Proxy Graph Matching with Proximal Matching Networks

Hao-Ru Tan[1, 2], **Chuang Wang**[1, 2] , **Si-Tong Wu**[2],
Tie-Qiang Wang[1, 2], **Xu-Yao Zhang** [1,2],**Cheng-Lin Liu**[1,2, 3] *

[1]National Laboratory of Pattern Recognition, Institute of Automation of Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]CAS Center for Excellence of Brain Science and Intelligence Technology
tanhaoru2018@ia.ac.cn, wusitong98@gmail.com, {chuang.wang, tieqiang.wang, xyz, liucl}@nlpr.ia.ac.cn

Abstract

Estimating feature point correspondence is a common technique in computer vision. A line of recent data-driven approaches utilizing the graph neural networks improved the matching accuracy by a large margin. However, these learning-based methods require a lot of labeled training data, which are expensive to collect. Moreover, we find most methods are sensitive to global transforms, for example, a random rotation. On the contrary, classical geometric approaches are immune to rotational transformation though their performance is generally inferior. To tackle these issues, we propose a new learning-based matching framework, which is designed to be rotationally invariant. The model only takes geometric information as input. It consists of three parts: a graph neural network to generate a high-level local feature, an attention-based module to normalize the rotational transform, and a global feature matching module based on proximal optimization. To justify our approach, we provide a convergence guarantee for the proximal method for graph matching. The overall performance is validated by numerical experiments. In particular, our approach is trained on the synthetic random graphs and then applied to several real-world datasets. The experimental results demonstrate that our method is robust to rotational transform and highlights its strong performance of matching accuracy.

Introduction

Graph matching (GM) is a ubiquitous technique in computer vision and pattern recognition to establish the correspondence between two sets of feature points. Its applications range from object recognition (Brendel and Todorovic 2011), tracking (Hwann-Tzong Chen, Horng-Horng Lin, and Tyng-Luh Liu 2001) and shape matching (Berg, Berg, and Malik 2005). Practitioners encounter two major challenges when deploying GM. First, one has to make considerable efforts to seek a robust feature representation for constructing the GM objective. Second, the graph matching algorithm is often hard to reach satisfiable performance due to its intrinsic nature of NP-hardness.

Traditionally, people consider developing a GM algorithm and finding a good feature extractor as two separated problems. The GM problem is commonly formulated as a quadratic assignment optimization problem (Koopmans and Beckmann 1957; Rainer et al. 1998), which can be solved approximately by adopting a variety of relaxation strategies (Leordeanu and Hebert 2005; Caelli and Caetano 2005; Marius, Martial, and Sukthankar 2009). Meanwhile, the features fed into the GM objective are majorly handcrafted, for example, coordinate positions for geometric features (Zhou and la Torre 2016).

A line of recent works considered the above two tasks for solving feature matching as a joint problem by utilizing the graph neural network (GNN). The general idea is to build an end-to-end model consisting of a feature extractor and a feature matching module. For visual input, the feature extractor is a deep convolutional neural network, *e.g.* VGG (Simonyan and Zisserman 2015), and for geometric features that only involve coordinate information, the higher-order local feature is generated by a graph neural network (Zhang and Lee 2019). The feature matching module can either be the Sinkhorn-Knopp transform of a proper mutual distance of node features (Wang, Yan, and Yang 2019) or a differentiable module of a more involved GM algorithm (Zanfir and Sminchisescu 2018). Since the whole model is differentiable with respect to the model parameters, the model can be trained in an end-to-end fashion. More recently, SuperGlue (Sarlin et al. 2020) proposed an approach based on a combination of the convolutional neural network and graph neural network for the feature matching tasks.

Compared with the traditional methods, the contemporary data-driven approaches improve the matching accuracy by a large margin. However, these learning-based methods usually require a large number of labeled training data, which are expensive to collect. In addition, we found existing methods are highly sensitive to global transform, e.g. rotational transform, raising a generalization issue to many real-world applications, where global calibration is not possible at inference stage (Zhang and Lee 2019). On the contrary, traditional methods with geometric feature input are robust to global rotational transform though the overall performance is not as good as the GNN-based methods. See Figure 1 for a detailed experimental example.

In this work, we proposed a new GNN framework. Our model enjoys the advantages of both the classical method and the GNN based method. Using purely geometric information as input, the performance is on a par with the

*Corresponding author.

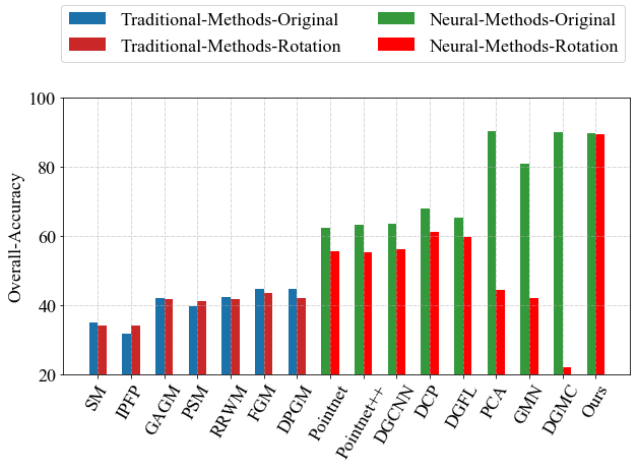

Figure 1: Overall matching accuracy (%) on Willow-Object dataset. For each method, we did two experiments. Left bar shows the accuracy on the original data, and the right bar indicates the accuracy on the rotated data.

state-of-art method. On the other hand, our method is insensitive to global rotational transform. The proposed model consists of three modules: a high-order local feature extractor based on GNN, a differentiable global feature matching module based on the proximal method, and an attention-based calibration module to deal with rotation ambiguity. Specifically, we deploy the same feature extractor as the work used in (Zhang and Lee 2019) to get a local feature vector for each node, where the input is a graph with only 2D coordinate node locations. We implement the global node-wise matching module by using the proximal optimization method, which decomposes the quadratic assignment problem for graph matching into a sequence of convex optimization problems. The rotation calibration module generates a set of candidate graphs by rotating one input graph at different degrees and applies an attention mechanism to encourage the candidate graph with the aligned angle to match the target graph. Our model was trained in an end-to-end fashion by using synthetic random graphs. It can directly generalize to real-life datasets without further fine-tuning on human-labeled data.

In summary, our contributions are threefold.

- We proposed a neural network for geometric 2D graph matching, which is designed to be rotationally invariant. The model only takes keypoints' coordinate information as input, and it uses only synthetic random graphs for training.
- We provide a convergence guarantee for the proximal graph matching algorithm, which acts as a node-wise matching module in the proposed network.
- Numerical results show that the proposed method is on par with the state-of-art geometric method in the standard rotation-aligned setting, and improves the prediction accuracy with a large margin in the rotation-misaligned scenario.

Classical Proximal Graph Matching

In this section, we first introduce the classical graph matching problem, which is formulated as a quadratic assignment problem. Then, we introduce the proximal method for solving this quadratic optimization problem. Finally, we discuss a special matching problem that only considers the local feature as input without taking any graph structure into consideration.

Formulation Let $\mathcal{G}_1 = (V_1, E_1)$ and $\mathcal{G}_2 = (V_2, E_2)$ be two undirected graphs with an equal number of nodes, *i.e.*,$|V_1| = |V_2| = n$, where V_i and E_i for $i = 1, 2$ represents nodes set and edges set of the graph $\mathcal{G}_i$ respectively. For the case $|V_1| \neq |V_2|$, one can add additional auxiliary nodes to meet the equality condition.

Formally, graph matching is formulated as a quadratic assignment problem (Koopmans and Beckmann 1957; Rainer et al. 1998). Let $\boldsymbol{x} \in \{0,1\}^{n^2}$ be an assignment vector such that if node i in $\mathcal{G}_1$ is mapped to node j in $\mathcal{G}_2$, then $x_{ij} = 1$, $x_{ij'} = 0$ and $x_{i'j} = 0$ for all $i' \in V_1 \setminus \{i\}$ and $j' \in V_2 \setminus \{j\}$. The optimal assignment of graph matching is then formulated as

$$\max_{\boldsymbol{x} \in \{0,1\}^{n^2}} \quad \boldsymbol{x}^\top \boldsymbol{M} \boldsymbol{x} \qquad \text{s.t.} \qquad \boldsymbol{C}\boldsymbol{x} = \boldsymbol{1}, \tag{1}$$

where the binary matrix $\boldsymbol{C} \in \{0,1\}^{n^2 \times n^2}$ encodes n^2 linear constraints ensuring that $\sum_{i \in V_1} x_{ij'} = 1$ and $\sum_{j \in V_2} x_{i'j} = 1$ for all $i' \in V_1$ and $j' \in V_2$. The affinity matrix is of size $\boldsymbol{M} \in \mathcal{R}^{n^2 \times n^2}$. The diagonal terms $\boldsymbol{M}_{ii,jj}$ represents the reward if node i in $\mathcal{G}_1$ is mapped to node j in $\mathcal{G}_2$. The off-diagonal terms $\boldsymbol{M}_{ii',jj'}$ represents the reward if pair (i, i') in $\mathcal{G}_1$ corresponds to pair (j, j') in $\mathcal{G}_2$.

Constructing the Affinity Matrix Traditionally, the affinity matrix $\boldsymbol{M}$ is defined manually based on concrete applications. For example, Koopman-Beckmann's formula (Koopmans and Beckmann 1957) constructs the affinity matrix by setting $\boldsymbol{M} = \boldsymbol{A}_1 \otimes \boldsymbol{A}_2$, where $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ are adjacent matrices of $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively. Intuitively, $[\boldsymbol{M}]_{ii',jj'} = 1$ if and only if both pairs of (i, i') and (j, j') are directly connected in $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively, and $[\boldsymbol{M}]_{ii',jj'} = 0$ otherwise. If only adjacent matrix is used, the diagonal terms are all zero.

In many practical situations, we usually have additional prior information on nodes correspondence, such as locations of nodes in an image. Those prior information forms node feature vectors $\boldsymbol{f}_{1,i}, i \in V_1$ and $\boldsymbol{f}_{2,j}, j \in V_2$. One way to construct affinity matrix utilizing this prior information is to set $\boldsymbol{M}$ as

$$[\boldsymbol{M}]_{ii,jj} = \exp(-\|\boldsymbol{f}_{1,i} - \boldsymbol{f}_{2,j}\|^2/\rho), \tag{2}$$

for diagonal terms, and for off-diagonal entries,

$$[\boldsymbol{M}]_{ii',jj'} = \exp[-(d_{ii'} - d_{jj'})^2/\rho] \tag{3}$$

if $(i, i') \in E_1$ and $(j, j') \in E_2$, and $[\boldsymbol{M}]_{ij',i'j} = 0$ otherwise, where $d_{ii'} = \|\boldsymbol{f}_{1,i} - \boldsymbol{f}_{1,i'}\|$ and $d_{jj'} = \|\boldsymbol{f}_{2,j} - \boldsymbol{f}_{2,j'}\|$, and ρ is a decay parameter.

Finally, based on this physical meaning of the affinity matrix, it is convenient to decompose the affinity matrix $\boldsymbol{M}$ into two parts

$$\boldsymbol{M} = \mathrm{Diag}(\boldsymbol{u}) + \boldsymbol{P}, \tag{4}$$

where $\boldsymbol{u}$ is the diagonal vector of $\boldsymbol{M}$, and the matrix $\boldsymbol{P}$ contains all off-diagonal entries of $\boldsymbol{M}$. Then, the original objective (1) is equivalent to the following objective function:

$$\max_{\boldsymbol{x}\in\{0,1\}^{n^2}} \quad \boldsymbol{u}^\top \boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{P}\boldsymbol{x} \qquad \text{s.t.} \qquad \boldsymbol{C}\boldsymbol{x} = \mathbf{1}, \tag{5}$$

Proximal Method We relax the integer optimization problem (5) to a continuous version with an additional entropy regularizer $h(\cdot)$, and obtain

$$\begin{aligned} \min_{\boldsymbol{z}\in\mathcal{R}_+^{n^2\times 1}} \; & -\boldsymbol{u}^\top \boldsymbol{z} - \boldsymbol{z}^\top \boldsymbol{P}\boldsymbol{z} + h(\boldsymbol{z}) \\ \text{s.t.} \quad & \boldsymbol{C}\boldsymbol{z} = \mathbf{1}, \end{aligned} \tag{6}$$

where $\boldsymbol{z}$ is the continuous counterpart of the assignment vector $\boldsymbol{x}$ in (5). The function $h(\boldsymbol{z}) = \boldsymbol{z}^\top \log(\boldsymbol{z})$ is an entropy regularizer. The matrix $\boldsymbol{C}$ is the same as the one in (1), which ensures that the reshaped matrix version of $\boldsymbol{z}$ is an $n \times n$ doubly stochastic matrix.

Following the proximal variational inference method adopted in (Mohammad et al. 2015), we decompose the non-convex objective in (6) into a sequence of convex subproblems

$$\boldsymbol{z}_{t+1} = \underset{\boldsymbol{z}\in\mathcal{Z}}{\mathrm{argmin}} \; \boldsymbol{z}^\top \nabla g(\boldsymbol{z}_t) + h(\boldsymbol{z}) + \frac{1}{\beta_t} D(\boldsymbol{z}, \boldsymbol{z}_t), \tag{7}$$

where $g(\boldsymbol{z}) = -\boldsymbol{u}^\top \boldsymbol{z} - \boldsymbol{z}^\top \boldsymbol{P}\boldsymbol{z}$ and $\mathcal{Z}$ is the set of all $n \times n$ doubly stochastic matrices. The non-negative proximal operator $D(\cdot,\cdot)$ satisfies that $D(\boldsymbol{x},\boldsymbol{y}) = 0$ if and only if $\boldsymbol{x} = \boldsymbol{y}$, $\beta_t > 0$ is a scalar parameter controlling the proximal weight. The proximal function $D(\cdot,\cdot)$ can be understood as a distance function for the geometry of $\mathcal{Z}$. We choose the Kullback-Leibler divergence as the proximal function

$$D(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{x}^\top \log(\boldsymbol{x}) - \boldsymbol{x}^\top \log(\boldsymbol{y}). \tag{8}$$

The KL divergence has a better description of the geometry (Mohammad et al. 2015) of the variable $\boldsymbol{z}$, as z_{ij} can be interpreted as the probability mapping node i in $\mathcal{G}_1$ to node j in $\mathcal{G}_2$.

In (7), the two parts $g(\boldsymbol{z})$ and $h(\boldsymbol{z})$ are treated separately. The first part $g(\boldsymbol{z})$ contains complicated terms that we should use the first-order approximation $\boldsymbol{z}^\top \nabla g(\boldsymbol{z}_t)$, and the second part $h(\boldsymbol{z})$ is a simple convex term such that we can directly solve (7) without any approximation on h. The convex optimization (7) has an analytical solution

$$\widetilde{\boldsymbol{z}}_{t+1} = \exp\Big[\frac{\beta_t}{\beta_t+1}(\boldsymbol{u} + \boldsymbol{P}\boldsymbol{z}_t) + \frac{1}{\beta_t+1}\log(\boldsymbol{z}_t)\Big] \tag{9}$$

$$\boldsymbol{z}_{t+1} = \mathrm{Sinkhorn}(\widetilde{\boldsymbol{z}}_{t+1}), \tag{10}$$

where $\mathrm{Sinkhorn}(\boldsymbol{z})$ is the Sinkhorn-Knopp transform (Richard and Paul 1967) that maps a nonegative matrix of size $n \times n$ to a doubly stochastic matrix. Here, the input and output variables are n^2 vector. When use the Sinkhorn-Knopp transform, we reshape the input (output) as an $n \times n$ matrix (n^2-dimensional vector) respectively.

Algorithm 1 : Differentiable Proximal Graph Matching

Input: node affinity vector $\boldsymbol{u}$, edge affinity matrix $\boldsymbol{P}$, a sequence of stepsize $\{\beta_0, \beta_1, \beta_2, ...\}$, and a maximum iteration T.
Initialization: $\boldsymbol{z}_0 \Longleftarrow \mathrm{Sinkhorn}(\boldsymbol{u})$, $t \Longleftarrow 0$.
for $t = 0$ **to** $T-1$ **do**
  $\widetilde{\boldsymbol{z}}_{t+1} \Longleftarrow \exp\Big[\frac{\beta_t}{1+\beta_t}(\boldsymbol{u} + \boldsymbol{P}\boldsymbol{z}_t) + \frac{1}{1+\beta_t}\log(\boldsymbol{z}_t)\Big]$
  $\boldsymbol{z}_{t+1} \Longleftarrow \mathrm{Sinkhorn}(\widetilde{\boldsymbol{z}}_{t+1})$
end for
Output $\boldsymbol{z}_T$

Matching without Graph Structure In some cases, the node feature $\boldsymbol{f}_i$'s are informative enough to estimate the node correspondence or one just needs a quick and rough estimation. Then, one can neglect the graph structure by setting the off-diagonal terms of the affinity matrix to be zero. The relaxed optimized problem (6) is then reduced to

$$\begin{aligned} \min_{\boldsymbol{z}\in\mathcal{R}_+^{n^2\times 1}} \; & \mathcal{L}(\boldsymbol{z}), \text{ with } \mathcal{L}(\boldsymbol{z}) = -\boldsymbol{u}^\top \boldsymbol{z} + h(\boldsymbol{z}) \\ \text{s.t.} \quad & \boldsymbol{C}\boldsymbol{z} = \mathbf{1}, \end{aligned} \tag{11}$$

where $h(\boldsymbol{z})$ is the entropy regularizer same as the one in (6). The above optimization problem can be solved by one-step Sinkhorn-Knopp transform (Patrini et al. 2018).

$$\mathcal{L}^* = \mathcal{L}(\boldsymbol{z}^*), \qquad \text{with } \boldsymbol{z}^* = \mathrm{Sinkhorn}(\exp(\boldsymbol{u})). \tag{12}$$

In the next section, we use this as a fast method to estimate the global matching score of a candidate graph pair, in which the node-wise affinity μ is set to be

$$\boldsymbol{u}_{ij} = -\|\boldsymbol{f}_{1,i} - \boldsymbol{f}_{2,j}\|^2, \tag{13}$$

with $i \in V_1$ and $j \in V_2$.

We note that the proximal method for solving graph matching was investigated based on a view of the Gromov-Wasserstein distance in (Xu, Luo, and Carin 2019; Xu et al. 2019) in the classical non-learnable setting. The optimization problem (11) is also related to the Wasserstein distance that L^* is an optimal transport distance between two set of feature vectors $\{\boldsymbol{f}_{1,i}\}_{i\in V_1}$ and $\{\boldsymbol{f}_{2,j}\}_{j\in V_2}$ (Titouan et al. 2019). In this work, we apply those two methods as two differentiable modules to build and learn an end-to-end matching network. Details are illustrated in the next section.

Proxy Graph Matching Network

In this section, we introduce our proxy graph matching network. We consider the setting of supervised geometric feature matching with synthetic training data and real-world test dataset. This setting is the same as previous works (Zhang and Lee 2019; Fey et al. 2020), and is particularly useful in many practical situations, where content-features (SIFT/CNN-feature) may not be available, and only geometric information is provided, e.g. 3D/2D point cloud, online-handwriting analysis. Specifically, we focus on the 2D geometric graph matching problem that only uses the graph structure and the keypoints' coordinate information as input for matching.

In particular, we consider the input is a pair of graphs embedded on a 2D surface. Each node i of the graph is associated with a 2D vector containing the node coordinate information. Instead of learning data from labeled training data, we synthesize the training graph pairs with known node correspondence. At the inference stage, given a pair of graphs from a real-word dataset, e.g. Willow, or Pascal PF (see details in the Experiments section), the proxy graph matching network is going to infer the node correspondence according to their topology of the graphs and coordinates information.

The whole learnable network consists of three modules: a local feature extractor, a proximal method based global node-wise matching module, and a global rotation calibration module. In what follows, we will introduce the three parts one by one.

High-order Local Feature Extractor The first module aims to extract higher-order features from low-dimensional vectors that only contain 2-D coordinate information. We implement this module with the same multi-layer graph neural network (GNN) from the work (Zhang and Lee 2019). Formally, the GNN module takes a graph $\mathcal{G} = (V, E, \{\boldsymbol{c}_i\}_{i\in V_1})$ as input, where $\boldsymbol{c}_i$ is the coordinate information of the node i. The GNN outputs a higher dimensional vector $\boldsymbol{f}_i$ associated with each node i

$$\{\boldsymbol{f}_i\}_{i\in V} = \text{GNN}(\mathcal{G}; \theta_{\text{GNN}}),$$

where θ_{GNN} represents the learnable parameters in this GNN module. In addition, one can use also other local GNNs as the backbone, for example, PointNet (Qi et al. 2017), or DGCNN (Wang et al. 2018).

Differentiable Proximal Graph Matching (DPGM)

The second module seeks node-wise correspondence based on the proximal optimization introduced in the previous section.

We use the higher-order feature vectors $\{\boldsymbol{f}_i\}$ outputted by the GNN module to construct the affinity matrix $\boldsymbol{M}$. The concrete construction is introduced in (2) and (3). Then, DPGM take the input of an affinity matrix $\boldsymbol{M}$, and output a continuous version of an assignment matrix $\boldsymbol{z}$. Note that the proximal graph matching algorithm described in the previous section can be considered as a differentiable map from the affinity matrix $\boldsymbol{M}$ to the node assignment probability vector $\boldsymbol{z}$,

$$\boldsymbol{z} = \text{DPGM}(\mathcal{G}_1, \mathcal{G}_2), \tag{14}$$

where $\mathcal{G}_\ell = (V_\ell, E_\ell, \{\boldsymbol{f}_{\ell,i}\}_{i\in V_\ell})$, $\ell = 1, 2$, and the node feature vectors $\{\boldsymbol{f}_{\ell,i}\}_{i\in V_\ell}$ is the output of the previous higher-order feature extractor. It is straightforward to check that the map (14) is differentiable function with respect to the input feature vectors $\{\boldsymbol{f}_{\ell,i}\}$, because both of (9) and (10) are differentiable with respective to its input iterand as well as $\boldsymbol{u}$ and $\boldsymbol{P}$. Therefore, DPGM can be treated as a differentiable module, which is integrated into the whole learnable matching network.

Global Rotational Calibration Module (GRCM) As shown in Figure 1, most learning-based methods suffer performance degradation by a random rotation transform. To tackle this issue, we proposed a calibration module to alleviate this rotational ambiguity.

We proposed an attention-based proxy matching module to learn and calibrate the rotation angle. Specifically, we first sample L angles η_ℓ, $\ell = 1, 2, \ldots, L$ evenly distributed in the range $[-\pi, \pi)$. Then, we generate L candidate graphs $\mathcal{G}_1^{(\ell)}$ by rotating $\mathcal{G}_1$ to η_ℓ degree. In particular, $\mathcal{G}_1^{(\ell)} = (V_1, E_1, \{\boldsymbol{f}_{1,i}^{(\ell)}\}_{i\in V_1})$ is the graph that have the same topology structure as $\mathcal{G}_1$ with rotated feature vectors

$$\{\boldsymbol{f}_{1,i}^{(\ell)}\}_{i\in V_1} = \text{GNN}(\{\boldsymbol{c}_i^{(\ell)}\}_{i\in V_1}, \mathcal{G}_1; \theta_{\text{GNN}}),$$

where $\boldsymbol{c}_i^{(\ell)}$ is the node coordinate vector rotated by η_ℓ degree. We then calculate the global matching score of each candidate pair $(\mathcal{G}_1^{(\ell)}, \mathcal{G}_1)$, in which $\mathcal{G}_1^{(\ell)}$ is acted as a proxy graph of $\mathcal{G}_\infty$ to matching $\mathcal{G}_\in$. The score is computed by solving the optimization problem (11)

$$\widetilde{\alpha}^{(\ell)} = -\text{NodeMatchingLoss}(\{\boldsymbol{f}_{1,i}^{(\ell)}\}_{i\in V_1}, \{\boldsymbol{f}_{2,i}\}_{i\in V_2}), \tag{15}$$

where MatchingLoss is the minimized loss got from (12).The estimation of this global matching score does not take the graph structure as input to boost the inference speed. We also tried a more involved module, *e.g.* using DPGM loss as a matching score. The overall improvement is marginal.

The matching scores of all pairs are then normalized by a softmax function

$$\alpha^{(\ell)} = \frac{\exp(\gamma\widetilde{\alpha}^{(\ell)})}{\sum_{\ell=1}^{L}\exp(\gamma\widetilde{\alpha}^{(\ell)})}, \tag{16}$$

where γ is a hyper-parameter acting as an inverse temperature to control the softness of the softmax function.

Meanwhile, for each rotated candidate, we compute the node-wise matching estimation by DPGM

$$\boldsymbol{z}^{(\ell)} = \text{DPGM}(\mathcal{G}_1^{(\ell)}, \mathcal{G}_2).$$

The overall node-wise matching score is a weighted average of $\{\boldsymbol{z}^{(\ell)}\}$

$$\boldsymbol{z} = \sum_{\ell=1}^{L}\alpha^{(\ell)}\boldsymbol{z}^{(\ell)}.$$

In the training phase, we use the soft-attention, *i.e.,* a finite γ, so that all direction pairs can be used for training. In the inference phase, we use the hard-attention *i.e.,* $\gamma = \infty$. Only the pair with the greatest global matching score is the pass to the next module to compute node-wise matching. The hard-attention drastically decreases the inference time by only run DPGM once. The experiments show that in a relatively wide range [0.1, 5], our model is insensitive to γ. Therefore, we choose the default temperature $\gamma = 1.0$ in the training stage of all experiments.

Loss and Prediction

The output of the proximal graph matching module $[\boldsymbol{z}]_{ij}$ can be interpreted as the probability that node i in $\mathcal{G}_1$ matches

node j in $\mathcal{G}_2$. Thus, it is reasonable to use cross-entropy to model the training loss (Wang, Yan, and Yang 2019)

$$\begin{aligned}\text{loss}(\boldsymbol{z}, \boldsymbol{z}^*) = -\sum_{i\in V_1, j\in V_2} \Big[&[\boldsymbol{z}]^*_{ij} \log[\boldsymbol{z}]_{ij} + \\ &(1-[\boldsymbol{z}^*]_{ij}) \log(1-[\boldsymbol{z}]_{ij})\Big]\end{aligned} \tag{17}$$

where $\boldsymbol{z}^*$ is the true assignment vector with $[\boldsymbol{z}^*]_{ij}$ being 1 if node i in $\mathcal{G}_1$ matches node j in $\mathcal{G}_2$, and $[\boldsymbol{z}^*]_{ij}$ being 0 otherwise.

Combining the above three modules, we get a trainable graph matching model, where the input is a pair of two graphs with each node assigned a 2D coordinate vector, and the true assignment vector $\boldsymbol{z}^*$. The trainable parameters of the whole model are optimized via the stochastic gradient descent where the gradients are automatically computed by a deep learning library (Paszke, Gross, and et al 2017).

Convergence Analysis

In this section, we show that under a mild assumption, the proposed DPGM method converges to a stationary point within a reasonable number of iterations. The main technique in this analysis follows (Emtiyaz et al. 2015), which studied a general variational inference problem using the proximal-gradient method.

Before presenting the main proposition, we first show two technical lemmas. The details of all proof are in the supplementary materials.

Lemma 1. *There exist a constant $\alpha > 0$ such that for all $\boldsymbol{z}_{t+1}$, $\boldsymbol{z}_t$ generated by Eq.*(7)*, we have*

$$(\boldsymbol{z}_{t+1} - \boldsymbol{z}_t)^\top \nabla_{\boldsymbol{z}_{t+1}} D(\boldsymbol{z}_{t+1}, \boldsymbol{z}_t) \geq \alpha \|\boldsymbol{z}_{t+1} - \boldsymbol{z}_t\|^2, \tag{18}$$

where D is the KL-divergence defined in (8)*.*

We remark that the largest valid α satisfying (18) is $\frac{1}{2}$.

Lemma 2. *For any real valued vector $\boldsymbol{g}$ which has the same dimension of $\boldsymbol{z}$, and for any $\beta > 0$, considering the problem*

$$\boldsymbol{z}_{t+1} = \underset{\boldsymbol{C}\boldsymbol{z}=\boldsymbol{I}}{\operatorname{argmin}} : \ \{\boldsymbol{z}^\top \boldsymbol{g} + h(\boldsymbol{z}) + \frac{1}{\beta} D(\boldsymbol{z}, \boldsymbol{z}_t)\}, \tag{19}$$

where $h(\boldsymbol{z}) = \boldsymbol{z}^\top \log(\boldsymbol{z})$ and D is the KL-divergence, then the following inequality holds

$$\boldsymbol{g}^\top (\boldsymbol{z}_t - \boldsymbol{z}_{t+1}) \geq \frac{\alpha}{\beta} ||\boldsymbol{z}_t - \boldsymbol{z}_{t+1}||^2 + [h(\boldsymbol{z}_{t+1}) - h(\boldsymbol{z}_t)].$$

Based on the above two lemmas, we show the main result of our convergence analysis.

Proposition 1. *Let L be the Lipschitz constant of the gradient function $\nabla g(\boldsymbol{z})$, where $g(x)$ is defined in* (7)*, and let α be the constant used in Lemma 1. If we choose a constant step-size $\beta_t = \beta < 2\alpha/L$ for all $t \in \{0, 1, \ldots, T\}$, then*

$$\mathbb{E}_{t\sim \textit{uniform}\{0,1,\ldots,T\}} \|\boldsymbol{z}_{t+1} - \boldsymbol{z}_t\|^2 \leq \frac{C_0}{T(\alpha - L\beta/2)}, \tag{20}$$

where $C_0 = |\mathcal{L}^* - \mathcal{L}(\boldsymbol{z}_0)|$ is the objective gap of (6) *between the initial guess $\mathcal{L}(\boldsymbol{z}_0)$ and the optimal one $\mathcal{L}^*$.*

On the Lipschitz constant L, if we construct $\boldsymbol{M}$ by Koopman-Beckmann's form, then $L = 4|E_1| \cdot |E_2|$, where $|E_1|$ and $|E_2|$ are the numbers of edges of the graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively. This is due to the fact that

$$\|\nabla f(\boldsymbol{z}) - \nabla f(\boldsymbol{y})\| = \|\boldsymbol{M}(\boldsymbol{z} - \boldsymbol{y})\| \leq \|\boldsymbol{M}\| \|(\boldsymbol{z} - \boldsymbol{y})\|,$$

where $L = \|\boldsymbol{M}\| = \|\boldsymbol{A}_1\|_F \cdot \|\boldsymbol{A}_2\|_F = |E_1| \cdot |E_2|$.

Proposition 1 guarantees that DPGM converges to a stationary point within a reasonable number of iterations. In particular, the average difference $\|\boldsymbol{z}_{t+1} - \boldsymbol{z}_t\|^2$ of the iterand converges at a rate $\mathcal{O}(1/T)$.

Experiments

In this section, we conducted a series of experiments to study the performance of our approach. We introduce two categories of peer methods: traditional matching algorithms and GNN-based matching models. For those traditional matching methods, we list several existing widely used methods as baselines in the experiments: graduated assignment graph matching (GAGM)(Gold and Rangarajan 1996), reweighted random walk matching (RRWM) (Minsu, Jungmin, and Mu 2010), probabilistic spectral matching (PSM) (Egozi, Keller, and Guterman 2013), spectral matching (SM), integer projected fixed point method (IPFP) (Marius, Martial, and Sukthankar 2009), factorized graph matching method (FGM) (Zhou and la Torre 2016).

We also compare several deep learning matching approaches. The work DGFL (Zhang and Lee 2019) learns the node-wise feature and the structure information by employing a graph convolutional network. The work DGMC (Fey et al. 2020) proposed a two-stage matching strategy to generate robust matching. We also compare several works that intend to process 3D point cloud for example DCP (Yue and M. 2019), Pointnet (Qi et al. 2017), Pointnet++ (R et al. 2017), DGCNN (Wang et al. 2018). Following the protocol in (Zhang and Lee 2019), we remove the depth input channel from the model and make them suitable for 2D feature points. We further remove the rest except for their feature extraction part and transform them into the Siamese network structure introduced as (Zhang and Lee 2019).

Training and Testing Protocol

For model training, we first generate the reference graph $\mathcal{G}_1$ with 30–60 inliers. The node coordinate vector is uniformly drawn from $[-1, 1]^2$. Each node is set to be connected with its top k-nearest neighbors (k = 8). The query graph $\mathcal{G}_2$ is generated by adding Gaussian noise from $N(0, 0.05)$ to these nodes from $\mathcal{G}_1$. Furthermore, we add 0–20 outliers to each graph, where the outliers are randomly distributed in $[-1, 1]$. Following the training protocol in (Zhang and Lee 2019), the input keypoint features are normalized by subtracting the statistic mean and divided by its statistic variance. We keep the maximum iteration of DPGM to be 5 for both training and testing stages. By following (Zheng et al. 2015), we consider the proximal operator coefficient β as an additional learnable parameter besides the parameters θ_{GNN} in the first local GNN module. Finally, all methods are implemented in the deep learning package PyTorch (Paszke,

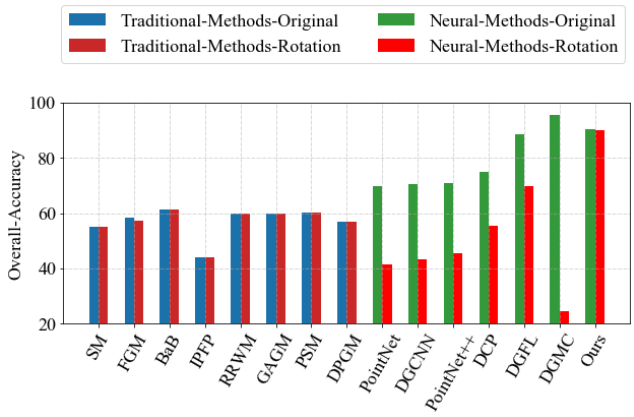

Figure 2: Matching accuracy (%) on PASCAL-PF dataset. Results of Bab is taken from (Zhang and Lee 2019).

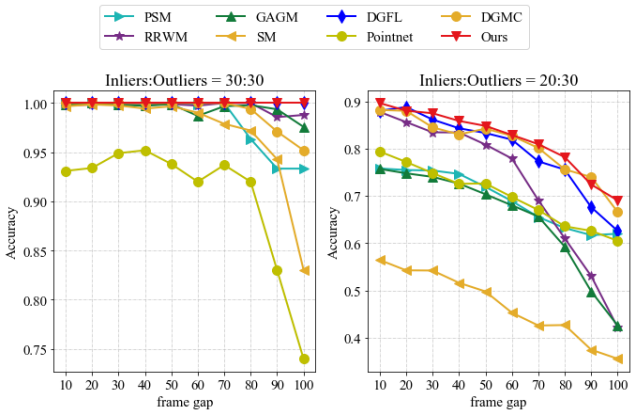

Figure 3: The matching accuracy, robustness on CMU House dataset.

Gross, and et al 2017). The computing platform is Intel CPU E5-2650 v4, 256G RAM with 8 Titian X 12GB GPUs. Only a single GPU is used when we run experiments.

Willow-Object

First, we use the Willow-Object dataset (Cho, Alahari, and Ponce. 2013) to study the performance of those matching approaches. This dataset contains 5 categories (face, duck, and wine-bottle collected from Caltech-256, car, and motorbike from Pascal VOC 2007) and 256 images in total. Each image consists of exactly 10 key points. The goal is to match those key-points with only the keypoint coordinate information. It should be noted that this dataset is often used to test the performance of deep visual matching methods (Zanfir and Sminchisescu 2018; Wang, Yan, and Yang 2019; Yu et al. 2020) which use additional visual keypoint features extracted with a VGG model (Simonyan and Zisserman 2015). Following (Zhang and Lee 2019), we introduce two different experimental settings: (1). directly using the original coordinate as input; (2). applying a random rotation to the keypoint coordinate and use the rotated coordinate as input. The experiment results are shown in Figure 1. We put the detailed quantitive results in the appendix. The results highlight the good performance and generalization ability of our method.

Pascal PF Dataset

We further study the performance on another real-life PASCAL-PF dataset (Cho, Alahari, and Ponce. 2013). The PASCAL-PF dataset consists of 1351 image pairs within a total of 20 classes. There are 6-18 manually annotated ground-truth correspondences in each image pair. Same as the previous experiment, we also introduce two experimental settings: (1) matching with the original keypoint coordinate; (2) matching with the rotated coordinate. We report the overall accuracy in Figure 2 and put the quantitive detailed results in the appendix. The results in Figure 2 demonstrate the leading performance of our approach in the rotated setting. In the standard unrotated setting, our method falls behind the state-of-art method DGMC (90.6 vs. 95.7), but in the rotated setting, our method surpasses DGMC by a large margin (90.2 vs. 24.5). As a comparison, the second-best method DGFL gets 69.9% matching accuracy.

CMU House Sequence

The widely used CMU House Sequence dataset contains 111 images of a house. Each image has 30 labeled landmarks. For all traditional matching algorithm, we build graphs by using these landmarks as nodes and adopting Delaunay triangulation to generate edges. For all deep learning matching approaches, we normalize the keypoint coordinates by subtracting the statistic mean and dividing by its statistic variance. We report the final average matching accuracy in Figure 3 under the frame gap setting of 10:10:100 frames. The results show that our method can keep a relatively good performance even when the number of inliers is small, and the frame gap is large.

Speed Test

We establish an experiment to test the inference time cost. The computational complexity of our model mainly depends on two aspects. First, the DPGM need to solve a quadratic assignment with the complexity of $\mathcal{O}(|V_1|^2|V_2|^2)$. Second, the angle calibration module need to calculate the similarity between the reference graph and a set of candidate graphs. This step requires a time cost of $\mathcal{O}(C|V_1||V_2|)$, where C is the number of those candidates. In our experiment, we set $C = 10$. As a comparison, the neural network method only with the local GNN module has the complexity of $\mathcal{O}(|V_1||V_2|)$). In practice, our overall time consumption is still at an acceptable level. Details are shown in Figure 4.

Ablation Study

In this section, we conduct several groups of ablation experiments to study the effect of design choices on the PASCAL-PF dataset. The results from Table.1 suggest that the DPGM could promote the matching performance under the original experimental setting and the GRCM significantly improves

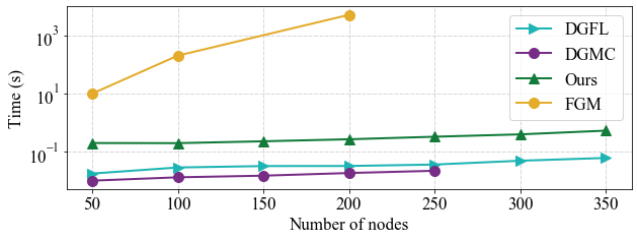

Figure 4: The inference time consumption under graph scales.

GNN	DPGM	GRCM	Ori-Acc	Rot-Acc
DGFL	×	×	88.5	69.9
DGFL	✓	×	**93.5** ↑	70.6 ↑
DGFL	×	✓	80.7 ↓	**80.6** ↑
DGFL	✓	✓	**90.6** ↑	**90.0** ↑

Table 1: Ablation study (%) on PASCAl-PF dataset.

the robustness to random rotations. However, the GRCM has some negative effects on the accuracy of the system for the unrotated data.

The GRCM The previous ablation study indicates that the GRCM module does not always improve performance. This may due to two possible reasons. The first reason is the insufficient amount of the candidate graphs, and the second reason is interference from other candidate graphs to the truly aligned graph. To verify this assumption, we establish an experiment by increase the number of candidates C. We also run an oracle model, where the model also chooses the closest aligned graph from the candidate set with $C = 1000$. We find that as the number of candidates C increases, the negative effect from the GRCM module could be compensated at a cost to further increase time complexity. However, we also notice that there is a small performance gap from the oracle-candidate model. This is due to an inaccurate estimation of the candidate graph.

Differentiability matters The proposed model consists of three modules. Only the first backbone network (Local-GNN) for the local feature extractor contains lots of parameters that need to learn. One can either train this model independently as DGFL did in their work or jointly train the whole model by combining the other two modules. We conduct experiments on the PASCAL-PF dataset to show the advantage of the joint training. In the right figure of Fig.5, the results suggest that jointly training improves the matching performance of the backbone+DPGM system in a large margin and further promotes the robustness. This experiment shows that designing a better differentiable matching module and jointly train the whole network can learn a better backbone network, which further benefits the overall performance.

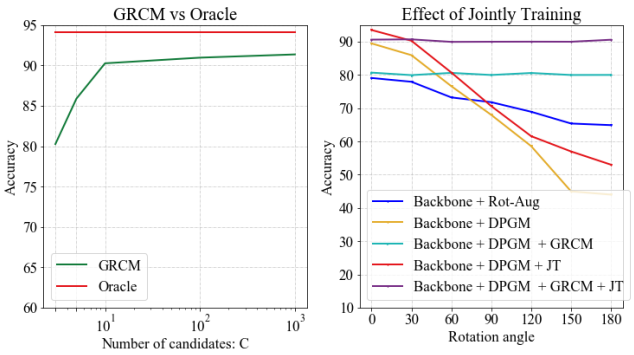

Figure 5: Left: by increasing the number of candidates, the performance of GRCM could get closer to the Oracle bound. Right: The effect of jointly training (JT).

Conclusion

This paper presented a novel end-to-end approach for robust geometric points matching. Our method is designed to be rotational invariant. The proposed model consists of three parts: a graph neural network to generate a high-level local feature, an attention-based module to normalize the rotational transform, and a global feature matching module based on proximal optimization. In the rotation calibration model, we use a proxy graph to replace the original graph, which is aligned with the target graph's rotation. Thus, the proposed method can significantly enhance the robustness of the matching results. Besides, we deploy a differentiable proximal module for node-wise matching with theoretical convergence analysis. Extensive experimental results show the satisfactory performance and robustness of our approach.

Acknowledgments

This work has been supported by the Major Project for New Generation of AI under Grant No. 2018AAA0100400, the National Natural Science Foundation of China (NSFC) grants U20A20223, 61836014, 61721004, the Youth Innovation Promotion Association of CAS under Grant 2019141, and the Pioneer Hundred Talents Program of CAS under Grant Y9S9MS08 and Y9S9MS08.

References

Berg, A. C.; Berg, T.; and Malik, J. 2005. Shape matching and object recognition using low distortion correspondences. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Brendel, W.; and Todorovic, S. 2011. Learning spatio-temporal graphs of human activities. In *International Conference on Computer Vision*.

Caelli, T.; and Caetano, T. 2005. Graphical models for graph matching: Approximate models and optimal algorithms. *Pattern Recognition Letters* 26(3): 339 – 346.

Cho, M.; Alahari, K.; and Ponce., J. 2013. Learning graphs to match. *In CVPR* .

Egozi, A.; Keller, Y.; and Guterman, H. 2013. A Probabilistic Approach to Spectral Graph Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(1): 18–27.

Emtiyaz, K. M.; Pierre, B.; François, F.; and Pascal, F. 2015. Kullback-Leibler Proximal Variational Inference. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28*, 3402–3410. Curran Associates, Inc.

Fey, M.; JanE.Lenssen; Morris, C.; Masci, J.; and Kriege, N. 2020. Deep graph matching consensus. *In ICLR* .

Gold, S.; and Rangarajan, A. 1996. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18(4): 377–388.

Hwann-Tzong Chen; Horng-Horng Lin; and Tyng-Luh Liu. 2001. Multi-object tracking using dynamical graph matching. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*.

Koopmans, T. C.; and Beckmann, M. 1957. Assignment Problems and the Location of Economic Activities. *Econometrica* 25(1): 53–76.

Leordeanu, M.; and Hebert, M. 2005. A spectral technique for correspondence problems using pairwise constraints. In *IEEE International Conference on Computer Vision*.

Marius, L.; Martial, H.; and Sukthankar, R. 2009. An Integer Projected Fixed Point Method for Graph Matching and MAP Inference. In Bengio, Y.; Schuurmans, D.; Lafferty, J. D.; Williams, C. K. I.; and Culotta, A., eds., *Advances in Neural Information Processing Systems*.

Minsu, C.; Jungmin, L.; and Mu, L. K. 2010. Reweighted Random Walks for Graph Matching. In Daniilidis, K.; Maragos, P.; and Paragios, N., eds., *ECCV*.

Mohammad, K.; Pierre, B.; Francois, F.; and Pascal, F. 2015. Kullback-Leibler Proximal Variational Inference. In *Advances in neural information processing systems*.

Paszke, A.; Gross, S.; and et al. 2017. Automatic differentiation in pytorch. In *NIPS Workshop*.

Patrini, G.; Carioni, M.; Forre, P.; Bhargav, S.; Welling, M.; van den Berg, R.; Genewein, T.; and Nielsen, F. 2018. Sinkhorn AutoEncoders. *CoRR* abs/1810.01118.

Qi, C.; Su, H.; Mo, K.; and Guibas, L. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* .

R, Q.; Li, Y.; Hao, S.; and Leonidas, G. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *arXiv preprint arXiv:1706.02413* .

Rainer, B.; Eranda, C.; Panos, P.; Leonidas, P.; and Panos, P. 1998. *The Quadratic Assignment Problem*. INFORMS.

Richard, S.; and Paul, K. 1967. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific J. Math.* 21(2): 343–348.

Sarlin, P. E.; DeTone, D.; Malisiewicz, T.; and Rabinovich, A. 2020. SuperGlue: Learning Feature Matching With Graph Neural Networks. In *CVPR*.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

Titouan, V.; Courty, N.; Tavenard, R.; Laetitia, C.; and Flamary, R. 2019. Optimal Transport for structured data with application on graphs. In *roceedings of the 36th International Conference on Machine Learning*.

Wang, R.; Yan, J.; and Yang, X. 2019. Learning Combinatorial Embedding Networks for Deep Graph Matching. In *The IEEE International Conference on Computer Vision*.

Wang, Y.; Sun, Y.; Liu, Z.; Bronstein, M. M.; ; and Solomon., J. M. 2018. Dynamic graph cnn for learning on point clouds. In *arXiv preprint arXiv:1801.07829*.

Xu, H.; Luo, D.; and Carin, L. 2019. Scalable gromov-Wasserstein learning for graph partitioning and matching. In *Advances in neural information processing systems*.

Xu, H.; Luo, D.; Zha, H.; and Carin, L. 2019. Gromov-Wasserstein Learning for Graph Matching and Node Embedding. In *ICML*.

Yu, T.; Wang, R.; Yan, J.; and Li., B. 2020. Learning deep graph matching with channel-independent embedding and Hungarian attention. *In ICLR* .

Yue, W.; and M., S. J. 2019. Deep Closest Point: Learning Representations for Point Cloud Registration. In *The IEEE International Conference on Computer Vision*.

Zanfir, A.; and Sminchisescu, C. 2018. Deep Learning of Graph Matching. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Zhang, Z.; and Lee, W. S. 2019. Deep Graphical Feature Learning for the Feature Matching Problem. *In Proceedings of the IEEE International Conference on Computer Vision* .

Zheng, S.; Jayasumana, S.; Romera-Paredes, B.; Vineet, V.; Su, Z.; Du, D.; Huang, C.; and Torr, P. H. S. 2015. Conditional Random Fields as Recurrent Neural Networks. In *International Conference on Computer Vision*.

Zhou, F.; and la Torre, D. 2016. Factorized Graph Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence* .