Coupling Point Cloud Completion and Surface Connectivity Relation Inference for 3D Modeling of Indoor Building Environments

Xiao, Y.; Taguchi, Y.; Kamat, V.R.

TR2018-096 July 13, 2018

Abstract

Due to occlusions and limited measurement ranges, three-dimensional (3D) sensors are often not able to obtain complete point clouds. Completing missing data and obtaining spatial relations of different building components in such incomplete point clouds are important for several applications, for example, 3D modeling for all objects in indoor building environments. This paper presents a framework that recovers missing points and estimates connectivity relations between planar and nonplanar surfaces to obtain complete and high-quality 3D models. Given multiple depth frames and their sensor poses, a truncated signed distance function (TSDF) octree is constructed to fuse the depth frames and estimate the visibility labels of octree voxels. A normalbased region growing method is utilized to detect planar and nonplanar surfaces from the octree point cloud. Based on the surfaces and the visibility labels, missing points are completed by estimating the connectivity relations between pairs of the surfaces and by filling individual planar surfaces. Experimental results demonstrate that the proposed method can correctly identify at least 78% of the connectivity relations between the detected surfaces, and 87% of added points are correct and help to generate high-quality 3D models compared to the ground truth model.

Journal of Computing in Civil Engineering

Coupling Point Cloud Completion and Surface Connectivity Relation Inference for 3D Modeling of Indoor Building Environments

Yong Xiao[1], Yuichi Taguchi[2], and Vineet R. Kamat[3]

ABSTRACT

Due to occlusions and limited measurement ranges, three-dimensional (3D) sensors are often not able to obtain complete point clouds. Completing missing data and obtaining spatial relations of different building components in such incomplete point clouds are important for several applications, for example, 3D modeling for all objects in indoor building environments. This paper presents a framework that recovers missing points and estimates connectivity relations between planar and nonplanar surfaces to obtain complete and high-quality 3D models. Given multiple depth frames and their sensor poses, a truncated signed distance function (TSDF) octree is constructed to fuse the depth frames and estimate the visibility labels of octree voxels. A normal-based region growing method is utilized to detect planar and nonplanar surfaces from the octree point cloud. Based on the surfaces and the visibility labels, missing points are completed by estimating the connectivity relations between pairs of the surfaces and by filling individual planar surfaces. Experimental results demonstrate that the proposed method can correctly identify at least 78% of the connectivity relations between the detected surfaces, and 87% of added points are correct and help to generate high-quality 3D models compared to the ground truth model.

Keywords

3D modeling; Point cloud completion; Surface connection; Depth camera.

[1]Ph.D. Candidate, Tishman Construction Management Program, Dept. of Civil and Environmental Engineering, Univ. of Michigan, 2350 Hayward St., Suite 1306 G.G. Brown Bldg., Ann Arbor, MI 48109 (corresponding author). E-mail: yongxiao@umich.edu
[2]Senior Principal Research Scientist, Mitsubishi Electric Research Laboratories, 201 Broadway, 8th Floor Cambridge, MA 02139. E-mail: taguchi@merl.com
[3]Professor, Tishman Construction Management Program, Dept. of Civil and Environmental Engineering, Univ. of Michigan, 2350 Hayward St., Suite 2105 G.G. Brown Bldg., Ann Arbor, MI 48109. E-mail: vkamat@umich.edu

Introduction

3D models, e.g., building information modelings (BIMs), contain rich geometric properties and spatial relations of various building entities (Tang et al. 2010) and can play an important role in different project stages, including design, construction, and maintenance phases (Azhar 2011; Hardin and McCool 2015). For facility maintenance purposes or indoor robotic applications, 3D models for existing buildings are necessary. Current BIMs generation methods mainly focus on obtaining models for primary civil infrastructures, e.g., walls, floors, windows, and pipes (Pătrăucean et al. 2015). For indoor building environments, 3D models for various furniture are neglected during the modeling process. However, 3D models including furniture as well as building elements (e.g., walls and floors) are significant for comprehensive building facility management or indoor robotic applications. Three-dimensional (3D) point clouds collected by various sensors, e.g., laser scanners (Giel and Issa 2012; Hajian and Becerik-Gerber 2010) and depth cameras (Zhu and Donia 2013; Arnaud et al. 2016), are used to create 3D models for indoor building environments.

When using the sensors to obtain 3D point clouds, due to object occlusions or sensor limitations, observed point clouds usually cover only some parts of scenes and miss some other parts. These point clouds will be referred to as incomplete point clouds in this paper. For example, when using the 3D sensors (e.g., laser scanners, and depth sensors) to obtain a point cloud of a typical classroom, any fixed furniture such as anchored tables or chairs may block each other or the building elements (e.g., walls and floors) such that the resulting point clouds do not contain the complete geometry of the building elements. Based on the incomplete point clouds, it is challenging to recover complete 3D object models or identify object labels.

In order to mitigate this problem, this paper presents a framework to recover missing points and infer connectivity relations between surfaces for creating complete 3D models in indoor environments. Our framework exploits the fact that indoor building environments are dominated by planar surfaces and that intersections of the planar surfaces provide a strong cue for completing missing data: if two planar surfaces are physically intersecting and connected, there is likely no gap between them and missing points between them can be filled. Thus the main process of our framework consists of extracting planar surfaces from the incomplete point cloud, estimating such connectivity relations between intersecting planar surfaces, and filling the missing points between the planar surfaces if the connectivity relations are found. For estimating the connectivity relations and filling the missing points, the framework uses the visibility information of points in 3D space, which is obtained by generating a truncated signed distance function (TSDF) octree (Steinbruecker et al. 2014) from the incomplete point cloud, such that we do not connect planar surfaces and fill missing points when there are free space measurements between them. To obtain more comprehensive connectivity relations and fill more missing points, our framework also includes additional processes such as (1) estimating connectivity relations between parallel planar surfaces located close to each other; (2) extracting nonplanar surfaces and connecting each of them to a planar surface that supports it by filling the gap between them; and (3) filling missing points within individual planar surfaces.

The rest of the paper is organized as follows: Section *Previous Work* reviews related work on completing point clouds and estimating spatial characteristics for 3D reconstruction and modeling. Section *Methodology* introduces the proposed method in detail. The experimental results and discussion on real-world and synthetic datasets are presented in Section *Experimental Results and Discussion*. Section *Conclusions and Future Work* draws conclusions of the paper as well as discusses its limitations and future work.

Previous Work

Our work involves point cloud completion and spatial relation inference, each of which is discussed separately in this section.

Point Cloud Completion

To complete point clouds of surfaces, a common approach is to apply interpolation using geometric properties (e.g., symmetry and smoothness) of the surfaces. Janaszewski et al. (2010) filled holes by extracting the Euclidean skeleton and closing holes in the skeleton using a modified hole closing algorithm and thickness of objects. Kroemer et al. (2012) utilized planar reflection symmetries to detect extruded shapes and then employed the parametric representation of the extruded shapes to complete the point cloud. Wang and Oliveira (2007) used moving least squares to interpolate both geometry and shading information to fill holes. Sharf et al. (2004) estimated the characteristics of the surfaces and filled holes by copying the best matching patches from valid regions. Carr et al. (2001) utilized radial basis functions to reconstruct smooth surfaces and complete holes by interpolation. When reconstructing 3D models with known parametric representations, e.g., cylindrical objects (Ahmed et al. 2014; Son et al. 2015), identifying the exact parameters also helps complete missing point clouds. The parameters are used to infer unobserved points on its surface and thus obtain 3D complete models. Li et al. (2011) proposed a method to simultaneously fit primitives and recover their global mutual relations from noisy and incomplete point sets. By estimating the global relations and shape alignments, complete models are constructed. Chauve et al. (2010) presented a piecewise-planar 3D reconstruction and completion method from point clouds with noise and outliers. They added ghost primitives composed of planar primitives to ensure the continuation of detected primitives and the prevalence of vertical structures. Xiong et al. (2013) utilized a ray-tracing method to detect occluded regions of the walls and filled them using a 3D inpainting algorithm. Silberman et al. (2014) utilized a probabilistic model, Contour

Completion Random Field, to complete the boundaries and then complete the internal planar surfaces using the boundaries. The proposed method quantitatively and qualitatively outperforms standard methods. This method relies on the accuracy of plane segmentation and the boundaries of the planar surfaces.

Another type of methods for completing point clouds is to reconstruct the models from partial point clouds by referring to existing 3D object model libraries. Kim et al. (2012) first acquired 3D models of common objects and their variability models and then recognized these objects from a single scan. Sung et al. (2015) collected examples of 3D shapes to build structural part-based priors and learned the distribution of positions and orientations of each part of the shapes. When processing incomplete point clouds, they estimated the parts and symmetries of the data and fused data source, symmetry, and database to reconstruct 3D complete models. Nan et al. (2012) trained a classifier on a set of shape features and performed the segmentation and classification simultaneously. The 3D completion models are obtained by a template deform-to-fit reconstruction method. Song and Xiao (2014) created a collection of 3D Computer-Aided Design (CAD) models, rendered each model from different viewpoints to get synthetic depth maps, and then trained a support vector machine (SVM) classifier for each depth rendering. A 3D detection window was used to detect objects and construct the 3D complete models. Shao et al. (2012) developed an interactive approach to generate better segmentation results and then replaced the segments with objects from a 3D model database to get semantic models of indoor scenes. Song et al. (2016) proposed a semantic scene completion network to estimate the occupancy and semantic labels for all voxels in the camera view frustum for a single depth image. The experiments demonstrated better results compared to methods for semantic scene completion. However, this approach relies on comprehensive training dataset so as to predict the correct semantic label for objects as well as the completion.

Spatial Relations Inference

Apart from geometric property of objects, 3D models also need spatial relations (or topological relations) of building components to facilitate complicated analysis and decision making, e.g., building object classification (Brilakis et al. 2010). Apart from merely representing simple spatial information (e.g., connection, adjacency, and intersection), spatial relations can also depict or be used to infer physical relations, which helps object detection and scene understanding. Existing BIMs can be directly employed to estimate spatial relations between 3D objects for spatial queries or analysis. Nguyen et al. (2005) proposed algorithms to automatically estimate topological information of building components from 3D CAD models. Based on the boundary representation of 3D objects, the following topological relations were computed: adjacency, separation, containment, intersection, and connectivity. Nepal et al. (2008) analyzed topological relations to derive construction features from a BIM model using the Industry Foundation Classes. Borrmann and Rank (2009) extracted directional relations (e.g., above, below, and north of) between 3D spatial objects for BIMs. Daum and Borrmann (2014) estimated topological relations for spatial queries based on a novel boundary representation of 3D models for BIMs. These methods rely on an existing BIMs as well as specific representation of models to efficiently compute topological relations of building entities.

Instead of using existing 3D models, spatial relations are also estimated when generating 3D models. Silberman et al. (2012) presented a supervised framework to segment visible regions and infer their support relations by utilizing physical constraints and statistical priors on support. This method processed single RGB-D images individually instead of processing registered point clouds. Shao et al. (2014) extrapolated the cuboids around objects to recover the geometric attributes and their spatial relations by making the cuboids physically stable. This method aimed to recover the support relations and thus provided cues for retrieving models from 3D model libraries. Zheng et al. (2013) estimated the geometric primitives by segmenting the point clouds and completing the volumetric space. The completion mainly utilized the occlusion information and the Manhattan assumption. After the completion and segmentation, they used Swendsen-Wang Cut (Barbu and Zhu 2005) to optimize the stability of surfaces.

Different from the aforementioned previous work, this paper explores to complete the missing points and recover the spatial relations (especially connections between surfaces) simultaneously from a registered point cloud. Considering the noisy data, the proposed method couples the surface detection process with the point cloud completion and connectivity relation inference, so that the connectivity relations and surface completion are performed robustly. In addition, based on the assumption that planar surfaces are dominant structures in indoor environments, the modeling process in our method starts from major planar surfaces and then handles iteratively using small planar surfaces and nonplanar surfaces, which generates 3D complete point clouds for detected surfaces and surface connectivity relations.

Methodology

Fig. 1 shows how the proposed method simultaneously completes a point cloud and recovers the connectivity relations of surfaces from multiple RGB-D frames. The input to the method is a series of organized point clouds (depth maps) registered with each other by a simultaneous localization and mapping (SLAM) system. A truncated signed distance function (TSDF) octree is created to label the visibility of each octree voxel using the observed point clouds and sensor poses. From the octree point cloud, a normal-based region growing algorithm is first employed to extracted major planar surfaces. The system tries to create connections between the planar surfaces by filling the gap between surfaces if necessary. A connectivity graph $\boldsymbol{G}$ is created using the planar surfaces where each node denotes a surface while an edge represents the connection between two surfaces. Then, the normal-based region growing algorithm is utilized to extract small planar surfaces and nonplanar surfaces from the remaining point cloud. These new detected surfaces are utilized to update $\boldsymbol{G}$ by estimating the connections between them and the surfaces in $\boldsymbol{G}$. Therefore, $\boldsymbol{G}$ is updated by three different types of surface sets, i.e., major planar surfaces, small planar surfaces, and nonplanar surfaces, and the surface completion and connectivity inference methods depend on the surface type. The surface detection is iteratively performed until no surface is detected. Finally, 3D complete point clouds for detected surfaces and the connectivity relations of surfaces are reconstructed.

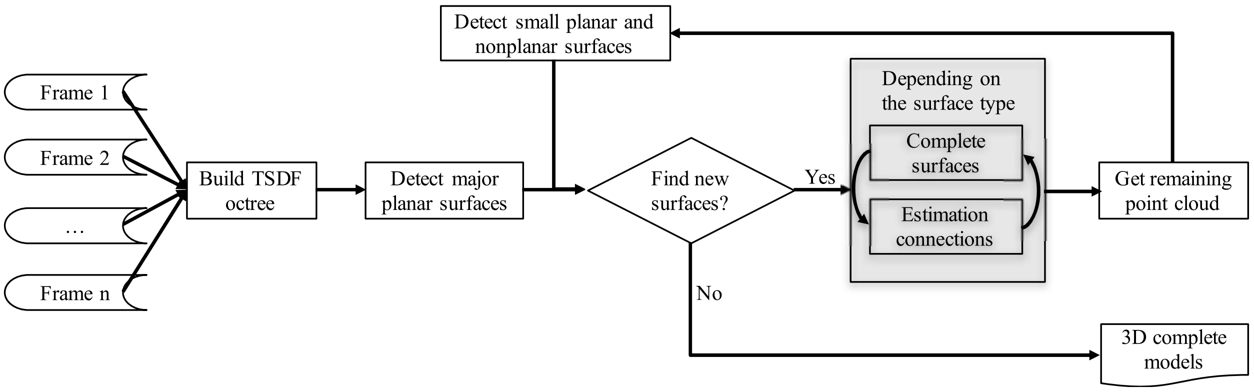

Detect small planar and nonplanar surfaces
Frame 1
Frame 2
…
Frame n
Build TSDF octree
Detect major planar surfaces
Find new surfaces?
Yes
No
Depending on the surface type
Complete surfaces
Estimation connections
Get remaining point cloud
3D complete models

Fig. 1. System framework.

TSDF Octree Construction

The registered organized point clouds are fused by using a truncated signed distance function (TSDF) octree (Curless and Levoy 1996; Newcombe et al. 2011; Steinbruecker et al. 2014) to reduce the measurement noise and to obtain a single fused point cloud $\boldsymbol{P_{oct}}$ with fewer points compared to the raw point cloud. The TSDF octree representation can efficiently handle large-scale scenes while incorporating uncertainty information of observed points. The TSDF octree generation is performed using the organized point cloud frames and the sensor poses (positions and orientations) computed by a SLAM system which registers all frames to the same coordinate system.

For each point in a frame of the organized point cloud, a ray from the sensor position to the point is cast to the TSDF volume. Then the TSDF values of octree voxels (that are near the observed point) on the ray according to the depth measurement of the point are calculated for the first time or updated if they are computed using other points. The octree is incrementally expanded to cover all the measured points when the depth measurement falls into an uninitialized region. The octree generation iterates for all the points in all the frames.

After processing all the frames, the TSDF value of an octree voxel reflects the distance between the voxel and its nearest surface point. The TSDF value is close to zero for a measured point, while the TSDF value is positive and negative for a point in front of and behind a measured point, respectively. The visibility of an octree voxel is then determined according to its TSDF value. The voxels with zero TSDF values are categorized as occupied voxels, which form the single fused point cloud $\boldsymbol{P_{oct}}$. The voxels with positive and negative distance values respectively correspond to free space voxels (free space between the sensor and occupied voxels) and invisible voxels (occluded behind occupied voxels). In addition to these visibility labels, the voxels will be assigned a surface identification during the completion process.

Surface detection

Instead of detecting surfaces from the point cloud in advance for later processes as previous methods (Zheng et al. 2013), in this paper, the surface detection is coupled with estimating the connectivity relations and completing the point cloud. Since the point clouds contain noises due to the measurement noises or registration errors of multiple frames, it is challenging to attain the optimal and general threshold for both planar and nonplanar surface detection (e.g., the distance for a point belonging to a surface). For example, when detecting planes for the noisy point clouds, the surface detection method tends to find multiple planar clusters for the plane containing large noises. Therefore, in this paper, the surface detection contains two separate steps, (1) major planar surface detection, and (2) small planar surface and nonplanar surface detection. Since most indoor objects contain planes, the proposed method processes the major planar surfaces before handling small planar and nonplanar surfaces. After the connection estimation and completion for the major planar surfaces, the small planar surface and nonplanar surface detection is iteratively performed on the remaining point cloud and the detected surfaces are processed for point cloud completion and surface relation inference.

A normal-based region growing algorithm (Xiao et al. 2014) is utilized to detect major planar surfaces. The normal vectors and curvatures of the points are estimated by performing principal component analysis of neighboring points. In order to find a planar cluster, the point with the smallest curvature is selected as the initial seed point from the points that are not classified to any cluster. Starting with this seed point $\boldsymbol{p_s}$ whose normal vector is $\boldsymbol{n_{p_s}}$, for each point in its neighborhood, $\boldsymbol{p} \in \mathbb{N}_{\boldsymbol{p_s}}$, if the difference between its normal vector and $\boldsymbol{n_{p_s}}$, is smaller than a threshold, $\boldsymbol{p}$ is assigned to the cluster $\boldsymbol{C_{p_s}}$ containing $\boldsymbol{p_s}$ and used as a new seed point. This process is iteratively performed until no point is added to $\boldsymbol{C_{p_s}}$ and all seed points are explored. Then another qualified seed point, i.e., it has the smallest curvature among the remaining unassigned points while the curvature is smaller than the curvature threshold, is selected and the iterative growing process is performed again to find another cluster. The method stops until no qualified seed point is available or no cluster meeting the requirements (in this work, a cluster has to contain a minimum number of points) is found using the growing strategy. A small curvature threshold and a small normal vector discrepancy threshold are utilized to extract planes with high confidence. Based on these major planar surfaces, the connectivity graph $\boldsymbol{G}$ is constructed.

After the connectivity graph is created and updated by the detected major planar surfaces, for the remaining point cloud, the normal-based region growing algorithm is adopted to identify nonplanar surfaces and small planar surfaces by relaxing the thresholds for the curvature and the normal vector difference. Due to noise and irregular objects, some spurious clusters whose points scatter widely in 3D space may be obtained. To eliminate them, the point density of the cluster, i.e., the ratio of the number of points to the volume of its bounding box, is checked. The valid nonplanar surfaces and small planar surfaces are utilized to update the connectivity graph by finding connections between them and the planar surfaces in $\boldsymbol{G}$.

Since the algorithm of updating $\boldsymbol{G}$ by a small planar surface is different from that using a nonplanar surface, this work utilizes the cluster point distribution to distinguish a nonplanar surface from a small planar surface. The principal component analysis (PCA) is performed on the cluster points and the eigenvalues $\lambda_0, \lambda_1, \lambda_2$ ($\lambda_0 \leq \lambda_1 \leq \lambda_2$) and eigenvectors of the covariance matrix of the points are computed. The value $p \leftarrow 1 - \lambda_0/\lambda_1$ can reflect whether these points are from a plane. For a perfect plane, p is 1 because the points have zero variance along the normal vector (which is the same as the eigenvector corresponding to λ_0) of the plane, and thus λ_0 is 0. For the points on a sphere, the three eigenvalues are identical and p is 0. If p is greater than a threshold (in this work, 0.9), the surface is viewed as a planar surface and used to update $\boldsymbol{G}$ using the corresponding method. Otherwise, it is processed as a nonplanar surface to update $\boldsymbol{G}$.

Connectivity Graph Construction

To represent connectivity relations between surfaces, an undirected graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ is constructed where the set of nodes $\boldsymbol{V}$ denotes surfaces segmented from the point cloud, and the set of edges $\boldsymbol{E}$ represents connections between nodes. If there exists a connection between two surfaces $\boldsymbol{v_i}$ and $\boldsymbol{v_j}$, an edge $\boldsymbol{e_{ij}}$ is added to $\boldsymbol{G}$. Since the planar surfaces are the major components of objects in indoor environment, the paper first utilizes the major planar surfaces to construct $\boldsymbol{G}$ by recovering the connections among them. Then, $\boldsymbol{G}$ is updated using small planar surfaces and nonplanar surfaces by finding connections between them and the planar surfaces in $\boldsymbol{G}$.

Connection Inference and Point Completion for Planar Surfaces

Algorithm 1 describes the method of updating $\boldsymbol{G}$ based on the major planar surface set $\boldsymbol{S}$. In indoor environments, large planar surfaces (measured by the number of points in the observed point cloud) usually dominate the main structures of a scene and play an important role in surface connections within the scene. Thus, Algorithm 1 handles planar surface according to their sizes so as to recover the connections between larger planar surfaces before processing small planar surfaces. The algorithm contains two sub-processes where the first one finds connections between $\boldsymbol{S}$ and $\boldsymbol{G}$ while the other seeks connections within $\boldsymbol{S}$ and then adds them to $\boldsymbol{G}$. When building a connection between two surfaces, some points are added to the two surfaces and the completion can lead to changes of distances between surfaces, which improves the possibility of building more connections. Since $\boldsymbol{G}$ already contains surface connections of major planar surfaces and partially filled surfaces during the building connection process, connections between $\boldsymbol{S}$ and $\boldsymbol{G}$ have higher confidence than those within $\boldsymbol{S}$. Therefore, Algorithm 1 first searches connections between $\boldsymbol{S}$ and $\boldsymbol{G}$ and then estimates connections within $\boldsymbol{S}$.

As shown in Algorithm 1 Line 5, first, for each surface $\boldsymbol{s} \in \boldsymbol{S}$, its candidate connected surfaces $\boldsymbol{S_{cs}}$ are searched from the nodes in $\boldsymbol{V}$ by checking the spatial relations of surfaces. If the distance between two surfaces $\boldsymbol{s}$ and $\boldsymbol{s'} \in \boldsymbol{V}$ is smaller than a distance threshold ($15v_s$ where v_s is the octree voxel size) and their planes intersect with each other, there might exist a connection between the two surfaces. Thus $\boldsymbol{s}'$ is a candidate connected surface for $\boldsymbol{s}$, i.e. $\boldsymbol{S_{cs}} \leftarrow \boldsymbol{S_{cs}} \cup \{\boldsymbol{s}'\}$. In this paper, the distance between two different (planar or nonplanar) surfaces, $\boldsymbol{s_1}$ and $\boldsymbol{s_2}$ is computed as the distance between the closest pair of points $(\boldsymbol{p_1}, \boldsymbol{p_2})$ while the two points are from different surfaces, i.e., $\boldsymbol{p_1} \in \boldsymbol{s_1}, \boldsymbol{p_2} \in \boldsymbol{s_2}$.

Algorithm 1 Update Connectivity Graph by the Major Planar Surface Set

```
1   function UpdateGraphByMajorPlaneSet($\boldsymbol{G}, \boldsymbol{S_P}$)
2     $\boldsymbol{S_P} \leftarrow$ SortPlanesBySize($\boldsymbol{S_P}$)
3     do
4       for $\boldsymbol{s} \in \boldsymbol{S_P}$ do
5         $\boldsymbol{S_{cs}} \leftarrow$ FindCandidateConnectedPlanes($\boldsymbol{s}, \boldsymbol{V}$)
6         for $\boldsymbol{s_{cs}} \in \boldsymbol{S_{cs}}$ do
7           if ThereIsAConnection($\boldsymbol{s}, \boldsymbol{s_{cs}}$) = true then
8             $\boldsymbol{S_P} \leftarrow \boldsymbol{S_P} \backslash \{\boldsymbol{s}\}$
9             $\boldsymbol{V} \leftarrow \boldsymbol{V} \cup \{\boldsymbol{s}\}$
10            $\boldsymbol{E} \leftarrow \boldsymbol{E} \cup \{\boldsymbol{e}(\boldsymbol{s}, \boldsymbol{s_{CS}})\}$
11          end if
12        end for
13      end for
14      for $\boldsymbol{s_1} \in \boldsymbol{S_P}, \boldsymbol{s_2} \in \boldsymbol{S_P}, \boldsymbol{s_1} \neq \boldsymbol{s_2}$ do
15        if ThereIsAConnection($\boldsymbol{s_1}, \boldsymbol{s_2}$) = true then
16          $\boldsymbol{S_P} \leftarrow \boldsymbol{S_P} \backslash \{\boldsymbol{s_1}, \boldsymbol{s_2}\}$
17          $\boldsymbol{V} \leftarrow \{\boldsymbol{s_1}, \boldsymbol{s_2}\} \cup \boldsymbol{V}$
18          $\boldsymbol{E} \leftarrow \boldsymbol{E} \cup \{\boldsymbol{e}(\boldsymbol{s_1}, \boldsymbol{s_2})\}$
19        end if
20      end for
21    while IsChanged($\boldsymbol{G}$) = true
22    return $\boldsymbol{G}$
23  end function
```

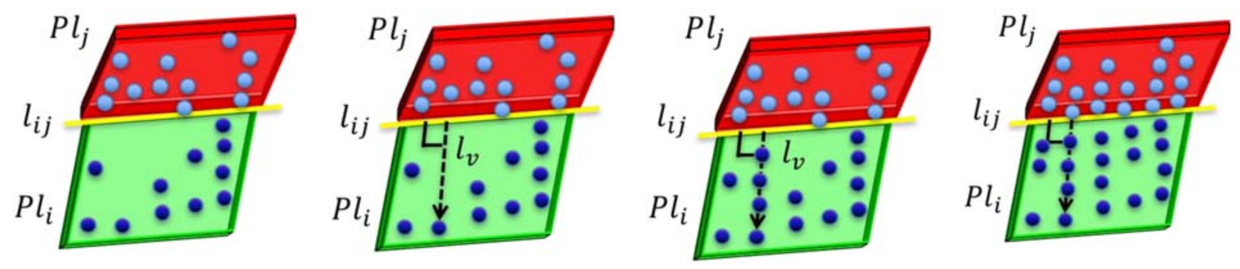
Once candidate connected planar surfaces $\boldsymbol{S_{cs}}$ are found, for each candidate surface $\boldsymbol{s_{cs}} \in \boldsymbol{S_{cs}}$ the algorithm will check whether there is a valid connection between $\boldsymbol{s_{cs}}$ and $\boldsymbol{s}$ (Algorithm 1 Line 7). The validity of a connection is related to the type of connections being estimated. This paper estimates two connections for planar surfaces, i.e., the connection between two intersecting planar surfaces and the connection between parallel but not coplanar planar surfaces. The validity of the connections will be discussed later. If a valid connection is detected, the edge is constructed between the two surfaces and added to $\boldsymbol{G}$ (Algorithm 1 Line 10). Meanwhile, the surface $\boldsymbol{s}$ is moved from $\boldsymbol{S}$ to $\boldsymbol{G}$ (Algorithm 1 Line 8-9). After trying to connect $\boldsymbol{S}$ to $\boldsymbol{G}$, as shown in Lines 14-20, Algorithm 1 detects connections within $\boldsymbol{S}$. If a connection between two planar surfaces, $\boldsymbol{s_1}$ and $\boldsymbol{s_2}$, is found, $\boldsymbol{s_1}$ and $\boldsymbol{s_2}$ are added to $\boldsymbol{V}$ while the edge between them is added to $\boldsymbol{E}$.

Algorithm 1 iteratively performs the two sub-processes until $\boldsymbol{G}$ is not updated, i.e., no surface is added to $\boldsymbol{G}$ and no more connection is detected.

Algorithm 1 is designed for the major planar surface set detected at the first stage the surface detection method. For the remaining planar surfaces (usually having a small number of points), this paper only tries to build a connection between them and the surfaces in $\boldsymbol{G}$, which is the same as Algorithm 1 Lines 4-13.

1) Connection Inference for Intersecting Planar Surfaces

The completion between two intersecting planar surfaces is performed by growing the two planar surfaces toward the intersection line. The intersection line $\boldsymbol{l_{ij}}$ (Fig. 2(a)) of two planar surfaces $\boldsymbol{Pl_i}$ and $\boldsymbol{Pl_j}$ is estimated using the plane equations. Then as shown in Fig. 2(b), for each plane, e.g., $\boldsymbol{Pl_i}$ starting from a voxel $\boldsymbol{v}$ on $\boldsymbol{l_{ij}}$, a segment $\boldsymbol{l_v}$ orthogonal with $\boldsymbol{l_{ij}}$ is drawn toward the centroid of the surface until it hits a point $\boldsymbol{v_{Pl_i}}$ on that surface.



(a) Compute the intersection line $\boldsymbol{l_{ij}}$ (b) Draw a segment $\boldsymbol{l_v}$ perpendicular to $\boldsymbol{l_{ij}}$ (c) Fill the voxels on $\boldsymbol{l_v}$ (d) Iterate all the points on $\boldsymbol{l_{ij}}$

Fig. 2. Completion between two intersecting planar surfaces

If all voxels on the segment $\boldsymbol{l_v}$ are unassigned to any surfaces or invisible, these voxels will be filled with points and added to the plane $\boldsymbol{Pl_i}$ (Fig. 2(c)). Otherwise, no voxels on $\boldsymbol{l_v}$ will be added to the octree. Thus, if $\boldsymbol{l_v}$ contains at least one free space voxel, none of these voxels are added since they have a high probability of being free space too. The process is iterated at all voxels on $\boldsymbol{l_{ij}}$. This completion process will fill the invisible or unassigned voxels around the intersection line of the two planar surfaces. This completion is temporarily performed first and finalized when the connection is valid. When the connection is valid, all the added points are maintained in the TSDF octree permanently.

Whether the connection between two intersecting planar surfaces is valid depends on the quality of the intersection segment between them. The intersection segment $\boldsymbol{seg_{ij}}$ is a segment on the intersection line $\boldsymbol{l_{ij}}$ between two surfaces and contains at least a certain number of points from both planar surfaces. If one of the planar surfaces have few points on $\boldsymbol{l_{ij}}$, there exists no intersection segment between the two planar surfaces. In order to compute the intersection segment, a segment $\boldsymbol{seg_i}$ is estimated from the points that are in $\boldsymbol{Pl_i}$ and also on $\boldsymbol{l_{ij}}$. Similarly, $\boldsymbol{seg_j}$ is computed. Finally, the intersection segment $\boldsymbol{seg_{ij}}$ is estimated as the intersection of $\boldsymbol{seg_i}$ and $\boldsymbol{seg_j}$, i.e.,

$\boldsymbol{seg_{ij}} \leftarrow \boldsymbol{seg_i} \cap \boldsymbol{seg_j}$.

If there exists an intersection segment $\boldsymbol{seg_{ij}}$ between two planar surfaces, whether the connection between them is created depends on the length and the point density of $\boldsymbol{seg_{ij}}$. In this paper, if the segment length is greater than $5v_s$ and the ratio of the number of points it contains to its length is greater than $0.9/v_s$, the intersection segment is good and the connection becomes valid. If a valid connection is detected, the connection is constructed between the two surfaces and the added points will be permanently assigned to the surfaces as well as the octree.

2) Connection inference for parallel planes

Other than intersecting connections, there exist connectivity relations between two parallel planar surfaces in the real world, e.g. a book lying on the table, and a television hanging on the wall. To estimate this connectivity relation, the two planar surfaces and their parallel relation (In this paper, we assume that for plane relations, coplanar planes have the same parameters and parallel planes only share the same normal vector.) should be correctly identified. Instead of simply utilizing a distance threshold between two parallel planes, this work also integrates the uncertainty of the planar surface to determine the parallel relation. If two planar surfaces are identified as parallel and can be connected, a connection between them is created and added to the graph $\boldsymbol{G}$.

In this work, to decide whether two parallel planar surfaces, $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$ are connected to each other, their normal vectors should be parallel, i.e., the minimum angle between them is smaller than an angle threshold (10 degrees), and the distance between the planar surfaces is smaller than a threshold ($2.5v_s$). Then, all the points of the two planar surfaces are projected to a plane parallel to the planar surfaces and two corresponding 2D convex hulls, $\boldsymbol{ch_1}$ and $\boldsymbol{ch_2}$ for the projected points are computed. The overlapping value is computed as the ratio of the points of the smaller planar surface (for example, $\boldsymbol{P_1}$) falling within $\boldsymbol{ch_2}$. If the ratio is smaller than a threshold (20%

in this paper), there is no connection between $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$, and they are just parallel to each other. Otherwise, the distance $d(\boldsymbol{c_1}, \boldsymbol{P_2})$ from the centroid of the smaller surface $\boldsymbol{P_1}$ to $\boldsymbol{P_2}$ is computed. If the distance is greater than $\max(0.25(thick_{CBD_1} + thick_{CBD_2}), d_{thresh_p})$ (in this work, $d_{thresh_p} = 3v_s$) where $thick_{CBD_1}$ and $thick_{CBD_2}$ represent the thickness of the cuboids of $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$ (which will be explained in Section *Completion within Individual Planar Surfaces*), there is a connection between $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$. If the overlapping value is greater than the threshold and $d(\boldsymbol{c_1}, \boldsymbol{P_2})$ is smaller than the threshold, $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$ are viewed as coplanar planes and will be merged into one planar surface.

Connection Inference and Point Completion for Nonplanar Surfaces

After the major planar surfaces are processed, nonplanar surfaces are utilized to update the connectivity graph $\boldsymbol{G}$. By assuming that a nonplanar surface is supported by a planar surface, this paper aims to connect a nonplanar surface to at most one planar surface and build a connection between them. The candidate connected planar surfaces for a nonplanar surface are determined according to the distances between surfaces.

To find the best candidate, the method computes the weights for the candidate planar surfaces based on the gravity direction, the surface size, and the distances between surfaces. The gravity direction is employed to obtain physically reasonable connections. This paper assumes that when capturing the first frame, the sensor is held almost horizontally and all the other frames are registered to the first frame. Therefore, the gravity direction is set using this prior knowledge according to the sensor coordinate system. Meanwhile, the surface size is utilized to ensure that the connection creation prefers large planar surfaces than small planar surfaces. The distance between surfaces is also considered to favor closer surfaces. Let $\omega_{\boldsymbol{NP_i}}(\boldsymbol{P_j})$ denote the weight of a planar surface $\boldsymbol{P_j}$ with respect to a nonplanar surface $\boldsymbol{NP_i}$. Then,

$$\omega_{\boldsymbol{NP}_i}(\boldsymbol{P_j}) = a_1 \cdot f(\boldsymbol{g}, \boldsymbol{n}_{\boldsymbol{P}_i}) + a_2 \cdot g(|\boldsymbol{P_j}|) + a_3 \cdot h\big(d(\boldsymbol{NP_i}, \boldsymbol{P_j}), dT_0\big) \qquad (1)$$

where a_1, a_2, a_3 are weight coefficients (in the paper, a_1, a_2, a_3 are 0.4, 0.3 and 0.3, respectively). The first term$f(\boldsymbol{g}, \boldsymbol{n}_{\boldsymbol{P}_i}) = 1 - \angle(\boldsymbol{g}, \boldsymbol{n}_{\boldsymbol{P}_i})/\pi$ is a function of the gravity $\boldsymbol{g}$ and the normal vector of $\boldsymbol{P_j}$, $\boldsymbol{n}_{\boldsymbol{P}_i}$ where $\angle(\boldsymbol{g}, \boldsymbol{n}_{\boldsymbol{P}_i})$ is the minimum angle between $\boldsymbol{g}$ and $\boldsymbol{n}_{\boldsymbol{P}_i}$. The second term $g(|\boldsymbol{P_j}|)$ is a function of the planar surface size and favors large surfaces than small surfaces. In this work, if $|\boldsymbol{P_j}| > T_s$, $g(|\boldsymbol{P_j}|) = 0.7$. Otherwise, $g(|\boldsymbol{P_j}|) = 0.3$. As the octree is utilized, T_s can represent the number of voxels of an object model. In this paper, T_s is set as the number of voxels of a square planar surface with a side length of $50v_s$, i.e. $T_s \leftarrow 2{,}500$. The third term $h\big(d(\boldsymbol{NP_i}, \boldsymbol{P_j}), dT\big)$ is defined as

$$h\big(d(\boldsymbol{NP_i}, \boldsymbol{P_j}), dT\big) = \begin{cases} 1 - \frac{d(\boldsymbol{NP_i}, \boldsymbol{P_j})}{dT} & d(\boldsymbol{NP_i}, \boldsymbol{P_j}) \leq dT \\ -\infty & d(\boldsymbol{NP_i}, \boldsymbol{P_j}) > dT \end{cases} \qquad (2)$$

It is a function of the distance between $\boldsymbol{P_j}$ and $\boldsymbol{NP_i}$, and a distance threshold dT (in this work, $dT \leftarrow 15v_s$) which is utilized to ignore those surfaces that are too far away from $\boldsymbol{NP_i}$. When $d(\boldsymbol{NP_i}, \boldsymbol{P_j})$ is too large (greater than dT), the weight is negative infinite and a connection between them will not be created.

The candidate planar surfaces are sorted by the weights. Starting from the planar surface with the largest weight, this paper tries to construct a connection between the planar surface and the nonplanar surface by filling voxels that are not free space. For each point $\mathbf{p}$ on the nonplanar surface, its projection point on the planar surface $\mathbf{p_{proj}}$ is computed. If none of the voxels between $\mathbf{p}$ and $\mathbf{p_{proj}}$ are free space, these voxels are temporarily labeled as the nonplanar surface. If new points can be added between the two surfaces, there exists a valid connection between the two surfaces and the new points will be permanently added to the octree as well as the nonplanar surface.

Completion within Individual Planar Surfaces

To generate compact models, the planar surfaces are also completed using their parameters and the visibility information apart from completion when creating the connections between planar surfaces. Since the nonplanar surfaces in this paper are represented by a cluster of points without any parametric representation, it is difficult to define complete models for them and thus they are only filled when finding the connections.

To facilitate the completion of individual planar surfaces, this work estimates a cuboid for each planar surface while considering the measurement error. Algorithm 2 shows the pseudocode of estimating the cuboid from the point cloud $\boldsymbol{P}$ assigned to the planar surface. Firstly, in Line 2 all the points on that planar surface are rotated to a plane parallel to XZ according to the normal vector of the planar surface. After this process, points of the rotated point cloud $\boldsymbol{P}'$ have nearly the same Y values. Then in Line 4, the minimum enclosing rectangle $rect$ of the 2D point sets containing X and Z of $\boldsymbol{P}'$ is estimated by ignoring Y values of these points. In Line 5, the mean μ_y and standard deviation σ_y of Y of $\boldsymbol{P}'$ are calculated. In Lines 6-11, the eight corners of the cuboid are computed based on $rect$, μ_y, and σ_y while the thickness of the estimated cuboid is $3\sigma_y$. Line 12 rotates the cuboid corners back as Line 2 transforms all points to a plane parallel to XZ.

The thickness of the cuboid of a planar surface reflects the uncertainty of this plane. For a planar surface close to the sensor, the thickness of its cuboid is usually smaller compared to the plane far away from the sensor. By taking into consideration the thickness of cuboids, the cuboid representation helps to distinguish coplanar and parallel relations between planar surfaces, and thus benefits the inference of the connectivity relations between parallel planes.

Algorithm 2 Estimate a cuboid for a planar surface

```
1   function EstimateCuboid ($\boldsymbol{P}$)
2       $\boldsymbol{P'} \leftarrow$ RotateToXZ($\boldsymbol{P}$)
3       $\boldsymbol{P_{XZ}} \leftarrow \{\boldsymbol{P'}.x, \boldsymbol{P'}.z\}$
4       $rect \leftarrow$ FindMinimumEnclosingRectangle($\boldsymbol{P_{XZ}}$)
5       $(\mu_y, \sigma_y) \leftarrow$ ComputeMeanAndSTD($\boldsymbol{P'}.y$)
6       for $i \leftarrow [1,4]$ do
7           $cuboid[i] \leftarrow$Point($rect[i].x, \mu_y - 1.5\sigma_y, rect[i].z$)
8       end for
9       for $i \leftarrow [1,4]$ do
10          $cuboid[i+4] \leftarrow$Point($rect[i].x, \mu_y + 1.5\sigma_y, rect[i].z$)
11      end for
12      $cuboid \leftarrow$RotateBack($cuboid$)
13      return $cuboid$
14  end function
```

Another advantage of the cuboid representation is that combined with the visibility of voxels, the cuboid can be used to complete planar surfaces to get a complete planar surface model. After updating the connectivity graph and completing intersecting planar surfaces, the cuboid is utilized to add points to the planar surfaces.

The connected component analysis is performed to further complete a planar surface. Based on the voxels that are not labeled as free space within the cuboid of the planar surface, the Euclidean clustering method is performed to detect connected clusters of voxels. If the distance between a cluster and the planar surface is smaller than a threshold (in this paper $2.5v_s$), the voxels of this cluster will be added to the planar surface. In addition, to avoid noise in computing the visibility information, all the free space voxels within the cuboid are also clustered using the Euclidean clustering method. The small connected components are assigned to the current planar surface while the others are not filled so as to maintain large free space of planar surfaces.

Experimental Results and Discussion

Experimental Setup

To evaluate the proposed method on real-world scenarios, we collected datasets of three different indoor scenes: (i) a cubic office desk, (ii) a typical officer corner with printers, and (iii) a table. The datasets were collected using an ASUS Xtion PRO while the point-plane SLAM algorithm (Taguchi et al. 2013) was employed to obtain the registered point clouds and the sensor poses. The octree voxel size v_s was set as 0.02m.

In addition, to quantitatively evaluate the completion correctness and the model quality, the ICL-NUIM living room dataset (Handa et al. 2014) is utilized as it has ground truth mesh models. The ICL-NUIM dataset is a synthetic RGB-D dataset designed for evaluation of visual odometry and SLAM methods and contains the ground truth poses of the sensors. These sensor poses are used to register all the frames and build the TSDF octree. The octree voxel size v_s was set as 0.01m.

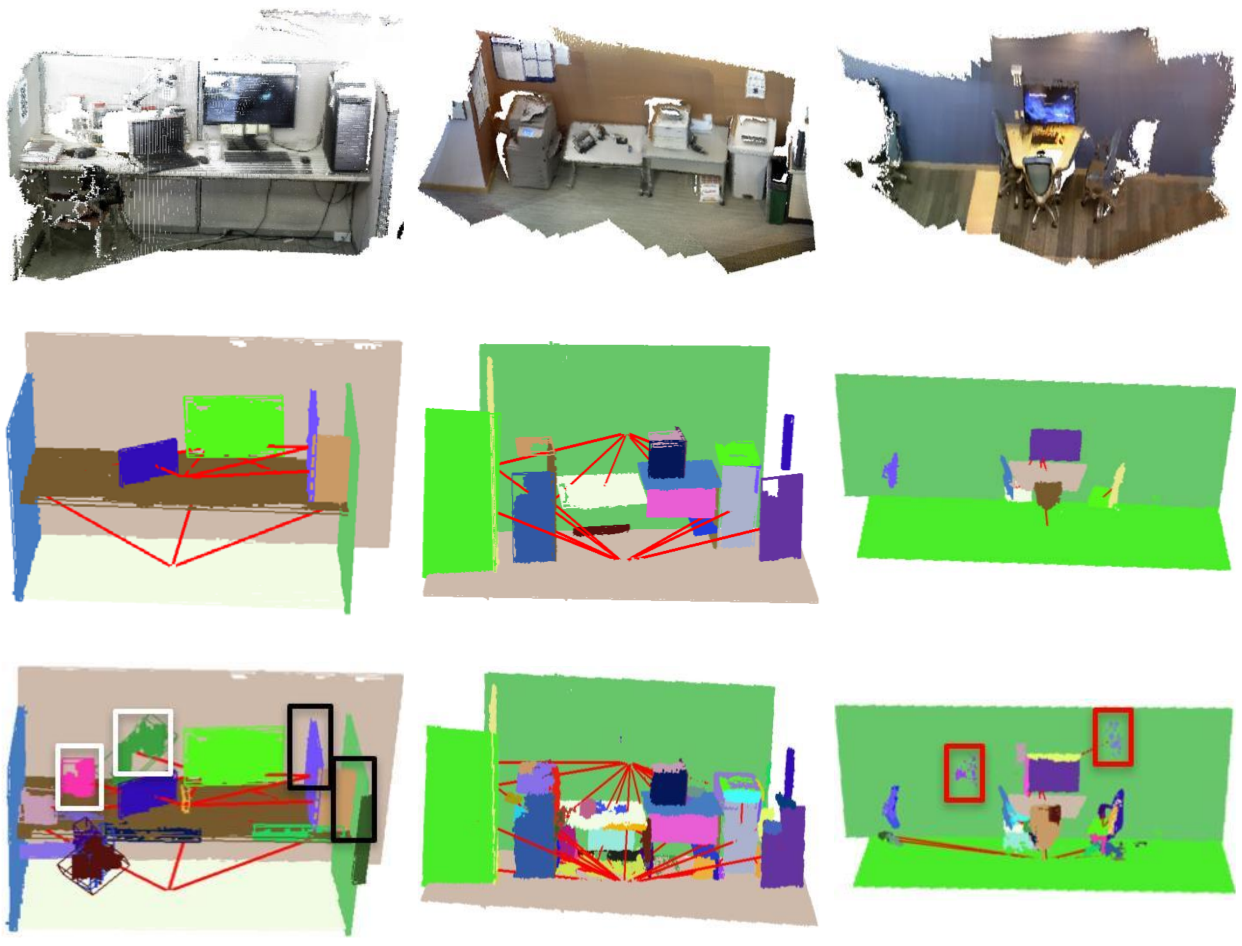
Regarding the thresholds in the surface detection , when processing the real-world datasets, to detect the major planar surfaces, the neighboring search radius is $5v_s$, the angle threshold for the normal difference is 6^o, the curvature threshold is 2, and the minimum cluster size is 300. For the ICL-NUIM datasets, the thresholds for detecting the major planar surfaces are $5v_s$, 3^o, 1 and 300, respectively. When detecting small planar and nonplanar surfaces, the thresholds are $3v_s$, 10^o, 10 and 150, respectively for all the datasets.

Results on Real-World Datasets

The accuracy of the connectivity relations between detected surfaces is evaluated using the real-world datasets. The ground truth connectivity relations of detected surfaces are manually identified and compared to those estimated by the proposed method. As the connectivity relations rely on the surface detection results, the connections related to undetected surfaces (small or irregular surfaces) are not considered.

Fig. 3 displays the results of processing the real-world datasets, including the original point cloud of the TSDF octree, the results after processing major planar surfaces, and the final results after processing small planar and nonplanar surfaces. Table 1 shows the connectivity evaluation results of the three real-world datasets. For each scene, the results after processing the major planar surfaces (the *Plane* column in Table 1), $\boldsymbol{M_P}$ are also evaluated as well as the final results $\boldsymbol{M_F}$ which are generated by processing small planar and nonplanar surfaces based on $\boldsymbol{M_P}$**.** As shown in Table 1, the number of false detected connections $|\boldsymbol{E_{err}}|$ for $\boldsymbol{M_F}$ is low, equal or less than seven for all the three scenes. The ratios of false detected connections, $|\boldsymbol{E_{err}}|/|\boldsymbol{E}|$ are equal or less than 22%, which demonstrates that at least 78% of the detected surface connections are correct.

The false connections of $\boldsymbol{M_F}$ are caused by overfilling of intersecting planar surfaces and from wrong detected surfaces. The overfilling of intersecting planar surfaces denotes the case that when two intersecting planar surfaces are not connected in the real-world, the proposed method creates a connection between them by adding points between them. It occurs when a valid connection can be created due to a lack of visibility information of the voxels between them. For example, the desktop box in Scene (i) is not connected to any of the walls. However, the proposed method fills the gap between the desktop box and the walls and thus create connections between them as shown in the black rectangles in the bottom image of Fig. 3(a).



(a) Scene (i) (b) Scene (ii) (c) Scene (iii)

Fig. 3. Results on the real-world datasets. The first row shows the original point cloud in the octree $\boldsymbol{P_{oct}}$. The second row displays the results after processing the major planar surfaces $\boldsymbol{M_P}$, where each surface is rendered using a random color and the red segments represent the connections. The last row shows the results after processing the small planar and nonplanar surfaces $\boldsymbol{M_F}$. The black rectangles in (a) show the false connections due to over-filling of intersecting planar surfaces while the red rectangles in (c) include the false connections from wrong detected surfaces. The white rectangles in (a) cover example areas where the nonplanar surfaces and the connections are correctly identified.

Table 1. Evaluation of detected connections of the real-world datasets. $|\boldsymbol{V}|$ is the number of surfaces in $\boldsymbol{V}$ while $|\boldsymbol{E}|$ represents the number of detected connections among surfaces. $|\boldsymbol{E}_{miss}|$ denotes the number of undetected connections while $|\boldsymbol{E}_{err}|$ is the number of false detected connections.

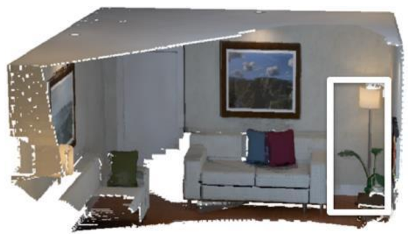
	Scene (i)		Scene (ii)		Scene (iii)	
	Plane	Final	Plane	Final	Plane	Final
$\|\boldsymbol{V}\|$	9	20	21	63	10	36
$\|\boldsymbol{E}\|$	15	21	24	65	6	27
$\|\boldsymbol{E}_{err}\|$	2	3	3	7	1	6
$\|\boldsymbol{E}_{err}\|/\|\boldsymbol{E}\|$	13%	14%	13%	11%	17%	22%
$\|\boldsymbol{E}_{miss}\|$	0	0	0	0	0	0

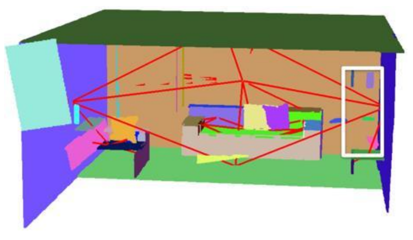
False detected planar surfaces are mainly caused by registration errors of multiple depth frames and the uncertainty of sensor measurements. This is the reason why $|\boldsymbol{E}_{err}|$ increases after processing small planar and nonplanar surfaces, i.e., as shown in Table 1, the *Final* columns have larger $|\boldsymbol{E}_{err}|$ compared to the *Plane* columns. For example, for the wall in Scene (iii) in Fig. 3(c), the nonplanar surface detection method detects multiple clusters around the wall as shown in the red rectangles of the bottom image of Fig. 3(c) and these clusters create false connections in the final results. Even though processing small planar and nonplanar surfaces leads to the increasing of errors in estimating connections, it can still identify some correct small planar or nonplanar surfaces and find the correct connections. As shown in Fig. 3(a), within the white rectangles, a lamp (in green) and a cup (in magenta) are correctly added to the model with correct connectivity relations.

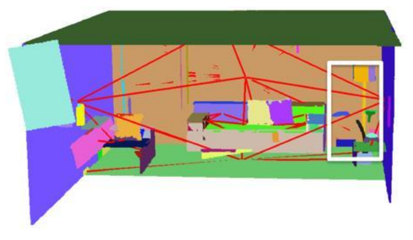
The number of undetected connections $|\boldsymbol{E}_{miss}|$ is zero, which demonstrates that the proposed method is able to recover all the connections of surfaces in $\boldsymbol{V}$. According to our method, there exist three types of connections, (1) the connection between intersecting planar surfaces, (2) the connection between two parallel planar surfaces, and (3) the connection between a nonplanar surface and a planar surface. Based on the criteria for connecting surfaces, missing connections might occur when (1) many points between two actually connected surfaces (two planar surfaces, or a nonplanar surface and a planar surface) are not observed and their distance is too larger to be considered for creating connections, and (2) a nonplanar surface is connected to more than one planar surface while our method only detects a single connection to a planar surface.

Results on ICL-NUIM Datasets

We utilize the ICL-NUIM living room dataset which contains four scenes, kt0, kt1, kt2, and kt3, to evaluate (1) the overall accuracy of the final models by comparing the final model point cloud with the ground truth point cloud, and (2) the completion results by counting the number of correctly filled points. As the four scenes have some overlapping areas, Fig. 4 shows the modeling and connectivity inference results of kt0 which contains the major part of the living room dataset. Fig. 4 (b) displays the results after processing the major planar surfaces $\boldsymbol{M_P}$ while (c) shows the results by adding small planar surfaces and nonplanar surfaces $\boldsymbol{M_F}$. By comparing Fig. 4 (b) and (c) and using the close-up views in (d), i.e. the second and third rows of (d), it can be found that the results after processing small planar and nonplanar surfaces successfully add many small planar or nonplanar surfaces to the results, e.g., the plant pot and lamp in the white rectangles.


(a) Point cloud in the octree $\boldsymbol{P_{oct}}$


(b) Results after processing the major planar surfaces


(c) Final results after adding small planar and nonplanar surfaces

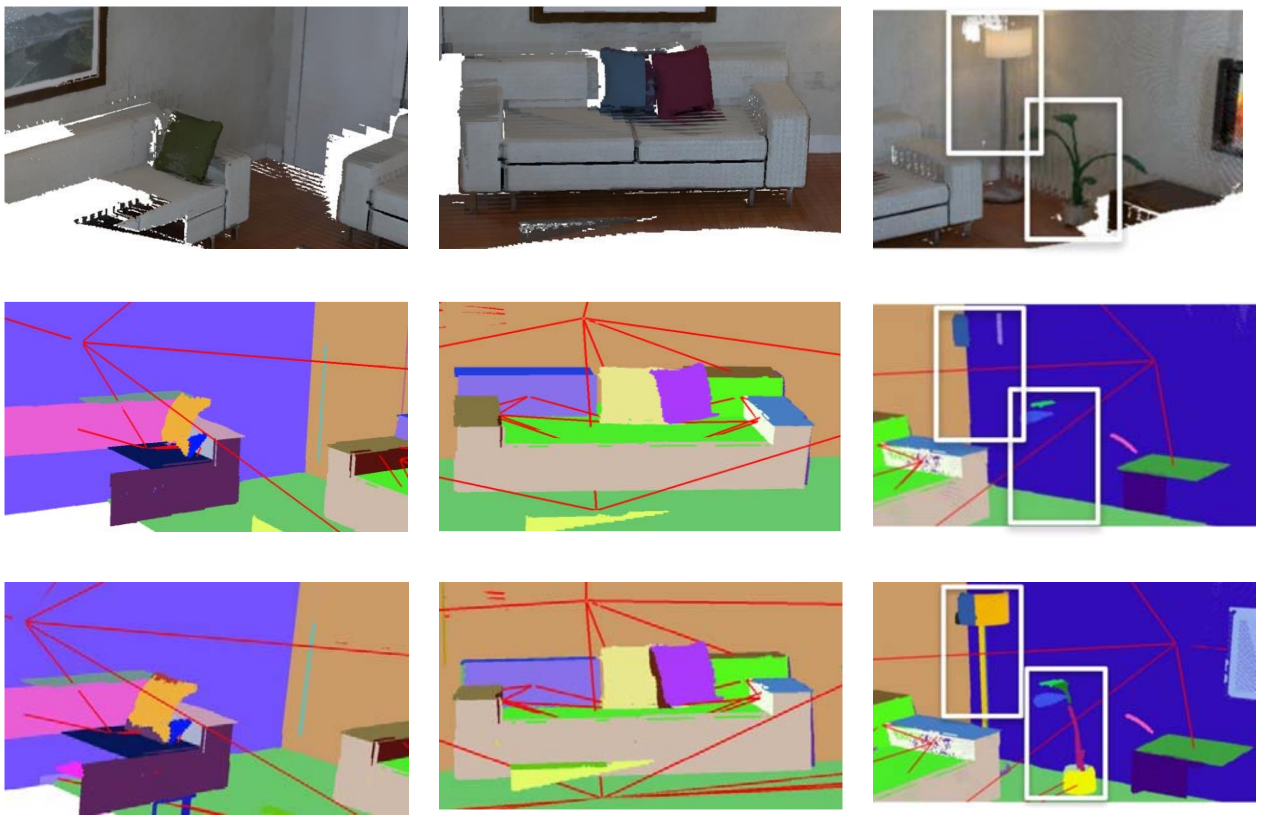
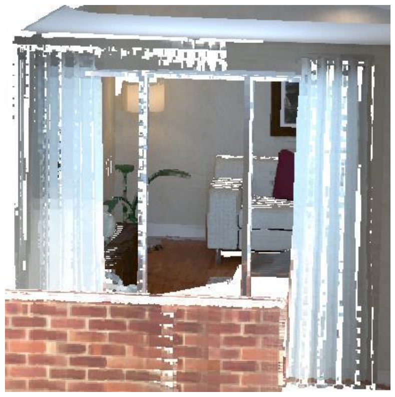

(d) Close-up views of some areas of the three point clouds from left to right.

Fig. 4. Results of kt0. (a) displays the original point cloud of the octree $\boldsymbol{P_{oct}}$ while (b) and (c) respectively show the results after processing the major planar surfaces and adding nonplanar surfaces. In (b) and (c), each surface is rendered by a random color while the red segment denotes that there is a connection between two surfaces. (d) shows the close-up view of some areas of the point clouds in the above row. The white rectangles show example areas that nonplanar surfaces from a lamp and a plant pot are added to the final model after processing small planar and nonplanar

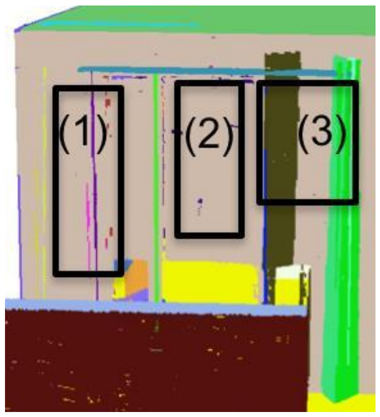
surfaces.

Table 2. Evaluation of the original point cloud $\boldsymbol{P_{oct}}$ and the final model point cloud $\boldsymbol{P_m}$ with respect to the ground truth point cloud $\boldsymbol{P_{gt}}$.

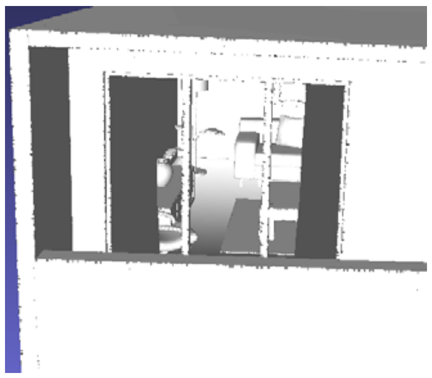
Point Cloud	Error (m)	kt0	kt1	kt2	kt3
$\boldsymbol{P_{oct}}$	Mean	0.006	0.006	0.006	0.007
	Median	0.005	0.006	0.007	0.006
	Std.	0.003	0.003	0.003	0.003
	Max.	0.019	0.021	0.021	0.023
$\boldsymbol{P_m}$	Mean	0.008	0.016	0.012	0.011
	Median	0.007	0.011	0.008	0.007
	Std.	0.007	0.024	0.022	0.02
	Max.	0.188	0.351	0.352	0.351



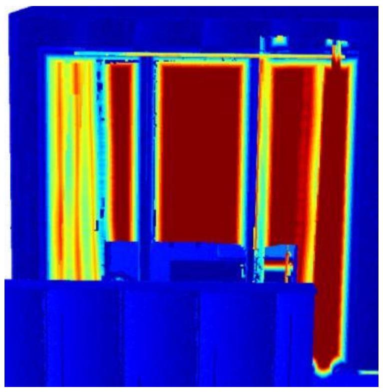
(a) Original point cloud of the window area.



(b) Final models of the window area.



(c) Ground truth for that area.



(d) Errors of that area.

Fig. 5. Errors around the French window area in kt2. (a) shows the original point cloud of the French window are in the octree and (b) displays the models of that area where each surface is rendered by a random color. (c) displays the ground truth points while (d) depicts the errors of the models in (b) by rendering the errors from cool colors to warm colors, i.e. the dark red denotes large errors while the blue represents small errors. The three rectangles in (b) from left to right cover the areas of errors due to (1) processing nonplanar surfaces as planar surfaces, (2) over-filling of planar surfaces, and (3) over-growing of intersecting planar surfaces.

The comparison between the final model point cloud $\boldsymbol{P_m}$ and the ground truth model $\boldsymbol{P_{gt}}$ is performed by estimating the distance between a point pair that one is from $\boldsymbol{P_m}$ and the other in $\boldsymbol{P_{gt}}$. For each point $\boldsymbol{p_m} \in \boldsymbol{P_m}$, its nearest point in $\boldsymbol{P_{gt}}$ is searched and denoted as $\boldsymbol{p_{gm}}$. The Euclidean distance between $\boldsymbol{p_m}$ and $\boldsymbol{p_{gm}}$, $\left\|\boldsymbol{p_m} - \boldsymbol{p_{gm}}\right\|_2$ is calculated and referred to as the error of $\boldsymbol{p_m}$. Then the mean, median, standard deviation, and maximum values of the distances for all the four scenes, i.e., kt0, kt1, kt2, and kt3, are computed and displayed in Table 2. In addition, using the same method, the original point cloud of the octree $\boldsymbol{P_{oct}}$ is evaluated and the results are presented in Table 2. To mitigate the distance differences due to the voxelization in creating the octree, both $\boldsymbol{P_{oct}}$ and $\boldsymbol{P_m}$ are aligned to $\boldsymbol{P_{gt}}$ using the Iterative Closest Point (ICP) (Besl and McKay 1992) algorithm.

The lower mean values (less than or equal to 0.16m) in Table 2 demonstrate that the proposed method is able to reconstruct high-quality models. The fact that all the median values of $\boldsymbol{P_m}$ are lower than the mean values indicates half of the point errors are lower than the mean values. As shown in Table 2, the maximum errors of $\boldsymbol{P_m}$ (the row *Max.*) are larger than those of $\boldsymbol{P_{oct}}$, especially for the last three scenes. The main discrepancy for the last three scenes mainly occurs around a French window area. As shown in Fig. 5, many points (for example the points within the middle rectangle, Rectangle (2) in Fig. 5(b)) are filled within the window frame. This is mainly because those filled voxels are not labeled as free space since there are no points behind the window glasses. However, Fig. 5(d) also indicates that the errors of points on the walls are small, which demonstrates that the proposed method is capable of recovering reliable points if the visibility information is correctly estimated.

The final model point errors can be categorized into three types, (1) errors of using planar surfaces to approximate nonplanar surfaces, (2) errors of over-filling of planar surfaces, and (3) errors of over-growing of planar surfaces, which correspond to the areas covered by the three rectangles in Fig. 5(b).

The errors of using planar surfaces to approximate nonplanar surfaces occur when the surface detection method detects planar surfaces from non-planar objects. For example, the surface detection method detects several planar surfaces, in particular for the area in the left rectangle, Rectangle (1) in Fig. 5(b), for the accordion folding doors before the French windows. During the processing of the planar surfaces, many wrong points are added in finding the connections and filling of the planar surfaces and thus cause large point errors.

The over-filling error is due to the missing of free space information for the voxels within a planar surface. For the French window (Fig. 5), only a small number of the points (mainly the lower parts of the window) within the window area are identified as free space by the points on the ground behind the window. The visibility information of other points (e.g., the points within the middle rectangle, Rectangle (2) in Fig. 5(b)) is unknown due to the visibility information computation strategy. Therefore, as shown in Fig. 5(b), those points are added to the point cloud during the planar surface filling process and lead to large (actually the largest) point errors.

The over-growing of planar surfaces occurs when the voxels between two intersecting planar surfaces are not labeled as free space due to a lack of information. When the distance between the intersecting planar surfaces is smaller than a distance threshold ($15v_s$), the proposed method will add points between them and create a connection between them. In this dataset, there exist many intersecting planar surfaces due to the accordion folding doors before the French window. The surface detection method generates some planar surfaces and the proposed method connects them by adding points behind the accordion folding doors. The large point errors within the right rectangle, Rectangle (3) in Fig. 5(d) are mainly caused by over-growing of planar surfaces.

The number of correct points in $\boldsymbol{P_m}$ with respect to $\boldsymbol{P_{gt}}$ is also estimated using the point error. A point $\boldsymbol{p_m} \in \boldsymbol{P_m}$ is correct if $\left\|\boldsymbol{p_m} - \boldsymbol{p_{gm}}\right\|_2 \leq 2.5v_s$. By excluding $\boldsymbol{P_{oct}}$ from $\boldsymbol{P_m}$, we obtain the number of added points n_{add} and the number of correctly added points n_{corr} in $\boldsymbol{P_m}$. That is, $n_{add} = |\boldsymbol{P_m}| - |\boldsymbol{P_{oct}}|$ and n_{corr} is the difference between the size of correct added points in $\boldsymbol{P_m}$ and the size of $\boldsymbol{P_{oct}}$. The ratio of n_{corr} to n_{add} is also shown in Table 3. The ratios of correctly added points show that more than 87% of the added points from this proposed method are correct. In addition, Table 3 displays the ratio of the final model point cloud size $|\boldsymbol{P_m}|$ to the original point cloud size $|\boldsymbol{P_{oct}}|$. The results demonstrate that although the proposed method does not perform completion for isolated planar surfaces, it is able to at least double the point cloud size.

Table 3. Evaluation of completion results

	kt0	kt1	kt2	kt3
$\lvert\boldsymbol{P_m}\rvert/\lvert\boldsymbol{P_{oct}}\rvert$	2.74	3.20	2.60	2.39
n_{corr}/n_{add}	98%	87%	90%	93%

Computational Analysis

Regarding the computational time on the experimental datasets, the processing time of the proposed method ranges from several minutes to about half an hour as shown in Table 4. The data processing was conducted on a personal desktop with Intel Core^{TM} i7-4790K @ 4.00 GHz equipped with the NVIDIA GeForce GTX 970 graphic card. The program was implemented in C++ with OpenMP while no GPU or other parallel computing methods were utilized. Table 4 shows that the computational time is positively related to the number of frames. The larger the number of frames is, the more computational time the system takes. The computational time is also related to the scale of the scene, which can be reflected by the size of the original point cloud $|\boldsymbol{P}_{oct}|$. A large-scale scene usually contains more points and surfaces and thus requires more computational expense in filling the surfaces and estimating the connections.

Table 4. Computational time of all the datasets. $|\boldsymbol{P}_{oct}|$ represents the number of points of the octree. $|\boldsymbol{V}|$ is the number of surfaces in the connectivity graph $\boldsymbol{G}$ while $|\boldsymbol{E}|$ denotes the number of surfaces in $\boldsymbol{G}$.

	# of frames	$\lvert\boldsymbol{P}_{oct}\rvert$	$\lvert\boldsymbol{V}\rvert/\lvert\boldsymbol{E}\rvert$	Time (minutes)
Scene (i)	60	54,567	20/21	0.78
Scene (ii)	44	169,631	63/65	0.84
Scene (iii)	70	188,967	36/27	1.42
kt0	1,509	660,577	68/64	34.73
kt1	966	832,596	117/158	29.23
kt2	881	1,046,417	127/176	31.66
kt3	1,241	742,691	93/104	32.77

In the proposed method, there are two main processes that affect the computational time: (i) creating the TSDF octree, and (ii) updating the connectivity graph using the major planar surfaces. When constructing the TSDF octree, the system loads each depth frame and utilizes each observed point of the frame to update the TSDF tree. Therefore, the number of frames greatly affects the computational time. Moreover, the graph updating using the major planar surfaces involves surface parameter updating (e.g., computing plane equations and estimating the cuboid) and traversing a large number of voxels in order to decide their labels. Thus a large number of planar surfaces in the scene generally leads to more computational time compared to a scene containing a small number of planar surfaces.

Conclusions and Future Work

This paper presented a framework that integrates point cloud completion and surface connectivity relation inference into a combined process to obtain complete 3D models and surface connections. The framework utilizes geometric properties of surfaces and the visibility of octree voxels to estimate the connections of surfaces and recover missing points between the surfaces. The method first processes the major planar surfaces to estimate their connectivity relations and fill the missing points. Then small planar surfaces and nonplanar surfaces are utilized to find more connections between them and the major planar surfaces by adding points if necessary. Furthermore, individual planar surfaces are further filled using the connected component analysis within the surface cuboid to obtain complete surface models. Experimental results demonstrated that the proposed method is able to recover all connectivity relations between surfaces, double the point cloud size by adding points of which more than 87% are correct, and obtain high-quality 3D models.

The proposed method handles nonplanar surfaces using a basic strategy, i.e. growing their points toward planar surfaces if possible. It does not incorporate the geometric properties of the nonplanar surfaces. In addition, the connectivity relations between nonplanar surfaces are not estimated in this paper. Future work will explore to segment more primitives other than planes (e.g., cylinder, sphere, etc.) using non-uniform B-Spline surface fitting methods (Dimitrov et al.

2016) and infer connectivity relations between nonplanar surfaces as well. Future work will also improve the computational efficiency and apply the method to large-scale indoor scenes.

References

Ahmed, M. F., Haas, C. T., and Haas, R. (2014). "Automatic detection of cylindrical objects in built facilities." *Journal of Computing in Civil Engineering*, 28(3), 4014009.

Arnaud, A., Christophe, J., Gouiffes, M., and Ammi, M. (2016). "3D reconstruction of indoor building environments with new generation of tablets." *Proceedings of ACM Conference on Virtual Reality Software and Technology*, 187–190.

Azhar, S. (2011). "Building Information Modeling (BIM): trends, benefits, risks, and challenges for the AEC industry." *Leadership and Management in Engineering*, 11(3), 241–252.

Barbu, A., and Zhu, S.-C. (2005). "Generalizing Swendsen-Wang to sampling arbitrary posterior probabilities." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8), 1239–1253.

Besl, P. J., and McKay, N. D. (1992). "A Method for Registration of 3-D Shapes." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2), 239–256.

Bhatla, A., Choe, S. Y., Fierro, O., and Leite, F. (2012). "Evaluation of accuracy of as-built 3D modeling from photos taken by handheld digital cameras." *Automation in Construction*, 28, 116–127.

Borrmann, A., and Rank, E. (2009). "Specification and implementation of directional operators in a 3D spatial query language for building information models." *Advanced Engineering Informatics*, 23(1), 32–44.

Brilakis, I., Lourakis, M., Sacks, R., Savarese, S., Christodoulou, S., Teizer, J., and Makhmalbaf, A. (2010). "Toward automated generation of parametric BIMs based on hybrid video and laser scanning data." *Advanced Engineering Informatics*, 24(4), 456–465.

Carr, J. C., Beatson, R. K., Cherrie, J. B., Mitchell, T. J., Fright, W. R., McCallum, B. C., and Evans, T. R. (2001). "Reconstruction and representation of 3D objects with radial basis functions." *Proceedings of Annual Conference on Computer Graphics and Interactive Techniques*, 67–76.

Chauve, A. L., Labatut, P., and Pons, J. P. (2010). "Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data." *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1261–1268.

Curless, B., and Levoy, M. (1996). "A volumetric method for building complex models from range images." *Proceedings of Annual Conference on Computer Graphics and Interactive Techniques*, 303–312.

Dimitrov, A., Gu, R., and Golparvar-Fard, M. (2016). "Non-uniform B-spline surface fitting from unordered 3D point clouds for as-built modeling." *Computer-Aided Civil and Infrastructure Engineering*, 31(7), 483–498.

Giel, B., and Issa, R. R. A. (2012). "Using Laser Scanning to Access the Accuracy of As-Built BIM." *Proceedings of International Workshop on Computing in Civil Engineering*, 665–672.

Golparvar-Fard, M., Peña-Mora, F., and Savarese, S. (2011). "Integrated sequential as-built and as-planned representation with D4 AR tools in support of decision-making tasks in the AEC/FM industry." *Journal of Construction Engineering and Management*, 137(12), 1099–1116.

Hajian, H., and Becerik-Gerber, B. (2010). "Scan to BIM: factors affecting operational and computational errors and productivity loss." *Proceedings of International Symposium on Automation and Robotics in Construction*, 265–272.

Handa, A., Whelan, T., McDonald, J. B., and Davison, A. J. (2014). "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM." *Proceedings of IEEE International Conference on Robotics and Automation*, 1524–1531.

Hardin, B., and McCool, D. (2015). *BIM and construction management: proven tools, methods, and workflows*. John Wiley & Sons, Incorporated, Indianapolis, Indiana.

Janaszewski, M., Couprie, M., and Babout, L. (2010). "Hole filling in 3D volumetric objects." *Pattern Recognition*, 43(10), 3548–3559.

Kim, Y. M., Mitra, N. J., Yan, D.-M., and Guibas, L. (2012). "Acquiring 3D indoor environments with variability and repetition." *ACM Transactions on Graphics*, 31(6), 138:1–138:11.

Klein, L., Li, N., and Becerik-Gerber, B. (2012). "Imaged-based verification of as-built documentation of operational buildings." *Automation in Construction*, 21, 161–171.

Kroemer, O., Ben Amor, H., Ewerton, M., and Peters, J. (2012). "Point cloud completion using extrusions." *Proceedings of IEEE-RAS International Conference on Humanoid Robots*, 680–685.

Li, Y., Wu, X., Chrysathou, Y., Sharf, A., Cohen-Or, D., and Mitra, N. J. (2011). "GlobFit: consistently fitting primitives by discovering global relations." *ACM Transactions on Graphics*, 30(4), 52:1–52:12.

Nan, L., Xie, K., and Sharf, A. (2012). "A search-classify approach for cluttered indoor scene understanding." *ACM Transactions on Graphics*, 31(6), 137:1–137:10.

Nepal, M. P., Staub-French, S., Zhang, J., Lawrence, M., and Pottinger, R. (2008). "Deriving construction features from an IFC model." *Proceedings of Annual Conference of the Canadian Society for Civil Engineering*, 426–436.

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). "KinectFusion: Real-Time Dense Surface Mapping and Tracking." *Proceedings of IEEE International Symposium on Mixed and Augmented Reality*, 127–136.

Nguyen, T.-H., Oloufa, A. A., and Nassar, K. (2005). "Algorithms for automated deduction of topological information." *Automation in Construction*, 14(1), 59–70.

Pătrăucean, V., Armeni, I., Nahangi, M., Yeung, J., Brilakis, I., and Haas, C. (2015). "State of research in automatic as-built modelling." *Advanced Engineering Informatics*, 29(2), 162–171.

Shao, T., Monszpart, A., Zheng, Y., Koo, B., Xu, W., Zhou, K., and Mitra, N. J. (2014). "Imagining the unseen: stability-based cuboid arrangements for scene understanding." *ACM Transactions on Graphics*, 33(6), 209:1–209:11.

Shao, T., Xu, W., Zhou, K., Wang, J., Li, D., and Guo, B. (2012). "An interactive approach to semantic modeling of indoor scenes with an RGBD camera." *ACM Transactions on Graphics*, 31(6), 136:1–136:11.

Sharf, A., Alexa, M., and Cohen-Or, D. (2004). "Context-based surface completion." *ACM*

Transactions on Graphics, 23(3), 878–887.

Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). “Indoor segmentation and support inference from RGBD images.” *Proceedings of European Conference on Computer Vision*, 746–760.

Silberman, N., Shapira, L., Gal, R., and Kohli, P. (2014)." A contour completion model for augmenting surface reconstructions." In European Conference on Computer Vision (pp. 488-503). Springer, Cham.

Son, H., Kim, C., and Kim, C. (2015). “Fully automated as-built 3D pipeline extraction method from laser-scanned data based on curvature computation.” *Journal of Computing in Civil Engineering*, 29(4), B4014003.

Song, S., and Xiao, J. (2014). “Sliding shapes for 3D object detection in depth images.” *Proceedings of European Conference on Computer Vision*, 634–651.

Song, S., *Yu, F., Zeng, A., Chang, A. X., Savva, M., & Funkhouser, T. (2016)."Semantic scene completion from a single depth image". arXiv preprint arXiv:1611.08974.*

Sung, M., Kim, V. G., Angst, R., and Guibas, L. (2015). “Data-driven structural priors for shape completion.” *ACM Transactions on Graphics*, 34(6), 175:1–175:11.

Taguchi, Y., Jian, Y. D., Ramalingam, S., and Feng, C. (2013). “Point-plane SLAM for hand-held 3D sensors.” *Proceedings of IEEE International Conference on Robotics and Automation*, 5182–5189.

Tang, P., Huber, D., Akinci, B., Lipman, R., and Lytle, A. (2010). “Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques.” *Automation in Construction*, 19(7), 829–843.

Volk, R., Stengel, J., and Schultmann, F. (2014). “Building Information Modeling (BIM) for existing buildings: Literature review and future needs.” *Automation in Construction*, 38, 109–127.

Wang, J., and Oliveira, M. M. (2007). “Filling holes on locally smooth surfaces reconstructed from point clouds.” *Image and Vision Computing*, 25(1), 103–113.

Xiao, Y., Wang, C., Xi, X. H., and Zhang, W. M. (2014). “A comprehensive framework of building model reconstruction from airborne LIDAR data.” *IOP Conference Series: Earth and Environmental Science*, 12178.

Xiong, X., Adan, A., Akinci, B., and Huber, D. (2013). “Automatic creation of semantically rich 3D building models from laser scanner data.” *Automation in Construction*, 31, 325–337.

Zheng, B., Zhao, Y., Yu, J. C., Ikeuchi, K., and Zhu, S. C. (2013). “Beyond point clouds: Scene understanding by reasoning geometry and physics.” *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 3127–3134.

Zhu, Z., and Donia, S. (2013). “Potentials of RGB-D Cameras in As-Built Indoor Environment Modeling.” *Proceedings of International Workshop on Computing in Civil Engineering*, 605–612.