Lifting 3D Manhattan Lines from a Single Image

Srikumar Ramalingam and Matthew Brand
Mitsubishi Electric Research Labs (MERL), Cambridge, MA, USA

{ramalingam,brand}@merl.com

Abstract

We propose a novel and an efficient method for reconstructing the 3D arrangement of lines extracted from a single image, using vanishing points, orthogonal structure, and an optimization procedure that considers all plausible connectivity constraints between lines. Line detection identifies a large number of salient lines that intersect or nearly intersect in an image, but relatively a few of these apparent junctions correspond to real intersections in the 3D scene. We use linear programming (LP) to identify a minimal set of least-violated connectivity constraints that are sufficient to unambiguously reconstruct the 3D lines. In contrast to prior solutions that primarily focused on well-behaved synthetic line drawings with severely restricting assumptions, we develop an algorithm that can work on real images. The algorithm produces line reconstruction by identifying 95% correct connectivity constraints in York Urban database, with a total computation time of 1 second per image.

1. Introduction

Consider Fig. 1(a) of an outdoor scene. Such man-made structures predominantly consist of lines in three orthogonal directions. It is easy to observe that several pairs of lines intersect. However, this does not necessarily mean that the corresponding 3D lines in the world also intersect. Lines that share a common vanishing point are a trivial counterexample; all appear to intersect at the vanishing point but none intersect in world space, where they are parallel. Knowing which apparent intersections correspond to real 3D intersections would be highly informative about the scene's 3D geometry. In fact, by identifying a minimal set of such intersections we can completely reconstruct the 3D lines up to an unknown global scale. However, there are several challenges in employing this strategy to infer the 3D geometry of lines. The most notorious one is occluding edges in the image producing several false intersections. Line detection algorithms in real images often miss important lines and produce spurious ones. Detected lines are often broken and cropped, obliterating evidence of intersections. In this paper, we propose a method that addresses these problems by identifying a few real 3D intersections as shown in Fig. 1(b) and use them as connectivity constraints to lift the 2D lines to 3D space. We show two perspective views of the reconstructed 3D lines in Fig. 1(c) and Fig. 1(d).

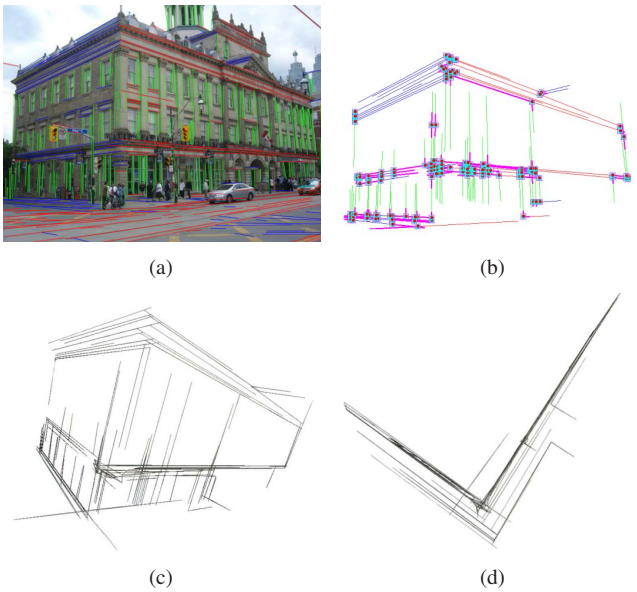

Figure 1. *(a) A typical image of an outdoor scene. The detected Manhattan lines and their intersections are shown on the image. In all the figures shown in this paper, red, green and blue lines denote the lines oriented along x, y and z directions respectively. (b) Intersections from several pairs of lines are shown on the image as squares. The pair of 3D lines corresponding to each one of these also intersect in 3D space. We automatically detect such intersections and use them as connectivity constraints to lift the 2D lines to 3D space as shown in two different perspective views (c) and (d).*

1.1. Related Work

Single view 3D reconstruction is a well-studied but distinctly unsolved problem in computer vision. Most of the techniques in this domain can be classified into purely ge-

ometrical and semantically driven approaches. Geometrical approaches have a lineage dating back a half century [16, 20, 23, 27, 30]. The most popular approach is based on labeling lines into convex, concave or occluding lines. The line labeling problem is in general NP-hard and several challenging line drawings were studied and novel constraint satisfaction algorithms were developed to solve the single view reconstruction problem. The algorithms were primarily tested on synthetic line drawings and ultimately proved too brittle for inputs derived from real images. Most of the other geometrical algorithms that show promising results on real images rely on some kind of user interaction [4, 26].

Recently, there has been a renewed interest in single view reconstruction (SVR) problem as more holistic approaches are starting to emerge. Hoiem refers to this as the geometric layout estimation where we classify the image pixels into several classes like sky, buildings and ground [12, 13]. This classification along with an estimation of surface orientations can produce 3D models that are sufficient for several applications like synthesizing walkthroughs, stereoscopic content generation for movies and 3D context for object detection. Along with several image features and weak assumptions on coplanarity and collinearity, Saxena et al. [24] developed an algorithm for depth estimation from a single image. Indoor scenes are particularly challenging due to the presence of clutter. Hedau et al. [11] approximated the room geometry using a cuboid and proposed an algorithm that samples different hypotheses and selects the best one. Being a severely ill-posed problem, SVR has led to several novel ideas such as the computation of orientation maps [18], inferring geometry from human activities [8] and even physics-driven stability and mechanical constraints [9]. Recently Schwing et al. [25] showed improvement in the performance of the indoor layout estimation algorithm using efficient inference techniques.

Most man-made scenes, both indoor and outdoor, satisfy Manhattan world assumption [3]. Delage et al. [5] used this assumption in an MRF framework to reconstruct indoor scenes. Constraints based on Manhattan grammar has been used for modeling buildings from aerial photos [28]. Our work is related to the geometrical constraints used in [19, 7]. In their work, corners are classified into concave, convex and occluding to build Manhattan worlds from single and multiple images. Structures like rectangles were detected in [10, 21], and they can be ordered according to their depth [32]. Example-based approaches have been used to reconstruct 3D scenes from line drawings [2]. Jain et al. [15] used connectivity constraints for reconstructing lines from multiple images. Recently, it was shown that junction features can be extracted from real images using an efficient voting scheme [22]. We use junctions for designing penalty terms in a linear programming (LP) formulation. However, our main focus is detecting intersection points that occur from only a pair of lines, and these intersection points need not be junctions.

1.2. Contributions

The main contributions of this paper are as follows:

- We build a graph that models connectivity constraints between nearby lines in real images.
- We use LP to lift the line segments in 3D space using Manhattan world assumption [3].
- Junction features are used to design the penalty terms in the LP.
- Our approach is computationally efficient taking about 1 second per image.
- We show automatic single view line reconstruction for challenging real images.

2. Lifting 2D lines to 3D

Early approaches to the interpretation of line drawing centered on labeling lines and detecting junctions [27, 29, 30]. Based on the orientations of the projected line segments in the image, junctions can be classified into an "alphabet" $\{\mathbf{L}, \mathbf{T}, \mathbf{Y}, \mathbf{W}, \mathbf{X}\}$. In Fig. 2(a), we show the line drawing of a tetrahedron with one truncated vertex. The points A, B and C correspond to $\mathbf{W}$ junctions and the points D, E and F correspond to $\mathbf{Y}$ junctions. It is straightforward to detect and classify junctions in synthetic line drawings. Once the junctions are detected, the incident lines are labeled using four labels ($+$,$-$,$\leftarrow$,$\rightarrow$). As shown in Fig. 2(a), $+$ denotes convex lines, $-$ denotes concave lines, and the arrows ($\leftarrow, \rightarrow$) denote occluding lines. The direction of the occluding label indicates that the plane to its right is the occluding plane. Although every line can take four possible labels, the adjacent junctions restrict certain labeling. Based on this restriction, Huffman [14] and Clowes [1] produced catalogs that provide all possible labeling for lines adjoining a given junction. Existing constraint satisfaction algorithms for the line labeling problem are generally successful on most synthetic line drawings, except a few pathological cases. Given the labeling, Sugihara proposed a lifting procedure to decide whether a line drawing represents a physically realizable polyhedral object or not. The basic idea is very simple: Using camera calibration, we know the projection rays for every pixel in the image. The line labels and junctions provide many constraints. For example, the quadruplet (A, B, E, D) lies on a single plane. The line labeling forces the point D to be closer to the camera than the point A. Such constraints can be written in the form of linear inequalities and if the linear program has a

feasible solution, then the line drawing is physically realizable. This implies that the 3D points can be lifted along the projection rays to form a physically realizable polyhedron as shown in Fig. 2(b). By studying the rank of this linear system, we can also understand if a line drawing can have multiple 3D interpretations. Sugihara's solution works for trihedral line drawings, where only three planes can intersect at a point. Whiteley extended Suhihara's approach to general line drawings using combinatorial structures like matroids [31].

In our work, we also produce a system of linear inequalities to lift the 3D points along the projection rays. Our formulation is quite different than Sugihara's because the real world images pose a completely different set of challenges that are hard to be accurately modeled using only geometrical and combinatorial structures. In real images, it is not easy to know which 3D points lie on a plane and which don't. Due to missing and spurious lines, we can not detect junctions with very high confidence. Therefore we cannot use planarity constraints or hard inequalities from detected junctions. Furthermore, real images have many false intersections that will always lead to infeasible solutions. To address these challenges, we do not derive any hard constraints from apparent junctions. Instead, evidence about junctions is used to inform the penalty terms in an LP to obtain a consistent 3D structure that "breaks" the smallest number of infeasible apparent junctions.

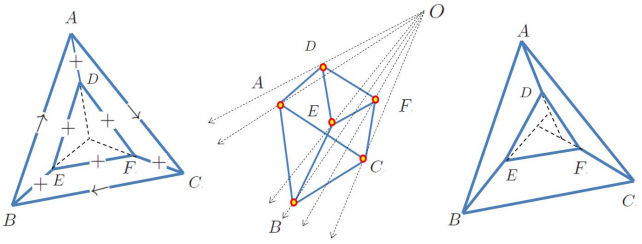

Figure 2. *Left: Line drawing of a physically realizable truncated tetrahedron. Middle: The lifting procedure where the 3D points are computed on the projection rays satisfying all the constraints from projections and junctions. Right: This line drawing is physically not realizable because the edges joining the two triangles do not concur at a point.*

3. 3D Manhattan Line Reconstruction

3.1. Camera Alignment

Our goal is to recover the 3D orthogonal line structure of a scene from an image. It is convenient to work in the world coordinate system where every 3D line is aligned along one of the three orthogonal axes. To achieve this, we compute the camera orientation with respect to the world and perform the necessary rotation, as per [3, 17]. First, we compute the three vanishing points $\mathbf{vp}_x$, $\mathbf{vp}_y$ and $\mathbf{vp}_z$ using the detected lines. Using two of the three vanishing points, we can compute the rotation R between the camera to the world coordinate system. This rotation is used to orient the camera rays such that the 3D line segments we reconstruct are aligned with the world. Once we compute the 3×3 camera matrix K, every pixel p_i in the image corresponds to a 3D projection ray in the world coordinate system that is given by the unit vector: $\mathbf{d}_i = \mathsf{RK}^{-1}\begin{pmatrix} x & y & 1 \end{pmatrix}^T$. Let d_{ix}, d_{iy} and d_{iz} denote the x, y and z components of the 3×1 direction vector $\mathbf{d}_i$.

3.2. Constraint Graph

We consider connectivity constraints between two line segments, if the shortest distance between them is less than a threshold. We consider two types of connectivity:

- Two orthogonal lines can (be extended to) intersect at a point, which we refer to as the intersection.
- Two collinear lines can (be extended to) meet at a point, which we refer to as incidence. As shown in Fig. 3, we can choose any point collinear to the lines as the incidence.

The intersections and incidences basically provide the necessary coupling constraints to lift the 2D lines to 3D space. In addition to merging broken line segments, the incidence relationship can also connect two lines coming from two different objects that are collinear in the world. Such relationships are essential to build a large connected graph. However, accidental collinearities are fairly common in real images. This is not a problem because our LP formulation is designed to handle constraints that may not be true, by using penalty terms based on the junction features.

3.3. Linear Program

We encode the scene interpretation problem in an LP as follows: Let n be the number of lines in an image and $l_i, i \leq n$ denote the i^{th} line. We can represent the intersection and incidence relationship using a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the vertices $\mathcal{V} = \{1, 2, ..., n\}$ represent the lines and the edges $(i, j) \in \mathcal{E}$ represent possible intersections or incidences between lines l_i and l_j. For example, in Fig. 3(a), we show four lines l_1, l_2, l_3 and l_4. These lines lead to three intersections I_{12}, I_{13} and I_{23}. Note that the intersection can happen between any two lines that belong to two different orientations. Accordingly, the lines l_4 and l_2 can intersect if the shortest distance between them is less than a threshold. Lines l_1 and l_4 are collinear and they lead to a point of incidence I_{14}. As shown in Fig. 3(b), the vertices and edges of the graph are given by $\mathcal{V} = (1, 2, 3, 4)$ and $\mathcal{E} = \{(1, 2), (1, 3), (1, 4), (2, 3)\}$ respectively.

The 3D point corresponding to every pixel p_i lies on its projection ray and it can be represented by $\lambda_i \mathbf{d}_i$, where λ_i

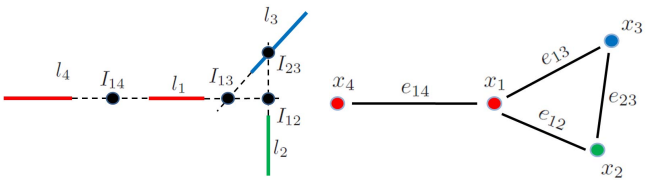

Figure 3. *(a) We consider four lines in the image space denoted by l_1,l_2,l_3 and l_4 that produce three intersection points I_{12}, I_{13} and I_{23}. For a pair of nearby collinear lines, we use a point that is collinear to both the lines as a point of incidence I_{14}. (b) We build a constraint graph where all the lines in the image are the vertices and all the intersections and incidences are edges.*

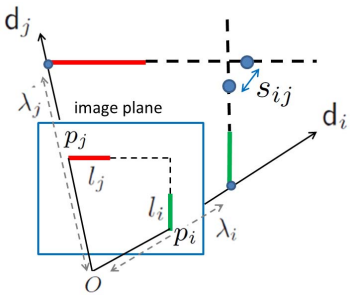

Figure 4. *We show the intersection of two lines l_i and l_j. The projection rays for two image pixels p_i and p_j, lying on lines l_i and l_j respectively, are also shown. The 3D points corresponding to these image pixels can slide anywhere on its projection rays and the two unknown parameters (λ_i, λ_j) denote the distances of these points from the camera center (O). The directions of these two points are given by $\mathbf{d}_i$ and $\mathbf{d}_j$. We parametrize a 3D line using one of the end points. As this point moves along the projection ray, the corresponding 3D line slides while maintaining its Manhattan orientation $\mathcal{D}_i$. The distance between the two 3D lines is given by s_{ij}.*

is an unknown depth parameter that is equal to the distance of the 3D point from the camera center. In general, to lift an image line, we require two parameters to reconstruct both its end points in 3D space. However, reconstructing one point is sufficient to recover the other one under Manhattan world assumption. Let p_i be the pixel corresponding to one of the end points of the line l_i. We use a single depth parameter λ_i to denote the distance of this end point from the camera center. Accordingly, the 3D point $\lambda_i \mathbf{d}_i$ encodes the position of the 3D line corresponding to line l_i. In Fig. 4, we show two lines l_i and l_j that intersect. Let the Manhattan direction of a line l_i be given by $\mathcal{D}_i$ where $\mathcal{D}_i \in \{x, y, z\}$. One can see that by varying the λ_i parameter we can slide the 3D line along the projection ray using one of its end points while maintaining the Manhattan direction $\mathcal{D}_i$. With respect to the constraint graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we have one unknown λ_i parameter for every vertex $i \in \mathcal{V}$. Every intersection or incidence relationship indicates that the two lines should intersect or be collinear. Each edge in the graph represents a potential constraint between two lines. Our goal is to find the unknown depth parameters, such that we satisfy the maximum number of such constraints as shown in the following optimization problem:

$$\begin{aligned} \min_{\lambda_i} \quad & \sum_{(i,j)\in\mathcal{E}} (||s_{ij}||_0) \\ \text{s.t} \quad & |\lambda_i d_{ia} - \lambda_j d_{ja}| \leq s_{ij}, a \in \{x, y, z\} \setminus \{\mathcal{D}_i, \mathcal{D}_j\} \\ & \lambda_i \geq 1, \ i \in \mathcal{V}. \end{aligned} \tag{1}$$

From a single image, the reconstruction is only possible up to an unknown global scale. The constraints $\lambda_i \geq 1$ ensures that the line segments are at least unit distance from the camera and also in front of the camera. We have two inequalities for every collinearity constraint because the set $\{x, y, z\} \setminus \{\mathcal{D}_i, \mathcal{D}_j\}$ has two elements given $\mathcal{D}_i = \mathcal{D}_j$. Accordingly, we have one inequality for the orthogonality constraint since $\mathcal{D}_i \neq \mathcal{D}_j$. We have one slack variable s_{ij} for every constraint. As shown in Fig. 4, minimizing the $L0$ norm of the slack variable s_{ij} is equivalent to depth-shifting and depth-bridging the smallest subset of 3D lines to make the scene graph connected and consistent in 3D.

Since this $L0$ norm minimization is NP-hard, we relax the problem to a sparsity-promoting $L1$ norm and solve for a minimum-weight set of depth shifts and bridges that makes the scene graph 3D-consistent:

$$\begin{aligned} \min_{\lambda_i} \quad & \sum_{(i,j)\in\mathcal{E}} (||w_{ij} s_{ij}||_1) \\ \text{s.t} \quad & |\lambda_i d_{ia} - \lambda_j d_{ja}| \leq s_{ij}, a \in \{x, y, z\} \setminus \{\mathcal{D}_i, \mathcal{D}_j\} \\ & \lambda_i \geq 1, \ i \in \mathcal{V}. \end{aligned} \tag{2}$$

The weight parameters w_{ij} in the objective function present an opportunity to incorporate more evidence from the image, which is obtained from junction features and will be explained in Sec. 3.4. The LP provides a solution for all slack variables s_{ij}. For every edge (i, j) the slack variables s_{ij} give us the minimum depth separation between line i and line j needed to obtain the lowest-cost globally consistent 3D interpretation of the scene. To then extract a minimal-cost fully-connected 3D structure from the LP solution, we use each slack value s_{ij} as the edge cost and compute the minimal spanning tree (MST) as shown in Fig. 5(c). The depth parameters λ_i can be computed directly from the linear system that corresponds to the tree-structured graph. The various stages of the algorithm are shown in Fig. 5.

3.4. Junction-breaking Costs

Under the Manhattan world assumption, an efficient voting scheme was introduced recently for computing the junc-

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

Figure 5. *The various stages of our automatic single view line reconstruction algorithm. (a) Given an input image, we extract lines, compute vanishing points and cluster the lines according to three principal orientations. In (b), we show the intersections of the Manhattan lines in the constraint graph . We solve the LP and use the solution to select a subset of constraints using a minimal spanning tree (MST) algorithm. The selected subset is shown in (c). Using the MST we compute the λ_i parameters using a linear system. These depth parameters are sufficient to lift the 2D lines to 3D space in the world coordinate system. Two different perspective views of the reconstructed lines are shown in (d) and (e).*

tion features [22]. In our work, we use these junction features for designing the penalty terms in the LP. The weight parameters w_{ij} in Eqn. 2 corresponds to the penalty we impose if the connectivity constraint corresponding to the edge (i, j) is violated and relaxed to a depth-bridge. These weights are modeled as a function of junction features. Since junctions play an important role in our work, we show Fig. 6 to illustrate the idea behind junctions and how to compute them. For every pixel in the image, there are 6 possible oriented regions with respect to the vanishing points as shown in Fig. 6. These orientations are denoted by the set $S = \{\overrightarrow{x}, \overleftarrow{x}, \overrightarrow{y}, \overleftarrow{y}, \overrightarrow{z}, \overleftarrow{z}\}$. Every subset $A \subseteq S, |A| \geq 2$, denotes a junction and one can compute a function $\mathcal{F}(A, p)$ for every possible subset A and pixel p. The value of the function gives a score to indicate the presence of a junction A at pixel p. For example, the **Y** junction at pixel p in Fig. 6 corresponds to the set $\{\overrightarrow{x}, \overrightarrow{y}, \overrightarrow{z}\}$. If the function has a high value for a particular junction, then it is very unlikely that it may have a high score for a different junction. In Fig. 7, we show pixels where there is non-zero junction scores for various types of junctions. The distribution of junctions depends on the scene geometry. It is easy to observe the following behavior from junctions on Manhattan scenes.

- **L** and **X** junctions occur on planar surfaces.
- **T** junctions occur on both planar surfaces and occluding boundaries.
- **Y** and **W** junctions are common on convex and concave edges.

Let $\mathcal{J}(p)$ denote the type of junction at pixel p. Let p denote the intersection point that maps to edge (i, j) in the constraint graph. In the LP, we set the weight for w_{ij} as shown below:

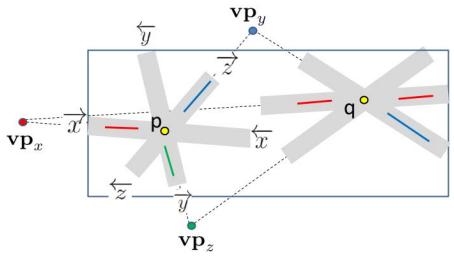

Figure 6. *The basic idea in computing junction features [22]. For a pixel p, we consider 6 oriented regions ($\{\overrightarrow{x}, \overleftarrow{x}, \overrightarrow{y}, \overleftarrow{y}, \overrightarrow{z}, \overleftarrow{z}\}$) with respect to the three vanishing points $\mathbf{vp}_x$, $\mathbf{vp}_y$ and $\mathbf{vp}_z$. Depending on the presence of lines in each of these 6 regions, we can detect different junctions. We show a* **Y** *junction at pixel p and a* **T** *junction at pixel q.*

$$w_{ij} = \begin{cases} \mathcal{C}_h & \text{if } \mathcal{J}(p)\text{=}\mathbf{Y} \text{ or } \mathbf{W} \\ \mathcal{C}_m & \text{if } \mathcal{J}(p)\text{=}\mathbf{X} \\ 0 & \text{if } \mathcal{J}(p)\text{=}\mathbf{T} \\ 1 & \text{otherwise.} \end{cases}$$

$\mathcal{C}_h$ and $\mathcal{C}_m$ are constants that are scaled according to the quality of visual evidence for each junction. We did not give any preference to **L** junctions because **T** can sometimes be detected as **L** due to missing lines.

4. Experiments

Dataset and Parameters: We tested our algorithm on images from York Urban database [6]. The images are of size 640×480 and the camera calibration is included in the database. We considered intersection points between two line segments if the minimum distance between the line segments is less than 40 pixels. The collinearity constraint is considered for lines that are separated by as much as $1/4$ of the image. After the line detection, we find candidate connections between lines by finding intersections and incidences. We then solve the LP and compute its MST to find the largest connected component in the constraint graph and reconstruct only the lines in this component. We could

Figure 7. *We show the prominent junctions detected in two outdoor images. First column displays* **L** *junctions. The second displays* **T** *junctions. The third column displays* **X** *junctions, while the last column displays both* **Y** *and* **W** *junctions.*

also reconstruct other connected components up to different scale factors. The results shown and evaluated in this paper are based on the maximum connected component. Based on the detected junctions we used $\mathcal{C}_h = 10$ and $\mathcal{C}_m = 5$. The average statistics for the constraint graph is given in Table 1.

Quantitative Evaluation: We used two different methods for quantitative evaluation. In the first method, we manually marked the correct and incorrect constraints in 93 images in York Urban database. We identify a minimum number of edges (constraints) to unambiguously lift the lines to 3D space using a minimum spanning tree (MST) algorithm. The quantitative evaluation is based on the number of incorrect constraints with respect to the total number of constraints in the MST as shown in Fig. 8. The mean error is 4.81%.

Our second method measures how well our algorithm obtains a globally consistent interpretation of the visual evidence. Recall that our goal is to identify a maximal set of pairwise constraints that spans the detected lines and supports a 3D reconstruction without error (ignoring small VP, line, and calibration errors). This is NP-hard, so we construct a convex relaxation using the $L1$ optimization of the LP to bound the (intractable) L0 optimization of the number of consistent constraints. The LP computes the λ_i parameters and obtains line reconstruction. We also compute the λ_i parameters using the linear system based on the constraints identified by the MST algorithm. A geometric match between the (relaxed) LP-based reconstruction and the (exact) MST-based reconstruction is a proof that the LP has found a globally consistent interpretation of the evidence with a fully connected set of completely satisfied geometric constraints. In 87 out of 102 images, the maximum disagreement between results of the spanning tree reconstruction and the results of the LP reconstruction was less than 5% of the total induced depth of the scene.

Qualitative Evaluation: We show the line reconstruction results in Fig. 9. For qualitative evaluation, we also tested images from Flickr. The camera parameters for Flickr images were obtained using vanishing points. Most of the reconstructions looked correct when viewed with a VRML viewer. The VRML models are included in the Supplementary Materials. A few challenging cases are shown in Fig. 10.

Mean	Value
Detected lines	235
Lines in max. connected component	152
Intersections	659
Incidences	27

Table 1. The average statistics for York Urban Database.

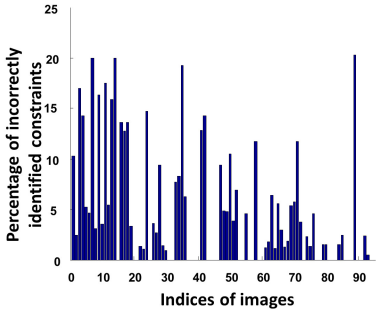

Figure 8. *Quantitative Evaluation: Using the minimum spanning tree (MST) algorithm, we identify a minimum number of constraints to unambiguously lift the lines to 3D space. In this graph, we show the percentage of incorrectly identified edges with respect to the total number of edges in the MST.*

Computation time: All the algorithmic components are implemented in C++ and we use CVX for the LP. Overall, the algorithm takes about 1 second per image.

5. Discussion

The success of recent methods can be attributed to the use of important features [13, 18] and priors on the scene geometry [11]. The classical approaches considered more general and challenging polyhedral scenes [16, 20, 23, 27, 30]. Despite several decades of research, the generality of such techniques posed severe challenges to their application on real images. We rely on Manhattan assumption. Otherwise, our problem formulation is also general and applies to any connected set of lines and these could come from non-planar regions like lampposts or even cables running between buildings. However in practice, as shown in Fig. 10, the performance degrades in the case of too much clutter, which leads to missing and spurious lines. Constraints from classical algorithms [27] or features like geometric context [13] can provide additional penalty terms in our LP to decrease the number of incorrect edges.

Lee et al.[18] computes orientation map for every pixel

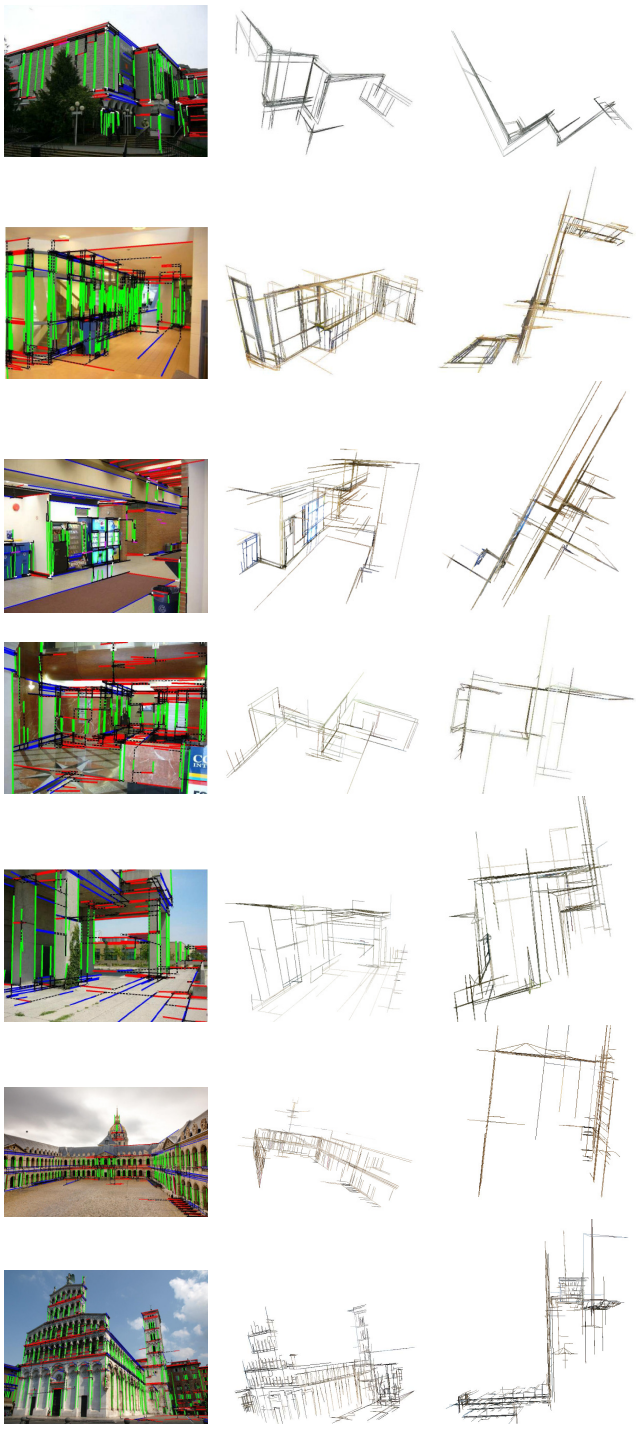

Figure 9. *Line Reconstruction Results. The first column displays the original images with the Manhattan lines marked. The second and third columns show perspective views of the 3D lines. During the visualization of the 3D lines, we used the same color as they appear on the image.*

using the nearby lines. We take a more global approach

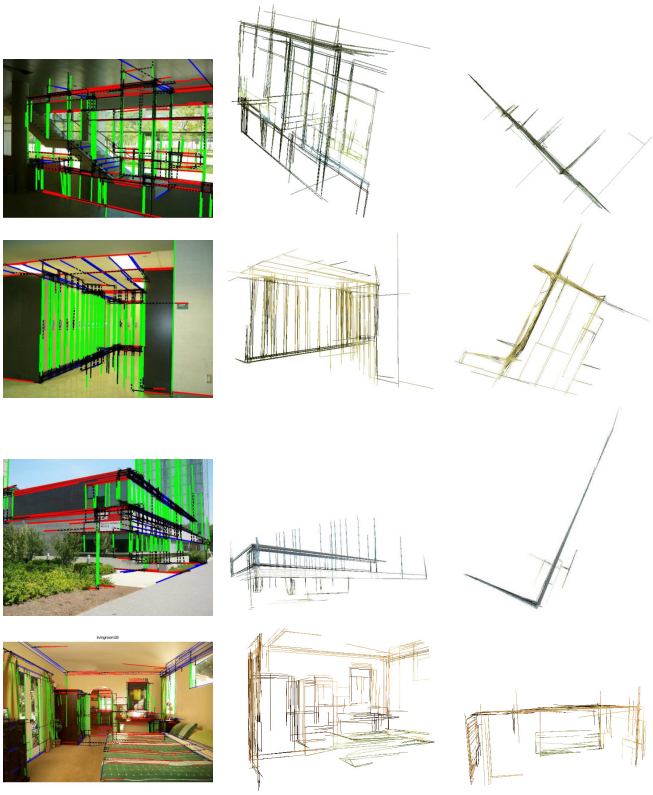

Figure 10. *Failure cases in the line reconstruction. In the first row, missing lines along the z direction led to the collapse of the two parallel planes to a single plane. In the second row, spurious lines from the reflections on the shiny floor are mistaken for vertical structure in the scene. In the third row, missing horizontal lines on the background building deprives the LP of evidence that it is a separate structure. In the last row, incorrect focal length estimation using the vanishing points led to smaller estimates for the depth parameters.*

by considering all plausible connectivity constraints between lines and try to find a consistent interpretation of the scene. Such an approach can improve the existing techniques for orientation maps by giving evidence about occluding boundaries. In the case of scenes containing non-Manhattan lines, the algorithm either ignores or approximates them using Manhattan ones as shown in Fig. 11. One of the main advantages of our algorithm can be seen in the reconstruction shown in Fig. 12. On a careful observation, we can also see a second door on a parallel wall behind the right wall. The line reconstruction can recover such partially hidden structures. Due to planarity, smoothness, and the opaqueness assumptions typically employed in many approaches, existing techniques will find it difficult to understand such structures.

Acknowledgments: We thank J. Thornton, S. Miwa, J. Yedidia, P. Torr, K. Alahari, A. Jain, J. Pillai, Y. Taguchi, P. Varley, ACs and anonymous reviewers for useful discus-

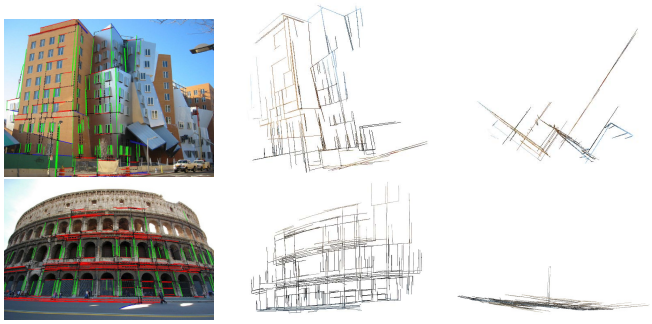

Figure 11. *Typical behavior of the line reconstruction algorithm in non-Manhattan scenes. In the first row, MIT Stata center is approximated with Manhattan 3D lines. In the second row, the curved surface of the Colosseum is approximated by a single plane.*

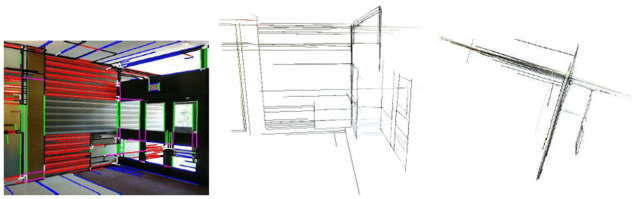

Figure 12. *Left: We show an image of an indoor scene with two orthogonal walls. In the middle and right, we show two different perspectives of the line reconstruction. Note that the line reconstruction recovers the second door behind the right wall. Understanding such structures would be challenging for many existing techniques. [Best viewed by zooming]*

sions and constructive feedback.

References

[1] M. B. Clowes. On seeing things. *AI*, 1971.

[2] F. Cole, P. Isola, W. T. Freeman, F. Durand, and E. H. Adelson. Shapecollage: Occlusion-aware, example-based shape interpretation. In *ECCV*, 2012.

[3] J. Coughlan and A. Yuille. Manhattan world: Compass direction from a single image by bayesian inference. In *ICCV*, 1999.

[4] A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. *IJCV*, 2000.

[5] E. Delage, H. Lee, and A. Ng. Automatic single-image 3d reconstructions of indoor manhattan world scenes. In *ISRR*, 2005.

[6] P. Denis, J. Elder, and F. Estrada. Efficient edge-based methods for estimating manhattan frames in urban imagery. In *ECCV*, 2008.

[7] A. Flint, D. Murray, and I. Reid. Manhatten scene understanding using monocular, stereo, and 3D features. In *ICCV*, 2011.

[8] D. Fouhey, V. Delaitre, A. Gupta, A. Efros, I. Laptev, and J. Sivic. People watching: Human actions as a cue for single view geometry. In *ECCV*, 2012.

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

[10] F. Han and S.-C. Zhu. Bottom-up/top-down image parsing with attribute grammar. *PAMI*, 2009.

[11] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, 2009.

[12] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. *ACM Trans. Graph.*, 2005.

[13] D. Hoiem, A. A. Efros, and M. Hebert. Recovering surface layout from an image. *IJCV*, 2007.

[14] D. A. Huffman. Impossible objects as nonsense sentences. *Machine Intelligence*, 1971.

[15] A. Jain, C. Kurz, T. Thormahlen, and H. Seidel. Exploiting global connectivity constraints for reconstruction of 3d line segment from images. In *CVPR*, 2010.

[16] T. Kanade. A theory of origami world. *AI*, 1980.

[17] J. Kosecha and W. Zhang. Video compass. In *ECCV*, 2002.

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

[19] D. Lee, M. Hebert, and T. Kanade. Geometric reasoning for single image structure recovery. In *CVPR*, 2009.

[20] J. Malik. Interpreting line drawings of curved objects. *IJCV*, 1987.

[21] B. Micusik, H. Wildenauer, and J. Kosecka. Detection and matching of rectilinear structures. In *CVPR*, 2008.

[22] S. Ramalingam, J. Pillai, A. Jain, and Y. Taguchi. Manhattan junction catalogue for spatial reasoning of indoor scenes. In *CVPR*, 2013.

[23] L. Roberts. *Machine perception of three-dimensional solids*. PhD thesis, MIT, 1963.

[24] A. Saxena, S. H. Chung, and A. Y. Ng. 3-D depth reconstruction from a single still image. *IJCV*, 2008.

[25] A. G. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun. Efficient structured prediction for 3D indoor scene understanding. In *CVPR*, 2012.

[26] P. Sturm and S. Maybank. A method for interactive 3d reconstruction of piecewise planar objects from single images. In *BMVC*, 1999.

[27] K. Sugihara. *Machine Interpretation of Line Drawings*. MIT Press, 1986.

[28] C. Vanegas, D. Aliaga, and D. Benes. Building reconstruction using manhattan-world grammars. In *CVPR*, 2010.

[29] P. Varley. *Automatic Creation of Boundary-Representation Models from Single Line Drawings*. PhD thesis, Cardiff University, 2003.

[30] D. Waltz. Generating semantic descriptions from line drawings of scenes with shadows. *Technical Report, MIT*, 1972.

[31] W. Whiteley. A matroid on hypergraphs, with applications in scene analysis and geometry. *Discrete and Computational Geometry*, 1989.

[32] S. Yu, H. Zhang, and J. Malik. Inferring spatial layout from a single image via depth-ordered grouping. In *Proc. Workshop on Perceptual Organization in Computer Vision*, 2008.