Surface Reconstruction from 3D Line Segments

Pierre-Alain Langlois[1]   Alexandre Boulch[2]   Renaud Marlet[1,3]

[1]LIGM (UMR 8049), ENPC, UPE, France   [2]ONERA, Université Paris-Saclay, Palaiseau, France   [3]valeo.ai, Paris, France

Abstract

In man-made environments such as indoor scenes, when point-based 3D reconstruction fails due to the lack of texture, lines can still be detected and used to support surfaces. We present a novel method for watertight piecewise-planar surface reconstruction from 3D line segments with visibility information. First, planes are extracted by a novel RANSAC approach for line segments that allows multiple shape support. Then, each 3D cell of a plane arrangement is labeled full or empty based on line attachment to planes, visibility and regularization. Experiments show the robustness to sparse input data, noise and outliers.

1. Introduction

Numerous applications make use of 3D models of existing objects. In particular, models of existing buildings (e.g., BIMs) allow virtual visits and work planning, as well as simulations and optimizations, e.g., for thermal performance, acoustics or lighting. The building geometry is often reconstructed from 3D point clouds captured with lidars or using cameras and photogrammetry. But with cameras, registration and surface reconstruction often fail on indoor environments because of the lack of texture and strong view points changes: salient points are scarce, point matching is difficult and less reliable, and when calibration nonetheless succeeds, generated points are extremely sparse and reconstructed surfaces suffer from holes and inaccuracies.

Yet, recent results hint it is possible to rely on line segments rather than points. Lines are indeed prevalent in man-made environments, even if textureless. From robust detection [17, 41] and matching [60, 56, 18] to camera registration [13, 42, 43, 36] and 3D segment reconstruction [22, 21], lines can be used when photometric approaches fail for lack of texture. But as opposed to point processing, line-based surface reconstruction has little been studied [57, 34]. This paper presents a novel approach to do so.

A change of paradigm is needed to consider 3D line segments rather than points. Transposing point-based methods to lines is difficult as many point-related assumptions do not hold for line segments. Indeed, points should be numer-
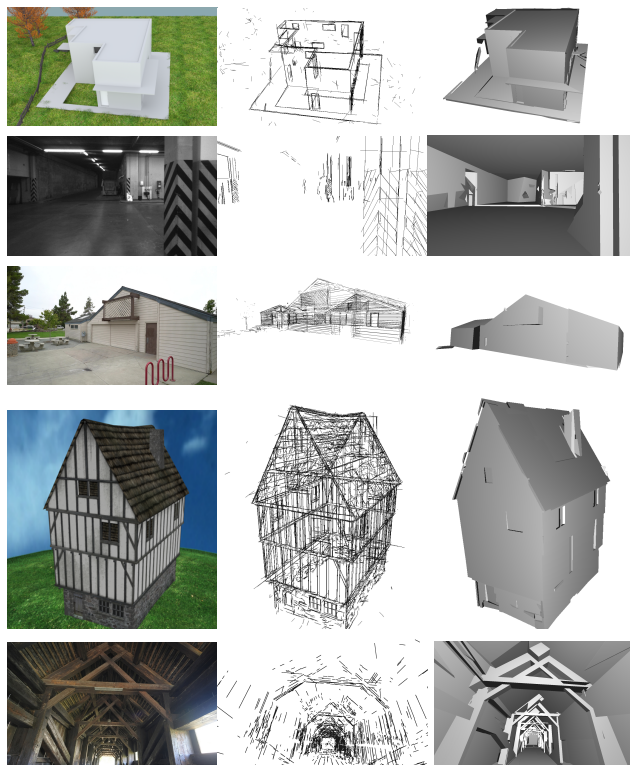

Figure 1: Datasets (from top to bottom) Andalusian, DeliveryArea, Barn, TimberFrame, Bridge: (from left to right) image sample, 3D line segments, our reconstruction.

ous enough (often, in thousands), with a uniform enough sampling, with an accurate enough detection and matching, and most of all, they must belong at most to one primitive. On the contrary, only a few tens of lines (rarely hundreds) are typically detected, and their density and sampling uniformity is so low that they cannot directly support a good surface reconstruction. Also, due to noise in local gradients and varying occlusions depending on viewpoints, segment detection is less accurate and often leads to over-segmentation and unstable end-points, ignored by most 2D line matchers. Only after image registration and 3D segment reconstruction can 2D detections be related to actual fragments of a 3D line segment, moreover possibly differing according to the different viewpoints. Besides, curvy shapes as cylinders may yield unstable occlusion edges (sil-
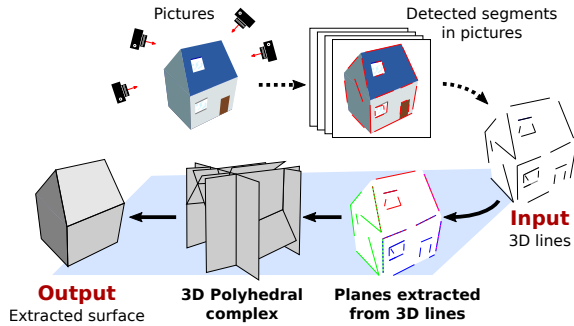
Figure 2: Line-based 3D reconstruction pipeline. This paper covers from **Input** to **Output**.

houettes), yielding noise or outliers. Finally, some 3D lines identify straight edges that are creases between two planar surfaces, and thus support two shapes, contrary to points.

Belonging to two primitives rather than one requires reconsidering shape detection. In particular, in greedy iterative methods, removing all data supporting a detected shape could prevent detecting other shapes because all or a significant fraction of features would then be missing. For instance, it would not be possible to detect all the faces of a cube given only its edges. And even if enough 3D data remains for detection, shape sampling would be affected and some shapes would be less likely or unlikely to be detected.

Overview. We propose the first complete reconstruction pipeline that inputs 3D line segments with visibility information and outputs a watertight piecewise-planar surface without self-intersection (cf. Fig. 2). We first extract primitive planes from the line cloud, distinguishing two kinds of line segments: *textural lines*, supporting a single plane, and *structural lines*, at the edge between two planes. Then we label each 3D cell of the plane arrangement as full or empty by minimizing an energy based on line type, line segment support, visibility constrains and regularization.

Our main contributions are as follows:
- We define a robust and scalable plane detection method from 3D line segments, without scene assumptions. This novel non-straightforward RANSAC formulation takes into account a key specificity of lines vs points, namely that they can support up to two primitives (at edges), which breaks the greedy iterative detection traditionally used with points.
- We totally recast the surface reconstruction approach of [10, 6] into a line-based setting. We meaningfully and efficiently generalize data fidelity and visibility from points to line segments, taking care of lines supporting two planes. We also feature a simpler and lighter treatment of noise.
- We validate our method on existing datasets, and provide new ones to assess line-based reconstruction quality. Examples of our reconstructions are illustrated on Fig. 1.

2. Related work

Surface reconstruction has been extensively studied from 3D points [5] and/or images [16]. We consider here the input to be 3D line segments (with viewpoints), that can be sparse, missing, noisy and corrupted with outliers. We aim at an idealized piecewise-planar and watertight surface.

To deal with sparse data, some methods detect planes based on 3D features and dominant orientations [49], possibly with a Manhattan-world assumption [15], and create piecewise-planar depth maps taking into account visibility and local consistency. Other approaches consider 2D image patches back-projected on 3D planes [35, 7]. In contrast, our method produces a watertight mesh, does not impose a few specific orientations, and can work with 3D features only, not requiring images and not working at pixel level.

Another approach to little input data is to extend boundaries and primitives until they intersect [9]. It however does not ensure a watertight reconstruction either. This is only achieved by methods that create a volumetric partition of the 3D space and extract the surface from full and empty cells. The partition can be a voxel grid [45], a 3D Delaunay triangulation [30, 55] or a plane arrangement [10, 6].

Wireframe reconstruction is what most lined-based methods focus on: rather than surfaces, they study how to generate meaningful 3D line segments [25, 22, 21, 23], after line matching is performed [44]. And more general curves than lines are not used beyond structure from motion [39].

Surface reconstruction with lines in addition to points has received a modest attention. [2] reconstruct planes from a 3D line and a neighboring detected point. It requires lines surrounded with texture and is outlier-sensitive. It also does not prevent self-intersections nor guarantees watertightness. [4] segments images into likely planar polygons based on 3D corner junctions and use best supporting lines to reconstruct polygons in 3D. For 2.5D reconstruction, extracted 3D lines [44] are used with a dense height map to build a line arrangement on the ground plane and create geometric primitives and building masks [59]. In [49], pairs of 3D lines generated from vanishing directions provide plane hypotheses, validated by 3D points. The surface is a set of planar patches created from plane assignment to pixels. [50] adds points uniformly sampled on the 3D lines to the Delaunay triangulation, introducing extra parameters, and although visibility is treated without sampling, the method is unlikely to work on scenes with only sparse lines. [20] also shows a meshing improvement using 3D line segments.

Surface reconstruction from line segments only, when points fail due to the lack of texture, has little been studied. [58] presents a single-view surface reconstruction based on 2D line segments. Lines are paired from segment extensions along their direction, and planes orientations are sought by RANSAC, hypothesizing mutually orthogonal corresponding 3D lines. Articulating lines are found at plane inter-

sections to construct a multi-plane structure. Our *structural lines* are called 'articulation' or 'articulating' lines in [58]. They are discovered late, to set plane offsets, whereas we differentiate them early at plane detection. For robotic mapping, [57] considers all combinations of two non-collinear coplanar line segments as plane hypotheses. Line segments are then assigned to possibly multiple planes in a face complex built from plane intersections. The reconstructed surface is made of faces depending on an occlusion score. Compared to our approach, this method does not scale well to many lines, is sensitive to outliers, relies on a number of conservative heuristics that can be detrimental to surface recall, involves no regularization, and does not reconstruct a watertight mesh. As for [34], it first reprojects 3D lines into images that see them, studies the intersection of segments in 2D rather than planes in 3D, and infers plane hypotheses. The surface is made from image faces back-projected onto a possible 3D plane. Although less sensitive to outliers, this method involves heuristics and no proper regularization, and it reconstructs a non-watertight mesh with floating polygons and possible self-intersections.

Extracting 3D planes from line segments has little been treated; the literature focuses on point clouds, chiefly ignoring line clouds. The most popular scheme for points, which is robust to sparsity contrary to region growing as in [10, 6], is RANSAC [11, 46]. But as explained below, it cannot straightforwardly be applied to line segments because it relies on different distribution hypotheses and because of the possible association of a segment to several primitives, also invalidating line discretization into points. Still, [57] takes line segments as input, but plane detection is somewhat exhaustive, hence with scalability issues, and sensitive to outliers. Using laser data, [8] exploits 3D lines to detect planes, but it uses strong properties of lidar acquisition, namely line parallelism and large and dense data.

An open question is if multi-model methods [61, 51, 24, 33], which assume non-overlapping segmented data, can be adapted not only to large inputs but also to multiple shape support [29, 1], as absolutely required for line segments.

Surface reconstruction from a plane arrangement is a common topic, with variants enforcing plane regularity [32, 37] or level of detail [54], or offering reconstruction simplicity [38]. It is largely orthogonal to our work. Here we build on [6], with line-specific data and visibility terms.

3. Plane detection from 3D line segments

The first step of our approach is to detect planes that are supported by line segments in the input line cloud $\mathcal{L}$. We use the RANSAC framework [14] as it scales well to large scenes and deals well with a high proportion of outliers, which are unavoidable in photogrammetric data.

As argued above and shown experimentally (cf. Sect. 5), a key requirement is to allow a line to belong to two planes.

Lines supporting one plane are considered *textural*; lines supporting two planes are deemed *structural*. Yet some actual texture lines may support additional "virtual" planes, as when a line is drawn around an object, e.g., at the borders of a frieze around the walls of a room, which belongs both to the vertical walls and to an non-physical horizontal plane.

Candidate plane construction. We generate candidates by sampling the minimum number of observations required to create a model, i.e., two non-collinear line segments to define a plane. Two 3D segments l_a, l_b can be coplanar in two ways: they can be parallel, or their supporting infinite lines can intersect. With noisy real data, the latter can be relaxed using a maximum small distance ϵ between the lines. We discard parallelism because, when reconstructing man-made environments such as buildings, it may generate many bad planes. Indeed, two random vertical segments (e.g., detected on windows) are parallel but statistically unlikely to support an actual, physical plane (e.g., segments on different facades). We thus threshold the angle $\angle(l_a, l_b)$, which also excludes the degenerate case of collinear segments.

Greedy detection and multi-support issues. We sample planes as line pairs and perform an iterative extraction of the most significant planes, i.e., with the largest number of supporting segments after a given number of sampling trials. However, contrary to usual RANSAC, we cannot remove supporting segments at once as they may actually belong to two planes; it would lead to detecting the main planes only, missing planes with a smaller support. The supplementary material (supp. mat.) illustrates failure cases. Conversely, we cannot consider all segments as available at each iteration: it would statistically lead to multiple detections of the same large planes and again miss planes with small support.

A natural way to allow a datum to be part of several detection in greedy RANSAC is to remove inliers for model sampling but not for data assignment to models [58]. But for sparse data (which is the case with line segments), it fails to detect models with little data support, e.g., preventing detecting all the faces of a cube from its sole edges.

Another way to allow the same datum to seed several models is to bound their number, i.e., 2 for lines supporting planes. But it does not work either as it often associates a line twice to more or less the same plane. As illustrated in the supplementary material too, this yields very bad results.

Our solution is to bound the number of supported planes per line segment, but with an additional condition to prevent shared segments to belong to similar planes.

Candidate plane generation. We note $\Lambda(P)$ the set of line segments supporting a plane P, $\Pi(l)$ the set of planes supported by a line segment $l \in \mathcal{L}$, with $|\Pi(l)| \leq 2$, and $\mathcal{L}_i$ the set of segments supporting i plane(s) for i in $0, 1, 2$.

We construct these sets iteratively by generating candidates planes P and assigning them segments $l \in \mathcal{L}$, some

of which may have already been assigned to another plane $\Pi(l)$. Only line segments in $\mathcal{L}_2$ are discarded from the pool of available segments to support a plane, as they already support two planes. Initially, $\mathcal{L}_0 = \mathcal{L}$, and $\mathcal{L}_1 = \mathcal{L}_2 = \emptyset$.

As line segments are not put aside as soon as they are assigned to a plane, they can be drawn again to generate new candidate models. However, generating several times the same plane (with the same supporting line segments) would not only reduce efficiency, but also make some models little likely to be drawn, as models with a large support would be sampled much more often. To prevent it, after drawing a first line segment $l_a \in \mathcal{L}_0 \cup \mathcal{L}_1$, there are two cases. If $l_a \in \mathcal{L}_0$, i.e., if l_a has not been assigned to any plane yet, then the second segment l_b can be drawn unconditionally in $\mathcal{L}_0 \cup \mathcal{L}_1$ as it will always yield a new model. If $l_a \in \mathcal{L}_1$, i.e., if l_a has already been assigned to some plane P', with $\Pi(l_a) = \{P'\}$, then lines in $\Lambda(P')$, i.e., supporting P', are excluded when drawing the second segment l_b. This ensures l_a, l_b cannot participate to the same already existing model. As the number of extracted planes is typically less than a few hundred, this drawing can be optimized by incrementally keeping track of the sets $\bar{\Lambda}(P) = \mathcal{L} \setminus (\mathcal{L}_2 \cup \Lambda(P))$, that have *not* already been assigned to a detected plane P.

We do not prevent a line pair to be redrawn when it previously failed to generate an accepted model (for lack of planarity, parallelism or poor support) because it does not lead to unbalanced chances to detect a plane. And if $|\mathcal{L}|$ is not too large, we can draw systematically all line pairs.

Note that we do not prevent a line pair to be redrawn when it previously failed to generate an accepted model (for lack of planarity, parallelism or poor support). It is not an issue as it does not lead to unbalanced chances to detect a plane. Yet, when the number of input line segments is not too large, we can perform a systematic drawing of all line pairs, possibly exploiting the above filtering. In this case, all possible models are considered and at most once.

Inlier selection. After picking a candidate plane P, we populate the support $\Lambda(P)$. For this, we go through each segment $l \in \mathcal{L}_0 \cup \mathcal{L}_1$ and assign it to $\Lambda(P)$ if close enough to P, i.e., if $d(l, P) \leq \epsilon$. Several distances can be used, such as the average or the maximum distance to the plane.

If l already supports some other plane P', i.e., if $\Pi(l) = \{P'\}$, then also assigning l to P would make it a structural segment. As such, we impose that it lies close to the line at the intersection of both planes, i.e., $d(l, P \cap P') \leq \epsilon$. This condition is stronger than imposing both $d(l, P) \leq \epsilon$ and $d(l, P') \leq \epsilon$ as the angle between P and P' could be small and l could then be close to both P and P' although far from their intersection. This condition is actually crucial. Without it, we would tend to associate l to two planes P and P' which are very similar, and fail to detect crease lines.

Plane selection. Last, we sample N_{iter} models and keep the plane with the largest number of inliers. (See the

supp. mat. for the abstract version of the algorithm.)

This plane detection differs from [58], that samples and populates planes from 2D line pairs instead of 3D lines, making inlier search quadratic, not linear, and requiring heuristically to only consider pairs defined by intersecting segment extensions, which is highly unstable due to noise in endpoints and which induces plane splitting at occlusions. We have none of these downsides. Besides, structural lines in [58] are found with heuristics after RANSAC, considering plane pairs and candidate lines, which only makes sense as they have few (<10) planes. We get them directly, without heuristics, in greater number, and for many more planes.

Plane refitting. After each plane P_{best} is selected, it is actually refitted to its inliers Λ_{best} before being stored into Π, based on the (signed) distance of the segment endpoints, weighted by the segment length. As it changes the plane equation, we check whether the slice centered on the refitted plane P' with thickness ϵ now contains extra segments. If so, they are added as inliers and refitting is repeated.

Plane fusion. Modeling a building may require different levels of details, including small plane differences such as wall offsets for door jambs, baseboards or switches. But setting a small ϵ to do so may easily break a wall or a ceiling into several fragments because it is not perfectly planar due to construction inaccuracies or load deflections. Each country actually has standards (official or not) defining construction tolerances, e.g., 1 cm error every 2 m for walls.

To prevent this arbitrary fragmentation while preserving details, we add a plane fusion step with a tolerance higher than ϵ, i.e., with a maximal distance threshold $\epsilon_{\text{fus}} > \epsilon$ to the plane refitted on the union of inliers. This allows merging at ϵ_{fus} accuracy several plane fragments detected at ϵ. However, to make sure it applies only to cases described above, we impose a maximum angle θ_{fus} when merging two planes and minimum proportion p_{fus} of common inliers. Concretely, we consider all pairs of planes in Π whose angle is less than θ_{fus}, sort them, pick the pair with the smallest angle, and try merging it. If it succeeds, the two planes are removed, the new refitted plane is added, and the priority queue based on angles is updated before iterating. If it fails, the pair of planes is discarded and the next pair is considered. This is similar to a heuristics used in Polyfit [38].

Plane limitation. To make sure not too many planes are given to the surface reconstruction step, because of possible limitations (cf. Sect. 6), the algorithm may be stopped after at most N_{max} (best) greedy detections.

4. Surface reconstruction

The second step of our approach is surface reconstruction based on detected planes and observations of 3D line segments. Rather than selecting plane-based faces with hard constraints for the the surface to be manifold and watertight

[38], we follow [10, 6] and consider a scene bounding box, partition it into 3D cells constructed from the planes, and assign each cell with a status 'full' or 'empty' depending on segment visibility, with a regularization prior coping with sparse and missing data. The reconstructed surface is then the interface between full and empty cells. By construction, it is watertight and free from self-intersections. Our contribution is a total reformulation of [10, 6] in terms of lines, making the difference between textural and structural lines, and with a lighter treatment of noise in data.

The volume partition is given by a cell complex $\mathcal{C}$ made from an arrangement of planes detected in the line cloud. For each cell $c \in \mathcal{C}$, we represent occupancy by a discrete variable $x_c \in \{0, 1\}$: 0 for empty and 1 for full. A surface is uniquely defined by a cell assignment $\mathbf{x} : \mathcal{C} \mapsto \{0, 1\}$, where $\mathbf{x}(c) = x_c$. The optimal cell assignment $\mathbf{x}$ is defined as the minimum of an energy $E(\mathbf{x})$ which is the sum of three terms: a primitive term $E_{\text{prim}}(\mathbf{x})$ penalizing line segments not lying on the reconstructed surface, a visibility term $E_{\text{vis}}(\mathbf{x})$ penalizing surface reconstructions on the path between observations and their viewpoints, and a regularization term $E_{\text{regul}}(\mathbf{x})$ penalizing complex surfaces.
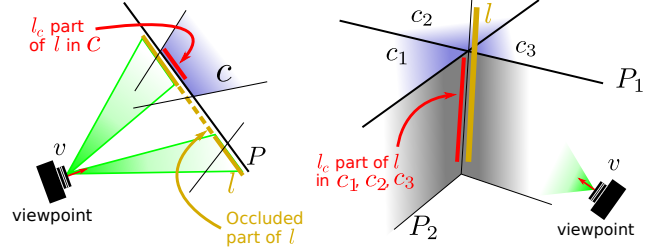
$$E(\mathbf{x}) = E_{\text{prim}}(\mathbf{x}) + E_{\text{vis}}(\mathbf{x}) + E_{\text{regul}}(\mathbf{x}) \quad (1)$$

Dealing with noise. To deal with noise in input data, [6] introduces slack in the choice of cells penalized for not being at the reconstructed surface and lets regularization make the right choices. The resulting formulation and resolution is heavy. Instead, we assume that plane extraction (Sect. 3) did a good-enough job: any segment supporting a plane (resp. two planes) is considered as a noisy inlier and is projected on the plane (resp. the intersection of the two planes). A segment not supporting any plane is treated as an outlier for data fidelity (no penalty for not being on the reconstructed surface) but not for visibility (penalty for not being seen from viewpoints if hidden by reconstructed surface).

Primitive term. $E_{\text{prim}}(\mathbf{x})$ penalizes line segments that support planes but do not lie on the reconstructed surface. But it does not penalize the presence of matter in front of segments w.r.t. viewpoints, letting the visibility term do it. Segments that support no plane are ignored as if outliers.

For a segment l supporting one plane P, and for each viewpoint v seeing at least a part of l, we consider the set C of all cells c immediately behind l w.r.t. v, possibly only along a fraction l_c of l due to occlusions (cf. Fig. 3(a)). Each $c \in C$ is penalized if not full, with a cost $1 - x_c$.

For a segment l supporting two planes P_1, P_2, a cell behind l w.r.t. viewpoint v need not be full. (Penalizing emptiness actually yields terrible results, as the supp. mat. shows.) Any configuration is valid as long as the space around l is not empty (cf. Fig. 3(b)): salient edges, reentrant edges or planes (if the seemingly structural line happens to only be textural). To penalize only when all three cells c

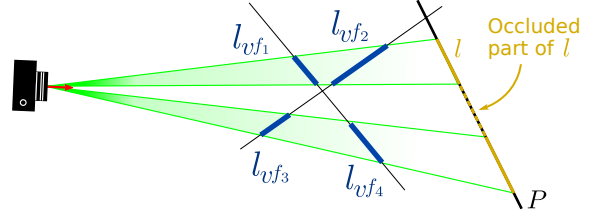(a) Primitive term, $l \in P$ (b) Primitive term, $l \in P_1 \cap P_2$

Figure 3: (c) Visibility term

around a visible fraction of l are empty (ignoring the cell in front), we consider a cost of $\max(0, 1 - \sum_c x_c)$, which is equal to 1 in this case, and 0 in other configurations.

Both textural and structural cases can be covered with a single formula, where we weigh the cost by the length of the visible fraction of l and normalize it by a scale of interest σ:

$$E_{\text{prim}}(\mathbf{x}) = \sum_{l \in \mathcal{L}_1 \cup \mathcal{L}_2} \sum_{v \in \mathcal{V}(l)} \sum_{C \in \mathcal{C}(l,v)} \frac{|l_C|}{\sigma} \max\left(0, 1 - \sum_{c \in C} x_c\right)$$
$$(2)$$

where $\mathcal{L}_1 \cup \mathcal{L}_2$ is the set of segments l supporting at least one plane, $\mathcal{V}(l)$ is the set of viewpoints v seeing l, $\mathcal{C}(l, v)$ is the set of cells c adjacent to l but not in the triangles of sight from v to non occluded fragments of l (locally 1 or 3 cells as to whether l belongs to 1 plane or 2 planes), l_C is the set of fragments of l in each cell $c \in C$, and $|l_C|$ is the sum of the lengths of segment fragments in l_C.

Visibility term. $E_{\text{vis}}(\mathbf{x})$ penalizes reconstructed surface boundaries between viewpoints and segments, as [10, 6]. It measures the number of times a 3D segment l is to be considered an outlier as it should not be visible from a given viewpoint v, weighted by the length of the visible parts $l_{v,f}$ of l on the offending faces f (possibly fragmented due to occlusions). Contrary to $E_{\text{prim}}(\mathbf{x})$, all segments are considered in $E_{\text{vis}}(\mathbf{x})$, not just segments supporting a plane:

$$E_{\text{vis}}(\mathbf{x}) = \lambda_{\text{vis}} \sum_{l \in \mathcal{L}} \sum_{v \in \mathcal{V}(l)} \sum_{f \in \mathcal{F}(l,v)} \frac{|l_{v,f}|}{\sigma} |x_{c_f^{+v}} - x_{c_f^{-v}}|$$
$$(3)$$

where $\mathcal{F}(l, v)$ is the set of faces f of the complex intersected by the visibility triangle (l, v), at some unoccluded segment fractions $l_{v,f}$ totaling a length of $|l_{v,f}|$, and c_f^{+v}, c_f^{-v} are the cells on each side of f (c_f^{+v} being nearest to v).

Regularization term. $E_{\text{regul}}(\mathbf{x})$ penalizes surface complexity as the sum of the length of reconstructed edges and
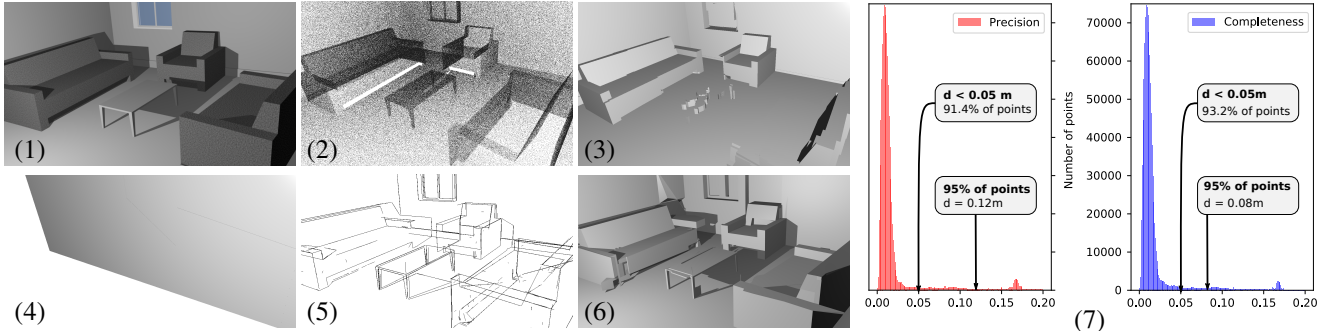
Figure 4: HouseInterior: (1) an image of the dataset, (2) points densely sampled on surface, (3) reconstruction with [10], (4) failed reconstruction with [10] from points sampled on lines, (5) 3D lines detected with Line3D++ [19], with noise and outliers, (6) our reconstruction, which is nonetheless superior, (7) histograms of distance errors w.r.t. ground truth (m).

the number of corners, with relative weights $\lambda_{\text{edge}}, \lambda_{\text{corner}}$, as defined in [6]. Area penalization makes little sense here due to the low density of observations in some regions.

Solving. Minimizing this higher-order energy is a non linear integral optimization problem (max in eq. (2)). As in [6], integral variables are relaxed to real values and slack variables are introduced. The resulting linear problem is solved and fractional results are rounded to produce the final integral values. See details in the supplementary material.

Properties of reconstructed surface. By construction, the surface we produce is watertight, even if the input data is very sparse, and not self-intersecting. Our process treats outliers (with RANSAC at plane detection stage, and regularization during reconstruction) and noise (with a model tolerance at plane detection stage and via projections when reconstructing). It has also several positive properties:

• *Insensitivity to line over-segmentation:* if a 3D line segment l is split, $E(\mathbf{x})$ does not change and thus the same surface is reconstructed. This provides robustness to over-segmentation, which is a common weakness of line segment detectors. (It may however change inlier-ness.)

• *Little sensitivity at endpoints*: given a line segment l, slightly changing its endpoint only makes a marginal change to $E(\mathbf{x})$. (Yet it may change inlier-ness too.)

• *Insensitivity to dummy planes:* given a 3D cell assignment $\mathbf{x}$, if an extra plane is randomly inserted in the arrangement, the value of $E_{\text{vis}}(\mathbf{x})$ does not change as it only depends on surface transitions encountered on visibility path.

5. Experiments

We experimented both with real and synthetic data, for qualitative and quantitative analysis. The real datasets consist of images of a 'MeetingRoom' from [43], of a 'Barn' from Tanks and Temples [28], of a 'DeliveryArea', a 'Bridge' and of a corridor named 'Terrains' from ETH3D [48]. All scenes are poorly textured (walls of uniform colors). The synthetic datasets include a 'Timber-Frame' house [25] as well as two new synthetic datasets, to

σ_p†	ϵ	ϵ_{fus}	θ_{fus}	p_{fus}	N_{iter}	N_{max}	λ_{vis}	λ_{edge}	λ_{corner}
2.5	2 cm	3 ϵ	10°	20%	50k	160	0.1	0.01	0.01

Table 1: Parameters (all datasets are metric). † in Line3D++

Dataset	#img	$\|\mathcal{L}\|$	$\|\Pi\|$	$\|\Pi_{\text{fus}}\|$	$\|\mathcal{L}_0\|$	$\|\mathcal{L}_1\|$	$\|\mathcal{L}_2\|$	#res
TimberFrame	241	7268	140	131	264	4507	2497	79024
Andalusian	249	1234	160	148	242	597	395	14503
MeetingRoom	32	831	135	130	25	383	423	9028
Terrains	42	3223	120	105	9	356	2858	18189
DeliveryArea	948	1586	160	160	30	771	785	29222
Barn	410	7936	160	141	41	2157	5738	83989
Bridge	110	7437	150	102	338	4168	2931	48315
HouseInterior	159	1995	120	106	1	286	1708	18304

Table 2: Dataset statistics: number of images #img, number of 3D line segments $\|\mathcal{L}\|$, number of 3D planes before fusion $\|\Pi\|$, number of 3D planes after fusion $\|\Pi_{\text{fus}}\|$, number of segments supporting no plane $\|\mathcal{L}_0\|$, one plane $\|\mathcal{L}_1\|$ or two planes $\|\mathcal{L}_2\|$, and total number of sub-segments #res.

be publicly released. 'HouseInterior' is a living room, with both large planar areas (walls, floor and ceiling) and smaller details (chair and table legs). 'Andalusian' is the outside of a modern house; it is piecewise-planar and uniformly white.

MeetingRoom was calibrated with LineSfM [40, 43] and we recalibrated the other real datasets using Colmap [47], with distortion correction as it impacts line detection. The synthetic datasets came with their exact calibration.

We then ran Line3D++ [19], as defined in [21], to detect and reconstruct 3D line segments. As seen on Figs. 1, 4, 5 and in the supplementary material, line segments obtained from Line3D++ are extremely noisy: lines that are mostly parallel, orthogonal, planar or colinear in the original scene turn out to be reconstructed with visible discrepancies. There are also many missing lines and many outliers. For instance, in MeetingRoom, many segments are floating in the air in the middle of the room. Line3D++ also tends to duplicate the same segment many times with a little displacement, leading to a treatment as noise or outlier.

Finally, we ran our plane detection and surface reconstruction, using a complete plane arrangement as baseline
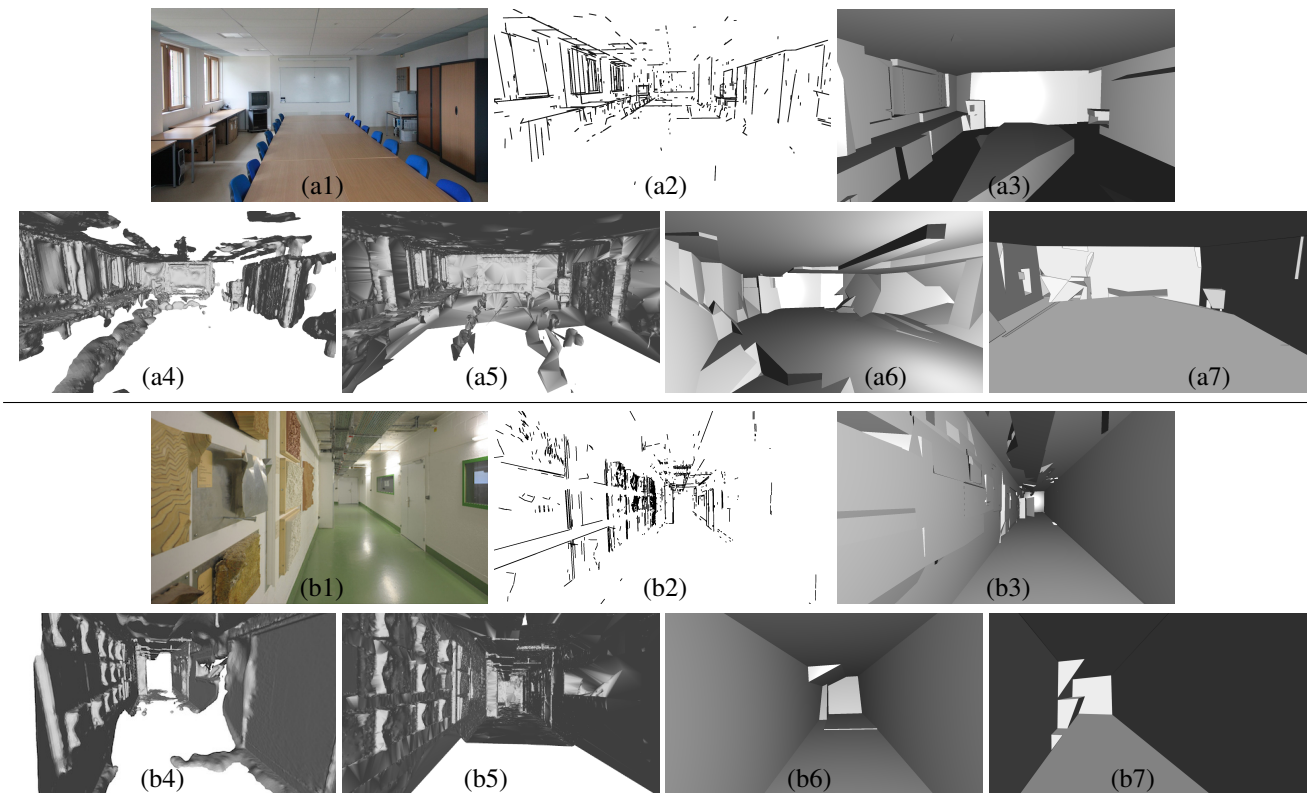
Figure 5: MeetingRoom (a), Terrains (b): (1) image sample, (2) segments from Line3D++ [19, 21], (3) our reconstruction, point-based reconstructions with Colmap [47] then (4) Poisson [27], (5) Delaunay [31], (6) Chauve et al.[10], (7) Polyfit [38].

(see Sect. 6). Tab. 1 lists default parameters for all datasets. We often had to tweak σ_p of Line3D++ to get decent input lines, and sometimes our $\lambda_{\mathsf{edge}} = \lambda_{\mathsf{corner}}$ (see the supp. mat. for a sensitivity study). Tab. 2 reports detection statistics.

Comparing to point-based reconstruction. To show the relevance of lines for scenes with little or no texture, in contrast to point-based methods (which are doubtlessly superior on textured scenes), we compare our method to a point-based piecewise-planar reconstruction [10] on House-Interior (cf. Fig. 4). Even when densely sampling point on the ground-truth surface as seen from the viewpoints, [10] yields a reconstruction with missing details (e.g., the lounge table) due to missing primitives in hidden area (e.g., under the table). Moreover, [10] uses a regularization that minimizes the reconstructed area, which is relevant for points uniformly sampled on the surface but strongly penalizes unsampled regions (e.g., invisible planes of lounge table). In contrast, our method leads to a better plane discovery and a reconstruction robust to non-uniform sampling. (We also tried reconstructing from points sampled on the 3D lines, but the result is terrible; many planes are missed as points belong at most to one plane. As lines mostly lie on edges, the area cost also dominates the data term and creates holes in large planar regions.) More comparisons, also with Colmap [47] and Polyfit [38], are on Fig. 5. The supp. mat.

also studies the sensitivity to the number of images.

Comparing to other line-based reconstruction methods. As said above, there are very few reconstruction methods based on lines. [34] mostly reconstructs a soup of planes, sometimes with adjacencies, but without any topological guarantee. [57] provides a slightly more behaved mesh, but reconstructions still look messy and overly simple, although usable enough for robotic planning. No code nor data are available for comparing with either of these methods.

Quantitative evaluation. We evaluate the quality of reconstruction with two criteria: precision (proximity to the ground truth) and completeness (how much of the ground truth is reconstructed). For this, we pick 2M points both on the reconstruction and on the ground truth, and we compute the nearest-neighbor distance from one set to the other.

Histograms of distances for HouseInterior are plotted on Fig. 4(7). Regarding precision, most of the points sampled on the reconstruction (91.4%) lie at less than 5 cm to the ground truth, showing that our RANSAC planes fit well the underlying surface and that our energy properly balances data fidelity and regularization. The error profile for completeness is similar, and 95% of the points on the ground truth are less than 8 cm to the reconstruction. It shows our regularization term do not over-smooth too much the surface by erasing details that would penalize completeness.
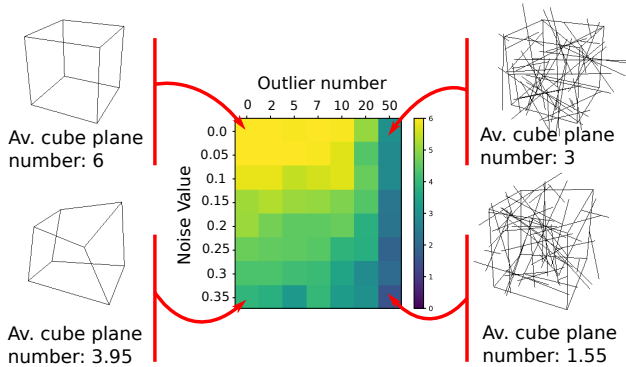
Figure 6: Robustness of RANSAC on lines for a cube defined by its edges. Value in grid is the average number of cube planes found, depending on the perturbation.

Qualitative evaluation. Figs. 1, 4, 5 illustrates our reconstructions with sparse data (Andalusian, DeliveryArea, MeetingRoom), and more (TimberFrame, Barn) or less texture (HouseInterior), possibly with thin objects like beams (Bridge). Compared to usual point clouds, our 3D line clouds are extremely sparse. Despite the noise on inliers and the number of outliers due to Line3D++, our method is able to reconstruct a good approximation of the scenes, which illustrates the robustness of our approach. Still Barn shows that it is hard to reconstruct a sieve-like shape (balcony) due to the visibility lines traversing it.

Computation times. Although computing the visibility term is linear in the number of sub-segments, it is the most time-consuming part as it depends mostly on the number of cells in the plane arrangement, which is up to cubic in the number of planes. Time required for performing a whole reconstruction varies from 30 minutes (MeetingRoom) to 3 hours 30 minutes (TimberFrame). Creating the linear program from scene data takes more time than solving it.

Robustness of plane detection. To explore the robustness of our RANSAC formulation, we experimented with a toy example made of the 12 edges of a cube. We seek to extract the 6 planes associated to the 6 faces of the cube. We consider two types of perturbations: noise and outliers.

The cube has an edge length of 2. We add noise to each segment endpoint, drawn from a uniform distribution with standard deviation ranging from 0 to 0.35. Outliers, from 0 to 50, are segments generated by uniformly picking pairs of points in a 2-radius ball. Finally for each couple (noise, #outliers), we report the number of planes that include the 4 edges of an actual face of the cube, using parameters $\epsilon = 0.06$ and $N_{\mathrm{iter}} = 100$, and averaging over 20 iterations.

Results are presented on Fig. 6. As expected, with a low level of perturbation, all planes are perfectly extracted. As the level of perturbation increases, for both noise and outliers, the rate of missed detections increases. Yet, even with a high level of noise, corresponding to a highly distorted cube (very non planar faces), we get a mean of 3.95 planes.

Ablation study. We compared with variants of RANSAC where (a) one line supports at most one plane, which leaves fewer lines for ulterior extractions and detects less planes, (b) we only consider $d(l, P) \leq \epsilon$ to decide if segment l is an inlier to candidate plane P, ignoring if $l \in \mathcal{L}_1$, which misses many planes. We also tried ignoring the notion of structural lines at reconstruction time, treating segments with two supports as two ordinary lines (one for each plane), which fails miserably. Last, we compared with a regularization using only corners or edges, which yields lower quality reconstructions. See supp. mat. for details and illustrations.

6. Conclusion

We studied the specifics of line-based reconstruction and proposed the first method to create an intersection-free, watertight surface from observed line segments. Experiments on synthetic and real data show it is robust to sparsity, outliers and noise, and that it outperforms point-based methods on datasets with little or no texture.

Limitations and perspectives. The quality of 2D and 3D line segments at input (from Line3D++) is the main bottleneck of our method. Improving them would be very helpful.

Mainly, it would be specially relevant too to merge points and lines treatments into a single framework to offer a smooth transition from textured regions to textureless areas.

Also, in our experiments, we used the full-extent plane arrangement, i.e., with planes extending all the way to the scene bounding box. This is not intrinsic to our method; it merely provides a baseline. Because of a cubic complexity in the number of planes, the acceptable number of planes is limited to a few hundreds, which is in practice often enough for a single room or the exterior of a building, but not enough for a complete BIM model. (It is also easy to keep the best few hundred planes after RANSAC detection to make sure the pipeline succeeds.) Yet, preliminary experiments with a coarse-to-fine approach show promising results for scaling to large scenes. In the cell complex, limiting the plane extent with a heuristic on a coarse voxel-based partition [10] or adapting 2D kinetic polygonal plane partitioning [3] to 3D would also reduce the complexity.

Moreover, defining a notion of extent for line-detected planes, similar to α-shapes in the case of points [12] but adapted to lines [52, 53], could also be used to introduce so-called 'ghost planes', corresponding to unobserved, hidden planes at occluding edges of observed surfaces [10, 6].

Last, global regularization weights favor highly sampled surfaces. Adapting them to be more sensitive to weakly supported surfaces as in [26] could improve the results.

References

[1] S. Baadel, F. Thabtah, and J. Lu. Overlapping clustering: A review. In *SAI Computing Conference (SAI 2016)*, pages 233–237, July 2016. 3

[2] C. Baillard and A. Zisserman. Automatic reconstruction of piecewise planar models from multiple views. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 1999)*, pages 2559–2565, Ft. Collins, CO, USA, 1999. 2

[3] J. Bauchet and F. Lafarge. KIPPI: kinetic polygonal partitioning of images. In *IEEE Conference on Computer Vision and Pattern Recognitio (CVPR 2018)*, pages 3146–3154, Salt Lake City, UT, USA, June 2018. 8

[4] H. Bay, A. Ess, A. Neubeck, and L. Van Gool. 3D from line segments in two poorly-textured, uncalibrated images. In *3rd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT 2006)*, pages 496–503, June 2006. 2

[5] M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, J. Levine, A. Sharf, and C. Silva. State of the Art in Surface Reconstruction from Point Clouds. In *Eurographics 2014 - State of the Art Reports*, volume 1 of *EUROGRAPHICS star report*, pages 161–185, Strasbourg, France, Apr. 2014. 2

[6] A. Boulch, M. de La Gorce, and R. Marlet. Piecewise-planar 3D reconstruction with edge and corner regularization. *Computer Graphics Forum (CGF 2014)*, 2014. 2, 3, 5, 6, 8

[7] A. Bourki, M. de La Gorce, R. Marlet, and N. Komodakis. Patchwork stereo: Scalable, structure-aware 3D reconstruction in man-made environments. In *IEEE Winter Conference on Applications of Computer Vision (WACV 2017)*, Mar. 2017. 2

[8] C. Cabo, S. G. Cortes, and C. Ordonez. Mobile laser scanner data for automatic surface detection based on line arrangement. *Automation in Construction*, 58:28 – 37, 2015. 3

[9] U. Castellani, S. Livatino, and R. B. Fisher. Improving environment modelling by edge occlusion surface completion. In *1st International Symposium on 3D Data Processing Visualization and Transmission (3DPVT 2002)*, pages 672–675. IEEE, 2002. 2

[10] A. L. Chauve, P. Labatut, and J. P. Pons. Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2010)*, pages 1261–1268, June 2010. 2, 3, 5, 6, 7, 8

[11] S. Choi, T. Kim, and W. Yu. Performance evaluation of ransac family. In *British Machine Vision Conference (BMVC 2009)*, 2009. 3

[12] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–559, July 1983. 8

[13] A. Elqursh and A. Elgammal. Line-based relative pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2011)*, pages 3049–3056, June 2011. 1

[14] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. 3

[15] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Manhattan-world stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, pages 1422–1429, June 2009. 2

[16] Y. Furukawa and C. Hernández. Multi-view stereo: A tutorial. *Foundations and Trends in Computer Graphics and Vision (CGV 2015)*, 9(1-2):1–148, 2015. 2

[17] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. LSD: a line segment detector. *Image Processing On Line (IPOL 2012)*, 2:35–55, 2012. 1

[18] K. Hirose and H. Saito. Fast line description for line-based SLAM. In *British Machine Vision Conference (BMVC 2012)*, 2012. 1

[19] M. Hofer. Line3D++, 2016. https://github.com/manhofer/Line3Dpp. 6, 7

[20] M. Hofer, M. Maurer, and H. Bischof. Improving sparse 3D models for man-made environments using line-based 3D reconstruction. In *2nd International Conference on 3D Vision (3DV 2014)*, volume 1, pages 535–542, Dec. 2014. 2

[21] M. Hofer, M. Maurer, and H. Bischof. Efficient 3D scene abstraction using line segments. *Computer Vision and Image Understanding (CVIU 2016)*, 3 2016. 1, 2, 6, 7

[22] M. Hofer, A. Wendel, and H. Bischof. Incremental line-based 3D reconstruction using geometric constraints. In *British Machine Vision Conference (BMVC 2013)*, 2013. 1, 2

[23] N. Ienaga and H. Saito. Reconstruction of 3D models consisting of line segments. In C.-S. Chen, J. Lu, and K.-K. Ma, editors, *Asian Conference on Computer Vision workshops (ACCVw 2017)*, pages 100–113. Springer International Publishing, 2017. 2

[24] H. Isack and Y. Boykov. Energy-based geometric multi-model fitting. *International Journal of Computer Vision (IJCV 2012)*, 97(2):123–147, Apr 2012. 3

[25] A. Jain, C. Kurz, T. Thormählen, and H. P. Seidel. Exploiting global connectivity constraints for reconstruction of 3D line segments from images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2010)*, pages 1586–1593, June 2010. 2, 6

[26] M. Jancosek and T. Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2011)*, pages 3121–3128, 2011. 8

[27] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *4th Eurographics Symposium on Geometry Processing (SGP 2006)*, pages 61–70, 2006. 7

[28] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (TOG 2017)*, 36(4), 2017. 6

[29] H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM ransactions on Knowledge Discovery from Data (TKDD 2009)*, 3(1):1:1–1:58, Mar. 2009. 3

[30] P. Labatut, J.-P. Pons, and R. Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, De-

launay triangulation and graph cuts. In *IEEE 11th International Conference on Computer Vision (ICCV 2007)*, 2007. 2

[31] P. Labatut, J.-P. Pons, and R. Keriven. Robust and efficient surface reconstruction from range data. *Computer Graphics Forum (CGF 2009)*, 28(8):2275–2290, 2009. 7

[32] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, and N. J. Mitra. GlobFit: Consistently fitting primitives by discovering global relations. In *ACM SIGGRAPH 2011*, pages 52:1–52:12, New York, NY, USA, 2011. ACM. 3

[33] L. Magri and A. Fusiello. T-linkage: A continuous relaxation of J-linkage for multi-model fitting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2014)*, pages 3954–3961, 2014. 3

[34] G. Mentges and R.-R. Grigat. Surface reconstruction from image space adjacency of lines using breadth-first plane search. In *IEEE International Conference on Robotics and Automation (ICRA 2016)*, pages 995–1002, May 2016. 1, 3, 7

[35] B. Mičušík and J. Košecká. Multi-view superpixel stereo in urban environments. *International Journal of Computer Vision (IJCV 2010)*, 89(1):106–119, Aug. 2010. 2

[36] P. Miraldo, T. Dias, and S. Ramalingam. A minimal closed-form solution for multi-perspective pose estimation using points and lines. In *European Conference on Computer Vision (ECCV 2018)*, 2018. 1

[37] A. Monszpart, N. Mellado, G. J. Brostow, and N. J. Mitra. RAPter: Rebuilding man-made scenes with regular arrangements of planes. *ACM Transactions on Graphics (TOG 2015)*, 34(4):103:1–103:12, 2015. 3

[38] L. Nan and P. Wonka. Polyfit: Polygonal surface reconstruction from point clouds. In *IEEE International Conference on Computer Vision (ICCV 2017)*, pages 2372–2380, Oct. 2017. 3, 4, 5, 7

[39] I. Nurutdinova and A. Fitzgibbon. Towards pointless Structure from Motion: 3D reconstruction and camera parameters from general 3D curves. In *IEEE International Conference on Computer Vision (ICCV 2015)*, pages 2363–2371, 2015. 2

[40] Y. Salaün. LineSfM, 2017. https://github.com/ySalaun/LineSfM. 6

[41] Y. Salaün, R. Marlet, and P. Monasse. Multiscale line segment detector for robust and accurate SfM. In *23rd International Conference on Pattern Recognition (ICPR 2016)*, pages 2000–2005, Dec. 2016. 1

[42] Y. Salaün, R. Marlet, and P. Monasse. Robust and accurate line- and/or point-based pose estimation without Manhattan assumptions. In *European Conference on Computer Vision (ECCV 2016)*, 2016. 1

[43] Y. Salaün, R. Marlet, and P. Monasse. Robust SfM with little image overlap. In *5th International Conference on 3D Vision (3DV 2017)*, Qingdao, China, Oct. 2017. 1, 6

[44] C. Schmid and A. Zisserman. Automatic line matching across views. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 1997)*, pages 666–671, Washington, DC, USA, June 1997. 2

[45] R. Schnabel, P. Degener, and R. Klein. Completion and reconstruction with primitive shapes. *Computer Graphics Forum (CGF 2009)*, 28(2):503–512, 2009. 2

[46] R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum (CGF 2007)*, 26(2):214–226, June 2007. 3

[47] J. L. Schönberger and J.-M. Frahm. Structure-from-Motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, 2016. 6, 7

[48] T. Schöps, J. L. Schönberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, and A. Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Conference on Computer Vision and Pattern Recognition (CVPR 2017)*, 2017. 6

[49] S. N. Sinha, D. Steedly, and R. Szeliski. Piecewise planar stereo for image-based rendering. In *IEEE 12th International Conference on Computer Vision (ICCV 2009)*, pages 1881–1888, Sept. 2009. 2

[50] T. Sugiura, A. Torii, and M. Okutomi. 3D surface reconstruction from point-and-line cloud. In *International Conference on 3D Vision (3DV 2015)*, pages 264–272, Oct. 2015. 2

[51] R. Toldo and A. Fusiello. Robust multiple structures estimation with J-linkage. In *10th European Conference on Computer Vision (ECCV 2008)*, pages 537–547, 2008. 3

[52] M. van Kreveld, T. van Lankveld, and R. C. Veltkamp. On the shape of a set of points and lines in the plane. *Computer Graphics Forum (CGF 2011)*, 30(5):1553–1562, 2011. 8

[53] M. van Kreveld, T. van Lankveld, and R. C. Veltkamp. Watertight scenes from urban LiDAR and planar surfaces. *Computer Graphics Forum (CGF 2013)*, 32(5):217–228, 2013. 8

[54] Y. Verdie, F. Lafarge, and P. Alliez. LOD Generation for Urban Scenes. *ACM Transactions on Graphics (TOG 2015)*, 34(3):15, 2015. 3

[55] M. Wan, Y. Wang, and D. Wang. Variational surface reconstruction based on Delaunay triangulation and graph cut. *International Journal for Numerical Methods in Engineering*, 85(2):206–229, 2011. 2

[56] Z. Wang, F. Wu, and Z. Hu. MSLD: A robust descriptor for line matching. *Pattern Recognition (PR 2009)*, 42(5):941–953, 2009. 1

[57] J. Witt and G. Mentges. Maximally informative surface reconstruction from lines. In *IEEE International Conference on Robotics and Automation (ICRA 2014)*, pages 2029–2036, May 2014. 1, 3, 7

[58] A. Zaheer, M. Rashid, and S. Khan. Shape from angle regularity. In *12th European Conference on Computer Vision (ECCV 2012)*, pages 1–14, Florence, Italy, Oct. 2012. 2, 3, 4

[59] L. Zebedin, J. Bauer, K. F. Karner, and H. Bischof. Fusion of feature- and area-based information for urban buildings modeling from aerial imagery. In *10th European Conference on Computer Vision (ECCV 2008)*, pages 873–886, Marseille, France, Oct. 2008. 2

[60] Y. Zhang, H. Yang, and X. Liu. A line matching method based on local and global appearance. In *4th International Congress on Image and Signal Processing (ICISP 2011)*, pages 1381–1385, Oct. 2011. 1

[61] M. Zuliani, C. S. Kenney, and B. S. Manjunath. The multi-RANSAC algorithm and its application to detect planar homographies. In *IEEE International Conference on Image Processing (ICIP 2005)*, pages 153–156. IEEE, Sept. 2005. 3