Line-Based Structure From Motion for Urban Environments

Grant Schindler, Panchapagesan Krishnamurthy, and Frank Dellaert
Georgia Institute of Technology
College of Computing
{schindler, kpanch, dellaert}@cc.gatech.edu

Abstract

We present a novel method for recovering the 3D-line structure of a scene from multiple widely separated views. Traditional optimization-based approaches to line-based structure from motion minimize the error between measured line segments and the projections of corresponding 3D lines. In such a case, 3D lines can be optimized using a minimum of 4 parameters. We show that this number of parameters can be further reduced by introducing additional constraints on the orientations of lines in a 3D scene.

In our approach, 2D-lines are automatically detected in images with the assistance of an EM-based vanishing point estimation method which assumes the existence of edges along mutally orthogonal vanishing directions. Each detected line is automatically labeled with the orientation (e.g. vertical, horizontal) of the 3D line which generated the measurement, and it is this additional knowledge that we use to reduce the number of degrees of freedom of 3D lines during optimization. We present 3D reconstruction results for urban scenes based on manually established feature correspondences across images.

1 Introduction

We are interested in building line-based 3D models of urban environments from multiple images taken at or near ground level and separated by a wide baseline. We show that there is a large amount of knowledge about the 3D structure of urban scenes that we can exploit at every step of the reconstruction process, from detecting features, to matching features, to finally optimizing the 3D structure of the scene.

The problems of urban scene reconstruction, line-based structure from motion, and the related issue of 3D line representation have received a large amount of recent attention. Architectural reconstruction has been approached via both manual [10] and automatic [13, 5] methods, many of which produce polyhedral models as output and use a mix of point- and line-features for matching and reconstruction. Werner and Zisserman present a well-developed method for automated architectural reconstruction across image triplets in [13]. They perform a preliminary step in which lines are classified according to principal orthogonal directions, and these classifications are used to constrain 3D line estimation. A line-based reconstruction method was presented in [3] which extracts and matches line features across image triplets under trifocal constraints. The method recursively updates structure over an extended sequence of images separated by a small amount of motion. More recently, Rother presented a linear reconstruction method for lines using a reference plane [8]. Metric reconstruction can be achieved using a virtual reference plane, such as that provided by assuming three mutually orthogonal vanishing directions in a scene. Bartoli and Sturm [2] developed an orthonormal representation for 3D lines which they show can be updated during non-linear minimization with the minimum 4 parameters. They list several scenarios, including the case of fixed 3D line direction, in which prior knowledge can be exploited by optimizing over a subset of the 4 update parameters.

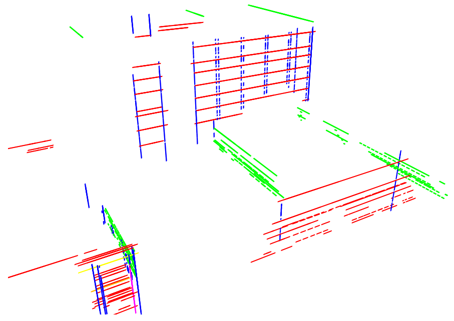


Figure 1: 3D reconstruction based on automatically detected and classified lines in 11 images. Optimization over structure and motion takes advantage of additional knowledge provided by line classification.

We fully explore this idea from [2] by enumerating the specific types of prior knowledge we can exploit in the case of urban environments and demonstrating how to acquire

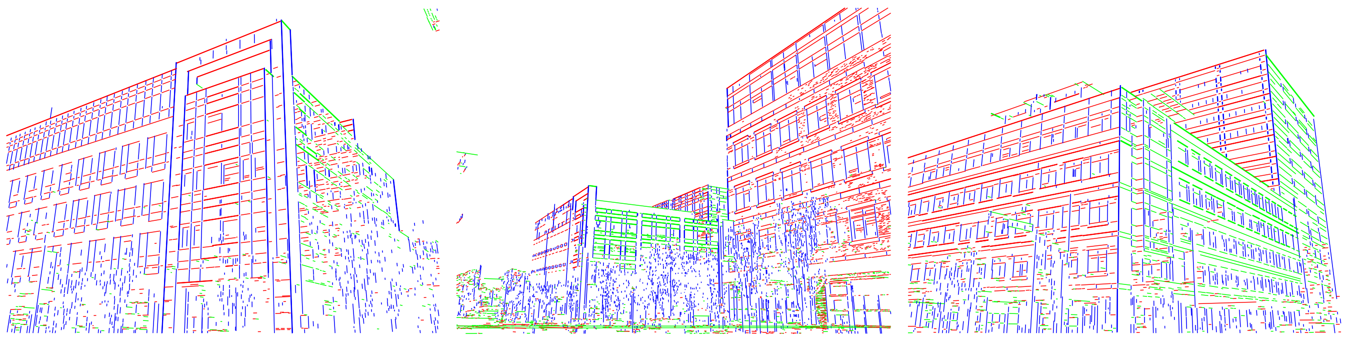


Figure 2: 2D line features are automatically detected by grouping edge pixels according to vanishing directions.

this knowledge directly from images. Specifically, we extend the framework of [11] to develop distinct differentiable mappings for a each type of 3D line which we encounter in urban scenes. The major contribution of this paper is to demonstrate how knowledge about the structure of 3D lines in urban environments can be used to improve the processes of feature detection, feature matching, and structure optimization. An illustrative example of our approach is that given a specific pixel in an image, we will already know most of the parameters of the 3D line which generated this pixel before we have detected 2D lines in the image, matched 2D lines across images, or performed any structure optimization on the 3D lines. Thus, we are free to use this 3D knowledge throughout each of the remaining steps in the reconstruction.

2 2D Line Features

The first step toward a line-based 3D reconstruction is the detection of 2D lines in photographs. We are interested specifically in detecting lines that correspond to real 3D structure lying along dominant orthogonal vanishing directions. In the case of images separated by a wide baseline, both manual [8] and automatic [1] methods of line detection and matching have been employed. We adopt the approach of [4, 9] to classify image edges at the pixel level and then group pixels into a large number of line segments, each labeled with the direction of the corresponding 3D line.

2.1 Pixel Classification with Vanishing Points

Our 2D line detection method is based on the assumption that the 3D scenes depicted in our images have at least 3 mutually orthogonal vanishing directions. These vanishing directions are represented as a set of 3D vanishing points VP where we always fix the first three points as $VP_{1..3} = \{[0,0,1,0]^T, [0,1,0,0]^T, [1,0,0,0]^T\}$. Given an image I, we would like to estimate a label λ for each pixel in the image to indicate that the pixel was generated by a 3D line vanishing at VP_λ. In order to do this, we must also estimate any additional 3D vanishing points $VP_{4..N}$ (for a scene with N vanishing directions), a camera calibration $C = \{K, R\}$ with rotation R and intrinsic parameters K.

We briefly outline the expectation-maximization method of [9] used to estimate vanishing point assignments for every pixel in the image I. In the E-Step, given fixed camera calibration C and fixed set of vanishing points $VP_{1..N}$, for each image pixel we compute the probability that it belongs to an edge oriented along each vanishing direction. In the M-Step, the camera parameters C and vanishing points $VP_{4..N}$ are optimized based on the computed distributions over each pixel's vanishing direction. After EM converges, we compute an image I_λ containing the *maximum a posteriori* vanishing point label for each pixel in the original image I. For pixels which do not fall on any edge in the image, or which fall on edges not oriented along a dominant vanishing direction, we set the value of λ to OFF or $OTHER$, respectively. Note that we can also project the 3D vanishing points $VP_{1..N}$ into the recovered camera C to obtain the set of 2D vanishing points $vp_{1..N}$.

2.2 2D Line Detection

The features we detect from our images are labeled 2D line segments $f = \{(x_1, y_1), (x_2, y_2), \lambda\}$ parameterized by two endpoints and a label λ indicating the vanishing point VP_λ of the 3D line to which the feature corresponds. We begin extracting 2D line features by first performing connected components analysis on the resulting image I_λ from Section 2.1, grouping neighboring pixels that have the same vanishing point label λ. For each connected component, we create a corresponding feature f which takes on the common label λ of pixels in the component. To determine the endpoints of the line segment, we first compute the line l_{vp} joining the centroid c of the pixels in a component to the 2D vanishing point vp_λ corresponding to the component's label. The endpoints $(x_1, y_1), (x_2, y_2)$ of the detected line segment f are computed by intersecting the line $l_{vp}$with the bounding box of pixels in the connected component. The result is a set of labeled line segments oriented precisely along the image's vanishing directions.

The significance of the label λ for each 2D feature is that it directly determines the class of the corresponding 3D line, and as we we will see, the number of free parameters in the structure. All 2D line features for which $\lambda = 1$ correspond to vertical 3D lines, while those with $\lambda = 2..N$ correspond to horizontal 3D lines. Note that when $\lambda = OTHER$, the corresponding 3D line is unconstrained, and we determine the line l_{vp} according to the the method in [7]. We ignore connected components for which $\lambda = OFF$ as they, by definition, do not correspond to edges in the image.

2.3 2D Line Matching

The establishment of correspondences between 2D lines across images is a necessary step in our 3D reconstruction. A standard approach similar to [1] is to use RANSAC to find a set of line correspondences which satisfy trifocal constraints [6] across image triplets. RANSAC demands a set of putative correspondences over which to sample, and in the case of small motion over sequential images, potential feature matches are usually constrained to lie within a local neighborhood of each other. In the case of wide baseline matching, we cannot take advantage of the physical proximity of features. Instead we can constrain the set of putative correspondences by matching line features f against only those lines with the same vanishing point label λ, thus reducing the combinatorial complexity of the matching process and allowing RANSAC to draw fewer samples. Once we have established matchings between sets of line features, we consider corresponding features to be *measurements* of the same 3D line.

3 Reconstruction

We formulate the reconstruction problem as in [12], in terms of an objective function $\mathcal{O}$ that measures the total squared distance between the observed line segments (measurements) and the projections of the reconstructed lines in the image, minimizing $\mathcal{O}$ with respect to the camera and structure parameters. We differ from the approach of [12] in our representation of 3D lines (structure), the computation of error function in the image (rather than the image plane), and use of non-linear least squares optimization to minimize $\mathcal{O}$. We extend the work of C.J. Taylor and D.J. Kriegman [11] by defining different local parameterizations for several classes of 3D lines in light of available prior knowledge about the scene.

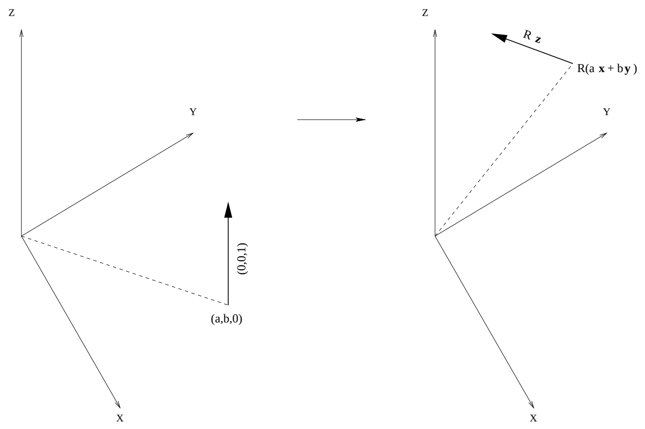


Figure 3: 3D Line Representation. The line can be visualized as being initially (left) parallel to the Z-axis and passing through the point (a, b) (point on the line closest to the origin) in the XY-plane, and then rotated by R (right) into its new position (passing through $R(a\hat{\mathbf{x}} + b\hat{\mathbf{y}})$) and orientation (along the direction $R\hat{\mathbf{z}}$).

3.1 3D Line Geometry

We represent lines in $\mathbb{R}^3$ by the set of tuples $\mathcal{L} = \{\langle R, (a, b)\rangle \in SO(3) \times \mathbb{R}^2\}$ (see Figure 3). $\mathcal{L}$ defines an algebraic set which is a 4-dimensional manifold embedded in $SO(3) \times \mathbb{R}^2$.

3.1.1 The Projection Function

We project 3D lines down to 2D lines in an image via the usual steps of transformation of the 3D line into the camera coordinate frame, perspective projection of the transformed line onto the image plane at z=1, and bringing the projection into image space according to the camera calibration.

1. *Transformation into camera coordinate frame*
A line in the world coordinate frame $\langle R_w, (a_w, b_w)\rangle$ is transformed according to the rotation R_c^w and translation t_c^w of a camera in world coordinates. During the transformation, the rotation R_w is affected only by the orientation of the camera R_c^w and (a_w, b_w) is affected only by the translation t_c^w of the camera. Thus, we can obtain the line $\langle R_c, (a_c, b_c)\rangle$ in the camera coordinate frame as follows:

$$\begin{aligned} R_c &= R_c^w R_w \\ (t_x, t_y, t_z) &= (R_w)^T t_c^w \\ (a_c, b_c) &= (a_w - t_x, b_w - t_y) \end{aligned}$$

2. *Perspective Projection*
We represent projections in $\mathbb{R}^2$ as the set of homogenous vectors $\ell = \{\langle m_x, m_y, m_z\rangle : m_x x + m_y y +$

3D Line Class	Degrees of Freedom	Free Parameters	Exponential Mapping
general	4	$\alpha, \beta, \omega_x, \omega_y$	L_g
horizontal	3	α, β, ω_z	L_{h3}
vertical	2	α, β	L_v
horizontal (fixed direction)	2	α, β	L_{h2}
fixed	0	-	-

Table 1: 3D Line Classes. We define different local parameterizations for a number of line classes. By classifying lines in images of urban scenes, we reduce the degrees of freedom of the 3D lines over which we optimize.

$m_z = 0\}$. We compute the projection $m_{cip} \in \ell$ in the image plane as the cross product of two vectors [12]: $R_c\hat{\mathbf{z}}$ and $R_c(a_c\hat{\mathbf{x}} + b_c\hat{\mathbf{y}})$, respectively the direction of the 3D line in the camera coordinate frame and the point on the line closest to the origin. The result of this cross product is:

$$m_{cip} = a_c R_{c2} - b_c R_{c1}$$

where R_{c1} and R_{c2} denote the first and second columns of R_c respectively.

3. *Image Space*
Given the internal calibration matrix K of the camera, the projection m_{ci} (in the image) corresponding to the projection m_{cip} (in the image plane) can be computed as:

$$m_{ci} = (K^T)^{-1} m_{cip}$$

3.1.2 The Objective Function

We define our objective function $\mathcal{O}$ as the sum, over all images and 3D lines, of the errors between the projection of a 3D line and the corresponding measurement in an image [12], computing this error in the image, rather than the image plane as in [12].

3.2 Optimization

Our representation parameterizes a 3D line with 11 parameters (9 for R and 2 for (a, b)) while it has only 4 degrees of freedom (DOF). This causes singularity issues when we try to perform non-linear optimization over the lines. We avoid such singularities by constructing a local parameterization [11] around any point $\langle R, (a, b)\rangle \in \mathcal{L}$ mapping an open region in $\mathbb{R}^4$ (in the case of generic 3D lines) onto a local neighborhood of $\langle R, (a, b)\rangle$ on the manifold ($L_g : \mathbb{R}^4 \rightarrow \mathcal{L}$) as follows:

$$L_g(\omega_x, \omega_y, \alpha, \beta) = \langle R\, exp\{J \begin{pmatrix} \omega_x \\ \omega_y \\ 0 \end{pmatrix}\}, (a + \alpha, b + \beta)\rangle$$

where exp is the matrix exponential operator, and $J(\omega)$ is the skew symmetric operator [11].

3.2.1 3D Line Classes

The local parameterization presented above can be applied to any general 3D line resulting in a line with 4 degrees of freedom. However, in reconstructing an urban scene predominantly composed of mutually orthogonal lines, not all 3D lines will have the full 4 degrees of freedom. For example, a vertical line in the 3D world has only 2 degrees of freedom (its intersection with the XY plane). Our 2D line detection algorithm presents us with this valuable information about the orientation of the corresponding 3D line which we can use to reduce its degrees of freedom, thus resulting in quicker convergence during optimization. We now present the different local parameterizations of the various line types (summarized in Table 1).

1. *Horizontal Lines*
A horizontal line has only 3 degrees of freedom - translation along $\hat{\mathbf{z}}$ (the z-axis), translation within the horizontal plane, and rotation about $\hat{\mathbf{z}}$. The local parameterization $L_{h3} : \mathbb{R}^3 \rightarrow \mathcal{L}$ for a horizontal line is defined as:

$$L_{h3}(\omega_z, \alpha, \beta) = \langle R\, exp\{J \begin{pmatrix} 0 \\ 0 \\ \omega_z \end{pmatrix}\}, (a + \alpha, b + \beta)\rangle$$

These lines can be initialized with an R corresponding to a pure rotation of $\frac{\pi}{2}$ about $\hat{\mathbf{y}}$ (the y-axis).

2. *Vertical Lines*
A vertical line has only 2 degrees of freedom, corresponding to its intersection with the XY plane. The local parameterization $L_v : \mathbb{R}^2 \rightarrow \mathcal{L}$ for a vertical line is defined as:

$$L_v(\alpha, \beta) = \langle R, (a + \alpha, b + \beta)\rangle$$

where $R = I$, the identity matrix.

3. *Horizontal Lines with fixed direction*
Often, it is the case that a horizontal line in our scene lies parallel to $\hat{\mathbf{x}}$ (the x-axis) or $\hat{\mathbf{y}}$. The direction of such a line is fixed, leaving only 2 degrees of freedom.

The local parameterization $L_{h2} : \mathbb{R}^2 \rightarrow \mathcal{L}$ for this line type is:

$$L_{h2}(\alpha, \beta) = \langle R, (a + \alpha, b + \beta) \rangle$$

Lines of this type can be initialized with an R corresponding to a rotation of $\frac{\pi}{2}$ either about $\hat{\mathbf{y}}$ (in the case of lines parallel to $\hat{\mathbf{x}}$) or $\hat{\mathbf{x}}$ (in the case of lines parallel to $\hat{\mathbf{y}}$).

3.2.2 Elimination of Gauge Freedoms

Every reconstruction task faces the problem of gauge freedoms – the fact that we can translate (3 DOF), rotate (3 DOF), or scale (1 DOF) the entire scene (cameras and 3D lines) as a single unit and still observe the same error. Failure to eliminate these 7 gauge freedoms can cause the optimization to get stuck along flat regions in the error surface. Our problem is not amenable to the traditional approach of fixing a camera at the origin because the world coordinate frame must be consistent with the horizontal and vertical lines in the scene. Instead, we can completely remove these degrees of freedom by placing additional constraints on the scene in the form of three completely fixed 3D lines L_1,L_2, and L_3. We do this in the following manner:

- L_1 - *Vertical line fixed as the Z-axis*
  We choose an arbitrary line that we know to be vertical, and fix it to be the Z-axis of the world. We initialize this line as $L_1 = \langle I, (0, 0) \rangle$. By doing so, we remove all degrees of freedom of the reconstruction that may alter the position and orientation of this line. In this case, we remove 4 degrees of freedom viz. rotation about $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$, and translation in the XY plane.

- L_2 - *Horizontal line fixed as the X axis*
  We choose any horizontal line coplanar with L_1 and fix it to be the X-axis of the world. We initialize this line as $L_2 = \langle R_0, (0, 0) \rangle$ where R_0 corresponds to a rotation about $\hat{\mathbf{y}}$ of $\frac{\pi}{2}$. By doing so, we reduce 2 more degrees of freedom of the reconstruction viz. translation along $\hat{\mathbf{z}}$, and rotation about $\hat{\mathbf{z}}$.

- L_3 - *Horizontal line parellel to L_2, intersecting L_1*
  We choose any horizontal line parallel to L_2, and coplanar with L_1 and L_2, and fix it to be a line parallel to $\hat{\mathbf{x}}$ and intersecting the Z-axis (now L_1) of the world at $(0, 0, 1)$. This removes the final remaining degree of freedom viz. scale. We initialize this line as $L_3 = \langle R_0, (1, 0) \rangle$ where R_0 is the same as that of L_2.

Note that while lines L_1,L_2, and L_3 remain fixed throughout the optimization (i.e. they have no degrees of freedom), the errors between the projections of these lines ℓ and their measurements f in the images still contribute to the objective function against which the optimization is performed.

4 Results

We tested our method on an urban scene containing multiple buildings with orthogonal structure. We captured 11 images of the scene, 10 of which were taken from ground level while walking around a city block, and one of which was taken from the fifth floor of one of the buildings in the scene. All images were taken with the same 3.2 megapixel digital camera.

Line detection was performed on each image at half resolution (1024x768 pixels). Despite the presence of trees and cars, the vanishing directions were successfully estimated in all images (see Figure 2), resulting in a large number of automatically detected and classified features (roughly 4000 per image). The resulting 2D lines were used as the basis of a manual feature correspondence step in which 469 of the features were chosen as measurements on 110 different 3D lines. Note that more than one feature in a single image can correspond to the same infinite 3D line (e.g. in a row of windows on a building, although the top edges of all the windows correspond to the same 3D line, the feature detector returns many short line segments corresponding to the edges of individual windows). While the large number of features generated for a given image suggests an almost impossibly large combinatorial problem for any RANSAC-based trifocal matching scheme, this many-to-one mapping (from features to 3D structure) simplifies the true combinatorics of the problem.

Of the 110 3D lines, 38 were vertical (depicted as blue lines in all figures) and 72 were horizontal, with 47 parallel to the x-axis (red in all figures), and 25 lines parallel to the y-axis (green in all figures). Note that we automatically extract this information about the *3D structure* of the scene before any reconstruction or optimization.

Initialization of the scene structure and camera parameters is an important step, and the method of [8], for example, would provide a suitable unconstrained estimate of structure as an initialization to our non-linear minimization method. In the present case, since we have classifications for all lines, we can initialize exactly half of the 3D line parameters in the scene according to Section 3, as well as the three lines we choose to fix our gauge freedoms. Camera rotation and intrinsic parameters are initialized based on the results of the feature detection stage, which computes these values in the process of classifying edge pixels. The reconstruction results presented here were initialized with manual estimates of camera translation and some scene structure.

Using the optimization method of Section 3, we were able to successfully reconstruct the 3D structure of the scene (see Figure 1 and Figure 5). Figure 5 shows synthesized views of the 3D structure and optimized camera poses. All images of the 3D reconstruction show the projections of the 2D measurements onto the reconstructed 3D lines.

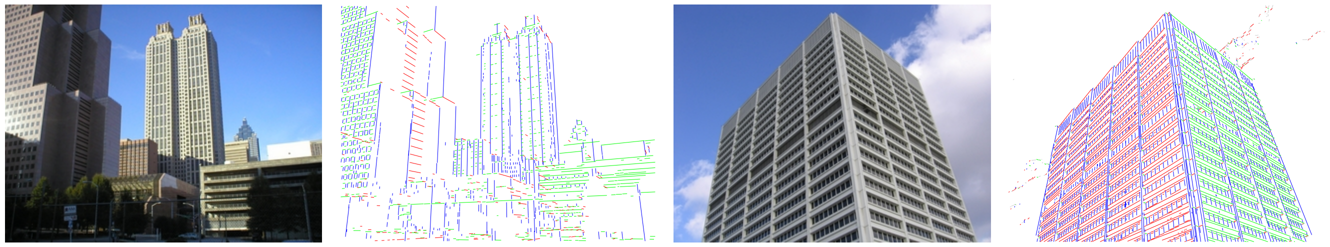


Figure 4: Line features are reliably detected and classified across a wide range of urban scenes.

5 Conclusion

We have demonstrated how knowledge of urban scene structure can be used effectively throughout the entire process of line-based structure from motion estimation, improving feature detection, feature matching, and optimization. We have defined a number of local parameterizations of different line types for use during non-linear optimization, and demonstrated the effectiveness of this approach through experimental results for 3D line-based reconstruction. In future work, we plan to fully automate the feature matching and initialization steps.

References

[1] C. Baillard, C. Schmid, A. Zisserman, and A. Fitzgibbon. Automatic line matching and 3D reconstruction of buildings from multiple views. In *ISPRS Conference on Automatic Extraction of GIS Objects from Digital Imagery, IAPRS Vol.32, Part 3-2W5*, pages 69–80, September 1999.

[2] A. Bartoli and P. Sturm. Structure-from-motion using lines: Representation, triangulation and bundle adjustment. *CVGIP:Image Understanding*, 100(3):416–441, 2005.

[3] P.A. Beardsley, P.H.S. Torr, and A. Zisserman. 3D model acquisition from extended image sequences. In *Eur. Conf. on Computer Vision (ECCV)*, pages II:683–695, 1996.

[4] J. Coughlan and A. Yuille. Manhattan World: Compass Direction from a Single Image by Bayesian Inference. In *Intl. Conf. on Computer Vision (ICCV)*, pages 941–947, 1999.

[5] A.R. Dick, P.H.S. Torr, and R. Cipolla. A Bayesian estimation of building shape using MCMC. In *Eur. Conf. on Computer Vision (ECCV)*, pages 852–866, 2002.

[6] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[7] P. Kahn, L. Kitchen, and E.M. Riseman. A fast line finder for vision-guided robot navigation. *IEEE Trans. Pattern Anal. Machine Intell.*, 12(11):1098–1102, Nov 1990.

[8] C. Rother. Linear multi-view reconstruction of points, lines, planes and cameras, using a reference plane. In *Intl. Conf. on Computer Vision (ICCV)*, pages 1210–1217, 2003.

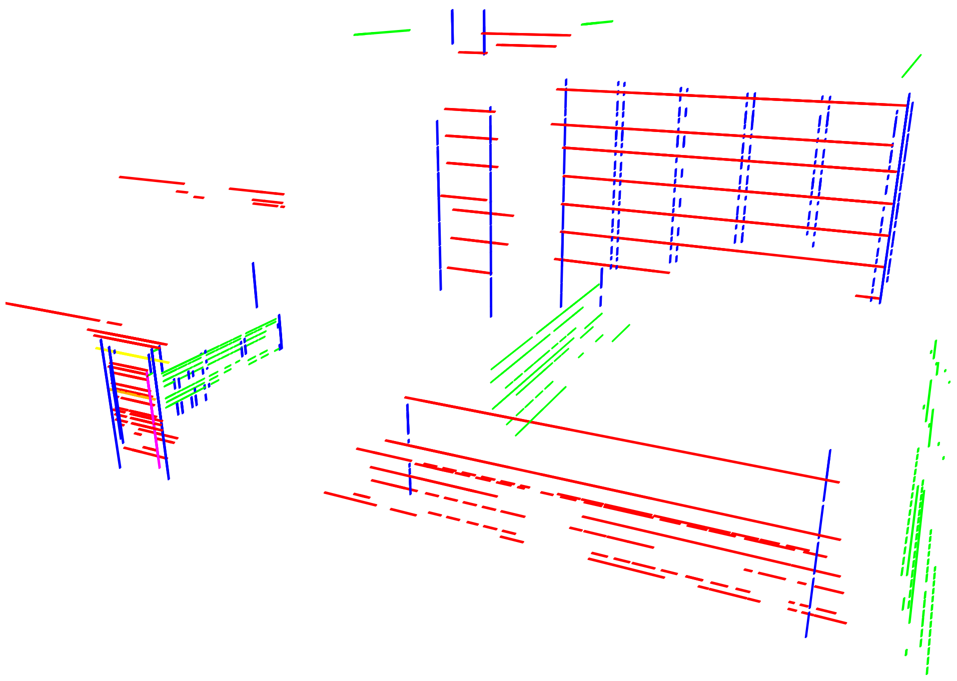
[9] G. Schindler and F. Dellaert. Atlanta World: An expectation-maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[10] C.J. Taylor, P.E. Debevec, and J. Malik. Reconstructing polyhedral models of architectural scenes from photographs. In *Eur. Conf. on Computer Vision (ECCV)*, pages 659–668, 1996.

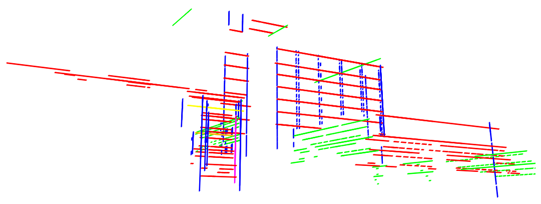
[11] C.J. Taylor and D.J. Kriegman. Minimization on the lie group SO(3) and related manifolds. Technical Report 9405, Yale University, New Haven, CT, April 1994.

[12] C.J. Taylor and D.J. Kriegman. Structure and motion from line segments in multiple images. *IEEE Trans. Pattern Anal. Machine Intell.*, 17(11):1021–1032, November 1995.

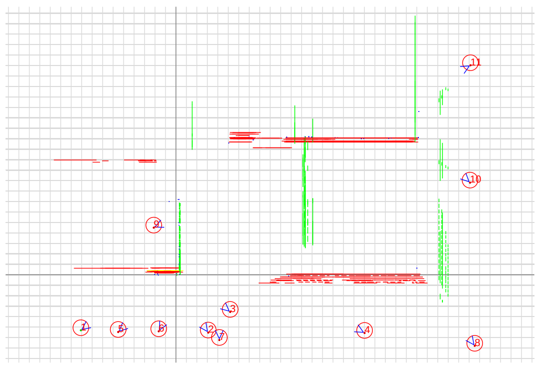
[13] T. Werner and A. Zisserman. New techniques for automated architecture reconstruction from photographs. In *Eur. Conf. on Computer Vision (ECCV)*. Springer-Verlag, 2002.



(a)



(b)



(c)

Figure 5: 3D line structure was successfully reconstructed based on 439 measurements of 110 3D lines across 11 images. Figures (a) and (b) show synthesized views from novel viewpoints. Red, green, and blue lines correspond to lines along the X, Y, and Z axes, respectively. The single yellow, orange, and pink lines in each image indicate those lines fixed to eliminate gauge freedoms. Figure (c) shows the reconstructed lines from directly overhead, as well as the position and orientation of all 11 cameras. To provide scale, grid lines in the bottom image are spaced at intervals of 5 meters.

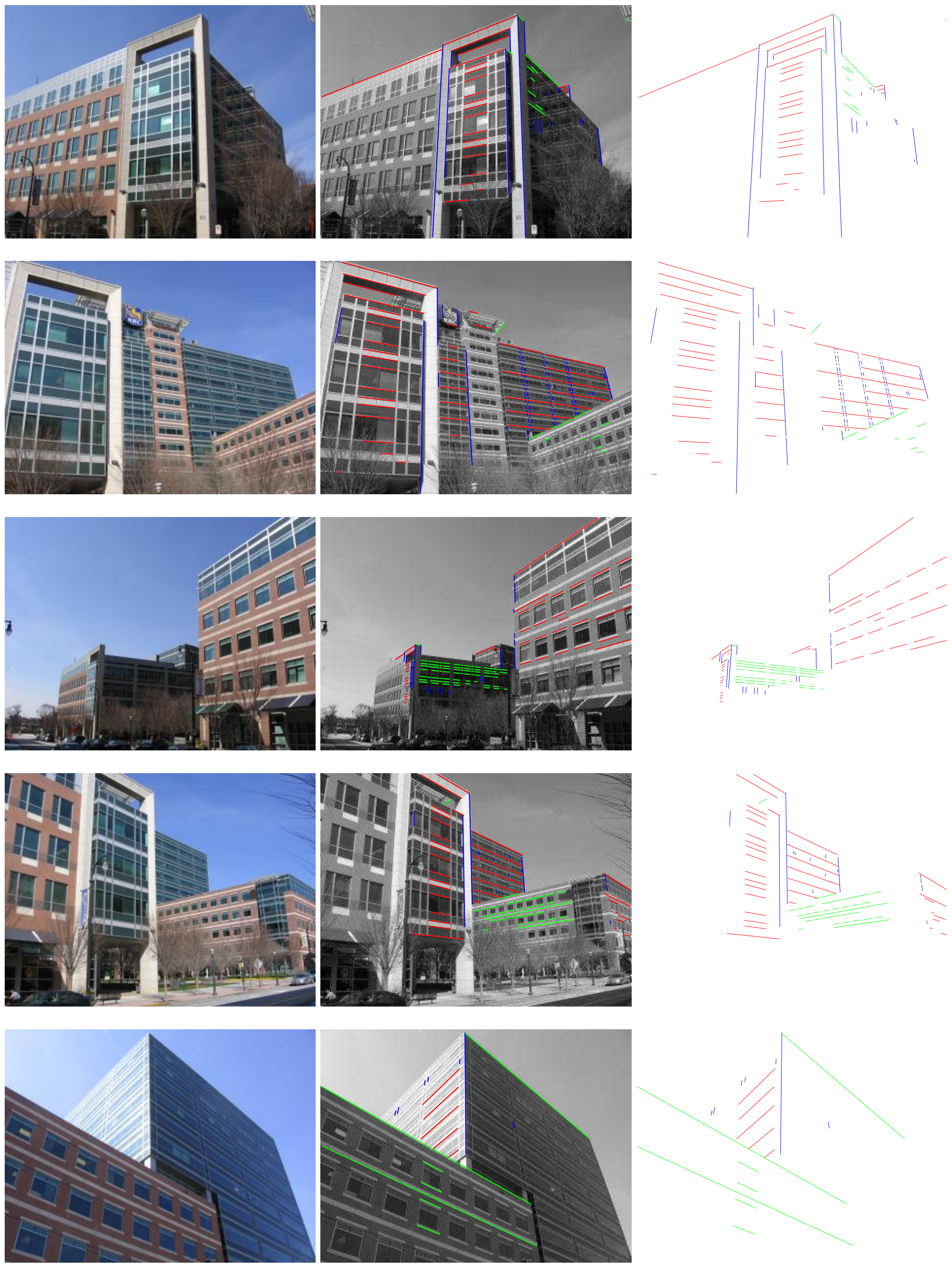


Figure 6: We captured 11 images of an urban scene, 5 of which are displayed above in the left-hand column. The middle column shows the subset of automatically detected features from each image which were used as measurements during reconstruction. The right column displays the projections of the reconstructed 3D lines into the recovered camera associated with each image.