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].

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
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 Attention-Fused Aggregation (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 plug-in.

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.

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 GCN [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 backpropagation, to discard outliers. 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 backpropagation, to discard outliers.

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.

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

**Input:** matching confidence d; k; regularization factor \(\epsilon\).
1. construct \(D, c, r\) by Eq. 1; \(\Gamma = \exp(-D/\epsilon)\).
2. repeat
3. \(\Gamma = \text{diag}((\Gamma^1 \odot r))^{1/\epsilon} \Gamma\); \(\triangleright \) Sinkhorn optimal transport
4. \(\Gamma = \text{diag}((\Gamma^1 1 \odot c))^{-1} \Gamma\); \(\triangleright \) row norm
5. until \(\Gamma\) is converged
6. \(S_r = \text{Reshape}(\Gamma_{r,2})\); \(\triangleright \) rebuild matching matrix
7. if training then
8. return \(S_r\); \(\triangleright \) differentiable result, for training
9. else
10. return GreedyTopK(Hungarian(\(S_r\)));
11. end if

\(d = [d_1, d_2, \ldots, d_{n_1n_2}]\). To select the top-k matches from \(d\) in a differentiable manner, the OT problem is formulated as moving each element in \(d\) 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 \(r, c\) as the marginal distributions, \(D\) as the distance matrix, \(1\) as an all-one vector, we have:

\[
\begin{align*}
    r &= \Gamma^1_{n_1n_2}, \quad c = [n_1n_2 - k, k]^T, \\
    D &= \begin{bmatrix}
        d_1 - d_{\min} & d_2 - d_{\min} & \cdots & d_{n_1n_2} - d_{\min} \\
        d_{\max} - d_1 & d_{\max} - d_2 & \cdots & d_{\max} - d_{n_1n_2}
    \end{bmatrix}.
\end{align*}
\]

The OT problem is formulated as

\[
\min_{\Gamma} \text{tr}(\Gamma^T D) \quad s.t. \quad \Gamma 1 = r, \Gamma^T 1 = c,
\]

where \(\Gamma \in \{0,1\}^{n_1n_2 \times 2}\) is the transportation matrix, \(\Gamma_{i,1} = 0, \Gamma_{i,2} = 1\) means the \(i\)-th match is selected, \(\Gamma_{i,1} = 1, \Gamma_{i,2} = 0\) 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 \(\Gamma \in [0,1]^{n_1n_2 \times 2}\):

\[
\min_{\Gamma} \text{tr}(\Gamma^T D) + \epsilon h(\Gamma) \quad s.t. \quad \Gamma 1 = r, \Gamma^T 1 = c
\]

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

\[
\Gamma = \text{diag}(\Gamma^1 1 \odot r)^{-1} \Gamma, \quad \Gamma = \text{diag}((\Gamma^T 1 \odot c)^{-1} \Gamma,
\]

where \(\odot\) means element-wise division, \(\text{diag}()\) means building a diagonal matrix from a vector. The 2nd row of converged \(\Gamma\), denoted as \(\Gamma_{:,2}\), are the probabilities that each match should be selected in top-k, and we have \(\sum \Gamma_{:,2} = k\).
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, 34], 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, 34], we adopt intra-graph aggregation layers and cross-graph attention layers. Consider $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, with $|V_1| = n_1$, $|V_2| = n_2$, the intra-graph aggregation updates the feature of node $i$ through:

$$x_i^{l+1} = \xi_{\text{intr}} \left( \frac{1}{|N(i)|} \sum_{j \in N(i)} (\phi_{\text{intr}} (x_j^l)) \right),$$

where $x_i^l$ is the the feature vector of node $i$ in layer $l$, $N(i)$ is the set of neighbors of node $i$, $\xi$, $\phi$, $\zeta$ 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 $i \in V_1$):

$$x_i^{l+1} = \xi_{\text{crs}} \left( \sum_{j \in V_2} (\tilde{M}_{i,j}) \phi_{\text{crs}} (x_j^l)) \right),$$

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

$$A_{i,j} = \exp \left( \frac{x_i^l \cdot W_{\text{affy}} x_j^l}{\tau} \right), i \in V_1, j \in V_2,$$

$$\tilde{M}^l = \text{Sinkhorn}(A^l),$$

where $A \in \mathbb{R}^{n_1 \times n_2}$ is the affinity matrix, representing the similarity of node features from the graph pair. $A_{i,j}$ indicates the affinity score between node $i \in V_1$ and node $j \in V_2$. $W_{\text{affy}} \in \mathbb{R}^p \times p$ contains learnable parameters used for affinity matrix construction, where $p$ is the feature dimension. $\tau$ is a hyperparameter for scaling. $M$ 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 $X_1 \in \mathbb{R}^{n_1 \times p}$, $X_2 \in \mathbb{R}^{n_2 \times p}$ for $G_1$, $G_2$, respectively, where $p$ is the feature dimension. We first perform average pooling to reach $\bar{x}_1 \in \mathbb{R}^p$, $\bar{x}_2 \in \mathbb{R}^p$ (column vectors). The global features are designed to be a weighted summation of node features with attention weights:

$$w_i = \text{Sigmoid} (X_i W_{ap,i} \bar{x}_i), x_{gh, i} = \sum_{n=1}^{n_1} w_{i,n} x_{i,n},$$

where $W_{ap,i} \in \mathbb{R}^p \times p$ contains learnable parameters for $G_i$, $w_{i,n} \in \mathbb{R}$ is the attention weight of $G_i$, $w_{i,n} \in \mathbb{R}$ is the $n$-th element of $w_i$, and $x_{i,n} \in \mathbb{R}^p$ is the $n$-th vector of $X_i$. $x_{gh, i}$ is the global feature of $G_i$.

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 $s$ is computed by:
weights of the multi-head dimension of each query, key, and value vector. In each head of nodes in value pairs to an output. We take the embeddings of nodes cross-set multi-head attention module in transformers, the multi-head is mainly built upon transformers [17], the edge weight matrix $S$ (in the bipartite graph) is concatenated to $w_{ori}$ 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 $w_{final}$ (size $n_1 \times n_2$):

$$w_{final} = \text{softmax} \left( \text{MLP} \left( \begin{bmatrix} w_{ori} \\ S \end{bmatrix} \right) \right)$$

(11)

where $w_{final}$ is multiplied with the embedded $V$ to produce the output in each head: $\text{Attn}(Q, K, V) = w_{final}(VV^T)$. $W^V$ 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 $S$, the initialization of node embeddings does not have a significant impact. In our implementation, following [21], we set node embeddings in $V_1$ to be zero vectors, and in $V_2$ to be one-hot vectors to enhance the effect of $S$ in feature aggregation. Moreover, the positions of $V_1$ and $V_2$ can be swapped in our AFA-U module to produce a new $k$ for double-checking. We build 2 parallel AFA-U modules for $V_1$ and $V_2$ 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.

<table>
<thead>
<tr>
<th>GM Network</th>
<th>PMH</th>
<th>bike</th>
<th>bird</th>
<th>boat</th>
<th>bottle</th>
<th>bus</th>
<th>cat</th>
<th>chair</th>
<th>cow</th>
<th>dog</th>
<th>horse</th>
<th>mbike</th>
<th>person</th>
<th>plant</th>
<th>sheep</th>
<th>sofa</th>
<th>train</th>
<th>tv</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZACR [43]</td>
<td>ZACR [43]</td>
<td>12.2</td>
<td>31.8</td>
<td>31.7</td>
<td>23.0</td>
<td>35.0</td>
<td>28.3</td>
<td>21.8</td>
<td>32.6</td>
<td>19.6</td>
<td>23.8</td>
<td>33.8</td>
<td>29.9</td>
<td>28.8</td>
<td>21.4</td>
<td>10.8</td>
<td>39.0</td>
<td>26.9</td>
<td>15.5</td>
</tr>
<tr>
<td>BBGM [29]</td>
<td>LPPM [38]</td>
<td>35.6</td>
<td>60.3</td>
<td>43.7</td>
<td>34.5</td>
<td>81.5</td>
<td>54.9</td>
<td>30.1</td>
<td>47.8</td>
<td>30.4</td>
<td>46.4</td>
<td>43.9</td>
<td>44.5</td>
<td>52.4</td>
<td>29.4</td>
<td>78.7</td>
<td>40.7</td>
<td>30.4</td>
<td>58.6</td>
</tr>
<tr>
<td>GM Network</td>
<td>PMH</td>
<td>bike</td>
<td>bird</td>
<td>boat</td>
<td>bottle</td>
<td>bus</td>
<td>cat</td>
<td>chair</td>
<td>cow</td>
<td>dog</td>
<td>horse</td>
<td>mbike</td>
<td>person</td>
<td>plant</td>
<td>sheep</td>
<td>sofa</td>
<td>train</td>
<td>tv</td>
<td>mean</td>
</tr>
<tr>
<td>------------</td>
<td>-----</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>--------</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>-------</td>
<td>--------</td>
<td>-------</td>
<td>-------</td>
<td>------</td>
<td>-------</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>Sparse NGMv2 [45]</td>
<td>None</td>
<td>45.5</td>
<td>65.3</td>
<td>55.3</td>
<td>45.8</td>
<td>88.4</td>
<td>64.3</td>
<td>45.9</td>
<td>58.6</td>
<td>43.3</td>
<td>59.1</td>
<td>39.2</td>
<td>55.7</td>
<td>58.0</td>
<td>65.3</td>
<td>44.4</td>
<td>55.4</td>
<td>50.3</td>
<td>41.2</td>
</tr>
<tr>
<td>Dense NGMv2 [45]</td>
<td>Thresholding</td>
<td>48.3</td>
<td>65.4</td>
<td>55.9</td>
<td>48.6</td>
<td>87.6</td>
<td>63.0</td>
<td>51.1</td>
<td>61.1</td>
<td>39.6</td>
<td>63.3</td>
<td>33.6</td>
<td>59.2</td>
<td>59.3</td>
<td>63.4</td>
<td>46.9</td>
<td>55.2</td>
<td>55.1</td>
<td>45.5</td>
</tr>
<tr>
<td>AFAT-I (ours)</td>
<td>45.7</td>
<td>67.7</td>
<td>57.3</td>
<td>44.9</td>
<td>90.1</td>
<td>65.5</td>
<td>49.9</td>
<td>59.3</td>
<td>44.0</td>
<td>62.0</td>
<td>54.9</td>
<td>58.4</td>
<td>58.6</td>
<td>63.8</td>
<td>45.9</td>
<td>59.4</td>
<td>50.9</td>
<td>37.3</td>
<td>74.2</td>
</tr>
<tr>
<td>AFAT-II (ours)</td>
<td>45.0</td>
<td>67.3</td>
<td>55.9</td>
<td>46.5</td>
<td>90.3</td>
<td>64.6</td>
<td>48.7</td>
<td>58.0</td>
<td>44.7</td>
<td>60.2</td>
<td>54.8</td>
<td>57.2</td>
<td>57.5</td>
<td>63.4</td>
<td>45.2</td>
<td>55.3</td>
<td>49.3</td>
<td>41.6</td>
<td>73.6</td>
</tr>
</tbody>
</table>

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 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. Based on IMC-PT, we re-make 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 keypointsp, 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 (0.025δ 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...
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 city. 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 regenerating. 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 $P_{pred}$ containing $k_{pred}$ matches, and ground truth permutation matrix $P_{gt}$ containing $k_{gt}$ matches, denote $\odot$ as element-wise product, matching F1 score is computed as precision $= \sum (P_{pred} \odot P_{gt}) / k_{pred}$, recall $= \sum (P_{pred} \odot P_{gt}) / k_{gt}$, $F1 = 2 \times$ precision-recall / precision + 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 Attention-Fused Aggregation with Top-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, 52] 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>
<thead>
<tr>
<th>Dataset name</th>
<th>Willow Object Class (+random outliers) and IMC-PT-SparseGM (50/100 anchors). Our methods are marked as gray.</th>
<th>IMC-PT-SparseGM (50 anchors)</th>
<th>IMC-PT-SparseGM (100 anchors)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GM Network</td>
<td>PMH</td>
<td>car</td>
<td>duck</td>
</tr>
<tr>
<td>ZACR [43]</td>
<td>None</td>
<td>47.3</td>
<td>44.7</td>
</tr>
<tr>
<td>ZACR [43]</td>
<td>None</td>
<td>55.8</td>
<td>56.5</td>
</tr>
<tr>
<td>BBGM [29]</td>
<td>LPM [38]</td>
<td>None</td>
<td>65.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># outlier (Pascal VOC, intersection)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAGM [28]</td>
<td>70.2</td>
<td>65.9</td>
<td>61.2</td>
<td>57.5</td>
<td>53.8</td>
<td>50.3</td>
</tr>
<tr>
<td>NGMv2 [45] + AFAT-U (ours)</td>
<td>72.2</td>
<td>65.9</td>
<td>61.2</td>
<td>57.5</td>
<td>53.8</td>
<td>50.3</td>
</tr>
</tbody>
</table>

| | Willow Object Class | | IMC-PT-SparseGM (50 anchors) | IMC-PT-SparseGM (100 anchors) |
| | | | | |
| | | | | |
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.

**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 k_{ns} 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


