Computer Vision and Image Understanding 000 (2016) 1–12

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Efficient 3D scene abstraction using line segments

Manuel Hofer*, Michael Maurer, Horst Bischof

Institute for Computer Graphics and Vision, Graz University of Technology, Graz, Austria

ARTICLE INFO

Article history:
Received 10 November 2015
Revised 17 March 2016
Accepted 24 March 2016
Available online xxx

Keywords:
Structure-from-Motion
3D reconstruction
Line segments
Scene abstraction
Multi-view Stereo

ABSTRACT

Extracting 3D information from a moving camera is traditionally based on interest point detection and matching. This is especially challenging in urban indoor- and outdoor environments, where the number of distinctive interest points is naturally limited. While common Structure-from-Motion (SfM) approaches usually manage to obtain the correct camera poses, the number of accurate 3D points is very small due to the low number of matchable features. Subsequent Multi-view Stereo approaches may help to overcome this problem, but suffer from a high computational complexity. We propose a novel approach for the task of 3D scene abstraction, which uses straight line segments as underlying features. We use purely geometric constraints to match 2D line segments from different images, and formulate the reconstruction procedure as a graph-clustering problem. We show that our method generates accurate 3D models with low computational costs, which makes it especially useful for large-scale urban datasets.

1. Introduction

Recovering 3D information from an image sequence used to be a very challenging and time consuming task. Today, thanks to freely available software (Moulon et al.; Schoenberger et al., 2014; Snavely et al., 2006; Sweeney; Wu, 2013) or sophisticated professional tools (Agisoft photoscan; Capturingreality; Strecha et al.), even non-expert users are able to generate accurate 3D models from arbitrary scenes within hours. Since these so-called Structure-from-Motion (SfM) approaches operate on a sparse set of distinctive feature points (e.g., SIFT Lowe (2004) features), the resulting 3D point cloud is usually quite sparse as well. The important part of the SfM result are the camera poses for each input image, which enable subsequent Multi-View Stereo (MVS) pipelines (e.g., PMVS Furukawa et al. (2010) or SURE (Rothermel et al., 2012)) to create a (semi-) dense point cloud.

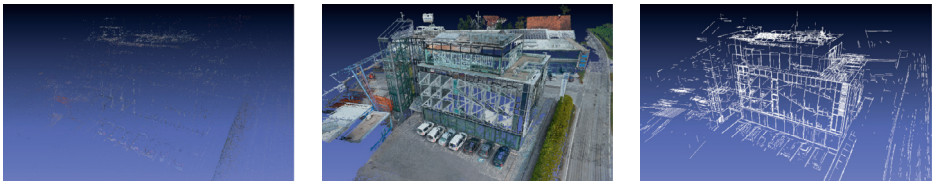
While the first part of this two-step procedure (pose estimation via SfM) can be computed very efficiently even for large crowd-sourced datasets (Frahm et al., 2010; Havlena and Schindler, 2014), the second part (dense reconstruction via MVS) is still computationally expensive and can take up to several days on modern desktop computers. Moreover, the resulting 3D point cloud easily consists of millions of points and just viewing it in a point-cloud viewer becomes a very tedious task. The same holds for any kind of automatic data analysis or post processing (e.g., meshing Labatut et al. (2007) and texturing (Waechter et al., 2014)). This is due to using point clouds as 3D representation. On the one hand, shapes of arbitrary complexity can be described by a set of 3D points, but on the other hand, the number of points needed to do so can explode very quickly.

Desirable is an efficient way of abstracting the 3D model, to decrease the amount of data without reducing the embedded 3D information. A natural choice is to use more complex geometric primitives as data representation, such as planes (e.g., Kim and Manduchi (2014); Raposo et al. (2014); Sinha et al. (2009)) or lines (e.g., Hofer et al. (2015); Micusik and Wildenauer (2014); Zhang and Koch (2014)). While this might not be sufficient for natural scenes (e.g., forests, etc.), it is especially useful for urban indoor- and outdoor environments, where most of the structures are piecewise planar/linear.

In this paper, we propose a system, denoted as *Line3D++*, to generate a semantically meaningful 3D model of urban environments, by using 2D line segments as image features. Our method uses an oriented image sequence as input, whose camera poses can be obtained by any conventional SfM pipeline. We use weak epipolar constraints to establish a large set of potential line correspondences between images, and use a scoring formulation based on mutual support to separate correct from incorrect matches for each segment individually. We obtain a final line-based 3D model by clustering 2D segments from different views, using an efficient graph-clustering formulation. In addition, we show how the SfM result as well as the 3D line model can be further improved by employing a combined bundle adjustment over the reconstructed points and lines. Our method scales almost linearly with the

* Corresponding author. Tel.: +433168735090.
E-mail addresses: hofer@icg.tugraz.at (M. Hofer), maurer@icg.tugraz.at (M. Maurer), bischof@icg.tugraz.at (H. Bischof).
URL: http://www.icg.tugraz.at (M. Hofer)

http://dx.doi.org/10.1016/j.cviu.2016.03.017
1077-3142/



SfM [23]
97,699 points
(a) runtime: 0.75 hours

PMVS [15]
12,156,664 points
(b) runtime: > 11 hours

Line3D++ (proposed)
7,877 lines
(c) runtime: **210.65 sec**

Fig. 1. Three different 3D representations of the *Building* sequence (344 images). (a) Sparse 3D model (Irschara et al., 2007) (SIFT Lowe (2004) features). (b) Semi-dense point-cloud (PMVS Furukawa et al. (2010)). (c) 3D line model using *Line3D++*. It is hardly possible to recognize the building in the sparse 3D model, while it is clearly recognizable in both the semi-dense- and the line-based 3D model. Compared to PMVS, our method has much lower runtime- and memory requirements.

number of input images and has low memory requirements, to easily reconstruct scenes from hundreds of images. We demonstrate our approach on several challenging real-world datasets, as well as a synthetic sequence with ground truth. The implementation of this work is freely available,[1] and works off-the-shelf with the output of several state-of-the-art SfM pipelines (Moulon et al.; Schoenberger et al., 2014; Snavely et al., 2006; Strecha et al.; Wu, 2013).

Fig. 1 shows a comparison between a sparse-, a dense-, and a line-based 3D model for an urban scene. As we can see, our reconstruction provides a high degree of semantically meaningful 3D information, despite its sparsity compared to the dense model. Moreover, our method is much more efficient than computing a dense 3D model, with a runtime of just a few minutes even for this large-scale dataset.

2. Related work

While line segments have been used for tasks such as image registration or 3D reconstruction for a long time (e.g., Ayache and Faverjon (1987); Baillard et al. (1999); Schmid and Zisserman (1997)), in recent years image-based 3D reconstruction has been dominated by the use of image feature-points and their invariant descriptors (e.g., SIFT Lowe (2004)). Only quite recently, the principles of feature-point descriptors have been successfully adapted to the task of line segment matching (e.g., Wang et al. (2009); Zhang et al. (2012, 2011)), but line-based 3D reconstruction for real-world scenarios is still rarely used.

Bartoli and Sturm (2005) proposed a full SfM pipeline based on line segments, including bundle adjustment. They make use of Plücker coordinates as 3D line representations and a trifocal tensor, since two views do not put enough constraints on the camera motion when using lines alone. In an earlier work, they successfully used line segments for SfM with only two images at a time, by establishing point-wise correspondences between lines from different images (Bartoli et al., 2004). Bay et al. (2006) use line segments from two un-calibrated images to determine relative camera poses, and to compute a piecewise planar 3D model. They establish correspondences between the images by using a color histogram based line descriptor (Bay et al., 2005). Further, Schindler et al. (2006) demonstrated how the Manhattan-world assumption can be incorporated into the reconstruction procedure, to decrease the computational complexity and to reconstruct buildings from two views.

Recently, Zhang and Koch (2014) proposed a very sophisticated line-based SfM pipeline. They initialize the reconstruction from matched lines across three images in closed from, and incrementally add new images by establishing 3D-2D line correspondences for absolute pose estimation. Additionally, they introduce the *Cayley* 3D line representation, which can be efficiently derived from Plücker coordinates, and only requires four parameters to encode a 3D line. This representation allows a more efficient optimization of 3D lines during bundle adjustment. In the same year, Micusik and Wildenauer (2014) presented a SLAM-like line-based SfM system, which is able to reconstruct large-scale urban scenes with line segments alone. They use relaxed endpoint constraints for line matching, which requires narrow baselines. They showed impressive results, especially for indoor scenes.

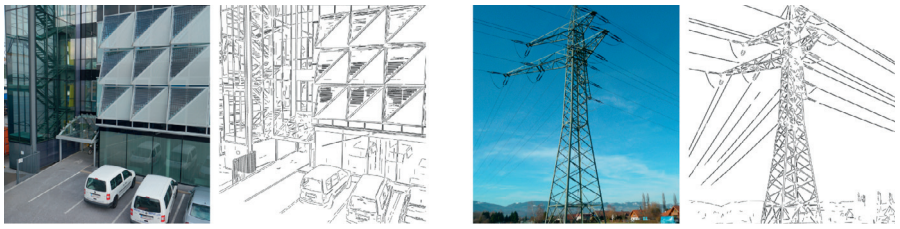
The aforementioned methods try to solve both camera pose estimation and 3D reconstruction using only line segments. This is especially challenging for pose estimation, since line correspondences between two images do not put enough constraints on the epipolar geometry for the general case. Hence, additional constraints are needed to solve for the unknown camera orientations (e.g., explicit point-wise correspondences between lines (Bartoli et al., 2004), a trifocal tensor (Bartoli and Sturm, 2005), narrow baselines (Micusik and Wildenauer, 2014), or the Manhattan-world assumption (Kim and Manduchi, 2014)). Related to these methods, several approaches for the task of line-based pose estimation have been presented over the years, e.g., by utilizing special line triplets for relative- (Elqursh and Elgammal, 2011), and 2D-3D line correspondences for the absolute pose problem (Zhang and Koch, 2012).

In contrast to pure line-based SfM, several MVS algorithms that focus on the reconstruction procedure with given camera poses have been presented. Jain et al. (2010) proposed a method that does not require explicit correspondences between line segments from different images, which enables 3D reconstruction under difficult lighting conditions or around highly non-planar objects (such as power pylons), where patch-based line descriptors would fail. They formulate the reconstruction procedure as an optimization problem, where the unknown depth of the endpoints of 2D line segments in the images is modelled as a random variable. They compute the most probable 3D locations for the segment endpoints by minimizing the re-projection error among several neighboring views, and compute a final 3D model by merging individual 3D hypotheses that are sufficiently close together. While their approach generates very promising and visually pleasant results, the continuous optimization of the endpoint depths (in a potentially large range) renders the method inefficient for large-scale datasets.

Since the approach by Jain et al. (2010) is very versatile, we decided to go a similar way in our research. To overcome the high runtime requirements, we switch from using no explicit 2D line correspondences at all to using weak epipolar matching constraints, without any kind of appearance information. We compute a set of potential matches for each 2D line segment in its neighboring images, and limit its potential 3D locations to a discrete set coinciding with these matches. We have shown how this simple modification significantly boosts the performance without negatively affecting the accuracy (Hofer et al., 2013a; 2013b). We further replace the greedy line-merging from Jain et al. (2010) with a scale invariant graph clustering formulation (Hofer et al., 2014b),

[1] https://github.com/manhofer/Line3Dpp.

Please cite this article as: M. Hofer et al., Efficient 3D scene abstraction using line segments, Computer Vision and Image Understanding (2016), http://dx.doi.org/10.1016/j.cviu.2016.03.017



(a) Building (b) Pylon

Fig. 2. Line segment detection using the LSD algorithm (von Gioi et al., 2010). (a) Example image from the *Building* sequence (3801 segments) (b) Example image from the *Pylon* sequence (1875 segments).

which can also be evaluated on-the-fly for incremental SfM applications (Hofer et al., 2014a). As a conclusion of our research, we presented a publicly available line-based 3D reconstruction tool, denoted as *Line3D*, which improves the accuracy and completeness of the obtained 3D models even further (Hofer et al., 2015). In this paper, we describe our principle of line-based 3D reconstruction in more detail, and additionally show how the obtained 3D line models as well as the underlying SfM results can be further improved by a combined bundle adjustment procedure at the end. The resulting algorithm *Line3D++* is an extension of our most recent work (Hofer et al., 2015), with slight modifications which mostly concern usability and runtime performance.

In the following sections we describe all necessary steps of our 3D reconstruction method in detail. We assume the camera poses to be known, which can be achieved by running an SfM pipeline beforehand. We conclude with extensive experiments on synthetic- as well as real-world datasets.

3. 3D reconstruction using line segments

Given an (unordered) image sequence $I = \{I_1, \dots, I_N\}$, we first run an arbitrary SfM pipeline to obtain the corresponding camera poses as well as a sparse set of 3D points $X = \{X_1, \dots, X_K\}$, which is needed solely to define which images are visual neighbors (i.e., see the same thing). We further define $X(i) \subset X$ to be the set of 3D points which are visible in image I_i, where we only consider points which are visible in at least three images. We require a set of 2D line segments $L_i = \{l_1^i, \dots, l_{m_i}^i\}$ per image, where each segment l_m^i simply consists of two endpoints $p_m^i, q_m^i \in \mathbb{R}^2$. The line segments can be obtained by any line segment detector, such as LSD (von Gioi et al., 2010) or EDL (Akinlar and Topal, 2011). Both algorithms are very reliable and deliver accurate and virtually outlier free detections, in a very efficient way. Fig. 2 shows two example images with their respective line segments, obtained by LSD (von Gioi et al., 2010). In our approach, we only consider the k longest detected line segments for 3D reconstruction ($k = 3000$ by default), and also throw away very small segments with a size smaller than ρ times the image diagonal ($\rho = 0.005$).

Our method consists of several steps:

1. Establishing potential correspondences between line segments from different images (Section 3.1)
2. Evaluating these correspondences based on their support in neighboring views (Section 3.2)
3. Selecting the most plausible correspondence for each 2D segment as its 3D position hypothesis (Section 3.3)
4. Clustering 2D segments based on their spatial proximity in 3D (Section 3.4)
5. Optimizing the SfM result and the 3D lines using combined bundle adjustment [*optional*] (Section 3.5)

In the following sections we will describe all these steps in more detail.

3.1. Establishing line segment correspondences

To generate a line-based 3D model we need to establish correspondences between 2D line segments from different images. Theoretically, this could be done by one of the numerous line-matching approaches presented in the past (e.g., Wang et al. (2009); Zhang et al. (2012); 2011)). However, most of these approaches are patch-based and are therefore only suitable for line segments located on planar surfaces. Most of the line segments in natural images correspond to depth discontinuities, which results in line descriptors describing the potentially far away background. To overcome this drawback, we use purely geometric constraints to obtain potential correspondences between 2D line segments from different images, based on their known epipolar geometry.

Since it would be infeasible (and unnecessary) to match all images with each other, we first compute a set of visual neighbors $V_i \subset \{1, \dots, N\} \setminus \{i\}$ for each image I_i, by finding its M nearest neighbors in terms of Dice's similarity coefficient

$$S_I(i, j) = \frac{2 \cdot |X(i) \cap X(j)|}{|X(i)| + |X(j)|}, \tag{1}$$

which sets the number of common world-points in relation to the total number of world-points for each image (the higher the more similar). This procedure assumes that an SfM pipeline has been run beforehand. However, since we only require the camera poses to compute the fundamental matrices between neighboring views, this could technically also be obtained by an external tracking system. To obtain the visual neighbor sets for this case, one can of course also use different definitions of view similarity, e.g., small distances between their camera centers and small angles between their optical axes (Hofer et al., 2013b).

We then match all segments in L_i to all segments in L_j (if $j \in V_i$). For a specific segment pair, $l_m^i \in L_i$ and $l_{\tilde{m}}^j \in L_j$, we compute the epipolar lines of the endpoints of l_m^i in the opposite image I_j. We then simply intersect the infinite line passing through the segment $l_{\tilde{m}}^j$ with these epipolar lines, and obtain two intersection points x_1 and x_2 which are collinear with the endpoints $p_{\tilde{m}}^j$ and $q_{\tilde{m}}^j$ of the segment $l_{\tilde{m}}^j$ (for the sake of simplicity we use x_1 and x_2 as the name for all intersection points here, even though they of course depend on the 2D segment pair to be matched). We then define a matching score between l_m^i and $l_{\tilde{m}}^j$ as

$$s(l_m^i, l_{\tilde{m}}^j) = \frac{\text{inner}\left(\left\{p_{\tilde{m}}^j, q_{\tilde{m}}^j, x_1, x_2\right\}\right)}{\text{outer}\left(\left\{p_{\tilde{m}}^j, q_{\tilde{m}}^j, x_1, x_2\right\}\right)}, \tag{2}$$

where $\text{inner}(\{\cdots\})$ and $\text{outer}(\{\cdots\})$ stand for the Euclidean distance between the inner- and the outer pair of the four collinear points ($p_{\tilde{m}}^j$, $q_{\tilde{m}}^j$, x_1, and x_2) respectively. If the matching score is above a fixed threshold τ, we consider l_m^i and $l_{\tilde{m}}^j$ to be potentially matching ($\tau = 0.25$ in all our experiments). However, we only consider constellations as valid matches where an overlap between the line segment $l_{\tilde{m}}^j$ and the virtual segment defined by the

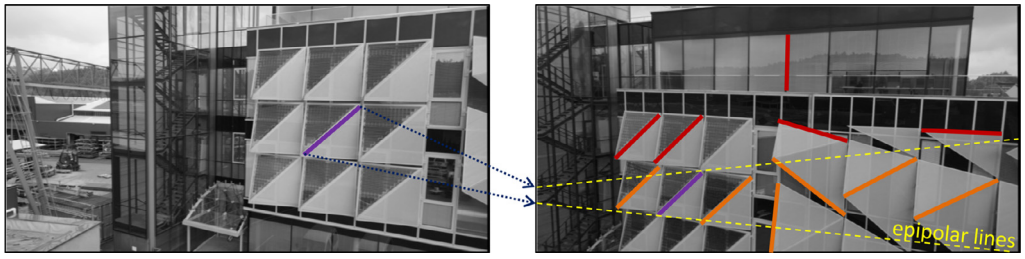


Fig. 3. An example for the epipolar-based matching procedure. *Left:* image I_i with one specific 2D line segment l_m^i (purple). *Right:* The epipolar lines of the endpoints of l_m^i in image I_j are shown in yellow. Accepted matches are shown in orange and purple (correct match), and rejected matches in red. Please note that this is just a visualization and not necessarily the complete set of accepted/rejected matches!. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

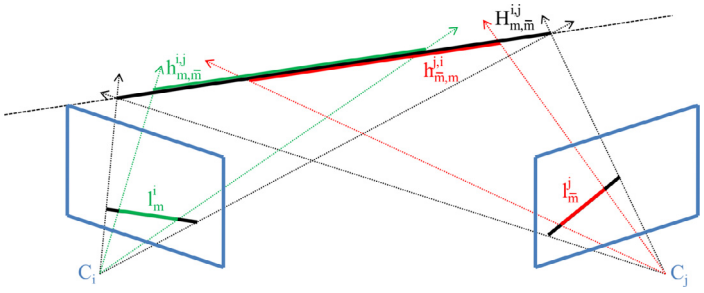


Fig. 4. An illustration how the two 3D hypotheses $h^{i,j}_{m,\bar{m}}$ and $h^{j,i}_{\bar{m},m}$ are computed from two matched 2D segments l^i_m (green) and $l^j_{\bar{m}}$ (red). Both hypotheses are collinear on the underlying (infinite) 3D line $H^{i,j}_{m,\bar{m}}$ (black), but in general not identical, since the endpoints of l^i_m and $l^j_{\bar{m}}$ must not necessarily coincide. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

intersection points x_1 and x_2 actually exists (i.e., when either one or both of the points x_1 and x_2 are in between $l^j_{\bar{m}}$, or vice versa). Fig. 3 illustrates the matching procedure for one source segment l^i_m in one of the neighboring views. This procedure generates a large set of potential matches for each 2D segment, from which only few are correct (in most cases only one per neighboring view is correct). Since all of these matches make some sense in a geometric way (epipolar overlap), we cannot reliably determine incorrect ones during the matching procedure. In the following section we show how we can efficiently filter these wrong correspondences, relying purely on geometric properties.

Since we know all camera poses, we can transform each 2D correspondence into a 3D line $H^{i,j}_{m,\bar{m}}$ by intersecting the two planes passing through the respective camera centers $C_i, C_j \in \mathbb{R}^3$, and the 2D segments. We compute two 3D line segment hypotheses ($h^{i,j}_{m,\bar{m}}$ and $h^{j,i}_{\bar{m},m}$) for each correspondence, which are defined as 3D line segments on $H^{i,j}_{m,\bar{m}}$, whose projected endpoints coincide with the endpoints of the 2D line segments l^i_m and $l^j_{\bar{m}}$ respectively. Similar to the 2D case, a 3D line segment consists of two 3D points ($h^{i,j}_{m,\bar{m}} = \{P^{i,j}_{m,\bar{m}}, Q^{i,j}_{m,\bar{m}}\}$). Note that $H^{i,j}_{m,\bar{m}} = H^{j,i}_{\bar{m},m}$, while in general $h^{i,j}_{m,\bar{m}} \neq h^{j,i}_{\bar{m},m}$ (due to occlusions and imprecise 2D segment detections). However, $h^{i,j}_{m,\bar{m}}$ and $h^{j,i}_{\bar{m},m}$ are always collinear on $H^{i,j}_{m,\bar{m}}$. For a visualization see Fig. 4.

3.2. Evaluating line segment correspondences

The matching procedure enables us to establish a potentially large set of correspondences, most of which are of course incorrect. To distinguish between correct and incorrect matches, we want to compute some kind of confidence value for each of them, after L_i has been matched with all visual neighbors. This can either be done using gradient-based backprojection and scoring of the 3D hypotheses over multiple images (Hofer et al., 2013b; Jain et al., 2010) (which is time consuming), or by directly analysing their 3D similarity to each other (Hofer et al., 2013a, 2014a, 2014b, 2015) (which requires some scale information). Both methods are based on the observation that correct hypotheses of a 2D segment always support each other (e.g., they are close together in 3D space, and project to similar locations in the images), while this does not hold for incorrect ones. We focus on the latter method, which operates directly in 3D and is much more efficient than projective gradient scoring.

The main idea is to find for each 3D segment $h^{i,j}_{m,\bar{m}}$ (associated with the correspondence between $l^i_m \in L_i$ and $l^j_{\bar{m}} \in L_j$) all other 3D segments $h^{i,\cdot}_{m,\cdot}$ (originating from l^i_m as well), that are spatially close to $h^{i,j}_{m,\bar{m}}$ (i.e., that *support* it). The number of different views from which these underlying correspondences emerge is a good indicator whether a 3D segment is likely to exist in reality or not, since it is unlikely that random incorrect correspondences are triangulated to the same spatial position, while all correct matches for one specific 2D segments always end up at the (approximate) same position in space.

In our method, we use a similarity measure based on spatial- and angular errors between 3D hypotheses. We assign a confidence

$$c(h^{i,j}_{m,\bar{m}}) = \sum_{x \in V_i \setminus \{j\}} \max_{y \in \{1,\dots,m_x\}} \left\{ A(h^{i,j}_{m,\bar{m}}, h^{i,x}_{m,y}) \right\}, \tag{3}$$

to a correspondence $h^{i,j}_{m,\bar{m}}$, where A computes an affinity between two 3D hypotheses originating from the same source segment (l^i_m). This affinity is defined as

$$A(h^{i,j}_{m,\bar{m}}, h^{i,x}_{m,y}) = \begin{cases} \min\left\{ S^q(h^{i,j}_{m,\bar{m}}, h^{i,x}_{m,y}), \right. \\ \quad \left. S^p(h^{i,j}_{m,\bar{m}}, h^{i,x}_{m,y}) \right\} & \text{if } \min\{\cdots\} > \frac{1}{2} \\ 0 & \text{else} \end{cases}, \tag{4}$$

with S^a being an *angular* similarity in 3D, and S^p being a *positional* similarity in 3D, defined as

$$S^a(h^{i,j}_{m,\tilde{m}}, h^{i,x}_{m,y}) = \exp\left(-\frac{\angle(h^{i,j}_{m,\tilde{m}}, h^{i,x}_{m,y})^2}{2\sigma_a^2}\right) \tag{5}$$

and

$$S^p(h^{i,j}_{m,\tilde{m}}, h^{i,x}_{m,y}) = \min_{Z \in h^{i,j}_{m,\tilde{m}}} \exp\left(-\frac{d_\perp(Z, h^{i,x}_{m,y})^2}{2\sigma_p^i(d_i(Z))^2}\right), \tag{6}$$

where $\angle(h_1, h_2)$ denotes the angle between the two line segments (in degrees), $d_\perp(Z, h_2)$ is the normal distance between the 3D point Z and the infinite line passing through h_2, and $d_i(Z) = \|C_i - Z\|_2$ is the Euclidean distance between the camera center C_i of I_i and Z (i.e., its depth along its corresponding camera ray). To prevent that the confidence gets high if we only have several weak supporters (i.e., many non-zero elements in the sum in Eq. 3, but no strong support among them), we truncate the affinity and only accept values above 1/2.

Both formulas require a regularization parameter σ. For the angular case (Eq. 5), we can chose a reasonable value very easily, since angles are scale invariant. However, for the spatial case (Eq. 6) this is non trivial. Even if we know the scale of the reconstruction, choosing a constant value for all 3D hypotheses is not recommended, since the positional uncertainty of 3D line hypotheses h also depends strongly on the distances to the cameras from which they were triangulated. Hence, choosing a value that is too large will produce a bias towards hypotheses that are very close to the cameras, while a small value will prevent that hypotheses farther away from the cameras can reach reasonable scores. To prevent this, one possibility would be to use a 2D re-projection error for scoring, as we have shown in Hofer et al. (2015). However, since the re-projection error can easily be very small despite a potentially very large spatial displacement (which is especially frequent when dealing with small baselines), we recommend to do the scoring directly in the 3D space. We use a *depth adaptive* spatial regularization function $\sigma_p^i(d_i(Z))$, similar to the one we first proposed in Hofer et al. (2014b).

This regularization function is defined for each image I_i individually as

$$\sigma_p^i(d) = d \cdot \mu_\sigma, \tag{7}$$

which is a linear function in the depth d. Its slope, μ_σ, can be derived from the underlying camera geometry. Given a standard pinhole camera model, we do this by shifting the principle point $\tilde{p}_p$ horizontally by a 2D regularizer σ (defined in pixels), to obtain a second point $\tilde{p}_p^\sigma$ (in homogeneous coordinates), and then computing the angle β between the two 3D rays $K^{-1} \cdot \tilde{p}_p$ and $K^{-1} \cdot \tilde{p}_p^\sigma$, where K^{-1} are the cameras' inverse intrinsics. We then simply compute $\mu_\sigma = \sin(\beta)$, which is basically the maximum distance we can move the unprojection of the principal point at depth $d = 1$, such that the distance between the re-projection of the moved point and the principal point in the image is less or equal to σ (see Fig. 5). This formulation ensures that higher 3D point-to-line distances ($d_\perp(Z, h)$ in Eq. 6) are punished less when the respective hypotheses are farther away from their observing cameras, and vice versa.

As the angular similarity S^a, the positional similarity S^p is scale invariant as well, since it only depends on a regularization parameter σ defined in the image space. This is especially beneficial when the exact reconstruction scale is unknown at runtime, which is often the case in SfM applications. However, when the scale is known (e.g., metric) one can of course also incorporate this information by e.g., reformulating the regularization function as

$$\sigma_p^{3D}(d) = d \cdot \frac{\mu_{3D}}{d_{\text{med}}}, \tag{8}$$

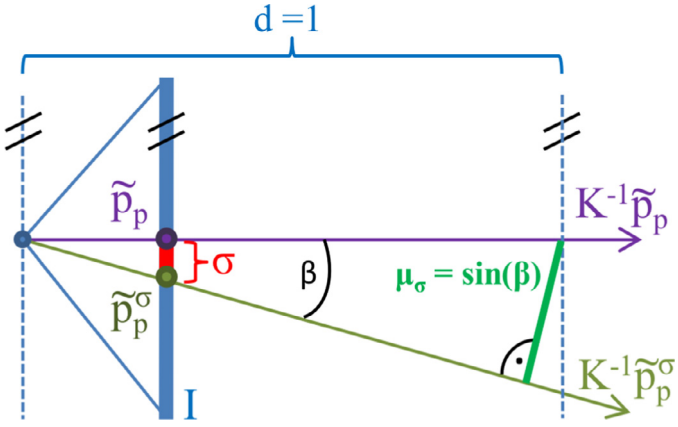


Fig. 5. Derivation of the slope μ_σ of the spatial regularization function $\sigma_p^i(d)$, based on the camera geometry and the regularization parameter σ. The image plane I is shown from the top.

where μ_{3D} denotes a user specified constant spatial regularizer (e.g., $\mu_{3D} = 0.05$, for 5 cm when the scale is metric), and d_{med} stands for the median scene depth (e.g., the median distance over all 3D points X to their respective camera centers). As above, this allows depth adaptive scoring as well, since it linearly scales the regularizer with the deviation from the median scene depth. However, for the sake of simplicity we will consider the scale invariant case (Eq. 7) throughout the remainder of the paper (our open-source implementation can handle both cases).

With the confidence formulation from Eq. 3 we are now able to determine whether a matching hypothesis makes sense or not. We only keep hypotheses for further processing for which $c(h^{i,j}_{m,\tilde{m}}) > 1$, which means that at least two segments from two additional images (apart from I_i and I_j) have to support $h^{i,j}_{m,\tilde{m}}$. We end up with a much sparser set of correspondences, with a significantly lower number of outliers, while correct hypotheses are only seldom removed (e.g., when a segment is occluded in many visual neighbors). Fig. 6a shows a visualization of all verified correspondences, that remained after the previous verification step. The building can be clearly recognized, even though several outlier hypotheses remain (e.g., in the sky).

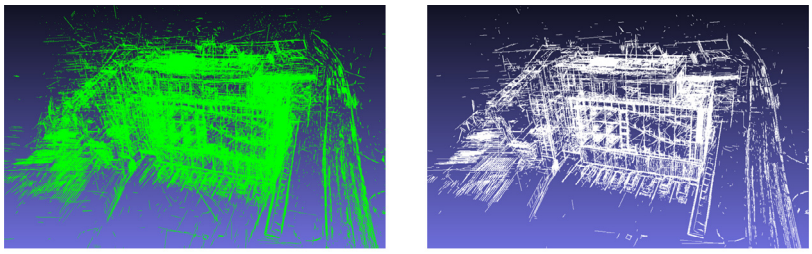
3.3. Assigning 3D locations to 2D segments

Given all hypotheses $h^{i,j}_{m,\tilde{m}}$ for a 2D segment l^i_m, we want to estimate its most probable 3D position, since each 2D segment can only be a projection of one specific 3D structure. We use this 3D information for the following clustering procedure, as we have first shown in Hofer et al. (2014a, 2014b). For each 2D segment l^i_m we define its 3D location as

$$\hat{h}^i_m = \arg\max_{h^{i,j}_{m,\tilde{m}}} \left\{ c\left(h^{i,j}_{m,\tilde{m}}\right) \right\}, \tag{9}$$

which is simply its 3D hypothesis with the highest confidence. As stated above, we only keep hypotheses with a confidence bigger than one, which means that each remaining hypothesis is supported by ≥ 4 images (see Section 3.2). We have seen that 3D segment hypotheses verified by four or more images are rarely incorrect, which can also be observed for SfM point-clouds on the 3D point level. Fig. 6b shows all selected 3D hypotheses $\{\hat{h}^i_m\}$ for the *Building* sequence. As we can see, the majority of the 2D segments selects its correct 3D position, with only few isolated outliers remaining.

Please cite this article as: M. Hofer et al., Efficient 3D scene abstraction using line segments, Computer Vision and Image Understanding (2016), http://dx.doi.org/10.1016/j.cviu.2016.03.017



(a) All scored hypotheses $\{h^{i,j}_{m,\bar{m}}\}$ (b) Best hypotheses $\{\hat{h}^i_m\}$

Fig. 6. *Left:* All hypotheses $h^{i,j}_{m,\bar{m}}$ for which $c(h^{i,j}_{m,\bar{m}}) > 1$ ($\approx$650K). *Right:* The set of all selected hypotheses $\{\hat{h}^i_m\}$ (i.e., the one hypothesis with the highest confidence for each 2D segment across all images, $\approx$ 100K). The building is clearly recognizable in both images, while the majority of the remaining outliers has been removed on the right.

3.4. Clustering 2D segments across images

After the previous step we end up with an individual 3D estimate $\hat{h}^i_m$ for each 2D line segment l^i_m. To compute one consistent 3D model, we have to fuse corresponding 2D segments (i.e., segments that originate from the same 3D entity) and their 3D estimates together. As we have shown in our previous work (Hofer et al., 2014b, 2015), this can be efficiently done by employing a graph clustering procedure, which operates on a global affinity matrix W, encoding the pairwise similarities between potentially corresponding 2D segments across all images. Since we only need to consider segment pairs which have been successfully matched (i.e., for which a valid 3D hypothesis exists), this matrix is usually very sparse. To compute these similarities, we use the same metric as for the hypothesis confidence above (see Eq. 4). Hence, the affinity between two potentially matching 2D segments l^i_m and $l^j_{\bar{m}}$ can be computed straightforwardly as

$$W(l^i_m, l^j_{\bar{m}}) = \begin{cases} \min\left\{S^a(\hat{h}^i_m, \hat{h}^j_{\bar{m}}), \right. \\ \quad \left. \tilde{E}^p(\hat{h}^i_m, \hat{h}^j_{\bar{m}})\right\} & \text{if } \min\{\cdots\} > \frac{1}{2} \\ 0 & \text{else} \end{cases}, \tag{10}$$

where S^a is the basic angular similarity (Eq. 5), and $\tilde{E}^p$ is a symmetric positional similarity defined as

$$\tilde{E}^p(\hat{h}^i_m, \hat{h}^j_{\bar{m}}) = \min\left\{\tilde{S}^p(\hat{h}^i_m, \hat{h}^j_{\bar{m}}),\ \tilde{S}^p(\hat{h}^j_{\bar{m}}, \hat{h}^i_m)\right\}, \tag{11}$$

which is computed using a slightly modified version of S^p (referring to Eq. 6)

$$\tilde{S}^p(\hat{h}^i_m, \hat{h}^j_{\bar{m}}) = \min_{Z \in \hat{h}^i_m} \exp\left(-\frac{d_\perp(Z, \hat{h}^j_{\bar{m}})^2}{2\tilde{\sigma}^i_p(d_i(Z))^2}\right), \tag{12}$$

with a different spatial regularizer

$$\tilde{\sigma}^i_p(d) = \begin{cases} d \cdot \mu_\sigma & \text{if } d < D_i \\ D_i \cdot \mu_\sigma & \text{otherwise} \end{cases}, \tag{13}$$

which is truncated at a certain depth D_i, to avoid the possibility that the allowed spatial uncertainty grows too large for points far away from the camera center. This is not necessary during the scoring procedure, but for the reconstruction we want to prevent potentially imprecise line clusters that might occur far away from the observing cameras. To obtain a reasonable estimate for D_i (even when the scale is unknown), we define it as the median depth over all final 3D hypotheses $\hat{h}^i_m$ (for each image I_i individually), by using both segment endpoints.

The resulting affinity matrix W could now be directly fed to an arbitrary graph clustering algorithm, which takes a simple pairwise affinity matrix as an input. In our earlier approaches (Hofer et al., 2014a, 2014b), we used the popular method by Felzenszwalb and Huttenlocher (2004) as a clustering algorithm, which delivers visually pleasant results for the general case, without any kind of parameter tuning. In our most recent work (Hofer et al., 2015), we experimented with a different clustering strategy (Donoser, 2013), which is based on diffusing the given affinity matrix W, by implicitly considering the underlying data manifold. The diffused matrix is then segmented by the same graph clustering method as before Felzenszwalb and Huttenlocher (2004). In both cases, we only accept clusters if they contain 2D segments from at least three different images.

In practice the choice of the clustering method has a minor influence on the resulting 3D model. In general, the result obtained using Donoser (2013) manages to cluster together more 3D segments and obtains a slightly denser result, at no additional cost regarding computational time (due to a sparser affinity matrix). However, the matrix diffusion step requires extra memory which is in our experience only relevant for very large datasets (>1000 images) and on small machines. In these cases, we recommend to use Felzenszwalb and Huttenlocher (2004) only, which scales almost linearly with the number of images in terms of runtime and memory consumption.

Regardless of the method, the output of the clustering algorithm is always a set of 3D line clusters $\Pi = \{\Pi_1, \ldots, \Pi_T\}$, where each cluster Π_t consists of a set of 2D residuals $L_{\Pi_t} = \{l_1, l_2, \ldots\}$, a representative 3D line H_{Π_t}, and one or more collinear 3D line segments $h_{\Pi_t} = \{h_1, \ldots\}$ (which are a part of the infinite line H_{Π_t}). To obtain the underlying 3D line H_{Π_t} for a cluster Π_t, we use the 3D depth estimates ($\hat{h}$) of its 2D residuals, as first shown in Jain et al. (2010). The line direction can be computed by a Singular Value Decomposition ($\{U, \Sigma, V\}$) of the scatter matrix containing all endpoints of the clustered 3D segment hypotheses, and taking the column of U corresponding to the highest singular value. A point on the line can easily be obtained by computing the center of gravity of all the 3D segment endpoints of the cluster. To obtain the final 3D line segment set h_{Π_t} on H_{Π_t} (i.e., the actually visible part of the infinite line H_{Π_t}, with respect to its 2D observations L_{Π_t}), we project the 3D estimates ($\hat{h}$) of all 2D residuals L_{Π_t} onto the newly estimated 3D line H_{Π_t}, and compute a set of non overlapping 3D line segments on this line, such that each of these segments is fully covered by at least three of the projected hypotheses (from at least three different images). Fig. 7 shows the final 3D models for the *Building* sequence using the two different clustering approaches.

3.5. Combined bundle adjustment

The resulting set of 3D line clusters Π obtained in the previous section can be used as the final output of our pipeline, and is in general very accurate (with respect to the used regularization parameters σ and σ_a). However, since the obtained 3D lines are more or less an average over several 3D hypotheses, which are only two-view triangulations, there is no guarantee that they fit the 2D observations in an optimal way. To overcome this issue,

Please cite this article as: M. Hofer et al., Efficient 3D scene abstraction using line segments, Computer Vision and Image Understanding (2016), http://dx.doi.org/10.1016/j.cviu.2016.03.017

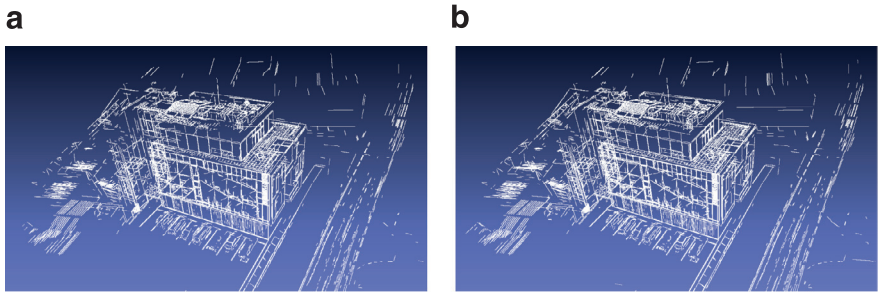


Fig. 7. Final line-based 3D models of the *Building* sequence obtained by *Line3D* when (a) directly using the graph-clustering method by Felzenszwalb and Huttenlocher (2004), or (b) when pre-processing the affinity matrix W by using matrix diffusion (Donoser, 2013). Both results are fairly similar. In general, the diffusion based approach manages to reconstruct slightly more parts of the scene.

one could easily formulate a bundle adjustment procedure (Triggs et al., 2000), to optimize the 3D lines with respect to their 2D residuals (i.e., minimizing the re-projection error). In addition, we could go even further and don't just optimize the 3D lines alone, but also the whole SfM result (points and camera poses) using the newly obtained 2D to 3D correspondences, that are provided by the reconstructed line clusters.

While there are slight differences in the exact formulation in various SfM pipelines, bundling is usually done by minimizing a non-linear least squares problem, consisting of the re-projection error of the reconstructed 3D points to their 2D residuals, with respect to the camera poses. Analogue to a 3D line cluster Π_t, a reconstructed 3D point by the SfM can be seen as a cluster Ω_t of 2D points $P_{\Omega_t} = \{p_1, p_2, \ldots\}$ $(p \in \mathbb{R}^2)$, and an associated 3D position $h_{\Omega_t} \in \mathbb{R}^3$. For points and lines, the combined optimization problem can be defined as

$$f^* = \min_{\{R,t\},\Omega,\Pi} \sum_{\Omega_t \in \Omega} f^P(\Omega_t) + \lambda \sum_{\Pi_t \in \Pi} f^L(\Pi_t), \tag{14}$$

where R, t are the rotation and translation of the cameras, and f^P and f^L are error functions for 3D points and lines respectively, with a scalar weighting factor λ. For the sake of simplicity, we do not include the cameras' intrinsics and distortion coefficients into the bundle adjustment (but they can be added straightforwardly). In our case, the error functions are defined as

$$f^P(\Omega_t) = \sum_{p^x \in P_\Omega} \rho_\epsilon \left(d^P_{2D}(\Gamma_x(h_\Omega), p^x) \right) \tag{15}$$

and

$$f^L(\Pi_t) = \sum_{l^x \in L_\Pi} \rho_\epsilon \left(d^L_{2D}(\Gamma_x(h_\Pi), l^x) \right), \tag{16}$$

where ρ_ϵ is the robust Huber loss function (linearised at $\epsilon = 2$px), Γ_x projects a 3D point or line into image I_x, and d^P_{2D} and d^L_{2D} are error functions for 2D points and lines respectively. For the point case d^P_{2D}, a common choice is the Euclidean distance between the projected $(\Gamma_x(h_\Omega))$ and the observed (p^x) 2D point.

For lines this is slightly more complicated. We define the error function as

$$d^L_{2D}(l_1, l_2) = \left(\sum_{p \in l_2} d^{2D}_\perp(p, l_1) \right) \cdot \exp(2 \cdot \angle(l_1, l_2)), \tag{17}$$

where $d^{2D}_\perp(p, l)$ is the normal distance from an endpoint p to a line l (analogue to Eq. 6 for the 3D case), and $\angle(l_1, l_2)$ is the angle between the two lines l_1 and l_2 ($\in [0, \pi/2]$). The second term punishes directional deviations by increasing the positional error if the angle between the projected and the observed 2D segment is large. We have seen that this additional term leads to a much faster convergence, without negatively affecting the results. To avoid over-parametrization, we use the Cayley 3D line representation for bundling, as suggested in Zhang and Koch (2014). It has the benefit that it only needs four parameters to define a 3D line, which is an optimal encoding since a 3D line has exactly four degrees of freedom. We solve the optimization problem by using the powerful CERES solver (Agarwal et al.), which offers built-in support for bundle adjustment and is very efficient even for large-scale problems.

In the next section, we evaluate the effects of the combined bundle adjustment procedure on several real- and synthetic datasets, with given camera poses and a ground truth 3D surface model.

4. Experimental results

We demonstrate the capabilities of our algorithm on several challenging real-world datasets, and quantitatively evaluate our method on several publicly available datasets with ground truth. We further set our line-based reconstructions in relation to conventional dense point-clouds, obtained from PMVS (Furukawa et al., 2010), to give an idea of the pros and cons of both methods in terms of runtime vs. level of abstraction.

The parameters are kept fixed for all datasets. We set the confidence regularisation parameters to $\sigma = 2.5$ pixels and $\sigma_a = 10^\circ$, and the number of visual neighbors $M = 10$. For the bundle adjustment optimization, we dynamically compute the weighting term between the point and line features as $\lambda = |\Omega| / |\Pi|$ (Eq. 14), which balances the weight between points and lines such that both parts contribute equally.

As a line segment detector we use LSD (von Gioi et al., 2010). To speed-up the line segment detection procedure, we resize large images to a fixed with of 1920 pixels ($\approx$ FullHD), where *width* actually means the larger of the two image dimensions. We upscale the line segment coordinates to the proper size after the detection procedure, and operate on the full image dimensions in the rest of the algorithm. The line segment detection is of course also possible on the full image scale, but we have seen that this unnecessarily increases the runtime with almost no benefits for the accuracy or completeness of the 3D model. As an SfM method we use *ICG3D* (Irschara et al., 2007), which is a conventional pipeline based on SIFT (Lowe, 2004) features and a vocabulary tree (Nister and Stewenius, 2006) for matching (we limit ourselves to the 5000 largest SIFT features per image). For all tests, we use the diffusion-based clustering approach. Our algorithm is implemented in C++ and (optionally) also CUDA, making use of parallel computing whenever possible (especially during the matching and hypotheses scoring). Our test system is an ordinary desktop machine,

Please cite this article as: M. Hofer et al., Efficient 3D scene abstraction using line segments, Computer Vision and Image Understanding (2016), http://dx.doi.org/10.1016/j.cviu.2016.03.017

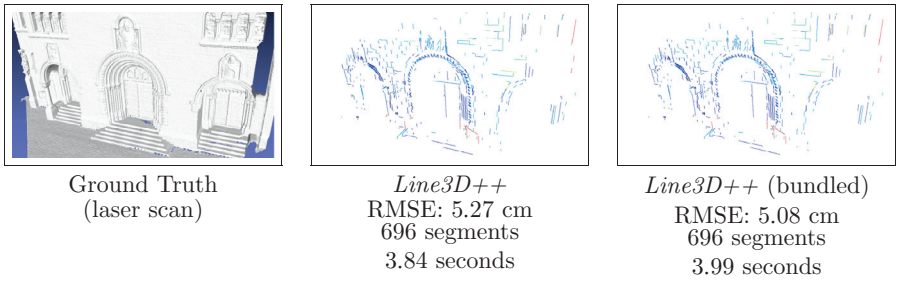


Fig. 8. Quantitative evaluation on the *Herz-Jesu-P8* (Strecha et al., 2008) dataset (8 images). The Root-Mean-Square Error (RMSE) is computed for densely sampled points on the 3D lines to the ground truth mesh. The colors indicate the RMSE using a linear color mapping (blue: 0.0, green: 0.05, red: ≥ 0.1). In the middle we see the unbundled-, and on the right the bundled result. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

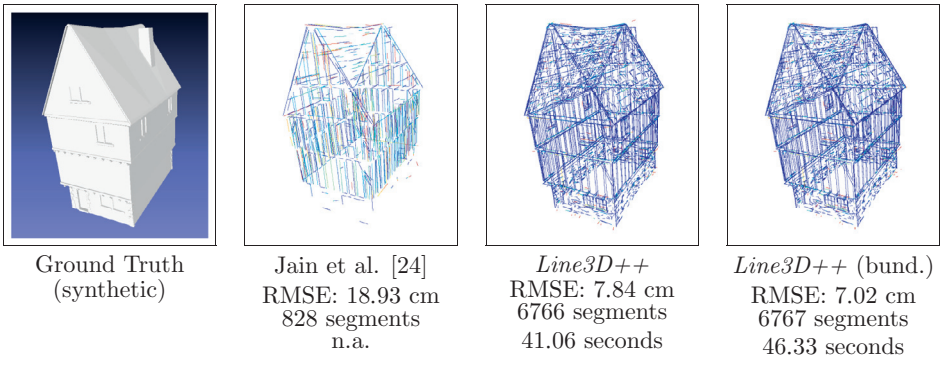


Fig. 9. Quantitative evaluation on the *Timberframe* (Jain et al., 2010) synthetic dataset (240 images). The RMSE is computed for densely sampled points on the 3D lines to the ground truth CAD model. The colors indicate the RMSE using a linear color mapping (blue: 0.0, green: 0.25, red: ≥ 0.5). On the left, we see the original result by Jain et al. (2010). In the middle we see our unbundled-, and on the right our bundled result. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

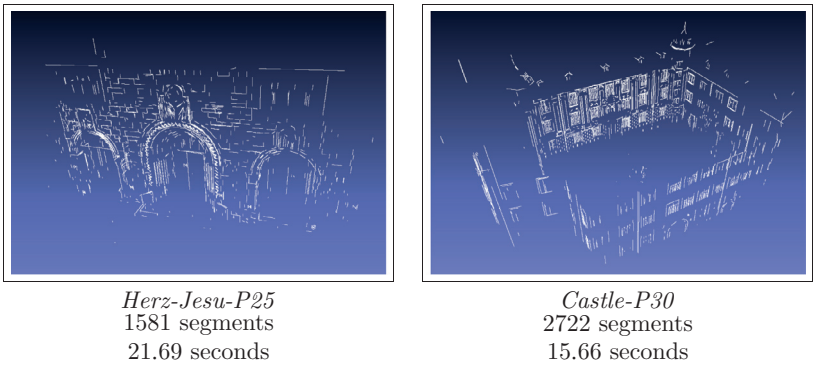


Fig. 10. Reconstruction results for the *Herz-Jesu-P25* (25 images), and the *Castle-P30* (30 images) datasets (Strecha et al., 2008).

equipped with an *Intel Core i5* CPU (4 × 3.4 GHz), 12 GB of main memory, and an *nVidia GeForce GTX 580 Ti* GPU.

Fig. 8 shows a quantitative evaluation of our method on the *Herz-Jesu-P8* (Strecha et al., 2008) dataset (8 images), using the unbundled (*left*) and the bundled result (*right*). The lines are color-coded by their RMSE to the ground truth surface. As we can see, the bundled result has a slightly higher accuracy, but the difference is only marginal for this small-scale dataset. Please note that not all valid 3D lines are actually contained in the ground truth. This is especially notable on the railings at the main entrance (colored in red).

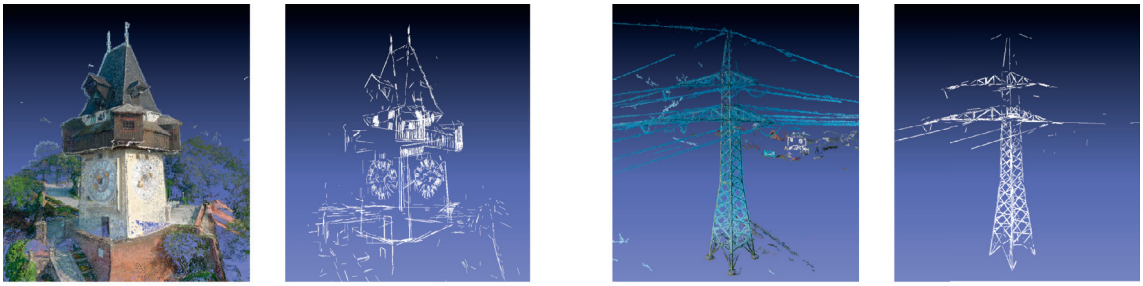
Fig. 9 shows an evaluation on the synthetic *Timberframe*[2] dataset (240 images), as well a comparison to the original reconstruction result by Jain et al. (2010). As we can see, our method manages to reconstruct the house more densely and with a lower RMSE to the ground truth surface (in both variants). In their paper no explicit runtimes are given, but it is stated that the reconstruction can easily take several hours. Our method finishes in less than a minute, which is mostly due to the more efficient matching procedure and the massive parallelism. As expected, the bundled result (*right*) has a slightly lower RMSE than the unoptimized version (*left*).

Additionally, we evaluate the accuracy of the camera parameters before- and after the combined bundle adjustment. For this purpose, we use three ground truth test sequences: the *Herz-Jesu-P25* and *Castle-P30* from Strecha et al. (2008) (reconstruction results can be seen in Fig. 10), and the synthetic *Timberframe* sequence, already introduced above. To evaluate the influence of the line segments on the estimation of the camera poses, we compute positional- (distance between the measured and reconstructed camera centers) and orientation (angle between the measured and reconstructed optical axes) errors to the ground truth poses, as well as the mean re-projection error over all features.

[2] http://resources.mpi-inf.mpg.de/LineReconstruction/.

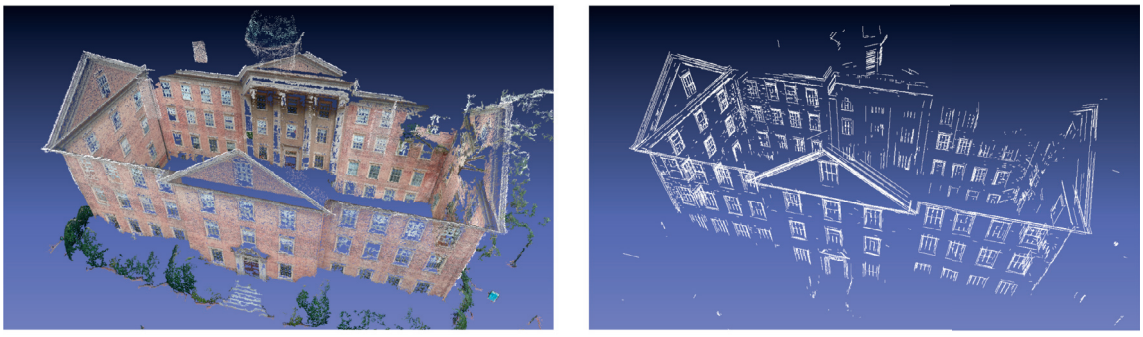
Table 1
Camera pose evaluation results on three ground truth datasets.

Dataset	SfM (Irschara et al., 2007) (points only)					SfM (Irschara et al., 2007) + Lines (proposed)				
	Pos.Err. [cm]		Ori.Err. [deg]		Repr.-err.	Pos.Err. [cm]		Ori.Err. [deg]		Repr.-err.
	mean	max	mean	max	mean	mean	max	mean	max	mean
Herz-Jesu-P25 (Strecha et al., 2008)	0.646	1.678	0.174	0.230	1.217	**0.608**	**1.268**	**0.156**	**0.191**	**0.665**
Castle-P30 (Strecha et al., 2008)	3.584	9.773	**0.227**	0.375	1.272	**3.267**	**6.947**	0.292	**0.373**	**0.555**
Timberframe (Jain et al., 2010)	3.462	8.763	**0.065**	0.225	0.695	**2.417**	**7.176**	0.068	**0.209**	**0.560**



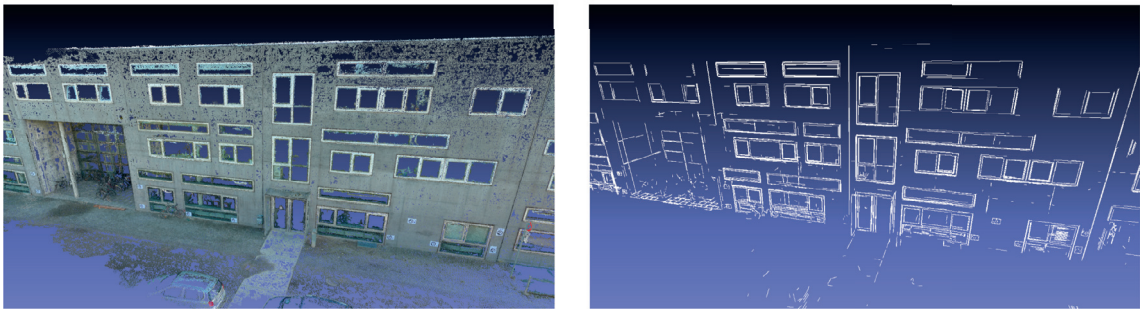
2, 290, 578 pts, 3.38 hrs | 2, 587 lines, 207.02 sec | 662, 459 pts, 1.58 hrs | 2, 119 lines, 40.04 sec

Clocktower, 403 images, 3968 × 2232px | *Pylon*, 89 images, 4320 × 3240px



1, 703, 759 pts, 1.41 hrs | 5, 953 lines, 89.79 sec

Brickstone, 127 images, 3072 × 2304px



5, 329, 945 pts, 6.37 hrs | 4, 321 lines, 91.34 sec

Facade, 317 images, 3684 × 2736px

Fig. 11. Qualitative reconstruction results on real-world datasets. *Left:* PMVS (Furukawa et al., 2010), *Right: Line3D++* (with matrix diffusion (Donoser, 2013) and bundle adjustment).

Please cite this article as: M. Hofer et al., Efficient 3D scene abstraction using line segments, Computer Vision and Image Understanding (2016), http://dx.doi.org/10.1016/j.cviu.2016.03.017

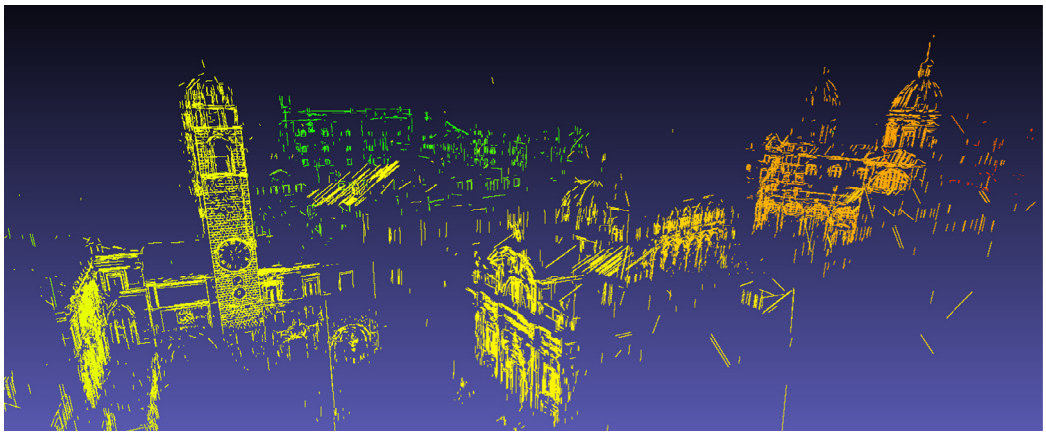


Fig. 12. Reconstruction result for the large crowdsourced *Dubrovnik6K* (Li et al., 2010) dataset (6844 images; 21, 040 3D lines). The total runtime was approximately 90 min (excluding SfM).

Table 1 shows the evaluation results. As we can see, adding the reconstructed 3D lines to the optimization procedure improves the camera positions in all cases, while the orientation errors stay almost the same. However, the differences are quite small, since the SfM pipeline already manages to obtain very accurate camera poses to begin with.

Fig. 11 shows additional qualitative results for several real-world test sequences. Please note that the runtime for PMVS is measured in hours, while for our method it is in seconds (the stated runtime includes all steps from line segment detection to the final 3D model). As can be seen, our method generates virtually outlier-free results very efficiently. The comparison to the dense point-clouds underlines once more how a lot of 3D information can be extracted in a very short time when only straight line segments are used as features. Our 3D line models give the viewer a very good impression of what is going on in the scene, but in a more compact way than a large dense point-cloud.

To further emphasize the efficiency of our algorithm, we show a line-based 3D reconstruction for the crowdsourced *Dubrovnik6K* (Li et al., 2010) dataset (6844 images) in Fig. 12. As we can see, our method is able to successfully process this very large-scale dataset, with a total runtime of just 90 min. Since the scale is metric, and the images have various different resolutions (since they have been taken by several different cameras), we use a spatial regularizer $\mu_{3D} = 0.025$ (which is 2.5 cm) defined in world coordinates (see Eq. 8). As the underlying SfM result we used the *bundler* file (Snavely et al., 2006) which is provided with the dataset. The obtained result shows satisfying 3D models, especially in densely photographed areas of the city (e.g., churches). However, due to the generally low image resolutions the amount of noise is slightly higher compared to the results shown in Fig. 11. In general, we recommend the usage of one consistent high-resolution camera over crowdsourced data, but this experiment shows that our method can handle such cases as well.

5. Conclusion

We presented our method to generate abstract 3D models for built environments, by using straight line segments as features. We have shown how a significant amount of 3D information about a scene can be encoded with a small amount of data in a very short time, in contrast to using large point-clouds. However, our goal was not to replace dense 3D reconstruction, but rather to provide an alternative for all scenarios in which 3D edge information is preferred over a point-cloud (e.g., plane detection (Kim and Manduchi, 2014), or surface reconstruction (Sugiura et al., 2015)).

Our method is suitable for reconstructing large-scale urban datasets, due to the efficient matching and clustering strategies, which scale almost linearly with the number of input images. Most of the parts of the pipeline can be parallelized, which further improves the performance on modern machines. An implementation in C++ and (optionally) CUDA is available for download on github: https://github.com/manhofer/Line3Dpp.

At the moment, our approach can only reconstruct piecewise linear 3D structures, since it is based on 2D line segments. As future work, we intend to extend our method to curved edges as well, e.g., by using lines and elliptical arcs combined (Patraucean et al., 2012), which both frequently occur in man-made environments. In addition, we want to investigate the possibility of using the obtained 3D edge features for subsequent tasks, such as surface reconstruction, or semantic scene classification.

Acknowledgements

This work has been supported by the Austrian Research Promotion Agency (FFG) project FreeLine (Bridge1/843450) and OMICRON electronics GmbH.

References

Agisoft photoscan. http://www.agisoft.com/.
Agarwal, S., Mierle, K., Others,. Ceres solver. http://ceres-solver.org.
Akinlar, C., Topal, C., 2011. Edlines: real-time line segment detection by edge drawing. Int. Conf. Image Process 18, 2837–2840.
Ayache, N., Faverjon, B., 1987. Efficient registration of stereo images by matching graph descriptions of edge segments. Int. J. Comput. Vis 1 (2), 107–131.
Baillard, C., Schmid, C., Zisserman, A., Fitzgibbon, A., 1999. Automatic line matching and 3D reconstruction of buildings from multiple views. ISPRS Confer. Autom. Extract. GIS Objects Digit. Imagery 32, 69–80.
Bartoli, A., Coquerelle, M., Sturm, P., 2004. A framework for pencil-of-points structure-from-motion. Eur. Confer. Comput. Vis 8, 28–40.
Bartoli, A., Sturm, P., 2005. Structure-from-motion using lines: representation, triangulation, and bundle adjustment. Comput. Vis. Image Understanding 100 (3), 416–441.
Bay, H., Ess, A., Neubeck, A., van Gool, L., 2006. 3D from line segments in two poorly-textured, uncalibrated images. Int. Symp. 3D Data Process. Vis. Transmission 3, 496–503.
Bay, H., Ferrari, V., van Gool, L., 2005. Wide-baseline stereo matching with line segments. Int. Confer. Comput. Vis. Pattern Recog 1, 329–336.
Capturingreality. https://www.capturingreality.com/.
Donoser, M., 2013. Replicator graph clustering. British Mach. Vis. Confer 24.
Elqursh, A., Elgammal, A., 2011. Line-based relative pose estimation. Int. Confer. Comput. Vis. Pattern Recog 3049–3056.
Felzenszwalb, P., Huttenlocher, D., 2004. Efficient graph-based image segmentation. Int. J. Comput. Vis 59 (2), 167–181.
Frahm, J.-M., Fite-Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y.-H., Dunn, E., Clipp, B., Lazebnik, S., Pollefeys, M., 2010. Building rome on a cloudless day. Eur. Confer. Comput. Vis 11, 368–381.
Furukawa, Y., Curless, B., Seitz, S., Szeliski, R., 2010. Towards internet-scale multi-view stereo. Int. Confer. Comput. Vis. Pattern Recog 1434–1441.

Havlena, M., Schindler, K., 2014. Vocmatch: efficient multiview correspondence for structure from motion. Eur. Confer. Comput. Vis.
Hofer, M., Donoser, M., Bischof, H., 2010a. Semi-global 3D line modeling for incremental structure-from-motion. British Mach. Vis. Confer.
Hofer, M., Maurer, M., Bischof, H., 2014b. Improving sparse 3D models for man-made environments using line-based 3D reconstruction.
Hofer, M., Maurer, M., Bischof, H., 2015. Line3D: Efficient 3D scene abstraction for the built environment. German Confer. Comput. Vis. Pattern Recog.
Hofer, M., Wendel, A., Bischof, H., 2013a. Incremental line-based 3D reconstruction using geometric constraints. British Mach. Vis. Confer.
Hofer, M., Wendel, A., Bischof, H., 2013b. Line-based 3D reconstruction of wiry objects. Comput. Vis. Winter Workshop.
Irschara, A., Zach, C., Bischof, H., 2007. Towards wiki-based dense city modeling. Int. Confer. Comput. Vis.
Jain, A., Kurz, C., Thormaehlen, T., Seidel, H.-P., 2010. Exploiting global connectivity constraints for reconstruction of 3D line segments from images.
Kim, C., Manduchi, R., 2014. Planar structures from line correspondences in a manhattan world. Asian Confer. Comput. Vis.
Labatut, P., Pons, J.-P., Keriven, R., 2007. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. Int. Confer. Comput. Vis.
Li, Y., Snavely, N., Huttenlocher, D.P., 2010. Recognition using prioritized feature matching. Eur. Confer. Comput. Vis.
Lowe, D., 2004. Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis.
Micusik, B., Wildenauer, H., 2014. Structure from motion with line segments under relaxed endpoint constraints. Int. Confer. 3D Vis.n (3DV).
Moulon, P., Monasse, P., Marlet, R., Others,. OpenMVG: an open multiple view geometry library. https://github.com/openMVG/openMVG.
Nister, D., Stewenius, H., 2006. Scalable recognition with a vocabulary tree. Int. Confer. Comput. Vis. Pattern Recog. Int.
Patraucean, V., Gurdjos, P., von Gioi, R.G., 2012. A parameterless line segment and elliptical arc detector with enhanced ellipse fitting. Eur. Confer. Comput. Vis.
Raposo, C., Antunes, M., Barreto, J.P., 2014. Piecewise-planar stereoscan: Structure and motion from plane primitives. Eur. Confer. Comput. Vis.
Rothermel, M., Wenzel, K., Fritsch, D., Haala, N., 2012. SURE: Photogrammetric surface reconstruction from imagery. LCD Workshop.
Schindler, G., Krishnamurthy, P., Dellaert, F., 2006. Line-based structure from motion for urban environments. Int. Symp. 3D Data Process. Vis. Transmission.
Schmid, C., Zisserman, A., 1997. Automatic line matching across views. Int. Conf. Comput. Vis. Pattern Recog.
Schoenberger, J.L., Fraundorfer, F., Frahm, J.-M., 2014. Structure-from-motion for MAV image sequence analysis with photogrammetric applications. Int. Archives Photogram. Remote Sensing Spat. Inf.n Sci.
Sinha, S.N., Steedly, D., Szeliski, R., 2009. Piecewise planar stereo for image-based rendering. Int. Confer. Comput. Vis.
Snavely, N., Seitz, S., Szeliski, R., 2006. Photo tourism: Exploring image collections in 3D. ACM Trans. Graph.
Strecha, C., von Hansen, W., Gool, L.V., Fua, P., Thoennessen, U., 2008. On benchmarking camera calibration and multi-view stereo for high resolution imagery. Int. Confer. Comput. Vis. Pattern Recog.
Strecha, C., Kueng, O., Others,. Pix4D, UAV mapping software. https://pix4d.com/.
Sugiura, T., Torii, A., Okutomi, M., 2015. 3D surface reconstruction from point-and–line cloud. Int. Confer. 3D Vis.
Sweeney, C.,. Theia multiview geometry library: Tutorial and Reference. University of California Santa Barbara.
Triggs, B., McLauchlan, P., Hartley, R., Fitzgibbon, A., 2000. Bundle adjustment: a modern synthesis. Vis. Algorithms: Theory and Practice.
von Gioi, R.G., Jakubowicz, J., Morel, J.-M., Randall, G., 2010. LSD: a fast fine segment detector with a false detection control.
Waechter, M., Moehrle, N., Goesele, M., 2014. Let there be color! large-scale texturing of 3D reconstructions. Eur.n Confer. Comput. Vis.
Wang, Z., Wu, F., Hu, Z., 2009. MSLD: a robust descriptor for line matching. Pattern Recog.
Wu, C., 2013. Towards linear-time incremental structure-from-motion. Int. Confer. 3D Vis.
Zhang, L., Koch, R., 2012. Line matching using appearance similarities and geometric constraints. Lecture Notes in Comput. Sci.: Pattern Recog.
Zhang, L., Koch, R., 2014. Structure from motion from line correspondences: Representation, projection, initialization and sparse bundle adjustment. J. Vis. Commun. Image Representation.
Zhang, L., Xu, C., Lee, K.-M., Koch, R., 2012. Robust and efficient pose estimation from line correspondences. Asian Confer. Comput. Vis.
Zhang, Y., Yang, H., Liu, X., 2011. A line matching method based on local and global appearance. Int. Congress Image Signal Process.

Please cite this article as: M. Hofer et al., Efficient 3D scene abstraction using line segments, Computer Vision and Image Understanding (2016), http://dx.doi.org/10.1016/j.cviu.2016.03.017

Manuel Hofer received his B.S. and M.S. degree in Computer Science from Graz University of Technology in 2010 and 2012, respectively. His studies were focused on computer vision and computer graphics. Currently he is a PhD student at the Institute for Computer Graphics and Vision, Graz University of Technology, Austria. His research interests are focused on the efficient 3D reconstruction of urban indoor- and outdoor scenarios, using edge features (such as line segments).

Michael Maurer received his B.S. and M.S. degree in Telematics (computer science and electronics) from Graz University of Technology in 2008 and 2010, respectively. His studies were focused on computer vision and mobile robots and he finished with highest distinction. Currently he is a PhD student at the Institute of Computer Vision and Graphics at Graz University of Technology, Austria. His current research interests are focused on joint semantic segmentation and 3D reconstruction.

Horst Bischof received his M.S. and Ph.D. degree in computer science from the Vienna University of Technology in 1990 and 1993, respectively. In 1998 he got his Habilitation (venia docendi) for applied computer science. Currently he is Professor at the Institute for Computer Graphics and Vision at Graz University of Technology, Austria.

Please cite this article as: M. Hofer et al., Efficient 3D scene abstraction using line segments, Computer Vision and Image Understanding (2016), http://dx.doi.org/10.1016/j.cviu.2016.03.017