# Fully Geometric Panoramic Localization

Supplementary Material

## A. Method Details

### A.1. Input Preparation

As explained in Section 4.1, our method operates in a fully geometric manner using lines and their intersections for localization. Below we explain the detailed procedures for how the inputs are prepared prior to localization.

**Line Extraction** Similar to LDL [16], our method extracts line segments in 2D and 3D using off-the-shelf line detectors. For 2D, we create perspective crops of the input panorama and apply LSD [10]. The detected lines are then converted to the spherical coordinate frame compatible to our method. For 3D, we extract lines from the colored point cloud provided in OmniScenes [14] and Stanford 2D-3D-S [2] by applying the 3D line detection algorithm from Xiaohu et al. [28].

**Principal Direction Extraction** To imbue spatial context to lines, we additionally extract principal directions from lines as in LDL [16]. For 2D, we find the principal directions by estimating vanishing points. Specifically, we extrapolate all lines in 2D and find their intersections. Then the intersection points are binned to a spherical grid, from which the top-3 bins with the largest number of points are selected and used as 2D principal directions. For 3D, we similarly bin the line directions on the spherical grid and extract the top-3 directions. Note we can apply more sophisticated vanishing point estimation algorithms such as Pautrat et al. [23], which will lead to an enhanced localization performance since principal directions are first used for rotation estimation. We leave such extensions to future work.

**Line Intersection Extraction** Our method uniquely leverages line intersections as important cues for accurate pose search and refinement. As mentioned in Section 4.1, we intersect lines from distinct principal directions. First for 2D line pairs, we extrapolate the lines on the sphere and obtain the intersection coordinates, and then keep intersections only if its spherical distance to both line segments are below a designated threshold $\delta_{\text{2D}} = 0.1$rad. Similarly for 3D line pairs, we extrapolate lines in the 3D space to get intersections, and then keep the intersections only if the distances to both line segments are below a threshold $\delta_{\text{3D}} = 0.15m$.

### A.2. Theoretical Analysis of Efficient Distance Function Comparison

In this section, we theoretically analyze the efficient distance function comparison presented in Section 4.2.2. Recall that instead of exhaustively computing distance functions on-the-fly as in prior work [16], we propose to cache distance function values prior to localization to enable scalable and efficient localization. We start by proving Theorem 1 from Section 4.2.2 which justifies our efficient distance function comparison. The proof builds upon the following lemma:

**Lemma 1.** Given a metric $d(\cdot,\cdot)$ defined over the unit sphere $\mathbb{S}^2$, let $f(x;S) := \min_{s\in S} d(x,s)$ denote a distance function to a set of spherical points $S \subset \mathbb{S}^2$. For two arbitrary points $p_1, p_2$ on the unit sphere $\mathbb{S}^2$, there exists a point $s^* \in S$ that satisfies the following inequality,

$$|f(p_1;S) - f(p_2;S)| \leq |d(p_1,s^*) - d(p_2,s^*)|. \quad \text{(A.1)}$$

**Proof** Without loss of generality, suppose that $f(p_1;S) \geq f(p_2;S)$. Also, let $s_i = \arg\min_{s\in S} d(p_i,s)$ for $i = 1,2$. Then we have the following,

$$|f(p_1;S) - f(p_2;S)| = d(p_1,s_1) - d(p_2,s_2) \quad \text{(A.2)}$$
$$\leq d(p_1,s_2) - d(p_2,s_2) \leq |d(p_1,s_2) - d(p_2,s_2)|, \quad \text{(A.3)}$$

and thus by setting $s^* = s_2$, Equation A.1 holds true, which proves the lemma.

Using the lemma, we prove the following theorem stated in Section 4.2:

**Theorem 1.** *Consider a countable, finite set of spherical points $Q \subset \mathbb{S}^2$ that satisfy $\max_{q\in Q} \min_{\hat{q}\in Q} d(q,R\hat{q}) \leq \max_{q\in Q} \min_{\hat{q}\neq q} d(q,\hat{q}) = \delta$ for all $R \in SO(3)$. For an arbitrary rotation $\tilde{R} \in SO(3)$, the following inequality holds:*

$$\frac{1}{|Q|}\sum_{q\in Q} |f(q;S) - f(\arg\min_{\hat{q}\in Q} d(\hat{q},\tilde{R}q);\tilde{R}S)| \leq \delta. \quad \text{(A.4)}$$

**Proof** We prove the following inequality for an arbitrary query point $q \in Q$, which when summed for all points in $Q$ equivalent to Equation A.4,

$$|f(q;S) - f(\arg\min_{\hat{q}\in Q} d(\hat{q},\tilde{R}q);\tilde{R}S)| \leq \delta. \quad \text{(A.5)}$$

First, since distance function values are preserved under rotation of both the query point $q$ and the point set, we have $f(q;S) = f(\tilde{R}q;\tilde{R}S)$. Further, Lemma 1 suggests that there exists a point $s^* \in S$ satisfying the following inequality,

$$|f(\tilde{R}q;\tilde{R}S) - f(\arg\min_{\hat{q}\in Q} d(\hat{q},\tilde{R}q);\tilde{R}S)| \tag{A.6}$$

$$\leq |d(\tilde{R}q,\tilde{R}s^*) - d(\arg\min_{\hat{q}\in Q} d(\hat{q},\tilde{R}q),\tilde{R}s^*)|. \tag{A.7}$$

Using the triangle inequality and the dense point assumption $\max_{q\in Q}\min_{\hat{q}\in Q} d(q,\tilde{R}\hat{q}) \leq \delta$, we have

$$|d(\tilde{R}q,\tilde{R}s^*) - d(\arg\min_{\hat{q}\in Q} d(\hat{q},\tilde{R}q),\tilde{R}s^*)| \tag{A.8}$$

$$\leq d(\tilde{R}q, \arg\min_{\hat{q}\in Q} d(\hat{q},\tilde{R}q)) = \min_{\hat{q}\in Q} d(\hat{q},\tilde{R}q) \leq \delta, \tag{A.9}$$

which in turn proves Equation A.5.

**Derivation of Efficient Distance Function Comparison** Using the theorem, we can derive the efficient distance function comparison presented in Section 4.2.2. Given an arbitrary rotation $\tilde{R}$, from Equation A.4 we have that

$$\frac{1}{|Q|}\sum_{q\in Q} |f(q;S) - f(\arg\min_{\hat{q}\in Q} d(\hat{q},\tilde{R}q);\tilde{R}S)| \tag{A.10}$$

$$= \frac{1}{|Q|}\sum_{q\in Q} |f(q;S) - f(\arg\min_{\hat{q}\in \tilde{R}^T Q} d(\hat{q},q);S)| \leq \delta. \tag{A.11}$$

This implies that for a one-to-one mapping $m(\cdot) : Q \to \tilde{R}^T Q$ that satisfies $d(m(q),q) \leq \delta$ for all $q \in Q$,

$$\frac{1}{|Q|}\sum_{q\in Q} |f(q;S) - f(m(q);S)| \leq \delta. \tag{A.12}$$

Note that the existence of such a mapping is given from the Hall's marriage theorem [11], assuming that for an arbitrary subset of points $\mathcal{Q} \subset Q$, we can find a subset of points $\mathcal{M} \subset \tilde{R}Q$ that satisfies $\max_{q\in\mathcal{Q},m\in\mathcal{M}} d(q,m) \leq \delta$ and $|\mathcal{Q}| \leq |\mathcal{M}|$.

We can now use the results to derive a error bound on our approximation scheme. For line distance functions, the cumulative deviation between using our approximation (Equation 8 from Section 4.2.2) and the original LDL [16] formulation (Equation 3 from Section 3) is bounded as follows,

$$\frac{1}{|Q|}\Big|\underbrace{\sum_{q\in Q} |f_{2D}(q;L_{2D}) - f_{3D}(q;L_{3D},R,t)|}_{\text{original (LDL)}} \tag{A.13}$$

$$- \underbrace{\sum_{q\in Q} |f_{2D}(q;R^T L_{2D}) - f_{3D}(q;L_{3D},I,t)|}_{\text{ours}}\Big| \tag{A.14}$$

$$= \frac{1}{|Q|}\Big|\sum_{q\in Q} |f_{2D}(q;L_{2D}) - f_{3D}(q;L_{3D},R,t)| \tag{A.15}$$

$$- \sum_{q\in Q} |f_{2D}(Rq;L_{2D}) - f_{3D}(Rq;L_{3D},R,t)|\Big| \tag{A.16}$$

$$\leq \frac{1}{|Q|}\Big|\sum_{q\in Q} f_{2D}(q;L_{2D}) - f_{3D}(q;L_{3D},R,t) \tag{A.17}$$

$$- \sum_{q\in Q} f_{2D}(Rq;L_{2D}) + f_{3D}(Rq;L_{3D},R,t)\Big| \tag{A.18}$$

$$= \frac{1}{|Q|}\Big|\sum_{q\in Q} \big(f_{2D}(q;L_{2D}) - f_{2D}(m(q);L_{2D})\big) \tag{A.19}$$

$$+ \big(f_{3D}(m(q);L_{3D},R,t) - f_{3D}(q;L_{3D},R,t)\big)\Big| \tag{A.20}$$

$$\leq \frac{1}{|Q|}\sum_{q\in Q} \big(|f_{2D}(q;L_{2D}) - f_{2D}(m(q);L_{2D})| \tag{A.21}$$

$$+ |f_{3D}(m(q);L_{3D},R,t) - f_{3D}(q;L_{3D},R,t)|\big) \leq 2\delta, \tag{A.22}$$

where $m(\cdot) : Q \to RQ$ is the one-to-one mapping defined similarly as in Equation A.12. Therefore, when we have a sufficiently small $\delta$ from a dense set of query points $Q$, our efficient approximation closely follows the original line distance function comparison from LDL. A similar derivation can also be made for point distance functions, but note that for point distance functions we have the sharpening parameter $\gamma = 0.2$ applied to the spherical distance (Equation 4, 5). Theorem 1 and its consequences still hold however, due to the fact that for a metric $d(\cdot,\cdot)$ defined on a bounded set, the composition of the metric with an increasing concave function $f(\cdot)$, namely $f(d(\cdot,\cdot))$ is still a metric [9, 18] (also known as the *snowflake* metric).

### A.3. Translation / Rotation Pool Generation

We further elaborate on how the translation and rotation pools are generated for our method. As explained in Section 3, we follow the pool generation method from LDL [16]. Specifically, we generate translation pools by first creating a bounding box of the 3D map and subdividing the bounding box into grid partitions. The center of each grid is used as the translation pool. We then generate rotation pools by combinatorially associating principal directions in 2D and 3D. Given three principal directions in 2D and 3D, namely $d_i \in D_{2D}$ and $\tilde{d}_i \in D_{3D}$, each rotation is determined from a permutation that associates directions in 2D with 3D. To elaborate, a rotation matrix can be obtained from an arbitrary permutation $\sigma(\cdot)$

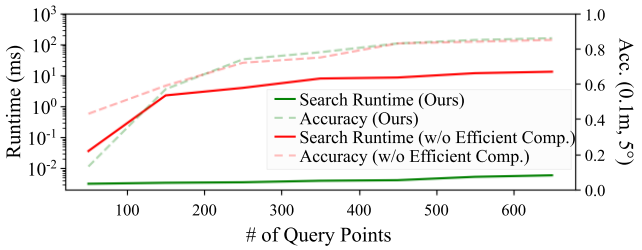



Figure B.1. Efficacy of our efficient distance function comparison on localization accuracy and pose search runtime. We plot the metrics under varying number of query points $|Q|$ on Room 5 from OmniScenes [14]. Larger number of query points lead to significantly enhanced accuracy, but the distance function comparison originally propose from LDL [16] shows large increase in runtime. On the other hand, our formulation shows an almost constant pose search runtime with nearly no loss in localization accuracy. Note the runtime is plotted in log scale.

by applying the Kabsch algorithm [13] on direction pairs $\{(d_1, \tilde{d}_{\sigma(1)}), (d_2, \tilde{d}_{\sigma(2)}), (d_3, \tilde{d}_{\sigma(3)})\}$. Note that due to the additional sign ambiguity when associating the principal directions (i.e., for a fixed $\sigma(\cdot)$, $d_i$ can be associated with $\pm d_{\sigma(i)}$), there can exist $2^3 \times 3$associations.

### A.4. Hyperparameter Setup

We report details about the hyperparameters not reported in the main paper. First for pose search, we filter short lines to reduce the effect of noisy line misdetections. We specifically filter 3D lines whose lengths are over $20cm$, and then filter 2D lines by length to match the ratio of filtered 3D lines. For pose refinement, we use Adam [17] for optimizing the cost functions (Equation 14, 15), with a step size of 0.1 for 100 iterations.

## B. Additional Experimental Results

### B.1. Scalability against Number of Query Points

One of the key factors for accurate localization of our method is the dense set of query points used during pose search. In all our experiments, we use $|Q| = 642$ query points, which is much larger than that of LDL ($|Q| = 42$). By using denser query points, we can better compare the fine-grained scene details, leading to enhanced pose search performance. However, naively increasing the number of query points leads to a significant increase in pose search runtime.

The efficient distance function comparison enables our method to handle dense query points without a noticeable increase in runtime. To illustrate the effect of using our efficient formulation, we assess the runtime and localization accuracy in Room 5 from OmniScenes [14] using varying number of query points. Figure B.1 plots the accuracy and pose search runtime with respect to the number of query points. First, one can observe that there is a clear, positive correlation between the number of query points and localization accuracy. However, without the efficient comparison scheme, runtime also dramatically increases. As our distance function comparison scheme pre-computes and caches 3D distance functions, which is the largest bottleneck for scaling query points, pose search runtime remains almost constant with the increase in query points. Further, note that one can attain a reasonable localization accuracy once query points are sufficiently dense ($|Q| \geq 350$ for Figure B.1). Therefore, for memory critical applications, one may choose to employ a smaller number of query points to reduce the map size.

| Method | *t*-error | *R*-error | Accuracy (0.1m, 5°) |
|---|---|---|---|
| Only rot. refine | 0.21 | 1.56 | 0.08 |
| Alternating trans. & rot. refine | 0.07 | 1.55 | 0.74 |
| No intersection match update | 0.17 | 1.56 | 0.13 |
| Ours | **0.06** | **1.05** | **0.77** |

Table B.1. Ablation study of key components of our pose refinement method, using the Room 2 subset from OmniScenes [14].

### B.2. Additional Ablation Study for Pose Refinement

In this section we conduct additional ablation experiments for the pose refinement module. Recall from Section 4.3 that we refine poses by aligning line intersections on the sphere, where translation is first optimized followed by rotation. We consider three ablated variations of pose refinement: (i) only optimizing rotation, (ii) alternating translation and rotation refinement each step, and (iii) omitting match updates during each translation refinement step. The results are shown in Table B.1, where we conduct evaluation in the identical setup as in Section 5.3. Only optimizing rotation leads to a large drop in localization accuracy, since the translations from initial pose pools are usually at least $0.10m$ away from the ground truth. Further, alternating translation and rotation instead of optimizing them sequentially lead to a slight drop in performance. As the initial rotation estimate is already fairly accurate (obtained by aligning principal directions), placing more weight on optimizing translation during the initial stage of refinement is beneficial. Finally, omitting the match update scheme leads to a dramatic decrease in performance. As the initial translations are imperfect, the iterative updates during translation refinement perform a crucial role in both obtaining good translations and intersection point matches.

### B.3. Robustness Against Line Detector Variations

We further evaluate the robustness of our method against line detector variations. As mentioned in Section A.1, we apply LSD [10] for 2D line detection. In this section we test if our method can also handle line segments from other detection methods. We conduct evaluations in the Om-

| Line Detector | *t*-error | *R*-error | Accuracy (0.1m, 5°) |
|---|---|---|---|
| ELSED [27] | **0.06** | 0.80 | 0.71 |
| DeepLSD [22] | **0.06** | **0.73** | 0.71 |
| LSD [10] (Ours) | **0.06** | 0.96 | **0.77** |

Table B.2. Robustness evaluation against line detector variations, using the Extreme split from OmniScenes [14].

niScenes dataset [14], using the top-1 retrieval results for refinement. To fairly test the generalizability of our method, all the hyperparameters are fixed to the setup used for the original set of detectors.

We test variations in 2D line detectors using two recently proposed methods: ELSED [27] and DeepLSD [22]. Similar to how we applied LSD [10] on panoramas, we first make perspective crops of the panorama and apply the detectors. As shown in Table B.2, our method shows consistent performance amidst changes in the 2D line detectors. The results suggest that our formulation is sufficiently generalizable and versatile to handle varying line detectors.

### B.4. Full Experimental Results

We finally report the full evaluation results for pose refinement in Stanford 2D-3D-S [2] and OmniScenes [14]. Note that the results presented in Table 3 and 5 are the results aggregated from the two datasets. Table B.3 and B.4 shows the full results for pose refinement in regular setups and lighting variations. In both cases, our method shows competitive performance against the visual descriptor-based methods while constantly outperforming the geometry-based methods.

## C. Baseline Details

In this section, we explain the implementation details of the baselines compared against our method.

**Pose Search Baselines** We consider three types of baselines for comparison: neural network based (SFRS [8], Cosplace [3]), color distribution-based (PICCOLO [14], CPO [15]), and line-based (LDL [16], Chamfer [21], SFRS$^{\text{L}}$, Cosplace$^{\text{L}}$). Neural network-based methods operate by first extracting global feature descriptors for image renderings in the 3D map and establishing comparisons against that obtaine from the query image. We specifically use the colored point clouds available in our test datasets to render synthetic views. Color distribution-based methods operate by directly comparing the color values between the panorama and point cloud. PICCOLO [14] operates using a loss function defined over various candidate poses that measures the color discrepancy between the point cloud projections and the panorama color values sampled from the projection locations. CPO [15] takes a more holistic approach by comparing the patch-level color histograms of the query and synthetic views in the point cloud. Line-based methods solely utilize the geometry of the line maps for pose search. LDL [16] uses line distance functions for pose search as explained in Section 3, while Chamfer-based method [21] uses the pairwise distances between the lines in 2D and 3D for pose search. We additionally test variants of the neural network-based methods, namely SFRS$^{\text{L}}$ and Cosplace$^{\text{L}}$, that extracts global descriptors from the 2D line extractions and those from synthetic views in 3D. As demonstrated in Section 5.2, our method performs competitively against the baselines by leveraging the holistic geometry of lines and their intersections, while exhibiting a very short runtime due to the efficient search scheme.

**Pose Refinement Baselines** Similar to pose search evaluation, we consider three types of baselines: line-based (GlueStick [24], Line Transformer [29]), point-based (SuperGlue [25], LightGlue [20], LoFTR [26], PICCOLO [14], CPO [15]), and geometry-based (GoMatch [31], BPnPNet [4], Gao et. al [7], PDF Minimization, LoFTR$^{\text{L}}$, SuperGlue$^{\text{L}}$, LightGlue$^{\text{L}}$, GlueStick$^{\text{L}}$, Line Transformer$^{\text{L}}$). Line-based methods operate by matching visual descriptors for lines and additionally for points, where we apply PnL-RANSAC [1, 6] similar to Yoon et al. [29] on the point and line matches to obtained refinement results.

Point-based methods can be further divided into neural network-based methods (SuperGlue [25], LightGlue [20], LoFTR [26]) and color matching-based methods (PICCOLO [14], CPO [15]). Neural network-based methods match point descriptors using graph neural networks or transformers, where we apply PnP-RANSAC [6, 12, 19] on the matches to get a refined pose. Note for LightGlue [20] we used the SuperPoint [5] keypoint descriptors as input to the matcher. Both color matching-based methods tested in our experiments minimize a loss function termed sampling loss [14], which directly compares the point color values against the panorama image's color values at projected locations. Here, the distinction between PICCOLO and CPO is in that CPO additionally exploits 3D score maps to place weights on regions less likely to contain changes during sampling loss optimization.

Finally, for geometry-based methods we specifically test neural network-based methods (GoMatch [31], BPnPNet [4]), optimization-based methods (Gao et al. [7], PDF minimization), and line image-based methods (LoFTR$^{\text{L}}$, SuperGlue$^{\text{L}}$, LightGlue$^{\text{L}}$, GlueStick$^{\text{L}}$, Line Transformer$^{\text{L}}$). For neural network-based methods which operate by matching learned descriptors for keypoint locations, we use SuperPoint [5] keypoint detections. In addition, as the bearing vector representation of GoMatch cannot fully handle the 360° view of panoramas, we subdivide the panoramas into $N_{\text{split}}$ horizontally split regions and sep-

| Dataset | | | OmniScenes | | | Stanford 2D-3D-S | | |
|---|---|---|---|---|---|---|---|---|
| Refinement | Method | Visual Desc. | Accuracy (0.1m, 5°) | Accuracy (0.2m, 10°) | Accuracy (0.3m, 15°) | Accuracy (0.1m, 5°) | Accuracy (0.2m, 10°) | Accuracy (0.3m, 15°) |
| Line-Based Refinement | Line Transformer [29] | ◯ | 0.88 | 0.92 | 0.93 | 0.70 | **0.72** | **0.73** |
| | Gluestick [24] | ◯ | 0.89 | 0.93 | 0.94 | 0.68 | 0.71 | 0.72 |
| Point-Based Refinement | SuperGlue [25] | ◯ | 0.90 | **0.95** | **0.95** | **0.71** | **0.72** | **0.73** |
| | LightGlue [20] | ◯ | **0.93** | **0.95** | **0.95** | **0.71** | **0.72** | **0.73** |
| | LoFTR [26] | ◯ | 0.88 | 0.94 | **0.95** | 0.67 | 0.69 | 0.69 |
| | PICCOLO [14] | ◯ | 0.58 | 0.60 | 0.61 | 0.57 | 0.59 | 0.60 |
| | CPO [15] | ◯ | 0.76 | 0.78 | 0.78 | 0.55 | 0.56 | 0.58 |
| Geometric Refinement | GoMatch [31] | ✗ | 0.67 | 0.84 | 0.88 | 0.57 | 0.65 | 0.67 |
| | BPnPNet [4] | ✗ | 0.01 | 0.18 | 0.41 | 0.05 | 0.24 | 0.41 |
| | Gao et. al [7] | ✗ | 0.11 | 0.43 | 0.74 | 0.26 | 0.63 | 0.62 |
| | PDF Minimization | ✗ | 0.28 | 0.52 | 0.63 | 0.18 | 0.31 | 0.38 |
| | LoFTR$^{L}$ [26] | ✗ | 0.06 | 0.26 | 0.55 | 0.10 | 0.33 | 0.47 |
| | SuperGlue$^{L}$ [25] | ✗ | 0.06 | 0.28 | 0.60 | 0.06 | 0.37 | 0.56 |
| | LightGlue$^{L}$ [20] | ✗ | 0.45 | 0.69 | 0.82 | 0.33 | 0.51 | 0.58 |
| | GlueStick$^{L}$ [24] | ✗ | 0.36 | 0.60 | 0.75 | 0.30 | 0.46 | 0.55 |
| | Line Transformer$^{L}$ [29] | ✗ | 0.05 | 0.24 | 0.54 | 0.07 | 0.31 | 0.47 |
| | Ours | ✗ | 0.77 | 0.89 | 0.91 | 0.62 | 0.66 | 0.67 |

Table B.3. Pose refinement evaluation in OmniScenes [14] and Stanford 2D-3D-S [2]. We retrieve top-1 poses using point distance functions and perform refinement with various baseline methods. Note the superscript X$^{L}$ denotes that the baseline takes line images as input.

| Method | Visual Desc. | Orig. | Intensity$^{1}$ | Gamma$^{1}$ | White Balance$^{1}$ | Intensity$^{2}$ | Gamma$^{2}$ | White Balance$^{2}$ | Range | Std |
|---|---|---|---|---|---|---|---|---|---|---|
| Line Transformer [29] | ◯ | 0.88 | 0.67 | 0.88 | **0.89** | 0.73 | 0.83 | 0.89 | 0.22 | 0.09 |
| Gluestick [24] | ◯ | **0.89** | **0.77** | **0.89** | **0.89** | **0.83** | **0.89** | **0.90** | 0.13 | 0.05 |
| Line Transformer [29] | ✗ | 0.67 | 0.64 | 0.66 | 0.67 | 0.68 | 0.60 | 0.66 | 0.08 | 0.03 |
| Gluestick [24] | ✗ | 0.36 | 0.47 | 0.45 | 0.30 | 0.51 | 0.29 | 0.31 | 0.22 | 0.09 |
| Ours | ✗ | 0.77 | 0.72 | 0.75 | 0.77 | 0.75 | 0.72 | 0.77 | **0.05** | **0.02** |

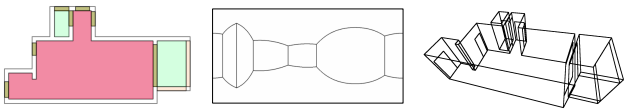
Table B.4. Pose refinement evaluation under varying lighting conditions in OmniScenes [14]. We report the localization accuracy at 0.1m and 5°, along with their range and standard deviations. Note we test two levels of variations for each lighting change type, totalling six variations.

arately apply GoMatch. We set $N_{\text{split}} = 8$ in all our experiments, as this attained the highest performance. For optimization-based methods, first Gao et al. [7] matches lines geometrically by inspecting the amount of line overlaps and then refines the matches by aligning line midpoint distances and directions. PDF minimization is a conceived baseline to test if point distance functions can be further used for refining poses. Here we apply gradient descent optimization on the cost function from Equation 6, using Adam [17] with a step size of 0.1. Similar to pose search evaluation, the line image-based methods considered here also take the line images in 2D and 3D, and apply visual descriptor-based matching to obtain the refined pose.

## D. Details on Experiment Setup

**Lighting Robustness Evaluation** We elaborate on the details of the lighting changes in Section 5.2, which are used to evaluate the illumination robustness of our method. As shown in Table B.4, we apply three types of color variations to images in the entire Extreme split of OmniScenes [14]: average intensity, gamma, and white balance change. For average intensity, we lower the pixel intensity by 25% (Intensity$^{1}$) and 33% (Intensity$^{2}$). For gamma, we test image gamma values with 0.3 (Gamma$^{1}$) and 1.5 (Gamma$^{2}$). For white balance, we apply the following linear transformations to the RGB values: $\begin{pmatrix} 0.9 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.7 \end{pmatrix}$ (White Balance$^{1}$), $\begin{pmatrix} 0.6 & 0 & 0 \\ 0 & 0.9 & 0 \\ 0 & 0 & 0.4 \end{pmatrix}$ (White Balance$^{2}$).

**Floorplan Localization Evaluation** To test the generalizability of our method against sparser line maps, we



(a) Raw Floorplan (b) 2D Floorplan Lines (c) 3D Floorplan Lines

Figure D.2. Visualization of lines in 2D and 3D extracted from the floorplans in Structured3D [30] dataset. Given the raw floorplan annotations and the room height information, we extract the 2D and 3D floorplan lines.

evaluated localization performance using floorplans in Section 5.4. Here we describe the details on the experiment setup. As shown in Figure D.2, from the raw floorplan annotations and known room height, we extract the 2D and 3D lines. Then, without any modifications in the hyperparameter setup, the lines are given as input to our localization pipeline. Despite any floorplan-specific tuning, our method performs competitively against the state-of-the-art methods, as demonstrated in Section 5.3.

## References


[1] Sérgio Agostinho, João Pedro Gomes, and Alessio Del Bue. Cvxpnpl: A unified convex solution to the absolute pose estimation problem from point and line correspondences. *Journal of Mathematical Imaging and Vision*, 65:492–512, 2019. 4

[2] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017. 1, 4, 5

[3] Gabriele Berton, Carlo Masone, and Barbara Caputo. Rethinking visual geo-localization for large-scale applications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4878–4888, 2022. 4

[4] Dylan Campbell, Liu Liu, and Stephen Gould. Solving the blind perspective-n-point problem end-to-end with robust differentiable geometric optimization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, page preprint. Springer, 2020. 4, 5

[5] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPR Deep Learning for Visual SLAM Workshop*, 2018. 4

[6] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981. 4

[7] Shuang Gao, Jixiang Wan, Yishan Ping, Xudong Zhang, Shuzhou Dong, Jijunnan Li, and Yandong Guo. Pose refinement with joint optimization of visual points and lines. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2888–2894, 2021. 4, 5

[8] Yixiao Ge, Haibo Wang, Feng Zhu, Rui Zhao, and Hongsheng Li. Self-supervising fine-grained region similarities for large-scale image localization. In *European Conference on Computer Vision*, 2020. 4

[9] Lee-Ad Gottlieb and Shay Solomon. Light spanners for snowflake metrics. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, page 387–395, New York, NY, USA, 2014. Association for Computing Machinery. 2

[10] Rafael Grompone von Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4): 722–732, 2010. 1, 3, 4

[11] Peter Hall. On representatives of subsets. *Journal of The London Mathematical Society-second Series*, pages 26–30, 1935. 2

[12] J. A. Hesch and S. I. Roumeliotis. A direct least-squares (dls) method for pnp. In *2011 International Conference on Computer Vision*, pages 383–390, 2011. 4

[13] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 32(5):922–923, 1976. 3

[14] Junho Kim, Changwoon Choi, Hojun Jang, and Young Min Kim. Piccolo: Point cloud-centric omnidirectional localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3313–3323, 2021. 1, 3, 4, 5

[15] Junho Kim, Hojun Jang, Changwoon Choi, and Young Min Kim. Cpo: Change robust panorama to point cloud localization. ECCV, 2022. 4, 5

[16] Junho Kim, Changwoon Choi, Hojun Jang, and Young Min Kim. Ldl: Line distance functions for panoramic localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17882–17892, 2023. 1, 2, 3, 4

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 3, 5

[18] William Leeb. Approximating snowflake metrics by trees. *Applied and Computational Harmonic Analysis*, 45(2):405–424, 2018. 2

[19] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *Int. J. Comput. Vision*, 81(2):155–166, 2009. 4

[20] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. LightGlue: Local Feature Matching at Light Speed. In *ICCV*, 2023. 4, 5

[21] Branislav Micusik and Horst Wildenauer. Descriptor free visual indoor localization with line segments. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3165–3173, 2015. 4

[22] Rémi Pautrat, Daniel Barath, Viktor Larsson, Martin R. Oswald, and Marc Pollefeys. Deeplsd: Line segment detection and refinement with deep image gradients. In *Computer Vision and Pattern Recognition (CVPR)*, 2023. 4

[23] Rémi Pautrat, Shaohui Liu, Petr Hruby, Marc Pollefeys, and Daniel Barath. Vanishing point estimation in uncalibrated images with prior gravity direction. In *International Conference on Computer Vision (ICCV)*, 2023. 1

[24] Rémi Pautrat, Iago Suárez, Yifan Yu, Marc Pollefeys, and Viktor Larsson. GlueStick: Robust image matching by sticking points and lines together. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 4, 5

[25] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 4, 5

[26] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. *CVPR*, 2021. 4, 5

[27] Iago Suárez, José M. Buenaposada, and Luis Baumela. Elsed: Enhanced line segment drawing. *Pattern Recognition*, 127:108619, 2022. 4

[28] Lu Xiaohu, Liu Yahui, and Li Kai. Fast 3d line segment detection from unorganized point cloud. *arXiv preprint arXiv:1901.02532*, 2019. 1

[29] 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. 4, 5

[30] Jia Zheng, Junfei Zhang, Jing Li, Rui Tang, Shenghua Gao, and Zihan Zhou. Structured3d: A large photo-realistic dataset for structured 3d modeling. In *Proceedings of The European Conference on Computer Vision (ECCV)*, 2020. 6

[31] Qunjie Zhou, Sérgio Agostinho, Aljoša Ošep, and Laura Leal-Taixé. Is geometry enough for matching in visual localization? In *Computer Vision – ECCV 2022*, pages 407–425, Cham, 2022. Springer Nature Switzerland. 4, 5