Learning Multi-view Camera Relocalization with Graph Neural Networks

Fei Xue[1], Xin Wu[2], Shaojun Cai[1], and Junqiu Wang[3]

[1]UISEE Technology Inc.
[2]Key Laboratory of Machine Perception, Peking University
[3] AVIC Beijing Changcheng Aeronautical Measurement and Control Technology Research Institute
{fei.xue, shaojun.cai}@uisee.com
wuxin1998@pku.edu.cn, jerywangjq@foxmail.com

Abstract

We propose to construct a view graph to excavate the information of the whole given sequence for absolute camera pose estimation. Specifically, we harness GNNs to model the graph, allowing even non-consecutive frames to exchange information with each other. Rather than adopting the regular GNNs directly, we redefine the nodes, edges, and embedded functions to fit the relocalization task. Redesigned GNNs collaborate with CNNs in guiding knowledge propagation and feature extraction respectively to process multi-view high-dimensional image features iteratively at different levels. Besides, a general graph-based loss function beyond constraints between consecutive views is employed for training the network in an end-to-end fashion. Extensive experiments conducted on both indoor and outdoor datasets demonstrate that our method outperforms previous approaches especially in large-scale and challenging scenarios. Our code is publicly available (`https://github.com/feixue94/grnet`*).*

1. Introduction

Visual relocalization estimates the absolute pose of the camera in a known scene from an image or multiple images. It plays an important role in various applications including robotics, autonomous driving, and virtual/augmented reality. In the last two decades, many geometry-based algorithms [37, 38, 23, 30, 29, 35] have been proposed. Recent studies formulate the task as a regression problem using Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). Most of them estimate the camera pose from a single image [18, 25, 17, 19, 5, 46, 44, 16, 4]. Therefore, their performances degrade severely due to visual ambiguities in challenging conditions, *e.g.*, textureless regions, repetitive textures, local similarities, dynamic objects, and various weather states. In contrast to single images, sequential images provide more information from

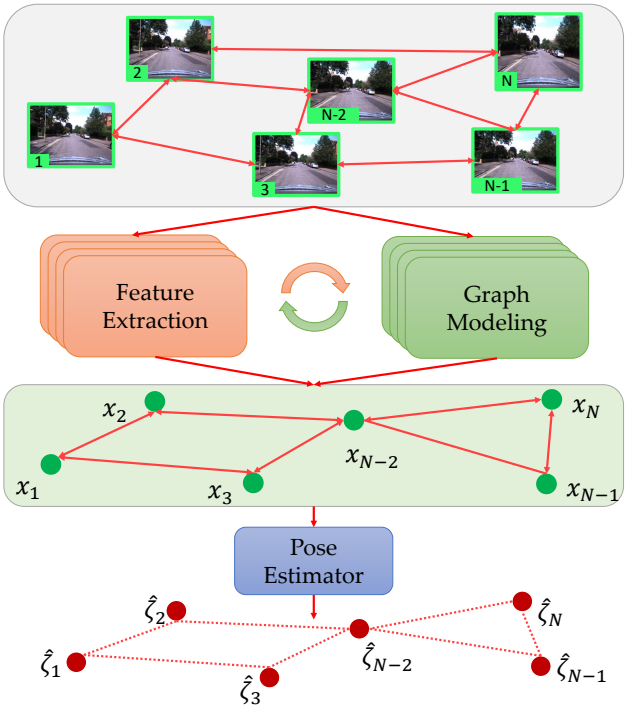


Figure 1: Overview of our framework. We construct a graph with nodes representing observations of different viewpoints. CNNs and GNNs are exploited for feature extraction and message exchange, respectively. The two processes are executed iteratively at multiple levels with edges dynamically adjusted to retain the most important connections. Enhanced feature of each node is fed into the pose estimator for absolute pose prediction. The preserved edges are utilized to build additional constraints in the training process.

different viewpoints, and thus can improve both the robustness and accuracy of relocalization.

To deal with visual ambiguities, temporal consistency is utilized by fusing sequential observations via LSTMs (Long-Short Term Memories) [50, 9] or adding VO (visual odometry) results as constraints to guarantee the consistency [3, 40, 50, 28, 9]. Unfortunately, due to limitations of LSTMs, temporal consistency is maintained in short-term periods [36]. As consecutive frames are much overlapped, temporally close views bring little new information to each other, leading to inadequate usage of multiple views for resolving visual ambiguities [48, 49].

Inspired by classic 3D reconstruction and SLAM systems [32, 26], we try to mine the correlation of all potentially related frames by constructing a view graph where observations are represented as nodes, and thus even temporally far images can contribute to each other. We harness the prevalent GNNs (Graph Neural Networks) which have been widely utilized to tackle tasks with unstructured inputs [51, 13, 41, 22, 45, 52], to process the relationships of multiple views. Considering the significant differences between relocalization and other tasks, we redesign the nodes, edges, and embedded functions of regular GNNs to fit the relocalization task. Moreover, to facilitate information propagation among multiple frames, we take advantages of CNNs and GNNs in governing feature extraction and knowledge exchange respectively to process the high-dimensional image features progressively at multiple levels.

An overview of our framework is shown in Fig. 1. Images are initially organized in a graph. All features in the graph are represented as 3D tensors so that the spatial correlations can be retained. Taking these 3D tensors as input, CNNs and GNNs cooperate to extract features and propagate messages along edges, respectively. The two processes are executed iteratively at multiple levels with both feature resolution and number of edges reduced. Finally, fused features absorbing the knowledge of other views are utilized to regress the absolute poses. The whole process is differentiable and can be trained and tested in an end-to-end fashion. Our contributions are summarized as follows:

- To the best of our knowledge, we are the first to tackle the multi-view camera relocalization task with GNNs, enabling the messages of different frames to be transferred beyond temporal connections.
- We redefine the nodes, edges, and embedded functions of GNNs to fit the relocalization task, allowing more effective information exchange between different frames.
- We leverage GNNs and CNNs for feature extraction and graph modeling respectively, which process multi-view image features iteratively at different levels to enhance the performance of our relocalization system.

We organize the rest of this paper as follows. In Sec. 2, we describe the related works and GNNs. In Sec. 3, our framework is introduced in detail. We compare our algorithm with previous methods in Sec. 4 and conclude the paper in Sec. 5.

2. Related Work

We mainly discuss visual relocalization methods in this section, and then give a brief introduction to GNNs along with their applications in computer vision tasks.

2.1. Visual Relocalization

The absolute pose of a camera can be recovered approximately using pose of the most similar image in the database [39, 1, 29] or further estimating the relative pose between the query image and the candidate to improve the accuracy [2, 10, 21]. However, query images may differ drastically from those in the database due to changing environments, making these methods prone to failure. Structure-based algorithms build 3D maps of the scene and then establish 2D-3D correspondences. Poses are calculated by performing the RANSAC and solving a Perspective-n-Point problem [8, 30, 35]. Despite their promising performance, they require accurate intrinsic calibration and intricate initialization to establish correct correspondences. Moreover, when dealing with large-scale scenes, they need excessive amounts of memory storage and computational resource in order to build a complete 3D model, which further impair their feasibility in real-world applications.

Recently, direct absolute pose regression with CNNs has achieved promising performance. PoseNet [19] and its variations [17, 18, 25, 42, 5, 44, 46, 16, 4] calculate poses from single images. These methods suffer from high uncertainty in challenging scenarios, e.g., texture-less regions, local similarities, and illumination/weather variations. Since sequential images contain more information than a single frame, they are found helpful for mitigating such ambiguities [9, 3, 50]. VidLoc [9] models image sequences using bidirectional LSTMs [33] so that observations from multiple images can be fused for pose estimation. MapNet [3] introduces additional information from IMU, GPS, and visual SLAM systems [11] as constraints to guarantee the pose consistency between consecutive frames. VLocNet [40] leverages the temporal consistency of two nearby views and imposes relative poses as constraints. LsG [50] proposes to support global pose recovery with the local properties of sequential images. It achieves state-of-the-art performance because the uncertainty of pose estimation is alleviated by augmenting the observations in local maps and optimizing poses in VO streams.

Previous methods mainly emphasize the temporal consistency by modeling the multi-view images via LSTMs or imposing relative constraints on consecutive frames. How-

ever, this unidirectional process cannot take full advantages of multi-view images. Besides, LSTMs are incapable of keeping memories for a long time [36], causing the loss of global knowledge. Instead, we formulate the multi-view images using graphs and exploit GNNs to propagate information between different views more adequately.

2.2. Graph Neural Network

GNNs model the relationships of unstructured data by passing messages between connected nodes and have been investigated intensively in the literature [52, 47]. Recently, GNNs are getting increasingly popular for their outstanding performance on tackling computer vision tasks including classification [51, 13, 41], segmentation [45, 43], human pose estimation [6], and scene representation [53]. However, visual relocalization is essentially different from these tasks as the nodes are defined on high-dimensional image features rather than vectors, and edges are utilized to exchange information between two views. Adopting regular GNNs directly by compressing image features to vectors leads to a large amount of information loss. In addition, this process introduces a large number of fully-connected layers, which further cause over-fitting and instability of the model [22].

In this paper, we employ GNNs to handle the topological relationships of multiple views. Rather than employing the regular GNN directly, we redefine the nodes, edges, and embedded functions of GNNs to fit the relocalization task.

3. Method

We first introduce how to formulate the sequence-based relocalization task using a graph in Sec. 3.1. Next, we describe the message exchange protocol among multiple frames in Sec. 3.2. Finally, we define a graph-based loss function in Sec. 3.4.

3.1. Graph Definition

A graph is usually defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = \{v_i\}$ and $\mathcal{E} = \{e_{ij}\}$ representing the nodes and edges, respectively. Each node v_i contains a feature x_i denoting its attribute and each edge $e_{ij} = (v_i, v_j)$ connects v_i and v_j. The overall process of our network can be described as:

$$\mathcal{G}_{pose} = \mathcal{F}(\mathcal{G}_{img}) \,, \tag{1}$$

where $\mathcal{G}_{img} = (\mathcal{V}_{img}, \mathcal{E}_{img})$ and $\mathcal{G}_{pose} = (\mathcal{V}_{pose}, \mathcal{E}_{pose})$ represent the input and output graphs, respectively. $\mathcal{F}$ is the model proposed in this paper.

Node Definition. Instead of compressing features to vectors in standard GNNs, we define the representation of each node as a 3D tensor considering the spatial connections of image features, which are crucial for knowledge sharing between different views. Thus, each feature map

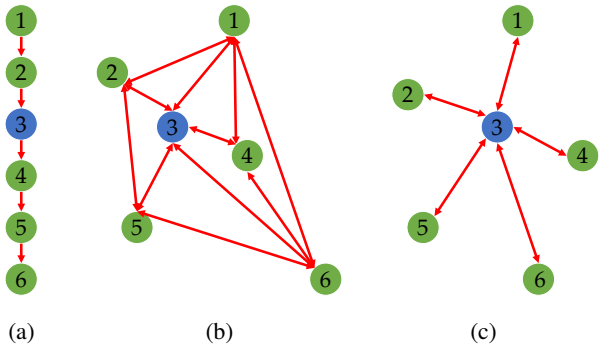


Figure 2: Graph structure. Previous methods [3, 9, 50] focus mainly on the temporal consistency by adding connections between consecutive frames (a). In contrast, we initialize the graph with dense connections (b), so that even temporally distant views can contribute to each other (c).

$x_i \in \mathcal{R}^{H \times W \times C}$ (H, W, and C denote the height, width, and channels) can be directly put into the graph without losing any information. Accordingly, we redefine the message passing and node updating functions so that information can be propagated in the formulation of 3D tensors.

Edge Initialization. As shown in Fig 2, each edge determines which two views should be connected and allows them to exchange information. Traditionally, edges are defined between consecutive frames as $\mathcal{E} = \{e_{ij} \,|\, |i-j| = 1\}$ (in Fig. 2a) and LSTMs are adopted to model the sequential images along the temporal direction or the inverse. Theoretically, two views can potentially contribute to each other if they share enough content overlap, which can be treated as priors defined as the *attribute* of edges. However, such priors are unachievable without depth information [2].

In order to retain the potential connections, we initially construct a dense direct graph by connecting every two nodes with edges (in Fig. 2b). Therefore, even temporally far images can contribute to each other (in Fig. 2c).

It can be seen clearly that modeling the sequence using LSTMs in [9, 50] is just a special case of graph model proposed in this paper. The graph model allows the global information of the whole sequence to be transferred to each node, which is unattainable in LSTMs.

3.2. Message Passing

Once the graph is built, the key becomes how to transfer valuable information from the *source* nodes to the *target* ones. In the following, we introduce how to produce messages and how to aggregate them, as shown in Fig. 3.

Message Generation. The message provided by node v_j to node v_i carries the correlation between image I_j and I_i. We calculate the message from node features using convolution operations so that the spatial correlations are retained.

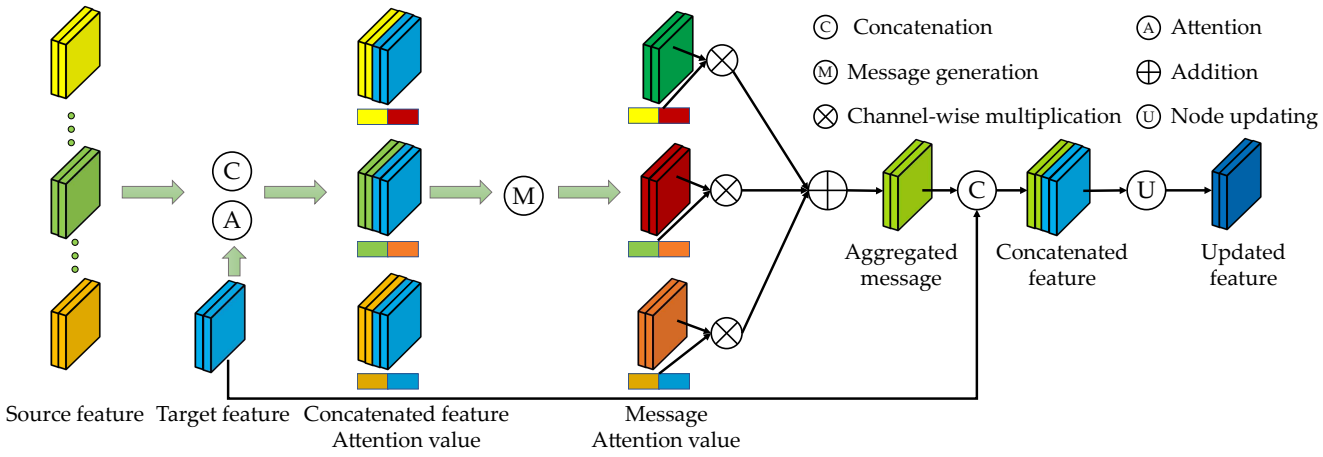


Figure 3: The pipeline of message passing and node updating. Each source feature concatenated with the target one is utilized to generate the message which is then re-weighted along the channel dimension for selection. Messages from all source features are aggregated by adding pixel-wise values and concatenated with the target feature for the final node updating.

The message generation function is defined as:

$$m_{j\to i} = f_m(x_i, x_j) \,, \tag{2}$$

where f_m is the function calculating message $m_{j\to i}$ from x_i and x_j. x_i and x_j are first concatenated along the channel dimension and then fed into two convolutional layers with kernel size of 3×3, padding size of 1 to produce the message. The output channels of both two layers are the same as x_i, hence $m_{j\to i}$ shares the same size with x_i and x_j.

Message Selection. Different views contribute discriminatively, resulting in some messages being more important than others. Hence, we employ a soft attention mechanism to calculate the channel-wise correlations between connected nodes. The soft attention is performed as:

$$a_{j\to i} = f_{atten}(x_i, x_j) \,, \tag{3}$$

$$a^{(k)}_{j\to i} = \sigma(cs(vec(x^{(k)}_i), vec(x^{(k)}_j))) \,, \tag{4}$$

where f_{atten} calculates the cosine similarity cs between each corresponding *vectorized* kth channel of connected features x_i and x_j. The attention value is then normalized to $(0, 1)$ by the sigmoid function σ.

Message Aggregation. The final message m^{agg}_i for node v_i is aggregated by gathering messages from all source nodes with soft attentions, as:

$$m^{agg}_i = \frac{1}{N_i} \sum_{e_{ij}\in\mathcal{E}} a_{j\to i} \otimes m_{j\to i} \,, \tag{5}$$

where N_i is the number of source nodes connected to node v_i and $\otimes$ denotes the channel-wise multiplication.

Node Updating. The feature of node v_i is augmented by the aggregated message m^{agg}_i as:

$$x^{'}_i = f_u(x_i, m^{agg}_i) \,, \tag{6}$$

where f_u is the updating function sharing the same structure with f_m, yet using another group of parameters. x_i and m^{agg}_i are first passed through a convolutional layer with kernel size of 1×1 individually and then concatenated along the channel dimension and fed into f_u to calculate the updated feature $x^{'}_i$.

3.3. Multi-level Dynamic Updating

The redesigned message passing and node updating functions facilitate the cooperation between GNNs and CNNs in processing image features. To promote the information sharing among multiple views, we extend the framework to operate on multi-level features progressively with additional edge pooling incorporated to abandon redundant connections, as shown in Fig. 4.

Multi-level Graph Modeling. The interaction among different views is computed iteratively as:

$$\mathcal{V}^l = f^l_c(\mathcal{V}^{l-1}) \,, \tag{7}$$

$$\mathcal{V}^l_{fused}, \mathcal{E}^l = f^l_g(\mathcal{V}^l, \mathcal{E}^{l-1}) \,, \tag{8}$$

where $\mathcal{V}^l$, $\mathcal{E}^l$, $\mathcal{V}^{l-1}$, and $\mathcal{E}^{l-1}$ indicate the nodes and edges at the lth and the previous level, respectively. $\mathcal{V}^l_{fused}$ denotes the nodes with fused information.

As shown in Fig. 4, ResNet34 [15] is divided into 4 parts as f^1_c, f^2_c, f^3_c, and f^4_c at the end of 4 groups of *residual layers*. Each f^i_c is followed by a GNN block f^i_g for graph modeling.

Adaptive Edge Pooling. Densely initialized edges retain the potential connections, yet inevitably lead to redundancy, extra computational time, over-fitting, and thus degrade performance of the whole system. Moreover, as the information is propagated among multiple views from the low to high levels progressively, there is no need to keep

connections with small values. Here, we enable the network to make the decision adaptively according to the correlation between connected nodes, as:

$$c_{j\to i} = cs(maxpool(x_i), maxpool(x_j)) \ , \tag{9}$$

where the correlation $c_{j\to i}$ is calculated with cosine similarity on the downsampled features of x_i and x_j. For each node, only k neighbors with the largest correlations are preserved while the rest are discarded.

Pose Estimation. Each node gives an absolute pose calculated from its updated feature $x_i^{'}$. Instead of estimating the pose from the output of the last GNN block which contains only high-level information, we aggregate fused features from the output of all GNN blocks and use the aggregated feature to estimate the absolute pose. Therefore, both low and high level visual clues can contribute to the final calculation. As shown in Fig. 4, outputs of GNN blocks are first passed through a GAP (global average pooling) layer for downsampling and then concatenated along the channel dimension, and finally fed into two FC (fully connected) layers to predict the position $\hat{\mathbf{t}}_i \in \mathcal{R}^3$ and orientation $\hat{\mathbf{r}}_i \in \mathcal{R}^4$ (quaternion) for image I_i, respectively.

3.4. Graph-based Loss Function

The output of our model contains the predicted poses and connections between them. The loss function is designed considering both of them, as:

$$\mathcal{L} = \frac{1}{N_v} \sum_{v_i \in \mathcal{V}_{pose}} d(\zeta_i, \hat{\zeta}_i) + \frac{1}{N_e} \sum_{e_{ij} \in \mathcal{E}_{pose}} d(\omega_{ij}, \hat{\omega}_{ij}) \ , \tag{10}$$

where $\zeta_i = (\mathbf{t}_i, \mathbf{r}_i)$ and $\hat{\zeta}_i = (\hat{\mathbf{t}}_i, \hat{\mathbf{r}}_i)$ are the ground-truth and predicted 6-DoF absolute poses. $\mathbf{t}$ and $\mathbf{r}$ denote the position and orientation, respectively. $\hat{\omega}_{ij}$ is the relative pose between predicted poses $\hat{\zeta}_i$ and $\hat{\zeta}_j$ if e_{ij} exists in the output edges. $d(.)$ is the geometric loss function adopted to balance the position and orientation errors as in [18, 3, 50]. $d(.)$ is defined as:

$$d(\zeta_i, \hat{\zeta}_i) = ||\mathbf{t}_i - \hat{\mathbf{t}}_i||_1 e^{-\beta_p} + \beta_p + ||\mathbf{r}_i - \hat{\mathbf{r}}_i||_1 e^{-\gamma_p} + \gamma_p \ , \tag{11}$$

where β_p and γ_p are the parameters balancing the position and orientation errors. We adopt another group of β_g and γ_g to balance the translations and rotations in relative poses. Both βs and γs are optimized jointly with the parameters in the neural networks.

It is noteworthy that our graph-based loss function introduces constraints among multiple views beyond solely consecutive poses in MapNet [3] and LsG [50]. Therefore, the loss function is close to the standard pose-graph [7], which has been an indispensable component of current reconstruction and SLAM systems [26, 32] to improve the pose accuracy.

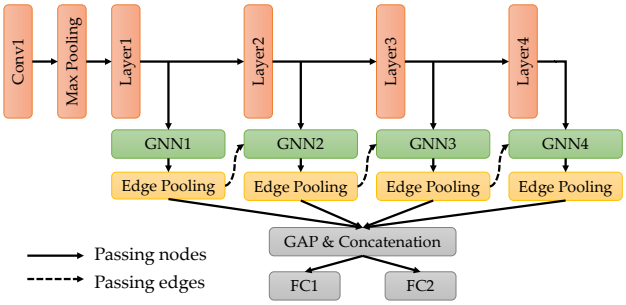


Figure 4: Multi-level modeling. ResNet34 [15] is integrated with GNNs for iterative feature processing at four levels. After each GNN block, an edge pooling is performed to discard redundant connections. Fused features from four GNN blocks are passed through the GAP (global average pooling) layer, concatenated along the channel dimension, and fed into two FC (fully connected) layers for position and orientation prediction, respectively.

4. Experiments

We first describe the implementation details, then introduce the datasets used for evaluation in Sec. 4.1 and the performance of our approach against previous methods on the 7Scenes [34], Cambridge [19], and Oxford RobotCar [24] datasets in Sec. 4.2, Sec. 4.3, and Sec. 4.4, respectively. Finally, we perform an ablation study to test the efficacy of each component in our framework in Sec. 4.5. More details and results can be found in our **Supplementary Material**.

Implementation. The input of our model is (but not limited to) monocular RGB sequential images. The length is 8 for all datasets, while our network is flexible to accept sequences of arbitrary lengths. Following [3, 50, 18], all images are resized with the height as 256 and normalized by subtracting mean pixel values. The βs and γs are set to -4.0 and -1.0 for initialization, respectively. k is set to 8, 8, 6, and 4 for four levels. ResNet34 [15] is pretrained on the ImageNet, while the rest are initialized with the MSRA method [14]. We implement our model using the prevalent pytorch-geometric [12] on the NVIDIA GeForce GTX 1080TI GPU. The Adam [20] with initial learning rate 10^{-4} is utilized to optimize the parameters with batch size of 8 for 200 epochs in total.

Baseline Methods. We compare our model against previous networks that regress absolute camera poses in an end-to-end fashion. Both image-based [19, 17, 18, 25, 5, 46, 44, 42, 16, 4]) and sequence-based algorithms (VidLoc [9], MapNet [3], and LsG [50]) are considered.

4.1. Dataset

We conduct experiments on three public datasets with different scales and conditions including 7Scenes [34],

Method	Sequence							
	Chess $6m^2$	Fire $2.5m^2$	Heads $1m^2$	Office $7.5m^2$	Pumpkin $5m^2$	Kitchen $18m^2$	Stairs $7.5m^2$	Avg
PoseNet15 [19]	0.32m, 8.12°	0.47m, 14.4°	0.29m, 12.0°	0.48m, 7.68°	0.47m, 8.42°	0.59m, 8.64°	0.47m, 13.8°	0.44m, 10.4°
PoseNet16 [17]	0.37m, 7.24°	0.43m, 13.7°	0.31m, 12.0°	0.48m, 8.04°	0.61m, 7.08°	0.58m, 7.54°	0.48m, 13.1°	0.47m, 9.81°
PoseNet17 [18]	0.14m, 4.50°	0.27m, 11.80°	0.18m, 12.10°	0.20m, 5.77°	0.25m, 4.82°	0.24m, 5.52°	0.37m, 10.60°	0.24m, 7.87°
PoseNet17 (geo) [18]	0.13m, 4.48°	0.27m, 11.30°	0.17m, 13.00°	0.19m, 5.55°	0.26m, 4.75°	0.23m, 5.35°	0.35m, 12.40°	0.23m, 8.12°
Hourglass [25]	0.15m, 6.17°	0.27m, 10.84°	0.19m, 11.63°	0.21m, 8.48°	0.25m, 7.01°	0.27m, 10.15°	0.29m, 12.46°	0.23m, 9.53°
LSTM-Pose [42]	0.24m, 5.77°	0.34m, 11.9°	0.21m, 13.7°	0.30m, 8.08°	0.33m, 7.00°	0.37m, 8.83°	0.40m, 13.7°	0.31m, 9.85°
ANNet [4]	0.12m, 4.30°	0.27m, 11.60°	0.16m, 12.40°	0.19m, 6.80°	0.21m, 5.20°	0.25m, 6.00°	0.28m, **8.40**°	0.21m, 7.90°
BranchNet [46]	0.18m, 5.17°	0.34m, 8.99°	0.20m, 14.15°	0.30m, 7.05°	0.27m, 5.10°	0.33m, 7.40°	0.38m, 10.26°	0.29m, 8.30°
GPoseNet [5]	0.20m, 7.11°	0.38m, 12.3°	0.21m, 13.8°	0.28m, 8.83°	0.37m, 6.94°	0.35m, 8.15°	0.37m, 12.5°	0.31m, 9.95°
MLFBPPose [44]	0.12m, 5.82°	**0.26**m, 11.99°	**0.14**m, 13.54°	**0.18**m, 8.24°	0.21m, 7.05°	**0.22**m, 8.14°	0.38m, 10.26°	0.22m, 9.29°
Ours	**0.08**m, **2.82**°	**0.26**m, **8.94**°	0.17m, **11.41**°	**0.18**m, **5.08**°	**0.15**m, **2.77**°	0.25m, **4.48**°	**0.23**m, 8.78°	**0.19**m, **6.33**°
VidLoc [9]	0.18m, NA	**0.26**m, NA	**0.14**m, NA	0.26m, NA	0.36m, NA	0.31m, NA	0.26m, NA	0.25m, NA
MapNet [3]	**0.08**m, 3.25°	0.27m, 11.69°	0.18m, 13.25°	**0.17**m, 5.15°	0.22m, 4.02°	**0.23**m, 4.93°	0.30m, 12.08°	0.21m, 7.77°
LsG [50]	0.09m, 3.28°	**0.26**m, 10.92°	0.17m, 12.70°	0.18m, 5.45°	0.20m, 3.69°	**0.23**m, 4.92°	**0.23**m, 11.3°	**0.19**m, 7.47°
Ours	**0.08**m, **2.82**°	**0.26**m, **8.94**°	0.17m, **11.41**°	0.18m, **5.08**°	**0.15**m, **2.77**°	0.25m, **4.48**°	**0.23**m, **8.78**°	**0.19**m, **6.33**°

Table 1: Median errors of image- and sequence-based methods on the 7Scenes dataset [34]. The best results are highlighted.

Method	Sequence						
	College $5.6 \times 10^3 m^2$	Shop $8.8 \times 10^3 m^2$	Church $4.8 \times 10^3 m^2$	Hospital $2.0 \times 10^3 m^2$	Avg	Court $8 \times 10^3 m^2$	Street $5.0 \times 10^3 m^2$
ADPoseNet [16]	1.30m, 1.67°	1.22m, 6.7°	2.28m, 4.80°	-	1.60m, 4.21°	-	-
PoseNet15 [19]	1.66m, 4.86°	1.41m, 7.18°	2.45m, 7.96°	2.62m, 4.90°	2.04m, 6.23°	-	-
PoseNet16 [17]	1.74m, 4.06°	1.25m, 7.54°	2.11m, 8.38°	2.57m, 5.14°	1.92m, 6.28°	-	-
LSTM-Pose [42]	0.99m, 3.65°	1.18m, 7.44°	1.52m, 6.68°	1.51m, 4.29°	1.30m, 5.52°	-	-
GPoseNet [5]	1.61m, 2.29°	1.14m, 5.73°	2.93m, 6.46°	2.62m, 3.89°	2.08m, 4.59°	-	-
SVS-Pose [27]	1.06m, 2.81°	0.63m, 5.73°	2.11m, 8.11°	**1.50**m, 4.03°	1.33m, 5.17°	-	-
MLFBPPose [44]	0.76m, 1.72°	0.75m, 5.10°	**1.29**m, 5.01°	1.99m, 2.85°	1.20m, 3.67°	-	-
PoseNet17 [18]	0.99m, 1.06°	1.05m, 3.97°	1.49m, 3.43°	2.17m, 2.94°	1.43m, 2.85°	7.00m, 3.65°	20.70m, 25.70°
PoseNet17 (geo) [18]	0.88m, 1.04°	0.88m, 3.78°	1.57m, 3.32°	3.20m, 3.29°	1.63m, 2.86°	6.83m, 3.47°	20.30m, 25.50°
MapNet [3]	1.07m, 1.89°	1.49m, 4.22°	2.00m, 4.53°	1.94m, 3.91°	1.63m, 3.64°	7.85m, 3.76°	22.23m, 27.55°
Ours	**0.59**m, **0.65**°	**0.50**m, **2.87**°	1.90m, **3.29**°	1.88m, **2.78**°	**1.12**m, **2.40**°	**6.67**m, **2.79**°	**14.72**m, **22.44**°

Table 2: Median errors on the Cambridge dataset [19]. Results of MapNet [3] on College, Shop, Church, and Hospital are from [31] and others are from retrained model. " $-$ " denotes no data provided. The best results are highlighted.

Cambridge [19], and Oxford RobotCar [24] datasets.

7Scenes. The 7Scenes dataset [34] was recorded continuously by a kinect in seven different indoor environments. It contains scenes with highly repetitive textures (Chess and Stairs) and many textureless regions (Fire and Pumpkin), and thus is very challenging for relocalization algorithms.

Cambridge. The Cambridge dataset [19] was captured by a mobile phone in urban environments. Dynamic objects, various illumination and weather conditions, and distinct walking paths between training and testing sequences make the dataset more difficult for relocalization methods.

Oxford RobotCar. The Oxford RobotCar dataset was collected by a car driving in the central Oxford city. It contains much longer trajectories (from $1 \times 10^3 m$ to $9 \times 10^3 m$) and larger areas (from $8 \times 10^4 m^2$ to $1 \times 10^6 m^2$) than 7Scenes and Cambridge benchmarks, and hence is extremely challenging for relocalization algorithms. Additionally, this dataset consists of images captured under various illuminations, weather conditions, and dynamic objects, which may further increase the relocalization difficulties. We follow the train/test split in [3, 50] and evaluate the performance of our approach on the LOOP and FULL scenes.

4.2. Experiments on the 7Scenes Dataset

Table 1 shows the quantitative results of our method and previous image- and sequence-based approaches. It can be seen that our model outperforms PoseNet [19] and its variations [17, 18, 25, 42] considerably in terms of both position and orientation prediction, especially in scenes with texture-less regions (Fire and Pumpkin) and highly repetitive textures (Chess and Stairs). That is because estimating poses from single images suffers severely from visual ambiguities resulting from texture-less regions and highly repetitive textures. These problems can be effectively mitigated by exploiting correlations of multiple frames.

Additionally, we can find that our model performs much

Method	Scene				
	LOOP1	LOOP2	FULL1	FULL2	Avg
	$8.8 \times 10^4 m^2$	$8.8 \times 10^4 m^2$	$1.2 \times 10^6 m^2$	$1.2 \times 10^6 m^2$	
PoseNet [18, 19, 17]	28.81m, 19.62°	25.29m, 17.45°	125.6m, 27.1°	131.06m, 26.05°	77.85m, 22.56°
MapNet [3]	8.76m, 3.46°	9.84m, 3.96°	41.40m, 12.50°	59.30m, 14.81°	29.83m, 8.68°
LsG [50]	9.07m, 3.31°	9.19m, 3.53°	31.65m, 4.51°	53.45m, 8.60°	25.84m, 4.99°
Ours	**7.76**m, **2.54**°	**8.15**m, **2.57**°	**17.35**m, **3.47**°	**37.81**m, **7.55**°	**17.77**m, **4.03**°
Ours (s)	23.43m, 9.75°	24.65m, 10.55°	34.47m, 4.07°	58.16m, 9.73°	35.18m, 8.53°
Ours (l_4)	10.60m, 4.54°	10.77m, 4.12°	20.26m, 4.78°	39.57m, 8.05°	20.30m, 5.37°
Ours (l_4 + $p.g.$)	9.07m, 3.15°	9.16m, 3.22°	19.70m, 4.46°	39.83m, 8.17°	19.44m, 4.75°
Ours ($l_{4,3}$ + $p.g.$)	8.49m, 3.11°	8.62m, 3.19°	18.76m, 4.35°	38.76m, 9.41°	18.66m, 5.02°
Ours ($l_{4,3,2}$ + $p.g.$)	8.46m, 3.02°	**7.68**m, 2.78°	**17.35**m, 3.59°	**36.84**m, 8.22°	**17.58**m, 4.40°
Ours ($l_{4,3,2,1}$ + $p.g.$)	**7.76**m, **2.54**°	8.15m, **2.57**°	**17.35**m, **3.47**°	37.81m, **7.55**°	17.77m, **4.03**°

Table 3: Mean errors on the Oxford RobotCar dataset [24]. Ours (s) is the network using standard GNNs. l_i and $p.g.$ denote the model using the ith level GNN block (see Fig. 4) with pose graph as additional constraint. The best results are highlighted.

better than sequence-based methods including VidLoc [9], MapNet [3], and LsG [50] in orientation estimation. The improvement comes from the adequate message passing among multiple images via the graph modeling. Recovered trajectories can be found in our **Supplementary Materials**.

4.3. Experiments on the Cambridge Dataset

Covering large regions of outdoor scenes, the Cambridge dataset [19] validates the potential of our method in handling large-scale challenging conditions. As VidLoc and LsG didn't report results on this dataset, we compare our approach with image-based methods and MapNet. As shown in Table 2, our model outperforms previous networks consistently with large margins. Note that the improvement gets progressively larger as the size of the scene increases (College, Shop, Court, and Street).

4.4. Experiments on the RobotCar Dataset

Table 3 demonstrates the quantitative comparison between previous methods and our model. Both position and orientation errors are considerably reduced by the proposed graph-based relocalization system. As the FULL1-2 sequences contain an area of $1.2 \times 10^6 m^2$ with the length of trajectories up to $9562m$, these sequences are extremely challenging for relocalization algorithms. From Table 3 we can see that our framework is more effective in handling these challenges than other methods.

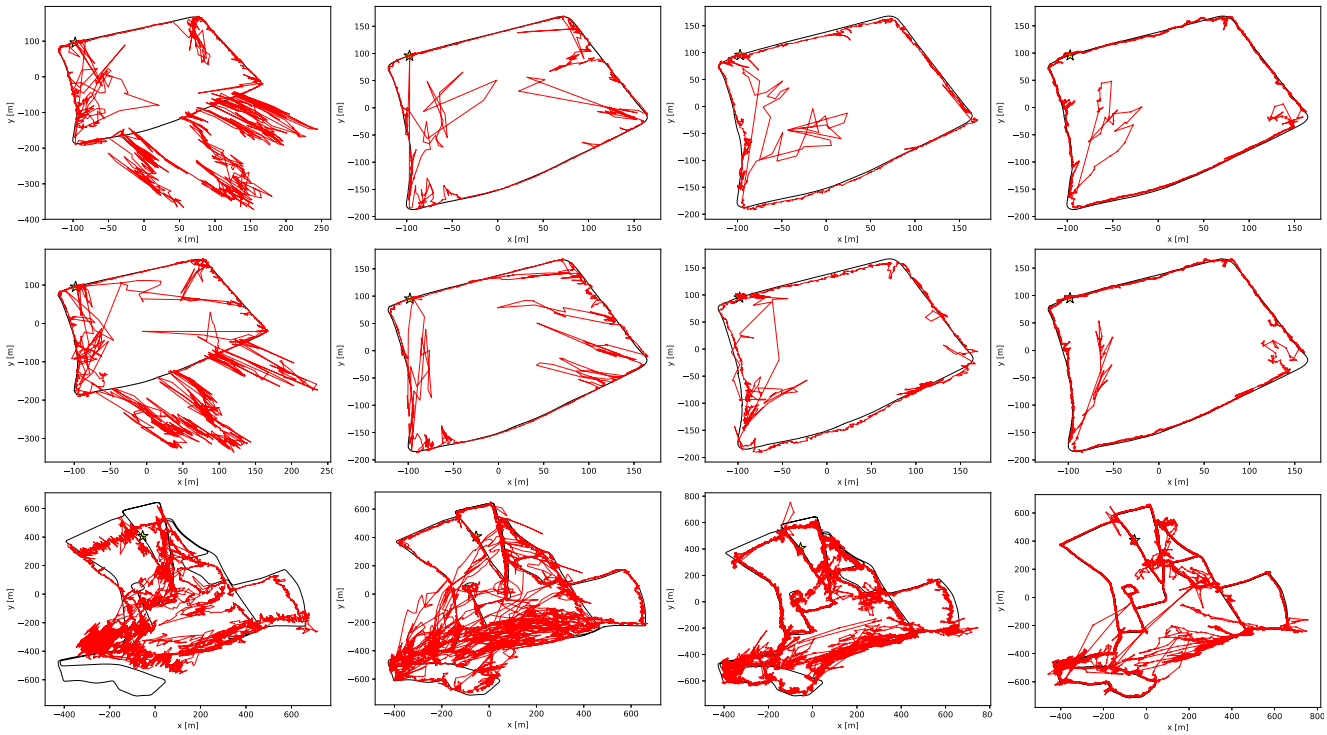
Fig. 5 illustrates the trajectories recovered by PoseNet [19, 17, 18], MapNet [3], LsG [50], and our method. PoseNet produces inaccurate estimations with lots of outliers due to local similarities and over-exposure. MapNet yields more accurate results and reduces many outliers by introducing relative poses between consecutive frames as additional constraints. However, there still exist a large number of unstable predictions, especially in the FULL scenes. By employing content augmentation, although LsG ameliorates this problem to a certain extent, it scarifies the accuracy. In contrast, our approach reduces the number of outliers more effectively than LsG and guarantees the accuracy at the same time. This improvement can be seen in both LOOP and FULL sequences.

We additionally calculate the cumulative distribution errors to further compare the performance regarding both position and orientation estimation in Fig. 6. Fig. 6b and 6d show that our method outperforms PoseNet, MapNet, and LsG in orientation estimation consistently in both the LOOP and FULL sequences. Fig. 6a and 6c further verify the efficacy of our model for retaining the accuracy and reducing the number of outliers. Besides, the two figures also reveal the limitations of MapNet and LsG in dealing with hard and simple cases, respectively. Fortunately, both of these two cases can be effectively handled by our approach.

4.5. Ablation Study

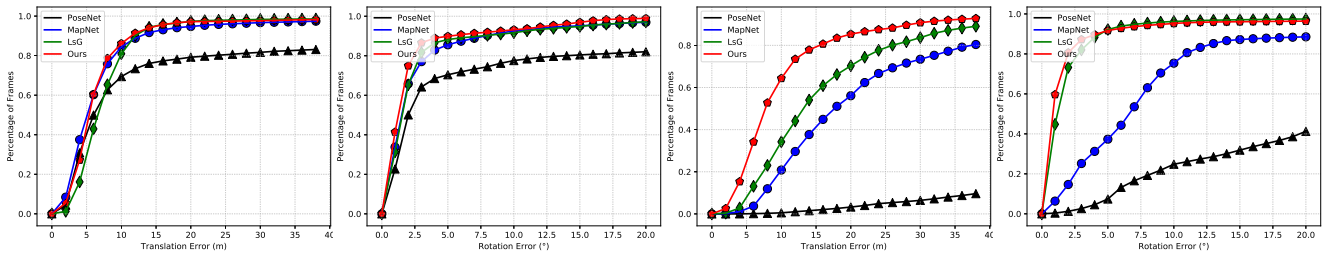
We conduct an ablation study to validate the effectiveness of redesigned GNN, multi-scale modeling, and the graph-based loss function. As shown in Table 3, compared with PoseNet [19, 17, 18], standard GNN (Ours (s)) can boost the performance by exchanging information among different views, while the redefined GNN can yield more accurate results by sharing knowledge in the formulation of 3D tensors (Ours (l_4)), which can be further improved by the pose graph (Ours ($l_4 + p.g.$)).

From Table 3, we can see that even our network with only a single level of GNN block (Ours ($l_4 + p.g.$)) outperforms PoseNet [19, 17, 18], MapNet [3], and LsG [50] considerably, especially in larger environments (Full1-2). The improvement on position is much more significant than orientation because the Oxford RobotCar dataset was collected by moving cars, and its rotation patterns are relatively sim-



PoseNet [19, 17, 18] MapNet [3] LsG [50] **Ours**

Figure 5: Trajectories of PoseNet, MapNet, LsG, and our model on the LOOP1 (top), LOOP2 (middle), and FULL1 scenes (bottom) of the Oxford RobotCar dataset [24] The red and black lines indicate predicted and ground truth poses, respectively.



(a) Translation error on LOOP. (b) Rotation error on LOOP. (c) Translation error on FULL. (d) Rotation error on FULL.

Figure 6: Cumulative distributions of the mean position and orientation errors of PoseNet, MapNet, LsG, and our method on the Oxford RobotCar LOOP and FULL scenes. X-axis and y-axis denote the error and the percentage of frames, respectively.

ple than those captured by handheld cameras (7Scenes and Cambridge datasets). Both orientation and position errors are reduced progressively by incorporating multiple GNN blocks for information propagation.

5. Conclusion

In this paper, we propose to estimate the absolute camera poses from multiple images. In contrast to previous works which rely heavily on the temporal consistency, we construct a view graph to exploit the dependency among multiple frames and leverage GNNs for graph modeling. Rather than adopting the regular GNNs with brute-force, we redefine the nodes and edges and redesign the embedded functions for message passing and node updating to fit the relocalization task. Moreover, we take full advantages of CNNs and GNNs in feature extraction and knowledge propagation respectively to process high-dimensional image features at multiple levels. Finally, we employ a graph-based loss function to train our model in an end-to-end fashion. We conduct extensive experiments on the 7Scenes, Cambridge, and Oxford RobotCar datasets. Results demonstrate that our approach outperforms previous methods, especially in large-scale and challenging scenarios.

References

[1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. NetVLAD: CNN architecture for Weakly Supervised Place Recognition. In *CVPR*, 2016.

[2] Vassileios Balntas, Shuda Li, and Victor Prisacariu. RelocNet: Continuous Metric Learning Relocalisation using Neural Nets. In *ECCV*, 2018.

[3] Samarth Brahmbhatt, Jinwei Gu, Kihwan Kim, James Hays, and Jan Kautz. MapNet: Geometry-aware Learning of Maps for Camera Localization. In *CVPR*, 2018.

[4] Mai Bui, Christoph Baur, Nassir Navab, Slobodan Ilic, and Shadi Albarqouni. Adversarial Networks for Camera Pose Regression and Refinement. In *ICCV Workshops*, 2019.

[5] Ming Cai, Chunhua Shen, and Ian Reid. A Hybrid Probabilistic Model for Camera Relocalization. In *BMVC*, 2018.

[6] Yujun Cai, Liuhao Ge, Jun Liu, Jianfei Cai, Tat-Jen Cham, Junsong Yuan, and Nadia Magnenat Thalmann. Exploiting Spatial-Temporal Relationships for 3D Pose Estimation via Graph Convolutional Networks. In *ICCV*, 2019.

[7] Giuseppe Calafiore, Luca Carlone, and Frank Dellaert. Pose Graph Optimization in the Complex Domain: Lagrangian Duality, Conditions for Zero Duality Gap, and Optimal Solutions. *T-RO*, 2016.

[8] Federico Camposeco, Torsten Sattler, Andrea Cohen, Andreas Geiger, and Marc Pollefeys. Toroidal Constraints for Two-point Localization under High Outlier Ratios. In *CVPR*, 2017.

[9] Ronald Clark, Sen Wang, Andrew Markham, Niki Trigoni, and Hongkai Wen. Vidloc: A Deep Spatio-temporal Model for 6-DOF Video-clip Relocalization. In *CVPR*, 2017.

[10] Mingyu Ding, Zhe Wang, Jiankai Sun, Jianping Shi, and Ping Luo. CamNet: Coarse-to-Fine Retrieval for Camera Re-Localization. In *ICCV*, 2019.

[11] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct Sparse Odometry. *TPAMI*, 2018.

[12] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop*, 2019.

[13] Hongyang Gao and Shuiwang Ji. Graph U-Nets. In *ICML*, 2019.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-level Performance on Imagenet Classification. In *ICCV*, 2015.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

[16] Zhaoyang Huang, Yan Xu, Jianping Shi, Xiaowei Zhou, Hujun Bao, and Guofeng Zhang. Prior Guided Dropout for Robust Visual Localization in Dynamic Environments. In *ICCV*, 2019.

[17] Alex Kendall and Roberto Cipolla. Modelling Uncertainty in Deep Learning for Camera Relocalization. In *ICRA*, 2016.

[18] Alex Kendall and Roberto Cipolla. Geometric Loss Functions for Camera Pose Regression with Deep Learning. In *CVPR*, 2017.

[19] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A Convolutional Network for Real-time 6-DoF Camera Relocalization. In *ICCV*, 2015.

[20] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

[21] Zakaria Laskar, Iaroslav Melekhov, Surya Kalia, and Juho Kannala. Camera Relocalization by Computing Pairwise Relative Poses using Convolutional Neural Network. In *ICCV*, 2017.

[22] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. Can GCNs Go as Deep as CNNs? In *ICCV*, 2019.

[23] Liu Liu, Hongdong Li, and Yuchao Dai. Efficient Global 2D-3D Matching for Camera Localization in a Large-scale 3D Map. In *ICCV*, 2017.

[24] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *IJRR*, 2017.

[25] Iaroslav Melekhov, Juha Ylioinas, Juho Kannala, and Esa Rahtu. Image-based Localization Using Hourglass Networks. In *ICCV Workshops*, 2017.

[26] Raul Mur-Artal and Juan D Tardós. ORB-SLAM2: An Open-source SLAM System for Monocular, Stereo, and RGB-D Cameras. *T-RO*, 2017.

[27] Tayyab Naseer and Wolfram Burgard. Deep Regression for Monocular Camera-based 6-DoF Global Localization in Outdoor Environments. In *IROS*, 2017.

[28] Noha Radwan, Abhinav Valada, and Wolfram Burgard. Vlocnet++: Deep Multitask Learning for Semantic Visual Localization and Odometry. *RA-L*, 2018.

[29] Torsten Sattler, Michal Havlena, Konrad Schindler, and Marc Pollefeys. Large-scale Location Recognition and the Geometric Burstiness Problem. In *CVPR*, 2016.

[30] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & Effective Prioritized Matching for Large-scale Image-based Localization. *TPAMI*, 2017.

[31] Torsten Sattler, Qunjie Zhou, Marc Pollefeys, and Laura Leal-Taixe. Understanding the Limitations of CNN-based Absolute Camera Pose Regression. In *CVPR*, 2019.

[32] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *CVPR*, 2016.

[33] Mike Schuster and Kuldip K Paliwal. Bidirectional Recurrent Neural Networks. *TSP*, 1997.

[34] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images. In *CVPR*, 2013.

[35] Pablo Speciale, Johannes L. Schonberger, Sudipta N. Sinha, and Marc Pollefeys. Privacy Preserving Image Queries for Camera Localization. In *ICCV*, 2019.

[36] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end Memory Networks. In *NIPS*, 2015.

[37] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. InLoc: Indoor Visual Localization with Dense Matching and View Synthesis. In *CVPR*, 2018.

[38] Carl Toft, Erik Stenborg, Lars Hammarstrand, Lucas Brynte, Marc Pollefeys, Torsten Sattler, and Fredrik Kahl. Semantic Match Consistency for Long-term Visual Localization. In *ECCV*, 2018.

[39] Akihiko Torii, Relja Arandjelovic, Josef Sivic, Masatoshi Okutomi, and Tomas Pajdla. 24/7 Place Recognition by View Synthesis. In *CVPR*, 2015.

[40] Abhinav Valada, Noha Radwan, and Wolfram Burgard. Deep Auxiliary Learning for Visual Localization and Odometry. In *ICRA*, 2018.

[41] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep Graph Infomax. In *ICLR*, 2019.

[42] Florian Walch, Caner Hazirbas, Laura Leal-Taixe, Torsten Sattler, Sebastian Hilsenbeck, and Daniel Cremers. Image-based Localization Using LSTMs for Structured Feature Correlation. In *ICCV*, 2017.

[43] Wenguan Wang, Xiankai Lu, Jianbing Shen, David J. Crandall, and Ling Shao. Zero-Shot Video Object Segmentation via Attentive Graph Neural Networks. In *ICCV*, 2019.

[44] Xin Wang, Xiang Wang, Chen Wang, Xiao Bai, Jing Wu, and Edwin R Hancock. Discriminative Features Matter: Multi-layer Bilinear Pooling for Camera Localization. In *BMVC*, 2019.

[45] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic Graph CNN for Learning on Point Clouds. *ACM TOG*, 2019.

[46] Jian Wu, Liwei Ma, and Xiaolin Hu. Delving Deeper into Convolutional Neural Networks for Camera Relocalization. In *ICRA*, 2017.

[47] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A Comprehensive Survey on Graph Neural Networks. *arXiv preprint arXiv:1901.00596*, 2019.

[48] Fei Xue, Qiuyuan Wang, Xin Wang, Wei Dong, Junqiu Wang, and Hongbin Zha. Guided Feature Selection for Deep Visual Odometry. In *ACCV*, 2018.

[49] Fei Xue, Xin Wang, Shunkai Li, Qiuyuan Wang, Junqiu Wang, and Hongbin Zha. Beyond Tracking: Selecting Memory and Refining Poses for Deep Visual Odometry. In *CVPR*, 2019.

[50] Fei Xue, Xin Wang, Zike Yan, Qiuyuan Wang, Junqiu Wang, and Hongbin Zha. Local Supports Global: Deep Camera Relocalization with Sequence Enhancement. In *ICCV*, 2019.

[51] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An End-to-end Deep Learning Architecture for Graph Classification. In *AAAI*, 2018.

[52] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph Neural Networks: A Review of Methods and Applications. *arXiv preprint arXiv:1812.08434*, 2018.

[53] Yang Zhou, Zachary While, and Evangelos Kalogerakis. SceneGraphNet: Neural Message Passing for 3D Indoor Scene Augmentation. In *ICCV*, 2019.