Deep Learning of Partial Graph Matching via Differentiable Top-K

Runzhong Wang†, Ziao Guo†, Shaofei Jiang, Xiaokang Yang, Junchi Yan*
MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University
{runzhong.wang,ziao.guo,jiangshaofei,xkyang,yanjunchi}@sjtu.edu.cn

Abstract

Graph matching (GM) aims at discovering node matching between graphs, by maximizing the node- and edge-wise affinities between the matched elements. As an NP-hard problem, its challenge is further pronounced in the existence of outlier nodes in both graphs which is ubiquitous in practice, especially for vision problems. However, popular affinity-maximization-based paradigms often lack a principled scheme to suppress the false matching and resort to handcrafted thresholding to dismiss the outliers. This limitation is also inherited by the neural GM solvers though they have shown superior performance in the ideal no-outlier setting. In this paper, we propose to formulate the partial GM problem as the top-k selection task with a given/estimated number of inliers k. Specifically, we devise a differentiable top-k module that enables effective gradient descent over the optimal-transport layer, which can be readily plugged into SOTA deep GM pipelines including the quadratic matching network NGMv2 as well as the linear matching network GCAN. Meanwhile, the attention-fused aggregation layers are developed to estimate k to enable automatic outlier-robust matching in the wild. Last but not least, we remake and release a new benchmark called IMC-PT-SparseGM, originating from the IMC-PT stereo-matching dataset. The new benchmark involves more scale-varying graphs and partial matching instances from the real world. Experiments show that our methods outperform other partial matching schemes on popular benchmarks.

1. Introduction

The importance of graph matching (GM) is recognized by its successful applications in machine learning [26], molecule matching [47], and various vision tasks [15, 30, 49]. Existing GM methods mainly follow the optimization formulation by maximizing the node-wise and edge-wise affinities of the matched elements, yielding an NP-hard problem known as Quadratic Assignment Problem [22].

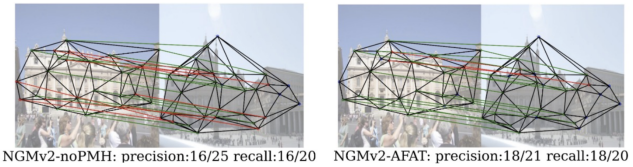


Figure 1. Illustrative comparison of injective graph matching (left) and partial graph matching (right) on our remade and released benchmark: **IMC-PT-SparseGM** (originated from IMC-PT [18], see Sec. 4 for details). Green for correct matches, red for wrong matches. Without partial matching, the GM solver greedily matches all nodes, leading to inferior accuracy. This paper presents a general learning method to mitigate this issue.

The ubiquitous challenge of partial matching, however, is less addressed by the existing affinity-maximization pipeline. Partial matching means only a partial set of the nodes are matched, due to the existence of outliers on both sides. As shown in Fig. 1, this is commonly encountered in (visual) graph matching [13, 16, 52, 53], where the existence of outliers is usually unavoidable due to the errors in keypoint detectors and (self-)occlusion of objects.

The recent line of deep graph matching papers [11, 45, 52] sheds light on the partial GM, whereby the higher capacity offered by deep neural networks on both the feature stage [52] and the solver stage [27, 45] will hopefully distinguish the outliers from inliers. However, there still lacks a general, principled approach that could enable the partial matching capability of existing graph matching neural networks. Some straightforward treatments such as thresholding [30], and adding dummy rows and columns [17] are relatively inflexible because their threshold and dummy values are manually assigned.

We aim at developing a general, unified partial matching handling approach, which can be readily integrated into SOTA GM networks. However, it is still an open question about how to determine whether or not a matching pair should be discarded. Besides, directly discarding the unwanted matching pairs causes non-differentiability and

*Runzhong Wang and Ziao Guo contributed equally. Junchi Yan is the correspondence author who is also with Shanghai AI Laboratory. The work was in part supported by National Key Research and Development Program of China (2020AAA0107600), NSFC (62222607, U19B2035), Shanghai Committee Science and Technology Project (22511105100). IMC-PT-SparseGM dataset: https://github.com/Thinklab-SJTU/IMCPT-SparseGM-dataset; Code: https://github.com/Thinklab-SJTU/ThinkMatch.

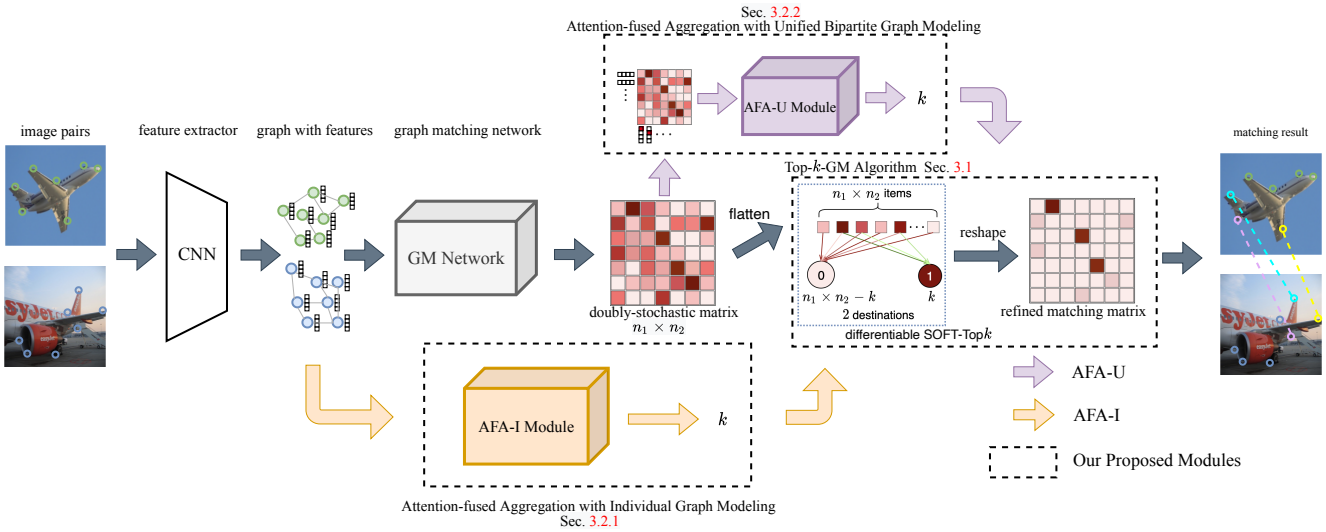


Figure 2. Deep learning paradigm for partial graph matching. The input images are sent to CNN to extract features and build the input graphs. The GM network predicts a doubly-stochastic matrix based on graphs with features, followed by a differentiable top-k algorithm. The number of inliers k is estimated by our proposed attention-fused aggregation networks.

truncated gradient. We resort to the doubly-stochastic matrix available in most SOTA GM pipelines [17, 44, 45], whereby the values in the doubly-stochastic matrix could be viewed as the matching confidence. The matchings with the top-k confidence values should be preserved, assuming that the number of inliers k is known. To this end, we present a top-k deep graph matching approach to address the partial matching challenge, whereby the differentiable top-k formulation [48] is followed to enable the end-to-end training.

Another challenge that naturally arises is how to estimate the value of k (i.e. number of inliers) from scratch. We identify the connection between the k-estimation problem and the graph-level similarity learning problem, whereby some prior successful models [1, 2] could be further exploited. In this paper, we present two networks to efficiently reuse the mid-layers that are available in most deep GM pipelines, namely **A**ttention-**F**used **A**ggregation (AFA) modules, based on the similarity of graphs aggregated from either individual graph features (Sec. 3.2.1) or the doubly-stochastic similarities on a unified bipartite graph (Sec. 3.2.2). The AFA modules are further integrated into the learning pipeline.

Besides, the severe partial matching issue that usually exists in downstream vision tasks is less addressed by existing graph matching evaluation protocols. We are thus motivated to collect and relabel a new benchmark, namely IMC-PT-SparseGM, which is originated from the stereo matching task in Image Matching Challenge PhotoTourism (IMC-PT) 2020 [18]. As summarized in Tbl. 1, the new benchmark addresses the severe partial matching challenge, and its graphs are of larger sizes than existing benchmarks.

The main contributions of the paper are as follows.

1) We formulate the partial (graph) matching problem as a top-k scheme, i.e. in the presence of outliers in both graphs. The scheme can be integrated into SOTA GM neural networks e.g. [17, 45] as a plugin.

2) Based on the top-k formulation, we devise an end-to-end and outlier-aware neural pipeline for partial GM learning and show its effectiveness. In contrast, we show that directly combining the top-k scheme with either a traditional solver like RRWM [7] or an outlier-blind neural solver e.g. NGMv2 [45] still produce poor results (see Tbl. 6), suggesting the necessity of our integrated learning paradigm.

3) To estimate the number of inliers k which is often unknown in practice, we devise an attention-based supervised graph neural network whose input consists of similarity information available in GM networks. With this estimation module, we enable a fully automatic neural solver for partial GM which to our best knowledge is new in the literature.

4) Our approach outperforms existing methods on popular GM benchmarks notably in the setting of partial GM. Last but not least, we remake and release a new benchmark to advance the research in GM by introducing larger graphs for matching and more partial matching instances.

Table 1. Visual GM datasets. "partial rate" means the mean percentage of occluded keypoints w.r.t. the universe of all keypoints.

dataset name	# visual graphs	avg # nodes	# universe	partial rate
Willow Object Class [6]	404	10	10	0.0%
Pascal VOC Keypoint [5]	8702	9.07	6 to 23	28.5%
IMC-PT-SparseGM-50 (ours)	25765	21.36	50	57.3%
IMC-PT-SparseGM-100 (ours)	25765	44.48	100	55.5%

2. Related Work

Deep learning of graph matching. The graph matching problem used to be formulated in its optimization form [22], whereby traditional algorithms are developed to tackle the matching problem [14, 23, 24]. The recent efforts on deep learning models for graph matching [27, 44, 51–53] are motivated by the higher model capacity offered by deep learn-

ing, in consideration of the robustness against noises and outliers. Existing deep graph matching models could be categorized into two main paradigms: 1) The quadratic matching paradigm [28, 29, 52] where both node and edge affinities are explicitly modulated, and the SOTA method is NGMv2 [45]; and 2) The linear matching paradigm [13, 44, 51, 53] where edge information is embedded into node features, resulting in a linear matching problem, and the SOTA method is GCAN [17]. It might be too early to judge which paradigm is better, but as their common challenge, the important partial matching setting is merely studied. These previous deep graph matching networks are viewed orthogonal to our approach, where we aim to develop a principled partial matching handling method that fits into most deep graph matching models. There also exist traditional solvers [8, 43] and learning model [28] tailored for partial graph matching, yet their architectures lack the flexibility to directly fuse with SOTA deep learning models, and their performances are inferior in experiments.

Partial matching handling. There exist methods to handle the challenges of partial matching and outliers, by exploiting additional priors. There is success in discovering common inliers among multiple graphs, which is addressed by a few multi-graph matching papers [4, 37, 40]. Maximum common subgraph (MCS) solvers [3, 50] rely on strict graph isomorphism to discard outliers, yet such isomorphism is not preserved in visual graph matching because of the geometric deformations in images. Another line of papers [12, 39] exploits transformation priors such as motion, homography, pose, etc. In this paper, however, we intend to relax the previous restrictions on priors and handle the general cases of partial matching in a data-driven manner.

3. Proposed Method

An overview of our deep learning of partial graph matching pipeline is shown in Fig. 2. In Sec. 3.1, we present a principled top-k graph matching approach that works with general doubly-stochastic matrices and supports gradient backpropagation, to discard outliers. In Sec. 3.2.1 and 3.2.2, we present attention-fused aggregation models with individual graphs and a unified graph, respectively, to estimate the number of inliers k in a data-driven manner.

3.1. Top-k-based Deep Partial Graph Matching

To encode partial matching constraint into deep graph matching pipeline, inspired by [48], we formulate choosing the top-k matches as an Optimal Transport (OT) problem, whereby all elements in the doubly-stochastic matrix are treated as matching confidences, and the k most confident matches are preserved. The procedure of our top-k-GM algorithm is summarized in Algorithm 1.

OT formulation of top-k. As shown in Fig. 2, we flatten the doubly-stochastic matrix $\mathbf{S}$ into its vectorized version

Algorithm 1 Top-k-based Deep Learning of Partial GM

Input: matching confidence $\mathbf{d}$; k; regularization factor ϵ.
construct $\mathbf{D}, \mathbf{c}, \mathbf{r}$ by Eq. 1; $\mathbf{\Gamma} = \exp\left(-\mathbf{D}/\epsilon\right)$;
repeat ▷ Sinkhorn optimal transport
  $\mathbf{\Gamma} = \text{diag}((\mathbf{\Gamma}\mathbf{1} \oslash \mathbf{r}))^{-1}\mathbf{\Gamma}$; ▷ row norm
  $\mathbf{\Gamma} = \text{diag}\big((\mathbf{\Gamma}^\top\mathbf{1} \oslash \mathbf{c})\big)^{-1}\mathbf{\Gamma}$; ▷ column norm
until $\mathbf{\Gamma}$ is converged
$\mathbf{S}_r = \text{Reshape}(\mathbf{\Gamma}_{:,2})$; ▷ rebuild matching matrix
if training **then**
  return $\mathbf{S}_r$; ▷ differentiable result, for training
else
  return GreedyTopK(Hungarian($\mathbf{S}_r$)); ▷ for testing
end if

$\mathbf{d} = [d_1, d_2, \dots, d_{n_1n_2}]$. To select the top-$k$ matches from $\mathbf{d}$ in a differentiable manner, the OT problem is formulated as moving each element in $\mathbf{d}$ to one of the two destinations: d_{max} and d_{min}. The capacities of d_{max} and d_{min} are k and $n_1n_2 - k$ respectively, so that the matches moved to d_{max} are selected, and the others moved to d_{min} are discarded. Denote $\mathbf{r}, \mathbf{c}$ as the marginal distributions, $\mathbf{D}$ as the distance matrix, $\mathbf{1}$ as an all-one vector, we have

$$\mathbf{r} = \mathbf{1}_{n_1n_2}^\top, \quad \mathbf{c} = [n_1n_2 - k, k]^\top, \tag{1}$$
$$\mathbf{D} = \begin{bmatrix} d_1 - d_{min} & d_2 - d_{min} & \cdots & d_{n_1n_2} - d_{min} \\ d_{max} - d_1 & d_{max} - d_2 & \cdots & d_{max} - d_{n_1n_2} \end{bmatrix}.$$

The OT problem is formulated as

$$\min_{\mathbf{\Gamma}} \text{tr}(\mathbf{\Gamma}^\top\mathbf{D}) \quad s.t.\ \mathbf{\Gamma}\mathbf{1} = \mathbf{r}, \mathbf{\Gamma}^\top\mathbf{1} = \mathbf{c}, \tag{2}$$

where $\mathbf{\Gamma} \in \{0, 1\}^{n_1n_2 \times 2}$ is the transportation matrix, $\mathbf{\Gamma}_{i,1} = 0, \mathbf{\Gamma}_{i,2} = 1$ means the i-th match is selected, $\mathbf{\Gamma}_{i,1} = 1, \mathbf{\Gamma}_{i,2} = 0$ means the i-th match is discarded.

Solving OT differentiably. Directly solving Eq. 2 does not offer a meaningful gradient for neural networks. To build an end-to-end learning pipeline, we modify the objective function by adding an entropic regularizer [9], and relax the constraints to continuous values $\mathbf{\Gamma} \in [0, 1]^{n_1n_2 \times 2}$:

$$\min_{\mathbf{\Gamma}} \text{tr}(\mathbf{\Gamma}^\top\mathbf{D}) + \epsilon h\left(\mathbf{\Gamma}\right) \quad s.t.\ \mathbf{\Gamma}\mathbf{1} = \mathbf{r}, \mathbf{\Gamma}^\top\mathbf{1} = \mathbf{c} \tag{3}$$

where $h\left(\mathbf{\Gamma}\right) = \sum_{i,j} \mathbf{\Gamma}_{i,j} \text{log} \mathbf{\Gamma}_{i,j}$. We then adopt the differentiable top-k algorithm [48] based on Sinkhorn algorithm [34]. $\mathbf{\Gamma}$ is firstly initialized via $\mathbf{\Gamma} = \exp\left(-\mathbf{D}/\epsilon\right)$, where ϵ is the regularization factor. The solving procedure is composed of alternative row/column-normalization on $\mathbf{\Gamma}$:

$$\mathbf{\Gamma} = \text{diag}\left(\mathbf{\Gamma}\mathbf{1} \oslash \mathbf{r}\right)^{-1}\mathbf{\Gamma}, \ \mathbf{\Gamma} = \text{diag}\left(\mathbf{\Gamma}^\top\mathbf{1} \oslash \mathbf{c}\right)^{-1}\mathbf{\Gamma}, \tag{4}$$

where $\oslash$ means element-wise division, $\text{diag}(\cdot)$ means building a diagonal matrix from a vector. The 2nd row of converged $\mathbf{\Gamma}$, denoted as $\mathbf{\Gamma}_{:,2}$, are the probabilities that each match should be selected in top-k, and we have $\sum \mathbf{\Gamma}_{:,2} = k$.

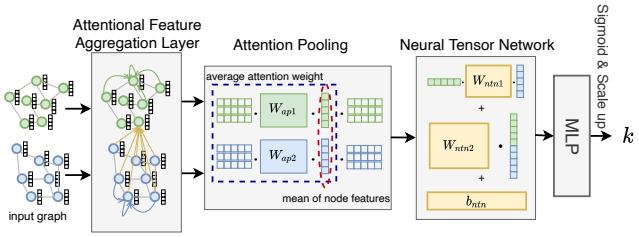


Figure 3. AFA-I module for the number of inliers k prediction with **A**ttention-**F**used **A**ggregation and **I**ndividual graph modeling.

Rebuilding matching matrix. We reshape $\mathbf{\Gamma}_{:,2}$ into $n_1 \times n_2$, which is viewed as the refined matching matrix $\mathbf{S}_r$. $\mathbf{S}_r$ is used for loss computation during training, whereby standard loss functions e.g. permutation loss [44] work smoothly. Note that we do not guarantee the 1-on-1 matching constraint among the selected k matches. Therefore, for testing, we perform Hungarian algorithm [19] on $\mathbf{S}_r$, and greedily select matches with top-k based on the confidence from the refined matching matrix. GreedyTopK in Algorithm 1 is elaborated in Appendix.

3.2. Attention-Fused Aggregation for k Prediction

Attention-Fused Aggregation (AFA) modules are our principled approach to predicting the number of inliers k. In this paper, we are motivated by the previous success of graph-level similarity learning models [1, 2, 25], whereby estimating k could be viewed as an extension of graph similarity learning, and higher similarity implies larger k. In this section, we present graph neural network-style aggregations to exploit the similarity information available at different stages of the GM pipeline, on either individual graphs (AFA-I, Sec. 3.2.1) or a unified graph (AFA-U, Sec. 3.2.2).

3.2.1 AFA-I: Individual Graph Modeling

Previous efforts [1, 25] demonstrate the feasibility of graph similarity learning from individual graphs, whereby its general motivation could be adapted to k prediction. In our AFA-I approach, we aggregate global features in each individual graph, and then compute global similarity score based on their global features. An overview of the AFA-I module is shown in Fig. 3. As shown in Fig. 2, AFA-I shares the CNN feature extractor with the GM network, and the graphs with features are inputs to AFA-I.

Attentional Feature Aggregation Layer. In this layer, we try to aggregate the features of nodes exploiting graph structure. Inspired by [25, 44], we adopt intra-graph aggregation layers and cross-graph attention layers. Consider $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$, with $|\mathcal{V}_1| = n_1$, $|\mathcal{V}_2| = n_2$, the intra-graph aggregation updates the feature of node i through:

$$\mathbf{x}_i^{l+1} = \xi_{\text{intr}}\left(\frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)}\left(\phi_{\text{intr}}\left(\mathbf{x}_j^l\right)\right), \zeta_{\text{intr}}\left(\mathbf{x}_i^l\right)\right), \tag{5}$$

where $\mathbf{x}_i^l$ is the the feature vector of node i in layer l, $\mathcal{N}(i)$ is the set of neighbors of node i. ξ, ϕ, ζ refer to the update function, message passing function, and self-update function, respectively, and can be implemented as any differentiable function that maps vectors to vectors. The cross-graph aggregation adopts an attention mechanism [41], and aggregate node features from the other graph, weighted by the similarity of features (suppose $i \in \mathcal{V}_1$):

$$\mathbf{x}_i^{l+1} = \xi_{\text{crs}}\left(\sum_{j\in\mathcal{V}_2}\left(\tilde{\mathbf{M}}_{i,j}^l \phi_{\text{crs}}\left(\mathbf{x}_j^l\right)\right), \zeta_{\text{crs}}\left(\mathbf{x}_i^l\right)\right), \tag{6}$$

where $\tilde{\mathbf{M}}^l \in [0,1]^{n_1\times n_2}$ is the similarity matrix among nodes from the graph pair in layer l, and $\tilde{\mathbf{M}}_{i,j}$ is at row i, column j. The building of $\tilde{\mathbf{M}}$ follows [44]:

$$\begin{aligned}\mathbf{A}_{i,j}^l &= \exp\left(\frac{{\mathbf{x}_i^l}^\top \mathbf{W}_{\text{afty}}\mathbf{x}_j^l}{\tau}\right), i\in\mathcal{V}_1, j\in\mathcal{V}_2,\\ \tilde{\mathbf{M}}^l &= \text{Sinkhorn}(\mathbf{A}^l),\end{aligned} \tag{7}$$

where $\mathbf{A} \in \mathbb{R}^{+n_1\times n_2}$ is the affinity matrix, representing the similarity of node features from the graph pair. $\mathbf{A}_{i,j}$ indicates the affinity score between node $i \in \mathcal{V}_1$ and node $j \in \mathcal{V}_2$. $\mathbf{W}_{\text{afty}} \in \mathbb{R}^{p\times p}$ contains learnable parameters used for affinity matrix construction, where p is the feature dimension. τ is a hyperparameter for scaling. $\tilde{\mathbf{M}}$ is generated by taking Sinkhorn operation [34] on affinity matrix and is guaranteed to be a doubly-stochastic matrix, which avoids numerical scale issues of cross-graph attention. Note that the Sinkhorn method here is for node-matching (see [44]), whose formulation is different from the one tackling the top-k OT problem in Sec. 3.1.

Attention Pooling. After feature aggregation, we obtain the node features of each graph via stacking node feature vectors. The node features are denoted as $\mathbf{X}_1 \in \mathbb{R}^{n_1\times p}$, $\mathbf{X}_2 \in \mathbb{R}^{n_2\times p}$ for $\mathcal{G}_1, \mathcal{G}_2$, respectively, where p is the feature dimension. We first perform average pooling to reach $\bar{\mathbf{x}}_1 \in \mathbb{R}^p$, $\bar{\mathbf{x}}_2 \in \mathbb{R}^p$ (column vectors). The global features are designed to be a weighted summation of node features with attention weights:

$$\mathbf{w}_i = \text{Sigmoid}\left(\mathbf{X}_i\mathbf{W}_{\text{ap},i}\bar{\mathbf{x}}_i\right), \ \mathbf{x}_{\text{glb},i} = \sum_{n=1}^{n_i} w_{i,n}\mathbf{x}_{i,n}, \tag{8}$$

where $\mathbf{W}_{\text{ap},i} \in \mathbb{R}^{p\times p}$ contains learnable parameters for $\mathcal{G}_i$. $\mathbf{w}_i \in \mathbb{R}^{n_i}$ is the attention weight of $\mathcal{G}_i$. $w_{i,n} \in \mathbb{R}$ is the n-th element of $\mathbf{w}_i$, and $\mathbf{x}_{i,n} \in \mathbb{R}^p$ is the n-th vector of $\mathbf{X}_i$. $\mathbf{x}_{\text{glb},i}$ is the global feature of $\mathcal{G}_i$.

Neural Tensor Network. The similarity between the global graph-level features is measured through a neural tensor network with multi-neurons [35]. For each neuron, the similarity score $\mathbf{s}$ is computed by:

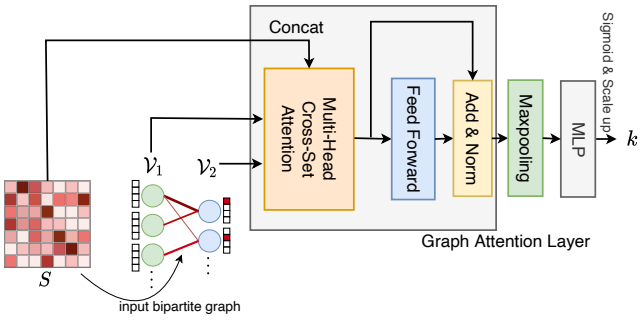


Figure 4. AFA-U module for k prediction with **A**ttention-**F**used **A**ggregation and **U**nified bipartite graph modeling.

$$\mathbf{s} = \mathbf{x}_{\text{glb},1}^{\top}\mathbf{W}_{\text{ntn},1}\mathbf{x}_{\text{glb},2} + \mathbf{W}_{\text{ntn},2}\begin{bmatrix}\mathbf{x}_{\text{glb},1}\\ \mathbf{x}_{\text{glb},2}\end{bmatrix} + \mathbf{b}_{\text{ntn}}, \tag{9}$$

where $[\cdot]$ denotes concatenation operation, and $\mathbf{W}_{\text{ntn},1}$, $\mathbf{W}_{\text{ntn},2}$, $\mathbf{b}_{\text{ntn}}$ are learnable parameters. The adopted multi-neuron architecture enables the model to focus on different information in different subspaces. Finally, we merge the similarity scores from all neurons via a 2-layer MLP with a final Sigmoid activation to generate k-proportion, and we multiply the k-proportion with $\min(n_1, n_2)$ to scale up and generate the final k by rounding.

3.2.2 AFA-U: Unified Bipartite Graph Modeling

The unified bipartite graph built from the doubly-stochastic matrix $\mathbf{S}$ also carries graph similarity information, and such information is explored previously in [2] by CNN. An overview of AFA-U is shown in Fig. 4, whereby the doubly-stochastic output of the GM network is exploited for k prediction. Note that $\mathbf{S}_{i,j}$ is the match confidence at row i, column j, also reflecting the similarity between node i in $\mathcal{G}_1$ and node j in $\mathcal{G}_2$, and $\mathbf{S}$ contains abundant local similarity information. Inspired by [21], we modulate $\mathbf{S}$ as a bipartite graph with weighted edges, whereby the node sets are $\mathcal{V}_1 = \{a_1, a_2, \dots, a_{n_1}\}$ and $\mathcal{V}_2 = \{b_1, b_2, \dots, b_{n_2}\}$. The edge weight between $a_i \in \mathcal{V}_1$ and $b_j \in \mathcal{V}_2$ is $\mathbf{S}_{i,j}$.

Graph Attention Layer. We aggregate the features on the bipartite graph via a graph attention layer [42], to aggregate similarity information (i.e. edge weights) into node embeddings. The architecture of the graph attention layer is mainly built upon transformers [41]. Similar to the multi-head attention module in transformers, the multi-head *cross-set* attention module maps a query and a set of key-value pairs to an output. We take the embeddings of nodes in $\mathcal{V}_1$ as the input query $\mathbf{Q} \in \mathbb{R}^{n_1 \times p_{\text{qkv}}}$, and the embeddings of nodes in $\mathcal{V}_2$ as both the key $\mathbf{K} \in \mathbb{R}^{n_2 \times p_{\text{qkv}}}$ and the value $\mathbf{V} \in \mathbb{R}^{n_2 \times p_{\text{qkv}}}$ in the input key-value pair. p_{qkv} is the dimension of each query, key, and value vector. In each head of the multi-head *cross-set* attention, the original attention weights $\mathbf{w}_{\text{ori}}$ (size $n_1 \times n_2$) are first computed by $\mathbf{Q}$ and $\mathbf{K}$:

$$\mathbf{w}_{\text{ori}} = \frac{\left(\mathbf{Q}\mathbf{W}^{Q}\right)\left(\mathbf{K}\mathbf{W}^{K}\right)^{\top}}{\sqrt{p_{\text{qkv}}}} \tag{10}$$

where $\mathbf{W}^{Q}$ and $\mathbf{W}^{K}$ are learnable parameters and independent in each head. The output is correspondingly the weighted summation of embeddings of nodes in $\mathcal{V}_2$. Following [21], the edge weight matrix $\mathbf{S}$ (in the bipartite graph) is concatenated to $\mathbf{w}_{\text{ori}}$ at a new dimension, then the new dimension is merged through a 2-layer MLP with 2-dim input and 1-dim output to generate the final attention weights $\mathbf{w}_{\text{final}}$ (size $n_1 \times n_2$):

$$\mathbf{w}_{\text{final}} = \text{softmax}\left(\text{MLP}\left(\begin{bmatrix}\mathbf{w}_{\text{ori}}\\ \mathbf{S}\end{bmatrix}\right)\right) \tag{11}$$

$\mathbf{w}_{\text{final}}$ is multiplied with the embedded $\mathbf{V}$ to produce the output in each head: $\text{Attn}\left(\mathbf{Q}, \mathbf{K}, \mathbf{V}\right) = \mathbf{w}_{\text{final}}(\mathbf{V}\mathbf{W}^{V})$. $\mathbf{W}^{V}$ is learnable and independent in each head. Outputs from all heads are merged via concatenation and linear transformation. Then an add & norm and a feed forward module from transformers [41] aggregate the final node embeddings.

Maxpooling and final MLP. The aggregated node embeddings are then max-pooled to retain only the most significant features, and a 2-layer MLP with Sigmoid activation is adopted to generate the final k-proportion. k is finally generated via scaling up, in the way same as Sec. 3.2.1.

Note that the information is mainly from $\mathbf{S}$, the initialization of node embeddings does not have a significant impact. In our implementation, following [21], we set node embeddings in $\mathcal{V}_1$ to be zero vectors, and in $\mathcal{V}_2$ to be one-hot vectors to enhance the effect of $\mathbf{S}$ in feature aggregation. Moreover, the positions of $\mathcal{V}_1$ and $\mathcal{V}_2$ can be swapped in our AFA-U module to produce a new k for double-checking. We build 2 parallel AFA-U modules for $\mathcal{V}_1$ and $\mathcal{V}_2$ respectively and take the mean of the 2 output ks as the final k.

3.2.3 Training the AFA Modules

In supervised training, the ground truth of k is available by counting the number of ground truth matches. Both AFA-I and AFA-U are trained with a Mean Square Error (MSE) loss, following standard regression learning pipelines.

3.3. Implementation Details

Network details and training setups. We select VGG16 [33] as the CNN feature extractor of GM networks following past methods [17, 29, 44, 45]. To avoid the error introduced by AFA modules and keep the training stable, we take the ground truth k_{gt} as the input of Algorithm 1 when training the GM network. The predicted k_{pred} is used during testing. The AFA modules are dependent on the performance of the GM network and CNN feature extractor, so we extend the original GM network training pipeline into three stages. In stage 1, we only train the CNN and GM network to improve their performance and generate better inputs for AFA modules. In stage 2, we only train the AFA modules for a small number of epochs as a warm-up. In

Table 2. F1 (%) on Pascal VOC Keypoints (unfiltered). PMH means Partial Matching Handling. Our methods are marked as gray.

GM Network	PMH	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
ZACR [43]	ZACR [43]	12.2	31.8	31.7	23.0	35.0	28.3	21.8	32.6	19.6	23.8	33.8	29.9	28.8	21.4	10.8	39.0	26.9	15.5	55.8	82.5	30.2
PCA-GM [44]	None	35.6	60.3	43.7	34.5	81.5	54.9	30.1	47.8	30.4	46.4	43.9	44.5	46.1	52.4	29.4	78.7	40.7	30.4	58.6	81.2	48.6
BBGM [29]	LPMP [38]	42.2	66.7	54.9	46.1	85.7	66.5	39.8	60.3	38.9	65.1	60.1	58.4	58.1	62.4	41.3	96.1	53.5	26.3	75.9	82.6	59.0
NGMv2 [45]	None	45.5	65.3	55.3	45.8	88.4	64.3	45.9	58.6	43.3	59.1	39.2	55.7	58.0	65.3	44.4	95.4	50.3	41.2	72.4	81.8	58.8
	Thresholding	48.3	65.4	55.3	48.6	87.6	63.0	51.1	61.1	39.6	63.3	33.6	59.2	59.3	63.4	46.9	95.2	53.5	45.5	73.4	81.4	59.4±0.4
	Dummy node	44.7	61.9	57.1	41.9	83.9	63.9	54.1	60.8	40.5	64.2	36.2	60.6	60.8	61.9	48.7	91.2	56.2	37.4	63.2	82.2	58.6±0.5
	AFAT-U (ours)	45.7	67.7	57.3	44.9	90.1	65.5	49.9	59.3	44.0	62.0	54.9	58.4	58.6	63.8	45.9	94.8	50.9	37.3	74.2	82.8	**60.2**±0.4
	AFAT-I (ours)	45.0	67.3	55.9	45.6	90.3	64.6	48.7	58.0	44.7	60.2	54.8	57.2	57.5	63.4	45.2	95.3	49.3	41.6	73.6	82.4	59.9±0.3
GCAN [17]	Dummy node	46.3	67.7	57.4	45.0	87.1	64.8	57.5	61.2	40.8	61.6	37.3	59.9	59.2	64.6	49.7	95.1	54.5	28.5	77.9	83.1	59.7±0.3
	AFAT-U (ours)	47.1	70.8	58.1	45.8	90.8	66.5	49.6	58.8	50.6	64.6	47.2	60.5	62.3	65.7	46.3	95.4	52.7	47.4	74.2	83.8	**62.0**±0.2
	AFAT-I (ours)	46.1	69.9	56.1	46.6	90.7	66.1	48.1	57.9	49.9	63.9	50.4	59.0	61.6	65.0	44.7	95.5	50.9	49.2	74.0	83.8	61.6±0.3

Table 3. The GPU memory cost (GB) of matching two graphs (batch size=1) w.r.t. dense and sparse NGMv2. The original dense version exceeds memory limit of RTX 3090 (24GB) at 110 nodes, whereas our sparse version scales-up more efficiently.

Number of Nodes	40	50	60	70	80	90	100	110
Dense NGMv2 [45]	2.82	3.05	4.42	6.26	8.80	12.48	17.59	26.06
Sparse NGMv2 (ours)	2.73	2.73	2.74	2.75	2.75	2.76	2.77	2.78

stage 3, we jointly train the GM network and AFA modules to make the models fit each other.

Sparse implementation of NGMv2 [45]. We also notice that the demand on GPU memory grows significantly for quadratic assignment-based GM networks as the number of nodes increases. Therefore, we implement a sparse version of SOTA quadratic GM network NGMv2 [45]. In Tbl. 3, we show that the scalability issue is well-addressed by our sparse version in terms of the GPU memory cost.

4. Our IMC-PT-SparseGM benchmark

We provide a new visual graph matching benchmark addressing partial matching and graphs with larger sizes, based on the novel stereo benchmark Image Matching Challenge PhotoTourism (IMC-PT) 2020 [18]. In Sec. 4.1 we discuss the original IMC-PT benchmark, and Sec. 4.2 presents how to transform the original stereo benchmark to GM. Sec. 4.3 provides statistics for our new benchmark.

4.1. Introducing The Original IMC-PT Benchmark

The original IMC-PT 2020 [18] contains photos of 16 tourism attractions around the world collected from Yahoo Flickr (https://www.flickr.com/). Based on the large collection of tourism photos, the authors of [18] reconstruct dense 3D point clouds and camera poses as ground truth labels with off-the-shelf Structure from Motion (SfM) software colmap [31, 32]. The IMC-PT is proposed based on a similar insight as ours: traditional local features are often evaluated on small datasets through proxy metrics, which sometimes may not translate into downstream applications, distorting the actual performance and impeding the development of new technologies. Therefore, training and evaluating new image matching strategies on large-scale and challenging benchmarks are in great need. To this end, the authors of [18] hold an open challenge in image matching with a new dataset, called Image Matching Challenge PhotoTourism (IMC-PT) 2020. Based on IMC-PT, we remake and release our IMC-PT-SparseGM benchmark for visual graph matching.

4.2. Building IMC-PT-SparseGM benchmark

We emphasize that the visual GM task, which is the main focus of this paper and the new benchmark, is notably different from general image matching which often refers to dense matching without explicit graph structure e.g. [30, 36]. Visual GM models are usually evaluated on matching sparse semantic keypoints, which emphasizes more on exploiting graph structures. Here we set out below our steps to transform the image matching benchmark (IMC-PT) to visual GM.

Building 3D point clouds. The IMC-PT provides dense point coordinates detected by colmap [31, 32] for each image, together with a camera pose matrix for 2D-3D projection. However, these labels are not ready to build a graph matching benchmark. Our first step is building a dense 3D point cloud based on the 2D points and camera poses.

Selecting anchors. To build consistent correspondence for a GM benchmark, our next step is selecting a batch of anchors. Anchors are randomly selected from the 3D point clouds, and the selected anchors must satisfy the following requirements: 1) an anchor must have keypoints from at least 10 different images within its 0.015δ pixels neighborhood, where δ denotes the diameter of the point cloud. This prevents selecting rare anchor points that seldom appear in the collection (e.g. a random person passing by). 2) we set a minimal Euclidean distance (0.02δ pixels) between any two anchors to ensure that the anchors are evenly distributed in the scene. To offer a practical scale-up w.r.t. existing benchmarks, as shown in Tbl. 1, we set 50 and 100 anchors when building the new benchmark.

Judging the visibility of anchors in 2D images. Our IMC-PT-SparseGM benchmark addresses the challenge of partial matching, i.e. not all anchors appear in an image due to different camera pose, zoom-in, self-occlusion, etc. After selecting the anchors, we determine the visibility of anchors in each image by checking the minimal Euclidean distance between an anchor and the nearest visible point in the image. If the distance is below the threshold of 0.015δ pixels, it is marked as a visible anchor.

Projecting to 2D Image. Finally, as the anchors are with

Table 4. F1 (%) on Willow Object Class (+random outliers) and IMC-PT-SparseGM (50/100 anchors). Our methods are marked as gray.

Dataset name		Willow Object Class						IMC-PT-SparseGM (50 anchors)				IMC-PT-SparseGM (100 anchors)			
GM Network	PMH	car	duck	face	mbike	bottle	mean	reichstag	sacre_coeur	st_peters_square	mean	reichstag	sacre_coeur	st_peters_square	mean
ZACR [43]	ZACR [43]	47.3	44.7	77.7	39.9	53.6	52.6	72.1	33.7	29.5	45.1	39.4	33.1	30.4	34.3
PCA-GM [44]	None	55.8	56.5	81.2	46.4	58.1	59.6	83.4	47.5	58.5	63.1	70.7	43.1	58.8	57.5
BBGM [29]	LPMP [38]	65.1	60.7	85.5	71.6	65.5	69.7	85.4	55.1	59.3	66.6	88.1	55.0	56.4	66.5
NGMv2 [45]	None	78.9	66.6	84.3	63.1	76.0	73.8	90.8	55.9	64.3	70.3	78.4	54.9	69.3	67.6
	Thresholding	86.8	74.5	91.2	71.0	83.8	81.4±0.2	91.4	56.8	65.8	71.3±0.3	80.3	56.9	71.6	69.6±0.3
	Dummy node	83.3	69.7	95.7	68.8	86.7	80.8±0.4	88.5	56.1	63.0	69.2±0.5	80.0	57.0	71.3	69.5±0.3
	AFAT-U(ours)	82.6	74.5	90.6	73.9	87.0	81.7±0.5	90.5	58.7	66.9	72.0±0.3	81.7	57.0	72.2	**70.3**±0.2
	AFAT-I(ours)	84.6	75.7	92.0	74.5	88.6	**83.1**±0.2	92.3	58.7	66.7	**72.8**±0.4	82.0	57.0	71.4	70.1±0.3
GCAN [17]	Dummy node	74.8	75.7	92.8	77.1	83.5	80.8±0.2	87.2	55.1	63.0	68.4±0.5	80.4	55.7	72.8	69.6±0.4
	AFAT-U(ours)	80.1	78.0	90.6	76.0	87.0	82.3±0.3	86.9	59.4	67.1	71.1±0.4	82.6	58.2	73.8	**71.5**±0.2
	AFAT-I(ours)	82.2	77.7	92.7	77.2	88.6	**83.7**±0.3	91.0	60.3	67.3	**72.9**±0.6	82.7	57.8	72.4	70.9±0.4

Table 5. Mean F1 (%) with random outliers (in line with [28]).

# outlier (Pascal VOC, intersection)	0	1	2	3	4	5
EAGM [28]	70.5	60.6	53.3	47.6	42.6	38.4
NGMv2 [45]+AFAT-U (ours)	**72.2**	**65.9**	**61.2**	**57.5**	**53.8**	**50.3**

3D coordinates, all visible anchors are projected back to the 2D image based on the camera pose matrices provided by IMC-PT. The 2D coordinates and the corresponding anchor indices are saved together with the RGB image to form our new benchmark. Please refer to the supplementary material for more details about building the new benchmark.

4.3. Details of IMC-PT-SparseGM

Our IMC-PT-SparseGM benchmark consists of images of 16 tourism attractions around the world. Each image has a file storing its visible keypoints. Each tourism attraction corresponds to an `npz` file of meta-information including the image names and the number of keypoints. As shown in Tbl. 1, among all visual graph matching benchmarks, our IMC-PT-SparseGM has the largest number of images (25,765), the largest number of nodes (21.36/44.48), and the highest partial rate (57.3%/55.5%). Our benchmark also owns the flexibility to further scale up by simply adjusting the anchor numbers and regerenrating. Compared to [5, 6] that mainly consider matching semantic keypoints from single objects, the IMC-PT-SparseGM benchmark involves matching larger scenes and we take a step closer to the real-world downstream tasks e.g. structure from motion.

5. Experiments

5.1. Metric and Peer Methods

Evaluation metric. We report the matching F1 scores between given graph pairs. The matching F1 score is the harmonic mean of matching precision and matching recall. Given predicted permutation matrix $\mathbf{P}_{\text{pred}}$ containing k_{pred} matches, and ground truth permutation matrix $\mathbf{P}_{\text{gt}}$ containing k_{gt} matches, denote $\odot$ as element-wise product, matching F1 score is computed as $\text{precision} = \sum(\mathbf{P}_{\text{pred}} \odot \mathbf{P}_{\text{gt}}) / k_{\text{pred}}, \text{recall} = \sum(\mathbf{P}_{\text{pred}} \odot \mathbf{P}_{\text{gt}}) / k_{\text{gt}}, \text{F1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$. We also run 5 random restarts for peer methods whose performances are close, and report the 95% confidence intervals as error bars.

Our Approach. Our method can serve as a **partial matching handling** (**PMH**) plugin and can be integrated with any GM network whose output is a doubly stochastic matrix. We name our PMH method **A**ttention-**F**used **A**ggregation with **T**op-k-GM (**AFAT**). AFAT-I and AFAT-U refer to using AFA-I module and using AFA-U module to estimate k, respectively. To show our flexibility, we implement based on NGMv2 [45], a SOTA quadratic matching network, and GCAN [17], a SOTA linear matching network.

PMH baselines. In [17, 30], partial matching is handled by adding dummy rows and columns to the Sinkhorn algorithm. In [17], the discretization step is done by Gurobi commercial solver. We name this method as *dummy node*, and we replace the commercial solver with an adapted Hungarian algorithm [20], whose output is exactly the same. We also compare with the straightforward *thresholding* method whereby the threshold value is carefully tuned.

GM baselines. We also conduct experiments on PCA-GM [44] and BBGM [29] without PMH for comparison. Conceptually, PCA-GM and many other networks [11, 51, 53] could also fit into our top-k-GM learning paradigm because their output is doubly-stochastic, yet BBGM cannot because it is based on discrete black-box solver. The learning-free partial GM solver ZACR [43] is also selected as a baseline, which owns the ability to predict the inlier number k during GM solving. The GM network EAGM [28] also considers partial graph matching in its network architecture. Its official implementation is based on TensorFlow, and could not be easily adapted to the more popular PyTorch experiment protocol. Therefore, instead of reimplementing EAGM in PyTorch, we compare our AFAT on the same setting as [28]. Note that we refrain from directly comparing with multi-graph methods in experiments because they use extra information.

Testbed. All experiments are done on our workstation with AMD 3970X, RTX 3090, and 128GB memory.

5.2. Results and Discussions

Pascal VOC Keypoints. We conduct experiments on Pascal VOC Keypoints dataset [10] with Berkeley annotations [5], containing 20 classes of instances. Outliers are prevalent due to occlusion and variances in scale and pose. We crop the instances around bounding boxes and resize the

Table 6. Ablation study addressing the necessity of learning. We report mean F1 (%), assuming that the ground truth k is given.

Dataset	Solver	Alg. 1 in test	Train	Alg. 1 in train	F1
PascalVOC	RRWM	✗	✗	✗	20.3
	RRWM	✓	✗	✗	19.4
	NGMv2	✓	✓	✗	60.7
	NGMv2	✓	✓	✓	**62.3**
WillowObject	RRWM	✗	✗	✗	20.1
	RRWM	✓	✗	✗	18.9
	NGMv2	✓	✓	✗	85.2
	NGMv2	✓	✓	✓	**85.5**
IMC-PT-SparseGM-50	RRWM	✗	✗	✗	39.6
	RRWM	✓	✗	✗	39.1
	NGMv2	✓	✓	✗	72.3
	NGMv2	✓	✓	✓	**73.6**
IMC-PT-SparseGM-100	RRWM	✗	✗	✗	34.6
	RRWM	✓	✗	✗	34.0
	NGMv2	✓	✓	✗	70.7
	NGMv2	✓	✓	✓	**71.8**

images to 256 × 256, and take 7,020 images as the training set and other 1,682 images as the test set. The "unfiltered" setting is followed, and experimental results are shown in Tbl. 2. The improvements w.r.t. other PMH methods brought by our AFAT-U and AFAT-I are consistent with different GM network embodiments.

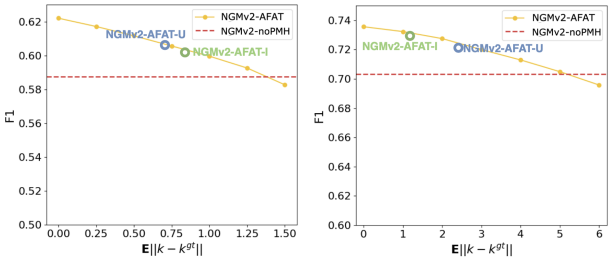
We also conduct a fair comparison with [28] following their setting, where all outliers are firstly removed (intersection filtering), and random outliers are then added to the second graph. The number of outliers is from 0 to 5. We directly run an evaluation on our pretrained NGMv2+AFAT-U model, and the results are listed in Tbl. 5, and our method also demonstrates better robustness against random outliers.

Willow Object Class. The dataset [6] contains 5 categories of images that have been aligned in pose, and its original version does not have outliers. Each image has 10 keypoints. To build a partial matching setting, we randomly add 1 to 10 background outliers to each image. We select 20 images as the training set in each category and take the remaining images for test. The left half of Tbl. 4 shows that our methods outperform other PMH methods.

IMC-PT-SparseGM. We also perform experiments on our released IMC-PT-SparseGM benchmark, which is more relevant to partial graph matching in real world. Following the train-test split in [18], we take 13 classes as the training set and the other 3 classes as the test set. Experiments are conducted on the benchmark with both 50 anchors and 100 anchors, and results are reported in the right half of Tbl. 4. The standard, dense implementation of NGMv2 [45] cannot fit into the GPU memory, thus our sparse implementation introduced in Sec. 3.3 is adopted. The improvement of our AFAT-U and AFAT-I is consistent in all settings.

5.3. Further Studies

Necessity of learning in top-k-GM. We propose a learning paradigm by integrating differentiable top-k-GM algorithm instead of a trivial combination of top-k selection and GM solvers. In this ablation study, we prove the necessity of learning in our paradigm. We consider RRWM and NGMv2, a traditional GM solver and an outlier-blind neural GM solver, respectively, and perform the ablation study in Tbl. 6. We decouple the possible influence of k estimation error by dropping the AFA module and using ground truth k instead. We can see that directly combining top-k algorithm with the traditional solver RRWM leads to inferior performance, and coupling differentiable top-k in training distinctly improves the performance of the neural solver NGMv2, indicating the effectiveness of our proposed learning paradigm by integrating differentiable top-k-GM.



(a) Pascal VOC Keypoints (b) IMC-PT-SparseGM-50

Figure 5. Relations between matching F1 score and k error.

Impact of the accuracy of k prediction. k is crucial in our top-k-based GM, so it is necessary to understand (at least empirically) the relation between matching accuracy and k error. In this study, we add Gaussian random noises to k_{gt} to simulate different levels of k errors and evaluate the matching accuracy of NGMv2 based on the noisy ks on Pascal VOC and IMC-PT-SparseGM (50 anchors). Fig. 5 shows the relationship between matching F1 and k error. We also plot the accuracy and k error of our AFA models, worth noting that the error pattern of AFA networks may be different from Gaussian noise, thus they do not strictly lie on the yellow curve. As shown in Fig. 5, there is an "acceptable k-error range", where the acceptable k-error is also proportional to the number of nodes. Besides, our AFA models all lie within the "acceptable k-error range".

6. Conclusion

We have presented a top-k-based framework to tackle the partial graph matching problem, which is ubiquitous in vision. Specifically, we devise an end-to-end and outlier-aware neural pipeline. Then an attention-based graph neural network is devised to estimate k. A new benchmark based on IMC-PT 2020, which is better suited for partial graph matching problem is remade and will be released. Extensive experimental results on both classic and our new benchmarks, including the comparison with peer partial matching handling methods and the studies on our integrated learning paradigm and the acceptable range of k error, show the effectiveness and significance of our work.

References

[1] Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 384–392, 2019. 2, 4

[2] Yunsheng Bai, Hao Ding, Yizhou Sun, and Wei Wang. Convolutional set matching for graph similarity. *arXiv preprint arXiv:1810.10866*, 2018. 2, 4, 5

[3] Yunsheng Bai, Derek Xu, Yizhou Sun, and Wei Wang. Glsearch: Maximum common subgraph detection via learning to search. In *ICML*, pages 588–598, 2021. 3

[4] Florian Bernard, Johan Thunberg, Paul Swoboda, and Christian Theobalt. HiPPI: Higher-order projected power iterations for scalable multi-matching. In *Int. Conf. Comput. Vis.*, pages 10284–10293, 2019. 3

[5] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *Int. Conf. Comput. Vis.*, pages 1365–1372. IEEE, 2009. 2, 7

[6] Minsu Cho, Karteek Alahari, and Jean Ponce. Learning graphs to match. In *Int. Conf. Comput. Vis.*, pages 25–32, 2013. 2, 7, 8

[7] Minsu Cho, Jungmin Lee, and Kyoung Mu Lee. Reweighted random walks for graph matching. In *Eur. Conf. Comput. Vis.*, pages 492–505. Springer, 2010. 2, 11

[8] Minsu Cho, Jian Sun, Olivier Duchenne, and Jean Ponce. Finding matches in a haystack: A max-pooling strategy for graph matching in the presence of outliers. In *CVPR*, pages 2083–2090, 2014. 3

[9] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Neural Info. Process. Systems*, pages 2292–2300, 2013. 3

[10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 7

[11] Matthias Fey, Jan E Lenssen, Christopher Morris, Jonathan Masci, and Nils M Kriege. Deep graph matching consensus. In *Int. Conf. Learn. Rep.*, 2020. 1, 7

[12] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 3

[13] Quankai Gao, Fudong Wang, Nan Xue, Jin-Gang Yu, and Gui-Song Xia. Deep graph matching under quadratic constraint. In *Comput. Vis. Pattern Recog.*, pages 5069–5078, June 2021. 1, 3

[14] Steven Gold and Anand Rangarajan. A graduated assignment algorithm for graph matching. *Trans. Pattern Anal. Mach. Intell.*, 18(4):377–388, 1996. 2

[15] Jiawei He, Zehao Huang, Naiyan Wang, and Zhaoxiang Zhang. Learnable graph matching: Incorporating graph partitioning with deep feature learning for multiple object tracking. In *Comput. Vis. Pattern Recog.*, 2021. 1

[16] Bo Jiang, Pengfei Sun, Ziyan Zhang, Jin Tang, and Bin Luo. Gamnet: Robust feature matching via graph adversarial-matching network. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 5419–5426, 2021. 1

[17] Zheheng Jiang, Hossein Rahmani, Plamen Angelov, Sue Black, and Bryan M Williams. Graph-context attention networks for size-varied deep graph matching. In *Comput. Vis. Pattern Recog.*, pages 2343–2352, 2022. 1, 2, 3, 5, 6, 7

[18] Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. Image Matching across Wide Baselines: From Paper to Practice. *Int. J. Comput. Vis.*, 2020. 1, 2, 6, 8, 11

[19] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 4

[20] H. W. Kuhn. The hungarian method for the assignment problem. In *Export. Naval Research Logistics Quarterly*, pages 83–97, 1955. 7

[21] Yeong-Dae Kwon, Jinho Choo, Iljoo Yoon, Minah Park, Duwon Park, and Youngjune Gwon. Matrix encoding networks for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 34:5138–5149, 2021. 5

[22] E. L. Lawler. The quadratic assignment problem. *Management Science*, 9(4):586–599, 1963. 1, 2

[23] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *Int. Conf. Comput. Vis.*, pages 1482–1489, 2005. 2

[24] Marius Leordeanu, Martial Hebert, and Rahul Sukthankar. An integer projected fixed point method for graph matching and map inference. In *Neural Info. Process. Systems*, pages 1114–1122. Citeseer, 2009. 2

[25] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *Int. Conf. Mach. Learn.*, pages 3835–3845. PMLR, 2019. 4

[26] Chang Liu, Chenfei Lou, Runzhong Wang, Alan Yuhan Xi, Li Shen, and Junchi Yan. Deep neural network fusion via graph matching with applications to model ensemble and federated learning. In *Int. Conf. Mach. Learn.*, pages 13857–13869. PMLR, 2022. 1

[27] A. Nowak, S. Villar, A. Bandeira, and J. Bruna. Revised note on learning quadratic assignment with graph neural networks. In *Data Science Workshop*, 2018. 1, 2

[28] Jingwei Qu, Haibin Ling, Chenrui Zhang, Xiaoqing Lyu, and Zhi Tang. Adaptive edge attention for graph matching with outliers. In *Int. Joint Conf. Artificial Intell.*, 2021. 3, 7, 8

[29] Michal Rolínek, Paul Swoboda, Dominik Zietlow, Anselm Paulus, Vít Musil, and Georg Martius. Deep graph matching via blackbox differentiation of combinatorial solvers. In *Eur. Conf. Comput. Vis.*, pages 407–424. Springer, 2020. 3, 5, 6, 7, 11

[30] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Comput. Vis. Pattern Recog.*, pages 4938–4947, 2020. 1, 6, 7

[31] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 6, 11

[32] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 6, 11

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5

[34] R. Sinkhorn and A. Rangarajan. A relationship between arbitrary positive matrices and doubly stochastic matrices. *Ann. Math. Statistics*, 1964. 3, 4

[35] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. *Advances in neural information processing systems*, 26, 2013. 4

[36] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou. Loftr: Detector-free local feature matching with transformers. In *Comput. Vis. Pattern Recog.*, 2021. 6

[37] Paul Swoboda, Ashkan Mokarian, Christian Theobalt, Florian Bernard, et al. A convex relaxation for multi-graph matching. In *Comput. Vis. Pattern Recog.*, pages 11156–11165, 2019. 3

[38] P. Swoboda, C. Rother, H.A. Alhaija, D. Kainmuller, and B. Savchynskyy. A study of lagrangean decompositions and dual ascent solvers for graph matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 6, 7

[39] Lorenzo Torresani, Vladimir Kolmogorov, and Carsten Rother. A dual decomposition approach to feature correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(2):259–271, 2012. 3

[40] R. Tron, X. Zhou, C. Esteves, and K. Daniilidis. Fast multi-image matching via density-based clustering. In *Comput. Vis. Pattern Recog.*, pages 4057–4066, 2017. 3

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. 4, 5

[42] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *stat*, 1050:20, 2017. 5

[43] Fudong Wang, Nan Xue, Jin-Gang Yu, and Gui-Song Xia. Zero-assignment constraint for graph matching with outliers. In *CVPR*, pages 3033–3042, 2020. 3, 6, 7

[44] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Combinatorial learning of robust deep graph matching: an embedding based approach. *Trans. Pattern Anal. Mach. Intell.*, 2020. 2, 3, 4, 5, 6, 7, 11

[45] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Neural graph matching network: Learning lawler's quadratic assignment problem with extension to hypergraph and multiple-graph matching. *Trans. Pattern Anal. Mach. Intell.*, 44(9):5261–5279, 2022. 1, 2, 3, 5, 6, 7, 8

[46] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Unsupervised learning of graph matching with mixture of modes via discrepancy minimization. *Trans. Pattern Anal. Mach. Intell.*, 2023. 11

[47] Runzhong Wang, Tianqi Zhang, Tianshu Yu, Junchi Yan, and Xiaokang Yang. Combinatorial learning of graph edit distance via dynamic embedding. In *Comput. Vis. Pattern Recog.*, pages 5241–5250, 2021. 1

[48] Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. Differentiable top-k with optimal transport. *Advances in Neural Information Processing Systems*, 33:20520–20531, 2020. 2, 3

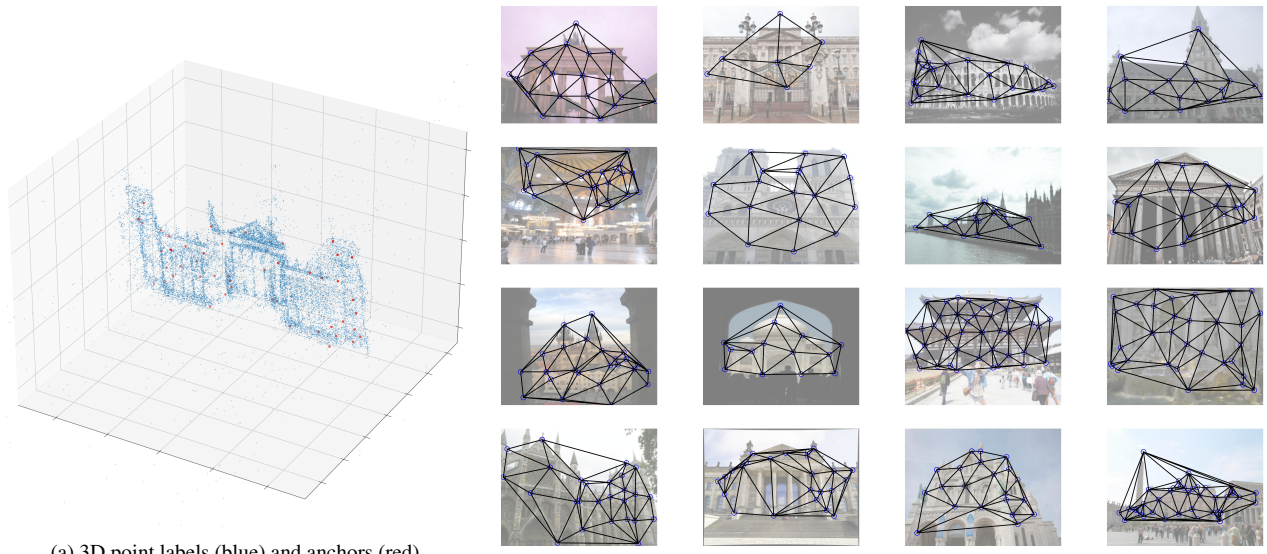
[49] Yu Xiong, Qingqiu Huang, Lingfeng Guo, Hang Zhou, Bolei Zhou, and Dahua Lin. A graph-based framework to bridge movies and synopses. In *Int. Conf. Comput. Vis.*, pages 4592–4601, 2019. 1

[50] Zhitao Ying, Andrew Wang, Jiaxuan You, Chengtao Wen, Arquimedes Canedo, and Jure Leskovec. Neural subgraph matching. *arXiv preprint arXiv:2007.03092*, 2020. 3

[51] Tianshu Yu, Runzhong Wang, Junchi Yan, and Baoxin Li. Learning deep graph matching with channel-independent embedding and hungarian attention. In *Int. Conf. Learn. Rep.*, 2020. 2, 3, 7

[52] A. Zanfir and C. Sminchisescu. Deep learning of graph matching. In *Comput. Vis. Pattern Recog.*, pages 2684–2693, 2018. 1, 2, 3

[53] Kaixuan Zhao, Shikui Tu, and Lei Xu. Ia-gm: A deep bidirectional learning method for graph matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3474–3482, 2021. 1, 2, 3, 7



(a) 3D point labels (blue) and anchors (red)

(b) Examples (raw images with built graphs) for visual graph matching

Figure 6. The 3D points in (a) are detected by colmap [31, 32] which are available as labels in IMC-PT [18]. The blue points denote our selected anchors, based on which our IMC-PT SparseGM-50 is built, as shown in (b). The lines connecting anchors are the edges we build through Delaunay triangulation.

Table 7. Comparison of existing visual graph matching benchmarks.

dataset name	# instances	# classes	avg # nodes	avg # edges	# universe	partial rate	best-known f1
CMU house/hotel	212	2	30	\	30	0.00%	100% (RRWM [7])
Willow ObjectClass	404	5	10	\	10	0.00%	97.8% (GANN [46])
CUB2011	11,788	200	12	\	15	20.00%	83.2% (PCA-GM [44])
Pascal VOC Keypoint	8,702	20	9.07	\	6 to 23	28.50%	62.8% (BBGM-Multi [29])
IMC-PT-SparseGM-50 (ours)	25,765	16	21.36	54.71	50	57.28%	72.9% (ours)
IMC-PT-SparseGM-100 (ours)	25,765	16	44.48	123.99	100	55.52%	71.5% (ours)

A. Details of IMC-PT-SparseGM Benchmark

A.1. Visualization

See Fig. 6a for a visualization of the 3D point cloud built from the collection of Reichstag photos, where most points gather near the main part of the building. The red points are the anchors (50 anchors in this example) that are regarded as the keypoints in our visual graph matching benchmark. These anchor points are then projected back to the 2D images, whereby the visibility of anchors is judged by the method described in the main paper. Examples of images and keypoints from our benchmark are visualized in Fig. 6.

A.2. Details about hyperparameters

We elaborate on the following three insights of our proposed approach to transforming the original IMC-PT image matching dataset to our IMC-PT-SparseGM benchmark for visual graph matching. These insights are omitted in the main paper due to page limitations. Our approach only involves four hyperparameters:

1) To extract some keypoints that can well represent the feature of the original building and to reduce the impact of noise, we set one hyperparameter of the frequency threshold of keypoints existence, screening out keypoints frequently appearing in the sample images.

2) To reduce the complexity of graph matching, we randomly extract a hyperparameter of 50 keypoints from the keypoints we selected before. During the extraction, to maintain a good representation of the original building, we set a hyperparameter of the minimal euclidean distance of two keypoints to ensure that the extracted keypoints are relatively evenly distributed in the main part of the building.

3) For every sample image of a certain building, we intend to check whether the extracted keypoints exist in the image. Using the annotation of the whole keypoints existing in the image, for every selected keypoints, we calculate its minimal euclidean distance between the keypoints in the image and judge its existence in the image based on another minimal euclidean distance hyperparameter threshold which indicates whether the keypoint is close to the present image or not, removing some of the keypoints covered by the exterior scene to some extent.

Table 8. Number of visual graphs in each class of IMC-PT-SparseGM benchmark. * refers to test class.

class name	brandenburg_gate	grand_place_brussels	palace_of_westminster	reichstag*
# visual graphs	1,363	1,083	983	75
class name	taj_mahal	westminster_abbey	buckingham_palace	hagia_sophia_interior
# visual graphs	1,312	1,061	1,676	889
class name	pantheon_exterior	sacre_coeur*	temple_nara_japan	colosseum_exterior
# visual graphs	1,401	1,179	904	2,063
class name	notre_dame_front_facade	prague_old_town_square	st_peters_square*	trevi_fountain
# visual graphs	3,765	2,316	2,504	3,191

Table 9. F1 (%) on SPair-71k (unfiltered setting). Our methods are marked as gray.

GM Network	PMH	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	dog	horse	mbike	person	plant	sheep	train	tv	mean
ZACR	ZACR	32.9	33.3	45.7	24.6	62.0	13.5	36.0	56.2	17.4	47.5	32.7	19.0	40.7	42.7	37.3	34.8	52.5	60.0	38.3
PCA-GM	None	36.5	25.6	48.9	24.7	50.7	29.1	19.2	54.6	30.1	39.1	42.9	34.0	31.3	27.1	70.5	31.1	56.6	75.2	40.4
BBGM	None	42.9	43.8	65.3	34.6	62.6	47.6	25.6	68.0	38.6	62.0	57.8	42.8	44.1	36.0	83.2	45.4	86.7	90.3	54.3
Ngmv2	None	45.4	42.3	61.0	31.2	62.2	53.3	34.2	65.3	37.0	59.5	54.7	41.3	44.8	38.9	77.5	44.2	77.8	89.9	53.4
	Thresholding	50.2	42.9	63.4	29.9	62.1	53.9	34.8	65.7	37.3	62.7	56.1	43.8	45.7	41.8	77.1	45.2	79.0	90.4	54.6±0.5
	Dummy node	47.7	41.6	62.1	30.3	59.0	49.7	27.4	68.3	33.9	62.4	57.3	46.7	46.4	42.7	78.7	43.5	80.5	89.5	53.8±0.4
	AFAT-U(ours)	50.3	43.5	63.8	32.4	59.0	60.1	39.7	68.6	36.1	63.6	56.5	46.3	51.4	43.3	77.0	51.2	81.1	89.4	56.3±0.4
	AFAT-I(ours)	50.4	43.6	63.9	32.1	61.2	58.5	38.0	68.4	35.7	62.7	56.4	47.7	51.9	44.3	78.5	50.7	79.2	91.2	**56.4**±0.6
GCAN	Dummy node	49.0	41.3	64.0	30.3	57.3	55.0	37.4	64.8	36.6	63.0	58.0	44.4	46.4	42.6	68.4	42.3	83.2	91.9	54.2±0.3
	AFAT-U(ours)	46.7	43.3	65.8	33.3	61.5	54.9	35.2	68.4	37.7	59.9	56.0	47.6	47.2	43.5	80.3	47.7	83.8	89.0	**55.7**±0.4
	AFAT-I(ours)	46.8	44.3	65.9	32.4	61.5	53.8	33.7	68.4	38.1	60.1	56.3	47.9	48.3	43.8	81.2	48.4	82.9	88.0	**55.7**±0.4

A.3. More Details

Tbl. 7 shows comparison among our released IMC-PT-SparseGM benchmark and other existing vision graph matching benchmarks. Note that in our released IMC-PT-SparseGM benchmark, the edges are previously built through Delaunay triangulation, thus saving users' time of online graph-building. Tbl. 8 exhibits number of visual graphs (with raw images) in each class of IMC-PT-SparseGM benchmark.

In addition, the anchors are not fixed in IMC-PT-SparseGM, and can be edited via tuning hyperparameters, allowing users to build data that fulfills their own demands. We provide code and instructions for users to build their own data in IMC-PT-SparseGM dataset page: https://github.com/Thinklab-SJTU/IMCPT-SparseGM-dataset.

B. Results on SPair-71k Dataset

To further validate the general effectiveness of our proposed methods, we also perform experiments on SPair-71k (http://cvlab.postech.ac.kr/research/SPair-71k/) dataset. The dataset contains 18 categories of total 70,958 image pairs, including 53,340 for training and 12,234 for testing. The image pairs are different in scale, truncation, and occlusion, whereby outliers are prevalent. We still follow the "unfiltered" setting and show our experimental results in Tbl. 9. Consistent with the experimental results on other datasets, our methods outperform other PMH methods on both GM network embodiments.

Algorithm 2 GreedyTopK

Input: confidence matrix $\mathbf{D}_{conf}$; k; permutation matrix $\mathbf{P}$.
Output: final permutation matrix $\tilde{\mathbf{P}}$.

$\mathbf{D}_{conf} = \mathbf{D}_{conf} \odot \mathbf{P}$; ▷ filter the matching confidence
set $\tilde{\mathbf{P}}$ to all-zero matrix; set $m = 0$; ▷ initialization
while $m < k$ **do**
  $r, c = \text{argmax}\,(\mathbf{D}_{conf})$; ▷ the most confident match
  $\tilde{\mathbf{P}}_{r,c} = 1$; ▷ select this match
  $\mathbf{D}_{conf\,r,c} = 0$; ▷ to select next match
  $m = m + 1$; ▷ count selected matches
end while
return $\tilde{\mathbf{P}}$; ▷ final matching result, for testing

C. GreedyTopK Algorithm for Post-process

In our proposed framework, a GreedyTopK algorithm is adopted in inference stage to greedily select top-k matches based on the confidence from the output of Hungarian algorithm, i.e., the permutation matrix $\mathbf{P}$. Algorithm 2 shows the procedure of GreedyTopK algorithm, where $\odot$ denotes element-wise product.