Indoor Manhattan Spatial Layout Recovery from Monocular Videos via Line Matching

Chelhwon Kim and Roberto Manduchi

University of California, Santa Cruz

Abstract

We present an end-to-end system for structure and motion computation in a Manhattan layout from monocular videos. Unlike most SFM algorithms that rely on point feature matching, only line matches are considered in this work. This may be convenient in indoor environment characterized by extended textureless walls, where point features may be scarce. Our system relies on the notion of "characteristic lines", which are invariants of two views of the same parallel line pairs on a surface of known orientation. Experiments with indoor video sequences demonstrate the robustness of the proposed system.

1. Introduction

Structure from motion (SFM) has a long history in computer vision [1, 2]. Traditional SFM relies on the ability to detect and match across two or more views a substantial number of point features. Robust point detection and matching, however, can be challenging in indoor environments, where the density of detectable points (e.g. corners) may be low. This is particularly true in the presence of extended textureless walls. Specularities, which often occur with shiny surfaces or floor covers, may contribute to invalidate an already small pool of point feature matches.

If point features may be relatively scarce, line features are almost invariably present in these environment, due to plane intersections and other linear structures. In many cases, extended line features can be localized reliably in individual images, and geometric constraints can be used to ensure correct matching across views. A problem with SFM from lines is that, in the general case, at least three images are necessary for epipolar geometry reconstruction. If, however, the lines being matched are known to be coplanar, then four lines seen from two views suffice, provided that no line is parallel to the camera motion, and that no triplets of line have a common point of intersection or are mutually parallel. If the plane orientation and the rotation between the cameras are known, three coplanar lines are sufficient for reconstruction of the camera motion from two views, provided that the lines are not all mutually parallel.

In this contribution we introduce a technique for SFM over extended sequences from line matching in a Manhattan world, that is, a layout containing only planes at mutually orthogonal orientations. The Manhattan world represents an adequate model for most indoor buildings with vertical walls and square corners. In a Manhattan world, lines are normally mutually parallel or orthogonal, and parallel or orthogonal to the planar surfaces; estimation of the parallel lines' vanishing points enables computation of the planes' orientation in the scene with respect to each camera, and thus of the mutual camera orientation.

In previous work [3] we introduced the *characteristic lines (CL)* algorithm to find sets of coplanar lines from two views of a Manhattan world, thus enabling SFM computation. The CL algorithm performs a clustering of $\vec{n}$-characteristic lines, which are invariant representations of two views of parallel lines lying on a common plane with known orientation $\vec{n}$. The CL algorithm is fast and robust, and was shown to produce good results with challenging image pairs [3]. In this paper we extend the CL method for the analysis of videos from a monocular camera. In principle, it would be possible to simply extract motion estimates from pairs of views using 2-frames CL, and then feed these motion vectors to any existing algorithm for global motion computation from two-view constraints. We show that robustness can be increased dramatically by using a new multi-view CL technique that looks for clusters of vectors formed by characteristic lines over multiple view pairs. This technique requires individual lines to be tracked across multiple views; an algorithm for reliable line matching between two frames leading to the formation of "line chains" across multiple frames is presented here. Cluster centers of multi-view characteristic lines represent estimates of the camera motion between any two views, normalized by the distance from a planar surface of the first camera location in the pair. This information is passed on to a modified version of Özyeşil and Singer's "least unsquared deviations" (LUD) algorithm[4] that computes the global camera motion. While the original algorithm takes as input unit-norm translation vectors (directions) between view pairs, we modified it to take existing geometric constraints into account. Specifically, we leverage the fact that the translation vectors produced by multi-view CL for all view pairs that share one view have the same (unknown) scale factor. Finally, we introduce a new technique for planar fitting of the reconstructed lines that makes explicit use of the Manhattan world geometry.

This paper is organized as follows. After presenting the related work in Sec. 2, we describe the characteristic lines algorithm for motion and structure reconstruction from two views in Sec. 3. This section also introduces a new "line ordering constraint" method for robust matching. Sec. 4 describes the extension of the CL algorithm to multiple views; it includes our technique for line chain construction (4.1), multi-view characteristic lines clustering (4.2), motion reconstruction via modified LUD algorithm (4.3), and Manhattan structure computation (4.4). Sec. 5 presents results from experiments with videos taken from mostly textureless indoor environments. In the same section, we describe the techniques used for line detection and vanishing point computation. Sec. 6 has the conclusions.

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, 5, 6, 7, 8, 9, 10]. 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 [6, 11]. Kalman filter approaches for reconstruction from multiple views have also been proposed [12, 13].

A few authors have attempted to recover structure and motion using line features from only two views (as in our contribution), under strong assumptions (e.g., reliable estimation of segment endpoints across views [14]) or geometric priors (Manhattan world). Košecka and Zhang [15] 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 [16] 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 [17, 18]. Zhou et al. [19] 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 [20, 21, 22].

A more recent research direction looks to recover the spatial layout of an indoor scene from a single image [23, 24, 25]. Lee et al. [26] 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. [27] 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. [28] 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. [29] 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. Structure from Lines from Two Views

3.1. Notation and Basic Concepts

By *Manhattan world* [30] we mean an environment comprising only planar surfaces, each of which is oriented along one of three *canonical* mutually orthog-

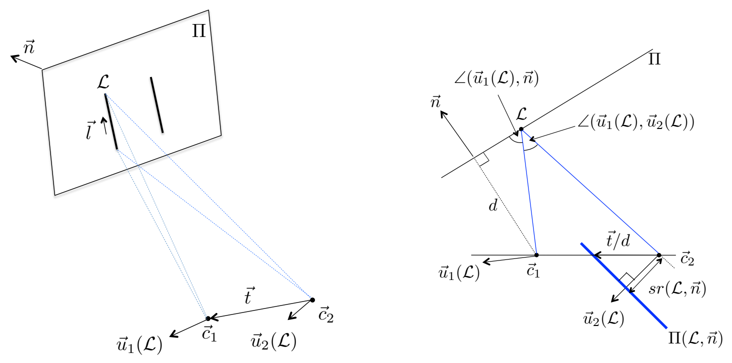


Figure 1: Left: The two camera centers $\vec{c}_1$, $\vec{c}_2$ and the lever vectors $\vec{u}_1(\mathcal{L})$, $\vec{u}_2(\mathcal{L})$ for line $\mathcal{L}$. Right: Line $\mathcal{L}$ lies on the plane $\Pi \equiv (\vec{n}, d)$ (both line and plane orthogonal to this page). The thick blue line is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

onal vectors[1] ($\vec{n}_1$, $\vec{n}_2$, $\vec{n}_3$). In addition, we will assume that each line[2] 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 from 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 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 [31]. 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 [32]. We will assume that the characteristic calibration matrices K_1, 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 (and, consequently, that R is known). We will also assume that lines visible in both images have been correctly matched; the algorithms used in our implementation for line detection and matching are presented in Sec. 5.1.1 and 3.3.1.

[1]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.

[2]For the sake of simplicity, we use the term "line" to indicate both a 3-D line and its projection onto an image. If there is risk of confusion, the latter will be termed "line image". "Characteristic lines" are geometric representations of linear constraints, and should not be confused with actual lines visible in the scene.

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). In a Manhattan world, surface planes Π and visible lines $\mathcal{L}$ 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 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 line images in the two cameras are related to one another as by $\mathbf{L}_1 = H^T \mathbf{L}_2$. The relationship between lines in space and line images is best described in terms of the *lever vector* $\vec{u}(\mathcal{L})$, which is a unit-norm vector orthogonal to the *projection plane* [14] (the plane containing $\mathcal{L}$ and the optical center of the camera; see Fig. 1, left panel). Expressed in terms of the associated camera reference frames, the lever vectors can be written as $\mathbf{u}_1 = K_1^T \mathbf{L}_1$ and $\mathbf{u}_2 = K_2^T \mathbf{L}_2$. The lever vectors are thus easily computed from the image of the line $\mathcal{L}$ in the two cameras. The following relation holds:

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

where $H_c = K_2^{-1} H K_1$ is the *calibrated homography matrix* induced by the plane, which can be decomposed [2] as

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

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.

3.2. Motion from Lines on a Plane with Known Orientation

By combining (1) and (2), one sees that the lever vectors associated with the same line $\mathcal{L}$ seen by two cameras are related as by

$$\mathbf{u}_1 = R^T \mathbf{u}_2 + \mathbf{n}\mathbf{u}_2^T \mathbf{t} / d \tag{3}$$

Thus, a single line on a plane with known normal $\vec{n}$ defines one linear constraint on $\vec{t}/d$ (since the matrix $\mathbf{n}\mathbf{u}_2^T$ has rank 1). The null space of solutions coincides with the second camera's projection plane of $\mathcal{L}$. The only information we can derive about $\vec{t}/d$ is its projection $\langle \vec{t}/d, \vec{u}_2 \rangle$ (as the lever vector $\vec{u}_2$ is orthogonal to this projection plane), which is equal to [3] (see the Appendix for a proof):

$$\langle \vec{t}/d, \vec{u}_2 \rangle = \frac{\sin \underline{/\vec{u}_1, \vec{u}_2}}{\sin \underline{/\vec{u}_1, \vec{n}}} \tag{4}$$

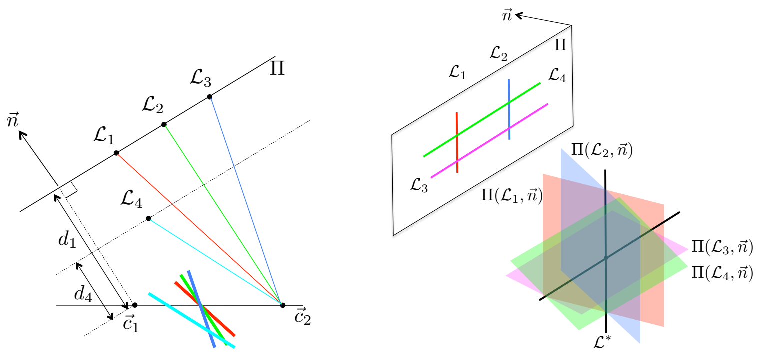


Figure 2: Left: Lines $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$ (orthogonal to this page) are $\vec{n}$-coplanar. Their associated $\vec{n}$-characteristic planes all intersect at a characteristic line through the baseline (also orthogonal to this page). They also individually intersect with 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, the vector $\vec{t}/d$ lies on a plane that is parallel to the projection plane of $\mathcal{L}$ on the second camera, at a (signed) distance $\langle \vec{t}/d, \vec{u}_2 \rangle$ from it. We call this the *$\vec{n}$-characteristic plane* $\Pi(\mathcal{L}, \vec{n})$ [3] (see Fig. 1, right panel). Note that $\Pi(\mathcal{L}, \vec{n})$ can be easily computed using (4), provided that the plane orientation $\vec{n}$ is known.

If a second line $\mathcal{L}_2$ is also seen that is coplanar with $\mathcal{L}$, one more linear constraint is added on $\vec{t}/d$. The space of solutions for $\vec{t}/d$ is the intersection of the two $\vec{n}$-characteristic planes $\Pi(\mathcal{L}, \vec{n})$ and $\Pi(\mathcal{L}_2, \vec{n})$. If $\mathcal{L}$ and $\mathcal{L}_2$ are parallel, the space of solutions is a line that is parallel to both $\mathcal{L}$ and $\mathcal{L}_2$ (as should be expected: moving either camera parallel to the lines does not change the line images). This line takes the name of *$\vec{n}$-characteristic line* $\mathcal{L}^*$ [3]. The *$\vec{n}$-characteristic line* of a pair of $\vec{n}$-coplanar lines can be computed easily from their images in the two views, as the intersection of the associated $\vec{n}$-characteristic planes.

If a third coplanar line is added, the vector $\vec{t}/d$ is fully determined, unless the three lines are mutually parallel (as in this case the associated lever vectors are all coplanar). In fact, for a bundle of parallel *$\vec{n}$-coplanar lines* (i.e., lying on the common plane oriented as $\vec{n}$), the $\vec{n}$-characteristic planes associated with the lines intersect in a common $\vec{n}$-characteristic line (see Fig. 2, left panel). Otherwise stated, any two parallel $\vec{n}$-coplanar lines in the bundle share the same $\vec{n}$-characteristic line. Thus, $\vec{n}$-characteristic lines represent an *invariant* of any number of $\vec{n}$-coplanar parallel lines. This property is at the basis of the characteristic lines algorithm for motion and structure computation proposed in [3], and briefly summarized in the following.

3.3. The Characteristic Lines Algorithm

As described in the previous section, matching a set of $\vec{n}$*-coplanar* lines across two views enables at least partial reconstruction of the "normalized" motion vector $\vec{t}/d$, provided that the plane orientation $\vec{n}$ is known. The remaining problem is to find, for each planar orientation $\vec{n}$, the groups of $\vec{n}$-coplanar lines. The characteristic lines (CL) algorithm [3] provides a fast and robust solution to this problem.

We'll start by considering a simple case with a bundle of parallel lines, all oriented along a direction $\vec{n}_i$ (in which case, as discussed earlier, only the projection of $\vec{t}/d$ on a plane orthogonal to the lines can be found). In a Manhattan world, each line in the bundle can belong to a plane with orientation of either $\vec{n}_j$ or $\vec{n}_k$ with $i \neq j \neq k$ (or to two planes, if the line is at a surface junction). For each orientation $\vec{n} = \vec{n}_j$ or $\vec{n} = \vec{n}_k$, we compute the $\vec{n}$-characteristic lines of pairs of lines in the bundle. The line pairs that are $\vec{n}$-coplanar share the same $\vec{n}$-characteristic line, whereas the $\vec{n}$-characteristic lines of non-$\vec{n}$-coplanar lines may be expected to distribute randomly. Hence, identifying the groups of $\vec{n}$-coplanar becomes a problem of finding the clusters of closely located characteristic lines. Each such cluster indicates the presence of a planar surface oriented as $\vec{n}$. Clustering can be performed (for example, using mean shift [33]) on the traces of the characteristic lines on the plane oriented as $\vec{n}_i$ and containing the second camera's optical center. The m-th cluster center represents an estimate of the projection of $\vec{t}/d_m$ on this plane, where d_m is the distance of the m-th planar surface to the first camera. Importantly, all clusters are expected to lie on the same line through the origin. For example, Fig. 3 shows the traces of $\vec{n}_2$- and $\vec{n}_3$-characteristic lines generated by pairs of vertical lines. The cluster center for each group of characteristic lines identifies a specific surface in the scene. Note that, due to the different orientation of the two planes, the clusters are at opposite sides with respect to the origin. The orientation of the translation $\vec{t}/d$ can be determined by a visibility test of the reconstructed lines.

In the general case with bundles of lines aligned along all three canonical orientations, we proceed as follows. For each possible plane orientation $\vec{n}_i$, we consider each bundle of parallel lines aligned along $\vec{n}_j$ and $\vec{n}_k$ in turn. We compute the $\vec{n}_i$-characteristic lines of each parallel line pair. If two line pairs, one pair oriented along $\vec{n}_j$ and one pair oriented along $\vec{n}_k$, are coplanar, then their $\vec{n}_i$-characteristic lines should intersect at $\vec{t}/d$. Based on this intuition, the CL algorithm finds points

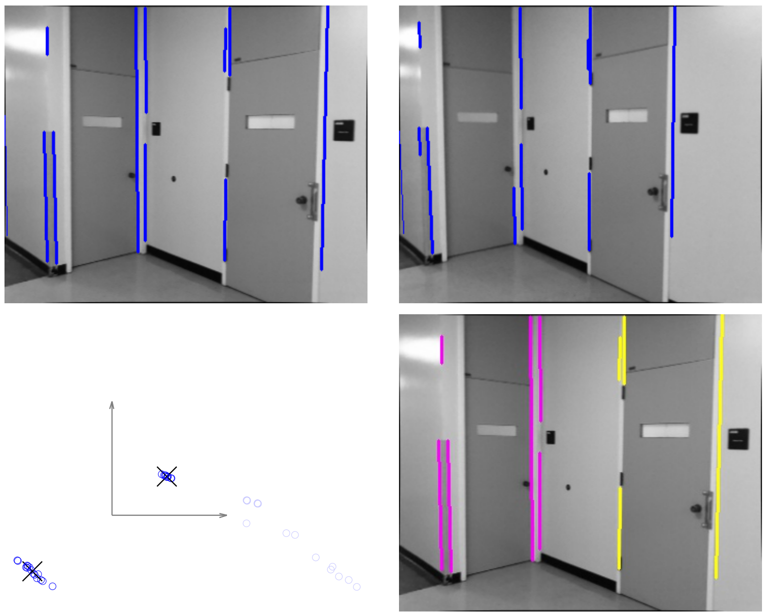


Figure 3: Top row: Image pair with detected lines oriented along one canonical direction ($\vec{n}_1$). Only lines that have been matched across images are shown. Bottom left: Traces of the $\vec{n}_2$- and $\vec{n}_3$-characteristic lines on a plane oriented as $\vec{n}_1$. The cluster centers, found 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).

in 3-D space that have a high density of nearby $\vec{n}_i$-characteristic lines (in either direction) using a modified mean-shift algorithm [3]. Fig. 4 shows $\vec{n}$-characteristic lines in all three directions, for two different orientations of $\vec{n}$. The cluster centers (shown by crosses, and found using the modified mean shift procedure) identify the three visible vertical surfaces

3.3.1. Line Matching

Our algorithm requires lines detected in each frame to be correctly matched across consecutive frames. Line matching is a notoriously difficult task. Part of the problem stems from the fact that, at least for the indoor environments considered here, different lines in the same image may appear very similar to each other. To characterize line appearance, we define a descriptor that takes into account both textural and color information in the areas adjacent to the line. We use the mean-standard deviation line descriptor (MSLD [34]), which defines pixel

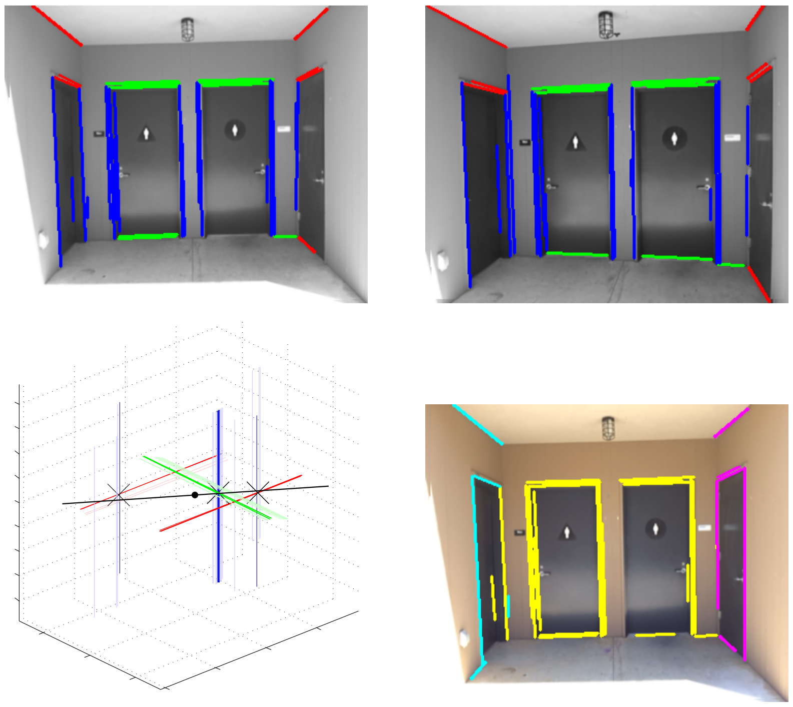


Figure 4: Top row: Image pair with detected lines oriented along the three canonical directions (the color of each line identifies its orientation). Only lines that have been matched across images are shown. Bottom left: Characteristic lines for the different orientations. The color of a characteristic line matches the color of the lines it represents. Clusters centers identified by the modified mean shift algorithm [3] 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).

support regions for each point in the line, from which SIFT-like descriptor (edge orientation histograms) are generated. In addition, as suggested by Bay et al. [35], we compute color histograms within each such region. More specifically, we first vector quantize the color of each pixel into one of the 11 "color names" defined in [36], then compute color vectors histogram on these 11 bins over the same regions used for MSLD. When matching two lines $\mathcal{L}_1$ and $\mathcal{L}_2$, we first compute the Euclidean distances d_T and d_C between the MSLD and color histograms associated with the two lines. Then, a "compound" similarity measure is defined as follows:

$$\text{Sim}(\mathcal{L}_1, \mathcal{L}_2) = w_T e^{-d_T^2/\sigma} + w_C^{-d_C^2/\sigma} \tag{5}$$

The coefficients w_T and w_C (with $w_T + w_C = 1$) are chosen adaptively so as to weigh the color term d_C more in poorly textured areas (as measured by the average brightness gradient magnitude at both sides of both matching lines), and less in texture-rich regions.

Textural and color information alone, however, cannot guarantee robust line matching. One approach to increasing robustness is to include nearby point feature correspondences and their topological layout [37, 38, 35]. We propose a different strategy, one that restricts the set of possible matches by defining a *line ordering constraint (LOC)*. LOC generalizes the well-known ordering constrain of points in individual epipolar lines, often used in stereo matching [39]. A counter-clockwise radial ordering of the images of a set of visible parallel lines is defined with respect to the line images' common vanishing point (Fig. 5). LOC posits that the pairwise ranking of two line images induced by the radial ordering in one view is preserved for the matching line images in the second view. Note that this constraint makes no assumptions about the epipolar geometry of the two cameras. Similarly to the standard ordering constraint on points in conjugate epipolar lines used in stereo matching, LOC may break down in the case of thin objects that are parallel to the considered 3-D lines. Dynamic programming can be used to find a set of line matches that are LOC-consistent while minimizing a global cost comprising an appearance term (the similarity $\text{Sim}(\mathcal{L}_1, \mathcal{L}_2)$ defined in (5)) and a "skip" term (a constant cost per line skipped in each image). A similar approach was taken by Cornelis et al. [40], who defined an ordering constraint on the vertical lines detected in properly warped images. Note that image warping (which requires estimation the epipolar geometry) is not needed using the counter-clockwise radial line ordering introduced here.

To evaluate the effectiveness of the LOC algorithm in terms of line matching accuracy, we compared it against the commonly used Nearest-Neighbor-Ratio criterion (NNR) that computes the ratio of the distances to the nearest and to the second nearest neighbor [41, 34]. Specifically, we only kept matches with a compound similarity measure (5) larger than 0,65, and with a NNR value larger than 0.8. Table. 1 shows comparative results over a set of 6 indoor image pairs in terms of the number of correct matches, the total number of matches, and their ratio (correct match ration, CR), as in [34]. Overall, the LOC algorithm produced a higher average number of total matches as well as a higher average CR than the NNR criterion.

Images	1	2	3	4	5	6	Total	CR
NNR	14(21)	22(32)	27(32)	26(26)	16(16)	18(21)	123(148)	83%
LOC	22(26)	29(32)	38(39)	34(34)	20(21)	16(19)	159(167)	95%

Table 1: Line matching accuracy tests on 6 image pairs using the NNR criterion [41, 34] and the proposed LOC alorithm. For each column, the first number is the number of correct line matches, while the number in parentheses is the number of total matches. CR is the correct match ratio.

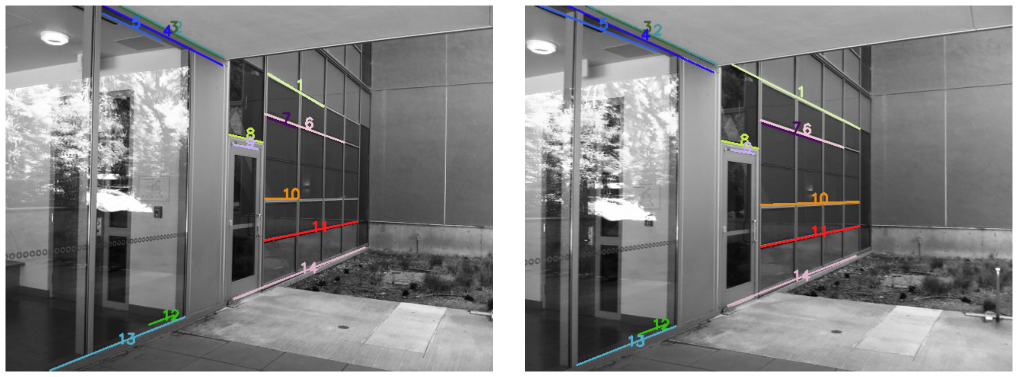


Figure 5: The line ordering constraint (LOC) improves robustness of line matching in the case of repeated linear structures. Only matched lines oriented along one of the three canonical directions are shown (the color of each line identifies its match pair). The numbers on the matched lines indicate the counter-clock wise radial ordering with respect to the lines' common vanishing point.

4. Structure from Lines – Multiple Views

We now extend the CL algorithm to the case of multiple views of the same scene. The first step is to compute "line chains", that is, sets of line images across multiple views that are projections of the same line in space. We then compute a clustering in a higher dimension space, where data points are vectors formed by concatenating 2-D characteristic lines for the same line pair seen from multiple views. The cluster centers directly provide estimation of scaled motion between two camera views; a modified LUD algorithm is then employed for robust motion estimation. Finally, planar patches are fitted to the triangulated lines using Manhattan world geometric constraints.

4.1. Line Chain Construction

As will result clear in Sec. 4.2, our algorithm for multi-view SFM benefits from tracking the same line across multiple frames. The simplest approach to building *line chains* (sequences of matching line images in consecutive frames) would be to match lines in consecutive frame pairs, assigning the same label to the two lines, one per frame, being matched (each label identifying one line chain). Unfortunately, this algorithm may lead to undesired effects in the event of occasional line breakage. In order to analyze this problem and to justify the proposed solution, let us first formally define a line chain as a connected component of the directed graph $\mathcal{G} = (V, E)$ whose nodes V represent individual image lines, and whose edges E connect two nodes if the lines associated with the nodes belong to different frames and have been matched by our algorithm (the direction of the edge pointing to the more recent frame). Note that in the simple case of lines matched across two consecutive views, each node has maximum degree of 2. The fragmentation problem mentioned above can be formalized as a node split in the graph $\mathcal{G}$, where the nodes resulting from the split are disconnected from each other. While line fragmentation is not an issue with the 2-frames characteristic lines algorithm (as each such segment correctly identifies the line it belongs to), it becomes a problem for line chain construction, since a node split may break an otherwise connected line chain (Fig. 6).

To mitigate problems associated with fragmentation, we implemented a 3-frames line matching algorithm that uses dynamic programming to generate LOC-compliant solutions (in a similar fashion as for the two-frame LOC matching of Sec. 3.3.1). Given the small number of lines typically found in each image (about 70 on average), the additional computational cost is well affordable. 3-frames line matching ensures that sporadic line splits do not break a line chain (see Fig. 7). The price to pay is that now nodes in the graph may have a degree as high as 4, which brings on the risk of merging connected components (line chains) in the case of fragmentation or mismatches (Fig. 7).

To avoid this, we implement a pruning procedure on the graph that limits the number of incoming edges (indegree) and of outgoing edges (outdegree) at each node. Specifically, we endow the edge linking the nodes associated with lines $\mathcal{L}_i$ and $\mathcal{L}_j$ with the "similarity" value $\text{Sim}(\mathcal{L}_i, \mathcal{L}_j)$ defined in (5). In the case of a node with indegree (outdegree) larger than 1, only the incoming (outcoming) edge with highest similarity is kept. This simple solution has given us good results; an example of resulting line chain is shown in Fig. 8.

After the line chains in the sequence have been formed, we define an ordering on them; this is used both for characteristic line clustering (Sec. 4.2) and for plane fitting (Sec. 4.4). The ordering is induced by the radial line ordering around the vanishing points (LOC, Sec. 3.3.1) at each image. The line chains are ordered in such a way that, for any two line chains and for any view that sees both lines in the chain, the ranking of the two line chains and of the lines in the view (as defined

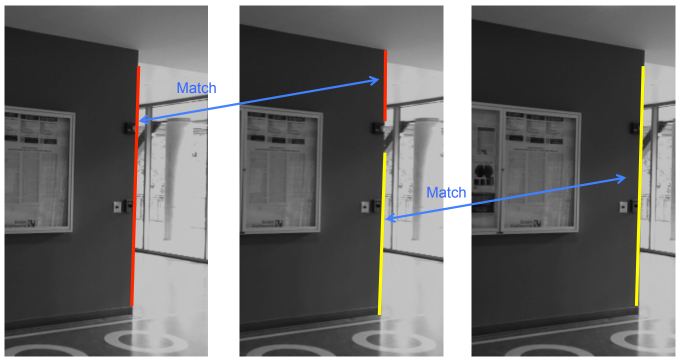


Figure 6: The line fragmentation problem for line chain construction. The same line segment (corresponding to the edge of the red wall) is seen in three views, but it is split in two segments in the second view. The segment in the first view is matched to one of the two segments, while the segment in the third view is matched with the other segment, impeding formation of a line chain.

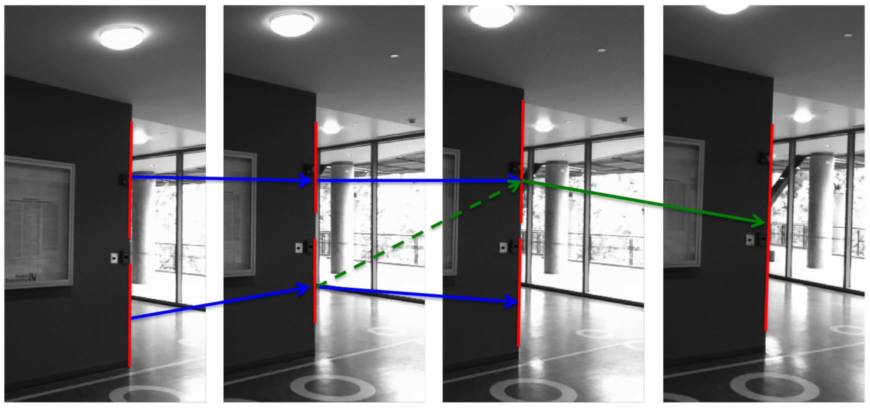
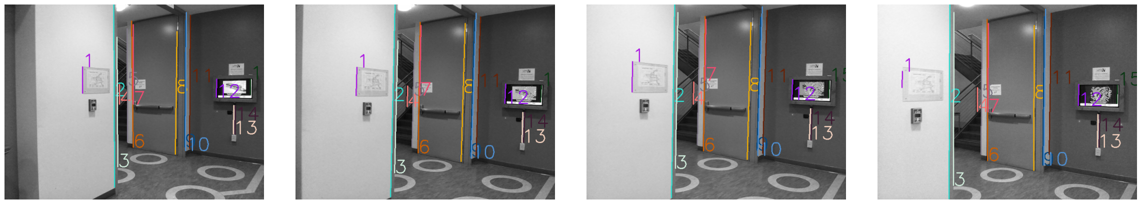


Figure 7: A sequence of images with the 3-frames line matching results. Two triplets of lines are found by the 3-frames line matching algorithm for the first three frames (two pairs of blue arrows). One triplet of lines are found for the next three frames (shown by a pair of green arrows). The bottom line segment in the second frame is mistakenly matched with the upper line segment in the third frame. Our algorithm removes the incoming edge (a green dashed arrow) to the upper line segment in the third frame (which is also the outgoing edge from the bottom line segment in the second frame).

by the LOC) is the same.

4.2. *The Multi-View Characteristic Lines Algorithm*

The 2-frames characteristic lines algorithm [3], summarized in Sec. 3.3, attempts to determine which line pairs are $\vec{n}$-coplanar by looking at local concen-



frame 1 frame 4 frame 7 frame 10

Figure 8: Line chains correctly identified across four image frames. Only lines that oriented along the vertical direction are shown.

trations of $\vec{n}$-characteristic lines. Unfortunately, depending on the camera motion relative to the scene geometry, characteristic lines clustering from a single pair of views may be challenging or impossible. For example, when camera motion between the two views is small or is almost parallel to the direction of parallel lines in the scene, clusters of characteristic lines may be located very close to each other, which complicates individual cluster identification in the presence of noise.

It is reasonable to assume that, if the same lines are seen from multiple views, evidence from all such views could contribute to understanding whether the lines in the pair are indeed $\vec{n}$-coplanar, and thus increase reliability of camera motion estimation. The idea is that view pairs with large baseline, or with baseline that is not aligned with visible parallel lines, could compensate for other, less informative view pairs. We bear this intuition to fruition by the algorithm described next.

4.2.1. A Simple Case: All Lines Seen by All Views

Suppose for the time being that a set of N views see the same set of parallel lines in the scene, and that we are interested in finding the $\vec{n}$-coplanar groups of lines in this set (where $\vec{n}$ is a canonical planar direction that is orthogonal to the lines). Also assume that each line has been correctly tracked across the N views, forming an N-long line chain. One could, in principle, consider all $\begin{pmatrix} N \\ 2 \end{pmatrix}$ pairs of views, run the regular CL algorithm on each view pair (by computing a 2-D clustering of the characteristic line traces), and "digest" this set of results to decide which lines are $\vec{n}$-coplanar. We propose a different solution, one that computes the characteristic lines clustering only once, but in a space with dimension of $N(N-1)$. Clustering is performed on vectors formed by concatenating the 2-D characteristic line traces obtained from all view pairs. Mean shift is used to find the modes of this data point distribution.

Remember that mean shift finds modes of a probability density function as linear combinations of samples generated by this density, where the weights of the combination are refined at each iteration. Specifically, the weight of each sample is proportional to the value at that sample location of a kernel (e.g. a Gaussian kernel) centered on the current mode estimate at that iteration. Now consider an indexing $\{m\}$ of all (ordered) view pairs, and an indexing $\{p\}$ of all (ordered) parallel line pairs. Let $\mathbf{l}^*_{m,p}$ be the trace of the p-th $\vec{n}$-characteristic line on an orthogonal plane for the m-th view pair, and let $\mathbf{l}^*_p$ be the vector formed by concatenating the p-th line traces for all view pairs. Given the current mode estimate $\hat{\mathbf{l}}^*$, the p-th vector is assigned weight w_p with

$$w_p = K \exp\left(-c\|\hat{\mathbf{l}}^* - \mathbf{l}^*_p\|^2\right) = K \exp\left(-c\sum_m \|\hat{\mathbf{l}}^*_m - \mathbf{l}^*_{m,p}\|^2\right) \tag{6}$$

where $\hat{\mathbf{l}}^*_m$ is the component of the current mode estimate for the m-th view pair, and c and K are constant (K is chosen to ensure that the weights w_p sum up to 1). Updates are performed for each view pair independently, but using the global weights $\{w_p\}$:

$$\hat{\mathbf{l}}^*_m = \sum_p w_p \mathbf{l}^*_{m,p} \tag{7}$$

Fig. 9 shows a comparison of multi-view CL clustering vs. regular 2-view CL clustering in the case of small camera motion. Both methods are tested on four image pairs with different baseline lengths. These images were extracted from a sequence; we considered the image pairs with indices (0,3) (shown in the figure), (0,6), (0,9), and (0,12). The camera moved of uniform motion approximately along the y direction (horizontal and parallel to the side walls), resulting in increasing baseline $\vec{t}$ for the selected image pairs (e.g. (0,12) has a larger baseline than (0,3)).

For each image pair, the characteristic lines for the y and for the z (vertical) directions are plotted on the x-y plane (the y-characteristic lines are shown as dashed green lines, while the intersections of the z-characteristic lines with the x-y plane are shown with blue dots). The characteristic lines cluster centers computed by the two algorithms are shown for each image pair as circles (2-view algorithm) or crosses (multi-view algorithm). Ideally, the characteristic lines corresponding to coplanar parallel lines should all intersect at point $\vec{t}/d$, where d is the (approximately constant) distance to the side wall, and this point should lie close to the y axis (as $\vec{t}$ is approximately parallel to y). The cluster centers produced by multi-view clustering conform to this expectation (note that the distance of the cluster to the origin increases with the baseline $\vec{t}$). 2-view clustering produces less accurate results, especially for the pairs with smaller baseline. Although these results could conceivably be improved by adjusting the size of the mean-shift kernel, we noticed that finding a kernel size that works for different baselines is challenging for the 2-view clustering algorithm. In contrast, multi-view clustering seems to be less sensitive to the choice of kernel size.

4.2.2. The General Case: Visibility Sets

In general, different views from a moving camera see different sets of lines, with large overlap only between nearby views. Fig. 10 (a) shows the *visibility table* for a certain sequence, where each column represents a view pair and each row represents a parallel line pair. A '1' entry in the (p, m) position means that both views in the m-th pair see both lines in the p-th pair. The visibility table is built from the computed line chains (Sec. 4.1).

The *visibility set* $\mathcal{V}_p$ of the p-th line pair is defined as the set of view pairs that see both lines in the p-th pair (i.e., the set of column indices with non-null entries in the p-th row of the visibility table). We modify the equation for weight computation (6) to take visibility sets into account as follows:

$$w_p = K\,|\mathcal{V}_p| \exp\left(-c \sum_{m\in\mathcal{V}_p} \frac{\|\hat{\mathbf{l}}^*_m - \mathbf{l}^*_{m,p}\|^2}{|\mathcal{V}_p|}\right) \tag{8}$$

The Gaussian kernel is computed on the average squared distance between the line trace $\mathbf{l}^*_{m,p}$ and the current mode estimate at each view in the visibility set. The factor $|\mathcal{V}_p|$ gives higher weight to lines that are seen by many views.

Care must be taken during component-wise update. Let $\mathcal{W}_m$ be the set of line pairs seen by the m-th view pair (i.e., the set of row indices with non-null entries in the m-th column of the visibility table). The update equation (7) is modified as follows:

$$\hat{\mathbf{l}}^*_m = R_m \sum_{p\in\mathcal{W}_m} w_p \mathbf{l}^*_{m,p} \text{ with } R_m = \frac{1}{\sum_{p\in\mathcal{W}_m} w_p} \tag{9}$$

4.2.3. Multiple Line Orientations

Until now we have considered the case of a parallel lines bundle. In order to estimate camera motion, we need to consider both line directions orthogonal to the planar orientation $\vec{n}$. For this purpose, we use the modified mean shift algorithm introduced in [3]. We give only a short summary of the algorithm in the 2-views

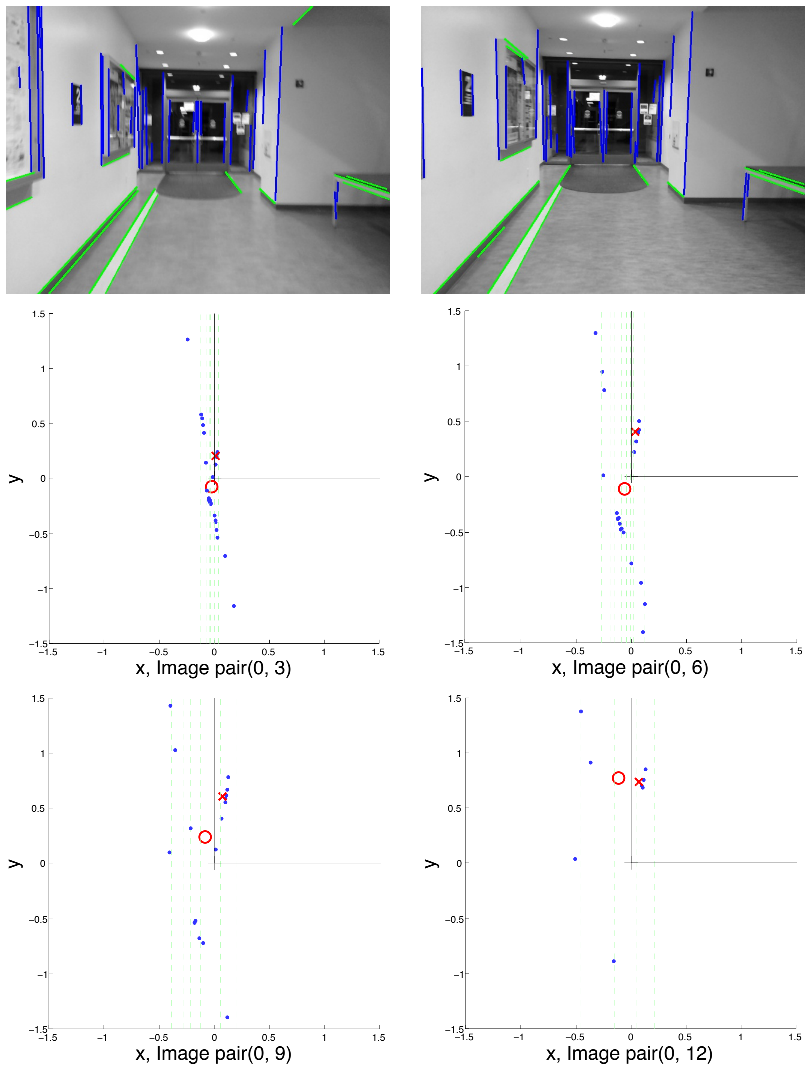


Figure 9: 2-view vs. multi-view characteristic line clustering (Sec. 4.2.1). Top: Image frames 0 and 3 with detected lines oriented along the y (green) and z (blue) canonical directions. Second and third rows: Characteristic lines for the y (dashed green lines) and z (blue dots) directions plotted on the x-y plane for the view pairs (0, 3), (0, 6), (0, 9), and (0, 12). Red circles: Cluster centers identified by 2-view clustering. Red crosses: Cluster centers identified by multi-view clustering.

case here; the reader is referred to [3] for a detailed description. Remember that the characteristic line of a parallel line pair has the same orientation as the lines in the pair. The characteristic lines of line pairs in two orthogonal bundles are thus mutually orthogonal. The goal is to find a 3-D point that has in its neighborhood a high concentration of characteristic lines in both directions. The modified mean shift algorithm achieves this goal by alternating mode estimation update in the three canonical directions, each time using only one appropriate set (or both) of characteristic lines for weight computation. Extension to the multi-view case using the update equations (8) and (9) is relatively straightforward.

4.2.4. Characteristic Lines Selection

While the original CL algorithm operates on all characteristic lines, we found it beneficial to select a subset with high likelihood of being $\vec{n}$-coplanar before clustering. Specifically, we select the characteristic lines formed by pairs of parallel lines whose rank (induced by the ordering of the associated line chains introduced in Sec. 4.1) does not differ by more than 3. These lines may be expected to be relatively close in the image, and thus have a good chance of belonging to the same visible surface.

4.2.5. Clusters Selection

We run the iterative clustering procedure starting from a random set of line chain pairs, resulting in a number of clusters for each planar orientation $\vec{n}$.

Each cluster is characterized by its *cluster visibility set*, which is the union of visibility sets $\mathcal{V}_p$ of the line pairs associated with the characteristic lines contained in the cluster. As seen in Fig. 10 (c), clusters visibility sets often overlap with each other, which may indicate that they are generated by the same surface. To select representative clusters, we follow the following strategy. We create a graph with clusters as nodes, and edges weighted by the Jaccard distance (1 minus the ratio of the cardinalities of the intersection and of the union) of the cluster visibility sets for the two nodes linked by the edge. The connected components of this graphs are found (Fig. 10 (d)), and only one representative per component (specifically, the one with highest associate density value as computed by mean shift) is kept (Fig. 10 (e) and (f)).

Each cluster center $\hat{\mathbf{l}}^*$ encodes the estimated normalized camera translation vectors $\{\vec{t}_m/d_m\}$ between the views in the m-th pair, where d_m is the distance between the surface identified by the cluster and the first camera in the m-th view pair.

4.3. Global Motion Computation

The multi-view intrinsic line clustering described in the previous section produces a number of "normalized" motion vector estimates $\{\vec{\tau}_{i,j,k} = \vec{t}_{i,j}/d_{i,k}\}$, where $\vec{t}_{i,j}$ represents $\vec{c}_j - \vec{c}_i$, the motion from the i-th to the j-th view, and $d_{i,k}$

is the distance from the camera in the i-th view to the k-th visible surface in the scene. (Remember that different parallel surfaces produce distinct characteristic line clusters.) We use the Least Unsquared Deviations (LUD) algorithm [4] to reconstruct the camera motion vectors $\{\vec{c}_i\}$. This was shown to be more robust than the least squared deviation formulation [42] in the presence of outliers. We also use a parallel rigidity test as in [4] for unique realizability of camera locations from pairwise directions. The parallel rigidity test [43] is used to maintain well-posed instances of the camera location estimation problem. For ill-posed instances of the problem we extract maximally parallel rigidity components in the camera location formation as in [44].

The original LUD algorithm takes in input a set of unit-norm vectors $\{\vec{\gamma}_{i,j}\}$, and solves the following problem:

$$\begin{aligned} &\underset{\{\vec{t}_i\},\{\delta_{i,j}\}}{\arg\min} \textstyle\sum_{(i,j)} \|\vec{c}_j - \vec{c}_i - \delta_{i,j}\vec{\gamma}_{i,j}\| \\ &\text{s.t. } \textstyle\sum_i \vec{c}_i = 0 \; ; \; \delta_{i,j} \geq D \end{aligned} \tag{10}$$

In the equation above, the index pairs (i, j) include all view pairs for which a normalized motion estimate could be computed, and D is a constant. Unit norm vectors $\{\vec{\gamma}_{i,j}\}$ represent the motion directions computed for the view pairs.

We slightly modified the formulation in (10) to include all available constraints. Instead of using unit-norm motion vectors $\{\vec{\gamma}_{i,j}\}$, we use our computed values $\{\vec{\tau}_{i,j,k}\}$:

$$\begin{aligned} &\underset{\{\vec{t}_i\},\{\delta_{i,k}\}}{\arg\min} \textstyle\sum_{(i,j,k)} \|\vec{c}_j - \vec{c}_i - \delta_{i,k}\vec{\tau}_{i,j,k}\| \\ &\text{s.t. } \textstyle\sum_i \vec{c}_i = 0 \; ; \; \delta_{i,k} \geq D \end{aligned} \tag{11}$$

This formulation is very similar to (10), except for the fact that the vectors $\{\vec{\tau}_{i,j,k}\}$ are not unit-norm, and the sum now extends not only to all view pairs, but to the combination view pairs/surfaces. In addition, the values $\{\delta_{i,k}\}$ are only defined for combinations view/surfaces, rather than for view pairs (since all view pairs (i, j) for fixed i share the same distance $d_{i,k}$ to the k-th surface from the i-th view). The solution can be computed using exactly the same procedure as for the original LUD problem. Importantly, in the noiseless case ($\vec{\tau}_{i,j,k} = (\vec{c}_j - \vec{c}_i)/d_{i,k}$), any solution of (11) is congruent with the true set of motion vectors $\{\vec{c}_i\}$. Note that there are usually much fewer variables to optimize than with the original formulation (10), as only a small number of different surfaces are visible in typical

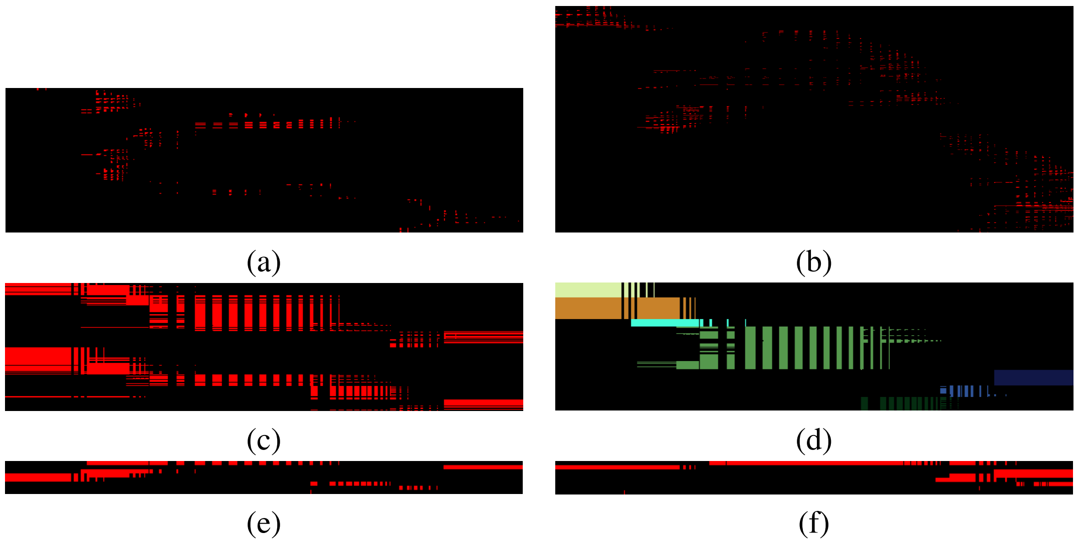


Figure 10: (a),(b): Visibility tables. The (p, m) cell of each table indicates whether the p-th line pair (oriented along the X direction (a) or the Z direction (b)) is visible by the m-th view pair. (c): The cluster visibility table for lines in both the X and Z direction. Clusters were computed on $\vec{n}$-characteristic lines (with $\vec{n}$ oriented in the Y direction) in both the X and Z direction using the modified mean shift algorithm. (d): Cluster visibility table highlighting the connected components of the cluster visibility table (Sec. 4.2.5). (e): The selected clusters. (f) Selected clusters computed from $\vec{n}$-characteristic lines, with $\vec{n}$ oriented along the X direction.

images. The modified LUD problem is solved by an iteratively reweighted least squares (IRLS) solver [45, 46].

4.4. Manhattan Structure Computation

Given the sequence of camera locations (Sec. 4.3) and orientations (from vanishing points), we can reconstruct the geometry of the environment from features (lines) triangulation. The Manhattan world assumption provides strong priors on the geometry of the scene; in particular, for wall layout reconstruction, it can be assumed that each wall is vertical and oriented in one of two possible known directions (indicated as X or Y in the following). We leverage this prior information to build a piecewise planar fit to the triangulated data.

The first step in our algorithm is the computation of the location of the visible lines in space. For each canonical direction, we consider each line chain (Sec. 4.1), determine its visibility set, and compute a 2-D least-squares triangulation of the line's traces on an orthogonal plane [2]. We then perform bundle adjustment [2] using the Ceres solver [47] on the computed lines, motion vectors, rotation matrices, and focal lengths (from prior camera calibration). The segments whose trace on an orthogonal plane has reprojection error larger than 5 pixels after bundle adjustment are discarded.

For each horizontal canonical direction (say, X), we consider the vertical lines' traces on the horizontal plane, as well as the horizontal lines oriented as Y, and project these points/lines orthogonally onto the X axis (each horizontal line contributes one point to the projection). The modes of the resulting distribution (computed via mean shift) determine the main vertical surfaces $\{\Pi_j\}$ in the scene oriented as X (see Fig. 11). The next step is to assign each vertical line to one surface in either orientation. This operation is complicated by noise in the line position measurement, and by the fact that, in the proximity of surface intersections, line-surface association can be ambiguous (see Fig. 11). We frame this task as a minimum cost path problem in a directed graph, leveraging spatial coherence priors.

We define a directed graph, where each node represents an association between a vertical line and one of the planes found in the previous step. A generic node associating the i-th line in the sequence with the plane Π_j is indexed as $N_{i,j}$. Two nodes that are associated with consecutive lines (according to the ordering defined in Sec. 4.1) are linked by an edge. Nodes and edges are assigned cost values; the minimum cost path determines the line/plane association. We define nodes and edges costs as follows.

4.4.1. Node Costs

The cost at node $N_{i,j}$ is simply a function d of the distance of the i-th vertical line to the plane Π_j (equal to $1 - exp(-d^2/c)$ for a suitable constant c) – see Fig. 11. While some lines can be safely assigned to the closest plane, in other cases distance-based association would lead to incorrect results.

4.4.2. Edge Costs

The edge linking $N_{i,j}$ with $N_{i+1,k}$ is assigned a cost that is equal to 1 minus the conditional probability of the assignment $N_{i+1,k}$ given $N_{i,j}$. This probability is computed in terms of the *reconstructed lines alignment* and *deviation*, which measure how well the i-th and the $(i+1)$-th line traces align with either the X or the Y axis. The reconstructed lines alignment term $rla^X_{i,i+1}$ or $rla^Y_{i,i+1}$ is equal to the magnitude of the cosine of the angle formed by the vector $\vec{r}_{i,i+1}$ joining the traces of the i-th and of the $(i+1)$-th line with the Y or X axis, respectively. The reconstructed lines deviation term $rld_{i,i+1}$ is the magnitude of the cosine of the angle formed by $\vec{r}_{i,i+1}$ and a line at $45°$ in the X-Y plane.

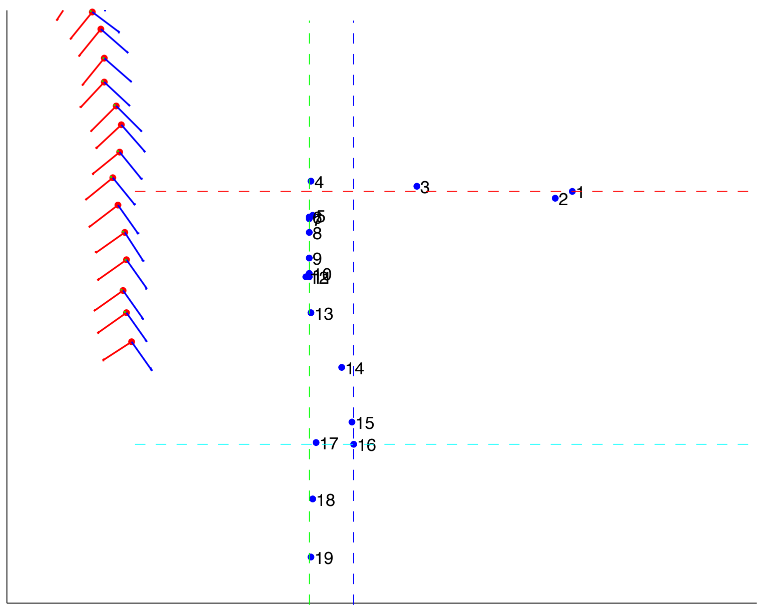


Figure 11: The main vertical surfaces in the scene oriented along X and Y (dashed lines) are shown together with the traces of the vertical lines on the horizontal plane (blue points) and the estimated camera location and orientation in the trajectory.

The alignment/deviation terms measure the evidence provided by observations o (characteristic line, reconstructed lines) to the hypotheses that the two lines are aligned with the X axis, the Y axis, or neither. By assigning proper prior probabilities, we can compute the posterior probabilities that the i-th and $(i+1)$-th lines are $\vec{n}$-coplanar with $\vec{n}$ oriented as X (event denoted by $\|^X_{i,i+1}$), as Y ($\|^Y_{i,i+1}$), or that they are not $\vec{n}$-coplanar for either axis ($\bar{\|}_{i,i+1}$):

$$\begin{aligned} P(\|^X_{i,i+1}|o) &= K \cdot P(o|\ \|^X_{i,i+1}) \cdot P(\|^X_{i,i+1}) = K \cdot \mathit{rla}^X_{i,i+1} \cdot P(\|^X_{i,i+1}) \\ P(\|^Y_{i,i+1}|o) &= K \cdot \mathit{rla}^Y_{i,i+1} \cdot P(\|^Y_{i,i+1}) \\ P(\bar{\|}_{i,i+1}|o) &= K \cdot \mathit{rld}_{i,i+1} \cdot P(\bar{\|}_{i,i+1}) \end{aligned} \tag{12}$$

where the normalization constant K ensures that the three posteriori probabilities sum up to 1. In our experiments, we set the priors as follows: $P(\|^X_{i,i+1}) = P(\|^Y_{i,i+1}) = 0.4$, $P(\bar{\|}_{i,i+1}) = 0.2$.

The formulas above express our belief that the two lines belong to a plane oriented as X or Y, or neither. We now leverage simple geometric reasoning and spatial coherence priors to transform these in conditional probabilities $P(N_{i+1,k}|N_{i,j})$ (where we use the same notation for a node in the graph and for the association event it represents). Using the total probability theorem,

$$\begin{aligned} P(N_{i+1,k}|N_{i,j},o) &= P(N_{i+1,k}|N_{i,j},\|_{i,i+1}^{X}) \cdot P(\|_{i,i+1}^{X}|o) \\ &+ P(N_{i+1,k}|N_{i,j},\|_{i,i+1}^{Y}) \cdot P(\|_{i,i+1}^{Y}|o) \\ &+ P(N_{i+1,k}|N_{i,j},\overline{\|}_{i,i+1}) \cdot P(\overline{\|}_{i,i+1}|o) \end{aligned} \tag{13}$$

where we took into consideration the fact that the observations considered here (characteristic line, reconstructed lines) can only help determine the coplanarity (or lack thereof) of the lines. We assign to the first factor in each term in the sum the value v if $j = k$, or ϵv if $j \neq k$, where $0 < \epsilon < 1$. Multiplication by ϵ is meant to discourage frequent jumps in plane assignment. v can take one of two values: l or h, with $0 < l < h = 2l < 1$. The choice between l and h depends on the assumed $\vec{n}$-coplanarity of the lines, as well as on the orientation of the planes $\Pi_j$and Π_k. If the lines are $\vec{n}$-coplanar in the X orientation ($\|_{i,i+1}^{X}$), v is assigned the value h only when the planes are identical ($j = k$) and oriented as X; in all other cases, $v = l$. Similar reasoning apply to the event $\|_{i,i+1}^{Y}$. If the lines are not $\vec{n}$-coplanar in either direction ($\overline{\|}_{i,i+1}$), v is assigned the value l only when the planes are identical ($j = k$); v is set equal to h otherwise.

5. Experiments

We have tested our system with video sequences taken inside a university building characterized by mostly textureless walls, with relatively narrow corridors linked by hallways. The videos were taken with an iPhone 4S at VGA resolution and at a frame rate of 7 frames/s, while walking and holding the iPhone straight up, the camera looking forward. Timestamped measurements from the accelerometers were also taken; these were used to aid in vanishing point computation (Sec. 5.1.2). We used all frames in the video to compute line chains, but computed SFM only using one frame out of three; this was done to reduce computational time (Sec. 5.1.4). Our results are presented in Sec. 5.2; implementation details are presented next.

5.1. Implementation Details

5.1.1. Line Detection

We use the LSD (Line Segment Detector) algorithm [48] to find line segments. This algorithm works in linear time, and does not require any parameter tuning. Short line segments (less than 30 pixels in length) are removed, and nearby line segments of similar orientation are merged using the algorithm in [49]. More specifically, a fitting line is computed through the center of mass of the vertices of the two segments, with orientation angle equal to the mean of the orientation angles of the two segments, weighted by their lengths. The "merged segment" is defined as the shortest segment on this line containing the projections of all endpoints of the original segments. However, if the average distance of the original segments endpoints to the fitting line is larger than 1 pixel, or if there is a gap of more than 50 pixels between the projections of the two segments on the fitting line, the merged segment is discarded and the original segments are retained.

5.1.2. Vanishing Points Estimation

Following [32], we assume that one canonical direction is aligned with the gravity vector $\vec{v}_Z$, which can be found using the smartphone's accelerometers. The segments in the image that converge towards the associated vanishing point are removed. The goal now is to find the two vanishing points for the remaining (horizontal) segments. Towards this goal, we use the following RANSAC-like procedure. At each iteration, we randomly select one line segment and compute the associated lever vector $\vec{u}$ (Sec. 3.1). We then compute the vanishing points of two horizontal orthogonal directions $\vec{v}_Z \times \vec{u}$ and $(\vec{v}_Z \times \vec{u}) \times \vec{v}_Z$, and count the line segments aligned with either vanishing point (specifically, we measure the angle between each line's lever vector and each vanishing points, and assign a line to a vanishing point if the angle has magnitude less than $5°$).

After a number of iteration, the vanishing points with highest number of supported segments are retained. Finally, three mutually orthogonal vectors are computed by minimizing the alignment errors with all line segments using non-linear optimization [47]. Once the vanishing points have been found, 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.

5.1.3. Orientation Ambiguity

Identification of vanishing points in a Manhattan world provides a direct estimation of the camera orientation with respect to the canonical frame of reference.

Care must be taken, however, to correctly assign axis labels when the camera is moving, lest the direction that was assigned to the X axis, say, in the current frame may be identified as the Y axis at the next frame. We solve this gauge ambiguity by simply labeling with the same name the pairs of axes (one per view in a pair of consecutive views) that (with respect to the camera's reference frame) have the smallest angular difference.

5.1.4. Computational Cost

The original sequences used for our experiments were 1356 frames long. We divided them into 6 chunks of 240 frames each, with an overlap of 9 frames each. As mentioned earlier, all frames are used for line chain computation, but only one frame out of three was used for SFM (resulting in 80 views per chunk, with an overlap of 3 views). Multi-view characteristic lines clustering and global motion computation was performed over each chunk of 80 views. Consecutive chunks were then aligned by moving the locations computed for the second chunk by a common vector aligning the centers of the position computed for the overlapping views in the two chunks to a common location (after which, the locations of each overlapping view computed for the two chunks were averaged together).

On a Mac Air 1.7 GHz Intel Core i5 with 4GB RAM, each chunk of 240 frames (80 views used for SFM) took on average 157 second to process, divided as follows:

- Line chain computation: 26.2 s (on 240 frames – 0.11 s/frame)
- Multi-view characteristic lines clustering: 38.9 s (on 80 views – 0.49 s/view)
- Global motion estimation: 86.22 s (1.1 s/view)
- Triangulation + Bundle adjustment: 3.9 s (0.05 s/view)
- Manhattan structure computation: 1.5 s

5.2. Results: Qualitative

We present results of reconstructed trajectories and structure over three videos in Figs. 12–14. A bird-eye view of the reconstructed trajectory and structure is displayed over a floor plan of the building, which has been rescaled isotropically (by hand) to visually match the reconstructed structure. We show reconstructions with two different variations of our algorithm: 2-view characteristic lines clustering [3], followed by global motion estimation (Sec. 4.3) with additional *scale normalization* and Manhattan layout computation (sec. 4.4) (c); same as in (c), but with multi-view clustering (Sec. 4.2) (d). The *scale normalization* procedure attempts to enforce a common reconstruction scale even when, due to sporadic paucity of features, different segments are reconstructed with different scales. A fixed distance between camera locations at two views with 8-view distance (constant velocity model) is imposed by scanning the sequence of camera locations after global motion estimation, and moving each camera location $\vec{c}_i$ by a proper amount along the line joining $\vec{c}_{i-1}$ with $\vec{c}_i$. This is clearly a very crude approach; a more complete dynamical model could be used [50], possibly also using inertial measurements, when available [51]. In each figure, we also show the trajectories computed via SFM from point features, using two open source software packeges: VisualSFM [52], by Changchang Wu; and the Theia multiview geometry library (TheiaSFM) [53], by Chris Sweeney. In our experiments with TheiaSFM, we obtained our best results using its global SFM implementation with brute force point feature matching and regular SIFT descriptors. TheiaSFM's cascade hashing matching implementation and Root-SIFT didn't work well with our indoor video sequence. All other parameters were left to their default values.

Sequence 1 (Fig. 12) is characterized by a S-shaped trajectory. VisualSFM reconstructs the trajectory fairly well up to a point, after which no camera location estimates are returned. TheiaSFM, the 2-view characteristic lines clustering, and the multi-view clustering all produce fairly good results.

We also show the reconstructed Manhattan structure, as shown by red rectangles, obtained from the vertical lines associated with each planar structure as per Sec. 4.4.

Sequence 2 (Fig. 13) was taken while walking in a U-shaped trajectory. Again, VisualSFM is unable to reconstruct the trajectory beyond a certain point, while TheiaSFM reconstructs the whole trajectory. However, the structure reconstructed by TheiaSFM appears expanded at the bottom of the image, while conforming well to the actual scene structure in the top part of the image. This is likely an artifact resulting from inconsistent reconstruction scale, as explained above. 2-view and multi-view clustering (with scale normalization) produce satisfactory trajectories and structures. Note that the sequence was taken by one of the authors walking at constant speed, thus approximately satisfying the constant velocity assumption at the basis of our scale normalization algorithm.

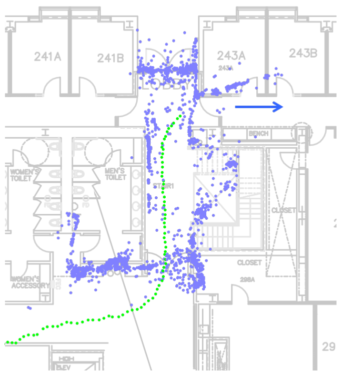
Sequence 3 (Fig. 14) was taken while walking in a loop. VisualSFM is unable to reconstruct the trajectory. TheiaSFM produces four disconnected trajectory segments, each of which could be moved to match a segment of the floor plane, but it fails to reconstruct the complete trajectory. This is probably due to a momentary paucity of point features at the four corners of the trajectory.

2-view clustering reconstructs the trajectory fairly well up to a point. Multi-view reconstruction produces a satisfactory trajectory with loop closure, and an acceptable reconstruction of the main visible surfaces.

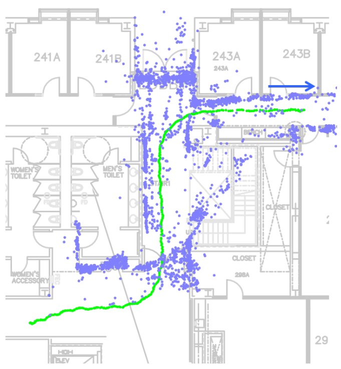
5.3. Results: Quantitative

In order to provide a quantitative evaluation of our 2-view and multi-view algorithms, we followed a procedure originally proposed in [3]. Note that we don't have ground truth motion or structure information, and thus cannot directly assess the quality of reconstruction. Instead, we evaluate our results based on whether, given any three coplanar 3-D lines, their reconstructions appear to be coplanar. Since line coplanarity is easy to assess from images, this method results in a convenient indirect way to validate the 3-D reconstruction.

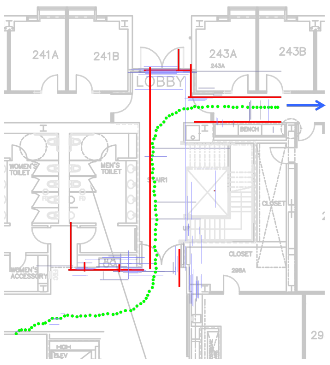
We first sampled 31 key frames uniformly out of the 1356 frames of Sequence 3 (Fig. 14). Then, for each key frame, we manually enumerated all planes visible in the image and assigned each line segment to one or two of those planes containing it. All possible line triplets in the key frame were labeled as coplanar or non-coplanar based on their manually assigned plane labels. Given the set of reconstructed 3-D lines, we consider all line triplets, and test each triplet for coplanarity using Plücker matrices [54]. Specifically, 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 $\mathcal{L}_2$ [54]. In practice, we declare the three reconstructed lines to be coplanar if the norm of $\mathbf{L}_1\mathbf{L}_2^*\mathbf{L}_3$ is smaller than a certain threshold δ_P (where we first normalize each Plücker matrix to unit norm). By validating the estimated coplanarity with the ground-truth values for all line triplets, we obtain a precision-recall pair. We also vary the threshold δ_P as well as a parameter used in mean-shift clustering (specifically, the number of nearby characteristic lines used to update the cluster center in every iteration) to create the precision-recall curves shown in Fig. 15. In this figure we compare the results of our 2-view and multi-view CL clustering, as well as those obtained with an implementation that integrates Micusik and Wildenauer's line-based method [55] in our SFM pipeline, as explained in the following. We estimate three camera poses using RANSAC based on the linear constraints on relative camera translations between three views as proposed in [55], using the relative rotations and line matches produced by our algorithm. Although 5 lines matches are sufficient for this algorithm [55], we used 6 matches to obtain better result. Unlike our SFM pipeline, which only analyzes view pairs, the algorithm of [55] considers all triplets of views. This results in a large number of relative camera translation estimates (on the order of N^3, where N is the number of views), making this



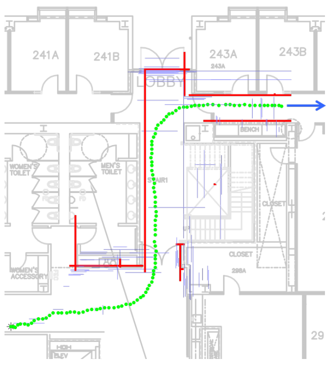
(a)



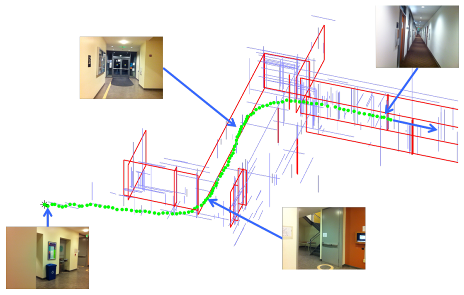
(b)



(c)



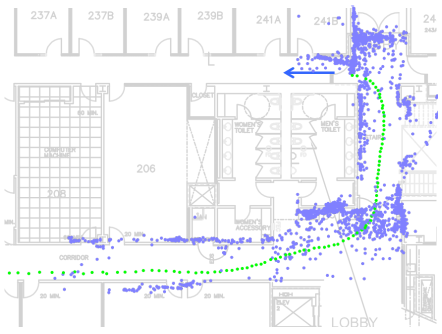
(d)



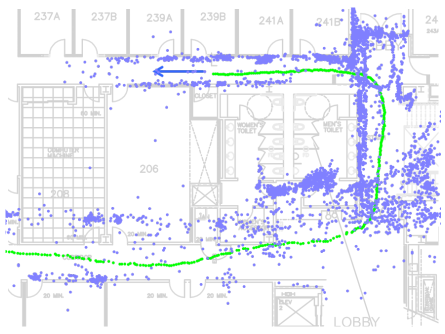
(e)

Figure 12: Results for Sequence 1. Camera trajectory, shown with green dots; feature points (a-b) or lines (c-e), shown with blue dots/lines; and Manhattan structure (c-e), shown with red lines/retangles. (a): VisualSFM [52]; (b) TheiaSFM [53]; (c) 2-view characteristic lines clustering [3]; (d-e): Multi-view clustering (Sec. 4.2).

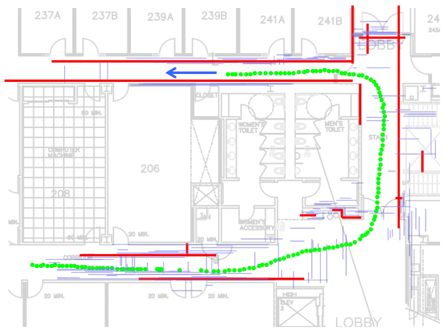
problem intractable once these estimates enter the global camera translation step (Sec. 4.3). In order to reduce the size of the LUD problem (10), we visit all sets



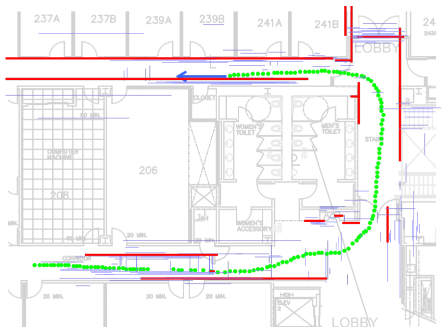
(a)



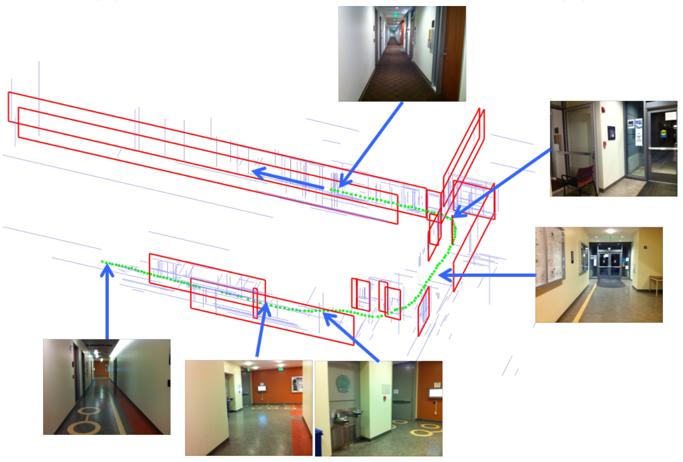
(b)



(c)



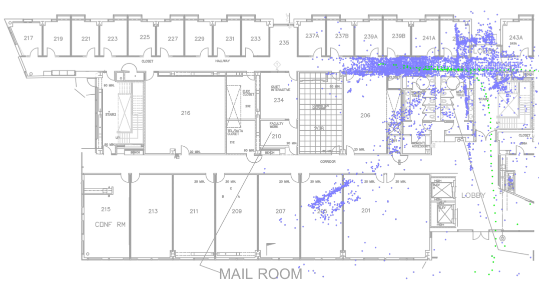
(d)



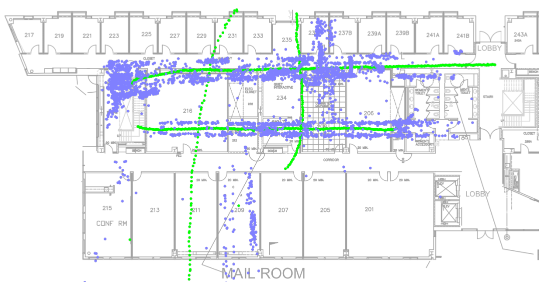
(e)

Figure 13: Results for Sequence 2 (see caption of Fig. 12)

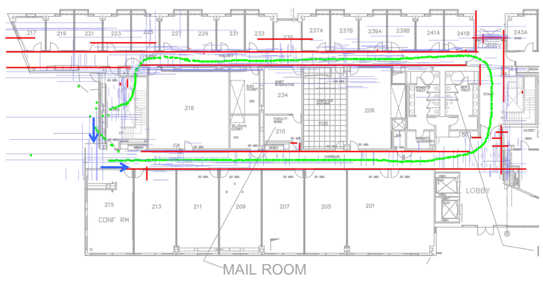
of view triplets with a common view, and only retain the triplet with the smallest reprojection error of its reconstructed lines.



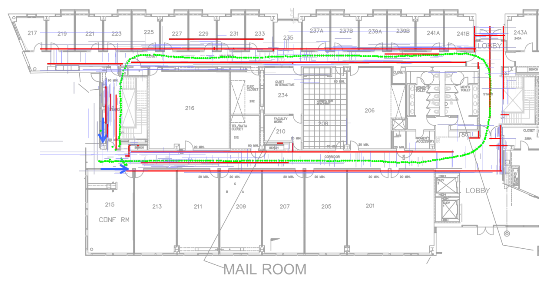
(a)



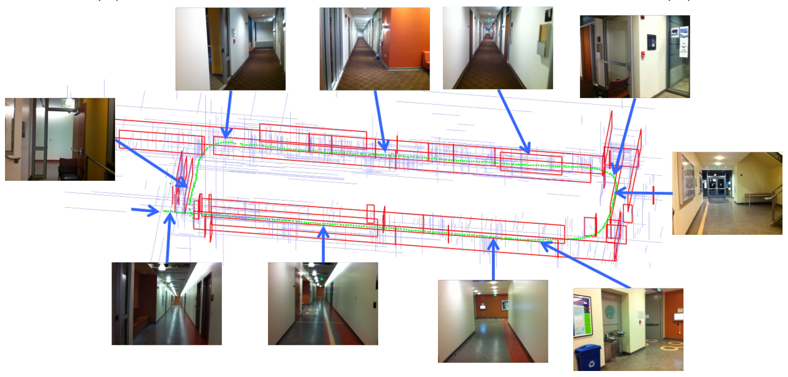
(b)



(c)



(d)



(e)

Figure 14: Results for Sequence 3 (see caption of Fig. 12)

Fig. 15 shows that the multiview CL clustering method produces best overall results in terms of estimated line triplet coplanarity. To get some insight into the difference of this method vis-a-vis the algorithm of Micusik and Wildenauer [55], we show in Fig. 16 a bird-eye and a side view of the trajectories and of the structure as reconstructed by the two methods. While the trajectories look similar from the bird-eye view, the side view shows that the method of [55] seems to produce incorrect vertical values for the camera pose.

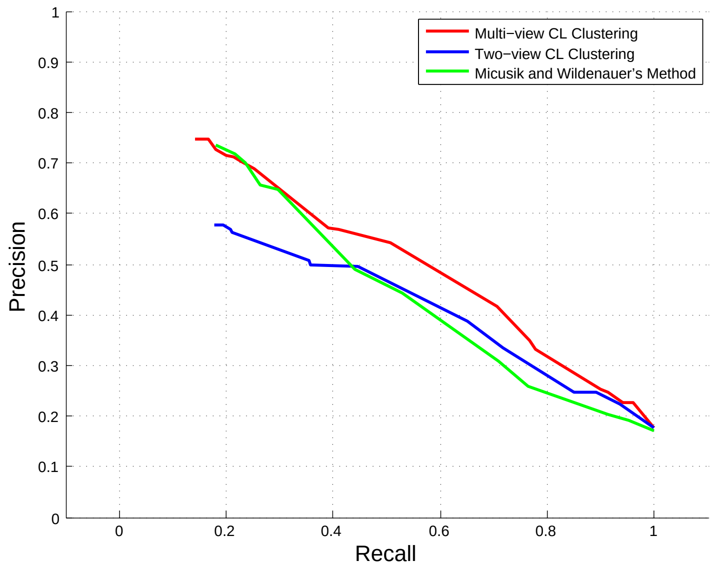


Figure 15: Precision/recall curves for the algorithms (see Sec. 5.3).

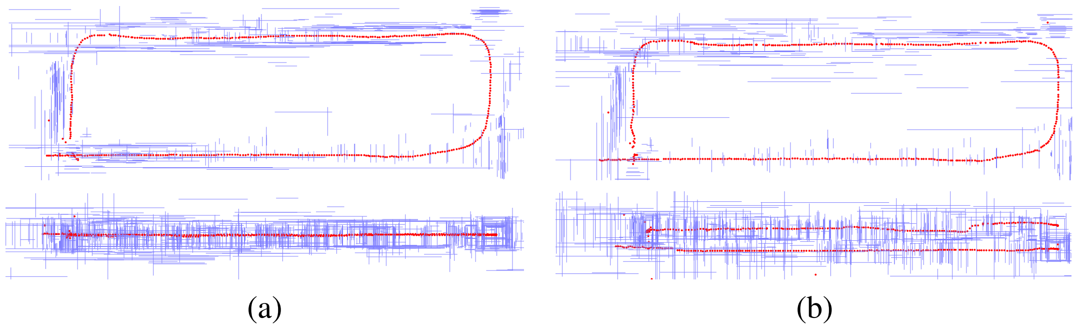


Figure 16: Bird-eye view (top) and side view (bottom) of the trajectory and structure reconstructed by our multi-view CL clustering (a) and by our implementation of Micusik and Wildenauer's linear constraints [55] (b); see Sec. 5.3.

6. Conclusions

We have described a SFM system that is based solely on line matching, under the assumption that the environment layout complies with the Manhattan world hypothesis. This system is based on the idea of "characteristic lines", which can be seen as an invariant of two views of a parallel line pair laying on a plane with known orientation. The characteristic lines algorithm enables segmentation of planar patches from visible lines, and thus reconstruction of motion and structure. We have shown how to extend the characteristic lines algorithm to the multi-view case, resulting in a more robust planar segmentation. Clusters of characteristic lines indicate the presence of a planar surface, and provide a measurement of pairwise camera translation, normalized with the distance to the plane. This information is fed to a modified LUD algorithm for globally consistent motion estimation. The structure of the environment is then computed with an algorithm that leverages the strong Manhattan world geometric constraints.

While this paper focused solely on line matching, it should be clear that maximum robustness and accuracy would be obtained by considering both point *and* line features. We are currently exploring different approaches for combining these heterogenous features in an unified framework. We are also looking at ways to expand our characteristic line method to weak Manhattan layouts [56], which have only horizontal and vertical surfaces, but the vertical surfaces can be oriented in any direction. Another intriguing research direction would be to combine the type of knowledge that can be acquired from individual images (under strong geometric assumptions) with information from multiple views. Given that in many cases the line configuration in a single image already suffice for indoor layout reconstruction [23, 24, 25], this prior information could be very beneficial for SFM algorithms.

Appendix

In this Appendix, we prove that if a line $\mathcal{L}$ lying on a plane with normal $\vec{n}$ is matched across two views separated by the baseline vector $\vec{t}$, then (4)

$$\langle \vec{t}/d, \vec{u}_2 \rangle = \frac{\sin \angle \vec{u}_1, \vec{u}_2}{\sin \angle \vec{u}_1, \vec{n}} \tag{14}$$

where d is the distance of the plane from the first camera, and $\vec{u}_1$, $\vec{u}_2$ are the lever vectors induced by the line $\mathcal{L}$ on the two cameras (as defined in Sec. 3.1).

From (1) one derives

$$\mathbf{u}_1 \times H_c^T \mathbf{u}_2 = 0 \tag{15}$$

Combining (15) with (2), one obtains

$$(R^T \mathbf{u}_2) \times \mathbf{u}_1 = \mathbf{u}_1 \times \mathbf{n} \mathbf{u}_2^T \mathbf{t}/d \tag{16}$$

hence (noting that $\mathbf{u}_1$, $\mathbf{u}_2$ and $\mathbf{n}$ are unit vectors)

$$\mathbf{u}_2^T \mathbf{t}/d = \frac{\|(R^T \mathbf{u}_2) \times \mathbf{u}_1\|}{\|\mathbf{u}_1 \times \mathbf{n}\|} \cdot ((R^T \mathbf{u}_2) \times \mathbf{u}_1)^T (\mathbf{u}_1 \times \mathbf{n}) \tag{17}$$

which is identical to (14).

References

[1] C. G. Harris, J. Pike, 3D positional integration from image sequences, Image and Vision Computing 6 (2) (1988) 87–90.

[2] R. Hartley, A. Zisserman, Multiple view geometry in computer vision, Cambridge Univ Press, 2000.

[3] C. Kim, R. Manduchi, Planar structures from line correspondences in a manhattan world, in: Computer Vision–ACCV 2014, Springer, 2015, pp. 509–524.

[4] O. Ozyesil, A. Singer, Robust camera location estimation by convex programming, arXiv preprint arXiv:1412.0165.

[5] A. W. Fitzgibbon, A. Zisserman, Automatic camera recovery for closed or open image sequences, in: Computer Vision—ECCV'98, Springer, 1998, pp. 311–326.

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

[7] J. Weng, T. S. Huang, N. Ahuja, Motion and structure from line correspondences; closed-form solution, uniqueness, and optimization, IEEE Transactions on Pattern Analysis and Machine Intelligence (3) (1992) 318–336.

[8] M. E. Spetsakis, J. Y. Aloimonos, Structure from motion using line correspondences, International Journal of Computer Vision 4 (3) (1990) 171–183.

[9] R. Hartley, et al., Projective reconstruction from line correspondences, in: Proc. IEEE Computer Vision and Pattern Recognition, IEEE, 1994, pp. 903–907.

[10] A. Tang, T. Ng, Y. Hung, C. H. Leung, Projective reconstruction from line-correspondences in multiple uncalibrated images, Pattern Recognition 39 (5) (2006) 889–896.

[11] N. Navab, O. D. Faugeras, The critical sets of lines for camera displacement estimation: A mixed euclidean-projective and constructive approach, International Journal of Computer Vision 23 (1) (1997) 17–44.

[12] J. L. Crowley, P. Stelmaszyk, T. Skordas, P. Puget, Measurement and integration of 3-d structures by tracking edge lines, International Journal of Computer Vision 8 (1) (1992) 29–52.

[13] C. J. Taylor, D. J. Kriegman, Structure and motion from line segments in multiple images, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (11) (1995) 1021–1032.

[14] Z. Zhang, Estimating motion and structure from correspondences of line segments between two perspective images, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (12) (1995) 1129–1139.

[15] J. Košecká, W. Zhang, Extraction, matching, and pose recovery based on dominant rectangular structures, Computer Vision and Image Understanding 100 (3) (2005) 274–293.

[16] A. Elqursh, A. Elgammal, Line-based relative pose estimation, in: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, 2011, pp. 3049–3056. doi:10.1109/CVPR.2011.5995512.

[17] E. Vincent, R. Laganiére, 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, pp. 182–187.

[18] E. Montijano, C. Sagues, Position-based navigation using multiple homographies, in: Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on, IEEE, 2008, pp. 994–1001.

[19] Z. Zhou, H. Jin, Y. Ma, Robust plane-based structure from motion, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, 2012, pp. 1482–1489.

[20] Z. Zhou, H. Jin, Y. Ma, Plane-based content-preserving warps for video stabilization, in: Computer Vision and Pattern Recognition, 2013. CVPR 2013., IEEE, 2013.

[21] R. Toldo, A. Fusiello, Robust multiple structures estimation with j-linkage, in: Computer Vision–ECCV 2008, Springer, 2008, pp. 537–547.

[22] S. N. Sinha, D. Steedly, R. Szeliski, Piecewise planar stereo for image-based rendering., in: ICCV, Citeseer, 2009, pp. 1881–1888.

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

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

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

[26] D. C. Lee, M. Hebert, T. Kanade, Geometric reasoning for single image structure recovery, in: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 2136–2143.

[27] A. Flint, D. Murray, I. Reid, Manhattan scene understanding using monocular, stereo, and 3d features, in: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, 2011, pp. 2228–2235.

[28] S. Ramalingam, J. K. Pillai, A. Jain, Y. Taguchi, Manhattan junction catalogue for spatial reasoning of indoor scenes, in: Computer Vision and Pattern Recognition, 2013. CVPR 2013., IEEE, 2013.

[29] G. Tsai, B. Kuipers, 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, pp. 4695–4701.

[30] J. M. Coughlan, A. L. Yuille, Manhattan world: Compass direction from a single image by bayesian inference, in: Proc. IEEE International Conference on Computer Vision, Vol. 2, IEEE, 1999, pp. 941–947.

[31] J. Košecká, W. Zhang, Video compass, in: Computer Vision—ECCV 2002, Springer, 2006, pp. 476–490.

[32] M. Hwangbo, T. Kanade, Visual-inertial uav attitude estimation using urban scene regularities, in: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE, 2011, pp. 2451–2458.

[33] D. Comaniciu, P. Meer, Mean shift: A robust approach toward feature space analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (5) (2002) 603–619.

[34] Z. Wang, F. Wu, Z. Hu, Msld: A robust descriptor for line matching, Pattern Recognition 42 (5) (2009) 941–953.

[35] H. Bay, V. Ferrari, L. Van Gool, Wide-baseline stereo matching with line segments, in: Proc. Computer Vision and Pattern Recognition (CVPR 2005), Vol. 1, IEEE, 2005, pp. 329–336.

[36] J. Van De Weijer, C. Schmid, J. Verbeek, D. Larlus, Learning color names for real-world applications, IEEE Transactions on Image Processing 18 (7) (2009) 1512–1523.

[37] B. Fan, F. Wu, Z. Hu, Line matching leveraged by point correspondences, in: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE, 2010, pp. 390–397.

[38] S. Ramalingam, M. Antunes, D. Snow, G. H. Lee, S. Pillai, Line-sweep: Cross-ratio for wide-baseline matching and 3d reconstruction, in: Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on, IEEE, 2015, pp. 1238–1246.

[39] D. Marr, T. Poggio, A theory of human stereo vision., Tech. rep., DTIC Document (1977).

[40] N. Cornelis, K. Cornelis, L. Van Gool, Fast compact city modeling for navigation pre-visualization, in: Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, Vol. 2, 2006, pp. 1339–1344. doi:10.1109/CVPR.2006.118.

[41] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International journal of computer vision 60 (2) (2004) 91–110.

[42] R. Tron, R. Vidal, Distributed image-based 3-d localization of camera sensor networks, in: Proceedings of the 48th IEEE Conference on Decision and Contro, IEEE, 2009, pp. 901–908.

[43] O. Özyeşil, A. Singer, R. Basri, Stable camera motion estimation using convex programming, SIAM Journal on Imaging Sciences 8 (2) (2015) 1220–1262.

[44] R. Kennedy, K. Daniilidis, O. Naroditsky, C. J. Taylor, Identifying maximal rigid components in bearing-based localization, in: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, IEEE, 2012, pp. 194–201.

[45] I. Daubechies, R. DeVore, M. Fornasier, C. S. Güntürk, Iteratively reweighted least squares minimization for sparse recovery, Communications on Pure and Applied Mathematics 63 (1) (2010) 1–38.

[46] T. Zhang, G. Lerman, A novel m-estimator for robust pca, The Journal of Machine Learning Research 15 (1) (2014) 749–808.

[47] S. Agarwal, K. Mierle, Others, Ceres solver, http://ceres-solver.org.

[48] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, G. Randall, LSD: a Line Segment Detector, Image Processing On Line 2012. doi:10.5201/ipol.2012.gjmr-lsd.

[49] J. M. R. S. Tavares, A. J. M. N. Padilha, A new approach for merging edge line segments, RecPad95.

[50] A. Chiuso, P. Favaro, H. Jin, S. Soatto, Structure from motion causally integrated over time, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (4) (2002) 523–535.

[51] E. S. Jones, S. Soatto, Visual-inertial navigation, mapping and localization: A scalable real-time causal approach, The International Journal of Robotics Research 30 (4) (2011) 407–430.

[52] C. Wu, VisualSFM, http://ccwu.me/vsfm/ (last checked: 6/15/2014).

[53] C. Sweeney, Theia multiview geometry library: Tutorial & reference.

[54] J. I. Ronda, A. Valdés, G. Gallego, Line geometry and camera autocalibration, Journal of Mathematical Imaging and Vision 32 (2) (2008) 193–214.

[55] B. Micusik, H. Wildenauer, Structure from motion with line segments under relaxed endpoint constraints, in: 3D Vision (3DV), 2014 2nd International Conference on, Vol. 1, IEEE, 2014, pp. 13–19.

[56] O. Saurer, F. Fraundorfer, M. Pollefeys, Homography based visual odometry with known vertical direction and weak manhattan world assumption, in: Proc. IEEE/IROS Workshop on Visual Control of Mobile Robots, 2012.