Boosting Self-localization with Graph Convolutional Neural Networks

Takeda Koji and Tanaka Kanji
Department of Engineering, University of Fukui, 3-9-1, Bunkyo, Fukui, Japan

Keywords: Visual Robot Self-localization, Graph Convolutional Neural Network, Map to DNN.

Abstract: Scene graph representation has recently merited attention for being flexible and descriptive where visual robot self-localization is concerned. In a typical self-localization application, the objects, object features and object relationships of the environment map are projected as nodes, node features and edges, respectively, on to the scene graph and subsequently mapped to a query scene graph using a graph matching engine. However, the computational, storage, and communication overhead costs of such a system are directly proportional to the number of feature dimensionalities of the graph nodes, often significant in large-scale applications. In this study, we demonstrate the feasibility of a graph convolutional neural network (GCN) to train and predict alongside a graph matching engine. However, visual features do not often translate well into graph features in modern graph convolution models, thereby affecting their performance. Therefore, we developed a novel knowledge transfer framework that introduces an arbitrary self-localization model as the teacher to train the GCN-based self-localization system i.e., the student. The framework, additionally, facilitated lightweight storage and communication by formulating the compact output signals from the teacher model as training data. Results on the Oxford RobotCar datasets reveal that the proposed method outperforms existing comparative methods and teacher self-localization systems.

1 INTRODUCTION

The graph-based scene model has recently received significant attention as being a flexible and descriptive scene model for visual robot self-localization. In self-localization applications, the objects, object features, and object relationships of the environment map are generally transposed as nodes, node features, and edges, respectively, in the scene graph, which are then matched against a query scene graph by a graph matching engine; such a scene graph model can be used with various types of scene data. In (Gawel et al., 2018), the input scene is segmented semantically to procure the graph nodes, which are linked to their neighbours via graph edges. Conversely, a view sequence-based localization can be modelled as a scene graph wherein nodes become the image frames and the edges connect successive image frames (Naseer et al., 2014). For this study, the view sequence-based scene graph representation, as shown in Fig. 1, is utilised.

Here, we attempt to analyse the scalability of a graph-based representation for large-scale applications such as long-term map-learning (Milford and Wyeth, 2012). The storage cost of a scene graph is proportional to the number of dimensionalities of the graph nodes, i.e., graphs, nodes per graph and dimensionality of the node features, which escalates with the size of the environment. Moreover, the computational cost of a graph matching engine is reliant on the graph size and often requires approximations, such as dimension-reduction, to achieve considerable computational speed. To address these issues, we propose a novel framework to improve the efficiency of a scene graph-based self-localization system without compromising the accuracy.

In this study, we demonstrate the viability of a graph-convolutional neural network (GCN), a popular graph neural network (GNN), as an efficient tool to train and predict with a graph matching engine (Wang et al., 2019). In GCN, a graph-convolutional layer is initially harnessed to extract graph features, which are then supplied to the graph-summarisation process to enrich the features. GCN has been successfully applied to various types of graphical data applications, including chemical reactivity and web-scale recommender systems (Coley et al., 2019; Ying et al., 2018). The GCN training and prediction process is computa-

Koji, T. and Kanji, T.
Boosting Self-localization with Graph Convolutional Neural Networks.
DOI: 10.5220/0010212908610868
In *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2021) - Volume 5: VISAPP*, pages 861-868
ISBN: 978-989-758-488-6

tionally efficient and the complexity is in the order of $O(m+n)$, where m and n are the edges and nodes respectively.

To assess the performance of a visual robot self-localization system, it is important to determine how intuitively the robot converts a given visual feature to a graph feature. As visual features typically aid in visual self-localization tasks, such direct conversion can adversely affect the quality of the resultant graph features. To this effect, we propose a novel knowledge transfer (KT) framework, which introduces an arbitrary self-localization model as a teacher to train the GCN-based self-localization system as the student. The proposed framework adopts the standard KT framework for knowledge distillation, and our feature learning strategy is inspired by the multimedia information retrieval (MMIR) domain (Hinton et al., 2015; Imhof and Braschler, 2018).

The contributions of this study can be summarised as follows: a) to evaluate the benefits of GCN in not only augmenting the self-localization performance but also economising the computational, storage and communication costs; and b) to conceive a versatile framework for feature learning based on a novel teacher-to-student KT model. Results on the Oxford RobotCar datasets highlighted the superior performance of the proposed method when compared to other existing methods and teacher self-localization systems.

2 RELATED WORK

Robot self-localization using vision is one of the most important subdomains of mobile robotics and has been studied in various contexts, including multi-hypothesis pose tracking, map matching, image retrieval and view sequence matching (Himstedt and Maehle, 2017; Neira et al., 2003; Cummins and Newman, 2008; Milford and Wyeth, 2012). Our study borrows from view sequence matching, wherein a real-time short-term view sequence is supplied as a query to obtain the corresponding component on the map view sequence.

Unlike previous studies, the proposed approach models self-localization as a classification problem. The problem consists of a) partitioning the robot workspace into different place classes; b) training a visual place classifier using a class-specific training set; c) predicting the place class for a given query image using the pre-trained classifier. For mobile robotics, training a deep convolutional neural network (DCN) as a visual place classifier is relatively straightforward. Recently, in (Kim et al., 2019), it

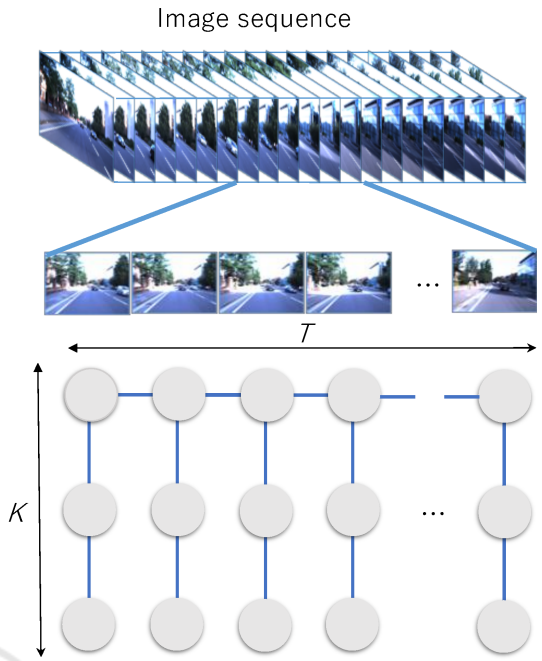

Figure 1: Overview of GCN-based self-localization framework used in conjunction with view-sequence-based scene graphs; the bottom panel illustrates nodes (circles) and time/attribute (horizontal/vertical line-segments) edges of a scene graph.

is successfully implemented for a 3-D point cloud-based self-localization using scan context image representation. However, the current study differs in two aspects viz. it focuses on the graph-based view sequence representation that can accommodate interactions between image frames, and it further addresses KT from a teacher self-localization model to a student GCN-based self-localization system.

GNNs have merited interest among the pattern recognition community as being flexible and efficient for pattern recognition and machine learning, and GCN is the most widely used GNN that generalizes the traditional convolution to data of graph structures. In the past, GCN has been successfully harnessed in applications where the traditional DCN proved to be either inefficient or unsuitable (Coley et al., 2019; Ying et al., 2018; Zhang and Zhu, 2019). However, in this study, we revisit a conventional visual robot self-localization application with the aim to improve existing solutions.

3 VISUAL SELF-LOCALIZATION PROBLEM

Here, the self-localization process is modelled as a classification problem constituting three distinct

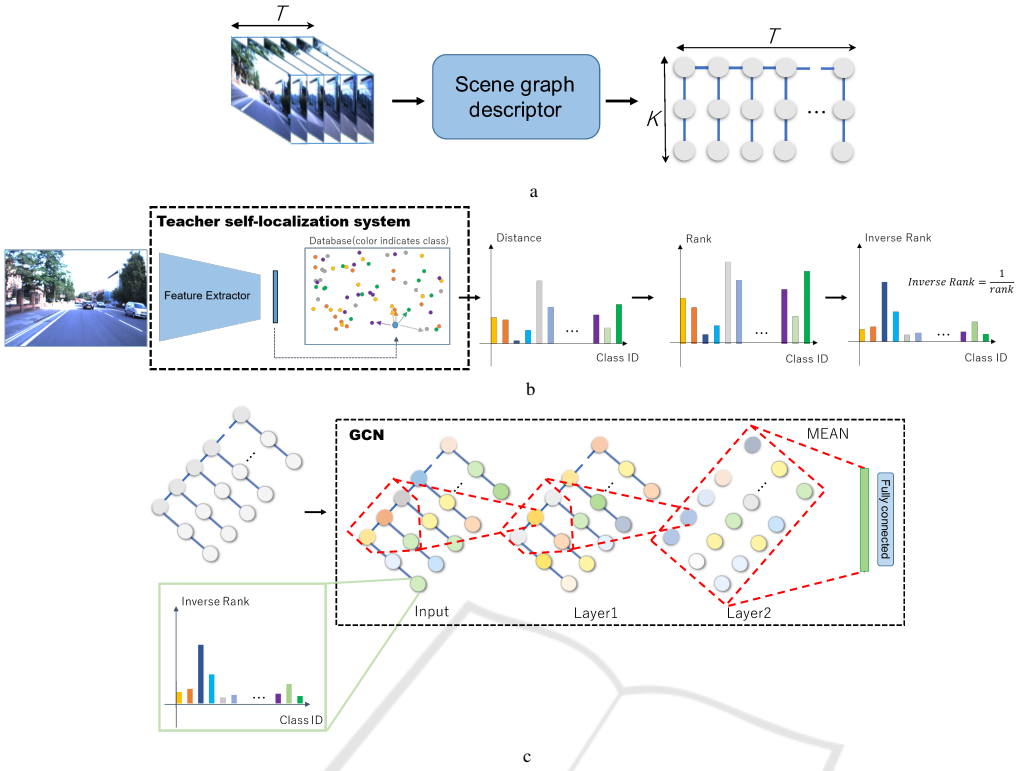

Figure 2: Pipeline of graph matching engine. (a) Scene graph descriptor; (b) KT from teacher self-localization model; and (c) Supervised learning of the GCN model.

stages: (1) Place partitioning that partitions the robot workspace into a collection of place classes; (2) Mapping (i.e., training) that takes a visual experience with ground-truth viewpoint information 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.

To reduce storage costs, the trained visual place classifier is utilised instead of the original training data during testing. Additionally, the post-verification techniques like random sample consensus (RANSAC) are omitted to alleviate the overall computational burden (Raguram et al., 2012). Experimental results nevertheless revealed the robustness of the proposed framework toward outliers in measurements.

A standard grid-based place partitioning method is employed to define the place classes. First, a regular 2-D grid is imposed on the robot workspace i.e., a moving plane, and each grid cell is subsequently viewed as a place class. It should be noted that place partitioning can be enhanced by adopting pertinent state-of-the-art techniques.

4 GCN-BASED SELF-LOCALIZATION

Figure 2 depicts the pipeline of the graph-matching engine. It consists of three modules, which are detailed in the following.

1. A scene graph descriptor to translate an input view sequence of length T to a scene graph.
2. A KT module to facilitate communication between the teacher (arbitrary self-localization model) and student (GCN-based self-localization system).
3. A supervised learning module that utilises the view sequences to train the classifier, which is used to predict the place class for a given query sample.

A supervised learning procedure is applied to train the scene graph classifier. In the mapping stage, a collection of overlapping sub-sequences of length T are sampled from the visual experience and divided into place class-specific training sets according to the available viewpoint information as well as the predefined place partitioning labels.

We emphasize that all the training set can be thrown away once the GCN classifier is trained. Considering the proposed framework employs overlapping sub-sequences as training data, the final dataset size as well as the number of graph nodes are expected to be significantly larger than that estimated originally with the view sequences. Nevertheless, the training data has no impact on the storage overhead after compressing the training data into a GCN classifier.

The domain invariance is elicited by modifying the length and intervals of the map/query (for training/testing respectively) view sequences, as highlighted in Fig. 3.

A uniform length T is initially assumed for all map/query view sequences to develop the invariance across different domains. Moreover, these T frames are selected such that the travel distance between successive frames approximately matches a predetermined value to obtain invariance against the vehicle's ego-motion speed. This setup is also empirically corroborated to highlight the efficiency of the methodology for visual self-localization. It should be noted that the GCN theory is not limited to homogeneous graphs, and extending the proposed approach to tackle heterogeneous graphs is envisaged in future.

First, a collection of K different image feature extractors i.e., $F_1, \cdots, F_K$, are collated using several image processing techniques like NetVLAD, Canny operation, depth regression and semantic segmentation, as shown in section 4 (Arandjelović et al., 2016; Canny, 1986; Alhashim and Wonka, 2018; Chen et al., 2018b). Then, each graph node, $n = (t,k,f_k[t])$, represents an attribute feature vector $f_k[t]$ of the k-th extractor from the tth image frame. Conversely, two types of graph edges viz. time and attribute, are applied such that the time edge, $e = (t,t+1,k)$, connects two graph nodes with successive time indices as $(t,t+1)$ with the attribute index k and the attribute edge, $e = (t,k1,k2)$, connects two graph nodes with different attribute indices as k_1 and k_2 having the same time index t.

We now elucidate how a robot can translate input view images to graph features required for training and, subsequently, validating the model. A straightforward way to achieve this is by directly translating the visual features, originally designed for visual self-localization tasks, to graph node features. Designing visual features has been a topic of interest in recent self-localization literature, with past studies alternatively proposing to apply compact, yet discriminative, visual features like autoencoder-based methods, GAN-based methods, and CNN-based methods (Merrill and Huang, 2019; Hu et al., 2019; Arandjelović et al., 2016). In particular, NetVLAD is an

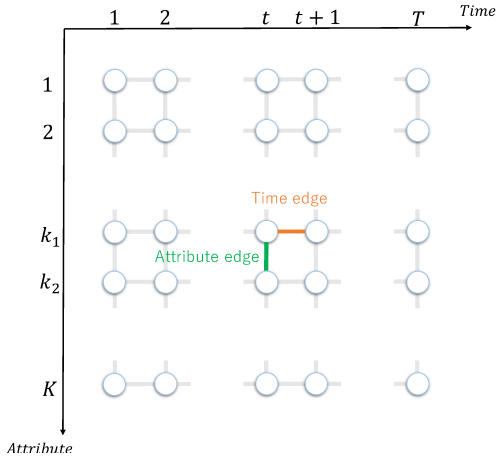

Figure 3: Time and attribute edges.

emerging visual feature extractor in computer vision and robotics, and hence has been used against the proposed methodology for comparison (Arandjelović et al., 2016).

One of the main concerns with incorporating GNNs in this study is that visual features are not optimised for graph convolutions. In theory, their superior performances of the past may not necessarily be replicated in GCN-based self-localization tasks. Results of our experiments, in fact, showed that the self-localization performance deteriorated when visual features were directly used as node features in the GCN model.

To address this issue, we engage a class-specific probability distribution vector (PDV) output along with a teacher self-localization model as the training data, which is derived from the standard KT approach for knowledge distillation (Hinton et al., 2015). The PDV representation facilitates applicability to a broad range of teacher output signals, including the tf-idf scores for the bag-of-words image retrieval models, RANSAC scores in the post-verification stage and mean average intersection-over-union in object matching systems (Sivic and Zisserman, 2003; Garcia-Fidalgo and Ortiz, 2018; Sünderhauf et al., 2015).

A node image, I, is converted to a graph feature vector using a teacher self-localization system, Y, and an image-to-feature translator, M:

$$f = M(Y(I)). \quad (1)$$

The conversion procedure is as follows: a) I is first supplied to Y to obtain the output PDV signal, $o = Y(I)$, from the teacher system; and b) o is then mapped to a graph feature vector as $f = M(o)$.

Four teacher systems, Y_1, Y_2, Y_3 and Y_4, were designed for this study, as shown in Fig. 4, by combining four different image filters, $Z_i(I)$ $(i \in [1,4])$, with

a single nearest-neighbour (NN)-based VLAD matching engine, Y_o, given by:

$$Y_i = Y_o(Z_i(I)). \quad (2)$$

An NN matching engine represents a place class via a collection of VLAD descriptors that are extracted from images in the class-specific training set (Chen et al., 2018a). Then, it computes the image-to-class distance (i.e., dissimilarity) between a given query VLAD vector and its nearest neighbour among the class-specific VLAD vectors.

Four image filters were implemented as depicted along the horizontal labels in Fig. 4. Z_1 is a basic identity mapping function, Z_2 represents a Canny image filter that emphasises the gradient of the input image, Z_3 is a depth image regressor trained as an unsupervised task that predicts a depth image from a monocular image (Alhashim and Wonka, 2018), and Z_4 highlights a semantic segmentation filter that converts an input image to a semantic label image whose pixel colour is derived from the pixel-wise class labels defined in the original colour palette in (Chen et al., 2018b). Furthermore, four different mapping functions were applied to the image filters viz. $M_1(\cdot)$, $M_2(\cdot)$, $M_3(\cdot)$ and $M_4(\cdot)$ as highlighted along the vertical labels in Fig. 4. M_1 is an identity mapper used solely with Z_1. M_2 is a class-specific distance value vector wherein each c-th place class is assigned the L2 norm of the distance between the query feature and its nearest neighbour feature in the same class. M_3 employs a ranking function such that the c-th place class in a given PDV is assigned a rank by sorting the PDV elements in the ascending order of their probability scores and the resultant vector of rank values is used as the node feature vector; such rank-based representation was administered based on the recent success it has found in MMIR applications where ranks are used as features to fuse information across several domains (for example, the domain of individuals using MMIR systems) (Imhof and Braschler, 2018). M_4 is different from M_3 in that the inverse rank values are exploited instead, which was inspired by the rank fusion approach in (Hsu and Taksa, 2005).

We adopt the standard training procedure for GCN, as delineated in (Wang et al., 2019), to train the proposed self-localization system. Initially, a graph is defined as $G = (V, E)$, where V is the set of nodes and E is the set of edges. All graphs are undirected i.e., a special case of directed graphs where a pair of connected nodes are indicated by a pair of edges with inverse directions. Let v_i in V denote a node and $e_{ij} = (v_i, v_j)$ in E denote the edge pointing from v_j to v_i. Then, the neighbourhood of node v can then be defined as $N(v) = \{u \in V \mid (u, v) \in E\}$. Each node has a corresponding feature vector expressed as $\mathbf{h} \in R^D$. The representation of v is generated by aggregating its own features $\mathbf{h}_v$ and those in $N(v)$ connected to v via edges, $\mathbf{h}_u$ ($u \in N(v)$), computed as follows: a) each node initially receives features from $N(v)$; b) the features are thereafter summarised in a summation operation; and c) the summarised features are then supplied to a single layer fully-connected neural network, followed by a non-linear ReLU transformation expressed as:

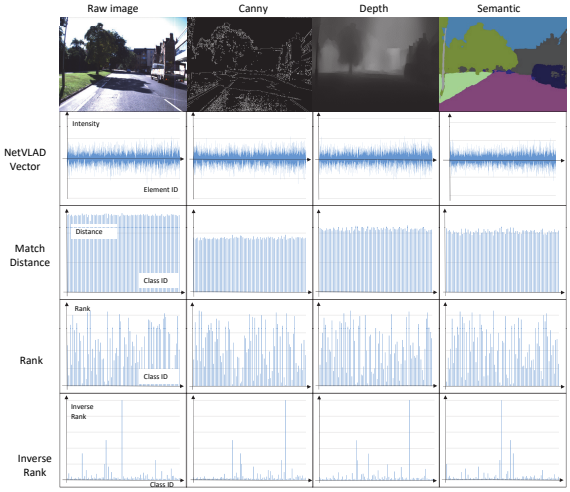

Figure 4: Image and feature vectors generated by individual image filters and image-to-feature translators.

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

where W is weight matrix $\mathbf{W} \in \mathbf{R}^{D \times D'}$ for applying the linear transformation, and D and D' are the dimensions of the feature vector before and after the operation. The operation at the l-th GCN layer is generalised as:

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

This operation is applied to all nodes to update the node features and is repeated L times, corresponding to the number of layers, which was configured as 2 for this study. Finally, the features of all nodes are summarised as an average, and then passed to a fully-connected (FC) and softmax operation given by:

$$\mathbf{p} = \text{Softmax}\left(FC\left(\frac{1}{|V|}\sum_{u \in V} \mathbf{h}_u^K\right)\right), \quad (5)$$

where $\mathbf{h}_u$ is the feature at the node u being the output of the final GCN layer. The system was implemented using the deep graph library with a Pytorch backend, as in (Wang et al., 2019).

Table 1: 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	×	×

5 EXPERIMENTS

We evaluated the proposed methodology on the Oxford RobotCar dataset (Maddern et al., 2017). Table 1 enumerates the characteristics of the dataset. For the grid-based place partitioning process described in 3, we used a 14×17 grid with a resolution of 0.1 degree horizontally and vertically (approximately 110×70 m). Resultantly, the average number of place classes was 81-86 and a place class was eliminated from the training and test sets if the number of images belonging to the class was less than or equal to 6. Every image was cropped to 1080×800 pixels to eliminate regions occluded by the vehicle itself (i.e., 100 pixels from each side and 180 pixels from the bottom). The length of the map and query view sequences was set to $T = 10$ and the intervals between successive frames in the travel distance was approximately 2[m]. Finally, the sequences spanning adjacent places were removed altogether from both training and test sets. For simplicity, we consider scene graphs with two image filters (i.e., one attribute edges per image frame) and as the default setting, the combination of the image filters with Z_1 and Z_4 is used.

To compare the performance accuracy, we used the implementation of an NN matching system with the NetVLAD descriptor (Arandjelović et al., 2016) (adapting the implementation in (Cieslewski et al., 2018)) which uses the first image frame in each view-sequence as the query. Conversely, the image-to-class distance described in 4 was used to measure the class dissimilarity.

Table 2: 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	

The number of GCN layers was set to 2 and feature dimensionality of the GCN layers was configured as C, 256, 256, and C for a size C class set. For node summarisation, the SUM and ReLU operations were administered. The number of epochs, batch size and learning rate were set to 5, 32, and 0.001, respectively. The training was conducted for 170 s on 31,835 samples on a personal computer running the Intel(R) Xeon(R) GOLC 6130 CPU at 2.10 GHz. The self-localization performance was measured by the top-1 accuracy, as highlighted in Table 4. The horizontal and vertical indexes in the table are IDs of query and map datasets, respectively. The prediction turnaround time amounted to 15.5 ms per query graph, rendering the proposed framework as computationally expeditious. This implies significant reduction in computational complexity compared with previous approaches such as graph matching.

Table 4 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, the 1st, 2nd and 3rd lines correspond to the combinations of image filters (Z_1, Z_2), (Z_1, Z_3) and (Z_1, Z_4), while the 4th line corresponds the result with the comparing method. 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 4 enumerates the comparative results of administering different combinations of image filters on the proposed method against existing methods. From top to bottom, each line corresponds to that image filters Z_1, Z_2, Z_3 and Z_4. Among the various combinations applied, that of Z_1 and Z_4 yielded the best outcome. It can be seen that the proposed method surpassed its competitors in most of the settings considered here.

Two experiments were performed as part of an ablation study. In the first instance, the scene graphs were modified by removing the edges to train the model and, in the second, the graph topology was modified by removing one of the time and attribute edges at random. Table Table 3 outlines the experimental results. It is apparent that graphs with both time and attribute edges worked significantly better in almost all cases. The use of attribute edges fa-

Table 3: Performance comparison.

Method	Average Top-1 accuracy
Ours	**92.3**
NetVLAD	87.9
Ours w/o edge	89.0
Ours w/o attribute edge	91.5
Ours w/o time edge	88.7
Ours w/o attribute node/edge	91.7

Table 4: 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	

cilitated resistance against feature ambiguity by compensating individual features' drawbacks. The use of time edges facilitated resistance against partial occlusions incurred from changes in illumination between the training and test domains. Consequently, the proposed framework showed potential to solve a variety of problems by integrating the available cues from different image filters as well as the time and spatial graphs.

6 CONCLUSIONS

We investigated the utility of a GCN model to augment the performance of visual robot self-localization systems whilst alleviating the computational, storage and communication costs. Furthermore, a novel and versatile KT framework was conceived to facilitate information transfer from an arbitrary self-localization model (teacher) that integrated the available cues from different image filters as well as the time and spatial contextual information. Results on the Oxford RobotCar datasets substantiated the robustness of the proposed framework when compared to other existing methods and teacher self-localization systems. Although we harnessed a view sequence-based scene graph representation for this study, other scene graph representations can also be employed, including attribute grammar-based scene graphs (Steinlechner et al., 2019). We attempt to explore other general heterogeneous scene graphs so as to tackle map/query scene graphs of variable sizes and shapes in future.

ACKNOWLEDGEMENTS

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

REFERENCES

Alhashim, I. and Wonka, P. (2018). High quality monocular depth estimation via transfer learning. *CoRR*, abs/1812.11941.

Arandjelović, R., Gronat, P., Torii, A., Pajdla, T., and Sivic, J. (2016). NetVLAD: CNN architecture for weakly supervised place recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698.

Chen, G. H., Shah, D., et al. (2018a). *Explaining the success of nearest neighbor methods in prediction*. Now Publishers.

Chen, L., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018b). Encoder-decoder with atrous separable convolution for semantic image segmentation. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII*, volume 11211 of *Lecture Notes in Computer Science*, pages 833–851. Springer.

Cieslewski, T., Choudhary, S., and Scaramuzza, D. (2018). Data-efficient decentralized visual SLAM. In *2018 IEEE International Conference on Robotics and Automation, ICRA*, pages 2466–2473.

Coley, C. W., Jin, W., Rogers, L., Jamison, T. F., Jaakkola, T. S., Green, W. H., Barzilay, R., and Jensen, K. F. (2019). A graph-convolutional neural network model for the prediction of chemical reactivity. *Chemical science*, 10(2):370–377.

Cummins, M. and Newman, P. (2008). Fab-map: Probabilistic localization and mapping in the space of appearance. *Int. J. Robotics Research*, 27(6):647–665.

Garcia-Fidalgo, E. and Ortiz, A. (2018). ibow-lcd: An appearance-based loop-closure detection approach using incremental bags of binary words. *IEEE Robotics and Automation Letters*, 3(4):3051–3057.

Gawel, A., Del Don, C., Siegwart, R., Nieto, J., and Cadena, C. (2018). X-view: Graph-based semantic multi-view localization. *IEEE Robotics and Automation Letters*, 3(3):1687–1694.

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

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hsu, D. F. and Taksa, I. (2005). Comparing rank and score combination methods for data fusion in information retrieval. *Information retrieval*, 8(3):449–480.

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

Imhof, M. and Braschler, M. (2018). A study of untrained models for multimodal information retrieval. *Information Retrieval Journal*, 21(1):81–106.

Kim, G., Park, B., and Kim, A. (2019). 1-day learning, 1-year localization: Long-term lidar localization using scan context image. *IEEE Robotics and Automation Letters*, 4(2):1948–1955.

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

Merrill, N. and Huang, G. (2019). 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.

Milford, M. J. and Wyeth, G. F. (2012). Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. In *2012 IEEE Int. Conf. Robotics and Automation*, pages 1643–1649. IEEE.

Naseer, T., Spinello, L., Burgard, W., and Stachniss, C. (2014). Robust visual robot localization across seasons using network flows. In *AAAI*, pages 2564–2570.

Neira, J., Tardós, J. D., and Castellanos, J. A. (2003). Linear time vehicle relocation in slam. In *ICRA*, pages 427–433. Citeseer.

Raguram, R., Chum, O., Pollefeys, M., Matas, J., and Frahm, J.-M. (2012). Usac: a universal framework for random sample consensus. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):2022–2038.

Sivic, J. and Zisserman, A. (2003). Video google: A text retrieval approach to object matching in videos. In *null*, page 1470.

Steinlechner, H., Haaser, G., Maierhofer, S., and Tobler, R. F. (2019). Attribute grammars for incremental scene graph rendering. In *VISIGRAPP (1: GRAPP)*, pages 77–88.

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

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

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

Zhang, L. and Zhu, Z. (2019). Unsupervised feature learning for point cloud understanding by contrasting and clustering using graph convolutional neural networks. In *IEEE Int. Conf. 3D Vision*, pages 395–404.