3D Indoor Mapping of Buildings with Drones

Haoda Li, Puyuan Yi, Yunhao Liu

May 12, 2023

Executive Summary

Drones have been widely used in construction, agriculture, and urban planning by providing a fast, easy, and efficient geometry mapping mechanism. In recent years, drones have been made smaller, and hence are able to fly through and capture indoor environments. Compared to humans, drones' maneuverability provides opportunities for indoor mapping on the fly. However, the indoor environment poses unique challenges for drones, including the absence of GPS positioning, limited payloads, and the lack of texture features. In this project, we examined both potentials and limitations of indoor drone mapping and proposed an end-to-end system for quick and efficient indoor mapping using a drone.

In order to tackle these challenges, we propose a novel approach that leverages both the traditional structure-from-motion method and the cutting-edge deep learning approaches. To evaluate the effectiveness of the proposed method, we captured videos inside a large-scale, complex indoor scene using commercially available drones. The experimental results demonstrate that our proposed method can robustly model indoor environments using drone-captured RGB images. Specifically, our pipeline produces high-quality 3D models of two stories of an academic building from a twenty-minute video capture.

Currently, our pipeline is limited by the processing speed, whose components will each take a few hours to reconstruct a single 3D model. For future work, we will optimize the pipeline to achieve real-time positioning and modeling. In addition, our approach fits seamlessly with robotics and automation. With the current advancements in autonomous drone piloting, it is foreseeable that the indoor mapping task can be made fully automated with drones in the near future.

Contents

Contents **i**

1 Introduction **1**

2 Related Work **3**
2.1 Image based Reconstruction 3
2.2 Implicit neural representation 4

3 Proposed Approach **6**
3.1 Structure from Motion 7
3.2 2D segmentation 7
3.3 Neural Surface Reconstruction 8
3.4 Geometry Extraction and Polygonal Surface Fitting 12

4 Experiments **15**
4.1 Data Capture 15
4.2 Implementation 20
4.3 Results 20

5 Discussion and Limitations **30**
5.1 Limitations 30
5.2 Future Work 30

6 Conclusion **32**

Bibliography **33**

Introduction

Miniature unmanned aerial vehicles (UAV), or more widely known as "drones", offer unique benefits by taking high-quality aerial images with precise GPS positioning. Many industries have adopted UAV technologies to reconstruct 3D maps and models. Currently, most applications are focused on outdoor scenes. In recent years, the availability of smaller and cheaper drones has created an opportunity for indoor 3D reconstruction applications such as recovering floor plans of buildings. Using drones for indoor 3D reconstruction has many advantages over traditional human-based scanning. Firstly, the smaller size of these drones allows for greater flexibility and maneuverability in indoor spaces, enabling them to capture images from a variety of positions and perspectives. Secondly, their lightweight design means they can be operated safely and easily in confined spaces. Finally, the cost of acquiring and operating drones is relatively low compared to existing devices, making it a more affordable option for businesses and organizations.

However, indoor environments pose several challenges that need to be addressed.[1] One of the major challenges is the difficulty of navigating and mapping indoor environments, which can be more complex and varied than outdoor environments. In indoor spaces, drones may encounter obstacles, such as walls and furniture, that can interfere with their flight path and make it difficult to capture images from all angles. Another challenge is the need for accurate and reliable localization methods. Unlike outdoor environments, indoor spaces block GPS signals, making it challenging to accurately determine the position and orientation of the drone. This is especially important for generating accurate 3D models. Finally, low-texture and reflective surfaces, including walls, floors, and ceilings, are common in indoor environment. However, these surfaces are challenging for current 3D modeling algorithms. Specifically, traditional algorithms struggle to correctly match images on low-texture areas, hence fail to generate accurate models.

To solve these questions, we propose an end-to-end system shown in Fig.3.1 for indoor 3D reconstruction using commercially available drones without extra sensors and only a forward-looking RGB camera. We leverage traditional structures from motion (SfM) algorithms with a deep-learning-based neural surface reconstruction. We recover camera poses and reconstruct an initial 3D point cloud using traditional SfM methods,[2] then enhance the point cloud

[1] Victor Sanchez and Avideh Zakhor, "Planar 3D modeling of building interiors from point cloud data," in *2012 19th IEEE International Conference on Image Processing* (2012), 1777–1780, https://doi.org/10.1109/ICIP.2012.6467225.

[2] Johannes Lutz Schönberger et al., "Pixelwise View Selection for Unstructured Multi-View Stereo," in

model using neural surface reconstruction.[3] The neural surface approach reduces the impact of outliers and significantly enhances reconstruction quality. Following Guo's approach,[4] we further embed the Manhattan-world assumption to deal with low-texture surfaces. Specifically, we assume that all surfaces in the world are aligned with three orthonormal directions.[5] Existing neural surface methods have limited scalability and fail on large, non-rectangular scenes. As such, we employ a divide-and-conquer strategy by automatically partitioning a scene into blocks, performing block-wise neural surface reconstruction, and finally merging reconstructions with depth refinement.

We conducted an evaluation of our system on a challenging multi-floored indoor scene, composed of staircases, and corridors. The scene was captured using a commercially available drone as an approximately 20-minute video footage for both corridors and staircases. Our system is able to reconstruct this large and complex indoor scene and export an accurate 3D model. The 3D model is compatible with existing 3D viewing and editing softwares. Our experiments demonstrate the effectiveness of our system in producing scalable, robust, and efficient 3D indoor representations using only monocular images captured by a single drone. Compared to existing human-based capture, our system significantly reduces the capture time while maintaining accuracy.

Overall, our pipeline represents a significant step forward in the field of indoor 3D reconstruction, offering an effective approach for capturing and modeling complex indoor environments. As technology continues to advance, we believe that our approach will become increasingly valuable in a wide range of applications such as interior design, construction maintenance, and augmented reality.

European Conference on Computer Vision (ECCV) (2016); Johannes Lutz Schönberger and Jan-Michael Frahm, "Structure-from-Motion Revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[3]Lior Yariv et al., "Volume rendering of neural implicit surfaces," *Advances in Neural Information Processing Systems* 34 (2021): 4805–4815.

[4]Haoyu Guo et al., "Neural 3D Scene Reconstruction with the Manhattan-world Assumption," in *CVPR* (2022).

[5]James M Coughlan and Alan L Yuille, "Manhattan world: Compass direction from a single image by bayesian inference," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2 (IEEE, 1999), 941–947.

Related Work

The related work comprises two subparts: image-based reconstruction and implicit neural representation. In recent years, image-based reconstruction has gained significant attention in the field of computer vision, showing remarkable progress in reconstructing 3D geometry from 2D images. On the other hand, implicit neural representation has emerged as a promising alternative to traditional explicit surface representations in computer graphics, being successfully applied in various applications such as shape modeling and scene synthesis. The use of these techniques has been explored in various domains, including robotics, medical imaging, and entertainment. In our project, we utilize both traditional image-based reconstruction methods and implicit neural representation strategies.

2.1 Image based Reconstruction

Given a set of RGB images without location information, 3D scene reconstruction usually involves estimating per-image depth maps and fusing them into a 3D model. Structure-from-Motion (SfM) algorithms utilize feature matching to find pixel correspondence across images to compute depth.[1] Traditional methods[2] formulate the reconstruction task as an optimization problem and utilize hand-crafted metrics to refine the task. Although traditional methods show great performance in the reconstruction of single objects, they still suffer from unreliable matching correspondences under illumination change, low-texture regions, and optical reflection, which are common in the indoor environment. Some works address these issues by introducing planar priors,[3] which assumes that surfaces are mostly planar, but they still perform poorly in large-scale indoor environments.

With the success of the convolutional neural networks (CNN), many learning-based methods have been developed to address these problems and have achieved superior results to

[1] Schönberger and Frahm, "Structure-from-Motion Revisited."

[2] Schönberger et al., "Pixelwise View Selection for Unstructured Multi-View Stereo"; Yasutaka Furukawa and Jean Ponce, "Accurate, Dense, and Robust Multiview Stereopsis," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, no. 8 (2010): 1362–1376, https://doi.org/10.1109/TPAMI.2009.161.

[3] Michal Jancosek and Tomas Pajdla, "Multi-view reconstruction preserving weakly-supported surfaces," in *CVPR 2011* (2011), 3121–3128, https://doi.org/10.1109/CVPR.2011.5995693; Andrea Romanoni and Matteo Matteucci, "Tapa-mvs: Textureless-aware patchmatch multi-view stereo," in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), 10413–10422.

traditional methods. Learning-based multi-view stero (MVS) approaches[4] have become more popular in recent years. These works first use 2D CNN to extract image features and then leverage the variance equation to build a single cost volume from extracted features. Later, these approaches construct 3D CNNs or 2D RNNs for cost volume regularization and predict depth using soft `argmin`. Utilizing the 3D CNNs or 2D RNNs to finish the cost regularization and to find the cross-image correspondences, the data-driven approaches can hence address the aforementioned low-texture-region challenge, while the edges and corners are often over-smoothed.

To simplify the problem, specialized devices are used to obtain additional sensory data. These hardware can obtain high-precision positions and depth. However, such devices are often expensive and heavy. Recently, LiDAR cameras have become smaller and more accessible. Many indoor scanning applications utilize the LiDAR camera on mobile phones[5] to scan houses and create 3D models. However, the accuracy of depth estimation suffers significantly when objects are distant. In general, LiDAR cameras provide erroneous estimations when the scene becomes large and complex, hence most indoor scanning applications fail on large-scale indoor modeling.

2.2 Implicit neural representation

Alternatively, implicit neural representations describe the scene as a 3D field estimated by neural networks (NNs). Mildenhall et al.[6] leverages volume rendering to learn the implicit radiance field from images, Yariv et al.[7] and Wang et al.[8] further combine neural volume rendering with implicit surfaces to enable high-fidelity object reconstruction. These methods perform well on single objects with rich texture. However, they tend to result in erroneous or incomplete surfaces in low-texture regions common in indoor scenes. Recently, novel approaches specifically tackle indoor scene reconstruction by introducing additional priors

[4]Yao Yao et al., "MVSNet: Depth Inference for Unstructured Multi-view Stereo," *European Conference on Computer Vision (ECCV)*, 2018, Yao Yao et al., "Recurrent MVSNet for High-resolution Multi-view Stereo Depth Inference," *Computer Vision and Pattern Recognition (CVPR)*, 2019, Gu Xiaodong et al., "Cascade Cost Volume for High-Resolution Multi-View Stereo and Stereo Matching," *Computer Vision and Pattern Recognition (CVPR)*, 2020,

[5]Francesco Di Stefano et al., "Mobile 3D scan LiDAR: a literature review," *Geomatics, Natural Hazards and Risk* 12, no. 1 (2021): 2387–2429, https://doi.org/10.1080/19475705.2021.1964617.

[6]Ben Mildenhall et al., "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," in *ECCV* (2020).

[7]Yariv et al., "Volume rendering of neural implicit surfaces."

[8]Peng Wang et al., "NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction," *NeurIPS*, 2021,

such as depth,[9] geometric consistency,[10] and planar region assumptions.[11] These methods perform well in rectangular rooms, but they fail to demonstrate reasonable reconstructions in more complex indoor scenes. Our method further extends these methods beyond rectangular-shaped rooms to large multi-floor constructions.

[9]Zehao Yu et al., "MonoSDF: Exploring Monocular Geometric Cues for Neural Implicit Surface Reconstruction," *Advances in Neural Information Processing Systems (NeurIPS)*, 2022,

[10]Qiancheng Fu et al., "Geo-Neus: Geometry-Consistent Neural Implicit Surfaces Learning for Multi-view Reconstruction," *Advances in Neural Information Processing Systems (NeurIPS)*, 2022,

[11]Guo et al., "Neural 3D Scene Reconstruction with the Manhattan-world Assumption"; Yusen Wang et al., "NeuralRoom: Geometry-Constrained Neural Implicit Surfaces for Indoor Scene Reconstruction," *ACM Trans. Graph.* (New York, NY, USA) 41, no. 6 (November 2022), ISSN: 0730-0301, https://doi.org/10.1145/3550454.3555514, https://doi.org/10.1145/3550454.3555514.

Proposed Approach

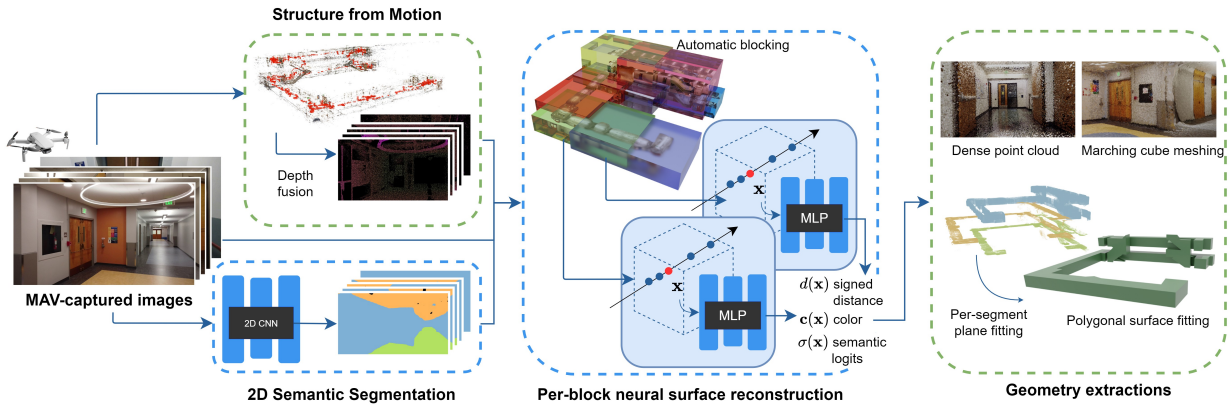


Figure 3.1: Workflow of the proposed system

Our system takes an RGB image sequence as the input and produces a color-textured 3D model as the output. The model can be exported in various forms including point cloud, 3D triangle mesh, or simplified polygonal mesh. Figure 3.1 is an overview of our proposed workflow. We first recover camera poses using Pix4D, a commercially available SfM software. Next, we recover a sparse point cloud by using poses and RGB images in COLMAP. Then, we generate 2D depth maps by projecting each point back to its corresponding image. Simultaneously, we segment out walls, floors, and ceilings in RGB input images with a 2D CNN. We choose the state-of-the-art network DeepLabV3+ as our segmentation network baseline. The model is trained on the ADE20k dataset. We set all the classes except walls, floors, and ceilings to other. The details will be shown in Section 3.2. In order to apply the neural surface reconstruction mentioned in Section 3.3 to a larger scale, a divide-and-conquer strategy is leveraged by partitioning the image sequences into blocks, based on their estimated locations. Each block is then reconstructed with a modified ManhattanSDF method[1] with 2D depth maps and segmentation supervision. Thereafter, the blocks are aligned by a depth-based refinement step to be explained later within this chapter. At this stage, the scene is represented as block-wise signed distance and appearance fields.

[1]Guo et al., "Neural 3D Scene Reconstruction with the Manhattan-world Assumption."

Finally, we extract various geometry representations for different downstream tasks. The representations include a dense color point cloud similar to other reconstruction systems, a high-fidelity mesh using marching cubes, and an accurate polygonal model using plane fitting refer to PolyFit.[2]

3.1 Structure from Motion

As has been briefly summarized in Section 2.1, the Structure from Motion (SfM) phase in the aforementioned pipeline aims to reconstruct a baseline 3D geometry from the input images. We incorporate SfM implementation by commercial software Pix4D which provides us with the camera poses. Next, we use camera poses and RGB images as input to COLMAP to create a sparse point cloud. Each point in the point cloud corresponds to places that are common across multiple images. The SfM algorithm guarantees statistically accurate estimations of point locations.[3] However, due to the sparsity of the image features in indoor scenarios, the point cloud is expected to be sparse in low-texture regions and has outliers across the scene. Directly performing surface reconstruction on such point clouds results in numerous errors such as holes in the walls and floaters in empty space, necessitating refinement with neural surface approaches to be described in the next section. To utilize the point cloud as a depth prior, we project pixels in the drone's raw images to point cloud and acquire per-image sparse depth maps.

3.2 2D segmentation

2D segmentation is a fundamental technique in computer vision that aims to divide an image into distinct regions or objects. It involves the process of assigning pixel-level labels to different parts of an image, allowing for the extraction of valuable information and the identification of specific objects or regions of interest. In section 3.3 dealing with the neural surface reconstruction, we apply Manhanttan assumptions to our neural surface. Thus, it is pivotal to help the network identify wall, ceiling, and floor regions in our given images. We adopted DeepLabV3+[4] as our baseline to help us segment out walls, floors, and ceilings. Given the segmentation results, we can later apply the geometric constraints to the regions of floors, walls, and ceilings.

[2]Liangliang Nan and Peter Wonka, "Polyfit: Polygonal surface reconstruction from point clouds," in *Proceedings of the IEEE International Conference on Computer Vision* (2017), 2353–2361.

[3]Schönberger and Frahm, "Structure-from-Motion Revisited."

[4]Liang-Chieh Chen et al., "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence* 40, no. 4 (2017): 834–848.

3.3 Neural Surface Reconstruction

Among prior work on neural scene reconstruction, we observe that embedding the Manhattan world assumption produces the most likely robust outcome for our task settings. Thus, we adopt the ManhattanSDF by Guo et al.[5] as our backbone.ManhattanSDF addresses the challenge of reconstructing 3D indoor scenes from multi-view images. Specifically, ManhattanSDF uses an MLP network to represent the signed distance function as the scene geometry. Based on the Manhattan-world assumption, planar constraints are employed to regularize the geometry in floor and wall regions predicted by a 2D semantic segmentation network. In addition to this baseline, we utilize the sparse depth maps generated from structure-from-motion as new constraints to further improve the performance. In addition, we propose a divide-and-conquer strategy to address the scalability problem while trying to reconstruct a large-scale environment.

Our neural surface reconstruction represents the scene geometry and appearance with signed distance and color fields. Specifically, given a 3D point $\mathbf{x}$, our model will map it to signed distance $d(\mathbf{x})$. The mapping function is defined as:

$$d(\mathbf{x}) = F_d(\mathbf{x}) \tag{3.1}$$

where F_d is implemented using a Multi-Layer perceptron. To estimate the color field, the appearance model takes the spatial point $\mathbf{x}$, the view direction $\mathbf{v}$, the point's corresponding normal $\mathbf{n}(\mathbf{x})$ as inputs and outputs the color $\mathbf{c}(\mathbf{x})$, which is defined as:

$$\mathbf{c}(\mathbf{x}) = F_{\mathbf{c}}(\mathbf{x}, \mathbf{v}, \mathbf{n}(\mathbf{x})) \tag{3.2}$$

where $F_{\mathbf{c}}$ is also implemented as a Multi-Layer perceptron and we obtain the normal $\mathbf{n}(\mathbf{x})$ by computing the gradient of the signed distance $d(\mathbf{x})$ at point $\mathbf{x}$. We adopt volume rendering following ManhattanSDF to render the image pixels given signed distance field and color field. After rendering the images, we can optimize the scene representation network using multi-view images with photometric loss as follows:

$$\mathcal{L}_{img} = \sum_{\mathbf{r} \in \mathcal{R}} ||C(\mathbf{r}) - \hat{C}(\mathbf{r})|| \tag{3.3}$$

where $C(\mathbf{r})$ is the ground-truth color of our captured pixel data, $\hat{C}(\mathbf{r})$ is the rendered pixel color, and $\mathcal{R}$ is the set of camera rays going through sampled pixels. In addition, we apply Eikonal loss shown as follows:

$$\mathcal{L}_E = \sum_{\mathbf{y} \in \mathcal{Y}} (||\nabla_{\mathbf{y}} d(\mathbf{y})||_2 - 1)^2 \tag{3.4}$$

where $\mathcal{Y}$ denotes the combination of points sampled along the rays and points randomly sampled from the 3D space and $d(\mathbf{y})$ is the signed distance estimated by the MLP. As shown

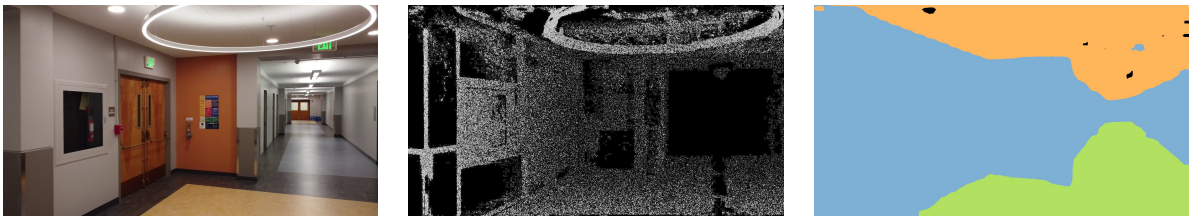
[5]Guo et al., "Neural 3D Scene Reconstruction with the Manhattan-world Assumption."

by Gropp et al.,[6] the Eikonal loss regularizes the MLP to correctly converge to a signed distance field.

We now illustrate how to utilize generated depth maps to create an additional loss function in computing the neural surface reconstruction network parameter optimization. First, we acquire per-view sparse depth maps by projecting the reconstructed point cloud back to 2D. The SfM pipeline, which estimates these points from pixel correspondence and stereo views, guarantees statistical accuracy on the predicted depths. Therefore, they are reliable for supervision. In addition, since indoor-scene images are taken inside-out, we filter out obvious outliers by a maximum depth. Next, we use the filtered depth maps and to be predicted depth maps to calculate the depth loss:

$$\mathcal{L}_D = \sum_{\mathbf{r}\in\mathcal{D}} |D(\mathbf{r}) - \hat{D}(\mathbf{r})| \tag{3.5}$$

where $\mathcal{D}$ denotes rays going through pixels with depth values, and $D(\cdot)$ denotes the depth value along each ray in $\mathcal{D}$, $D(\mathbf{r})$ is our filtered depth map, and $\hat{D}(\mathbf{r})$ is the depth predicted by the Multi-layer Perceptron of our neural surface reconstruction part. Fig.3.2 shows a captured RGB image, its filtered depth map generated from structure-from-motion, and the corresponding segmentation result.



(a) RGB image (b) sparse depth map (c) segmentation result

Figure 3.2: (a)The original drone captured RGB images; (b)Its reprojected sparse depth map; (c)The segmentation result.

Next, we develop yet another loss function for our neural surface reconstruction network parameter optimization problem, as it relates to the Manhattan world assumption. Specifically, Manhattan world assumption puts a geometric constraint on the final 3D reconstructed model by assuming floors, ceilings, and walls are mutually orthogonal. As our task primarily provides no information distinguishing floors, ceilings, and walls, to extend ManhattanSDF to our scenario, we also train a 2D semantic segmentation network[7] to mask walls, floors,

[6] Amos Gropp et al., "Implicit Geometric Regularization for Learning Shapes," in *Proceedings of Machine Learning and Systems 2020* (2020), 3569–3579.

[7] Chen et al., "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs."

and ceilings in each captured RGB image as described in Section 3.2. The masks regularize the planar region surface as follows:

$$\mathcal{L}_f = \sum_{\mathbf{r}\in\mathcal{F}} |1 - \mathbf{n}(\mathbf{x_r}) \cdot \mathbf{n}_f| \tag{3.6}$$

$$\mathcal{L}_c = \sum_{\mathbf{r}\in\mathcal{C}} |1 + \mathbf{n}(\mathbf{x_r}) \cdot \mathbf{n}_f| \tag{3.7}$$

$$\mathcal{L}_w = \sum_{\mathbf{r}\in\mathcal{W}} |\mathbf{n}(\mathbf{x_r}) \cdot \mathbf{n}_f| + \alpha \min_{i\in\{-1,0,1\}} |i - \mathbf{n}(\mathbf{x_r}) \cdot \mathbf{n}_w| \tag{3.8}$$

$$\mathcal{L}_{\text{geo}} = \mathcal{L}_f + \mathcal{L}_c + \mathcal{L}_w \tag{3.9}$$

where $\mathcal{F}, \mathcal{C}$, and $\mathcal{W}$ denote rays going through pixels masked with floor, ceiling, and wall, respectively. The function $\mathbf{n}(\mathbf{x_r})$ computes the normal at a 3D point $\mathbf{x}$ along a ray $\mathbf{r}$. The floor normal $\mathbf{n}_f$ and wall normals $\mathbf{n}_w$ are specified as:

$$\mathbf{n}_f = [\sin(\theta)\cos(\phi), \sin(\theta)\sin(\phi), \cos(\theta)] \tag{3.10}$$

$$\mathbf{n}_w = [\sin(\theta + \frac{\pi}{2})\cos(\phi), \sin(\theta + \frac{\pi}{2})\sin(\phi), \cos(\theta + \frac{\pi}{2})] \tag{3.11}$$

where θ, ϕ are initialized as 0 to result in $\mathbf{n}_f = [0, 0, 1]$ and $\mathbf{n}_w = [1, 0, 0]$ indicating the floor is initially horizontal and the wall is initially vertical and aligned with the global x axis. θ and ϕ are then optimized during training to allow deviation from the above initial condition. Intuitively, for the points on floors and ceilings, $\mathbf{n}(\mathbf{x_r})$ and $\mathbf{n}_f$ align resulting in $1 - \mathbf{n}(\mathbf{x_r}) \cdot \mathbf{n}_f$ to be close to zero for floor and ceiling points, thus minimize the loss functions $\mathcal{L}_f$ and $\mathcal{L}_c$ in Equations 3.6 and 3.7 respectively. For the second term of Equation 3.8, we design a loss that enforces the normal directions of surface points on walls to be either parallel or orthogonal to the learnable wall normal $\mathbf{n}_w$. This is intuitive because walls usually have two degrees of freedom. So the $i \in \{-1, 0, 1\}$ are different cases that represent different situations: parallel or orthogonal to $\mathbf{n}_w$. $i \in \{-1, 1\}$ means that the wall is parallel to $\mathbf{n}_w$ and $i = 0$ means the wall is perpendicular to $\mathbf{n}_w$. During training, the parameter i will be set to one of $\{-1, 0, 1\}$ which minimizes the second term of $\mathcal{L}_w$, via back-propagation. We also add another hyper-parameter $\alpha \in [0, 1]$ to soften the Manhattan world constraint, because in practice walls are not always orthogonal or parallel to $\mathbf{n}_w$.

For large scenes, neural surface reconstruction loses details and eventually becomes intractable. Therefore, we apply a divide-and-conquer strategy to make our approach scalable. We partition camera poses along the trajectory into several segments with enough overlapping poses for adjacent segments. Specifically, each segment is bounded within a $30m \times 30m \times 5m$ box and the overlapping volume between each pair of segments is at least $10m \times 10m \times 5m$. An example partition is shown in Fig.3.3. Thereafter we initialize neural surface reconstruction on each block. To connect the outcomes from each block and overcome the mismatches between adjacent overlapping blocks, we propose a depth-based boundary refinement step to align each pair of implicit surfaces. Specifically, for each camera view in

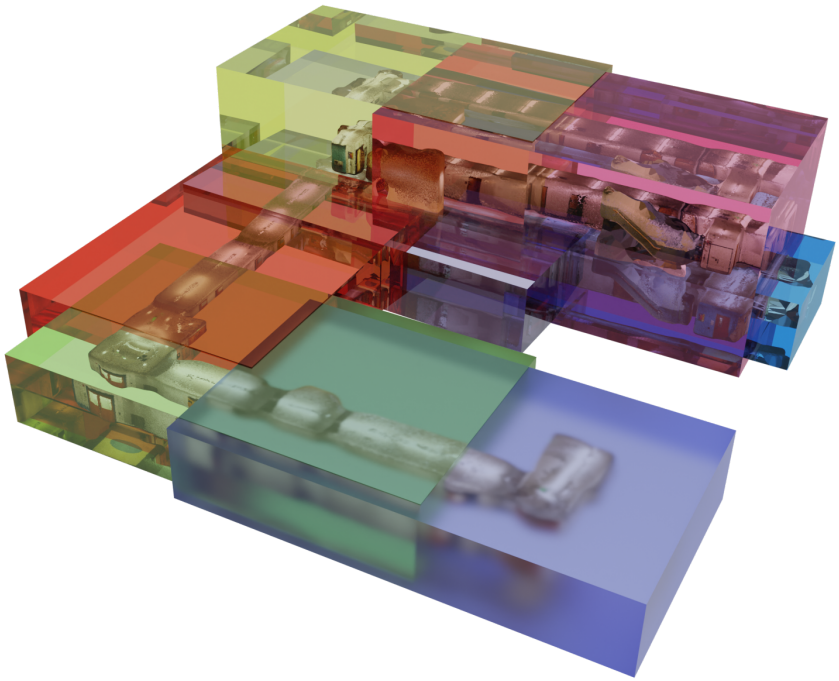


Figure 3.3: An example partition result

the overlapping area of blocks i and j, we calculate a new loss term using their respective predicted depth maps:

$$\mathcal{L}_{D,i} = \sum_{\mathbf{r}\in\mathcal{D}} |\hat{D}_i(\mathbf{r}) - \hat{D}_j(\mathbf{r})| \tag{3.12}$$

where $\mathcal{D}$ denotes rays going through pixels with depth values, and $D(\cdot)$ denotes the depth value along each ray in $\mathcal{D}$, and $\hat{D}_i(\mathbf{r})$ and $\hat{D}_j(\mathbf{r})$ are the depth estimations for blocks i and j, respectively. We utilize this loss term to align each of our blocks so that their predicted depth maps are as close as possible. This loss term is utilized in the boundary refinement step, in which we will replace the original sparse depth map loss, $\mathcal{L}_D$ in Eq. 3.5, with this new loss term. The details of the training strategy are illustrated in Sec. 4.2.

In summary, our final loss $\mathcal{L}_{total}$ is illustrated as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{\mathcal{D}}(\mathcal{L}_{D,i}) + \mathcal{L}_{geo} + \mathcal{L}_{img} + \mathcal{L}_E \tag{3.13}$$

where $\mathcal{L}_D$ denotes the loss calculated from sparse depth map constraints from structure-from-motion sparse depth maps, $\mathcal{L}_{D,i}$ denotes depth-based boundary refinement loss which will replace the sparse depth map loss $\mathcal{L}_D$ and only be utilized in the boundary refinement process, $\mathcal{L}_{geo}$ denotes the Manhanttan assumption regularization loss, $\mathcal{L}_{img}$ means the photometric loss and $\mathcal{L}_E$ denotes the Eikonal loss. Fig. 3.4 is an example of our final signed distance field visualization for one block.

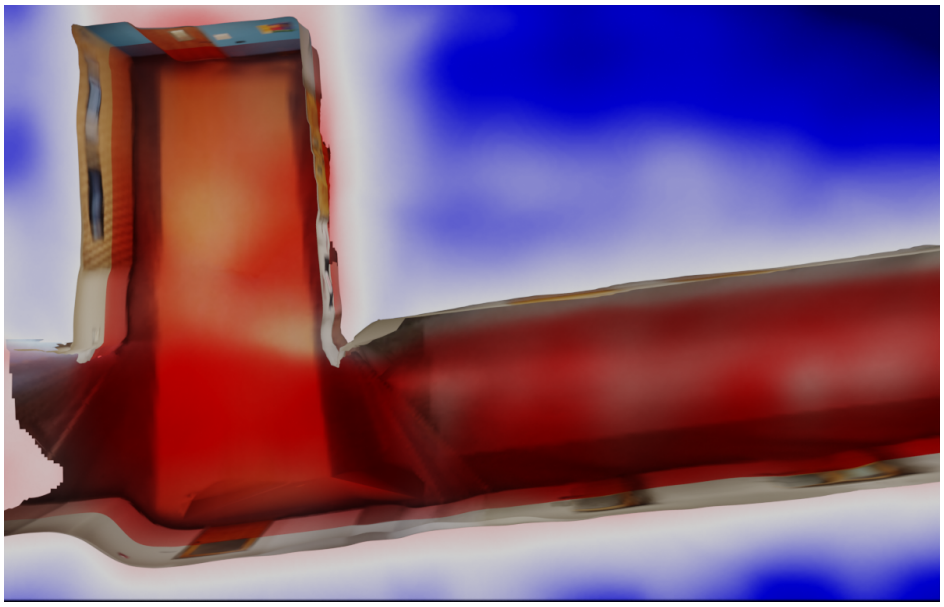


Figure 3.4: A cross-section of signed distance field, the boundary regions of the surface are shown in white color. Blue is used to represent points located far outside the surface and red represents points located inside the surface, where the signed distance values are negative and relatively large.

3.4 Geometry Extraction and Polygonal Surface Fitting

For a signed distance field, the surface is the zero-level set

$$\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 : d(\mathbf{x}) = 0\} \tag{3.14}$$

Therefore, To extract an explicit geometry, we bound the whole scene using a 3D grid and estimate the signed distance at each grid point. Specifically, we find all blocks that bound the grid point and query the corresponding neural surface reconstruction models. Because the blocks are overlapped, each grid point must be evaluated by more than one model, hence we assign the minimum signed distance to the grid point. Thereafter, we use the marching cubes algorithm[8] to find the zero-level set and extract the mesh from the grid. An example extracted mesh of our single block is shown in Fig.3.5(a).

Alternatively, we simply cast rays from each camera view. For each ray $\mathbf{r}$, our neural reconstruction model estimates its color $\tilde{C}(\mathbf{r})$ and depth $\tilde{D}(\mathbf{r})$. Therefore, we obtain a set of

[8]William E. Lorensen and Harvey E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *SIGGRAPH Comput. Graph.* (New York, NY, USA) 21, no. 4 (August 1987): 163–169, ISSN: 0097-8930, https://doi.org/10.1145/37402.37422, https://doi.org/10.1145/37402.37422.

RGBD images. We can simply generate a dense point cloud by fusing these RGBD images. For example, we can do depth fusion by truncated signed distance integration.[9] An example of an extracted point cloud of our single block is shown in Fig.3.5(b).

Polygonal surface reconstruction generates simplified models from point clouds. The method is suitable for many downstream tasks in man-made environments by eliminating unnecessary surface details while preserving sharp features. This process involves analyzing the spatial relationship between the input points, estimating surface normals, and connecting adjacent points to form triangles or polygons. In our project, we applied PolyFit[10] to find plane boundaries and to assemble planes into a polygonal model. PolyFit uses a combination of local fitting and global optimization to construct a polygonal mesh that approximates the input point cloud. The local fitting step involves fitting a polynomial surface to a set of neighboring points, while the global optimization step minimizes the energy of the entire mesh with respect to the input point cloud. For our experiments, we do not use PolyFit to fit the planar model block by block, instead, we concatenate our point cloud and utilize our whole point cloud to directly fit our final planar model. Our generated planar model is shown in Fig. 4.11(a).

[9]Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun, "Open3D: A Modern Library for 3D Data Processing," *arXiv:1801.09847*, 2018,

[10]Nan and Wonka, "Polyfit: Polygonal surface reconstruction from point clouds."

(a) An extracted mesh on our single block

(b) An extracted point cloud on our single block

Figure 3.5: We extracted mesh and point cloud from a single block using the marching cubes algorithm and ray casting method.

Experiments

4.1 Data Capture

We deployed a DJI Mini 2 drone, which only weighs 249g, to capture videos. The drone traversed a trajectory whereby the optical axis of its camera was varying over time from being perpendicular to parallel to the main axis of the hallway. The reason for this flying trajectory is illustrated in the next paragraph. All videos are captured in 2720×1530 at 24 frames per second(fps). Our example reconstruction used three clips captured in two flights for a total of 20 minutes. Frames are extracted at 2 fps. We further filtered out overly blurred images by a threshold over the variance of image Laplacian. We utilize OpenCV Laplacian function to calculate the image Laplacian. If the image Laplacian is lower than 120, we assert the image is blurry. We also removed redundant views by a threshold of 2 on distances among estimated poses. After filtering, the remaining 1971 frames were extracted and resized to 1360×765. Our camera trajectory and the sparse point cloud generated from structure-from-motion are shown in Fig.4.1. As the sample scene spans across multiple floors, we captured in three flights: in **Path 1**, we flew the drone along the 3rd floor's corridor in a U-shaped fashion; in **Path 2**, the drone entered the stairwell on the third floor, flew along the stairs and captured a portion of the 4th floor due to the battery capacity. In the last **Path 3**, the drone entered another stairwell on the third floor, flew upwards to the 4th floor and towards the region that had been covered by Path 3, after which it turned back to scan towards the other end of the 4th floor's corridor. A bird-eye view of our drone's third-floor trajectory is shown in Fig.4.3 and the fourth-floor trajectory is shown in Fig.4.4.

At first, we tried to scan the geometry with the canonical drone trajectory along which the optical axis of the drone's camera was kept perpendicular to the walls while the drone traverses the corridors. The method, illustrated in Fig. 4.2 (b), would create a smoothly sliding window, and the perpendicularity minimizes the impact of perspective views and lens distortion and hence is widely adopted for scanning tasks like document scanning. However, it is not recommended. Specifically, the method failed in our case due to the limited dimensions of the indoor environment. In order for computer vision and the neural network to capture sufficient feature points to obtain geometry, the drone needs to be far away enough from the target wall. Yet, indoor scenes such as a narrow hallway will not allow for such a distance. Hence, we pointed the drone camera to the main axis of the hallway so that the optical axis is parallel to the direction of the motion, as illustrated in Fig. 4.2 (a). Since

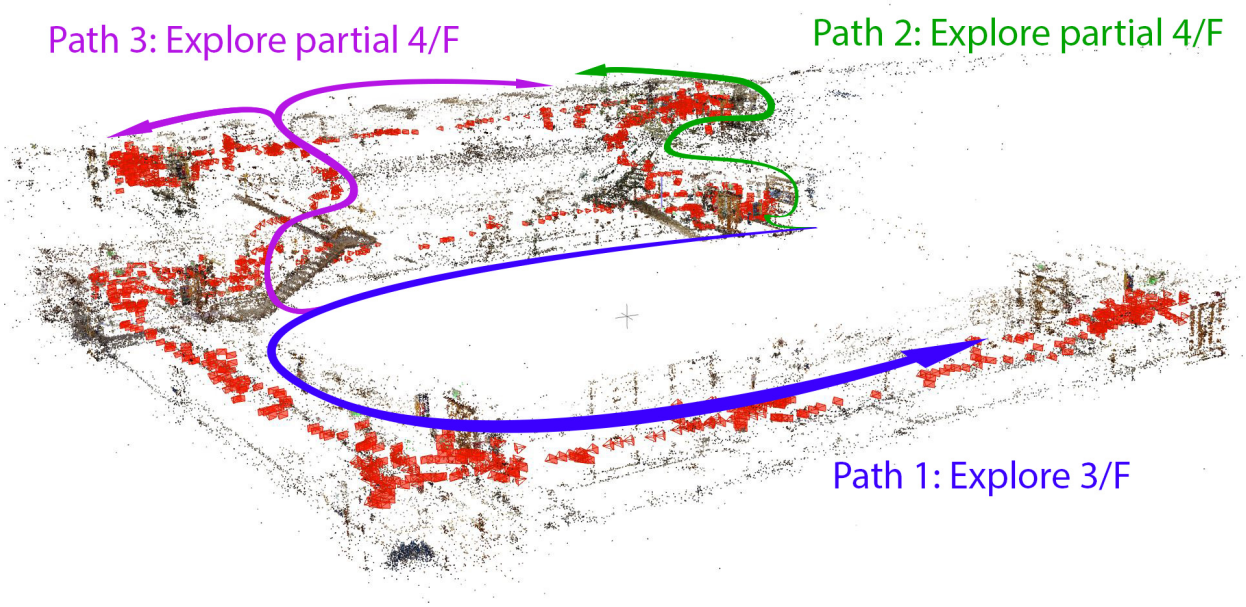
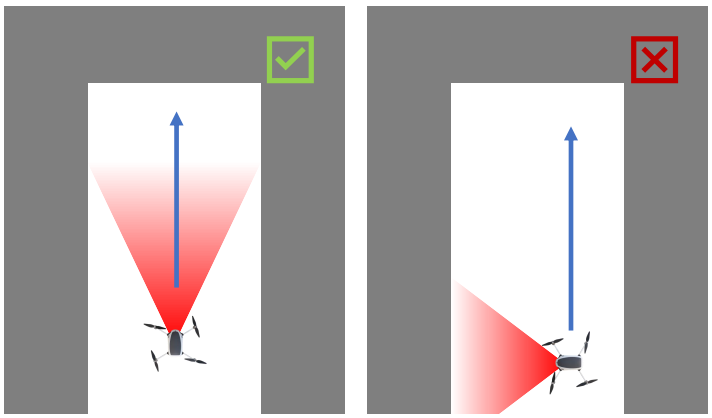


Figure 4.1: Our camera trajectory and Structure-from-motion generated sparse point cloud



(a) (b)

Figure 4.2: Drone's camera shall be pointed (a) parallel and (b) perpendicularly to corridor walls.

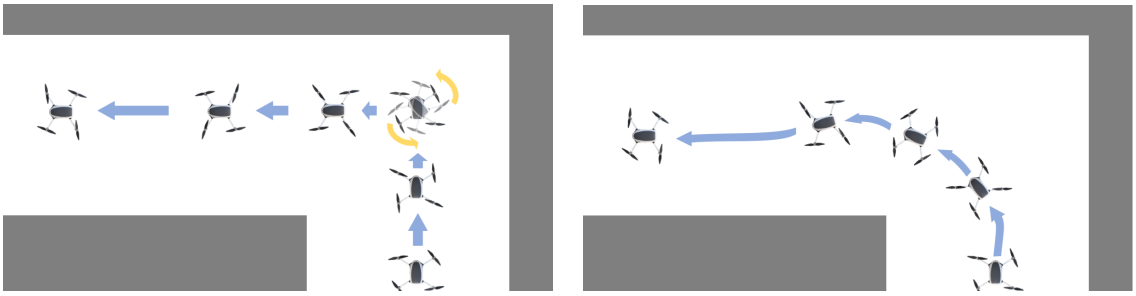
a hallway, despite its narrowness, usually extends relatively long, this change of camera direction, therefore, allows for the required scanning distance.

Figure 4.3: Our third floor's drone trajectory

However, a camera parallel to the hallway leads to a focus issue, as now the distances between the camera and those pixels belonging to the wall vary over a wide range. The camera automatically sets the focus at the end of the hallway, which can lead to a significant blur of the nearer regions. An approach to alleviate the issue is to slightly sway the drone's camera direction side-to-side, which explains the wavy trajectory in Figs.4.3 and 4.4. Additionally, the robustness of the structure from motion and the 2D segmentation models may also help to resolve the blur, which can be ensured by the data augmentation technique used in model training and tuning. Moreover, the neural network we utilized in the following parts of our pipeline also possesses the capability of accepting inputs of lower quality.

Furthermore, we discovered the structure from motion models tackle the turning of cameras around corners unsatisfactorily. When the drone approaches a corner, instead of adopting a "stop-spin-and-go" trajectory as shown in Fig. 4.5(a), we therefore utilize a smooth

Figure 4.4: Our fourth floor's drone trajectory



(a) A spinning trajectory

(b) A trajectory where the drone rotates with translation

Figure 4.5: A spinning one which shall be avoided (a) v.s. the desired trajectory (b)

transition between two sides of a corner. The drone rotates around the corner while keeping a forward translation, as shown in Fig. 4.5(b).

Note that the aforementioned methods of data capture work specifically for a "narrow-but-long" corridor. If the capture target is a room instead, it will suffice to directly point the drone's camera perpendicular to walls while maintaining a distance away from the walls so that portions of both the ceiling and the floor can be included together with the walls. In this case, the direction of the drone's motion is perpendicular to the optical axis of the camera. Then, the drone can simply scan the walls of the room one by one, without the

necessity to sway or rotate in the manner illustrated in Figs.4.3 and 4.4.

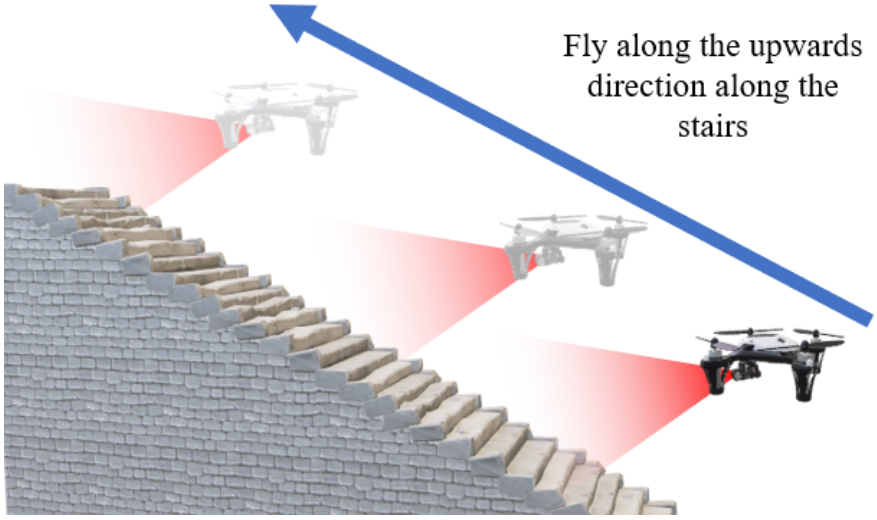


Figure 4.6: Drone trajectory for capturing in stairwells

When the drone enters a stairwell, it would be better to fly the drone **along** the stairs, while its camera is pointed at the upward direction of the stairs, as is shown in Fig. 4.6. Two issues deserve notice here: **1)** maintain an about 1m distance between the drone to the stairs surface below, and **2)** always point the camera approximately parallel to the horizontal surface of each step.

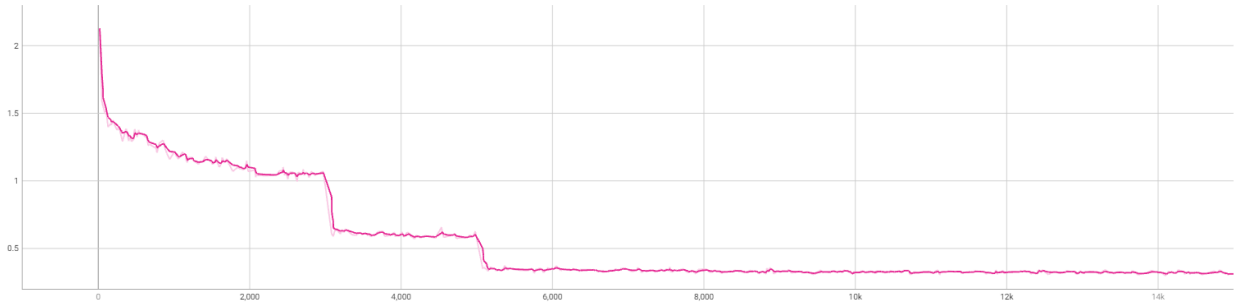


Figure 4.7: Our training convergence curve

4.2 Implementation

We used commercial software Pix4D Mapper for camera pose recovery, and subsequently COLMAP's reconstruction[1] to generate a point cloud. We used Pix4D because the software makes specific capture assumptions about drones and produces better camera calibration results. On the other hand, COLMAP provides more flexibility in doing triangulation and point-to-pixel projection. For 2D semantic segmentation, we trained a DeepLab[2] model on ADE20K dataset.[3] For neural surface reconstruction, we partitioned the space into 11 blocks, each containing 300-400 images. The training was performed on an NVIDIA TITAN RTX GPU. We first trained each block with batches of 2048 rays for $15,000$ iterations and then perform boundary refinement for $2,000$ iterations. Each block took approximately 2 hours to train. For each block, we follow the same training strategy in ManhanttanSDF. We use Adam optimizer with a learning rate of $5e-4$. In the first $1,000$ iterations, we omit the Manhattan loss $\mathcal{L}_{geo}$ to let the network quickly learn a rough shape. In addition, we apply a half decay on depth loss $\mathcal{L}_D$ at $3,000$ and $5,000$ iterations. The sparse depth is statistically accurate and provides an initial guidance for learning the general surface shape, while the minor noise and outliers can be misleading for learning the finer features. Our training convergence curve is shown in Fig.4.7. For polygonal surface modeling, our system fitted planes to each of the wall, floor, and ceiling segments as is mentioned in Section 3.4. PolyFit[4] is applied to find plane boundaries and assemble planes into a polygonal model. We compared our proposed 3D reconstruction framework with a traditional robust pipeline COLMAP[5] with its built-in Poisson meshing.[6] We list our procedure time for each part in our pipeline in Tab.4.1. It is worth mentioning that from Pix4D pose recovery to neural surface reconstruction, the processing for each block is independent so we can run the algorithms for all blocks together in parallel. In our experiments, we ran our 11 blocks one by one. For PolyFit, cause it directly creates a manifold and watertight surface model, it can only be applied to the whole generated point cloud. So we list the processing time for our total point cloud of the PolyFit method.

4.3 Results

In this section, we compare our method with COLMAP in terms of dense point cloud quality, meshing, and polygonal surface fitting. As seen in Fig.4.8, the point cloud from our method

[1]Schönberger et al., "Pixelwise View Selection for Unstructured Multi-View Stereo."

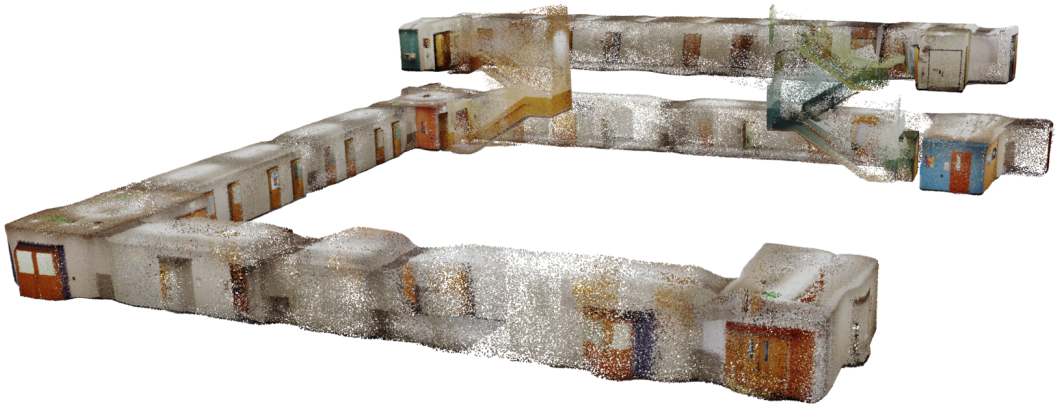
[2]Chen et al., "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs."

[3]Bolei Zhou et al., "Semantic understanding of scenes through the ade20k dataset," *International Journal of Computer Vision* 127, no. 3 (2019): 302–321.

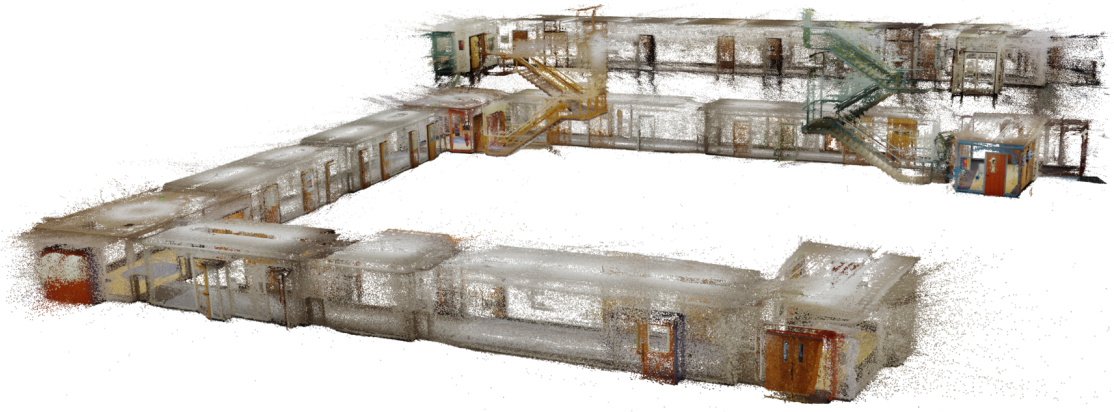
[4]Nan and Wonka, "Polyfit: Polygonal surface reconstruction from point clouds."

[5]Schönberger et al., "Pixelwise View Selection for Unstructured Multi-View Stereo."

[6]Michael Kazhdan and Hugues Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics (ToG)* 32, no. 3 (2013): 1–13.

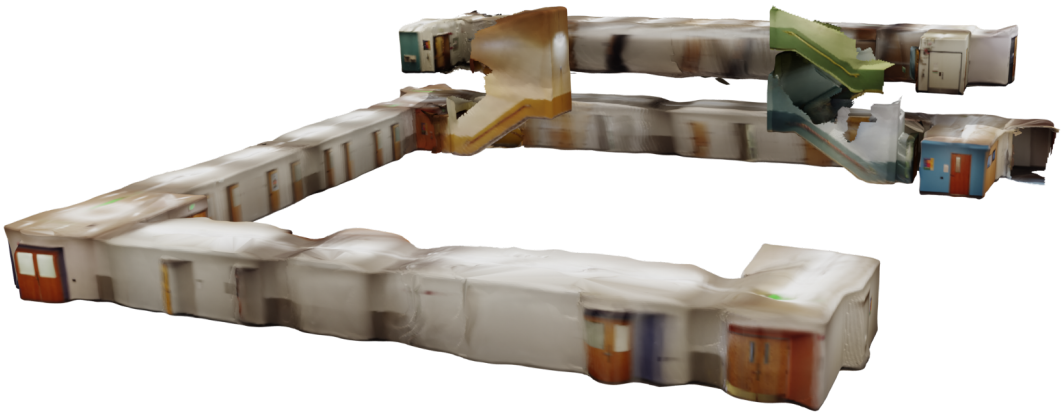


(a) Our dense point cloud



(b) COLMAP dense point cloud

Figure 4.8: Dense point cloud comparison between ours and COLMAP



(a) Our generated mesh

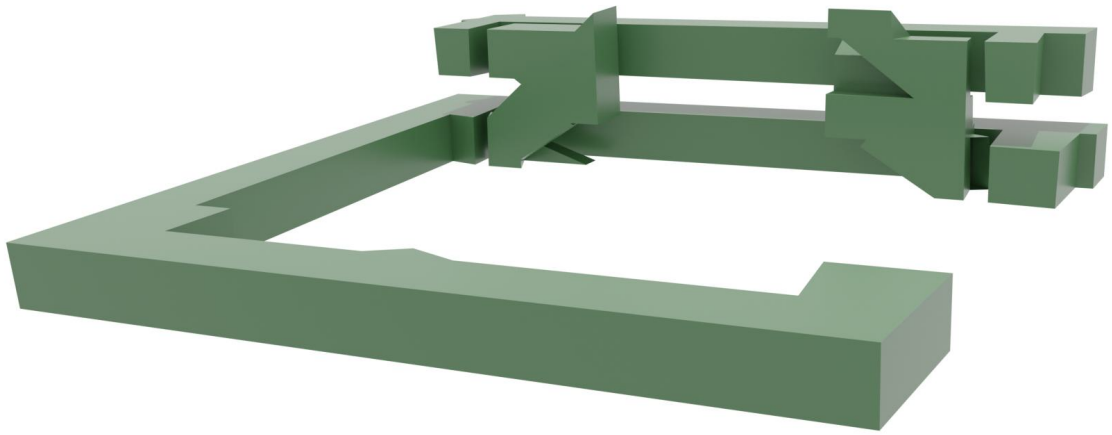
(b) COLMAP generated mesh

Figure 4.9: Mesh Comparison between ours and COLMAP

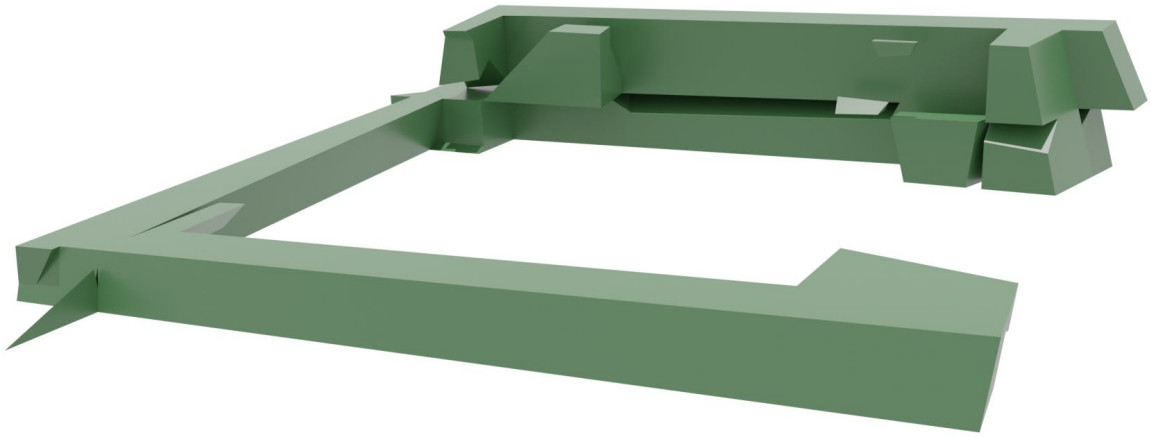
(a) Our close-up observation

(b) COLMAP close-up observation

Figure 4.10: Mesh's close-up comparison between ours and COLMAP



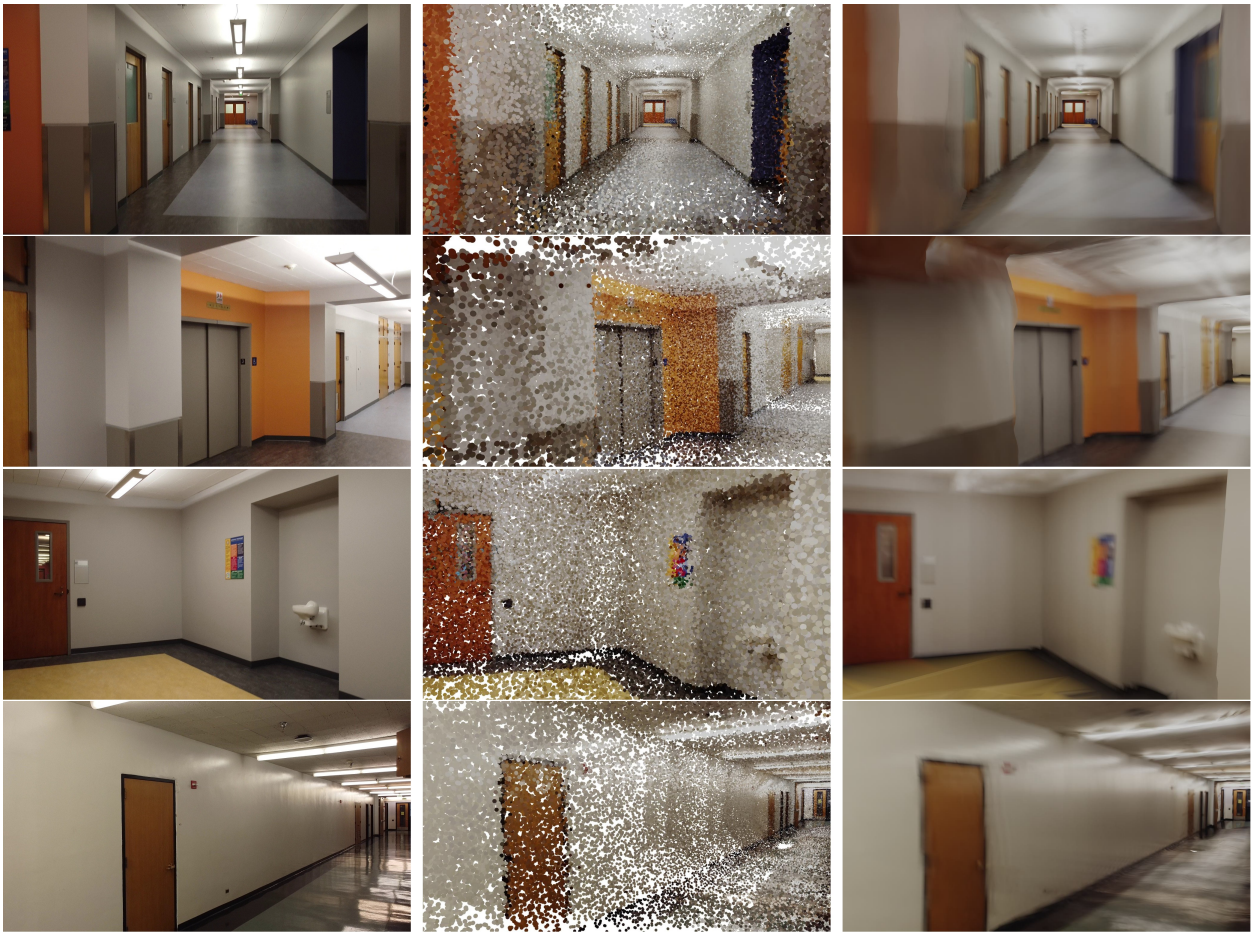
(a) Our Planar fitting model



(b) COLMAP Planar fitting model

Figure 4.11: Planar fitting model comparison between ours and COLMAP

is dense and accurate even though the selected indoor scene contains a large number of low-textured white walls, while the point cloud from COLMAP contains a large number of holes and outliers. In Fig.4.9, our mesh accurately models the surface, while the one from COLMAP is erroneous due to point cloud noises. The close-up observation of our generated mesh and COLMAP's are shown in Fig.4.10. As shown in Fig.4.11, our system successfully generates a simple polygonal model by filtering out non-planar noise.



GT Image Point cloud Textured mesh

Figure 4.12: Close-up observation of our reconstructions. A complete walk-through is provided in this video: https://haoda-li.github.io/scale_indoor.mp4

Fig.4.12 contrasts our reconstructed geometry (generated dense point cloud and textured mesh) against the RGB pictures at approximately the same viewpoint. Our reconstruction accurately predicts the surface even for low-texture regions. However, the surface details in our reconstruction are somewhat smooth, as illustrated in Fig. 4.13, especially in corners

(a) Sharp geometry shown in a captured image

(b) Over-smooth geometry in our reconstruction result

Figure 4.13: Comparison between the raw captured image (a) and our construction result (b) with the same camera pose.

associated with short wall segments. Block 1 in red shows that the bulging pillar is smoothed and can hardly be recognized. Additionally, block 2 in purple shows that the edges of the hanging lights are missing as well, merging into the ceiling plane. We speculate it to be caused by the Manhattan-world assumption being too strong to recover some details.

Run time Analysis

Table 4.1 indicates the run time for our proposed pipeline, for each block, in addition to all 11 blocks. As seen, the most computationally expensive part of our pipeline is the NeRF surface reconstruction, followed by the COLMAP sparse depth reconstruction. It is possible to run each block on a separate GPU in parallel, consequently reducing up the overall run time. As explained later in Chapter 5, future work could involve speeding up the NeRF reconstruction process.

Different procedures in the pipeline	Run time (per block)	Run time (total)
Data capture	N/A	20 minutes
Pix4D pose recovery	20 minutes	3.5 hours
COLMAP sparse point cloud recovery	1.5 hours	16.5 hours
2D segmentation	3 - 5 minutes	45 minutes
Neural surface reconstruction	3 hours	33 hours
PolyFit	N/A	3 - 5 minutes

Table 4.1: Run time for different parts in our pipeline.

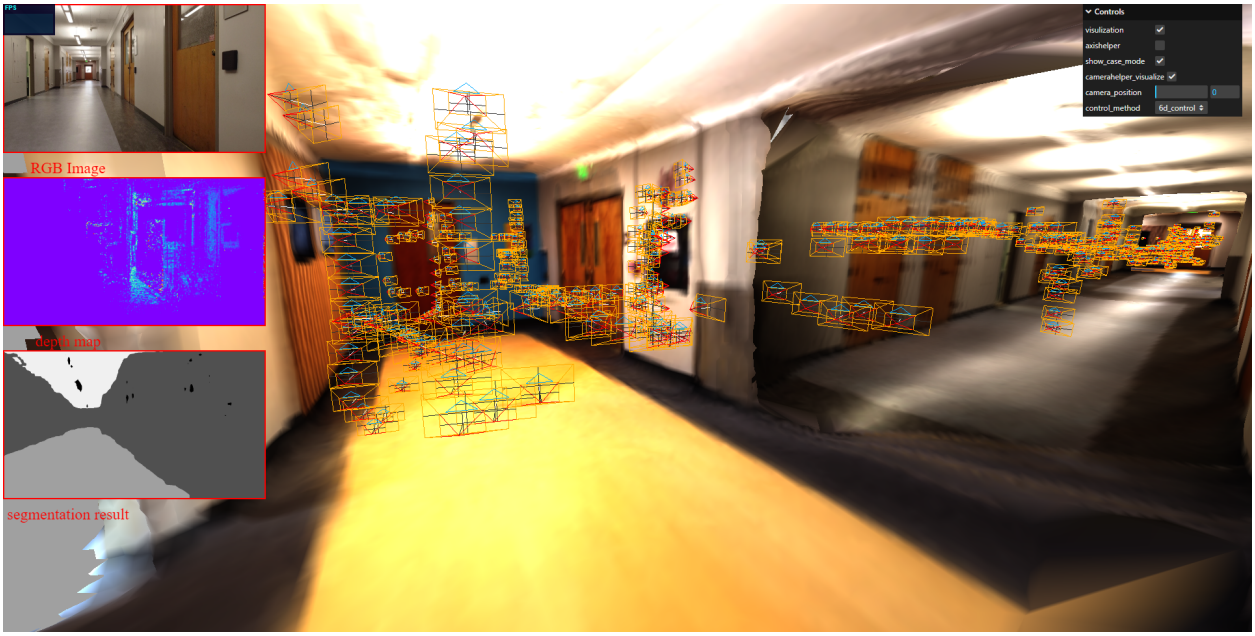
GUI Visualization

To better visualize our result, we have created a graphic user interface for users to experiment with. Our GUI enables users to have a virtual tour in our generated mesh using provided 6D control. Also, users can switch to orbit control to better view our entire mesh. Besides, users can interact with each camera to view its corresponding RGB image, sparse depth map, and segmentation result. Figure 4.14 shows some example screenshots of our GUI.

To experiment with our GUI, you need to connect with Berkeley Internet via VPN first. After connecting, you could tunnel to Corscia machine using the following command:

```
ssh –L 5000:localhost:5000 <username>@corsica.eecs.berkeley.edu
```

This command sets up a tunnel between the local machine (such as your laptop) port 5000 and the Corsica port 5000, as seemingly the EECS firewall blocks direct access to Corsica's customized ports. This also requires the users to have an account on the Corsica machine.



(a) 6-Dof control mode

(b) orbit control mode

Figure 4.14: Our user-friendly GUI. (a)The 6-Dof control mode. Users can enjoy a virtual tour in the generated mesh; (b)Orbit control mode. Users can have an observation of the entire mesh.

Then, on the local machine, open the browser and access `http://localhost:5000/`. The access to the localhost's port 5000 shall be automatically directed to Corsica's port 5000 if the previous command keeps running. Hence, our GUI service can show up.

GUI Usage

The 6-DoF control method works as follows: Using the keyboard **WASD** to move in the x and z directions and **J**, **K** to move in the y direction. **Q** and **E** can be used for turning left and right, and **Z** and **C** can be used for looking up and down. Users may need to left-click their mouse to open the control.

To visualize each camera's original RGB image, our projected sparse depth map, and the segmentation result, users can click on the **show case method** button on the right then they can use their mouse to choose a specific camera they want to observe the result.

The example screenshot of the 6-Dof control mode is shown in Fig.4.14(a). Users can enjoy a virtual tour in the generated mesh. The screenshot of the orbit control mode is shown in Fig.4.14(b). Users can easily have an observation of the entire mesh.

Discussion and Limitations

5.1 Limitations

Currently, our pipeline is limited by processing time and computational complexity. The aforementioned implicit neural representation in Section 3.3 belongs to a machine learning method family, namely the Neural Radiance Fields method (NeRF), whose training can be computationally expensive and time-consuming. A larger-scale and complex geometry further increases the compute time. In our experiments, training a single large-scale indoor scene may take up to several hours to finish. Our pipeline is also limited by the storage of computational devices. The "divide-and-conquer" ideology naturally prefers parallel computation. However, on one single NVIDIA TITAN RTX GPU, multiple training processes cannot be executed simultaneously. Therefore, the "divide-and-conquer" strategy has to be reorganized in a sequential way: each block trains separately, which also extends the run time of our pipeline.

In addition, our pipeline relies on the traditional SFM method. We utilize the SFM to acquire camera poses and a sparse point cloud. We thereafter project the sparse point cloud to form sparse depth maps for further supervision. Hence, despite the capability of fixing the imperfectness of point clouds of the NeRF model, the output quality is still intuitively bottlenecked by the ability of the SFM.

5.2 Future Work

For future work, we suggest optimizing the pipeline to achieve real-time positioning and modeling. Currently, the implicit neural representation model in our pipeline is a canonical Nerf MLP (Multi-layer Perceptron), which is time-consuming and also memory-consuming as mentioned above. There are many other novel NeRF-related research projects wielding better model structures to alleviate the time complexity problem, such as Plenoxels[1] and InstantNGP.[2] It is possible for us to replace our current model with these novel designs to accelerate our pipeline. With a more rapid model reconstruction, we can replace

[1]Sara Fridovich-Keil et al., "Plenoxels: Radiance fields without neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), 5501–5510.

[2]Thomas Müller et al., "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Transactions on Graphics (ToG)* 41, no. 4 (2022): 1–15.

our structure-with-motion module with a simultaneous localization and mapping (SLAM) module and achieve real-time modeling.

In addition, our approach fits seamlessly with robotics and automation. As mentioned in Section 4.1, we manually operate the drones to capture data. Recent works[3] attempt to solve obstacle avoidance and navigation autonomy using reinforcement learning. Such methods require real-time perception of the scene, thereafter automatically plan for the drone trajectories. Our method provides a detailed modeling of the environment and can be an alternative to their current perception module. We believe that indoor mapping can be made fully automatic with drones in the near future. Combining full automation with drones' maneuverability, drone-based indoor mapping can be much faster and more efficient than current solutions.

[3]Zhihan Xue and Tad Gonsalves, "Vision based drone obstacle avoidance by deep reinforcement learning," *AI* 2, no. 3 (2021): 366–380; Victoria J Hodge, Richard Hawkins, and Rob Alexander, "Deep reinforcement learning for drone navigation using sensor data," *Neural Computing and Applications* 33 (2021): 2015–2033.

Conclusion

Albeit the wide utilization of drones in modeling the terrain and building facades, indoor mapping with MAVs to generate 3D geometry remains unsolved and challenging. In this project, we propose a fully automated pipeline for reconstructing large and complex indoor scenes with RGB images collected from drones. First, we leverage traditional structure-from-motion methods to obtain camera poses and reconstruct the initial point cloud. Finally, we devise a divide-and-conquer strategy to utilize neural surface reconstruction under the Manhattan-world assumption. Our method is expected to reduce point clouds' outliers and improve reconstruction quality on low-texture regions.

Our method demonstrate the great scalability, efficiency, and accuracy of drone-based indoor mapping. In the future, our method can adapt recent advancements in drone autonomy and open up opportunities such as virtual experience creations, construction design, and digital twining in metaverse.

Bibliography

Chen, Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." *IEEE transactions on pattern analysis and machine intelligence* 40, no. 4 (2017): 834–848.

Coughlan, James M, and Alan L Yuille. "Manhattan world: Compass direction from a single image by bayesian inference." In *Proceedings of the seventh IEEE international conference on computer vision*, 2:941–947. IEEE, 1999.

Fridovich-Keil, Sara, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. "Plenoxels: Radiance fields without neural networks." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5501–5510. 2022.

Fu, Qiancheng, Qingshan Xu, Yew-Soon Ong, and Wenbing Tao. "Geo-Neus: Geometry-Consistent Neural Implicit Surfaces Learning for Multi-view Reconstruction." *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Furukawa, Yasutaka, and Jean Ponce. "Accurate, Dense, and Robust Multiview Stereopsis." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, no. 8 (2010): 1362–1376. https://doi.org/10.1109/TPAMI.2009.161.

Gropp, Amos, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. "Implicit Geometric Regularization for Learning Shapes." In *Proceedings of Machine Learning and Systems 2020*, 3569–3579. 2020.

Guo, Haoyu, Sida Peng, Haotong Lin, Qianqian Wang, Guofeng Zhang, Hujun Bao, and Xiaowei Zhou. "Neural 3D Scene Reconstruction with the Manhattan-world Assumption." In *CVPR*. 2022.

Hodge, Victoria J, Richard Hawkins, and Rob Alexander. "Deep reinforcement learning for drone navigation using sensor data." *Neural Computing and Applications* 33 (2021): 2015–2033.

Jancosek, Michal, and Tomas Pajdla. "Multi-view reconstruction preserving weakly-supported surfaces." In *CVPR 2011*, 3121–3128. 2011. https://doi.org/10.1109/CVPR.2011.5995693.

Kazhdan, Michael, and Hugues Hoppe. "Screened poisson surface reconstruction." *ACM Transactions on Graphics (ToG)* 32, no. 3 (2013): 1–13.

Lorensen, William E., and Harvey E. Cline. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm." *SIGGRAPH Comput. Graph.* (New York, NY, USA) 21, no. 4 (August 1987): 163–169. ISSN: 0097-8930. https://doi.org/10.1145/37402.37422. https://doi.org/10.1145/37402.37422.

Mildenhall, Ben, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis." In *ECCV*. 2020.

Müller, Thomas, Alex Evans, Christoph Schied, and Alexander Keller. "Instant neural graphics primitives with a multiresolution hash encoding." *ACM Transactions on Graphics (ToG)* 41, no. 4 (2022): 1–15.

Nan, Liangliang, and Peter Wonka. "Polyfit: Polygonal surface reconstruction from point clouds." In *Proceedings of the IEEE International Conference on Computer Vision*, 2353–2361. 2017.

Romanoni, Andrea, and Matteo Matteucci. "Tapa-mvs: Textureless-aware patchmatch multi-view stereo." In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10413–10422. 2019.

Sanchez, Victor, and Avideh Zakhor. "Planar 3D modeling of building interiors from point cloud data." In *2012 19th IEEE International Conference on Image Processing*, 1777–1780. 2012. https://doi.org/10.1109/ICIP.2012.6467225.

Schönberger, Johannes Lutz, and Jan-Michael Frahm. "Structure-from-Motion Revisited." In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

Schönberger, Johannes Lutz, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. "Pixelwise View Selection for Unstructured Multi-View Stereo." In *European Conference on Computer Vision (ECCV)*. 2016.

Stefano, Francesco Di, Stefano Chiappini, Alban Gorreja, Mattia Balestra, and Roberto Pierdicca. "Mobile 3D scan LiDAR: a literature review." *Geomatics, Natural Hazards and Risk* 12, no. 1 (2021): 2387–2429. https://doi.org/10.1080/19475705.2021.1964617.

Wang, Peng, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. "NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction." *NeurIPS*, 2021.

Wang, Yusen, Zongcheng Li, Yu Jiang, Kaixuan Zhou, Tuo Cao, Yanping Fu, and Chunxia Xiao. "NeuralRoom: Geometry-Constrained Neural Implicit Surfaces for Indoor Scene Reconstruction." *ACM Trans. Graph.* (New York, NY, USA) 41, no. 6 (November 2022). ISSN: 0730-0301. https://doi.org/10.1145/3550454.3555514. https://doi.org/10.1145/3550454.3555514.

Xiaodong, Gu, Fan Zhiwen, Zhu Siyu, Dai Zuozhuo, Tan Feitong, and Tan Ping. "Cascade Cost Volume for High-Resolution Multi-View Stereo and Stereo Matching." *Computer Vision and Pattern Recognition (CVPR)*, 2020.

Xue, Zhihan, and Tad Gonsalves. "Vision based drone obstacle avoidance by deep reinforcement learning." *AI* 2, no. 3 (2021): 366–380.

Yao, Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. "MVSNet: Depth Inference for Unstructured Multi-view Stereo." *European Conference on Computer Vision (ECCV)*, 2018.

Yao, Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. "Recurrent MVSNet for High-resolution Multi-view Stereo Depth Inference." *Computer Vision and Pattern Recognition (CVPR)*, 2019.

Yariv, Lior, Jiatao Gu, Yoni Kasten, and Yaron Lipman. "Volume rendering of neural implicit surfaces." *Advances in Neural Information Processing Systems* 34 (2021): 4805–4815.

Yu, Zehao, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. "MonoSDF: Exploring Monocular Geometric Cues for Neural Implicit Surface Reconstruction." *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Zhou, Bolei, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. "Semantic understanding of scenes through the ade20k dataset." *International Journal of Computer Vision* 127, no. 3 (2019): 302–321.

Zhou, Qian-Yi, Jaesik Park, and Vladlen Koltun. "Open3D: A Modern Library for 3D Data Processing." *arXiv:1801.09847*, 2018.