Scene wireframes sketching for Unmanned Aerial Vehicles

Roi Santos, Xose M. Pardo, Xose R. Fdez-Vidal

CiTIUS, University of Santiago de Compostela. Galicia, Spain
Emails: {roi.santos,xose.pardo,xose.vidal}@usc.es

Abstract

This paper introduces novel insights to improve the state-of-the-art line-based unsupervised observation and abstraction models of man-made environments. The increasing use of autonomous UAVs inside buildings and around human-made structures demands new accurate and comprehensive representation of their operation environments. Most of the 3D scene abstraction methods use invariant feature point matching, nevertheless some sparse 3D point clouds do not concisely represent the structure of the environment. The presented approach is based on observation and representation models using the straight line segments.

The goal of the work is a complete method based on the matching of lines, that provides a complementary approach to state-of-the-art methods when facing 3D scene representation of poor texture environments for future autonomous UAV. Oppositely to other recently published methods obtaining 3D line abstractions, the proposed method features 3D segment abstraction in the absence of a previously generated point based reconstruction. Another advantage is the ability to group the resulting 3D lines according to different planes, for exploiting coplanar line intersections. These intersections are used like feature points in the reconstruction process. It has been proved that this method exclusively based on lines can obtain spatial information in the adverse situations when a SIFT-like SfM pipeline fails to generate a dense point cloud.

Keywords: 3D abstraction, reconstruction, line-based sketching, UAV.

1. Introduction

Modern unmanned aerial vehicles (UAV), commonly known as drones, feature on-board video cameras that stream real time visual information about their environment. These frames captured by UAVs need to be processed to allow a feasible understanding of the environment in real time. Fetching an usable representation of the scene is crucial, and the result compromises the success of ongoing mission specific analysis. A proper representation of the environment eases the programming of autonomous movement routines, by identifying its location in space, finding obstacle-free trajectories, or learning movement routines. In this work, we consider the first stage of computing structural 3D information for an autonomous drone from its own video captures or, in general, for multiple views of an environment of interest.

The vast majority of the current approaches for 3D scene reconstruction are based on point clouds [1]. Commonly, points are matched between pairs of views based on their descriptors, then triangulated [2] to make an initial estimation of their location in 3D space, and finally their poses are adjusted by least squares minimization [3]. We can distinguish three common stages of a reconstruction process: The first stage is the estimation of camera poses by Structure-from-Motion (SfM), which outputs the estimated camera poses and reconstructed features. The second step is the computationally more expensive Multi-View Stereo (MVS) [4] [5] that can add up to millions of points into a dense cloud. The third stage is intended for post processing, and based on fitting more complex 3D elements to the estimated point clouds, including planes [6] [7] or lines [8] [9]. However, these approaches are heavily dependent on the dense point clouds, and will not permit a feasible solution when a dense point cloud is not available. That applies in the case of UAVs, when an on-flight real time reconstruction is required, because expensive adjustments can not be delivered on time, when the video stream lacks high definition, the digital noise is persistent, or the received images have to be converted from an analog video source. Even with the appropriate setup, conditions related to poor texture of

the surrounding objects, a poor environment illumination, blurring or the lack of sufficient available high definition pictures of the scene when the vehicle is moving fast, might compromise the building of a dense cloud.

The logical evolution of the environment abstraction from multiple views is to incorporate line-based pipelines that do not require a detailed point-based description of the areas of interest, and are resilient to difficulties on the point cloud creation. Point based 3D reconstructions may be accurate when created from high definition frames and after expensive bundle adjustments. Nevertheless, in addition to provide geometrical information richer than interest points, lines and planes do summarize the limits of man-made structures. Besides, coplanar line primitives can be intersected to further reveal spatial information. These advantages make lines a good candidate to team with point feature detectors and descriptors [10] [8] [11]. They offer the possibility of combining individual similarities of pairs of segments, related to strong constraints of parallelism and orthogonality [12, 13], and specially coplanarity constraints [14]. This paper exploits the latter ones, and the inlier groups are determined based on the homography between different views. The constraint that can be exploited is that the intersection of a pair of coplanar lines, even if it might not resemble a physical point, is still geometrically invariant under perspective projection.

Our approach is rooted on line based abstraction in the absence of high density point clouds [15], and it takes advantage of the robustness of a state-of-the-art line matching algorithm that works among pairs of images [16]. The inputs are both the images and the camera intrinsic parameters, and the outputs are both the camera extrinsics and the line based 3D sketch. Line detection and matching algorithms are prone to errors, so the system is designed for outliers rejection, is robust to changes in illumination conditions, blurring due fast camera movements, and also capable to real-time reconstruction from streamed video.

1.1. Problem formulation and our approach

A 3D reconstruction, abstraction or spatial sketch is the estimation of the position of singular primitives captured in several images, by using geometrical relationships and minimization adjustments. The first problem is to filter the geometrical features that have been detected and put them in correspondence among views, and segment them into clusters. The second task is to apply geometrical projection of line endpoints with just two views by assuming a Gaussian noise model for perturbation of their location [2]. The third objective involves adjustment algorithms to fetch all the camera poses and estimate the whole environment abstraction. The forth and last stage is to use the adjusted spatial configuration to improve the abstraction itself, attending to structural properties. The main hypothesis of this work is that straight segments resemble a source of geometrical information that can be employed, independently of dense point clouds, for creating spatial 3D reconstructions. The premise was to use line matchings as the cross-view relationship to develop the abstraction on, oppositely to fitting these primitives to described feature points [10]. Our method does not assume any Manhattan configuration, this means that no alignment is performed according to any preferred direction. The model is engineered considering the most common case for UAV of unknown camera poses. Therefore, camera extrinsics are not provided for the line matching process, neither any parameter or limitation for the degrees of freedom that could ease their estimation. Camera pose must also be recovered through an SfM process. On the other hand, just the camera intrinsic parameters and the target images are required for the abstraction. The main result of this work is a set of improvements for the complete sequence of algorithms required for 3D abstraction using lines, mainly focused on extracting the most information from the images.

The goal of this work is to obtain a real-time three dimensional representation of a scene by using a limited number of matched straight segments. The optimal abstraction the authors look for is built by meaningful descriptive lines whose resemble the limits of the 3D planes of the scene, avoiding representations of several redundant atomic short segments. In order to assure that

we improved the state-of-the-art scene abstraction based on lines, we need to prove: Firstly, the implemented multi-scale line detection, and the structure based matching algorithms are profitable in poorly illumined environments and low texture scenes. Secondly, the proposed exploitation of intersections brings valuable information into the least-squares adjustment. For the first task, the observed lines are firstly detected in frames, and then put in correspondence, dragging outliers during the matching process. Our approach takes advantage of multi-scale line detection and matching [16] to increase the accuracy of the line endpoints triangulation among pairs of line-matched frames. Secondly, our method goes one step ahead in the least squares adjustment of cameras and lines by exploiting geometrical relationships of the coplanar lines. After classifying the spatial lines according to their co-planarity, the intersection of the observed lines are brought into a second run of the least squares adjustment of cameras and lines.

1.2. Related work based on lines

Lines need to be detected, and optionally, junctions [17]. Segments can be put in correspondence by characterizing their appearance and structure [18, 19]. Sparse Bundle Adjustment (SBA) is a method for simultaneously optimizing a set of camera poses and visible points, based on minimization of an objective function. The input in the case of lines can be the endpoints of the matched lines or the probability distribution of their location shen2000uncertainty. [20] goes an step forward and brings up the Plücker coordinates 4-parameter representation of lines, for the will of a more suitable least-squares based optimization SBA. They also include the trifocal tensor [21] because they just use lines as inputs. Same as did in [8], where in addition avoided enforcing Plücker constraints to improve the SBA efficiency, by employing the Cayley representation of the Plücker coordinates. Their cost function computes the squared re-projection error from the sum of the squared shortest distance from each observed endpoint to the reprojected infinite line. Several approaches base the line reconstruction on previous results of a feature point based SfM pipeline. In this group, [22]

match these segments to other images according to the distances from their back-projected endpoints to neighboring observed line endpoints. This is also the case of the stereo line reconstruction introduced in [23], based on block matching. This work is able to deal with both lines and feature points, and uses RANSAC for outliers rejection. [10] teamed the accurate feature point detector and descriptor SIFT with a line detection based on LSD [24], and matching based on the mentioned previously run point-based SfM. Camera extrinsics are an input to the algorithm, and are obtained from the third-party feature-point based reconstruction before matching the lines [10]. The limitation is the requirement of a dense SfM reconstruction as a base for the spatial lines abstraction and the camera extrinsics given as inputs. They define a line cluster as a group of spatial lines discriminated by spatial proximity in the reconstruction, and the segmentation is done by projecting from the input camera poses. They also implemented a cost function for the SBA optimization similar to that presented in [8]. On the other hand there are other approaches based on geometric relationships on the structure of the lines proposing to locate discrete point counterparts on a line over different views [25]. The present work is framed in the group of studies that are not dependant of feature point based SfM pipelines, and uses 4 parameters for updating the line representation during the non-linear optimization.

1.3. Main contributions

We propose a method with a set of improvements over state-of-the-art algorithms for automatic 3D scene abstraction based on lines:

1. The proposed method does not require a dense point cloud. It makes feasible a reconstruction based exclusively on line matchings, performed independently over pairs of images. Our approach estimates camera extrinsics, but does not root the line matching on these spatial projections, preventing the uncertainty from SfM to propagate and merge with the uncertainty related to 2D line detection and matching. Previous reconstruction methods rely on a third party

SfM pipeline [23] [10], providing the camera poses and dense point clouds, to base line matching and 3D reconstruction on.

2. We propose to segment the set of spatial lines by fitting them to different planes through RANSAC. The observed intersections of 2D co-planar lines are therefore described according to the observed matched lines. These segmented group of intersections are projected from every camera plane. They are finally included into the cost function for a second SBA run, taking advantage of this accurate source of observed points in correspondence. Most of the published methods are intended for urban environments, where many lines are coplanar. Nevertheless, they do not retrieve additional information from the images according to the spatial structure. Hence, the projected lines use to be the sole primitive input to the cost function for a least-squares minimization.

3. Our proposed *Line Observation* is built by merging independent line matchings over pairs of scale-space images. These groups of matched 2D lines define unique global entities before the 3D stages of reconstruction, in order to avoid the problem of redundant lines and to reduce the number of matching outliers when dealing with multiple views. On the other hand, some recent methods verify the matching candidates based on their support on neighboring views [10], for later clustering them based on their spatial proximity, instead of performing a global matching of the observed lines individually. This is a source of uncertainty, as the matching criteria is tied to the accuracy of every camera pose that was adjusted with point-based SfM, and used as input.

The current implementation include the following differences compared to the previous methods:

1. The whole set of algorithms is implemented into a ROS node that includes capturing the streamed video from a commercial drone. This allows experimental testing without the requirement of external image processing pipelines. To our knowledge, there are no other line based reconstruction pipelines completely integrated in ROS. It runs on real-time up to approximately 800×600 resolution,

depending on how many images are stored in the buffer for the reconstruction. The illumination conditions and number of edges on the image are also crucial for the processing times.

2. The complete sequence of algorithms runs without a human in the loop, and the process goes from the detection of lines to the output of the scene abstraction described by a set of 3D line segments. It can also run in real-time if image resolutions are not big. It must be stressed that several published works employ human assistance, mainly during the line matching stage [26], or need the input of camera poses and an external point feature based SfM pipeline, avoiding the feasibility of a real-time execution [10, 8].

The rest of the paper is structured as follows. A high level overview of our method along with the definitions and problems to solve are presented in Section II. Section III goes through the low level details of the method, explanation of algorithms and the implications. In Section IV, the embedding of intersections of coplanar lines and their contribution to the improvement of the 3D abstraction is described. Experimental results are presented and discussed in Section V. Finally, Section VI presents the main conclusions of this work.

2. High level overview

The architecture of the line sketching model comprises three layers as shown in Fig. 1. i) The *2D layer* extracts and unifies the observed projections of edges from the set of views of a scene, where in general there might be large camera translations and rotations among the different views. ii) The *3D abstraction layer* builds stereo subsystems for every possible pair of views, which are fed into a first SBA, that outputs a 3D wireframe-based sketch and the camera poses, valid up to scale. iii) Finally, the *optimization layer* detects the intersections of the coplanar lines in the sourced images and add them to the least squares optimization for the sake of a more accurate reconstruction.

Line detection and matching between pairs of views are performed in the *2D layer* as described in [16]. Segments that remain unmatched are definitely

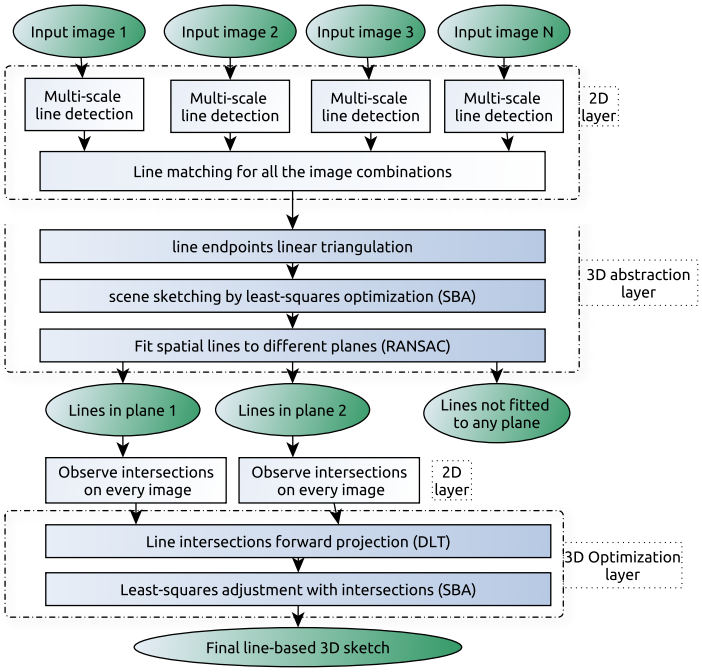


Figure 1: Flow for the proposed method.

discarded. Scene abstraction will be based on these initial pair-wise correspondences of line segments.

Given an unordered set of images, related to a set of unknown camera extrinsics Υ=$\{\Upsilon^1,\Upsilon^2,\Upsilon^3,...,\Upsilon^M\}$, the *3D abstraction layer* computes the initial estimations for each 3D line segment in the set Γ=$\{\Gamma_1, \Gamma_2, \Gamma_3, ..., \Gamma_N\}$. These are estimated from two or more observed projections, this is why Γ_i may firstly embed one or more estimations of the 3D line before they are unified into a single estimation. The set of line projections observed in Υ is represented as l=$\{l_1^1, l_2^1, ..., l_N^1, ..., l_N^M\}$. A Line Observation is defined as a subgroup of the set of line observations l that are the projections of the same 3D line Γ_i. Therefore, $L_i = \{l_i^1, l_i^2, ..., l_i^M\}$. The set of Line Observations is denoted by L=$\{L_1, L_2, ..., L_N\}$.

The 3D line estimations Γ are obtained by linear triangulation of the end-

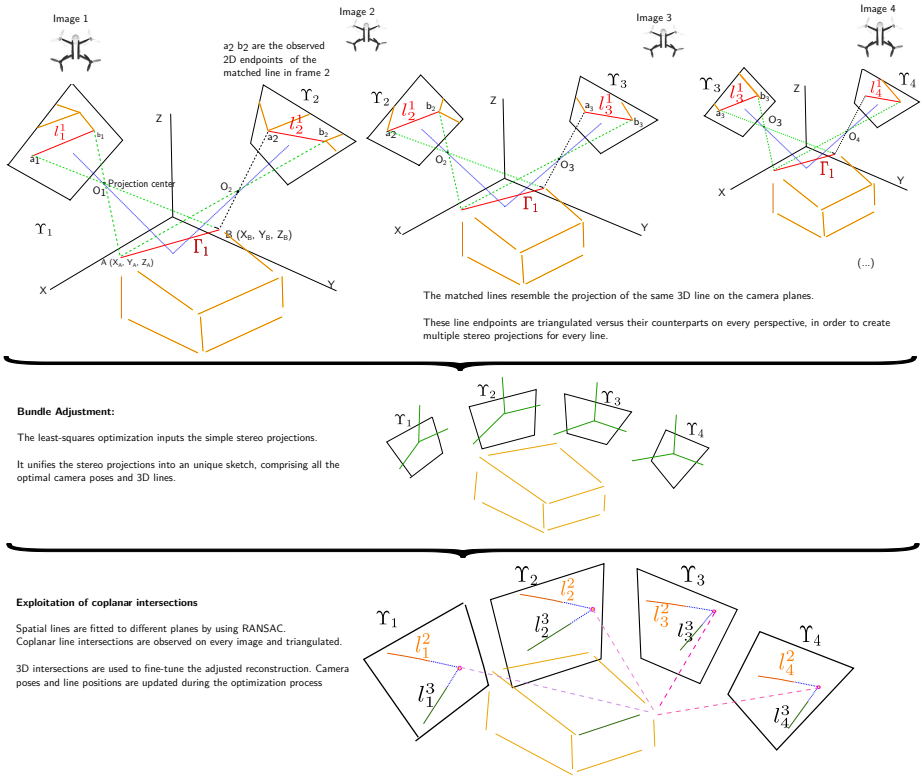


Figure 2: Reconstruction after unifying the observed projections of lines. The pairwise triangulations of endpoints are the input of the adjustment process. The output is the 3D sketch featuring the camera poses and the spatial lines.

points of L from pairs of camera planes in Υ, as can be done using the Direct Linear Transformation (DLT) method [21]. This is represented on the upper region of Fig. 2 and on the middle region of the flow diagram Fig. 1.

The 3D stereo system $\{\Upsilon,\Gamma\}$ generated by linear triangulation is used as input to the *3D abstraction layer*. The cameras Υ are sequentially bundled in the same reference frame, by stacking the new ones according to the L-to-Γ correspondences computed in the previous stereo pair of cameras. The estimations for each $\{\Gamma_i\}$ is unified as one unique line, computed as the center of gravity of all the embed estimations for the 3D line. This is shown on the middle region of Fig. 2.

The least-squares optimization named SBA is based on the Levenberg–Marquardt algorithm, and its inputs are the estimated camera extrinsics (including the camera planes Υ) and the set Γ, now containing unique estimations for each 3D line. This process is visually explained on the lower region of Fig. 2 and is embed in the "SBA" block in the flow diagram Fig. 1.

The *optimization layer* fits the spatial line segments Γ to a set of different planes $\mathcal{P}$. Γ is therefore segmented into different groups according to the planes $\mathcal{P}$, and so is done with their projections L. It is drawn in the lower region of Fig. 1. The group of Line Observations fitted to the plane $\boldsymbol{\mathcal{P}}_t$ is noted as F_t. The intersections of the coplanar lines F_t on the camera plane Υ^j is the set of points $\mathcal{T}_t^j$. These intersections are observed on the input images, and described like a feature point, being the descriptor the pair of coplanar lines drawing it. Like was performed in the *3D abstraction layer* with the endpoints of l, now the point correspondences in $\mathcal{T}_t^j$ are fed into the linear triangulation algorithm, in order to create initial estimates for the 3D intersections by forward projecting $\mathcal{T}_t^j$. The set of estimations for the 3D intersections is a sparse cloud and is denoted as $\mathcal{R}$.

The 3D intersections $\mathcal{R}$ enter the least-squares optimization. These new iterations of SBA are based on a simple 3D point reprojection distance to the observations on Υ. The outputs are the new optimized estimations for Υ and the optimal 3D intersections $\mathcal{R}$. The line poses are corrected by forward projecting them from the newly estimated camera planes Υ, generating the final sketch $\{\Upsilon,\Gamma\}$.

3. Low level description

The 3D line based sketch $\{\Upsilon,\Gamma\}$ is built based on the knowledge of correspondences among observed lines l and the intrinsics of all the cameras. These observed lines are obtained from edge detection on scale-spaces, and this is a novelty of the present method compared to other recently published ones [10, 8, 20]. This detection method brings up the number of image scales where the line was

detected, an additional segment appearance attribute that is used in the matching algorithm. The practical consequence for the matching of the observed lines, is that a segment with inconsistent endpoint location among scales can be avoided, as these lines might introduce uncertainty in the camera extrinsics. Segment matching is performed between pairs of pictures because each couple of images have a different set of detected lines in common. Correspondences are made by characterizing not just individual lines, but also their group of neighbor segments, and these neighborhoods of detected lines will be different for each image. An iterative process goes through the pairwise correspondences, and puts together all the lines marked as counterparts into an unique instance of a multi-image spatial segment. The process ends when all the segments that are related get into the same instance. Therefore, an unique 3D segment will be computed from all the matched lines in each instance.

The first problem to solve for the computation of a 3D sketch from the matched observations is that the camera poses P are unknown. We have chosen to estimate them from the endpoint correspondences of l. For this task we first estimate the Essential matrix E for each camera pair, by using the Five-Point Algorithm [27], and RANSAC for hypothesis generation. Having E and l, the relative camera rotation and translation among the first pair of cameras $\boldsymbol{P}^j = \boldsymbol{K} \times [\boldsymbol{R}|\boldsymbol{t}]$ are estimated using cheirality check [27] and discarding the triangulated endpoints that are not in front of the cameras. The left camera is provided with the pose $\boldsymbol{P}^1 = \boldsymbol{K} \times [\boldsymbol{I}|0]$, and the rest of cameras will be stacked from this position in the world reference frame. Once we have the camera matrices for the first pair of cameras, a linear triangulation method [21] can be used to retrieve the first estimations for 3D lines, i.e. the members of Γ with observed counterpart on both camera planes.

The inaccuracy in the observation of the 2D lines l implies that there will not be a 3D point $\boldsymbol{X}$ which exactly satisfies that their projections on cameras Υ^1 and Υ^2 are $\boldsymbol{x_1} = \boldsymbol{P}^1\boldsymbol{X}$, $\boldsymbol{x_2} = \boldsymbol{P}^2\boldsymbol{X}$ respectively, and that the image points do not satisfy the epipolar constraint $\boldsymbol{x_2}\boldsymbol{F}\boldsymbol{x_1} = 0$. Therefore, a projective-invariant triangulation method, that only minimizes the image distances to the

observations, is required. A linear triangulation [21] method does not depend on the projective frame in which $\boldsymbol{X}$ is defined.

The linear triangulation is performed in homogeneous coordinates because it allows to consider line endpoints in the infinite.The execution order for this DLT-based method over the pairs of cameras obeys to the inlier ratio of line matching, i.e. the number of matched lines divided by the total number of detected lines for each camera pair. Therefore, the linear triangulation of line segment endpoints starts with the pair of cameras with the highest inlier ratio, noted as $\{\Upsilon^a, \Upsilon^b\}$. Next, the camera Υ^c is chosen among the ones with the higher inlier ratio of line matching with Υ^a and Υ^b. At successive steps, every new camera Υ^j is chosen among the ones with the higher inlier ratio of line matching with previous selected cameras.

For linear triangulation it is required a set of observed line correspondences, $\boldsymbol{l}_j^m$ to $\boldsymbol{l}_j^n$. This set of 2D features are undistorted and translated as coordinates on the camera plane. The projection on the image plane of camera Υ^m of the endpoint $\boldsymbol{X}_{i,a}$ of the spatial line $\boldsymbol{\Gamma}_i$ is $\boldsymbol{x}_{i,a}^m = \boldsymbol{P}^m \boldsymbol{X}_{i,a}$. This point on Υ^m is matched to its counterpart on the Υ^n, noted as $\boldsymbol{x}_{i,a}^n = \boldsymbol{P}^n \boldsymbol{X}_{i,a}$. Both equations can be combined into $\boldsymbol{A}\boldsymbol{X}_{i,a} = 0$, where $\boldsymbol{A}$ is the matrix of equation coefficients. It is built from the matrix rows $\boldsymbol{A}_r$ contributed from each correspondence, whose resemble the movement of each line between both views. $\boldsymbol{X_{i,a}}$ contains the unknowns for the 3D line endpoint position. The observed line endpoints $\boldsymbol{x}_i$ on the camera planes are in homogeneous coordinates, but the homogeneous scale factor is eliminated by a cross product, to give three equations for each endpoint, of which two are linearly independent. By using the cross product for the camera Υ^m: $\boldsymbol{l}_i^m \times (\boldsymbol{P}^m \boldsymbol{X}_{i,a}) = 0$, analogously as followed in [21], it is obtained

$$x_m(\boldsymbol{p_m^{3T}} \boldsymbol{X_{i,a}}) - (\boldsymbol{p_m^{1T}} \boldsymbol{X_{i,a}}) = 0, \tag{1}$$

$$y_m(\boldsymbol{p}_m^{3T} \boldsymbol{X}_{i,a}) - (\boldsymbol{p}_m^{2T} \boldsymbol{X}_{i,a}) = 0, \tag{2}$$

$$x_m(\boldsymbol{p}_m^{2T} \boldsymbol{X}_{i,a}) - y_m(\boldsymbol{p}^{1T} \boldsymbol{X}_{i,a}) = 0, \tag{3}$$

where (x_m, y_m) and (x_n, y_n) are the coordinates of $\boldsymbol{x}_{i,a}^m$ and $\boldsymbol{x}_{i,a}^n$ respectively. $\boldsymbol{p}_m^{rT}$

is the r-th row of $\boldsymbol{P}^m$. It can be decomposed similarly for $\boldsymbol{P}^n$, and compose the equation of the form $\boldsymbol{A}\boldsymbol{X}_{i,a} = 0$. Solving:

$$\boldsymbol{A} = \begin{bmatrix} x_m \boldsymbol{p}_m^{3T} - \boldsymbol{p}_m^{1T} \\ y_m \boldsymbol{p}_m^{3T} - \boldsymbol{p}_m^{2T} \\ x_n \boldsymbol{p}_n^{3T} - \boldsymbol{p}_n^{1T} \\ y_n \boldsymbol{p}_n^{3T} - \boldsymbol{p}_n^{2T} \end{bmatrix}. \tag{4}$$

The solution for the 4 equations of the over-determined problem (four equations for four homogeneous variables) is only valid up to scale. The set of points in space mapping to a 3D line $\boldsymbol{\Gamma_i}$ via $\boldsymbol{P}^m$, is the plane $\boldsymbol{P}^m\boldsymbol{\Gamma}_i$.

The result of the linear triangulation process is $\boldsymbol{\Gamma_i}$ and $\boldsymbol{v^j}$, represented in cartesian coordinates. The next subsection will cover the change of coordinate system that will allow an efficient least-squares optimization for the pose of the lines and cameras.

3.1. SBA optimization cost function

The Plücker coordinates are employed as a middle stage for moving into the Cayley representation. The objective of changing to a 4-parameters Cayley representation is to profit from the properties of the spin tensor and the Cayley transformation. A spin tensor is an skew-symmetric matrix, it resembles an infinitesimal rotation, and is useful in the construction of approximations for moderate rotations. In the present method it is employed to update the parameters for each line during the least-squares optimization process.

A 3D line $\boldsymbol{\Gamma_i}$ is represented in Plücker coordinates, following [21, 20], as

$$\boldsymbol{\Gamma_i} = (\hat{\mathbf{u}}_i, \boldsymbol{t_i}), \tag{5}$$

$$\hat{\mathbf{u}}_i = \frac{\boldsymbol{X_{i,0}} - \boldsymbol{X_{i,1}}}{\|\boldsymbol{X_{i,0}} - \boldsymbol{X_{i,1}}\|}, \tag{6}$$

$$\boldsymbol{t_i} = \boldsymbol{X_{i,d}} \times \hat{\mathbf{u}}_i, \tag{7}$$

where $\hat{\mathbf{u}}_i$ is the unit vector in the direction of the line, $\boldsymbol{X_{i,0}}$ and $\boldsymbol{X_{i,1}}$ are any two points on the 3D segment, e.g., its endpoints, and $\boldsymbol{t_i}$ is the moment vector around any 3D point $\boldsymbol{X_{i,d}}$ laying on $\boldsymbol{\Gamma}_i$. As this moment vector $\boldsymbol{t_i}$ is

independent of the choice of $\boldsymbol{X}_{i,d}$ on the segment, we have used the center of gravity for the group of forward projections obtained by linear triangulation with all the camera pairs. Now we are ready to move into the Cayley representation for the 3D lines $\mathbf{\Gamma}_i$, the transformation is the same as employed by [8].

For each reprojection of a 3D line Γ_i on a camera plane Υ^j, the center of projection for Υ^j had been previously subtracted from the coordinates of the reprojection, in order to compare with the observed line. In order to avoid the singularity around the center of the image plane, the error distance for each line is chosen as the squared shortest distance from observed endpoints to the reprojected infinite line, as suggested by [8]. This squared distance is $\{\epsilon_i^j\}^2$:

$$\{\epsilon_i^j\}^2 = \{\{d_A\}_i^j\}^2 + \{d_B\}_i^j\}^2 \tag{8}$$

$$\{d_A\}_i^j = d(proj_{\mathbf{\Upsilon^j}}\mathbf{\Gamma_i}, A_i^j) \tag{9}$$

$$\{d_B\}_i^j = d(proj_{\mathbf{\Upsilon^j}}\mathbf{\Gamma_i}, B_i^j). \tag{10}$$

In Eq. 8 the measure ϵ_i^j is used to generate the residuals during the SBA. $proj_{\mathbf{\Upsilon^j}}\mathbf{\Gamma_i}$ is the reprojection of the infinite line in the direction of $\mathbf{\Gamma}_i$ over the camera plane Υ^j

This SBA process is performed incrementally, starting with the pair of views with the lowest rate of matching inliers, as suggested in [28], and then sequentially adding the remaining cameras with their respective DLT results. After each camera has been added to the set of views to be adjusted, the algorithm executes a line-based SBA run, based on the cameras that have been added so far, and the lines that have counterpart on these views. The process explained in this subsection is expressed in the first loop of Algorithm 1. The output of the algorithm will be the estimated optimal pose of 3D line segments Γ, and the optimal poses for the cameras (Υ).

4. Line intersections

Any line segment detection process is prone to misalignments, wrong segmentation, incorrect placement of endpoints and also cases of unsuited merging

of two curved edges resembling a real straight line. Likewise, some line endpoints are neither well defined in an image, when resembling the edges of shadows, or over-exposed photos. Even with these non optimal conditions, edge segments might be correctly matched on several pairs of view. While this kind of inlier matchings would be welcome in a comparison of matching algorithms, they introduce uncertainty in the location of endpoints set used for the spatial abstraction. The application of the SBA returns the optimized poses of cameras and line segments in 3D, but further improvement can be achieved by exploiting coplanarity constraints.

The goal of the *optimization layer* is to extract structural information from groups of coplanar lines and exploit them to improve the accuracy of camera poses and the placement of 3D lines. This is a novelty of this work, and its main contribution. This approach is rooted on the hypothesis that coplanar line intersections encode accurate geometric information for the reconstruction. The points we are looking for are not just joints of segments touching each other, but also more relevant intersections of the extensions of long structural segments.

The point of intersection of coplanar lines is computed on the original 2D input frames, once we have computed the first 3D line based abstraction as followed in Section 3. The 3D estimations for the lines Γ are fitted to a set planes $\mathcal{P} = \{\boldsymbol{\mathcal{P}}^1, \boldsymbol{\mathcal{P}}^2, ..., \boldsymbol{\mathcal{P}}^p\}$, being p the total number of planes. Two spatial lines $\mathbf{\Gamma_1}$ and $\mathbf{\Gamma_2}$ in Plucker coordinates are coplanar if and only if the reciprocal product of their Plucker coordinates is zero:

$$(\hat{\mathbf{u}}_1, \boldsymbol{t}_1) * (\hat{\mathbf{u}}_2, \boldsymbol{t}_2) = \hat{\mathbf{u}}_1 \cdot \boldsymbol{t}_2 + \hat{\mathbf{u}}_2 \cdot \boldsymbol{t}_1 = 0. \tag{11}$$

The group of coplanar Line Observations F_t, fitted to the plane t, contains all their counterparts on all the camera planes extended and intersected. The intersection of the observations $\boldsymbol{l}_i^j$ and $\boldsymbol{l}_{ii}^j$ is denoted as $\boldsymbol{\mathcal{T}}_{i,ii}^j = \{\boldsymbol{l}_i^j \cap \boldsymbol{l}_{ii}^j\}$. The complete group of observed intersections of F_t is $\mathcal{T}_t$. This is expressed right after the first loop in Algorithm 1.

Each element in the set of observed coplanar intersections $\mathcal{T}$ is dealt as a

point feature described by the pair of Line Observations that originate it, and put in correspondence on all images with counterparts for both original Line Observations. The first 3D estimations for the intersection points on all images $\mathcal{T}$ are obtained by linear triangulation, and denoted with $\mathcal{Z}$. This is done analogously as would be computed with described feature points, as followed in Algorithm 1 inside the second cluster of loops. For this method, every 2D line intersection is eligible to enter the spatial abstraction if and only if it matches a proximity rule and an structural threshold. The first rule limits the sum of the distances from the intersection to the segments to be less than 1.5 times the sum of lengths of both lines, in order to avoid short segments with inaccurate slope to intersect far from them and with high uncertainty. The second threshold is for the angle drawn by both lines, required to be more than red15$^{\text{o}}$deg for the sake of accuracy. Finally these estimations are used as input for the SBA.

4.1. SBA with lines and planes

The second run of the SBA employs a cost function that inputs the 3D estimations for the line intersections $\mathcal{Z}$, in addition to the previously optimized camera extrinsic parameters and lines. The squared cost that will be used to compute the residuals is expressed in the last cluster of loops in Algorithm 1. For this optimization process, the residuals are computed for each iteration by using the squared cost like a simple problem of feature point reprojection error:

$$\{\epsilon_k^j\}^2 = d(proj_{\mathbf{\Upsilon^j}}\{\mathcal{Z}_\mathbf{k}\}, \boldsymbol{\mathcal{T}}_k^j)^2, \tag{12}$$

where $proj_{\mathbf{\Upsilon^j}}\{\mathcal{Z}_\mathbf{k}\}$ is the reprojection of the 3D intersection $\mathcal{Z}_k$ over the camera plane Υ^j, and $\boldsymbol{\mathcal{T}}_k^j$ is the location of the detected intersection in frame j . $d(\cdot,\cdot)$ denotes the 2D Euclidean distance. An estimation of the location of a reconstructed intersections $\mathcal{Z}_k$ has been experimentally proven to be accurate enough, when verifying the following conditions:

1. Both lines are not parallel, which can be guaranteed using a threshold of $\theta > \pi/8$ for the intersection angle. In Plucker coordinates, Γ_a and Γ_b being

Algorithm 1: Line based sketching

Data: L, Camera intrinsics

Result: Υ , Γ, and $\mathcal{T}$

initialization;

Pairwise Linear Triangulation for $L \rightarrow \Gamma$

for $\boldsymbol{\Upsilon}^j \in \Upsilon$ **do**

  for $\boldsymbol{\Gamma}_i \in \Gamma$ **do**

    Add cost function with:

    $\{\epsilon_i^j\}^2 = d(proj_{\boldsymbol{\Upsilon^j}}\boldsymbol{\Gamma_i}, A_i^j)^2 + d(proj_{\boldsymbol{\Upsilon^j}}\boldsymbol{\Gamma_i}, B_i^j)^2$, with $\{A_i^j, B_i^j\} \in \boldsymbol{L}_i$

    observed endpoints of $\boldsymbol{L}_i$ on Υ^j

  end

  if $j >= 2$ **then**

    Solve SBA for Υ , Γ

  end

end

Fit Γ to several planes $\mathcal{P} \rightarrow \mathcal{F}$

for $\boldsymbol{\Upsilon}^j \in \Upsilon$ **do**

  for $\boldsymbol{\mathcal{P}}_h \in \mathcal{P}$ **do**

    for $\{\boldsymbol{l}_i^j, \boldsymbol{l}_{ii}^j\}_h \in \mathcal{F}_h$ **do**

      if $\{\boldsymbol{\mathcal{T}}_h = \{\boldsymbol{l}_i^j \cap \boldsymbol{l}_{ii}^j\} \in \mathcal{T}$,

      $\boldsymbol{l}_i^j \nparallel \boldsymbol{l}_{ii}^j$, $\theta(\boldsymbol{l}_i^j, \boldsymbol{l}_{ii}^j) > \pi/8\}$ **then**

        Add $\boldsymbol{\mathcal{T}}_h$ for Linear Triangulation

      end

    end

  end

end

Pairwise Linear Triangulation for $\mathcal{T} \rightarrow \mathcal{Z}$

for $\boldsymbol{\Upsilon}^j \in \Upsilon$ **do**

  for $\mathcal{Z}_k \in \mathcal{Z}$ **do**

    Add cost function with:

    $\{\epsilon_k^j\}^2 = d(proj_{\boldsymbol{\Upsilon^j}}\{\mathcal{Z}_{\mathbf{k}}\}, \boldsymbol{\mathcal{T}}_k)^2$

  end

end

Solve SBA for Υ , $\mathcal{Z}$

Forward-project L from $\Upsilon \rightarrow$ optimized Γ

parallel implies:

$$\hat{\Gamma}_a = \pm\hat{\Gamma}_b. \tag{13}$$

2. The perpendicular distance from a plane to each intersection is not greater than the average perpendicular distance from the planar surface to the lines associated to it:

$$d_\perp(\mathcal{P}_p, \mathcal{Z}_k) < 1/N_p \sum_k d_\perp(\mathcal{P}_p, \Gamma_k), \tag{14}$$

being Γ_k a 3D line fitted to the plane $\mathcal{P}_p$, and N_p the number of lines associated to plane $\mathcal{P}_p$.

After the convergence of the optimization process, the lines Γ are forward projected from the newly optimized camera planes Υ. The 3D intersections $\mathcal{Z}$ can be represented alongside the lines Γ.

The implementation has been done in ROS by using C++ and the CERES library [1] for bundle adjustment, and holds real-time performance with a compressed video stream of 640×368.

5. Experimental results

The experimental section of this work is willing to test the proposed line based reconstruction method and compare it against other methods under the same conditions. Our fully automatic approach performs all the processes from image datasets or video frames capture, through line detection and matching, to camera pose estimation and 3D abstraction. This section is willing to prove the following facts:

1. The proposed method, if based exclusively on lines, returns the pose of the cameras and a line-based reconstruction of the scene up to scale. An analysis compares the number of 3D segments against the number of 3D points obtained with a SIFT-based SfM [29]. Based on this point cloud it is computed the result for the method Line3D++ [10], and this line based abstraction is also

[1] http://ceres-solver.org

compared with the result of the proposed method. In these test cases the proposed method is not using any point-based SfM pipeline, but instead it just employs self detection and matching of lines. These are described in Sec. 5.1.

2. The proposed method improves the results of feature point based SfM pipelines in various scenarios. For proving this fact, the presented method is teamed with SfM pipelines based on KAZE. Our results are compared against Ground Truth, and the results obtained by [10]. These tests are followed in Sec. 5.2.

3. The current implementation is able to extract a reconstruction by its own, just requiring the images and the camera extrinsics as input.

In this study we have used a set of well known public image datasets [22], [30], [15] . These datasets feature a balanced compromise of line structure combined with large textured elements, it is intended to allow a fair comparison with feature point based reconstruction methods.

Camera intrinsics are provided as inputs for each set of views, and every method has to estimate the extrinsic parameters of all the cameras, the location of feature points or the poses of the principal lines defining the structure of the region of interest, and the planes adjusted to these lines, when applicable.

We distinguish two kinds of experiments depending on whether the objective is to validate both the camera poses and the 3D line sketch against the ground truth, or just to validate the 3D line sketch. In the first case the camera poses are estimated altogether with the set of 3D lines, as explained in this paper. Nevertheless, in the second case the employed camera extrinsics are primarly estimated from point-based SfM. The 3D lines obtained with our model are densely discretized into a set of 3D points laying on the line. Then the Hausdorff distance is computed from these points to the surface of the ground truth polygonal model. The RMSE of this distance is computed and provided as a measure of the error of the prediction compared against the ground truth. The figures referred from this section show the complete result returned by the algorithm, no alteration has been performed by human, and all the lines are shown, including outliers and wrongly posed lines.

Method	Line detection and matching	Depends on cloud	First 3D estimation	Final result
Proposed	Appearance and structure [16]	NO	Linear triangulation	SBA
Line3D++ [10]	LSD and SfM pipeline	YES	Depends on point cloud	SBA

Table 1: Overview of evaluated methods

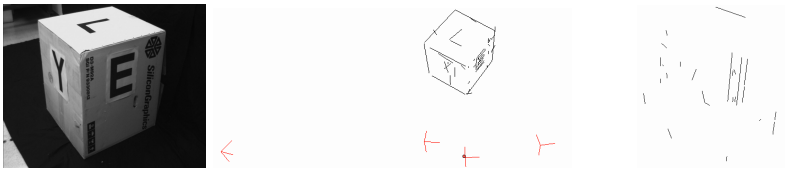
In Table 1 the main high level differences between the proposed method and the method Line3D++ [10] used for the evaluations are overviewed.

5.1. Experiments estimating camera poses and 3D lines

In this subsection we are evaluating the implementation of the model strictly as explained in this paper, meaning that the camera extrinsics are estimated right from the matched lines on the images and a solution obtained without using feature points at all. Therefore the images showing a common region of interest are selected from the datasets. The pictures showing just a fraction of this region were also avoided. The selected subsets comprise picture numbers {3-6} from the dataset CUBE [15] , and {0,2,4,5,7} from HALLWAY [15]. The result against the first subset is presented in Fig. 3. This set is characterized by low resolution and poor illumination. In order to compare this result, the same subset has been used with the VisualSfM [29] SIFT-based SfM pipeline teamed with Line3D++ [10], nevertheless no usable line based result has been obtained with this subset comprised of just 4 low resolution pictures. This is an example of test case for which the proposed method is superior to the competence. Being the line matching based on 2D appearance features, it can still reconstruct the scene on the absence of a thick point cloud. The figure shows 4 estimated camera poses and the line sketch formed by 50 spatial segments. The good fitting to the mesh of a cube proves that the proportions are fairly well represented with the proposed method.

For the dataset "HALLWAY" [15] the images showing the whole region of interest were selected. The 3D reconstruction is presented in Fig. 4, showing the spatial lines and the estimated cameras. Although a result with more lines are obtained in the original paper [15], this study relies on ground truth line

matching among all views. Oppositely, our approach starts with the sole set of pictures and generates geometric data exclusively from the results of line detection and matching algorithms.



(a) (b) (c)

Figure 3: Comparative results using the proposed method of line-based reconstruction exclusively based on lines. The selected subset from the dataset "BOX" [15] comprises the picture numbers {3-6}.a) Sample from the dataset. b) Result obtained with the proposed method. c) A SIFT-based reconstruction performed with VisualSfM [29] returned an sparse cloud, and therefore this result of Line3D++ [10] presents just 26 unconnected 3D lines, and it is not understandable.

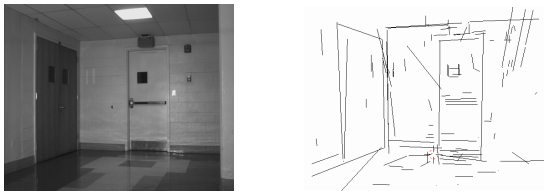


Figure 4: Selection of 5 pictures within the dataset "HALLWAY" [15] for the line based reconstruction: It comprises the picture numbers {0,2,4,5,7}. The second picture shows the line based 3D sketch and the 5 cameras estimated exclusively using the output from the line matching algorithm.

5.2. Experiments teaming with feature point descriptors

The most difficult scenario for line matching are sets of images featuring a large number of lines embed in an structure of repetitive pattern. Also, most of the ground-truth public datasets we found in the literature are intended for feature point based SfM pipelines. These sets feature wide baselines, large relative distances between pictures, and not all the structural elements are framed on each picture. The ideal dataset for line based reconstruction should include most of the structural elements on every picture, and minimize the number of

occlusions. A solution we took for creating the test cases was to pick subsets from these sequences of images that can fairly serve for evaluation and comparison. Secondly, matching outliers make difficult the convergence of the SBA of a large datasets. This happens also in the case of employing line intersections, because a single line mismatch can trigger the mis location of multiple intersections. For these cases it is unavoidable to team the pose estimator based on line correspondences with a point-based descriptor. In this study KAZE [31] point feature detector and descriptor is teamed with the proposed method and executed after the line detection and matching stages, and right before the triangulation among views. The addition of this SfM pipeline gives an initial estimation for the cameras, avoiding the increased uncertainty of obtaining it from triangulation of the line endpoints.

Having an initial estimation of the cameras allows to set a maximum distance between the cameras for the pairwise line matching to be performed among the respective images. In the implementation, this solution translates into setting a distance threshold between the camera centers for the line matching algorithm to be executed over the respective images. This prevents that the algorithm spend time on the search for line correspondences between two images showing opposite faces of an object, and avoids potential matching outliers due to similar sides of the same man-made object. Additionally, limiting the line matching to close neighboring views drops the total processing time dramatically.

The quantitative evaluation of the precision of the method is employing the synthetic dataset Timber-Frame-House [22], for performing two different comparative evaluations with the methods Line3D++ [10] and Jain [22]. Results are shown on Table 2, Fig. 5 and Fig. 6. The proposed method returns a 3D line based sketch of the model from as low as 6 images, whereas Line3D++ [10] requires 12 or more images to draw an abstraction that includes the main long structures of the house. Moreover, the number of images required for the model to output all or mostly all the edges visible in the original 3D representation, with fair accuracy, this is represented on Fig. 7. In order to consider a valid reconstruction for this comparison it was also required that most of the 3D seg-

ments extend to the whole length of their Ground Truth counterparts, oppositely to feature sequences of atomic short 3D segments. For a resulting sketch, the number of 3D segments correctly represented as an unique entity resembling the whole edge is denoted as P. Quantitative measurements have been performed on the results. Every 3D line is discretized into points, and the measure used to evaluate the precision of a resulting abstraction is the distance between these points and the Ground Truth mesh. The level of subdivisions of the octrees at which this Cloud-To-Mesh distance computation was performed is 3 [32]. The test cases are created looking for the minimum number of images that resulted in a reconstruction comprising most of the lines shown in the Ground Truth dataset. These include both the edges of the polygonal model and the lines present in the texture. In order to capture all the segments of the representation, each test case comprise two groups of pictures, one for the front and the second one capturing the opposite side of the house. The test cases are labeled as S_6, comprising image numbers {6,9,86,46,49,126}, S_8 further add two more images {89,129} to the list, S_{10} includes {8,10,12,88,90,48,50,52,128,130}, and S_{12} further adds images {92,132} to the latter. The resulting 3D line sketches from both sides of the house are aligned by using common lines. This completed sketch is finally aligned to the Ground Truth in order to measure the precision. The time employed by the proposed method is clearly outperformed by Line3D++ [10]. One of the reasons is the lower number of lines included in the results of the latter one. In the case of Line3D++ [10], the time taken by the required SfM pipeline[29] is added to the amount.

The quantitative comparison and Ground Truth evaluation is performed firstly using a low number of images, and then more images are incrementally added. The comparisons performed against S_6 and S_8, are shown in Fig. 5. The result proves that the proposed method is able to obtain a number of structures of the house from as low as 6 images, and still holding a decent accuracy. On the other hand, the method Line3D++ [10] returns sparse short segments for both cases. This sparsity complicates the understanding of what the spatial line cloud is resembling, and difficult the alignment to the Ground Truth mesh.

Note that this method also fails to retrieve any long segment of the house for these test cases.

The results for a larger number of images of 10 and 12, using S_{10} and S_{12}, are shown in Fig.6. S_{10} turns out to be the minimal test case to return a complete line-based reconstruction from the original synthetic images by using the proposed model. It is the subset with the minimum number of images required for the method to generate a complete abstraction, meaning that it includes most of the segments resembling the main edges and lines on the original 3D model and texture. It is proved that the proposed method is able to return a greater number of complete segments that resemble the original representation, while holding a level of precision equivalent to the one achieved by the method Line3D++ [10] in this case. For S_{12} the point cloud obtained by VisualSfM pipeline[29] is dense enough for Line3D++ [10] to generate a fair abstraction, showing completeness and continuity between the segments.

The obtained mean values, and the Ground Truth segment count for the mentioned test cases against the two methods are displayed on Table 2. The mean distance to GT mesh is lower for Line3D++ [10] than for the proposed method, mainly because their line poses are estimated based on SIFT-originated point clouds. Nevertheless, the number of 3D segments represented as a whole for the first three subsets is much lower than in the proposed method, as clearly seen on the images of Fig. 5 and Fig. 6.

Fig.8 is included in this subsection for reference, as the result provided by [22]. It is based on Ground Truth camera poses, instead of estimating these from the lines or SfM pipelines. The actual code neither the executables are provided by the authors, so it could not be tested against the proposed method. Nevertheless the shown result gives an idea of the precision achieved by the method, as reference. This result is provided by the authors, altogether with the complete dataset and the used subset, with 72 images.

Fig. 7 shows a comparison of the proposed method and [10] based on the same dataset [22], but including a larger number of cameras facing the same region of interest. For this test case we want to put in value that the proposed

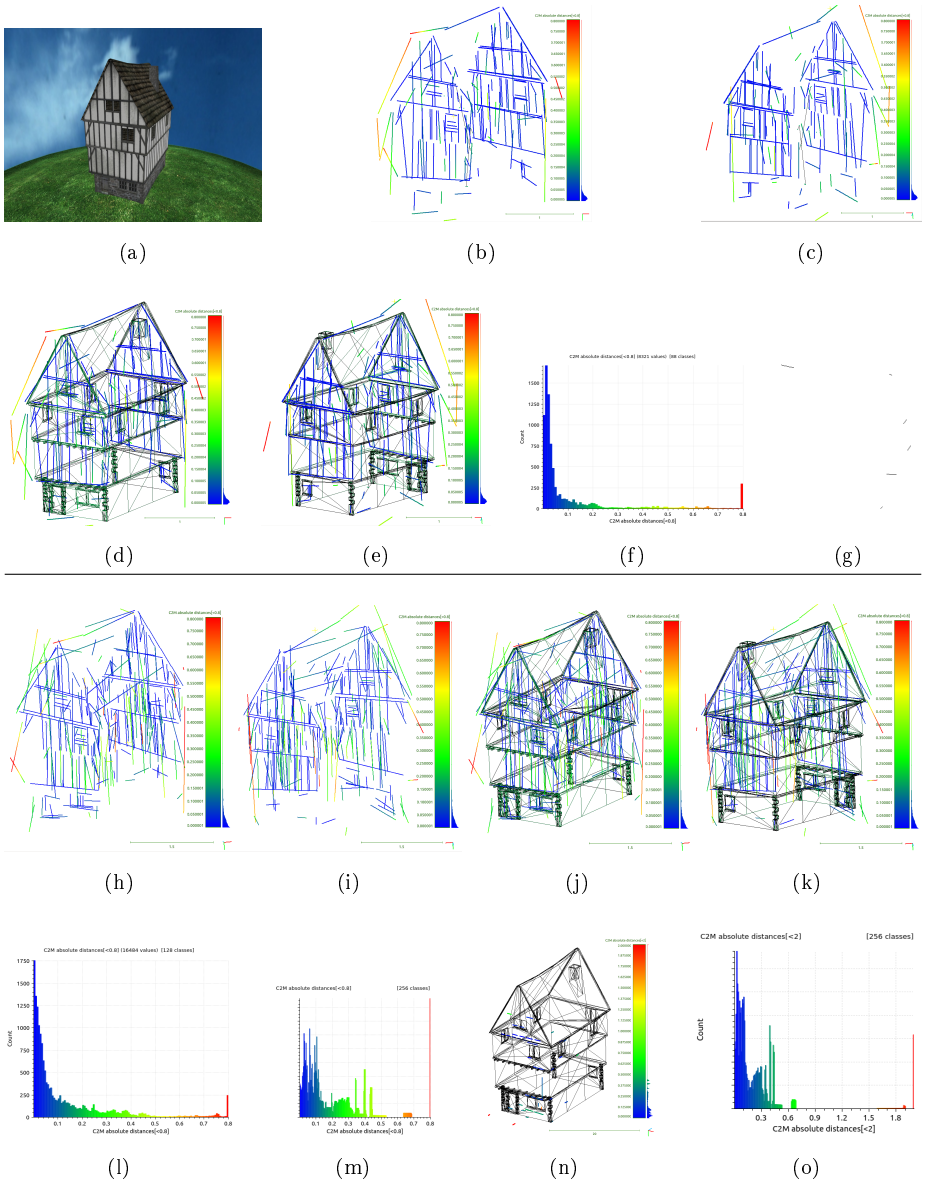


Figure 5: Quantitative comparison using the sets S_6 and S_8 [22]. This figure is better viewed on a screen with a 4x zoom. a) Sample of the set. b) and c) Proposed method against S_6, resulting 175 lines. The distance from each point in the cloud to the surface of the Ground Truth mesh is represented in colors. d) and e) Same superposed onto the Ground Truth mesh. f) Histogram of distances to Ground Truth with the proposed method. The maximum distance to be accounted is set to be 0.8, already considered as outlier. g) Sparse atomic lines returned by the Line3D++ [10] method. It has been aligned with the Ground Truth mesh. h) to l) The proposed method against the set S_8, with 294 segments. m) and n) same measurements for the result by Line3D++ [10]. o) shows the histogram for this latter result.

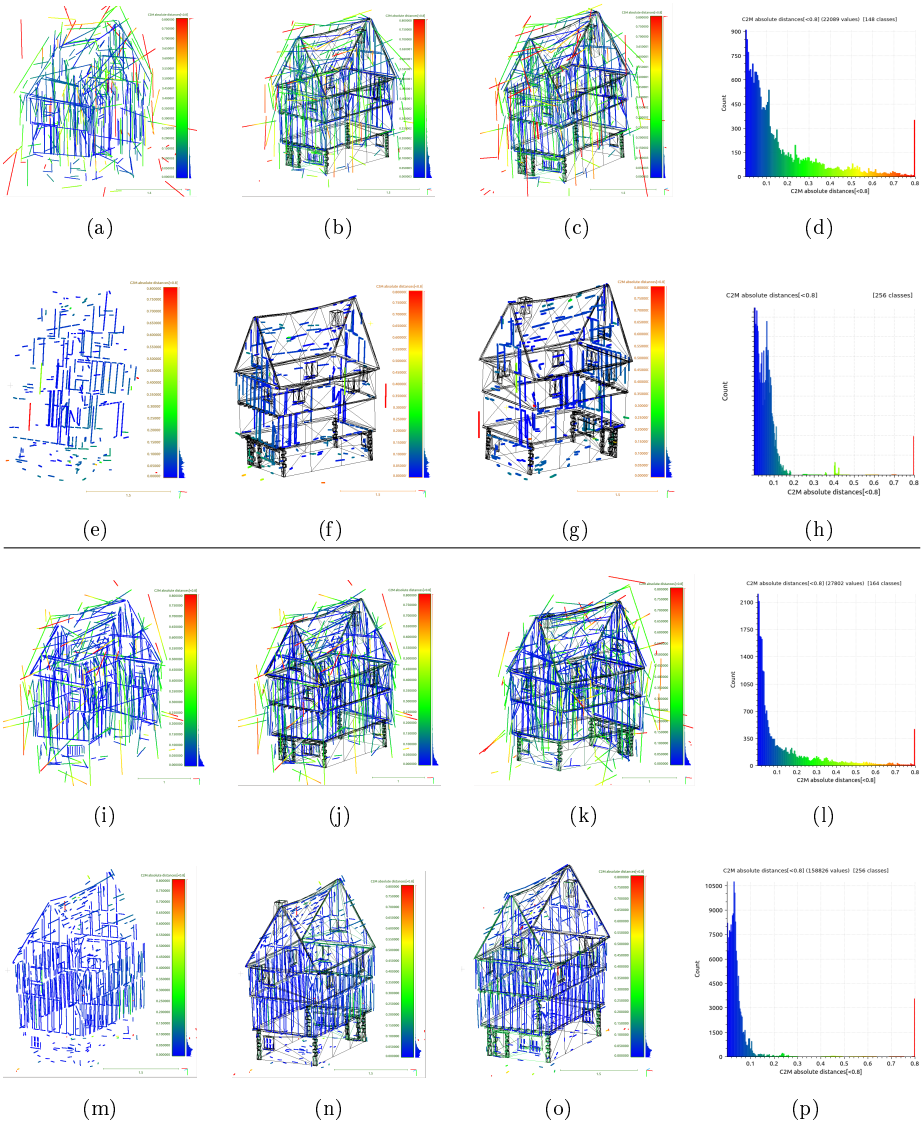
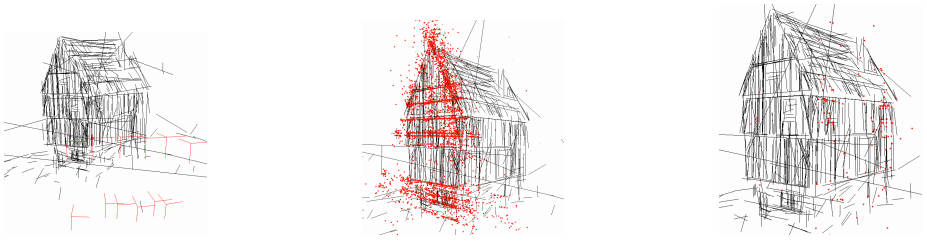
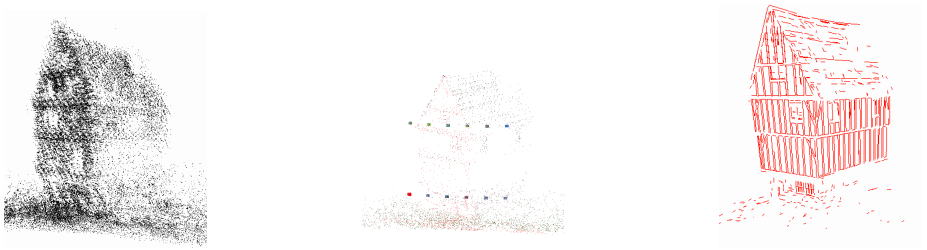


Figure 6: Quantitative comparison using the sets S_{10} and S_{12}[22]. This figure is better viewed on a screen with a 4x zoom. a), b) and c) Proposed method against S_{10}. The obtained 475 lines have been discretized in points. The distance from each point in the cloud to the surface of the Ground Truth mesh is represented in colors. d) and e) Same superposed onto the Ground Truth mesh. f) Histogram of distances to Ground Truth with the proposed method. The maximum distance to be accounted is set to be 0.8, already considered as outlier. g) Sparse atomic lines returned by the Line3D++ [10] method. It has been aligned with the Ground Truth mesh. h) to l) Same for the proposed method against the set S_{12}, with 556 segments. m) and n) same measurements for the result by Line3D++ [10]. o) shows the histogram for this latter result.

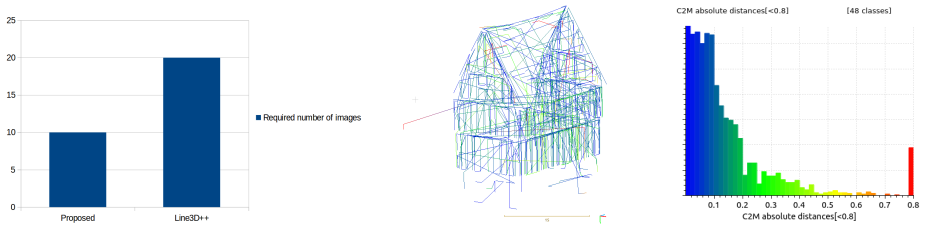


(a) (b) (c)



(d) (e) (f)

Figure 7: Qualitative comparison using the public dataset Timber-Frame-House [22]. This figure is better viewed on a screen with a 4x zoom. The used subset comprises image numbers {5-10},{85-90}, capturing the same corner of the house. a) Result of the proposed method, featuring the 12 camera poses and 583 lines. b) Proposed method. Intersections of the matched lines fitted to the plane with more inliers. c) Same for the next plane with more inliers. d) Point cloud comprising 54041 points by using KAZE [31] point features, used just for retrieving initial estimations of the camera poses for the proposed method, looking for a suitable comparison with Line3D++ [10]. e) SIFT based reconstruction obtained with VisualSFM [29], featuring 10608 points. f) Line based reconstruction obtained with Line3D++ [10], built over the result shown in (e). The abstraction features 617 segments, and more accurately located than the proposed method.



(a) (b) (c)

Figure 8: a) Required number of images for obtaining an abstraction featuring more than 85% of the lines featured in the GT representation. b) and c) The result of the method [22] versus a subset of 72 images is shown for reference.

Set	Method	Lines	Completed lines	Processing time	Mean distance to GT	std deviation
S_6	Proposed	175	52	122s	0.1	0.36
	Line3D++ [10]	6	0	18s	No alignment	No alignment
S_8	Proposed	294	74	160s	0.21	0.3
	Line3D++ [10]	91	1	20s	0.07	0.11
S_{10}	Proposed	475	244	235s	0.08	0.24
	Line3D++ [10]	290	112	30s	0.08	0.11
S_{12}	Proposed	556	305	374s	0.8	0.25
	Line3D++ [10]	622	176	42s	0.07	0.24

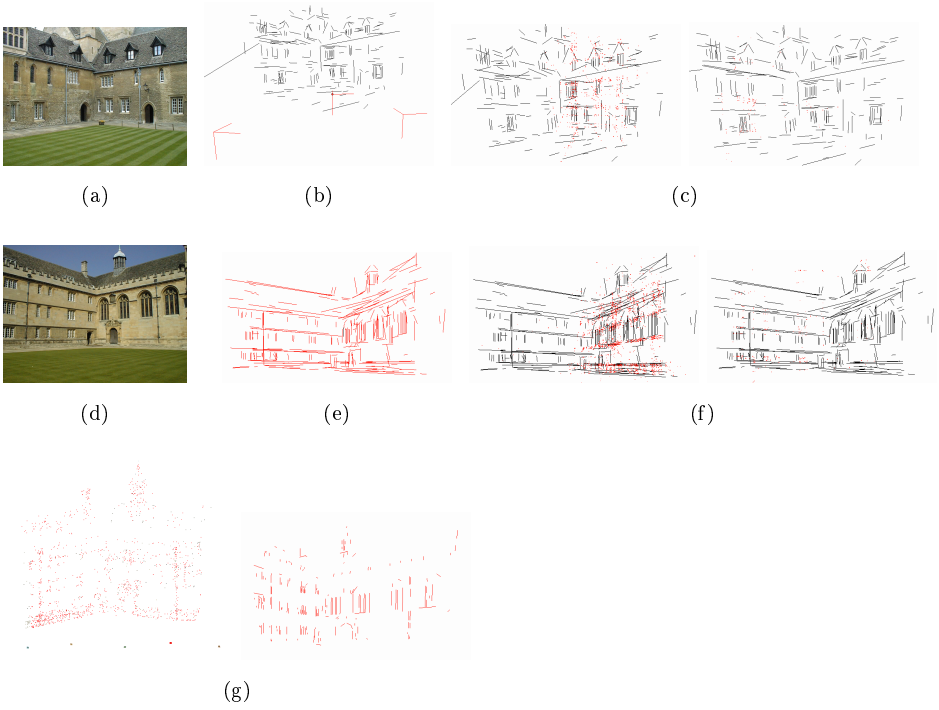
Table 2: Quantitative comparison of both methods compared against the dataset

method provides an additional grouping of the structural information. The coplanar lines and their intersections are grouped and the 3D abstraction of these intersections are plotted. This is shown as an analogy of a SIFT-like generated point cloud. With the higher amount of views of this test case, the result of Line3D++ [10] shows a large number of segments precisely located, nevertheless most of the shown edges are comprised by a discontinuous array of short segments. This result contrasts with the completeness and uniqueness of the segments returned by the proposed method.

It was found out that our method is able to obtain a reconstruction in adverse situations where Line3D++ [10] is not able to return a valid abstraction, for instance when the number of pictures is low and they do not present clear textures. The famous datasets [30] comprise few low resolution images, and are always difficult to represent by feature point based SfM pipelines. These have been selected because the addition of 3D segment abstractions improve the overall representation. In addition to plot the reconstructed lines by the proposed method, Fig. 10 also show the estimated spatial intersections of the coplanar lines. For the first one, built of just 3 pictures, the point cloud provided by VisualSfM [29] is too sparse for Line3D++ [10] to be able to construct a 3D abstraction. For the second dataset, comprising 5 views, it does place 3D segments because the obtained point cloud is denser, as the set comprises more views. Nevertheless, the cloud is not thick enough for Line3D++ [10] to represent all the main lines in the scene, as can be observed in the picture included in Fig. 10. On the other hand, the proposed method manages to obtain

(a) (b) (c)

Figure 9: Qualitative comparison using the dataset Building-Blocks [22], of resolution 1440 × 1080. The used subset comprises the pictures {0,2,4,6,8,10}. a) Sample. b) Result with the proposed method, featuring 329 lines and the estimations for the six camera pose. c) On the left, the point based obtained by VisualSfM [29], featuring 3052 points and the 6 camera poses. Based on this point cloud it was generated the result obtained by Line3D++ [10] that shown on the right picture, with just 102 segments and difficult to understand.



(a) (b) (c)

(d) (e) (f)

(g)

Figure 10: a) Sample from the dataset Merton College [30]. b) Proposed method, with 234 lines and the 3 estimations for the camera poses for the three pictures. c) Proposed method. Intersections of lines fitted to the two planes with more inliers. The obtained point cloud comprises just 2250 3D points, and is quite sparse and no valid result was obtained by Line3D++ [10]. d,e,f) Same for the Wadham College[30] dataset. g) Wadham College[30]. Point cloud result of the SIFT-based VisualSfM [29], that comprises just 3739 3D points. Based on this result, it was generated the result by Line3D++ [10] on the right, with fewer and shorter lines but located with more accuracy than the proposed method.

an usable result for both datasets. In addition, on both cases the lines were fitted to several planes, being the two planes with more inliers the corresponding to the vertical walls of each building. This allowed to use the intersections of coplanar lines to estimate the camera poses for the final bundled results shown on the figures.

The dataset Building-Blocks [22] is used for another quantitative analysis. Fig. 9 allows the comparison of the results of the proposed method with the ones provided by VisualSfM [29] and Line3D++ [10] . A subset of 6 views from close camera positions was selected in order to enhance the differences between the methods. In this case the proposed method is able to obtain a bundled result with more lines and with visually better grasped connectivity than the abstraction provided by Line3D++ [10].

6. Conclusions

This paper presents a novel integration of a set of algorithms to create a line-based spatial sketch, showing the main structures of the man-made environment, and the camera poses. It receives as input the camera intrinsic parameters and at least 3 pictures. The set of methods include novel observation relations of groups of straight segments that are captured from different camera poses. The method features the novelty of employing coplanar line intersections like feature points. The first noticeable advantage of the proposed method over the competition is that it does not require as input the results of an external SfM pipeline, neither bases the line reconstruction on point clouds. This advantage shows up when it is required a reconstruction performed by an algorithm exclusively based on line detection and matching among images, as the result can be obtained by using triangulated line endpoints as input for the SBA. Therefore, 3D sketches are obtained from sets of pictures that present corrupted texture, blurring, and low definition images where the feature point descriptor fails to detect any usable point. It has been proven that there is a feasible alternative to base the 3D line abstractions on feature point based SfM results, by estimating the camera

poses with the observed lines and their coplanar intersections. During the experimental testing we proved that the proposed method performs better than the competition for reconstructing scenarios captured on a reduced number of pictures, with low texture, poor illumination, low resolution or blurring.

It was obtained better results than the competition against public datasets with a reduced number of images. This low number of views limited the thickness of the point cloud generated by the SfM pipelines. The competition models were outperformed because they required a point cloud dense enough for a feasible line matching outlier detection, and in order to obtain an accurate estimation for the camera extrinsics. Oppositely, the proposed method was able to reconstruct simple line-based sketches with equivalent precision and number of lines featured in the results of the competition when using larger sets. We can conclude that our method requires less number of images for obtaining equivalent results than the competition, in terms of quantity of lines and precision of the abstraction.

Additional future work might include a more advanced use of the planes in the line based reconstruction of man-made environments. For the implementation in a desktop or laptop computers, all the code is parallelized for CPU, but in the future, some algorithms will benefit of the use of GPU implementation. On the other hand, an implementation on a Raspberry Pi is a feasible solution for onboard line based 3D reconstruction during UAV flight.

Acknowledgment

This work has received financial support from the Xunta de Galicia through grant ED431C 2017/69 and Xunta the Galicia (Centro singular de investigación de Galicia accreditation 2016-2019) and the European Union (European Regional Development Fund - ERDF) through grant ED431G/08.

References

References

[1] S. Yang, S. Scherer, Direct monocular odometry using points and lines, in: ICRA, 2017, IEEE, 2017, pp. 3871–3877.

[2] R. I. Hartley, P. Sturm, Triangulation, CVIU 68 (2) (1997) 146–157.

[3] B. Triggs, P. F. McLauchlan, R. I. Hartley, A. W. Fitzgibbon, Bundle adjustment—a modern synthesis (1999) 298–372.

[4] Y. Furukawa, J. Ponce, Accurate, dense, and robust multiview stereopsis, IEEE PAMI 32 (8) (2010) 1362–1376.

[5] M. Rothermel, K. Wenzel, D. Fritsch, N. Haala, Sure: Photogrammetric surface reconstruction from imagery, in: Proceedings LC3D Workshop, Berlin, 2012, pp. 1–9.

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

[7] C. Raposo, M. Antunes, J. P. Barreto, Piecewise-planar stereoscan: structure and motion from plane primitives, in: ECCV, Springer, 2014, pp. 48–63.

[8] L. Zhang, R. Koch, Structure and motion from line correspondences: representation, projection, initialization and sparse bundle adjustment, Journal of Visual Communication and Image Representation 25 (5) (2014) 904–915.

[9] B. Micusik, H. Wildenauer, Structure from motion with line segments under relaxed endpoint constraints, Int. J. Comput. Vis. 124 (1) (2017) 65–79.

[10] M. Hofer, M. Maurer, H. Bischof, Efficient 3d scene abstraction using line segments, CVIU 157 (2017) 167–178.

[11] X. Zuo, X. Xie, Y. Liu, G. Huang, Robust visual slam with point and line features, arXiv preprint arXiv:1711.08654.

[12] B. Micusik, J. Kosecka, Piecewise planar city 3d modeling from street view panoramic sequences, in: CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 2906–2912.

[13] A. Criminisi, I. Reid, A. Zisserman, A plane measuring device, Image and Vision Computing 17 (8) (1999) 625–634.

[14] H. Kim, S. Lee, Simultaneous line matching and epipolar geometry estimation based on the intersection context of coplanar line pairs, Pattern Recognit. Lett. 33 (10) (2012) 1349–1363.

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

[16] J. López, R. Santos, X. R. Fdez-Vidal, X. M. Pardo, Two-view line matching algorithm based on context and appearance in low-textured images, Pattern Recognit. 48 (7) (2015) 2164–2184.

[17] G.-S. Xia, J. Delon, Y. Gousseau, Accurate junction detection and characterization in natural images, Int. J. Comput. Vis. 106 (1) (2014) 31–56.

[18] B. Fan, F. Wu, Z. Hu, Robust line matching through line–point invariants, Pattern Recognit. 45 (2) (2012) 794–805.

[19] N. Xue, G.-S. Xia, X. Bai, L. Zhang, W. Shen, Anisotropic-scale junction detection and matching for indoor images, IEEE Trans. Image Process. 27 (1) (2018) 78–91.

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

[21] R. I. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, 2nd Edition, Cambridge University Press, ISBN: 0521540518, 2004.

[22] A. Jain, C. Kurz, T. Thormählen, H.-P. Seidel, Exploiting global connectivity constraints for reconstruction of 3d line segments from images, in: CVPR, 2010 IEEE Conference on, IEEE, 2010, pp. 1586–1593.

[23] T. Koletschka, L. Puig, K. Daniilidis, Mevo: Multi-environment stereo visual odometry, in: IROS 2014 IEEE/RSJ International Conference on, IEEE, 2014, pp. 4981–4988.

[24] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, G. Randall, Lsd: A fast line segment detector with a false detection control, IEEE Trans. Pattern Anal. Mach. Intell. 32 (4) (2010) 722–732.

[25] S. Ramalingam, M. Antunes, D. Snow, G. Hee Lee, S. Pillai, Line-sweep: Cross-ratio for wide-baseline matching and 3d reconstruction, in: Proceedings of the IEEE Conference on CVPR, 2015, pp. 1238–1246.

[26] M. Hofer, M. Maurer, H. Bischof, Line3d: Efficient 3d scene abstraction for the built environment, in: German Conference on Pattern Recognition, Springer, 2015, pp. 237–248.

[27] D. Nistér, An efficient solution to the five-point relative pose problem, IEEE transactions on pattern analysis and machine intelligence 26 (6) (2004) 756–770.

[28] N. Snavely, S. M. Seitz, R. Szeliski, Photo tourism: exploring photo collections in 3d, in: ACM transactions on graphics (TOG), Vol. 25, ACM, 2006, pp. 835–846.

[29] C. Wu, et al., Visualsfm: A visual structure from motion system.

[30] T. Werner, A. Zisserman, Model selection for automated architectural reconstruction from multiple views, BMVC, 2002, pp. 53–62.

[31] P. F. Alcantarilla, A. Bartoli, A. J. Davison, Kaze features, in: ECCV, Springer, 2012, pp. 214–227.

[32] DGM, http://www.cloudcompare.org (2015).