MapNet: Geometry-Aware Learning of Maps for Camera Localization

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

*Maps are a key component in image-based camera localization and visual SLAM systems: they are used to establish geometric constraints between images, correct drift in relative pose estimation, and relocalize cameras after lost tracking. The exact definitions of maps, however, are often application-specific and hand-crafted for different scenarios (*e.g. *3D landmarks, lines, planes, bags of visual words). We propose to represent maps as a deep neural net called MapNet, which enables learning a data-driven map representation. Unlike prior work on learning maps, MapNet exploits cheap and ubiquitous sensory inputs like visual odometry and GPS in addition to images and fuses them together for camera localization. Geometric constraints expressed by these inputs, which have traditionally been used in bundle adjustment or pose-graph optimization, are formulated as loss terms in MapNet training and also used during inference. In addition to directly improving localization accuracy, this allows us to update the MapNet (*i.e., *maps) in a self-supervised manner using additional unlabeled video sequences from the scene. We also propose a novel parameterization for camera rotation which is better suited for deep-learning based camera pose regression. Experimental results on both the indoor 7-Scenes dataset and the outdoor Oxford RobotCar dataset show significant performance improvement over prior work.*

1. Introduction

Camera localization *i.e.* recovering the 3D position and orientation of a moving camera is one of the fundamental tasks in computer vision with a wide variety of applications in robotics, autonomous driving, and AR/VR. A key component in camera localization, including various visual SLAM systems [19, 43, 57] and image-based localization methods [36, 47, 48] is the concept of a *map*. A map is an abstract summary of the input data that establishes geometric constraints between observations and can be queried to get the camera pose when tracking is drifting or lost. Maps, however, are usually defined in an application-specific manner with hand-crafted features. Examples include 3D landmarks for general visual SLAM methods [35, 36, 43], 3D lines and patches in semi-dense SLAM methods and indoor scenes [19, 44, 57], object-level context in semantic SLAM methods [10, 46], bag of visual word features on key frames for camera relocalization [12, 47, 48]. Being application-specific, these map representations may ignore useful (sometimes, the only available) features in environments they were not designed for, and are inflexible to update as more input data come in.

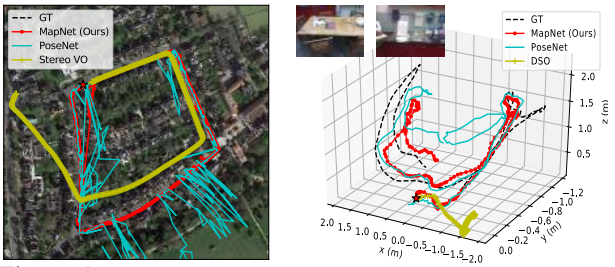


Figure 1: Camera localization results for outdoor (left) and indoor (right) scenes from the Oxford RobotCar [39] and 7-Scenes [49] datasets. As shown, prior DNN-based methods (*e.g.*, PoseNet [33, 31, 32]) result in noisy estimations, while traditional visual odometry based methods (*e.g.*, stereo VO or DSO [17]) often drift over time. In contrast, MapNet gives accurate camera pose estimates by including various geometric constraints in DNN training and inference.

Is there a general map representation for camera localization that addresses these drawbacks? In this paper, we take a step towards answering this question. We propose to represent maps as a DNN, called MapNet, which learns the map representation directly from input data, with the flexibility to fuse multiple sensory inputs and to improve over time using unlabeled data. MapNet aims to be a part that can be easily plugged into any visual SLAM or image-based localization systems. We are inspired by both the recent DNN-based camera localization work (*e.g.*, PoseNet [33] and its variants [11, 32, 41, 56]) in the context of structure-from-motion, as well as the traditional map optimization methods (*e.g.*, bundle adjustment (BA) [21, 22, 40], pose graph optimization (PGO) [8, 16, 38]) in the context of visual SLAM. Compared to these prior works, our approach makes the following contributions:

- Most prior DNNs for camera localization [33, 31, 32, 11, 41, 56] are trained using single images labelled with absolute camera pose. In MapNet we show how the geometric constraints between pairs of observations can be included as an additional loss term in training. These constraints can come from a variety of sources: pose constraint from the visual odometry (VO) between pairs of images, translation constraint from two GPS readings, rotation constraint from two IMU readings, *etc*. We call this *geometry-aware learning* and show that it significantly improves camera localization performance in Section 4.

- PoseNet and its variants are *offline* methods – the learned DNNs are fixed after training. In contrast, we propose MapNet+ that can use the geometric constraints between pairs of observations mentioned above to continuously update the DNN weights (*i.e*., maps) without absolute camera pose supervision, as additional unlabeled data come in. Moreover, at runtime, we also exploit the complementary noise characteristics of MapNet predictions (locally noisy but drift-free) and VO (locally smooth but drifty) by fusing them in a moving window fashion with PGO. We call this variant MapNet+PGO. We show in Section 4 that both MapNet+ and MapNet+PGO successively improve performance further.

- We propose a new parameterization for camera rotation, the logarithm of unit quaternion, which is better suited for deep-learning based camera pose regression. This improves the performance of PoseNet and MapNet, as shown in Table 2.

Figure 1 shows two examples of camera localization. Pure DNN-based methods (*e.g*., PoseNet [33, 31, 32]) result in noisy estimations, and the traditional VO-based methods (*e.g*., stereo VO or DSO [17]) often drift significantly over time. By incorporting the geometric constraints into DNN-based learning and inference, the proposed approach MapNet+PGO achieves the best result. We evaluate the proposed methods extensively on both the indoor 7-Scenes [49] and the outdoor Oxford RobotCar dataset [39].

2. Related Work

Maps in Visual SLAM and Image-based Localization Over the years, various types of map representations and their optimization techniques have been proposed for camera localization [53, 55]. In visual SLAM, 3D landmarks (with feature descriptors of the corresponding image patches) are often defined as maps in both Bayesian filtering approaches [13, 42, 54] and key-frame based approaches [35, 43, 50]. The features are often modeled in various forms such as points [13, 35], points and lines [20], points and planes [52], or built on more semantic (object) level features [10, 46]. However, the choice of the representation has been application-specific, and thus the performance can vary depending on a target scene (*i.e*., amount of lines, planes or texture present in the scene). To address this issue, recently, direct [28, 44] and semi-direct methods [17, 19] utilize all the pixels with high gradients rather than features to build maps. While they provide more stable pose estimate and denser information of a scene, they require a higher computational expense, and often need an accurate intrinsic calibration and initialization because they are sensitive to photometric consistency [5].

For map optimization, since Lu and Milios [38] first introduced a graph-based method to refine a map with global optimization of local nodes (measurements from odometry), various types of these local-to-global pose graph optimization methods have been proposed [14, 24, 53, 54]. Similarly, bundle adjustment [37, 55] has also been a popular choice for methods using structure from motion techniques (*i.e*., keyframe-based approaches) [35, 43, 50].

In the context of image-based localization, visual place recognition [47, 48, 36] and camera relocalization [43], image descriptors (*e.g*., bag-of-words (BoW) features [12], VLAD [4], Fisher vectors [29], and recent DNN-based features [3]) are used to build maps/vocabularies for image retrieval and pose estimation.

Compared to these prior application-specific map definitions, in this paper we primarily focus on learning a general map representation for sequential camera localization with deep neural networks, by leveraging statistical learning from big data and geometric constraints from pose graph optimization and bundle adjustment.

DNN-based Camera Localization A few recent works use deep neural networks for image-based localization in the context of structure-from-motion. PoseNet [33] first proposed to directly regress 6-DoF camera pose from an input image with GoogLeNet. Kendall *et al*. [31, 32] extend PoseNet by learning the weight between camera translation and rotation loss and incorporating the reprojection loss. Melekhov *et al*. [41] improved PoseNet with skip connections with ResNet34 architecture. Brachmann *et al*. [7] localize a camera in a dense 3D reconstruction by performing RANSAC on predicted 2D-3D correspondences. Recently, RNNs (*e.g*., LSTM) have been introduced to spatially [56] and temporally [11] improve camera localization.

MapNet is inspired by the PoseNet line of work, but has several major modifications. Table 1 shows a comparative summary. Clark *et al*. [11] used an LSTM to implicitly learn the temporal relationship between consecutive frames, but its performance is on-par or worse than prior methods [32, 41]. In contrast, MapNet uses single images as input during inference but still uses the geometric constraints between pairs as a meaningful learning signal for

Table 1: Comparison with prior DNN-based camera localization methods. Please refer to Section 2 for details.

	PoseNet [33, 31, 32]	Hourglass [41]	LSTM-Pose [56]	VidLoc [11]	(Proposed) MapNet	MapNet+	MapNet+PGO
Input	Images	Images	Images	Videos	Images	unlabeled videos + (VO, GPS, IMU)	
Fusion ability	No	No	No	No	No	Yes	Yes
Self-supervised update	No	No	No	No	No	Yes	No
Temporal constraint	No	No	No	Yes	Yes	Yes	Yes
Geometry aware	Reprojection [32]	No	No	No	Geometric constraints on camera poses		

training. MapNet+ and MapNet+PGO use unlabeled videos and multiple sensory input (*e.g.*, visual odometry, IMU, GPS) to further improve performance. Thus, they can fuse information from multiple modalities and improve in a self-supervised manner. In [32] Kendall *et al.* make PoseNet *scene*-geometry aware by minimizing the reprojection error of 3D points in multiple images. In contrast, MapNet is *camera motion*-geometry aware by utilizing the geometric constraints between camera poses.

3. Proposed Approach

In this paper, we learn a general map representation for sequential camera localization with deep neural networks (DNNs). Maps are represented as learned weights of a DNN trained to regress camera pose. Figure 2 shows all of our three proposed models. At the heart of MapNet is a DNN that regresses absolute camera pose from an input image, which is described in detail in Section 3.1. MapNet takes in tuples of images and additionally enforces constraints between pose predictions for pairs, as described in Section 3.2. MapNet+ improves a trained MapNet by utilizing the geometric constraints expressed by visual odometry (VO) on additional unlabeled videos from the same scene, or synchronized GPS readings (Section 3.3). Finally, we employ moving-window PGO during inference to obtain a smooth and drift free camera trajectory by fusing MapNet+ absolute pose predictions and VO (Section 3.4).

3.1. Camera Pose Regression with DNNs

Our work is built upon prior works in DNN-based pose estimation methods [11, 31, 32, 33, 41, 56], which regress 6-DoF camera pose from an input RGB image with a DNN. In our work, we made several modifications to PoseNet [32, 33]. First, we use ResNet-34 [25] and modify it by introducing a global average pooling layer after the last conv layer, followed by a fc layer with 2048 neurons, a ReLU and dropout with $p = 0.5$. This is followed by a final fc layer that outputs a 6-DoF camera pose.

Second, we propose to parameterize camera orientation as the logarithm of a unit quaternion [2], which is better suited for regression with deep learning. PoseNet and its variants [11, 32, 33, 41, 56] used 4-d unit quaternions to represent orientation, and regress it with l_1 or l_2 norm. This has two issues: (1) the quadruple is an over parameterization of the 3-DoF rotation, and (2) normalization of the output quadruple is required but often results in worse performance [33, 32, 41]. While Euler angles used in [51] are not over-parameterized, they are not suited for regression since they wrap around 2π.

The logarithm of a unit quaternion, $\log \mathbf{q}$ has 3 dimensions and is not over-parameterized. This allows us to directly use the l_1 or l_2 distance as the loss function without normalization. The logarithm of a unit quaternion $\mathbf{q} = (u, \mathbf{v})$, where u is a scalar and $\mathbf{v}$ is a 3-d vector, is defined as [26]

$$\log \mathbf{q} = \begin{cases} \frac{\mathbf{v}}{\|\mathbf{v}\|} \cos^{-1} u, & \text{if } \|\mathbf{v}\| \neq 0 \\ \mathbf{0}, & \text{otherwise} \end{cases} \quad (1)$$

The logarithmic form $\mathbf{w} = \log \mathbf{q}$ can be converted back to a unit quaternion by the formula $\exp \mathbf{w} = (\cos \|\mathbf{w}\|, \frac{\mathbf{w}}{\|\mathbf{w}\|} \sin \|\mathbf{w}\|)$. As shown in Table 2 in Section 4, using this rotation parameteration achieve better results than PoseNet [32]. We also implemented other metrics for rotation [27], and found they did not improve the performance.

3.2. MapNet: Geometry-Aware Learning

Similar to PoseNet [11, 32, 33, 41, 56], MapNet also learns a DNN Θ that estimates the 6-DoF camera pose $\mathbf{p} = (\mathbf{t}, \mathbf{w})$ from an input RGB image $\mathbf{I}$ on the training set $\mathcal{D} = \{(\mathbf{I}, \mathbf{p}^*)\}$ via supervised learning, $f(\mathbf{I}; \Theta) = \mathbf{p}$. The main difference, however, is that MapNet minimizes both the loss of the per-image absolute pose and the loss of the relative pose between image pairs, as shown in Fig. 2,

$$L_{\mathcal{D}}(\Theta) = \sum_{i=1}^{|\mathcal{D}|} h(\mathbf{p}_i, \mathbf{p}_i^*) + \alpha \sum_{i,j=1, i \neq j}^{|\mathcal{D}|} h(\mathbf{v}_{ij}, \mathbf{v}_{ij}^*), \quad (2)$$

where $\mathbf{v}_{ij} = (\mathbf{t}_i - \mathbf{t}_j, \mathbf{w}_i - \mathbf{w}_j)$ is the relative camera pose between pose predictions $\mathbf{p}_i$ and $\mathbf{p}_j$ for images $\mathbf{I}_i$ and $\mathbf{I}_j$. $h(\cdot)$ is a function to measure the distance between the predicted camera pose $\mathbf{p}$ and the ground truth camera pose $\mathbf{p}^*$, defined as [32]:

$$h(\mathbf{p}, \mathbf{p}^*) = \|\mathbf{t} - \mathbf{t}^*\|_1 e^{-\beta} + \beta + \|\mathbf{w} - \mathbf{w}^*\|_1 e^{-\gamma} + \gamma, \quad (3)$$

where β and γ are the weights that balance the translation loss and rotation loss. Both β and γ are learned dur-

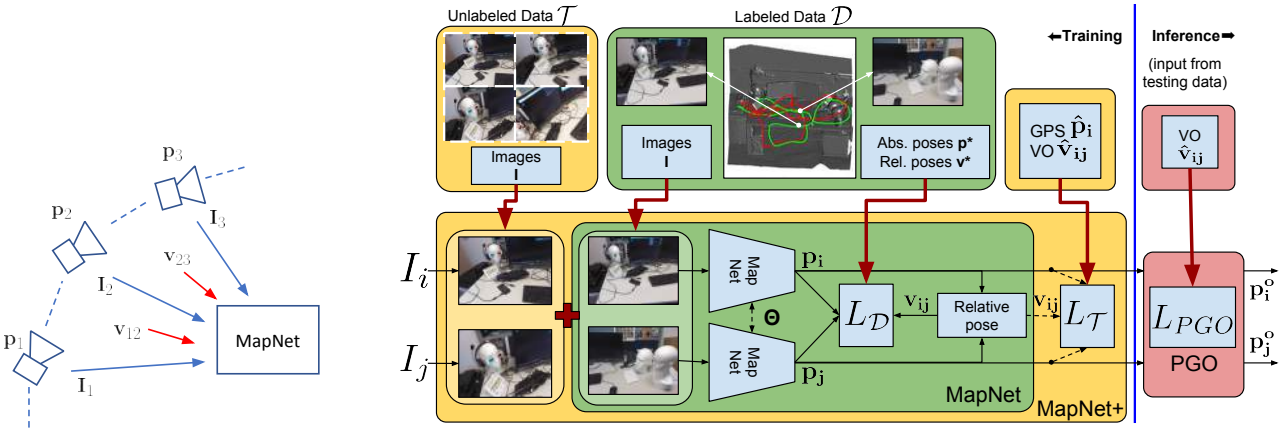
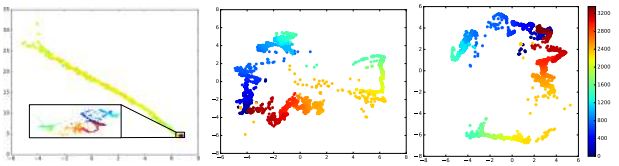


Figure 2: **Left**: MapNet learns a general map representation directly from input data, including images, visual odometry (VO), and other sensory inputs. **Right**: Data flow for our proposed algorithms. MapNet enforces geometric constraints between relative poses and absolute poses in network training. MapNet+ fuses other inputs such as visual odometry to update maps with self-supervised learning. MapNet+PGO performs PGO at testing time to further improve accuracy.



PoseNet MapNet MapNet+(2seq)

Figure 3: 2D multi-dimensional scaling (MDS) of penultimate layer features for various models trained on the LOOP sequence from Oxford RobotCar [39]. Input images are from a held-out test sequence (see Fig. 1 for ground-truth camera poses). The points are chronologically colored. Features learned by PoseNet do not correlate with the distribution of ground truth camera poses, while those learned by MapNet and MapNet+ show successively better correlation with the ground truth camera poses (see Fig. 5).

ing training with initialization β_0 and γ_0. $(\mathbf{I}_i, \mathbf{I}_j)$ are image pairs within each tuple of s images sampled with a gap of k frames from $\mathcal{D}$. Intuitively, adding the second loss of the relative camera poses between image pairs helps to enforce global consistency, which improves the performance of camera localization (see Section. 4).

To understand more about the representation MapNet learns, we visualize the distribution of the feature vectors of the last activation layer using 2D multi-dimensional scaling (MDS) [6]. We chose MDS rather than T-SNE because it is designed to preserve global structure of the feature space. In Fig. 3, we show that PoseNet [32] feature vectors for test images captured along a loop in the Oxford RobotCar dataset [39] do not correlate with the distribution of ground truth camera poses, while the features learned by MapNet and MapNet+ (next subsection) show successively better correlation. All models use the same network architecture.

3.3. MapNet+: Update with Unlabeled Data

Both PoseNet and MapNet require labelled data (*i.e.*, images with absolute camera poses) to train. In many real applications, we may also have lots of unlabeled data, *e.g.*, videos captured at different times or camera motions in the same scene. Off-the-shelf VO algorithms [17, 18] provide relative camera poses between image pairs from these videos. Other sensors (*e.g.*, IMU and GPS) can also provide measurements about camera pose, especially for challenging conditions (*e.g.*, textureless, low-light). MapNet+ fuses these additional data $\mathcal{T}$ to update the weights of MapNet with self-supervised learning.

Suppose the additional data are some videos of the same scene, $\mathcal{T} = \{\mathbf{I}_t\}$. We can compute the relative poses $\hat{\mathbf{v}}_{ij}$ between consecutive frames with visual odometry algorithms [17, 18, 45]. In order to update the map with $\mathcal{T}$, we fine-tune a pre-trained MapNet Θ by minimizing a loss function that consists of the original loss from the labelled dataset $\mathcal{D}$ and the loss from the unlabeled data $\mathcal{T}$,

$$L(\Theta) = L_{\mathcal{D}}(\Theta) + L_{\mathcal{T}}(\Theta), \tag{4}$$

where $L_{\mathcal{T}}(\Theta)$ is the distance between the relative camera pose $\mathbf{v}_{ij}$ (from predictions $\mathbf{p}_i, \mathbf{p}_j$) and visual odometry $\hat{\mathbf{v}}_{ij}$,

$$L_{\mathcal{T}}(\Theta) = \sum_{i,j=1, i\neq j}^{|\mathcal{T}|} h(\mathbf{v}_{ij}, \hat{\mathbf{v}}_{ij}) \tag{5}$$

Since VO algorithms compute $\hat{\mathbf{v}}_{ij}$ in the coordinate system of camera i, the relative pose $\mathbf{v}_{ij}$ is also computed in that coordinate system:

$$\begin{aligned}\mathbf{v}_{ij} = (&\exp(\mathbf{w}_j)(\mathbf{t}_i - \mathbf{t}_j)\exp(\mathbf{w}_j)^{-1}, \\ &\log(\exp(\mathbf{w}_j)^{-1}\exp(\mathbf{w}_i)),\end{aligned} \tag{6}$$

Note that we still keep the supervision loss $L_{\mathcal{D}}(\Theta)$ from $\mathcal{D}$ — this is important to avoid trivial solutions if we optimize only the self-supervised loss $L_{\mathcal{T}}(\Theta)$ from $\mathcal{T}$. Thus each mini-batch samples half from the labelled data $\mathcal{D}$ and half from the unlabeled data $\mathcal{T}$. The image pairs $(\mathbf{I}_i, \mathbf{I}_j)$ are sampled similarly from tuples of s images with a gap of k frames from both $\mathcal{D}$ and $\mathcal{T}$.

Intuitively, MapNet+ exploits the complimentary characteristics of VO and DNN-based pose prediction — VO is locally accurate but often drifts over time, and DNN-based pose predictions are noisy but drift-free. For other sensors such as IMU (which measures relative rotation) and GPS (which measures 3D locations), we can define similar loss terms $L_{\mathcal{T}}(\Theta)$ that minimize the difference between such measurements and the predictions from the MapNet.

3.4. MapNet+PGO: Optimizing During Inference

During inference, MapNet+PGO fuses the absolute pose predictions from MapNet+ and the relative poses from VO using pose graph optimization (PGO) [8, 38, 16] to get smooth and globally consistent pose predictions. It runs in a moving-window of T frames. Suppose the initial poses predicted by MapNet+ are $\{\mathbf{p}_i\}_{i=1}^T$, and the relative poses between two frames from VO are $\{\hat{\mathbf{v}}_{ij}\}$ where $i, j \in [1, T], i \neq j$. MapNet+PGO solves for the optimal poses $\{\mathbf{p}_i^o\}_{i=1}^T$ by minimizing the following cost:

$$L_{PGO}(\{\mathbf{p}_i^o\}_{i=1}^T) = \sum_{i=1}^{T} \bar{h}(\mathbf{p}_i^o, \mathbf{p}_i) + \sum_{i,j=1, i\neq j}^{T} \bar{h}(\mathbf{v}_{ij}^o, \hat{\mathbf{v}}_{ij}), \tag{7}$$

where $\bar{h}(\cdot)$ is the standard pose distance function used in PGO literature [24]. PGO is an iterative algorithm where internally $\mathbf{v}_{ij}^o$ is derived from $\mathbf{p}_i^o$ and $\mathbf{p}_j^o$ as in Equation (6). Details and derivation are included in the supplementary material. Note here we fix the DNN weights Θ and only optimize $\{\mathbf{p}_i^o\}_{i=1}^T$. As shown in Section 4, MapNet+PGO further improves the accuracy of pose estimation, with a minimal extra computational cost at testing.

3.5. Implementation Details

We implemented our algorithms with PyTorch [1], using the Adam optimizer [34] with a learning rate of 1e-3 and a weight decay of 5e-4. The input images are scaled to 341×256 pixels, and normalized by pixel mean subtraction and standard deviation division. We set the weight coefficient $\alpha = 1$ and initializations $\beta_0 = 0.0$ and $\gamma_0 = -3.0$. Image pairs are sampled from tuples of size $s = 3$ with spacing $k = 10$ frames for MapNet and MapNet+, and $T = 7$, $k = 150$ frames for PGO.[1] All models are trained for 300 epochs, except for MapNet+, which is finetuned with $\alpha = 0$ for 5 epochs from a trained MapNet.

[1] PGO for RobotCar sequences uses frame separation $T = 7$, $k = 10$.

4. Experimental Evaluations

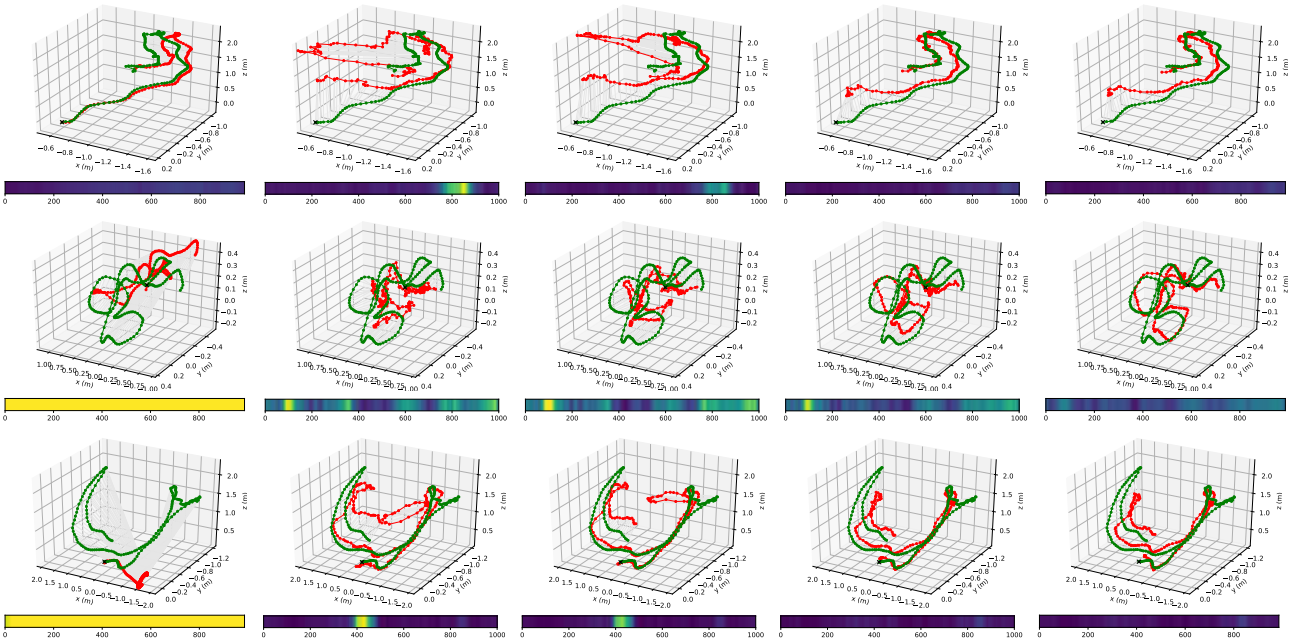
Datasets We evaluate our algorithms on two well-known public datasets — 7-Scenes [49] for small-scale, indoor, AR/VR-type scenarios, and Oxford RobotCar [39] for large-scale, outdoor, autonomous driving scenarios. 7-Scenes contains RGB-D image sequences of seven indoor environments (with the spatial extent less than 4 meters) captured with a Kinect sensor. Multiple sequences were captured for each environment, and each sequence is 500 or 1000 frames. The ground truth camera poses are obtained with KinectFusion. The 7-Scenes dataset has recently been evaluated extensively as a benchmark [33, 31, 32, 41, 56, 11, 7], which makes it ideal for us to compare with prior state-of-the-art methods.

Oxford RobotCar contains over 100 repetitions of a consistent route (about 10km) through central Oxford captured twice a week over a period of over a year. Thus the dataset captures different combinations of weather, traffic, pedestrians, construction and roadworks. In addition to the images captured with the six cameras mounted on the car, the dataset also contains LIDAR, GPS and INS measurements, as well as stereo visual odometry (VO). We extracted two subsets from this dataset: LOOP (Fig. 1) with a total length of 1120m, which was also used in VidLoc [11], and FULL (Fig. 6) with a total length of 9562m. Details of the training, validation, and testing sequences are provided in the supplementary material.

Baselines and Data Augmentation We compare our approach with two groups of prior methods on 7-Scenes. For the DNN-based prior work, we compare with PoseNet15 [33], PoseNet16 [31], PoseNet17 [32], Hourglass [41], LSTM-PoseNet [56], and VidLoc [11]. For the traditional visual odometry based methods, we used DSO [17] to compute the VO and integrate to obtain camera poses. We run the DSO with images at the same spatial resolution as MapNet. On Oxford RobotCar, only VidLoc [11] reported results on the LOOP scene but it did not provide training and testing sequences. Thus, the two baselines to compare are the provided stereo VO, as well as our version of PoseNet (with $\log \mathbf{q}$). In RobotCar, we randomly perturb the brightness, saturation, hue and contrast of images during training for experiments, which we found essential for performing cross-weather and cross-time localization.

4.1. Experiments on the 7-Scenes Dataset

Effects of Rotation Parameteriation In Section 3.1, we introduced a new parameterization of camera orientation for PoseNet and used ResNet34 as the base network. Table 2 shows the quantitative results of these modifications to the baseline PoseNet. Following the same convention of prior work [33, 31, 32, 41, 56, 11], we compute the median error



DSO [17] PoseNet [32, 33, 31] MapNet MapNet+ MapNet+PGO

Figure 4: **Camera localization results on 7-Scenes dataset [49].** For each subfigure, the top 3D plot shows the camera trajectory (green for the ground truth and red for the prediction), and the bottom color bar shows rotation error for all the frames. From top to bottom, the three testing sequences are: Redkitchen-seq-03, Heads-seq-01, and Redkitchen-seq-12. See Table 3 for quantitative comparison.

Table 2: Translation and rotation error on the 7-Scenes dataset.

Scene	PoseNet17 [32]	PoseNet (ResNet34)	PoseNet+log $\mathbf{q}$ (ResNet34)
Chess	0.13m, 4.48°	**0.11m, 4.24°**	**0.11m**, 4.29°
Fire	**0.27m, 11.30°**	0.29m, 11.68°	**0.27m**, 12.13°
Heads	**0.17m**, 13.00°	0.20m, 13.11°	0.19m, **12.15°**
Office	**0.19m, 5.55°**	**0.19m**, 6.40°	**0.19m**, 6.35°
Pumpkin	0.26m, **4.75°**	0.23m, 5.77°	**0.22m**, 5.05°
Red Kitchen	**0.23m**, 5.35°	0.27m, 5.81°	0.25m, **5.27°**
Stairs	0.35m, 12.40°	0.31m, 12.43°	**0.30m, 11.29°**
Average	0.23m, 8.12°	0.23m, 8.49°	**0.22m, 8.07°**

for camera translation and rotation.[2] As shown, our proposed rotation parameterization does improve performance.

Comparison with Prior Methods Figure 4 shows the camera trajectories for several testing sequences from the 7-Scenes dataset for DSO VO, PoseNet, MapNet, MapNet+, and MapNet+PGO. Table 3 shows quantitative comparisons. The unlabeled data used to fine-tune MapNet+ for these experiments are the unlabeled test sequences. This is a transductive learning scenario [9, 15]. As shown, DSO often drifts over time and PoseNet results in noisy predictions. In contrast, by including various geometric constraints into network training and inference our proposed approaches MapNet, MapNet+ and MapNet+PGO successively improve the performance. A complete table for all testing sequences from the 7-Scenes dataset is included in the supplementary material.

[2]Other statistics of the camera pose estimation errors are also provided in the supplementary material, which support the same conclusion.

4.2. Experiments on the Oxford RobotCar Dataset

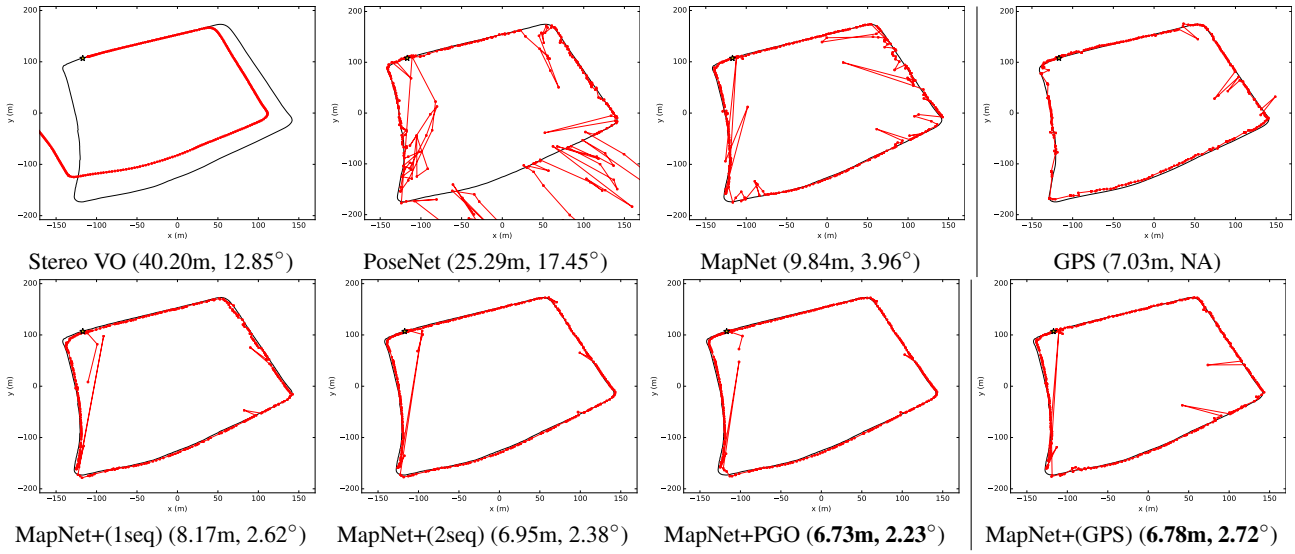
Results on the LOOP Route We first train a baseline PoseNet (with the $\log \mathbf{q}$ parameterization for rotation) and a MapNet model using two labelled sequences captured on the LOOP route under cloudy weather, while the testing sequence is captured under sunny weather. We then perform two experiments with different auxiliary data for MapNet+.

In the first experiment, MapNet+ is trained on additional unlabeled LOOP sequences separate from the testing sequence, with stereo VO provided with the dataset. To tease apart the influence of labeled and unlabeled data in the effectiveness of our MapNet+ models, we train them with varying amounts of labeled (one to two sequences) and unlabeled data (zero to three sequences). Figure 8 shows the mean translation and rotation errors of these models on the testing sequence. While labeled data is clearly more important than an equal amount of unlabeled data, we show that unlabeled data does consistantly improve performance as more becomes available. This trend bodes well for real-world scenarios, where the amount of unlabeled data available far exceeds the amount of labeled data.

In the second experiment, MapNet+ is trained with GPS *i.e.*, the dataset $\mathcal{T}$ contains two sequences of images and

Table 3: Translation error (m) and rotation error (°) for various methods on the 7-Scenes dataset [49].

Scene	PoseNet17 [32]	Hourglass [41]	LSTM-Pose [56]	VidLoc [11]	DSO [17]	MapNet	MapNet+	MapNet+PGO
Chess	0.13m, 4.48°	0.15m, 6.17°	0.24m, 5.77°	0.18m, NA	0.17m, 8.13°	**0.08m**, 3.25°	0.10m, **3.17°**	0.09m, 3.24°
Fire	0.27m, 11.30°	0.27m, 10.84°	0.34m, 11.9°	0.26m, NA	**0.19m**, 65.0°	0.27m, 11.69°	0.20m, **9.04°**	0.20m, 9.29°
Heads	0.17m, 13.00°	0.19m, 11.63°	0.21m, 13.7°	0.14m, NA	0.61m, 68.2°	0.18m, 13.25°	0.13m, 11.13°	**0.12m, 8.45°**
Office	0.19m, 5.55°	0.21m, 8.48°	0.30m, 8.08°	0.26m, NA	1.51m, 16.8°	**0.17m, 5.15°**	0.18m, 5.38°	0.19m, 5.42°
Pumpkin	0.26m, 4.75°	0.25m, 7.01°	0.33m, 7.00°	0.36m, NA	0.61m, 15.8°	0.22m, 4.02°	**0.19m, 3.92°**	**0.19m**, 3.96°
Kitchen	0.23m, 5.35°	0.27m, 10.15°	0.37m, 8.83°	0.31m, NA	0.23m, 10.9°	0.23m, 4.93°	0.20m, 5.01°	**0.20m, 4.94°**
Stairs	0.35m, 12.40°	0.29m, 12.46°	0.40m, 13.7°	**0.26m**, NA	0.26m, 21.3°	0.30m, 12.08°	0.30m, 13.37°	0.27m, **10.57°**
Average	0.23m, 8.12°	0.23m, 9.53°	0.31m, 9.85°	0.25m, NA	0.26m, 29.4°	0.21m, 7.77°	0.19m, 7.29°	**0.18m, 6.55°**



Stereo VO (40.20m, 12.85°) | PoseNet (25.29m, 17.45°) | MapNet (9.84m, 3.96°) | GPS (7.03m, NA)

MapNet+(1seq) (8.17m, 2.62°) | MapNet+(2seq) (6.95m, 2.38°) | MapNet+PGO (**6.73m, 2.23°**) | MapNet+(GPS) (**6.78m, 2.72°**)

Figure 5: **Camera localization results on the LOOP scene (1120m long) of the Oxford RobotCar dataset [39]**. The ground truth camera trajectory is the black line, the star indicates the first frame, and the red lines show the camera pose predictions. The caption of each figure shows the mean translation error (m) and mean rotation error (°). MapNet+(1seq) uses one unlabeled sequence, while MapNet+(2seq) uses two unlabeled sequences. **Left**: MapNet+ trained with unlabeled images and stereo VO. **Right**: MapNet+ trained with unlabeled images and GPS data.

their GPS locations, which are separate from the testing sequence. Since GPS measurements are sparse (less than 10% of images have corresponding GPS measurements), we first linearly interpolate GPS measurements for entire sequence. We define the loss of auxillary data $\mathcal{T}$ in Equation (5) as $L_{\mathcal{T}}(\Theta) = \sum_{i=1}^{|\mathcal{T}|} h(\mathbf{p_i}, \hat{\mathbf{p}}_\mathbf{i})$, where $\hat{\mathbf{p}}_i$ is the linearly interpolated GPS measurement of the 2D camera location.

Figure 5 shows the estimated camera poses for all the methods in these two experiments along with the mean translation error (m) and mean rotation error (°). Figure 7 shows the cumulative distributions of the translation errors for all the methods on the LOOP route. In both figures, the left part shows that MapNet significantly improves the estimation compared to PoseNet and stereo VO. MapNet+ and MapNet+PGO further improve the pose predictions. The right part shows that by fusing GPS signals, MapNet+(GPS) obtains better results compared to MapNet and GPS alone.

Results on the FULL Route We also evaluated our approach on the FULL route of the Oxford RobotCar dataset. This is challenging since the FULL route is 9562m long in total and has a complicated ground truth camera trajectory. Figure 6 shows the results of all the models with mean translation error (m) and rotation error (°). As shown, our method MapNet significantly outperforms the baseline PoseNet (both trained for 100 epochs) and the stereo VO (provided by the dataset). By fusing the stereo VO information, MapNet+ and MapNet+PGO further improve the result. The cumulative distributions of the translation errors also show the large improvement over the baselines.

We note in both the LOOP scene (Fig. 5) and FULL scene (Fig. 6), there are quite a few outlier estimations — this often corresponds to image frames with large overexposed regions. These outliers often can be filtered out with simple post-processing such as temporal median filtering, which we show in the supplementary material.

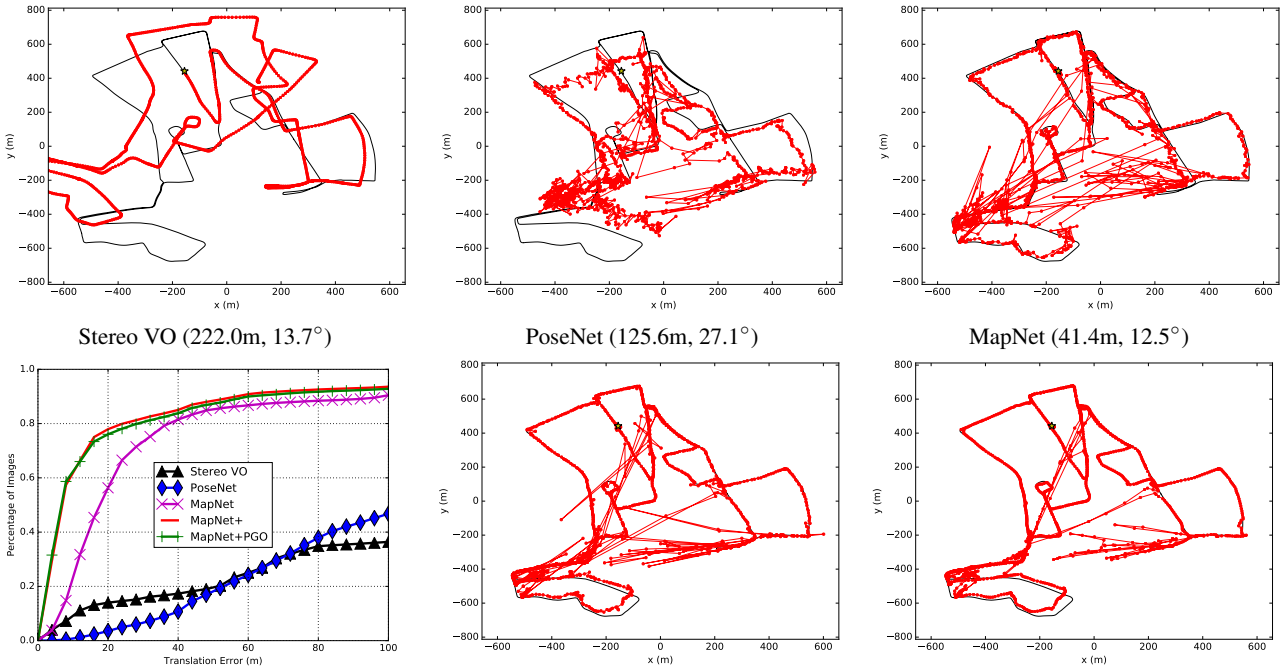


Figure 6: Comparison of camera localization results on the FULL scene (9562m long) of the Oxford RobotCar dataset [39]. The ground truth camera trajectory is the black line, and the star indicates the first frame. The red lines show the results of stereo VO (provided by the dataset), our version of PoseNet+log $\mathbf{q}$, MapNet, and its variations. The caption of each figure shows the mean translation error (m) and mean rotation error (°). A plot of the cumulative distribution of the translation error is also included.

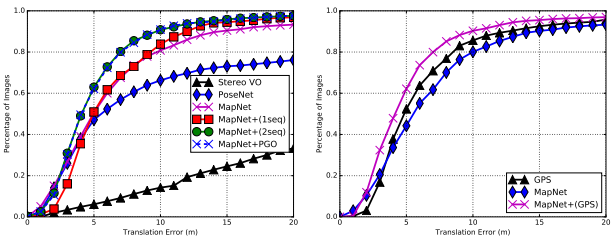


Figure 7: Cumulative distributions of the translation errors (m) for all the methods evaluated on Oxford RobotCar LOOP. x-axis is the translation error and y-axis is the percentage of frames with error less than the value.

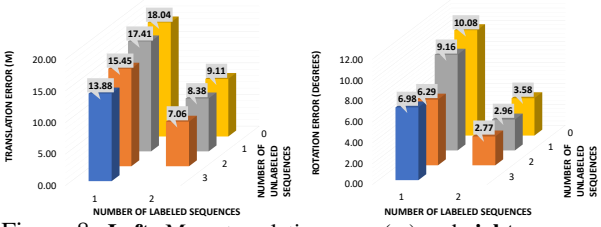


Figure 8: **Left**: Mean translation error (m) and **right**: mean rotation error (°) for MapNet+ models trained with varying amounts of labeled and unlabeled data on the Oxford RobotCar LOOP sequence. X-axis indicates the number of labeled sequences (1-2), Y-axis indicates the number of unlabeled sequences (0-3) and Z-axis indicates the error.

5. Conclusions and Discussions

In summary, MapNet learns a *general, data-driven* map representation for camera localization. Compared to prior DNN-based methods, our models bring geometric constraints (*e.g.*, pose graph optimization and bundle adjustment) widely used in visual SLAM and structure-from-motion into DNN-based learning, which allow us to learn from unlabeled data and to easily fuse other input sources (*e.g.*, visual odometry, GPS, IMU) for globally consistent pose estimation. We evaluate our approach on both indoor and outdoor datasets and show significantly better performance than both prior DNN-based methods. While we use only RGB images in this paper, our approach can be extended to stereo/RGBD inputs as well.

There are several future directions we plan to explore. Unlike the mapping in traditional visual SLAM systems, MapNet and MapNet+ cannot expand maps to unknown space. A tighter integration with visual SLAM systems may enable mapping of unknown regions. With recent success in extracting high-level semantic information (*e.g.*, objects and scene composition) from images, we plan to utilize such semantic information for camera localization.

Appendix 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 [23, 24].

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 [26]:

$$\mathbf{q} = (\cos\|\mathbf{w}\|, \frac{\mathbf{w}}{\|\mathbf{w}\|}\sin\|\mathbf{w}\|) \tag{8}$$

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.

$$\begin{aligned} E(z) &= \sum_c E_c(z) \\ &= \sum_c \bar{h}(f_c(c), k_c) \end{aligned} \tag{9}$$

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 [23, 24], we define $\bar{h}(\cdot)$ as:

$$\bar{h}(f_c(c), k_c) = (f_c(z) - k_c)^T S_c (f_c(z) - k_c) \tag{10}$$

where S_c the covariance matrix for the constraint.

Optimization Following [23], We first linearize f_c around $\bar{z}$, the current value of z:

$$f_c(\bar{z} + \Delta z) \approx f_c(\bar{z}) + \left.\frac{\partial f_c}{\partial z}\right|_{z=\bar{z}} \Delta z \tag{11}$$

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

$$\begin{aligned} E(\Delta z) &= \sum_c (f_c(\bar{z} + \Delta z) - k_c)^T S_c (f_c(\bar{z} + \Delta z) - k_c) \\ &\approx \sum_c \left(f_c(\bar{z}) + \left.\frac{\partial f_c}{\partial z}\right|_{z=\bar{z}} \Delta z - k_c \right)^T L_c \\ &\quad L_c^T \left(f_c(\bar{z}) + \left.\frac{\partial f_c}{\partial z}\right|_{z=\bar{z}} \Delta z - k_c \right) \\ &= \sum_c \left|\left| L_c^T \left(f_c(\bar{z}) + \left.\frac{\partial f_c}{\partial z}\right|_{z=\bar{z}} \Delta z - k_c \right) \right|\right|^2 \\ &= \sum_c ||J_c \Delta z - r_c||^2 \end{aligned} \tag{12}$$

where **Jacobian** $J_c = L_c^T \left.\frac{\partial f_c}{\partial z}\right|_{z=\bar{z}}$ 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:

$$\Delta z^* = \min_{\Delta z} ||J\Delta z - r||^2 \tag{13}$$

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

Finally, we update the state vector:

$$z = z \boxplus \Delta z \tag{14}$$

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

Detour: Manifolds for Quaternion Update As mentioned in [24], if we had used a simple addition in the update Equation (14), it would have broken the constraints introduced by the over-parameterization of quaternions. So we use manifolds. According to [24], "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" [26] to implement $\boxplus$. For this, we re-cast the objective function as a function of the update on the manifold, $\Delta\check{z}$:

$$E(\Delta\check{z}) = \sum_c (f_c(\bar{z} \boxplus \Delta\check{z}) - k_c)^T S_c (f_c(\bar{z} \boxplus \Delta\check{z}) - k_c) \tag{15}$$

The linearization step is:

$$f_c(\bar{z} \boxplus \Delta\check{z}) \approx f_c(\bar{z}) + \left.\frac{\partial f_c}{\partial z}\right|_{z=\bar{z}} \left.\frac{\partial \bar{z} \boxplus \Delta\check{z}}{\partial \Delta\check{z}}\right|_{\Delta\check{z}=0} \Delta\check{z} \tag{16}$$

So the Jacobian in this case is:

$$\check{J}_c = J_c \left.\frac{\partial \bar{z} \boxplus \Delta\check{z}}{\partial \Delta\check{z}}\right|_{\Delta\check{z}=0} \tag{17}$$

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

$$\bar{z} \boxplus \Delta\check{z} = \bar{z} \cdot \Delta\tilde{z} \tag{18}$$

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 (8)). 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,

$$\begin{aligned} \left.\frac{\partial \bar{z} \boxplus \Delta\check{z}}{\partial \Delta\check{z}}\right|_{\Delta\check{z}=0} &= \frac{\partial \bar{z} \cdot \Delta\tilde{z}}{\partial \Delta\tilde{z}} \left.\frac{\partial \Delta\tilde{z}}{\partial \Delta\check{z}}\right|_{\Delta\check{z}=0} \\ &= \frac{\partial \bar{z} \cdot \Delta\tilde{z}}{\partial \Delta\tilde{z}} M_e \end{aligned} \tag{19}$$

For the first term, we use the formula for derivative of quaternion product from [30].

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 (17) and (19) 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 [30].

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 [30].

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 (8), 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.

Appendix 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.*, $(\mathbf{I}_i, \mathbf{I}_{i+k}, \cdots, \mathbf{I}_{i+k(s-2)}, \mathbf{I}_{i+k(s-1)})$.

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.

Appendix 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

Appendix 4

Additional results

Experiments on the 7-Scenes Dataset Figure 9 and Figure 10 show the results for all the 18 testing sequences on the 7-Scenes dataset. Table 4 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.

Experiments on the RobotCar Dataset Figure 11 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 12.

References

[1] PyTorch, 2017. 5

[2] S. Altmann. *Rotations, Quaternions, and Double Groups*. Dover Publications, 2005. 3

[3] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[4] R. Arandjelovic and A. Zisserman. All about VLAD. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 2

[5] P. Bergmann, R. Wang, and D. Cremers. Online photometric calibration of auto exposure video for real-time visual odometry and slam. In *arXiv*, 2017. 2

[6] I. Borg and P. J. Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005. 4

[7] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother. DSAC: Differential RANSAC for camera localization. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 5

[8] L. Carlone, G. Calafiore, and F. Dellaert. Pose graph optimization in the complex domain: Duality, optimal solutions, and verification. *IEEE Transactions on Robotics*, 32(3):545–565, 2016. 1, 5

[9] O. Chapelle, B. Scholkopf, and A. Zien. *Semi-Supervised Learning*. MIT Press, 2006. 6

[10] J. Civera, D. Gaivez-Lopez, L. Riazuelo, J. Tardos, and J. Montiel. Towards semantic SLAM using a monocular camera. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011. 1, 2

[11] R. Clark, S. Wang, A. Markham, N. Trigoni, and H. Wen. VidLoc: A deep spatio-temporal model for 6-DoF videoclip relocalization. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 3, 5, 7

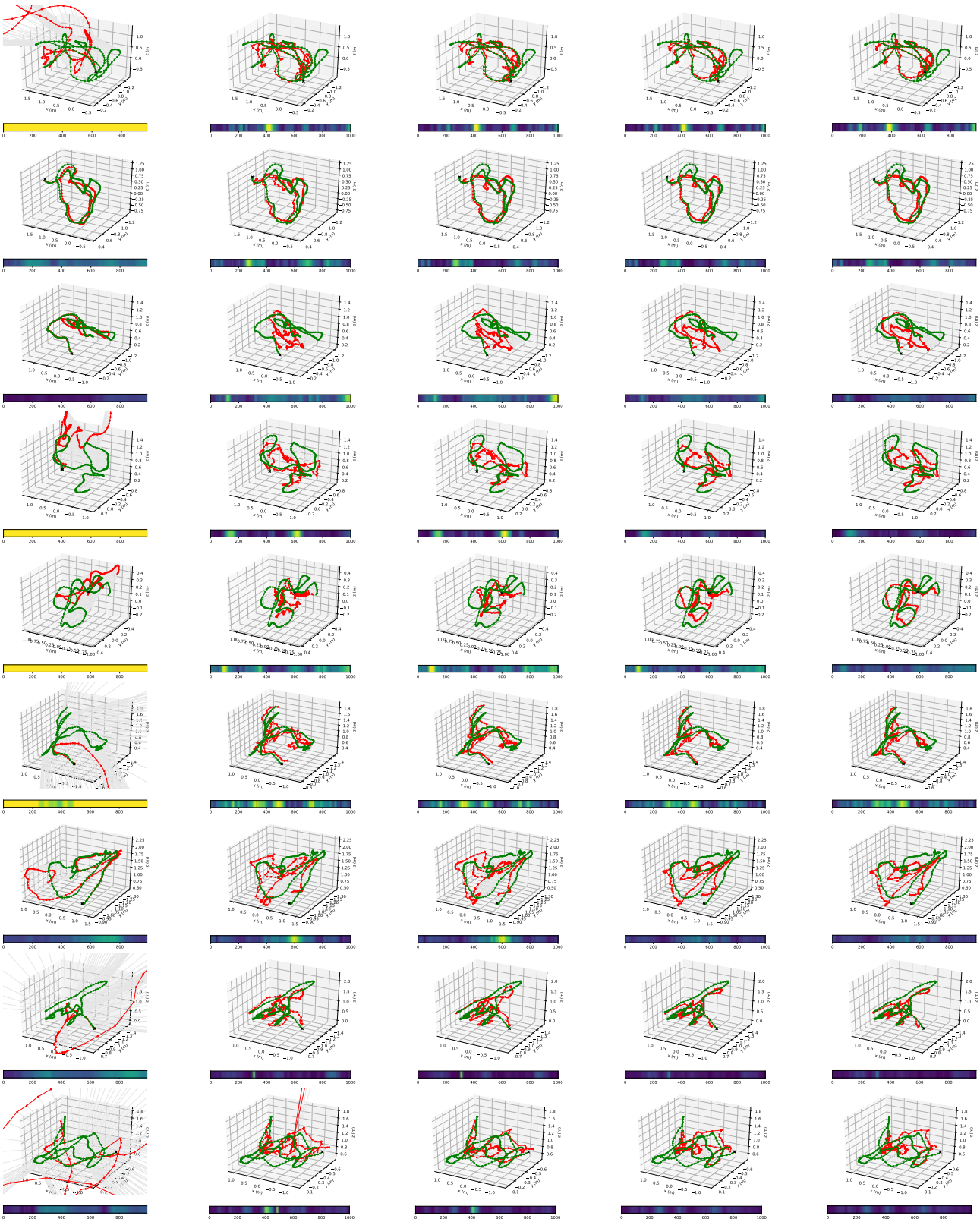
[12] M. Cummins and P. Newman. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008. 1, 2

[13] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2003. 2

[14] F. Dellaert and M. Kaess. Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *Intl. J. of Robotics Research (IJRR)*, 25(12):1181–1204, Dec. 2006. 2

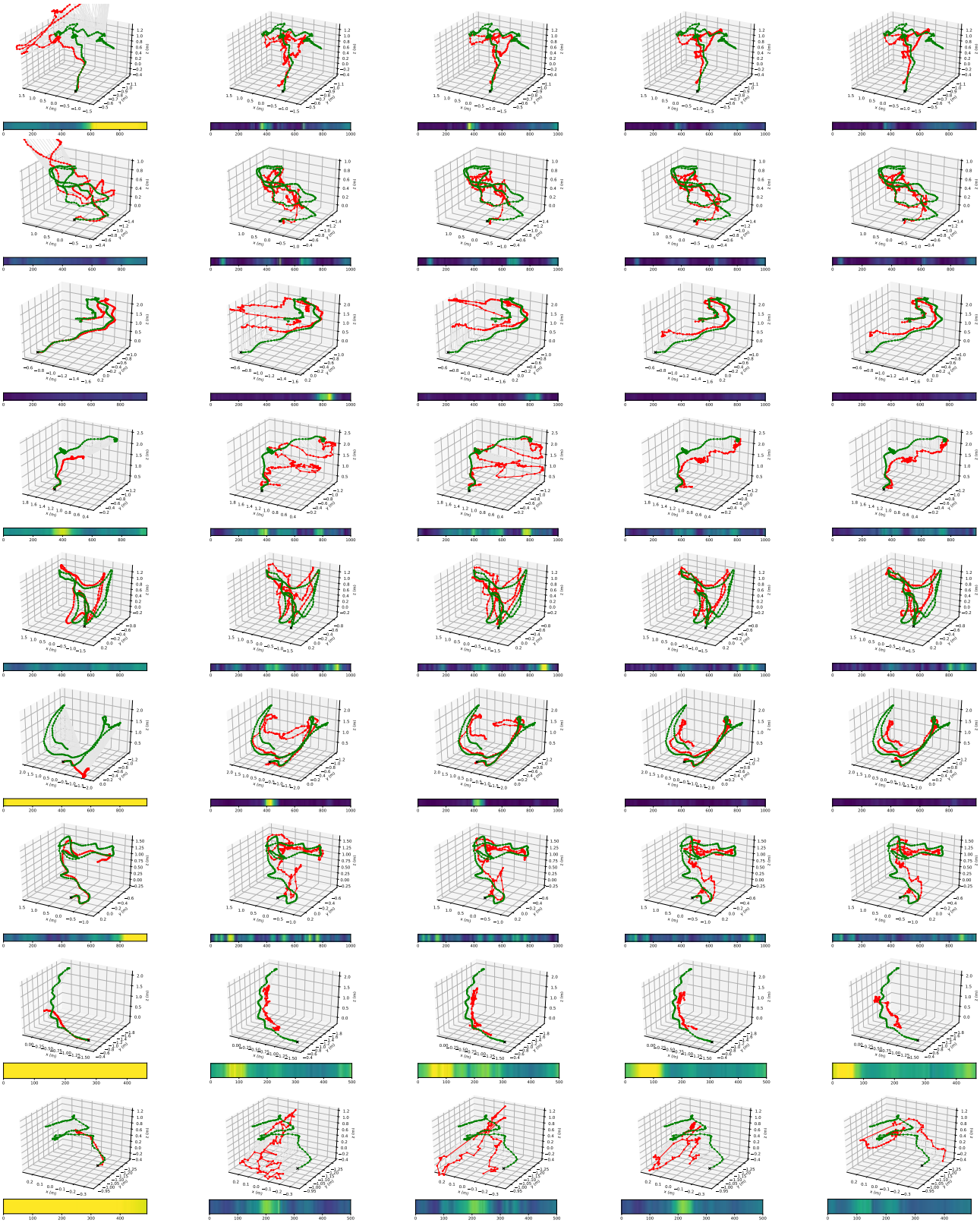
[15] O. Duchenne, J. Y. Audibert, R. Keriven, J. Ponce, and F. Segonne. Segmentation by transduction. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008. 6

[16] T. Duckett, S. Marsland, and J. Shapiro. Fast, online learning of globally consistent maps. *Autonomous Robots*, 12(3):287–300, 2002. 1, 5



DSO [17] PoseNet [32, 33, 31] MapNet MapNet+ MapNet+PGO

Figure 9: **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 [17] PoseNet [32, 33, 31] MapNet MapNet+ MapNet+PGO

Figure 10: **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.

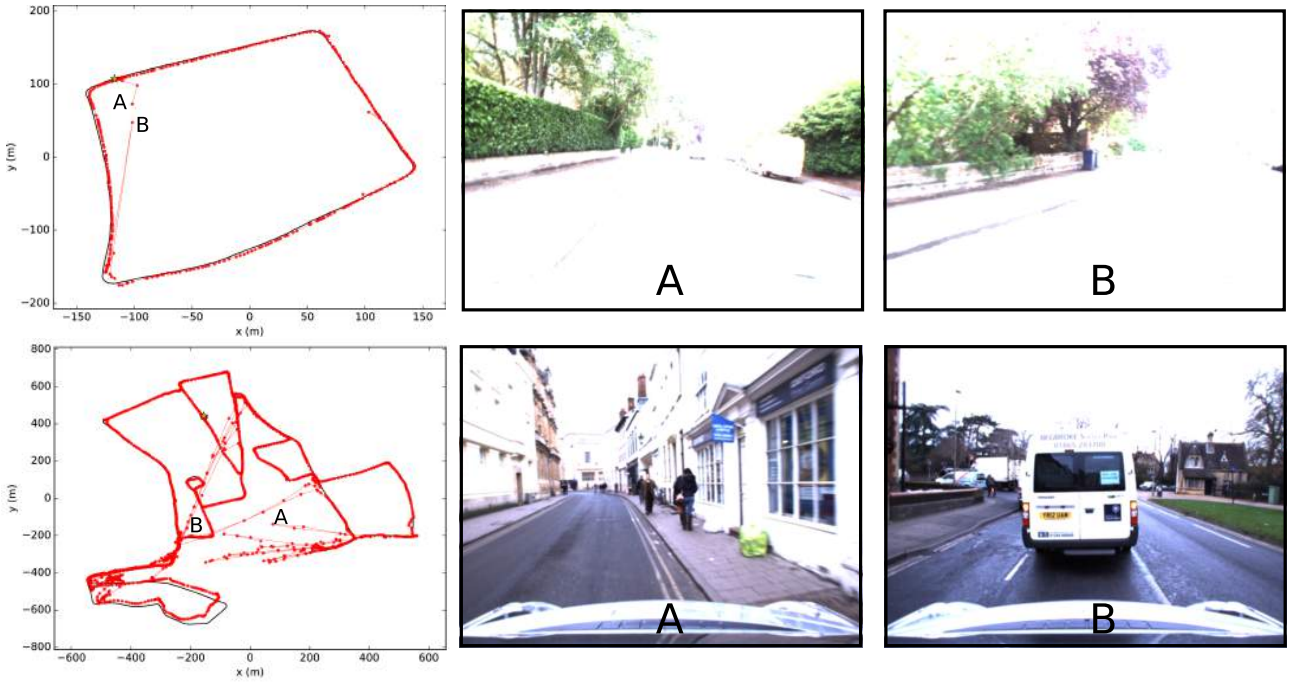
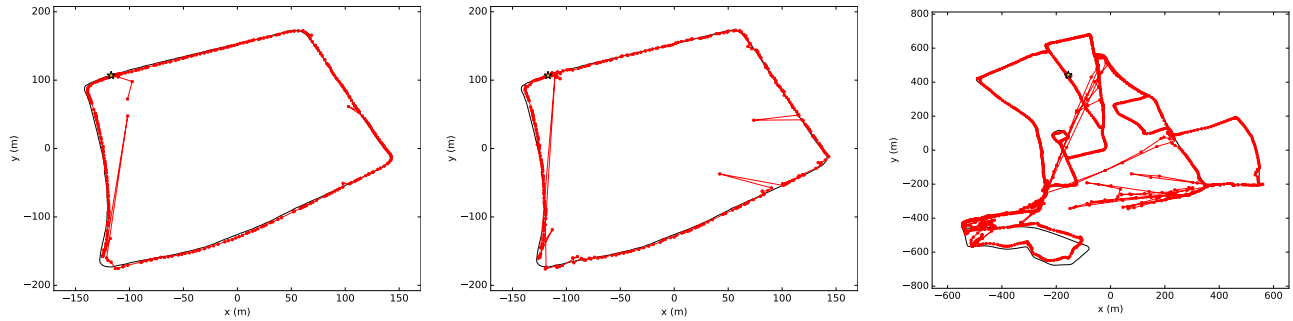
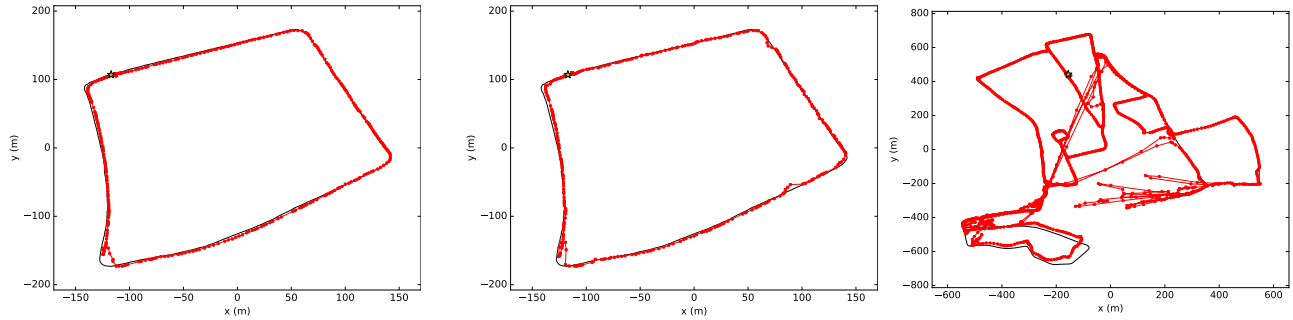


Figure 11: **Images corresponding to the spurious estimation of MapNet+PGO for the LOOP scene (top) and the FULL scene (bottom)**. These outliers usually corresponds to images with large over-exposed regions, or large regions on moving objects (*e.g.*, truck), which often can be filtered out with simple temporal median filtering (see Figure 12).



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 12: **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).

Table 4: Statistics of state-of-the-art methods on the 7-Scenes dataset.

Scene	PoseNet+log $\mathbf{q}$	DSO [17]	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**

[17] J. Engel, V. Koltun, and D. Cremers. DSO: Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017. 1, 2, 4, 5, 6, 7, 12, 13, 15

[18] J. Engel, T. Schops, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2013. 4

[19] J. Engel, T. Schops, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2014. 1, 2

[20] R. Gomez-Ojeda, F.-A. Moreno, D. Scaramuzza, and J. Gonzalez-Jimenez. PL-SLAM: a Stereo SLAM System through the Combination of Points and Line Segments. *arXiv preprint arXiv:1705.09479*, 2017. 2

[21] V. Govindu. Combining two-view constraints for motion estimation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001. 1

[22] V. Govindu. Lie-algebraic averaging for globally consistent motion estimation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004. 1

[23] 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. 9

[24] 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. 2, 5, 9

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[26] C. Hertzberg. A framework for sparse, non-linear least squares problems on manifolds, 2008. 3, 9

[27] D. Huynh. Metrics for 3D rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164, 2009. 3

[28] S. G. J. Stuhmer and D. Cremers. Real-time dense geometry from a handheld camera. In *Joint Pattern Recognition Symposium*, pages 11–20, 2010. 2

[29] H. Jegou, F. Perronnin, M. Douez, J. Sanchez, P. Perez, and C. Schmid. Aggregating local image descriptors into compact codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(9):1704–1716, 2012. 2

[30] Y.-B. Jia. Quaternion and Rotation. Com S 477/577 Lecture Notes at http://web.cs.iastate.edu/~cs577/handouts/quaternion.pdf, 2008. 10

[31] A. Kendall and R. Cipolla. Modeling uncertainty in deep learning for camera relocalization. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2016. 1, 2, 3, 5, 6, 12, 13

[32] 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. 1, 2, 3, 4, 5, 6, 7, 12, 13

[33] 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. 1, 2, 3, 5, 6, 12, 13

[34] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[35] G. Klein and D. Murray. Parallel tracking and maping for small AR workspaces. In *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007. 1, 2

[36] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua. Worldwide pose estimation using 3D point clouds. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2012. 1, 2

[37] M. A. Lourakis and A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Trans. Math. Software*, 36(1):1–30, 2009. 2

[38] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, pages 334–349, 1997. 1, 2, 5

[39] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017. 1, 2, 4, 5, 7, 8

[40] D. Martinec and T. Pajdla. Robust rotation and translation estimation in multiview reconstruction. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 1

[41] I. Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu. Image-based localization using hourglass networks. *arXiv*, abs/1703,07971, 2017. 1, 2, 3, 5, 7

[42] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Eighteenth National Conference on Artificial Intelligence*, pages 593–598, 2002. 2

[43] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *CoRR*, abs/1502.00956, 2015. 1, 2

[44] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *Pro-*

ceedings of IEEE International Conference on Computer Vision (ICCV), pages 2320–2327, 2011. 1, 2

[45] D. Nistr, O. Naroditsky, and J. Bergen. Visual odometry. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–659, 2004. 4

[46] R. Salas-Moreno, R. Newcombe, H. Strasdat, and P. Kelly. SLAM++: Simultaneous localization and mapping at the level of objects. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 1, 2

[47] T. Sattler, B. Leibe, and L. Kobbelt. Fast image-based localization using direct 2d-3d matching. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2011. 1, 2

[48] T. Sattler, B. Leibe, and L. Kobbelt. Efficient and effective prioritized matching for large-scale image-based localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(9):1744–1756, 2017. 1, 2

[49] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in RGBD images. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 1, 2, 5, 6, 7

[50] H. Strasdat, J. M. M. Montiel, and A. J. Davison. Visual SLAM: Why filter? *Image Vision Computing*, 30(2):65–77, Feb. 2012. 2

[51] H. Su, C. Qi, Y. Li, and L. Guibas. Render for CNN: Viewpoint estimation in images using CNNs trained with rendered 3D model views. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015. 3

[52] Y. Taguchi, Y.-D. Jian, S. Ramalingam, and C. Feng. Point-plane SLAM for hand-held 3d sensors. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5182–5189, 2013. 2

[53] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002. 2

[54] S. Thrun and M. Montemerlo. The GraphSLAM algorithm with applications to large-scale mapping of urban structures. *International Journal on Robotics Research*, 25(5/6):403–430, 2005. 2

[55] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV '99, pages 298–372, 2000. 2

[56] F. Walch, C. Hazirbas, L. Leal-Taixe, T. Sattler, S. Hilsenbeck, and D. Cremers. Image-based localization using LSTMs for structured feature correlation. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 2, 3, 5, 7

[57] H. Zhou, D. Zou, L. Pei, R. Ying, P. Liu, and W. Yu. StructSLAM: Visual SLAM with building structure lines. *IEEE Transactions on Vehicular Technology*, 64(4):1364–1375, 2015. 1