

Indoor Depth Completion via Geometry Processing with RGB Image

Zhenshan Liang^a, Tsz Ho Kwok^a, Yunbo Zhang^{b,1,*}

^aDepartment of Mechanical, Industrial and Aerospace Engineering, Concordia University, 1455 De Maisonneuve Blvd. W., Montreal, H3G1M8, Quebec, Canada

^bDepartment of Industrial and Systems Engineering, Rochester Institute of Technology, 1 Lomb Memorial Dr, Rochester, 14623, New York, USA

^cSchool of Information, Rochester Institute of Technology, 1 Lomb Memorial Dr, Rochester, 14623, New York, USA

Abstract

Depth sensing plays a pivotal role in a wide range of indoor manufacturing applications. However, the market-available low-cost depth sensors contain noises and missing values due to inherent limitations of their principles, and the existing methods still have mismatches and “stretched” regions. Therefore, this paper proposes a discrete geometric processing-based method to solve these problems and achieve more accurate indoor depth completion by explicitly processing depth discontinuity and modeling input uncertainties. This approach involves converting the raw depth image into a quadrilateral mesh surface and completing missing depth information through a mesh deformation framework. The optimization problem considers normal constraints, position constraints, and perspective projection constraints, solving it with element geometry projection and proximity function minimization. Uncertainties in input normal and observed depth information are addressed through constraints updating strategies. Experimental results on the ScanNet dataset, Matterport3D dataset, and five real-world scenarios demonstrate the superiority of the proposed method over existing techniques quantitatively and qualitatively. Specifically, the method achieves quantitative improvement of 13% to 32% (fully automated pipeline) in root mean square error (RMSE) of the depth values and excels in handling discontinuity around object boundaries by explicitly considering depth discontinuity, providing a clear boundary in the reconstruction results, while competing methods exhibit errors and “stretched” distortions. The robustness of the method is further affirmed by energy reductions in both normal and position aspects across iterations.

Keywords: Indoor depth completion, Multi-sensor fusion, Geometry processing

1. Introduction

In the ongoing Fourth Industrial Revolution (Industry 4.0), depth sensing and 3D reconstruction play pivotal roles in a wide range of indoor engineering applications, including industrial digital twins [1, 2], human-machine interaction [3], robot manipulation [4, 5], inspection and quality control [6, 7], redesign [8, 9] and remanufacturing [10, 11], and flexible manufacturing process control [12]. The proliferation of readily available depth sensors has fueled the rapid growth of these applications, making their success contingent upon the accuracy and reliability of depth sensors. Traditional depth sensors can be broadly categorized as passive and active. The passive approach relies on rich and distinctive features in paired color (RGB) images to estimate depth for corresponding points, but it tends to perform inadequately in regions with low texture or repetitive patterns. On the other hand, the active approach, encompassing techniques like time-of-flight (ToF) and structured light (SL), overcomes some of the limitations of stereovision by projecting structured patterns to discern the geometry of the scene. However, active methods may encounter challenges such as non-Lambertian surfaces and inherent constraints [13]. Consequently, both sensor types often yield noisy and incomplete

depth maps, which are insufficient for accurate indoor 3D information for localization and reconstruction. Therefore, the completion and refinement of raw depth maps are essential to enhancing their utility. In essence, indoor depth completion is the process of generating a comprehensive and more precise depth map from an initially incomplete and noisy depth map. Its primary objective is to fill the void left by missing depth values and elevate the overall quality of the depth data. Indoor depth completion methods encompass a range of techniques, including interpolation [14], machine learning [15], graph-based optimization [16], and sensor data fusion [17], to estimate and augment the missing depth values. Sensor fusion, in particular, involves the integration of data from multiple sensors and leverages the strengths of each type of data to enhance the depth map’s quality. This technique proves especially valuable when tackling intricate and challenging scenes. For instance, depth (D) cameras capturing depth information could easily encounter difficulties in achieving high precision or missing information when dealing with highly reflective surfaces. Meanwhile, RGB cameras excel at capturing high-resolution details, patterns, and textures on object surfaces [18, 19]. There are research works that utilize the RGB information to complete the depth through implicitly extracted feature representation by designing specific modules and networks [15, 20, 21, 22], or explicitly extracted geometric information [23, 24, 25], e.g., detected edges and surface normal. Diverse data from multiple sensors provide comple-

*Corresponding author.

Email address: ywzeie@rit.edu (Yunbo Zhang)

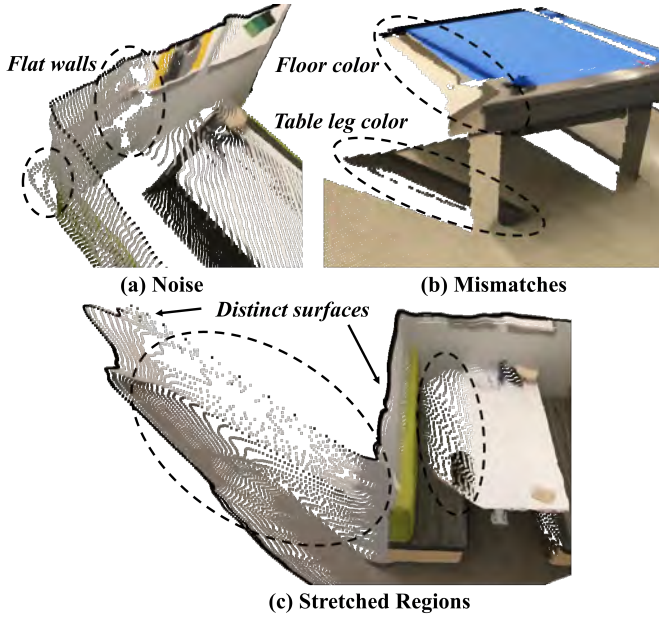


Fig. 1. Challenges in depth completion: noises, mismatches, and “stretched” regions. The colors from the RGB image are assigned to the corresponding points in the depth image.

mentary insights, enabling sensor fusion techniques to create more resilient and accurate depth maps.

Despite the progress made in sensor fusion approaches, several unresolved issues in the input data lead to significant errors in their results, including 1) *noise and mismatch*, and 2) *missing depth-discontinuity information*. Noise and mismatch among data from different sensors are common, even in outputs of dense reconstructions [26], which are often referred to as “ground truth.” For example, as shown in Fig. 1 (a) and (b), walls that should be flat exhibit gaps and redundancies, while the point cloud computed from the depth map using camera intrinsics does not semantically align well with the corresponding RGB image. With such problematic input data, methods such as calibration procedures [27, 28], the BundleFusion system [29], and VoxelHashing [30] fail to produce results that accurately reflect real scenes. Moreover, the lack of depth-discontinuity information in different regions of the input data can cause significant distortions in the resulting depth, leading to “stretched” regions (see Fig. 1 (c)). Such “stretched” regions appear in the results of several existing works [24, 31, 32, 21] due to inadequate handling of depth discontinuities and/or the absence of effective processing techniques. Therefore, a new approach is expected to 1) handle input data with noise and mismatched information and 2) utilize depth-discontinuity information to establish correct spatial relationships between objects and improve the accuracy of the depth.

To this end, we propose a geometric processing-based method derived from the existing geometric Surface-from-Gradients (SfG) method [33]. This method models an RGB-D image as a quadrilateral mesh surface and formulates the depth reconstruction problem as constrained geometric optimization. Compared with implicit functions [34, 35], quadrilateral discretizations can easily deal with depth discontinuity after ex-

PLICIT extraction from RGB images, and the completed depth will be obtained conveniently from the deformed mesh surface instead of re-projection from implicit representations. If two triangular facets are used to represent one pixel, there are two possible cases: 1) top-left and bottom-right, and 2) top-right and bottom-left, which creates ambiguity in choosing which case to use. Then, the mesh surface is separated according to the depth discontinuity extracted from the RGB image to avoid stretched regions in the completion results, while the observed depth trimming along the detected discontinuity avoids mismatches during the completion process. This work is motivated by applications in Industry 4.0, which are typically linked to indoor manufacturing scenarios. Such scenarios have a limited depth range and consist mainly of man-made objects. Therefore, our method is designed to complete depth within a room-sized range and to utilize a normal estimation method that is robust for indoor environments. The uncertainties of raw depth from the depth sensor and estimated normal from the RGB image are converted into geometric updatable ranges to reduce the noises. The workflow and key aspects of our approach are illustrated in 3, and the major contributions are highlighted as follows:

- A quadrilateral mesh representation-based framework for fusing multi-sensory data (i.e., RGB and depth) and completing missing depth concurrently, with explicit handling of depth discontinuities.
- A constrained geometric optimization method solved in an iterative least-squares scheme, consisting of pinhole model-based element geometry projection and proximity function minimization, with both estimated normal from the RGB image and input raw depth as constraints.
- Position and normal updating strategies with uncertainty awareness, based on their confidence maps, for denoising and improved depth accuracy.

The experimental results indicate that the proposed method outperforms other state-of-the-art techniques [24, 21], which also leverage RGB data to improve the depth map. Our results exhibit better qualitative performance and a higher degree of consistency with the RGB images. The remainder of the paper is structured as follows: Section 2 provides an overview of related works in the field of depth completion. Section 3 and Section 4 delve into the technical intricacies of the proposed method. Section 5 presents our experimental results along with a detailed analysis. Finally, in Section 7, we conclude by summarizing the proposed method and discussing potential directions for future work.

2. Related Works

This paper pertains to the field of depth completion, and we will examine related techniques within this domain.

2.1. RGB-D Data Fusion

Various sensors offer complementary depth information, enabling the creation of a more comprehensive and precise depth map through data integration. RGB image is a prevalent data source in depth completion. Certain methods have introduced specific modules [20] and network architectures [22, 36] designed to encode RGB images into multi-level features, and their adaptive fusion methods utilize U-Net structure to guide depth completion efficiently during the decoding process. Specifically, surface normals can be explicitly estimated from RGB images, providing local depth change as geometric information for depth completion [23]. Huang et al. [24] have enhanced performance by incorporating a self-attention mechanism and boundary consistency. The self-attention mechanism highlights critical features, while boundary consistency loss forces the attention network to preserve occlusion information for sharp depth edges. Ren et al. [25] extends this idea to depth structure completion, introducing surface normal constraints on flat regions and reflecting depth structures through Gaussian weights. A significant challenge lies in sensor alignment and calibration to ensure accurate fusion. The decoder modulation branch [21] utilizes a missing depth map mask to exploit spatially dependent features, reducing domain shift between RGB images and depth maps through a technique known as spatially adaptive denormalization (SPADE). In the fusion process, some approaches incorporate depth confidence for guidance during both training and inference [37, 38]. Wang et al. [15] introduces a constraint network branch that assigns high confidence to accurate raw depth data. The fused confidence map serves as a complementary guide for depth completion. Liu et al. [39] employs a confidence mask to reflect the reliability of depth obtained from Multiple-View Stereo (MVS). They mitigate outliers by reducing their influence through weighted loss functions. Nevertheless, data from different sensors may introduce varying errors [40], resulting in distinct uncertainties and mismatching issues that necessitate further investigation.

2.2. Depth Spatial Propagation

By considering the interrelationships between neighboring pixels, one can disseminate depth information across an image, thus completing the areas lacking depth data. Liu et al. [41] introduces the spatial propagation network (SPN), which aims to learn an affinity matrix for semantic similarity. Cheng et al. [42] extends this approach to the convolutional spatial propagation network (CSPN), which propagates information within a local area in all directions without sacrificing theoretical support. Subsequently, they further refine the method with CSPN++ [43]. The context-aware CSPN significantly improves depth completion accuracy by distinguishing between simple planes (e.g., ground and walls) and complex surface boundaries. The resource-aware CSPN optimizes kernel size and iterations for each pixel to reduce computational resources while maintaining comparable accuracy. To efficiently incorporate relevant neighbors and suppress unrelated ones, Park et al. [31] introduces learnable parameters that dynamically select K neighbors for each pixel based on the RGB image and raw

depth map. Liu et al. [16] pioneer a more dynamic, graph-based spatial propagation in 3D space. They utilize changeable graphs with estimated affinity patches to propagate information effectively, employing a self-attention mechanism. DySPN [44] further refines the spatial propagation model by generating different affinity matrices during iterations to account for long-range dependencies. This approach also addresses over-smoothing issues by considering the affinity matrix between the current refined result and the previous one. However, it's worth noting that spatial propagation may not excel at handling discontinuities and can sometimes result in "stretched" regions at the boundary.

2.3. Depth Residual Learning

Assuming the accuracy of the observed depth, depth completion can be seen as a regression problem, employing residual reconstruction mapping to estimate invalid depth [45]. The primary characteristic of depth residual learning involves adding the interpolated dense depth map to the output of the designed network through a global skip connection to calculate the completed depth map. Liao et al. [46] expands linear interpolation results from filtered laser scan readings along the gravity direction to create a "reference depth" map. Conversely, Chen et al. [47] and Hegde et al. [48] utilize nearest neighbor interpolation for acquiring a dense depth map. Subsequently, the resultant interpolated map is combined with RGB images as the input for the depth residual network, encouraging the network to predict residual depth. However, the hand-crafted interpolation kernels employed in these methods are data-independent and heuristic. To address these limitations, Liu et al. [14] introduces differentiable kernel regression to fully exploit the image through end-to-end learning. The interpolated dense depth map ensures a lower bound on the final results' quality. Nonetheless, most of these approaches considered rendered depth from mesh or inpainted depth as ground truth, such as in ScanNet [26] and NYU Depth v2 [49]. Unfortunately, even the ground truth depth map includes noisy areas, missing depth regions, and mismatched parts compared to RGB images [50]. Hence, our objective is to create a method that is not reliant on the training data.

3. A Geometric-Processing-Based Method

The computation of missing depth is formulated as a constrained mesh deformation [33] where the estimated normal vector at each facet and the observed depth information are constraints. The last constraint comes from the pinhole camera model, which constrains the deformation of each facet to follow the perspective projection in the camera coordinate system. The details about the pinhole camera model and the quadrilateral mesh representation of the depth image are as follows:

Pinhole Camera Model. In the classic pinhole model [51], a 3D point \mathbf{p} with coordinates (x, y, z) in the camera coordinate system is projected onto the physical imaging plane through a perspective projection, resulting in a point \mathbf{p}' with coordinates

$(\frac{f_x}{z}, \frac{f_y}{z}, f)^T$, where f is the focal length between the optical center \mathbf{o} and the physical imaging plane. The coordinates (u, v) of \mathbf{p}' on the pixel plane can be further obtained through the scaling α along u axis and β along v axis, and the translation c_x along u and c_y along v , which are represented as: $u = \alpha x' + c_x$, and $v = \beta y' + c_y$. The whole process can be rewritten into a matrix multiplication formation, as follows:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{z} \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{1}{z} \mathbf{K} \mathbf{p}, \quad (1)$$

where $f_x = \alpha f$, $f_y = \beta f$, and \mathbf{K} is also defined as the camera intrinsic matrix.

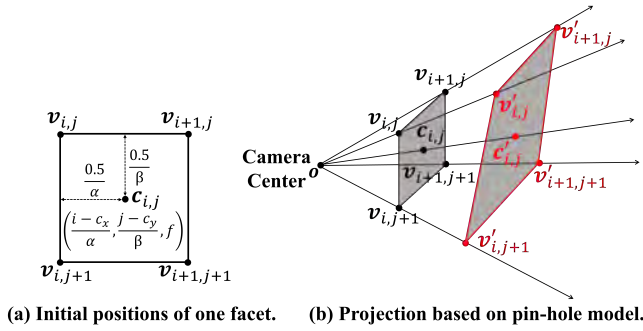


Fig. 2. (a) The initial positions of four vertices of one quadrilateral facet form a square. (b) The initial positions are projected in the camera coordinate system based on the pinhole model.

Quadrilateral Mesh Representation of Depth Image. For each pixel of the depth image with the pixel coordinates (i, j) , a quadrilateral facet $f_{i,j}$ is constructed with four vertices $\mathbf{v}_{i,j}$, $\mathbf{v}_{i+1,j}$, $\mathbf{v}_{i+1,j+1}$ and $\mathbf{v}_{i,j+1}$, and the pixel coordinates (i, j) become the position of the center $\mathbf{c}_{i,j}$ of the quadrilateral facet $f_{i,j}$ when it is tiled on the pixel plane. Based on the pinhole camera model, each vertex $\mathbf{v}_{i,j}$ is then initially positioned with depth f in the camera coordinate system with the coordinates

$$\left(\frac{(i-0.5)-c_x}{\alpha}, \frac{(j-0.5)-c_y}{\beta}, f \right).$$

As shown in the Fig. 2 (a), the initial positions of four vertices make the facet $f_{i,j}$ a square, and the vertices and center point are projected to the required positions marked in red color according to the local shaping in Section 4.2. The original depth image in the image space is thus converted to a mesh surface $M = (V, F, E)$ in Euclidean space, where V , F , and E represent the sets of vertices, facets, and edges.

3.1. Formulation for Constrained Optimization

To complete the missing depth, the mesh surface M is deformed by moving each vertex $\mathbf{v}_{i,j}$ and therefore reshaping each facet $f_{i,j}$ to satisfy the normal $\mathbf{n}_{i,j}$ constraint and position d_{ob}

constraint. Generally, normal constraints rotate the orientation of the facet, while position constraints determine the position of the vertices for each facet. The facets with raw observed depth can be considered as spatial anchors to constrain the deformation within a certain scale range. The last constraint comes from the pinhole camera model, which requires that the vertices of the facet do not move freely in space but only move along the ray formed by the camera center \mathbf{o} and the vertex's initial positions (see Fig. 2 (b)). Therefore, a straightforward formulation of this problem is to minimize the shape variation of M and enforce the normal, position, and pinhole projection constraints as follows:

$$\begin{aligned} \min_{\{d_{i,j}\}} E(M) \quad \text{s.t.} \quad & \mathbf{n}(f_{i,j}) = \mathbf{n}_{i,j}, \quad \forall f \in F \\ & d(f) = d_{ob}, \quad \forall f \in F_{ob} \\ & \mathbf{ov} \times \mathbf{ov}' = 0, \quad \forall \mathbf{v} \in V \end{aligned} \quad (2)$$

where $E(M)$ is a functional measuring the shape variation (and/or smoothness) of M , $\mathbf{n}(\cdots)$ returns the normal vector of a facet, $d(\cdots)$ returns the depth value of a facet, and F_{ob} is the set of facets with raw observed depth values. The third constraint enforces that the movement of each vertex of M during the deformation follows the pinhole camera model's perspective projection by ensuring the vector \mathbf{ov} (\mathbf{o} is the camera center and \mathbf{v} is the initial position of the vertex) and the vector \mathbf{ov}' (\mathbf{v}' is the deformed vertex) are co-linear. Then, we follow previous work [33, 52] to solve the optimization problem in a local-global iterative manner.

Since the solution to this optimization problem highly relies on the quality of normal and position constraints, an essential question to ask is: **Should we update these constraints during iterations?** Because there are potential uncertainties in the initial normal and positional inputs due to the limitations of their methods, the noise in the training dataset, and the instability of sensor performance. These uncertainties could result in increasingly larger errors in the depth of information throughout the iterations. In order to approach the real values of $\mathbf{n}_{i,j}$ and d_{ob} , a simple and effective strategy is proposed to update normal constraints and depth constraints based on the uncertainty modelings. More details can be found in Section 4.

3.2. Iterative Least-Squares Solution

Geometric constraints update, element geometry projection, and proximity function minimization are iteratively employed to compute the missing depth information. Before starting the iteration, an initial normal field containing normal vectors associated with each facet $f_{i,j}$ is estimated from the RGB image. The mesh edges, which are at the boundary of two regions with two discontinuous depths, are detected from the RGB image to explicitly split the mesh surface M to obtain clear boundaries without stretched regions. Meanwhile, the observed depths that are inconsistent with the depths near detected depth-discontinuity edges are deleted for solving the mismatches problem, which is called trimming. The last part of initialization is Laplacian diffusion for quickly generating initial depth values at the facets with missing depth values. During the element geometry projection process in each iteration,

¹For convenience, it is assumed that the object and image are on the same side and in front of the camera.

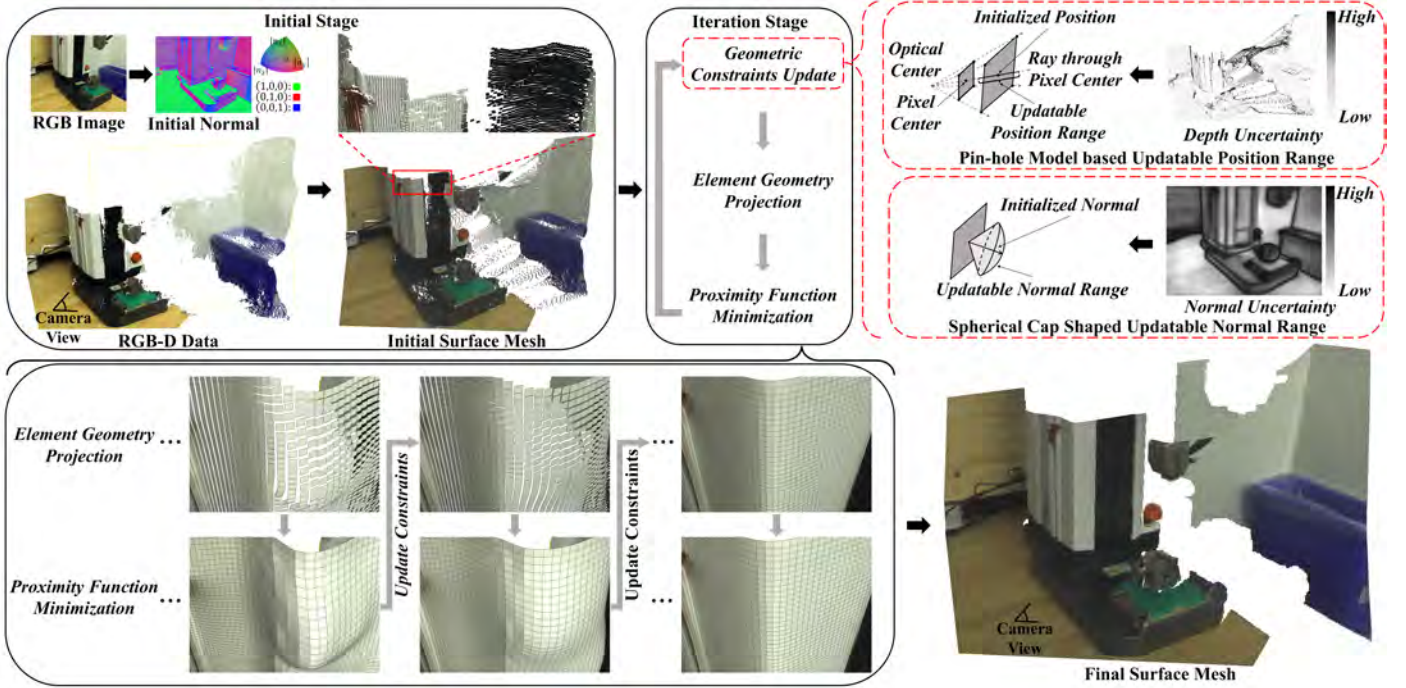


Fig. 3. Overview of the geometric-processing-based framework for indoor depth completion. The initial stage converts raw RGB-D data with estimated normal field into an initial surface mesh. Then, the iteration stage includes three steps: 1) The geometric constraints are updated based on the current surface mesh and updatable position/normal range; 2) Each element is projected onto the desired shape under the current geometric constraints; and 3) The proximity function is minimized by solving least-squares. Finally, the high-quality surface mesh is reconstructed, and the complete depth is easily extracted from the position constraints of the last iteration.

each facet is projected onto the desired geometry according to the current target normal, current target position, and pinhole perspective projection. Afterward, the proximity function minimization will be performed to obtain the new shape of M by computing the new position of each vertex. To address the potential uncertainties in the raw depth and input normal information, an effective strategy based on the trust region is proposed to update the target normal and target position during iteration. When the termination conditions are satisfied, the iteration stops and outputs the final completed depth. Please refer to Fig. 3 for an overview.

4. Formulation and Implementation Details

4.1. Initialization

Normal Estimation from RGB Image: The surface normal at each facet is required as the input of our iterative depth completion method. There are a variety of deep learning-empowered surface normal estimation methods [53, 54, 55, 56] with a single RGB image as input, some of which leverage the outstanding performance of CNN [57, 58]. We choose a work [59] to estimate the normal as shown in the initial stage of Fig. 3 since it achieved state-of-the-art performance on the ScanNet dataset [26] as well as providing the estimation of aleatoric uncertainty in the dataset. It firstly utilized a convolutional encoder-decoder [60] to extract a 1/8 resolution feature map, and the coarse prediction was made from this low-resolution

feature map. During training, the loss function was based on the proposed angular von Mises-Fisher distribution that modeled the normal probability density function, and it was applied to all pixels for the coarse prediction. Then, three pixel-wise refinement modules with the same architecture in sequence up-sampled the feature map by a factor of 2 and applied a pixel-wise MLP to acquire a higher-resolution output. In this work, the pre-trained model on ScanNet was utilized to estimate the normal, and the derived cumulative probability of angular error was utilized in the proposed update strategy. Also, the estimated normal is used for the other two works for comparison in experiments. The calculated concentration parameter κ [59] is taken as uncertainty for each facet. Based on each κ , we propose an update strategy for the normal, which will be detailed in Section 4.4.

Depth-discontinuity Edges Detection and Trimming: In a depth image, there are regions with quite distinct depths, i.e., adjacent objects at different distances, which indicates the discontinuity of the depth image. Existing methods considered the depth image as a whole and therefore cannot handle the discontinuity well (see “stretched” regions shown in Fig. 1). The proposed geometric processing-based method handles the discontinuity through a simple topological process: the boundaries of discontinuous regions, which are edges of the mesh surface M , are detected and used to trim the mesh surface as shown in Fig. 4. Specifically, these edges can be visualized by two adjacent pixels but with different predefined colors on the RGB

image, and at most three different predefined colors (purple, yellow, and green) are needed to visualize all possible discontinuity edges [61]. There are cases where the discontinuities detected from the depth values and those from the RGB image are inconsistent (see Fig. 4 left): some facets identified as part of the table based on RGB information have depth values similar to those of the floor. Since the RGB information is considered more accurate, these facets are trimmed from the floor and initialized with a depth value of 0 (see Fig. 4 right). Setting the depth value to 0 here just means discarding the raw observed depths, and the trimmed depths and the missing depths are actually regarded as unknown variables and initialized by neighboring reliable depths using the subsequent Laplacian diffusion. In the experiments, evaluations are conducted using manually annotated and automatically detected depth-discontinuity edges, respectively. In the following computation, the neighboring facets with trimmed edges as boundaries will be decoupled.

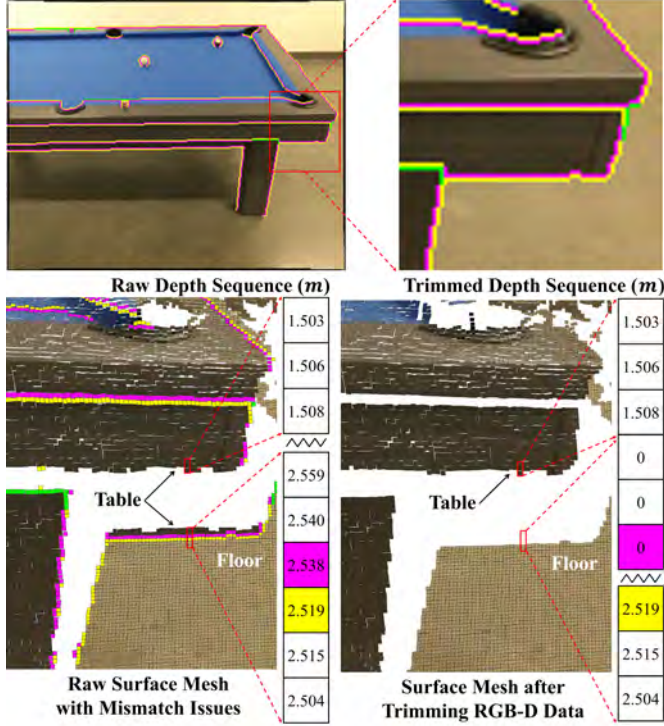


Fig. 4. The boundaries of discontinuous regions are detected and visualized by three predefined colors on the RGB image (see top). However, there are cases where the discontinuities detected from the depth values and those from the RGB image are inconsistent (see bottom left): some facets identified as part of the table based on RGB information have depth values similar to those of the floor. Since the RGB information is considered more accurate, these facets are trimmed from the floor and initialized with a depth value of 0 for the subsequent iterative optimization (see bottom right).

Initial Depth Generation Through Laplacian Diffusion: To start iterative depth completion, the initial depth values of the depth-missing regions are required. The Laplacian mesh diffusion method is simply applied, which has been used in mesh

hole filling [62], to generate initial depth values of depth-missing regions. For the mesh surface with trimming applied, a global Laplacian diffusion is solved by a linear equation system assembled according to the connectivity of the mesh surface and the known depth values as the boundary condition. In spite of more sophisticated methods that could be applied and tested in the future, in our implementation, we find that first-order Laplacian diffusion leads to good results.

4.2. Element Geometry Projection

For each facet $f_{i,j}$, the vertices of a quadrangular facet $f_{i,j}$ are projected onto the plane with the normal $\mathbf{n}_{i,j}$, where the plane is supposed to be passing through the *current* projected center, $\mathbf{c}'_{i,j}$, of $f_{i,j}$ as shown in Fig. 2(b). The projection of a vertex $\mathbf{v}_{k,l}$ ($k \in \{i, i+1\}$ and $l \in \{j, j+1\}$) along the vector (i.e., $\mathbf{ov}_{k,l}$ formed by the camera center \mathbf{o} and the initial position of the vertex $\mathbf{v}_{k,l}$ on the pixel plane), which is obtained as

$$\mathbf{p}_{i,j}(\mathbf{v}_{k,l}) = \mathbf{c}'_{i,j} + \left(\frac{(\mathbf{c}'_{i,j} - \mathbf{o}) \cdot \mathbf{n}_{i,j}}{\mathbf{ov}_{k,l} \cdot \mathbf{n}_{i,j}} \right) \mathbf{ov}_{k,l} \quad (3)$$

A vertex $\mathbf{v}_{k,l}$ of M that is not at image boundaries, is surrounded by four facets by default, which results in four projected positions computed by Eq.(3). However, the projection of $\mathbf{v}_{k,l}$ should be determined by the connections among its adjacent facets according to the detected depth-discontinuity edges. Therefore, a more sophisticated method is developed and presented as follows.

4.3. Proximity Function Minimization

Following the previous work [33], the vertices of $f_{i,j}$ are desired to move to their projections obtained in Section 4.2, represented by a column vector as:

$$\mathbf{p}(f_{i,j}) = \begin{bmatrix} \mathbf{p}_{i,j}(\mathbf{v}_{i,j}) \\ \mathbf{p}_{i,j}(\mathbf{v}_{i+1,j}) \\ \mathbf{p}_{i,j}(\mathbf{v}_{i+1,j+1}) \\ \mathbf{p}_{i,j}(\mathbf{v}_{i,j+1}) \end{bmatrix}. \quad (4)$$

The unknown depth values of $f_{i,j}$'s four vertices after fusing the contribution of adjacent facets can be represented as

$$\mathbf{z}(f_{i,j}) = \begin{bmatrix} d_{i,j} \\ d_{i+1,j} \\ d_{i+1,j+1} \\ d_{i,j+1} \end{bmatrix}. \quad (5)$$

The deformation of M is desirable to move the position of each vertex to its projected position of each facet, which can be formulated as minimizing the following proximity function:

$$\Phi_n(\{d_{k,l}\}) = \sum_{f_{i,j}} \|\mathbf{z}(f_{i,j}) - \mathbf{Z}(\mathbf{p}(f_{i,j}))\|^2, \quad (6)$$

where the function $\mathbf{Z}(\cdot)$ extracts the current desired depth values, i.e., z coordinates, of projection points because x and y are linearly determined by z in the pin-hole model. This proximity function equally measures the sum of squared distances of

the vertices to the corresponding geometric projections. Similarly, to allow translational motion as a degree of freedom to accelerate optimization, we subtract the mean value of the four vertices for both $\mathbf{z}(f_{i,j})$ and $\mathbf{Z}(\mathbf{p}(f_{i,j}))$ [52], and therefore, the proximity function becomes

$$\Phi_n(\{d_{k,l}\}) = \sum_{f_{i,j}} \|\mathbf{Nz}(f_{i,j}) - \mathbf{NZ}(\mathbf{p}(f_{i,j}))\|^2, \quad (7)$$

with $\mathbf{N} = \mathbf{I}_{4 \times 4} - \frac{1}{4}\mathbf{1}$, and $\mathbf{1}$ is a 4×4 matrix with all elements equal to 1. Minimizing Eq.7 solely could lead to ambiguity since there are multiple solutions for the facets fulfilling normal constraints under the perspective projection. Therefore, the observed depth information from the raw depth image is utilized as positional constraints to eliminate ambiguity. These positional constraints can be formulated as the following proximity problem as well:

$$\Phi_o(\{d_{k,l}\}) = \sum_{f_{i,j} \in F_o} \|(\mathbf{z}(f_{i,j})) - \mathbf{Z}(\mathbf{p}(f_{i,j}))\|^2, \quad (8)$$

where F_o is the set of facets with observed depth values from the raw depth image. The overall proximity function Φ can be obtained simply through the summation of Φ_n and Φ_o (i.e., $\Phi = \Phi_n + \Phi_o$).

Without loss of generality, the mesh surface M is assumed to have \mathbb{N} vertices and \mathbb{M} quadrangular facets, while \mathbb{M}_o of the quadrangular facets have observed depth values. Minimizing Φ is straightforward by solving a linear system of \mathbb{N} equations: $\partial\Phi/\partial d_{k,l} = 0$. Since both Φ_n and Φ_o are in quadratic forms, a simpler formulation with a more efficient numerical scheme is developed below.

$$\begin{aligned} \Phi(\{d_{k,l}\}) &= \Phi_n(\{d_{k,l}\}) + \Phi_o(\{d_{k,l}\}) \\ &= \|\mathbf{A}_n \mathbf{x} - \mathbf{b}_n\|^2 + \|\mathbf{A}_o \mathbf{x} - \mathbf{b}_o\|^2, \end{aligned} \quad (9)$$

where \mathbf{A}_n is a $4\mathbb{M} \times \mathbb{N}$ matrix derived from $\mathbf{Nz}(f_{i,j})$, \mathbf{b}_n is a vector with $4\mathbb{M}$ components derived from $\mathbf{NZ}(\mathbf{p}(f_{i,j}))$, \mathbf{A}_o is a $4\mathbb{M}_o \times \mathbb{N}$ matrix from $\mathbf{z}(f_{i,j})$ of those facets in the set F_o , \mathbf{b}_o is a vector with $4\mathbb{M}_o$ depth values obtained from the element projection $\mathbf{Z}(\mathbf{p}(f_{i,j}))$, the vector \mathbf{x} contains all the unknown depth values at the vertices of M . Finally, $\Phi(\{d_{k,l}\})$ can be rewritten as $\Phi(\{d_{k,l}\}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$, with $\mathbf{A} = \begin{bmatrix} \mathbf{A}_n \\ \mathbf{A}_o \end{bmatrix}$, and $\mathbf{b} = \begin{bmatrix} \mathbf{b}_n \\ \mathbf{b}_o \end{bmatrix}$. This is a standard least-squares formulation, which can be solved by finding out the solution of $\mathbf{A}^T \mathbf{A} \mathbf{x} = \mathbf{A}^T \mathbf{b}$, where \mathbf{A} is a $4(\mathbb{M} + \mathbb{M}_o) \times \mathbb{N}$ matrix, and \mathbf{b} is a vector with $4(\mathbb{M} + \mathbb{M}_o)$ values.

4.4. Uncertain-aware Updating Strategies

In each iteration, the solved new depth values lead to new positions of M 's vertices. For a facet, its four vertices after deformation are generally not on the same plane. Therefore, a plane is fitted based on the computed new positions of the four vertices using a Singular Value Decomposition (SVD) to solve a least-squares problem [63]. The depth value of each facet $f_{i,j}$ is defined as the intersection of the ray $\mathbf{oc}_{i,j}$ (formed by the camera center \mathbf{o} and the center point $\mathbf{c}_{i,j}$ of $f_{i,j}$ on the image plane) and the newly fitted plane. Given fitted planes, the

normal vectors associated with each facet can be easily obtained as well. As mentioned in Section 3.1, it is crucial to handle the uncertainties and noise existing in estimated normal and raw observed depth. Therefore, the above depth value and normal vectors are updated following the below strategies.

Normal Updating Strategy: The method we adopt to generate the initial normal input [59] provides the modeling of the noise distribution existing in the input. Specifically, an angular von Mises-Fisher distribution to describe the normal probability density function for each pixel with image coordinates i, j :

$$p_{i,j}(\mathbf{n}_{i,j}|\boldsymbol{\mu}_{i,j}, \kappa_{i,j}) = \frac{(\kappa_{i,j}^2 + 1)}{2\pi} \cdot \frac{\exp(-\kappa_{i,j} \cos^{-1}(\boldsymbol{\mu}_{i,j}^T \mathbf{n}_{i,j}))}{1 + \exp(-\kappa_{i,j}\pi)}, \quad (10)$$

where $\boldsymbol{\mu}_{i,j}$ is the estimated mean normal, $\kappa_{i,j}$ is the concentration parameter, and $\mathbf{n}_{i,j}$ is the ground truth normal. The larger $\kappa_{i,j}$ is, the more concentrated the probability density function of $\mathbf{n}_{i,j}$ is around $\boldsymbol{\mu}_{i,j}$. Accordingly, the cumulative probability of angular error α between $\mathbf{n}_{i,j}$ and $\boldsymbol{\mu}_{i,j}$ is derived as:

$$\begin{aligned} P[\alpha \leq \theta_{i,j}] &= \int_0^{2\pi} \int_0^{\theta_{i,j}} p_{i,j}(\phi) \sin \phi d\phi d\omega \\ &= \frac{1 - \exp(-\kappa_{i,j}\theta_{i,j})(\cos \theta_{i,j} + \kappa_{i,j} \sin \theta_{i,j})}{1 + \exp(-\kappa_{i,j}\pi)}, \end{aligned} \quad (11)$$

where $\theta_{i,j}$ is angular uncertainty between estimated normal $\boldsymbol{\mu}_{i,j}$ and ground truth $\mathbf{n}_{i,j}$. With an angle $\theta_{i,j}$, there is a probability of \hat{P} that the angle between $\boldsymbol{\mu}_{i,j}$ and $\mathbf{n}_{i,j}$ falls into the range $[0, \theta_{i,j}]$. In principle, a larger \hat{P} leads to a larger $\theta_{i,j}$, and vice versa. We utilize Eq.(11) to constrain the normal updating. By setting up a \hat{P} , a spherical cap shaped range $[0, \theta_{i,j}]$ is obtained as shown in Fig. 3. In each iteration, we calculate the angle $\theta'_{i,j}$ between the newly calculated normal vector and the initial normal estimated from RGB images and compare it with $\theta_{i,j}$ that is pre-computed after setting up \hat{P} . If the $\theta'_{i,j}$ is within $[0, \theta_{i,j}]$, the newly calculated normal vector will be the new target normal as the constraint in the next iteration. Otherwise, we rotate the initial normal towards the newly generated normal by the angle $\theta'_{i,j}$, and update the target normal for the next iteration with the rotated normal. Note that in principle, \hat{P} for each facet could be different, but we decided to choose the same value for all facets. In the next section, we will present the experiment about how we decide the value of \hat{P} . In the future, more sophisticated methods to determine \hat{P} could be explored.

Depth Updating Strategy: The uncertainty of depth comes mainly from the depth sensor's measurement precision and working range. Since we test our method on the ScanNet dataset and Matterport3D dataset, the camera information of the Structure Sensor used for their depth cameras is of interest. Based on the Structure Sensor's fact sheet [64] and the online resources, their precisions are 1% of the observed depth and 0.2% of the observed depth, respectively. Therefore, we can come up with the range of $[0.99\hat{d}_i^c, 1.01\hat{d}_i^c]$ for the ScanNet dataset and the range of $[0.998\hat{d}_i^c, 1.002\hat{d}_i^c]$, where \hat{d}_i^c is the measured depth. Based on pin-hole model, the updatable position range is actually a line segment of the ray formed by the camera center \mathbf{o} and the center point $\mathbf{c}_{i,j}$ of $f_{i,j}$ on the image plane, as shown in Fig. 3. For the facet with an observed

Table 1: Quantitative comparison on the manually selected 30 examples with manually annotated depth discontinuity from the ScanNet dataset. Our method, Huang et al.’s [24] and Senushkin et al.’s [21] are evaluated over all pixels, as well as only over pixels without observed depth values (values in parentheses). Bold font indicates the best one in a whole column. The mismatched depth values in rendered depth are removed as well.

Methods	<i>RMSE</i> ↓	<i>REL</i> ↓	<i>MAE</i> ↓	$\delta_{1.05}$ ↑	$\delta_{1.10}$ ↑	$\delta_{1.25^1}$ ↑	$\delta_{1.25^2}$ ↑	$\delta_{1.25^3}$ ↑	<i>MSSIM</i> ↑
Huang [24]	0.1191 (0.3378)	0.0167 (0.1018)	0.0339 (0.2026)	0.9374 (0.4356)	0.9676 (0.6960)	0.9875 (0.8888)	0.9974 (0.9853)	0.9991 (0.9952)	0.9667 (0.9950)
Senushkin [21]	0.1197 (0.3478)	0.0168 (0.1116)	0.0311 (0.1829)	0.9581 (0.6167)	0.9705 (0.7084)	0.9821 (0.8192)	0.9892 (0.8828)	0.9962 (0.9588)	0.9698 (0.9915)
Ours	0.0941 (0.2254)	0.0085 (0.0252)	0.0183 (0.0530)	0.9863 (0.9415)	0.9911 (0.9566)	0.9949 (0.9792)	0.9975 (0.9890)	0.9989 (0.9939)	0.9856 (0.9976)

Table 2: Quantitative comparison on 1699 examples randomly selected from the ScanNet dataset. Our method, Huang et al.’s [24] and Senushkin et al.’s [21] are evaluated over all pixels, as well as only over pixels without observed depth values (values in parentheses). Bold font indicates the best result in each column. Depth values at detected depth discontinuities are removed to reduce the effect of depth mismatches in the rendered depth.

Methods	<i>RMSE</i> ↓	<i>REL</i> ↓	<i>MAE</i> ↓	$\delta_{1.05}$ ↑	$\delta_{1.10}$ ↑	$\delta_{1.25^1}$ ↑	$\delta_{1.25^2}$ ↑	$\delta_{1.25^3}$ ↑	<i>MSSIM</i> ↑
Huang [24]	0.1820 (0.4111)	0.0351 (0.1466)	0.0549 (0.2194)	0.8678 (0.3722)	0.9224 (0.5950)	0.9673 (0.8317)	0.9878 (0.9432)	0.9944 (0.9770)	0.9490 (0.9890)
Senushkin [21]	0.1914 (0.4061)	0.0356 (0.1425)	0.0567 (0.2080)	0.8908 (0.5134)	0.9242 (0.6348)	0.9552 (0.7738)	0.9782 (0.8878)	0.9895 (0.9478)	0.9450 (0.9849)
Ours	0.1571 (0.3105)	0.0242 (0.0886)	0.0374 (0.1270)	0.9333 (0.6884)	0.9562 (0.7913)	0.9769 (0.8981)	0.9893 (0.9584)	0.9946 (0.9820)	0.9628 (0.9916)

depth, in each iteration, if the newly computed depth falls into the range, we update the depth value with the new depth value. Otherwise, we need to project the new depth value to the range and update the depth value with the end values of the range. For facets without an observed depth, we update the depth value to the newly computed one freely, unless it is beyond the camera sensing range of 0.1 to 10 meters.

5. Experimental Results

5.1. Dataset and Benchmark

In this work, we firstly evaluate the proposed method on 30 selected examples with manually annotated depth discontinuity edges from each of the ScanNet dataset [26] and Matterport3D dataset [65]. Then, we utilize the automatic depth discontinuity detection, RINDNet [66], to conduct more experiments on 1699 images of ScanNet that include complex surfaces and clustered scenarios as well as the selected 30 examples for validation. Both of them provide raw RGB-D images, camera intrinsics and extrinsics, and high-quality reconstructed meshes. For comparison experiments with Huang et al.’s method [24] and Senushkin et al.’s method [21], the resolution of RGB-D images and rendered depth is resized to 320×256, and the same normal map predicted from the trained model [59] is used as input for the proposed method and the other two works.

5.2. Evaluation Metrics

For evaluating the depth completion results, there are five metrics used to compare the completion results with the rendered depth map. It should be noted that the rendered depth map still has pixels without valid depth. Therefore, these pixels are discarded from both of the completion results and the rendered depth map during evaluation and $\mathbf{x}_i \in val$ is denoted as a pixel with valid depth on the rendered depth map. Given the rendered depth D^* and completion result D , the first metric is Root Mean Square Error (RMSE):

$$\sqrt{\frac{1}{|val|} \sum_{\mathbf{x}_i \in val} \|D(\mathbf{x}_i) - D^*(\mathbf{x}_i)\|^2} \quad (12)$$

The second metric is Mean Absolute Error (MAE):

$$\frac{1}{|val|} \sum_{\mathbf{x}_i \in val} \|D(\mathbf{x}_i) - D^*(\mathbf{x}_i)\| \quad (13)$$

The third metric is Absolute Relative Error (Abs. Rel.):

$$\frac{1}{|val|} \sum_{\mathbf{x}_i \in val} \frac{|D(\mathbf{x}_i) - D^*(\mathbf{x}_i)|}{|D^*(\mathbf{x}_i)|} \quad (14)$$

The fourth metric is the percentage of pixels δ_i , within the relative error range $e_r \in \{1.05, 1.10, 1.25, 1.25^2, 1.25^3\}$, and the counted pixels should satisfy the following inequality:

$$\max(\frac{D^*(pix)}{D(pix)}, \frac{D(pix)}{D^*(pix)}) < e_r, pix \in val \quad (15)$$

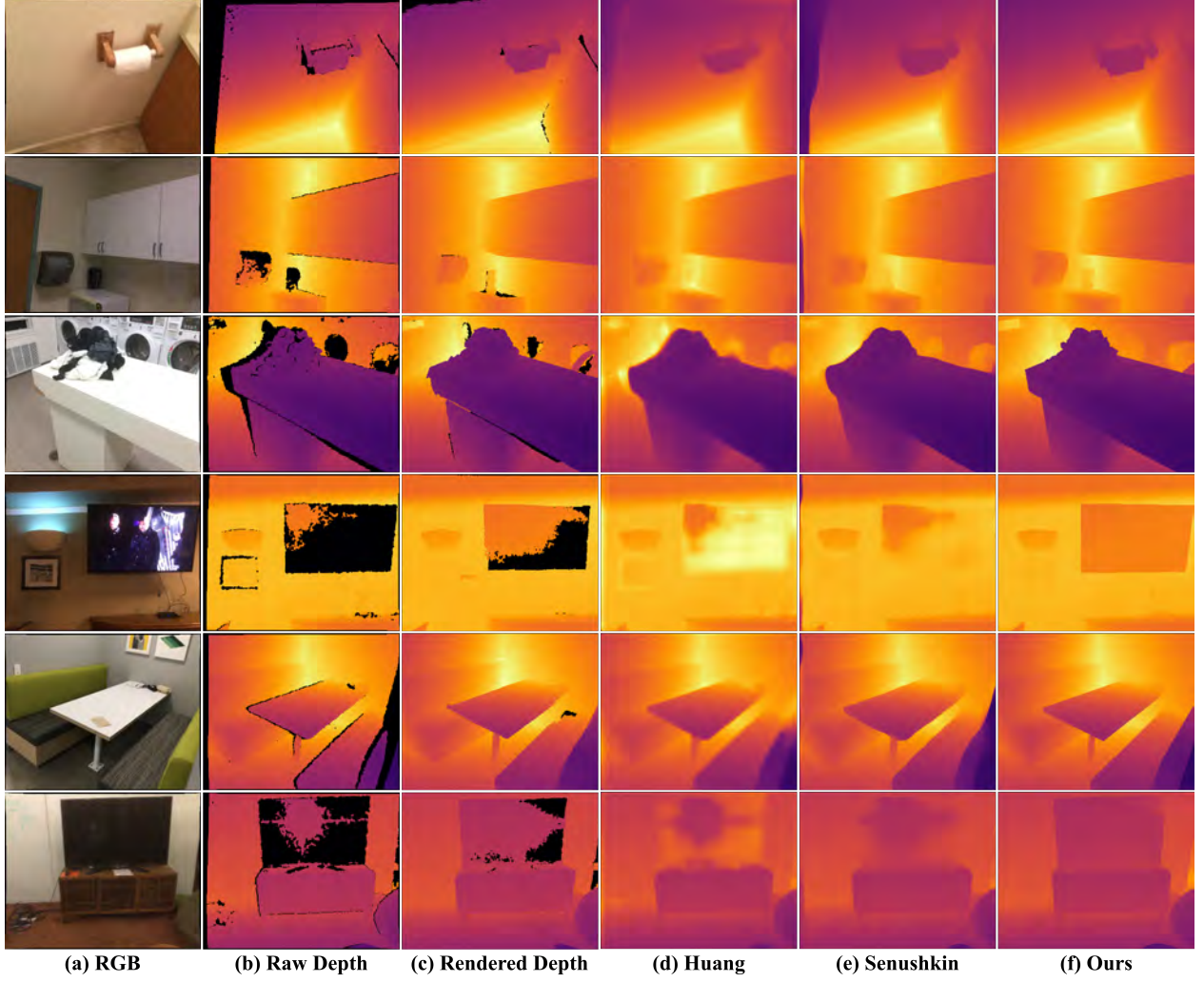


Fig. 5. The first column shows the selected (a) RGB images from the ScanNet dataset, and the other columns include qualitative comparisons of (b) raw depth, (c) rendered depth, (d) Huang et al.’s [24] depth completion, (e) Senushkin et al.’s [21] depth completion, and (f) **our results using manually annotated depth discontinuity**. Black pixels represent invalid depth values.

The fifth metric is the mean Structural Similarity Index Measure (MSSIM) [67]. It is used to evaluate the overall image quality.

$$MSSIM(X, Y) = \frac{1}{\mathfrak{N}} \sum_{i=0}^{\mathfrak{N}-1} SSIM(x_i, y_i), \quad (16)$$

where \mathfrak{N} is the number of pixels on the image. Before evaluating depth completion with MSSIM, the invalid values of the rendered depth map are assigned to 0, and the same pixels of completion results are assigned to 0 as well. According to [67], the definition of $SSIM(x_i, y_i)$ is:

$$SSIM(x_i, y_i) = \frac{(2\mu_{x_i}\mu_{y_i} + C_1)(2\sigma_{x_i y_i} + C_2)}{(\mu_{x_i}^2 + \mu_{y_i}^2 + C_1)(\sigma_{x_i}^2 + \sigma_{y_i}^2 + C_2)}, \quad (17)$$

where μ_{x_i} and μ_{y_i} are mean values of the local square window centered in x_i and y_i respectively, and $C_1 = 0.01^2$, $C_2 = 0.03^2$.

σ_{x_i} and σ_{y_i} are standard deviations:

$$\begin{cases} \sigma_{x_i} = \sqrt{\frac{1}{\mathfrak{M}-1} \sum_{j=0}^{\mathfrak{M}-1} (x_j - \mu_{x_i})^2} \\ \sigma_{y_i} = \sqrt{\frac{1}{\mathfrak{M}-1} \sum_{j=0}^{\mathfrak{M}-1} (y_j - \mu_{y_i})^2} \end{cases} \quad (18)$$

where \mathfrak{M} is the number of pixels in the local square window, and x_j and y_j belong to the corresponding local square windows. In practice, the size of the square local window is 11×11 , and a Gaussian filter with a standard deviation of 1.5 samples is applied.

5.3. Implementation Details

The proposed method is implemented with Python, and the function, *cholesky*, is imported from the *sksparse.cholmod* package for large matrix factorization. The code runs on the



Fig. 6. Qualitative comparison of point clouds generated from (a) raw depth from the ScanNet dataset, (b) rendered depth, (c) Huang et al.’s [24] depth completion, (d) Senushkin et al.’s [21] depth completion, and (e) **our results using manually annotated depth discontinuity**. As shown inside the circled regions, our method has significant advantages in dealing with depth discontinuities and thus avoids “stretched” distortions around object boundaries, i.e., the screen and the wall behind it. Besides, the edge of captured scenes still has high-quality depth completion results.

Ubuntu 20.04 server equipped with AMD Ryzen Threadripper PRO 3955WX@4.3GHz and 128 GB RAM.

The iteration termination condition depends on the position energy difference \mathcal{E}_d and normal energy difference \mathcal{E}_n . Specifically, after updating normal constraints and position constraints for \mathbf{b}_n^t and \mathbf{b}_o^t at the t th iteration, the solved \mathbf{x}^t is used to compute the normal energy $\|\mathbf{A}_n \mathbf{x}^t - \mathbf{b}_n^t\|^2$ and position energy $\|\mathbf{A}_o \mathbf{x}^t - \mathbf{b}_o^t\|^2$. Then, the position energy difference \mathcal{E}_d and normal energy difference \mathcal{E}_n measure the absolute difference of normal energy and position energy, respectively, between the current t and pre-

vious $t-1$ iteration. When both \mathcal{E}_d and \mathcal{E}_n are less than or equal to 10^{-4} , the iteration terminates.

5.4. Performance

We test our method over the depth images from ScanNet and Matterport3D, and the metric values are measured over 1) all pixels and 2) only the ones without observed depth values (see the values in parentheses in Table. 1, Table. 2 and Table. 3). The same set of tests was applied to Huang et al.’s [24] and Senushkin et al.’s [21] as well, and the results are shown in Ta-

Table 3: Quantitative comparison on the manually selected 30 examples with manually annotated depth discontinuity from the Matterport3D dataset. Our method, Huang et al.’s [24] and Senushkin et al.’s [21] are evaluated over all pixels, as well as only over pixels without observed depth values (values in parentheses). Bold font indicates the best one in a whole column. The mismatched depth values in rendered depth are removed as well.

Methods	$RMSE\downarrow$	$REL\downarrow$	$MAE\downarrow$	$\delta_{1.05}\uparrow$	$\delta_{1.10}\uparrow$	$\delta_{1.25^1}\uparrow$	$\delta_{1.25^2}\uparrow$	$\delta_{1.25^3}\uparrow$	$MSSIM\uparrow$
Huang [24]	0.2558 (0.3413)	0.0889 (0.1530)	0.0752 (0.1225)	0.6144 (0.2816)	0.7321 (0.4840)	0.9175 (0.8564)	0.9561 (0.9303)	0.9657 (0.9491)	0.9477 (0.9646)
Senushkin [21]	0.5247 (0.5059)	0.2955 (0.3676)	0.2352 (0.2367)	0.5409 (0.3485)	0.6157 (0.4809)	0.7448 (0.7234)	0.7929 (0.7818)	0.8350 (0.8294)	0.8848 (0.9354)
Ours	0.2009 (0.2477)	0.0523 (0.0762)	0.0474 (0.0712)	0.8314 (0.6965)	0.8969 (0.8160)	0.9435 (0.9119)	0.9657 (0.9492)	0.9718 (0.9617)	0.9683 (0.9802)

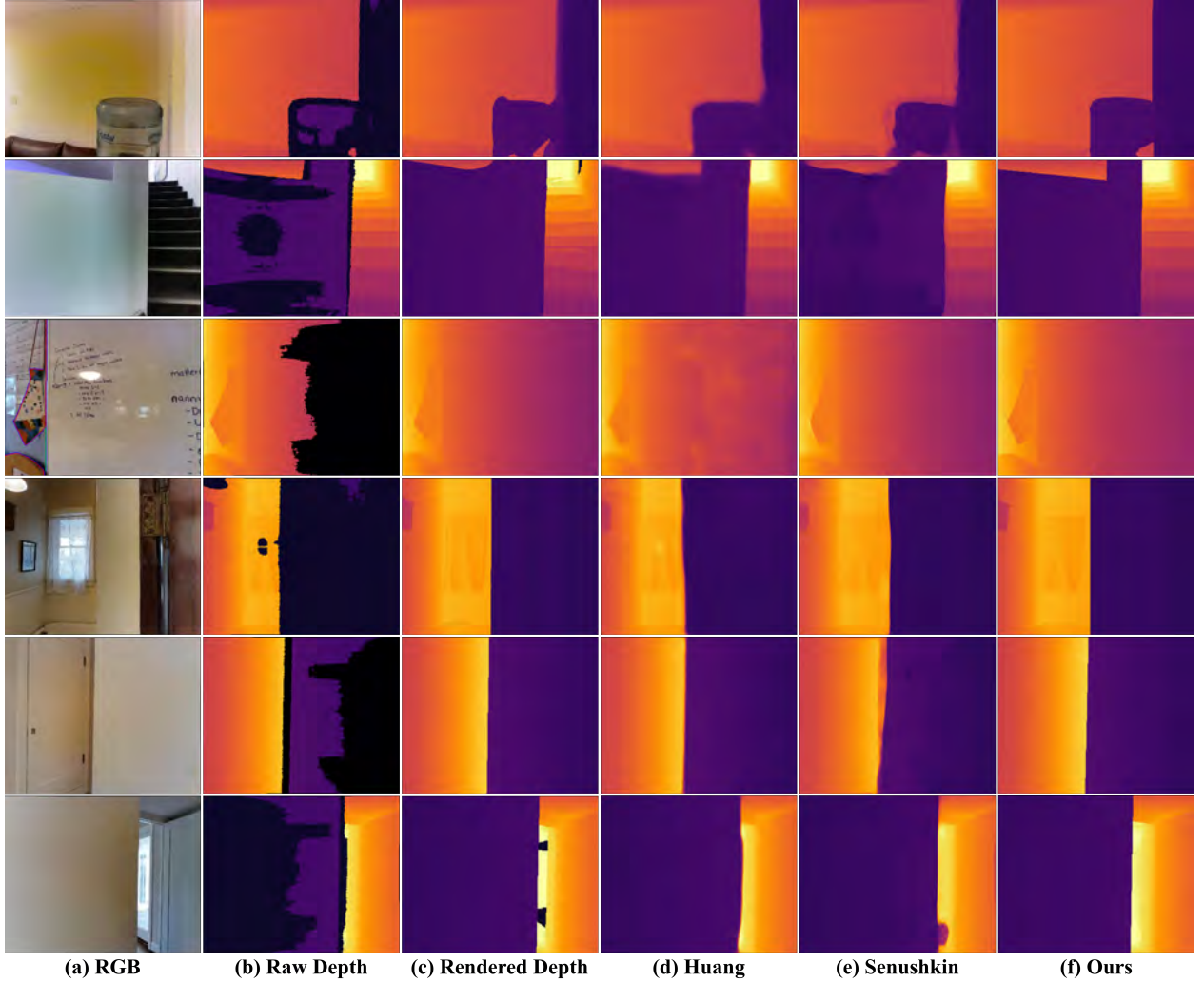


Fig. 7. The first column shows the selected (a) RGB images from the Matterport3D dataset, and the other columns include qualitative comparisons of (b) raw depth, (c) rendered depth, (d) Huang et al.’s [24] depth completion, (e) Senushkin et al.’s [21] depth completion, and (f) **our results using manually annotated depth discontinuity**. Black pixels represent invalid depth values.

ble. 1, Table. 2 and Table. 3. From the results shown in tables, our method outperforms two other methods on most metrics. If we only consider those pixels without observed depth, our method is even better. It is reasonable to pay more attention to

the errors on pixels without observed depth values instead of all pixels since pixels without observed depth values are the target of the depth completion. Moreover, the average error over all pixels will dilute the error, as those pixels with observed depth

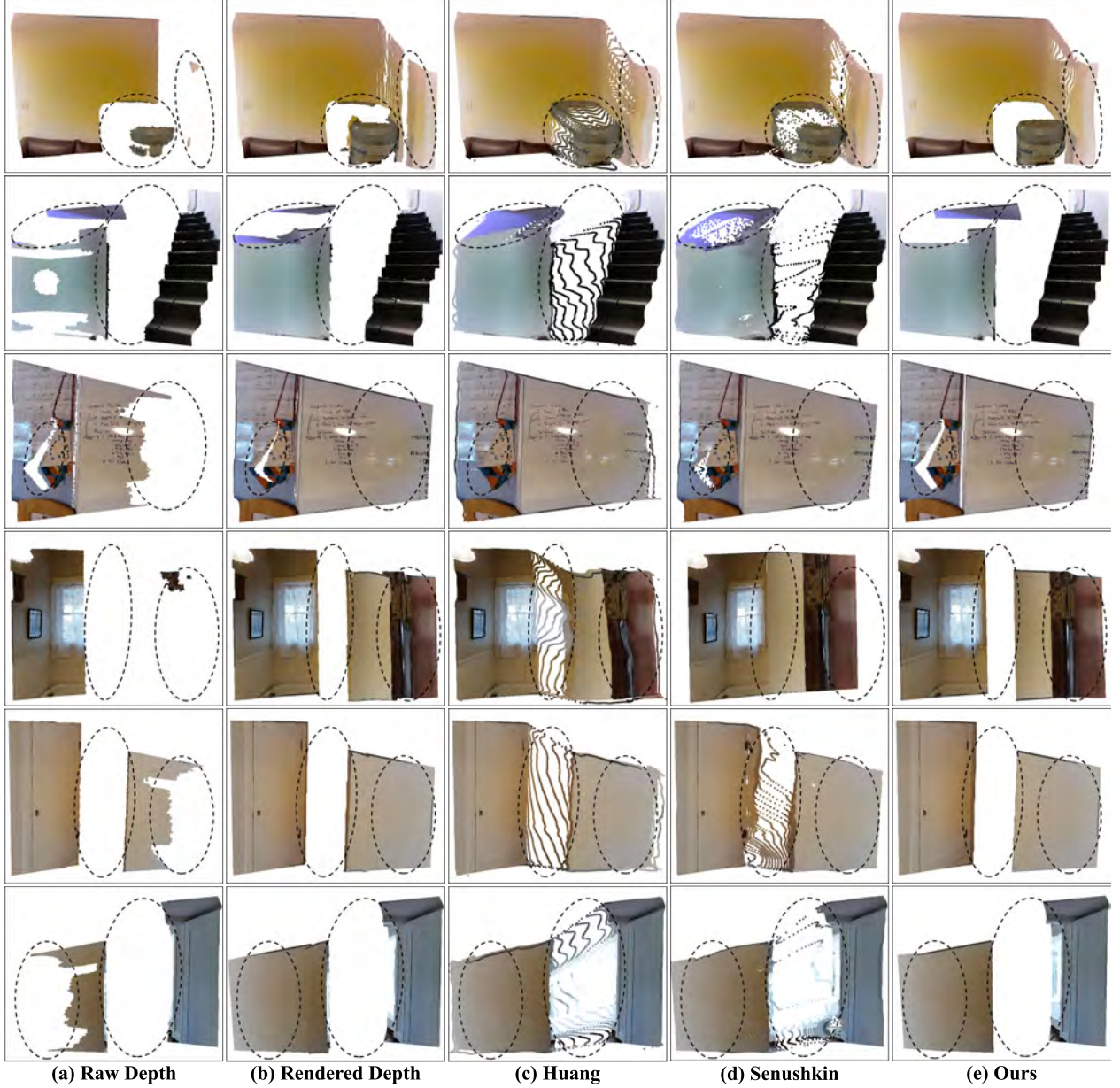


Fig. 8. Qualitative comparison of point clouds generated from (a) raw depth from the Matterport3D dataset, (b) rendered depth, (c) Huang et al.’s [24] depth completion, (d) Senushkin et al.’s [21] depth completion, and (e) **our results using manually annotated depth discontinuity**.

values have very low errors in general. Therefore, some results from Huang et al. and Senushkin et al. show low quality when visualized on the point cloud, even with reasonable error values (see Figs. 6 and 8). Considering this, in our other results, we only measure the errors over the pixels without observed depth values. Further, the results in Table. 3 show that our method has a particularly better performance compared with the other two over the metrics REL and $\delta_{1.05}$, which measure the relative depth accuracy. After applying the automatic depth discontinuity detection method, Table. 2 shows the results conducted on 1699 randomly selected examples that include complex surfaces and cluttered objects in the images, and our method re-

mains highly competitive. More results of manually annotated and automatically detected depth discontinuity are shown in Section 6.

Besides the point clouds in Figs. 6 and 8, we also visualize the depth images for a qualitative evaluation in Figs. 5 and 7. From the depth images, our method successfully generates the missing depth values and preserves the clear, sharp boundaries between different objects. The main reason is that we explicitly consider the discontinuity of depth across different objects and process it through mesh-based geometric processing with topological modification. In other words, our method supports mesh surfaces of arbitrary topology with extracted boundaries

as input, and the adjacent pixels on the RGB image do not affect each other during the optimization when they have distinct depths. This advantage allows for better processing of the object that is independent of surrounding objects, as shown in the fourth row of Fig. 5. However, Huang et al.’s and Senushkin et al.’s methods cannot handle the depth-discontinuity well and introduce large errors around the boundaries, as shown in the depth images (see Figs. 5 and 7). The point clouds further indicate the “stretched” distortions around the boundaries between distinct objects (shown in Figs. 6 and 8). Additionally, the range of deformation for each facet is determined only by its depth/normal uncertainty, so this “stretched” distortion can be eliminated in our method to maintain high-quality depth completion around the boundaries of objects as shown in the last row of Figs. 6 and 8.

Table 4: Time consumption for selected examples from the ScanNet Dataset. The index of examples is the same as in Fig. 9, and the unit of time consumption is seconds (s). The last line records the average time.

Index	Init.(s)	Projection(s)	Solve(s)	Update(s)	Iters.	Total(s)
01	15.79	0.82	0.40	1.27	42	120.63
02	15.15	0.83	0.45	1.26	56	157.21
03	17.59	0.82	0.56	1.27	51	153.04
04	15.31	0.82	0.45	1.25	65	178.64
05	16.85	0.81	0.35	1.20	47	127.54
06	14.77	0.82	0.54	1.28	102	283.84
07	14.42	0.84	0.21	1.29	60	154.66
08	15.74	0.87	0.40	1.32	41	122.04
09	17.17	0.88	0.59	1.26	51	156.48
10	17.88	0.90	0.66	1.32	45	147.39
11	17.24	0.86	0.59	1.30	68	204.29
12	18.30	0.89	0.55	1.28	59	179.00
13	17.98	0.90	0.58	1.30	83	248.61
14	16.67	0.84	0.46	1.26	86	237.15
15	17.07	0.80	0.06	1.26	35	91.38
16	18.08	0.87	0.50	1.29	31	100.66
17	16.96	0.88	0.49	1.35	78	229.19
18	15.21	0.85	0.26	1.28	76	196.47
19	17.54	0.88	0.26	1.30	96	250.72
20	16.52	0.87	0.55	1.33	87	255.94
21	17.42	0.86	0.19	1.27	71	182.57
22	15.76	0.88	0.21	1.29	67	175.24
23	15.99	0.86	0.40	1.30	74	205.63
24	17.60	0.90	0.40	1.36	87	248.79
25	15.46	0.90	0.29	1.32	59	163.68
26	16.82	0.87	0.25	1.29	60	161.79
27	17.09	0.87	0.50	1.28	55	163.07
28	23.77	0.86	0.36	1.30	88	245.45
29	16.04	0.90	0.40	1.29	61	174.16
30	15.39	0.92	0.50	1.34	99	288.35
Average	16.79	0.86	0.41	1.29	66	186.79
Huang [24]	-	-	-	-	-	0.07
Senushkin [21]	-	-	-	-	-	1.01

We also record the time statistics and take the results from ScanNet as an example. From the left column to the right, Table 4 reports the total time for initialization, the average time

for element geometry projection and proximity function minimization in one iteration, the average time for updating constraints in one iteration, the total number of iterations, and the overall time for completing the computation for one scene. On average, our method requires 186.79 seconds to complete the computation across 30 examples. There are two bottlenecks in computational efficiency: the initialization and the constraints updating. The initialization process only needs to be performed once and can potentially be pre-calculated and loaded when the program starts. The other bottleneck is the updating of normals and depth values, which requires significant computation based on the strategies presented in Section 4.4. For comparison, the average time consumption of Huang et al.’s [24] and Senushkin et al.’s [21] over 30 examples is recorded in Table 4 as well. **Indeed, our method currently is slow, and it cannot meet the real-time or near-real-time requirements for some Industry 4.0 applications. Nevertheless, it possesses the unique advantage of effectively and explicitly leveraging depth discontinuity information to fundamentally address stretched regions and mismatched depths. The results of using automatic depth discontinuity detection shown in Table 2 indicate the superior performance of our method in depth completion accuracy over the two benchmark methods. More directions, including reformulating the optimization problem for fewer iteration steps, localizing the computation to solve smaller linear systems, and implementing GPU-based parallelization, are promising for improving the efficiency in the future.**

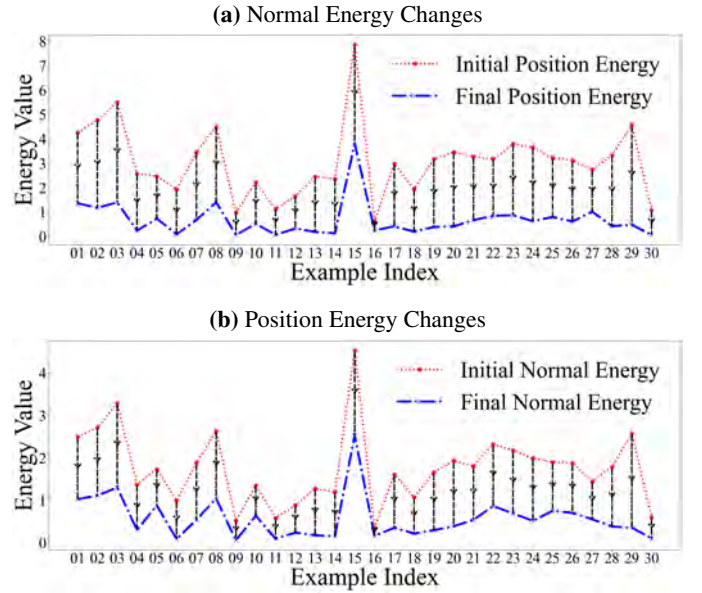


Fig. 9. Energy changes before and after iterations. The red points represent the initial position/normal energy, while the blue points represent the energy after convergence. The red dotted line on the upper side and the blue dotted line on the lower side illustrate the energy reduction after iterations.

As mentioned in Section 3.1, we convert the depth completion task into an optimization problem and solve it by minimizing the shape variation of the surface mesh. Therefore, the

Table 5: The quantitative evaluation of different conditions for normal and depth updating strategies on the selected 30 examples from the ScanNet dataset. Bold font indicates the best for the whole column. The mismatched depth values in rendered depth are removed.

Depth	P	$RMSE\downarrow$	$REL\downarrow$	$MAE\downarrow$	$\delta_{1.05}\uparrow$	$\delta_{1.10}\uparrow$	$\delta_{1.25^1}\uparrow$	$\delta_{1.25^2}\uparrow$	$\delta_{1.25^3}\uparrow$	$MSSIM\uparrow$
×	1.0	0.2262	0.0290	0.0613	0.9258	0.9526	0.9788	0.9890	0.9937	0.9972
✓	1.0	0.2263	0.0263	0.0552	0.9347	0.9533	0.9789	0.9889	0.9937	0.9975
×	0.75	0.2259	0.0259	0.0543	0.9382	0.9546	0.9790	0.9890	0.9938	0.9976
✓	0.75	0.2258	0.0256	0.0537	0.9388	0.9546	0.9790	0.9890	0.9938	0.9976
×	0.5	0.2258	0.0258	0.0541	0.9381	0.9551	0.9790	0.9890	0.9938	0.9976
✓	0.5	0.2254	0.0252	0.0530	0.9415	0.9566	0.9792	0.9890	0.9939	0.9976
×	0.25	0.2255	0.0253	0.0532	0.9390	0.9560	0.9791	0.9889	0.9938	0.9976
✓	0.25	0.2689	0.0339	0.0709	0.9139	0.9344	0.9660	0.9837	0.9916	0.9965
×	0	0.2255	0.0258	0.0541	0.9364	0.9540	0.9789	0.9889	0.9939	0.9976
✓	0	0.2256	0.0254	0.0533	0.9370	0.9544	0.9789	0.9890	0.9938	0.9976

Table 6: The quantitative evaluation of different conditions for normal and depth updating strategies on the manually selected 30 examples from the Matterport3D dataset. Bold font indicates the best for the whole column. The mismatched depth values in rendered depth are removed.

Depth	P	$RMSE\downarrow$	$REL\downarrow$	$MAE\downarrow$	$\delta_{1.05}\uparrow$	$\delta_{1.10}\uparrow$	$\delta_{1.25^1}\uparrow$	$\delta_{1.25^2}\uparrow$	$\delta_{1.25^3}\uparrow$	$MSSIM\uparrow$
×	1.0	0.7573	0.1880	0.1418	0.6609	0.7876	0.8840	0.9302	0.9438	0.9674
✓	1.0	1.0702	0.2397	0.2225	0.6598	0.7869	0.8836	0.9273	0.9349	0.9541
×	0.75	0.2477	0.0764	0.0713	0.6954	0.8151	0.9119	0.9492	0.9616	0.9802
✓	0.75	0.2477	0.0762	0.0712	0.6965	0.8160	0.9119	0.9492	0.9617	0.9802
×	0.5	0.2477	0.0773	0.0717	0.6922	0.8134	0.9075	0.9493	0.9619	0.9801
✓	0.5	0.2477	0.0774	0.0717	0.6913	0.8126	0.9075	0.9494	0.9619	0.9801
×	0.25	0.2483	0.0788	0.0725	0.6847	0.8097	0.9055	0.9493	0.9620	0.9800
✓	0.25	0.2483	0.0790	0.0726	0.6837	0.8079	0.9055	0.9493	0.9620	0.9800
×	0	0.2485	0.0806	0.0733	0.6622	0.8044	0.9042	0.9494	0.9621	0.9800
✓	0	0.2486	0.0808	0.0734	0.6628	0.8036	0.9041	0.9494	0.9621	0.9800

robustness of the proposed method can be evaluated by comparing the energy changes before and after optimization. We plot both the position and normal energies of the first and last iterations for all 30 examples from the ScanNet dataset (see Fig. 9). Obviously, both position and normal energies have a reduction in the last iteration compared with the ones in the first iteration, which indicates the robustness of our local-global optimization scheme.

5.5. Ablation Study

Depth/Normal Updating Strategies. To verify our depth and normal updating strategies, we set up an experiment with different conditions. For updating depth, there were two strategies: updating the depth or not. For normal updating, the key was to figure out the angular uncertainty $\hat{\theta}_{i,j}$, such that an updating range $[0, \hat{\theta}_{i,j}]$ was obtained. $\hat{\theta}_{i,j}$ can be calculated using Eq. 11 by setting a cumulative probability P . Therefore, what was the value of P is the essential question to answer. We uniformly sampled the value range of P (i.e., $[0, 1]$) by 0.25 and calculated the updating range with these values (i.e., 0, 0.25, 0.5, 0.75, and 1.0). By combining different conditions of depth updating and normal updating, we had a total of ten cases. Table. 5 and Table. 6 included all the results in different cases for ScanNet and

Matterport3D. The results indicated that the combination of updating depth & $P = 0.5$, and the combination of updating depth & $P = 0.75$ for ScanNet and Matterport3D, respectively, performed better on most metrics, and therefore, these setups are adopted in all our experiments. For qualitatively validating the normal updating strategies, some examples are selected from the ScanNet dataset (first row of Fig. 10) and the Matterport3D dataset (second and third rows of Fig. 10). Although the quantitative differences between these results were not significant, the qualitative results, as shown in Fig. 10, show that normal update strategies effectively improve the robustness of our method.

Topological Modification for Depth-Discontinuity. One assumption of our method is that the depth-discontinuity processed through a topological modification and taken into account in the following optimization will reduce the errors, especially around the boundaries across different objects with very distinct depths. Therefore, we conduct experiments to validate this assumption. We apply our method to the same set of depth images twice, with all other setups the same, but one with depth discontinuity considered and processed and one not. The results shown in Fig. 11 suggest that without discontinuity considered and processed, the regions around boundaries

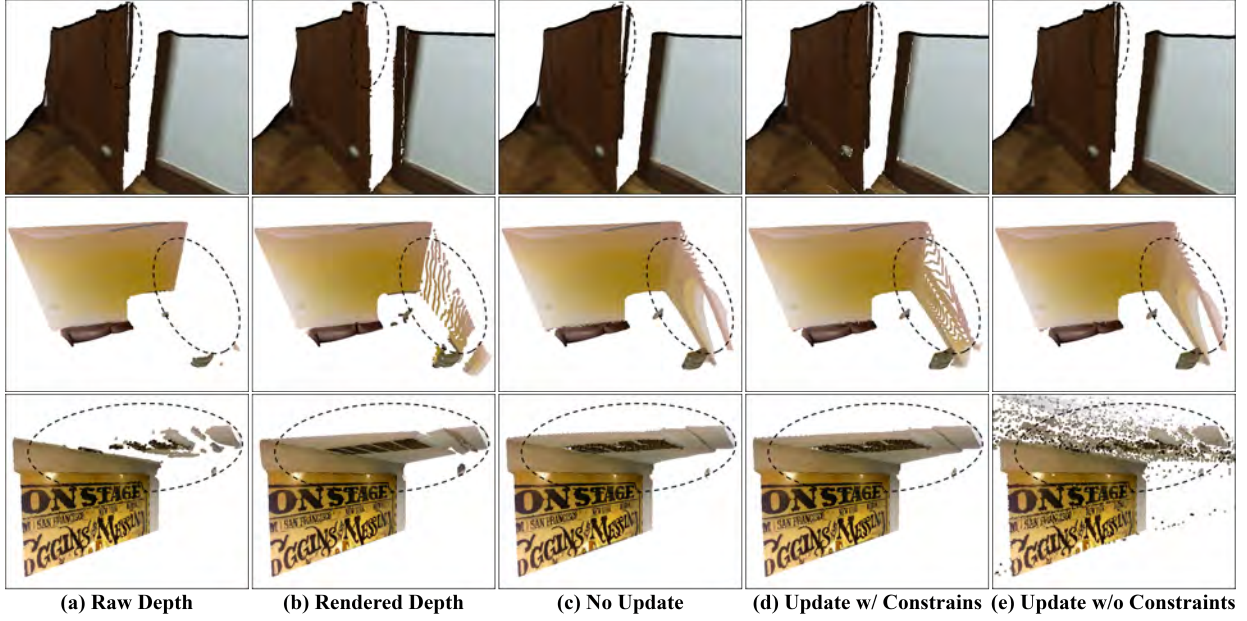


Fig. 10. Qualitative comparison of point clouds generated from (a) raw depths, (b) rendered depths, (c) the depths computed from the combination of updating depth but not updating normal, (d) the depths computed from the combination of updating depth and updating normal with constraints, and (e) the depths computed from the combination of updating depth and updating normal without constraints. These examples are selected from the ScanNet dataset (first row of Fig. 10) and the Matterport3D dataset (second and third rows of Fig. 10)

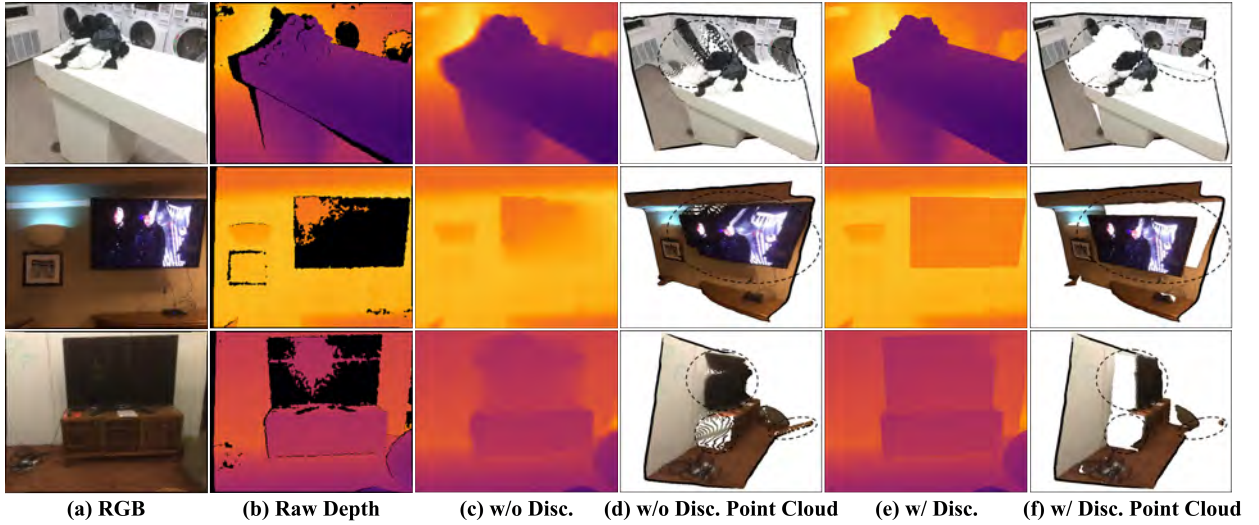


Fig. 11. Qualitative validation results reflect the importance of explicitly handling depth discontinuity. The blurred boundaries in (c) become clearer in (e), while the stretched regions in (d) are removed in (f).

had large errors and appear as “stretched” distortions. Besides, Table 7 and Table 8 records the quantitative comparison of whether to introduce topological modification and whether to remove mismatching depths. However, the results of the ScanNet dataset and the Matterport3D dataset show opposite behavior for whether depth-discontinuity is introduced or not. We find that the inconsistency between the RGB image and the rendered depth in the ScanNet dataset is more serious than in the Matterport3D dataset. Therefore, we remove the inconsistent

pixels during evaluation to obtain a more accurate comparison. Finally, both the qualitative and quantitative results validate our assumption of topological modification based on depth discontinuity.

5.6. Applications

To evaluate our approach in real-life applications, we select five scenarios as shown in Fig. 12. The raw depth maps are

Table 7: Quantitative comparison of whether to introduce topological modification and whether to remove mismatching depths when evaluating the manually selected 30 examples from the ScanNet dataset over all pixels, as well as only pixels without observed depth values (values in the parenthesis). Bold font indicates the best for the whole column.

Depth Disc.	Mismatching (Eval)	$RMSE\downarrow$	$REL\downarrow$	$MAE\downarrow$	$\delta_{1.05}\uparrow$	$\delta_{1.10}\uparrow$	$\delta_{1.25^1}\uparrow$	$\delta_{1.25^2}\uparrow$	$\delta_{1.25^3}\uparrow$	$MSSIM\uparrow$
WITH	KEEP	0.1233 (0.2687)	0.0106 (0.0338)	0.0227 (0.0707)	0.9797 (0.9164)	0.9859 (0.9350)	0.9921 (0.9660)	0.9963 (0.9837)	0.9984 (0.9916)	0.9573 (0.9965)
WITHOUT	KEEP	0.1012 (0.2284)	0.0108 (0.0430)	0.0228 (0.0862)	0.9742 (0.8213)	0.9836 (0.8881)	0.9914 (0.9496)	0.9966 (0.9842)	0.9988 (0.9946)	0.9615 (0.9966)
WITH	REMOVE	0.0941 (0.2254)	0.0085 (0.0252)	0.0183 (0.0530)	0.9863 (0.9415)	0.9911 (0.9566)	0.9949 (0.9792)	0.9975 (0.9890)	0.9989 (0.9939)	0.9856 (0.9976)
WITHOUT	REMOVE	0.0914 (0.2134)	0.0096 (0.0378)	0.0202 (0.0754)	0.9792 (0.8492)	0.9874 (0.9080)	0.9932 (0.9566)	0.9974 (0.9866)	0.9990 (0.9953)	0.9836 (0.9973)

Table 8: Quantitative comparison of whether to introduce topological modification and whether to remove mismatching depths when evaluating the manually selected 30 examples from the Matterport3D dataset over all pixels, as well as only pixels without observed depth values (values in the parenthesis). Bold font indicates the best for the whole column.

Depth Disc.	Mismatching (Eval)	$RMSE\downarrow$	$REL\downarrow$	$MAE\downarrow$	$\delta_{1.05}\uparrow$	$\delta_{1.10}\uparrow$	$\delta_{1.25^1}\uparrow$	$\delta_{1.25^2}\uparrow$	$\delta_{1.25^3}\uparrow$	$MSSIM\uparrow$
WITH	KEEP	0.2153 (0.2530)	0.0568 (0.0866)	0.0504 (0.0719)	0.8287 (0.6813)	0.8941 (0.8116)	0.9412 (0.9090)	0.9639 (0.9503)	0.9708 (0.9620)	0.9563 (0.9769)
WITHOUT	KEEP	0.2966 (0.4039)	0.1215 (0.2347)	0.0763 (0.1316)	0.8157 (0.6497)	0.8718 (0.7602)	0.9139 (0.8471)	0.9414 (0.8992)	0.9534 (0.9234)	0.9427 (0.9627)
WITH	REMOVE	0.2009 (0.2477)	0.0523 (0.0762)	0.0474 (0.0712)	0.8314 (0.6965)	0.8969 (0.8160)	0.9435 (0.9119)	0.9657 (0.9492)	0.9718 (0.9617)	0.9683 (0.9802)
WITHOUT	REMOVE	0.2913 (0.4007)	0.1154 (0.2245)	0.0725 (0.1264)	0.8243 (0.6634)	0.8807 (0.7754)	0.9219 (0.8615)	0.9463 (0.9079)	0.9553 (0.9260)	0.9545 (0.9670)

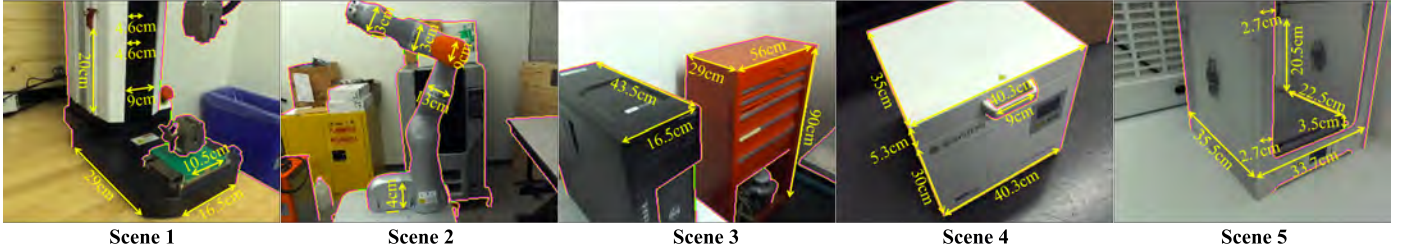


Fig. 12. Five real-world scenarios with physical measurements of some dimensions and depth discontinuity visualized by three colors. Some depth discontinuities around distant or trivial objects are ignored, and the physical measurements focus on objects of interest in these scenarios.

captured by ZED mini depth cameras at 1920×1080 resolution and 15 frames per second (FPS). These maps are center-cropped and scaled to 320×256 , as shown in the first and second columns of Fig. 13. We conduct a qualitative comparison (Fig. 13) between our method and the built-in methods of the ZED SDK, Huang et al., and Senushkin et al. From the completed depth images, both the ZED SDK and our proposed method generate clearer object boundaries compared to Huang et al. and Senushkin et al. However, the ZED SDK still fails to estimate depth in some regions (marked by black pixels). Meanwhile, Huang et al. and Senushkin et al. struggle to recover the correct spatial relationships between objects and the background wall. In contrast, our method fully utilizes depth-discontinuity information extracted from RGB im-

ages to guide the depth completion process. Furthermore, as shown in Fig. 14, the reconstructed point clouds from the ZED SDK, Huang et al., and Senushkin et al. exhibit “stretched” distortions near depth-discontinuity regions. For quantitative comparison, we physically measure the dimensions of labeled objects (Fig. 12) as a benchmark. These measurements are compared against counterparts from 3D point clouds generated by the ZED SDK, Huang et al., Senushkin et al., and our method, as well as those from raw data (Fig. 12). Table 9 summarizes the results across these five scenarios. Overall, our method outperforms all others in all scenarios. And the distances calculated from the raw depth map are the worst due to limitations of stereo vision principles in the ZED depth camera. The textureless surfaces lead to depth estimation er-

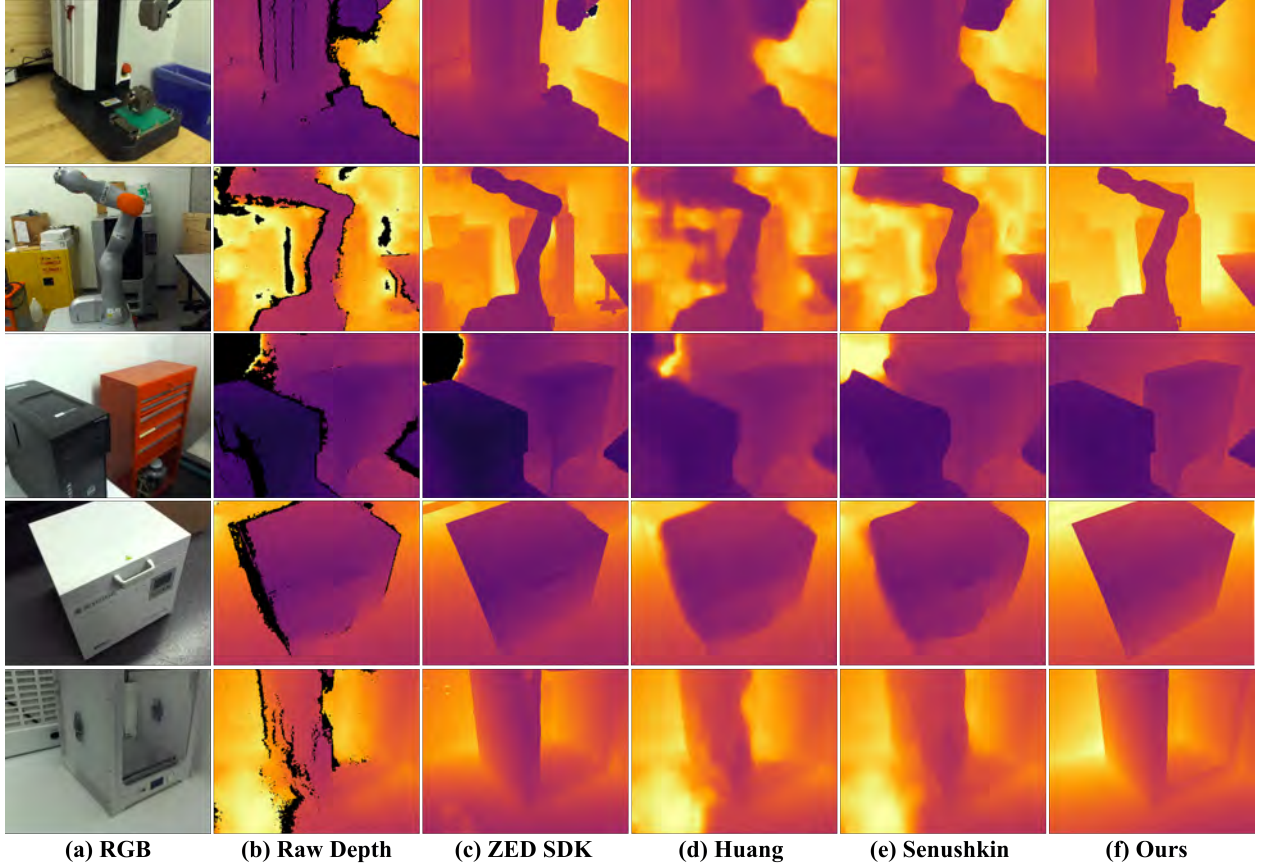


Fig. 13. The first column shows the (a) RGB images of three real-world scenarios, and the other columns include qualitative comparisons of (b) raw depths, (c) completed depths obtained from the ZED SDK built-in method, (d) Huang et al.’s [24] depth completion, (e) Senushkin et al.’s [21] depth completion, and (f) our results. Black pixels represent invalid depth values.

rors. Further, low-quality or even wrong raw depth can amplify the errors in reconstruction results of Huang et al.’s [24] and Senushkin et al.’s [21].

6. Discussion

Depth-Discontinuity Detection. Different from other methods, the depth-discontinuity in our work exists at the shared edges of adjacent pixels, which are represented by quadrilateral mesh facets. Therefore, we manually detect and annotate depth-discontinuity with three colors on RGB images for all selected examples from public datasets and the real-world scenarios (see Fig. 12). Although the proposed method excels at handling depth-discontinuity, the manual annotation is still a temporary limitation that prevents batch processing of large amounts of data. In addition, manual annotation may focus on the area of interest and ignore some other details, e.g., depth-discontinuity edges around the blue box in Scene 1 and the black table leg in Scene 2 (see Fig. 12). To reduce the impact of manual annotation of depth discontinuity, the depth discontinuity edge detection method, RINDNet [66], is applied on the selected examples from the ScanNet and Matterport3D datasets. The quantitative results are recorded in Table 10 and Table 11, while some qualitative results are displayed in Fig. 15. **RINDNet [66] is**

designed to estimate and distinguish four types of edges [68]: (1) surface-reflectance discontinuity, (2) illumination discontinuity, (3) surface-normal discontinuity, and (4) depth discontinuity. However, its detection results still suffer from incorrect or missing depth discontinuity edges, such as sharp shadows or high-contrast regions on a flat wall, or areas with discontinuous depth but similar RGB information. These failure cases can lead to incorrect mesh separation in our pipeline. If the mesh is incorrectly separated but the input depth and normal quality in that region are high, the generated depth will still be accurate. In contrast, failing to separate regions that should be distinct can cause our method to produce stretched regions in the results. Therefore, quantitatively and qualitatively, the results of manual annotation are better than those of automatic detection. A more robust discontinuity edge detection method remains an open problem.

Implications. This work has the potential to inspire other researchers to consider applying geometric processing techniques to depth completion tasks. In particular, we show that explicit operations on the mesh representation of a depth image allow geometric constraints (e.g., discontinuities) to be enforced effectively, providing a unique advantage that purely end-to-end methods do not possess. Researchers may further explore this direction by incorporating more explicit constraints and ge-

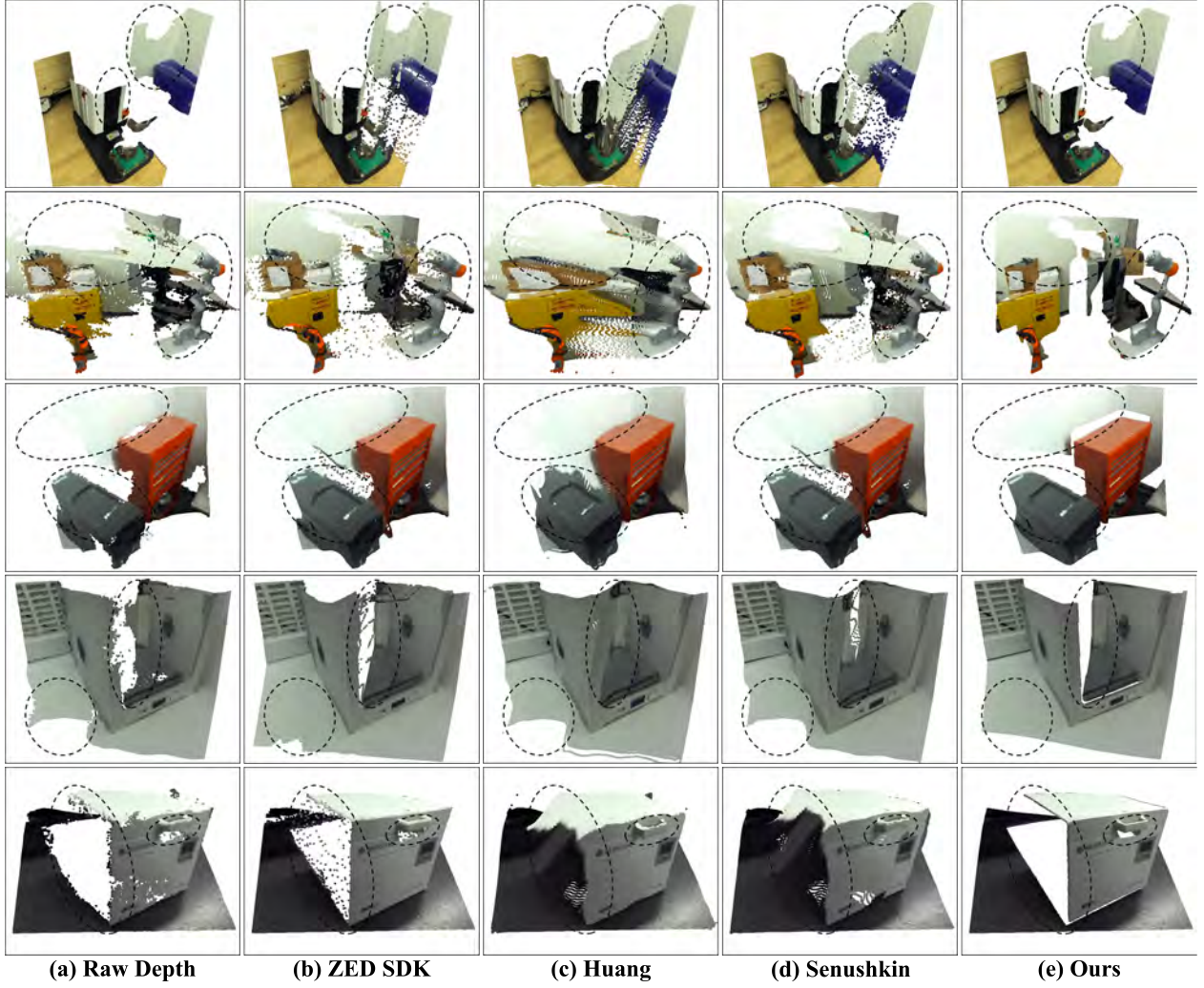


Fig. 14. Qualitative comparison of point clouds generated from (a) raw depths, (b) completed depths obtained from the ZED SDK built-in method, (c) complete depths of Huang et al.'s [24], (e) completed depths of Senushkin et al.'s [21] and our results.

Table 9: Quantitative comparisons are calculated based on the physical measurements in Fig. 12, and our results perform the best on all metrics.

Methods	$RMSE \downarrow$					$REL \downarrow$					$MAE \downarrow$				
	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5
<i>Raw Depth</i>	0.0269	0.7422	0.1559	0.0527	0.0639	0.0201	0.4742	0.0971	0.0840	0.2541	0.3394	3.6560	0.4709	0.0245	0.0323
<i>ZED SDK</i>	0.0112	0.1613	0.0514	0.0447	0.1220	0.0086	0.0956	0.0439	0.0997	1.8869	0.1407	0.7399	0.1118	0.0224	0.0558
<i>Huang [24]</i>	0.0212	0.2352	0.0808	0.2894	0.0336	0.0180	0.1549	0.0301	0.4978	0.4772	0.2637	1.3866	0.1459	0.1693	0.0213
<i>Senushkin [21]</i>	0.0227	0.3359	0.0404	0.2039	0.0242	0.0172	0.1556	0.0301	0.2706	0.1092	0.2880	1.2052	0.0761	0.0913	0.0149
<i>Ours</i>	0.0029	0.0083	0.0274	0.0186	0.0181	0.0024	0.0074	0.0189	0.0581	0.0603	0.0232	0.0642	0.0481	0.0157	0.0126

ometric processing algorithms into depth image completion. Moreover, this work highlights the importance of depth discontinuities in recovering spatial geometric relationships, pointing to a promising future research direction for depth completion. Because of the significantly enhanced depth accuracy around discontinuous regions of the depth map, our work also has the potential to benefit a variety of industrial applications, including robust object grasping, navigation for autonomous systems,

and accurate occlusion handling for XR visualization.

7. Conclusion and Future Work

We propose a depth image completion method based on discrete geometric processing. Our method converts a depth image into a quadrilateral mesh surface and explicitly processes depth

Table 10: Quantitative comparison on the manually selected 30 examples from the ScanNet dataset for studying the manual annotation effect. Our method with manual annotation and the method with automatic depth discontinuity detection are evaluated over all pixels, as well as only over pixels without observed depth values. Bold font indicates the best one in a whole column. The mismatched depth values in rendered depth are removed.

Methods	$RMSE\downarrow$	$REL\downarrow$	$MAE\downarrow$	$\delta_{1.05}\uparrow$	$\delta_{1.10}\uparrow$	$\delta_{1.25^1}\uparrow$	$\delta_{1.25^2}\uparrow$	$\delta_{1.25^3}\uparrow$	$MSSIM\uparrow$
<i>Ours (Manual)</i>	0.0941	0.0085	0.0183	0.9863	0.9911	0.9949	0.9975	0.9989	0.9856
	(0.2254)	(0.0252)	(0.0530)	(0.9415)	(0.9566)	(0.9792)	(0.9890)	(0.9939)	(0.9976)
<i>Ours (Automatic)</i>	0.1041	0.0105	0.0219	0.9766	0.9858	0.9923	0.9968	0.9989	0.9833
	(0.2355)	(0.0410)	(0.0795)	(0.8550)	(0.9126)	(0.9558)	(0.9825)	(0.9940)	(0.9970)

Table 11: Quantitative comparison of the manually selected 30 examples from the Matterport3D dataset for studying the manual annotation effect. Our method with manual annotation and the method with automatic depth discontinuity detection are evaluated over all pixels, as well as only over pixels without observed depth values. Bold font indicates the best one in a whole column. The mismatched depth values in rendered depth are removed.

Methods	$RMSE\downarrow$	$REL\downarrow$	$MAE\downarrow$	$\delta_{1.05}\uparrow$	$\delta_{1.10}\uparrow$	$\delta_{1.25^1}\uparrow$	$\delta_{1.25^2}\uparrow$	$\delta_{1.25^3}\uparrow$	$MSSIM\uparrow$
<i>Ours (Manual)</i>	0.2009	0.0523	0.0474	0.8314	0.8969	0.9435	0.9657	0.9718	0.9683
	(0.2477)	(0.0762)	(0.0712)	(0.6965)	(0.8160)	(0.9119)	(0.9492)	(0.9617)	(0.9802)
<i>Ours (Automatic)</i>	0.2170	0.0562	0.0534	0.8289	0.8875	0.9343	0.9596	0.9678	0.9691
	(0.2595)	(0.0850)	(0.0772)	(0.6831)	(0.7987)	(0.8969)	(0.9429)	(0.9582)	(0.9783)

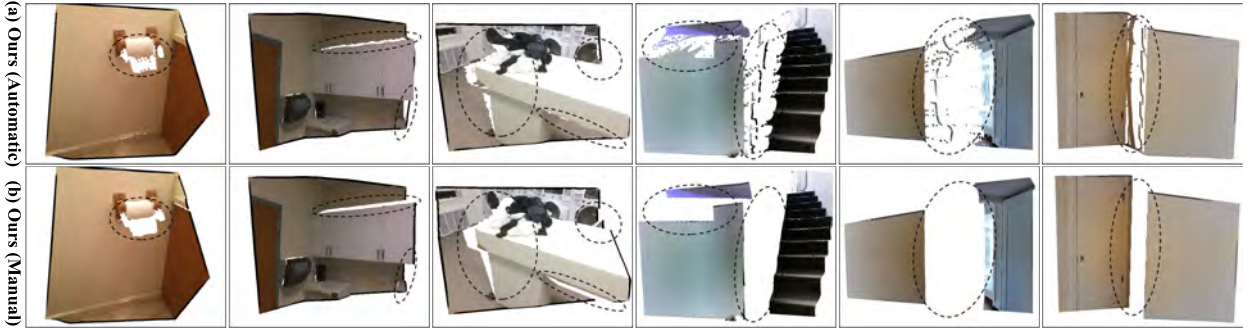


Fig. 15. Qualitative comparison of point clouds generated from (a) our method using manual annotation and (b) our method using automatic depth discontinuity detection [66].

discontinuities through edge detection and trimming. A constrained geometric optimization with normal and position constraints is formulated and solved with element geometry projection and proximity function minimization. We further address uncertainties in the input normal and raw depth data by developing dedicated updating strategies. Experiments on the ScanNet dataset and Matterport3D dataset, evaluated using five metrics, demonstrate that our method outperforms Huang et al.’s [24] and Senushkin et al.’s [21] approaches both quantitatively and qualitatively. Notably, our method achieves sharp discontinuities around object boundaries due to explicit discontinuity processing. Evaluations in five real-world scenarios further confirm superior performance compared to Huang et al., Senushkin et al., and the ZED SDK. While our proposed method shows promising results, it is still in its preliminary stage. Our current element geometry projection with normal constraints remains coupled with the observed depth from the raw depth image. In principle, it should only relate to the

shape of objects, not the position (the depth in this problem). Moreover, significant errors could be introduced if the observed depth contains inaccuracies. To address these problems, a new formulation for element geometry projection with position-independent normal constraints is planned for development. In theory, normal is the first-order derivative of a surface to describe geometric properties, and it is enough for shape deformation. Therefore, the idea is to construct normal constraints to be position-independent, which will be different from the current normal constraints described in Euclidean space. After decoupling normal and depth by the new formulation, position-independent normal constraints only describe the orientation of facets, while the observed depths are mainly responsible for providing the scale information of the whole captured scene. Additionally, depth-discontinuity detection remains a precondition that requires further study. The efficient and stable depth-discontinuity detection methods should be explored to provide accurate surface separation, i.e., filtering from edge detection

or incorporating advanced semantic segmentation based on the understanding of object surfaces. To speed up the computation for potential applications requiring real-time performance, we expect to develop an approximate but more efficient updating method, reduce the problem size to a smaller, localized region, and utilize GPU acceleration for parallel processing.

8. Acknowledgements

This paper acknowledges the support of the Natural Sciences & Engineering Research Council of Canada (NSERC) grant #RGPIN-2017-06707.

References

- [1] T. Czerniawski, F. Leite, Automated segmentation of rgb-d images into a comprehensive set of building components using deep learning, *Advanced Engineering Informatics* 45 (2020) 101131.
- [2] S. Chen, G. Fan, J. Li, Improving completeness and accuracy of 3d point clouds by using deep learning for applications of digital twins to civil structures, *Advanced Engineering Informatics* 58 (2023) 102196.
- [3] M. Radanovic, K. Khoshelham, C. Fraser, Aligning the real and the virtual world: Mixed reality localisation using learning-based 3d–3d model registration, *Advanced Engineering Informatics* 56 (2023) 101960.
- [4] T. Zhou, Q. Zhu, J. Du, Intuitive robot teleoperation for civil engineering operations with virtual reality and deep learning scene reconstruction, *Advanced Engineering Informatics* 46 (2020) 101170.
- [5] P. Ding, J. Zhang, P. Zheng, P. Zhang, B. Fei, Z. Xu, Dynamic scenario-enhanced diverse human motion prediction network for proactive human–robot collaboration in customized assembly tasks, *Journal of Intelligent Manufacturing* (2024) 1–20.
- [6] R. G. Lins, R. E. d. Santos, R. Gaspar, Vision-based measurement for quality control inspection in the context of industry 4.0: a comprehensive review and design challenges, *Journal of the Brazilian Society of Mechanical Sciences and Engineering* 45 (4) (2023) 229.
- [7] Z. Yuan, B. Xin, J. Zhang, Y. Xu, Three-dimensional fabric smoothness evaluation using point cloud data for enhanced quality control, *Journal of Intelligent Manufacturing* (2024) 1–17.
- [8] M. Rozmus, J. Tokarczyk, D. Michalak, M. Dudek, K. Szwedra, M. Rotkegel, A. Lamot, J. Roßer, Application of 3d scanning, computer simulations and virtual reality in the redesigning process of selected areas of underground transportation routes in coal mining industry, *Energies* 14 (9) (2021) 2589.
- [9] M. Akerdad, A. Aboutajeddine, M. Elmajdoubi, Development of an authentic concept of engineering activities based on product redesign, *Computer Applications in Engineering Education* 30 (3) (2022) 956–972.
- [10] M. U. Siddiqi, W. L. Ijomah, G. I. Dobie, M. Hafeez, S. Gareth Pierce, W. Ion, C. Mineo, C. N. MacLeod, Low cost three-dimensional virtual model construction for remanufacturing industry, *Journal of Remanufacturing* 9 (2019) 129–139.
- [11] Y. Zheng, J. Liu, Z. Liu, T. Wang, R. Ahmad, A primitive-based 3d reconstruction method for remanufacturing, *The International Journal of Advanced Manufacturing Technology* 103 (2019) 3667–3681.
- [12] T. H. Kwok, T. Gaasenbeek, Dynamic computer-aided process control with computer vision for industry 4.0, in: *International Conference on Flexible Automation and Intelligent Manufacturing*, Springer, 2022, pp. 510–518.
- [13] J. Wang, Y. Liang, Generation and detection of structured light: a review, *Frontiers in Physics* 9 (2021) 688284.
- [14] L. Liu, Y. Liao, Y. Wang, A. Geiger, Y. Liu, Learning steering kernels for guided depth completion, *IEEE Transactions on Image Processing* 30 (2021) 2850–2861.
- [15] H. Wang, M. Wang, Z. Che, Z. Xu, X. Qiao, M. Qi, F. Feng, J. Tang, Rgb-depth fusion gan for indoor depth completion, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6209–6218.
- [16] X. Liu, X. Shao, B. Wang, Y. Li, S. Wang, Graphcspn: Geometry-aware depth completion via dynamic gcns, in: *European Conference on Computer Vision*, Springer, 2022, pp. 90–107.
- [17] R. Fan, Z. Li, M. Poggi, S. Mattoccia, A cascade dense connection fusion network for depth completion, in: *The 33rd British Machine Vision Conference*, Vol. 1, 2022, p. 2.
- [18] B. Liu, C.-T. Lam, B. K. Ng, X. Yuan, S. K. Im, A graph-based framework for traffic forecasting and congestion detection using online images from multiple cameras, *IEEE Access* 12 (2024) 3756–3767. doi:10.1109/ACCESS.2023.3349034.
- [19] K.-H. Chan, G. Pau, S.-K. Im, Chebyshev pooling: An alternative layer for the pooling of cnns-based classifier, in: *2021 IEEE 4th International Conference on Computer and Communication Engineering Technology (CCET)*, 2021, pp. 106–110. doi:10.1109/CCET52649.2021.9544405.
- [20] C. Xian, D. Zhang, C. Dai, C. C. Wang, Fast generation of high-fidelity rgb-d images by deep learning with adaptive convolution, *IEEE Transactions on Automation Science and Engineering* 18 (3) (2020) 1328–1340.

- [21] D. Senushkin, M. Romanov, I. Belikov, N. Patakin, A. Konushin, Decoder modulation for indoor depth completion, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2021, pp. 2181–2188.
- [22] Z. Yan, K. Wang, X. Li, Z. Zhang, J. Li, J. Yang, Rignet: Repetitive image guided network for depth completion, in: European Conference on Computer Vision, Springer, 2022, pp. 214–230.
- [23] Y. Zhang, T. Funkhouser, Deep depth completion of a single rgb-d image, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 175–185.
- [24] Y.-K. Huang, T.-H. Wu, Y.-C. Liu, W. H. Hsu, Indoor depth completion with boundary consistency and self-attention, in: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, 2019, pp. 0–0.
- [25] D. Ren, M. Yang, J. Wu, N. Zheng, Surface normal and gaussian weight constraints for indoor depth structure completion, Pattern Recognition 138 (2023) 109362.
- [26] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, M. Nießner, Scannet: Richly-annotated 3d reconstructions of indoor scenes, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 5828–5839.
- [27] M. Di Cicco, L. Iocchi, G. Grisetti, Non-parametric calibration for depth sensors, Robotics and Autonomous Systems 74 (2015) 309–317.
- [28] A. Teichman, S. Miller, S. Thrun, Unsupervised intrinsic calibration of depth sensors via slam., in: Robotics: Science and systems, Vol. 248, Citeseer, 2013, p. 3.
- [29] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, C. Theobalt, Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration, ACM Transactions on Graphics (ToG) 36 (4) (2017) 1.
- [30] M. Nießner, M. Zollhöfer, S. Izadi, M. Stamminger, Real-time 3d reconstruction at scale using voxel hashing, ACM Transactions on Graphics (ToG) 32 (6) (2013) 1–11.
- [31] J. Park, K. Joo, Z. Hu, C.-K. Liu, I. So Kweon, Non-local spatial propagation network for depth completion, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16, Springer, 2020, pp. 120–136.
- [32] X. Qi, Z. Liu, R. Liao, P. H. Torr, R. Urtasun, J. Jia, Geonet++: Iterative geometric neural network with edge-aware refinement for joint depth and surface normal estimation, IEEE Transactions on Pattern Analysis and Machine Intelligence 44 (2) (2020) 969–984.
- [33] W. Xie, Y. Zhang, C. C. Wang, R. C.-K. Chung, Surface-from-gradients: An approach based on discrete geometry processing, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2195–2202.
- [34] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, R. Ng, Nerf: Representing scenes as neural radiance fields for view synthesis, cite arxiv:2003.08934Comment: ECCV 2020 (oral). Project page with videos and code: <http://tancik.com/nerf> (2020). URL <http://arxiv.org/abs/2003.08934>
- [35] B. Kerbl, G. Kopanas, T. Leimkühler, G. Drettakis, 3d gaussian splatting for real-time radiance field rendering., ACM Trans. Graph. 42 (4) (2023) 139:1–139:14. URL <http://dblp.uni-trier.de/db/journals/tog/tog42.html#KerblKLD23>
- [36] Y. Zhang, X. Guo, M. Poggi, Z. Zhu, G. Huang, S. Mattoccia, Completionformer: Depth completion with convolutions and vision transformers, in: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023, pp. 18527–18536. doi:10.1109/CVPR52729.2023.01777.
- [37] Y. Lee, S. Park, B. Kang, H. Park, Confidence guided depth completion network, arXiv preprint arXiv:2202.03257 (2022).
- [38] A. Conti, M. Poggi, F. Aleotti, S. Mattoccia, Unsupervised confidence for lidar depth maps and applications, in: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2022, pp. 8352–8359.
- [39] H. Liu, X. Tang, S. Shen, Depth-map completion for large indoor scene reconstruction, Pattern Recognition 99 (2020) 107112.
- [40] W. Zhao, S. Liu, Y. Wei, H. Guo, Y.-J. Liu, A confidence-based iterative solver of depths and surface normals for deep multi-view stereo, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 6168–6177.
- [41] S. Liu, S. De Mello, J. Gu, G. Zhong, M.-H. Yang, J. Kautz, Learning affinity via spatial propagation networks, Advances in Neural Information Processing Systems 30 (2017).
- [42] X. Cheng, P. Wang, R. Yang, Depth estimation via affinity learned with convolutional spatial propagation network, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 103–119.
- [43] X. Cheng, P. Wang, C. Guan, R. Yang, Cspn++: Learning context and resource aware convolutional spatial propagation networks for depth completion, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 10615–10622.
- [44] Y. Lin, H. Yang, T. Cheng, W. Zhou, Z. Yin, Dyspn: Learning dynamic affinity for image-guided depth completion, IEEE Transactions on Circuits and Systems for Video Technology (2023).
- [45] I. Makarov, V. Aliev, O. Gerasimova, Semi-dense depth interpolation using deep convolutional neural networks, in: Proceedings of the 25th ACM international conference on Multimedia, 2017, pp. 1407–1415.
- [46] Y. Liao, L. Huang, Y. Wang, S. Kodagoda, Y. Yu, Y. Liu, Parse geometry from a line: Monocular depth estimation with partial laser observation, in: 2017 IEEE international conference on robotics and automation (ICRA), IEEE, 2017, pp. 5059–5066.
- [47] Z. Chen, V. Badrinarayanan, G. Drozdov, A. Rabinovich, Estimating depth from rgb and sparse sensing, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 167–182.
- [48] G. Hegde, T. Pharale, S. Jahagirdar, V. Nargund, R. A. Tabib, U. Mudenagudi, B. Vandrotti, A. Dhiman, Deepdnet: Deep dense network for depth completion task, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 2190–2199.
- [49] N. Silberman, D. Hoiem, P. Kohli, R. Fergus, Indoor segmentation and support inference from rgb-d images, in: Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7–13, 2012, Proceedings, Part V 12, Springer, 2012, pp. 746–760.
- [50] Y. Zuo, Q. Wu, P. An, J. Zhang, Explicit measurement on depth-color inconsistency for depth completion, in: 2016 IEEE International Conference on Image Processing (ICIP), IEEE, 2016, pp. 4037–4041.
- [51] R. Hartley, A. Zisserman, Multiple view geometry in computer vision, Cambridge university press, 2003.
- [52] S. Bouaziz, M. Deuss, Y. Schwartzburg, T. Weise, M. Pauly, Shape-up: Shaping discrete geometry with projections, in: Computer Graphics Forum, Vol. 31, Wiley Online Library, 2012, pp. 1657–1667.
- [53] X. Wang, D. Fouhey, A. Gupta, Designing deep networks for surface normal estimation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 539–547.
- [54] J. Huang, Y. Zhou, T. Funkhouser, L. J. Guibas, Framenet: Learning local canonical frames of 3d surfaces from a single rgb image, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 8638–8647.
- [55] R. Wang, D. Geraghty, K. Matzen, R. Szeliski, J.-M. Frahm, Vplnet: Deep single view normal estimation with vanishing points and lines, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 689–698.
- [56] Q. Li, J. Guo, Y. Fei, Q. Tang, W. Sun, J. Zeng, Y. Guo, Deep surface normal estimation on the 2-sphere with confidence guided semantic attention, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16, Springer, 2020, pp. 734–750.
- [57] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al., Segment anything, arXiv preprint arXiv:2304.02643 (2023).
- [58] X. Zou, J. Yang, H. Zhang, F. Li, L. Li, J. Gao, Y. J. Lee, Segment everything everywhere all at once, arXiv preprint arXiv:2304.06718 (2023).
- [59] G. Bae, I. Budvytis, R. Cipolla, Estimating and exploiting the aleatoric uncertainty in surface normal estimation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 13137–13146.
- [60] S. F. Bhat, I. Alhashim, P. Wonka, Adabins: Depth estimation using adaptive bins, in: Computer Vision and Pattern Recognition, 2021.
- [61] W. Xie, M. Wang, M. Wei, J. Jiang, J. Qin, Surface reconstruction from normals: A robust dgp-based discontinuity preservation approach, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 5328–5336.
- [62] P. Liepa, Filling holes in meshes, in: Proceedings of the 2003 Eurograph-

- ics/ACM SIGGRAPH symposium on Geometry processing, 2003, pp. 200–205.
- [63] V. Schomaker, J. Waser, R. t. Marsh, G. Bergman, To fit a plane or a line to a set of points by least squares, *Acta crystallographica* 12 (8) (1959) 600–604.
 - [64] Washington, L. University, Structure 3d sensor, <https://my.wlu.edu/iq-center/equipment/prototyping/scanners/structure-sensor> [Accessed: January-11, 2024] (2024).
 - [65] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, Y. Zhang, Matterport3d: Learning from rgb-d data in indoor environments, *arXiv preprint arXiv:1709.06158* (2017).
 - [66] M. Pu, Y. Huang, Q. Guan, H. Ling, Rindnet: Edge detection for discontinuity in reflectance, illumination, normal and depth, in: *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 6879–6888.
 - [67] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: from error visibility to structural similarity, *IEEE transactions on image processing* 13 (4) (2004) 600–612.
 - [68] D. Marr, *Vision: A computational investigation into the human representation and processing of visual information*, MIT press, 2010.