Title

Planar Structures from Line Correspondences in a Manhattan World

Authors

Kim, Chelhwon

Manduchi, Roberto

Publication Date

2014-10-02

Peer reviewed

Planar Structures from Line Correspondences in a Manhattan World

Chelhwon Kim[1], Roberto Manduchi[2]

[1] Electrical Engineering Department
[2] Computer Engineering Department
University of California, Santa Cruz
Santa Cruz, CA, US

Abstract. Traditional structure from motion is hard in indoor environments with only a few detectable point features. These environments, however, have other useful characteristics: they often contain severable visible lines, and their layout typically conforms to a Manhattan world geometry. We introduce a new algorithm to cluster visible lines in a Manhattan world, seen from two different viewpoints, into coplanar bundles. This algorithm is based on the notion of "characteristic line", which is an invariant of a set of parallel coplanar lines. Finding coplanar sets of lines becomes a problem of clustering characteristic lines, which can be accomplished using a modified mean shift procedure. The algorithm is computationally light and produces good results in real world situations.

1 Introduction

This paper addresses the problem of reconstructing the scene geometry from pictures taken from different viewpoints. Structure from motion (SFM) has a long history in computer vision [1, 2], and SFM (or visual SLAM) algorithms have been ported on mobile phones [3, 4]. Traditional SFM relies on the ability of detecting and matching across views a substantial number of point features. Unfortunately, robust point detection and matching in indoor environments can be challenging, as the density of detectable points (e.g. corners) may be low. At the same time, indoor environments are typically characterized by (1) the presence of multiple line segments (due to plane intersections and other linear structures), and (2)"Manhattan world" layouts, with a relatively small number of planes at mutually orthogonal orientations.

This paper introduces a new algorithm for the detection and localization of planar structures and relative camera pose in a Manhattan world, using line matches from two images taken from different viewpoints. As in previous approaches [5–7], the orientation (but not the position) of the two cameras with respect to the environment is computed using vanishing lines and inertial sensors (available in all new smartphones). The main novelty of our algorithm is in the criterion used to check whether groups of lines matched in the two images may be coplanar. Specifically, we introduce a new invariant feature ($\vec{n}$-*characteristic line*) of the image of a bundle of coplanar parallel lines, and show how this

feature can be used to cluster visible lines into planar patches and to compute the relative camera pose. The algorithm has low complexity (quadratic in the number of matched line segments); implemented on an iPhone 5s, its average end-to-end execution time is of 0.28 seconds. Our algorithm fully exploits the strong constraints imposed by the Manhattan world hypothesis, and is able to produce good results even when very few lines are visible, as long as they are correctly matched across the two images.

2 Related Work

The standard approach to recovering scene structure and camera pose from multiple views is based on point feature matches across views [2]. When point features are scarce, line features can be used instead. Computation of 3-D line segments and camera pose from three images of a set of lines is possible using the trifocal tensor [2, 8, 9]. This approach follows three general steps: (1) trifocal tensor computation from triplets of line correspondences, producing the three camera matrices; (2) 3-D line computation via triangulation from line correspondences; (3) non-linear optimization for refinement. At least 13 triplets of line correspondences are necessary for computing the trifocal tensor [2]. Note that direct 3-D line computation requires at least three views because two views of 3-D lines in the scene do not impose enough constraints on camera displacements [9, 10].

A few authors have attempted to recover structure and motion using line features from only two views (as in our contribution), under strong geometric priors such as the Manhattan world assumption. Košecka and Zhang [11] presented a method to extract dominant rectangular structures via line segments that are aligned to one of the principal vanishing points, thus recovering camera pose and planar surfaces. Elqursh and Elgammal [7] introduced an SFM algorithm based on line features from a man-made environment. Three line segments, two of which parallel to each other and orthogonal to the third one, are used to recover the relative camera rotation, and the camera translation is computed from any two intersections of two pairs of lines. This algorithm was shown to work even in the absence of dominant structures.

An alternative approach is to detect dominant planes and compute the induced homographies, from which the camera pose and planar geometry can be recovered [12–15]. Zhou et al. [16] presented a SFM system to compute structure and motion from one or more large planes in the scene. The system detects and tracks the scene plane using generalized RANSAC, and estimates the homographies induced by the scene plane across multiple views. The set of homographies are used to self-calibrate and recover the motion for all camera frames by solving a global optimization problem. Another possibility for planar surface recovery is to fit multiple instances of a plane to 3-D point cloud obtained by SFM using a robust estimation algorithm [17–19].

A more recent research direction looks to recover the spatial layout of an indoor scene from a single image [20–22]. Lee et al. [23] proposed a method

based on an hypothesis-and-test framework. Layout hypotheses are generated by connecting line segments using geometric reasoning on the indoor environment, and verified to find the best fit to a map that expresses the local belief of region orientations computed from the line segments. Flint et al. [24] addressed the spatial layout estimation problem by integrating information from image features, stereo features, and 3-D point clouds in a MAP optimization problem, which is solved using dynamic programming. Ramalingam et al. [25] presented a method to detect junctions formed by line segments in three Manhattan orthogonal directions using a voting scheme. Possible cuboid layouts generated from the junctions are evaluated using an inference algorithm based on a conditional random field model. Tsai et al. [26] model an indoor environment as a ground plane and a set of wall planes; by analyzing ground-wall boundaries, a set of hypotheses of the local environment is generated. A Bayesian filtering framework is used to evaluate the hypotheses using information accumulated through motion.

3 The Characteristic Lines Method

3.1 Notation and Basic Concepts

By *Manhattan world* we mean an environment composed of planar surfaces, each of which is oriented along one of three *canonical* mutually orthogonal vectors[3] ($\vec{n}_1$, $\vec{n}_2$, $\vec{n}_3$). In addition, we will assume that each line visible in the scene lies on a planar surface (possible at its edge) and is oriented along one of the three canonical vectors. Two pictures of the environment are taken by two different viewpoints (camera centers, $\vec{c}_1$ and $\vec{c}_2$) with *baseline* $\vec{t} = \vec{c}_1 - \vec{c}_2$. The rotation matrix representing the orientation of the frame of reference of the first camera with respect to the second one is denoted by $\mathbf{R}$. Previous work has shown how to reconstruct the orientation of a camera from a single picture of a Manhattan world, using the location of the three vanishing points of the visible lines [5]. This estimation can be made more robust by measuring the gravity vector using a 3-axis accelerometer, a sensor that is present in any modern smartphones [6]. We will assume that the intrinsic calibration matrices $\mathbf{K}_1$, $\mathbf{K}_2$ of the cameras have been obtained offline, and that the orientation of each cameras with respect to the canonical reference system ($\vec{n}_1$,$\vec{n}_2$,$\vec{n}_3$) has been estimated using one of the methods mentioned above (and, consequently, that $\mathbf{R}$ is known). We will also assume that lines visible in both images have been correctly matched; the algorithm used in our implementation for line detection and matching is briefly discussed in Sec. 4.

A generic plane Π will be identified by the pair $(\vec{n}, d)$, where $\vec{n}$ is its orientation (unit-norm normal) and d is its signed offset with respect to the first camera ($d = \langle \vec{p} - \vec{c}_1, \vec{n} \rangle$, where $\vec{p}$ is a generic point on the plane, and $\langle \cdot, \cdot \rangle$ indicates inner product). A generic line $\mathcal{L}$ will be identified by its orientation $\vec{l}$

[3] A vector is represented by an arrowed symbol ($\vec{n}$) when the frame of reference is immaterial, and by a boldface symbol ($\mathbf{n}$) when expressed in terms of a frame of reference.

(unit-norm vector parallel to the line) and by the location of any point on the line. In a Manhattan world, surface planes and visible lines are oriented along one of the three canonical orientations.

It is well known that a plane $(\vec{n}, d)$ imaged by two cameras induces an homography $\mathbf{H}$ on the image points in the two cameras. Given a line $\mathcal{L}$ in the plane, the two homogeneous representations $\mathbf{L}_1$ and $\mathbf{L}_2$ of the image lines in the two camera are related to one another as by $\mathbf{L}_1 = \mathbf{H}^T\mathbf{L}_2$. The *lever vectors* $\vec{u}_1(\mathcal{L})$ and $\vec{u}_2(\mathcal{L})$ are unit-norm vectors orthogonal to the plane containing $\mathcal{L}$ and the optical center of camera 1 and camera 2, respectively (see Fig. 1, left panel). Expressed in terms of the associated camera reference frames, the lever vectors can be written as $\mathbf{u}_1 = \mathbf{K}_1^T\mathbf{L}_1$ and $\mathbf{u}_2 = \mathbf{K}_2^T\mathbf{L}_2$. The lever vectors are thus easily computed from the image of the line $\mathcal{L}$ in the two cameras. The following relation holds:

$$\mathbf{u}_1 = \mathbf{H}_c^T\mathbf{u}_2 \tag{1}$$

where $\mathbf{H}_c = \mathbf{K}_2^{-1}\mathbf{H}\mathbf{K}_1$ is the *calibrated homography matrix* induced by the plane, which can be decomposed [2] as

$$\mathbf{H}_c = \mathbf{R} + \mathbf{t}\mathbf{n}^T/d \tag{2}$$

In the above equation, the baseline $\mathbf{t}$ and plane normal $\mathbf{n}$ are expressed in terms of the reference frames defined at the second camera and at the first camera, respectively, and d is the distance between the plane and the first camera.

A set of lines will be termed $\vec{n}$-*coplanar* if the lines are all coplanar, and the common plane has orientation $\vec{n}$.

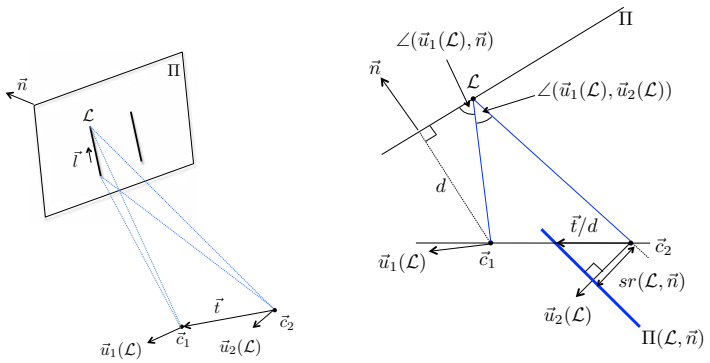

Fig. 1. Left: The two camera centers $\vec{c}_1$, $\vec{c}_2$ and the lever vectors $\vec{u}_1(\mathcal{L})$, $\vec{u}_2(\mathcal{L})$ for line $\mathcal{L}$. Right: Line $\mathcal{L}$ lies on the plane $\mathit{\Pi} \equiv (\vec{n}, d)$ (both line and plane orthogonal to this page). The thick blue line is the trace of the $\vec{n}$-characteristic plane $\mathit{\Pi}(\mathcal{L}, \vec{n})$ (also orthogonal to the page).

3.2 Characteristic Planes

Given a line $\mathcal{L}$ and a vector $\vec{n}$, we define by *sin-ratio* $sr(\mathcal{L}, \vec{n})$ the following quantity:

$$sr(\mathcal{L}, \vec{n}) = \frac{\sin \angle(\vec{u}_1(\mathcal{L}), \vec{u}_2(\mathcal{L}))}{\sin \angle(\vec{u}_1(\mathcal{L}), \vec{n})} \tag{3}$$

where $\angle(\cdot, \cdot)$ indicates the signed angle between two vectors. Note that the numerator of (3) has magnitude equal to $\|(\mathbf{u}_1 \times \mathbf{R}^T \mathbf{u}_2)\|$, while the denominator has magnitude equal to $\|\mathbf{u}_1 \times \mathbf{n}\|$, where $\mathbf{n}$ is defined with respect to the first camera's reference frame. Hence, $sr(\mathcal{L}, \vec{n})$ can be computed from the two images of $\mathcal{L}$ and from $\mathbf{R}$ without knowledge of the baseline $\vec{t}$. The sin-ratio has an interesting property:

Proposition 1. If the line $\mathcal{L}$ lies on plane $(\vec{n}, d)$, then the projection $\langle \vec{t}/d, \vec{u}_2(\mathcal{L}) \rangle$ of $\vec{t}/d$ onto $\vec{u}_2(\mathcal{L})$ is equal to $sr(\mathcal{L}, \vec{n})$.

Proof. From (1) one derives

$$\mathbf{u}_1 \times \mathbf{H}_c^T \mathbf{u}_2 = 0 \tag{4}$$

Combining (4) with (2), one obtains

$$(\mathbf{R}^T \mathbf{u}_2) \times \mathbf{u}_1 = \mathbf{u}_1 \times \mathbf{n}\mathbf{u}_2^T \mathbf{t}/d \tag{5}$$

The vectors $(\mathbf{R}^T \mathbf{u}_2) \times \mathbf{u}_1$ and $\mathbf{u}_1 \times \mathbf{n}$ are both parallel to the line $\mathcal{L}$. The ratio of their magnitudes (multiplied by -1 if they have opposite orientation) is equal to the sin-ratio $sr(\mathcal{L}, \vec{n})$. This value is also equal to $\mathbf{u}_2^T \mathbf{t}/d = \langle \vec{u}_2(\mathcal{L}), \vec{t}/d \rangle$ ■

This result may be restated as follows. Given a plane $(\vec{n}, d)$ and a line $\mathcal{L}$ on this plane, define by *characteristic plane* $\Pi(\mathcal{L}, \vec{n})$ the plane with normal equal to $\vec{u}_2(\mathcal{L})$ and offset with respect to the second camera center $\vec{c}_2$ equal to $sr(\mathcal{L}, \vec{n})$. Then, the "normalized" baseline vector $\vec{t}/d$ is guaranteed to lie on $\Pi(\mathcal{L}, \vec{n})$ (see Fig. 1, right panel). This constraint is at the basis of our *characteristic line* method, discussed in the next section. A parallel derivation of the characteristic plane and of its properties, based on algebraic manipulation, is presented in the Appendix.

3.3 Characteristic Lines and Coplanarity

Given a set of parallel lines $\{\mathcal{L}_i\}$, with common orientation $\vec{l}$, the associated characteristic planes $\{\Pi(\mathcal{L}_i, \vec{n})\}$ for a given unit norm vector $\vec{n}$ are all parallel to $\vec{l}$ by construction (since the lever vectors $\{\vec{u}_2(\mathcal{L}_i)\}$ are all coplanar and orthogonal to $\vec{l}$). Any two such planes intersect at a *$\vec{n}$-characteristic line* $\mathcal{L}^*$ oriented along $\vec{l}$. It may be interesting to study under which conditions *all* of the characteristic planes associated with $\{\mathcal{L}_i\}$ intersect at a common line, i.e., when the lines $\{\mathcal{L}_i\}$ *induce a $\vec{n}$-characteristic plane intersection at $\mathcal{L}^*$*.

Corollary 1. Let $\{\mathcal{L}_i\}$ be any number of parallel $\vec{n}$-coplanar lines. These lines induce a $\vec{n}$-characteristic plane intersection at $\mathcal{L}^*$, where the characteristic line $\mathcal{L}^*$ goes through $\vec{t}/d$, and d is the signed offset of the plane defined by the lines $\{\mathcal{L}_i\}$ to the first camera.

Proof. By definition of $\vec{n}$–coplanarity, all lines $\{\mathcal{L}_i\}$ lie on the plane $(\vec{n}, d)$. Hence, by Proposition 1, $\vec{t}/d$ is contained in all of the $\vec{n}$-characteristic planes defined by the lines. Since these planes are all parallel to the orientation of the lines $\{\mathcal{L}_i\}$, they must intersect at a single characteristic line containing $\vec{t}/d$■

Corollary 1 shows that a sufficient condition for a set of parallel lines to induce a $\vec{n}$-characteristic plane intersection is that they be $\vec{n}$-coplanar. The resulting characteristic line represents an *invariant* property of parallel, coplanar lines; importantly, it can be computed from the image lines, provided that the rotation $\mathbf{R}$ and the normal $\vec{n}$ of the plane are known. As discussed earlier in Sec. 3.1, this information can be easily obtained in a Manhattan world.

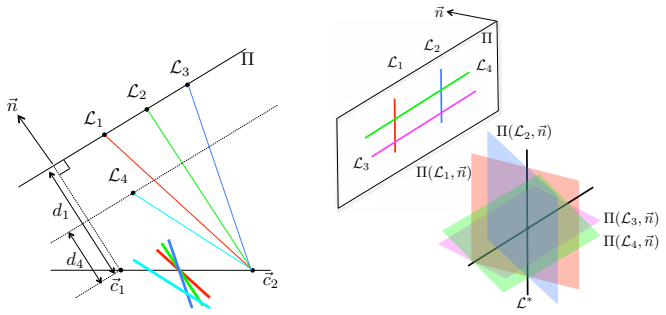

Fig. 2. Left: Lines $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$ (orthogonal to this page) are $\vec{n}$-coplanar. Their associated $\vec{n}$-characteristic planes all intersect at a characteristic line through the baseline (also orthogonal to this page). They also individually intersect with the $\vec{n}$-characteristic plane associated with line $\mathcal{L}_4$, parallel but not coplanar with the other lines, but these intersections are outside of the baseline. Right: The sets of parallel lines ($\mathcal{L}_1$, $\mathcal{L}_2$) and ($\mathcal{L}_3$, $\mathcal{L}_4$) are mutually orthogonal; all lines are $\vec{n}$-coplanar. The $\vec{n}$-characteristic line associated with ($\mathcal{L}_3$, $\mathcal{L}_4$) intersects the $\vec{n}$-characteristic line associated with ($\mathcal{L}_1$, $\mathcal{L}_2$), $\mathcal{L}^*$, at a point on the baseline.

Thus, for a given canonical orientation $\vec{n}$, one may test whether a group of parallel lines all belong to a plane oriented as $\vec{n}$ by observing whether the associated characteristic planes intersect at one line (see Fig. 2, left panel). In fact, one never needs to test many lines at once: the characteristic planes for multiple lines in a parallel bundle intersect at a single line if and only if the characteristic lines from pairwise plane intersection are identical. Hence, one needs only test two parallel lines at a time. This observation suggests the following algorithm to *cluster* parallel lines into coplanar groups for a given plane orientation $\vec{n}$:

1. For each pair of parallel lines, find the associated $\vec{n}$-characteristic line;

2. Find clusters of nearby characteristic lines. Each such cluster may signify the presence of a plane;
3. For all characteristic lines in a cluster, label the associated parallel lines $\{\mathcal{L}_i\}$ as belonging to the same plane $(\vec{n},d)$ for some d.

An example of application of this algorithm is shown in Fig. 3. Since all $\vec{n}$-characteristic lines for a given line orientation $\vec{l}$ are parallel to $\vec{l}$, in Fig. 3 we simply plot the intersections of these lines with a plane orthogonal to $\vec{l}$. To identify cluster centers, we run mean shift on these 2-D points.

A degenerate case occurs when the camera moves in the direction of $\vec{l}$. In this case, $\vec{u}_1 = \vec{u}_2$ and thus $sr(\mathcal{L}_i, \vec{n}) = 0$ for all lines $\mathcal{L}_i$, meaning that all $\vec{n}$-characteristic planes intersect at $\vec{c}_2$. This is consistent with the fact that the image of the lines does not change as the camera moves along $\vec{l}$.

Before closing this section, we note that from a set of coplanar parallel lines we cannot really say much about the baseline $\vec{t}$. As we will see next, multiple bundles of parallel lines allow us to also precisely compute $\vec{t}/d$.

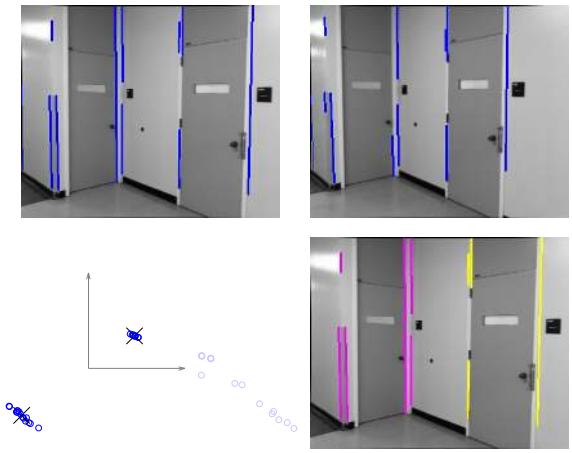

Fig. 3. Top row: Image pair with detected lines oriented along one canonical direction ($\vec{n}_1$). Only lines that have been matched across images are shown. Bottom left: Traces of the $\vec{n}_2$- and $\vec{n}_3$-characteristic lines on a plane oriented as $\vec{n}_1$. The cluster centers, found by mean shift, are marked by a cross. Note that the cluster centers for the $\vec{n}_2$- and $\vec{n}_3$-characteristic lines are found separately. Characteristic line traces are shown by circles in dark blue color when associated with a cluster, by circles in pale blue color otherwise. Bottom right: The coplanar line sets defined by the characteristic line clusters. Each set drawn with a characteristic color (For line segments at the intersection of two planes, we used one color corresponding to one of the two planes).

3.4 Multiple Line Orientations

In a Manhattan world, lines belonging to a plane orthogonal to a canonical orientation $\vec{n}_i$ must be oriented along one of the two other canonical orientations ($\vec{n}_j$ or $\vec{n}_k$), orthogonal to $\vec{n}_i$. For each such orientation, consider a bundle of coplanar parallel lines. These two line bundles induce an $\vec{n}_j$- and $\vec{n}_k$-characteristic line, respectively, where the first characteristic line contains $\vec{t}/d_j$ and the second one contains $\vec{t}/d_k$ (d_j and d_k being the offsets of the planes defined by the two line bundles to the first camera). If both bundles are coplanar, then obviously $d_j = d_k = d$, and the characteristic lines in both directions intersect at $\vec{t}/d$ (Fig. 2, right panel). This is a very interesting result: the intersection of characteristic lines induced by orthogonal coplanar lines directly provides the direction of camera translation. (Note again that this simple result is only possible in a Manhattan world, where the orientation of planes in the scene is known.)

The algorithm for testing coplanarity introduced in Sec. 3.3 can be easily modified to consider, for each canonical orientation $\vec{n}$, the two bundles of parallel lines in the two directions orthogonal to $\vec{n}$. This calls for an algorithm that can detect accumulation points of 3-D lines, defined as points in 3-D space that, within a cubic neighborhood, contain a high density of characteristic lines in both directions. For this purpose we propose a modified version of mean shift [27], described next.

3.5 A Modified Mean Shift Algorithm

Suppose we are looking for groups of $\vec{n}_1$-coplanar lines; each one of these lines is oriented along either $\vec{n}_2$ or $\vec{n}_3$. Given a cubic neighborhood around a point $\vec{p}$, it is convenient to consider the *traces* (intersections) of the lines on the cube's faces orthogonal to $\vec{n}_2$ and $\vec{n}_3$. Suppose to move the point $\vec{p}$ (and the cube around it) along $\vec{n}_2$; it is clear that only the density (within the cube) of lines oriented along the orthogonal direction $\vec{n}_3$ will change. Likewise, moving the point along $\vec{n}_3$ will change only the density of lines parallel to $\vec{n}_2$. If, however, the point is moved along $\vec{n}_1$, the density of both lines in the cube will change.

Let (p^1, p^2, p^3) be the coordinates of the point $\vec{p}$ in a canonically oriented reference system; let $(\mathcal{L}^1_{i,2}, \mathcal{L}^3_{i,2})$ be the coordinates of the trace on the $(\vec{n}_1, \vec{n}_3)$ plane of a generic $\vec{n}_2$-oriented line $\mathcal{L}_i$ crossing the cubic neighborhood of $\vec{p}$; and let $(\mathcal{L}^1_{j,3}, \mathcal{L}^2_{j,3})$ be the coordinates of the trace on the $(\vec{n}_1, \vec{n}_2)$ plane of a generic $\vec{n}_3$-oriented line $\mathcal{L}_j$ crossing the cube. Our algorithm iterates over a cycle of 3 steps, each requiring a 1-D (component-wise) mean shift update:

1. Implement a mean shift update of p^2 based on the measurements $\{\mathcal{L}^2_{j,3}\}$.
2. Implement a mean shift update of p^3 based on the measurements $\{\mathcal{L}^3_{i,2}\}$.
3. Implement a mean shift update of p^1 based on the measurements $\{\mathcal{L}^1_{i,2}\} \cup \{\mathcal{L}^1_{j,3}\}$.

At convergence, the point will be situated in a neighborhood with high density of lines in both directions. We also found it beneficial to assign a weight to each

line (which is used in the mean shift updates) equal to the mean of a function $g(D)$ (with $g(D) = e^{-D/\sigma}$) of the line's distance D to each other line oriented in an orthogonal direction; this ensures that characteristic lines with a high density of neighbors in the orthogonal direction are given high weight. An example of application of this algorithm is shown in Fig. 4.

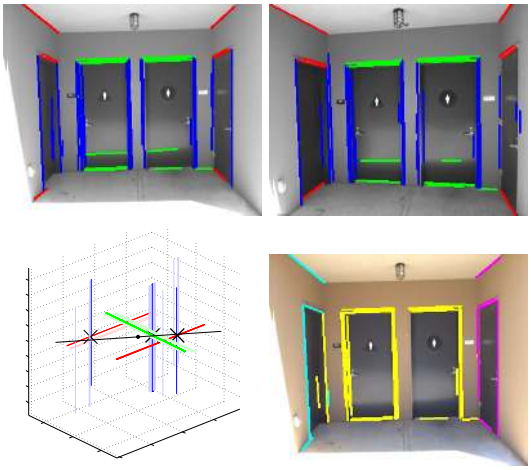

Fig. 4. Top row: Image pair with detected lines oriented along the three canonical directions (the color of each line identifies its orientation). Only lines that have been matched across images (including incorrect ones) are shown. Bottom left: Characteristic lines for the different orientations. The color of a characteristic line matches the color of the lines it represents. Clusters centers identified by the mean shift algorithm described in Sec. 3.5 are shown by black crosses. Characteristic lines not associated to a cluster are shown in pale color. The regressed baseline direction is represented by a black line through the origin (shown as a thick dot). Bottom right: The coplanar line sets defined by the characteristic line clusters (each set drawn with a characteristic color).

3.6 Limitations

Corollary 1 provides a sufficient condition for characteristic plane intersection. This condition, however, is not necessary: there may exist groups of parallel, non-$\vec{n}$-coplanar lines (but still individually oriented orthogonally to $\vec{n}$) that induce a $\vec{n}$-characteristic plane intersection. This means that a cluster of characteristic lines could potentially be found even for non-coplanar lines.

In general, the occurrence of such "spurious" clusters is unlikely in a Manhattan world. For example, if two parallel lines are $\vec{n}$-coplanar, addition of a third, non-coplanar parallel line will *not* induce a characteristic plane intersection, as shown by the following corollary (proof omitted for lack of space.)

Corollary 2: A bundle of parallel lines, two or more of which are $\vec{n}$-coplanar, induces a $\vec{n}$-characteristic plane intersection only if all lines in the bundle are $\vec{n}$-coplanar.

4 Implementation

We use the LSD (Line Segment Detector) algorithm [28] to find line segments. This algorithm works in linear time, and does not require any parameter tuning. A MSLD (Mean-Standard Deviation Line Descriptor) [29] feature vector is computed for each line; lines are matched based on a criterion that considers the Euclidean distance between feature vectors in a line match while ensuring that the angle between matched image lines in the two images is consistent across matches. For each image, the vanishing points of detected lines are computed. This information, together with data from the accelerometers (which measure the direction of gravity, assumed to be aligned to one of the canonical orientations), is used to compute the rotation of each camera with respect to the frame of reference defined by the canonical orientations.

Each image line segment is associated with one canonical direction. In addition, each line segment is rotated around its midpoint and aligned with the direction from the midpoint to the associated vanishing point. This pre-processing is particularly useful for short segments, whose estimated orientation can be noisy.

In addition to vanishing points, we compute the vanishing lines of planes in the canonical orientations. (In a Manhattan world, vanishing lines join the two vanishing points of visible lines.) Suppose that the vanishing line for planes orthogonal to $\vec{n}$ is visible in the image; since the image of a plane orthogonal to $\vec{n}$ cannot straddle the plane's vanishing line, when computing the $\vec{n}$-characteristic lines we can safely neglect to consider pairs of parallel lines whose images are on different sides of the vanishing line. This property, which is used extensively in the computation of structure from single images [23], helps reducing the risk of false positives.

We also implemented a simple procedure to remove characteristic lines from parallel line pairs that are unlikely to belong to the same planar surface. Given a pair of image segments from parallel lines (i.e. converging at one of the vanishing points), we compute the smallest quadrilateral $\mathcal{Q}$, two sides of which are collinear with the line segments, and the remaining sides converge to one of the other vanishing points, such that all four segments endpoints are contained in the quadrilateral. This quadrilateral could be construed as the image of a rectangular planar surface, with edges parallel to the canonical directions. If this were in fact the case (i.e., if there existed a planar rectangular surface patch projecting onto the image quadrilateral), one would not expect to see a line orthogonal to the surface within the image of the surface. Accordingly, if a line aligned towards the third vanishing point crosses $\mathcal{Q}$, the two line segments defining $\mathcal{Q}$ are assumed *not* to belong to the same planar patch, and the associated characteristic line is neglected. This simple procedure has given good results in our experiments, although it may lead to false negatives in more complex geometrical layouts.

As detailed in the previous sections, our algorithm searches for coplanar lines one canonical orientation at a time. All pairs of parallel lines that survive the tests discussed above generate characteristic lines, and 3-D clusters are found using our modified mean shift algorithm, seeded with multiple points chosen at the mid-point between nearby orthogonal characteristic line pairs. Each cluster represents a plane; the characteristic lines within each cluster identify coplanar 3-D lines. In some cases, the same line in space may be associated with two different characteristic lines, belonging to different clusters. This may be a legitimate situation for lines at the junction of two planes; otherwise, it represents an inconsistency. In order to reject outlier clusters, we exploit the property that clusters of characteristic lines defined by $\vec{n}$-coplanar lines must be collinear with the baseline vector $\vec{t}$ (Sec. 3.3). For each canonical orientation $\vec{n}_i$, we select the cluster of characteristic lines orthogonal to $\vec{n}_i$ with highest weight, where the weight of a cluster is equal to the sum of the weights of the characteristic lines it contains (with the characteristic line weights defined in Sec. 3.5). The selected cluster determines a tentative baseline direction $\vec{t}_i$. Among the remaining clusters, we only retain those that are at a distance to the line $\lambda\vec{t}_i$ closer than a threshold T. We repeat this for all canonical directions, obtaining up to three tentative baseline directions $\{\vec{t}_i\}$. Note that some canonical orientation may contain no characteristic lines, or the lines may not cluster. (In fact, in our experiments we never considered the vertical canonical orientation due to the general lack of line features on the floor and on the ceiling.) Finally, we linearly regress the direction of $\vec{t}$ from the vectors $\{\vec{t}_i\}$, and project the vectors $\{\vec{t}_i\}$ onto the resulting line to compute (up to a common scale) the distance of each plane to the first camera (and thus the location of the planes in space).

Our algorithm has been implemented on an iPhone 5s and tested in various scenarios. On images with resolution of 352×288, execution time is of 0.28 seconds on average, with 35% of the computation due to line detection, 6% to vanishing line detection, 7% to line matching, and the remaining 52% due to characteristic lines computation and clustering.

5 Experimental Evaluation

Quantitative comparative assessment of our algorithm was performed on a set of 49 image pairs. These image pairs were taken by hand, some with an iPhone 4 and some with an iPhone 5s. Examples can be seen in Fig. 5. The full set of images, with line detection and 3-D reconstruction, is provided in the Supplementary Material.

We devised an evaluation criterion based on a test for coplanarity of line triplets, that does not require ground truth measurements of relative camera pose (which are difficult to obtain without precisely calibrated instruments). This criterion requires manual evaluation of coplanarity of all line triplets seen in the image. In practice, we manually enumerated all planes in the scene and assigned each line to the one or two planes containing it. From this data, labeling of all line triplets as coplanar or not is trivial. Given three lines in space, one

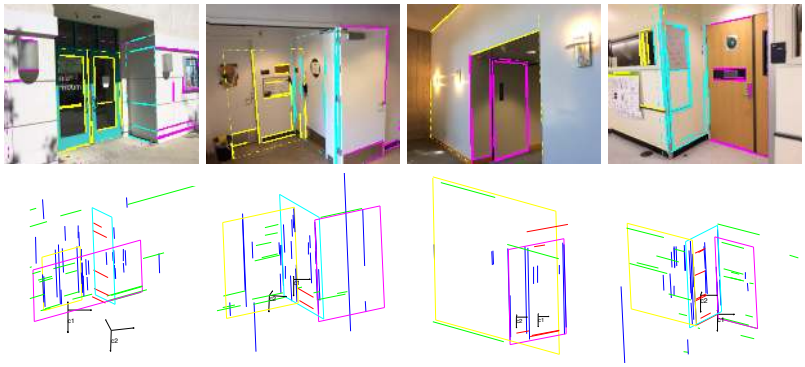

Fig. 5. Top row: Coplanar line sets produced by our algorithm for the image set considered in the evaluation. Only one image for each pair is shown. Different line sets are shown in different color. Note that some lines (especially those at a planar junction) may belong to more than one cluster (although they are displayed using only one color). All lines that have been matched (possibly incorrectly) across images are shown (by thick segments) and used for coplanarity estimation. The quadrilaterals $\mathcal{Q}$ shown by dotted lines represent potential planar patches. They contain all coplanar lines in a cluster, and are computed as described in Sec. 4. Bottom row: 3-D reconstruction of the visible line segments and camera center positions. Line segment are colored according to their orientation in space. The colored rectangles are the reconstructed planar patches corresponding to the quadrilateral $\mathcal{Q}$ shown with the same color as in the top row.

can test for their coplanarity using Plücker matrices [30]. More precisely, lines $(\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3)$ are coplanar if $\mathbf{L}_1\mathbf{L}_2^*\mathbf{L}_3 = \mathbf{0}$, where $\mathbf{L}_1, \mathbf{L}_3$ are the Plücker L-matrices associated with $\mathcal{L}_1, \mathcal{L}_3$ and $\mathbf{L}_2^*$ is the Plücker L^*-matrix associated with $\mathcal{L}_2$ [30]. The ability of an algorithm to determine line coplanarity is critical for precise reconstruction of Manhattan environments; in addition, this criterion gives us an indirect assessment of the quality of pose estimation (as we expect that good pose estimation should result in good 3-D reconstruction and thus correct coplanarity assessment).

We compared our algorithm against two other techniques. The first is traditional structure from motion from point features *(SFM-P)*. We used the popular VisualSFM application [31], created and made freely available by Changchang Wu. The second technique is Elqursh and Elgammal's algorithm [7], which uses lines (rather than point features) in a pair of images to estimate the relative camera pose *(SFM-L)*. Once the motion parameters $(\mathbf{R}, \mathbf{t})$ are obtained with either algorithm, 3-D lines are reconstructed from matched image line pairs. To check for coplanarity of a triplet of lines (at least two of which are parallel), we compute the associated Plücker matrices $\mathbf{L}_1$, $\mathbf{L}_2^*$ and $\mathbf{L}_3$, each normalized to unit norm (largest singular value), and threshold the norm of $\mathbf{L}_1\mathbf{L}_2^*\mathbf{L}_3$. By varying this threshold, we obtain a precision/recall curve. This evaluation was conducted with and without the "corrective" pre-processing step, discussed in Sec. 4, that rotates each line segment to align it with the associated vanishing point.

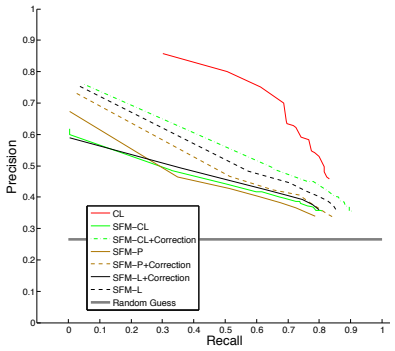

Fig. 6. Precision/recall curves for the algorithms considered (*SFM-P*, *SFM-L*, *SFM-CL*, *CL*) with and without the "correction" pre-processing step that aligns line segments with the associated vanishing point. (Note that the *CL* method is always computed with this correction.)

When assessing our characteristic line algorithm, we considered two different approaches for determining line triplet coplanarity: (a) From the estimated relative camera pose $(\mathbf{R}, \mathbf{t})$, as discussed above *(SFM-CL)*; (b) From clusters of characteristic lines *(CL)*. In the second approach, we rely on the fact that each characteristic line cluster represents a set of $\vec{n}$-coplanar lines. If all three lines in a triplet are contained in one such set of $\vec{n}$-coplanar lines, they are classified as coplanar. For the *CL* approach, the precision/recall curve was replaced by the Pareto front [32] of precision/recall values computed by varying the following parameters: (1) the constant σ in the function $g(D)$ defined in Sec. 3.5; (2) the threshold T, defined in Sec. 4, used to select the inlier characteristic line clusters.

Note that line detection and matching across images was performed automatically as described in Sec. 4. In some cases, lines were incorrectly matched; in this situation, line triplets containing the incorrectly matched lines were removed from the evaluation set (although both correctly and incorrectly matched lines were fed to the algorithms).

The precision/recall curves for all methods (with and without line re-orientation pre-processing) are shown in Fig. 6. Note that for two of the 49 image pairs considered, the VisualSFM application could not find any reliable point features and thus did not produce any results. Those two images were removed from the set used for the construction of the precision/recall curves. Without the "correction" step, the curves for *SFM-P*, *SFM-L* and *SFM-CL* are fairly similar (with *SFM-P* showing higher precision than the other two for low recall). When the correction pre-processing step is implemented, *SFM-CL* produces better results than *SFM-L* and *SFM-P*. This suggests that our algorithm can reconstruct the relative camera pose as well as or better than the other methods. The curve for *CL*, which does not require explicit 3-D line reconstruction, shows a substantial improvement. This demonstrates the power of the proposed algorithm for planar surface modeling and reconstruction.

6 Conclusions

We have introduced a new algorithm for the explicit detection of coplanar line sets and for the estimation of the camera motion in a Manhattan world. The algorithm is simple, easy to implement, fast, and produces comparatively excellent experimental results in terms of detection of coplanar lines. The main drawback of this approach, of course, is that it doesn't work in non–Manhattan environments, although it could conceivably be extended to support multiple plane orientations. The newly proposed characteristic line criterion allows for the analysis of line sets even when they are small in number and even when the lines are all parallel to each other (in which case, though, the camera motion cannot be recovered). It is, however, only a sufficient criterion, meaning that false positives are possible, although arguably rare. Future work will extend this technique to the case of line matches over more than two images.

Appendix: Characteristic Planes Revisited

We present here a different derivation of the characteristic planes concept, obtained through algebraic manipulations. For simplicity's sake, we will restrict our attention to one canonical plane Π_i, assuming that both cameras are located on it. A 2-D reference system is centered at the first camera. In this 2-D world, each camera only sees an image line, and the cameras' relative pose is specified by the (unknown) 2-D vector $\mathbf{t}$ and the (known) 2-D rotation matrix $\mathbf{R}$. We'll assume that both cameras have identity calibration matrices. Consider a plane Π_j with (known) normal $\mathbf{n}_j$, orthogonal to Π_i. A line $\mathcal{L}$ in Π_j intersects Π_i at one point, $\mathbf{X}$. Note that, from the image of this point in the first camera and knowledge of the plane normal $\mathbf{n}_j$, one can recover $\mathbf{X}/d$, where d is the (unknown) distance of Π_j from the first camera. Let $\hat{\mathbf{x}}_2$ be the location of the projection of $\mathbf{X}$ in the second camera's (line) image, expressed in homogeneous coordinates. From Fig. 1 one easily sees that $\lambda\hat{\mathbf{x}}_2 = \mathbf{R}\mathbf{X} + \mathbf{t}$ for some λ, and thus

$$\mathbf{t}/d = \lambda\hat{\mathbf{x}}_2/d - \mathbf{R}\mathbf{X}/d = \lambda_2\hat{\mathbf{x}}_2 - (\mathbf{R}\mathbf{X})_\perp/d \tag{6}$$

for some λ_2, where $(\mathbf{R}\mathbf{X})_\perp = \mathbf{R}\mathbf{X} - (\hat{\mathbf{x}}_2^T\mathbf{R}\mathbf{X})\hat{\mathbf{x}}_2/(\hat{\mathbf{x}}_2^T\hat{\mathbf{x}}_2)$ is the component of $\mathbf{R}\mathbf{X}$ orthogonal to $\hat{\mathbf{x}}_2$. This imposes a linear constraint on $\mathbf{t}/d$. It is not difficult to see that $\hat{\mathbf{x}}_2$ is orthogonal to the lever vector $\vec{u}_2$ in Fig. 1, and that $\|(\mathbf{R}\mathbf{X})_\perp/d\|$ is equal to the modulus of the sin ratio for the line $\mathcal{L}$ seen by the two cameras. Hence, the linear constraint in (6) is simply an expression of the intersection of the characteristic plane $\Pi(\mathcal{L}, \vec{n}_j)$ with Π_i.

Acknowledgement. The project described was supported by Grant Number 1R21EY021643-01 from NEI/NIH. The authors would like to thank Ali Elqursh and Ahmed Elgammal for providing the implementation of their method.

References

1. Harris, C.G., Pike, J.: 3D positional integration from image sequences. Image and Vision Computing **6** (1988) 87–90
2. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge Univ Press (2000)
3. Arth, C., Klopschitz, M., Reitmayr, G., Schmalstieg, D.: Real-time self-localization from panoramic images on mobile devices. In: 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR). (2011) 37–46
4. Tanskanen, P., Kolev, K., Meier, L., Camposeco, F., Saurer, O., Pollefeys, M.: Live metric 3d reconstruction on mobile phones. In: IEEE International Conference on Computer Vision (ICCV). (2013) 65–72
5. Košecká, J., Zhang, W.: Video compass. In: Computer Vision—ECCV 2002. Springer (2006) 476–490
6. Hwangbo, M., Kanade, T.: Visual-inertial uav attitude estimation using urban scene regularities. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE (2011) 2451–2458
7. Elqursh, A., Elgammal, A.: Line-based relative pose estimation. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. (2011) 3049–3056
8. Fitzgibbon, A.W., Zisserman, A.: Automatic camera recovery for closed or open image sequences. In: Computer Vision—ECCV'98. Springer (1998) 311–326
9. Bartoli, A., Sturm, P.: Structure-from-motion using lines: Representation, triangulation, and bundle adjustment. Computer Vision and Image Understanding **100** (2005) 416–441
10. Navab, N., Faugeras, O.D.: The critical sets of lines for camera displacement estimation: A mixed euclidean-projective and constructive approach. International Journal of Computer Vision **23** (1997) 17–44
11. Košecká, J., Zhang, W.: Extraction, matching, and pose recovery based on dominant rectangular structures. Computer Vision and Image Understanding **100** (2005) 274–293
12. Vincent, E., Laganiére, R.: Detecting planar homographies in an image pair. In: Image and Signal Processing and Analysis, 2001. ISPA 2001. Proceedings of the 2nd International Symposium on, IEEE (2001) 182–187
13. Sagüés, C., Murillo, A., Escudero, F., Guerrero, J.J.: From lines to epipoles through planes in two views. Pattern Recognition **39** (2006) 384–393
14. Guerrero, J.J., Sagüés, C.: Robust line matching and estimate of homographies simultaneously. In: Pattern Recognition and Image Analysis. Springer (2003) 297–307
15. Montijano, E., Sagues, C.: Position-based navigation using multiple homographies. In: Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on, IEEE (2008) 994–1001
16. Zhou, Z., Jin, H., Ma, Y.: Robust plane-based structure from motion. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE (2012) 1482–1489
17. Zhou, Z., Jin, H., Ma, Y.: Plane-based content-preserving warps for video stabilization. In: Computer Vision and Pattern Recognition, 2013. CVPR 2013., IEEE (2013)
18. Toldo, R., Fusiello, A.: Robust multiple structures estimation with j-linkage. In: Computer Vision–ECCV 2008. Springer (2008) 537–547

19. Sinha, S.N., Steedly, D., Szeliski, R.: Piecewise planar stereo for image-based rendering. In: ICCV, Citeseer (2009) 1881–1888
20. Hoiem, D., Efros, A.A., Hebert, M.: Recovering surface layout from an image. International Journal of Computer Vision **75** (2007) 151–172
21. Hedau, V., Hoiem, D., Forsyth, D.: Thinking inside the box: Using appearance models and context based on room geometry. Computer Vision–ECCV 2010 (2010) 224–237
22. Delage, E., Lee, H., Ng, A.Y.: A dynamic bayesian network model for autonomous 3d reconstruction from a single indoor image. In: Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on. Volume 2., IEEE (2006) 2418–2428
23. Lee, D.C., Hebert, M., Kanade, T.: Geometric reasoning for single image structure recovery. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE (2009) 2136–2143
24. Flint, A., Murray, D., Reid, I.: Manhattan scene understanding using monocular, stereo, and 3d features. In: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE (2011) 2228–2235
25. Ramalingam, S., Pillai, J.K., Jain, A., Taguchi, Y.: Manhattan junction catalogue for spatial reasoning of indoor scenes. In: Computer Vision and Pattern Recognition, 2013. CVPR 2013., IEEE (2013)
26. Tsai, G., Kuipers, B.: Dynamic visual understanding of the local environment for an indoor navigating robot. In: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, IEEE (2012) 4695–4701
27. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. Pattern Analysis and Machine Intelligence, IEEE Transactions on **24** (2002) 603–619
28. Grompone von Gioi, R., Jakubowicz, J., Morel, J.M., Randall, G.: LSD: a Line Segment Detector. Image Processing On Line **2012** (2012)
29. Wang, Z., Wu, F., Hu, Z.: Msld: A robust descriptor for line matching. Pattern Recognition **42** (2009) 941–953
30. Ronda, J.I., Valdés, A., Gallego, G.: Line geometry and camera autocalibration. Journal of Mathematical Imaging and Vision **32** (2008) 193–214
31. Wu, C.: VisualSFM. http://ccwu.me/vsfm/ (last checked: 6/15/2014)
32. Boyd, S., Vandenberghe, L.: Convex optimization. 2004. Cambridge Univ. Pr (2004)