Active Exploration for Sparse Visual Localization

Supplementary Material

A. Pose-to-Pixel Uncertainty Propagation

In this section, we provide additional details on how we propagate pose uncertainty. First, the Jacobian of the projection with respect to the translation is calculated by the following formula, where (u, v) are pixel coordinates.

$$J_t = \frac{\partial(u,v)}{\partial \mathbf{t}} = \frac{\partial(u,v)}{\partial(x_d,y_d)}\frac{\partial(x_d,y_d)}{\partial(x,y)}\frac{\partial(x,y)}{\partial \mathbf{X}_c}\frac{\partial \mathbf{X}_c}{\partial \mathbf{t}} \tag{6}$$

where

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} = \begin{bmatrix} f_x x_d + c_x \\ f_y y_d + c_y \\ 1 \end{bmatrix} \tag{7}$$

with f_x, f_y as the focal lengths and (c_x, c_y) as the principal point. Furthermore,

$$x_d = g(x,y), \quad y_d = h(x,y) \tag{8}$$

are the distorted projected coordinates, where we assume g, h are polynomial functions. Moreover,

$$x = \frac{X_c}{Z_c}, y = \frac{Y_c}{Z_c} \tag{9}$$

are the projected points on the virtual image plane. Finally,

$$\mathbf{X}_c = \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \mathbf{R}\mathbf{X} + \mathbf{t} \tag{10}$$

is a 3D point in the camera coordinate system, transformed by the *world-to-camera* pose $(\mathbf{R}, \mathbf{t})$. Now, we see here that

$$\frac{\partial \mathbf{X}_c}{\partial \mathbf{t}} = \mathbf{I}_3. \tag{11}$$

We continue from Eq. (7) and write

$$\frac{\partial(u,v)}{\partial(x_d,y_d)} = \begin{bmatrix} \frac{\partial u}{\partial x_d} & \frac{\partial u}{\partial y_d} \\ \frac{\partial v}{\partial x_d} & \frac{\partial v}{\partial y_d} \end{bmatrix} = \begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix}. \tag{12}$$

Additionally, from Eq. (8) we get

$$\begin{aligned} \frac{\partial(x_d,y_d)}{\partial(x,y)} &= \begin{bmatrix} \frac{\partial x_d}{\partial x} & \frac{\partial x_d}{\partial y} \\ \frac{\partial y_d}{\partial x} & \frac{\partial y_d}{\partial y} \end{bmatrix} \\ &= \begin{bmatrix} g_x(x,y) & g_y(x,y) \\ h_x(x,y) & h_y(x,y) \end{bmatrix}. \end{aligned} \tag{13}$$

Continuing from Eq. (9), we obtain

$$\begin{aligned} \frac{\partial(x,y)}{\partial \mathbf{X}_c} &= \begin{bmatrix} \frac{\partial x}{\partial X_c} & \frac{\partial x}{\partial Y_c} & \frac{\partial x}{\partial Z_c} \\ \frac{\partial y}{\partial X_c} & \frac{\partial y}{\partial Y_c} & \frac{\partial y}{\partial Z_c} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{Z_c} & 0 & -\frac{X_c}{Z_c^2} \\ 0 & \frac{1}{Z_c} & -\frac{Y_c}{Z_c^2} \end{bmatrix}. \end{aligned} \tag{14}$$

Thus, in total, we get

$$\begin{aligned} J_t = \frac{\partial(u,v)}{\partial \mathbf{t}} = \frac{\partial(u,v)}{\partial(x_d,y_d)}\frac{\partial(x_d,y_d)}{\partial(x,y)}\frac{\partial(x,y)}{\partial \mathbf{X}_c}\frac{\partial \mathbf{X}_c}{\partial \mathbf{t}} = \\ \begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix} \begin{bmatrix} g_x(x,y) & g_y(x,y) \\ h_x(x,y) & h_y(x,y) \end{bmatrix} \begin{bmatrix} \frac{1}{Z_c} & 0 & -\frac{X_c}{Z_c^2} \\ 0 & \frac{1}{Z_c} & -\frac{Y_c}{Z_c^2} \end{bmatrix} \end{aligned} \tag{15}$$

which can easily be implemented. In the case of a pinhole camera, the middle matrix becomes identity.

For the rotation we get a similar expression:

$$J_r = \frac{\partial(u,v)}{\partial \mathbf{R}} = \frac{\partial(u,v)}{\partial(x_d,y_d)}\frac{\partial(x_d,y_d)}{\partial(x,y)}\frac{\partial(x,y)}{\partial \mathbf{X}_c}\frac{\partial \mathbf{X}_c}{\partial \mathbf{R}} \tag{16}$$

where we use the exponential map as a local parametrization of rotations. We write

$$\mathbf{R} = \mathbf{R_0} e^{[\delta]_\times} \tag{17}$$

with ($[\cdot]_\times$ denotes the cross-product matrix)

$$\mathbf{R}(\delta) \approx \mathbf{R_0}(\mathbf{I} + [\delta]_\times) \tag{18}$$

giving

$$\frac{\partial \mathbf{X}_c}{\partial \mathbf{R}} = \nabla_\delta(\mathbf{R_0}(\mathbf{I} + [\delta]_\times)\mathbf{X} + \mathbf{t}) = \tag{19}$$

$$\nabla_\delta \mathbf{R_0}[\delta]_\times \mathbf{X} = -\nabla_\delta \mathbf{R_0}[\mathbf{X}]_\times \delta = -\mathbf{R_0}[\mathbf{X}]_\times \tag{20}$$

In summary, the rotational part of the Jacobian becomes:

$$J_r = \frac{\partial(u,v)}{\partial \mathbf{R}} = -\frac{\partial(u,v)}{\partial(x_d,y_d)}\frac{\partial(x_d,y_d)}{\partial(x,y)}\frac{\partial(x,y)}{\partial \mathbf{X}_c}\mathbf{R_0}[\mathbf{X}]_\times \tag{21}$$

To combine the two components to a single Jacobian (assuming no cross-covariances and isotropic covariances), we get:

$$JJ^T = \sigma_R^2 J_r J_r^T + \sigma_t^2 J_t J_t^T \tag{22}$$

We conducted ablations using the full projected pose uncertainty $\Sigma_{R,t}$, with $\sigma_R = \sigma_t$. Results are shown in Tab. 8 (Bottom). In practice, we found that only using the translation uncertainty worked well.

Method	25cm,2°	50cm,5°	100cm,10°
No refinement (MNN)	**18.9**	**23.9**	**27.7**
↳ SuperGlue matching	16.5	22.0	26.0
Refinement with Σ_t	**20.3**	**26.5**	**30.3**
↳ with $\Sigma_{R,t}$	19.8	25.8	29.4

Table 8. **Ablation on Oxford Night validation**. *Top:* Comparing mutual nearest neighbour matcher (MNN) with SuperGlue. *Bottom:* Impact of full pose uncertainty in the refinement step.

B. Incorporating a Learning-Based Matcher

Current state-of-the-art matchers focus on 2D-2D matching and are not directly applicable to our 2D-3D setting (as the positional encoding is trained for 2D points). We experimented with projecting the expanded point cloud into the retrieved reference view, followed by SuperGlue matching on the projections (see Tab. 8 (Top)). While it works reasonably well, it is still inferior to standard mutual nearest-neighbour matching. We believe this behaviour is likely due to a distribution mismatch with the training data, affecting both the keypoints (*e.g.*, inclusion of non-visible or occluded points) and the descriptors (which are averaged across multiple images). We think that re-training the matcher specifically for the 2D-3D setting (without the projection) is a promising future work.

C. RobotCar per Camera per Location

For completeness and easier comparison with other papers, we also report results when the image retrieval is performed per-location instead of against the entire map, i.e. only retrieving images in the same region as the query image. The results are shown in Tab. 9.

We compare HLOC+SuperGlue to FUSELOC and our method with SuperPoint and NetVLAD descriptors. Our method and HLOC clearly outperform FUSELOC on night queries. However, this scenario is not well-suited for Active Exploration, as the per-location retrieval already limits the 3D search space significantly. This likely explains why HLOC performs better than our method in this case.

Method	**Day (%)**	**Night (%)**
HLOC SuperGlue	**64.2 / 94.3** / 99.9	**40.1 / 78.8 / 92.8**
FUSELOC	63.3 / 93.9 / **100.0**	15.2 / 39.9 / 60.1
AE (ours)	63.9 / 94.2 / 99.9	32.4 / 69.5 / 89.5

Table 9. **Results Robotcar per Camera per Location** with SuperPoint and NetVLAD descriptors. Localization recall (%) at three thresholds (0.25 m, 2°) / (0.5 m, 5°) / (5 m, 10°) for day and nighttime query images.

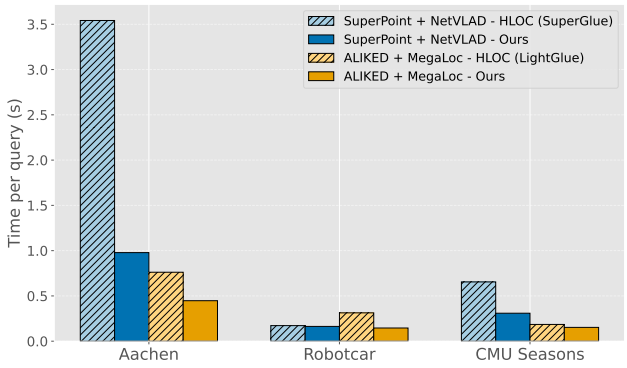

Figure 9. **Localization Times per Query** for three outdoor datasets. For HLOC, the descriptors used for SuperGlue [26] are SuperPoint [8] and NetVLAD [1], and for LightGlue we use ALIKED [36] and MegaLoc [4]. Naturally, we show the performance of Active Exploration with the same descriptors. Our method is consistently faster on the given datasets. The corresponding results are shown in Tab. 6.

D. Runtime Comparison

In Fig. 9 we provide full runtime comparisons, including HLOC with SuperPoint+SuperGlue. As expected, our method is also faster compared to HLOC with SuperGlue.

E. Additional Experiment Details

Here we provide additional details for the experiments in Sec. 4.3. For a fair comparison with HLOC, we use the same number of evaluated images in our search as the number of retrieved images in HLOC. For Aachen, this corresponds to 50, for RobotCar to 20, and for CMU to 10. Gangnam Station is not currently implemented in HLOC, so here we have chosen $k = 10$. For global and local descriptors as well as the SfM model generation we follow the configuration used in HLOC. The hyperparameters were set to $p_{ij} = 0.5$ and $e_{ij} = 0.5$ across all experiments.