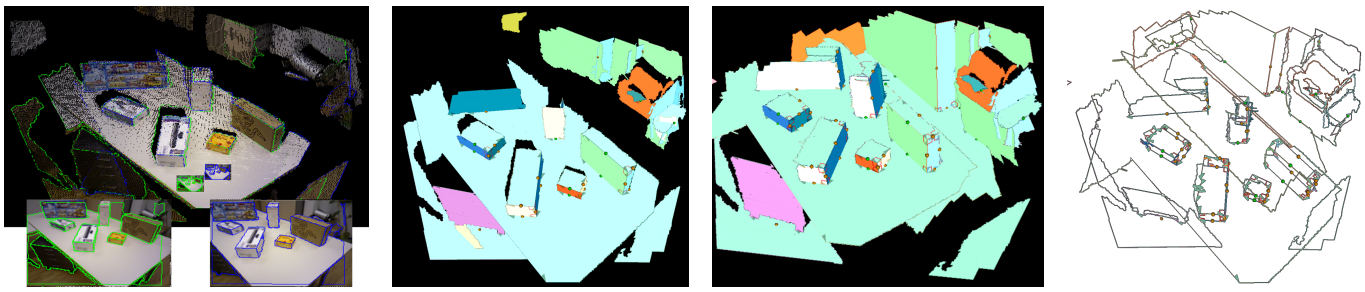
To appear in an IEEE VGTC sponsored conference proceedings

Structural Modeling from Depth Images

Thanh Nguyen, *Student Member, IEEE*, Gerhard Reitmayr, *Member, IEEE*, and Dieter Schmalstieg, *Senior Member, IEEE*



(a) (b) (c) (d)

Fig. 1: Overview of the structural modeling pipeline. (a) We first obtain a 3D scan of the environment and respective keyframes, to which we apply an efficient plane segmentation algorithm to find planar regions in the scene. (b) The regions found in the first two keyframes are integrated in an initial model. (c) Further regions from subsequent keyframes are integrated with existing planes to recover topologically meaningful geometry. (d) The structural model correctly captures joint edges and vertices between recovered planes, marked as dots and cubes in the image.

Abstract—In this work, we present a new automatic system for scene reconstruction of high-level structural models. We start with identifying planar regions in depth images obtained with a SLAM system. Our main contribution is an approach which identifies constraints such as incidence and orthogonality of planar surfaces and uses them in an incremental optimization framework to extract high-level structural models. The result is a manifold mesh with a low number of polygons, immediately useful in many Augmented Reality applications such as inspection, interior design or spatial interaction.

Index Terms—Structural modeling, geometric scene understanding, topology construction

1 Introduction

Augmented Reality (AR) relies on knowledge of the scene surrounding the user for registering virtual objects and for tangible interfaces. Commodity depth sensors and advances in simultaneous localization and mapping (SLAM) have revolutionized rapid 3D scanning, but the resulting purely geometric models do not provide clues on the *structure* of the environment. By structure, we mean high-level semantic information, such as the shape, class or purpose of geometric entities in the environment. Such *geometric scene understanding* is considered a very challenging artificial intelligence problem. Nonetheless, the topic has gained significant attention in recent years, especially in real-time computer vision and robotics.

In this work, we address structural modeling, which is concerned with recovering topologically sound high-level geometric primitives from 3D scanned geometry. We focus on planar structures, which can be robustly identified and processed into topologically connected polygonal models. Planar structures do not cover all possible shapes, but they are prevalent in human-made environments and immediately useful for AR applications.

Consider a simple office room (Figure 1): It consists of a desk, a floor, four walls, and a few boxes on the desk or floor. A conventional 3D scan may approximate one wall with thousands of sample points up to millimeter-level accuracy, but it is unable to return a simple wall-sized rectangle. If structural modeling provides such a rectangle registered to the real environment, an AR application could use simple texture-mapping onto a quad to visualize a new tapestry for interior decoration or just to display a webpage. A kitchen planning application could measure and count the shelves of an existing cupboard for refurbishing. A clean, topologically sound model of the environment has many more applications in AR, for instance, geometric object detection, scene measurements, parametric computer-aided design, illumination simulation and spatial interaction.

We present a new automatic system that explicitly focuses on obtaining a topologically correct model. As in previous work, we first identify planar regions in depth images obtained with a SLAM system. On these planar regions, we identify geometric constraints (coplanarity, incidence, orthogonality) of planar surfaces. High-level structural models are created with incremental optimization using these constraints. The final result is a polyhedron with a low number of polygons, immediately useful for AR applications such as geometric and semantic manipulation.

2 Related work

Several approaches exist for establishing structural models from geometric input. They can roughly be categorized into bottom-up and top-down approaches. In bottom-up approaches, the raw measurement data is abstracted, until the desired level of abstraction is obtained. This approach fits well with global reconstruction, where a large body of raw data can be integrated into a global model, such as point clouds from LIDAR or depth cameras. Furukawa *et al*. [7] demonstrated this approach on indoor image data sets. Similarly, Whelan *et al*. [26] introduced a new method to simplify dense point clouds into 3D polygon models. Their method computes incremental plane segmentations on existing point clouds to derive a global mesh model consisting of planar triangulated surfaces as well as non-planar surfaces. This approach shows qualitatively interesting reconstructed models. However, it is designed for offline processing and has a high computational cost.

- *Thanh Nguyen is with Graz University of Technology. E-mail: thanh.nguyen@icg.tugraz.at.*
- *Gerhard Reitmayr is with Qualcomm Research Austria. E-mail: gerhardr@qti.qualcomm.com.*
- *Dieter Schmalstieg is with Graz University of Technology. E-mail: schmalstieg@tugraz.at.*

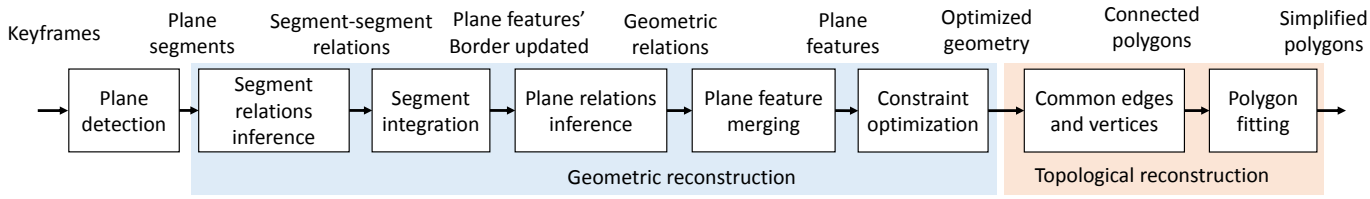


Fig. 2: This diagram illustrates the incremental structural modeling pipeline

An alternative approach proceeds in a top-down fashion and explicitly fits prior models to the raw data. Again, the first methods in this category worked offline on global data sets. Schnabel *et al*. [20] showed how to efficiently fit a set of parameterized 3D primitives such as spheres, cylinders, boxes and planes into a large dense point cloud. This was further refined by finding and enforcing spatial relationships between primitives [15]. Another approach using prior models is to detect a fixed set of classes of objects based on 3D models [21] or learned classifiers [13].

Prior models can also include more than simple geometric primitives or object detection. Knowledge about constraints between objects, such as volumetric reasoning [14] or physical support and mechanics [8], have been exploited in estimating block-based or object-based models. These approaches usually require an optimization over a label space to incorporate the constraints in the modeling. The above approaches employ expensive detection and estimation algorithms on large data sets and cannot be run in real-time or on data that becomes available incrementally.

Bottom-up approaches can be adapted to operate incrementally and, thus, can be embedded in a SLAM system, for example based on 3D feature points [4]. With the advent of cheap and robust depth sensors, several SLAM systems have emerged which target the reconstruction of simplified maps of complex environments. Trevor *et al*. [23] proposed planar SLAM for reconstructing the planar surfaces of a scene. Their system exploits advantages of both close-range and far-range depth sensors. However, such a system is difficult to deploy in AR. Recently, Salas-Moreno *et al*. [18] introduced a SLAM system that partitions a dense reconstruction into planar and non-planar parts and enforces planarity on the plane segments. While the above approaches demonstrate promising results, they require significant computational power, when integrating planar regions in a point-wise fashion. Furthermore, they do not aim for establishing higher-level relationships between the plane primitives.

For interactive systems, user input can provide the necessary information to build geometrically sound reconstructions. Several systems provide interactive modeling tools supported by 3D reconstruction information, such as camera poses, 3D points and plane estimates [5, 24, 22, 2]. These approaches require all information to be available up-front and do not work incrementally. Furthermore, the user interfaces resemble rather complicated 3D modeling software and are not easily re-usable in mobile setups.

Online interaction for 3D reconstruction is an even larger challenge, because fundamental information, such as the camera pose, needs to be estimated online with limited computational resources. However, this situation also presents two distinct opportunities: First, automatic computation of higher-level objects can be used to improve the robustness and accuracy of the SLAM process [19]. Second, the SLAM operator can immediately inspect the model and plan the next steps. Several SLAM systems demonstrate that users can build edge-based models [3] or fully polygonal models [25]. Here, the SLAM system provides camera poses that are used with interactive modeling tools to define 3D vertices, edges and polygon outlines of the 3D model.

Nguyen *et al*. [17] presented a simplified approach based on the aforementioned concept. The user annotates only vertices of a room model, and the system uses a combination of a normal camera and a single-point laser range finder to estimate the adjacent wall structures. A global estimation step establishes the recovery of the overall shape of the room. The system was limited to pure camera rotations and made strong assumptions about the shape of the environment. As a result, this system could only reconstruct simple models and left several details unsolved. A similar system was described by Kim *et al*. [11] using a Kinect camera combined with a projector for output. User input is provided through a single button to add a planar surface to the reconstruction. The system models vertical, rectangular walls and enforces orthogonality between segments.

O-Snap [2] operates as an interactive post-processing step on high quality dense point-clouds, but, otherwise, shares many aspects with our work. The point-cloud is processed using a semi-automatic approach to combine advantages of both low-level automatic construction and interactive user input. They apply plane fitting on the point-cloud, which results in initial polygons. The user can interactively modify vertices, until achieving the desired reconstruction results. The authors claim that their method achieves high accuracy within reasonable time. However, O-Snap is designed for a desktop setup.

We propose a fully automatic system, which works incrementally, in-situ, and has a low computational cost. This makes it much more amenable to mobile AR. The system delivers high-level structural information, which makes it easy to build mobile user interaction on top. We do not assume a reliable global reconstruction and work with a mobile SLAM system. Consequently, our work is easily applicable to consumer applications on low cost hardware.

3 System Overview

Our system incrementally builds a structural model from depth image keyframes with known camera poses. Figure 2 shows an overview of the processing steps. The pipeline works incrementally, while the user is actively observing the scene. Keyframes are selected by the underlying SLAM system, which runs in parallel to the reconstruction. Every time a new keyframe is added by the SLAM system, the pipeline is invoked and improves the structural model (Figure 3).

The three main stages each deal with different aspects of the reconstruction problem. The *plane detection* groups the individual measurements of a single depth image into *plane segments* that describe a single plane (see section 4). The *geometric reconstruction* estimates global *plane features* by combining individual segments, and optimizes the plane geometry by applying different geometric constraints between the global plane features (see section 5). Finally, the *topological reconstruction* combines the polygon outlines of the plane features (see section 6). The input and output of each stage is described in more detail in the following.

Simultaneous localization and mapping Our system obtains input from a consumer depth sensor (Kinect). We use RGB-D SLAM [10], which scales better to large environments than volumetric SLAM [16] and does not require a GPU. It uses concurrent threads for camera tracking and for mapping. The scene is mapped as a set of RGB-D keyframes, which provide the input to our structural modeling pipeline. A keyframe K stores a camera pose T as the transformation from the local camera frame to the world coordinate frame, a color image and a depth image.

Plane detection A new keyframe K is handed to the plane detection stage, which extracts *plane segments*. A plane segment S is a connected component in image space that can be modeled with a plane in 3D space. It consists of a set of pixel indices in the keyframe, a plane equation and an in-plane 3D polygon $\mathbf{b}$ corresponding to the segment border.

Row	Key stages of the structural modeling pipeline
1	Detect plane segments $\{S_i\}$ in new keyframe, and collect existing visible plane features $\{P_i\}$ in this keyframe
2	Infer geometric relations G_S between detected segments $\{S_i\}$
3	For each segment S in $\{S_i\}$ P = existing plane feature in $\{P_i\}$ overlapping to S or new plane feature corresponding to S Add P to $\{P_i\}$ if P is new plane feature
4	Use G_S to infer plane relations G_P from P to visible plane features $\{P_i\}$ in this keyframe; Add G_P to constraint set
5	For all P_i which overlap P Merge P_i into P Delete P_i
6	Constraint optimization
7	Topological reconstruction

Fig. 3: Incremental structural modeling pipeline executed for each new keyframe. (Row 1): Within the new keyframe, we project the existing plane features (consisting of plane segments), and detect new plane segments. (Row 2): Between new and existing plane segment pairs, geometric relations are computed (e.g., plane segment pairs may be parallel or orthogonal). (Row 3): Plane segments are assigned to a plane feature. (Row 4): We infer plane relations between the plane features from the assigned plane segments. (Row 5): Overlapping plane features are detected and merged. (Row 6): Exploiting the geometric relationships between plane features, we perform a constrained global optimization on the geometric reconstruction. (Row 7): Finally, we derive a topological reconstruction from the geometric reconstruction.

Geometric reconstruction The geometric reconstruction step processes one new plane segment S at a time, with the goal of merging overlapping segments from one or more keyframes into *plane features*. A plane feature P is estimated from one or more overlapping segments. It contains a set of associated plane segments S, a border 3D polygon $\mathbf{b}$, and a plane equation $\mathbf{p}$. While segments are treated as immutable measurement data, the plane features are continuously estimated, which means they are built incrementally and subject to optimization.

The geometric reconstruction stage infers geometric relations between plane features, such as parallel, coplanar, overlapping, incident and orthogonal relations. Plane equations and keyframe locations are optimized to obtain an improved model. The geometric relations are added as geometric constraints to this optimization step. This step is done before the outlines of the plane features are updated, to ensure that the best geometric information is available for the last step.

Topological reconstruction The information from the geometric reconstruction is forwarded to the topological reconstruction. This stage is responsible for simplifying the borders of plane features and for constructing a manifold polygonal mesh from incidence relationships.

We illustrate the main ideas of the structural modeling system with a simple scene shown in Figure 4. In this example, we only use a single RGB-D frame as input (Figure 4(a)). The system detects plane segments in the scene (labeled with different colors in Figure 4(b)). Before newly detected segments are integrated into the global model, geometric relations between segments are recovered (Figure 4(c)). Parallel segments are rendered with the same color. For pairs of incident and orthogonal segments, an orange intersection line is displayed. These segments are integrated into the global model as new plane features with common edges and corners (Figure 4(d)). Incident edges are rendered with arrows. Right angle corners between two consecutive edges on planes are highlighted with L shapes on the associated corners' position. Incident and orthogonal planes are highlighted with orange spheres on the associated incident lines. Finally, occluding, unbounded plane borders are further simplified. This results in a clean polyhedron (Figure 4(e)), comprising a few simple polygons. Simplified edges are rendered with arrows.

4 Plane Detection

Plane detection is the first stage of the pipeline and contributes the basic measurements. Each keyframe is analyzed independently without any prior information. The plane detection module creates a segmentation of the depth map into a set of contiguous planar regions and jointly estimates the plane equation for each component and the set of pixels belonging to it. It uses an inverse depth parametrization for the plane equation in camera coordinates, as described by Erdogan *et al*. [6], but applies a greedy algorithm instead of a global optimization for reasons of speed. We only look at the depth map values, while ignoring RGB data, since RGB is not very reliable for the Kinect sensor we used in our experiments. See Figure 5 for an example of the results of plane detection on a single frame.

In inverse depth or disparity representation, a visible plane – which must not contain the origin – can be described using three coefficients (a,b,c), so that for a pixel (u,v) with depth value z, the following equation holds

$$\frac{1}{z} = au + bv + c \tag{1}$$

Thus, we have a linear constraint for the plane coefficients from the inverse depth at a single pixel. Plane fitting can be expressed as a linear least squares problem of the form

$$\underset{(a,b,c)}{\operatorname{argmin}} \left\| \begin{pmatrix} u_1 & v_1 & 1 \\ u_2 & v_2 & 1 \\ \vdots & \vdots & \vdots \\ u_n & v_n & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} - \begin{pmatrix} \frac{1}{z_1} \\ \frac{1}{z_2} \\ \vdots \\ \frac{1}{z_n} \end{pmatrix} \right\|^2 \tag{2}$$

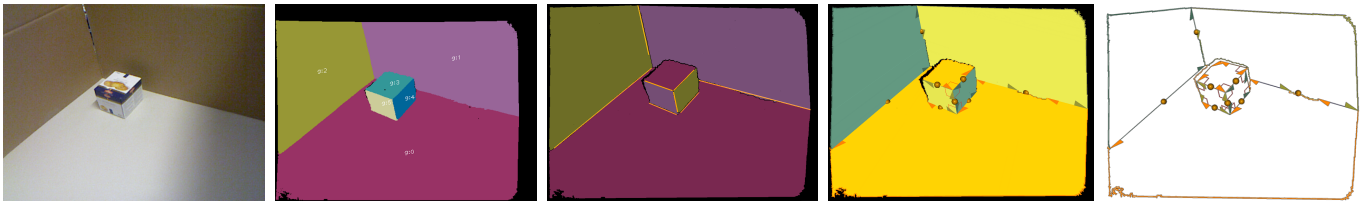
The estimation of the plane coefficients (a,b,c) proceeds incrementally, by updating the normal equations for each new inverse depth measurement and recomputing the solution. Overall, this is a very fast operation per pixel, because we only need to solve a linear system of rank 3.

The plane estimation is embedded in a greedy region growing operation. A set of seed locations is used to start new components, and an initial plane estimation is computed from the immediate four neighbors of the seed location. The set of pixels S for the current component is initialized with the seed location and its neighbors. All 4-connected neighbors are added to the candidate locations C.

For each candidate in C, the inverse depth error to the estimated plane is computed using equation 1. If it is below a threshold, the location is added to S, and the plane parameters are updated from equation 2. New 4-connected neighbors of the pixel are added to C. This flood fill operation continues, until C is empty and no new pixels can be added anymore. Note that the threshold is given in disparity space, therefore, it represents a uniform noise threshold independent of the depth for any stereo-based depth measurement system. This holds for structured light sensors, such as the Kinect, as well as for stereo camera pairs.

Seed locations that already belong to a set S are immediately rejected and do not generate new components. If a new neighbor already belongs to another component S', its distance to the original component is compared to the distance to the new component S. If it is closer to S than to S', it is reassigned to S, and both plane estimates are updated.

After the joint segmentation and plane estimation, we perform two cleanup steps. First, small holes and isolated pixels are removed by reassigning pixels to the component that is assigned to most of their 4-connected neighbors. This also simplifies the outline of components. Second, a connected component algorithm is run on the labeled pixels to identify individual components. While the initial region growing



(a) (b) (c) (d) (e)

Fig. 4: Geometric Reconstruction Example. (a): RGB-D input frame. (b): Detected plane segments labeled in different colors. (c): Inferred initial geometric relations between segments. (d): Planes reconstructed with common edges or corners snapped. (e): Simplified occluding boundary.

(a) (b) (c) (d)

Fig. 5: Plane Detection Example. (a): RGB-D input frame. (b): Detected plane regions labeled in different colors. (c): Input frame overlaid with transparent plane regions. (d): Error heatmap indicating the per-pixel error.

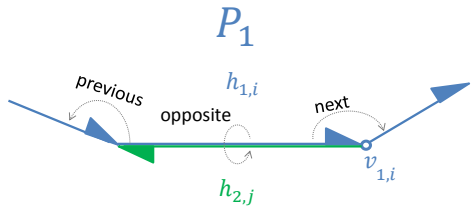


Fig. 6: Halfedge concepts and the internal data structure used in our implementation.

algorithm always produces connected components, the re-assignment step can break up a single component again. Therefore, the outcome of the connected component analysis provides us with a true segmentation into individual components. A final plane estimate is computed for each segment.

For further processing, we compute a polygonal representation of the *border* of each segment S. A border consists of one *outer boundary* and potentially multiple *inner boundaries*, representing holes in the component. Because the individual components are 4-connected, all boundaries are simple polygons, i.e., they may be concave, but not self-intersecting. The 2D polygons are reprojected onto the estimated plane to provide a 3D polygon in camera coordinates. The complete output for each segment S comprises the plane equation, a set of pixels belonging the component, including a neighbor graph listing neighbor components, and the 3D border $\mathbf{b}$.

We ensure that our border representation is a simple polygon with holes, and there is no self-intersection in the boundary or between the boundaries. We found that self-intersections emerge from single-pixel paths identified in the connected component analysis. To prevent single-pixel paths, we add surrounding pixels for those paths in the individual connected components.

5 Geometric Reconstruction

The objective of this stage is to robustly compute plane features P_i. This requires computing their border, merging overlapping plane segments into common planes and inferring geometric relationships. We write $S_{i,j,k}$ for a plane segment that belongs to plane feature P_i and was detected in keyframe K_j. Note that multiple such segments can exist per keyframe, hence the need for index k.

Initially, segments are not associated with a plane feature yet. We use the special index $i = \emptyset$ to denote this case. Thus, $S_{\emptyset,j,k}$ denotes a segment detected in keyframe K_j but not assigned to any plane feature yet.

5.1 Halfedge data-structure

We use a *halfedge data-structure* (see Fig. 6) to store all polygons and any relationships between them. Furthermore, halfedges are used to represent the plane border polygon of a plane feature and the geometric relations between plane features. A *plane border polygon* is a 3D polygon with vertices on the associated plane. The polygon is stored as a double-linked list of 3D halfedges. A halfedge has one *previous* halfedge and one *next* halfedge. When two polygons share a common edge, the halfedge has an *opposite* halfedge in another polygon. A halfedge $h_{1,i}$ in plane P_1 is represented as a 3D arrow, which has one vertex $v_{1,i}$ at the head of the arrow. The arrow indicates the order of the plane border, counterclockwise with respect to the plane's normal.

5.2 Segment relations

The first step of the geometric reconstruction is to find geometric relations between the segments $S_{\emptyset,j,k}$. We distinguish between *parallel*, *coplanar*, *incident*, and *incident-orthogonal* relations. We search for relations between pairs of segments of the same keyframe K_j:

Parallel: We are looking for parallel segments, e.g., located on the floor and on the ceiling. We consider two segments as parallel in 3D space, if the angle between their normals is below a threshold (e.g., 3°).

Coplanar: We are looking for segments which are not only parallel, but also located on the same 3D plane, e.g., two segments which are located on a floor. First, the segments must be parallel. Second, to ensure that the two segments lie on the same 3D plane, more than 90% of the border vertices of one segment must lie on (or near) the 3D plane of the other segment, and vice versa. We consider a border vertex to lie on a given segment plane, if the point-to-plane distance is less than a given threshold (e.g., 1 cm).

Incident: We are seeking segments which border at each other (note that segments cannot have an overlap, but may merely "touch" each other), e.g., located at the intersection edge of the floor and a wall. We consider two segments as incident, if they share a common edge, which we call an *incident edge*. To determine the incident edge, we first collect touching pairs of border vertices of the two segments, which are neighbors in pixels space. For instance, segment $S_{1,j,1}$ and segment $S_{2,j,1}$ have touching border points $(b_{1,j,1}, b_{2,j,1})$ (see Figure 7

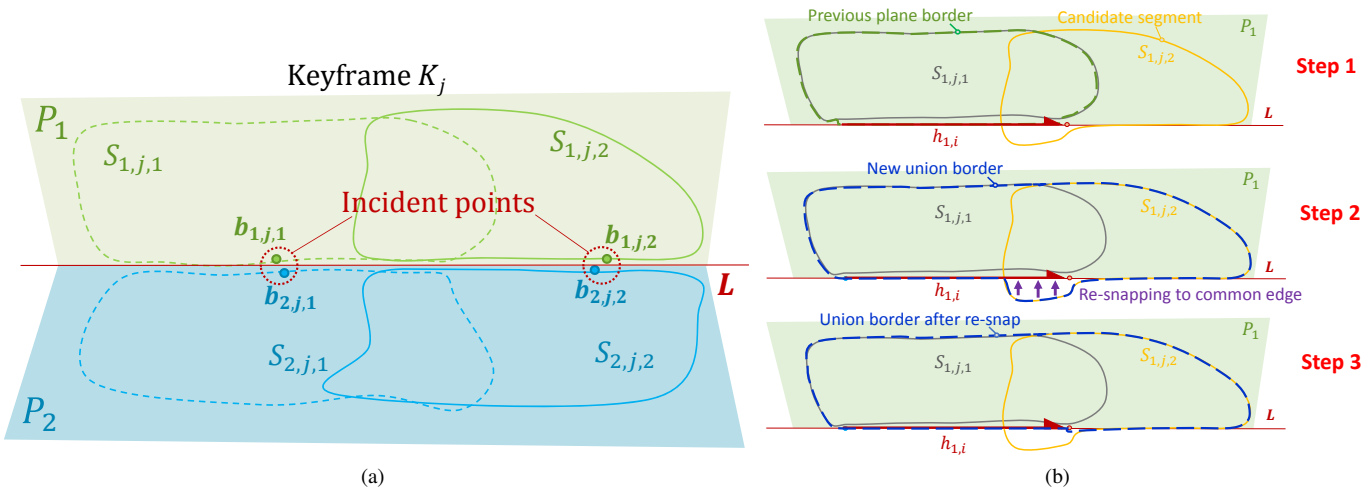


Fig. 7: Segment Integration. (a): Two planes with two overlapping segments each. All segments are from the same keyframe (although this is not required). Two incident points are highlighted. (b): The polygon border is computed as the union of the segment borders, when integrating segment $S_{1,j,2}$ to plane feature P_1. The union polygon is re-snapped to incident line.

(a)). We compute the 3D plane intersection line between the two segments. We verify this line by computing the support from the border vertices of the segments. For that purpose, we compute the line-point distance of all touching border vertices in 3D space. The plane intersection line is considered as an incident edge, if a sufficient number of touching border vertices (e.g., 50) is located within a threshold distance from that line (e.g., 1cm). Border vertices which support the incident edge are called *incident points*.

Incident-orthogonal: We are looking for segments which are not only incident, but also orthogonal to each other, such as segments located on the edge of the floor and a wall. Two segments are considered incident-orthogonal, if they are incident, and their normals are orthogonal up to a threshold (e.g., 3°).

These segment-segment relations are later used to infer relations between associated plane features of the segments. We found that using segment relations to infer plane relations is much more robust than directly checking plane feature relations. Additionally, it is easier to select uniform error thresholds (angular error and distance error) for extracting segment relations.

5.3 Segment integration

For the integration of segments, we check if a candidate segment $S_{1,j,2}$ overlaps an existing visible plane feature P_1 in this keyframe (see Figure 7 (b)). Segment $S_{1,j,2}$ overlaps plane feature P_1, if there is an intersection between the 2D polygon projection of the segment's border on the plane feature and the 2D polygon of the plane feature's border. Otherwise, if there is no overlap with any known plane feature, a new plane feature is created from the single segment.

If a segment $S_{1,j,2}$ is overlapping with a plane feature (see Section 5.2), we integrate this segment into the plane feature. The integrating operation includes three steps: preparation, union border computation, and re-snapping.

In the preparation step (step 1 in Figure 7 (b)), we store the existing state of the plane feature for rolling back, should one of the subsequent steps fail.

In the union border computation, we compute a new polygon border as the union of the current plane feature's border and a segment's border. Since the borders are simple polygons with holes, the union must also be a simple polygon, but the number of holes can change (step 2 in Figure 7 (b)).

The union border ignores information on incident edges that are shared with another existing plane feature and can move a border over the incident line. Therefore, in the final step, we re-snap the new union border to existing incident edges of the plane feature (step 3 in Figure 7 (b)). We proceed to test if the newly re-snapped union border consists of simple polygons with holes and there is no intersection between the outer polygon and any holes. If the test fails, we revert the integration operation to the stored state from the first step and put this segment on hold for later processing.

5.4 Plane relations

By now, all segments of the new keyframe K_j are either integrated into an existing plane feature or have created new plane features. Next, we update the geometric relations between all plane features visible in the keyframe. This includes both newly created plane features and existing ones that are visible.

For each pair of plane features P_1, P_2 we use segment-segment relations to infer new or update existing geometric relations between the plane features. For instance, P_1 and P_2 are *parallel*, if there exists a *parallel* relation between a segment $S_{1,j,k}$ of P_1 and a segment $S_{2,j,l}$ of P_2. Note that segments $S_{1,j,k}$ and $S_{2,j,l}$ belong to the same keyframe K_j. Similar to segment-segment relations, we infer the following plane-plane relations: *parallel*, *coplanar*, *incident*, and *incident-orthogonal*.

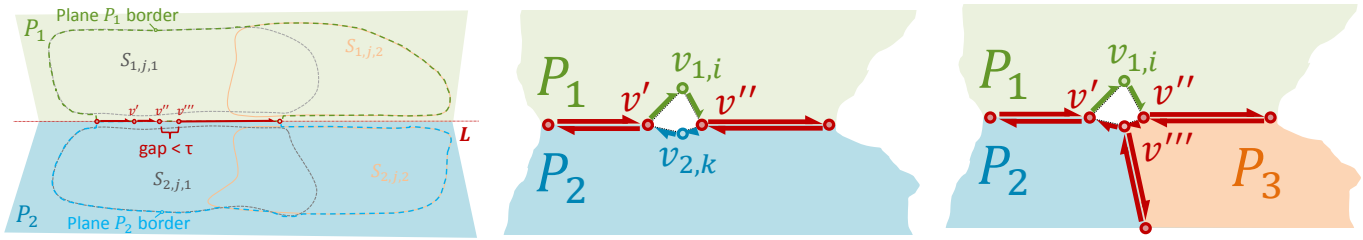
5.5 Plane feature merging

Next, we check if two plane features overlap. We first check for the existence of a *coplanar* relation between a segment $S_{1,j,k}$ of plane feature P_1 and a segment $S_{2,j,l}$ of plane feature P_2. If there is at least one large enough border interval of P_1 is inside P_2 (including the border of P_2), the planes are considered overlapping. A border interval is a consecutive sequence of halfedges along a border. A border interval is inside the border of another plane, if all the points of the plane are inside the border. A point is considered to be inside of a plane, if its normal projection onto that plane lies within the border polygon.

If two plane features are overlapping, we perform a merging operation on the plane features. The two plane features are replaced by a common plane feature that contains the union of both plane features' segments. The merging operation of a second plane feature with a first plane feature is equivalent to integrating all segments of the second plane feature into the first plane feature.

Similar to integrating a segment (see section 5.3), a union border of all new segment borders is computed and snapped against known incident edges of both original plane features. This step ensures that incident edges and the reciprocal relationship of halfedges between planes are still valid.

After plane merging, it is necessary to re-check for geometric relations. Existing relations may have changed, and new ones may have emerged as a result of the new larger plane feature.



(a) (b) (c)

Fig. 8: Merging of edges. (a): Halfedges along incident vertices v', v'', v''' can be merged, since v' and v'' are consecutive, and the gap width between v'' and v''' is smaller than τ. (b): The gap width $w_1 = |v_{1,i} - v'| + |v'' - v_{1,i}|$ is larger than τ, but not too large. The difference $|w_2 - w_1|$ to the backward walking distance $w_2 = |v_{2,k} - v''| + |v' - v_{2,k}|$ is small enough, so that the gap can be closed. (c): The backwards walking encounters a point v''' incident between P_2 and P_3. Since the incident points v', v'' and v''' are all close enough, they can be merged into a new corner point. We use v', v'', v'' for incident vertices to simplify notation. The vertices $v_{1,i}, v_{2,k}$ are halfedge vertices, as explained in Section 5.1 and Figure 6.

5.6 Constraint optimization

Finally, we improve the accuracy of the reconstructed model by optimizing the plane equations and keyframe camera poses. The optimization minimizes a geometric error between the measured segments and the associated plane features. In addition, it also observes the geometric constraints that were found in the prior steps.

The constraint optimization can be seen as an incremental bundle adjustment approach, similar to recent work on map optimization for SLAM. We must assume that the camera poses associated with the depth keyframes suffer from non-negligible amounts of drift, which affects the robustness of recovering the higher level geometric features of the structural model. Therefore, we jointly optimize camera poses $\mathbf{T}_j$ of keyframes K_j and the plane equation parameters $\mathbf{p}_i$ of plane features P_i in a graph-based approach [12]. We write $\mathbf{T} = (\mathbf{T}_1, ..., \mathbf{T}_n)$ for the vector of keyframes transformation parameters and $\mathbf{p} = (\mathbf{p}_1, ..., \mathbf{p}_m)$ for the vector of plane features equations.

We use the well-known least-squares Ceres solver [1] to perform the optimization.

The cost function C is the weighted sum of cost functions representing the different types of constraints: camera constraints (C_C), plane observation constraints (C_O) and plane relation constraints (C_R):

$$C = w_\mathrm{C} \cdot C_\mathrm{C} + w_\mathrm{O} \cdot C_\mathrm{O} + w_\mathrm{R} \cdot C_\mathrm{R} \tag{3}$$

Camera constraints A constraint exists between two camera poses $\mathbf{T}_j, \mathbf{T}_{j'}$, if these cameras have an overlapping view.

$$C_\mathrm{C}(\mathbf{T}) = \sum_{j,j'} \rho(log_{se3}((\mathbf{T}_j^{-1} \cdot \mathbf{T}_{j'}) \cdot \mathbf{R}_{j,j'})) \tag{4}$$

where $\mathbf{R}_{j,j'}$ is relative transformation from keyframe K_j to keyframe $K_{j'}$, $log_{se3}(\cdot)$ is the inverse of the exponential maps for $SE3$, and ρ is a robust metric minimizing a weighted combination of the residual translation and rotation. The relative transformations $\mathbf{R}_{j,j'}$ are treated as immutable measurements, since they are computed directly between two keyframes using intensity-depth alignment.

Plane observation constraints This term aligns measured plane segments to their corresponding plane feature. For such an alignment, we use the border polygons $\mathbf{b}_{i,j,k}$ of all plane segments $S_{i,j,k}$ which belong to plane feature P_i and keyframe K_j. Our cost function ensures that vertices $v \in \mathbf{b}_{i,j,k}$ of the plane segments are coplanar with the corresponding plane feature.

$$C_\mathrm{O}(\mathbf{p}, \mathbf{T}) = \sum_i \sum_{v \in \mathbf{b}_{i,j,k}} \| \mathbf{p}_i \begin{pmatrix} \mathbf{T}_j v \\ 1 \end{pmatrix} \|^2 \tag{5}$$

Plane relation constraints The plane relation constraint is a sum of constraints on incident planes (C_I), orthogonal planes (C_T) and parallel planes (C_P).

$$C_\mathrm{R} = C_\mathrm{I} + C_\mathrm{T} + C_\mathrm{P} \tag{6}$$

For unordered pairs of incident or incident-orthogonal planes $(P_i, P_{i'})$, we store the incident border points. Incident constraints ensure that the incident border points of incident planes move along the intersection line of the two planes during optimization. Let $\mathbf{b}_{i,j,k}$ be the set of incident border points of segments $S_{i,j,k}$ of plane P_i, and $\mathbf{b}_{i',j',k'}$ be the set of incident border points of segments $S_{i',j',k'}$ of plane $P_{i'}$ (Figure 7). Then, the incident constraint cost function ensures that the incident border points are coplanar with both planes $(P_i, P_{i'})$.

$$C_\mathrm{I}(\mathbf{p}, \mathbf{T}) = \sum_{v \in \mathbf{b}_{i,j,k}} \| \mathbf{p}_{i'} \begin{pmatrix} \mathbf{T}_j v \\ 1 \end{pmatrix} \|^2 + \sum_{v' \in \mathbf{b}_{i',j',k'}} \| \mathbf{p}_i \begin{pmatrix} \mathbf{T}_{j'} v' \\ 1 \end{pmatrix} \|^2 \tag{7}$$

Equation 7 expresses that incident border points of plane P_i must lie on plane $P_{i'}$ and vice versa. Note that this equation does not enforce that the incident points lie in their associated plane, because Equation 5 already enforces such a coplanarity constraint.

For pairs $(P_i, P_{i'})$ of orthogonal planes, the dot product of the associated normal vectors $\mathbf{n}_i, \mathbf{n}_{i'}$ should be zero. Likewise, for pairs $(P_i, P_{i'})$ of parallel planes, the cross product of the associated normal vectors should be zero.

$$C_\mathrm{T}(\mathbf{p}) = \sum_{i,i'} \| \mathbf{n}_i \cdot \mathbf{n}_{i'} \|^2 \tag{8}$$

$$C_\mathrm{P}(\mathbf{p}) = \sum_{i,i'} \| \mathbf{n}_i \times \mathbf{n}_{i'} \|^2 \tag{9}$$

6 Topological Reconstruction

This stage constructs an explicit topological geometry representation of the scene. The border of each plane feature will be simplified, and incident relationships are used to connect polygons into a manifold mesh.

6.1 Common edge snapping

Common edge snapping is applied to planes P_1 and P_2 if they are incident or incident-orthogonal.

First, we check for incident edges between the borders of P_1 and P_2. Two incident edges overlap, if they are either consecutive, or if the gap width between any incident edges is very small (see Figure 8 (a)). A gap consists of one or multiple edges, which are not incident, i.e., the neighboring plane identified by the opposite halfedge is nil. The gap width is the cumulative edge length. If the gap width is below a small threshold τ (empirically set to 1cm), the incident edges are merged.

Next, we iterate over all merged incident edges for both planes. As discussed in Section 5.5, we snap all segment border vertices which are close enough to an incident edge of their plane border onto that edge.

6.2 Common incident edges with gaps

Reconstruction errors can create gaps between incident edges that are small, but not negligible. In these cases, it is likely that the incident edges still belong to a single continuous edge. If the gap width is within a preset range (empirically set to 1cm-5cm), we apply additional checks to determine if the gap may be closed (Figure 8 (b)).

We compare the gap width in forward and backward direction by walking along the halfedges in either direction between the incident edges. If the forward and backward gap width are the same up to a threshold (again, we use 1cm), the gap is closed by merging the incident vertices.

6.3 Common vertices from at least three planes

Sometimes, the backward walking described above may reveal that the backward gap sequence contains halfedges that point to a third incident plane P_3. In this case, we could be facing a corner at the intersection between three or more planes. This is particularly important for artificial objects with regular structures and strong corners, such as furniture.

This situation can be identified by looking for a cycle in the forward-backward walking (Figure 8 (c)). Assume that the common corner between the three planes P_1, P_2, P_3 is missing. We start from an incident edge between P_1 and P_2 and walk along a gap. After the gap, we encounter an incident edge between P_1 and P_3. We walk backwards and continue walking forward-backward, until we either encounter the original incident edge again, or the permissible gap width is exceeded. All endpoint vertices of the incident edges encountered along this walk must be sufficiently close, so they can be merged into one corner vertex. Moreover, the corner vertex must be incident to all participating planes.

6.4 Polygon fitting

Topological reconstruction simplifies the reconstructed geometry and produces sound polygonal model in most cases. However, non-incident border intervals remain, in particular, if we are lacking sufficient observations in the keyframes. These omissions are mainly caused by limited camera movement during the SLAM mapping or by occlusion.

There are two types of non-incident border intervals: *Occluding* intervals are intervals corresponding to halfedges pointing to a plane feature closer to the camera of the keyframe in which they were observed; *occluded* intervals are the opposite. In this section, we propose a method to tackle occluding intervals.

The overarching idea is to employ polygon fitting to further simplify an occluding interval V of plane P. An optimal polygon must agree with all observed segment borders corresponding to V, while minimizing its number of vertices. We search through all segments S associated with P. For each vertex $\mathbf{v}_i$ of interval V, we search for corresponding border vertices $\mathbf{b}_j \in S$, which are occluding. For convenience, we write s_i for a line segment $(\mathbf{v}_i, \mathbf{v}_{i+1})$.

We use expectation maximization (EM) to estimate the positions of the vertices $\mathbf{v}_i$, such that the distances of the points $\mathbf{b}_j$ to the border interval V are minimal. EM alternates between assigning observed points $\mathbf{b}_j$ to collinear groups in the E-step and estimating vertices $\mathbf{v}_i$ in the M-step. This is done over multiple iterations, until convergence.

In the E-step, we assign observed points to collinear groups by computing the likelihood $p_{i,j}$ of a point $\mathbf{b}_j$ belonging to s_i as

$$p_{i,j} = p(\mathbf{b}_j \in s_i \mid \mathbf{b}_j, s_i) \propto exp\left(\frac{-dist(s_i, \mathbf{b}_j)^2}{2\sigma^2}\right) \tag{10}$$

where $dist(s_i, \mathbf{b}_j)$ is the perpendicular distance between point $\mathbf{b}_j$ and line segment s_i, and σ^2 is the variance of these distances. This equation assumes that the error follows a normal distribution. After each E-step, the likelihoods are normalized, such that $\sum_j p_{i,j} = 1$. At the start, we initialize the likelihoods by setting $p_{i,j} = 1$, if the point $\mathbf{b}_j$ is in the group corresponding to s_i.

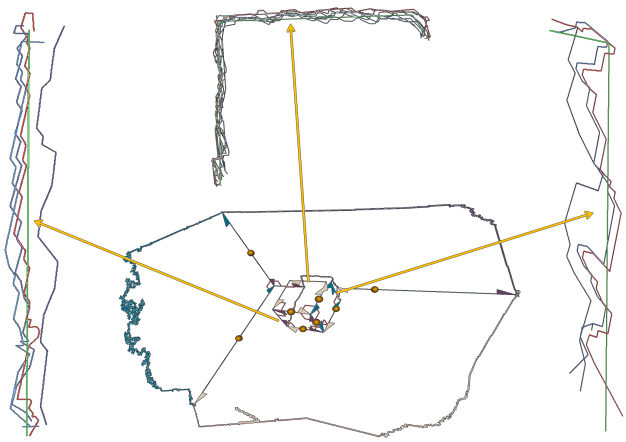


Fig. 9: Results of the polygon fitting step. Green polygons (top, left, right) represent final polygon fitted to non-incident edges in the reconstructed model (center). The other colored polygons (top, left, right) are contours of the plane's segments occluding borders in different keyframes.

In the M-step, we minimize the objective function $O(\mathbf{s})$, where $\mathbf{s} = (s_1, ..., s_n)$ is a vector of parameters.

$$O(\mathbf{s}) = \sum_{i,j} p_{i,j} \left\| dist(s_i, \mathbf{b}_j) \right\|^2 \tag{11}$$

Such minimization is to maximize the likelihood function $L(\mathbf{s})$:

$$L(\mathbf{s}) = \prod_{i,j} p(\mathbf{b}_j \mid s_i) \tag{12}$$

Note that the assignment likelihood $p_{i,j}$ is constant in the M-step.

To reduce the number of vertices of the simplified polygon, if three consecutive, collinear vertices are found, such redundant vertices are removed in each iteration of the M-step.

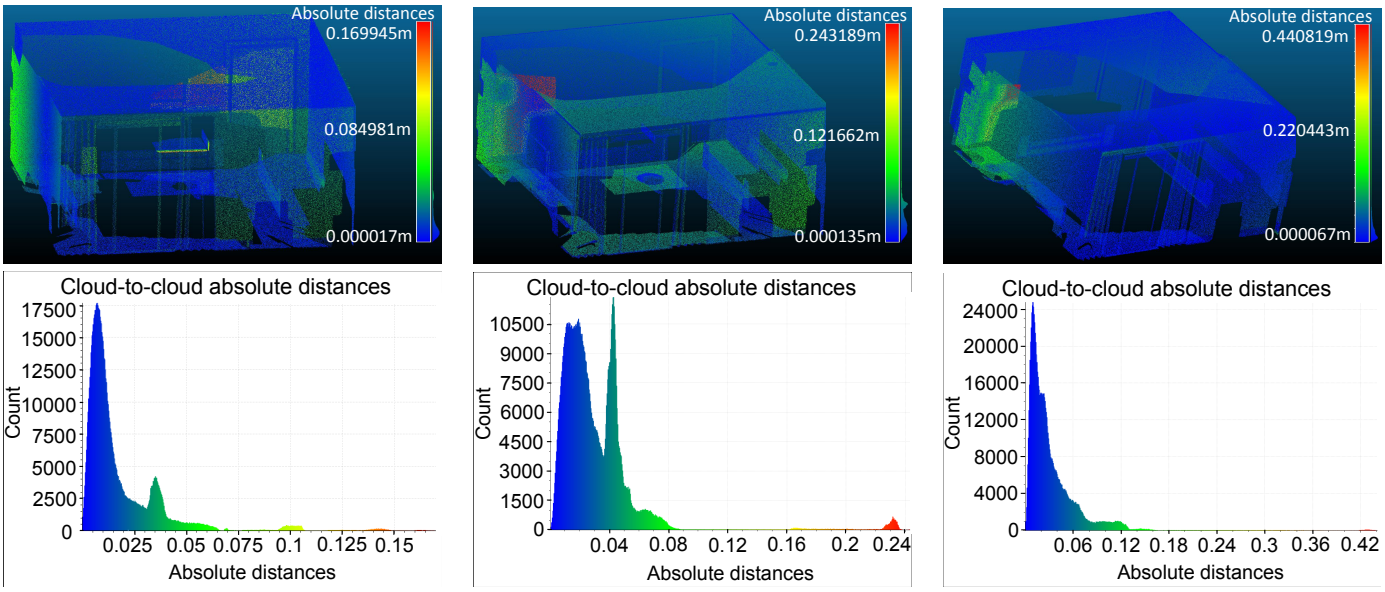
7 Evaluation

7.1 Quantitative evaluation

First, we study the accuracy of our proposed approach. We used the ICL-NUIM dataset [9] to compare the reconstruction results to a known ground-truth model. We added Gaussian noise of increasing standard deviation to the reference depth maps. Camera poses were obtained by tracking the input sequence using the RGB-D SLAM system. Then, we sampled both the reconstructed polygonal model as well as the ground-truth model into point clouds to compute point-wise distances between the two models.

As shown in Figure 10, our method achieves reasonably high accuracy. Overall, 80% of the surface area of the reconstructed model agrees to the ground-truth model within 2cm. However, the plane segmentation can over-fit nearly – but not entirely – planar surface areas, and erroneously simplify these into planes. Even though our proposed optimization can compensate for some drift from SLAM, significant amounts of drift are still hard to handle.

Furthermore, we also validated the correctness of the topological reconstruction. For this evaluation, we manually check correctness of topological relations between planar surfaces in the selected scene (Table 1). In a few cases, our method is not able to reconstruct the topology correctly. This is mostly due to a lack of observations available from a given camera trajectory. For instance, the right brown wall (green plane in column 3) of office scene #2 in Figure 12 (f) is always more than 2m away from given camera trajectory. The observed depth data in that region is very noisy. As a consequence, the system fails to reconstruct correct topology in this area. However, given sufficient quality and quantity of observations, the topology reconstruction is highly reliable.



(a) (b) (c)

Fig. 10: **Top:** Error heatmaps of the reconstructed models. **Bottom:** Color-coded distance error distribution when comparing the sampled point clouds of reconstructed models and ground-truth model. (a): 2mm depth-noise magnitude. (b): 10mm depth-noise magnitude. (c): 20mm depth-noise magnitude.

Scene	Incidence	InciOrtho	Parallel
Living room (dataset from [9])	7/9	33/36	25/27
Table scene (Figure 12 (c))	13/18	27/29	32/34
Corridor (Figure 12 (d))	0	5/8	5/5
Office scene #1(Figure 12 (e))	35/56	33/38	57/67
Office scene #2 (Figure 12 (f))	18/23	25/26	24/24
Avg. correctness	77.91%	86.05%	94.36%

Table 1: Number of correctly recovered topology relations per all existing topology relations. The existing relations were counted manually for the data sets, summarizing the results of three different topology relations. For incident and orthogonal-incident, we count number of correct relations per all associated relations. For parallel relations, we count number of planes assigned as parallel plane with another plane, instead of counting plane-to-plane relations, and check the number of correct parallel planes.

Table 1 demonstrates that our topology construction is generally robust. The proposed topology construction recovers on average 86% of all relations. The ground-truth relations were hand-annotated in the data sets. To our knowledge, this is the first attempt to evaluate topology construction. However, topology is a qualitative concept, and our results should, therefore, only be interpreted as a demonstration of feasibility.

7.2 Qualitative evaluation

Besides accuracy, we also validate our method in terms of quality. As illustrated in Figure 12, our method performs well in a wide range of indoor conditions. Each row shows the results of a different scene, with increasing complexity. The scenes range from simple scenes, comprising a box in a corner, to room-sized scenes. The method is able to reconstruct high-level geometric models of the scenes. Moreover, the topological information of the reconstructed models is correct in most of the cases.

Some restrictions are owed to the underlying SLAM system. As mentioned in previous sections, our SLAM system uses both color and depth information for camera pose estimation. This works well for scenes with a lot of local geometric structure and surface textures. Our structural modeling does not rely on such detailed local structures. In fact, the best results are obtained when viewing larger, flat surfaces, such as furniture or walls. As long as the underlying SLAM system, which is independent from our structural modeling, does not lose camera tracking, the structural modeling can handle a wide range of scene details (Figure 12).

8 Augmented Reality Application

We demonstrate a simple AR application, which superimposes virtual content on rectangular surface in the environment (see Figure 11). In order to achieve this, we check for possible rectangles in the reconstructed plane features of the structural model. We search for a plane with four incident edges and check for right angle corners. If these conditions are meet, we consider the plane to be a rectangle. Given a gravity vector input, we then apply up-right visualization of virtual content on the discovered rectangles.

9 Conclusion

We present an automatic structural modeling system, which can reconstruct high-level polygonal models with proper geometry and topology. Our system exploits a SLAM approach, which allows to track the camera and operate incrementally. Thereby, an AR application can be extended with scene understanding.

Our system extracts plane segments from SLAM keyframes using real-time plane detection and integrates them into a global model using automatically detected geometric constraints for improved accuracy. Furthermore, we also reconstruct the topology the model. This enables further simplification to obtain a geometric model consisting of a small set of polygons. For validation, we conducted a quantitative evaluation on a recently published dataset [9]. The results confirmed that our system can deliver high-level geometric models with good accuracy.

Our approach heavily depends on the underlying SLAM system. Therefore, it suffers when camera pose estimation is poor. As pointed out in the results section, RGB-D SLAM works best in highly cluttered scenes, which provide enough information for good depth image alignment. Scenes with many flat areas are very suitable for the structural modeling, but less suitable for RGB-D SLAM. However, our approach can work with any kind of real-time camera tracking approach, and, thus, we will be able to benefit from further improvements in SLAM systems.

For future work, we aim to formalize the geometric relations inference using a suitable probabilistic model. Moreover, using probabilistic models will allow easier integration of priors into the system. Such priors can be drawn from assumptions of the scene or from user input in an interactive application setup.

Fig. 11: Simple AR application. We render virtual content on rectangle surfaces in the scene, as shown from different view points. **Top:** Tracking and augmenting a single box. **Bottom:** Tracking and augmenting multiple boxes simultaneously.

We also plan to further examine occluded areas, which appear as holes in the final reconstructed polygons. For example, objects standing on a table occlude parts of the table surface. Further careful investigation can find an appropriate solution to fill those holes.

Reconstructing a wider range of geometric primitives is another interesting direction we would like to pursue. Primitives such as cylinders, spheres, and boxes are common in our environment. Thus, adding reconstruction of these primitives to plane primitives would produce more complete geometric models of the environment.

Acknowledgments

This work was partially funded by the Christian Doppler Laboratory for Handheld Augmented Reality and Qualcomm Technologies Inc. We would like to thank Christian Pirchheim, Clemens Arth and Vincent Lepetit for their support.

References

[1] S. Agarwal, K. Mierle, and Others. Ceres solver. http://ceres-solver.org.

[2] M. Arikan, M. Schwärzler, S. Flöry, M. Wimmer, and S. Maierhofer. O-snap: Optimization-based snapping for modeling architecture. *ACM Transactions on Graphics*, 32:6:1–6:15, Jan. 2013.

[3] P. Bunnun and W. Mayol-Cuevas. Outlinar: an assisted interactive model building system with reduced computational effort. In *ISMAR*, pages 61–64, Sept 2008.

[4] D. Chekhlov, A. Gee, A. Calway, and W. Mayol-Cuevas. Ninja on a plane: Automatic discovery of physical planes for augmented reality using visual slam. In *ISMAR*, November 2007.

[5] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *Proc. SIGGRAPH '96*, pages 11–20, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

[6] C. Erdogan, M. Paluri, and F. Dellaert. Planar segmentation of rgbd images using fast linear fitting and markov chain monte carlo. In *Computer and Robot Vision*, 2012.

[7] Y. Furukawa, B. Curless, S. Seitz, and R. Szeliski. Reconstructing building interiors from images. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 80–87, Sept 2009.

[8] A. Gupta, A. A. Efros, and M. Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *ECCV*, 2010.

[9] A. Handa, T. Whelan, J. McDonald, and A. Davison. A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In *ICRA*, pages 1524–1531, May 2014.

[10] C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for rgb-d cameras. In *Proc. of the Int. Conf. on Intelligent Robot Systems (IROS)*, 2013.

[11] Y. M. Kim, J. Dolson, M. Sokolsky, V. Koltun, and S. Thrun. Interactive acquisition of residential floor plans. In *Proc. ICRA 2012*, pages 3055–3062, 2012.

[12] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *ICRA*, Shanghai, China, May 2011.

[13] K. Lai, L. Bo, X. Ren, and D. Fox. Detection-based object labeling in 3d scenes. In *ICRA*, 2012.

[14] D. C. Lee, A. Gupta, M. Hebert, and T. Kanade. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. *NIPS*, 24, November 2010.

[15] Y. Li, X. Wu, Y. Chrysanthou, A. Sharf, D. Cohen-Or, and N. J. Mitra. Globfit: Consistently fitting primitives by discovering global relations. *ACM Transactions on Graphics*, 30(4):52:1–52:12, 2011.

[16] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. *ISMAR*, 0:127–136, 2011.

[17] T. Nguyen, R. Grasset, D. Schmalstieg, and G. Reitmayr. Interactive Syntactic Modeling With a Single-Point Laser Range Finder and Camera. In *ISMAR*, Adelaide, SA, Australia, 2013. IEEE Computer Society.

[18] R. Salas-Moreno, B. Glocken, P. Kelly, and A. Davison. Dense planar slam. In *ISMAR*, pages 157–164, Sept 2014.

[19] R. Salas-Moreno, R. Newcombe, H. Strasdat, P. Kelly, and A. Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *CVPR*, pages 1352–1359, June 2013.

[20] R. Schnabel, P. Degener, and R. Klein. Completion and reconstruction with primitive shapes. *Computer Graphics Forum (Proc. of Eurographics)*, 28(2):503–512, Mar. 2009.

[21] T. Shao, W. Xu, K. Zhou, J. Wang, D. Li, and B. Guo. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Trans. Graph.*, 31(6):136:1–136:11, Nov. 2012.

[22] S. N. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys. Interactive 3d architectural modeling from unordered photo collections. In *Proc. Siggraph Asia*, December 2008.

[23] A. Trevor, J. Rogers, and H. Christensen. Planar surface slam with 3d and 2d sensors. In *ICRA*, pages 3041–3048, May 2012.

[24] A. van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. H. S. Torr. Videotrace: Rapid interactive scene modelling from video. *ACM Trans. Graph.*, 26(3):86–90, July 2007.

[25] A. van den Hengel, R. Hill, B. Ward, and A. Dick. In situ image-based modeling. *ISMAR*, 0:107–110, 2009.

[26] T. Whelan, L. Ma, E. Bondarev, P. de With, and J. McDonald. Incremental and batch planar simplification of dense point cloud maps. *Robotics and Autonomous Systems*, 69:3–14, 2015.

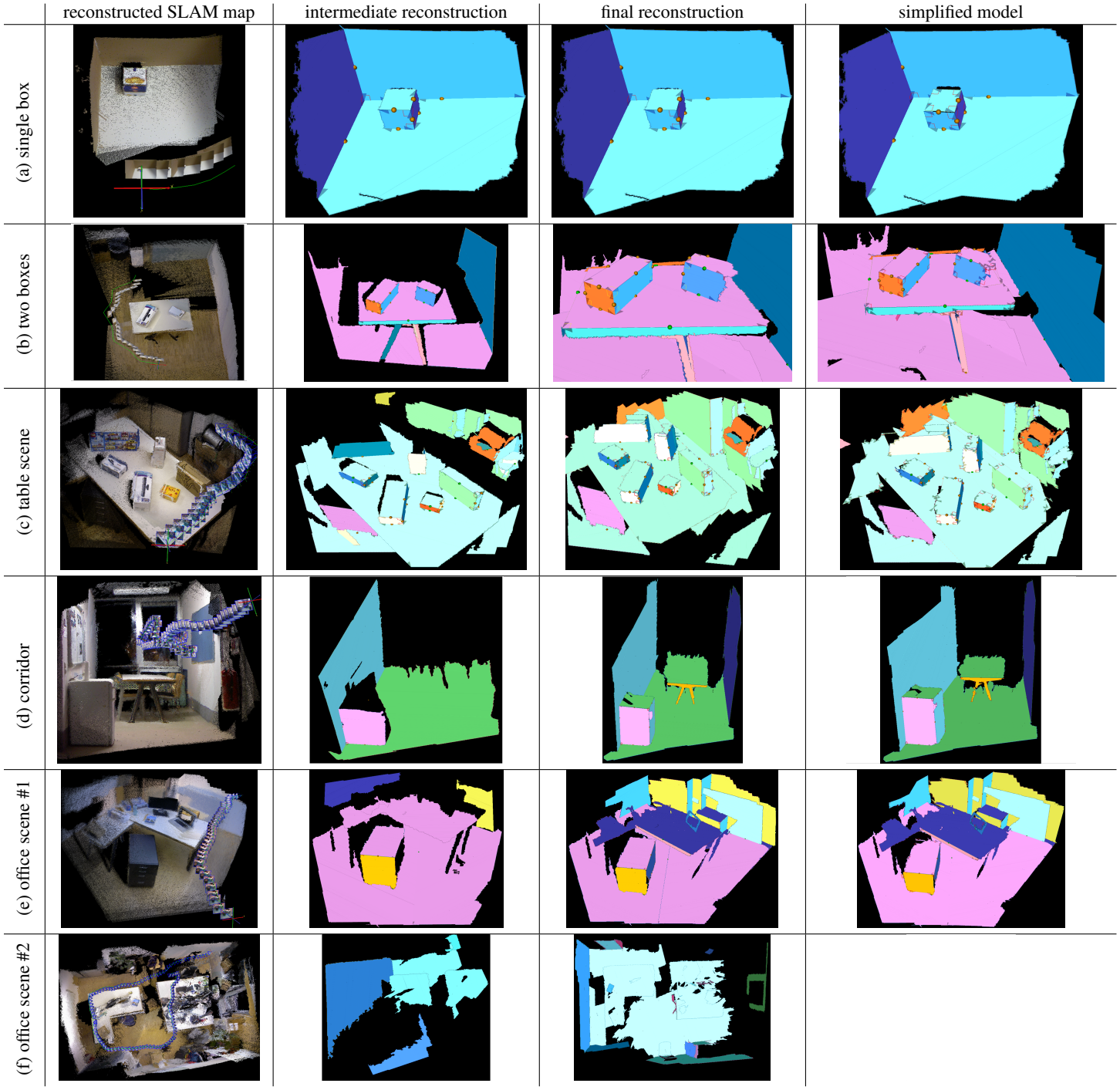


Fig. 12: Qualitative results of different scenes. In column 1, the reconstructed slam map of different scenes is depicted. The map includes the reconstructed point cloud and keyframes poses. Column 2 shows intermediate reconstructed structural models consisting of incomplete planes. Incident halfedges are highlighted with prominent arrows. For pairs of incident and orthogonal planes, we display orange balls on the associated incident lines. For incident only planes, we show green balls on the incident lines. Right angle corners of two consecutive halfedges are highlighted with L shapes on the associated corners' position. Column 3 shows the final reconstructed structural models comprising of few plane polygons. Column 4 shows further simplified models through employing polygon fitting on non-incident occluding borders. Simplified halfedges are also highlighted with prominent arrows. In column 4 of office scene #2 (f), polygon fitting fails to simplify any non-incident occluding edge because of too much drift in SLAM; thus, we do not present a result snapshot there.