UNIVERSITY OF CALIFORNIA
SANTA CRUZ

INDOOR MANHATTAN SPATIAL LAYOUT RECOVERY FROM MONOCULAR VIDEOS

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL ENGINEERING

by

Chelhwon Kim

December 2016

The Dissertation of Chelhwon Kim
is approved:

Professor Michael Isaacson, Chair

Professor Gabriel Hugh Elkaim

Professor Roberto Manduchi

Tyrus Miller
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Indoor Manhattan Spatial Layout Recovery

from Monocular Videos

by

Chelhwon Kim

Traditional Structure-from-Motion (SfM) 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.

In this thesis, I present a novel approach 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. The proposed 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. Finding coplanar sets of lines becomes a problem of clustering characteristic lines (CL), which can be accomplished using a modified mean shift procedure. The CL algorithm is fast and robust, and computationally light and produces good results in real world situations.

The CL algorithm is extended to the case of multiple views for the analysis of videos from a monocular camera. I present a novel 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 "least unsquared deviations" (LUD) algorithm that computes the global camera motion. Finally, I introduce a new technique for planar fitting of the reconstructed lines that makes explicit use of the Manhattan world geometry.

To my wife Youra, my daughter Seongju, and my parents for their love, support, and sacrifices.

Acknowledgments

I would like to express my sincere gratitude to my advisor Prof. Roberto Manduchi for the continuous support of my Ph.D study and related research, for his patience guidance, commitment to my growth and development. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study. Besides my advisor, I would like to thank Prof. Michael Isaacson and Prof. Gabriel Elkaim for serving on my committee and reviewing this thesis. I also want to thank all my fellow labmates from Computer Vision Lab.

I would like to thank my family, especially my parents, for all their support during all my years of studying. I would not have gotten that far without you. Last, but definitely not least, I want to thank my wife Youra and my daughter Seongju for their love, and for always being there for me. I love you, and I always will.

Chapter 1

Introduction

Structure from motion (SfM) has a long history in computer vision [31, 33]. 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. In recent years, several point feature-based SfM open source software packages [67, 64, 52, 88, 72] have been published and provide efficient and reliable implementations of the essential algorithms required in the SfM pipeline. However, in many cases, the reconstructed scene geometry of an indoor environment by these point feature-based SfM is represented as a non-uniformly

distributed sparse 3D point cloud due to the low density of detectable points in the textureless walls, and hence it is difficult to get a meaningful 3D model of the scene (e.g. planar surfaces). Fig. 1.1 shows one of the point-feature based SfM's result [72] and our line-feature based SfM result [42] for two views of an indoor scene. As we can see in this example, indoor environments are often characterized by a relatively low density of point features (e.g. corners), and matching across view is challenging, whereas lines are typically visible (e.g. plane intersections) and have better potential in representing a semantically meaningful 3D scene model. Point-based structure from motion often gives unsatisfactory results in these cases.

If point features may be relatively scarce, line features are almost invariably present in these environments, 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

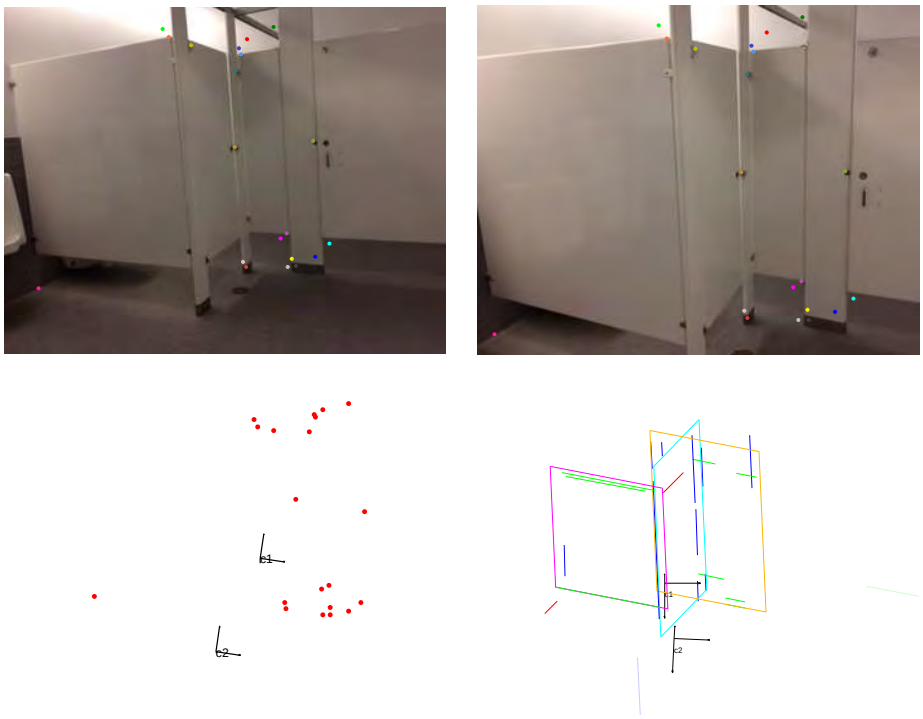


Figure 1.1: Top row: Two different views of an indoor environment and detected point matches by [72]. The indoor environment is characterized by a relatively low density of point features (e.g. corners), and matching across views may be challenging. However, lines are typically visible (e.g. plane intersections), oriented along the three canonical axes. Bottom left: Point-based structure from motion gives unsatisfactory results in these cases [72]. Bottom right: Line-based SfM result by our method [42]. Planar patches are detected by grouping coplanar lines in a Manhattan world geometry (described in Ch. 2).

that the lines are not all mutually parallel.

In this thesis, we present a novel approach 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. We first introduce 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 [42]. We then 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[54] 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 tech-

nique for planar fitting of the reconstructed lines that makes explicit use of the Manhattan world geometry.

1.1 Previous Work

In this section, we give an overview of the standard SfM pipeline and a survey of line matching, line-based SfM, and alternative approaches of scene reconstruction.

The standard SfM pipeline consists of the following steps to recover the camera motion and the 3D structure of the scene from 2D images. A collection of 2D images of the same scene from different viewpoints are acquired and point (or line) feature correspondences are identified across the images. Generally, the standard point feature based SfM requires at least two images with at least 7 or 8 point feature matches to compute the 3x3 fundamental matrix that encapsulates the camera's internal parameters and relative pose in the epipolar geometric constraint [33, 47]. Line feature based SfM requires at least three images to retrieve the camera poses using trifocal tensor with at least 13 triplets of line correspondences [33, 25, 4, 86, 70, 32, 74]. By imposing geometric constraints on the structure of the scene or the camera motion, however, only two images can be considered with less numbers of line correspondences [23, 42].

To identify point feature correspondences from different images, numerous robust appearance-based point descriptors have been used in a number of modern SfM pipelines [88, 72] such as SIFT [46], SURF [6], ORB [61] which are widely

known as robust rotation and scale invariant features, to just name a few, while only a few methods are reported in the literatures for line segment descriptors due to its inherent difficulties such as inaccurate locations of end points of line segment, fragmentation of a line on the same edge etc. We will review several line descriptors and matching methods in the next section.

Once a sufficient number of correspondences between two views has been identified the fundamental matrix (or the essential matrix if the camera intrinsic parameters are given) is estimated from where the relative camera orientation and translation are extracted [33, 47]. By triangulating the matched features given the camera motions, the 3D structure of the scene is computed. Here, either incremental SfM approach [33] or global SfM approach [87, 73] can be considered to reconstruct the global camera locations and orientations. Incremental methods [2, 57, 68, 56] build the 3D model by iteratively growing an initial model computed from first several images followed by the bundle adjustment. Incremental SfMs could be more accurate than the global SfM for small size of images because of its extensive use of bundle adjustment for every new images entered in the pipeline but it is typically slow and subject to large drifting errors on the final reconstruction result for large size of image set. The global SfM approaches [87, 73, 54, 55, 38] solve for global camera locations and orientations simultaneously from all available pairwise relative camera motions. The main challenge of this approach is to compute the global camera locations since the essential ma-

trix encodes only the relative direction of two views [18]. Generally, the global approach can be considered in case of reconstruction from unordered irregular collection of images such as internet photos [68]. Sequential SfM approaches may be considered for relatively ordered and small size of image collections such as sequential video frames. Finally, the bundle adjustment [79, 68, 27] is performed to refine the camera motion and the reconstructed structure by minimizing the reprojection error.

1.1.1 Line Matching

While the use of appearance measures based on SIFT-like descriptors has been very successful in disambiguating point feature matches, line matching is still a challenging task due to inaccurate locations of end points of line segment, fragmentation of a line on the same edge, and any geometric constraint such as the epipolar constraint popularly used in point matching is not available [24, 85]. In this section, we give a brief survey of line matching approaches.

In general, straight line segments are extracted from an image by detecting edge pixels (e.g. Canny edge detector [11]) and building chains of connected edge pixels. A split step is performed at a point with high curvature and straight lines are fitted to the connected edge pixels [7, 40, 5]. Some of recent line detectors [22, 10, 30] exploit the Helmholtz principle to control the false positive rates and produce robust and accurate line segments. Once the straight line segments are

detected from two different images, visually similar line pairs are identified by any given appearance based descriptor. Wang et al. [85] proposed a mean-standard deviation line descriptor (MSLD), which defines pixel support regions for each point in the line, from which SIFT-like descriptor (edge orientation histograms) are generated. This descriptor, however, may fail in the lack of textures in the local neighborhood region of the line segment. Zhang and Koch [89] also designed the SIFT-like line descriptor called Line Band Descriptor (LBD) similar to MSLD except that the support region is divided into a set of bands where each band is parallel with the line segment. In order to improve the matching performance, the multi-scale line extraction approach is adopted by detecting lines and computing the descriptors in multi scale space.

Textural information alone cannot guarantee robust line matching due to the weak appearance distinctiveness of line segments. Several approaches have attempted to exploit more geometric information by nearby point/line feature correspondences or their topological layout to reduce the number of false line matches [24, 58, 5, 63]. Bay et al. [5] exploits a topological relation of triplet or pair of line matches to discriminate the correct matches from the false matches. A color histogram based matching is performed to find a set of tentative line matches, and then the spatial arrangement of line segments and regions in two views is used to remove mismatched line pairs. Finally, new matches consistent with the topological structure of the current ones are added iteratively to increase the correct

line match number. The authors also attempted to estimate the epipolar geometry from the line matches. The line matches are grouped by a RANSAC-like algorithm based on the homography of coplanar line correspondences and then the epipolar geometry is estimated by using the intersecting points of every set of potentially coplanar line segments as point matches and using them to estimate the fundamental matrix. The result shows that the proposed matching method finds a good number of line matches on untextured scenes. Schmid and Zisserman [63] also exploits the homography, which is characterized by a single scalar parameter given a line pair and the epipolar geometry. The authors compute a cross-correlation between a rectangular strip on one side of the line segment in one image and its projected region by the homography in the other image. The homography is determined by searching for its scalar parameter given the prior knowledge of the epipolar geometry such that the cross-correlation is maximized. Finally, the maximum cross-correlation determines whether the line pair is correct or not. This approach, however, still requires rich textures in the nearby region of the line segments and the epipolar geometry. Fan et al. [24] invented a similarity measure for line matching based on the ratio of distances from two image points to a line segment, which is invariant under the view point change when their associated two points and a line in 3D space are coplanar. This approach also requires rich textures nearby the line segments and coplanar point correspondences nearby the given line pair. The main drawback of these approaches is the requirements

of the epipolar geometry, the point correspondences, and the rich textures in the local region of the line segment.

1.1.2 SfM based on Line Feature

The standard approach to recovering scene structure and camera pose from multiple views is based on point feature matches across views [33]. 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 [33, 25, 4, 86, 70, 32, 74]. 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 [33]. 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 [4, 53]. Kalman filter approaches for reconstruction from multiple views have also been proposed [17, 77].

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 [91]) or geometric priors (Manhattan world). Košecka and Zhang [43] presented a method to ex-

tract 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 [23] 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.

Hofer et al. [35] also presented a method that generates accurate 3D models of a scene using straight line segments as features with low computational costs. The proposed method establishes potential line matches based on the prior knowledge of the epipolar geometry constraint between images by executing any existing point feature-based SfM pipeline. The more plausible line matches are selected further by clustering line segments based on their spatial proximity in 3D space. By optimizing the reconstructed 3D lines and camera poses by the bundle adjustment, the method generates accurate camera poses and 3D lines. However, this approach requires the prior knowledge of the epipolar geometry.

Micusik and Wildenauer [50] presented a purely line based SfM pipeline to reconstruct a wiry 3D models of indoor scenery. The authors first decoupled the SfM problem into rotation and translation estimations, and proposed a linear solver for relative translation estimation given the relative rotations and five line

matches in three views. The final camera poses are refined by the bundle adjustment, in which the reconstructed 3D lines' end points were taken into account in the cost function to be minimized. The proposed method was shown to work on long indoor sequences.

1.1.3 3D layout reconstruction from single image

A more recent research direction looks to recover the spatial layout of an indoor scene from a single image [36, 34, 21]. Lee et al. [45] 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. Recently, rather than relying on the hand crafted features based on line segments and the geometric reasoning, Dasgupta et al. [19] presented a method that uses a fully convolutional neural network (FCNN) for generating the orientation map and demonstrated robust layout estimation results with challenging indoor scenes. Flint et al. [26] 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. [59] 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. [81] 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.

1.2 Organization of this thesis

This thesis is organized as follows. We describe the characteristic lines algorithm (CL) for motion and structure reconstruction from two views in Ch. 2. This section also introduces a new “line ordering constraint" method for robust line matching. Ch. 3 describes the extension of the CL algorithm to multiple views; it includes our technique for line chain construction (3.1), multi-view characteristic lines clustering (3.2), motion reconstruction via modified LUD algorithm (3.3), and Manhattan structure computation (3.4). Ch. 4 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. 5 has the conclusions.

Chapter 2

Structure from Lines from Two Views

In this chapter, we introduce 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 [44, 37, 23], 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

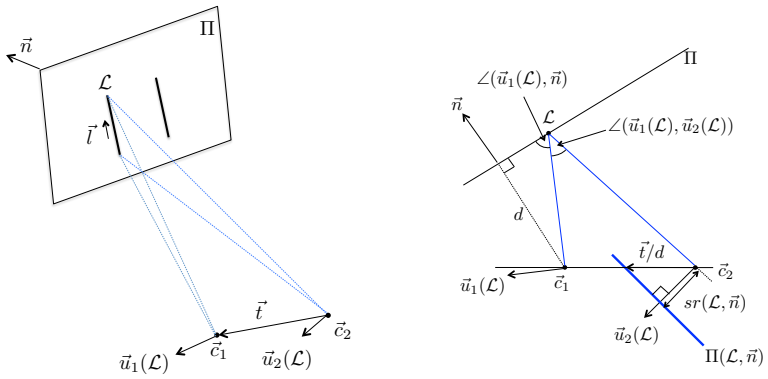


Figure 2.1: Left: The two camera centers $\vec{c}_1$, $\vec{c}_2$ and the lever vectors $\vec{u}_1(\mathcal{L})$, $\vec{u}_2(\mathcal{L})$ for line $\mathcal{L}$. Right: Line $\mathcal{L}$ lies on the plane $\Pi \equiv (\vec{n}, d)$ (both line and plane orthogonal to this page). The thick blue line is the trace of the $\vec{n}$-characteristic plane $\Pi(\mathcal{L}, \vec{n})$ (also orthogonal to the page).

camera pose. 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.

2.1 Notation and Basic Concepts

By *Manhattan world* [16] we mean an environment comprising only planar surfaces, each of which is oriented along one of three *canonical* mutually orthogonal 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 (possibly at its edge) and is oriented along one of the three canonical vectors.

[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.

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 [44]. 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 [37]. 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. 4.1.1 and 2.4.

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). 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* [91] (the plane containing $\mathcal{L}$ and the optical center of the camera; see Fig. 2.1, 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:

$$\mathbf{u}_1 = H_c^T\mathbf{u}_2 \tag{2.1}$$

where $H_c = K_2^{-1}HK_1$ is the *calibrated homography matrix* induced by the plane, which can be decomposed [33] as

$$H_c = R + \mathbf{t}\mathbf{n}^T/d \tag{2.2}$$

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.

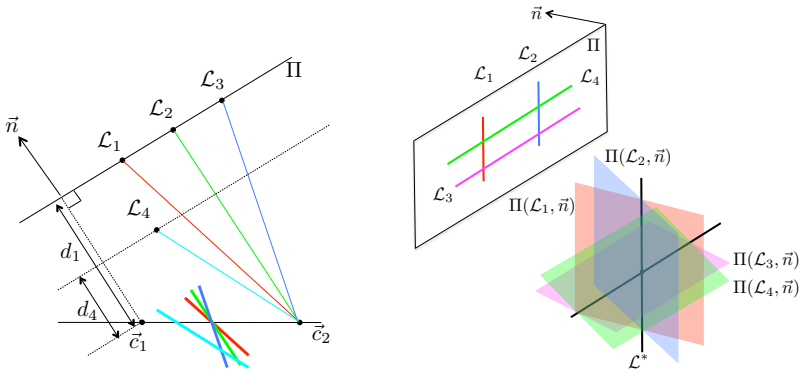


Figure 2.2: Left: Lines $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$ (orthogonal to this page) are $\vec{n}$-coplanar. Their associated $\vec{n}$-characteristic planes all intersect at a characteristic line through the baseline (also orthogonal to this page). They also individually intersect 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.

2.2 Motion from Lines on a Plane with Known Orientation

By combining (2.1) and (2.2), one sees that the lever vectors associated with the same line $\mathcal{L}$ seen by two cameras are related as by

$$\mathbf{u}_1 = R^T \mathbf{u}_2 + \mathbf{n}\mathbf{u}_2^T \mathbf{t}/d \tag{2.3}$$

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 [42] (see the Appendix for a proof):

$$\langle \vec{t}/d, \vec{u}_2 \rangle = \frac{\sin \angle(\vec{u}_1, \vec{u}_2}{\sin \angle(\vec{u}_1, \vec{n}} \tag{2.4}$$

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* $\Pi(\mathcal{L}, \vec{n})$ [42] (see Fig. 2.1, right panel). Note that $\Pi(\mathcal{L}, \vec{n})$ can be easily computed using (2.4), 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* $\mathcal{L}^*$ [42]. 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. 2.2, 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 [42], and briefly summarized in the following.

2.3 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 [42] 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. This observation suggests the following clustering algorithm:

1. For each pair of parallel lines oriented along $\vec{n}_i$, 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 as belonging to the same plane ($\vec{n}$, d_m) for the m-th cluster center.

Each such cluster indicates the presence of a planar surface oriented as $\vec{n}$. Clustering in the second step can be performed (for example, using mean shift [14]) on the traces of the characteristic lines on the plane oriented as $\vec{n}_i$ and containing the second camera's optical center (Left bottom of Fig. 2.3). The m-th cluster center represents an estimate of the projection of $\vec{t}/d_m$ on this plane, where d_m is the signed 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. 2.3 shows the traces of $\vec{n}_2$- and $\vec{n}_3$-characteristic lines generated by pairs of vertical lines (oriented along $\vec{n}_1$ in this case). The cluster center for each group of characteristic lines identifies a specific surface in the scene. Note that, due to the different sign of d_m of the two planes, the clusters can be at opposite sides with

respect to the origin. The orientation of the translation $\vec{t}$ 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 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 described in the next subsection. Fig. 2.4 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.

2.3.1 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. 2.4.

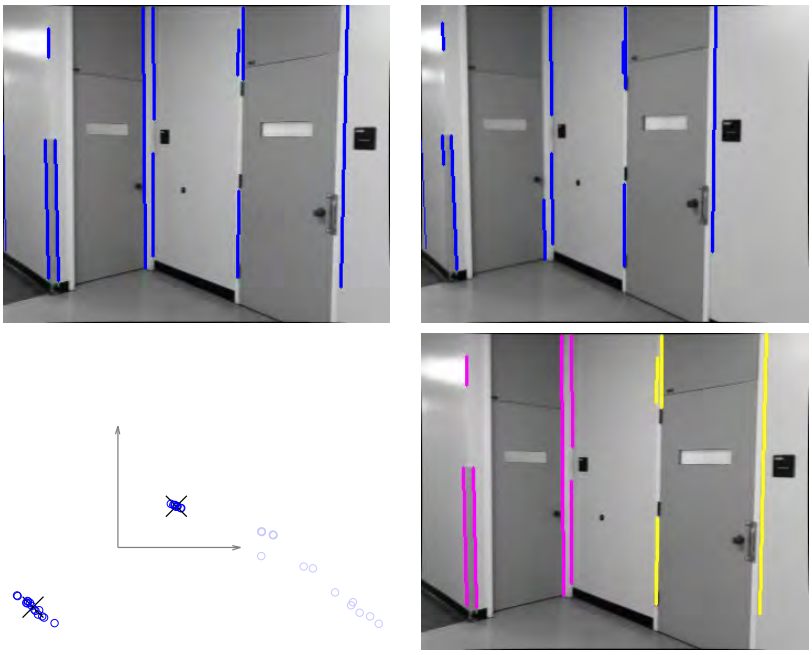


Figure 2.3: Top row: Image pair with detected lines oriented along one canonical direction ($\vec{n}_1$). Only lines that have been matched across images are shown. Bottom left: Traces of the $\vec{n}_2$- and $\vec{n}_3$-characteristic lines on a plane oriented as $\vec{n}_1$. The cluster centers, found 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).

2.4 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

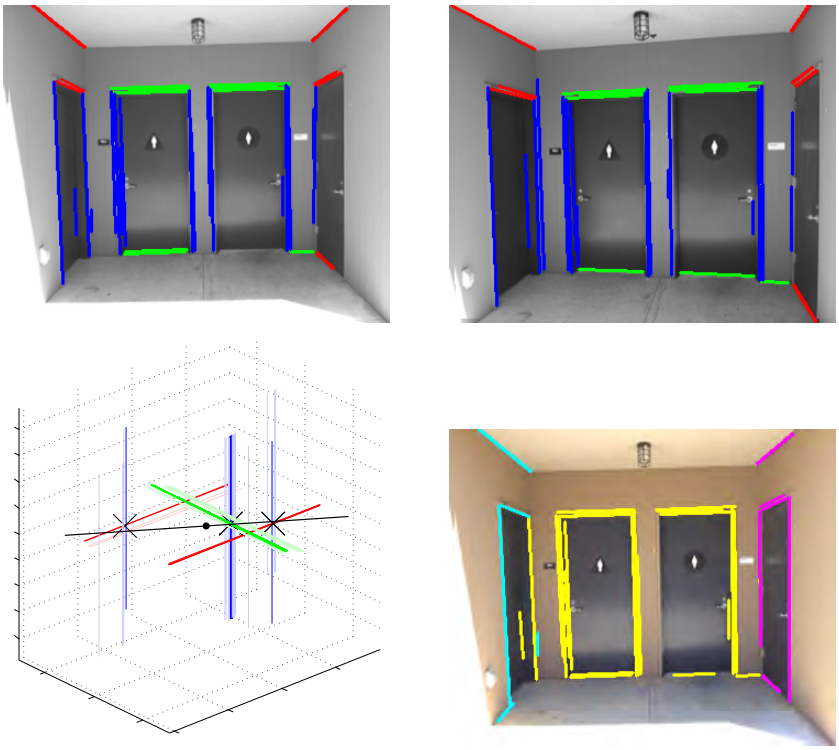


Figure 2.4: Top row: Image pair with detected lines oriented along the three canonical directions (the color of each line identifies its orientation). Only lines that have been matched across images are shown. Bottom left: Characteristic lines for the different orientations. The color of a characteristic line matches the color of the lines it represents. Clusters centers identified by the modified mean shift algorithm [42] 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).

line. We use the mean-standard deviation line descriptor (MSLD [85]), which defines pixel 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. [5], we compute color histograms within each such region. More specifically, we first vector quantize the color of each pixel into one of the 11

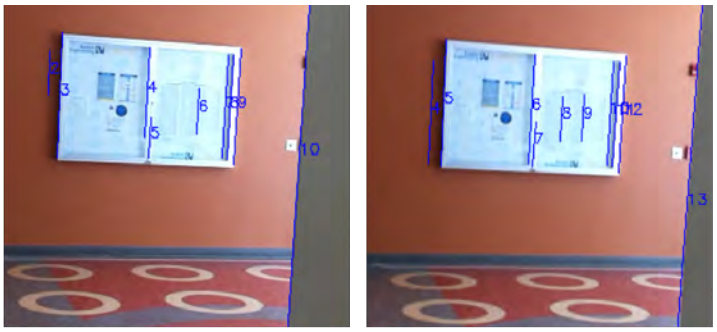
"color names" defined in [82], then compute color vectors histogram on these 11 bins over the same regions used for MSLD. When matching two lines $\mathcal{L}_1$ and $\mathcal{L}_2$, we first compute the Euclidean distances d_T and d_C between the MSLD and color histograms associated with the two lines. Then, a "compound" similarity measure is defined as follows:

$$\mathrm{Sim}(\mathcal{L}_1, \mathcal{L}_2) = w_T e^{-d_T^2/\sigma} + w_C^{-d_C^2/\sigma} \tag{2.5}$$

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. Fig. 2.5 (a) shows two images of the same scene from two different view points and detected line segments. The table below the two images shows three different similarity measures of one reference line segment, $\mathcal{L}_5$ in the left image to its nearby line segments, $\mathcal{L}_{k\in\{6,7,8,9\}}$ in the right image, where the three similarity measures are based on the texture term only ($\mathrm{Sim}_{MSLD}(\mathcal{L}_5, \mathcal{L}_k)$), the color term only ($\mathrm{Sim}_{Color}(\mathcal{L}_5, \mathcal{L}_k)$), and the compound one ($\mathrm{Sim}(\mathcal{L}_5, \mathcal{L}_k)$), respectively. Since the reference line segment is in textured areas the similarity measure based on the texture term only gives the highest score to the correct match $\{\mathcal{L}_5, \mathcal{L}_7\}$ relative to the other false matches, $\{\mathcal{L}_5, \mathcal{L}_{k\in\{6,8,9\}}\}$ while the similarity measure based on the color term only could not distinguish the correct match from the other false matches (all the matching scores are above 0.9). The

reference line segment in Fig. 2.5 (b), $\mathcal{L}_{17}$ lies in poorly textured areas, and thus the similarity measure based on the color histogram gives the higher score to the correct match, $\{\mathcal{L}_{17}, \mathcal{L}_{13}\}$ than the false one, $\{\mathcal{L}_{17}, \mathcal{L}_{12}\}$ while the MSLD based similarity gives the lower score to the correct match. w_C was chosen adaptively to 0.2 or 0.8 based on the average brightness gradient magnitude at both sides of the reference line segment and the compound similarity based on these coefficients was able to always give the higher score to the correct match in the both cases. The decaying parameter σ in Eq. 2.5 was chosen to be 0.28 in all experiments.

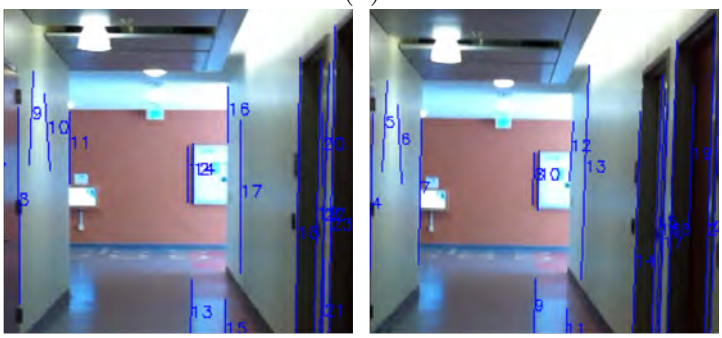
Textural and color information alone, however, cannot guarantee robust line matching. One approach to increasing robustness is to include nearby point feature correspondences and their topological layout [24, 58, 5, 63]. We propose a different strategy, one that restricts the set of possible matches by defining a *line ordering constraint (LOC)*. LOC generalizes the well-known ordering constraint of points in individual epipolar lines, often used in stereo matching [49]. A counterclockwise radial ordering of the images of a set of visible parallel lines is defined with respect to the line images' common vanishing point (Fig. 2.6). 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



k	$\text{Sim}_{\text{MSLD}}(\mathcal{L}_5, \mathcal{L}_k)$	$\text{Sim}_{\text{Color}}(\mathcal{L}_5, \mathcal{L}_k)$	$\text{Sim}(\mathcal{L}_5, \mathcal{L}_k)$*
6	0.21	0.90	0.34
7	0.83	0.97	0.86
8	0.18	0.99	0.34
9	0.19	0.99	0.35

*(W_T=0.8, W_C=0.2)

(a)



k	$\text{Sim}_{\text{MSLD}}(\mathcal{L}_{17}, \mathcal{L}_k)$	$\text{Sim}_{\text{Color}}(\mathcal{L}_{17}, \mathcal{L}_k)$	$\text{Sim}(\mathcal{L}_{17}, \mathcal{L}_k)$*
12	0.19	0.70	0.60
13	0.05	0.98	0.80

*(W_T=0.2, W_C=0.8)

(b)

Figure 2.5: Similarity measures of one reference line segment ($\mathcal{L}_5$ for (a) and $\mathcal{L}_{17}$ for (b)) in the left image to its nearby line segments ($\mathcal{L}_{k\in\{6,7,8,9\}}$ for (a) and $\mathcal{L}_{k\in\{12,13\}}$ for (b)) in the right image, where the three similarity measures are based on the textures term only ($\text{Sim}_{MSLD}(\mathcal{L}_*, \mathcal{L}_k)$), the color term only ($\text{Sim}_{Color}(\mathcal{L}_*, \mathcal{L}_k)$), and the compound one ($\text{Sim}(\mathcal{L}_*, \mathcal{L}_k)$), respectively. w_C was chosen adaptively to 0.2 or 0.8 based on the average brightness gradient magnitude at both sides of the reference line segment and the compound similarity (Eq. 2.5) based on these coefficients always gives the higher score to the correct match in the both cases.

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 $\mathrm{Sim}(\mathcal{L}_1, \mathcal{L}_2)$ defined in (2.5)) and a "skip" term (a constant cost per line skipped in each image). The method will be described in the next section. A similar approach was taken by Cornelis et al. [15], 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.

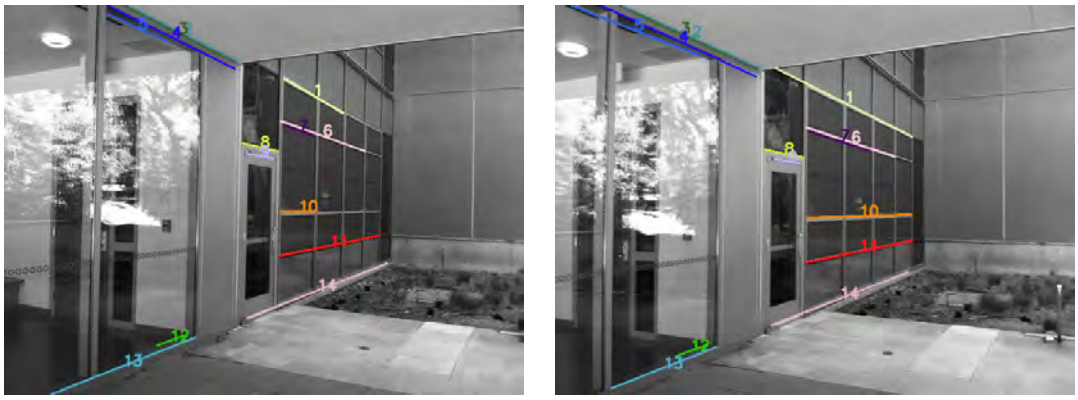


Figure 2.6: The line ordering constraint (LOC) improves robustness of line matching in the case of repeated linear structures. Only matched lines oriented along one of the three canonical directions are shown (the color of each line identifies its match pair). The numbers on the matched lines indicate the counter-clock wise radial ordering with respect to the lines' common vanishing point.

2.4.1 Line Matching by Dynamic Programming with LOC

Let us consider a set of parallel lines lying on a common plane in the scene and their projections in two different viewpoints. The pairwise ranking of two line images induced by the radial ordering in the first image is preserved in the second image. If we have a convex object that occludes some of the lines lying on the plane then the order between a line on the convex object and the line on the behind plane can change. However, since this is rare case in the indoor environment consisting of chains of vertical walls, and when the camera translation is small relative to the depth of the scene it happens only a few consecutive frames, we exploit this line ordering constraint to reduce the number of false line matches.

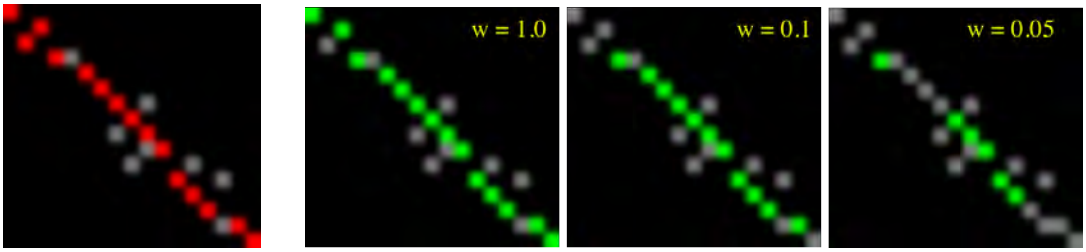
Suppose that we have a bundle of line segments and their common vanishing point in the first image, and same for the second image. We assume that the vanishing point in the second image corresponds to the one in the first image, and thus the two bundles of image line segments are the projections of a bundle of parallel lines in space. Also, a counter- clockwise radial ordering of the line segments have been computed with respect to the common vanishing point for each image. Now, for each line segment in the first image, best two candidate line matches are found in the second image based on the compound similarity measure (Eq. 2.5). If a line segment does not have any candidate line matches then we remove it. From the candidate line matches, we build a graph, where each node is one of the candidate line matches. Let (x_j, y_k) represent a candidate match

between j-th and k-th line segment in image x and y, respectively. A generic edge in the graph goes from node (x_j, y_k) to node (x_m, y_n), where $m > j$ and $n > k$ by the ordering constraint. The cost of a node is the compound similarity subtracted from one, $1 - \text{Sim}(\mathcal{L}_j, \mathcal{L}_k)$. The cost of an edge is defined as:

$$E((j,k),(m,n)) = (m-j-1) + (n-k-1) \tag{2.6}$$

This is the "skip" cost: if we skip one line left or right, we pay a penalty. For the line matching problem, we are looking for a minimum cost path from node (x_1, y_1) to node (x_{N_x}, y_{N_y}) where N_x and N_y are the number of line segments in image x and image y, respectively. We solve this using the dynamic programming. Since node (x_1, y_1) and node (x_{N_x}, y_{N_y}) could be incorrect line matches, we add two fake nodes (x_0, y_0) and (x_{N_x+1}, y_{N_y+1}) and connect (x_0, y_0) with all nodes (x_j, y_k) in the graph and connect (x_{N_1+1}, y_{N_2+1}) with all nodes (x_j, y_k) in the graph. The cost of the edge from (x_0, y_0) to (x_j, y_k) is $(j-1) + (k-1)$. Likewise, the cost of the edge from (x_{N_1+1}, y_{N_2+1}) to (x_j, y_k) is $(N_1 - j) + (N_2 - k)$. The node cost of the two fake nodes is zero. Now we compute a minimum cost path from (x_0, y_0) to (x_{N_1+1}, y_{N_2+1}) and remove (x_0, y_0) and (x_{N_1+1}, y_{N_2+1}) from the path. We use following recursive equation:

$$C(j,k) = n(j,k) + \min_{m<j, n<k} \{C(m,n) + wE((m,n),(j,k))\} \tag{2.7}$$



Ground Truth (Red) Line Matching Results (Green)

Figure 2.7: The minimum cost paths on 2D grid tables found by the dynamic programming with different w, where (j, k) cell represents a pair of j-th line segment in the first image and k-th line segment in the second image. Black cells indicate no line matches are found. Colored cells indicate the line match candidates are detected by the compound similarity measure (i.e. the nodes in the graph), and green or red cells represent correct matches (nodes) in the minimum cost path.

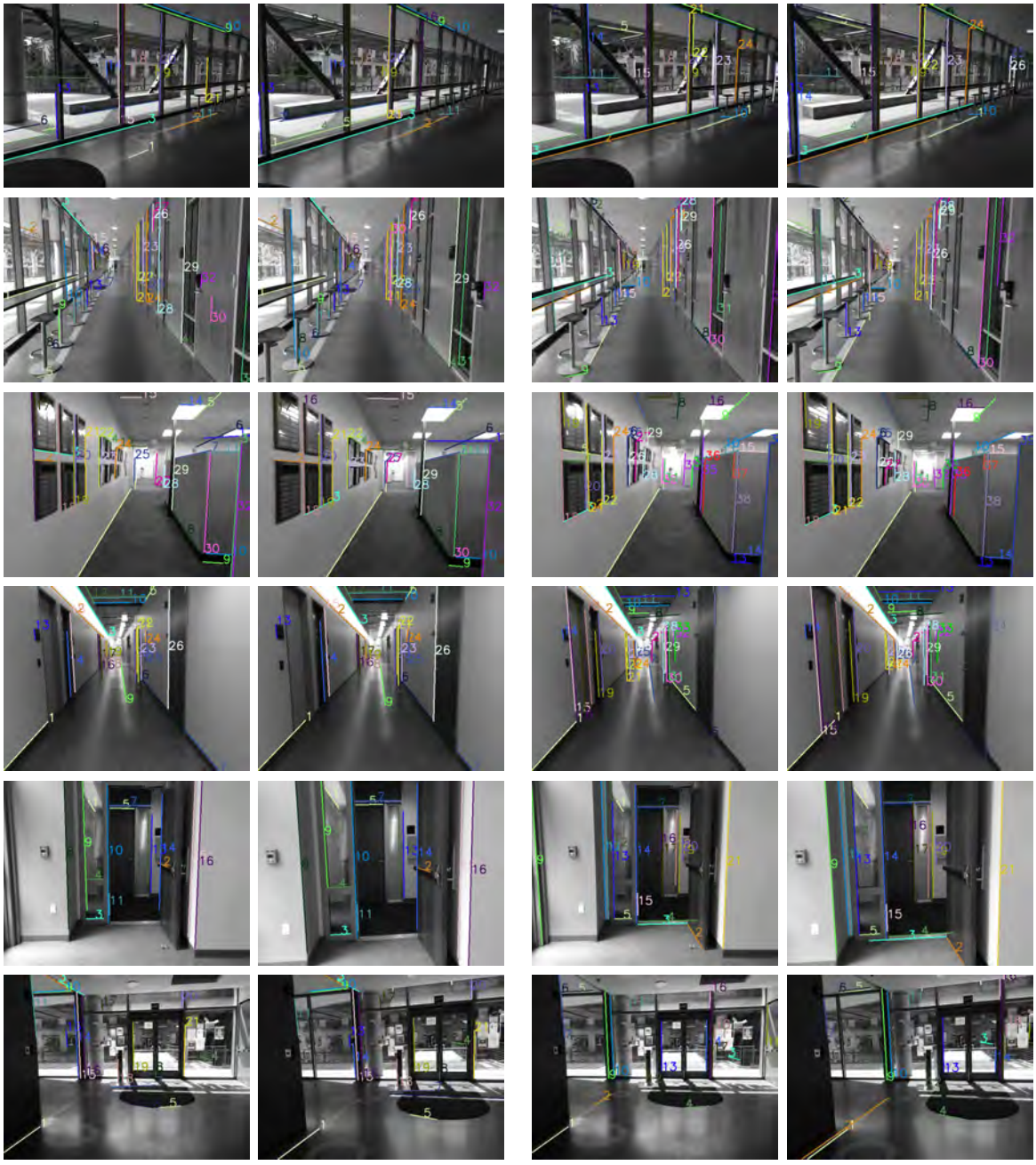
where $C(j, k)$ is the cost of a minimum path from (x_0, y_0) to (x_j, y_k) and w is a parameter to control how much we put a penalty on the skip. Fig, 2.7 shows the minimum cost paths on 2D grid tables found by the dynamic programming with different w, where (j, k) cell represents a pair of j-th line segment in the first image and k-th line segment in the second image. Black cells indicate no line matches are found. Colored cells indicate the line match candidates are detected by the compound similarity measure (i.e. the nodes in the graph), and green or red cells represent correct matches (nodes) in the minimum cost path. The penalty on the skip cost and its control parameter w was chosen empirically to be 1.0 to avoid missing many correct line matches ($w = 0.1$ or 0.05 in Fig, 2.7).

2.4.2 Evaluation

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 [46, 85]. Specifically, we only kept matches with a compound similarity measure (Eq. 2.5) larger than 0,65, and with a NNR value larger than 0.8. Table. 2.1 and Fig. 2.8 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 [85]. 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 2.1: Line matching accuracy tests on 6 image pairs using the NNR criterion [46, 85] and the proposed LOC algorithm. 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.



(a) NNR (b) LOC

Figure 2.8: Line matching results based on Nearest-Neighbor-Ratio criterion (NNR) (a), and our line matching results with the line ordering constraint (LOC) (b).

Chapter 3

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.

3.1 Line Chain Construction

As will result clear in Sec. 3.2, 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. 3.1).

To mitigate problems associated with fragmentation, we implemented a 3-

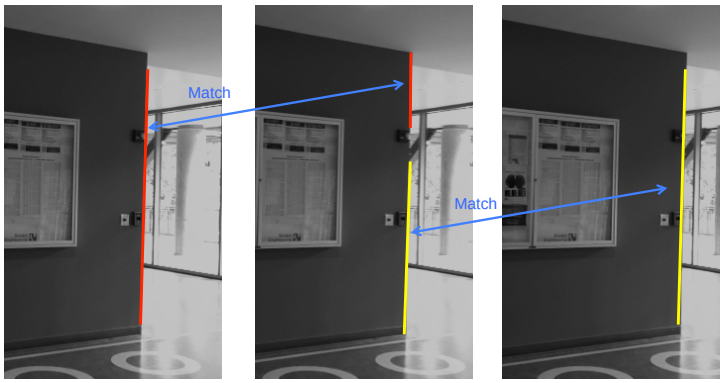


Figure 3.1: The line fragmentation problem for line chain construction. The same line segment (corresponding to the edge of the dark gray wall) is seen in three views, but it is split in two segments in the second view. The segment in the first view is matched to one of the two segments, while the segment in the third view is matched with the other segment, impeding formation of a line chain.

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. 2.4 and by the following recursive equation:

$$C(i,j,k) = n(i,j,k) + \min_{m<i, n<j, l<k} \{C(m,n,l) + wE((i,j,k),(m,n,l))\} \tag{3.1}$$

where $C(i,j,k)$ is the cost of a minimum path from a fake node, (x_0, y_0, z_0) to (x_i, y_j, z_k) for consecutive image frames $\{x, y, z\}$, and w is a parameter to control how much we put a penalty on the skip. $n(i,j,k)$ is the average cost value of $n(i,j)$ and $n(j,k)$, and $E((i,j,k),(m,n,l))$ is the edge cost. We set the edge cost to $(i-m-1)+(j-n-1)+(k-l-1)$. Fig. 3.2 shows a diagram illustrating a

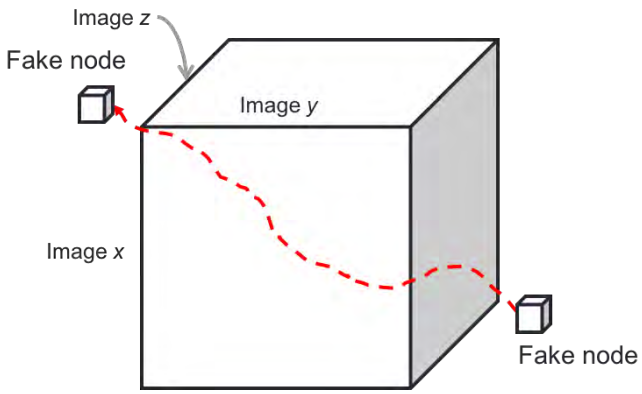


Figure 3.2: A diagram illustrating the minimum cost path for three consecutive image frames by dynamic programming

minimum cost path for three consecutive image frames.

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. 3.3). 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. 3.3).

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 $\mathrm{Sim}(\mathcal{L}_i, \mathcal{L}_j)$ defined in (2.5). 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. 3.4.

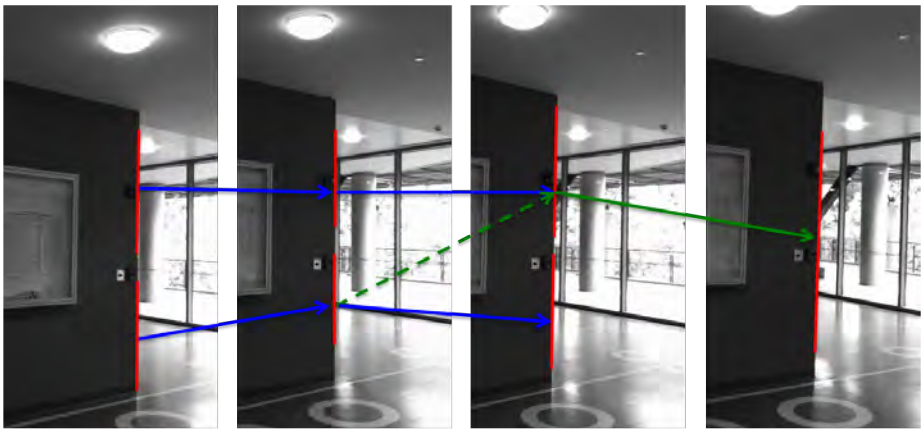


Figure 3.3: A sequence of images with the 3-frames line matching results. Two triplets of lines are found by the 3-frames line matching algorithm for the first three frames (two pairs of blue arrows). One triplet of lines are found for the next three frames (shown by a pair of green arrows). The bottom 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).

frame 1 frame 4 frame 7 frame 10

Figure 3.4: Line chains correctly identified across four image frames. Only lines that oriented along the vertical direction are shown.

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. 3.2) and for plane fitting (Sec. 3.4). The ordering is induced by the radial line ordering around the vanishing points (LOC, Sec. 2.4) 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 by the LOC) is the same.

3.2 The Multi-View Characteristic Lines Algorithm

The 2-frames characteristic lines algorithm [42], summarized in Sec. 2.3, attempts to determine which line pairs are $\vec{n}$-coplanar by looking at local concentrations 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.

3.2.1 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 $\vec{n}$-characteristic line traces for all view pairs. Given the current mode estimate $\hat{\mathbf{l}}^*$, the p-th vector is assigned weight w_p with

$$w_p = K \exp\left(-c\|\hat{\mathbf{l}}^* - \mathbf{l}_p^*\|^2\right) = K \exp\left(-c \sum_m \|\hat{\mathbf{l}}_m^* - \mathbf{l}_{m,p}^*\|^2\right) \tag{3.2}$$

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 $\{w_p\}$:

$$\hat{\mathbf{l}}_m^* = \sum_p w_p \mathbf{l}_{m,p}^* \tag{3.3}$$

Fig. 3.5 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.

3.2.2 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. 3.6 (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. 3.1).

The *visibility set* $\mathcal{V}_p$ of the p-th line pair is defined as the set of view pairs

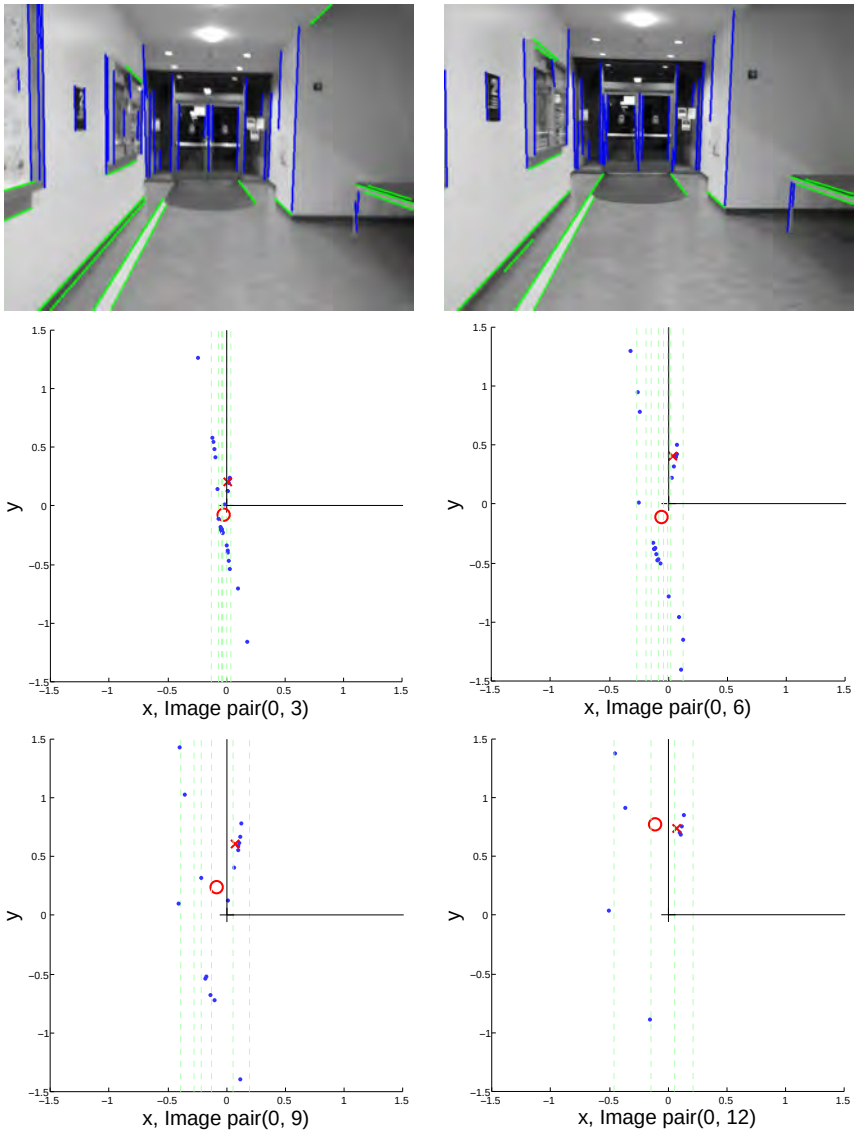


Figure 3.5: 2-view vs. multi-view characteristic line clustering (Sec. 3.2.1). Top: Image frames 0 and 3 with detected lines oriented along the y (green) and z (blue) canonical directions. Second and third rows: Characteristic lines for the y (dashed green lines) and z (blue dots) directions plotted 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.

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 (3.2) to take visibility sets into account as follows:

$$w_p = K\ |\mathcal{V}_p| \exp\left(-c \sum_{m \in \mathcal{V}_p} \frac{\|\hat{\mathbf{l}}^*_m - \mathbf{l}^*_{m,p}\|^2}{|\mathcal{V}_p|}\right) \tag{3.4}$$

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 (3.3) is modified as follows:

$$\hat{\mathbf{l}}^*_m = R_m \sum_{p \in \mathcal{W}_m} w_p \mathbf{l}^*_{m,p} \ \text{ with } \ R_m = \frac{1}{\sum_{p \in \mathcal{W}_m} w_p} \tag{3.5}$$

3.2.3 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 [42]. We give only a short summary of the algorithm in the 2-views case here; the reader is referred to 2.3.1 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 (3.4) and (3.5) is relatively straightforward.

3.2.4 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. 3.1) 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.

3.2.5 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. 3.6 (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. 3.6 (d)), and only one representative per component (specifically, the one with highest associate density value as computed by mean shift) is kept (Fig. 3.6 (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.

3.3 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 [54] to reconstruct

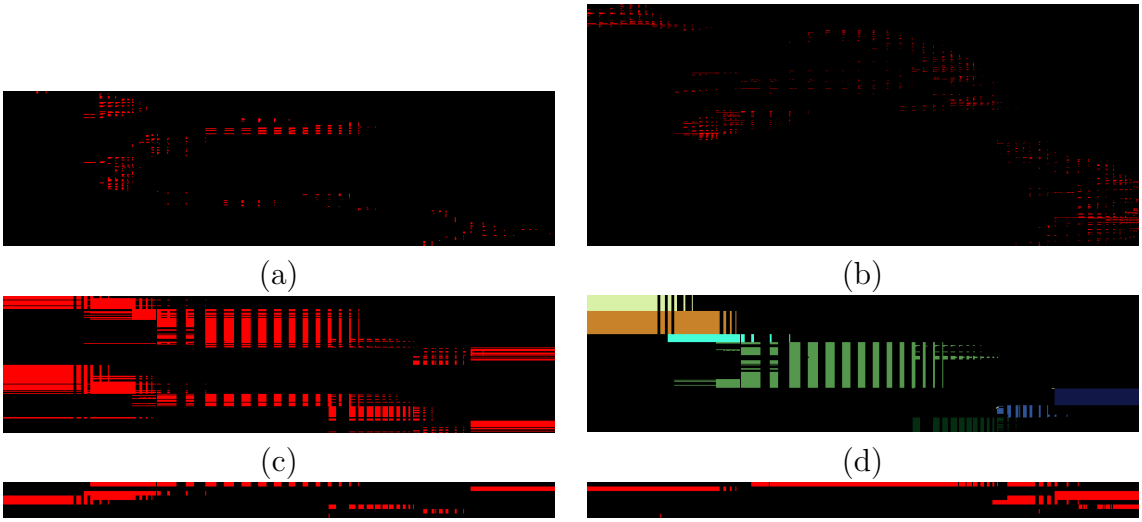


Figure 3.6: (a),(b): Visibility tables. The (p, m) cell of each table indicates whether the p-th line pair (oriented along the X direction (a) or the Z direction (b)) is visible by the m-th view pair. (c): The cluster visibility table for lines in both the X and Z direction. Clusters were computed on $\vec{n}$-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. 3.2.5). (e): The selected clusters. (f) Selected clusters computed from $\vec{n}$-characteristic lines, with $\vec{n}$ oriented along the X direction.

the camera motion vectors $\{\vec{c}_i\}$. This was shown to be more robust than the least squared deviation formulation [80] in the presence of outliers. We also use a parallel rigidity test as in [54] for unique realizability of camera locations from pairwise directions. The parallel rigidity test [55] 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 [41].

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

and solves the following problem:

$$\begin{aligned} &\underset{\{\vec{c}_i\},\{\delta_{i,j}\}}{\arg\min} \textstyle\sum_{(i,j)} \|\vec{c}_j - \vec{c}_i - \delta_{i,j}\vec{\gamma}_{i,j}\| \\ &\text{s.t. } \textstyle\sum_i \vec{c}_i = 0 \; ; \; \delta_{i,j} \geq D \end{aligned} \tag{3.6}$$

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

$$\begin{aligned} &\underset{\{\vec{c}_i\},\{\delta_{i,k}\}}{\arg\min} \textstyle\sum_{(i,j,k)} \|\vec{c}_j - \vec{c}_i - \delta_{i,k}\vec{\tau}_{i,j,k}\| \\ &\text{s.t. } \textstyle\sum_i \vec{c}_i = 0 \; ; \; \delta_{i,k} \geq D \end{aligned} \tag{3.7}$$

This formulation is very similar to (3.6), 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

(3.7) 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 (3.6), as only a small number of different surfaces are visible in typical images. The modified LUD problem is solved by an iteratively reweighted least squares (IRLS) solver [20, 90].

3.4 Manhattan Structure Computation

Given the sequence of camera locations (Sec. 3.3) 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. 3.1), determine its visibility set, and compute a 2-D least-squares triangulation of the line's traces on an orthogonal plane [33]. We then perform bundle adjustment [33] using the Ceres solver [1] 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. 3.7). 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. 3.7). 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. 3.1) 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.

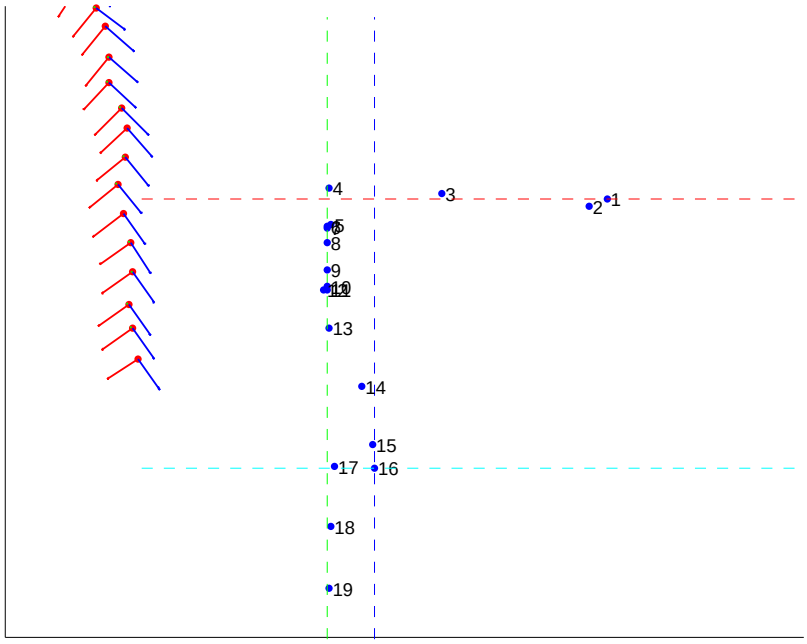


Figure 3.7: The main vertical surfaces in the scene oriented along X and Y (dashed lines) are shown together with the traces of the vertical lines on the horizontal plane (blue points) and the estimated camera location and orientation in the trajectory.

3.4.1 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. 3.7. While some lines can be safely assigned to the closest plane, in other cases distance-based association would lead to incorrect results.

3.4.2 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.

The alignment/deviation terms measure the evidence provided by observations o (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 ($\bar{\|}_{i,i+1}$):

$$
\begin{aligned}
P(\|^X_{i,i+1}|o) &= K \cdot P(o|\ \|^X_{i,i+1}) \cdot P(\|^X_{i,i+1}) = K \cdot rla^X_{i,i+1} \cdot P(\|^X_{i,i+1}) \\
P(\|^Y_{i,i+1}|o) &= K \cdot rla^Y_{i,i+1} \cdot P(\|^Y_{i,i+1}) \\
P(\bar{\|}_{i,i+1}|o) &= K \cdot rld_{i,i+1} \cdot P(\bar{\|}_{i,i+1})
\end{aligned}
\tag{3.8}
$$

where the normalization constant K ensures that the three posteriori probabili-

ties 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,

$$\begin{aligned} P(N_{i+1,k}|N_{i,j},o) &= P(N_{i+1,k}|N_{i,j},\|^X_{i,i+1}) \cdot P(\|^X_{i,i+1}|o) \\ &+ P(N_{i+1,k}|N_{i,j},\|^Y_{i,i+1}) \cdot P(\|^Y_{i,i+1}|o) \\ &+ P(N_{i+1,k}|N_{i,j},\bar{\|}_{i,i+1}) \cdot P(\bar{\|}_{i,i+1}|o) \end{aligned} \tag{3.9}$$

where we took into consideration the fact that the observations considered here (reconstructed lines) can only help determine the coplanarity (or lack thereof) of the lines (ex. The first term in the sum, $P(N_{i+1,k}|N_{i,j},\|^X_{i,i+1}) \cdot P(\|^X_{i,i+1}|o)$ was reduced from $P(N_{i+1,k}|N_{i,j},\|^X_{i,i+1},o) \cdot P(\|^X_{i,i+1}|N_{i,j},o)$). 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 ($\|^X_{i,i+1}$), 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 $\|^Y_{i,i+1}$. If the lines are not $\vec{n}$-coplanar in either direction ($\bar{\|}_{i,i+1}$), v is assigned the value l only when the planes are identical ($j = k$); v is set equal to h otherwise.

Chapter 4

Results

In this chapter, we show quantitative/qualitative comparative assessment of 2-view CL based SfM (Ch. 2) and multi-view CL based SfM (Ch. 4.3) on several challenging test sequences taken inside a university building characterized by mostly textureless walls. In order to provide a quantitative evaluation of proposed algorithms, 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.

4.1 Implementation Details

4.1.1 Line Detection

We use the LSD (Line Segment Detector) algorithm [30] 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 [76]. 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.

4.1.2 Vanishing Points Estimation

Following [37], 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. 2.1). 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 [1]. 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.

4.1.3 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.

4.2 Results: Two-view CL

Quantitative comparative assessment of our two-view CL based SfM (described in Ch. 2) 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. 4.1.

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 can test for their coplanarity using Plücker matrices [60]. 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 $\mathcal{L}_2$ [60]. 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 assess-

Figure 4.1: 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 shown by dotted lines represent potential planar patches. They contain all coplanar lines in a cluster, and are computed as described in Ch. 2. 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 shown with the same color as in the top row.

ment 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 [88], created and made freely available by Changchang Wu. The second technique is Elqursh and Elgammal's algorithm [23], 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 we declare the three 3-D 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.

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. 4.1.2, that rotates each line segment to align it with the associated vanishing point.

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 [9] of precision/recall values computed by varying the following parameters: (1) the constant σ in the function $g(D)$ defined in Sec. 2.3.1; (2) a threshold used to select the inlier characteristic line clusters.

Note that line detection and matching across images was performed automatically. 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. 4.2. 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.

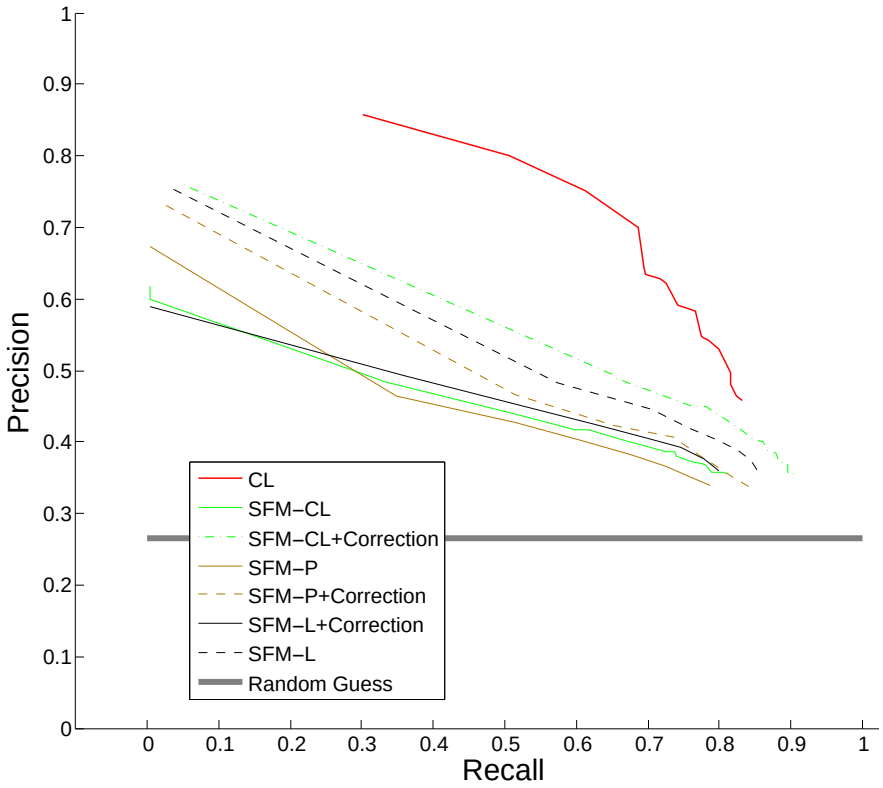


Figure 4.2: Precision/recall curves for the algorithms considered (*SFM-P*, *SFM-L*, *SFM-CL*, *CL*) with and without the "correction" pre-processing step that aligns line segments with the associated vanishing point. (Note that the *CL* method is always computed with this correction.)

4.3 Results: Multiple-view CL

We have tested our multi-view CL based SfM (described in Ch. 3) 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. 4.1.2). 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. 4.3.3).

4.3.1 Results: Qualitative

We present results of reconstructed trajectories and structure over three videos in Figs. 4.3–4.5. 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 [42], followed by global motion estimation (Sec. 3.3) with additional *scale normalization* and Manhattan layout computation (sec. 3.4) (c); same as in (c), but with multi-view clustering (Sec. 3.2) (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 [13], possibly also using inertial measurements, when available [39]. In each figure, we also show the trajectories computed via SfM from point features, using two open source software packages: VisualSFM [88], by Changchang Wu; and the Theia multiview geometry library (TheiaSFM) [72], 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 did not work well with our indoor video sequence. All other parameters were left to their default values.

Sequence 1 (Fig. 4.3) 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. 3.4.

Sequence 2 (Fig. 4.4) 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. 4.5) 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.

4.3.2 Results: Quantitative

In order to provide a quantitative evaluation of our 2-view and multi-view algorithms, we followed the same procedure in Sec. 4.2. Note that we do not 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. 4.5). 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 [60] as in Sec. 4.2. By validating the estimated coplanarity with the ground-truth values for all line triplets, we obtain a precision-recall pair. We vary the threshold δ_P used to threshold the norm of $\mathbf{L}_1\mathbf{L}_2^*\mathbf{L}_3$ 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. 4.6. 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 [50] 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 [50], using the relative rotations and line matches produced by our algorithm. Although 5 lines matches are sufficient for this algorithm [50], we used 6 matches to obtain better result. Unlike our

SfM pipeline, which only analyzes view pairs, the algorithm of [50] 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 problem intractable once these estimates enter the global camera translation step (Sec. 3.3). In order to reduce the size of the LUD problem (3.6), we visit all sets of view triplets with a common view, and only retain the triplet with the smallest reprojection error of its reconstructed lines.

Fig. 4.6 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 [50], we show in Fig. 4.7 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 [50] seems to produce incorrect vertical values for the camera pose.

4.3.3 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 s 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

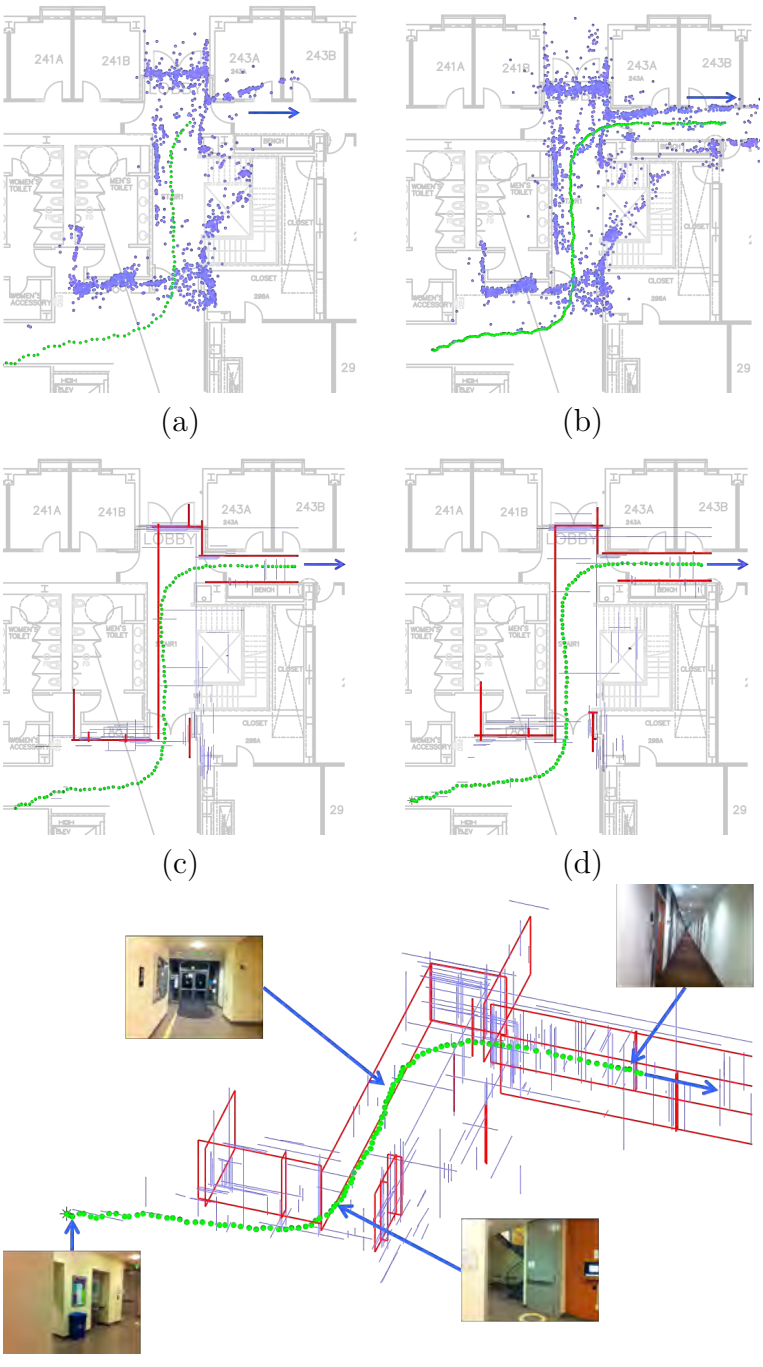


Figure 4.3: Results for Sequence 1. Camera trajectory, shown with green dots; feature points (a-b) or lines (c-e), shown with blue dots/lines; and Manhattan structure (c-e), shown with red lines/retangles. (a): VisualSFM [88]; (b) TheiaSFM [72]; (c) 2-view characteristic lines clustering [42]; (d-e): Multi-view clustering (Sec. 3.2).

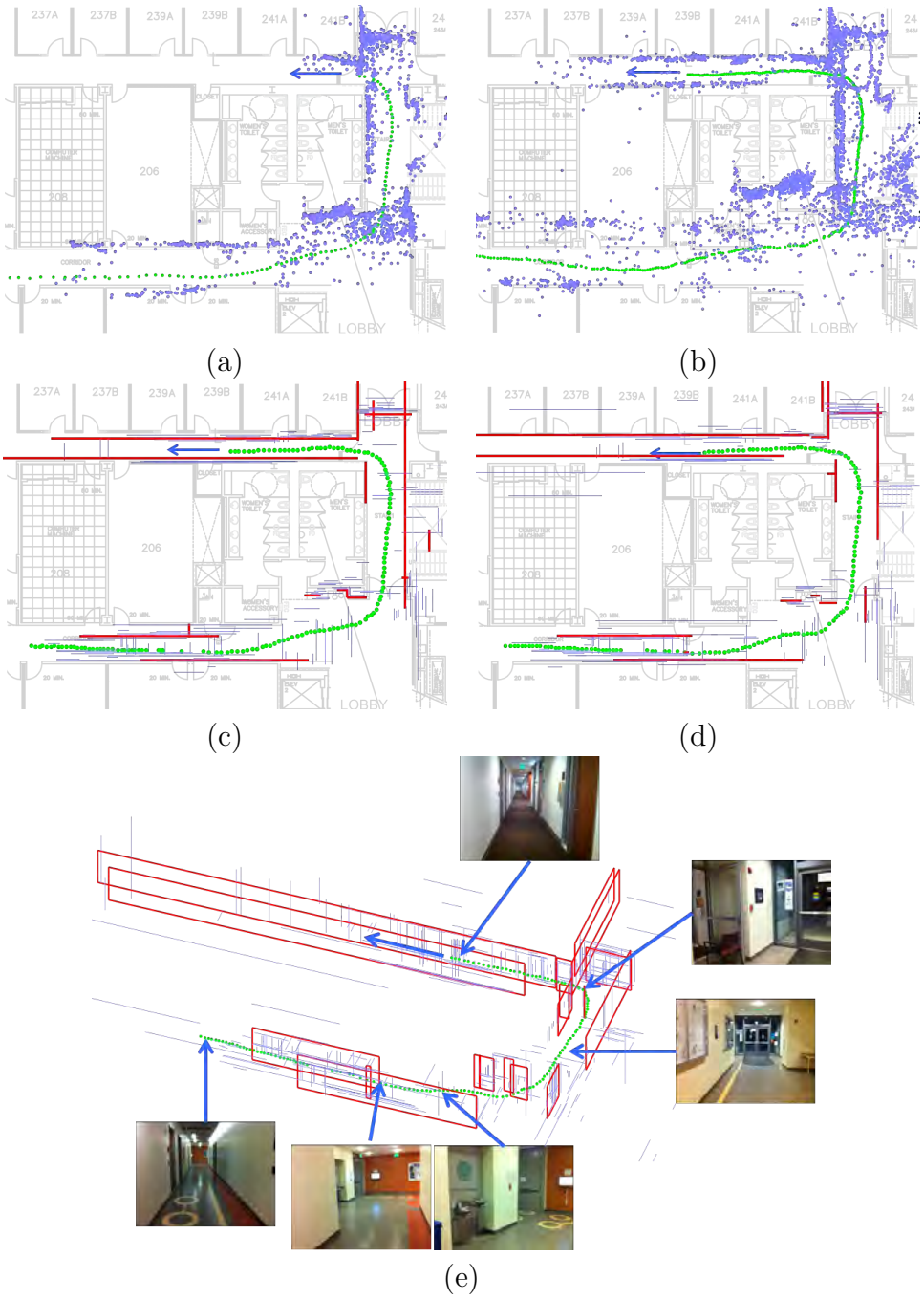


Figure 4.4: Results for Sequence 2 (see caption of Fig. 4.3)

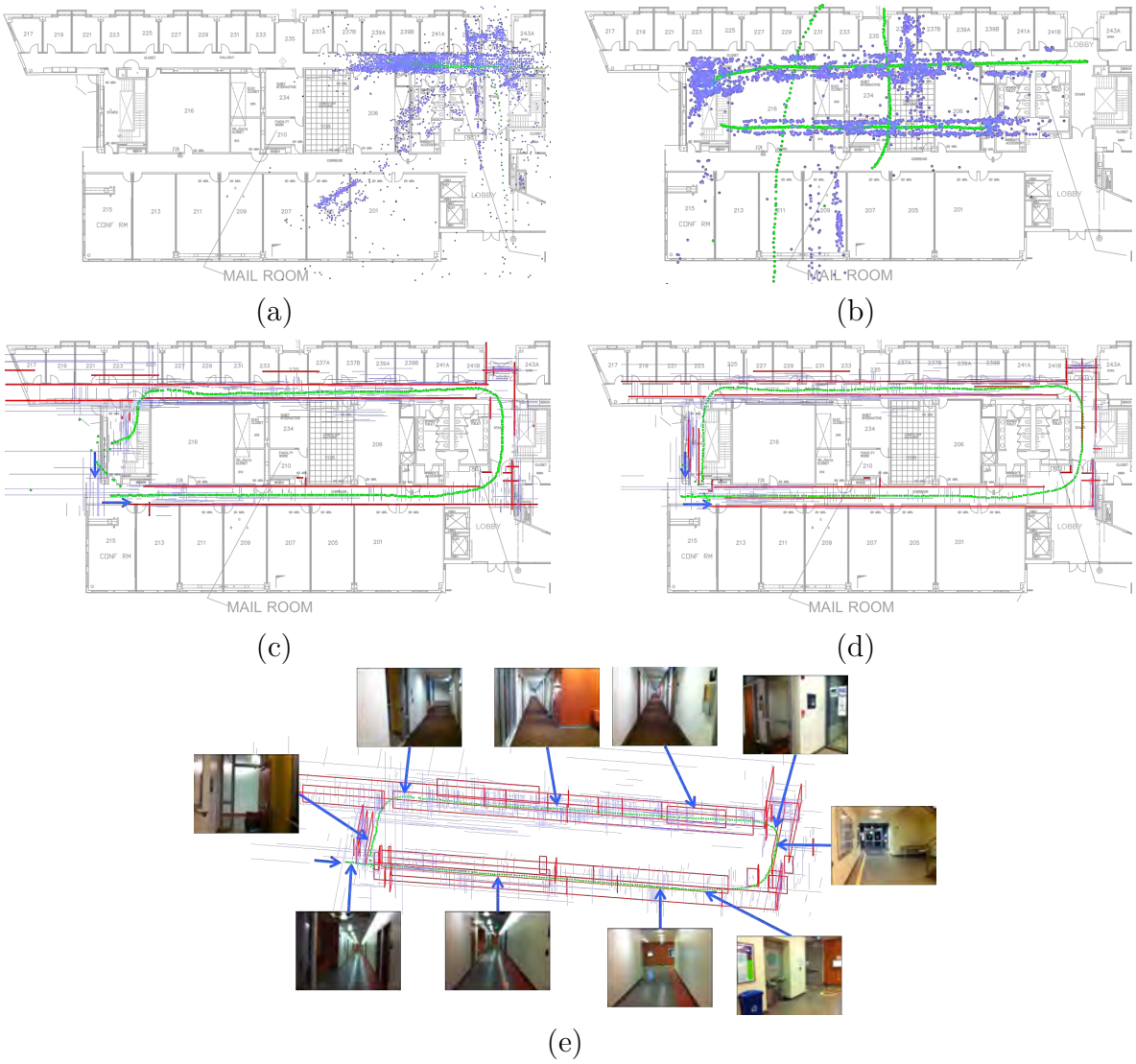


Figure 4.5: Results for Sequence 3 (see caption of Fig. 4.3)

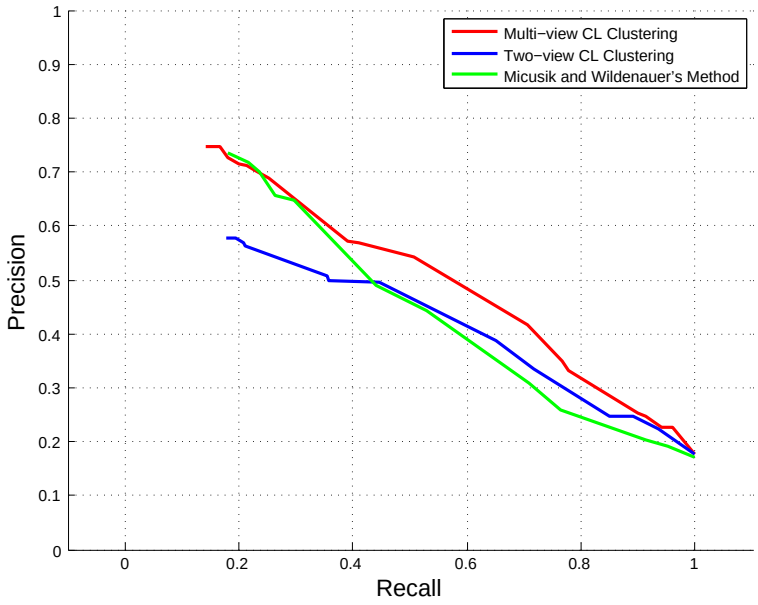
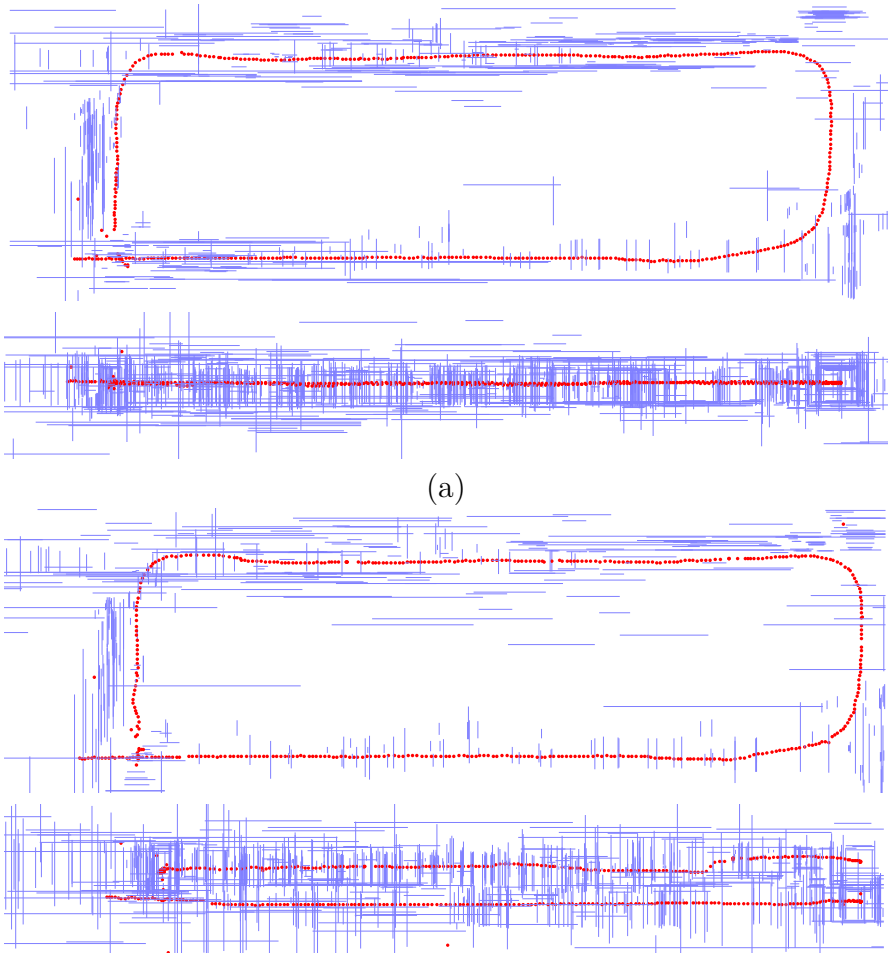


Figure 4.6: Precision/recall curves for the algorithms (see Sec. 4.3.2).



(a)

(b)

Figure 4.7: 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 [50] (b); see Sec. 4.3.2.

Chapter 5

Conclusion

In this thesis, 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 lying 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 thesis 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 heterogeneous features in a unified framework. We are also looking at ways to expand our characteristic line method to weak Manhattan layouts [62], 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 [36, 34, 21], this prior information could be very beneficial for SfM algorithms.

Appendix A

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 (2.4)

$$\langle \vec{t}/d, \vec{u}_2 \rangle = \frac{\sin \angle(\vec{u}_1, \vec{u}_2}{\sin \angle(\vec{u}_1, \vec{n}} \tag{A.1}$$

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. 2.1).

From (2.1) one derives

$$\mathbf{u}_1 \times H_c^T \mathbf{u}_2 = 0 \tag{A.2}$$

Combining (A.2) with (2.2), one obtains

$$(R^T \mathbf{u}_2) \times \mathbf{u}_1 = \mathbf{u}_1 \times \mathbf{n}\mathbf{u}_2^T \mathbf{t}/d \tag{A.3}$$

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

$$\mathbf{u}_2^T \mathbf{t}/d = \frac{\|(R^T \mathbf{u}_2) \times \mathbf{u}_1\|}{\|\mathbf{u}_1 \times \mathbf{n}\|} \cdot ((R^T \mathbf{u}_2) \times \mathbf{u}_1)^T (\mathbf{u}_1 \times \mathbf{n}) \tag{A.4}$$

which is identical to (A.1).

Bibliography

[1] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. http://ceres-solver.org.

[2] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M Seitz, and Richard Szeliski. Building rome in a day. In *2009 IEEE 12th international conference on computer vision*, pages 72–79. IEEE, 2009.

[3] Clemens Arth, M. Klopschitz, Gerhard Reitmayr, and D. Schmalstieg. Real-time self-localization from panoramic images on mobile devices. In *10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 37–46, 2011.

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

[5] Herbert Bay, Vittorio Ferrari, and Luc Van Gool. Wide-baseline stereo matching with line segments. In *Proc. Computer Vision and Pattern Recognition (CVPR 2005)*, volume 1, pages 329–336. IEEE, 2005.

[6] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[7] Jean-Charles Bazin, Inso Kweon, Cedric Demonceaux, and Pascal Vasseur. Rectangle extraction in catadioptric images. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–7. IEEE, 2007.

[8] Jean-Yves Bouguet. Camera calibration toolbox for matlab. 2004.

[9] Stephen Boyd and Lieven Vandenberghe. Convex optimization. 2004. *Cambridge Univ. Pr*, 2004.

[10] J Brian Burns, Allen R Hanson, and Edward M Riseman. Extracting straight lines. *IEEE transactions on pattern analysis and machine intelligence*, (4):425–455, 1986.

[11] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[12] Avhishek Chatterjee and Venu Madhav Govindu. Efficient and robust large-scale rotation averaging. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 521–528. IEEE, 2013.

[13] Alessandro Chiuso, Paolo Favaro, Hailin Jin, and Stefano Soatto. Structure from motion causally integrated over time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):523–535, 2002.

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

[15] N. Cornelis, K. Cornelis, and L. Van Gool. Fast compact city modeling for navigation pre-visualization. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1339–1344, 2006.

[16] James M Coughlan and Alan L Yuille. Manhattan world: Compass direction from a single image by bayesian inference. In *Proc. IEEE International Conference on Computer Vision*, volume 2, pages 941–947. IEEE, 1999.

[17] James L Crowley, Patrick Stelmaszyk, Thomas Skordas, and Pierre Puget. Measurement and integration of 3-d structures by tracking edge lines. *International Journal of Computer Vision*, 8(1):29–52, 1992.

[18] Zhaopeng Cui and Ping Tan. Global structure-from-motion by similarity averaging. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 864–872, 2015.

[19] Saumitro Dasgupta, Kuan Fang, Kevin Chen, and Silvio Savarese. Delay: Robust spatial layout estimation for cluttered indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 616–624, 2016.

[20] Ingrid Daubechies, Ronald DeVore, Massimo Fornasier, and C Sinan Güntürk. Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics*, 63(1):1–38, 2010.

[21] Erick Delage, Honglak Lee, and Andrew 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*, volume 2, pages 2418–2428. IEEE, 2006.

[22] Agnes Desolneux, Lionel Moisan, and Jean-Michel Morel. *From gestalt theory to image analysis: a probabilistic approach*, volume 34. Springer Science & Business Media, 2007.

[23] A. Elqursh and A. Elgammal. Line-based relative pose estimation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3049–3056, 2011.

[24] Bin Fan, Fuchao Wu, and Zhanyi Hu. Line matching leveraged by point correspondences. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 390–397. IEEE, 2010.

[25] Andrew W Fitzgibbon and Andrew Zisserman. Automatic camera recovery for closed or open image sequences. In *Computer Vision—ECCV'98*, pages 311–326. Springer, 1998.

[26] Alex Flint, David Murray, and Ian Reid. Manhattan scene understanding using monocular, stereo, and 3d features. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2228–2235. IEEE, 2011.

[27] Jan-Michael Frahm, Pierre Fite-Georgel, David Gallup, Tim Johnson, Rahul Raguram, Changchang Wu, Yi-Hung Jen, Enrique Dunn, Brian Clipp, Svetlana Lazebnik, et al. Building rome on a cloudless day. In *European Conference on Computer Vision*, pages 368–381. Springer, 2010.

[28] Venu Madhav Govindu. Combining two-view constraints for motion estimation. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–218. IEEE, 2001.

[29] Venu Madhav Govindu. Lie-algebraic averaging for globally consistent motion estimation. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–684. IEEE, 2004.

[30] Rafael Grompone von Gioi, Jérémie Jakubowicz, Jean-Michel Morel, and Gregory Randall. LSD: a Line Segment Detector. *Image Processing On Line*, 2012, 2012.

[31] Christopher G Harris and JM Pike. 3D positional integration from image sequences. *Image and Vision Computing*, 6(2):87–90, 1988.

[32] Richard Hartley et al. Projective reconstruction from line correspondences. In *Proc. IEEE Computer Vision and Pattern Recognition*, pages 903–907. IEEE, 1994.

[33] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge Univ Press, 2000.

[34] Varsha Hedau, Derek Hoiem, and David Forsyth. Thinking inside the box: Using appearance models and context based on room geometry. *Computer Vision–ECCV 2010*, pages 224–237, 2010.

[35] Manuel Hofer, Michael Maurer, and Horst Bischof. Efficient 3d scene abstraction using line segments. *Computer Vision and Image Understanding*, 2016.

[36] Derek Hoiem, Alexei A Efros, and Martial Hebert. Recovering surface layout from an image. *International Journal of Computer Vision*, 75(1):151–172, 2007.

[37] Myung Hwangbo and Takeo Kanade. Visual-inertial uav attitude estimation using urban scene regularities. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2451–2458. IEEE, 2011.

[38] Nianjuan Jiang, Zhaopeng Cui, and Ping Tan. A global linear method for camera pose registration. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 481–488. IEEE, 2013.

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

[40] Philip Kahn, L Kitchen, and Edward M. Riseman. A fast line finder for vision-guided robot navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(11):1098–1102, 1990.

[41] Ryan Kennedy, Kostas Daniilidis, Oleg Naroditsky, and Camillo J Taylor. Identifying maximal rigid components in bearing-based localization. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 194–201. IEEE, 2012.

[42] Chelhwon Kim and Roberto Manduchi. Planar structures from line correspondences in a manhattan world. In *Computer Vision–ACCV 2014*, pages 509–524. Springer, 2015.

[43] Jana Košecká and Wei Zhang. Extraction, matching, and pose recovery based on dominant rectangular structures. *Computer Vision and Image Understanding*, 100(3):274–293, 2005.

[44] Jana Košecká and Wei Zhang. Video compass. In *Computer Vision—ECCV 2002*, pages 476–490. Springer, 2006.

[45] David C Lee, Martial Hebert, and Takeo Kanade. Geometric reasoning for single image structure recovery. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2136–2143. IEEE, 2009.

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

[47] Yi Ma, Stefano Soatto, Jana Kosecka, and S Shankar Sastry. *An invitation to 3-d vision: from images to geometric models*, volume 26. Springer Science & Business Media, 2012.

[48] Roberto Manduchi and James Coughlan. (computer) vision without sight. *Commun. ACM*, 55(1):96–104, January 2012.

[49] David Marr and Tomaso Poggio. A theory of human stereo vision. Technical report, DTIC Document, 1977.

[50] Branislav Micusik and Horst Wildenauer. Structure from motion with line segments under relaxed endpoint constraints. In *3D Vision (3DV), 2014 2nd International Conference on*, volume 1, pages 13–19. IEEE, 2014.

[51] Eduardo Montijano and Carlos Sagues. Position-based navigation using multiple homographies. In *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*, pages 994–1001. IEEE, 2008.

[52] Pierre Moulon, Pascal Monasse, and Renaud Marlet. Global fusion of relative motions for robust, accurate and scalable structure from motion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3248–3255, 2013.

[53] Nassir Navab and Olivier 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):17–44, 1997.

[54] Onur Ozyesil and Amit Singer. Robust camera location estimation by convex programming. *arXiv preprint arXiv:1412.0165*, 2014.

[55] Onur Özyeşil, Amit Singer, and Ronen Basri. Stable camera motion estimation using convex programming. *SIAM Journal on Imaging Sciences*, 8(2):1220–1262, 2015.

[56] Marc Pollefeys, David Nistér, J-M Frahm, Amir Akbarzadeh, Philippos Mordohai, Brian Clipp, Chris Engels, David Gallup, S-J Kim, Paul Merrell, et al. Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision*, 78(2-3):143–167, 2008.

[57] Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, 2004.

[58] Srikumar Ramalingam, Michel Antunes, Daniel Snow, Gim Hee Lee, and Sudeep Pillai. Line-sweep: Cross-ratio for wide-baseline matching and 3d reconstruction. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 1238–1246. IEEE, 2015.

[59] Srikumar Ramalingam, Jaishanker K Pillai, Arpit Jain, and Yuichi Taguchi. Manhattan junction catalogue for spatial reasoning of indoor scenes. In *Computer Vision and Pattern Recognition, 2013. CVPR 2013.* IEEE, 2013.

[60] José I Ronda, Antonio Valdés, and Guillermo Gallego. Line geometry and camera autocalibration. *Journal of Mathematical Imaging and Vision*, 32(2):193–214, 2008.

[61] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. IEEE, 2011.

[62] O. Saurer, F. Fraundorfer, and 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.

[63] Cordelia Schmid and Andrew Zisserman. Automatic line matching across views. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 666–671. IEEE, 1997.

[64] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision*, pages 501–518. Springer, 2016.

[65] Sudipta N Sinha, Drew Steedly, and Richard Szeliski. Piecewise planar stereo for image-based rendering. In *ICCV*, pages 1881–1888. Citeseer, 2009.

[66] Sudipta N Sinha, Drew Steedly, and Richard Szeliski. A multi-stage linear approach to structure from motion. In *Trends and Topics in Computer Vision*, pages 267–281. Springer, 2012.

[67] Noah Snavely, Steven Seitz, and R Szeliski. Photo tourism: Exploring image collections in 3d (2006). *URL http://www. cs. cornell. edu/˜ snavely/bundler.*

[68] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM transactions on graphics (TOG)*, volume 25, pages 835–846. ACM, 2006.

[69] Stefano Soatto, Ruggero Frezza, and Pietro Perona. Motion estimation via dynamic vision. *IEEE Transactions on Automatic Control*, 41(3):393–413, 1996.

[70] Minas E Spetsakis and John Yiannis Aloimonos. Structure from motion using line correspondences. *International Journal of Computer Vision*, 4(3):171–183, 1990.

[71] Henrik Stewenius, Christopher Engels, and David Nistér. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60(4):284–294, 2006.

[72] Chris Sweeney. Theia multiview geometry library: Tutorial & reference.

[73] Chris Sweeney, Torsten Sattler, Tobias Hollerer, Matthew Turk, and Marc Pollefeys. Optimizing the viewing graph for structure-from-motion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 801–809, 2015.

[74] AWK Tang, TP Ng, YS Hung, and Cheung H Leung. Projective reconstruction from line-correspondences in multiple uncalibrated images. *Pattern Recognition*, 39(5):889–896, 2006.

[75] Petri Tanskanen, Kalin Kolev, Lorenz Meier, Federico Camposeco, Olivier Saurer, and Marc Pollefeys. Live metric 3d reconstruction on mobile phones. In *IEEE International Conference on Computer Vision (ICCV)*, pages 65–72, 2013.

[76] João Manuel Ribeiro Silva Tavares and Armando Jorge Monteiro Neves Padilha. A new approach for merging edge line segments. *RecPad95*, 1995.

[77] Camillo J Taylor and David J Kriegman. Structure and motion from line segments in multiple images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(11):1021–1032, 1995.

[78] Roberto Toldo and Andrea Fusiello. Robust multiple structures estimation with j-linkage. In *Computer Vision–ECCV 2008*, pages 537–547. Springer, 2008.

[79] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.

[80] Roberto Tron and René Vidal. Distributed image-based 3-d localization of camera sensor networks. In *Proceedings of the 48th IEEE Conference on Decision and Contro*, pages 901–908. IEEE, 2009.

[81] Grace Tsai and Benjamin 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*, pages 4695–4701. IEEE, 2012.

[82] Joost Van De Weijer, Cordelia Schmid, Jakob Verbeek, and Diane Larlus. Learning color names for real-world applications. *IEEE Transactions on Image Processing*, 18(7):1512–1523, 2009.

[83] Thierry Vieville. Estimation of 3d-motion and structure from tracking 2d-lines in a sequence of images. In *Computer Vision—ECCV 90*, pages 281–291. Springer, 1990.

[84] Etienne Vincent and Robert 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*, pages 182–187. IEEE, 2001.

[85] Zhiheng Wang, Fuchao Wu, and Zhanyi Hu. Msld: A robust descriptor for line matching. *Pattern Recognition*, 42(5):941–953, 2009.

[86] Juyang Weng, Thomas S. Huang, and Narendra Ahuja. Motion and structure from line correspondences; closed-form solution, uniqueness, and optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (3):318–336, 1992.

[87] Kyle Wilson and Noah Snavely. Robust global translations with 1dsfm. In *Computer Vision–ECCV 2014*, pages 61–75. Springer, 2014.

[88] Changchang Wu. VisualSFM. http://ccwu.me/vsfm/, last checked: 6/15/2014.

[89] Lilian Zhang and Reinhard Koch. An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency. *Journal of Visual Communication and Image Representation*, 24(7):794–805, 2013.

[90] Teng Zhang and Gilad Lerman. A novel m-estimator for robust pca. *The Journal of Machine Learning Research*, 15(1):749–808, 2014.

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

[92] Zihan Zhou, Hailin Jin, and Yi Ma. Robust plane-based structure from motion. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1482–1489. IEEE, 2012.

[93] Zihan Zhou, Hailin Jin, and Yi Ma. Plane-based content-preserving warps for video stabilization. In *Computer Vision and Pattern Recognition, 2013. CVPR 2013.* IEEE, 2013.