Title

Multi-planar Monocular Reconstruction of Manhattan Indoor Scenes

ISBN

978-1-5386-8425-2

Authors

Kim, Seongdo
Manduchi, Roberto
Qin, Siyang

Publication Date

2018-09-01

DOI

10.1109/3dv.2018.00076

Peer reviewed

Multi-planar Monocular Reconstruction of Manhattan Indoor Scenes

Seongdo Kim Roberto Manduchi Siyang Qin
University of California, Santa Cruz
1156 High St, Santa Cruz, CA 95064, USA
{seongdo,manduchi,siqin}@soe.ucsc.edu

Abstract

We present a novel algorithm for geometry and camera pose reconstruction from image sequences that is specialized for indoor Manhattan scenes. Unlike general-purpose SfM/SLAM, our system represents geometric primitives in terms of canonically oriented planes. The algorithm starts by computing multi-planar segmentation and motion estimation from image pairs using constrained homographies. It then proceeds to recover the relative scale at each frame and to determine chains of match clusters, where each cluster is associated with a plane in the scene. Motion and scene geometry (expressed in terms of planar models) are then optimized using a novel formulation of Bundle Adjustment. Compared with other state-of-the-art SfM/SLAM algorithms, our technique is shown to produce superior and realistic surface reconstruction for a monocular indoor scene.

1. Introduction

The problem of joint reconstruction of camera motion and 3-D scene geometry from images (called Structure from Motion (SfM) or SLAM, depending on the context) has been studied for decades. Impressive results have been obtained, both with vast collections of unordered images [1, 31], and with video sequences taken from a moving camera [26, 6]. Rather than attempting to raise the state of the art in general-purpose SfM or SLAM, this work proposes a new approach for a very specific scenario: indoor scenes characterized by a Manhattan World (MW) geometry [5]. The MW geometry assumption is appropriate for most indoor environments. Scenes with vertical walls not intersecting at right angles can be modeled by weak MW [30], which inherits many of the general properties of the MW geometry. Of course, there are cases in which the MW geometry would be inadequate, such as in the presence of curved surfaces, ramps, or generic objects or people visible in the scene; in these cases, our technique would not be directly applicable.

(a) (b)

Figure 1: 3-D textured rendering of an indoor scene from the output of our algorithm. (a) shows a selected frame from the input image sequence, and (b) represents the corresponding perspective in the reconstructed 3-D scene.

The MW geometry is inherently simple, which facilitates reconstruction. For example, by estimating the three vanishing points (an operation that is feasible in edge-rich indoor scenes), one obtains the camera orientation with respect to the "canonical" directions (plane normals) [17]. The homography induced on images of the same plane seen by a moving camera has only three degrees of freedom, which facilitates multi-planar segmentation and estimation [30, 15]. The images of multiple parallel and coplanar lines can be characterized by an invariant descriptor ("characteristic line") that enables robust co-planar line clustering [14]. Our work builds on these previous results, and proposes a technique for SfM/SLAM that makes careful use of the intrinsic properties of the MW geometry.

The main characteristic of our system lies in the fact that all surface elements are represented in terms of canonically oriented planes. Although we use feature points, matched across image pairs, to estimate the plane locations and to validate geometric reconstruction, we never maintain a representation of individual points in space. This is a major departure from traditional reconstruction techniques. The advantage of this approach is highlighted by our novel formulation of Bundle Adjustment, which uses planar primitives, jointly optimized with the camera poses by minimization of a specially designed reprojection error. The output

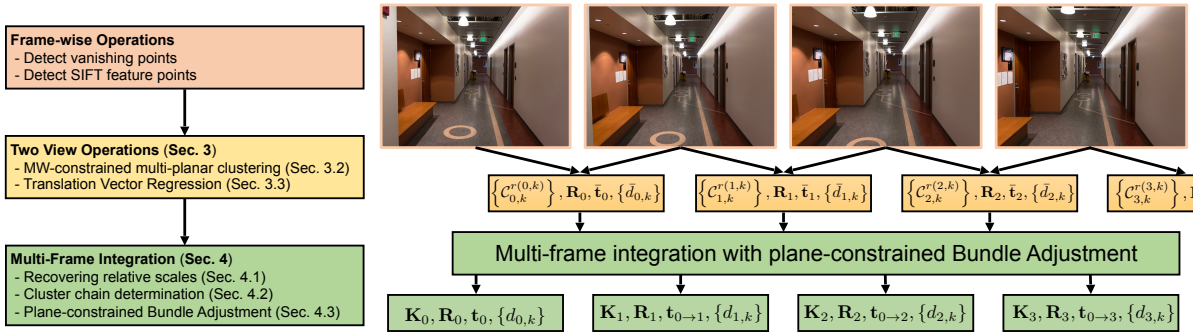

Figure 2: Overview of the different steps of our algorithm

of our algorithm is a set of canonically oriented planes, together with the reconstructed camera poses and a sparse set of back-projected feature points. This information can be used for realistic patch-based reconstruction, such as the one shown in Fig. 1. Our algorithm takes around a second (end-to-end) per image on a GPU-enabled computer. In our experiments, it produced camera trajectories comparable to the state of the art, with superior geometric reconstruction.

The main contribution of this paper is a novel approach for using planar primitives when reconstructing structure and motion in a Manhattan world. We propose new methods for regressing motion vectors, recovering relative scale, and determining chains of matching planar structures in a sequence, all of whom exploit the strong MW constraints. In addition, we propose a new bundle adjustment procedure that uses planar (rather than point) primitives.

This paper is organized as follows . After surveying the related work in the next section, we present our approach for multi-planar segmentation and motion recovery from points matched across two images in Sec. 3. This builds on prior work by Saurer et al. [30] and Kim and Manduchi [15]. The extension to image sequence analysis is presented in Sec. 4. This comprises recovering the relative scale at each frame, creating consistent chains of match clusters, and Bundle Adjustment using our novel formulation using planar primitives. Comparative experiments are presented in Sec. 5. Sec. 6 has the conclusions. A general overview of the system components is shown in Fig. 2

2. Related Work

SfM and visual SLAM algorithms can be roughly divided in three categories: those that match specific features across images (typically points, e.g. [16, 26, 25], or lines, e.g. [14, 24]); those that use direct image alignment to track the camera pose (e.g. [6]); and those that use a volumetric representation of space (e.g. [27]). Some works explicitly represent the presence of planes [32, 3, 21, 39, 8, 12]. While planar fitting is relatively simple when 3-D data (from a RGB-D camera) is available [33, 19, 13], plane tracking is also possible using regular monocular cameras. For example, ORB-SLAM [26] includes a "model selection" component that decides whether the scene is planar; in this case, an homography is more effective at describing view changes than the associated epipolar geometry. π-Match [29] computes planar structures from two views. The camera pose is tracked from the associated homographies, and relative scale between subsequent view pairs is recovered from point triangulation. Pop-up SLAM [38] detects the extent of the visible ground plane (floor) for each image using a convolutional network [37], and uses this information to extract the visible vertical planes. Planes are represented using the minimal representation proposed in [13], and optimization relies on the factor graph of visibility, also introduced in [13]. Point-based LSD SLAM is incorporated [6] to provide the necessary odometry information. Instead of optimizing 3D points, [39] refines positions of the feature points and decrease the loss defined for each pair of frames using homographies. Each homography model is found by a simple RANSAC procedure and its inliers points are iteratively updated during its bundle adjustment.

3. Multi-planar Fitting from Two Views

3.1. Manhattan-constrained Homography Computation

Saurer et al. [30] introduced the concept of constrained homographies for weak Manhattan World scenes (i.e. scenes that contain horizontal planes and vertical planes, where the latter are not necessarily mutually parallel or orthogonal). In the more restrictive Manhattan World case (all visible planes either orthogonal or parallel to one another), the homography $\mathbf{H}$ relating two images of the same plane has three (rather than eight) degrees of freedom. This can be seen by writing the canonical homography decomposition [11]:

$$\mathbf{H} = \mathbf{K}\left(\mathbf{R} + \frac{1}{d}\mathbf{t}\mathbf{n}^T\right)\mathbf{K}^{-1} \tag{1}$$

where $\mathbf{K}$ is the intrinsic parameters matrix, $\mathbf{R}$ represents the rotation of the camera between the two views, d is the distance between the plane and the first camera position, $\mathbf{t}$ is the camera translation, defined with respect to the first camera frame, and $\mathbf{n}$ is the plane unit-norm normal. One useful characteristic of Manhattan wold scenes is that the rotation aligning the camera with the *canonical Manhattan orientation* (i.e., with axes mutually parallel to the three planes normals) can be computed from the three vanishing points [17]. This in turns provides a convenient way to estimate $\mathbf{R}$.

Let $\mathbf{n}^r$ represent the one-hot 3-vector with its r-th entry equal to 1 while the other entries are 0. In the canonical Manhattan orientation, the plane normal $\mathbf{n}$ must be aligned with $\mathbf{n}^r$ for some r. This defines a constraint on the homography induced by the plane. For example, for r=1,

$$\mathbf{H}^1 = \mathbf{K}\mathbf{R}\mathbf{K}^{-1} + \mathbf{K} \begin{bmatrix} \frac{\mathbf{t}}{d} & \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{K}^{-1} \qquad (2)$$

Hence, computing an homography from two images of a plane with known orientation only requires solving for the three entries of $\mathbf{t}/d$ (note that we assume that $\mathbf{K}$ is known from calibration, and that $\mathbf{R}$ is computed from vanishing points estimation). This can be done, for example, using a modified version of the DLT algorithm [11], starting from a set of matching points identified in the two images.

One advantage of using constrained homographies is that they can be computed from a small number of matches. While four matches are necessary in general, for the constrained case we can use *minimal sets* of just two matches[1]. This is particularly important when using random sampling methods (e.g. RANSAC) for multi-planar estimation, as discussed next.

3.2. Manhattan-constrained Multi-Planar Clustering

When multiple planar structures are visible in the image, a (constrained) homography can be computed for each such structure, assuming that the point matches across the two images have been properly clustered. Joint clustering and homography computation can be done using sequential RANSAC or its variants [36, 40, 18, 28]. In particular, T-linkage [23] has been shown to generate reliable multi-planar segmentations. T-linkage, like its predecessor, J-linkage [35], starts from a large number of randomly sampled minimal matches sets, each of which determines its own planar model (hypothesis) by homography fitting. Each match is assigned a distance value to each planar model (e.g., the squared residual after application of the corresponding homography). The matches that are at a distance less than a threshold ϵ to a certain planar model (inliers) form the *consensus set* of this model. Since the planar models are computed from a large number of randomly sampled minimal sets, many of these models should be expected to be similar to one another, meaning that their consensus sets are likely to overlap. T-linkage performs aggregative clustering of matches in such a way that all matches in a cluster support the same set of planar models, i.e., they are all in the intersection of the consensus sets of these models. Clusters are aggregated using a criterion (Jaccard distance of their *preference sets* [35] or Tanimoto distance of their *preference functions* [23]) that encourages clusters to explain a variety of models, rather than allocate one cluster per model.

Kim and Manduchi [15] proposed an extension to the T-linkage algorithm that accounts for the reduced degree of freedom of the homographies to be computed. Minimal sets of two matches are sampled. Each such set determines one plane for each canonical direction (homographies $\mathbf{H}^1, \mathbf{H}^2, \mathbf{H}^3$). Then, for each canonical direction, agglomerative clustering of matches is conducted using T-linkage. A number of variations of the original T-linkage algorithm have proven to be effective:

Visibility constraint. The image of a plane with normal $\mathbf{n}^r$ cannot cross the vanishing line associated with that planar direction (i.e. the line formed by the vanishing points associated with the other two directions). Hence, matches containing points at either side of this vanishing line cannot be placed in the same cluster.

Sample selection. In the initial phase of the algorithm, sets of matches may be sampled so as to maximize the likelihood that they come from the same planar structure in the scene. This can be done by identifying (using orientation maps [20] or other reasoning [15]) regions in one of the two images that are likely to represent whole planar patches. Then, rather than sampling minimal sets, the whole set of matches within each such region could be used to generate individual planar models. This procedure has shown to produce a small number of reliable planar hypotheses, enabling considerable computational savings.

Final cluster aggregation. Due to inaccurate estimation of vanishing points or camera parameters, or to poor localization of feature points, T-linkage often produces oversegmented results. We apply a simple greedy procedure of cluster aggregation after T-linkage clustering, by merging together clusters corresponding to planes with the same normal direction, and with similar values of their scaled distance. The *scaled distance* is the plane's distance to the camera, divided by the magnitude of the camera translation (i.e., the inverse of the norm of the scaled translation vector $\mathbf{t}/d$.) Given two candidate clusters to be merged, we first make sure that each cluster has at least one point in the consensus set of the planar model associated with the other cluster. If this is the case, we merge the two clusters into

[1] In fact, one could potentially compute the constrained homography with a single match, using a method similar to [30].

one, and assign to it the model with the larger consensus set.

3.3. Translation Vector Regression

Each cluster of point matches $\mathcal{C}_{m,k}^{r(m,k)} = \{(\mathbf{x}_m, \mathbf{x}_{m+1})\}$ (where m is the image pair index, k is the cluster index, and $r(m,k)$ identifies the canonical direction of the k-th planar model) determines its own homography, defined by the scaled translation vector $\mathbf{t}_{m,k}/d_{m,k}$. Let us denote by $\bar{\mathbf{t}}_{m,k}$ the unit-norm normalized translation, and by $\bar{d}_{m,k}$ the scaled distance as defined earlier. Since the actual camera translation is unique, all unit vectors $\bar{\mathbf{t}}_{m,k}$ estimated for the same image pair should be identical. We enforce this constraint, and find a common unit-norm translation vector $\bar{\mathbf{t}}_m$ while simultaneously refining the planes' location, by solving the following constrained minimization problem for each image pair:

$$\min_{\bar{\mathbf{t}}_m, \{\bar{d}_{m,k}\}} \sum_k \sum_{(\mathbf{x}_m, \mathbf{x}_{m+1}) \in \mathcal{C}_{m,k}^{r(m,k)}} \|EN\left(\mathbf{H}^{r(m,k)}(\mathbf{R}_m, \bar{\mathbf{t}}_m, \bar{d}_{m,k})\, \tilde{\mathbf{x}}_m\right) - \mathbf{x}_{m+1}\|^2 \quad \text{s.t. } \|\bar{\mathbf{t}}_m\| = 1 \tag{3}$$

where $\tilde{\mathbf{x}}$ is the normalized homogeneous representation of the 2-D point $\mathbf{x}$, and EN is an operator that computes the Euclidean normalization [7] of a homogeneous vector (divides its entries by the last one) then removes the homogeneous part (last entry). $\mathbf{H}^r(\mathbf{R}, \mathbf{t}, d)$ is the constrained homography induced by camera rotation $\mathbf{R}$ and translation $\mathbf{t}$ on the images of a plane with normal of $\mathbf{n}^r$, located at a distance d from the first camera (see e.g. (2)).

Proper parameter initialization is important for convergence to the correct solution. We noted in our experiments that the translation vectors $\bar{\mathbf{t}}_{m,k}$ estimated for the planes that are further away from the camera tend to be incorrect, due to small parallax. This suggests initializing minimization from a translation vector computed from a plane close to the camera. In our experiments, we simply selected the plane with the smallest value of the scaled distance $\bar{d}_{m,k}$.

4. Multi-frame Integration

4.1. Recovering Relative Scale

The translation vectors computed for each frame pair are defined up to an unknown scale. To recover the relative scale of each translation vector, we define a metric normalized by the distance between the camera locations in the first two frames ($\mathbf{t}_0$). Under this metric, the translation vectors and plane distances are $\mathbf{t}_m = \sigma_m \bar{\mathbf{t}}_m$ and $d_{m,k} = \sigma_m \bar{d}_{m,k}$, respectively (as a reminder, m is the frame number and k is the index of the cluster, which is associated with a plane with normal $\mathbf{n}_{m,k}$). We can recover the sequence of *scale factors* $\{\sigma_1, \sigma_2, \dots\}$ by the recursion:

$$\sigma_0 = 1 \;;\; \mathbf{t}_{0\to 0} = \mathbf{0} \;;\; d_{0,k} = \bar{d}_{0,k} \;;\; \mathbf{R}_{0\to 0} = \mathbf{R}_0$$

$$\mathbf{R}_{0\to m} = \mathbf{R}_{m-1}\mathbf{R}_{0\to m-1}$$

$$\mathbf{t}_{0\to m} = \mathbf{t}_{0\to m-1} + \sigma_{m-1}\mathbf{R}_{0\to m-1}^T \bar{\mathbf{t}}_{m-1}$$

$$d_{m,k} = d_{0,k} + (\mathbf{n}_{m,k})^T \mathbf{t}_{0\to m} \;;\; \sigma_m = F\left(\{d_{m,k}, \bar{d}_{m,k}\}\right)$$

In the equation above, $\mathbf{t}_{0\to m}$ represents the vector moving the camera from its position at time 0 to time m, expressed in the reference system of the first camera (m=0). $\mathbf{R}_{0\to m}$ represents the camera orientation at time m with respect to the camera frame at time 0. F is a function that regresses the distances $d_{m,k}$, estimated through propagation $d_{m,k} = d_{0,k} + (\mathbf{n}_k)^T \mathbf{t}_{0\to m}$, against the measured scaled distance $\bar{d}_{m,k}$. Note that, due to noise and accumulated errors, the ratios $d_{m,k}/\bar{d}_{m,k}$ may not be identical for different values of k. In our experiments, we obtained good results by simply setting $\sigma_m = d_{m,k}/\bar{d}_{m,k}$ for the plane k closest to the camera at time m (i.e. with minimum value of $\bar{d}_{m,k}$). When a new plane of index k is identified at time m that was not visible at time 0, we can set $d_{0,k} = d_{m,k} - (\mathbf{n}_k)^T \mathbf{t}_{0\to m}$.

4.2. Cluster Chain Determination

Determining chains of pairwise image matches is a critical step in classical SLAM algorithms [22], as it allows one to associate a point in space with a number of feature points detected over multiple images, and thus to compute the reprojection error for a putative set of poses. In our case, we are interested in not only finding match chains, but also *cluster chains*, which identify the same planar model across subsequent images. We need to ensure that two separate planar models don't get mistakenly merged into one, or that the same model gets split in two.

Formally, the set of points matched across subsequent pairs of the N acquired images can be represented as a directed N-partite graph $\mathcal{G}$, whose m-th partition contains nodes associated with points detected in the m-th image. Each node in the m-th partition may be connected to at most one node associated with a matching point in the previous image (partition m-1) or in the next image (partition m+1). Note that the image ordering implicitly defines an ordering of the edges adjacent to a node. Since a point in an image can be matched to only one point in the next/previous image, both the indegree and the outdegree of any node in $\mathcal{G}$ can be at most 1.

The *line graph* [10] of $\mathcal{G}$, $\bar{\mathcal{G}}$, is the graph whose nodes are the edges of $\mathcal{G}$, and whose edges link the pairs of edges of $\mathcal{G}$ that have a common endpoint. In practice, $\bar{\mathcal{G}}$ represents the set of point pairs, matched across subsequent images. We assign a direction to each edge of $\bar{\mathcal{G}}$ such that its endpoints are ordered congruently to the associated edges in $\mathcal{G}$. Note

that $\bar{\mathcal{G}}$ is $(N-1)$-partite, and that its nodes have indegree and outdegree equal to at most 1.

The clustering of matches described in Sec. 3.2 defines a clustering of the nodes of $\bar{\mathcal{G}}$, which induces a new directed graph $\mathcal{G}_C$ as explained in the following. Each cluster of nodes of $\bar{\mathcal{G}}$ associated with the cluster of matches $\mathcal{C}_{m,k}^{r(m,k)}$ defines one node of $\mathcal{G}_C$. This node inherits the index triplet (m,k,r) from the cluster. Two nodes (m,k_1,r_1) and $(m+1,k_2,r_2)$ of $\mathcal{G}_C$ are connected if (1) r_1=r_2 (planes with the same normal direction), and (2) any two nodes from the corresponding clusters of $\bar{\mathcal{G}}$ are connected. This connection indicates that the planar models defined by the two associated clusters are the same. However, geometric inconsistencies (different planar models merging into one, or the same model splitting in two) may occur at nodes of $\mathcal{G}_C$ with indegree or outdegree larger than 1. To deal with these situations, we first define a weight on the edge linking two nodes (m,k_1,r) and $(m+1,k_2,r)$ of $\mathcal{G}_C$ based on a measure of similarity between $\mathcal{E}^{out}$, the set of outgoing edges from the nodes of $\bar{\mathcal{G}}$ in $\mathcal{C}_{m,k_1}^{r}$, and $\mathcal{E}^{in}$, the set of ingoing edges to the nodes of $\bar{\mathcal{G}}$ in $\mathcal{C}_{m+1,k_2}^{r}$. This weight is equal to their Jaccard distance (intersection over union: $\|\mathcal{E}^{out} \cap \mathcal{E}^{in}\| / \|\mathcal{E}^{out} \cup \mathcal{E}^{in}\|$); it rewards consistency of connections in $\bar{\mathcal{G}}$. Then, for each node of $\mathcal{G}_C$ with indegree (outdegree) larger than 1, we prune off all incoming (outgoing) edges except for the one with maximum weight. This in turn induces a pruning in $\bar{\mathcal{G}}$: any edge linking two nodes of $\bar{\mathcal{G}}$ associated with disconnected nodes of $\mathcal{G}_C$ is pruned off. After this operation, each maximal paths of $\mathcal{G}_C$ (cluster chain) identifies a unique plane. Consequently, any path of the pruned $\bar{\mathcal{G}}$ is formed by nodes (point pairs matched across subsequent images) that are all associated with the same plane in the scene.

To this simple procedure, we added two twists:

Single-frame gap filling. We allow for propagation of a planar model through time even when, for whatever reason, a cluster was not produced at one frame pair. We do this by allowing nodes in the m-th partition of $\bar{\mathcal{G}}$ with incoming and outgoing degree of 1 that do not belong to any cluster to inherit the clustering from the (m-1)-th partition. More precisely, let $\mathcal{N}_m$ be the set formed by the nodes whose adjacent node in the (m-1)-th and in the (m+1)-th partition are associated with clusters with the same plane normal direction. All nodes in $\mathcal{N}_m$ adjacent to nodes in the same cluster in the (m-1)-th partition are assigned to a new cluster with the same plane normal r, before generation of $\mathcal{G}_C$ and pruning. If, however, this newly created cluster ends up being disconnected from the cluster in the (m+1)-th partition (because of pruning in $\mathcal{G}_C$), its nodes are removed from further consideration.

Ground plane. We use the algorithm of [37] to extract, in each image, a region of pixels corresponding to the ground plane. Points in the ground plane region are treated separately than other points. They are matched across subsequent frames, but are never matched with points from outside the ground region. All points in this region are assigned to the same "ground plane" cluster. Unlike [37], we don't use the boundary of the ground plane region to identify vertical planes; rather, ground plane information is relied upon to build a long and consistent cluster chain. This is very helpful for maintaining odometry in the case of discontinuous cluster chains from the side walls.

4.3. Plane-Constrained Bundle Adjustment

After cluster chain construction, the following information is available:

- A set of K planar structures, defined by their one-hot normal vectors ($\mathbf{n}_k$) and distances to the camera at time 0 ($d_{0,k}$).
- A set of image points $\mathbf{x}_{m,i}^k$, where the i-th point in the m-th frame has been associated with the k-th plane.
- A set of matched point chains $\mathcal{M}_j^k = (\mathbf{x}_{m,i_1}^k, \mathbf{x}_{m+1,i_2}^k, \dots)$ that correspond to the maximal paths in the pruned graph $\bar{\mathcal{G}}$. Each $\mathcal{M}_j^k$ represents a single point $\mathbf{p}_j^k$ in space, belonging to the k-th plane, expressed in terms of its image projections.
- A set $(\mathbf{t}_{0\to m})$ of camera locations, and a set $(\mathbf{R}_{0\to m})$ of camera rotations, defined with respect to the reference frame of the first camera.
- A set of identical intrinsic camera matrices $(\mathbf{K}_m)$.

Bundle adjustment (BA) modifies a vector of model parameters (normally, the set of 3-D points, camera poses, and possibly intrinsic parameters) with the goal to ensure that observations are consistent with the model under an appropriate metric. Unlike typical BA, we do not attempt to optimize the location of individuals 3-D points. Rather, we modify the location (but not the orientation) of the K planes to which these points belong.

An image point $\mathbf{x}_{m,i}^k \in \mathcal{M}_j^k$ associated with the plane $(\mathbf{n}_k, d_{0,k})$ defines a 3-D point by the intersection of the line of sight through the $\mathbf{x}_{m,i}^k$ in the m-th camera and the plane. In the reference frame of the first camera, this point can be expressed as:

$$\mathbf{p}_{j,(m,i)}^k = \mathbf{R}_{0\to m}^T \left(\frac{d_{m,k}\, \mathbf{K}_m^{-1} \tilde{\mathbf{x}}_{m,i}^k}{\mathbf{n}_k^T \mathbf{K}_m^{-1} \tilde{\mathbf{x}}_{m,i}^k} - \mathbf{t}_{0\to m} \right) \quad (4)$$

In the equation above, $d_{m,k} = d_{0,k} + (\mathbf{n}_k)^T \mathbf{t}_{0\to m}$ is the distance of the k-th plane to the m-th camera location. $\mathbf{K}_m^{-1} \tilde{\mathbf{x}}_{m,i}^k$ represents the line of sight through the pixel, expressed in terms of the m-th camera frame.

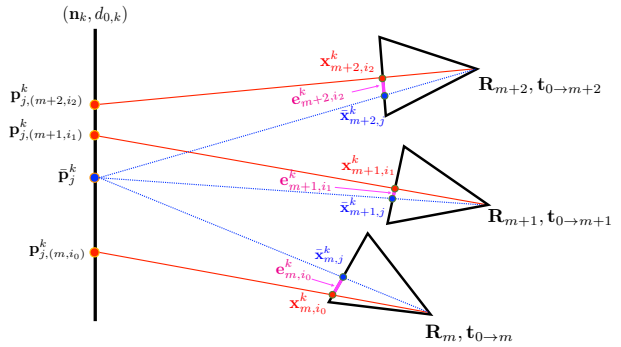

Figure 3: Computation of the reprojecton error for Bundle Adjustment (Sec. 4.3)

Ideally, the lines of sight through all pixels in $\mathcal{M}^k_j$ should intersect at the same point, $\mathbf{p}^k_j$, in the k-th plane. In practice, some amount of dispersion of the points $\{\mathbf{p}^k_{j,(m,i)}\}$ should be expected. Our BA procedure is designed to minimize a measure of such dispersion, defined as follow (see Fig. 3). We first compute the mean over the indices (m, i) in $\mathcal{M}^k_j$ of the back-projected points $\{\mathbf{p}^k_{j,(m,i)}\}$. We then reproject this mean point, $\bar{\mathbf{p}}^k_j$, onto the individual cameras at their estimated poses, obtaining:

$$\bar{\mathbf{x}}^k_{m,j} = EN\left(\mathbf{K}_m(\mathbf{R}_{0\to m}\bar{\mathbf{p}}^k_j + \mathbf{t}_{0\to m})\right) \qquad (5)$$

Finally, we compute the reprojection errors $\mathbf{e}^k_{m,i} = \bar{\mathbf{x}}^k_{m,j} - \mathbf{x}^k_{m,i}$ for all $\mathbf{x}^k_{m,i} \in \mathcal{M}^k_j$. We minimize, over camera poses and plane locations (and, optionally, camera focal lengths), the norm of $\mathbf{e}^k_{m,i}$ as measured by the Huber loss, summed over all planes and all matched point chains.

It is instructive to compare our error criterion with the criterion used by typical reconstruction algorithms that store and optimize individual 3-D point locations. These algorithms reproject the 3-D points onto the cameras and compute the difference with the locations of the associated point features. The position of all 3-D points is adjusted at each iteration. We too produce individual 3-D point locations in output (in the form of backprojected points $\bar{\mathbf{p}}^k_j$), however, we only optimize the position of the visible planes. Much fewer parameters are involved (as there are typically few planes visible), hence computational savings are achieved. More importantly, the resulting 3-D points are guaranteed to lie on the detected planes. This is very important when planar reconstruction is desired. While planes could be fitted post-facto to a 3-D point cloud, our algorithm naturally produces the planar structures containing the 3-D points. This is a critical difference with respect to existing algorithms for bundle adjustment with plane constraints (e.g. [21, 39]). Even while enforcing planar constraints, these algorithms try to estimate either individual 3-D points or 2-D observations, resulting in system with many more variables to optimize compared to ours.

We also found it beneficial to apply a simple cluster merging procedure at the end of BA. Parallel planes that are closer than a certain threshold are merged into one (along with the associated match clusters), unless they are seen at very different times (i.e., there is a large gap between the last time the first plane is seen and the first time the second plane is seen). This operation is repeated greedily until no more planes can be merged. Then, BA is run again on the merged data.

5. Implementation and Experiments

5.1. Implementation Details

Vanishing points detection. Vanishing points are computed using the technique of [34] on line segments detected by the LSD algorithm [9]. Following [15], line segments with length of 20 pixels or more are clustered using T-linkage [23] (implemented on GPU). A candidate set of vanishing points is found, which is then refined by minimizing a form that penalizes geometric discrepancies.

Multi-planar clustering. SIFT features are matched across subsequent frames, then plane-constrained T-linkage (implemented on GPU) is run starting from non-minimal sets of matches identified using the region-based sample selection scheme of [15].

Non-linear minimization. Minimization of (3) as well as of the reprojection errors with Huber loss (Sec. 4.3) is accomplished using the "Schur complement trick" [4] as implemented by the Ceres solver [2] .

Bundle Adjustment sequence. We first run a round of BA optimizing only plane locations and camera locations. Then, the plane merging procedure described at the end of Sec. 4.3 is applied. Finally, another round of BA is run, this time optimizing all parameters (plane locations, camera locations, camera rotation, and optionally focal lengths).

Visualization. The algorithm produces in output a set of (infinite) planes, as well as a number of 3-D points located on these planes. In order to visualize the results, we generate, for each plane, minimal rectangular patches aligned with the Manhattan canonical frame containing all 3-D points identified on the plane. Texture is then projected on these patches from the images, using camera pose information.

Processing times, averaged over 100 runs (20 runs for each of five 20-image-sequence) and divided by task, are shown in Tab. 1. The code ran on a GPU-enabled computer (only vanishing point detection and multi-planar clustering were optimized for GPU), equipped with Intel core i-7 6700 CPU and NVidia GTX 1080 GPU.

SIFT detection + matching	185 ms
Vanishing points detection	223 ms
CNN ground segmentation	22 ms
Multi-planar clustering	322 ms
Translation vector regression	20 ms
Cluster chain computation	30 ms
Bundle Adjustment (focal length fixed)	135 ms
Bundle Adjustment (focal length optimized)	500 ms
Total (focal length fixed)	**937 ms**
Total (focal length optimized)	**1,302 ms**

Table 1: Average processing time per frame.

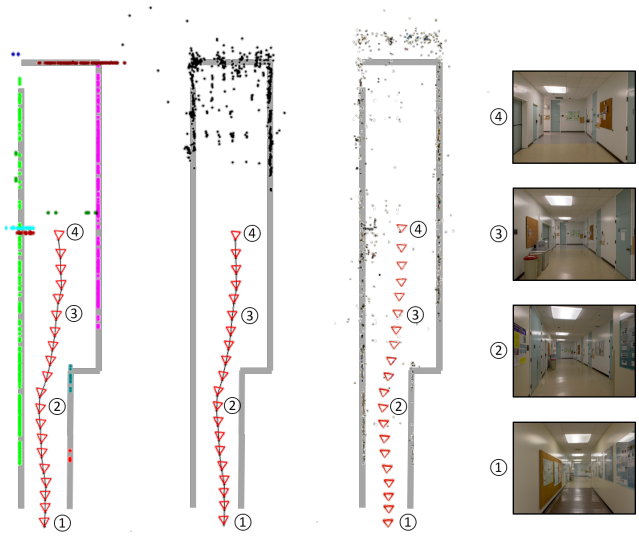

Figure 4: Bird-eye view of reconstructed points from the Corridor 1 sequence. Left: our algorithm (points in different detected planes are shown with different color). Center: ORB-SLAM [26]. Right: SfM Revisited [31]. Note that the sky blue and cinnamon points in our result represent planar surface induced by frontal surfaces of trashcan and printer.

Figure 5: 3-D textured rendering of one of the walls of the reconstructed Corridor 1 scene.

5.2. Experiments

We collected a number of sequences of corridors in our buildings using an iPhone 6 (1024 × 768 pixels). Each sequence contains 20 images, which were taken by hand at each step of walking approximately 0.4 meters of distance from each other. Our reconstructions have been computed without optimization of the camera's focal length. We noticed that optimizing the focal length did not improve the reconstruction results noticeably, while using more processing time (see Table 1). We show reconstruction results (together with the camera poses) for two scenes (Corridor 1 and 2) in Figs. 4 and 6 as bird-eye views of the 3-D points, using different colors for points belonging to different detected planes (Note that points belonging to the horizontal planes were removed from our result, for the clear visualization). For each scene, we also show the result using the open source implementation[2] of the ORB-SLAM algorithm [26], as well as the result using the open source implementation[3] of the "SfM Revisited" algorithm of [31]. The reconstructed points are shown on top of the floor plan, which was manually adjusted in all three cases to best fit the points. In addition, we show a textured 3-D rendering of the planar patches produced by our system for Corridor 1 and 2 in Figs. 5 and 7. Also, Fig. 8 shows the comparison of 3-D textured renderings between our algorithm and SfM Revisited. Different from the point-based reconstruction, our algorithm produced the dense reconstruction even for the regions without textures if such regions are in between the detected feature points on the same wall.

The reconstructed path is very similar for all three algorithms, except for SfM Revisited in the Corridor 2 scene. The most noticeable difference is in the quality of the reconstructed points, which are clearly more sparser in the other two general-purpose algorithms than in ours. The 3-D rendering is particularly impressive considering few input images, and insufficient feature points, which caused by the indoor environment, enabling walk-through visualization of the reconstructed scenes. The 3-D rendering also highlights several reconstruction errors, which may be imputed to incorrect relative scale recovery (resulting in planes with inaccurate location), mismatches (which are amplified by our simple planar patch extrapolation algorithm, generating "ghost" patches), and incorrect camera rotation (resulting in inaccurate image warping, e.g. in correspondence to the large door in the left half of Fig. 5.)

6. Conclusions

We have introduced a technique for motion recovery and surface reconstruction that makes use of the Manhattan World geometry at every step of the way. Our approach relies on pairwise matching of feature points, but represents geometric primitives in terms of planes. This enables a novel formulation of Bundle Adjustment that optimizes plane locations, rather than point locations. The result is

[2] https://github.com/raulmur/ORB_SLAM2
[3] https://github.com/colmap/colmap

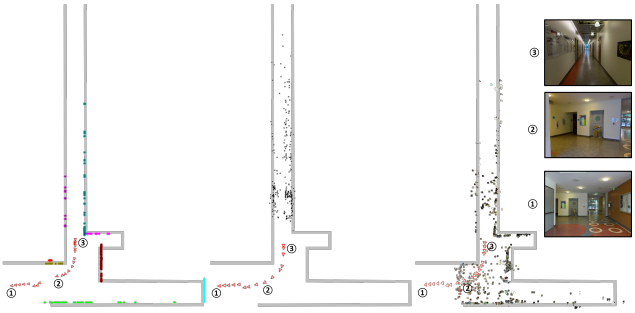

Figure 6: Bird-eye view of reconstructed points from the Corridor 2 sequence. See caption of Fig. 4

Figure 7: 3-D textured rendering of one of the walls of the reconstructed Corridor 2 scene.

(a) (b)

Figure 8: 3-D textured renderings of the reconstructed Corridor 3 scene. (a) our algorithm. (b) SfM Revisited [31] with dense reconstruction option turned on.

expressed in terms of planar structures, a natural representation for indoor scenes.

While we have achieved good results in our experiments, several situations may hamper the quality of reconstruction. First of all, if the scene geometry does not comply with the MW assumption, multiple system components will fail. However, extension of our approach to the less restrictive weak MW geometry is possible, which would expand its domain of applicability. Our system relies on the presence of point features that must be matched across images. As with other feature-based approaches, large textureless areas, as well as multiple specularities, can cause it to fail. Finally, our simple approach for patch-based visualization of the system output, as shown in our figures, is unable to deal with large textureless areas, even when these areas are perfectly modeled by the planes estimated by our algorithm. Improved visualization could be obtained by combining image-based segmentation with the geometric information produced by our system.

Our algorithm has shown very promising results in relatively short sequences with a few dozen images. Further work will be needed to evaluate its performances in very long data sets (including loop closure), as well as in situations with multiple non-planar objects.

References

[1] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011. 1

[2] S. Agarwal, K. Mierle, et al. Ceres solver. http://ceres-solver.org. 6

[3] A. Bartoli and P. Sturm. Constrained structure and motion from multiple uncalibrated views of a piecewise planar scene. *International Journal of Computer Vision*, 52(1):45–64, 2003. 2

[4] D. C. Brown. A solution to the general problem of multiple station analytical stereotriangulation. Technical Report 43, Patrick Airforce Base, 1958. 6

[5] J. M. Coughlan and A. L. Yuille. Manhattan world: Compass direction from a single image by bayesian inference. In *IEEE International Conference on Computer Vision*, volume 2, pages 941–947. IEEE, 1999. 1

[6] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014. 1, 2

[7] W. Förstner and B. P. Wrobel. *Photogrammetric Computer Vision: Statistics, Geometry, Orientation and Reconstruction*. Springer, 2016. 4

[8] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Reconstructing building interiors from images. In *IEEE 12th International Conference on Computer Vision*, pages 80–87. IEEE, 2009. 2

[9] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. LSD: a Line Segment Detector. *Image Processing On Line*, 2012. 6

[10] F. Harary and R. Z. Norman. Some properties of line digraphs. *Rendiconti del Circolo Matematico di Palermo*, 9(2):161–168, 1960. 4

[11] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2003. 2, 3

[12] N. Jiang, D. Lin, M. N. Do, and J. Lu. Direct structure estimation for 3D reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2655–2663, 2015. 2

[13] M. Kaess. Simultaneous localization and mapping with infinite planes. In *IEEE International Conference on Robotics and Automation*, pages 4605–4611. IEEE, 2015. 2

[14] C. Kim and R. Manduchi. Indoor Manhattan spatial layout recovery from monocular videos via line matching. *Com-*

puter Vision and Image Understanding, 157, April 2017. 1, 2

[15] S. Kim and R. Manduchi. Multi-planar fitting in an indoor Manhattan world. In *IEEE Winter Conference on Applications of Computer Vision*, pages 11–19. IEEE, 2017. 1, 2, 3, 6

[16] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234. IEEE, 2007. 2

[17] J. Košecká and W. Zhang. Video compass. In *European Conference on Computer Vision*, pages 476–490. Springer, 2006. 1, 3

[18] N. Lazic, G. Inmar, F. Brendan, and A. Parham. Floss: Facility location for subspace segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 825–832, 2009. 3

[19] P.-H. Le and J. Košecka. Dense piecewise planar RGB-D SLAM for indoor environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4944–4949. IEEE, 2017. 2

[20] D. C. Lee, M. Hebert, and T. Kanade. Geometric reasoning for single image structure recovery. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2136–2143. IEEE, 2009. 3

[21] G. H. Lee, F. Fraundorfer, and M. Pollefeys. MAV visual SLAM with plane constraint. In *IEEE International Conference on Robotics and Automation*, pages 3139–3144. IEEE, 2011. 2, 6

[22] A. Levin and R. Szeliski. Visual odometry and map correlation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2004. 4

[23] L. Magri and A. Fusiello. T-linkage: a continuous relaxation of J-linkage for multi-model fitting. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 3, 6

[24] B. Micusik and H. Wildenauer. Structure from motion with line segments under relaxed endpoint constraints. *International Journal of Computer Vision*, 124(1):65–79, 2017. 2

[25] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Generic and real-time structure from motion using local bundle adjustment. *Image and Vision Computing*, 27(8):1178–1193, 2009. 2

[26] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. 1, 2, 7

[27] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *2011 IEEE International Conference on Computer Vision*, pages 2320–2327. IEEE, 2011. 2

[28] R. Raguram, O. Chum, M. Pollefeys, J. Matas, and J. Frahm. USAC: A universal framework for random sample consensus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):2022–2038, 2013. 3

[29] C. Raposo and J. P. Barreto. π-Match: Monocular vSLAM and piecewise planar reconstruction using fast plane correspondences. In *European Conference on Computer Vision*, pages 380–395. Springer, 2016. 2

[30] 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. 1, 2, 3

[31] J. L. Schonberger and J.-M. Frahm. Structure-from-motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016. 1, 7, 8

[32] R. Szeliski and P. H. Torr. Geometrically constrained structure from motion: Points on planes. In *European Workshop on 3D Structure from Multiple Images of Large-Scale Environments*, pages 171–186. Springer, 1998. 2

[33] Y. Taguchi, Y.-D. Jian, S. Ramalingam, and C. Feng. Point-plane SLAM for hand-held 3D sensors. In *IEEE International Conference on Robotics and Automation*, pages 5182–5189. IEEE, 2013. 2

[34] J. P. Tardif. Non-iterative approach for fast and accurate vanishing point detection. In *International Conference on Computer Vision*, 2009. 6

[35] R. Toldo and A. Fusiello. Robust multiple structures estimation with J-linkage. In *European Conference on Computer Vision*, pages 537–547. Springer, 2008. 3

[36] E. Vincent and R. Laganiére. Detecting planar homographies in an image pair. In *International Symposium on Image and Signal Processing and Analysis*, pages 182–187. IEEE, 2001. 3

[37] S. Yang, D. Maturana, and S. Scherer. Real-time 3D scene layout from a single image using convolutional neural networks. In *IEEE International Conference on Robotics and Automation*, pages 2183–2189. IEEE, 2016. 2, 5

[38] S. Yang, Y. Song, M. Kaess, and S. Scherer. Pop-up slam: semantic monocular plane slam for low-texture environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1222–1229. IEEE, 2016. 2

[39] Z. Zhou, H. Jin, and Y. Ma. Robust plane-based structure from motion. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1482–1489. IEEE, 2012. 2, 6

[40] M. Zuliani, C. S. Kenney, and B. Manjunath. The multi-ransac algorithm and its application to detect planar homographies. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 3, pages III–153. IEEE, 2005. 3