Knowledge Transfer from Map to DNN: Use of Graph Convolutional Neural Network for Augmenting Visual Robot Self-localization System

Takeda Koji Tanaka Kanji

***Abstract*— Graph-based scene model has been receiving increasing attention as a flexible and descriptive scene model for visual robot self-localization. In a typical self-localization application, objects, object features, and object relationship in an environment map are described respectively by nodes, node features, and edges in a scene graph, which are then matched against a query scene graph by a graph matching engine. However, its overhead for computation, storage, and communication, is proportional to the number and feature dimensionality of graph nodes, and can be significant in large-scale applications. In this study, we observe that graph-convolutional neural network (GCN) has a potential to become an efficient tool to train and predict with a graph matching engine. However, it is non-trivial to translate a given visual feature to a proper graph feature that contributes to good self-localization performance. To address this issue, we introduce a new knowledge transfer (KT) framework, which introduces an arbitrary self-localization model as a teacher to train the student, GCN-based self-localization system. Our KT framework enables lightweight storage/communication by using compact teacher's output signals as training data. Results on RobotCar datasets show that the proposed method outperforms existing comparing methods as well as the teacher self-localization system.**

I. Introduction

Graph-based scene model has been receiving increasing attention as a flexible and descriptive scene model for visual robot self-localization. In a typical self-localization application, objects, object features, and object relatonship in an environment map are described respectively by nodes, node features, and edges in a scene graph, which are then matched against a query scene graph by a graph matching engine. Such a scene graph model is sufficiently general and applicable to various types of scene data. For example, in [1], an input scene is segmented into semantic segments which serve as graph nodes and are connected with their neighbors via graph edges. An alternative representative example is to model a view-sequence as a scene graph whose nodes are image frames and edges connect successive image frames [2]. In our experiments, we also focus on such a view-sequence -based scene graph representation (Fig. 1).

This paper is concerned with the scalability of a graph-based representation to large-scale applications, such as long-term map-learning [3]. Firstly, the storage cost for a scene graph is proportional to the number and dimensionality of graph nodes (i.e., #graphs, #nodes per graph, dimensionality of node feature), and grows rapidly with the environment size. Moreover, the computational cost for the graph matching engine is proportional to the graph size and often requires approximations such as dimension-reduction to reach reasonable computation speed. Therefore, a new framework is desired for improving efficiency without sacrificing accuracy of a scene graph -based self-localization system.

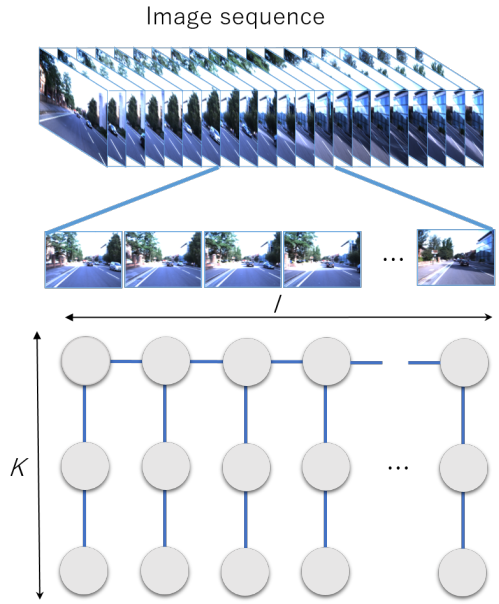

Fig. 1. In this study, a general framework of GCN-based self-localization is proposed and verified in a specific application of view-sequence -based scene graphs. The bottom panel illustrates nodes and time/attribute edges of a scene graph by circles and horizontal/vertical line-segments.

In this study, we observe that graph-convolutional neural network (GCN) [4] has a potential to become an efficient tool to train and predict with a graph matching engine. GCN is recently developed, one of most widely used graph neural networks. In GCN, a graph-convolutional operation is introduced to produce graph features, which are then passed to a graph-summarizing operation to produce higher-order graph features. GCN has been successfully applied to various kinds of graph data applications, including chemical reactivity [5] and web-scale recommender systems [6]. GCN training and prediction process is computationally efficient, whose complexity is in the order of $O(m+n)$ where m and n are #edges and #nodes.

From the perspective of visual robot self-localization, a non-trivial issue is how a robot can translate a given visual feature to a proper graph feature that contributes to good self-localization performance. This is a non-trivial issue because

Our work has been supported in part by JSPS KAKENHI Grant-in-Aid for Scientific Research (C) 17K00361, and (C) 20K12008.

The authors are with Graduate School of Engineering, University of Fukui, Japan. {takedakoji00, tanakakanji}@gmail.com

typical visuial features are originally designed for visual self-localization task and can often be deteriorated when they are directly used as graph features. We tackle this issue by introducing a novel knowledge transfer (KT) framework, which introduces an arbitrary self-localization model as a teacher to train the student, GCN-based self-localization system. Our KT strategy is inspired by the standard KT framework of knowledge distillation [7]. Our feature learning strategy is derived from the field of multi-media information retrieval (MMIR) [8].

Our contributions in this paper are summarized as follows. (1) We first study the use of GCN for augmenting self-localization performance, while suppressing computation, storage and communication cost. (2) We formulate the versatile framework for feature learning by introducing a novel teacher-to-student knowledge transfer model. (3) Results on RobotCar datasets show that the proposed method clearly outperforms the existing comparison methods as well as the teacher self-localization system.

II. Related Work

Visual robot self-localization is one of most important issues in mobile robotics and has been studied in many different contexts, including multi-hypothesis pose tracking [9], map matching [10], image retrieval [11], view-sequence matching [3], etc. Our self-localization scenario is most closely related to the view-sequence matching scenario, which takes a short-term live view-sequence as query and searches for corresponding part in the map view-sequence.

Unlike many existing works, the proposed approach formulates self-localization as a classification problem, which (1) Partitions the robot workspace into place classes, (2) Trains a visual place classifier from a class-specific training set, and (3) Predicts the place class for a given query image with the pre-trained classifier. In this line of researches, it is straightforward to train a deep convolutional neural network (DCN), as a visual place classifier, as demonstrated in our previous study [12]. More recently, in [13], DCN has been successfully used for visual place classification in an alternative context of 3D point cloud -based self-localization with the scan-context image representation. However, the current study is different from these existing studies in the following two aspects. (1) We focus on the problem of graph-based view-sequence representation, which can deal with interactions between image frames. (2) We further address knowledge transfer from a teacher self-localization model to the student GCN-based self-localization system.

Graph neural network (GNN) has attracted recent interest in pattern recognition community as a flexible and efficient model for pattern recognition and machine learning. GCN is a most widely used GNN, which generalizes the traditional convolution to data of graph structures. In previous studies, GCN has been successful in applications where traditional DCN are very inefficient or not applicable [5], [6], [14]. On the other hand, in this study, we revisit the traditional important application of visual robot self-localization and aim to augment and outperform existing solutions.

III. Approach

A. System Overview

We formulate self-localization as a classification problem [12], which consists of three distinct stages: (1) Place partitioning that aims to partition the robot workspace into a collection of place classes; (2) Mapping (i.e., training) that takes a visual experience with ground-truth viewpoint information [12] collected in the workspace as training data and trains a visual place classifier; (3) Self-localization (i.e., testing) that takes a query graph representing a short-term live view-sequence with length T, and predicts the place class.

We suppose that the original training data is no longer available at the test stage, but only the trained classifier is available, to save the overall long-term storage cost.

We do not rely on any post-verification stage, such as RANSAC post-verification [15], in order to reduce the overall computational burden. Nevertheless, experimental results will show that the proposed framework is surprisingly robust against outliers in measurements.

We use a standard grid-based place partitioning method to define place classes [12]. First, a 2D regular grid is imposed on the robot workspace (i.e., moving plane). Then, each grid cell is viewed as a place class. It should be noted that the place partitioning could be improved by introducing state-of-the-art place partitioning techniques, as we also demonstrated in our recent study [16].

B. Graph Matching Engine

Figure 2 depicts the pipeline of the graph matching engine. It consists of three modules:

1) A scene graph descriptor that translates an input length T view-sequence to a scene graph.
2) A knowledge transfer module that transfers knowledge from a teacher self-localization model to the student GCN-based self-localization system.
3) A supervised learning module that takes length T view-sequences as training samples, and trains a classifier that predicts the place class given a query sample.

These three modules are detailed in the subsequent subsections III-C, III-D, and III-E.

We follow the supervised learning procedure to train the scene graph classifier. In the mapping stage, a collection of overlapping sub-sequences with the same length T is sampled from the visual experience, and divided into place-class-specific training sets according to the available viewpoint information as well as the pre-defined place partitioning.

We emphasize that all the training set can be thrown away once after training the GCN classifier. Since our framework uses overlapping sub-sequences as training data, the total data size as well as the number of graph nodes can be significantly larger than the original training view-sequence. Nevertheless, the training data has no impact on the storage overhead after compressing the training data into a GCN classifier.

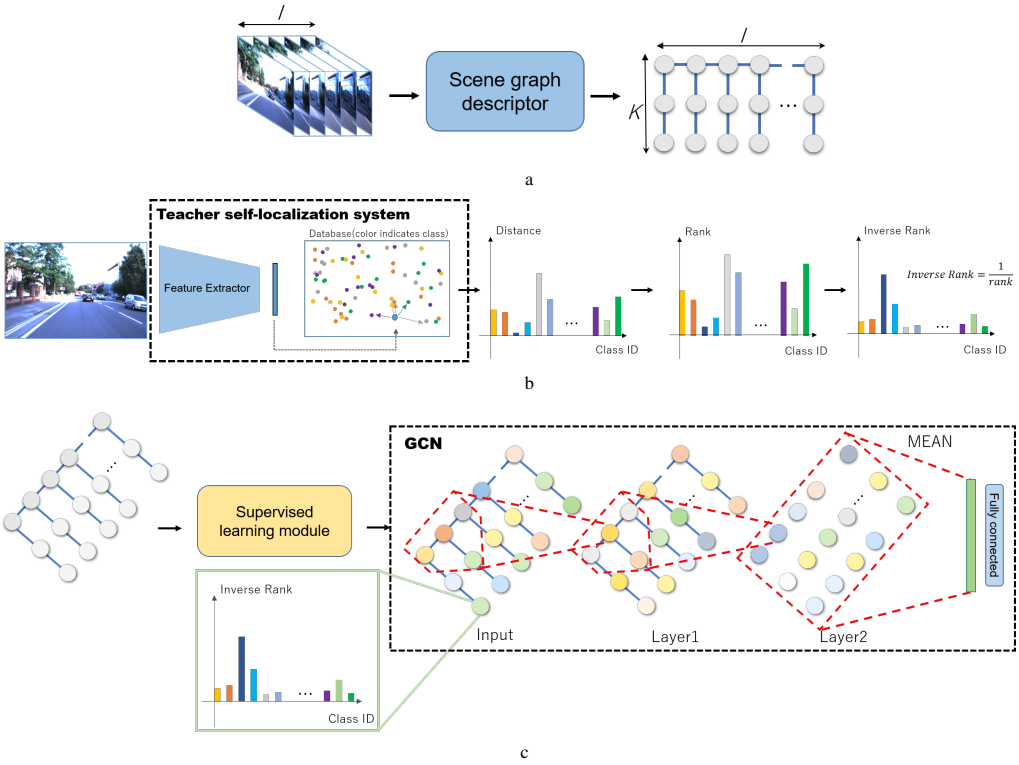

Fig. 2. Pipeline of graph matching engine. (a) Scene graph descriptor. (b) Knowledge transfer from teacher self-localization model. (c) Supervised learning of GCN-based self-localization network.

C. Scene Graph Descriptor

We build the domain invariance by controlling the length and intervals of map/query (i.e., training/testing) view-sequences (Fig. 3). Firstly, the same length T is assumed for all training/testing view-sequences, to build invariance across different domains. Moreover, these T frames are selected so that travel distances between successive frames are approximately equal to a pre-set value, 2[m], to build invariance against the vehicle's ego-motion speed. We will experimentally show that the above strategy contributes to good self-localization performance. It should be noted that the GCN theory is not restricted to such homogeneous graphs (with the same size and shape), and extending this approach to deal with heterogeneous graphs is a future direction of research.

We build a collection of K different image feature extractors $F_1, \cdots, F_K$, by combining several image processing techniques, such as NetVLAD [17], Canny operation [18], depth regression [19], and semantic segmentation [20], as shown in Section III-D. Then, each graph node $n = (t, k, f_k[t])$ represents an attribute feature vector $f_k[t]$ extracted by k-th extractor from t-th image frame. On the other hand, two types of graph edges, time edge and attribute edge, are employed in our approach (Fig. 3). A time edge $e = (t, t+1, k)$ connects two graph nodes with successive time indexes $t, t+1$ with the same attribute index k. An attribute edge $e = (t, k_1, k_2)$ connects two graph nodes with different attribute indexes k_1, k_2 with the same time index t.

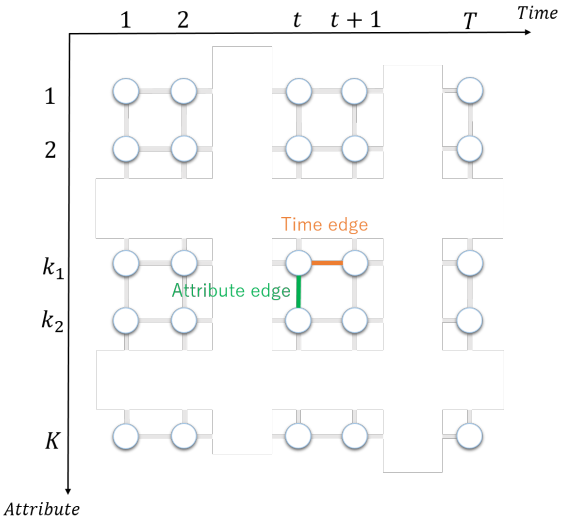

Fig. 3. Time and attribute edges.

D. Knowledge Transfer

We now discuss how a robot can translate input view-images to graph features that are required by GCN training/testing.

A naive and intuitive way is to use visual features that are originally designed for visual self-localization tasks, directly as graph node features. Designing visual features has been a dominant topic in recent self-localization literature [17], [21], [22]. Many works have proposed compact yet discriminative visual features. Recent examples include autoencoder-based

method [21], GAN-based method [22], CNN-based method [17]. In particular, NetVLAD [17] is recently developed and one of most widely used visual features in computer vision and robotics, which is used also in our experiments as a comparing method.

A concern is that typical visual features are not optimized for graph convolutions. In theory, their good performance in the original applications does not guarantee their good performance in GCN-based self-localization. In fact, results in our experiments will show that the self-localization performance is deteriorated when they are directly used as graph node features in a GCN-based self-localization.

To address this, we propose to use class-specific probability distribution vector (PDV) output by the teacher self-localization model as the training data. This strategy is similar in concept with the standard KT approach of knowledge distillation [7]. It should be noted that the PDV representation ensures the versatile applicability to a broad range of teacher output signals, including tf-idf score in bag-of-words image retrieval [23], RANSAC scores in a post-verification stage [24], as well as mean average intersection-over-union in an object matching system [25].

We convert a node image I to a graph feature vector

$$f = M(Y(I)) \tag{1}$$

by using a teacher self-localization system Y and an image-to-feature translator M. The conversion procedure is as follows: (1) Input the node image I to a teacher self-localization system Y, and obtain the output PDV signal $o = Y(I)$ from the teacher system. (2) Map the output PDV o to a graph feature vector $f = M(o)$. This strategy is similar in concept with our previous study in [26], where outputs from a teacher self-localization system are used as visual features for a student self-localization system. The above two functions Y, M are detailed in the following.

We build four teacher systems Y_1, Y_2, Y_3, and Y_4 (Fig. 4), by combining four different image filters $Z_i(I)$ $(i \in [1,4])$ with a single nearest-neighbor (NN) [27] -based VLAD matching engine Y_o, in the form:

$$Y_i = Y_o(Z_i(I)). \tag{2}$$

An NN matching engine represents each place class by a collection of VLAD descriptors, each of which has been extracted from each image in the class-specific training set. Then, it computes the image-to-class distance (i.e., dissimilarity) as the distance from a given query VLAD vector to the nearest-neighbor in the class-specific VLAD vectors. The image filters are implemented as follows. Z_1 is a simple identity mapping function (Fig. 4 "raw image"). Z_2 is Canny image filter that converts an input image to a gradient-emphasized image (Fig. 4 "canny"). Z_3 is depth image regressor that is trained in an unsupervised manner and aims to predict a depth image from a monocular image (Fig. 4 "depth"). Z_4 is semantic segmentation filter [20] that converts an input image to a semantic label image, whose pixel color represents the pixel-wise class label that is defined in the original color palette in [20] (Fig. 4 "semantic").

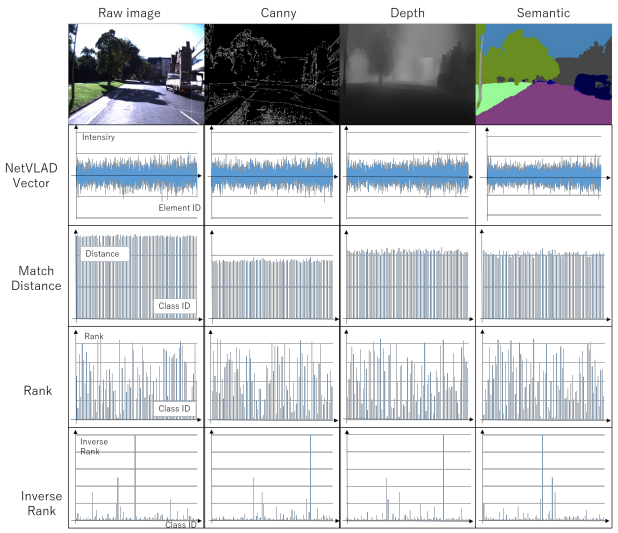

Fig. 4. Images and feature vectors generated by individual image filters and image-to-feature translators.

We consider and investigate the following four types of mapping functions $M_1(\cdot)$, $M_2(\cdot)$, $M_3(\cdot)$, and $M_4(\cdot)$. M_1 is an identity mapping that can be used only when $Z = Z_1$ (Fig. 4 "NetVLAD vector"). M_2 is a class-specific distance value vector, in which each c-th place class for the vector is assigned with the L2 norm from the query feature to its nearest neighbor feature in the class (Fig. 4 "match distance"). M_3 uses a ranking function, in which each c-th place class for a given PDV is assigned with a rank value by sorting the elements in a PDV in ascending order of the probability scores, and then the vector of rank values is used as the node feature vector (Fig. 4 "rank"). This ranking based representation is inspired by the recent findings in the field of MMIR [8], in which rank values are used for information fusion across different domains (e.g., individual users' domains of MMIR). M_4 is different from M_3 only in that inverse-rank values are used in place of the rank values (Fig. 4 "inverse rank"). This strategy can be viewed as an application of inverse rank fusion methods [28] to our graph feature fusion task.

E. Self-localization

The training stage of the self-localization system basically follows the standard training procedure for GCN in [4].

A Graph is represented as $G = (V, E)$ where V is the set of nodes, and E is the set of edges. Let $v_i \in V$ denote a node and $e_{ij} = (v_i, v_j) \in E$ denote edge pointing from v_j to v_i. We define all graphs as undirected graph, i.e., a special case of directed graphs where there is a pair of edges with inverse directions if two nodes are connected. The neighborhood of a node v is defined as $N(v) = \{u \in V \mid (u,v) \in E\}$. Each node v has a feature vector $\mathbf{h} \in R^D$. The representation of a node v is generated by aggregating its own features $\mathbf{h}_v$ and features $\mathbf{h}_u$ $(u \in N(v))$ of neighboring nodes $N(v)$ that are connected to v via edges, in the following steps. First, each node receives features from the neighbors

$N(v)$, which are then summarized via SUM operation. Then, these summarized features are passed to single layer fully connected neural network, which is followed by non-linear ReLU transformation, in the form:

$$\mathbf{h}_i^{new} = \mathrm{ReLU}(\mathbf{W}(\sum_{u \in N(v_i) \cup v_i} \mathbf{h}_u)), \quad (3)$$

where W is weight matrix $\mathbf{W} \in \mathbf{R}^{D \times D'}$ to apply linear transformation. D and D' respectively are the dimensions of feature vector before and after applying this operation. The operation at l-th GCN layer is generalized in the form:

$$\mathbf{h}_i^{(l)} = \mathrm{ReLU}(\mathbf{W}^{(l-1)}(\sum_{u \in N(v_i) \cup v_i} \mathbf{h}_u^{(l-1)})) \quad (4)$$

This operation is applied to all nodes. In this way, all node features are updated. This operation is repeated L times, where L is the number of layers and set $L = 2$ in this study. Finally, features of all nodes are summarized by an averaging operation, and then passed to an fully-connected (FC) softmax operation:

$$\mathbf{p} = \mathrm{Softmax}(\frac{1}{|V|} \sum_{u \in V} \mathbf{h}_u^K). \quad (5)$$

K is the number of final GCN layers, and $\mathbf{h}_u$ is the feature at the node u output by the final GCN layer. Our implementation is based on the Deep Graph Library with Pytorch backend in [4].

IV. Evaluation

We evaluated the proposed algorithm on RobotCar dataset [29]. Table I shows characteristics of the dataset.

For the grid-based place partitioning in III-A, we used a 14×17 grid with the resolution of 0.1 degree in latitude and longitude (approximately 110[m]×70[m] resolution). As a result, the number of training images per place class becomes 80-90 on average. A place class is eliminated from the training/testing set if the number of images belonging to the class equals to or lower than 6. Every image is cropped by a 1080×800 region to eliminate those regions that are occluded by the vehicle itself (i.e., 100 pixels from each size, and 180 pixels from the bottom). The length of a map/query view-sequence is set T =10. The intervals between successive frames in travel distance are set approximately 2[m]. Those sequences that span adjacent places are removed from training/testing set.

Self-localization performance in terms of top-1 accuracy is shown in Tab. II. For comparing method, we use NN matching of NetVLAD descriptor [17], adapting the implementation in [30]. For the dissimilarity measure, the image-to-class distance defined in III-D is reused.

The number of GCN layers is 2. The feature dimensionality of GCN layers are set as C, 256, 256 and C for a size C class set. For the node summarization, SUM and ReLU operations are used. The number of epochs, batch size, and learning rate are set as 5, 32, and 0.001. The GCN operation works in CPU using Intel(R) Xeon(R) GOLC 6130 CPU @ 2.10GHz. The self-localization performance is measured by top-1 accuracy. The timing result for training was 170 [sec] for 5 epochs and 31,835 training samples. The time cost for prediction is 15.5 [msec] per query graph. It can be said that the proposed method is computationally very efficient and realtime capable.

Table IV shows results for the proposed method with different choices of the image filter Z_i as well as the comparing method. From top to bottom, each line corresponds to the image filters Z_1, Z_2, Z_3, and Z_4. By comparing the different combinations of image filters, the combination of Z_1 and Z_4 yielded the best performance. It can be seen that the proposed method outperforms the comparing method for almost all settings considered here.

TABLE I

Dataset characteristics

dataset ID	weather	#images	detour	roadworks
15-08-28-09-50-22 (A)	sun	31,855	×	×
15-10-30-13-52-14 (B)	overcast	48,196	×	×
15-11-10-10-32-52 (C)	overcast	29,350	×	○
15-11-12-13-27-51 (D)	clouds	41,472	○	○
15-11-13-10-28-08 (E)	overcast	42,968	×	×

TABLE II

Top-1 accuracy

	A	B	C	D	E
A		92.3	88.5	80.8	88.4
B	92.4		97.3	87.3	97.6
C	91.5	94.8		90.9	95.7
D	88.2	88.0	93.2		91.6
E	92.7	97.4	99.2	94.4	

TABLE III

Performance comparison

Method	Average Top-1 accuracy
Ours	**92.1**
NetVLAD	87.6
Ours w/o edge	86.7
Ours w/o attribute edge	91.6
Ours w/o time edge	91.3
Ours w/o attribute node/edge	86.9

TABLE IV

Results for different choices of image filters.

	A	B	C	D	E
A		92.3	88.5	80.8	88.4
		92.8	90.0	82.0	90.2
		93.2	89.9	83.3	91.5
		83.1	85.0	79.7	82.2
B	92.4		97.3	86.6	97.8
	92.3		98.5	85.0	97.6
	91.1		96.7	84.2	97.6
	86.9		95.2	81.6	95.5
C	91.5	94.8		90.9	95.7
	91.2	94.9		89.8	95.8
	91.7	94.7		90.7	95.8
	83.6	91.7		88.7	94.0
D	88.2	88.0	93.2		91.6
	87.6	87.7	87.6		91.9
	87.2	87.3	93.2		92.6
	78.0	87.1	92.4		88.4
E	92.7	97.4	99.2	94.4	
	93.3	97.4	99.0	93.1	
	94.5	96.9	99.2	94.1	
	83.9	93.3	97.3	91.1	

As an ablation study, we performed two experiments. In the first case, we modified the scene graphs by removing all the edges, then train and test the GCN. In the second case, we modified the graph topology by removing one of the two types of edges, either time or attribute edge. Table III shows results for the ablation study. It is apparent that graph with both time and attribute edges work significantly better for almost all cases. Specifically, the use of time edges often contributed to improve the robustness against partial occlusions against illumination changes between training and test domains. In consequence, the proposed GCN framework was successfully shown to solve the variety of problems by integrating the available cues from different image filters, as well as time and spatial graphs.

V. Conclusions

In this paper, we first explore the use of GCN for augmenting visual robot self-localization systems, and improve the self-localization performance without sacrificing efficiency in computation, storage and communication. A novel versatile KT framework is introduced for transferring knowledge from a teacher self-localization model to integrate the available cues from different image filters as well as time and spatial contextual information. Results on RobotCar datasets show that the proposed method clearly outperforms the existing comparing methods as well as the teacher self-localization system. In the scope of this paper, we worked with view-sequence -based scene graph representaion, but other scene graph representations may be used such as attribute grammar -based scene graphs [31]. Another direction is to consider more general heterogeneous scene graphs to deal with map/query scene graphs with variable sizes and shapes.

References

[1] A. Gawel, C. Del Don, R. Siegwart, J. Nieto, and C. Cadena, "X-view: Graph-based semantic multi-view localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1687–1694, 2018.

[2] T. Naseer, L. Spinello, W. Burgard, and C. Stachniss, "Robust visual robot localization across seasons using network flows." in *AAAI*, 2014, pp. 2564–2570.

[3] M. J. Milford and G. F. Wyeth, "Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights," in *2012 IEEE Int. Conf. Robotics and Automation*. IEEE, 2012, pp. 1643–1649.

[4] M. Wang, L. Yu, D. Zheng, Q. Gan, Y. Gai, Z. Ye, M. Li, J. Zhou, Q. Huang, C. Ma, Z. Huang, Q. Guo, H. Zhang, H. Lin, J. Zhao, J. Li, A. J. Smola, and Z. Zhang, "Deep graph library: Towards efficient and scalable deep learning on graphs," *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[5] C. W. Coley, W. Jin, L. Rogers, T. F. Jamison, T. S. Jaakkola, W. H. Green, R. Barzilay, and K. F. Jensen, "A graph-convolutional neural network model for the prediction of chemical reactivity," *Chemical science*, vol. 10, no. 2, pp. 370–377, 2019.

[6] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining*, 2018, pp. 974–983.

[7] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[8] M. Imhof and M. Braschler, "A study of untrained models for multimodal information retrieval," *Information Retrieval Journal*, vol. 21, no. 1, pp. 81–106, 2018.

[9] M. Himstedt and E. Maehle, "Semantic monte-carlo localization in changing environments using rgb-d cameras," in *2017 European Conference on Mobile Robots (ECMR)*. IEEE, 2017, pp. 1–8.

[10] J. Neira, J. D. Tardós, and J. A. Castellanos, "Linear time vehicle relocation in slam," in *ICRA*. Citeseer, 2003, pp. 427–433.

[11] M. Cummins and P. Newman, "Fab-map: Probabilistic localization and mapping in the space of appearance," *Int. J. Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[12] N. Yang, K. Tanaka, Y. Fang, X. Fei, K. Inagami, and Y. Ishikawa, "Long-term vehicle localization using compressed visual experiences," pp. 2203–2208, 2018.

[13] G. Kim, B. Park, and A. Kim, "1-day learning, 1-year localization: Long-term lidar localization using scan context image," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1948–1955, 2019.

[14] L. Zhang and Z. Zhu, "Unsupervised feature learning for point cloud understanding by contrasting and clustering using graph convolutional neural networks," in *IEEE Int. Conf. 3D Vision*, 2019, pp. 395–404.

[15] R. Raguram, O. Chum, M. Pollefeys, J. Matas, and J.-M. Frahm, "Usac: a universal framework for random sample consensus," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 2022–2038, 2012.

[16] K. Tanaka, "Self-supervised map-segmentation by mining minimal-map-segments," in *IEEE Intelligent Vehicles Symposium (IV)*, 2020.

[17] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[18] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[19] I. Alhashim and P. Wonka, "High quality monocular depth estimation via transfer learning," *CoRR*, vol. abs/1812.11941, 2018.

[20] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII*, ser. Lecture Notes in Computer Science, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., vol. 11211. Springer, 2018, pp. 833–851.

[21] N. Merrill and G. Huang, "CALC2.0: Combining appearance, semantic and geometric information for robust and efficient visual loop closure," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Macau, China, Nov. 2019.

[22] H. Hu, H. Wang, Z. Liu, C. Yang, W. Chen, and L. Xie, "Retrieval-based localization based on domain-invariant feature learning under changing environments," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2019, pp. 3684–3689.

[23] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *null*, 2003, p. 1470.

[24] E. Garcia-Fidalgo and A. Ortiz, "ibow-lcd: An appearance-based loop-closure detection approach using incremental bags of binary words," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3051–3057, 2018.

[25] N. Sünderhauf, S. Shirazi, F. Dayoub, B. Upcroft, and M. Milford, "On the performance of convnet features for place recognition," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2015, pp. 4297–4304.

[26] K. Tanaka, Y. Chokushi, and M. Ando, "Mining visual phrases for long-term visual SLAM," in *2014 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2014, pp. 136–142.

[27] G. H. Chen, D. Shah, *et al.*, *Explaining the success of nearest neighbor methods in prediction*. Now Publishers, 2018.

[28] D. F. Hsu and I. Taksa, "Comparing rank and score combination methods for data fusion in information retrieval," *Information retrieval*, vol. 8, no. 3, pp. 449–480, 2005.

[29] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset," *The International Journal of Robotics Research (IJRR)*, vol. 36, no. 1, pp. 3–15, 2017.

[30] T. Cieslewski, S. Choudhary, and D. Scaramuzza, "Data-efficient decentralized visual SLAM," in *2018 IEEE International Conference on Robotics and Automation, ICRA*, 2018, pp. 2466–2473.

[31] H. Steinlechner, G. Haaser, S. Maierhofer, and R. F. Tobler, "Attribute grammars for incremental scene graph rendering." in *VISIGRAPP (1: GRAPP)*, 2019, pp. 77–88.