

# Scalable Indoor Novel-View Synthesis using Drone-Captured 360 Imagery with 3D Gaussian Splatting

Yuanbo Chen<sup>\*1</sup> , Chengyu Zhang<sup>\*1</sup> , Jason Wang, Xuefan Gao, and Avideh Zakhor<sup>1</sup> 

UC Berkeley, Berkeley CA 94720, USA  
 {yuanbo\_chen, chengyu\_zhang, jasoncwang, xuefangao, avz}@berkeley.edu

**Abstract.** Scene reconstruction and novel-view synthesis for large, complex, multi-story, indoor scenes is a challenging and time-consuming task. Prior methods have utilized drones for data capture and radiance fields for scene reconstruction, both of which present certain challenges. First, in order to capture diverse viewpoints with the drone’s front-facing camera, some approaches fly the drone in an unstable zig-zag fashion, which hinders drone-piloting and generates motion blur in the captured data. Secondly, most radiance field methods do not easily scale to arbitrarily large number of images. This paper proposes an efficient and scalable pipeline for indoor novel-view synthesis from drone-captured 360° videos using 3D Gaussian Splatting. 360° cameras capture a wide set of viewpoints, allowing for comprehensive scene capture under a simple straightforward drone trajectory. To scale our method to large scenes, we devise a divide-and-conquer strategy to automatically split the scene into smaller blocks that can be reconstructed individually and in parallel. We also propose a coarse-to-fine alignment strategy to seamlessly match these blocks together to compose the entire scene. Our experiments demonstrate marked improvement in both reconstruction quality, *i.e.* PSNR and SSIM, and computation time compared to prior approaches.

**Keywords:** Novel-view synthesis · Scalable scene reconstruction pipeline · 360° image processing · 3D Gaussian Splatting

## 1 Introduction

Novel-view synthesis for indoor environments is a valuable task with wide applications in areas such as heritage preservation [9] and digital twin modeling [28]. Other use cases include generating visual assets for further downstream tasks such as in robot navigation [3], AR/VR [10, 15, 40], and medical imaging [7, 29]. Classical pipelines for novel-view synthesis are as follows: (1) data acquisition, (2) 3D scene reconstruction, and (3) novel-view rendering. Due to the complexities involved in this multi-step process, there are many areas of improvement for better performance, especially when scaling up to large, multi-story, scenes.

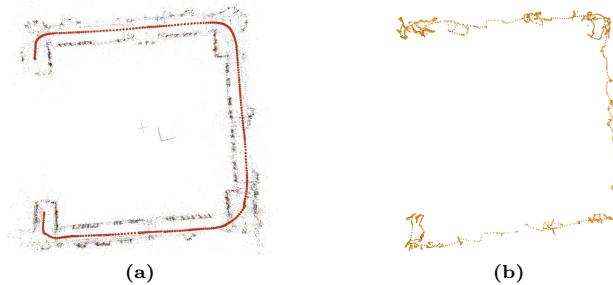
---

<sup>\*</sup> Both authors contributed equally to the paper.

Conventional data acquisition utilizes either a wheeled robot [42] or human operator [5, 8, 18, 21, 35–37]. The former is limited in capturing all surfaces in the scene, especially in areas near the ceiling. The latter faces limitations in Simultaneous Localization and Mapping (SLAM) reconstruction due to inevitable human body pitch and roll movements [18, 30, 31]. Consequently, many high-fidelity systems such as Matterport [23] resort to a tripod-based stop-and-go capture process, which is both labor-intensive and time-consuming. Drones, however, can enable rapid and versatile capture from various perspectives, including in confined spaces. Moreover, they yield stable camera trajectories unattainable with robots or humans. Thus, this paper focuses on drone-based data capture.

Post data capture, classical approaches [13, 27] use structure-from-motion (SfM) and multi-view stereo (MVS) to reconstruct the scene before synthesizing novel-views. Recently, radiance-fields methods [12, 24, 25] have revolutionized the field with their photorealistic results. They implicitly represent the density and color fields of the scene and synthesize novel-views via volume rendering, allowing for view-dependent effects not easily achievable with classical approaches. Much work has been done to extend this class of methods to large scale scenes [6, 32, 34], but they mainly focus on outdoor scenes. In this paper, we develop a pipeline for large scale indoor novel-view synthesis using radiance-field approaches.

Recent work by Haoda *et al.* [19, 20] propose a pipeline for large scale indoor 3D scene reconstruction from drone images. They reconstruct an explicit 3D scene representation from neural implicit surfaces under Manhattan-world assumption before synthesizing novel-views. In addition, they scale their pipeline by splitting up a large scene into smaller blocks. However, since they perform view synthesis after generating explicit 3D geometry, scene details and view-dependent effects are not sufficiently preserved. Furthermore, utilizing neural implicit surfaces for scene reconstruction is time consuming, taking about 5 hours for a modern machine to reconstruct a single block. Since they utilize the drone’s frontal camera for data capture, they have to fly the drone in a zig-zag fashion as seen in Fig. 1b in order to capture more diverse viewpoints. This makes drone piloting harder and the data capture process less reproducible.



**Fig. 1:** Comparison of our drone trajectory (a) and Li *et al.*’s [19] (b). Our drone trajectory (in red) is straight, simple to pilot, and more reproducible than (b).

We propose an indoor novel-view synthesis pipeline that is fast, photorealistic, and scalable to large complex scenes. We follow Haoda *et al.* [20] and use a divide-and-conquer strategy to automatically split the scene into smaller blocks for both capture and processing. Rather than generating an explicit 3D structure, we use 3D Gaussian Splatting [16] as our scene representation and directly synthesize novel-views. In addition, we mount a 360° camera on top of the drone to capture a larger set of viewpoints. This allows us to fly in a simple straight drone path while capturing the scene, as seen in Fig. 1a. Furthermore, we do not employ further Laplacian filtering for blurry images like in [20] since our stable drone trajectory creates less motion blur. We also develop a system that loads each reconstructed block into memory only when necessary, reducing the computational complexity of scene loading. We utilize a simple coarse-to-fine alignment method to align blocks together during the rendering stage. Our pipeline is evaluated on a large complex indoor scene with multiple rooms and corridors using a commercially available drone. Our results demonstrate marked improvement over [20]’s approach on the same scene by as much as 13.88dB in PSNR and 0.22 in SSIM, while achieving this at 10 times faster processing times.

## 2 Related Work

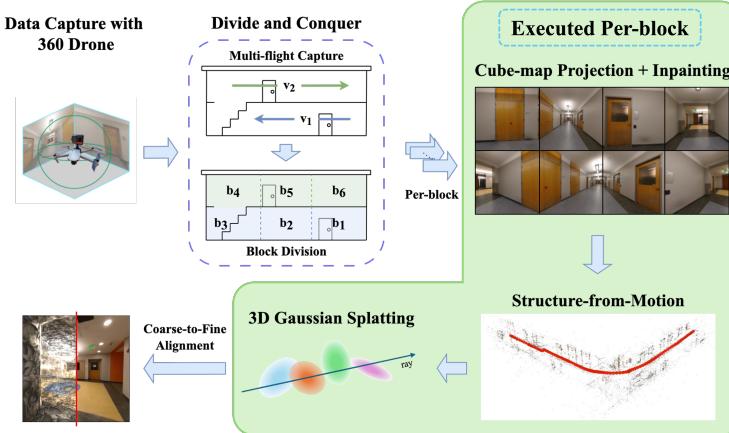
### 2.1 Large-Scale Scene Reconstruction with Radiance Fields

Intrinsic limitations of NeRF [24] and its long, memory-intensive process has resulted in a variety of approaches for large-scale scene reconstruction [6, 14, 32, 34]. In order to address model capacity issues, pipelines such as Block-NeRF separate a scene 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 scene without retraining [32]. In general, the way blocks are split in a large scale scenes is highly scene-specific. Some scalable pipelines such as Mega-NeRF utilize a geometric clustering algorithm that allows data to be parallelized [34] while others such as SCALAR-NeRF utilize KMeans [6].

### 2.2 Explicit Point-Based Representation: 3D Gaussian Splatting

Recently, explicit point-based scene representations [1, 26, 41] provide a plausible approach to solve the limitations of scalable pipelines. Point-based methods explicitly represent the scene with discrete and unstructured points, usually larger than a pixel. Coupled with differentiable point-based rendering techniques [39, 41] and neural features [1, 26], point-based methods optimize the position and opacity of points and perform fast or even real-time optimization and rendering. Nevertheless, they have poor performance in featureless or shiny areas. One seminal work that tackles both problems is 3D Gaussian Splatting [16]. By representing the scene with more flexible 3D Gaussians, it optimizes the scene representation with adaptive density control to prevent over- or under- reconstruction of regions. With its superior performance, we base our work on 3D Gaussian Splatting for indoor scene reconstruction as well as scaling it up to large scale data.

### 3 Method



**Fig. 2:** Our overall pipeline.

In this section we elaborate on our data capture and view synthesis pipeline. Our pipeline takes in 360° video sequences captured indoor from one or multiple drone flights and outputs 3D splats as seen in Fig. 2. From the single or multiple flight captures, we optionally further subdivide the captured video into multiple smaller blocks. Then, we unwarp the captured spherical images via cube-mapping. 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. Subsequently, we retrieve camera poses and the sparse point cloud of each block via SfM and reconstruct each block with 3D Gaussian Splatting. As such, a block can either correspond to a complete video from one flight capture or a subset of it. Finally, we match the blocks together with coarse-to-fine alignment and load each block only when necessary during rendering.

#### 3.1 Preliminary - 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) is a recent method for novel-view synthesis by Kerbl *et al.* [16]. 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 camera poses associated with each input image. 3DGS takes in an additional sparse

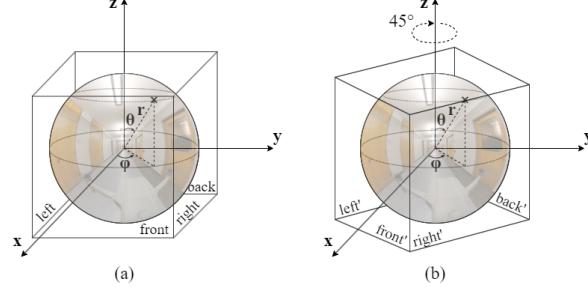
point cloud output from the SfM step as initialization. After training, the output 3D Gaussian Splat can be seen as a point cloud with additional effects such as anisotropic scaling, alpha-blending, and view-dependent color effects. This allows for much simpler post-processing manipulations such as our 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:

$$x = r \sin \theta \cos \phi, \quad y = r \sin \theta \sin \phi, \quad z = r \cos \theta, \quad (1)$$

with  $r = 1$ ,  $\theta \in [0, \pi]$ ,  $\phi \in [0, 2\pi]$ . As shown in Fig. 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  $\phi$ .<sup>1</sup> We use the four side faces front, left, back, right, in subsequent view synthesis. To provide sufficient number of 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 Fig. 3(b).



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

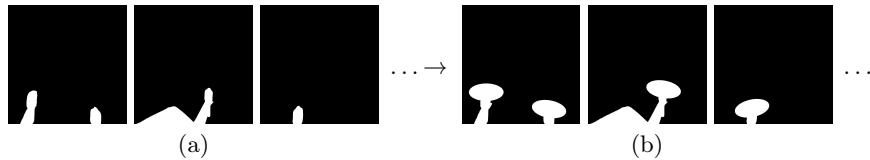
### 3.3 Drone Body Inpainting

The eight cube-map images contain static drone body across all frames, as there is no relative motion from the 360° camera. This hinders feature matching of SfM, leading to poor reconstruction. One solution is to lift the camera higher so that the drone does not appear in the side cube-map images. However, this drastically

<sup>1</sup> <https://stackoverflow.com/questions/29678510/convert-21-equirectangular-panorama-to-cube-map>

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 Segment Anything (SAM) [17] to generate accurate drone body segmentation masks for the eight cube-map images as shown in Fig. 4(a). We also find it beneficial to add elliptical masks to the tip of drone arms corresponding to the rotating propellers, shown in Fig. 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 [43] to inpaint out the drone whilst ensuring multi-view consistency. In our experiments, we also evaluated our pipeline with only masks without inpainting, and the results were worse.



**Fig. 4:** Left: SAM generated drone-body masks. Right: 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). This strategy allows our proposed scheme to be scalable with respect to both the number of flight captures and the number of video frames in each flight capture.

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 [6,34], we take advantage of the sequential nature of video data and automatically partition the video from the single flight capture 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 an approximately 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  $\mathcal{S}$  for different blocks. We further need to adjust rotation matrix  $\mathcal{R}$  and translation vector  $\mathcal{T}$  of each block to merge them back together.

To ensure a robust alignment, we first coarsely align  $\mathcal{S}$ ,  $\mathcal{R}$  and  $\mathcal{T}$  across neighboring blocks. Given two blocks, we want to find relative scale, rotation and translation  $\Delta\mathcal{S}$ ,  $\Delta\mathcal{R}$ ,  $\Delta\mathcal{T}$  to transform block 2 such that the overlapping section between blocks 1 and 2 are roughly aligned. For  $\Delta\mathcal{S}$ , 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 blocks 1 and 2 respectively, and calculate  $D_1$  and  $D_2$ , the average distance traveled in blocks 1 and 2, as in Eq. (2):

$$D_1 = \frac{1}{M_1} \sum_{i=1}^{M_1-1} (T_{1,i+1} - T_{1,i}), \quad D_2 = \frac{1}{M_2} \sum_{j=1}^{M_2-1} (T_{2,j+1} - T_{2,j}). \quad (2)$$

$D_1$  divided by  $D_2$  is the  $\Delta\mathcal{S}$  which we apply to block 2 for coarse scale alignment.

To obtain  $\Delta\mathcal{R}$  and  $\Delta\mathcal{T}$ , we use the SfM-estimated camera poses  $\{R_{1,i}, T_{1,i}\}_{i=1}^{N_1}$  and  $\{R_{2,j}, T_{2,j}\}_{j=1}^{N_2}$  of the  $N_1$  and  $N_2$  frames in overlapping sections in blocks 1 and 2 respectively, shown in Eq. (3):

$$\Delta\mathcal{R} = \text{diff}(\bar{R}_1, \bar{R}_2), \quad \Delta\mathcal{T} = \text{diff}(\bar{T}_1, \bar{T}_2), \quad (3)$$

where  $\bar{\cdot}$  denotes mean over set and  $\text{diff}(\cdot, \cdot)$  is the subtraction of the two poses. 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  $f_s^2$  with the most similar frame  $f_s^1$  in block 1, and match the last frame of block 1  $f_e^1$  with the most similar frame  $f_e^2$  in block 2. We choose the most similar frame by using a simple heuristic of retrieving the image with the highest number of matching SIFT [22] features, explained below.

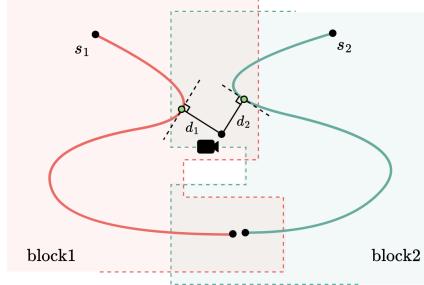
**Image Similarity with SIFT Features** Given a query image, we want to find the most similar image  $t_i$ , in a sequence of ordered image frames  $t_1 \dots t_i \dots t_m$ , where  $m$  is the total number of image frames in the comparison block. In practice, we can determine  $m$  using approximate overlap region between two blocks taking into account flight capture parameters. 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 by computing the softmax-probabilities using the number of feature matches. The weighted average is then rounded to the nearest integer. Since our 360° images generates 8 cube-mapped image frames, we perform this procedure for each side of the cube, yielding 8 numbers. Finally, we take the median of these 8 numbers to be the most similar image frame.

Lastly, we use a modified version of Iterative Closest Point (ICP) [2] in Cloud-Compare [11] to refine the alignment between the two blocks. We use the coarse alignment result  $\Delta\mathcal{S}$ ,  $\Delta\mathcal{R}$ , and  $\Delta\mathcal{T}$  as the initial condition for the ICP algorithm, and iteratively refine it to obtain the final  $\mathcal{S}_2$ ,  $\mathcal{R}_2$  and  $\mathcal{T}_2$ .

### 3.6 On-Demand Block Rendering

After block alignment, we refine the functionality of the Nerfstudio [33] interface 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 comprising multiple blocks. Leveraging our DAC’s block division, we optimize computation by only rendering the closest block to the viewer’s position.

Demonstrated in Fig. 5, we first conduct cubic spline interpolation on the drone trajectories of blocks, where  $s_1, s_2, \dots$  represent sets of piece-wise cubic polynomials with continuity  $C^2$  for drone trajectories in blocks 1, 2,  $\dots$ . We then compute the distance  $d_1, d_2, \dots$  from the viewer’s position to the splines  $s_1, s_2, \dots$ . Finally, the block given by:  $\arg \min_{i \in \{1, 2, \dots\}} d_i$  is rendered in the viewer. Note that distance calculation on splines is a minimization problem that could result in a local minima, leading to wrong block rendering. Following [38], we use several initial estimates for BFGS-B [4] optimization method, resulting in an accurate solution within five iterations.



**Fig. 5:** Block rendering protocol for complex drone trajectories.

Our dynamic block rendering techniques, coupled with coarse-to-fine alignment, achieves seamless transition between blocks, as seen in our video renderings.<sup>2</sup><sup>3</sup> Qualitative results are included in Sec. 4.3.

<sup>2</sup> <https://youtu.be/UsZUYquSSkM>

<sup>3</sup> <https://youtu.be/MhVooLRnmhE>

## 4 Experiments

### 4.1 Datasets and 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.

**Indoor Scene 1 - Cory Hall** We capture this scene to compare our pipeline against Li *et al.*'s [20] pipeline on the same scene.

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

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 consists of 3 floors with tall ceilings and multiple staircases. 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 Fig. 6a 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 Fig. 6b and Fig. 6c. We could not fly the drone in the center as that would trigger the fire alarm of the building. Hence, we opted to fly two loops along the corridors at different heights in order to capture sufficient viewpoints of the empty atrium in the center. We ensure overlap between floors along the staircases, as shown in Fig. 6.

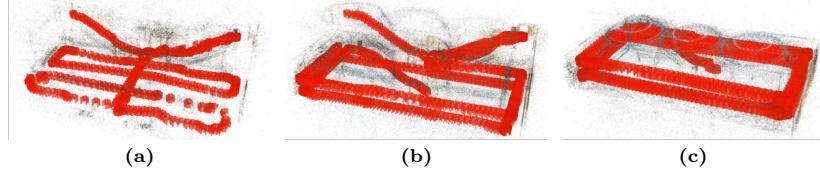
### 4.2 Implementation Details

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. 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. We then extract equirectangular frames from these videos at 3 FPS for scene 1 and 1 FPS for scene 2. Subsequently, these equirectangular frames are cube-mapped to  $768 \times 768$  sized images for the SfM step. We use sparse reconstruction in COLMAP [27] 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.* [16].

### 4.3 Results

**Scene 1 - Cory Hall** We compare our method with [20] in terms of rendered image quality and computation time. We use peak-signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) for image quality assessment.

In Tab. 1, we quantitatively compare three versions of our method, *1S1B*, *1S4B*, *4S4B* with [20] using PSNR and SSIM. As seen, all three versions of our approach outperform [20] by as much as  $13.88dB$  in PSNR and 0.22 in SSIM. Our



**Fig. 6:** Drone trajectories for all three floors of Hearst Memorial Mining Building. (a) 1F: the two loops in front is in the empty space of the first floor and the diagonal pathways towards the back are along the staircases leading to 2F. (b) 2F: the drone flew 2 loops along the corridor and on the staircases leading from 1F and to 3F. (c) 3F: similar to 2F but only one staircase leading from 2F.

Method	# Flight Captures	# Blocks	Frame Size	# Frames	PSNR↑	SSIM↑	Time↓(hrs)	# Pixels
[20]	1	5	1360 × 765	2000	20.76	0.74	25	2 B
<i>1S1B</i> (w/o DAC)	1	1	768 × 768	2552	31.31	0.95	3	1.5 B
<i>1S4B</i>	1	4	768 × 768	2552	<b>34.03</b>	<b>0.96</b>	<b>0.5+4·(0.5)</b>	<b>1.5 B</b>
<i>4S4B</i>	4	4	768 × 768	5632	<b>34.64</b>	<b>0.96</b>	<b>0.5+4·(0.5)</b>	<b>3.3 B</b>

**Table 1:** Quantitative comparison between pipelines.

method *1S4B* significantly outperforms [20] with an increase of 13.3dB 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 75% of [20]. Moreover, our method takes much less computation time overall, taking only 2.5 hours as opposed to 25 hours in [20], a 10 times speed improvement. Furthermore, if parallel processing is employed, our pipeline can complete scene reconstruction in as little as 1 hour.

As seen in Tab. 1, *4S4B* achieves 0.61dB higher PSNR than *1S4B*. This is likely 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 hours in pipeline time utilizing DAC. More importantly, DAC allows time per-block shown in brackets in Tab. 1 to be parallelized for more significant time reduction in our method.

A more qualitative comparison between our *1S4B* and [20] can be seen in Fig. 7. Column 1 shows that our method results in much sharper reconstruction than [20]. In corridors shown in column 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 column 3 when rendering “backwards” relative to the drone trajectory. Lastly, as seen in column 4, the use of 3DGS allows dynamic reflective surfaces.



**Fig. 7:** Novel-view renderings of *1S4B* (top) and [20] (bottom).



**Fig. 8:** Consecutive renderings at timestamps  $\mathbf{T} - \mathbf{1}$  and  $\mathbf{T}$  at block boundary with different data capture protocols. Note *1S4B* and *4S4B* have different block boundary since *1S4B* is automatically divided.

Fig. 8 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*. *4S4B* video can be found at Footnote 2.

**Scene 2 - Hearst Memorial Mining Building (HMMB)** We similarly compute PSNR and SSIM on the HMMB dataset, shown in Tab. 2. Here, time per-block shown in brackets can be parallelized due to our DAC method. The metrics for HMMB are lower than that of Cory Hall since HMMB is more complex compared to Cory Hall, with multiple floors, corridors and staircases. This increases the complexity of SfM, giving 3DGS a noisier initialization. Fig. 9 shows our qualitative rendering of HMMB at different perspectives. Fig. 10 shows consecutive renderings at block boundaries of HMMB. As seen, our proposed

Method	PSNR↑	SSIM↑	Time↓	# Pixels
<i>3S3B</i>	27.74	0.89	1+3·(6) hrs	7.1 B

**Table 2:** Quantitative results on Hearst Memorial Mining Building.

coarse-to-fine alignment and on-demand block rendering technique for complex data captures successfully aligns neighboring blocks and loads the block almost instantly. Video rendering on the HMMB scene can be found at Footnote 3.

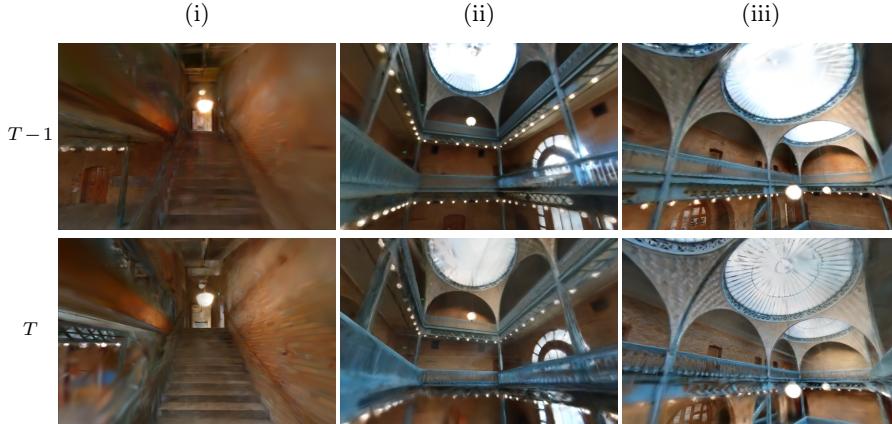
**Fig. 9:** Novel-view renderings of the Hearst Memorial Mining Building scene.

#### 4.4 Ablation Studies

Method	Block 0		Block 1		Block 2		Block 3		Avg.	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Inpainting	35.13	0.9651	33.76	0.9602	33.78	0.9645	35.89	0.9679	<b>34.64</b>	<b>0.9644</b>
Masking	34.10	0.9632	30.23	0.9521	30.24	0.9558	32.85	0.9620	<b>31.85</b>	<b>0.9583</b>

**Table 3:** Comparison of image quality between inpainting and masking on *4S4B*.

**Inpainting** 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 Sec. 3.3, and input them into SfM to mask the drone body out when doing feature matching. Then, we modify the original 3DGS code, adding



**Fig. 10:** Consecutive renderings at block boundaries of HMMB.  $\mathbf{T} - \mathbf{1}$  corresponds to the previous floor and  $\mathbf{T}$  corresponds to the next floor.

masking option for training to ignore the region with the drone body when computing loss. Quantitative results are presented in Tab. 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. We reason that inpainting explicitly introduces pixel information across frames for novel-view optimization, while masking does not propagate information in the masked area.

**Block alignment error** We evaluate the block alignment error of our coarse-to-fine method on *4S4B* Scene 1 Cory Hall using root mean squared error (RMSE) in meters. We compute their RMSE with respect to a “ground truth” obtained as follows: we input the frames of the overlapping regions from both blocks into COLMAP and hand-tune the COLMAP parameters to ensure a good reconstruction and pose recovery. RMSE is then computed per frame by taking the difference in the translation between the ground truth pose and the aligned pose.

Method	RMSE ↓(coarse + fine alignment)				Avg. Ratio↓	Time↓
	0→1	1→2	2→3	Avg.		
CA-DC	0.0399 m	0.0577 m	0.1442 m	0.0806 m	0.4344	20 min
CA-FM	0.0866 m	0.0387 m	0.1477 m	0.0910 m	0.4813	5 min

**Table 4:** Block alignment error on *4S4B*. CA-FM: coarse alignment via feature matching, used in our pipeline. CA-DC: coarse alignment via direct COLMAP. RMSE is computed after coarse alignment and fine alignment for both methods.

Given our two-stage coarse-to-fine block alignment strategy, we experimented with two different methods in the coarse alignment stage shown in Tab. 4: (1) our proposed coarse alignment via feature matching as described in Sec. 3.5, called CA-FM, and (2) a direct coarse alignment strategy via COLMAP, which we call CA-DC. CA-DC takes the last frame of the preceding block  $f_{\text{last}}^{\text{prev}}$  and the frames in the overlapping region of the succeeding block  $f_i^{\text{next}}$  and runs COLMAP to generate the relative pose of  $f_{\text{last}}^{\text{prev}}$  to  $f_i^{\text{next}}$ . This relative pose provides a good coarse estimate of the transformation between the two blocks. We similarly compute the relative pose the other way round with  $f_i^{\text{next}}$  and  $f_i^{\text{prev}}$  and take the average between the two computed poses as the final coarse transformation. After coarse alignment, we run ICP as proposed in Sec. 3.5.

Our experiments show that while CA-DC has a lower RMSE score than CA-FM, it is much slower than CA-FM since it performs SfM to estimate relative poses. We also include an average ratio metric to the aid understanding of the RMSE, which we define as the ratio of the alignment error to the average distance between each successive image frame. For both methods, the average ratio is less than 0.5, which means that the aligned frames are less than half a frame off their ground truth pose. Since the difference in RMSE score and average ratio between the two methods are minimal, we use CA-FM as our coarse alignment strategy.

#### 4.5 Limitations

One limitation is that SfM requires careful parameter adjustment and our pipeline time and novel-view synthesis result is partly bottle-necked by its performance. Another limitation comes from inpainting areas with high image frequencies. The inpainted result might not be multi-view accurate, producing visual artifacts in the reconstructed Gaussian Splat. Moreover, at block boundaries, color differences may occur since each block is trained separately, and viewing discrepancies can arise if occlusion is present, as in Fig. 10. Image blending techniques with appropriate weights [6, 32] could be applied to improve render quality at block boundaries. Lastly, at indoor corners where wall textures are plain and poorly lit, there could still be erroneous feature matching, leading to foggy rendering. Post-processing methods could be investigated to mitigate this problem.

### 5 Conclusions

We present an efficient and scalable pipeline for large-scale indoor novel-view synthesis, starting from data capture to scene reconstruction. Our usage of the 360° camera captures diverse viewpoints, making drone piloting and data capture much easier than with a standard perspective camera. By utilizing a divide-and-conquer strategy, our pipeline is parallelizable and scales up to large complex indoor scenes. Our experiments show marked improvement in both quality and speed over prior methods on the same scene. In addition, we evaluated our pipeline on a large, multi-story, building-scale scene and achieved photorealistic reconstruction results, demonstrating our pipeline’s overall effectiveness.

## References

1. Aliev, K.A., Sevastopolsky, A., Kolos, M., Ulyanov, D., Lempitsky, V.: Neural point-based graphics (2020)
2. Besl, P., McKay, N.D.: A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14**(2), 239–256 (1992). <https://doi.org/10.1109/34.121791>
3. Byravan, A., Humplik, J., Hasenclever, L., Brussee, A., Nori, F., Haarnoja, T., Moran, B., Bohez, S., Sadeghi, F., Vujatovic, B., Heess, N.: Nerf2real: Sim2real transfer of vision-guided bipedal motion skills using neural radiance fields (2022)
4. Byrd, R.H., Lu, P., Nocedal, J., Zhu, C.: A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing* **16**(5), 1190–1208 (1995). <https://doi.org/10.1137/0916069>, <https://doi.org/10.1137/0916069>
5. Chen, G., Kua, J., Shum, S., Naikal, N., Carlberg, M., Zakhori, A.: Indoor localization algorithms for a human-operated backpack system. In: *3D Data Processing, Visualization, and Transmission 2010*. vol. 3 (2010)
6. Chen, Y., Lee, G.H.: Scalar-nerf: Scalable large-scale neural radiance fields for scene reconstruction (2023)
7. Chen, Z., Yang, L., Lai, J.H., Xie, X.: Cunerf: Cube-based neural radiance field for zero-shot medical image arbitrary-scale super resolution. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. pp. 21185–21195 (October 2023)
8. Corso, N., Zakhori, A.: Indoor localization algorithms for an ambulatory human operated 3d mobile mapping system. *Remote Sensing* **5**(12), 6611–6646 (2013). <https://doi.org/10.3390/rs5126611>, <https://www.mdpi.com/2072-4292/5/12/6611>
9. Croce, V., Caroti, G., De Luca, L., Piemonte, A., Véron, P.: Neural radiance fields (nerf): Review and potential applications to digital cultural heritage. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* **XLVIII-M-2-2023**, 453–460 (2023). <https://doi.org/10.5194/isprs-archives-XLVIII-M-2-2023-453-2023>, <https://isprs-archives.copernicus.org/articles/XLVIII-M-2-2023/453/2023/>
10. Deng, N., He, Z., Ye, J., Duinkharjav, B., Chakravarthula, P., Yang, X., Sun, Q.: Fov-nerf: Foveated neural radiance fields for virtual reality. *IEEE Transactions on Visualization and Computer Graphics* **28**(11), 3854–3864 (2022). <https://doi.org/10.1109/TVCG.2022.3203102>
11. EDF R&D: Cloudcompare (2003), <https://cloudcompare.org/>
12. Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxtels: Radiance fields without neural networks. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 5501–5510 (June 2022)
13. Furukawa, Y., Ponce, J.: Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **32**(8), 1362–1376 (2010). <https://doi.org/10.1109/TPAMI.2009.161>
14. Gu, J., Jiang, M., Li, H., Lu, X., Zhu, G., Shah, S.A.A., Zhang, L., Bennamoun, M.: Ue4-nerf: neural radiance field for real-time rendering of large-scale scene (2023)
15. Kang, Y., Xu, Y., Chen, C.P., Li, G., Cheng, Z.: 6: Simultaneous tracking, tagging and mapping for augmented reality. In: *SID Symposium Digest of Technical Papers*. vol. 52, pp. 31–33. Wiley Online Library (2021)

16. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics* **42**(4) (July 2023), <https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/>
17. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., Dollár, P., Girshick, R.: Segment anything. arXiv:2304.02643 (2023)
18. Kua, J., Corso, N., Zakhor, A.: Automatic loop closure detection using multiple cameras for 3D indoor localization. In: Bouman, C.A., Pollak, I., Wolfe, P.J. (eds.) Computational Imaging X. vol. 8296, p. 82960V. International Society for Optics and Photonics, SPIE (2012). <https://doi.org/10.1117/12.916639>, <https://doi.org/10.1117/12.916639>
19. Li, H., Yi, P., Liu, Y., Zakhor, A.: 3D Indoor Mapping of Buildings with Drones. Master's thesis, University of California, Berkeley (2023)
20. Li, H., Yi, P., Liu, Y., Zakhor, A.: Scalable mav indoor reconstruction with neural implicit surfaces. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops. pp. 1544–1552 (October 2023)
21. Liu, T., Carlberg, M., Chen, G., Chen, J., Kua, J., Zakhor, A.: Indoor localization and visualization using a human-operated backpack system. In: 2010 International Conference on Indoor Positioning and Indoor Navigation. pp. 1–10 (2010). <https://doi.org/10.1109/IPIN.2010.5646820>
22. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *International journal of computer vision* **60**, 91–110 (2004)
23. Matterport Inc.: Matterport (2017), <https://matterport.com/>
24. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV (2020)
25. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.* **41**(4), 102:1–102:15 (Jul 2022). <https://doi.org/10.1145/3528223.3530127>, <https://doi.org/10.1145/3528223.3530127>
26. Rückert, D., Franke, L., Stamminger, M.: Adop: Approximate differentiable one-pixel point rendering. *ACM Transactions on Graphics (ToG)* **41**(4), 1–14 (2022)
27. Schönberger, J.L., Zheng, E., Frahm, J.M., Pollefeys, M.: Pixelwise view selection for unstructured multi-view stereo. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) Computer Vision – ECCV 2016. pp. 501–518. Springer International Publishing, Cham (2016)
28. Skydio Inc.: Skydio 3d scan, <https://www.skydio.com/3d-scan>
29. Song, Y.: Deep learning applications in the medical image recognition. *American Journal of Computer Science and Technology* **2**(2), 22–26 (Jul 2019). <https://doi.org/10.11648/j.ajcst.20190202.11>, <https://doi.org/10.11648/j.ajcst.20190202.11>
30. Song, Y., Arora, P., Singh, R., Varadharajan, S.T., Haynes, M., Starner, T.: Going blank comfortably: Positioning monocular head-worn displays when they are inactive. In: Proceedings of the 2023 ACM International Symposium on Wearable Computers. p. 114–118. ISWC '23, Association for Computing Machinery, New York, NY, USA (Oct 2023). <https://doi.org/10.1145/3594738.3611375>, <https://doi.org/10.1145/3594738.3611375>
31. Song, Y., Arora, P., Varadharajan, S.T., Singh, R., Haynes, M., Starner, T.: Looking from a different angle: Placing head-worn displays near the nose. In: Proceedings of the Augmented Humans International Conference 2024. p. 28–45. AHs '24,

- Association for Computing Machinery, New York, NY, USA (Apr 2024). <https://doi.org/10.1145/3652920.3652946>, <https://doi.org/10.1145/3652920.3652946>
32. Tancik, M., Casser, V., Yan, X., Pradhan, S., Mildenhall, B., Srinivasan, P.P., Barron, J.T., Kretzschmar, H.: Block-nerf: Scalable large scene neural view synthesis (2022)
  33. Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Kerr, J., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., McAllister, D., Kanazawa, A.: Nerfstudio: A modular framework for neural radiance field development. In: ACM SIGGRAPH 2023 Conference Proceedings. SIGGRAPH '23 (2023)
  34. Turki, H., Ramanan, D., Satyanarayanan, M.: Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs (2022)
  35. Turner, E., Cheng, P., Zakhor, A.: Fast, automated, scalable generation of textured 3d models of indoor environments. IEEE Journal of Selected Topics in Signal Processing **9**(3), 409–421 (2015). <https://doi.org/10.1109/JSTSP.2014.2381153>
  36. Turner, E., Zakhor, A.: Watertight as-built architectural floor plans generated from laser range data. In: 2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission. pp. 316–323 (2012). <https://doi.org/10.1109/3DIMPVT.2012.80>
  37. Turner, E., Zakhor, A.: Watertight planar surface meshing of indoor point-clouds with voxel carving. In: 2013 International Conference on 3D Vision - 3DV 2013. pp. 41–48 (2013). <https://doi.org/10.1109/3DV.2013.14>
  38. Wang, H., Kearney, J., Atkinson, K.: Robust and efficient computation of the closest point on a spline curve. In: Proceedings of the 5th International Conference on Curves and Surfaces. pp. 397–406 (2002)
  39. Wiles, O., Gkioxari, G., Szeliski, R., Johnson, J.: SynSin: End-to-end view synthesis from a single image. In: CVPR (2020)
  40. Yang, X., Kang, Y., Yang, X.: Retargeting destinations of passive props for enhancing haptic feedback in virtual reality. In: 2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW). pp. 618–619. IEEE (2022)
  41. Yifan, W., Serena, F., Wu, S., Öztireli, C., Sorkine-Hornung, O.: Differentiable surface splatting for point-based geometry processing. ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA) **38**(6) (2019)
  42. Zhang, B., Zhang, X., Chen, X., Fang, Y.: Grid map guided indoor 3d reconstruction for mobile robots with rgbd sensors. In: 2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM). pp. 498–503 (2018). <https://doi.org/10.1109/AIM.2018.8452296>
  43. Zhou, S., Li, C., Chan, K.C., Loy, C.C.: ProPainter: Improving propagation and transformer for video inpainting. In: Proceedings of IEEE International Conference on Computer Vision (ICCV) (2023)