GeoGraph: Graph-Based Multi-view Object Detection with Geometric Cues End-to-End

Ahmed Samy Nassar, Stefano D'aronco, Sébastien Lefèvre, Jan D Wegner

GeoGraph: Graph-based multi-view object detection with geometric cues end-to-end

Ahmed Samy Nassar[1,2], Stefano D'Aronco[2], Sébastien Lefèvre[1], Jan D. Wegner[2]

[1]IRISA, Université Bretagne Sud
[2]EcoVision Lab, Photogrammetry and Remote Sensing group, ETH Zurich

Abstract. In this paper we propose an end-to-end learnable approach that detects static urban objects from multiple views, re-identifies instances, and finally assigns a geographic position per object. Our method relies on a Graph Neural Network (GNN) to, detect all objects and output their geographic positions given images and approximate camera poses as input. Our GNN simultaneously models relative pose and image evidence, and is further able to deal with an arbitrary number of input views. Our method is robust to occlusion, with similar appearance of neighboring objects, and severe changes in viewpoints by jointly reasoning about visual image appearance and relative pose. Experimental evaluation on two challenging, large-scale datasets and comparison with state-of-the-art methods show significant and systematic improvements both in accuracy and efficiency, with 2-6% gain in detection and re-ID average precision as well as 8x reduction of training time.

Keywords: Object Detection · Re-identification · Graph Neural Networks · Urban Objects · Multi-view

1 Introduction

We present an end-to-end trainable multi-view object detection and re-identification approach centered on Graph Neural Networks (GNN). Unlike images, much data is not structured on a grid but naturally follows a graph structure. GNNs apply directly to graphs and thus their applications vary over many disciplines like predicting molecular properties for chemical compounds [15, 29] and proteins [12], social influence prediction [43], object tracking [3, 13], or detection of fake news [35]. Here, we propose to solve the problem of multi-view detection and re-identification of static objects in urban scenes using Graph Neural Networks. Given a set of ground-level images with coarse relative pose information, we detect, re-identify and finally assign geographic coordinates to thousands of urban objects with an end-to-end learnable approach.

Maintaining complete and accurate maps of urban objects is essential for a wide range of applications like autonomous driving, or maintenance of infrastructure by local municipalities. Despite much research in this field [49, 23, 54, 38], updating maps is often still carried out via field surveys, which is a

time-consuming and costly process. Here, we propose to accomplish this task by leveraging publicly available imagery that comes with coarse camera pose information.

What makes this task challenging is the relatively poor image quality of street-level panoramas (e.g., image stitching artefacts, motion blur) or dash cam image sequences (e.g., motion blur, narrow field of view) compared to data acquired through dedicated mobile mapping campaigns. Basically, wide baselines between consecutive acquisitions, and inaccurate camera poses information hinder establishing dense pixel-level correspondences between images. We thus propose to integrate image evidence and pose information into a single, end-to-end trainable neural network that uses images and coarse poses as input and outputs the geo-location of each distinct object in the scene. In contrast to a rigorous structure-from-motion approach, our method learns the joint distribution over different warping functions of the same object instance across multiple views together with the relative pose information. Unlike recent research in this domain, e.g., [49, 26, 2, 23], our method employs an end-to-end approach that helps to jointly learn features to carry out the detection, re-identification, and geo-localization tasks. And differently to other end-to-end works such as [38], our approach for re-identification is based on a GNN that enables the use of multiple views (2+) and is much more computationally efficient in comparison to a siamese approach.

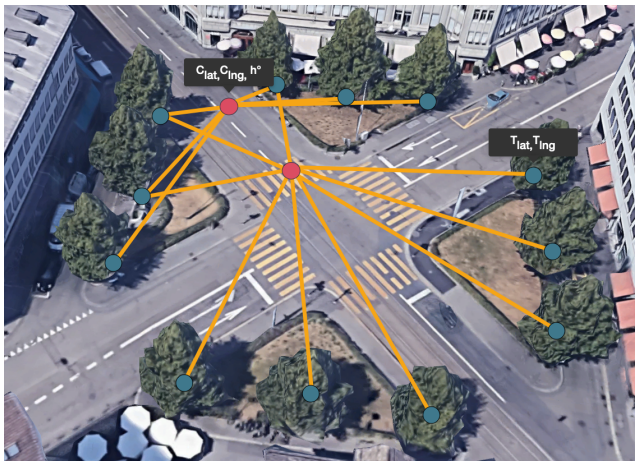

Fig. 1. Illustration of our multi-view scenario. Red circle: Camera acquisition location. Green circle: target object to be detected. Orange line: distance between camera and object.

As illustrated in Fig. 1, our method works as follows: using a set of street-level images that come with coarse camera pose information as input, an object

detector predicts several object bounding boxes for each of the views. We then construct a fully connected graph connecting all the object instances across the images. Next, the graph is fed through a GNN whose goal is to separate it into multiple disconnected sub-graphs, each representing a distinct object in the scene. The GNN has access to both image evidence and coarse pose information, so that it can learn to merge geometric view information with the corresponding object features from different viewpoints and to ensure high quality object re-identification. Finally, once that the distinct objects are re-identified, the proposed end-to-end architecture estimates their geo-location.

Our contributions are: i) an efficient end-to-end, multi-view detector for static objects *ii*) that implements a novel method for incorporating graphs inside any anchor-based object detector. We further *iii*) formulate a GNN approach that jointly uses coarse relative camera pose information and image evidence to detect distinct objects in the scene. We validate our method experimentally on two different datasets of street-level panoramas and dash cam image sequences. Our GNN formulation for multi-view object detection and instance re-identification outperforms existing methods while being much more computationally efficient.

2 Related Work

Our proposed method is related to many different research topics in computer vision like pose estimation, urban object detection, and instance re-identification. A full review is beyond the scope of this paper and we rather provide here some representative works for each different topic and highlight the differences with our proposal.

Urban object detection from ground-level images is an application closely related to our paper. In [49, 2], the authors propose a method to detect and geo-locate street-trees from Google street-view panorama and aerial images with a hierarchical workflow. Trees are first detected in all images, detections are then projected to geographic coordinates. The detection scores are back-projected into images and re-evaluated, finally a conditional random field integrates all image evidence with other learned priors. In [55] a method is proposed to detect and geo-locate poles in Google street-view panoramas using object detectors along with a modified brute-force line-of-bearing approach to estimate pole locations. Authors in [23] perform a semantic segmentation of images and estimate monocular depth before feeding both sources of evidence into a Markov random field to geo-locate traffic signs. [54] detects road objects from ground level imagery and places them into the right location using semantic segmentation and a topological binary tree. All previously mentioned methods have in common that they propose hierarchical, multi-step workflows where pose and image evidence are treated separately unlike ours, which models them jointly. The most similar work to ours is [38], which proposes to jointly leverage relative camera pose information and image evidence as an end-to-end trainable siamese CNN. [19] detects trees as blobs from satellite imagery and a Digital Surface Model

using semantic segmentation. Here, we propose to formulate the problem as a graph neural network, which, intuitively, better represents the underlying data structure of multi-view object detection. Our GNN provides greater flexibility to add an arbitrary number of views (as opposed to a siamese CNN), is computationally drastically less costly, and achieves significantly better quantitative results. A large body of literature addresses urban object detection from an autonomous driving perspective with various existing public benchmark datasets like famous KITTI [14], CityScapes [6], or Mapillary [39, 33]. As opposed to our case, in this scenario dense image sequences are acquired with minor viewpoint changes in driving direction with forward facing cameras and often relative pose information of good accuracy, enabling object detection and re-identification across views [5, 24, 56]. In our setup, relative pose information is coarse and the viewpoints among the images might be remarkably different, i.e., large baseline between the cameras, making the correspondences matching a much harder task.

Learning to predict camera poses using deep learning was made popular by PoseNet [20] and still motivates various studies [37, 9, 51]. Estimating a human hand's appearance from any viewpoint can be achieved by coupling pose with image content [42]. Full human pose estimation is another task that benefits from combined reasoning across pose and scene content, for instance in [32], authors employ a multi-task CNN to estimate pose and recognize action. We rely here on public imagery without fine-grained camera pose information, which requires a different approach.

Multiple object tracking (MOT) and person re-identification is related to our setting, but significantly differs in that objects are moving while cameras are usually fixed. Again, many deep learning-based solutions have been reported, usually employing a siamese CNN [28], as it is an effective technique to measure similarity between image patches. In [25], for instance, authors propose to learn features using a siamese CNN for multi-modal inputs (images and optical flow maps). In [48], a siamese CNN and temporally constrained metrics are jointly learned to create a tracklet affinity model. [44] uses a combination of CNNs and recurrent neural networks (RNN) in order to match pairs of detections. Authors in [52], instead, solve the re-identification problem with a so-called center loss that tries to minimize the distance between candidate boxes in the feature space. A work that is closely related to ours is [3]. Authors formulate their MOT for person re-identification into a graph setup. The graph is composed of nodes that hold CNN features of the image crops of the persons over time, with edges created between all these instances. A message passing network is then used to propagate the node features throughout the graph. Similar to our work, the edges between these nodes are then classified to re-identify the person. This paper however focuses on a single camera setup, whereas in our case we need to re-identify objects from different views.

Graph neural networks (GNN) naturally adapt to non-grid structured data like molecules, social networks, point clouds, or road networks. Graph convolutional networks (GCNs), as introduced by [4], originally proposed a convolutional approach on spectral graphs, which was further extended in [7, 22, 27].

Another line of research investigated GCNs in the spatial domain [1, 40, 15, 34, 11], where spatial neighborhoods of nodes inside graphs are convolved. In order to make processing on large graphs more efficient, recent methods pool nodes in order to perform subgraph-level classification [17, 47]. These methods are described as fixed-pooling methods because they are based on the graph topology. Another idea is pooling nodes into a coarser representation based on weighted aggregations that are learned from the graph directly [53, 13, 8].In our approach, a GNN network is used to capitalize on the data structure by classifying edges between nodes to find the correspondence. We use the spatial convolution operator GraphConv [36] since spatial-based convolutions have proven to be more efficient, and to generalize better on other data [17, 50].

3 Method

We now describe in detail the proposed method. It is convenient to think of our architecture as composed of three stages: object detector, a GNN object re-identification and a geo-localization predictor, see Fig. 2. Note that, though we can see the model as made of three stages, the training is still conducted in an end-to-end manner, as the all the different parts support back-propagation.

3.1 Object Detection

In general, any state-of-the-art object detector could be used here. We choose EfficientDet [46] as it can provide state-of-the-art accuracy performance with a moderate amount of parameters and computations. The benefit of using an anchor based detector, is that our method can be adapted to other state-of-the-art methods [18, 31] that also use an anchor based solution. The whole pipeline could, however, easily be adapted to work with anchor-free object detectors if desired.

The input to the object detector is a set of multiple images, which represent a scene, alongside with their metadata $M = \{C^*_{lat}, C^*_{lng}, h^\circ\}$ where $C^*_{lat}, C^*_{lng}, h^\circ$ represent the cameras's latitude, longitude and heading angle respectively, which corresponds to the location in the 3D world of the cameras. Similar to most object detectors, we feed the images through the *backbone*, which is in our case an ImageNet-pretrained CNN, to extract features. The features maps are then fed to both a classifier network and a regression network in order to predict the class of the objects and regress the coordinates of its bounding box, respectively. During the training phase we obviously have access to the bounding box ground truth with objects IDs of the annotated instances.

We use the Focal Loss introduced in [30] to train the object detection classifier. This loss was selected as it helps to handle the class imbalance between positive and negative samples. In order to calculate the detection losses, we measure Intersection Over Union (IoU) and select the anchors that have the best overlap with our ground truth bounding boxes. This task is achieved by a threshold that splits the anchors into positive and negative proposals; both proposal

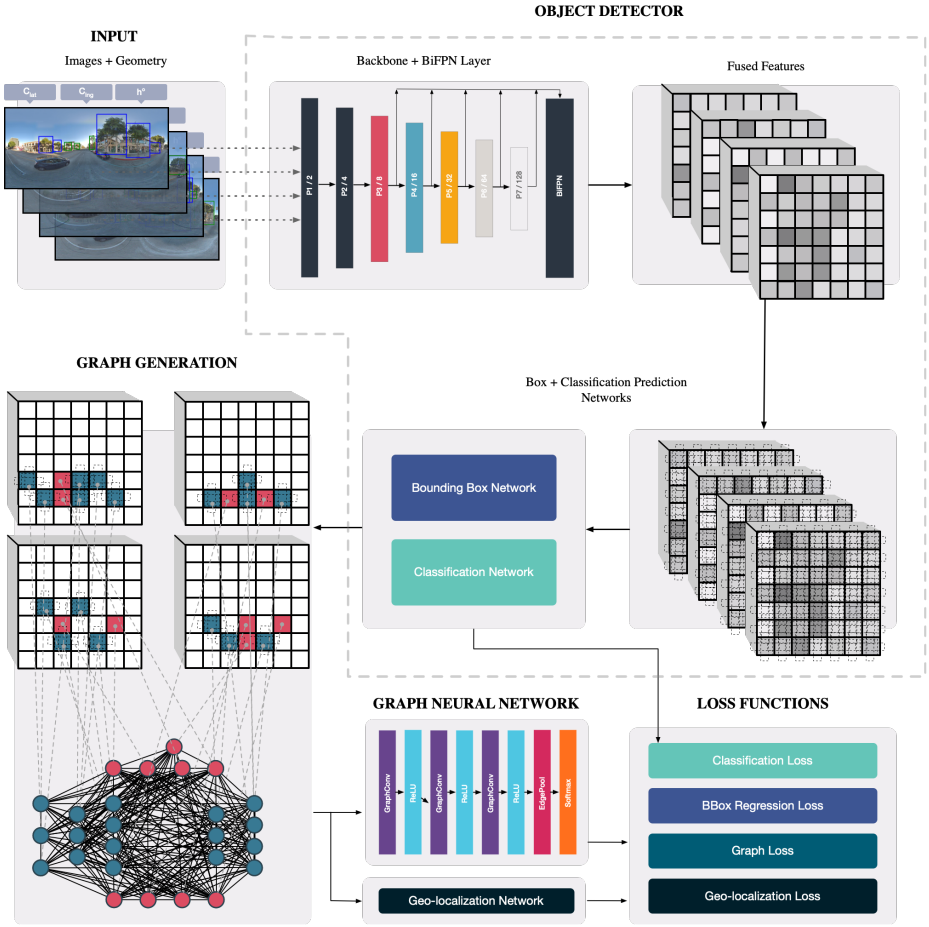

Fig. 2. Architecture of GeoGraph. A batch of images from multiple views and camera metadata information M pertaining to them are passed through the backbone network (EfficientNet) and the multi-scale feature aggregator (BiFPN) of the object detector that provides different levels of features. Anchors are then generated across the feature layers, and passed through two sub-networks to provide classification and box predictions. Based on the IoU of the ground truth with the anchors we select the positive and negative anchors. The features of these anchors are used to generate a dense fully connected graph. The graph is then fed to a GNN to predict if the nodes are corresponding by classifying their respective edge. In Parallel, the regressed bounding boxes of the positive anchors are passed to the Geo-Localization Network to regress the geo-coordinate.

will then be used in the next stage of our architecture (see Sec. 3.2). As for the object bounding box regression, we adopt a smooth-L_1 loss [16].

3.2 Object Re-identification

After detecting the objects across the different views, we need a method that is able to re-identify and recover all the distinct objects that appear of the scene. In this case both the number of input detections, and the number of distinct objects that occur in the scene, are not fixed and vary across different scene instances. This irregularity among the different instances poses some modelling challenges, which, fortunately, can be overcome by using a graph representation. Indeed, since graphs are usually used to represent irregular data, most of the graphs algorithms are suited to deal with a non-fixed number of nodes. As a result, we can map the object detections as nodes of a graph, and then use graph methods, which can handle graphs of different sizes, to perform object re-identification. In the following, we first define how the graph is created from the object detections, and then we describe how it is used to carry out the re-identification task.

Graph Generation We generate a graph $\mathcal{G} = (V, E)$ where V represents the set of N nodes, and E the set of edges connecting the nodes. From each view in the scene, we extract as many nodes as there are features vectors located inside the anchors proposed by the object detector. We then think of each node v as containing the associated feature vector extracted from the feature map to which we concatenate the coarse pose information of the image and the predicted bounding box values. The features selected from the feature map include all the ones contained in the positive and negative anchors predicted by the object detector. We then connect all the nodes in the graph to each other, building essentially an undirected fully connected graph.

During the training phase we build a second undirected graph, $\mathcal{G}_{\text{gt}}$, which contains the same nodes as $\mathcal{G}$ but with edges encoding the identification information. Using the groundtruth annotations, we set $e_{ij} = 1$ if nodes i and j belong to the same object, regardless of which images the nodes i and j are associated to. Otherwise, we set $e_{ij} = 0$, meaning that the nodes are disconnected. We basically connect each pair of nodes only if they come from the same object. Intuitively, it is convenient to think of $\mathcal{G}_{\text{gt}}$ as made of a set of disconnected sub-graph components, each representing a single individual object in the scene.

Graph Neural Network. At this stage we are provided with an input graph $\mathcal{G}$, which is a fully connected graph among all the feature cells of all the objects detected in the different views. The goal here is to train a GNN that receives as input $\mathcal{G}$, and disentangles the nodes of the graph that belong to different objects, i.e., the GNN should recover $\mathcal{G}_{\text{gt}}$.

We compose our GNN out of 3 GraphConv [36] layers with a ReLU activation after each convolution. The GraphConv uses message passing to aggregate

information from the neighborhood of i, denoted as $\mathcal{N}_i$, to update the feature representation H by:

$$H^{(k+1)}(v) = f_1^{W_1^{(k)}}\left(H^{(k)}(v), f_2^{W_2^{(k)}}\left(\left\{H^{(k)}(w)|w \in \mathcal{N}(v)\right\}\right)\right), \tag{1}$$

where k represents the layer of the GNN, W^k are the trainable/learned weights, $f_2^{W_2^{(k)}}$ is the aggregation function of $\mathcal{N}(v)$, and $f_1^{W_1^{(k)}}$ merges the neighborhood features. At this point in our network, we insert a dropout [45] layer for regularization. The output feature representations of the nodes are then passed to a modified or stripped down EdgePooling layer [8] for the edge classification. This operation consists in concatenating the features of all the neighboring nodes in the graph, and passing them through a linear transformation followed by a sigmoid non-linearity:

$$s_{ij} = \sigma\left(W \cdot (H_i \| H_j) + b\right), \tag{2}$$

where $\sigma()$ denotes the sigmoid function and $\|$ operator denotes a concatenation operation. s_{ij} represents the probability for the nodes i and j to belong to the same object in the scene. In this case the groundtruth value for each edge comes from $\mathcal{G}_{\text{gt}}$ and, as for the object detection, we train this classifier using focal loss to accommodate for dataset imbalance.

Note that by relying on a graph formulation we are able to effortlessly deal with a varying number of distinct objects in the scenes, focusing on "link prediction" instead of "graph classification". In fact, as we simply aim at disentangling the graph to separate the objects, the method is oblivious of the number of distinct objects in the scene (i.e., the object number is not hard-coded), and it is able to separate the graph into any number of disconnected components.

3.3 Geo-localization

We estimate the geo-coordinates of the identified objects similarly to [49, 38]. The regressed bounding boxes values are projected to real world geographic coordinates by taking advantage of the camera information that is coupled with the image. In order to perform this operation we further assume that the terrain is locally flat. By using the projection equations Eq. (3)-(4) we are able to map the object bounding box pixel locations x and y in East, North, Up (ENU) coordinates e_x, e_y, e_z and secondly recover the position of the object in the real world O_{lat}, O_{lng}.

$$(e_x, e_y, e_z) = \left(R\cos[\mathrm{C}_{lat}]\sin[\mathrm{O}_{lng} - \mathrm{C}_{lat}], R\sin[\mathrm{O}_{lat} - \mathrm{C}_{lat}], -\mathrm{h}^\circ\right) \tag{3}$$

$$\begin{aligned} x &= (\pi + \arctan(e_x, e_y) - \mathrm{h}^\circ)\, W/2\pi \\ y &= (\pi/2 - \arctan(C_h, z))\, H/\pi \end{aligned} \tag{4}$$

where R denotes the Earth's radius, W and H are the image's width and height, C_h denotes camera's height and $z = \sqrt{e_x^2 + e_y^2}$ is an estimate of the

object's distance from the camera. In order to improve the geo-localization accuracy we refine the predictions by feeding the geo-coordinates computed using Eq. (3)-(4) through a neural network. The geo-localization network is trained using the groundtruth through regression to obtain the object's geo-coordinate using a Mean Square Error (MSE) loss.

3.4 Inference operations

At inference time, the operations carried out in the whole pipeline are slightly different. First, the object detector generates the proposals for the anchors with classification confidences and bounding boxes. A threshold is then applied to the classification score to select only the detection proposals with a high classification confidence. Afterwards, Non Maximum Suppression (NMS) is used to filter further proposals and reduce redundancy.

With the remaining proposals, we create a fully connected graph as described before, with each node representing a feature vector contained inside the proposals, at which we concatenate the image's camera metadata information. The generated graph is then fed to the GNN network for edge classification. Finally, we classify the edges scores by applying a threshold on them with a decision boundary of 0.5. At this point, the obtained graph is supposed to be made of several disconnected components, one for each of the distinct objects in the scene. We remove sequentially each set of connected components until no nodes are left.

We finally compute the geo-location of the identified objects by utilizing the camera metadata information and the location of the bounding box in the image for all the views where the identified object appears. All the objects geo-coordinates computed from each different view are then separately refined with the geo-localization network and, finally, averaged to obtain the final prediction.

4 Experiments

4.1 Datasets

Pasadena Multi-View ReID. Instance labeled trees are ignored in most urban object datasets where they are rather labeled as vegetation. We consider the Pasadena Multi-View ReID [38] dataset that provides labeled instances of different trees acquired in Pasadena, California. The dataset offers approximately 4 Google Street View panoramic views for each tree instance. A scene of 4 views could contain multiple labeled instances, averaging 2 instances per scene, and other trees that are not instance labeled. There is a total of 6,020 individual tree instances.

The dataset consists of 6,141 panorama images of size 2048 x 1024 px. In total, there are around 25,061 annotated objects in the dataset. Each annotation includes the object's bounding box values, image geo-coordinates, camera heading, estimate of object distance from camera, ID and geo-coordinate of object. In our experiments, we follow the same data split introduced in [38], and allocate 4,298 images for training, 921 for validation, and 922 for testing.

Mapillary. The second dataset we consider is a crowdsourced one provided by Mapillary [1], that is different from the image segmentation dataset, Mapillary Vistas [39]. Normally, the dataset contains different types of objects, but this subset contains traffic signs only in an area of $2km^2$ London, England. In comparison to the Pasadena Multi-View ReID, the images acquired in this dataset are dominantly captured consecutively by forward-faced cameras mounted on vehicles with the object mostly facing change in scale with the same viewpoint, with the other instances being from pedestrians' smartphone cameras.

Given its crowdsourced nature, this dataset presents interesting challenges such as images being acquired at different times of the day, various camera sensors as well as image sizes. There are 31,442 different instances of traffic signs labeled in 74,320 images. An object instance appears on average in 4 images, and there are approximately 2 object instances in each image. Almost entirely, all instances of signs in the image are annotated. Each object instance in the dataset comes with its bounding box values, object ID, image geo-coordinates, instance geo-coordinates, heading of camera, in which images the object appears, and its height. It is important to note that the object geo-coordinate is attained through 3D Structure From Motion techniques (SFM). In contrast to the Pasadena Multi-View ReID, the objects are much smaller in comparison to trees, and much difficult to capture sideways due to the physical property of signs being thin. On the other hand, the dataset is much larger, thus easing the training process.

4.2 Implementation Details

We implemented GeoGraph using PyTorch [41]. For the object detector, a PyTorch implementation of EfficientDet was used. The backbone chosen for the EfficientDet was "EfficientNet-B5". As for the GNN component, we relied on the PyTorch Geometric package [10]. The Dropout [45] layers are used with a drop probability of 0.2. The learning rate is set to 0.001 initially with ADAM [21] as the optimizer. Each epoch takes approximately 45 minutes during training time on a NVIDIA 1080 Ti GPU ([38] needs 270 minutes). Our network uses 34.75M parameters while [38] uses 50M. We significantly reduce inference time per image from 0.78ms [38] to 0.32ms.

4.3 Object Detection & Re-identification

In Tab. 1 we report the results achieved by our method in the different datasets as well as state-of-the-art performance. The effectiveness of our approach is evaluated by correctly identifying an instance of an object across multiple views, as also visually illustrated in Fig. 3 and Fig. 4. Typically, the object detection method influences the re-identification process as a better mean Average Precision (mAP) ensures that the object is fed to the next stage of the pipeline for the re-identification. Moreover, note that the object detection scores for the proposed method does not change between the 2 and 2+ views, this is because the

[1] www.mapillary.com

object detection is always carried out on a single image at the time. Our method outperforms SSD-ReID-Geo [38] for detection mAP by 2.2% with the Pasadena dataset, and 1.7% with the Mapillary dataset. GeoGraph improvement can be attributed to the superiority of EfficientDet over SSD.

Our experiments for re-identification aim at validating whether using graphs with coarse pose information would assist in identifying the same object instance across views. The performance of the GNN component is based on whether the detections of the same object across different views have connecting edges. Therefore to calculate our mAP, we consider as true positives correctly predicted edges between a pair of detections with no other edge connections to different objects in the scene. Otherwise we consider them as false positives. In order to ensure a fair comparison with other methods, we perform experiments with a similar number of views. As shown in Tab. 1, increasing the views leads to higher re-identification mAP, with an improvement of 3.2% for Pasadena and 4.2% for Mapillary.

4.4 Geolocalization

The geo-localization component is evaluated in terms of a Mean Absolute Error (MAE) of the distance, measured in meters, between the predicted geo-coordinate averaged over the different views and the ground truth as shown in Fig. 5. The metric chosen to measure the distance is the Haversine distance which is defined as:

$$d = 2R \arcsin\left(\left(\sin^2\left(\frac{O_{lat} - G_{lat}}{2}\right) + \cos(G_{lat})\cos(O_{lat})\sin^2\left(\frac{O_{lng} - G_{lng}}{2}\right)\right)^{0.5}\right), \quad (5)$$

where O_{lat}, O_{lng} represent the detection's predicted geo-coordinates, and G_{lat}, G_{lng} represent the object's ground truth. As reported in Tab. 1, for the Pasadena dataset, the geo-localization error decreases as the number of views increases (12% improvement). As for Mapillary, we report again lower error with our GeoGraph and 6 views w.r.t. SSD-ReID-Geo [38] (3.4% improvement). However, we surprisingly did not outperform GeoGraph results achieved with only 2 views. This is probably be due to the way the ground truth of the Mapillary dataset is acquired (see Sec. 4.1), and adding more views brings a lot of noise in the representation of a scene and of the objects it contains.

4.5 Ablation Studies

Since the graphs are generated during training are created online, we assess this component separately to be able to evaluate its effect. In these experiments, we build the graph from our ground-truth by generating CNN features from the

Table 1. Quantitative assessment of our GeoGraph framework and related work on object detection, re-identification, and geo-localization tasks.

Method	# Views	Dataset	Detection mAP	Re-ID mAP	Geo-localization error (m)
MRF [23]	4	Pasadena	0.742	-	3.83
SSD-ReID-Geo [38]	2	Pasadena	0.682	0.731	3.13
Our GeoGraph	2	Pasadena	0.742	**0.754**	**2.94**
Our GeoGraph	4	Pasadena		**0.763**	**2.75**
MRF [23]	4	Mapillary	0.919	-	4.62
SSD-ReID-Geo [38]	2	Mapillary	0.902	0.882	4.36
Our GeoGraph	2	Mapillary	0.919	**0.902**	**3.88**
Our GeoGraph	6	Mapillary		**0.924**	4.21

image crops of multi-view images, which served as our node features. Using the labeled instances, edges were associated between the different instances of image crops across the different views. Throughout this experiment, the same settings of the GNN were used as mentioned in Sec. 3.2. We compare our method to a siamese CNN with a ResNet50 backbone trained with a contrastive loss that classified whether the two crops of the object are similar or not. We can observe

Table 2. Results for our graph matching method component evaluated with bypassing the object detector, and using image crops. We show a comparison between a Siamese CNN, a GNN based on CNN features and a GNN based on CNN features and camera metadata information to classify if pairs of objects are the same or not.

Method	Dataset	F1-Score
Siamese CNN	Pasadena	0.509
GNN	Pasadena	0.601
GNN-Geo	Pasadena	**0.640**
Siamese CNN	Mapillary	0.721
GNN	Mapillary	0.823
GNN-Geo	Mapillary	**0.873**

from results reported in Tab. 2 that adding geometric cues consistently helped to improve re-identification. Through different examination with different forms of the graph construction, we have found out that creating edges between nodes across multiple views but also within the same image in a fully connected dense manner led to better results than in the case where edges between nodes of the same image (i.e. detections within same image) are ignored. This can be explained by the fact that node aggregation performs better with fully connected graphs.

5 Conclusion

In this paper we have tackled the problems of object detection, re-identification across multiple views and geo-localization with a unified, end-to-end learnable framework. We propose a method that integrates both image and pose information and use a GNN to perform object re-identification. The advantage of the our GNN over standard Siamese CNN is the ability to deal with any number of views and be computational efficient. Experiments conducted on two public datasets have shown the relevance of our GeoGraph framework, which achieves high detection accuracy together with low geo-localization error. Furthermore, our approach is robust to occlusion, neighboring objects of similar appearance, and severe changes in viewpoints.

The proposed framework could be improved if based on a better geo-localization component. Furthermore, using the proposals generated from each view could improve the quality of the object detection step. Finally, while only street view images were used in our paper, our GeoGraph framework is compatible with other types of images. We thus would like to combine these ground-level views with aerial views in order to improve the overall performance and to make the best of multiple viewpoints following [26].

Acknowledgment: This project was supported by funding provided by the Hasler Foundation. We thank Mapillary for providing the dataset.

Fig. 3. Sample results obtained on the Pasadena dataset for multi-view object detection and re-identification. Trees were correctly detected (green) and further accurately re-identified across different views (cyan) when possible.

Fig. 4. Sample results obtained on the Mapillary dataset for multi-view object detection and re-identification. Here all detected objects (signs) were both detected and further re-identified (cyan) due to the higher similarity between views.

Fig. 5. Sample of geo-localization results for Pasadena comparing different methods. Green, blue, yellow and red circles represent the ground-truth, GeoGraph, SSD-ReID-Geo and MRF respectively. The orange bounding box exhibits how ground level imagery can be helpful to detect object obscured by buildings from aerial view.

References

1. Atwood, J., Towsley, D.: Diffusion-convolutional neural networks. In: Advances in Neural Information Processing Systems. pp. 1993–2001 (2016)
2. Branson, S., Wegner, J.D., Hall, D., Lang, N., Schindler, K., Perona, P.: From Google Maps to a fine-grained catalog of street trees. ISPRS Journal of Photogrammetry and Remote Sensing **135**, 13–30 (2018)
3. Brasó, G., Leal-Taixé, L.: Learning a neural solver for multiple object tracking. arXiv preprint arXiv:1912.07515 (2019)
4. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203 (2013)
5. Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3D object detection network for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1907–1915 (2017)
6. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3213–3223 (2016)
7. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in neural information processing systems. pp. 3844–3852 (2016)
8. Diehl, F.: Edge contraction pooling for graph neural networks. arXiv preprint arXiv:1905.10990 (2019)
9. En, S., Lechervy, A., Jurie, F.: RPNet: An end-to-end network for relative camera pose estimation. In: Proceedings of the European Conference on Computer Vision. pp. 738–745 (2018)
10. Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds (2019)
11. Fey, M., Lenssen, J.E., Weichert, F., Müller, H.: SplineCNN: Fast geometric deep learning with continuous b-spline kernels. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 869–877 (2018)
12. Fout, A., Byrd, J., Shariat, B., Ben-Hur, A.: Protein interface prediction using graph convolutional networks. In: Advances in neural information processing systems. pp. 6530–6539 (2017)
13. Gao, H., Ji, S.: Graph U-Nets. arXiv preprint arXiv:1905.05178 (2019)
14. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The KITTI dataset. The International Journal of Robotics Research **32**(11), 1231–1237 (2013)
15. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Proceedings of the International Conference on Machine Learning. vol. 70, pp. 1263–1272 (2017)
16. Girshick, R.: Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1440–1448 (2015)
17. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems. pp. 1024–1034 (2017)
18. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2961–2969 (2017)
19. Kampffmeyer, M., Salberg, A.B., Jenssen, R.: Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 1–9 (2016)

20. Kendall, A., Grimes, M., Cipolla, R.: Posenet: A convolutional network for real-time 6-dof camera relocalization. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2938–2946 (2015)
21. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
22. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
23. Krylov, V.A., Kenny, E., Dahyot, R.: Automatic discovery and geotagging of objects from street view imagery. Remote Sensing **10**(5), 661 (2018)
24. Ku, J., Mozifian, M., Lee, J., Harakeh, A., Waslander, S.L.: Joint 3D proposal generation and object detection from view aggregation. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 1–8 (2018)
25. Leal-Taixé, L., Canton-Ferrer, C., Schindler, K.: Learning by tracking: Siamese CNN for robust target association. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 33–40 (2016)
26. Lefèvre, S., Tuia, D., Wegner, J.D., Produit, T., Nassar, A.S.: Toward seamless multiview scene analysis from satellite to street level. Proceedings of the IEEE **105**(10), 1884–1899 (2017)
27. Levie, R., Monti, F., Bresson, X., Bronstein, M.M.: CayleyNets: Graph convolutional neural networks with complex rational spectral filters. IEEE Transactions on Signal Processing **67**(1), 97–109 (2019)
28. Li, W., Zhao, R., Xiao, T., Wang, X.: Deepreid: Deep filter pairing neural network for person re-identification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 152–159 (2014)
29. Li, Y., Vinyals, O., Dyer, C., Pascanu, R., Battaglia, P.: Learning deep generative models of graphs. arXiv preprint arXiv:1803.03324 (2018)
30. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2980–2988 (2017)
31. Liu, Y., Wang, Y., Wang, S., Liang, T., Zhao, Q., Tang, Z., Ling, H.: Cbnet: A novel composite backbone network architecture for object detection. arXiv preprint arXiv:1909.03625 (2019)
32. Luvizon, D.C., Picard, D., Tabia, H.: 2d/3d pose estimation and action recognition using multitask deep learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5137–5146 (2018)
33. Ma, D., Fan, H., Li, W., Ding, X.: The state of Mapillary: An exploratory analysis. ISPRS International Journal of Geo-Information **9**(1), 10 (2020)
34. Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model cnns. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5115–5124 (2017)
35. Monti, F., Frasca, F., Eynard, D., Mannion, D., Bronstein, M.M.: Fake news detection on social media using geometric deep learning. arXiv preprint arXiv:1902.06673 (2019)
36. Morris, C., Ritzert, M., Fey, M., Hamilton, W.L., Lenssen, J.E., Rattan, G., Grohe, M.: Weisfeiler and leman go neural: Higher-order graph neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 4602–4609 (2019)
37. Nakajima, Y., Saito, H.: Robust camera pose estimation by viewpoint classification using deep learning. Computational Vision Media **3**(2), 189–198 (2017)

38. Nassar, A.S., Lefèvre, S., Wegner, J.D.: Simultaneous multi-view instance detection with learned geometric soft-constraints. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 6559–6568 (2019)
39. Neuhold, G., Ollmann, T., Bulò, S.R., Kontschieder, P.: The Mapillary Vistas dataset for semantic understanding of street scenes. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 5000–5009 (2017)
40. Niepert, M., Ahmed, M., Kutzkov, K.: Learning convolutional neural networks for graphs. In: Proceedings of the International Conference on Machine Learning. pp. 2014–2023 (2016)
41. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch. In: NIPS Autodiff Workshop (2017)
42. Poier, G., Schinagl, D., Bischof, H.: Learning pose specific representations by predicting different views. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 60–69 (2018)
43. Qiu, J., Tang, J., Ma, H., Dong, Y., Wang, K., Tang, J.: Deepinf: Social influence prediction with deep learning. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 2110–2119 (2018)
44. Sadeghian, A., Alahi, A., Savarese, S.: Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 300–311 (2017)
45. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research **15**(1), 1929–1958 (2014)
46. Tan, M., Pang, R., Le, Q.V.: Efficientdet: Scalable and efficient object detection. arXiv preprint arXiv:1911.09070 (2019)
47. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
48. Wang, B., Wang, L., Shuai, B., Zuo, Z., Liu, T., Luk Chan, K., Wang, G.: Joint learning of convolutional neural networks and temporally constrained metrics for tracklet association. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 1–8 (2016)
49. Wegner, J.D., Branson, S., Hall, D., Schindler, K., Perona, P.: Cataloging public objects using aerial and street-level images-urban trees. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 6014–6023 (2016)
50. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks. arXiv preprint arXiv:1901.00596 (2019)
51. Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. In: Robotics: Science and Systems (2018)
52. Xiao, J., Xie, Y., Tillo, T., Huang, K., Wei, Y., Feng, J.: IAN: the individual aggregation network for person search. Pattern Recognition **87**, 332–340 (2019)
53. Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. In: Advances in Neural Information Processing Systems. pp. 4800–4810 (2018)
54. Zhang, C., Fan, H., Li, W., Mao, B., Ding, X.: Automated detecting and placing road objects from street-level images. arXiv preprint arXiv:1909.05621 (2019)
55. Zhang, W., Witharana, C., Li, W., Zhang, C., Li, X., Parent, J.: Using deep learning to identify utility poles with crossarms and estimate their locations from Google Street View images. Sensors **18**(8), 2484 (2018)

56. Zhao, J., Zhang, X.N., Gao, H., Yin, J., Zhou, M., Tan, C.: Object detection based on hierarchical multi-view proposal network for autonomous driving. In: Proceedings of the International Joint Conference on Neural Networks. pp. 1–6 (2018)