Surface Reconstruction from 3D Line Segments — Supplementary Material —

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

[1]LIGM (UMR 8205), ENPC, UPE, France [2]ONERA, Université Paris-Saclay, Palaiseau, France [3]valeo.ai, Paris, France

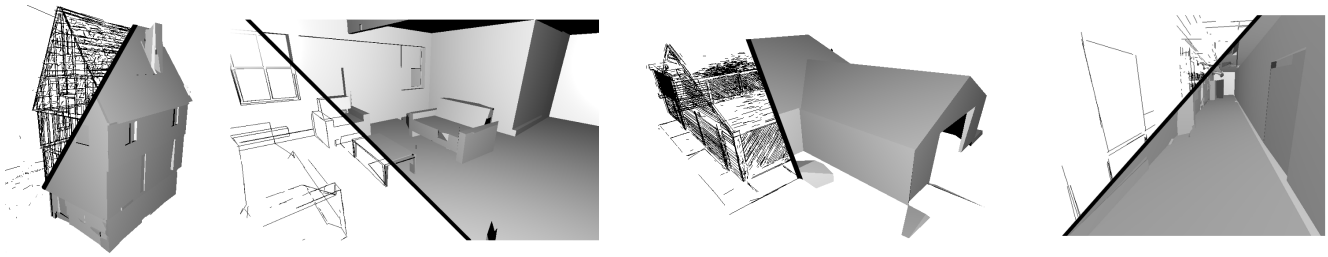


Figure 1: Input lines and reconstructed surfaces on 4 datasets (from left to right): TimberFrame, HouseInterior, Barn, Terrains.

This document is the supplementary material of the 3DV 2019 paper *Surface Reconstruction from 3D Line Segments*. It provides additional details and a more formal expression of our method. It explains our parameter setting via both a formal and an empirical study. It illustrates the noisy nature of our input data and analyzes its impact using a sensitivity study. It shows the relevance of the main ingredients of our approach through an extensive ablation study. More reconstruction examples are also on Figures 1 and 3.

Contents

1. Line-based plane detection 1

2. Surface reconstruction 3

3. Quantitative evaluation of reconstructions 6

4. Parameter setting and sensitivity study 7

5. Quality of input data and sensitivity study 9

6. Ablation study 11

7. Video, code and data 14

1. Line-based plane detection

Section 3 in the paper describes our line-specific method based on RANSAC for detecting planes in a line segment cloud. We provide here a more formal presentation of that same material, with an explicit algorithm (cf. Figure 2) and a few more details. The redundancies w.r.t. the paper are intended for readability, to make the section self-contained.

Candidate plane construction. In RANSAC, a candidate model is generated by sampling the minimum amount of data items required to create a model. To define a plane, 2 non-collinear line segments are enough. (A line and a point actually suffice.)

We define a plane P from two line segments l_a, l_b as follows. We note $\vec{l}$ a vector from one endpoint of l to the other (order does not matter), l^∞ the infinite line passing through l, and $|\, l \,| = ||\, \vec{l} \,||$ the length of l. The direction of the normal to P is given by $\vec{l_a} \times \vec{l_b}$, and a point belonging to P is given as the closest point to l_a^∞ and l_b^∞, i.e., the midpoint of the shortest line segment joining l_a^∞ and l_b^∞. This formulation give no particular role to the segment endpoints, that are notoriously noisy.

There are actually two ways for l_a and l_b to be co-planar: they can be parallel, or the distance between l_a^∞ and l_b^∞ can be small: $d(l_a^\infty, l_b^\infty) \leq \epsilon$, for some threshold $\epsilon > 0$. (Both criteria can be achieved at the same time.) For the reconstruction of man-made environments such as buildings, candidate models made from parallel lines are to be avoided to prevent the generation of many bad planes. Indeed, two random vertical line segments (e.g., detected on windows) are parallel but statistically unlikely to support an actual, physical plane. To address this issue, we threshold the angular measure $||\vec{l_a} \times \vec{l_b}||/(|\, l_a \,||\, l_b \,|)$, which also excludes the degenerate case of two collinear line segments, for which generating a plane normal is impossible or unstable.

Greedy detection and multi-support issues. Then, following RANSAC principle, we sample planes as line pairs and perform an iterative extraction of the most significant planes, i.e., with the largest line support after a given number of sampling trials.

However, contrarily 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 (with a large line support) and tend to miss smaller planes (with a smaller support). A failure case is illustrated on Figure 15.

Conversely, we cannot consider all segments as available at each plane detection iteration: it would statistically lead to multiple detections of the same large planes and again would miss detections of planes with small support.

A natural method to allow a datum to be part of several detected models in greedy RANSAC approaches is to remove inliers for succeeding model sampling but not for data assignment to models [14]. But for sparse data (which is the case for line segments in practice), 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, which also yield very bad results.

After a number of meaningful but unfruitful attempts to fix those issues, we propose below a solution based on this bounding of the number of detections, but with an additional condition to prevent shared lines to belong to similar planes (cf. "inlier selection").

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.

INPUT: set $\mathcal{L}$ of 3D line segments
$\mathcal{L}_0 \leftarrow \mathcal{L}$, $\mathcal{L}_1 \leftarrow \emptyset$, $\mathcal{L}_2 \leftarrow \emptyset$, $\Pi \leftarrow \emptyset$
while $|\,\mathcal{L}_0 \cup \mathcal{L}_1\,| \geq 2$ **and** $|\,\Pi\,| < N_{\text{max}}$ **do**
  $\Lambda_{\text{best}} \leftarrow \emptyset$ *// Current best set of coplanar lines*
  repeat N_{iter} **times**
    //// Sample a candidate plane by sampling 2 lines
    Pick $l_a \in \mathcal{L}_0 \cup \mathcal{L}_1$ *// 1st sample line*
    //// For 2nd sample, exclude lines of the plane of l_a if any
    Pick $l_b \in \mathcal{L}_0 \cup \mathcal{L}_1 \setminus \Lambda(\Pi(l_a))$ *// 2nd sample line*
    //// Make candidate plane and check consistency
    $P \leftarrow \text{plane}(l_a, l_b)$
    next if P degenerate **or** $d(l_a, P) > \epsilon$ **or** $d(l_b, P) > \epsilon$
    //// Gather line support for plane P
    $\Lambda \leftarrow \{l \in \mathcal{L}_0 \,|\, d(l, P) \leq \epsilon\} \cup \{l \in \mathcal{L}_1 \,|\, d(l, P \cap \Pi(l)) \leq \epsilon\}$
    //// Remember best candidate
    if $|\,\Lambda\,| > |\,\Lambda_{\text{best}}\,|$ **then** $\Lambda_{\text{best}} \leftarrow \Lambda$, $P_{\text{best}} \leftarrow P$ **end if**
  end repeat
  //// Update data structures
  $\Pi \leftarrow \Pi \cup \{P_{\text{best}}\}$ *// Set of detected planes*
  $\Lambda(P_{\text{best}}) \leftarrow \Lambda_{\text{best}}$ *// Support of best plane*
  $\forall\, l \in \Lambda_{\text{best}}, \Pi(l) \leftarrow \Pi(l) \cup \{P_{\text{best}}\}$ *// Planes supported by l*
  $\mathcal{L}_2 \leftarrow \mathcal{L}_2 \cup (\Lambda_{\text{best}} \cap \mathcal{L}_1)$
  $\mathcal{L}_1 \leftarrow (\mathcal{L}_1 \setminus (\Lambda_{\text{best}} \cap \mathcal{L}_1)) \cup (\Lambda_{\text{best}} \cap \mathcal{L}_0)$
  $\mathcal{L}_0 \leftarrow \mathcal{L}_0 \setminus (\Lambda_{\text{best}} \cap \mathcal{L}_0)$
end while
OUTPUT: set of planes Π, with related support $\Lambda(P)_{P \in \Pi}$

Figure 2: RANSAC-based plane detection from 3D lines segments, differentiating structural lines from textural lines.

- 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.

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 relatively poor support at a given iteration). 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 intersection of both planes, i.e., $d(l, P \cap P') \leq \epsilon$, that it is in the cylinder of axis $P \cap P'$ and radius ϵ. Again, several distances can be used, e.g., average or maximum distance to the line. Note that 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. As mentioned above, Without the $d(l, P \cap P') \leq \epsilon$ condition, the algorithm would tend to associate l to two planes P and P' which are very similar, thus failing to detect crease lines. This condition is actually a crucial ingredient in our algorithm.

Plane selection. Finally, we repeat model sampling N_{iter} times and keep the plane with the largest number of inliers, i.e., we maximize $|\Lambda(P)|$. The whole algorithm is summarized in Figure 2.

This plane construction is in contrast with [14], that samples and populates planes from (2D) line pairs instead of (3D) lines, making inlier search quadratic, not linear. To compensate, they heuristically only consider line pairs defined by intersected segment extensions, which is highly unstable due to noise in endpoints and induces plane splitting at occlusions (cf. Fig. 8 of their paper). We have none of these downsides.

Please also note that structural lines in [14] are found with heuristics after RANSAC, considering plane pairs and candidate lines, which only makes sense because they have few ($<$10) planes. In contrast, we get them directly in RANSAC, without heuristics, in greater number, and for a much larger set of planes.

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

Plane fusion. We observed that modeling a building may require different levels of details. On the one hand, we can be interested in small plane differences such as door or window jambs w.r.t. walls, or even baseboards and switches w.r.t. walls. This could be controlled by the ϵ parameter (assuming observations are made with this level of accuracy). On the other hand, setting a small ϵ may easily break a wall or a ceiling into several fragments because it is not perfectly planar due to construction inaccuracies or load deflections. While it is useful to retain a high degree of accuracy in low-level 3D capture for some applications, this arbitrary fragmentation is not desirable for abstract building modeling: such large mostly planar surfaces should be modeled as a single piece. For each country, there are actually standards (official or not) defining construction tolerances, e.g., 1 cm error every 2 m for walls. (Tolerances are actually more specific to the construction part and material.)

To cope with this practical issue, which actually occurs in real data, 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 additionally impose a maximum angle θ_{fus} when merging two planes and minimum proportion of common inliers of at least p_{fus}. 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 them. If it succeeds, the two planes are removed, the new refitted plane is added, and the priority queue based on angles is updated before the next merging attempt. If it fails, the pair of planes is discarded and the next pair is considered, until no merging can apply, yields a set of merged planes Π_{fus}. This procedure is similar to a heuristics used in Polyfit [9].

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

2. Surface reconstruction

Section 4 of the paper presents our method for reconstructing a surface on detected planes and observations of 3D line segments. We provide here a more formal expression of the energy we minimize (again with intended redundancies w.r.t. the paper, for self-containedness).

We follow [3, 2] and consider a scene bounding box, partition it into volumic cells constructed from the planes, and assign each cell with a status 'full' or 'empty' depending on observed lines segments, with a regularization prior coping with sparse or missing data. The reconstructed surface is then the interface between full and empty cells. By construction, it is guaranteed to be watertight and free from self-intersections.

The volume partition is given by a cell complex $\mathcal{C}$ made from a arrangement of planes detected in the line cloud. In our experiments, we use the full-extent plane arrangement, i.e., with planes extending all the way to the scene bounding box. Although it limits in practice $\mathcal{C}$ to a few hundred planes, as the complexity of building such a full-extent arrangement is cubic in the number of planes, it is generally enough to model a single room by focusing on the most salient planes, i.e., on planes with the largest number of supporting lines. In any case, while there can be more

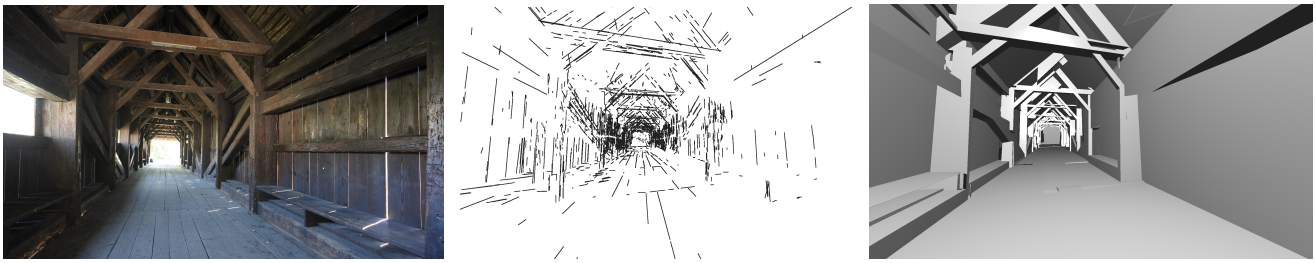


Figure 3: Dataset Bridge from ETH3D: input image sample, 3D line segments extracted by Line3D++ and reconstruction.

than thousands of planes detected in a dense point cloud, the number of detected 3D lines in a scene is in general on the order of a few thousands, leading anyway only to hundreds of detected planes.

Note however that using a full-extent arrangement is not intrinsic to our method; it merely provides a baseline, that is consistent with the sparsity of line detections and has the advantage of not depending on plane insertion order. Yet more planes could be fitted into the cell complex $\mathcal{C}$ by limiting their extent to a region around their supporting lines, which makes sense when there is little missing data, i.e., few unobserved surfaces or sparsely supported areas. For instance, a coarse voxel-based partition and heuritics could bound the extend of planes as in [3]. The 2D kinetic polygonal plane partitioning from line segments in images [1] could also be extended to 3D to provide a volume partitioning from planar regions, allowing a more principled complex, with a much larger set of planes, while reducing the number of cells and the complexity of their construction. (Preliminary unpublished results show that relevant complexes with 10k planes can be built this way.) Besides, defining a notion of extent for line-detected planes, similar to α-shapes in the case of points [4] but adapted to lines [12, 13], could also be used to introduce so-called 'ghost planes', corresponding to unobserved, hidden planes at occluding edges of observed surfaces [3, 2]. We leave that for future work.

Concretely, 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.

$$E(\mathbf{x}) = E_{\text{prim}}(\mathbf{x}) + E_{\text{vis}}(\mathbf{x}) + E_{\text{regul}}(\mathbf{x}) \qquad (1)$$

Dealing with noise. To deal with possible noise in input data, instead of introducing slack in the choice of cells penalized for not being at the reconstructed surface and letting regularization make the right choices [2], which induces a heavier formulation and resolution, we assume that plane extraction from 3D line segments did a good-enough job at detecting planes and assigning them a segment support: although there can be outliers among detected line segments in $\mathcal{L}$, detected planes in Π and plane supports $\Lambda(P)$ for $P \in \Pi$, the segments supporting a plane P are considered as noisy inliers, projected on P, and then treated as noiseless data.

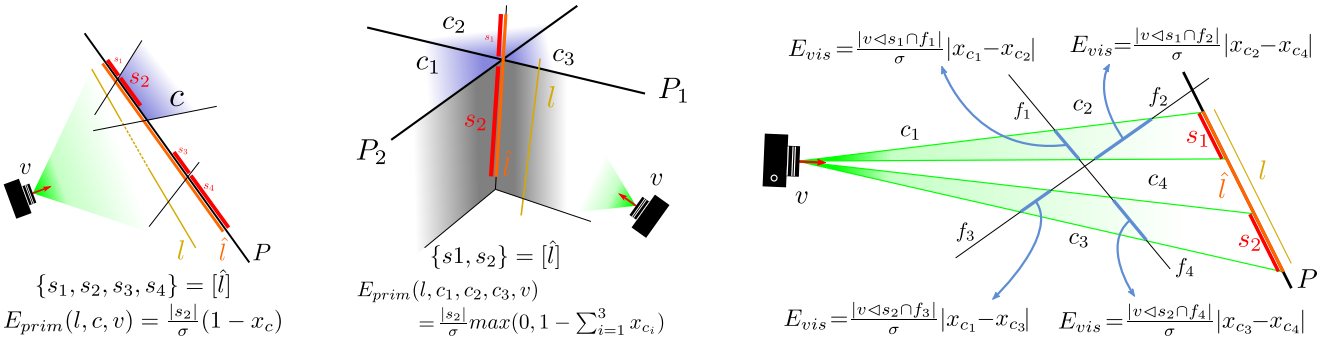
Formally, for any line segment $l \in \mathcal{L}_1$ (resp. $l \in \mathcal{L}_2$), i.e., if l supports a single plane P (resp. two planes P_1, P_2), we consider $\widehat{l}$, the orthogonal projection of segment l on plane P (resp. on the infinite line $P_1 \cap P_2$). A segment in $\mathcal{L}_0$, 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).

Dealing with viewpoints. Each 3D line segment $l \in \mathcal{L}$ is viewed from a set of viewpoints $\mathcal{V}(l)$. However, because of possible occlusions, not every part of l are seen from every viewpoint. Given a viewpoint $v \in \mathcal{V}(l)$, we consider l_v, the fraction of l that is viewed from v. Note that l_v is not necessarily continuous; it may consists of several disconnected sub-segments ℓ of l due to multiple occlusions when seen from v: $\bigcup_{\ell \in l_v} \ell = l_v$. We note $v \lhd l_v$ the visibility triangle(s) between v and the sub-segments ℓ of l_v.

For technical reasons, we also consider sub-fragments s of these visible sub-segments ℓ of l_v, corresponding to their intersection with the cell complex $\mathcal{C}$: we note $l_v = \{c \cap \ell \mid c \in \mathcal{C}, \ell \in l_v, c \cap l_v \neq \emptyset\}$ and we have $\bigcup_{s \in l_v} s = l_v$.

All this is combined with the above "noiseless" approximation: $\widehat{l}_v$ is the projection on the planes supported by l of the fractions ℓ of l seen from v, if any, and $\widehat{l}_v$ is its subsegmentation into sub-fragments s along complex $\mathcal{C}$.

Primitive term. The primitive term penalizes 3D line segments that support detected planes but do not lie on the reconstructed surface. More precisely, it actually only penalizes the absence of matter "behind" a line segment w.r.t. a viewpoint; it does not penalize the presence of matter just in front of it, letting the visibility term do it. Segments that support no plane are ignored here, consistently with the piecewise-planar approximation, as they would be far from



(a) Primitive term, $l \in \mathcal{L}_1$ (b) Primitive term, $l \in \mathcal{L}_2$ (c) Visibility term

Figure 4: Energy terms.

any possibly reconstructed surface. They are thus considered as if outliers regarding data fidelity. (They are however used below for visibility consistency.)

Concretely, for a textural line $l \in \mathcal{L}_1$, reprojected on $\Pi(l)$ as $\widehat{l}$ and seen from a viewpoint $v \in \mathcal{V}(l)$ as sub-segments $s \in \widehat{l}_v$, we consider each cell $c \in \mathcal{C}$ which is behind a sub-segment s w.r.t. v, and penalize it if not full, i.e., with a cost $1 - x_c$, multiplied by the length $|\, s \,|$ to give more weight to longer detections. To get a dimensionless cost, we actually normalize it by a scale of interest σ. This configuration is illustrated on Figure 4(a).

The case of a structural line $l \in \mathcal{L}_2$, at the intersection of two planes, is similar. But it cannot be treated as a texture line on each planes as it would lead to terrible results, as illustrated on Figure 17. In fact, as l can lie at an occluding edge w.r.t. the viewpoint, the cells right behind $\widehat{l}$ w.r.t. v do not necessarily have to be full. What we want to express is the fact that, apart from the cell in front of a sub-segment $s \in \widehat{l}_v$, at least one of the three other cells adjacent to s should be full. If only one is full, the sub-segment lies at a salient edge; if all three cells are full, the sub-segment lies at a reentrant edge; and if two adjacent cells are full, the structural line lies on a plane and is thus actually considered as textural. (There is also a degenerate case where the cells in front and behind s are empty, and the other, "side cells" are full.) To penalize only the case when all three cells are empty, i.e., when $\sum_c x_c = 0$, we consider a cost of $\max(0, 1 - \sum_c x_c)$, which is equal to 1 in this case, and to 0 in all the other configurations. As for textural lines, we weight it by $|\, s \,|$, normalized by the scale of interest σ. This configuration is illustrated on Figure 4(b).

Formally, textural and structural lines can be treated with this single expression covering both specific cases:

$$E_{\text{prim}}(\mathbf{x}) = \sum_{l \in \mathcal{L} \setminus \mathcal{L}_0} \sum_{v \in \mathcal{V}(l)} \sum_{s \in \widehat{l}_v} \frac{|\, s \,|}{\sigma} \max(0, 1 - \sum_{\substack{c \in \mathcal{C} \\ c \cap (v \lhd s) = s}} x_c) \quad (2)$$

Condition $c \cap (v \lhd s) = s$ says that the visibility triangle $v \lhd s$ intersects cell c only at sub-segment s, which is true only for cells adjacent to s and not in front of s w.r.t. v. If l is textural, i.e., in $\mathcal{L}_1$, then s lies on the interior of a face of the complex and only two cells are adjacent to s. The last part of the formula reduces to $\max(0, 1 - x_c)$, i.e., to $1 - x_c$, where c is the cell with no intersection with the visibility triangle but on s, i.e., behind s w.r.t. v.

Visibility term. As in [3, 2], the visibility term penalizes the number of reconstructed surface boundaries between viewpoints and segments. This somehow measures (twice) the number of times a 3D line segment is considered a detection outlier as it should not be visible from a given viewpoint. Besides, we consider that the longer the offending part of the outlier (as only a fraction of the segment might be "wrongly" visible), the higher the penalty. Our visibility cost is thus weighted by the reprojection length of erroneous segments on occluding surfaces, i.e., on surfaces encountered by visibility rays between viewpoint and segment. In contrast, we consider that the thickness of the occluding surfaces does not matter (nor does the the volume of reconstructed cells traversed by visibility rays).

More formally, we note c_f^{+v} the cell on the side of a face $f \in \mathcal{F}$ of $\mathcal{C}$ which is on the same side as v w.r.t. the plane of f, and c_f^{-v} the cell on the opposite side. Noting cell occupancy $x_f^{+v} = x_{c_f^{+v}}$ and $x_f^{-v} = x_{c_f^{-v}}$, we define:

$$E_{\text{vis}}(\mathbf{x}) = \lambda_{\text{vis}} \sum_{l \in \mathcal{L}} \sum_{v \in \mathcal{V}(l)} \sum_{f \in \mathcal{F}} \frac{|\, (v \lhd \widehat{l}_v) \cap f \,|}{\sigma} |\, x_f^{+v} - x_f^{-v} \,| \quad (3)$$

This configuration is illustrated on Figure 4(c).

Regularization term. E_{regul} is the sum of two terms penalizing the total length of reconstructed edges and the number of corners [2]. (Area penalization as in [2] makes little sense here due to the low density of observations in some regions.) We assume the optimal (simplest) surface is a balance of short edge length and few corners (with weights $\lambda_{\text{edge}}, \lambda_{\text{corner}}$) while being consistent with observations.

Energy minimization. Due to the specific treatment of structural lines, the primitive energy term is not linear: it involves maximum values (cf. Eq. (2). However, the optimization problem

$$\begin{aligned} &\underset{\mathbf{x}}{\text{minimize}} \sum_{l\in\mathcal{L}\setminus\mathcal{L}_0}\sum_{v\in\mathcal{V}(l)}\sum_{s\in\widehat{l}_v}\frac{|\,s\,|}{\sigma}\max(0,1-\sum_{\substack{c\in\mathcal{C}\\ c\cap(v\lhd s)=s}}x_c)\\ &\text{subject to}\quad 0\le x_c\le 1,\ c\in\mathcal{C}. \end{aligned} \tag{4}$$

can be rewritten as a standard linear program by introducing for each $\max$ term a new slack variable $z_s \in \mathbb{R}$:

$$\begin{aligned} &\underset{\mathbf{x},\mathbf{z}}{\text{minimize}} \sum_{l\in\mathcal{L}\setminus\mathcal{L}_0}\sum_{v\in\mathcal{V}(l)}\sum_{s\in\widehat{l}_v}\frac{|\,s\,|}{\sigma}z_s\\ &\text{subject to}\quad 0\le x_c\le 1,\ c\in\mathcal{C}.\\ &\qquad\quad z_s\ge 1-\sum_{\substack{c\in\mathcal{C}\\ c\cap(v\lhd s)=s}}x_c\\ &\qquad\quad z_s\ge 0 \end{aligned} \tag{5}$$

where $\mathbf{z}$ is the vector of all variables z_s.

The regularization term also is not linear. Minimizing the edges length and the number of corner indeed involves higher-order constraints: the presence of an edge (resp. corner) depends on the value of 4 (resp. 8) adjacent cells (full or empty). We reformulate these constraints, that are hard to solve, using linear terms only as proposed by [2], i.e., using the absolute value of linear combinations of cell values, which can also be turned into an equivalent linear program with extra slack variables.

The resulting energy minimization problem is formulated as mixed-integer programming, with integral values (0 or 1) for the occupancy of cells x_c and continuous values for slack variables. As solving it is NP-hard, we also do as in [2]: we relax the problem for optimization, i.e., the problem is solved for $x_c \in [0, 1]$ for all $c \in \mathcal{C}$. This corresponds to a linear program that can be solved efficiently using off-the-shelf solvers. The obtained fractional values for variables x_c are rounded independently of each other.

3. Quantitative evaluation of reconstructions

We performed quantitative evaluations of the quality of our reconstructions using our datasets with ground truth. In this section we describe the metrics we used in our experiments, as well as our setting and principles when varying a parameter or input data to study the sensitivity of our method (in following sections).

Metrics to assess the quality of surface reconstruction. We used 4 metrics derived from the Metro distance.

To compare a reconstructed mesh M_{recons} with a ground-truth mesh M_{gt}, we first sample 2 millions of points on the surface of each mesh. We then compute the distances of each point of M_{recons} to their nearest neighbour in M_{gt}; we note this set $D_{\text{recons}\to\text{gt}}$. Conversely, we also compute the set of distances $D_{\text{gt}\to\text{recons}}$.

The 4 metrics we use are the following:

- The max Metro distance measures the worst reconstruction error:

$$\max(D_{\text{recons}\to\text{gt}} \bigcup D_{\text{gt}\to\text{recons}})$$

- The mean Metro distance measures the average reconstruction error:

$$\text{mean}(D_{\text{recons}\to\text{gt}} \bigcup D_{\text{gt}\to\text{recons}})$$

- The "95%-completeness" is the 95% percentile of $D_{\text{gt}\to\text{recons}}$. It measures the distance under which most of the ground truth surface has been reconstructed.

- The "95%-precision" is the 95% percentile of $D_{\text{recons}\to\text{gt}}$. It measures the distance under which most of what has been reconstructed is close enough to the ground truth.

We use these metrics to assess the quality of reconstructed surfaces (cf. Figure 4(7) of the paper) as well as to define the value of our parameters (see following section).

Varying parameters or input data. In the following sections, starting from the default parameter setting in Table 1, we vary the value of a chosen parameter (e.g., λ_{vis}) or the quantity of input data (e.g., the number of detected 3D line segments), and we display a graph representing the impact on a quantitative assessment of the reconstruction.

For these experiments, we evaluate on the synthetic dataset HouseInterior, for which we have a ground truth. Although the results are specific to this dataset, we observed that the conclusions are relatively general. In particular, these variation studies on HouseInterior lead our choice of the best default parameter values (see Section 4), but we did not observe a strong need to alter this parameter setting when running on other datasets, although small changes could sometimes provide slightly better results.

Please note that although we may vary a parameter continuously, the labeling of the cells in the plane arrangement as full or empty is discrete and thus can lead to strong changes on the metrics when the altered cells are large or when the measure is based on a maximum distance. As a consequence, curves showing the impact of parameter variations can display significant discontinuities.

Also, due to the RANSAC stage, our algorithm is not deterministic. For each set of parameters, we actually ran our method 5 times, and we report in the graphs both the average value of the quantity we monitor over these 5 runs as well as its standard deviation.

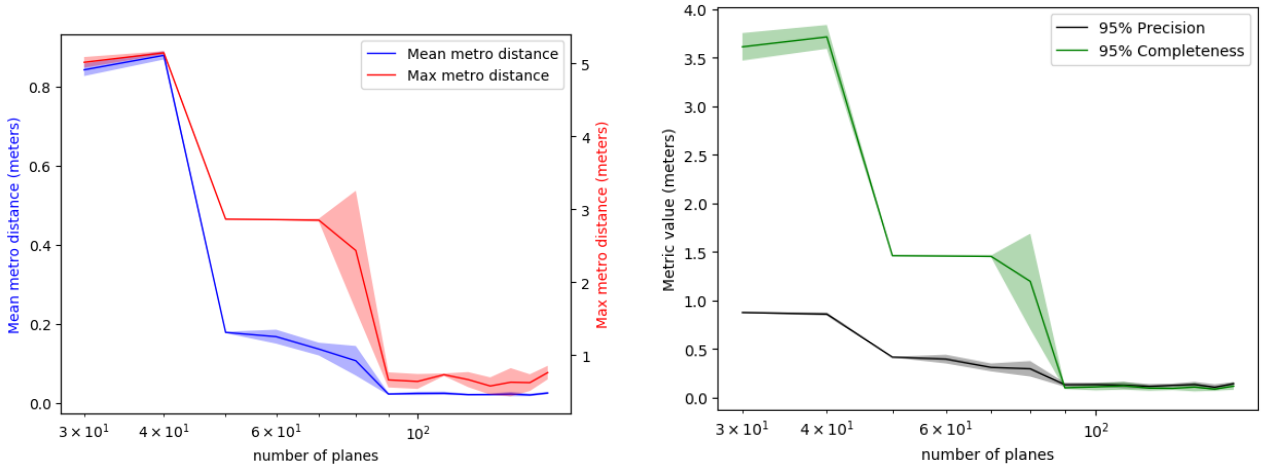


Figure 5: Impact of the number of planes N_{max} on the max/mean Metro distance and on the 95% precision/completeness.

4. Parameter setting and sensitivity study

We strove to reduce as much as possible the number of parameters of our method. Still, it has a few parameters, that have to be set. In this section, we study how to assign a value to these parameters, either using a formal argument (for the number of RANSAC iterations N_{iter}) or using an empirical justification (for the other parameters).

The parameter default setting is recalled in Table 1. Besides, the normalizing factor σ is set to 1 m.

ϵ	ϵ_{fus}	θ_{fus}	p_{fus}	N_{iter}	N_{max}	λ_{vis}	λ_{edge}	λ_{corner}
2 cm	$3\,\epsilon$	10°	20%	50k	160	0.1	0.01	0.01

Table 1: Parameters (all datasets have metric dimensions).

These parameters can be slightly adapted depending on the model. For instance, increasing λ_{vis} will dig more into the volumes, but will also make the model more sensitive to outliers. Likewise, increasing λ_{corner} and λ_{edge} will lead to more regularization, which is useful when data are missing, but can also lead to a loss of details.

In the following, we study the sensitivity of our method under different parameter settings, to discover ranges of parameter values leading to a good and relatively stable behavior. Table 1 resulted from this sensitivity study.

Number of RANSAC iterations N_{iter}. Our choice for setting the value of the number of RANSAC iterations N_{iter} can be justified as follows.

The number of 3D line segments detected in the scenes of our datasets varies between 1,000 and 10,000, and the number of detected planes is in practice limited to about 150 (see Table 2 of the paper). After the largest planes (with the most inliers) have been detected, the inlier rate of the current best plane can be quite low. Yet we must make sure that planes with a small line support are eventually detected, and first of all, sampled.

To define N_{iter}, we want to make sure at $\beta = 99\%$ chance that we sample the best plane assuming it is supported by at least $\alpha = 1\%$ of the line segments in the current value of $\mathcal{L}_0 \cup \mathcal{L}_1$. If we call X the event "not finding the best fitting plane after N_{iter} iterations among data which contain an inlier rate of α", its probability is:

$$P(X) = (1 - \alpha^k)^{N_{\text{iter}}}$$

where $k = 2$ is the number of line segment samples needed to generate a plane hypothesis. The criterion $P(X) \leq 1 - \beta$ in turn yields:

$$N_{\text{iter}} \leq \frac{\log(1 - \beta)}{\log(1 - \alpha^k)}$$

The right term evaluates to 46,049 with the given values for α and β. Given that the number of candidate line segments actually decreases at each iteration, as structural lines are detected, this is a worst case analysis.

In practice, in our experiments, we set $N_{\text{iter}} = 50000$. On a 12-core CPU, the whole RANSAC process (finding all planes) takes about 10 minutes for an upper bound of 10,000 lines with 50,000 iterations, which represents a minor fraction of the total reconstruction time.

Maximum number of detected planes N_{max}. We introduced a possible limit on the number of planes discovered by our RANSAC variant. The influence of that maximum number of detected planes N_{max} is shown on Figure 5.

As can naturally be expected, the more planes, the better. The mean Metro distance plateaus to a small value after 90 planes. So does the maximum Metro distance although with a small value but slightly larger variance. Both the 95%-precision and the 95%-completeness also plateau to a small value after 90 planes.

The only drawback of adding more planes, in the case of a surface reconstruction based on a full-extent plane arrangement, is the increasing computation time, as adding a plane has a cubic time complexity.

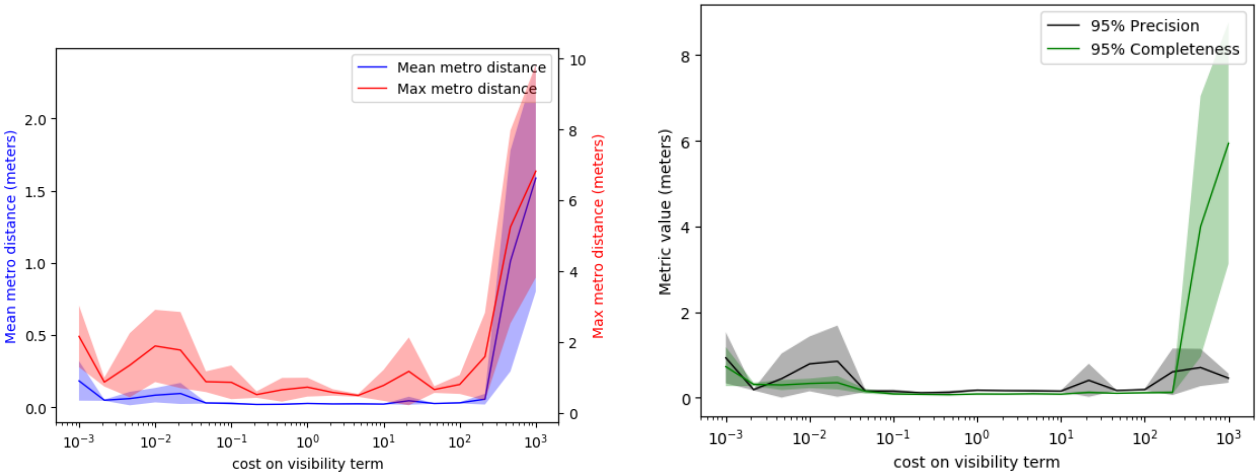


Figure 6: Impact of λ_{vis} on the max / mean Metro distance (left), and the 95% precision / completeness (right).

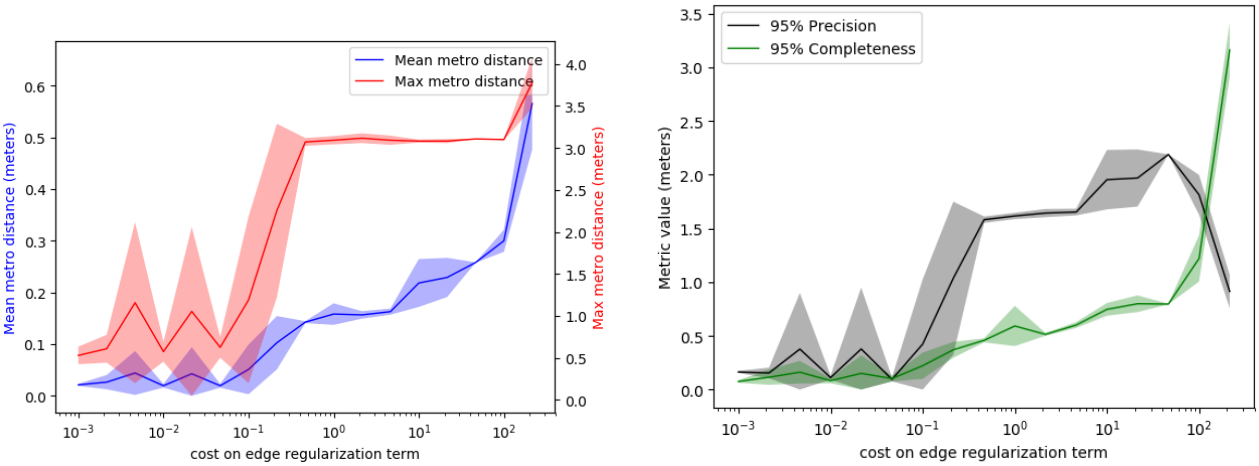


Figure 7: Impact of λ_{edge} on the max / mean Metro distance (left), and the 95% precision / completeness (right).

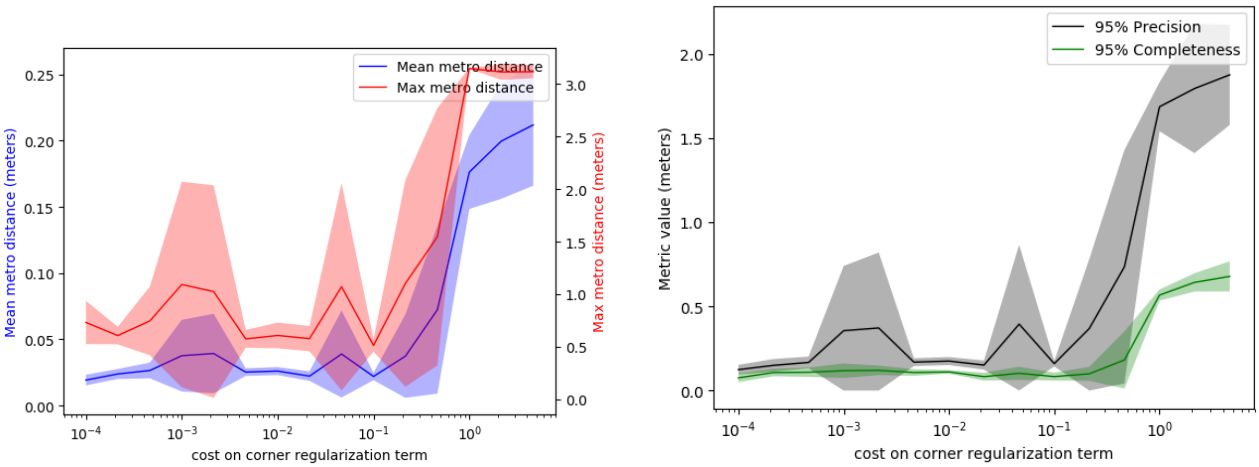


Figure 8: Impact of λ_{corner} on the max / mean Metro distance (left), and the 95% precision / completeness (right).

Weight of the visibility term λ_{vis}. The impact of varying the value of parameter λ_{vis} is represented on Figure 6. We observe a plateau starting a bit before 10^{-1} (our default parameter), which starts deteriorating after 10^{1} and even more after 10^{2}. As for the regularization parameters below, this observation qualitatively also applies to the other datasets.

Weight of the edge term λ_{edge}. The impact of varying the value of parameter λ_{edge} is represented on Figure 7. The error remains small when λ_{edge} is under 10^{-1}, in particular around value 10^{-2} (our default parameter).

Weight of the corner term λ_{corner}. The impact of varying λ_{corner} is represented on Figure 8. As for λ_{edge}, the error remains small when λ_{corner} is under 10^{-1}, in particular on the plateau around value 10^{-2} (our default parameter).

5. Quality of input data and sensitivity study

In this section, we analyze the impact of the level of quality of the 3D line segments (from Line3D++) that we use as input to our method. We also study more generally the sensitivity of our method to the number of input images or the number of input 3D lines.

3D line segments detectors. There has been some recent approaches to adapt structure-from-motion (SfM) methods from points to line segments [15, 10]. However, few methods focus on the actual production of 3D line segments [7, 6], possibly reconstructing more lines than just what the SfM algorithms would produces, by leveraging on the calibration to match or significatively augment matched lines. To our knowledge, only Line3D++ [5, 6] provides code for this denser 3D line segment reconstruction.

In our experiments, we thus use Line3D++ to detect 3D lines segments from a set of images, and to provide as well sub-segments associated the each viewpoints.

Modest quality of input data from Line3D++. Although quite efficient, Line3D++ produces a somewhat noisy output on which we have little control:

- It can miss some important lines (see Figure 10), which may lead to pertinent planes not being detected.
- It also generate many outliers (see Figure 11), which pressurize the visibility and regularization terms.
- A number of actual, physical 3D lines (including shadows) are given multiple reconstructions (see Fig. 12).

We expect any improvement on these aspects, for a 3D line segment detector to be used as a preliminary stage to our method (which is out of the scope of this work), to have a particularly positive impact on our results.

Impact of Line3D++ parameters. These 3D line segments were obtained using the default parameter setting of Line3D++ [5]. We tried playing around with the parameters but did not get significantly better segments. In particular, we studied the influence of σ_p, a major regularization parameter of Line3D++, on the HouseInterior dataset. It is represented on Figure 13. The default value of Line3D++ for σ_p is 2.5, which is in a low plateau area of the graph.

Impact of the number of input images. We made a variant of the dataset MeetingRoom with 100 images and studied qualitatively the impact of providing a variable number of images as input (keeping a calibration based on all 100 images). Results are show on Figure 9 and compared to a point-based approach, namely Colmap [11] + Poisson [8] reconstruction. When going down from 100 images to 50 images only, the quality of our reconstruction is progressively reduced, but the general shape of the room as well as a number of details are preserved. In contrast, the point-based reconstruction with 100 images is filled with holes and degrades rapidly when the number of images decreases.

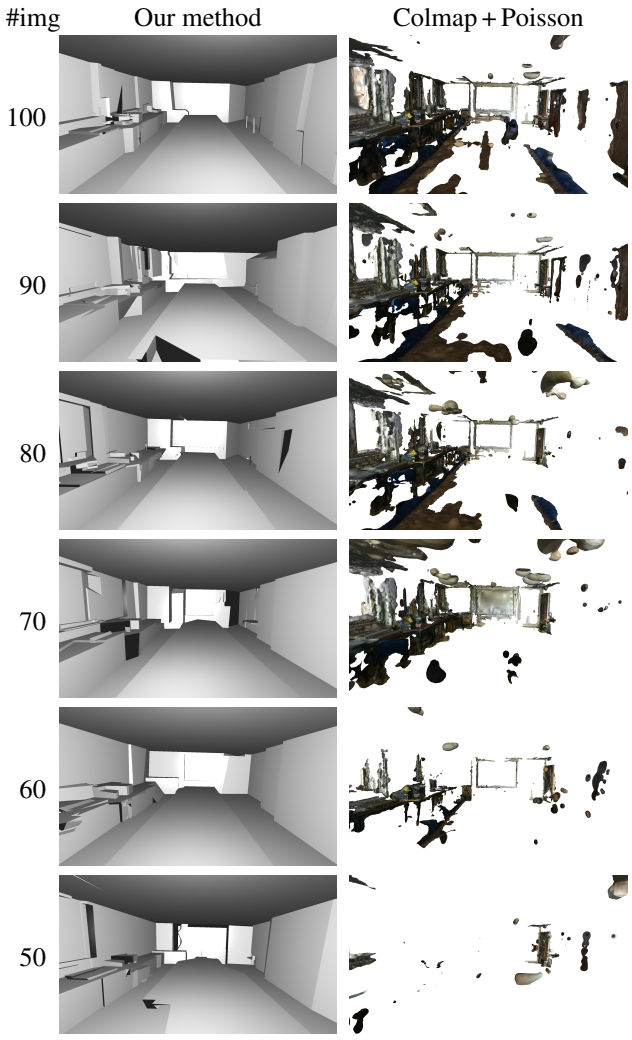


Figure 9: Variable number of input images: comparison of our method vs Colmap [11] + Poisson [8] reconstruction on a variant of MeetingRoom with 100 images.

Figure 10: Missing detections of 3D lines in the input: visible edge between floor and wall in a view of the MeetingRoom dataset (left), and extracted lines with Line3D++ where that important edge is missing to recover the floor (right).

Figure 11: Spurious detections of 3D lines in the input: view from the MeetingRoom dataset (left), and reconstructed line cloud using Line3D++ where vertical outlier lines float above the table (see also the video).

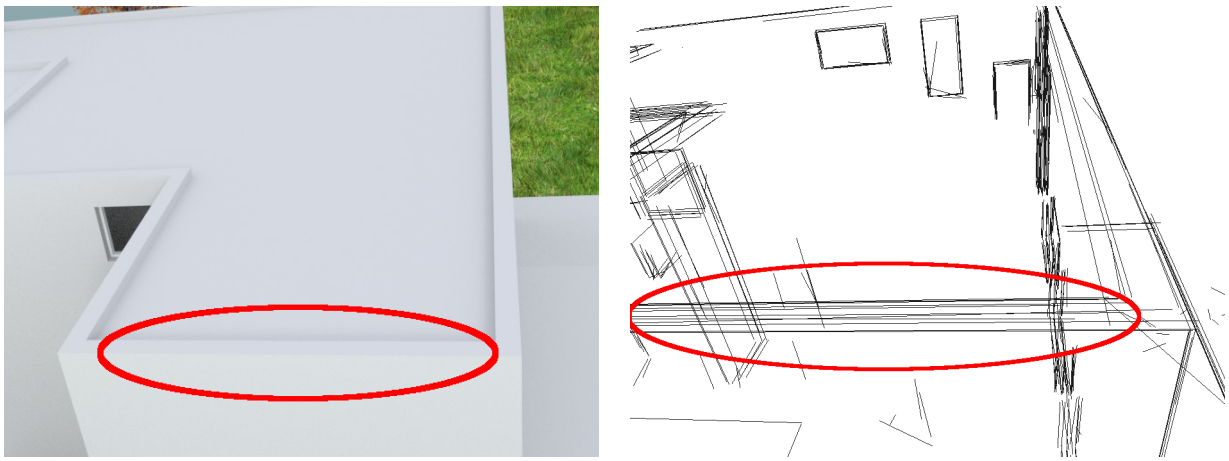


Figure 12: Spurious duplication of 3D lines in the input: view from the Andalusian dataset (left), and the extracted lines with Line3D++ where some actual 3D lines in the scene are detected many times at slightly different locations (right).

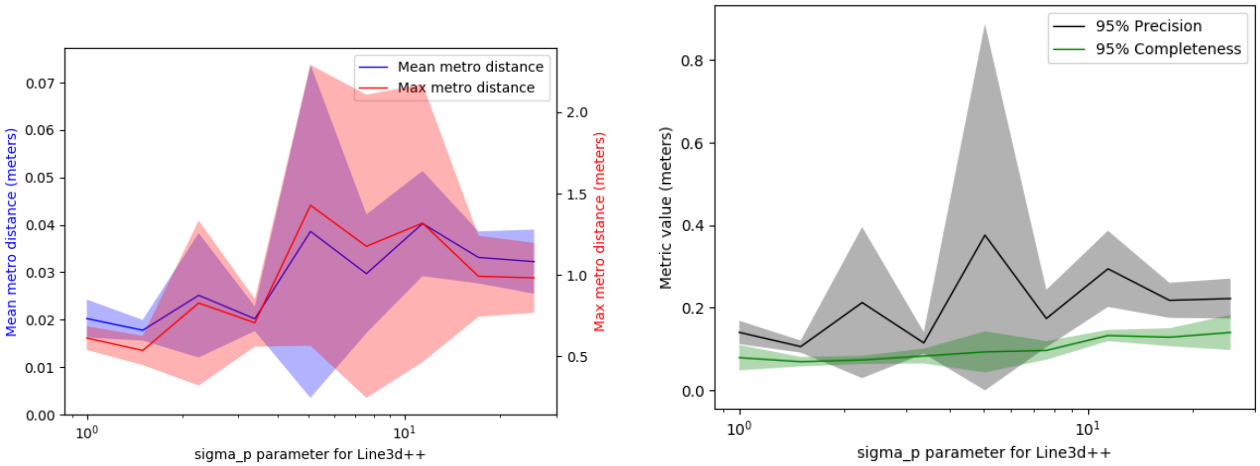


Figure 13: Impact of σ_p on the max / mean Metro distance (left), and the 95% precision / completeness (right).

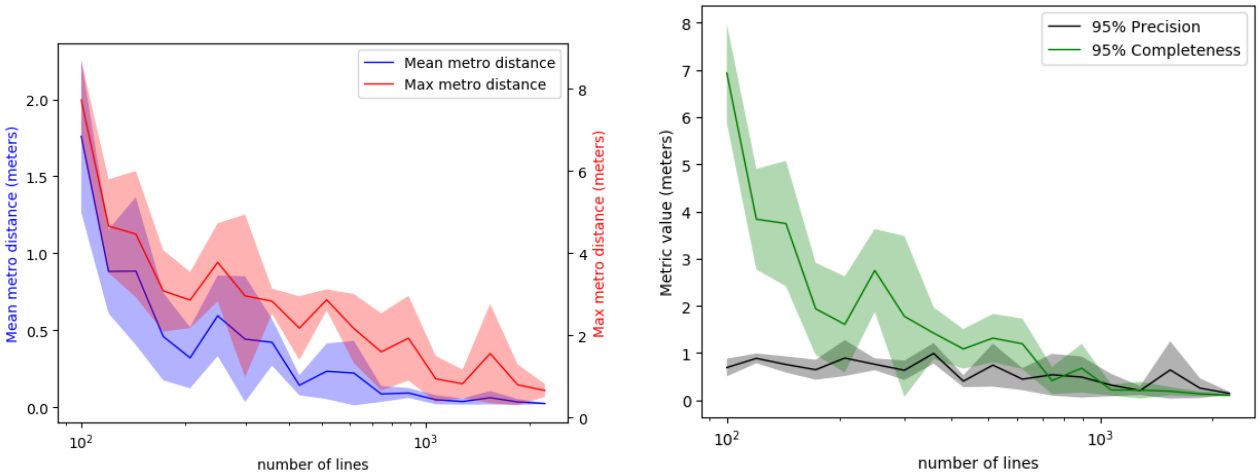


Figure 14: Impact of the number of input lines on the max/mean Metro distance, and the 95% precision/completeness.

.

Impact of the number of 3D line segments. We randomly sampled 3D line segments produced by Line3D++. The influence of the number of lines segments as input to our method is illustrated on Figure 14. As can be expected, the more lines, the better (in general), as it leads to a larger diversity of possible reconstructions, with more details. The mean Metro distance plateaus to a small value around 700 lines. Quite naturally, the max Metro distance remains sensitive until a few thousands lines are provided. 95%-precision also plateau a bit after a thousand lines, while 95%-completeness also retains a limited sensitivity. The computation time of the visibility term however increases, although mostly linearly in the number of lines.

6. Ablation study

To show that all line specificities in our method are useful if not crucial, we performed an extensive ablation study.

If one line supports at most one plane, as in an ordinary RANSAC framework, and if all lines supporting a detected plane are thus put aside before performing the following plane detection, as is the case in a greedy detection scheme, then fewer lines are available to detect succeeding planes, and less planes are detected. Experimentally, on HouseInterior, this detects only 45 planes instead of 120 with our method. The impact on reconstruction is illustrated on Figure 15: only the main planes are more or less reconstructed appropriately.

If inliers are decided only from the distance to the model, rather than from the distance to the intersection of the plane model with the plane that already supports the inlier in case the line is structural, i.e., if we make no difference between lines in $\mathcal{L}_0$ and lines in $\mathcal{L}_1$ when gathering inliers for a given model (see Section 3 of the paper, or Section 1 of this supplementary material, subsection "inlier selection"), then the algorithm tends to associate a line twice with more or less the same plane. Experimentally, on HouseInterior, this detects only 92 planes, vs 120 with our algorithm. See Figure 16 for an illustration of the impact on the reconstruction.

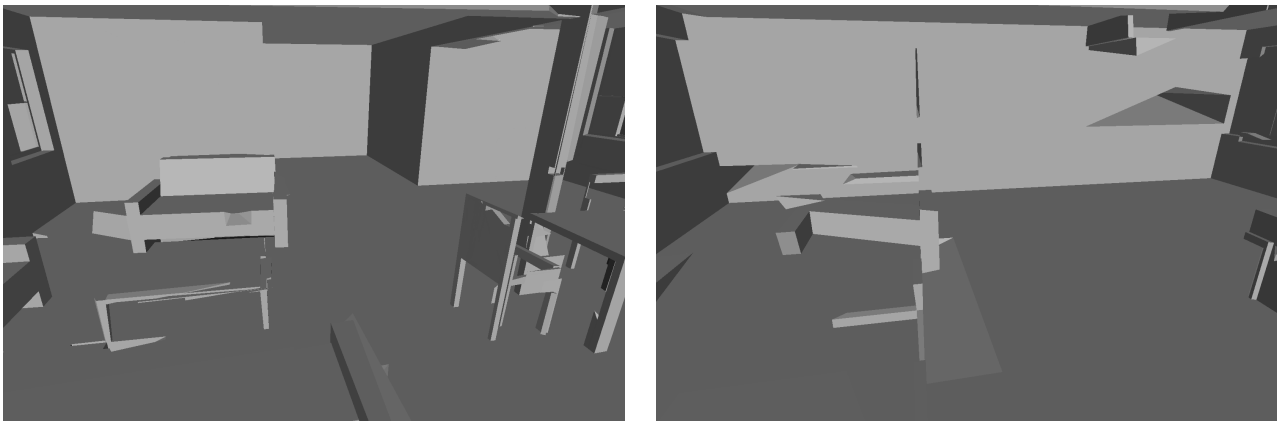


Figure 15: Reconstruction with our method where a line may support up to two planes (left, our RANSAC), or at most one plane (right, standard RANSAC).

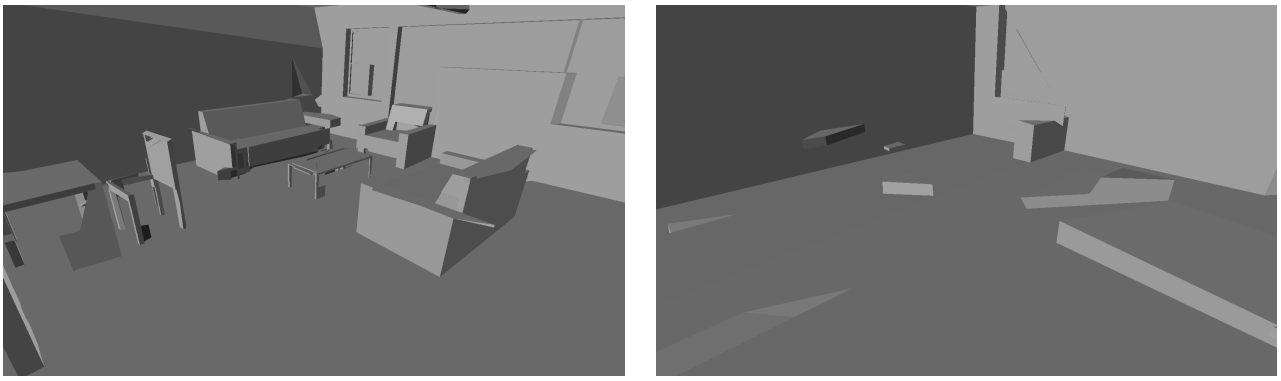


Figure 16: Reconstruction with our method where the gathering of inliers for a plane model relies on the distance to the plane intersection for lines already supporting a plane (left, our approach), and to the plane only (right, standard approach).

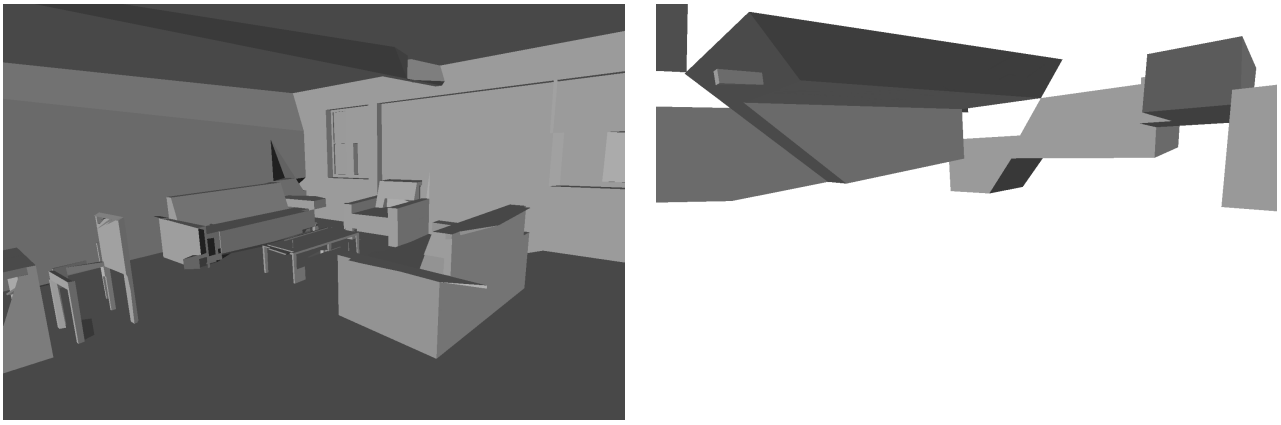


Figure 17: Reconstruction with our method where all structural lines are treated specifically (left), and considered as two textural lines, i.e., one for each supported plane (right).

If structural lines are treated as textural lines, i.e., if a structural line in $\mathcal{L}_2$ is simply treated as two textural lines in $\mathcal{L}_1$ (one for each supported plane), then reconstruction totally fails (see Figure 17). This illustrates the importance of distinguishing structural lines form textural lines, not only for plane detection but also for surface reconstruction.

If regularization penalizes only the length of edges, rather than only the number of corners, the reconstruction is not as good, as can be seen qualitatively on Figure 18. This fact has already been observed by [2]: providing a good balance between λ_{edge} and λ_{corner}, the regularization on both corners and edges is slightly better.

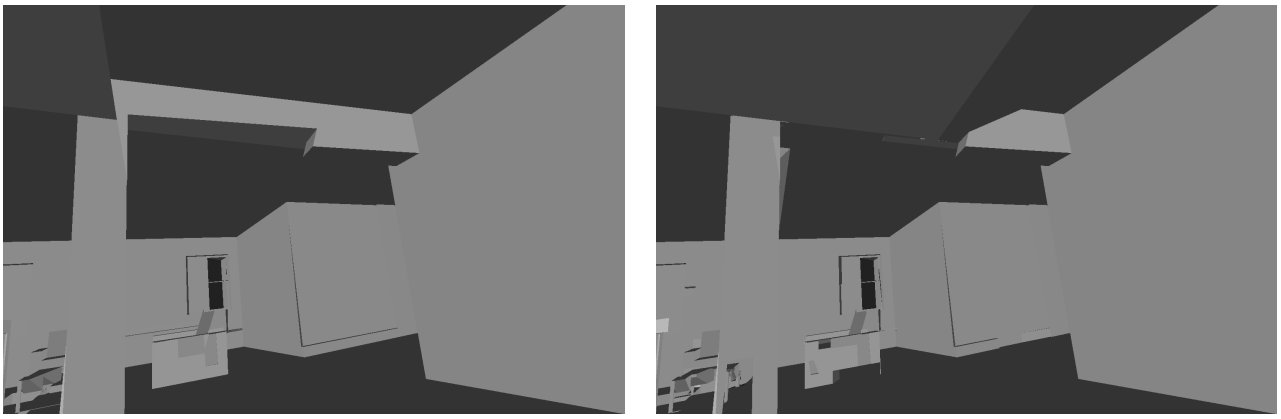


Figure 18: Reconstruction with a regularization on corners only (left), vs on edges only (right).

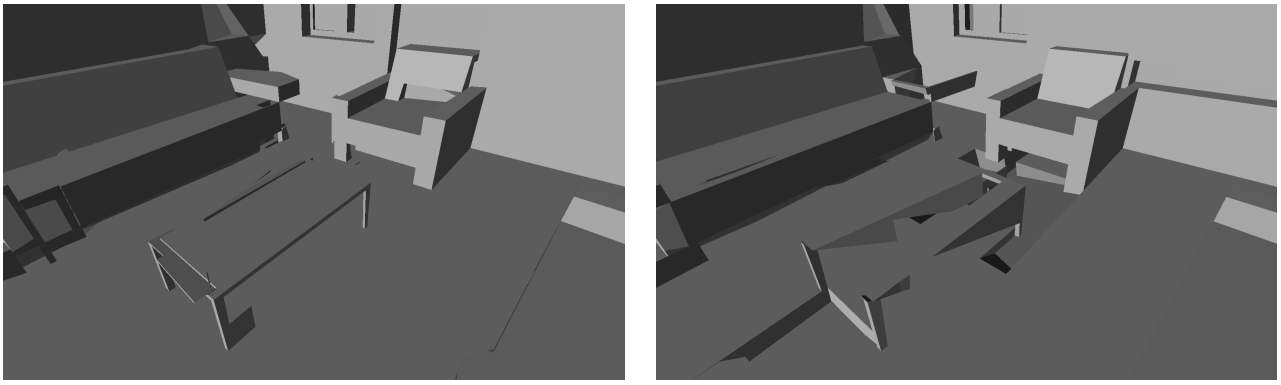


Figure 19: Reconstruction with a regularization on corners and edges (left), vs on corners only (right).

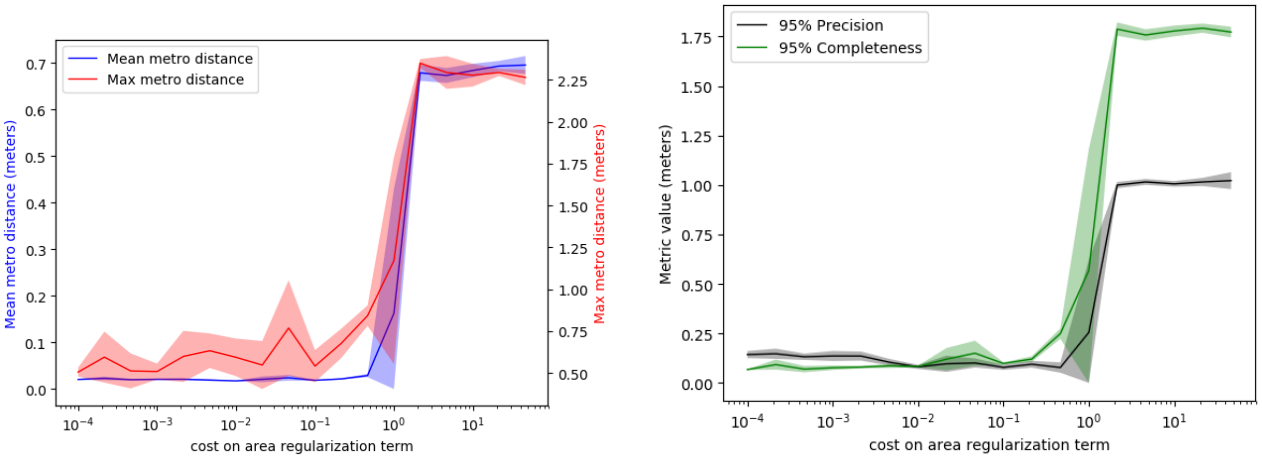


Figure 20: Impact of λ_{area} on the max / mean Metro distance (left), and the 95% precision / completeness (right).

If regularization penalizes only the number of corners, rather than both corners and edges, the reconstruction also is not as good (ground floor), although it can be locally better (armchair), as can be seen qualitatively on Figure 19. This fact has also already been observed by [2].

If regularization also penalizes the surface area, as a measure of surface simplicity like in [3, 2], the metrics do not improve but rather tend to deteriorate, as can be seen on Figure 20. The fact is this term makes little sense on the kind of building scenes we are considering because we want to be able to reconstruct large surfaces with little data, that mostly lies on their boundaries only.

7. Video, code and data

As described in the paper, we experimented on several datasets. To provide a better qualitative overview of the reconstructions, we propose along with this document a video showing through camera motion different viewpoints of the reconstruction of each scene. The video also offers relevant views of the 3D line segments we take as input (created with Line3D++ [5]), with occasional pauses to illustrate typical issues in the input 3D line cloud, i.e., noise, large number of outliers, missing segments, and multiply reconstructed 3D lines when there is only one in reality.

Code and new data (HouseInterior and Andalusian) will be released upon publication of this work.

References

[1] 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. 4

[2] A. Boulch, M. de La Gorce, and R. Marlet. Piecewise-planar 3D reconstruction with edge and corner regularization. *Computer Graphics Forum (CGF 2014)*, 2014. 3, 4, 5, 6, 12, 13

[3] 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. 3, 4, 5, 13

[4] 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. 4

[5] M. Hofer. Line3D++, 2016. https://github.com/manhofer/Line3Dpp. 9, 14

[6] M. Hofer, M. Maurer, and H. Bischof. Efficient 3D scene abstraction using line segments. *Computer Vision and Image Understanding (CVIU 2016)*, 3 2016. 9

[7] 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. 9

[8] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *4th Eurographics Symposium on Geometry Processing (SGP 2006)*, pages 61–70, 2006. 9

[9] 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

[10] 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. 9

[11] J. L. Schönberger and J.-M. Frahm. Structure-from-Motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, 2016. 9

[12] 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. 4

[13] 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. 4

[14] 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

[15] L. Zhang and R. Koch. Structure and motion from line correspondences: Representation, projection, initialization and sparse bundle adjustment. *Journal of Visual Communication and Image Representation (JVCIR 2014)*, 25(5):904–915, 2014. 9