Geometry-Aware Learning of Maps for Camera Localization Supplementary Material

Samarth Brahmbhatt[1] Jinwei Gu[2] Kihwan Kim[2] James Hays[1] Jan Kautz[2]
[1]{samarth.robo, hays}@gatech.edu [2]{jinweig, kihwank, jkautz}@nvidia.com
[1]Georgia Institute of Technology [2]NVIDIA

Abstract

In this supplementary document, we provide more implementation details of our method, names of the sequences used in the experiments on the Oxford RobotCar dataset, more analysis and visualization of the experimental results presented in the main paper, and the detailed derivation of pose-graph optimization (PGO) in MapNet+PGO. Please also refer to the supplementary video for more visualizations of the camera localization results.

1. Pose Graph Optimization in MapNet+PGO

The purpose of pose-graph optimization (PGO) is to refine the input poses such that the refined poses are close to the input poses (from MapNet+), and the relative transforms between the refined poses agree with the input visual odometries. It is an iterative optimization process [2, 3].

Inputs Pose predictions $\{\mathbf{p_i}\}_{i=1}^{T}$ and visual odometry (VOs) $\hat{\mathbf{v}}_{ij}$ between consecutive poses. Both poses and VOs are 6-dimensional (3d translation $\mathbf{t}$ + 3d log quaternion $\mathbf{w}$). For the rest of the algorithm, the log quaternions are converted to unit quaternion using the exponential map [4]:

$$\mathbf{q} = (\cos\|\mathbf{w}\|, \frac{\mathbf{w}}{\|\mathbf{w}\|}\sin\|\mathbf{w}\|) \tag{1}$$

Objective Function State vector z is the concatenation of all T pose vectors. The total objective function is the sum of the costs of all constraints. The constraints can be either for the absolute pose or for the relative pose between a pair of poses. For both of these categories, there are separate constraints for translation and rotation.

$$\begin{aligned} E(z) &= \sum_c E_c(z) \\ &= \sum_c \bar{h}(f_c(c), k_c) \end{aligned} \tag{2}$$

where $\bar{h}(\cdot)$ is the pose distance function from Equation (7) of the main paper. f_c is a function that maps the state vector to the quantity relevant for the constraint c. For example, it selects $\mathbf{p}_i$ from the state vector for a constraint on the absolute pose, or computes the VO between poses for $\mathbf{p}_i$ and $\mathbf{p}_j$ for a constraint on the relative pose. k_c is the observation for that constraint, and remains constant throughout the optimization process. For example:

- For the absolute pose constraints, k_c is the MapNet+ prediction.
- For the relative pose constraints, k_c is the input VO $\hat{\mathbf{v}}_{ij}$.

Following [2, 3], we define $\bar{h}(\cdot)$ as:

$$\bar{h}(f_c(c), k_c) = (f_c(z) - k_c)^T S_c (f_c(z) - k_c) \tag{3}$$

where S_c the covariance matrix for the constraint.

Optimization Following [2], We first linearize f_c around $\bar{z}$, the current value of z:

$$f_c(\bar{z} + \Delta z) \approx f_c(\bar{z}) + \left.\frac{\partial f_c}{\partial z}\right|_{z=\bar{z}} \Delta z \tag{4}$$

and take the Cholesky decomposition of S_c: $S_c = L_c L_c^T$. Hence the linearized objective function becomes:

$$\begin{aligned} E(\Delta z) &= \sum_c (f_c(\bar{z} + \Delta z) - k_c)^T S_c (f_c(\bar{z} + \Delta z) - k_c) \\ &\approx \sum_c \left(f_c(\bar{z}) + \left.\frac{\partial f_c}{\partial z}\right|_{z=\bar{z}} \Delta z - k_c \right)^T L_c \\ &\quad L_c^T \left(f_c(\bar{z}) + \left.\frac{\partial f_c}{\partial z}\right|_{z=\bar{z}} \Delta z - k_c \right) \\ &= \sum_c \left|\left| L_c^T \left(f_c(\bar{z}) + \left.\frac{\partial f_c}{\partial z}\right|_{z=\bar{z}} \Delta z - k_c \right) \right|\right|^2 \\ &= \sum_c ||J_c \Delta z - r_c||^2 \end{aligned} \tag{5}$$

where **Jacobian** $J_c = L_c^T \frac{\partial f_c}{\partial z}\Big|_{z=\bar{z}}$ and **residue** $r_c = L_c^T(k_c - f_c(\bar{z}))$. We will solve for Δz.

Stacking the individual Jacobians and residuals vertically, we arrive at the least squares problem:

$$\Delta z^* = \min_{\Delta z} ||J\Delta z - r||^2 \tag{6}$$

This can be solved by $\Delta z^* = (J^T J)^{-1} J^T r$.

Finally, we update the state vector:

$$z = z \boxplus \Delta z \tag{7}$$

where $\boxplus$ is the manifold update operation, needed because of the quaternions (more details below).

Detour: Manifolds for Quaternion Update As mentioned in [3], if we had used a simple addition in the update Equation (7), it would have broken the constraints introduced by the over-parameterization of quaternions. So we use manifolds. According to [3], "A mainfold is a space that is not necessarily Euclidean in a global scale, but can be seen as Euclidean on a local scale". The idea is to calculate the update for quaternion in a minimal 3d representation, and then apply this update to the 4d representation of quaterinion in z using $\boxplus$. We use the "exponential map" [4] to implement $\boxplus$. For this, we re-cast the objective function as a function of the update on the manifold, $\Delta\check{z}$:

$$E(\Delta\check{z}) = \sum_c (f_c(\bar{z}\boxplus\Delta\check{z}) - k_c)^T S_c (f_c(\bar{z}\boxplus\Delta\check{z}) - k_c) \tag{8}$$

The linearization step is:

$$f_c(\bar{z} \boxplus \Delta\check{z}) \approx f_c(\bar{z}) + \frac{\partial f_c}{\partial z}\Big|_{z=\bar{z}} \frac{\partial \bar{z} \boxplus \Delta\check{z}}{\partial \Delta\check{z}}\Big|_{\Delta\check{z}=0} \Delta\check{z} \tag{9}$$

So the Jacobian in this case is:

$$\check{J}_c = J_c \frac{\partial \bar{z} \boxplus \Delta\check{z}}{\partial \Delta\check{z}}\Big|_{\Delta\check{z}=0} \tag{10}$$

Let us see how $\bar{z} \boxplus \Delta\check{z}$ is implemented.

$$\bar{z} \boxplus \Delta\check{z} = \bar{z} \cdot \Delta\tilde{z} \tag{11}$$

where $\Delta\tilde{z}$ is the normal 4d quaternion that has been created from the 3d minimal representation $\Delta\check{z}$ using the exponential map (Equation (1)). So the derivative of the exponential map at $\Delta\check{z} = 0$ is $M_e = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. Hence,

$$\begin{aligned} \frac{\partial \bar{z} \boxplus \Delta\check{z}}{\partial \Delta\check{z}}\Big|_{\Delta\check{z}=0} &= \frac{\partial \bar{z} \cdot \Delta\tilde{z}}{\partial \Delta\tilde{z}} \frac{\partial \Delta\tilde{z}}{\partial \Delta\check{z}}\Big|_{\Delta\check{z}=0} \\ &= \frac{\partial \bar{z} \cdot \Delta\tilde{z}}{\partial \Delta\tilde{z}} M_e \end{aligned} \tag{12}$$

For the first term, we use the formula for derivative of quaternion product from [5].

Jacobian of Absolute Translation Constraint f_c just selects the appropriate 3 translation elements of a pose from the state vector z, so $\check{J}_c = L_c^T[\mathbf{0}, \ldots, I_3, \ldots, \mathbf{0}]$.

Jacobian of Absolute Rotation Constraint f_c selects the appropriate 4 quaternion elements of a pose from the state vector z. However, since the update is on the manifold, the Jacobian $\check{J}_c$ is computed as shown in Equations (10) and (12) with $J_c = L_c^T \cdot I_4$.

Jacobian of Relative Translation Constraint f_c computes the translation component of the VO $\mathbf{v}_{ij}$ between $\mathbf{p}_i$ and $\mathbf{p}_j$, which is $\mathbf{q}_j(\mathbf{t}_i - \mathbf{t}_j)\mathbf{q}_j^{-1}$ according to Equation (6) in the main paper. Hence J_c has $\frac{\partial \mathbf{q}_j \mathbf{t}_i \mathbf{q}_j^{-1}}{\partial \mathbf{t}_i}$ in the block corresponding to $\mathbf{t}_i$ and $-\frac{\partial \mathbf{q}_j \mathbf{t}_j \mathbf{q}_j^{-1}}{\partial \mathbf{t}_j}$ in the block corresponding to $\mathbf{t}_j$. Both these formulae can be found in [5].

Jacobian of Relative Rotation Constraint f_c computes the rotation component of the VO $\mathbf{v}_{ij}$ between $\mathbf{p}_i$ and $\mathbf{p}_j$, which is $q_j^{-1} \cdot q_i$ according to Equation (6) in the main paper. Hence J_c has $\frac{\partial \mathbf{q}_j^{-1} \cdot \mathbf{q}_i}{\partial \mathbf{q}_i}$ in the Jacobian block corresponding to $\mathbf{q}_i$ and $\frac{\partial \mathbf{q}_j^{-1} \cdot \mathbf{q}_i}{\partial \mathbf{q}_j}$ in the Jacobian block corresponding to $\mathbf{q}_j$. Both these formulae can be found in [5].

Update on the Manifold The updates for translation parts of the state vector are performed by simply adding the update vector to the state vector. For the quaternion parts, the minimal representations in the update need to be converted back to the 4d representation using the exponential map in Equation (1), and then quaternion-multiplied to the state vector quaternions.

Implementation Details The covariance matrix S_c is set to identity for all the translation constraints and tuned to σI_3 (σ=10 to 35) for different scenes in the 7-Scenes dataset. For the RobotCar dataset, we use $\sigma = 20$ for LOOP and $\sigma = 10$ for FULL.

2. Details of Image Pair Sampling

In both MapNet and MapNet+ (Sections 3.2 and 3.3 of the main paper) training, we need to sample image pairs $(\mathbf{I}_i, \mathbf{I}_j)$ from each input image sequence. This sampling is done within each tuple of s images sampled with a gap k frames. More specifically, suppose we have N images in an input sequence, $\mathbf{I}_1, \cdots, \mathbf{I}_N$. Each entry in each minibatch during the training of MapNet and MapNet+ consists of a tuple of s conseutive images that are k frames apart from each other, *i.e.*, $(\mathbf{I}_i, \mathbf{I}_{i+k}, \cdots, \mathbf{I}_{i+k(s-2)}, \mathbf{I}_{i+k(s-1)})$.

Table 1: Statistics of state-of-the-art methods on the 7-Scenes dataset.

Scene	PoseNet+log $\mathbf{q}$	DSO [1]	MapNet	MapNet+	MapNet+PGO
Avg Median (Scene)	0.23m, 8.49	0.51m, 29.44	0.21m, 7.77	0.19m, 7.29	**0.18m, 6.55**
Avg Median (Seq)	0.24m, 7.40	0.93m, 39.20	0.22m, 6.88	**0.20m**, 6.18	0.21m, **6.16**
Avg Mean (Scene)	0.28m, 10.43	1.27m, 46.48	0.27m, 10.08	0.23m, 8.27	**0.22m, 7.89**
Avg Mean (Seq)	0.30m, 9.84	1.62m, 40.28	0.28m, 9.12	0.24m, 7.42	**0.23m, 7.29**

Within this tuple of s images, each two neighboring elements will form an image pair for training. For example, both $(\mathbf{I}_i, \mathbf{I}_{i+k})$ and $(\mathbf{I}_{i+k(s-2)}, \mathbf{I}_{i+k(s-1)})$ are valid image pairs.

3. Details of the Sequences used in the Experiments on the RobotCar Dataset

Sequences in RobotCar are named by the date and time of their capture.

Experiments on the LOOP Scene To train the baseline PoseNet and MapNet, we used the following two sequences as the dataset $\mathcal{D}$ with ground truth supervision.

- 2014-06-26-09-24-58
- 2014-06-26-08-53-56

We used the following two sequences as the unlabeled dataset $\mathcal{T}$ to train MapNet+

- 2014-05-14-13-50-20
- 2014-05-14-13-46-12

MapNet+(1seq) used the first sequence in $\mathcal{T}$, and MapNet+(2seq) used both sequences in $\mathcal{T}$. These two sequences are also used in MapNet+(GPS) for updating the MapNet with GPS measurements.

We used the following sequences for testing, which are completely separated from all the sequences in $\mathcal{D}$ and $\mathcal{T}$.

- 2014-06-23-15-36-04
- 2014-06-23-15-41-25

Figure 5 in our main paper showed the testing results on 2014-06-23-15-41-25 for visualization (we obtained similar results on the other testing sequence).

Figure 8 of the main paper: The MapNet+ model trained with one sequence of labeled data used $\mathcal{D} = \{$2014-06-26-09-24-58$\}$ and increasingly larger subsets of unlabeled data $\mathcal{T} = \{$2014-06-26-08-53-56, 2014-05-14-13-50-20, 2014-05-14-13-46-12$\}$. The MapNet+ model trained with 2 sequences of labeled data used $\mathcal{D} = \{$2014-06-26-09-24-58, 2014-06-26-08-53-56$\}$ and increasingly larger subsets of unlabeled data $\mathcal{T} = \{$2014-05-14-13-50-20, 2014-05-14-13-46-12$\}$. All these models were tested on 2014-06-23-15-36-04.

Experiments on the FULL Scene To train the baseline PoseNet and MapNet, we used the following two sequences as the labeled dataset $\mathcal{D}$

- 2014-11-28-12-07-13
- 2014-12-02-15-30-08

We used the following sequence as the unlabeled dataset $\mathcal{T}$

- 2014-12-12-10-45-15

We used the following sequence for testing, which is completely separated from all the learning methods

- 2014-12-09-13-21-02

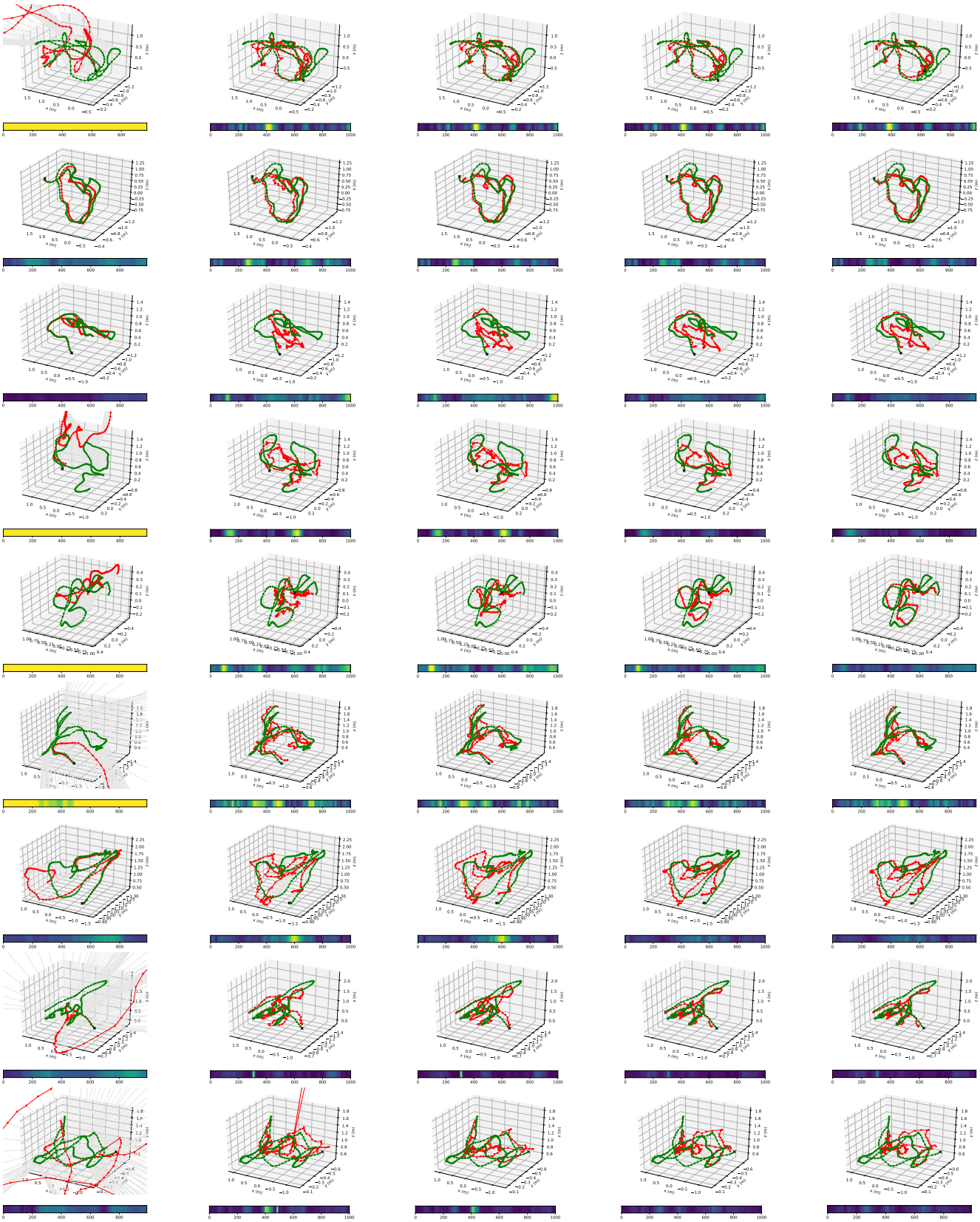
4. Experiments on the 7-Scenes Dataset

Figure 1 and Figure 2 show the results for all the 18 testing sequences on the 7-Scenes dataset. Table 1 lists a variety of statistics computed on all the 18 testing sequences, where Avg Median (Scene) means the averaged values of the median error over each scene, and Avg Median (Seq) means the averaged values of the median error over each sequence in the scene. As shown, both these two figures and the table support the same conclusion as described in the main paper.

5. Experiments on the RobotCar Dataset

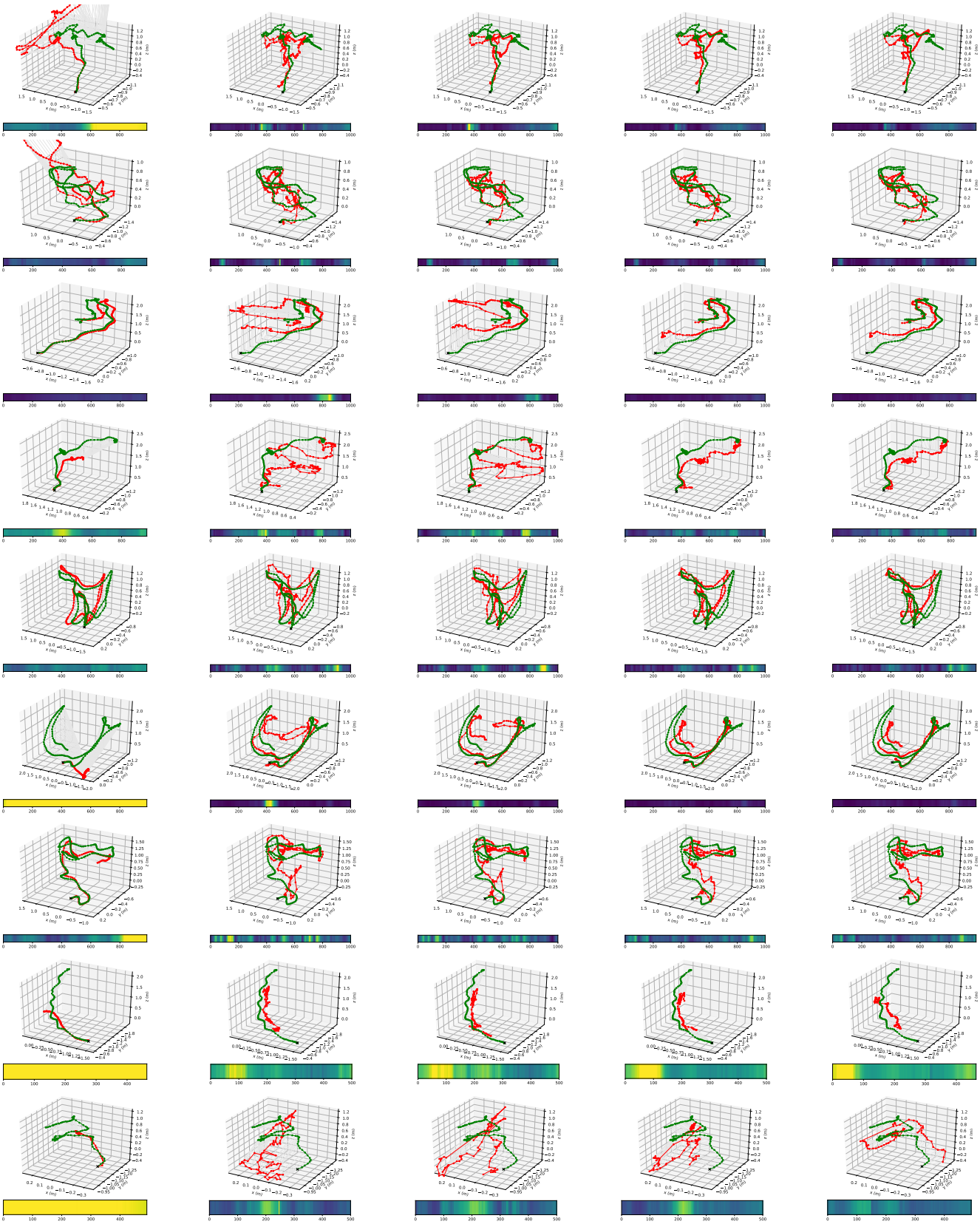
Figure 3 shows the images corresponding to the outliers in camera localization results of MapNet+PGO for both the LOOP scene and the FULL scene. As shown, these outliers often correspond to images with large over-exposed regions, or large regions covered with moving objects (*e.g.*, truck). Some of these outliers can be filtered out simply with temporal median filtering, as shown in Figure 4.

We also computed saliency maps $s(x, y) = \frac{1}{6}|\sum_{i=1}^{6} \frac{\partial p_i}{\partial I(x,y)}|$ (magnitude gradient of the mean of the 6-element output w.r.t. input image, maxed over the 3 color channels) of PoseNet and MapNet+ on both the 7-scenes and RobotCar dataset (`redkitchen` and `loop` sequences). As shown in Fig 5, compared to PoseNet, MapNet+ focuses more on geometrically meaningful regions and its saliency map is more consistent over time.



DSO [1] PoseNet [7, 8, 6] MapNet MapNet+ MapNet+PGO

Figure 1: **Results on the 7-Scenes dataset.** The 3d plots show the camera position (green for ground truth and red for predictions). The colorbars below show the errors of the predicted camera orientation (blue for small error and yellow for large error) with frame number on the X axis. From top to bottom are testing sequences: Chess-Seq-03, Chess-Seq-05, Fire-Seq-03, Fire-Seq-04, Head-Seq-01, Office-Seq-02, Office-Seq-06, Office-Seq-07, and Office-Seq-09.



DSO [1] PoseNet [7, 8, 6] MapNet MapNet+ MapNet+PGO

Figure 2: **Results on the 7-Scenes dataset (continued).** The 3d plots show the camera position (green for ground truth and red for predictions). The colorbars below show the errors of the predicted camera orientation (blue for small error and yellow for large error) with frame number on the X axis. From top to bottom are testing sequences: Pumpkin-Seq-01, Pumpkin-Seq-07, Redkitchen-Seq-03, Redkitchen-Seq-04, Redkitchen-Seq-06, Redkitchen-Seq-12, Redkitchen-Seq-14, Stairs-Seq-01, and Stairs-Seq-04.

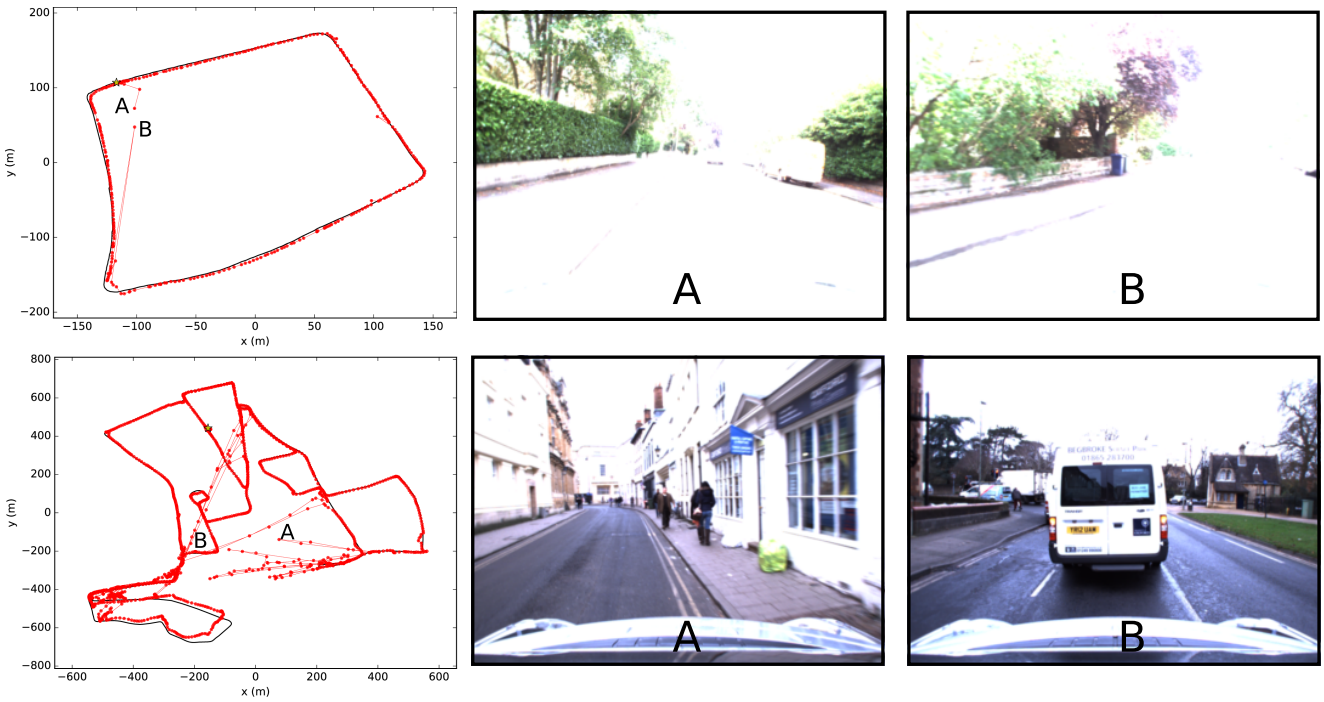
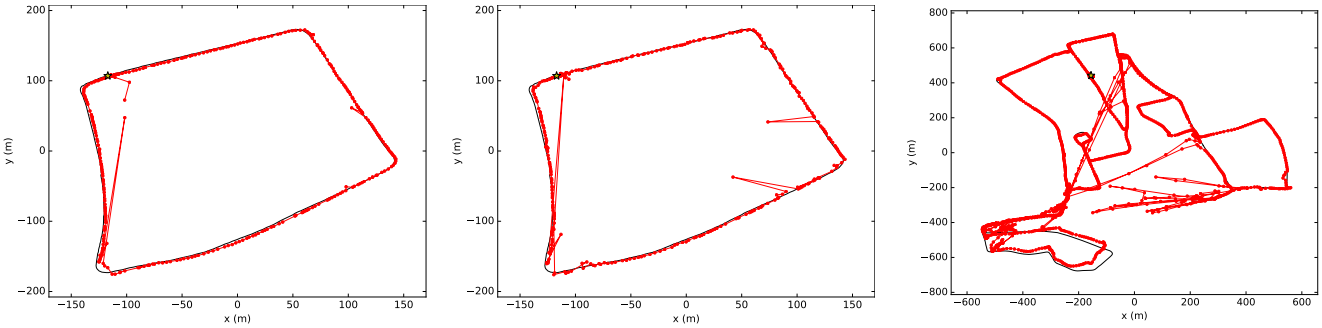
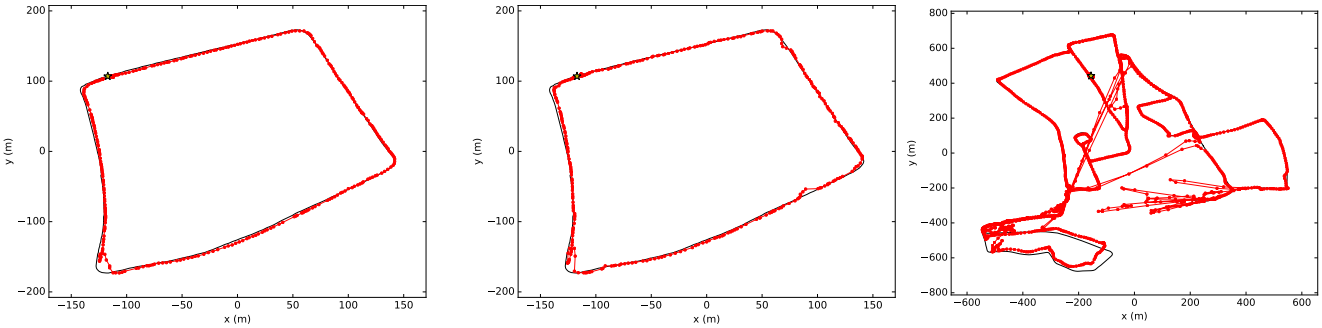


Figure 3: **Images corresponding to the spurious estimation of MapNet+PGO for the LOOP scene (top) and the FULL scene (bottom)**. These outliers usually corresponds to images with large over-exposed regions, or large regions on moving objects (*e.g.*, truck), which often can be filtered out with simple temporal median filtering (see Figure 4).



MapNet+PGO (6.73m, 2.23°) MapNet+(GPS) (6.78m, 2.72°) MapNet+PGO (29.5m, 7.8°)



MapNet+PGO (5.74m, 2.23°) MapNet+(GPS) (4.95m, 2.72°) MapNet+PGO (29.2m, 7.8°)

Figure 4: **Camera localization results before (TOP) and after (BOTTOM) temporal median filtering.** The spurious estimations can be effectively removed with a simple median filtering (with the window size of 51 frames).

Figure 5: Attention maps for example images from the 7 Scenes dataset (top) and RobotCar dataset (bottom). In all 4 examples, we observe that MapNet+ focuses more on geometrically meaningful regions compared to PoseNet, and its saliency map is more consistent over time. Please see videos at http://youtu.be/197N30A9RdE to observe temporal consistency and more example frames.

References

[1] J. Engel, V. Koltun, and D. Cremers. DSO: Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017. 3, 4, 5

[2] M. K. Grimes, D. Anguelov, and Y. LeCun. Hybrid hessians for flexible optimization of pose graphs. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2997–3004. IEEE, 2010. 1

[3] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010. 1, 2

[4] C. Hertzberg. A framework for sparse, non-linear least squares problems on manifolds, 2008. 1, 2

[5] Y.-B. Jia. Quaternion and Rotation. Com S 477/577 Lecture Notes at http://web.cs.iastate.edu/~cs577/handouts/quaternion.pdf, 2008. 2

[6] A. Kendall and R. Cipolla. Modeling uncertainty in deep learning for camera relocalization. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2016. 4, 5

[7] A. Kendall and R. Cipolla. Geometric loss functions for camera pose regression with deep learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 4, 5

[8] A. Kendall, M. Grimes, and R. Cipolla. PoseNet: A convolutional network for real-time 6-DOF camera relocalization. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015. 4, 5