13 - Large Scale Indoor 3D Reconstruction with Drone-Captured 360 Imagery

Yuanbo Chen, William Zhang, Jason Wang, Sophie Gao

May 2024

Advised by: Avideh Zakhor
Executive Summary

3D indoor scene reconstruction is crucial for applications such as historical site preservation, virtual asset creation, and digital twin development. In recent years, drones have been utilized in an indoor setting for scene reconstruction due to their highly mobile nature. They are versatile, easy to control, and can explore areas that are hard to reach for humans and other mobile robots. Thus, this report focuses on the area of 3D indoor scene reconstruction utilizing drones.

In terms of the reconstruction algorithm, many approaches exist such as Structure-from-Motion (SfM) and Simultaneous-Localization-and-Mapping (SLAM). Recently, radiance field methods such as Neural Radiance Fields (NeRFs) have been gaining popularity due to their photo-realistic results and ability to demonstrate view-dependent effects. However, most methods are suited for room-scale scenes and have difficulties scaling to larger scenes such as multi-story buildings. We address this challenge by proposing a scalable pipeline for large-scale 3D indoor scene reconstruction. Our strategy involves a divide-and-conquer approach, separating the entire scene into smaller manageable blocks. With this approach, we can massively speed up the processing speed since each sub-block can be reconstructed individually in parallel. Moreover, when rendering the large scene post-reconstruction, each sub-block can be loaded only when it is needed, removing memory constraints of loading a large scene.

We utilize 3D Gaussian Splatting as our main reconstruction method as it offers photo-realistic reconstruction in an efficient manner. To remedy the data-hungry nature of radiance field methods, we mount a 360 camera on the drone to capture wider set of camera viewpoints. Our pipeline’s efficacy is demonstrated by testing it on multi-story, large-scale indoor scenes, such as the Cory Hall and Hearst Memorial Mining Building at the University of California, Berkeley. Videos showing the reconstruction of both scenes can be viewed here: [https://drive.google.com/drive/folders/1TpsX1VkJ4F4oI0OZzEY0eoYD_Nswvxxd?usp=sharing](https://drive.google.com/drive/folders/1TpsX1VkJ4F4oI0OZzEY0eoYD_Nswvxxd?usp=sharing).
## Contents

1 **Introduction** 4  
   1.1 Background of 3D Indoor Scene Reconstruction 4  
   1.2 A Scalable Pipeline for Large-Scale 3D Indoor Scene Reconstruction 4  

2 **Related Works** 6  
   2.1 Existing Indoor Scene Reconstruction Methods 6  
   2.2 Implicit Neural Scene Representation Methods Such as NeRF 6  
   2.3 Explicit Point-Based Scene Representation: 3D Gaussian Splatting 7  

3 **Overall Approach** 8  
   3.1 Preliminary - 3D Gaussian Splatting 8  
   3.2 Processing 360° Images 9  
   3.3 Drone Body Inpainting 9  
   3.4 Divide-And-Conquer 10  
   3.5 Coarse-to-Fine Alignment 10  
   3.6 Image Similarity with SIFT features 11  
   3.7 On-Demand Block Rendering 12  

4 **Experiments and Results** 14  
   4.1 Data Capture Procedure 14  
   4.1.1 Indoor Scene 1 - Cory Hall 14  
   4.1.2 Indoor Scene 2 - Main Lobby of Hearst Memorial Mining Building 15  
   4.2 Implementation Details 15  
   4.3 Speed and Image Quality Comparison Against Previous Methods on Cory Hall Scene 17  
   4.4 Evaluation on Hearst Memorial Mining Building Scene 20  
   4.5 Comparing Inpainting with Masking 24  

5 **Discussion and Conclusion** 27  
   5.1 Application 27  
   5.2 Limitation and Future Works 27
1 Introduction

1.1 Background of 3D Indoor Scene Reconstruction

3D indoor scene reconstruction is the task of virtually reconstructing a physical indoor scene from data, usually from a set of images of the physical scene. This is highly useful and applicable to areas such as the preservation of historic buildings or the generation of 3D assets for augmented and virtual reality, gaming, and cinematic purposes. It is also used in the process of developing digital twins of factories or indoor facilities where movement and changes need to be monitored.

The overall process of 3D scene reconstruction involves capturing a series of images of the target scene along with their corresponding pose information, i.e., the camera’s location and orientation for each captured image. Subsequently, a reconstruction algorithm is applied to generate the 3D scene, usually in the form of a point cloud or 3D mesh. While there are many factors to consider in the entire process, this paper focuses on two key aspects: the method of data capture and the reconstruction algorithm.

Conventional data capture methods for mobile 3D indoor reconstruction utilize either a wheeled robot or a human operator [1]. The former is constrained in capturing detailed images comprehensively, especially in areas near the ceiling. The latter faces limitations in Simultaneous Localization and Mapping (SLAM) reconstruction due to inevitable human body movements, such as pitch and roll. Consequently, many high-fidelity systems resort to a stop-and-go capture process using a tripod, which is both labor-intensive and time-consuming. Drones, however, can enable rapid and versatile capture from various perspectives, including confined spaces. Moreover, they also yield stable camera trajectories unattainable with robots or humans. Hence, in this report we adopt a drone-based approach for 3D indoor scene reconstruction.

As for reconstruction algorithms, radiance field methods [2] have recently been gaining popularity for their ability to produce photo-realistic results and demonstrate view-dependent lighting effects. This is particularly advantageous when capturing the reflectances of shiny surfaces in real-life, such as windows, mirrors and other smooth surfaces. Given these benefits, this report employs radiance field methods as the main approach for reconstruction.

1.2 A Scalable Pipeline for Large-Scale 3D Indoor Scene Reconstruction

Currently, most radiance field-based methods work well for reconstructing room-scale scenes. However, for scenes extending beyond the room-scale, existing methods usually face the problem of slow reconstruction speed or expensive computation. Recent work by Li et al. [1, 3] propose a pipeline for large scale indoor 3D scene reconstruction from drone images. They reconstruct an explicit 3D representation of the scene from neural implicit surfaces under the Manhattan-world assumption before synthesizing novel-views. In addition, they make their pipeline scalable by utilizing a divide-and-conquer method to split up a large scene into smaller sub-blocks. However, since their method performs view synthesis after generating explicit 3D geometry, it does not preserve sufficient details, nor does it handle view-dependent effects. Furthermore, utilizing neural implicit surfaces for scene reconstruction is a time consuming process, taking about 5 hrs to reconstruct a single sub-block. Thus, the goal of this project is to develop a pipeline that can efficiently reconstruct a large-scale indoor scene such as a multi-story building in a photo-realistic manner.

We utilize a divide-and-conquer based strategy to pre-process and split the captured data into
multiple sub-blocks. Each sub-block can be reconstructed by itself, and thus all blocks can be
reconstructed in parallel, which massively improves the overall speed of the pipeline. We maintain
a degree of overlap between each sub-block so that they can be matched and combined post-
reconstruction. Furthermore, we implement a system that only loads each sub-block into memory
when necessary, reducing the computational load of loading the entire scenes directly. As for our
reconstruction algorithm, we utilize 3D Gaussian Splatting, a recent radiance field method recently
published by Kerbl et. al. [4] that can rapidly generate photo-realistic reconstruction results in an
efficient manner. To remedy the data-hungry nature of radiance field-based methods, we mount a
360 camera on top of the drone to capture a larger set of viewpoints, as shown in Figure 1. Our over-
all pipeline is shown in Figure 2. To demonstrate the efficacy of our pipeline, we test our approach
on multi-story large scale indoor scenes such as Cory Hall and the Hearst Memorial Mining Build-
ing in University of California, Berkeley. Videos showing the reconstruction of both scenes can be
viewed here: https://drive.google.com/drivefolders/1TpsX1VkJ4F4oI00ZzEY0eoYD_Nswvxd?usp=sharing

Figure 1: Image of our drone with a 360 camera mounted on top.


2 Related Works

2.1 Existing Indoor Scene Reconstruction Methods

Traditional 3D indoor reconstruction involves manual surveying and professional equipment. In comparison, using a drone makes the capture and reconstruction more accessible and cost efficient. Drones are also more flexible when capturing complex narrow scenes. However, many current drone reconstruction projects rely on other devices such as robots and LiDAR to ensure accuracy [5, 6], which makes the pipeline forsake the inherent convenience associated with the drone-based approaches.

3D scene reconstruction usually involves computing the depth of field and incorporating it into the 3D model. Structure from Motion (SfM) utilizes feature matching to achieve the computations. Nevertheless, when it comes to indoor scene reconstruction, traditional SfM methods [7, 8] face challenges due to lack of distinctive features. These approaches typically fail to match features from a corridor with repetitive plain walls and floors.

2.2 Implicit Neural Scene Representation Methods Such as NeRF

Recently, great progress has been made in utilizing deep learning for scene reconstruction. One particular approach is NeRF [2], which implicitly models the scene with a coordinate-based multi-layer perceptron (MLP) trained under a differentiable volume rendering formula. This series of methods [2, 9, 10] accurately predict color and opacity of pixels given 3D positions and 2D projection directions. Yet, they usually take up to several days to train on one scene, and their rendering is slow due to tedious integration along projection direction. This poses a huge resource challenge on large-scale reconstruction.

Consequently, methods have been proposed to speed up the rendering process. Plenoxels [11] incorporates a sparse voxel grid and trilinear interpolation for faster optimization; Instant-NGP [12] introduces a more versatile input encoding to permit the use of a smaller network, reducing computation counts. Still, these NeRF-based methods can have erroneous reconstruction, over-representing an empty space with blurry artifacts, which is common in indoor scenes full of low-texture regions such as walls and floors.

Intrinsic limitations of NeRF [2] and its long, memory-intensive process has resulted in a variety of approaches for large-scale environments. [13, 14, 15, 16]. In order to address model capacity issues, pipelines such as Block-NeRF separate an environment into blocks such that the NeRFs fit into memory. Scalable pipelines have also become dynamic, allowing users to update individual sections of a large-scale environment without retraining [14].

In general, the way blocks are split in a large scale scenes is highly environment-specific. Some scalable pipelines such as Mega-NeRF utilize a geometric clustering algorithm that allows data to be parallelized [15] while others such as SCALAR-NERF utilize KMeans [16]. Following the creation of blocks, one needs to deal with the overlapping regions. Most pipelines enhance overlapping regions through scaling a smaller point-cloud. Recently, the movement towards explicit point-based scene representations [17, 18, 19] provide an optimistic approach to solve the current limitations of scalable pipelines.
2.3 Explicit Point-Based Scene Representation: 3D Gaussian Splatting

Point-based methods explicitly represent the scene with discrete and unstructured points, usually with an extent larger than a pixel. Coupled with differentiable point-based rendering techniques [20, 17] and neural features [18, 19], point-based methods optimize the position and opacity of points and could perform fast or even real-time optimization and rendering. Nevertheless, point-based reconstructions have poor performance in featureless or shiny areas. One seminal work that tackles both problems is Gaussian Splatting [4]. By representing the scene with more flexible 3D Gaussians, it optimizes the scene representation with adaptive Gaussian density control to prevent over/under reconstruction of differing regions. With its superior performance, we base our work on Gaussian Splatting, aiming to convert its usage to indoor scene reconstruction as well as scaling it up for large scale data.
3 Overall Approach

In this section we elaborate on our data capture and view synthesis pipeline. Our pipeline takes in a 360° video sequence captured indoors from a drone and outputs a 3D Gaussian Splat as seen in Figure 2. We first unwarp the spherical images by projecting them into a 6-faced cube. This allows the images to be used further down the pipeline during SfM and 3D Gaussian Splatting. Next, we inpaint these cube-mapped images to mask out regions that contain the drone body to reduce feature matching errors. Before performing SfM, we utilize a Divide-and-Conquer (DAC) approach and automatically split the scene into multiple sub-blocks along the drone trajectory. Subsequently, we retrieve camera poses and the sparse point cloud of each sub-block via SfM and reconstruct each sub-block with 3D Gaussian Splatting. Finally, we combine the sub-blocks together during rendering with coarse-to-fine alignment, loading each sub-block only when necessary. The outline of this section is as follows: 3.1 3D Gaussian Splatting as preliminary, 3.2 360° image processing, 3.3 drone-body inpainting, 3.4 divide-and-conquer, 3.5 coarse-to-fine alignment, 3.7 on-demand block rendering.

3.1 Preliminary - 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) is a recent method for novel view synthesis by Kerbl et al. [4]. It represents a scene with a set of anisotropic 3D Gaussians, where each Gaussian is defined by its center position $\mu$ and a covariance matrix $\Sigma$. The covariance matrix $\Sigma$ is parameterized by a scale vector $s \in \mathbb{R}^3$ and a quaternion $q \in \mathbb{R}^4$ that encodes rotation of the 3D Gaussian. In addition, each 3D Gaussian is associated with an opacity $\alpha$ and a set of spherical harmonics that encodes its view-dependent color.

As with other radiance field-based methods, SfM is required to retrieve accurate camera poses associated with each input image. 3DGS takes in an additional sparse point cloud output from
the SfM step as initialization. After training, the output 3D Gaussian Splat can be seen as a point cloud with additional effects such as anisotropic scaling, alpha-blending, and view-dependent color effects. This allows for much simpler post-processing manipulations such as our proposed coarse-to-fine point cloud alignment compared to traditional NeRF-related methods.

3.2 Processing 360° Images

To facilitate SfM, we transform spherical 360° images into cube-map images, comprising six normal view cube faces. Mathematically, a 360° image sphere can be represented in polar coordinates:

\[ x = r \sin \theta \cos \phi, \quad y = r \sin \theta \sin \phi, \quad z = r \cos \theta, \]

with \( r = 1, \theta \in [0, \pi], \phi \in [0, 2\pi] \). As shown in Figure 3(a), placing a cube with side length 2 at the origin for projection, each side face corresponds to one of four evenly divided values of \( \phi \). We use the four side faces front, left, back, right, in subsequent view synthesis. To provide enough matching features for SfM convergence, we further introduce overlapping cube-map images by rotating the 360° image along the z-axis by 45° to obtain four additional cube faces overlapping the original four, as demonstrated in Figure 3(b).

![Figure 3](image)

**Figure 3:** (a) cube-map projection. (b) 45° rotated projection.

3.3 Drone Body Inpainting

The eight cube-map images contain static drone body across all frames, as there is no relative motion between the 360° camera and the drone. This hinders feature matching of SfM, leading to poor reconstruction results. One way to rectify this is to lift up the camera higher above the drone so that the drone does not appear in the projected side cube-map images. However, this drastically changes the drone’s center of gravity, leading to unstable flights. Therefore, we propose an alternative, i.e. before feeding the cube-map images to SfM, we introduce an inpainting network to remove the drone body from the images.

We first utilize a foundation segmentation model Segment Anything (SAM) [21] to generate accurate drone body segmentation masks for the eight cube-map images as shown in Figure 4(a). We also find it beneficial to add elliptical masks to the tip of drone arms for the rotating propellers, shown in Figure 4(b). To accommodate potential displacement of the mounted 360° camera during and across data captures, we dilate the mask by several pixels. Lastly, we employ a novel mask-guided inpainting model based on optical flow and spatiotemporal information propagation, namely, ProPainter [22] to inpaint out the drone whilst ensuring multi-view consistency.

Figure 4: Masks for drone body on cube-map images. Left: SAM generated masks. Right: masks after adding propeller masks.

3.4 Divide-And-Conquer

Novel view synthesis on large-scale scenes is computationally expensive due to the large number of images. Moreover, large-scale indoor environments often have repetitive features, such as hallways, leading to matching errors and poor reconstruction. We thus use a divide-and-conquer approach (DAC) to make our pipeline scalable.

We denote our data capture strategy and block division protocol as $mSnB$, indicating $m$ captured flight sequences divided into a total of $n$ blocks, where $n \geq m$. When $m = 1$, data is captured in a single contiguous drone flight and divided automatically into blocks. Rather than clustering as in previous works [16, 15], we take advantage of the sequential nature of video data and automatically partition the video into blocks with equal number of frames. We ensure 25% overlap between adjacent blocks to reliably merge them together after DAC. When $m > 1$, data is captured in multiple separate drone flights with a 10m straight line trajectory overlap. Note that $m = n = 1$ corresponds to the case without DAC.

DAC allows for the parallel reconstruction of each block, significantly reducing computation time. As seen later, it also mitigates erroneous feature matching when computing 3D geometries during SfM. During rendering, the blocks are loaded individually according to the viewing position.

3.5 Coarse-to-Fine Alignment

Under our proposed DAC method, each block is processed separately. Despite using the same camera intrinsics across blocks, the iterative optimization in SfM results in different scales $S$ for different blocks. We further need to adjust rotation matrix $R$ and translation vector $T$ of each block to merge them back together.

To ensure a robust alignment, we first coarsely align $S$, $R$ and $T$ across neighboring blocks. Given two blocks, we want to find relative scale, rotation and translation $\Delta S$, $\Delta R$, $\Delta T$ to transform block 2 such that the overlapping section between blocks 1 and 2 are roughly aligned.
For $\Delta S$, we utilize the fact that the drone speed during data capturing is generally the same for block 1 and 2, and the frames are extracted at the same frame rate post data acquisition. We use the Sfm-estimated translations $\{T_{1,i}\}_{i=1}^{M_1}, \{T_{2,j}\}_{j=1}^{M_2}$ for the $M_1, M_2$ total frames in block 1 and 2, and calculate $D_1$ and $D_2$, the average distance traveled between frames in block 1 and 2, as in Equation 2:

$$D_1 = \frac{1}{M_1} \sum_{i=1}^{M_1-1} (T_{1,i+1} - T_{1,i}),$$

$$D_2 = \frac{1}{M_2} \sum_{j=1}^{M_2-1} (T_{2,j+1} - T_{2,j}).$$

(2)

Now, $D_1$ divided by $D_2$ is the $\Delta S$ we apply to block 2 for coarse scale alignment.

To obtain $\Delta R$, $\Delta \tilde{T}$, we use the Sfm-estimated camera poses $\{R_{1,i}, T_{1,i}\}_{i=1}^{N_1}, \{R_{2,j}, T_{2,j}\}_{j=1}^{N_2}$ of the $N_1, N_2$ frames in overlapping sections in blocks 1 and 2 respectively to obtain $\Delta R, \Delta \tilde{T}$:

$$\Delta \tilde{T} = \text{diff} (\tilde{T}_1, \tilde{T}_2), \quad \Delta R = \text{diff} (\tilde{R}_1, \tilde{R}_2),$$

(3)

where $\bar{\cdot}$ denotes mean over set and $\text{diff} (\cdot, \cdot)$ is the subtraction of the two poses. Specifically, Equation 3 is implemented differently for translation and rotation. For translation $\Delta \tilde{T}$, we perform mean and subtraction computation directly on the translation matrix:

$$\Delta \tilde{T} = \frac{1}{N_1} \sum_{i=1}^{N_1} T_{1,i} - \frac{1}{N_2} \sum_{j=1}^{N_2} T_{2,j}.$$  

(4)

For rotation $\Delta R$, we need to first transform the rotation matrices to quaternion form with $r2q$ for mean computation, and then transform them back to rotation matrices with $q2r$ for difference computation using matrix inverse, shown in Equation 5:

$$\Delta R = q2r \left( \frac{1}{N_1} \sum_{i=1}^{N_1} r2q (R_{1,i}) \right) \times \left[ q2r \left( \frac{1}{N_2} \sum_{j=1}^{N_2} r2q (R_{2,j}) \right) \right]^{-1}.$$  

(5)

We define two situations for overlapping sections. When $m = 1$, images are from the same flight capture, the overlapping section is explicitly known and $N_1 = N_2$. When $m > 1$ i.e. with multiple flight captures, we use image similarity to match the first frame of block 2, denoted by $f_{2,s}$ with the most similar frame $f_{1,s}$ in block 1, and match the last frame of block 1 $f_{1,e}$ with the most similar frame $f_{2,e}$ in block 2. Then $N_1$ becomes the frame count between $f_{1,s}$ and $f_{1,e}$, $N_2$ becomes the frame count between $f_{2,s}$ and $f_{2,e}$. We choose the most similar frame by using a simple heuristic of retrieving the image with the highest number of matching SIFT [23] features. This is explained further in section 3.6.

Lastly, we use a modified version of Iterative Closest Point (ICP) [24] in CloudCompare [25] to refine the alignment between the two blocks. We use the coarse alignment result $\Delta R, \Delta \tilde{T}$ as the initial condition for the ICP algorithm, and iteratively refine it to obtain the final $S_2$, $R_2$ and $\tilde{T}_2$.

### 3.6 Image Similarity with SIFT features

Given a query image, we want to find the most similar image, $t_i$, in a sequence of ordered image frames $t_1, \ldots, t_i, \ldots, t_m$, where $m$ is the total number of image frames in the comparison block. In
practice, we can reduce $m$ to the approximate location of overlap since we know the length of overlap in the real world in meters and the drone speed. As mentioned above, we find the most similar image by choosing the image with the highest number of matching SIFT features. To further improve robustness, we take the top-3 images with the highest number of feature matches and take the weighted average of the frame numbers after applying soft-max on the number of feature matches. The weighted average is then rounded to the nearest integer. Since our $360^\circ$ images generates 8 cube-mapped image frames, we perform this procedure for each side of the cube, yielding 8 results. Finally, we take the median of these 8 results to be the most similar image frame.

### 3.7 On-Demand Block Rendering

After block alignment, we refine the functionality of the interactive NeRF interface, Nerfstudio [26], to render blocks dynamically as required. Nerfstudio allows user to navigate freely in a scene representation, generating new views in real-time depending on the viewer’s position. We adapt Nerfstudio to accommodate our large-scale scene reconstruction comprising multiple blocks. Leveraging our DAC’s block division, we optimize computation by only rendering the closest block to the viewer’s position. We establish two different rendering protocols depending on the complexity of drone trajectories: one for simple forward paths, and one for intricate arbitrary paths.

A simple forward path is usually utilized in corridor environments, and the blocks only have overlap with their preceding and succeeding blocks. We thus only need to check for new block rendering at the overlaps. Demonstrated in Figure 5, we define the center of block 1 as $c_1$, the center of block 2 as $c_2$, and the center of their overlap as $o$. Note that the coordinates of $c_1$, $c_2$, $o$ are obtained from the SfM-estimated camera poses. We further denote the vector from $o$ to the viewer’s position as $\vec{p}$, the vector from $o$ to $c_1$ and $c_2$ as $\vec{u}_1$ and $\vec{u}_2$. The angles $\theta_1$, $\theta_2$ from $\vec{p}$ to $\vec{u}_1$, $\vec{u}_2$ are then calculated, and the block given by

$$\arg\min_{i \in \{1,2\}} \theta_i$$

is rendered in the viewer.

![Overlap Diagram](image)

---

Figure 5: Block rendering protocol for simple drone trajectories, e.g. corridors.

An intricate path is employed in spaces with multiple floors, looping corridors and possibly atriums, where drone trajectories may intersect and the blocks have arbitrary and possibly complex
overlapping scenes. We thus need to check for new block rendering at every viewer’s position. Demonstrated in Figure 6, we first conduct cubic spline interpolation on the drone trajectories of blocks, where \( s_1, s_2, \ldots \) represent sets of piece-wise cubic polynomials with continuity \( C^2 \) for drone trajectories in block 1, 2, \ldots. We then compute the distance \( d_1, d_2, \ldots \) from the viewer’s position to the splines \( s_1, s_2, \ldots \), and the block given by

\[
\arg \min_{i \in \{1,2,\ldots\}} d_i
\]

is rendered in the viewer. Note that the calculation of \( d_1, d_2, \ldots \) requires finding the closest point on each spline, which is a minimization problem. Such a problem could get stuck in local minima, leading to wrong block rendering. Following [27], we use several initial estimates for BFGS-B [28] optimization method, resulting in an accurate solution within five iterations.

Figure 6: Block rendering protocol for complex drone trajectories.

Our dynamic block rendering techniques, coupled with coarse-to-fine alignment, achieves seamless transition between blocks. Qualitative results are included in Section 4.3.
4 Experiments and Results

4.1 Data Capture Procedure

To evaluate the effectiveness of our proposed pipeline, we captured and reconstructed two indoor scenes: (1) 3rd floor of Cory Hall, and (2) the main lobby of Hearst Memorial Mining Building. Both indoor scenes are academic buildings located in University of California Berkeley.

4.1.1 Indoor Scene 1 - Cory Hall

We capture this scene to evaluate our pipeline against Li et al.'s \cite{1} pipeline on the same scene. One issue faced by \cite{1} during data capture is the need to fly the drone in a zig-zag fashion as seen in Figure 7a. As they utilize the frontal camera embedded within the drone for data capture, it lacks sufficiently diverse viewpoints in order for structure from motion to work well. This is exacerbated by the fact that Cory Hall has texture-less and feature-less hallways, making feature extraction and matching even harder. Thus, they need to sway the drone side-to-side while flying in order to capture more diverse viewpoints, which increases the difficulty of the drone flight and makes the data capture process less reproducible. In contrast, our utilization of a 360° camera results in substantially more viewpoints compared to a normal camera, thus facilitating feature extraction and matching during the structure from motion. This allows us to fly in a simple and straight drone path while capturing the scene, as seen in Figure 7b. Furthermore, without the swaying of the drone, our drone flies in a more stable fashion and is less affected by motion blur. Unlike \cite{1}, we do not employ further Laplacian filtering to filter out blurry images.

As mentioned in section 3.4, the data capture process can be done in a single drone flight or multiple separate drone flights depending on the size of the scene and the drone’s battery capac-
ity. Thus to simulate these scenarios, we capture two datasets for this scene, one with only a single drone flight capture, and one with 4 drone flights captures. The single-flight capture dataset contains 400 equirectangular frames which we divide it into 4 sub-blocks according to our DAC strategy. This is referred to as $1S4B$. As for the 4-flight capture dataset, each flight capture results in its own sub-block with around 130 frames, and is referred to as $4S4B$. For both $1S4B$ and $4S4B$, we ensure an overlap of 32 frames between each sub-block. Finally, we also test our pipeline on the single-flight capture dataset but without using the DAC strategy for ablation purposes. This is referred to as $1S1B$.

### 4.1.2 Indoor Scene 2 - Main Lobby of Hearst Memorial Mining Building

We also captured the main lobby of the Hearst Memorial Mining Building at University of California Berkeley to evaluate our pipeline’s effectiveness on a large, building-scale scene. This scene contains 3 floors with tall ceilings and multiple staircases, seen in Figure 8. Due to its scale it is not feasible to capture the entire scene in a single drone flight. Thus, we separate the scene into 3 drone flights, one flight per floor. The first floor of the scene contains a large empty space. Thus, we fly the drone in the trajectory shown in Figure 9a in order to fully cover the floor area. As for the second and third floors, we fly the drone only along the corridors and staircases of the scene, as seen in Figures 9b and 9c. We could not fly the drone in the center as that would trigger the fire alarm of the building. We ensure overlap between each floor along the staircases, as shown in Figure 9.

![Figure 8](image8.jpg)

Figure 8: Hearst Memorial Mining Building consists of multiple stories, a large atrium, and complex light fixtures.

### 4.2 Implementation Details

For data acquisition of both scenes, we utilize the Insta360 ONE RS camera and mount it onto a DJI Mavic Air 2 drone to capture 360° videos, as shown in Figure 1. All videos are captured at 30 frames per second (FPS). After data capture, we use Insta360 Studio to export the 360° videos into $5760 \times 2880$ equirectangular MP4 videos. For scene 1 at Cory Hall, we extract equirectangular frames from the videos at 3 FPS with take-off and landing frames trimmed. As for scene 2 at
Hearst Memorial Mining Building, frames are extracted at 1 FPS. Subsequently, these equirectangular frames are cube-mapped to $768 \times 768$ sized images for the SfM step. We use sparse reconstruction in COLMAP [29] as the SfM method. Our data processing and model training is performed on a NVIDIA TITAN RTX 24GB GPU. For 3D Gaussian Splatting, we used the original implementation proposed by Kerbl et al. [4].

Figure 9: Drone trajectories for all three floors of Hearst Memorial Mining Building. (a) Floor 1. The two loops in front show the path the drone took while flying in the empty space of the first floor. The diagonal pathways towards the back show the drone path along the staircases leading up to floor 2. (b) Floor 2. The drone flew along the corridor and staircases leading from floor 1 and up to floor 3. (c) Floor 3. As with floor 2, the drone flew on the corridor and staircase leading from floor 2.
4.3 Speed and Image Quality Comparison Against Previous Methods on Cory Hall Scene

We compare our method with [1] in terms of rendered image quality and computation time. The dataset in [1] on the same environment uses around 2000 1360 × 765 frames, with each frame having about 22% of the total pixels of our cube-mapped 360° image. Its total pixel count is 1.25 times of our single flight capture dataset and 0.8 times of our 4-flight capture dataset in [1]. We use peak-signal-to-noise ratio (PSNR) and structural index similarity (SSIM) as our metrics for image quality assessment.

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR↑</th>
<th>SSIM↑</th>
<th>Time↓</th>
<th># Pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>20.76</td>
<td>0.74</td>
<td>25 hrs</td>
<td>2 B</td>
</tr>
<tr>
<td>1S1B (w/o DAC)</td>
<td>31.31</td>
<td>0.95</td>
<td>3 hrs</td>
<td>1.6 B</td>
</tr>
<tr>
<td>1S4B</td>
<td>34.03</td>
<td>0.96</td>
<td>0.5+4(0.5) hrs</td>
<td>1.6 B</td>
</tr>
<tr>
<td>4S4B</td>
<td>34.64</td>
<td>0.96</td>
<td>0.5+4(0.5) hrs</td>
<td>2.5 B</td>
</tr>
</tbody>
</table>

Table 1: Quantitative comparison between pipelines.

In Table 1 we quantitatively compare three versions of our method, 1S1B, 1S4B, 4S4B with [1] using PSNR and SSIM. As seen, all three versions of our approach outperform [1] by as much as 13.88 dB in PSNR and 0.22 in SSIM. Our method 1S4B significantly outperforms [1] with an increase of 13.3 dB PSNR and 0.22 SSIM, despite the fact that its total pixel count, i.e. number of pixels per frame times total number of frames, is 80% of [1]. Moreover, our method takes much less computation time overall.

A more qualitative comparison between our 1S4B and [1] can be seen in Figure 10. Row 1 shows that our method results in much sharper reconstruction than [1]. In corridors shown in row 2, 1S4B generates better details on walls even though they are parallel to the drone trajectory. We attribute this to our use of the 360° camera, which captures more viewpoints in a single forward flight. The image quality improvement from utilizing the 360° camera can also be seen in row 3 when rendering “backwards” relative to the drone trajectory. Lastly, as seen in row 4, the use of Gaussian Splatting allows dynamic reflective surfaces, greatly enhancing viewing experiences.

As seen in Table 1 4S4B achieves 0.61 dB higher PSNR than 1S4B. This is due to the introduction of 50% new pixels in the overlapping sections from separate drone flights. Comparing 1S4B and 1S1B, the number of pixels are the same due to the same underlying data, yet 1S4B achieves a higher PSNR and SSIM, indicating an inherent advantage of DAC. This is surprising as SfM on the whole scene without DAC is generally believed to have more constraints for optimization and should produce better results. We speculate that DAC mitigates erroneous feature matching by limiting number of repetitive/plain features. Our experiments also show a slight reduction of 0.5 hrs in pipeline time utilizing DAC. More importantly, DAC allows time per-block shown in brackets in Table 1 to be parallelized for more significant time reduction in our method.

Figure 11 shows consecutive renderings at block boundary with different methods. With 1S4B and 4S4B, the transition between two blocks is smooth, thanks to our coarse-to-fine alignment. Moreover, the DAC rendering quality in 1S4B and 4S4B is visually superior to 1S1B.
Figure 10: Novel-view renderings of $1S4B$ (left) and $1$ (right).
Figure 11: Consecutive renderings at block boundary with different data capture protocols. Note $1S4B$ and $4S4B$ have different block boundary since $1S4B$ is automatically divided.
4.4 Evaluation on Hearst Memorial Mining Building Scene

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR↑</th>
<th>SSIM↑</th>
<th>Time↓</th>
<th># Pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td>3S3B</td>
<td>28.58</td>
<td>0.90</td>
<td>0.5+3-(3) hrs</td>
<td>4.8 B</td>
</tr>
</tbody>
</table>

Table 2: Quantitative results on Hearst Memorial Mining Building.

We similarly compute PSNR and SSIM on the Hearst Memorial Mining Building (HMMB) dataset, shown in Table 2. The PSNR and SSIM for HMMB is lower than compared to Cory Hall. This is expected due to a number of reasons. Firstly, the HMMB scene is more complex compared to Cory Hall, with multiple floors, corridors and staircases. This increases the complexity of structure-from-motion, which results in a noisier initialization for 3D Gaussian Splatting. Furthermore, the repeating brick-wall texture of HMMB creates more noisy image features compared to the smooth surfaces of Cory Hall, as seen in Figure 12. Thus, this also results noisy feature matching, which decreases structure-from-motion performance.

Figure 12: A comparison of wall textures between (a) Cory Hall and (b) Hearst Memorial Mining Building. Unlike Cory Hall, Hearst Memorial Mining Building has more repeating textures due to its brick walls.

Figure 13 shows the final merged drone paths on this scene after coarse-to-fine alignment. It can be seen that the drone paths are closely aligned along the staircases of the scene. Our qualitative results are presented in Figure 14 which shows renderings of HMMB at different perspectives. Moreover, Figure 15 shows consecutive renderings at block boundaries of HMMB. As seen, our proposed coarse-to-fine alignment and on-demand block rendering technique for complex data captures successfully aligns neighboring blocks and loads the block almost instantly.

The second and third rows of Figure 15 demonstrates the complexities of the HMMB scene. As mentioned by our data capture protocol of the HMMB scene in section 4.1.2, we split the scene in 3 blocks, one for each floor with overlaps along the staircases only. This means that there are no overlapping viewpoints at areas other than the staircases connecting each floor. Thus, when transitioning between blocks in areas other than the staircase, it is highly likely to reach a spot in
Figure 13: Different views of the final merged drone paths after coarse-to-fine alignment.

the block where no existing viewpoints have been captured for that particular block. An example of this can be seen when transitioning from the center of the empty atrium in floor 1 to floor 2 as shown in the second row of Figure [15]. Since we were not allowed to fly the drone in the center of the building for floors 2 and 3, the center of floor 2 is poorly reconstructed with many floaters. This is an inherent challenge to the HMMB scene since every floor has a clear line of sight to other floors.
Figure 14: Qualitative results of reconstructed Hearst Memorial Mining Building scene.
Figure 15: Consecutive renderings at block boundaries of HMMB. $T - 1$ corresponds to the previous floor and $T$ corresponds to the next floor.
4.5 Comparing Inpainting with Masking

<table>
<thead>
<tr>
<th>Methods</th>
<th>Block 0</th>
<th>Block 1</th>
<th>Block 2</th>
<th>Block 3</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR</td>
<td>SSIM</td>
<td>PSNR</td>
<td>SSIM</td>
<td>PSNR</td>
</tr>
<tr>
<td>Inpainting</td>
<td>35.13</td>
<td>0.9651</td>
<td>33.76</td>
<td>0.9602</td>
<td>33.78</td>
</tr>
<tr>
<td>Masking</td>
<td>34.10</td>
<td>0.9632</td>
<td>30.23</td>
<td>0.9521</td>
<td>30.24</td>
</tr>
</tbody>
</table>

Table 3: Quantitative comparison of image quality between drone inpainting and drone masking on 4S4B.

We further compare our drone body inpainting approach with a simple drone masking approach, taking 4S4B as the target of comparison. For drone masking, we take the dilated drone body masks generated for inpainting, as described in Section 3.3, and input them into COLMAP to mask the drone body out when doing feature matching. Then, we modify the original 3DGS code, adding --masks option for training to ignore the region with the drone body when computing loss. Quantitative results is presented in Table 3. As seen, on average, drone body inpainting achieves 2.78dB higher PSNR and 0.0061 higher SSIM than simply masking out the drone body.

Figure 16: Masking (a) produces more blurry results in the bottom right of the wooden panels compared to inpainting (b).

The differences in rendering quality between these two methods are minimal in most areas. Some differences can be spotted in image areas containing high frequency, where inpainting produces a less blurry result as seen in Figure 16. In general, high frequency regions are much more easily affected by loss of information than areas of low frequency. With inpainting, some of the information is accurately restored, which leads to better results in high frequency areas. However, a downside of inpainting is that it hallucinates information, which generates noisy image features in the inpainted regions. This results in noisier feature matching and thus worse reconstruction quality, as seen from Figure 17b where the reconstructed gaussians are extruding out of the planar surface of the wooden panels. Moreover, smaller features such as the power outlet on the bottom right in Figure 18 could be lost when inpainting. We see the power outlet appears far fainter in the
inpainted rendering rather than the drone masked rendering.

Another culprit for the lower PSNR of masking can be seen in Figure 19a, where there are visual artifacts present in masked out areas. When we inpaint the masked out areas, the lost information in these regions are sufficiently recovered, hence producing fewer visual artifacts.

Figure 17: Inpainting generates noisy image features which worsens reconstruction quality in high frequency areas. It can be seen from (b) that the reconstructed gaussians are protruding out of the planar surface of the wooden panels, unlike in (a).

Figure 18: Small regions such as the wall plug can be inpainted inaccurately, thus producing blurrier reconstruction results.
Figure 19: Visual artifacts can be seen in areas that have been masked out (a) if there is no subsequent inpainting. When the masked areas are inpainted, these visual artifacts are not present (b).
5 Discussion and Conclusion

We present an efficient and scalable pipeline for large-scale indoor novel-view synthesis. By utilizing a divide-and-conquer strategy, our pipeline is parallelizable and scales up to large complex indoor scenes. Furthermore, our utilization of a 360° camera provides us with substantially more diverse viewpoints, improving both our image quality and reducing the complexity of the drone flight. Our experiments show marked improvement in both quality and speed over prior methods on the same scene.

5.1 Application

Our approach for a large-scale reconstruction pipeline opens the door for a myriad of impactful applications. For instance, indoor scene reconstruction can be a pivotal step in preserving heritage sites. As historically significant buildings continue to deteriorate, having viewable renderings of these locations can serve as a value tool to share these experiences with the rest of the world. Another application is the generation of 3D assets for AR/VR, gaming, and other sources of media. Since renderings are easily traversable, interacting with these scenes can be incredibly valuable to create the most immersive experiences as possible. Lastly, indoor scene reconstruction can provide digital twin modeling for factories and warehouses. Due to the complex nature of warehouses, a data-driven approach using digital twins provides operators with key insights into their decision making. As AR/VR approaches become more accessible, the applications of large-scale reconstruction will only get larger.

5.2 Limitation and Future Works

While our complete pipeline shows promising results, there is still for improvement in the future. One limitation is the presence of floaters within the final rendering. Despite the impressive renderings that 3DGS is able to provide, the method relies heavily on an accurate initial point cloud for good results. Oftentimes when a point cloud is noisy, the 3D Gaussians can be placed in the wrong location, causing blurry viewpoints. In our specific pipeline, we detect some evidence of floaters during our block transition since certain surfaces may be far from the drone. There has been recent work on removing these floaters by changing the growth logic when 3D Gaussians are optimized to fit a certain scene [30]. We hope to potentially incorporate this logic to remove such artifacts in our final rendering. Another limitation was related to the drone capture step. As indoor scenes struggle with bluetooth connectivity, drones can oftentimes lose connection with their human operators. Future work to alleviate this could include an automated drone flight or an explore algorithm for drones. Lastly, for larger scenes such as a concert hall or atrium, a more complicated drone flight is required to capture more viewpoints within a scene. Since our pipeline relies on COLMAP to find camera position in g, it can be a bottleneck for larger, more complex scenes. Recent work to remove this bottleneck have also been found [31]. Leveraging newer versions of 3DGS can open many doors for our pipeline and we hope to incorporate all these methods in future iterations.
References


