# Efficient 3D Room Shape Recovery from a Single Panorama

Hao Yang Hui Zhang
School of Software, Tsinghua University, Beijing, China
Tsinghua National Laboratory for Information Science and Technology (TNList)
yanghao14@mails.tsinghua.edu.cn, huizhang@tsinghua.edu.cn

## Abstract

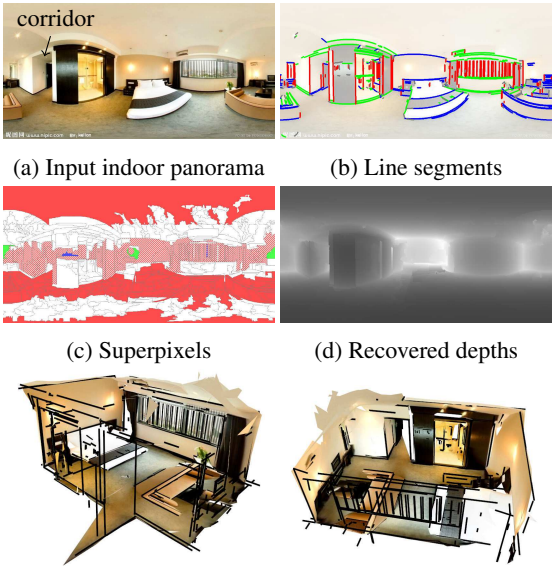
*We propose a method to recover the shape of a 3D room from a full-view indoor panorama. Our algorithm can automatically infer a 3D shape from a collection of partially oriented superpixel facets and line segments. The core part of the algorithm is a constraint graph, which includes lines and superpixels as vertices, and encodes their geometric relations as edges. A novel approach is proposed to perform 3D reconstruction based on the constraint graph by solving all the geometric constraints as constrained linear least-squares. The selected constraints used for reconstruction are identified using an occlusion detection method with a Markov random field. Experiments show that our method can recover room shapes that can not be addressed by previous approaches. Our method is also efficient, that is, the inference time for each panorama is less than 1 minute.*

## 1. Introduction

A 360° full-view indoor panorama is shown in Fig. 1a. We intend to recover the 3D room shape from this panorama. Several methods are available to solve this problem, either by adopting the Indoor World model, which consists of a single floor, a single ceiling, and vertical walls [3, 10], or by estimating a cuboid shape that fits the room layout [6, 17, 7, 11, 2, 15, 16, 14, 20][1]. Intrinsically, most of these approaches work in a discretized manner, that is, the results are selected from a set of candidates based on certain scoring functions. The generation rules of the candidates limit the scope of these algorithms. For example, the corridor of the room in Fig. 1a cannot be modeled either as part of a cuboid or within an Indoor World because the ceiling of the corridor is lower than that of the room.

A method is proposed to address this problem in a geometric manner. We extract lines and superpixels from the panorama and estimate their orientation information under

[1]Most of these methods deal with normal photographs, however, modifying these methods to cope with panoramas is a straightforward process.




(a) Input indoor panorama (b) Line segments

(c) Superpixels (d) Recovered depths

(e) Reconstructed 3D room model

Figure 1: **(a)** Input indoor panorama, with detected lines shown in **(b)**. Red, green, and blue indicate Manhattan directions that are assigned on lines in the preprocessing stage. **(c)** Superpixels generated by over segmentation. Pure colors (except white) indicate that the surface normals are restricted to certain Manhattan directions, striped colors suggest that the surface plane should be parallel with a Manhattan direction in 3D space. In particular, red regions in (c) represent horizontal faces such as floors or ceilings, whereas the striped red regions represent vertical faces such as walls. White indicates that no orientation restriction is imposed. **(d)** Automatically recovered layout depths and **(e)** reconstructed 3D lines (in black) and superpixel facets (all textured) from different views. Face culling is applied to better illustrate the inside model in each view.

*Manhattan world assumption*, as shown in Figs. 1b and 1c. A constraint graph, which includes all the lines and superpixels as vertices is then constructed; and geometric relations are encoded among them. A 3D reconstruction is performed in an iterative manner, which can solve constraints as constrained linear least squares (CLLS). We propose an occlusion detection method using a Markov random

field (MRF) to select plausible constraints for reconstruction. The reconstructed lines and superpixel facets of the room layout from Figs. 1b and 1c are shown in Fig. 1e.

### 1.1. Related Works

Interest in single-view reconstruction (SVR) problems has been constant. Methods in this domain can be roughly divided into two groups: geometric and semantic approaches. Most geometric approaches rely on lines. Lee *et al*. [10] recovered indoor layout by computing orientation maps (OMs) from lines. Xue *et al*. [19] reconstructed symmetric objects by detecting symmetric lines. Xiao *et al*. [18] recognized cuboids from single photographs based on both the appearance of corners and edges as well as their geometric relations. Ramalingam [13] proposed a method that lifted 2D lines to 3D by identifying true line intersections in space.

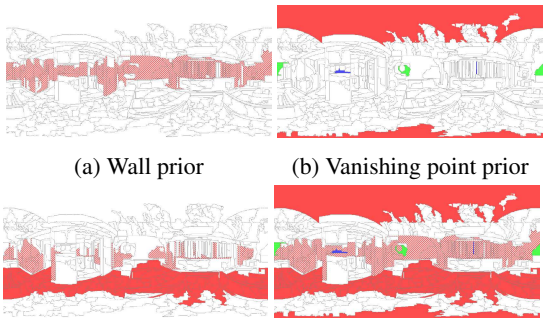
For semantic approaches, Hoiem *et al*. [8, 9] estimated geometric context label on each image pixel. Delage *et al*. [3] inferred room layout via floor–wall boundary estimation. Gupta *et al*. [5] proposed blocks world to predict 3D arrangement by explaining their volumetric and physical relations. Numerous studies have been presented in recent years by assuming room shape as a single box aligned with Manhattan direction. Hedau *et al*. [6] utilized structured learning to improve prediction accuracy. The inferences of both indoor objects and the box-like room layout have been continuously improved thereafter in [17, 7, 11, 2, 15, 16, 14] due to enhanced object presentation, novel features or more efficient inference techniques. Zhang *et al*. [20] recently stressed the limitation of a narrow view angle imposed by standard camera; they illustrated the advantage of panoramic images over normal photographs, where additional contexts could be considered. Cabral [1] took multiple indoor panoramas as input and utilized structural cues from single image for floorplan reconstruction.

Our work focuses on 3D room shape recovery. The algorithm is related to [13], in which a constraint graph is proposed, its vertices are fully orientated lines, and its edges are intersections/incidences between lines. In the present study, we additionally consider other entities, including lines with unknown orientation, and planar superpixels with varying degrees of freedom (DOF). The constraints are also extended to include connections between two superpixels or between a line and a superpixel. Hence, the graph is enhanced to be capable of representing a complex 3D shape including line segments and planar facets.

### 1.2. Contributions

The main contributions of our work are as follows.

- An expressive constraint graph is designed to encode the spatial configurations of line segments and superpixel facets in a uniform manner.



(a) Wall prior (b) Vanishing point prior

(c) Geometric context prior (d) All three merged

Figure 2: Orientation priors for Fig. 1a.

- An iterative algorithm is proposed to solve the constraint graph as CLLS to reconstruct both line segments and superpixel facets together.
- We propose a method to identify occluding lines using an MRF, which helps select plausible constraints for the graph.

## 2. Preprocessing

Our input is a panorama that covers a 360° horizontal field of view represented in an equirectangular projection. Under this projection, one-to-one correspondence occurs between a panorama pixel and a 3D view direction [2] ; therefore, we use the term *angle distance* to measure the distance between two pixels by computing the angle of their directions, and use *angle length* to measure the length of a pixel sequence (like a line segment or a superpixel boundary) by accumulating the vector angles between adjacent pixels.

An approach similar to that in [20] is used to detect lines, estimate Manhattan vanishing points, and identify the spatial directions of lines from the panorama. The vanishing point direction which is the most vertical in space is denoted as the *vertical direction* of the scene. A panorama version of graph cut [3] is utilized to over segment the panorama into superpixels. Orientations of the superpixels are restricted according to the following three priors:

**Wall prior**. We assume regions near the *horizon* to be parts of the walls. In particular, superpixels whose angle distances to *horizon* $< \theta^{tiny}$ are assigned to be *vertical* in space with $\theta^{tiny}$ as a threshold; that is, their surface planes must be parallel with the vertical direction. Fig. 2a illustrates the wall prior assignment for the superpixels generated from Fig. 1a.

[2] In our implementation, the correspondence is formulated as $\mathbf{t} = (cos\phi sin\theta, cos\phi cos\theta, sin\phi)^\intercal$ where $\phi$ and $\theta$ are the latitude and longitude of the point on the panorama and $\mathbf{t}$ is its corresponding spatial direction. Under the equirectangular projection, we define $\phi = \pi(y/h - 0.5), \theta = 2\pi x/w$ where $(x, y)$ are the coordinates of the 2D point in the panorama, and $(w, h)$ is the size of the panorama.

[3] We refer to the details in the supplementary material because they are similar to [4]. A similar approach is also utilized by [1].

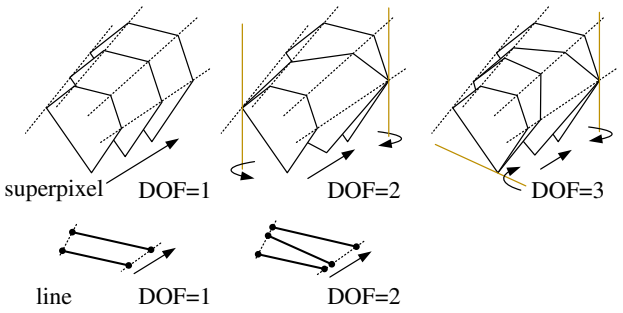



Figure 3: Five types of vertices in the constraint graph. Superpixels are categorized into three types with varying DOF, lines are categorized into two types. **First row**: A superpixel with DOF=1 has a known plane normal, it can only slide within its viewing cone; for example, the normal of a *horizontal* (floor/ceiling) superpixel is fixed to be vertical. By contrast, a superpixel with DOF=2 is restricted to be parallel with a known direction; for example, the plane of a *vertical* (wall) superpixel must be parallel with the vertical direction. No orientation restriction is imposed on superpixels with DOF=3. **Second row**: Lines are simpler; those with DOF=1 can only slide on two fixed rays with a fixed orientation, whereas those with DOF=2 are free. Note that the projection of the contours of superpixels and the endpoints of lines must remain unchanged in the panorama.

**Vanishing point prior**. We also assume that the superpixels with Manhattan vanishing points should face the corresponding Manhattan direction; therefore, their plane normals are assigned as indicated in Fig. 2b.

**Geometric context prior**. The geometric context (GC) extraction [7] on panorama follows the same strategy with [20]. We calculate the average label scores in each superpixel, and use the label with the maximum score to orient the superpixel. Only superpixels that have high scores in two labels are considered: *ground* and *wall*. The *ground* superpixels are restricted to be *horizontal* whereas the *wall* superpixels are all *vertical*. We do not utilize the wall facing information provided by GC. And only bottom half of the panorama is assigned by this prior as recommended in [20]. The assignment is illustrated in Fig. 2c.

Finally, all these three assignments are merged together as shown in Fig. 2d[4].

## 3. Constraint Graph

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is constructed by encoding all lines and superpixels as vertices $\mathcal{V}$. Two types of lines and three types of superpixels are considered to correspond to different *degrees of freedom* (DOF) imposed in preprocessing, as shown in Fig. 3. We consider two types of constraints: the constraints of *connectivity* $\mathcal{E}_{con}$ and the constraints of *coplanarity* $\mathcal{E}_{cop}$. $\mathcal{E}_{con}$ exhibits four types of connection between vertices: 1) connections between adjacent superpixels that share the same boundaries, 2) intersections of lines with different orientations, 3) collinearity of lines with the same orientation and 4) connections between adjacent lines and superpixels. By contrast $\mathcal{E}_{cop}$ is only concerned with superpixel facets that may lie on the same plane. Fig. 4 shows an example that illustrates part of a constraint graph.

[4]If any conflict occurs, the results of the geometric context prior is applied. No conflict will ever occur between the wall prior and the vanishing point prior.

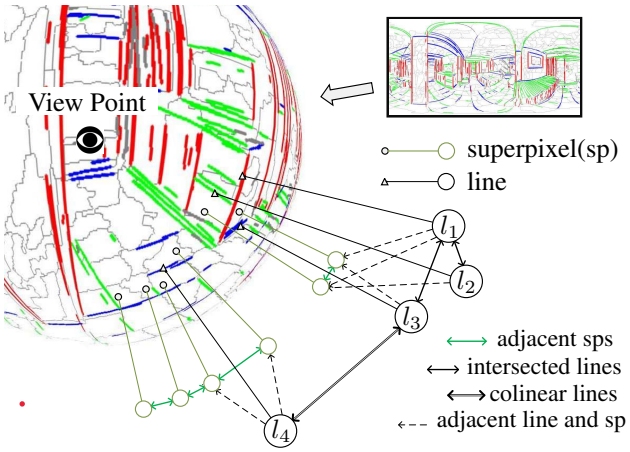



Figure 4: Constraint graph. The graph encodes lines and superpixels as vertices, and their relations as edges.

The following subsections are organized as follows. First in Sec. 3.1, we explain how we parameterize each kind of vertices to encode their types and orientation restrictions. Then in Sec. 3.2, a novel iterative approach is proposed to perform reconstruction by solving the constraints as CLLS. In Sec. 3.3, an occlusion detection algorithm is provided. Finally in Sec. 3.4, we explain how to build a constraint graph based on the occlusions.

### 3.1. Vertex Parameterization

Our objective is to predict the depths of all the visible surfaces to viewpoint. By assuming the surface planarity within each superpixel, the task is simplified to inferring the plane equations of the superpixels.

The number of parameters of each superpixel should correspond to its DOF; therefore, various parameters are designed for superpixels with different DOFs. We denote $\mathbf{x}_i$ as the vector that encodes the unknown parameters of vertex $i \in \mathcal{V}$ and use $\mathcal{C}_i$ to represent the set of known values of $i$. Let $df_i$ be the DOF of $i$.

| $df_i$ | $\mathbf{x}_i$ | $\mathcal{C}_i$ | $\mathbf{P}_i$ |
|---|---|---|---|
| 1 | $(1/d_i)$ | $\{\mathbf{n}_i\}$ | $[\mathbf{n}_i]$ |
| 2 | $(a_i, b_i)^\intercal$ | $\{\mathbf{u}_i\}$ | $\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -\frac{u_{ix}}{u_{iz}} & -\frac{u_{iy}}{u_{iz}} \end{bmatrix}$ |
| 3 | $(a_i, b_i, c_i)^\intercal$ | $\varnothing$ | $\mathbf{I}_{3\times 3}$ |

Table 1: Parameterizing vertices.

The proposed superpixel parameterization is presented in Table 1, where for each superpixel $i$

(1) If $df_i = 1$, then let $\mathbf{n}_i$ be the unit normal vector of its plane and $\mathcal{C}_i = \{\mathbf{n}_i\}$. Define $\mathbf{x}_i = (1/d_i)$, where $d_i$ represents the distance from the plane to the viewpoint.

(2) If $df_i = 2$, then $\mathcal{C}_i = \{\mathbf{u}_i\}$, where $\mathbf{u}_i$ is the unit direction vector that the surface plane must be parallel with. Define $\mathbf{x}_i = (a_i, b_i)^\intercal$, which corresponds to the two parameters in $(a_i, b_i, c_i)$ of the plane equation $a_i x + b_i y + c_i z = 1$.

(3) If $df_i = 3$, then $\mathcal{C}_i$ is empty, because no restriction is imposed regarding its orientation. $\mathbf{x}_i$ is directly defined as $(a_i, b_i, c_i)^\intercal$ which corresponds to the plane parameters.

Numerous parameterization methods are undoubtedly available for these superpixels. For example, we can simply define $\mathbf{x}_i = (d_i)$ if $df_i = 1$, or define $\mathbf{x}_i = (\theta_i, d_i)^\intercal$ if $df_i = 2$, where $\theta_i$ represents the angle of rotation along direction $\mathbf{u}_i$ and $d_i$ is the distance of its plane to the viewpoint. We decide to parameterize superpixels as shown in Table 1 because a linear transformation is available from the proposed superpixel parameter $\mathbf{x}_i$ to its plane coefficients $\boldsymbol{\pi}_i = (a_i, b_i, c_i)^\intercal$; this is required to solve constraints efficiently in Sec. 3.2. These transformation matrices $\mathbf{P}_i$ are presented in the fourth column of Table 1. All these matrices satisfy

$$\boldsymbol{\pi}_i = (a_i, b_i, c_i)^\intercal = \mathbf{P}_i \mathbf{x}_i. \tag{1}$$

For line segments, we propose using their *supporting planes* to encode their spatial arrangement. The supporting plane of a line is defined by two requirements: 1) it must contain the line in space, and 2) it must be orthogonal to the plane that passes through the viewpoint and that contains the line. For example, the supporting plane of line $(P_{1i}, P_{2i})$ in Fig. 5 should always contains the corresponding 3D line and should be orthogonal to the plane that contains rays $\overrightarrow{OP_{1i}}$ and $\overrightarrow{OP_{2i}}$.

A one-to-one correspondence exists between a lifted line in 3D and its supporting plane, and they share the same DOF value. Hence, we can regard lines as degenerated forms of superpixels and use the same method presented in Table 1 to parameterize them.

In particular, as shown in Fig. 5, for line $i = (P_{1i}, P_{2i})$ with DOF=1, $\mathbf{n}_i$ in Table 1 denotes the normal of its supporting plane. For a line $i$ with DOF=2, $\mathbf{u}_i$ in Table 1 represents the normal of the plane that contains line $i$ and passes through viewpoint $O$.

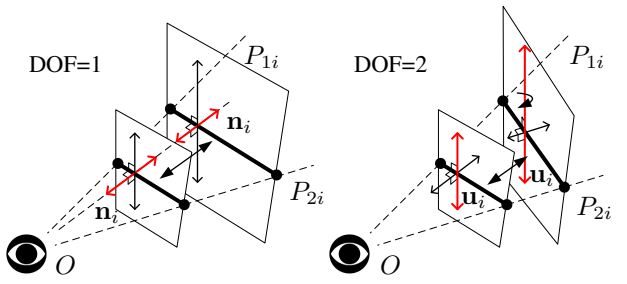



Figure 5: Supporting planes for lines.

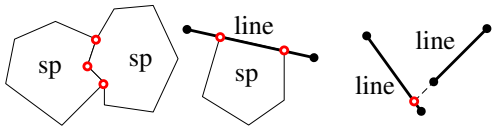



Figure 6: Connection points are shown in red. Each connection point in the panorama represents a view direction in space, along this direction the depths of two related vertices should be equal.

### 3.2. Constraint Solving

In this section, we propose an algorithm to solve all the constraints in $\mathcal{G}$ as CLLS, including $\mathcal{E}_{con}$ and $\mathcal{E}_{cop}$.

All types of connection constraints under $\mathcal{E}_{con}$ share the same objective: to equalize the depths of two vertices on certain *connection points*. As shown in Fig. 6, if the constraint is a connection between two adjacent superpixels or between a line and a superpixel, then its connection points correspond to their common pixels on the boundary or on the line in the panorama. If the constraint is an intersection or a collinear relation between two lines, then its connection point corresponds to the point shared by these two (extended) lines in the panorama.

Let $\mathcal{S}_{ij}$ be the set of connection points of a constraint $(i, j) \in \mathcal{E}_{con}$, and $\boldsymbol{\pi}_i = (a_i, b_i, c_i)$ $\boldsymbol{\pi}_j = (a_j, b_j, c_j)$ be the two (supporting) plane coefficients of vertices $i$ and $j$. Assume that $\mathbf{t} \in \mathcal{S}_{ij}$ is a unit vector representing the spatial direction that corresponds to a connection point. Then, the depth of vertex $i$ on direction $\mathbf{t}$, $d_i(\mathbf{t})$, which is actually the depth of its (supporting) plane $\boldsymbol{\pi}_i$, can be calculated as $d_i(\mathbf{t}) = 1/\mathbf{t}^\intercal \boldsymbol{\pi}_i$ or as $d_i(\mathbf{t}) = 1/(\mathbf{t}^\intercal \mathbf{P}_i \mathbf{x}_i)$ after substituting $\boldsymbol{\pi}_i$ with $\mathbf{P}_i \mathbf{x}_i$ using Equation 1.

To simplify the formulations, we denote vector $\mathbf{x}$ as the concatenation of all the parameter vectors $\mathbf{x}_i$. Let $N$ be the length of $\mathbf{x}$. Then we can construct a (0,1)-matrix $\mathbf{V}_i$, whose size is $df_i \times N$ and satisfies $\mathbf{x}_i = \mathbf{V}_i \mathbf{x}$, by mapping the position of each parameter in $\mathbf{x}_i$ from its corresponding position in $\mathbf{x}$. Therefore, the plane coefficients of vertex $i$ can be written as $\boldsymbol{\pi}_i = \mathbf{P}_i \mathbf{V}_i \mathbf{x}$, and the depth of vertex $i$ on direction $\mathbf{t}$ can be represented as $d_i(\mathbf{t}) = 1/(\mathbf{K}_i(\mathbf{t})\mathbf{x})$ with $\mathbf{K}_i(\mathbf{t}) = \mathbf{t}^\intercal \mathbf{P}_i \mathbf{V}_i$.

We utilize the squared sum energy to equalize the depths $d_i(\mathbf{t})$ and $d_j(\mathbf{t})$ of the two vertices on each connection direction $\mathbf{t}$ of all the constraints in $\mathcal{E}_{con}$, which is formulated as follows:

$$\begin{aligned} E_{con}(\mathbf{x}) &= \sum_{(i,j)\in\mathcal{E}_{con}} w_{ij}^{con} \sum_{\mathbf{t}\in\mathcal{S}_{ij}} \|d_i(\mathbf{t}) - d_j(\mathbf{t})\|^2 \\ &= \sum_{(i,j)\in\mathcal{E}_{con}} w_{ij}^{con} \sum_{\mathbf{t}\in\mathcal{S}_{ij}} \left\| \frac{(\mathbf{K}_i(\mathbf{t}) - \mathbf{K}_j(\mathbf{t}))\,\mathbf{x}}{\mathbf{x}^\intercal \mathbf{K}_i^\intercal(\mathbf{t}) \mathbf{K}_j(\mathbf{t}) \mathbf{x}} \right\|^2. \end{aligned} \tag{2}$$

The constraints of coplanarity are quantified by directly measuring the difference between plane coefficients $\boldsymbol{\pi}_i$ and

$\boldsymbol{\pi}_j$ of the related two superpixel vertices $i$ and $j$ as follows:

$$E_{cop}(\mathbf{x}) = \sum_{(i,j)\in\mathcal{E}_{cop}} w_{ij}^{cop} \|\boldsymbol{\pi}_i - \boldsymbol{\pi}_j\|^2 = \sum_{(i,j)\in\mathcal{E}_{cop}} w_{ij}^{cop} \|(\mathbf{P}_i\mathbf{V}_i - \mathbf{P}_j\mathbf{V}_j)\,\mathbf{x}\|^2. \quad (3)$$

$w_{ij}^{con}$ and $w_{ij}^{cop}$ are the weights given to constraints on $(i,j)$.

Directly minimizing $E_{con}(\mathbf{x}) + \alpha E_{cop}(\mathbf{x})$ ($\alpha$ is a weight) is intractable due to the non-linearity of $E_{con}$. Instead, we use an iterative approach which solves an approximated CLLS in each iteration to converge to the solution.

The approach is described in Algorithm 1. In each iteration: it first freezes the non-linear component $\phi_{ij}^k(\mathbf{t})$ as a constant and computes an approximated solution $\mathbf{x}^k$ by addressing the CLLS problem described in Line a, where an inequality is applied to avoid obtaining trivial results. Then, it updates the frozen component $\phi_{ij}^k(\mathbf{t})$ to $\phi_{ij}^{k+1}(\mathbf{t})$ based on the current solution $\mathbf{x}^k$, as is formulated in Line b.

**Algorithm 1:** Solving Constraints with CLLS

$k \leftarrow 0$; $\phi_{ij}^0(\mathbf{t}) \leftarrow 1, \forall \mathbf{t} \in \mathcal{S}_{ij} \forall (i,j) \in \mathcal{E}_{con}$;

**repeat**

a $\mathbf{x}^k \leftarrow$ the result of a CLLS problem:

$$\min_{\mathbf{x}} E_{con}^k(\mathbf{x}) + \alpha E_{cop}(\mathbf{x})$$
$$s.t.\ \mathbf{K}_{\{i,j\}}(\mathbf{t})\mathbf{x} \geqslant 1.0, \forall \mathbf{t} \in \mathcal{S}_{ij}, (i,j) \in \mathcal{E}_{con}$$

with

$$E_{con}^k(\mathbf{x}) = \sum_{(i,j)\in\mathcal{E}_{con}} w_{ij}^{con} \sum_{\mathbf{t}\in\mathcal{S}_{ij}} \left\| \frac{(\mathbf{K}_i(\mathbf{t}) - \mathbf{K}_j(\mathbf{t}))\,\mathbf{x}}{\phi_{ij}^k(\mathbf{t})} \right\|^2$$

b $\phi_{ij}^{k+1}(\mathbf{t}) \leftarrow \mathbf{x}^{k\intercal}\mathbf{K}_i^\intercal(\mathbf{t})\mathbf{K}_j(\mathbf{t})\mathbf{x}^k, \forall \mathbf{t} \in \mathcal{S}_{ij} \forall (i,j) \in \mathcal{E}_{con}$;

$k \leftarrow k + 1$ ;

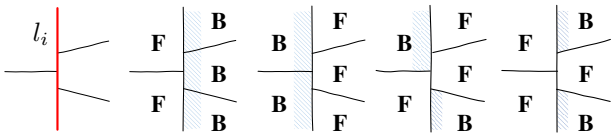
**until** *convergence* $\vee k =$ *maximum_iterations*;

**return** $\mathbf{x}^k$ ;

### 3.3. Occlusion Identification

Identifying constraints $\mathcal{E}_{con}$ and $\mathcal{E}_{cop}$ remains challenging because of occlusions. Room occlusions are likely to be covered by detected line segments, and tend to follow the rule of *coherence*. Using Fig. 7 for example, if any occlusion exists on $l_i$, then the sidings of the occlusions will likely remain coherent along $l_i$.

Therefore, we detect occlusions by labeling lines. Only oriented lines are considered for simplicity. To each line $l_i$ with DOF=1, we assign label $\mathbf{y}_i = (y_i^l, y_i^r)$. $y_i^l, y_i^r \in \{0,1\}$ are two flags that encode the occlusion status on the two sides of $l_i$. We use $y_i^{\{l,r\}}$ to indicate whether the superpixels on the {left, right} side of $l_i$ are in front ($= 1$) or are occluded behind ($= 0$). In particular, label $\mathbf{y}_i = (1,1)$ indicates that no occlusion exists on $l_i$, $\mathbf{y}_i = (1,0)$ indicates the left occludes the right, whereas $\mathbf{y}_i = (0,1)$ indicates that the right occludes the left. Finally, $\mathbf{y}_i = (0,0)$ suggests a dangling line that does not connect to the surfaces.




(a) (b) Plausible cases (c) Implausible cases

Figure 7: Coherence of occlusions. (a) Segmented superpixels near line $l_i$. (b) and (c) illustrate possible occluding statuses of adjacent superpixels, where **F** indicates *surface in front* and **B** denotes *surface occluded behind*. The cases shown in (b) are relatively more plausible than those in (c) based on logic.

Three kinds of evidence are used in occlusion detection: 1) *orientation violations* between lines and superpixels, 2) *T-junctions* formed by lines, and 3) the *coherence* between collinear lines. Based on the preceding discussion, we construct an MRF to infer the $\mathbf{y}_i$ of each $l_i$ by minimizing the following objective:

$$\mathbf{y}^* = \arg\min_{\mathbf{y}} \sum_{l_i\in\mathcal{V}_L^1} E_i^{unary}(\mathbf{y}_i) + \sum_{(l_i,l_j)\in\mathcal{E}_{colinear}, l_i,l_j\in\mathcal{V}_L^1} E_{ij}^{binary}(\mathbf{y}_i,\mathbf{y}_j) \quad (4)$$

with

$$E_i^{unary}(\mathbf{y}_i) = E_i^{ov}(\mathbf{y}_i) + E_i^{tj}(\mathbf{y}_i) + \tau(\mathbf{y}_i), \quad (5)$$

where $\mathcal{V}_L^1$ is the set of lines with DOF=1. The unary cost $E_i^{unary}$ on each line $l_i$ consists of three terms: the orientation violation cost $E_i^{ov}$, the T-junction cost $E_i^{tj}$, and a label prior $\tau(\mathbf{y}_i)$ to punish disconnections. $E_{ij}^{binary}$ is the coherence cost imposed on collinear lines.

**Orientation Violation Cost** Let $\mathbf{r}_i$ be the spatial orientation of line $l_i$, $vp_i^l$ and $vp_i^r$ be two closest vanishing points to $l_i$ that satisfy two conditions: 1) it lies on the {left, right} side of $l_i$, and 2) its spatial direction is orthogonal to $\mathbf{r}_i$. Then, we *sweep* $l_i$ toward $vp_i^{\{l,r\}}$ to an angle $\theta$ and form two spherical quadrilaterals that represent the *neighborhoods* of line $l_i$ in the panorama, denoted as $\Omega_i^{\{l,r\}}(\theta)$. Fig. 8 presents an example wherein two lines sweep toward the same vanishing point.

$N_i^{\{l,r\}}$ is then defined as the weighted number of nearby pixels that cause violations on the {*left*, *right*} side of $l_i$:

$$N_i^{\{l,r\}} = \sum_{p\in\Omega_i^{\{l,r\}}(\theta^{mid})} w(p)\, conflict(p, l_i), \quad (6)$$

where $\theta^{mid}$ is a threshold. $w(p)$ is the pixel weight used to rectify panorama distortion. $conflict(p, l_i)$ is 1 if the given orientation of the superpixel with $p$ conflicts with the direction of $l_i$, and 0 otherwise. A conflict occurs if the superpixel has DOF=1 and its plane normal coincides with the direction of $l_i$.

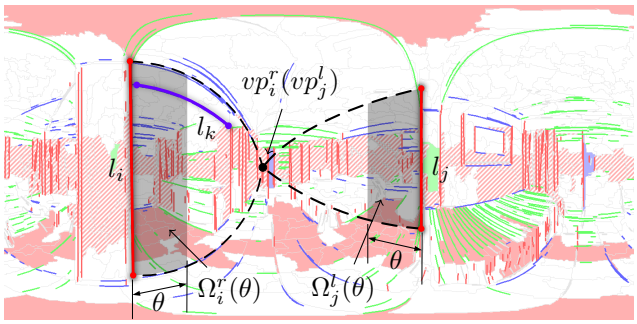



Figure 8: *Neighborhoods* of lines. The right neighborhood of line $l_i$, denoted as $\Omega_i^r(\theta)$, is generated by sweeping $l_i$ toward its right vanishing point $vp_i^r$ to angle $\theta$. The left neighborhood $\Omega_j^l(\theta)$ of line $l_j$ is similarly defined.

$E_i^{ov}$ is then defined as follows, where $c^{ov}$ is a weight:

$$E_i^{ov}(\mathbf{y}_i) = c^{ov}\left(\frac{y_i^l {N_i^l}^2}{\max\{{N_i^r}^2, 1\}} + \frac{y_i^r {N_i^r}^2}{\max\{{N_i^l}^2, 1\}}\right). \quad (7)$$

**T-junction Cost** T-junction is a commonly used evidence for occlusion recognition [13]. For example, in Fig. 8 line $l_k$ forms a T-junction on line $l_i$.

We detect T-junctions by collecting all pairs of lines with different orientations, which form T-structures, and their distances are $< \theta^{tiny}$. Define $M_i^{\{l,r\}} = \exp(-(\frac{m_i^{\{l,r\}}}{10})^2)$, where $m_i^{\{l,r\}}$ is the number of T-junctions on the {left, right} side of line $l_i$, and the T-junction cost is given by

$$E_i^{tj}(\mathbf{y}_i) = c^{tj}\left((1-y_i^l)M_i^l + (1-y_i^r)M_i^r\right). \quad (8)$$

**Coherence Cost** A binary term $E_{ij}^{binary}$ is imposed on each pair of collinear lines $l_i$ and $l_j$ to encourage the coherence of labels, which is formulated as

$$E_{ij}^{binary}(\mathbf{y}_i, \mathbf{y}_j) = c^{binary}\left(\begin{cases} \mathbb{1}(y_i^l \neq y_j^l) + \mathbb{1}(y_i^r \neq y_j^r) & \text{if } l_i \uparrow\uparrow l_j \\ \mathbb{1}(y_i^l \neq y_j^r) + \mathbb{1}(y_i^r \neq y_j^l) & \text{if } l_i \uparrow\downarrow l_j \end{cases}\right), \quad (9)$$

with $\mathbb{1}(\cdot)$ as the boolean to 0-1 conversion, $\uparrow\uparrow, \uparrow\downarrow$ denotes whether the two line directions are equal or opposite.

### 3.4. Graph Construction

Equation 4 is solved using the *convex max product* [12]. The resulting labels are used to build graph constraints.

A superpixel pair $(i, j)$ that shares a same boundary is collected into $\mathcal{E}_{con}$ and $\mathcal{E}_{cop}$ if its boundary is not covered by any occluding line. Connection points $\mathcal{S}_{ij}$ are set as the turning points on the boundary. Let $L_{ij}$ be the angle length of the boundary, then $w_{ij}^{con} = L_{ij}/\|\mathcal{S}_{ij}\|$. $w_{ij}^{cop} = L_{ij}$ if there are no lines lying on the boundary; otherwise $w_{ij}^{cop} = 0.1L_{ij}$ because non-occluding lines always suggest foldings.

(a) Annotation (b) Reconstruction from annotation

Figure 9: **(a)** shows the manually labeled face orientations (in red, green and blue) and occlusions (in white) of the panorama in Fig. 1a. **(b)** displays the directly reconstructed 3D model based on the annotation.

A line and a superpixel $(i, j)$ that are adjacent in the panorama are collected into $\mathcal{E}_{con}$ if the superpixel lies on the front side of the line. Connection points $\mathcal{S}_{ij}$ are set as the endpoints of the shared segment. $w_{ij}^{con} = L_{ij}/\|\mathcal{S}_{ij}\|$ with $L_{ij}$ as the angle length of the shared segment.

Recognizing constraints between lines $(i, j)$ is a slightly complex process. We consider a pair of lines with different orientations as intersecting if the angle distance between the line segments is $< \theta^{tiny}$. Collinearity is considered for lines with the same orientation that are separated by an angle $< \theta^{large}$. Then, we connect the nearest points of these two lines and check whether the connection is intercepted by any occluding line. If any of $i, j$ is an occluding line, then we place them into $\mathcal{E}_{con}$ only if 1) the other line lies on the front side of the occluding line, and 2) their connection is not intercepted. If neither $i, j$ is an occluding line, then $(i, j)$ is collected only if their connection is not intercepted. The $\mathcal{S}_{ij}$ of the two lines is set as the common pixel shared by these two (extended) lines, and $w_{ij}^{con} = \max\{J_{ij}, 1\}$, $J_{ij}$ is the junction score proposed by [13].

Finally, we apply searches on the graph from vertices with DOF=1 to find the largest determinable subgraph for reconstruction [5].

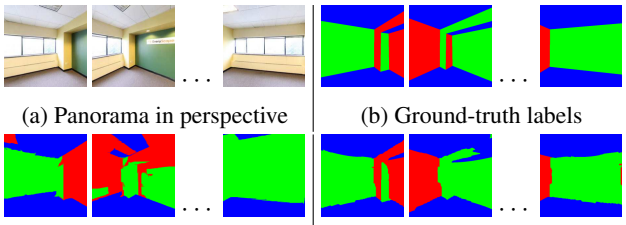
## 4. Experiments

We collected 88 indoor panoramas for evaluations with manual annotation of vps, room layout faces, face orientations and occluding lines (as is shown in Fig. 9a). In experiments, thresholds $\theta^{\{tiny,mid,large\}}$ are set as $2^\circ, 5^\circ, 15^\circ$. In Algorithm 1, the maximum iteration number is set to be 5, the weight $\alpha$ is set as $10^{-6}$. In the MRF terms, the label prior $\tau(\mathbf{y}_i)$ in Equation 5 is defined as $\begin{cases} 0 & \text{if } \mathbf{y}_i = (1,1) \\ 5 & \text{if } \mathbf{y}_i = (0,0) \\ 2 & \text{otherwise} \end{cases}$, the pixel weight $w(p)$ in Equation 6 is formulated as $sin(\frac{p_y}{H}\pi)$, where $H$ is the height of the panorama. The coefficient $c^{ov}$ in Equation 7 is set to be 100, $c^{tj}$ in Equation 8 is set as 1 and $c^{binary}$ in Equation 9 is set as 5.

#### Quantitative Evaluation

**Surface Label** First, the proposed method is evaluated in the aspect of surface label inference.

[5] Details are deferred to the supplementary material.



(a) Panorama in perspective (b) Ground-truth labels

(c) GC labels (d) Our reconstruction

Figure 10: Perspective conversion to compare surface labels.

Each panorama is projected onto 12 overlapping perspective images to extract GC. Five GC labels are considered: *floor*, *ceiling*, and {*front, left, right*} *wall*. We also project the surface normals of our reconstructed superpixels and the ground-truth face normals in the same manner. Each pixel is labeled in the aforementioned labels according to its normal direction in the perspective image. The normals of the superpixels that are not reconstructed in our algorithm are computed by averaging their adjacent superpixels. Fig. 10 illustrates examples of surface labels predicted by various methods.

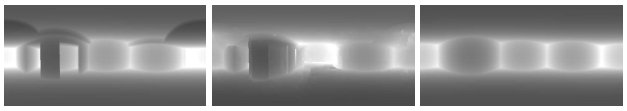
We use the per-pixel classification error (i.e., the percentage of pixels that have been wrongly classified) to quantify the predictions in each perspective image. Three types of evaluation are performed: 1) consider all pixels, 2) only considers pixels with $GC_{clutter} < 0.7$, 3) only considers pixels with $GC_{clutter} < 0.3$. The average error ratios are reported in Table 2, which shows that our result outperforms GC in all three cases. We also show results of our algorithm that do not use GC prior in the bottom row of Table 2, the errors rise but are still lower than those of GC.

**Depth Distribution** Second, we compare 3D models using their depth maps. A 3D model is directly reconstructed from the ground-truth annotation for each panorama, its depth map is used as the *reference* (as shown in Fig. 11a). Evaluations are performed by comparing depth maps with the *reference* using two metrics: *L2 distance* which is formulated as $\|\mathbf{d}_a - \mathbf{d}_b\|_2$, and *cosine distance* which is computed by $1 - \mathbf{d}_a^\intercal \mathbf{d}_b$. $\mathbf{d}_*$ is a normalized vector that comprises all weighted (by $w(p)$) depths of pixels in depth map.

Our method is compared with the *cuboid of best fit* (COBF), which is used to represent the performance upper bound of aligned-box-methods. We enumerate a discretized parameter space of cuboids that are aligned with the Manhattan direction and select the one that best fits the *reference* (measured by *L2 distance*) as the COBF of the panorama.

Results are presented in Table 3, showing that our method (second row) outperforms COBF (first row). Indirectly, this also suggests the advantage of the proposed approach in the aspect of room shape estimation over the algorithms that estimate room shapes by fitting cuboids aligned with Manhattan direction.

Results of the proposed method that assumes no occlusions are reported in the third row, and the results of that utilizing the ground-truth occlusions are shown in the fourth row. It could be observed that assuming no occlusions raises prediction errors. However, using the ground-truth occlusions does not bring meaningful improvement. We also report the average errors of a uniform sphere (centered at viewpoint) in the bottom row for comparison.



(a) Reference (b) Proposed (c) COBF

Figure 11: Depth maps of Fig. 1a. **(a)** The *reference* is set as the depth map from the model which is reconstructed directly from annotation, the corresponding model of (a) is shown in Fig. 9b. **(b)** The depth map of the proposed reconstruction. **(c)** The depth map of the corresponding COBF.

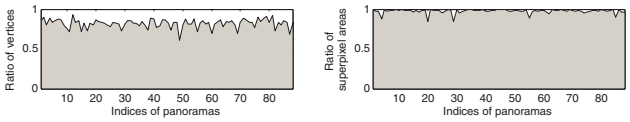



Figure 12: Coverage of our reconstruction. The first graph shows the ratios of recovered vertices; the second graph shows ratios of superpixel areas reconstructed in each panorama.

| $GC_{clutter}$: | all | $< 0.7$ | $< 0.3$ |
|---|---|---|---|
| test coverage | 1.0 | 0.95 | 0.60 |
| *GC* [6] | 44.5 | 44.4 | 48.7 |
| proposed | 26.9 | 26.8 | 26.7 |
| proposed(no gc prior) | 31.8 | 31.7 | 28.6 |

Table 2: Surface label classification error (%).

| metric | cosine dist. | L2 dist. |
|---|---|---|
| COBF | 5.23 | 28.48 |
| proposed | 4.27 | 27.02 |
| proposed(no occl) | 4.53 | 27.85 |
| proposed(gt occl) | 4.23 | 27.09 |
| sphere | 7.54 | 37.23 |

Table 3: Depth distribution error ($\times 10^{-2}$).

**Reconstruction Coverage** Algorithmically, our method does not reconstruct all the superpixels and the lines; hence we evaluate the coverage of the reconstruction. The statistics is shown in Fig. 12. On average, 81.9% of the vertices are reconstructed, and 97.5% of the superpixel areas are recovered.

### Qualitative Evaluation

Some of the results are presented in Figs. 13 and 14, which illustrate that our algorithm can reconstruct room shapes with occlusions and even some non-Manhattan parts. Note that in these cases lots of false occlusions are identified by

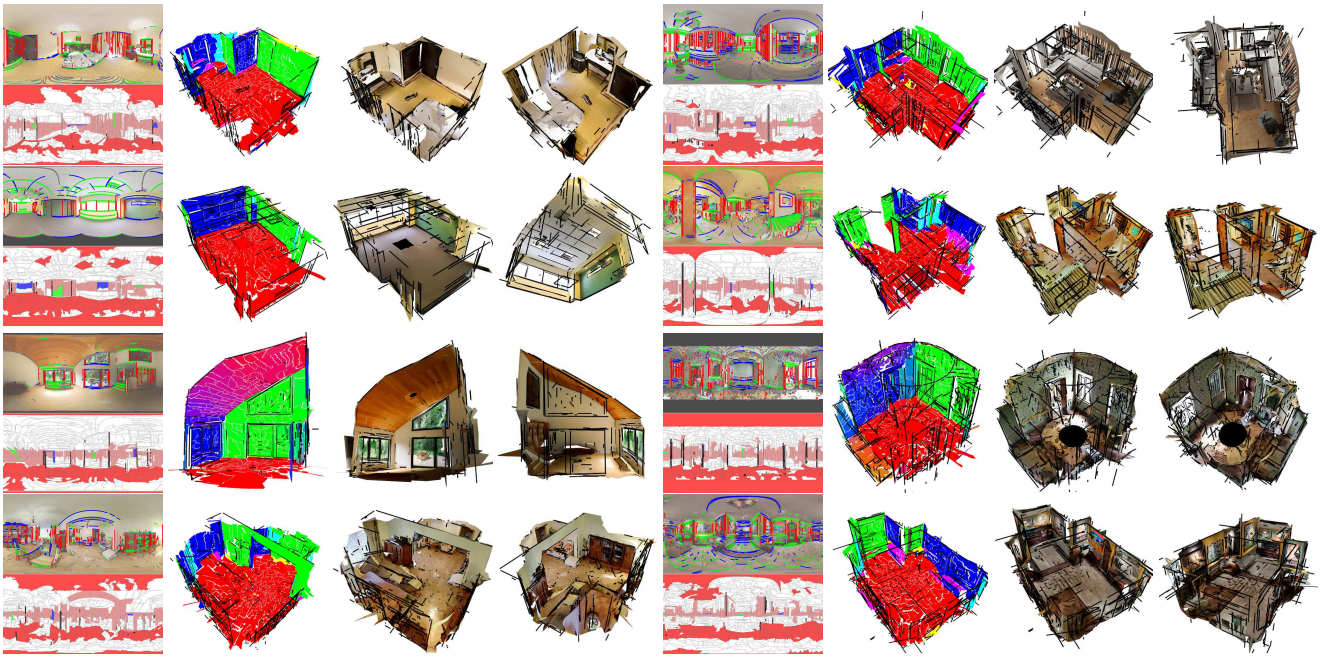


Figure 13: Reconstruction results. In each case, the leftmost two images show the input panorama, the extracted lines and the superpixels. Orientation restrictions of the lines and the superpixels are visualized in the same manner with Fig.1. Detected occlusions are rendered on the second images of the leftmost columns as black jagged lines, wherein the jagged side of each line represents the side of the front surface. The rest display reconstructed 3D models, the lines are all in black, the superpixels are colored by surface normals in the second columns, and are textured with the input panoramas in the rest. Face culling is applied to better illustrate the inside shape in each view.

the proposed occlusion detection. However, the recovered 3D models show the robustness of our reconstruction algorithm against the false occlusions. Fig. 14 shows two failed cases, where our algorithm is misled by wrongly oriented superpixels.

We invoke code of [6] to extract GC and use CVX to solve the CLLS, the rest is implemented in C++. On a PC with Intel Core i5 CPU (3.1/3.3GHz), the time cost of occlusion identification is approximately 3 seconds per panorama, and solving the CLLS costs less than 5 seconds per iteration. The total time cost of inference (without preprocessing stage) is within 1 minute for each panorama in our dataset.

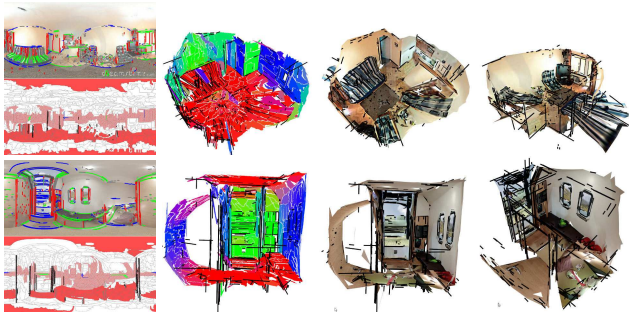


Figure 14: Two failed cases. In the first case, the reconstruction is misled by the false horizontal superpixels that cover the sofa in the room. In the second case, some wall superpixels are incorrectly identified as parts of the floor, which leads to a false illusion of doors open on the wall.

## 5. Conclusions

In this study a fully automatic method is proposed to reconstruct 3D room shape from a single indoor panorama. The method performs reconstruction based on partially oriented lines and superpixels; it identifies occlusions by labeling lines using an MRF. Geometric relations between lines and superpixels, such as connections and coplanarity, are identified using these labels. A constraint graph is then built; and the constraints are solved as CLLS by a novel iterative algorithm.

Experiments show that the proposed method outperforms geometric context in surface label inference; the proposed method is also advantageous over algorithms that model room layouts as aligned cuboids. The 3D reconstruction results illustrate that the proposed method can recover room shapes that include occlusions and even non-Manhattan parts from a single indoor panorama.

**Acknowledgement** This work is supported by the National Natural Science Foundation of China (61373070), the National Key Technologies R&D Program of China (2015BAF23B03), and Tsinghua University Initiative Scientific Research Program (2012Z02170).

## References


[1] R. Cabral and Y. Furukawa. Piecewise planar and compact floorplan reconstruction from images. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, 2014.

[2] W. Choi, Y.-W. Chao, C. Pantofaru, and S. Savarese. Understanding indoor scenes using 3d geometric phrases. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 33–40. IEEE, 2013.

[3] E. Delage, H. Lee, and A. Y. Ng. A dynamic bayesian network model for autonomous 3d reconstruction from a single indoor image. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2418–2428. IEEE, 2006.

[4] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[5] A. Gupta, A. A. Efros, and M. Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *Computer Vision–ECCV 2010*, pages 482–496. Springer, 2010.

[6] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *Computer vision, 2009 IEEE 12th international conference on*, pages 1849–1856. IEEE, 2009.

[7] V. Hedau, D. Hoiem, and D. Forsyth. Thinking inside the box: Using appearance models and context based on room geometry. *Computer Vision–ECCV 2010*, pages 224–237, 2010.

[8] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. *ACM Transactions on Graphics (TOG)*, 24(3):577–584, 2005.

[9] D. Hoiem, A. A. Efros, and M. Hebert. Recovering surface layout from an image. *International Journal of Computer Vision*, 75(1):151–172, 2007.

[10] D. Lee, M. Hebert, and T. Kanade. Geometric reasoning for single image structure recovery. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2136–2143. IEEE, 2009.

[11] D. C. Lee, A. Gupta, M. Hebert, and T. Kanade. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. *Advances in Neural Information Processing Systems (NIPS)*, 24:1288–1296, 2010.

[12] J. Peng, T. Hazan, D. McAllester, and R. Urtasun. Convex max-product algorithms for continuous mrfs with applications to protein folding. In *Proc. ICML*, 2011.

[13] S. Ramalingam and M. Brand. Lifting 3d manhattan lines from a single image. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 497–504. IEEE, 2013.

[14] S. Ramalingam, J. K. Pillai, A. Jain, and Y. Taguchi. Manhattan junction catalogue for spatial reasoning of indoor scenes. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3065–3072. IEEE, 2013.

[15] A. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun. Efficient structured prediction for 3d indoor scene understanding. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2815–2822. IEEE, 2012.

[16] A. G. Schwing, S. Fidler, M. Pollefeys, and R. Urtasun. Box in the box: Joint 3d layout and object reasoning from single images. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 353–360. IEEE, 2013.

[17] H. Wang, S. Gould, and D. Koller. Discriminative learning with latent variables for cluttered indoor scene understanding. *Computer Vision–ECCV 2010*, pages 435–449, 2010.

[18] J. Xiao, B. Russell, and A. Torralba. Localizing 3d cuboids in single-view images. In *Advances in Neural Information Processing Systems 25*, pages 755–763, 2012.

[19] T. Xue, J. Liu, and X. Tang. Symmetric piecewise planar object reconstruction from a single image. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2577–2584. IEEE, 2011.

[20] Y. Zhang, S. Song, P. Tan, and J. Xiao. Panocontext: A whole-room 3d context model for panoramic scene understanding. In *Computer Vision–ECCV 2014*, pages 668–686. Springer, 2014.