Supplementary Material for "3D Line Mapping Revisited"

Shaohui Liu[1] Yifan Yu[1] Rémi Pautrat[1] Marc Pollefeys[1,2] Viktor Larsson[3]
[1]Department of Computer Science, ETH Zurich [2]Microsoft [3]Lund University

Full Acknowledgements. We sincerely thank the reviewers for their constructive feedback. We are grateful to Philipp Lindenberger, Paul-Edouard Sarlin, Johannes Schönberger for their open-source projects, Iago Suárez, Hengkai Guo, Daniel Barath, Siyan Dong, Martin Oswald, Jing Ren, Iro Armeni for helpful discussions, Marcel Geppert, Daniel Thul, Peizhuo Li for technical support on Blender. Special thanks to Mihai Dusmanu and Wang Zhao for proof-reading. Viktor Larsson received funding by the strategic research project ELLIIT.

Appendix

This document provides a list of supplementary materials that accompany the main paper. The content is organized as follows:

- In Section A, we introduce the notation, parameterization, and transformations of Plücker coordinates that are used throughout the system, in particular at joint optimization.
- In Section B, we provide detailed derivations for different types of triangulations discussed in the main paper.
- In Section C, we provide details on constructing the association graph among lines and points / vanishing points, and further introduce some examples of its extensions to higher-level applications.
- In Section D, we provide details on computing the covariance used in the main paper for endpoint triangulation and algebraic line triangulation, and also give details on the setup of the corresponding synthetic tests.
- In Section E, we extend the discussion in the main paper to show how our system can be easily extended to map lines with available depth maps, and further present relevant experimental results on mapping and localization.
- In Section F, more implementation details are provided on datasets, hyperparameters, detailed distance measurements and the two baseline methods [20, 52] in the experiments.
- In Section G, we present more experimental results and additional analysis on our 3D line mapping system.
- In Section H, we provide details on our proposed line-assisted visual localization system, along with more results and comparisons against the baseline method [16], including point-line localization results on InLoc dataset [49].
- In Section I, we present more results on refining point-based structure-from-motion with our proposed line mapping system.
- In Section J, we show preliminary results on how to adapt the acquired 3D line maps into the PatchMatch Stereo pipeline [43] to improve the completeness of dense reconstruction.
- In Section K, we present how to extend featuremetric optimization over the acquired line tracks to improve the pixelwise alignment with deep features.
- Finally, in Section L, we expand the conclusions in the main paper and discuss more on the limitations and future work.

A. Background: Plücker Coordinate

Here we revisit how to represent an infinite line with its Plücker coordinate [17]. We first present the definition and its 4 DoF minimal parameterization [5]. Then, we show how to apply geometric operations on top of it.

A.1. Definition

A 3D line segment is compactly encoded with its two 3D endpoints $\boldsymbol{p}_s$ and $\boldsymbol{p}_e$, which exhibit six degrees of freedom. Its corresponding infinite 3D line, however, has only 4 degrees of freedom, as both points can be moving along the line direction. Thus, representing an infinite 3D line with two 3D points is not a compact representation in the sense that two coordinates can be both feasible and correspond to the same infinite 3D line. The Plücker coordinate [5, 18, 30] $(\boldsymbol{d}, \boldsymbol{m})$ is a compact representation for an infinite 3D line,

where $\boldsymbol{d}$ is the normalized direction of the 3D line, and $\boldsymbol{m}$ is the moment that is invariant to any point $\boldsymbol{p}$ along the line

$$\boldsymbol{d} = \frac{\boldsymbol{p}_e - \boldsymbol{p}_s}{\|\boldsymbol{p}_e - \boldsymbol{p}_s\|} \tag{1}$$

$$\boldsymbol{m} = \boldsymbol{p} \times \boldsymbol{d} = \boldsymbol{p}_s \times \boldsymbol{d} = \boldsymbol{p}_e \times \boldsymbol{d}. \tag{2}$$

The property in (2) is due to the fact that $(\boldsymbol{p} - \boldsymbol{p}_s) \times \boldsymbol{d} = \boldsymbol{0}$, where $\boldsymbol{p}$ is any point along the line. The coordinate is convenient in the sense that we can directly perform transformations and projections efficiently on top of it, which will be presented in the following subsections.

A.2. Minimal Parameterization

We first discuss here how to minimally parameterize a Plücker coordinate $(\boldsymbol{d}, \boldsymbol{m})$ in a non-linear optimization, e.g. our joint optimization scheme. The minimal parameterization was initially discussed in [5] as the orthonormal representation. A Plücker coordinate can be minimally represented with:

$$(\boldsymbol{U}, \boldsymbol{W}) \in SO(3) \times SO(2). \tag{3}$$

This results in the minimal $3 + 1 = 4$ degrees of freedom for the infinite 3D line. Specifically, since $\boldsymbol{d}$ is orthogonal to $\boldsymbol{m}$, we can represent the coordinate $(\boldsymbol{d}, \boldsymbol{m})$ with:

$$\boldsymbol{U} = (\boldsymbol{d} \quad \frac{\boldsymbol{m}}{\|\boldsymbol{m}\|} \quad \frac{\boldsymbol{d} \times \boldsymbol{m}}{\|\boldsymbol{d} \times \boldsymbol{m}\|}) \in SO(3), \tag{4}$$

$$\boldsymbol{W} = \begin{pmatrix} w_1 & w_2 \\ -w_2 & w_1 \end{pmatrix} \in SO(2), \tag{5}$$

$$w_1 = \frac{1}{\sqrt{1 + \|\boldsymbol{m}\|^2}}, \quad w_2 = \frac{\|\boldsymbol{m}\|}{\sqrt{1 + \|\boldsymbol{m}\|^2}}. \tag{6}$$

Denote $\boldsymbol{U}$ as $\boldsymbol{U} = (\boldsymbol{u}_1 \ \ \boldsymbol{u}_2 \ \ \boldsymbol{u}_3)$, we can easily recover the original Plücker coordinate $(\boldsymbol{d}, \boldsymbol{m})$ with the minimal parameterization by:

$$\boldsymbol{d} = \boldsymbol{u}_1, \quad \boldsymbol{m} = \frac{w_2}{w_1} \boldsymbol{u_2} \tag{7}$$

At optimization, we can parameterize $\boldsymbol{U} \in SO(3)$ with a quaternion, and $\boldsymbol{V} \in SO(2)$ with a 2-dimensional homogeneous parameterization using Ceres [3].

A.3. Perspective Projection

The Plücker coordinate $(\boldsymbol{d}, \boldsymbol{m})$ can be written in matrix form. The 4×4 Plücker matrix $\boldsymbol{L}$ is formulated as:

$$\boldsymbol{L} = \begin{pmatrix} [\boldsymbol{m}]_\times & \boldsymbol{d} \\ -\boldsymbol{d} & 0 \end{pmatrix}. \tag{8}$$

We here directly provide the clean formulation from [17] for projecting an infinite 3D line with Plücker matrix $\boldsymbol{L}$ perspectively with a 3×4 projection matrix $\boldsymbol{P}$:

$$[\boldsymbol{l}]_\times = \boldsymbol{P}\boldsymbol{L}\boldsymbol{P}^T, \tag{9}$$

where $\boldsymbol{l}$ is the 3-dimensional homogeneous coordinate for the resulting 2D infinite line, where on the 2D image we have $\boldsymbol{l}^T [x, y, 1] = 0$.

A.4. Point-to-Line Projection

Here we discuss how to project a 3D point $\boldsymbol{p}$ onto the infinite 3D line represented with Plücker coordinate $(\boldsymbol{d}, \boldsymbol{m})$. Specifically, we can compute the moment $\boldsymbol{m}_p$ of the line with respect to the 3D point $\boldsymbol{p}$ (rather than the origin):

$$\boldsymbol{m}_p = \boldsymbol{m} + \boldsymbol{d} \times \boldsymbol{p}. \tag{10}$$

Then, the projection of the 3D point $\boldsymbol{p}_\perp$ on the infinite 3D line can be computed as:

$$\boldsymbol{p}_\perp = \boldsymbol{p} + \boldsymbol{d} \times \boldsymbol{m}_p = \boldsymbol{p} + \boldsymbol{d} \times (\boldsymbol{m} + \boldsymbol{d} \times \boldsymbol{p}). \tag{11}$$

We can use this property to efficiently compute the projection of the 3D point without computing squared distances, which gives us robustness to numerical issues at joint optimization with soft point-line associations.

A.5. Line-to-Line Projection

The line-to-line projection aims to find the point on the line (line 1) with Plücker coordinate $(\boldsymbol{d}_1, \boldsymbol{m}_1)$ that is closest to the projected line (line 2) with Plücker coordinate $(\boldsymbol{d}_2, \boldsymbol{m}_2)$. This operation is particularly useful in our system at the following steps:

- **M1. Triangulation with multiple points.** We need to project the infinite 3D line fitted from multiple 3D points onto the camera rays of the two endpoints in the reference image.
- **Endpoint aggregation at joint optimization.** We aim to get a rough estimate of the 3D endpoints on the optimized infinite 3D line, without 3D line proposals. Here we can project the camera rays from the endpoints of each 2D support onto the infinite 3D line.
- **Cheirality test for point-line localization.** We need to test, for each 2D-3D line correspondence, if the 3D line segment has positive depth at the range of the unprojection of its 2D support. Here we can project the camera rays from the endpoints of the 2D supporting line segment onto the infinite 3D line of the 3D line segment to get the ranges where the cheirality test is applied.

Specifically, take $p_{2 \to 1 \perp}$ as the projection of infinite line 2 with $(\boldsymbol{d}_2, \boldsymbol{m}_2)$ onto infinite line 1 with $(\boldsymbol{d}_1, \boldsymbol{m}_1)$, the point can be computed with Plücker coordinate [17] as follows:

$$p_{2\to 1\perp} = \frac{-\boldsymbol{m}_1 \times (\boldsymbol{d}_2 \times (\boldsymbol{d}_1 \times \boldsymbol{d}_2)) + (\boldsymbol{m}_2^T(\boldsymbol{d}_1 \times \boldsymbol{d}_2))\boldsymbol{d}_1}{\|\boldsymbol{d}_1 \times \boldsymbol{d}_2\|^2} \tag{12}$$

The other way can be computed similarly by substitution of variables.

B. Detailed Derivations for Different Types of Triangulations

As discussed in the paper, we propose four types of different triangulation methods to generate the 3D proposals for each 2D line segment, including the straightforward algebraic line triangulation, plus three advanced triangulation methods utilizing commonly associated points and a vanishing point direction. To ensure completeness as well as support the later discussion in Section D, here we will provide detailed derivations for each of the four triangulation methods.

B.1. Algebraic Line Triangulation

We start with the conventional line triangulation with back-projected planes. Because we aim to get the 3D endpoints of the triangulated line segment rather than the infinite line, the algebraic line triangulation is geometrically two ray-plane intersection problems between the camera rays from the endpoints $\boldsymbol{x}_1^r$, $\boldsymbol{x}_2^r$ of the reference line segment and the back-projected plane from the matched line segment spanned by the camera rays of $\boldsymbol{x}_1^m$, $\boldsymbol{x}_2^m$ on the target image. Here $\boldsymbol{x}_1^r$, $\boldsymbol{x}_2^r$, $\boldsymbol{x}_1^m$, $\boldsymbol{x}_2^m$ are all in homogeneous coordinates normalized by the camera intrinsics.

As in the main paper, assume without loss of generality that the world coordinate system aligns with the reference view, while the camera pose of the matched view is $(R, \boldsymbol{t})$. Then the intersection point $\boldsymbol{X}_i = \lambda_i \boldsymbol{x}_i^r$ $(i = 1, 2)$ can be written in the linear combination of the two camera rays of the endpoints from the matched segments:

$$\boldsymbol{X}_i = \lambda_i \boldsymbol{x}_i^r = -R^T\boldsymbol{t} + \beta_1 R^T \boldsymbol{x}_1^m + \beta_2 R^T \boldsymbol{x}_2^m. \tag{13}$$

This results in a 3×3 linear system to solve for $[\lambda_i, \beta_1, \beta_2]$ for $i = 1, 2$, which will be used in the derivation of the covariance in Section D. By multiplying R and adding $\boldsymbol{t}$ in both sides we have:

$$R(\lambda_i \boldsymbol{x}_i^r) + \boldsymbol{t} = \beta_1 \boldsymbol{x}_1^m + \beta_2 \boldsymbol{x}_2^m. \tag{14}$$

Then, by multiplying $(\boldsymbol{x}_1^m \times \boldsymbol{x}_2^m)^T$ in both sides:

$$(\boldsymbol{x}_1^m \times \boldsymbol{x}_2^m)^T (R(\lambda_i \boldsymbol{x}_i^r) + \boldsymbol{t}) = 0, \tag{15}$$

which is similar to Eq. (2) in the main paper. Here the equation can be taken as the point $\boldsymbol{X}_i = \lambda_i \boldsymbol{x}_i$ satisfying the equation of the back-projected plane.

B.2. M1. Triangulation with Multiple Points

This triangulation applies to the case when multiple (≥ 2) common 3D points are available between the reference line segment and the matched one by traversing the 2D point-line association graphs. Here, we can fit an infinite 3D line by computing the mean and principle direction over all the points, and then project the infinite 3D line onto the two camera rays for $\boldsymbol{x}_1^r$ and $\boldsymbol{x}_2^r$ with Plücker coordinates discussed in Section A.5.

B.3. M2. Line + Point: Triangulation with a Known 3D Point

For each commonly shared 3D point between the reference line segment and the matched segment, we can formulate a line triangulation solution using a known 3D point. Compared to algebraic line triangulation, M2 can generate stable endpoints for weakly degenerate cases where one of the two endpoints has degenerate configurations.

As discussed in the paper, we ensure that the endpoints of the generated proposal lie on the camera rays of $\boldsymbol{x}_1^r$ and $\boldsymbol{x}_2^r$. such that $\boldsymbol{X}_i = \lambda_i \boldsymbol{x}_i^r$ for $i = 1, 2$. Therefore, the problem becomes a constrained least square problem with respect to the ray depths of the two endpoints $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)$. The least-square error is the residual from Eq. (2) (or equivalently, (15)), which is quadratic to $\boldsymbol{\lambda}$. We can denote the least square residual as $\boldsymbol{\lambda}^T A \boldsymbol{\lambda} + \boldsymbol{b}^T \boldsymbol{\lambda}$ without loss of generality.

Since both endpoints lie on the camera rays with $\boldsymbol{X}_i = \lambda_i \boldsymbol{x}_i^r$, we can convert the problem into a 2D subproblem by applying a global rotation R^r such that $R^r \boldsymbol{x}_i^r$ has zero value at the third dimension for $i = 1, 2$. Next, we will discuss how to acquire a closed-form solution for $\boldsymbol{\lambda}$ to this 2D subproblem.

B.3.1 Closed-form Solution to the 2D Subproblem

Let $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ be the 2D points (after applying R^r and removing the third dimension). The back-projected 3D endpoints become $\boldsymbol{v}_1 = \lambda_1 \boldsymbol{p}_1$, $\boldsymbol{v}_2 = \lambda_2 \boldsymbol{p}_2$. Take $\boldsymbol{p}_0$ as the 2D projection of the known 3D point on the plane, the constraints for the three points p_0, v_1, v_2 to be collinear is:

$$\det([\boldsymbol{v}_1 - \boldsymbol{p}_0,\ \boldsymbol{v}_2 - \boldsymbol{p}_0]) = 0 \tag{16}$$

This is a quadratic equation in λ_1 and λ_2, and can thus be written as

$$\boldsymbol{\lambda}^T Q \boldsymbol{\lambda} + \boldsymbol{q}^T \boldsymbol{\lambda} = 0 \tag{17}$$

Note that there is no constant term since $\det(\boldsymbol{p}_0, \boldsymbol{p}_0) = 0$. Combining the least square error $\boldsymbol{\lambda}^T A \boldsymbol{\lambda} + \boldsymbol{b}^T \boldsymbol{\lambda}$ and introducing Lagrange multiplier μ we have:

$$\mathcal{L}(\boldsymbol{\lambda}, \mu) = \boldsymbol{\lambda}^T A \boldsymbol{\lambda} + \boldsymbol{b}^T \boldsymbol{\lambda} + \mu(\boldsymbol{\lambda}^T Q \boldsymbol{\lambda} + \boldsymbol{q}^T \boldsymbol{\lambda}). \tag{18}$$

First-order constraints are then

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}} = 2A\boldsymbol{\lambda} + \boldsymbol{b} + 2\mu Q \boldsymbol{\lambda} + \mu \boldsymbol{q} = 0 \tag{19}$$

$$\frac{\partial \mathcal{L}}{\partial \mu} = \boldsymbol{\lambda}^T Q \boldsymbol{\lambda} + \boldsymbol{q}^T \boldsymbol{\lambda} = 0 \tag{20}$$

From (19) we get $\boldsymbol{\lambda}$ as a function of μ,

$$\boldsymbol{\lambda}(\mu) = \frac{-1}{2} (A + \mu Q)^{-1} (\boldsymbol{b} + \mu \boldsymbol{q}), \tag{21}$$

Then inserting into (20) we get

$$p(\mu) = \boldsymbol{\lambda}(\mu)^T Q \boldsymbol{\lambda}(\mu) + \boldsymbol{q}^T \boldsymbol{\lambda}(\mu) \tag{22}$$

which is a rational function in μ. The numerator is a degree quartic polynomial in μ which can be solved in closed form solution to recover μ. Backsubstituting into (21) yields the corresponding ray-depths $\boldsymbol{\lambda}$. We substitute into the cost for each of the (up to four) real solutions and take the one which minimizes the cost to get the final triangulation.

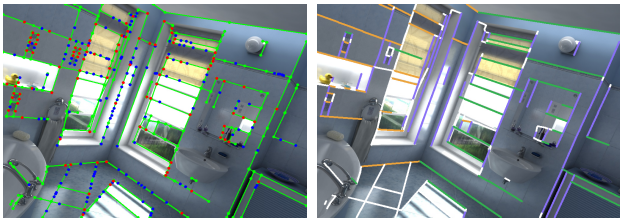
B.4. M3. Line + VP: Triangulation with a Known 3D Direction

One can also generate stable proposals for the weakly degenerate case when a known 3D direction is available, which can come from the vanishing point estimation. This, similarly, can also be converted into a 2D problem since we assume that the 3D endpoints lie on the two camera rays of the two endpoints $\boldsymbol{x_1^r}$ and $\boldsymbol{x_2^r}$. Specifically, take $\boldsymbol{v}$ as the 3D direction from the vanishing point, we assume that the projection of $\boldsymbol{v}$ on the plane spanned by $\boldsymbol{x_1^r}$ and $\boldsymbol{x_2^r}$ is collinear with the vector between the two endpoints, which results in Eq. (4) in the main paper. Since this equation is linear with $\boldsymbol{\lambda}$, the problem becomes a least square problem minimizing $\boldsymbol{\lambda}^T A \boldsymbol{\lambda} + \boldsymbol{b}^T \boldsymbol{\lambda}$ with a linear constraint on $\boldsymbol{\lambda}$. By introducing the Lagrange multiplier we can easily reduce the problem to a quadratic polynomial, which can be solved in closed form.

C. Details on Point-line Association

In this section, we present details on how we build the point-line association graphs initially in 2D and then in 3D, and further show two extensions of the recovered association graphs on generating local plane proposals and identifying the structural layout.

Points and lines are naturally associated in 3D structures. Most salient points lie on top of the lines and the corner points mostly come from the intersection of two or more lines. However, directly discovering point-line relations in 3D is not an ideal choice because the 3D distance is always at an unknown local scale, which is ambiguous to be tested with a predefined threshold and may result in wrong association. The idea of our approach is to first associate lines with points and vanishing points in 2D, and then employ the association graphs in triangulation and joint optimization, the latter of which results in 3D association graphs as a byproduct output.



(a) 2D line-point association (b) 2D line-VP association

Figure 1. **Visualization on the 2D association graphs** among lines and points (a) / vanishing points (VPs) (b). (a) 2D line-point association. We only show points that are associated with at least one line. The points with degree 1 are colored blue, while the points with degree ≥ 2 are colored red. (b) 2D line-VP association. The lines that are associated with the same VP are colored the same.

C.1. More Details on 2D Association

First of all, we aim to recover two association graphs for each image in 2D: a line-point association graph and a line-VP association graph, each of which is a bipartite graph. By recovering the relations we can traverse the graph on each image switching back and forth between lines and points / VPs by walking along the connected edges. This can be useful for the following steps in our pipeline:

- **Point-guided line triangulation**, where we can use the neighboring points and vanishing points to generate additional constraints.
- **Construction of 3D vanishing point tracks**, where we can associate vanishing points from different images using 2D-3D track associations in the line maps.
- **Joint optimization with soft association.** We can traverse the 2D association graph to measure how likely a line track is associated with a point / VP track in 3D by counting the 2D edges among their supports.

The advantages of these relational graphs are not limited to the aforementioned examples. Acting as fundamental geometric information for the sparse features, it can be beneficial to most sparse feature-based modules and applications with careful algorithmic designs. In the following parts we present details on how we build the 2D association graphs.

2D Line-Point Association. The line-point association graph is built over the 2D point and line features. We employ SuperPoint [13] as the point feature extractor as it aims to detect corners on the image. For each pair of a 2D point and a 2D line segment, we measure the distance between them (i.e. the distance between the 2D point and the nearest point on the 2D line segment) and add an edge in the bipartite graph if the distance is less than a pixel threshold, which in our case is set to 2 pixels. In theory, the threshold should

depend on the uncertainty of the line detector. An example illustration of the resulting line-point association graph is shown in Fig. 1(a).

2D Line-VP Association. The line-VP association graph is naturally built from the 2D vanishing point estimation. In this special bipartite graph, the line has at most one degree, while the vanishing point has at least 5 degrees to be considered valid. In our system, for vanishing point detection we use JLinkage [50], which aims to detect vanishing points for general parallel lines. One can also employ orthogonal vanishing point detectors [6] from which the orthogonality constraints can be acquired from 2D, yet it will lose the parallelism information for lines that are not aligned with the three main orthogonal axes. An example illustration of 2D vanishing point estimation and the resulting association is shown in Fig. 1(b).

C.2. More Details on 3D Association

C.2.1 Constructing 3D Vanishing Point Tracks

We can make use of the 2D association graph to build 3D vanishing point tracks from the recovered line tracks by transitively propagating the line correspondences.

Specifically, considering the graph with all detected vanishing points from each image as nodes, we aim to associate them together into a set of 3D vanishing point tracks. We connect two nodes from different images if:

- they share at least three common neighboring line tracks on its corresponding 2D line-VP bipartite.
- the angle between their vanishing point directions in the global frame is less than 10 degrees.

Once two nodes are connected, we also assign the weight of the edge to be the number of the common neighboring line tracks. Then, we sort the edges with respect to their weights in descending order and apply Kruskal-like VP track construction with the exclusion that each image only contributes one VP in a track, similar to the existing practice on constructing point tracks [14].

C.2.2 Joint Optimization

As discussed in the paper, the joint optimization consists of the energy terms E_P, E_L, and E_{PL}. We optimize the 3D lines, points, and vanishing points jointly. Specifically, each 3D line is converted into an infinite line and parameterized with Plücker coordinate (4 DoF) as discussed in Section A, and each 3D vanishing point is parameterized with a 3-dimensional homogeneous vector (2 DoF). The variables of the final problem exhibit $3N_P + 4N_L + 2N_{VP}$ degrees of freedom in total, where N_P, N_L, and N_{VP} are the number of 3D points, lines, and vanishing points, respectively. In the following, we will discuss the three energy terms in detail.

E_P. Data term for the point tracks. This term is defined as the squared reprojection error for each point track, which is exactly the same as in the regular bundle adjustment in COLMAP [42].

E_L. Data term for the line tracks. This term is defined as the reprojection error for each line track, termed geometric refinement in the main paper. For the line-only solutions in the experiments, we only employ this term in the final optimization solely over 3D line tracks. The residual is formulated as:

$$E_L(l) = \sum_k w_\angle^2(L_k, \ell_k) \cdot e_{\text{perp}}^2(L_k, \ell_k), \tag{23}$$

$$w_\angle(L_k, \ell_k) = \exp(\alpha(1 - \cos(\angle(L_k, \ell_k)))), \tag{24}$$

where α equals 10.0 in our system. This weighting term $w_\angle$ follows the design of L3D++ [20], which empirically promotes fast convergence by putting the emphasis to make the 2D direction consistent with the observation. Note that compared to $1-\cos(\angle(L_k, \ell_k))$ in our formulation, the residual in L3D++ [20] is directly built on the angle $\angle(L_k, \ell_k)$, which exhibits singularity on the gradient at zero angles and sometimes results in unstable optimization.

E_{PL}. 3D Association Term. This term encourages 3D association among lines and points / vanishing points. Specifically, it consists of three parts: line-point association, line-VP association, and VP orthogonality regularization.

- **3D Line-Point Association.** As discussed in the main paper, the association term is built between each pair of point track and line track that has at least three connected edges (on the 2D line-point association graphs from the corresponding images) among their 2D supports. Each residual is defined as the 3D point-line distance weighted by the number of 2D connections among supports, which can be efficiently computed with Plücker coordinate as discussed in Section A.4.
- **3D Line-VP Association.** As in the point case, the association term is built between each pair of VP tracks (built as in Section C.2.1) and line track that has at least three connected edges (on the 2D line-point association graphs from the corresponding images) among their 2D supports. Each residual is defined as the sine of the direction angle between the line and the vanishing point, again weighted by the 2D connections among supports.

- **VP Orthogonality Regularization.** Since we do not employ orthogonal vanishing point detection [6] to ensure the generality of the system, we do not have any orthogonal information from 2D. However, at joint optimization, we can enforce the nearly orthogonal pairs (in practice, when the angle is larger than 87 degrees) of vanishing points to be orthogonal. So we add a regularization residual to these pairs, defined as the cosine of the angle difference between the directions of the vanishing point pair.

Note that in the joint optimization we do not have data term for the 3D vanishing point. This means that we only employ the 2D line-VP association graphs and enforce parallelism with soft association, without relying on the actual vanishing point detection on 2D which can be sometimes noisy. In this way, we only enforce the lines that are associated with the same VP to become parallel. Both the line-point and line-VP association residuals are equipped with Huber loss function from Ceres Solver [3] to ensure robustness to outlier edges.

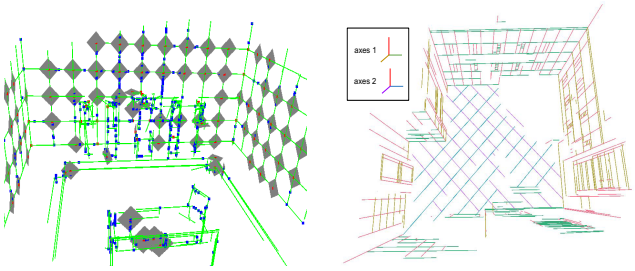
From the joint optimization, we can directly get the 3D association graphs as a byproduct output, by testing the validity of the active line-point / line-VP edges in the soft association problem. The validity check measures the 3D point-line distance and the 3D direction angle ($\leq$ 5 degrees) respectively. For the validity check of the 3D point-line distance, we keep a fixed threshold of 2.0 and re-scale the distance with the minimum uncertainty (defined as the depth divided by the focal length) between the measured 3D point and 3D line to ensure scale invariance. For the output 3D point-line association graph this step removes the outlier edges that are filtered out in the soft association problem at joint optimization. Note that both the resulting 3D association graphs are again bipartite graphs, among lines and points / vanishing points.

C.3. Extension: Generating Local Plane Proposals

As a byproduct output of the system, the 3D line-point association graph can be easily extended to benefit high-level problems. We show one most straightforward extension on generating local plane proposals in Fig. 2(a). From each degree-2 3D point in the graph, we can compute the plane normal by applying the cross product on its two neighboring 3D lines. From the resulting local planes, one can easily group the plane structures and further recover the scene layout. These planes are also potentially beneficial for visual localization pipelines.

C.4. Extension: Atlanta World

From the recovered orthogonality relationship we can easily parse the high-level structure relations in the output 3D maps. Figure 2(b) shows an example: From the six recovered orthogonality constraints we can get two groups of orthogonal axes with a shared vertical direction, resulting in an Atlanta World as discussed in [41].



(a) Local plane proposals (b) Atlanta World

Figure 2. **Illustration of the extensions on the recovered 3D association graphs.** (a) Local plane proposals can be directly acquired from the degree-2 junctions. (b) We show the recovered orthogonal axes from joint optimization at the top left of the figure, where two different Manhattan axes [11] are discovered, resulting in an Atlanta World [41].

D. Covariance Derivation and Setup for Synthetic Tests

In this section, we will provide detailed covariance derivations for endpoint triangulation and algebraic line triangulation respectively, which are used to compute the uncertainty (the largest eigenvalue of the covariance) to study the degeneracy problem in the main paper. We also provide details of the setup of the synthetic tests that are discussed in Figure 8 in the main paper.

In this section, we follow the convention of Section B, where the world coordinate is aligned with the reference frame without loss of generality, and the camera pose of the matched image is $(R, \boldsymbol{t})$. The intrinsic-normalized homogeneous coordinates for the endpoints of the reference segment are $\boldsymbol{x}_1^r$ and $\boldsymbol{x}_2^r$, and $\boldsymbol{x}_1^m$ and $\boldsymbol{x}_2^m$ for the endpoints of the matched segment.

D.1. Background: Covariance Propagation

Denote the 2D endpoints correspond to $\boldsymbol{x}_1^r$, $\boldsymbol{x}_2^r$, $\boldsymbol{x}_1^m$, and $\boldsymbol{x}_2^m$ as $\boldsymbol{p}_1^r$, $\boldsymbol{p}_2^r$, $\boldsymbol{p}_1^m$, and $\boldsymbol{p}_2^m$ respectively. The input for the triangulation problem is an 8-dimensional vector $\boldsymbol{p}_{2D}$, while the output for the triangulation problem is a 6-dimension vector $\boldsymbol{p}_{3D}$, where

$$\boldsymbol{p}_{2D} = \begin{pmatrix} \boldsymbol{p}_1^r \\ \boldsymbol{p}_2^r \\ \boldsymbol{p}_1^m \\ \boldsymbol{p}_2^m \end{pmatrix}, \quad \boldsymbol{p}_{3D} = \begin{pmatrix} \boldsymbol{X}_1^r \\ \boldsymbol{X}_2^r \end{pmatrix}. \tag{25}$$

We can assume unit covariance on each endpoint of the line detection and independence across different endpoints. Then the 8×8 covariance matrix of $\boldsymbol{p}_{2D}$ can be written as:

$$\mathbf{\Sigma}_{2D} = \begin{pmatrix} I_2 & 0 & 0 & 0 \\ 0 & I_2 & 0 & 0 \\ 0 & 0 & I_2 & 0 \\ 0 & 0 & 0 & I_2 \end{pmatrix} = I_8 \tag{26}$$

Take f_e and f_l be the functions that maps $\boldsymbol{p}_{2D}$ to $\boldsymbol{p}_{3D}$ with endpoint triangulation and algebraic line triangulation respectively, and J_e, J_l be their corresponding 6×8 Jacobian matrices.

With the rule of covariance propagation, we can compute the covariance Σ^e_{3D} and Σ^l_{3D} for endpoint triangulation and algebraic line triangulation by

$$\Sigma^e_{3D} = J_e \Sigma_{2D} J_e^T, \;\; \Sigma^l_{3D} = J_l \Sigma_{2D} J_l^T, \;\; . \tag{27}$$

The problem of measuring the covariance of the triangulated 3D line segment reduces to the computation of the corresponding Jacobian matrices J_e and J_l.

We first provide here the Jacobian J_d of a normalized direction vector $\boldsymbol{d}$ with respect to $\boldsymbol{x}$, where $\boldsymbol{d} = \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}$:

$$J_d = \frac{I - \boldsymbol{d}\boldsymbol{d}^{\boldsymbol{T}}}{\|\boldsymbol{x}\|}. \tag{28}$$

For ease of notation in the following sections, we use $\boldsymbol{d}_i^r$ and $\boldsymbol{d}_i^m$ $(i = 1, 2)$ to denote the normalized ray directions correspond to $\boldsymbol{x}_i^r$ and $R^T \boldsymbol{x}_i^m$ respectively, and $\boldsymbol{C}_r = \mathbf{0}$, $\boldsymbol{C}_m = -R^T \boldsymbol{t}$ be the corresponding camera center.

D.2. Endpoint Triangulation

We employ the mid-point triangulation [17] for computing each of the 3D endpoints respectively. The formulation of midpoint triangulation can be written as follows:

$$\begin{pmatrix} 1 & -\boldsymbol{d}_i^{rT} \boldsymbol{d}_i^m \\ -\boldsymbol{d}_i^{rT} \boldsymbol{d}_i^m & 1 \end{pmatrix} \begin{pmatrix} \lambda_i^r \\ \lambda_i^m \end{pmatrix} = \begin{pmatrix} \boldsymbol{d}_i^{rT} (C_m - C_r) \\ -\boldsymbol{d}_i^{mT} (C_m - C_r) \end{pmatrix}, \tag{29}$$

$$\boldsymbol{X}_i = \frac{1}{2} (\boldsymbol{C}_r + \lambda_i^r \boldsymbol{d}_i^r + \boldsymbol{C}_m + \lambda_i^m \boldsymbol{d}_i^m), \tag{30}$$

where λ_i^r and λ_i^m are the ray depths of the rays $\boldsymbol{d}_i^r$ and $\boldsymbol{d}_i^m$ respectively $(i = 1, 2)$.

By using the property on the derivative of matrix inverse:

$$(K^{-1})' = -K^{-1} K' K^{-1}, \tag{31}$$

we can compute the derivative of $\boldsymbol{X}_i$ with respect to the direction vectors $\boldsymbol{d}_i^r$ and $\boldsymbol{d}_i^m$. Combining (28) the final 6×8 Jacobian J_e can be computed with the chain rule.

D.3. Algebraic Line Triangulation

Similar to the endpoint triangulation, we can also formulate a linear system with respect to the direction vectors for algebraic line triangulation. Specifically, we can rewrite (14) into the following matrix form over $\boldsymbol{d}_i^r$, $\boldsymbol{d}_1^m$ and $\boldsymbol{d}_2^m$:

$$\begin{pmatrix} \boldsymbol{d}_i^r & -\boldsymbol{d}_1^m & -\boldsymbol{d}_2^m \end{pmatrix} \begin{pmatrix} \lambda_i^r \\ \beta_1^m \\ \beta_2^m \end{pmatrix} = \boldsymbol{C}_m - \boldsymbol{C}_r, \tag{32}$$

$$\boldsymbol{X}_i = \boldsymbol{C}_r + \lambda_i^r \boldsymbol{d}_i^r, \;\; i = 1, 2 \tag{33}$$

Again by using (28) and (31) the final 6×8 Jacobian J_l can be computed accordingly.

D.4. Setup for Synthetic Tests

Based on the derived covariance forms, we present study on the degeneracy problem of line triangulation in the main paper (Figure 8). We here discuss the detailed setup for the two experiments.

Uncertainty Visualization on AdelaideRMF [53]. We take an image pair from AdelaideRMF [53] and compute its two-view geometry with COLMAP [42]. Then, we manually annotate 42 line pairs that are perfectly matched. On top of the annotated line matches we perform algebraic line triangulation and compute the uncertainty as the largest eigenvalue of the covariance matrix. We also visualize the epipolar lines on the target image to better illustrate the relation to the degeneracy problem. When visualized in 3D, the lines with low uncertainty are reasonably accurate while the lines that are degenerate (with high covariance) locate "everywhere" in the 3D space. This also shows that the largest eigenvalue of the covariance matrix can be a good indicator of the reliability of the triangulation.

Synthetic Tests. We design a synthetic test to further study the stability of the line triangulation. Specifically, we first set up a horizontal plane ($z = 10$) and two stereo cameras that point orthogonal to the plane with a distance of 10.0. The baseline (lying along the x direction) of the stereo pair is 4.0 (so the two cameras locate at $(-2, 0, 0)$ and $(2, 0, 0)$) and the focal lengths of both cameras are 700. Under this setup, the epipolar lines are always horizontally aligned with the stereo baseline. We sample random 3D lines with a fixed direction on the horizontal plane ($z = 10$) within a range of $[-1, 1]$ on both x and y directions, and project them onto the two views, resulting in perfect 2D line matches with a fixed angle with the epipolar lines. Then, we perform endpoint triangulation and algebraic line triangulation respectively, and compute its covariance as discussed in Section D.2 and D.3. We measure the uncertainty as the largest eigenvalue of

Line type	Method	R1	R5	R10	P1	P5	P10	# supports
LSD [51]	Ours (line-only)	48.3	187.0	257.4	59.2	81.9	89.8	(15.8 / 19.1)
	Ours w/ depth	**89.7**	**315.3**	**330.8**	**63.0**	**99.7**	**100**	(**16.6** / **23.3**)
SOLD2 [31]	Ours (line-only)	50.8	143.5	180.8	74.4	86.9	91.2	(15.1 / 32.2)
	Ours w/ depth	**84.4**	**252.0**	**278.2**	**79.7**	**99.7**	**99.9**	(**16.0** / **38.4**)

Table 1. **Quantitative results of line mapping given depth maps** on Hypersim [34]. $R\tau$ and $P\tau$ are reported at 1mm, 5mm, 10 mm along with the average number of supporting images/lines.

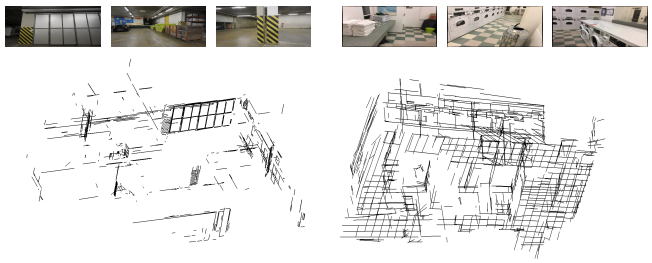


Figure 3. **Line mapping given depth maps. Left:** DSLR sequence of *delivery_area* (44 images) from ETH3D [44, 45] with LSD [51] and LiDAR scanner depth. **Right:** Laundry (*scene_0678_01*, 465 images at 6 FPS) from ScanNet [12] with SOLD2 [31] and depth from an RGB-D sensor.

the covariance matrix. The median uncertainty is computed for 10000 random lines for each tested angle.

E. Line Reconstruction given Depth Maps

E.1. System Details

As discussed in the paper, when depth maps are available (e.g. from an RGB-D sensor), we can apply a robust fitting to the back-projected 3D points to generate an accurate proposal, which can serve as the best candidate for the 2D line segment in the track building step.

Specifically, we sample points along the 2D line and collect a set of 3D points by back-projecting the points using the depth maps. Then, we apply 3D line fitting with LO-RANSAC [10, 28]. To ensure invariance to the scale changes, the inlier threshold is proportional to the median depth of all the points divided by the focal length, which shares similar spirits with the scale factor σ used in the *InnerSeg distance*. By associating those fitted 3D line segments with the same track-building strategy, we can acquire high-quality line tracks that align geometrically with the 3D depth maps.

E.2. Results on Line Mapping

We show quantitative results on line mapping given depth maps on the first eight scenes of Hypersim [34] in Table 1. As our solution on line mapping given depth maps does not employ points and vanishing points, we show the comparison to our triangulation with only line-line proposals. While

Scene	HLoc [35] w/ Depth	PtLine [16]	Ours w/ Depth
Chess	**2.4** / **0.81** / 94.8	**2.4** / **0.81** / **95.0**	**2.4** / 0.82 / 94.0
Fire	1.9 / 0.76 / 96.4	1.9 / 0.76 / **96.6**	**1.7** / **0.71** / **96.6**
Heads	1.1 / 0.73 / 99.0	1.1 / 0.74 / **99.4**	**1.0** / **0.72** / **99.4**
Office	2.7 / 0.83 / 83.7	2.7 / 0.83 / 83.9	**2.6** / **0.80** / **84.7**
Pumpkin	4.1 / **1.05** / **61.3**	4.0 / 1.06 / 60.8	**4.0** / **1.05** / 61.1
Redkitchen	3.3 / **1.12** / 72.1	**3.2** / **1.12** / 72.5	3.3 / **1.12** / **73.0**
Stairs	4.7 / 1.25 / 53.4	4.3 / 1.16 / 55.9	**3.2** / **0.86** / **76.0**
Avg.	2.9 / 0.94 / 80.1	2.8 / 0.93 / 80.6	**2.6** / **0.87** / **83.5**

Table 2. **Visual localization on 7Scenes [46] with depth maps [8].** We report the median translation and rotation error in cm and degrees, as well as the pose accuracy at a 5 cm / 5 deg threshold.

still far from perfect, the resulting 3D line maps are significantly better compared to the ones built without depth maps. Nearly all the recovered 3D lines are within 10 millimeters of the ground truth mesh model. The track supports are also significantly richer in comparison. Nevertheless, it is worth noting that our line maps built with the assistance of points and vanishing points can achieve a comparable number of supports with SOLD2 line detector [31]. This again demonstrates the advantages of point-guided line triangulation in being able to generate reasonable proposals on degenerate cases, which benefits the track-building process.

We further show qualitative results of line mapping with depth maps in Figure 3, on ETH3D [45] and ScanNet [12] respectively. The mapping solution can produce perceivable 3D line structures with either the LiDAR scanner depth from ETH3D [45] or the RGB-D sensor from ScanNet [12], with both conventional LSD detector [51] and the recent learning-based one [31].

E.3. Results on Line-Assisted Visual Localization

As in the RGB case, we show here that line mapping with depth maps is also able to help visual localization by combining points and lines. We run our solution of line mapping with depth maps on 7Scenes [46] with depth maps from [8]. Then, we run our proposed point-line visual localization system with hybrid RANSAC [9] as discussed in the paper (detailed in Sec. H). Results are shown in Table 2. Integrating line maps into visual localization improves the localization accuracy, in particular contributing to a large performance gain (53.4 $\rightarrow$ 76.0 on 5 cm / 5 deg) on the most challenging scene: *Stairs*. This again demonstrates the usefulness of the acquired 3D line maps.

F. More Implementation Details

F.1. Datasets

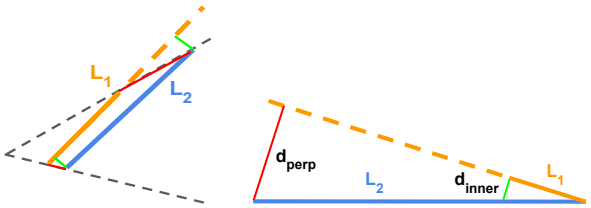
We test our method quantitatively on Hypersim [34] and *Tanks and Temples* [26]. For the qualitative results across datasets, we rely on two line detectors: SOLD2 and LSD [51]. For SOLD2 [31] detection, description, and matching, we employ the default parameters provided in

their released code repository. As SOLD2 [31] is originally focused on indoor and manmade structured environments, we test SOLD2 detection only on indoor datasets: Hypersim [34] and ScanNet [12] (Figure 3 in supp.), while for the other datasets LSD [51] is used to run our line mapping. For all the datasets, we undistort images with the calibration (either provided or estimated with COLMAP [42]) before performing line detection, which mitigates the issue of straight lines appearing curved due to radial distortion.

Hypersim [34] is a photorealistic synthetic dataset for holistic indoor scene understanding. For evaluation, we use the first 8 scenes and resize the image to a maximum dimension of 800 as input to all the tested methods. The average metrics over all the 8 scenes are reported. For neighborhood computation, we use the point triangulator from COLMAP [42] with SuperPoint [13] from HLoc [35] to build the 3D model from images with known camera poses, and rank neighboring images by the Dice coefficient on the common 3D points. The reconstructed 3D lines are evaluated with respect to the provided ground truth mesh model. To efficiently compute the distance between a query point sampled from the line and the mesh, we use the AABB hierarchy from [22] as the data structure. Specifically, we first build the hierarchy from the mesh model and sample points densely and uniformly for each line to compute the overall length recall $R\tau$ and the inlier percentage $P\tau$.

Tanks and Temples [26] is a benchmark for image-based 3D reconstruction and is widely used for evaluating multi-view stereo and novel view synthesis [33]. They provide dense ground truth point cloud from a FARO Focus 3D X330 HDR scanner. We input the images with their original resolution (around 2 Megapixels) for both our method and L3D++ [20]. The provided point cloud is cleaned such that it only contains the main subject in the middle of the scene. Since our method can reconstruct lines that are far away from the model (see Figure 5 and Figure 5 *Barn* of the main paper), we compute the axis-aligned bounding box for each point cloud and stretch it one meter in all three dimensions. At evaluation, only lines within the stretched bounding box are considered. We evaluate on the *train* split and take the average over all the scenes except for *Ignatius* which has no observable line structures. For efficient computation of the distance between the sampled point and the ground truth point cloud, we build a KD-Tree over the ground truth point cloud with nanoflann [7].

To further demonstrate the effectiveness and generalization of our system, we also present qualitative results on unstructured image collections on Aachen v1.1 Day-Night dataset [40] and Rome city from BigSFM [2, 47, 48]. In the supplementary material, we also present additional results of our line mapping on Cambridge [24], PhotoTourism [23, 47], and also line mapping given depth maps on ETH3D [44, 45] and ScanNet [12]. The oracle test in Figure 8 of the main paper is conducted on AdelaideRMF dataset [53]. The localization experiments are run on Cambridge [25], 7Scenes [8, 46], and InLoc [49] (in this supplementary material).



(a) Perspective distance (b) InnerSeg distance

Figure 4. **Benefits of our scoring methods.** (a) Ill-posed line triangulations may have a low perpendicular score (in green), while our perspective distance (in red) can filter them out. (b) The perpendicular distance (in red) is detrimental for very long segments, while our proposed innerseg distance (in green) remains unbiased.

F.2. Hyperparameters

Similar to all existing point-based solutions such as COLMAP [42], our library also has a number of hyperparameters that can be changed in each module while using our default ones at release should work on most in-the-wild cases due to our scale-invariant design. We keep the hyperparameters unchanged throughout all experiments across datasets.

The scaling factors introduced at scoring and track building are set as follows: $\tau_a = 10$ degrees for the angle in 3D and $\tau_a = 8$ degrees in 2D, $\tau_o = 0.05$ for the overlap in 2D and 3D, $\tau_p = 5$ pixels for the perpendicular distance in 2D, and $\tau_s = 0.015$ for the scale-invariant endpoint distance in 3D. The threshold for 2D point-line association is set to 2 pixels, and the inlier threshold for 2D VP-line association with JLinkage [50] is set to 1 pixel. The detected vanishing points with at least 5 inliers (associated lines) are kept on each image. The minimum triangulation angle between the camera ray and the plane spanned by camera rays $\boldsymbol{x}_1^m$ and $\boldsymbol{x}_2^m$ is 1 degree.

F.3. Additional Discussions on Distance Measurements

Benefits of *Perspective Distance*. The perspective distance was originally proposed to filter out ill-posed line triangulations that are almost colinear with the ray endpoints of the corresponding 2D segment. In Figure 4, we show on the left that such ill-posed triangulations may still have a small perpendicular error (in green), and thus, cannot be filtered out with such a classic distance. On the contrary, our proposed perspective scoring (proportional to the endpoint distances in red) will penalize such bad triangulations.

Benefits of *Innerseg Distance*. Another drawback of the perpendicular distance is that it penalizes long segments, as visualized on the right of Figure 4 in red. These long lines

are however quite important to get clean reconstructions. Our proposed InnerSeg distance (in green) can effectively avoid this negative bias.

Scale Factor σ. The uncertainty of the line segment depends on its depth with respect to the two views from which it is triangulated. To make our triangulation invariant to scale changes, we define the scale factor σ as the depth of the midpoint divided by the focal length. This essentially encodes how far the midpoint moves in 3D before reaching 1 pixel error on the image. When testing if the *InnerSeg Distance* is within a certain threshold, we rescale the distance with the minimum scale factors of the two, which results in the scale factor σ defined in the paper.

F.4. More Details on the System Design

Weak Epipolar Constraint. We also employ weak epipolar constraints for filtering out matches for line triangulation following [20]. We measure the IoU between the matched segment and the intersected segments from two epipolar lines from the reference endpoints, and filter out matches if the IoU is below 0.1. For a fair comparison, we also update this hyperparameter in L3D++ [20] to 0.1, as it empirically gives better performance than its default parameter 0.25 (see Tables 3 and 4).

Endpoint Aggregation. As discussed in the main paper, after associating the best candidates from 2D line segments into the 3D track, we take the mean and the principle directions over all the 3D endpoints (of the candidates) in the track to get an initial estimate of the infinite 3D line. As we eventually aim to get 3D line segments, we need to compute the 3D endpoints. This is done by projecting all the endpoints from the candidates in the track onto the estimated infinite 3D line. In practice, we take the third outermost endpoints on both sides to give better robustness to unstable triangulations in the track. This robust selection of the third outmost endpoint is also done after joint optimization, when the optimized infinite 3D line with Plücker coordinate is converted into a 3D line segment.

Track Remerging. Optionally, we also support remerging similar tracks together after track building, as some tracks may have very close 3D lines. Specifically, we can recompute the pairwise scores among the re-fitted 3D lines of each track, and greedily merge tracks with a stricter threshold.

F.5. Details on L3D++ [20] and ELSR [52]

For L3D++ [20], we use the open-sourced implementation from their official repository [19]. For SOLD2 detector [31], we detect line segments in advance and save the segments into a compatible format that can be processed from L3D++ [19]. We update two of their hyperparameters: visual neighbors from 10 to 20, and IoU threshold for the weak epipolar constraint from 0.25 to 0.10, to enable fair comparison, while we also present their results with the default hyperparameters in Tables 3 and 4.

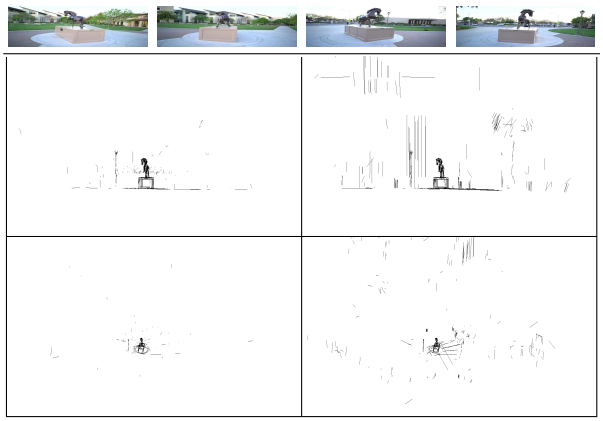


Figure 5. **Illustration on the advantages of our scale-invariant design. Left:** L3D++ [20]. **Right:** Ours. Both systems are run on *Horse* from *Tanks and Temples* [26]. We show two different views on the full scene where our method can reconstruct lines that are far away from the main subject. Refer back to Figure 4 in the main paper for zoom-in comparison on the horse.

For ELSR [52], we use the official release from the authors on their website. Since they only support VisualSfM input [54] with LSD detector [51], we convert the COLMAP model triangulated on Hypersim [34] and provided from *Tanks and Temples* [26] into the VisualSfM format [54] such that it is compatible with their implementation.

G. More Results on Line Mapping

G.1. Scale invariance

Our pipeline is robust to scale changes. This not only refers to the global scale of the scene, but also refers to the local scale of the sub-components of the whole scene: it is very common that one gets different layers of subjects and buildings with large depth changes.

To demonstrate the benefits of our scale-invariant design, we here show a visualization of how the scale-invariant design can help to reconstruct lines across very different scales in Figure 5. Specifically, We compare the reconstructions of *Horse* from *Tanks and Temples* [26] given by L3D++ [20] and our method. Our method is able to reconstruct many more far-away lines in the background compared to L3D++ [20], while providing a very accurate reconstruction for close-by details as well (see Figure 4 of the main paper).

G.2. More Comparisons with L3D++ [20]

To further highlight the advantage of our proposed line mapping over L3D++ [20], we study the recall-precision trade-off by relaxing the requirements for minimum number of supporting images in the final output 3D line tracks

Line type	Method	R1	R5	R10	P1	P5	P10	# supports
LSD [51]	(nv = 4) L3D++ default param. [20]	34.6	139.9	196.6	53.3	**82.6**	**92.6**	(11.6 / 12.5)
	(nv = 4) L3D++ [20]	37.0	153.1	218.8	53.1	80.8	90.6	(14.8 / 16.8)
	(nv = 4) ELSR [52]	13.9	59.7	96.5	55.4	72.6	82.2	(N/A / N/A)
	(nv = 4) Ours	**48.6**	**185.2**	**251.3**	**60.1**	82.4	90.0	(**16.4 / 20.5**)
	(nv = 3) L3D++ default param. [20]	40.9	166.2	235.8	49.3	76.7	86.9	(8.7 / 9.4)
	(nv = 3) L3D++ [20]	40.6	168.8	242.8	50.3	77.3	**87.6**	(12.1 / 13.6)
	(nv = 3) ELSR [52]	13.9	59.7	96.5	55.4	72.6	82.2	(N/A / N/A)
	(nv = 3) Ours	**51.9**	**198.1**	**271.0**	**56.7**	**78.2**	86.3	(**13.6 / 16.8**)
SOLD2 [31]	(nv = 4) L3D++ default param. [20]	29.7	84.7	102.3	67.2	**88.5**	**96.0**	(9.9 / 12.4)
	(nv = 4) L3D++ [20]	36.9	107.5	132.8	67.2	86.8	93.2	(13.2 / 20.4)
	(nv = 4) Ours	**54.3**	**151.1**	**191.2**	**69.8**	84.6	90.0	(**16.5 / 38.7**)
	(nv = 3) L3D++ default param. [20]	34.9	102.3	127.1	61.3	**82.2**	**90.4**	(7.4 / 9.2)
	(nv = 3) L3D++ [20]	40.3	118.7	148.0	62.3	82.0	89.6	(10.6 / 16.0)
	(nv = 3) Ours	**55.6**	**155.4**	**197.4**	**66.8**	82.0	88.0	(**14.1 / 32.5**)

Table 3. More results on Hypersim [34] with minimum 3 supporting images. We also add the results of L3D++ [20] with its default hyperparameters. "nv" denotes the minimum number of supporting images for a line track to be included in the final output. Note that even compared to L3D++ with nv = 3, our mapping with nv = 4 achieves significantly better recall while being better on precision at all thresholds as well. This demonstrates the superiority of our method over L3D++ [20] when moving along the recall-precision trade-off curve.

Method	R5	R10	R50	P5	P10	P50	# supports
(nv = 4) L3D++ default param. [20]	215.4	477.7	1543.6	41.3	55.8	**87.2**	(6.4 / 6.5)
(nv = 4) L3D++ [20]	373.7	831.6	2783.6	40.6	54.5	85.9	(8.8 / 9.3)
(nv = 4) ELSR [52]	139.2	322.5	1308.0	38.5	48.0	74.5	(N/A / N/A)
(nv = 4) Ours (line-only)	472.1	1058.8	3720.7	**46.8**	**58.4**	86.1	(10.3 / 11.8)
(nv = 4) Ours	**508.3**	**1154.5**	**4179.5**	46.0	56.9	83.7	(**10.4 / 12.0**)
(nv = 3) L3D++ default param. [20]	313.1	698.3	2351.6	32.2	44.0	72.5	(4.7 / 4.8)
(nv = 3) L3D++ [20]	473.7	1058.1	3622.4	35.6	48.5	79.6	(6.6 / 7.0)
(nv = 3) ELSR [52]	139.2	322.5	1308.0	38.5	48.0	74.5	(N/A / N/A)
(nv = 3) Ours (line-only)	564.8	1267.2	4539.0	**43.2**	**54.5**	**83.7**	(7.8 / 8.9)
(nv = 3) Ours	**606.7**	**1379.5**	**5047.1**	42.1	52.8	80.9	(**7.9 / 9.0**)

Table 4. More results on Tanks and Temples [26] with minimum 3 supporting images. We also add the results of L3D++ [20] with its default hyperparameters. "nv" denotes the minimum number of supporting images for a line track to be included in the final output.

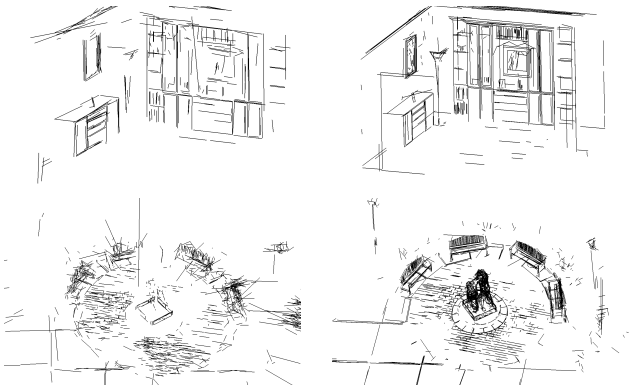


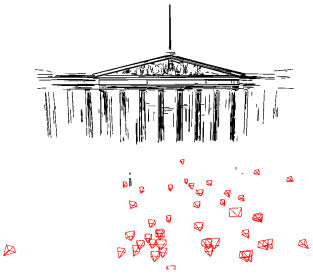
Figure 6. **Qualitative comparisons with ELSR [52].** We show two examples from Hypersim [34] and Tanks and Temples [26].

from 4 views to 3. This is actually the default setting for L3D++ [20] but we updated it in our main experiments for a fair comparison. We also compare with the default hyperparameters used in L3D++ [20] using 10 visual neighbors and 0.25 IoU threshold for the weak epipolar constraints.

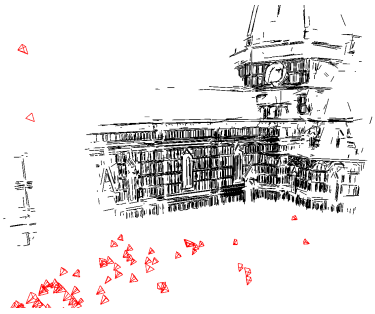
Tables 3 and 4 show the results on Hypersim [34] and *Tanks and Temples* [26] respectively. The relative positions of L3D++ [20] and our method are similar when we relax the required minimum supporting images to 3 views. Here it is worth mentioning that, when comparing our method with nv = 4 against L3D++ [20] with nv = 3 (the default in L3D++ release), we can see that our method is significantly better in both the length recall and precision on all thresholds. This further demonstrates our advantages over L3D++ [20] on the precision-recall curve. The performance gain becomes larger when comparing our line mapping with L3D++ [20] using its default parameters. Since ELSR [52] does not provide 2D-3D track association, we cannot filter their output lines with a minimum number of supporting views.

G.3. Qualitative Results of ELSR [52]

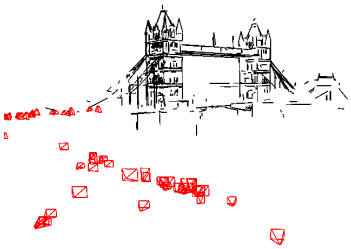
For completeness, we compare qualitatively our mapping results with those from ELSR [52] in Figure 6. While ELSR [52] is able to produce reasonable 3D line maps, it often fails to recover lines where the point and plane features are limited. On the contrary, our method recovers significantly more complete structures with better accuracy, and provides



British Museum from [47]

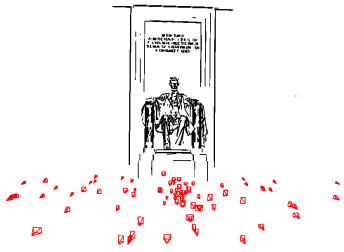


Florence Cathedral Side from [47]

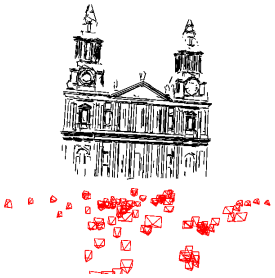


London Bridge from [47]

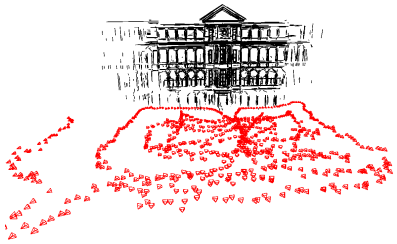
Piazza San Marco from [47]



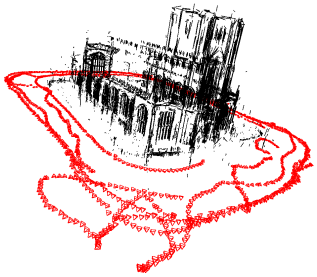
Lincoln Memorial Statue from [47]



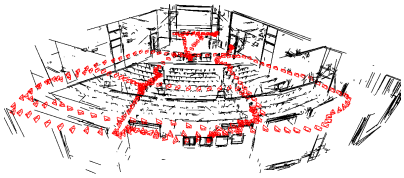
St. Paul's Cathedral from [47]



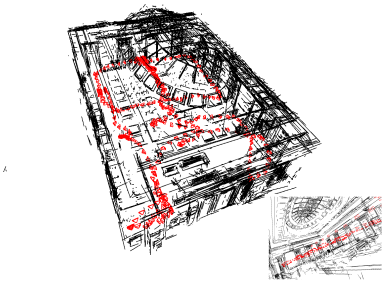
Old Hospital from [25]



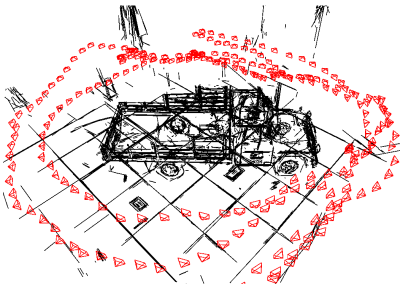
St. Mary's Church from [25]



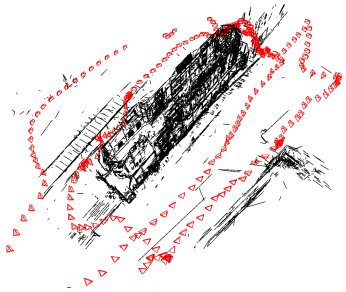
Auditorium from [26]



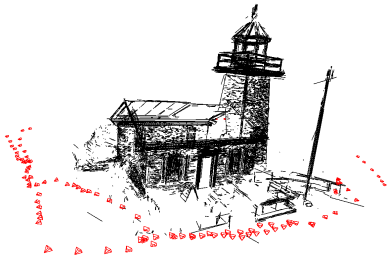
Courtroom (indoor and outdoor) from [26]



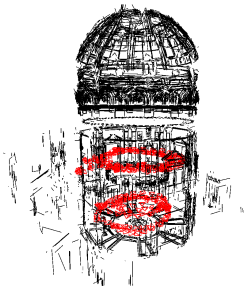
Truck from [26]



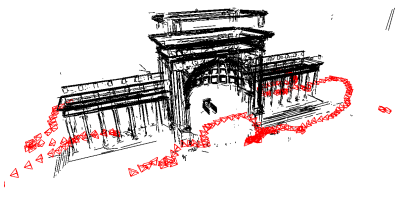
Train from [26]



Lighthouse from [26]



Museum from [26]



Temple from [26]

Figure 7. **More qualitative results of the 3D line maps recovered by our framework.**

Matcher \ Detector	LSD [51]	HAWPv3 [55]	TP-LSD [21]	SOLD2 [31]
LBD [57]	42.2 / 58.5 / (14.0 / 14.6)	6.0 / 58.0 / (7.8 / 9.8)	21.6 / 73.2 / (9.1 / 9.3)	30.7 / 69.3 / (12.2 / 18.7)
SOLD2 [31]	48.3 / 59.2 / (15.8 / 19.1)	14.7 / 62.7 / (11.2 / 20.1)	44.4 / 76.4 / (14.3 / 16.7)	**50.8** / 74.4 / (15.1 / 32.2)
L2D2 [1]	44.4 / 59.6 / (15.0 / 16.8)	13.5 / 63.4 / (10.7 / 18.3)	39.5 / **78.1** / (13.7 / 15.4)	43.9 / 72.8 / (13.7 / 24.9)
LineTR [56]	37.0 / 58.3 / (12.8 / 13.3)	5.4 / 60.5 / (8.4 / 10.7)	43.0 / 76.3 / (14.5 / 16.7)	29.0 / 70.1 / (12.3 / 19.9)
Endpoint SP [13] + NN	48.8 / 58.6 / (15.5 / 18.2)	16.2 / 63.2 / (11.2 / 20.0)	43.7 / 75.8 / (14.3 / 16.5)	49.1 / 73.7 / (14.7 / 31.4)
Endpoint SP [13] + SG [37]	48.4 / 58.0 / (15.8 / 18.9)	16.0 / 61.9 / (11.3 / 20.9)	47.1 / 76.1 / (14.5 / 16.8)	50.0 / 72.8 / (**15.5** / **34.4**)

Table 5. **Extensibility of the framework to different line detectors and matchers.** We show results of "R1 / P1 / #supports" for line mapping on Hypersim [34] with only line-line proposals. Also, we present here two customized line matchers by matching the line endpoints with either the nearest neighbor strategy ("Endpoint SP + NN") or an advanced point-based matcher SuperGlue [37] ("Endpoint SP + SG").

Line type	Triangulation	R1	R5	R10	P1	P5	P10	# supports
LSD [51]	Endpoints	27.5	102.3	140.9	57.4	**83.6**	**92.3**	(13.1 / 13.3)
	Line	**48.4**	**185.4**	**255.2**	**58.0**	80.7	88.6	(**15.8** / **18.9**)
SOLD2 [31]	Endpoints	29.4	87.6	111.3	67.0	83.8	**90.4**	(12.3 / 20.2)
	Line	**50.0**	**144.0**	**181.5**	**72.8**	**85.3**	90.2	(**15.5** / **34.4**)

Table 6. **Comparison between endpoint and line triangulation with "Endpoint SP + SG" matcher** on Hypersim [34]. While endpoint-based line matcher achieves very promising performance in Table 5, the endpoints of the matched line still do not necessarily match each other, indicating that the performance gain may come from the effectiveness of advanced point features [13].

rich 2D-3D track association that is critical for downstream applications.

G.4. Extensibility to Different Line Detectors and Matchers

To further show the flexibility of our framework to be extended to different line detectors and matchers, we test over several existing line detectors [21, 31, 51, 55] and matchers [1, 31, 56, 57] on the first eight scenes of Hypersim [34]. We also present two new matchers that are based on endpoint correspondences. Specifically, we extract SuperPoint features [13] over the two endpoints of the line and measure the structured endpoint distance with either nearest neighbor matching or an advanced point matcher SuperGlue [37].

Table 5 shows the results of all 6×4 combinations. Some interesting facts can be observed from the table. TP-LSD [21] achieves the highest precision with L2D2 [1], while SOLD2 [31] achieves the highest length recall. LSD [51] is consistently good on the length recall, while struggling on precision and track association due to its nature of being less structural. LineTR [56] is particularly good at matching TP-LSD [21] lines, while struggling on other detections compared to other matchers.

The endpoint-based line matcher is surprisingly effective, as "Endpoint SP + SG" consistently achieves the best track association under all detectors. We further test again the comparison between endpoint triangulation and algebraic line triangulation to see whether the endpoints of the matched line from the endpoint-based line matcher correspond to each other. Results in Table 6 show similar trends as Table 3 in the main paper, where performing algebraic line triangulation is significantly better than directly triangulating endpoints. This finding indicates that the endpoint-based matcher is surprisingly effective on matching lines with, however, unmatched endpoints, which may be due to the advantages of the rich point features [13]. This encourages more research towards integrating the success of existing point description and matching solutions to improve line matching.

We believe that, with the flexible design and modular Python bindings, our line mapping system can help benchmarking and facilitate the progress of developing advanced line detection and matching algorithms.

G.5. More Qualitative Results of Our Line Maps

We show more qualitative results of our reconstructed 3D line maps across datasets [25, 26, 47] in Figure 7.

H. More Results on Visual Localization

In this section, we first present the design details of our proposed visual localization pipeline with points and lines. Then, we provide experimental details on Cambridge [25] and 7Scenes [46] as well as per-scene results. Finally, we show additional results on the large-scale InLoc dataset [49].

H.1. Details on Our Visual Localization Pipeline

The input to the proposed visual localization pipeline is a set of 2D-3D point correspondences (from point-based SfM model, e.g. COLMAP [42], or depth maps) and line correspondences (from LIMAP). We directly use the point correspondences processed by HLoc [35]. Our pipeline is implemented within a hybrid RANSAC framework [9, 39] with local optimization [10, 28]. In the hybrid RANSAC we combine four different minimal solvers on 2D-3D point correspondences (PCs) and 2D-3D line correspondences (LCs) in the following:

- P3P [32]: 3 PCs.
- P2P1LL [58]: 2 PCs + 1 LC.
- P1P2LL [58]: 1 PC + 2 LCs.
- P3LL [58]: 3 LCs.

Scene	HLoc [35]	PtLine [16]	Ours
Great Court	**9.5 / 0.05 / 20.4**	11.2 / 0.07 / 17.8	9.6 / **0.05** / 20.3
King's College	6.4 / **0.10** / 37.0	6.5 / **0.10** / 37.0	**6.2 / 0.10 / 39.4**
Old Hospital	12.5 / 0.23 / 22.5	12.7 / 0.24 / 20.9	**11.3 / 0.22 / 25.4**
Shop Facade	2.9 / 0.14 / 78.6	**2.7 / 0.12** / 79.6	**2.7** / 0.13 / **81.6**
St.Mary's Church	**3.7** / 0.13 / 61.7	4.1 / 0.13 / 62.3	**3.7 / 0.12 / 63.8**
Avg.	7.0 / 0.13 / 44.0	7.4 / 0.13 / 43.5	**6.7 / 0.12 / 46.1**

Table 7. Per-scene results of visual localization on Cambridge Dataset [25]. We report the median translation and rotation error in cm and degrees, and the pose accuracy(%) at 5 cm / 5 deg threshold.

Scene	HLoc [35]	PtLine [16]	Ours
Chess	**2.4 / 0.84 / 93.0**	**2.4** / 0.85 / 92.7	2.5 / 0.85 / 92.3
Fire	2.3 / 0.89 / 88.9	2.3 / 0.91 / 87.9	**2.1 / 0.84 / 95.5**
Heads	**1.1 / 0.75 / 95.9**	1.2 / 0.81 / 95.2	**1.1** / 0.76 / **95.9**
Office	3.1 / 0.91 / 77.0	3.2 / 0.96 / 74.5	**3.0 / 0.89 / 78.4**
Pumpkin	5.0 / 1.32 / 50.4	5.1 / 1.35 / 49.0	**4.7 / 1.23 / 52.9**
Redkitchen	4.2 / **1.39** / 58.9	4.3 / 1.42 / 58.0	**4.1 / 1.39 / 60.2**
Stairs	5.2 / 1.46 / 46.8	4.8 / 1.33 / 51.9	**3.7 / 1.02 / 71.1**
Avg.	3.3 / 1.08 / 73.0	3.3 / 1.09 / 72.7	**3.0 / 1.00 / 78.0**

Table 8. Per-scene results of visual localization on 7Scenes [46]. We report the median translation and rotation error in cm and degrees, as well as the pose accuracy at a 5 cm / 5 deg threshold.

We take the implementation from PoseLib [27] for all four solvers to solve for the absolute camera pose. Following [9], the sampling probability and termination criterion of each solver depends on the inlier ratio. For scoring the model, we measure reprojection errors on both the PCs and LCs. Specifically, 2D perpendicular distance is employed for lines. Additionally, we perform cheirality tests on both PCs and LCs. The cheirality test for a 2D-3D line correspondence is done by unprojecting both 2D endpoints onto the 3D infinite line. This is achieved by projecting the camera rays onto the infinite 3D line as discussed in Section A.5. Also, a 2D-3D line correspondence is considered an outlier if the length of the 2D reprojection of the 3D line segment is less than 2 pixels, or their overlap ratio on the 3D infinite line is less than 0.4 (of the 3D line segment). For local optimization, the joint point-line refinement is applied with 2D reprojection error (perpendicular distance for lines) with an optional weighted Huber Loss via Ceres [3], where the weights for points and lines are set similarly to the weights according to the numbers of PCs and LCs. We apply the final least square optimization on all the inliers after RANSAC terminates.

H.2. Details and Per-Scene Results on Cambridge and 7Scenes

For the experiments on both Cambridge [25] and 7Scenes [46] Datasets, we run our line mapping system with LSD line detections [51] and SOLD2 matching [31]. On both datasets, the point-alone baseline employs the best method combination in HLoc [35]: NetVLAD [4] + SuperPoint [13] + SuperGlue [37]. For Cambridge [25], we also follow HLoc [35] to resize the images to 1024×576. The inlier thresholds of our hybrid RANSAC for both points and lines are set to 6 pixels on Cambridge [25] and 5 pixels on 7Scenes [46]. Up to the date of submission, the triangulated COLMAP model [42] for Cambridge from the official repository of HLoc [35] does not consider the radial distortion in the VisualSfM model [54]. Fixing the issue and re-triangulating the point-based 3D maps result in much better performance than the original one, so we use our updated one as the point-based baseline to evaluate our results. Our design with hybrid RANSAC enables direct comparison since disabling the three line solvers will fall into a point-based RANSAC with P3P [32] which is equivalent to HLoc [35].

To compare with the recently proposed PtLine method [16], we reimplement the match filtering and their midpoint-based post-refinement strategy. Because both their line detector and their strong point-based localization baseline is not publicly available, we apply their method with our line mapping over the initial poses retrieved by HLoc [35]. We tune the IoU threshold (0.4 for Cambridge, 0.2 for 7Scenes) for filtering to get the best results on both datasets.

We here provide the per-scene results of both the PtLine [16] and our method on both datasets, in Tables 7 and 8 respectively, where our method consistently outperforms the point-based baseline [35] and the joint point-line post-refinement from PtLine [16]. The results are further supported under settings when depth maps are available in Table 2, as already discussed in Section E.

		DUC 1	DUC 2
Points	HLoc [36]	49.0 / 69.2 / 80.3	52.7 / **77.1** / 80.9
Points + Lines	PtLine [16]	49.0 / 69.2 / **81.8**	56.5 / 76.3 / 80.2
	Ours	**49.5 / 72.2** / 81.3	**60.3** / 76.8 / **81.7**

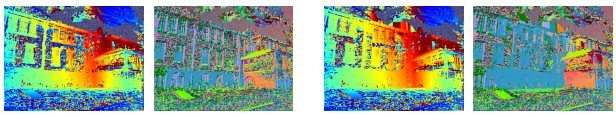
Table 9. **Results of visual localization on InLoc [49].** We report the pose AUC at 0.25m / 0.5m / 1m and 10 degrees error.

H.3. Results on InLoc dataset

We further test our method on InLoc dataset [49], again comparing with HLoc [35] as our point-only baseline and PtLine [16] as the only joint point-line visual localization method. Results on both DUC1 and DUC2 are shown in Table 9, where integrating line features again improves the performance of point-based solution, while our solution is consistently better than PtLine [16]. In particular, we improve over the point-only baseline HLoc [35] on AUC @ 0.25m by 7.6 on DUC2, by simply combining lines and points in the hybrid RANSAC framework.

	COLMAP [42]	[42] + LIMAP (line-only)	[42] + LIMAP
ai_001_001	68.0 / 87.0 / 91.3	78.3 / 91.1 / 93.8	**80.0 / 91.7 / 94.2**
ai_001_002	75.2 / 90.2 / 94.0	87.5 / 95.6 / 97.3	**88.5 / 96.0 / 97.6**
ai_001_003	83.8 / 94.4 / 96.6	82.9 / 94.0 / 96.4	**85.7 / 95.1 / 97.1**
ai_001_004	**79.2 / 88.9 / 90.9**	67.1 / 82.1 / 86.0	77.3 / 88.3 / 90.6
ai_001_005	85.1 / 94.9 / 97.0	88.4 / 96.1 / 97.7	**90.9 / 97.0 / 98.2**
ai_001_006	83.4 / 93.1 / 95.7	80.2 / 92.9 / 95.7	**84.4 / 93.8 / 96.3**
ai_001_007	59.0 / 68.5 / 70.6	64.5 / **70.6 / 71.9**	**65.0** / 70.3 / 71.7
ai_001_008	84.9 / 94.9 / 96.9	89.5 / 96.5 / 97.9	**91.3 / 97.1 / 98.2**
Average ↑	77.3 / 89.0 / 91.6	79.8 / 89.9 / 92.1	**82.9 / 91.2 / 93.0**
Median error ↓	0.188	0.173	**0.146**

Table 10. Per-scene results of joint bundle adjustment of points and lines on Hypersim [34]. Relative pose errors are measured on all image pairs. AUC @ (1° / 3° / 5°) are reported for each method and the average median error is reported an the bottom row.



Original COLMAP MVS [43] Same with line-based energy

Figure 8. **Line-assisted multi-view stereo (MVS).** Visualization (from COLMAP [42, 43]) of the depth and normal maps from COLMAP MVS [43]. Integrating line-based energy into PatchMatch Stereo largely improves completeness.

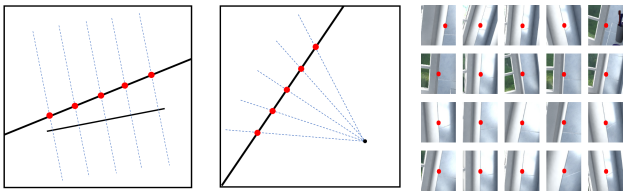
I. More Results on Refining Structure-from-Motion

We provide per-scene results in Table 10 on the joint point-line bundle adjustment experiment presented in the main paper. Combining lines with points consistently improves the accuracy of point-based Structure-from-Motion (SfM) on 7 of the 8 scenes, with notable improvement particularly on AUC@1° thanks to better pixelwise alignment with the line structures. For completeness, we also show the results with line-only optimization in Table 10. When the line structures are rich in the scene, optimizing solely over lines from the initialization of point-based SfM is able to achieve reasonable results, while combining both points and lines give the best accuracy and stability.

J. Line-Assisted Multi-view Stereo

In this section, we discuss on how to integrate the acquired line maps into PatchMatch Stereo [42] with the assumption of local planarity. Specifically, at each iteration of the PatchMatch Stereo pipeline, we can add an additional line-based energy that encourages the depth and normal at each pixel to span a plane that crosses some nearby 3D line segments. In practice, for each pixel, we collect all lines that have projections within a 2D perpendicular distance of half the line length. During the proposal selection in PatchMatch Stereo [43], we compute the perpendicular distances (sum of the perpendicular distance for both endpoints) of all collected lines to the corresponding plane proposal (spanned at each pixel with its depth and normal in PatchMatch), and sum the two minimum distances from the two closest lines as the line-based energy. This encourages the selected depth-normal pair to span a plane having at least two 3D lines that are very close, implicitly encouraging local planarity on the recovered surface with respect to the 3D line maps.

We show one qualitative result on AdelaideRMF [15] in Figure 8 to illustrate the advantage of line-assisted PatchMatch Stereo [43]. With the line-based energy, both the depth map and surface normal map become more complete in texture-less regions, and the surface normal map is comparably more smooth thanks to the local planarity implicitly enforced by the 3D line maps.



reference image target image correspondences

Figure 9. Illustration on how to extend featuremetric optimization [29] over 3D line tracks.

K. Extension: Featuremetric Line Refinement

Inspired by the recent success of featuremetric refinement for point-based Structure-from-Motion [29, 38], we here present ideas on how to extend its application for 3D line refinement. This type of refinement can be potentially very suitable for lines, since compared to point-based alternatives, line detectors usually have higher localization errors in the image. For points, it is straightforward to define featuremetric consistency loss by simply interpolating the feature map at the point locations and computing the difference. To apply the same framework for lines, it is also necessary to establish point-wise correspondences along the line to be able to measure the feature consistency. One approach is to parameterize the 3D line endpoints explicitly and sample points between them. The two main drawbacks of this approach are that directly optimizing over the endpoints might suffer from endpoint collapse, and that during optimization, the supporting images for each sampled point might change as the endpoints shift.

To avoid these issues, we here present an alternative formulation that instead optimizes over the infinite 3D line and defines the sample points directly in 2D. The idea is to parameterize the sampling of points through line intersection (Figure 9). Specifically, we first uniformly sample 3D points between the two initial endpoints. For each sampled 3D point, we determine the supporting images and 2D line segments. From these 2D line segments, we select the reference

line segment as the longest one and construct a perpendicular 2D line based on the projection. The sampled 3D points are then discarded. For the optimization, we project the infinite 3D lines onto the images, which are then intersected with the perpendicular 2D lines to give us the 2D sample points in the reference views. These sample points are then mapped to epipolar lines in the set of supporting images, and then by intersecting with the projections of the 3D line, we get the corresponding sample points in the other views. With these point-wise correspondences, we can compute the featuremetric consistency loss used in the optimization. This is illustrated in Figure 9. The infinite line can again be minimally parameterized with Plücker coordinate discussed in Section A.

To reduce memory requirements for the feature maps, a specially designed *line patching* strategy can be employed, where an oriented bounding box around the lines is extracted with bilinear sampling on the non-integral coordinates. The 2D rotation and translation are stored along with each line patch for the local-global coordinate transformation. This *line patching* can significantly save memory while still allowing us to accurately interpolate the features of the sampled points lying close to the line segment.

L. Limitations and Future Work

In this section, we discuss the current limitations of the proposed system and give an overview of potential areas for improvement and future work.

The current system is designed to reconstruct 3D lines from known camera poses, i.e. we focused on the mapping step of the reconstruction pipeline. While we show in the experiments that the system is robust to imperfect camera poses (e.g. obtained from SLAM or SfM), it is still dependent on the point-based reconstruction/tracking working and we cannot recover if they fail. Interesting future work is to integrate our mapping pipeline into an incremental Structure-from-Motion framework. We believe this is a promising direction as our experiments show that both localization (i.e. registering a new image to a reconstruction) and bundle adjustment can be improved with our line maps.

The system is also dependent on the reliability of the employed line detector and matcher, as shown in Table 5. Improving the detection, description, and matching of 2D lines can benefit a lot on the resulting 3D line maps. While this is beyond the scope of this paper, by sharing our library with the community we hope to facilitate relevant research developments on 2D line-related practice.

The current system design is partly due to the fact that we have weaker detectors and matchers for lines compared to points. This requires more excessive geometric verification before building tracks compared to point-based triangulation methods which can be more greedy in their selection. While line triangulation is inherently less stable than point triangulation, improvements in line detection and matching might lessen the need for geometric verification.

References

[1] Hichem Abdellali, Robert Frohlich, Viktor Vilagos, and Zoltan Kato. L2d2: Learnable line detector and descriptor. In *3DV*, 2021. 13

[2] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011. 9

[3] Sameer Agarwal and Keir Mierle. Ceres solver. http://ceres-solver.org. 2, 6, 14

[4] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *CVPR*, 2016. 14

[5] Adrien Bartoli and Peter Sturm. Structure-from-motion using lines: Representation, triangulation, and bundle adjustment. *Computer Vision and Image Understanding (CVIU)*, 100(3):416–441, 2005. 1, 2

[6] Jean-Charles Bazin and Marc Pollefeys. 3-line ransac for orthogonal vanishing point detection. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4282–4287. IEEE, 2012. 5, 6

[7] Jose Luis Blanco and Pranjal Kumar Rai. nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees. https://github.com/jlblancoc/nanoflann, 2014. 9

[8] Eric Brachmann and Carsten Rother. Visual camera relocalization from RGB and RGB-D images using DSAC. *TPAMI*, 2021. 8, 9

[9] Federico Camposeco, Andrea Cohen, Marc Pollefeys, and Torsten Sattler. Hybrid camera pose estimation. In *CVPR*, 2018. 8, 13, 14

[10] Ondrej Chum, Jiri Matas, and Josef Kittler. Locally optimized ransac. In *Joint Pattern Recognition Symposium*, pages 236–243, 2003. 8, 13

[11] James Coughlan and Alan L Yuille. The manhattan world assumption: Regularities in scene statistics which enable bayesian inference. In *NeurIPS*, 2000. 6

[12] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, pages 5828–5839, 2017. 8, 9

[13] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018. 4, 9, 13, 14

[14] Mihai Dusmanu, Johannes L Schönberger, and Marc Pollefeys. Multi-view optimization of local feature geometry. In *ECCV*, pages 670–686. Springer, 2020. 5

[15] Bin Fan, Fuchao Wu, and Zhanyi Hu. Robust line matching through line–point invariants. *Pattern Recognition*, 45(2):794–805, 2012. 15

[16] Shuang Gao, Jixiang Wan, Yishan Ping, Xudong Zhang, Shuzhou Dong, Yuchen Yang, Haikuan Ning, Jijunnan Li,

and Yandong Guo. Pose refinement with joint optimization of visual points and lines. In *IROS*, 2022. 1, 8, 14

[17] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 1, 2, 7

[18] William Vallance Douglas Hodge and Daniel Pedoe. *Methods of algebraic geometry*, volume 1. CUP Archive, 1947. 1

[19] Manuel Hofer. Line3D++. https://github.com/manhofer/Line3Dpp. 10

[20] Manuel Hofer, Michael Maurer, and Horst Bischof. Efficient 3d scene abstraction using line segments. *Computer Vision and Image Understanding (CVIU)*, 157:167–178, 2017. 1, 5, 9, 10, 11

[21] Siyu Huang, Fangbo Qin, Pengfei Xiong, Ning Ding, Yijia He, and Xiao Liu. Tp-lsd: Tri-points based line segment detector. In *ECCV*, 2020. 13

[22] Alec Jacobson, Daniele Panozzo, et al. libigl: A simple C++ geometry processing library, 2018. https://libigl.github.io/. 9

[23] Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. Image matching across wide baselines: From paper to practice. *IJCV*, 129(2):517–547, 2021. 9

[24] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *CVPR*, 2017. 9

[25] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A convolutional network for real-time 6-DoF camera relocalization. In *ICCV*, 2015. 9, 12, 13, 14

[26] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017. 8, 9, 10, 11, 12, 13

[27] Viktor Larsson. PoseLib - Minimal Solvers for Camera Pose Estimation. https://github.com/vlarsson/PoseLib. 14

[28] Karel Lebeda, Jiri Matas, and Ondrej Chum. Fixing the Locally Optimized RANSAC. In *BMVC*, 2012. 8, 13

[29] Philipp Lindenberger, Paul-Edouard Sarlin, Viktor Larsson, and Marc Pollefeys. Pixel-perfect structure-from-motion with featuremetric refinement. In *ICCV*, 2021. 15

[30] Matthew T Mason. *Mechanics of robotic manipulation*. MIT press, 2001. 1

[31] Rémi Pautrat, Juan-Ting Lin, Viktor Larsson, Martin R Oswald, and Marc Pollefeys. Sold2: Self-supervised occlusion-aware line description and detection. In *CVPR*, 2021. 8, 9, 10, 11, 13, 14

[32] Mikael Persson and Klas Nordberg. Lambda twist: An accurate fast robust perspective three point (p3p) solver. In *ECCV*, 2018. 13, 14

[33] Gernot Riegler and Vladlen Koltun. Stable view synthesis. In *CVPR*, 2021. 9

[34] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *ICCV*, 2021. 8, 9, 10, 11, 13, 15

[35] Paul-Edouard Sarlin. Visual localization made easy with hloc. https://github.com/cvg/Hierarchical-Localization/. 8, 9, 13, 14

[36] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, 2019. 14

[37] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 13, 14

[38] Paul-Edouard Sarlin, Ajaykumar Unagar, Mans Larsson, Hugo Germain, Carl Toft, Viktor Larsson, Marc Pollefeys, Vincent Lepetit, Lars Hammarstrand, Fredrik Kahl, et al. Back to the feature: Learning robust camera localization from pixels to pose. In *CVPR*, 2021. 15

[39] Torsten Sattler et al. RansacLib - A Template-based *SAC Implementation. https://github.com/tsattler/RansacLib. 13

[40] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, et al. Benchmarking 6dof outdoor visual localization in changing conditions. In *CVPR*, 2018. 9

[41] Grant Schindler and Frank Dellaert. Atlanta world: An expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments. In *CVPR*, 2004. 6

[42] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 5, 7, 9, 13, 14, 15

[43] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, pages 501–518. Springer, 2016. 1, 15

[44] Thomas Schops, Torsten Sattler, and Marc Pollefeys. Bad slam: Bundle adjusted direct rgb-d slam. In *CVPR*, pages 134–144, 2019. 8, 9

[45] Thomas Schops, Johannes L Schonberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *CVPR*, pages 3260–3269, 2017. 8, 9

[46] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in RGB-D images. In *CVPR*, 2013. 8, 9, 13, 14

[47] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM SIGGRAPH*, 2006. 9, 12, 13

[48] Noah Snavely, Steven M Seitz, and Richard Szeliski. Modeling the world from internet photo collections. *IJCV*, 80(2):189–210, 2008. 9

[49] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. Inloc: Indoor visual localization with dense matching and view synthesis. In *CVPR*, 2018. 1, 9, 13, 14

[50] Roberto Toldo and Andrea Fusiello. Robust multiple structures estimation with j-linkage. In *ECCV*, 2008. 5, 9

[51] Rafael Grompone Von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. Lsd: A fast line segment detector with a false detection control. *TPAMI*, 32(4):722–732, 2008. 8, 9, 10, 11, 13, 14

[52] Dong Wei, Yi Wan, Yongjun Zhang, Xinyi Liu, Bin Zhang, and Xiqi Wang. Elsr: Efficient line segment reconstruction with planes and points guidance. In *CVPR*, 2022. 1, 10, 11

[53] Hoi Sim Wong, Tat-Jun Chin, Jin Yu, and David Suter. Dynamic and hierarchical multi-structure geometric model fitting. In *ICCV*, 2011. 7, 9

[54] Changchang Wu. Visualsfm: A visual structure from motion system. *http://www. cs. washington. edu/homes/ccwu/vsfm*, 2011. 10, 14

[55] Nan Xue, Tianfu Wu, Song Bai, Fudong Wang, Gui-Song Xia, Liangpei Zhang, and Philip HS Torr. Holistically-attracted wireframe parsing. In *CVPR*, 2020. 13

[56] Sungho Yoon and Ayoung Kim. Line as a visual sentence: Context-aware line descriptor for visual localization. *IEEE Robotics and Automation Letters*, 6(4):8726–8733, 2021. 13

[57] Lilian Zhang and Reinhard Koch. An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency. *Journal of Visual Communication and Image Representation*, 24(7):794–805, 2013. 13

[58] Lipu Zhou, Jiamin Ye, and Michael Kaess. A stable algebraic camera pose estimation for minimal configurations of 2d/3d point and line correspondences. In *ACCV*, 2018. 13