*International Journal of Advanced Smart Convergence Vol.11 No.1 1-10 (2022)*
*http://dx.doi.org/10.7236/IJASC.2022.11.1.1*

IJASC 22-1-1

# Single-View Reconstruction of a Manhattan World from Line Segments

Suwon Lee[1] and Yong-Ho Seo[2]

[1]*Associate Professor, School of Computer Science and the Research Institute of Natural Science, Gyeongsang National University, Korea*
[2]*Professor, Department of AI and Robot Convergence, Mokwon University, Korea*
[1]*leesuwon@gnu.ac.kr,* [2]*yhseo@mokwon.ac.kr*

***Abstract***

*Single-view reconstruction (SVR) is a fundamental method in computer vision. Often used for reconstructing human-made environments, the Manhattan world assumption presumes that planes in the real world exist in mutually orthogonal directions. Accordingly, this paper addresses an automatic SVR algorithm for Manhattan worlds. A method for estimating the directions of planes using graph-cut optimization is proposed. After segmenting an image from extracted line segments, the data cost function and smoothness cost function for graph-cut optimization are defined by considering the directions of the line segments and neighborhood segments. Furthermore, segments with the same depths are grouped during a depth-estimation step using a minimum spanning tree algorithm with the proposed weights. Experimental results demonstrate that, unlike previous methods, the proposed method can identify complex Manhattan structures of indoor and outdoor scenes and provide the exact boundaries and intersections of planes.*

***Keywords:*** *single-view reconstruction, 3D reconstruction, Manhattan world, line segment detection*

## 1. Introduction

Single-view reconstruction (SVR) is a fundamental method in computer vision that has been studied over the past decades. However, although various algorithms have been presented, high-quality reconstruction from a single image remains a challenging issue. Unlike multi-view reconstruction, which restores a 3D space from images of multiple viewpoints, retrieving 3D information from a single image results in ambiguity in general cases. Therefore, most SVR algorithms utilize geometric or photometric information to eliminate ambiguity.

Many applications provide 3D information for street-side building exterior and interiors, increasing the interest in applying SVR to human-made structures. The algorithm can be used for scene understanding or restoring the 3D information of buildings from a single image. In addition, it can facilitate multi-view reconstruction when the image viewpoints are sparse. Because straight lines and repetitive patterns appear in

Manuscript Received: January. 3, 2022 / Revised: January. 7, 2022 / Accepted: January. 10, 2022
Corresponding Author: yhseo@mokwon.ac.kr
Tel: +82-42-829-7521, Fax: +82-42-829-7909
Professor, Department of AI and Robot Convergence, Mokwon University, Korea

typical human-made structures, additional information, such as vanishing points and line direction data, can be exploited.

Some additional assumptions are applicable to human-made environments. The Manhattan world assumption is the most widely used assumption. First cited in [1], the Manhattan world refers to planes and intersecting lines of structures aligned in mutually orthogonal directions in a 3D space. This assumption can be applied to most human-made structures, while significantly reducing 3D world complexity.

In this paper, an SVR method for reconstructing a Manhattan world is proposed. An image is segmented using the detected line segments, and the normal of each segment is determined by the surrounding line segment directions. An optimization step based on graph-cut segmentation is performed. Based on the experimental results, our algorithm shows effective performance in reconstructing complex Manhattan structures, which cannot be achieved with existing reconstruction methods.

## 2. Related work

Early SVR algorithms require additional user input. Sturm and Maybank [2] proposed an interactive algorithm to reconstruct piecewise planar objects. The directions of lines and planes in an image were manually specified, and geometric constraints were generated to reconstruct 3D information. Müller et al. [3] developed a procedural model that was applied to building facades. The method identifies the repetitive pattern of windows from a façade image in a frontoparallel view, and the user then specifies the depth for each component of the window.

Another approach for SVR is to use machine learning algorithms, which require an additional training step. Hoiem et al. [4] suggested the “photo pop-up” automatic SVR method. The method segments an image into superpixels and their groupings or “constellations,” where each constellation is labeled with the categories of “ground,” “sky,” and “vertical.” Color, texture, and geometric information were used as features for learning during the training phase. Saxena et al. [5] segmented an image into superpixels. Their method trains the relationship of neighboring superpixels using the Markov random field (MRF) model and exploits geometric relationships between superpixels, such as collinearity and coplanar structures, and image features of superpixels. Neither algorithms are restricted to human-made structures; they can be applied to any type of image, including natural outdoor scenes.

Algorithms that exploit geometric information, especially line segments, have been recently proposed. Lee et al. [6] proposed an indoor Manhattan world reconstruction algorithm. They defined a building hypothesis to classify the intersections of lines into several possible 3D structures. Flint et al. suggested a dynamic programming approach for reconstructing an indoor Manhattan world [7]. In [8], algorithms that do not require the Manhattan world assumption were presented. The normal of planes can be estimated through rectification, which uses the intersections of line segments. These algorithms do not require a training step; rather, they rely only on the geometric relationships of line segments.

Unlike the above methods, the approach of Wu et al. is based on the repetitive patterns of buildings [9]. The minimal repeating unit of a building façade was determined, and the depth of each pixel was estimated through the graph-cut optimization framework. Ramalingam and Brand reconstructed the line segments of Manhattan world images into a 3D space [10].

This study has several contributions: First, we solved the plane normal estimation problem using the graph-cut optimization framework. The data cost function and smoothness cost function were defined using the surrounding line of each segment. Then, we applied the graph-cut algorithm segment wise rather than pixel wise, which helps to reduce the optimization time. Second, we extended Zaheer’s method [8] to estimate the depth values from the oversegmented image. In particular, our proposed algorithm uses a minimum spanning tree (MST) to group segments that have the same depth. Consequently, the experimental results show that our algorithm can be applied to indoor and outdoor Manhattan world images. Moreover, better results were demonstrated as compared to those of previous methods, such as in [6] and [8]. Hence, our method can reconstruct complex Manhattan structures, which cannot be accomplished using existing methods.

## 3. Proposed method

### 3.1 Pre-processing

Line segments are the primary features of our reconstruction method. We used the line segment detection (LSD) algorithm [11] to extract line segments. Line segments extracted from LSD may be fragmented or broken when applied to real-world images. To segment an image into pieces using line segments, we must connect the gap between the line segments. We extend the line segment until it intersects with another line segment or before the RGB values of the pixels adjacent to the line segment suddenly change. A drastic intensity change indicates that a segment is occluded by another segment. We assume that if neighboring pixel values are similar, those pixels tend to be on the same plane. A line segment in the image is extended from the end points as shown in Figure 1(a). Then, the image is segmented using extended line segments as shown in Figure 1(b). All the closed polygons formed by the surrounding line segments became segments. We intentionally oversegmented the image using extended line segments because there is no step in dividing the segment after the line extension step.

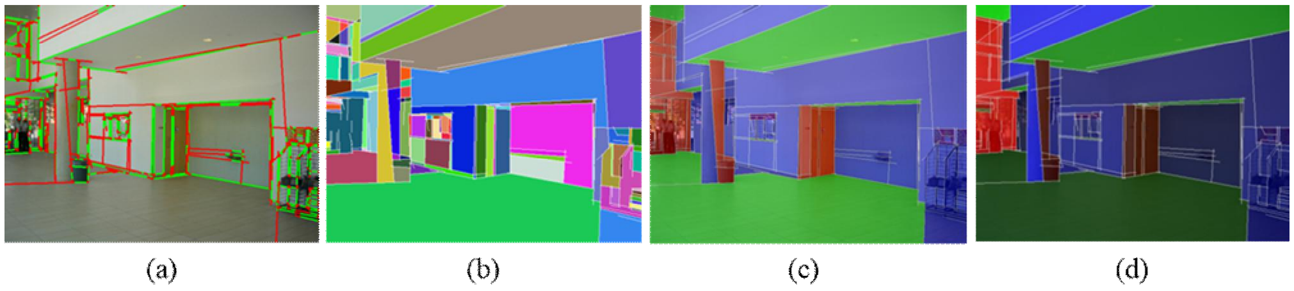

**Figure 1. Overview of the proposed algorithm: (a) Line segment extraction and extension. (b) Oversegmented image. (c) Estimating normal directions of segments. (d) Estimating segment depths.**

### 3.2 Vanishing point detection and finding Manhattan directions

Before we assign normal and depth values for each segment, we must estimate the three vanishing points of Manhattan directions from the extended line segments. Manhattan directions are mutually orthogonal; therefore, finding three directions with only one plane is possible. Line segments should orthogonally meet in Manhattan worlds, but they are not orthogonal in the 2D image because of perspective distortion. Perspective distortion may occur when the camera is rotated with respect to the plane. When the camera is in a canonical position, the perspective distortion induced by the camera rotation can be formulated as follows:

$$\mathrm{H} = \mathrm{K}\mathrm{R}_{\mathrm{Z}}^{\gamma}\mathrm{R}_{\mathrm{Y}}^{\beta}\mathrm{R}_{\mathrm{X}}^{\alpha}\mathrm{K}^{-1}, \tag{1}$$

where $\mathbf{K}$ is the intrinsic matrix of the camera and $\alpha, \beta, \text{and } \gamma$ are the respective degrees of rotation in the $\mathrm{x}, \mathrm{y}, \text{and } \mathrm{z}$ directions, respectively. In [8], Zaheer et al. proposed the cost function, which estimates $\alpha$ and $\beta$ as rectifying homographies. We calculated the rectification homography of the most dominant plane in the scene by minimizing the cost function. We first assumed that two pairs of intersecting lines are orthogonal in the 3D space. Then, we estimated α and β, which can be uniquely determined for two different pairs.

From the rotational matrix $\mathbf{R} = \mathrm{R}_Z^{\gamma}\mathrm{R}_Y^{\beta}\mathrm{R}_X^{\alpha}$, the normal vector of the dominant plane was calculated as the product of the inverse of **R** and the z-axis; the other axes were similarly calculated. The three Manhattan directions in the 3D space can be calculated as follows:

$$\boldsymbol{n}_X = \mathrm{R}^{\mathrm{T}}[0,0,1]^{\mathrm{T}}, \tag{2}$$

$$\boldsymbol{n}_Y = \mathrm{R}^{\mathrm{T}}[0,1,0]^{\mathrm{T}}, \tag{3}$$

$$\boldsymbol{n}_Z = \mathrm{R}^{\mathrm{T}}[1,0,0]^{\mathrm{T}}. \tag{4}$$

We used the notations $X, Y,$ and $Z$ to denote the Manhattan directions. From these directions, three vanishing points can be calculated as follows:

$$\boldsymbol{v_x} = \mathrm{K}\boldsymbol{n_x}, \tag{5}$$

$$\boldsymbol{v_y} = \mathrm{K}\boldsymbol{n_y}, \tag{6}$$

$$\boldsymbol{v_z} = \mathrm{K}\boldsymbol{n_z}. \tag{7}$$

After identifying the directions, we assigned each line segment in the image to one or more Manhattan directions. We used cosine similarity to measure how well the line segment is fitted to the respective vanishing points. We created a virtual line $\mathbf{l_v}$, which connects the center point of a line segment and the vanishing point. Then, we computed the cosine similarity of the $\mathbf{l_v}$ and original line segment $\mathbf{l}$. The cosine similarity of the two lines is calculated as follows:

$$\cos\theta = \frac{\mathbf{l}\cdot\mathbf{l_v}}{\|\mathbf{l}\|\|\mathbf{l_v}\|}. \tag{8}$$

The absolute value of the cosine similarity is close to 1 because the line is close to the vanishing point. A line segment is assigned to the direction of the corresponding vanishing point if $|\cos\theta| \leq 0.9$. The line segment can be assigned to multiple directions because a Manhattan world image often consists of two horizontal vanishing points and one vertical vanishing point. In this case, the horizontal lines in the image can be connected to either the left or right side of the vanishing point in the 3D space. Therefore, if more than two directions satisfy the condition $|\cos\theta| \leq 0.9$ and the difference between them is less than 0.01, then we assign the line segment to both vanishing points.

## 3.3 Segment normal estimation

Using line segment direction information, we determined the normal of each segment in the segmented image. We solved the normal labeling problem in the energy minimization framework, and graph-cut optimization [12,13] was used. Boykov et al. [12] proposed a fast algorithm that finds an approximated solution to minimize the energy function. The algorithm has the following forms:

$$\mathrm{E}(f) = \sum_{\{p,q\}\in N} V_{p,q}(f_p, f_q) + \sum_{p\in P} D_p(f_p). \tag{9}$$

The function is described as the sum of two terms. The first term is called the smoothness cost, which formulates the relationship between neighboring segments. It is used to penalize the case in which neighboring sites have different labels. $V_{p,q}(f_p, f_q)$ is the cost when sites $p$ and $q$ are assigned to labels $f_p$ and $f_q$, respectively. The second term is called the data cost, which exploits the characteristics of site $p$ itself. Generally, minimizing this form of the energy function is equivalent to finding the maximum a posteriori solution of the MRF structure. We constructed the MRF structure using the oversegmented image from Section 3.1. Each segment, which is a polygon surrounded by line segments, becomes a site. Two segments are defined as neighbors if they share one or more common line segments.

Our data cost represents how well the surrounding line segments support the normal of the corresponding segment. For instance, if a segment is surrounded by x- and y-direction lines, then the normal of the segment should be in the z direction under the Manhattan world assumption. However, the surrounding line segment may not be the real surrounding line in the 3D space because the segment can be occluded by another segment. Our data cost comprises the information on the directions of the surrounding lines and occlusion information.

First, the line supporting score represents the weight or confidence of the line segment, which is defined for each surrounding line of each segment. The line supporting score (lss) for line segment $l$ surrounding segment $S$ is defined as follows:

$$lss(S,l) = \frac{\|l\|}{\sum_{l_s \in S}\|l_s\|} + \frac{\|l_{LSD}\|}{\|l\|}, \tag{10}$$

where $l_s \in S$ means that $l_s$ is a surrounding line of segment $S$. Hence, $\sum_{l_s \in S}\|l_s\|$ is the perimeter of segment $S$. $l_{LSD}$ is a portion of the line segment, which is detected from the LSD detector. We consider that line segments from the LSD detector are more reliable than extended line segments because extended line segments often show unexpected results. For example, when a line segment is extended beyond the border of a plane, the extended portion divides another plane into parts. A line segment has a higher supporting score if the line has a greater portion of its perimeter and if the line is originally detected by the LSD detector.

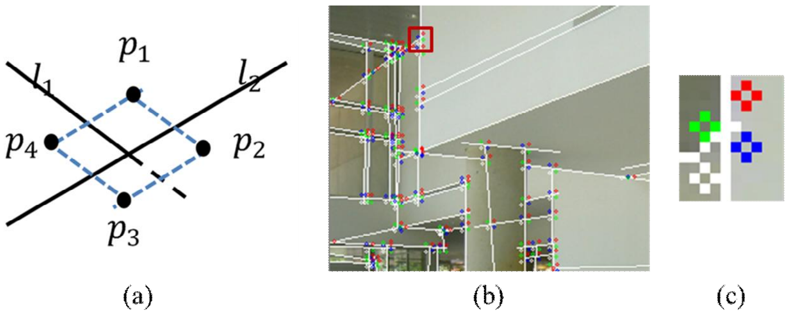

**Figure 2. (a) Pixel comparison for occlusion reasoning. Vertices of the parallelogram around the intersection point are sampled to compare the RGB values. (b),(c) Sampled points in real images are marked in green, blue, red, and white.**

Next, we defined the direction scores for each segment. Each segment has six directional scores. The Manhattan world has only three directions, which are denoted as x, y, and z. However, for line segments that are assigned to multiple directions, the direction scores are separately calculated. We refer to these directions as xy, yz, and zx. The xy direction is the direction score for line segments whose directions are assigned to both x and y. The direction score of segment $S$ with direction $d$ is defined as follows:

$$ds(S,d) = \sum_{l \in S \text{ AND } dir(l)=d} w_{occ} \cdot lss(S,l), \tag{11}$$

where $dir(l)$ denotes the direction assigned to line segment $l$, and $w_{occ}$ is the weight of the occlusion information. If line segment $l$ is occluding segment $S$, then $l$ is actually not a surrounding line segment of $S$. In this case, we set $w_{occ} = 0.5$ to penalize the occluding line segment. Then, $w_{occ} = 1$ was used. Figure 2 illustrates the occlusion inference method. We designed a simple occlusion detection algorithm by comparing the pixel values around the intersection of two line segments. The two lines divide the 2D plane into four parts. We sampled each part of the pixel value near the intersection point. Four points are the vertices of a parallelogram whose center point is the intersection point. These points are shown in Figure 2(a). Figure 2(b) shows how the points are sampled in a real image. The four points are represented by red, blue, green, and white dots for each intersection point. Figure 2(c) shows an example of an occlusion, which is a magnified image of the red box in Figure 2(b). Line segment $l_1$ is occluded by $l_2$ if the following condition is satisfied:

$$D_{eud}(p_1, p_4) \leq D_t \ AND \ D_{eud}(p_2, p_3) \leq D_t. \tag{12}$$

The occlusion inference depends only on the pixel values. We did not consider the geometric relationship of the line segment itself because it does not give reliable results in many cases due to unexpected expansion results.

The final data cost for labeling the normal of segment $S$ as $x$, y, and z is defined as follows:

$$D_S(x) = -w_{area} \cdot \{ds(S,y) + ds(S,z) + 0.5 \cdot ds(S,yz)\}, \tag{13}$$

$$D_S(y) = -w_{area} \cdot \{ds(S,x) + ds(S,z) + 0.5 \cdot ds(S,zx)\}, \tag{14}$$

$$D_S(z) = -w_{area} \cdot \{ds(S,x) + ds(S,y) + 0.5 \cdot ds(S,xy)\}. \tag{15}$$

A segment with line segments with directions assigned to both $y$ and $z$ can have $x$ as a normal. In this case, some line segments are in the $y$ direction, whereas the others are in the $z$ direction. Therefore, we multiplied 0.5 as the weight to the direction scores with multiple directions. $w_{area}$ is the weight of the area of the segment and is defined as follows:

$$w_{area} = 1.0 + \frac{\mathrm{n_s}}{100}, \tag{16}$$

where $n_s$ is the number of pixels of segment $S$. Because a high $w_{area}$ value stretches the gap between the data cost of different labels, a segment with a large area is less affected by the smoothness cost.

The smoothness cost is defined over the neighboring segments that share one or more common line segments. The main idea for designing the smoothness cost is that the directions of intersecting lines of two planes in a Manhattan world are determined by the normal of the planes. For instance, if a plane with x normal and a plane with y normal intersect, the intersection line should be in the z direction. Therefore, if neighboring segments have a different normal, then the direction of the intersecting line segment and the normal of neighboring segments should differ. The weight considering this concept of intersecting lines is used in the smoothness cost, which is defined as follows:

$$w_{dir} = \begin{cases} 0.1 & if\ normal(S) \neq dir(l_{neighbor}) \text{ and } normal(T) \neq dir(l_{neighbor}) \\ 1 & \text{otherwise} \end{cases}, \tag{17}$$

where $normal(S)$ refers to the normal of segment $S$ and $dir(l)$ refers to the direction of the line segment. $l_{neighbor}$ refers to the common line of segments $S$ and $T$.

Another weight for the smoothness cost is the ratio of the extended and non-extended line segments in neighboring line segments, which is similar to the line supporting score in the data cost. The weight, which is related to the neighboring line segment $w_l$, is defined as follows:

$$w_l = 1 - 0.9 * \left(\frac{\|l_{LSD}\|}{\|l\|}\right), \tag{18}$$

where $\|l\|$ and $\|l_{LSD}\|$ are the lengths of the common line segment and portion of the line segment extracted from the LSD, respectively. To balance the effect between line extension and line validation, we set the weight range of $w_l$ to 0.1–1.0. The value of $w_l$ decreases as the ratio of the original segment increases. Hence, the smoothness cost has a lower value for the original line segment and reduces the penalty for neighboring segments with different labels.

The final smoothness cost is defined as follows:

$$V_{S,T}(f_S, f_T) = \begin{cases} 0 & if\ f_S = f_T \\ w_{dir} \cdot w_l \cdot c_{smooth} & \text{otherwise} \end{cases}, \tag{19}$$

where $c_{smooth}$ is a constant that balances data and smoothness costs. We empirically set $c_{smooth} = 5$.

Combining the data cost and smoothness cost defined above, the entire energy function to be minimized is

$$\mathrm{E}(f) = \sum_{\{S,T\}\in N} V_{S,T}(f_S, f_T) + \sum_{S\in P} D_S(f_S). \tag{20}$$

We used the $\alpha-\beta-$ swap algorithm proposed in [12]. Conditions for the smoothness cost to use the $\alpha-\beta-$ swap were discussed in [12] and [13]. For any two labels $\alpha, \beta,$ the smoothness cost must satisfy the following conditions:

$$V(\alpha, a) + V(\beta, \beta) \leq V(\alpha, \beta) + V(\beta, \alpha). \tag{21}$$

Because our smoothness cost is 0 for the same labels, it satisfies the condition for using the $\alpha-\beta-$ swap. The normal estimation results are illustrated in Figure 1(c). The three Manhattan directions are shown in red, green, and blue.

## 3.5 Segment depth estimation

The depth estimation of connected planes can be formulated as a linear system in the following form:

$$\mathbf{Ad} = \mathbf{0}, \tag{22}$$

where $\mathbf{A}$ is the matrix containing the intersection point constraints and $\mathbf{d}$ is the column vector that contains the depth information of each plane. Matrix $\mathbf{A}$ should be a full-rank matrix to apply singular value decomposition and find a meaningful solution. We cannot directly apply the method to an oversegmented image because the matrix is a sparse matrix in most cases. To solve this problem, we developed a method for clustering segments with the same depth.

The fragmented segments must be connected to identify the actual intersection of the planes in the 3D world. We refer to this line as the “articulation line.” We used MST to cluster the connected planes and locate the articulation from the image. The graph structure is the same as that used in the graph-cut optimization. Each segment becomes a node of the graph, and an edge connects two nodes if two segments share a common line segment. If two segments are connected in the MST, then the two segments are connected in a real-world 3D space. If connected segments have a different normal, then the common line can be either an articulation line or an occluding line depending on the direction of the common line and normal of the segments. If the segments have the same normal, then they represent the same plane.

The weighting scheme for the MST is similar to the smoothness cost. Three criteria are considered when assigning the weight to each edge according to the relationship between the common line and neighboring segments. The first criterion is the weight of the articulation lines. We consider a common line of neighboring segments as an articulation line if the following condition is satisfied:

$$normal(p) \neq dir(l) \text{ and } normal(q) \neq dir(l) \text{ and } normal(p) \neq normal(q), \tag{23}$$

where $l$ is the common line segment of segments $p$ and $q$. This condition is the same as that used for the smoothness cost. The edges in this criterion have the lowest weights. The second criterion indicates the case in which the neighboring segments have the same normal. The segments adhering to this criterion may be on the same plane. Nevertheless, a plane may sometimes be occluded by another plane that has the same normal. To distinguish this case, we specified the weight by the portion that a common line segment shares with the planes. Finally, when the segments satisfy none of the above conditions, the highest weight is assigned to the edge of the segments. The final weighting scheme is defined as follows:

$$w_{pq} = \begin{cases} 1 - \dfrac{\|l_{valid}\|}{\|l\|} & if\ p, q, l\ satisfy\ condition\ (23) \\ 1 - \dfrac{\|l_{common}\|}{\max(\|l_p\|, \|l_q\|)} + 2 & if\ normal(p) = normal(q) \\ \infty & \text{otherwise} \end{cases}. \tag{24}$$

Different weights were assigned to the edges in the same criterion. $l_{valid}$ in the first criterion refers to the portion of the line segment that satisfies Equation (23). We formulated the MST algorithm using Kruskal's algorithm. After grouping the same plane using MST, segments were connected in the plane unit, and the linear system was constructed. We then estimated the depth of the planes up to scale, but only when all planes were connected. If an isolated segment exists, then the depth of the plane could be estimated incorrectly. The depth estimation results illustrated are illustrated in Figure 1(d). Brighter regions have greater depth values than darker regions.

## 4. Experimental results

We tested our method on the York Urban Database [14] and Manhattan world images. We compared our normal estimation method with those of the most similar methods to ours such as Lee et al. [6] and Zaheer et al. [8]. In Section 4.1, we outline the results, and in Section 4.2, we provide our reconstruction results. We excluded images that contained a significant portion of non-Manhattan structures, such as the sky, trees, or curved surfaces. Because our method highly depends on line segments, a sky detection or occlusion removal step should precede the process to address those images.

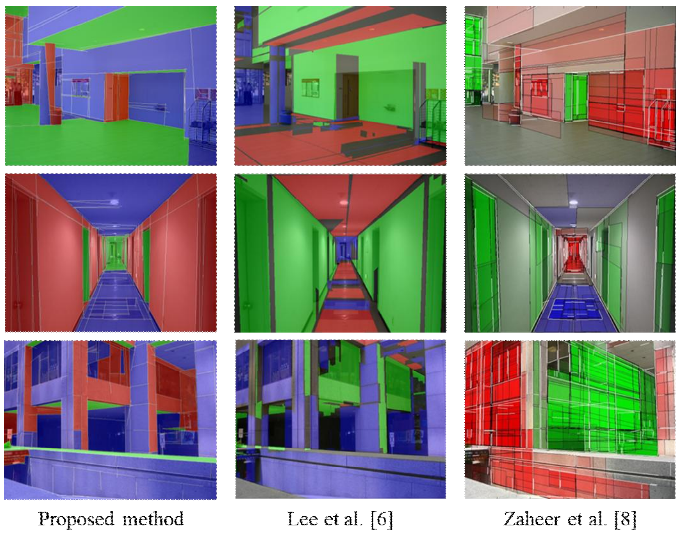

**Figure 3. Qualitative comparison of normal estimation. Only our method successfully captures detailed structures in all cases.**

### 4.1 Normal estimation results

In this experiment, we evaluated how well the proposed graph-cut optimization scheme estimates the normal of the planes. Similar SVR algorithms have slightly different assumptions, so it is difficult to directly compare these methods with ours. However, as the results demonstrate, our method performed better than the comparable methods in most cases. In particular, when the Manhattan world structure was complex, our method successfully captured detailed structures. Figure 3 presents a qualitative comparison of several Manhattan world images. Evidently, our method can effectively reconstruct the detailed structure of the Manhattan worlds.

### 4.2 3D reconstruction results

In this section, we present the 3D reconstruction results obtained after normal and depth estimations. In Figure 4, images in the left-most column are input images, and the remaining images are the reconstructed 3D models from various perspectives. Through our evaluation, we determined that the failure cases of our algorithm are primarily due to fragmented or over-extended line segments. Although our normal estimation method compensates for the oversegmented image using the smoothness cost, the fragmented line segments can enable different planes to occur in the same segments.

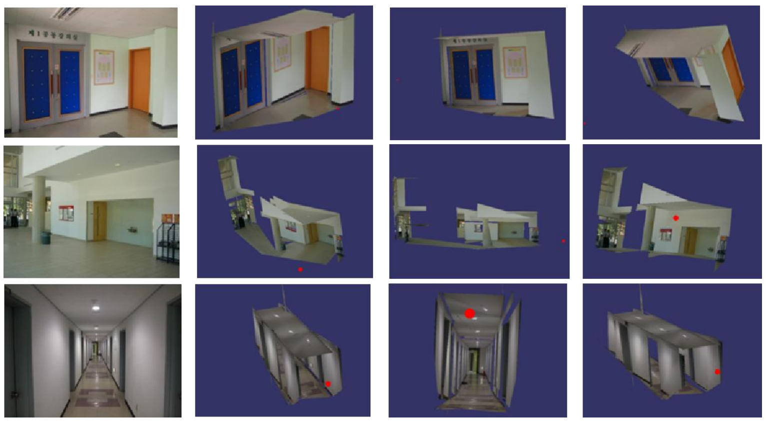
**Figure 4. 3D reconstruction result. The red dots in the reconstructed images denote the estimated location of the camera.**

## 5. Conclusions

In this paper, we propose an SVR algorithm that can be applied to Manhattan world images. After determining the Manhattan directions and vanishing points by exploiting the rotational matrix of the camera, we suggested a novel cost function for graph-cut optimization to estimate the normal of the planes in the image. We then used the MST to construct a linear system for depth estimation. The results of the proposed method showed that our algorithm can effectively reconstruct complex Manhattan world structures, which cannot be achieved with previous SVR methods.

The limitation of our work is that the result is highly dependent on the line detection results. A region where the line segment has not been detected will be handled in the same plane; consequently, the results may tend to be influenced by occluding objects or shade. Our future work will include an additional object- or background-removal step to produce a more accurate reconstruction result of outdoor scenes and a line refinement algorithm for pre-processing.

## References

[1] J.M. Coughlan and A.L. Yuille, "Manhattan world: Compass direction from a single image by Bayesian inference," in Proc. 7th IEEE International Conference on Computer Vision, pp. 941-947, Sep. 20-27, 1999. DOI: https://doi.org/10.1109/ICCV.1999.790349.

[2] P. Sturm and S. Maybank, "A method for interactive 3D reconstruction of piecewise planar objects from single images," in Proc. 10th British Machine Vision Conference, pp. 265-274, Sep. 13-16, 1999.

[3] P. Müller, G. Zeng, P. Wonka, and L. Van Gool, "Image-based procedural modeling of facades," ACM Transactions on Graphics, Vol. 26, No. 3, pp. 85-es, July 2007. DOI: https://doi.org/10.1145/1276377.1276484.

[4] D. Hoiem, A. Efros, and M. Hebert, "Automatic photo pop-up," ACM Transactions on Graphics, Vol. 24, No. 3, pp. 577-584, July 2005. DOI: https://doi.org/10.1145/1186822.1073232.

[5] A. Saxena, A, M. Sun, and A.Y. Ng, "Make3D: learning 3D scene structure from a single still image," IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 31, No. 5, pp. 824-840, May 2009. DOI: https://doi.org/10.1109/TPAMI.2008.132.

[6] D. Lee, M. Hebert, and T. Kanade, "Geometric reasoning for single image structure recovery," in Proc. 22nd IEEE Conference on Computer Vision and Pattern Recognition, pp. 2136-2143, June 23-25, 1999.
DOI: https://doi.org/10.1109/CVPR.2009.5206872.

[7] A. Flint, C. Mei, D. Murray, and I. Reid, "A dynamic programming approach to reconstructing building interiors," in Proc. 11th European Conference on Computer Vision, pp. 394-407, Sep. 5-11, 2010.
DOI: https://doi.org/10.1007/978-3-642-15555-0_29.

[8] A. Zaheer, M. Rashid, and S. Khan, "Shape from angle regularity," in Proc. 12th European Conference on Computer Vision, pp. 1-14, Oct. 7-13, 2012.
DOI: https://doi.org/10.1007/978-3-642-33783-3_1.

[9] C. Wu, J. Frahm, and M. Pollefeys, "Repetition-based dense single-view reconstruction," in Proc. 24th IEEE Conference on Computer Vision and Pattern Recognition, pp. 3113-3120, June 20-25, 2011.
DOI: https://doi.org/10.1109/CVPR.2011.5995551.

[10] S. Ramalingam and M. Brand, "Lifting 3D Manhattan lines from a single image," In Proc. 15th IEEE International Conference on Computer Vision, pp. 497-504, Dec. 1-8, 2013.
DOI: https://doi.org/10.1109/ICCV.2013.67.

[11] R.G. Von Gioi, J. Jakubowicz, J.M. Morel, and G. Randall, "LSD: a fast line segment detector with a false detection control," IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 32, No. 4, pp. 722-732, April 2010.
DOI: https://doi.org/10.1109/TPAMI.2008.300.

[12] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 23, No. 11, pp. 1222-1239, Nov. 2001.
DOI: https://doi.org/10.1109/34.969114.

[13] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?," IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 26, No. 2, pp. 147-159, June 2004.
DOI: https://10.1109/TPAMI.2004.1262177.

[14] P. Denis, J.H. Elder and F.J. Estrada, "Efficient edge-based methods for estimating Manhattan frames in urban imagery," in Proc. 10th European Conference on Computer Vision, pp. 197-210, Oct. 12-18, 2008.
DOI: https://doi.org/10.1007/978-3-540-88688-4_15.