

A DETAILS FOR OTGM

How to approximate the transport plan T can refer to Algorithm 1.

Algorithm 1 Computing OT Distance

Require: $\{x_i\}_{i=1}^n, \{y_j\}_{j=1}^m$, hyper-parameter λ

- 1: Compute intra-graph similarities:
- 2: $[C_x]_{ij} = \cos(x_i, x_j), [C_y]_{ij} = \cos(y_i, y_j)$,
- 3: $x'_i = g_1(x_i), y'_j = g_2(y_j)$ // g_1, g_2 denote two MLPs
- 4: Compute cross-graph similarities:
- 5: $C_{ij} = \cos(x'_i, y'_j)$
- 6: **if** T is shared **then**
- 7: Update \mathcal{L} in Algorithm 3 (Line 8) with:
- 8: $\mathcal{L}_{\text{unified}} = \lambda C + (1 - \lambda)\mathcal{L}$
- 9: Obtain T by calculating Eq.(1).
- 10: Compute D_{got}
- 11: **else**
- 12: Apply Algorithm 2 to obtain D_w
- 13: Apply Algorithm 3 to obtain D_{gw}
- 14: $D_{\text{OT}} = \lambda D_w + (1 - \lambda)D_{gw}$
- 15: **end if**
- 16: **return** D_{OT}

Algorithm 2 Computing Wasserstein Distance

Require: $\{x_i\}_{i=1}^n, \{y_j\}_{j=1}^n, \beta$

- 1: $\sigma \leftarrow \frac{1}{n}\mathbf{1}_n, T^{(1)} \leftarrow \mathbf{1}\mathbf{1}^\top$
- 2: $C_{ij} \leftarrow c(x_i, y_j), A_{ij} \leftarrow e^{-\frac{C_{ij}}{\beta}}$
- 3: **for** $t = 1, 2, 3, \dots$ **do**
- 4: $Q \leftarrow A \odot T^{(t)}$ // \odot is Hadamard product
- 5: **for** $k = 1, 2, 3, \dots, K$ **do**
- 6: $\delta \leftarrow \frac{1}{n\sigma}, \sigma \leftarrow \frac{1}{nQ^\top \delta}$
- 7: **end for**
- 8: $T^{(t+1)} \leftarrow \text{diag}(\delta)Q\text{diag}(\sigma)$
- 9: **end for**
- 10: $D_w \leftarrow \langle C^\top, T \rangle$ // $\langle \cdot, \cdot \rangle$ is the Frobenius dot-product
- 11: **return** T, D_w

Algorithm 3 Computing Gromov-Wasserstein Distance

Require: $\{x_i\}_{i=1}^n, \{y_j\}_{j=1}^n$, probability vectors p, q

- 1: Compute intra-domain similarities:
- 2: $[C_x]_{ij} = \cos(x_i, x_j), [C_y]_{ij} = \cos(y_i, y_j)$,
- 3: Compute cross-domain similarities:
- 4: $C_{xy} = C_x^2 \mathbf{1}_m + C_y q (C_y^2)^\top$
- 5: **for** $t = 1, 2, 3, \dots$ **do**
- 6: // Compute the pseudo-cost matrix
- 7: $\mathcal{L} = C_{xy} - 2C_x^\top T C_y$
- 8: Apply Algorithm 1 to solve transport plan T
- 9: **end for**
- 10: $D_{gw} = \langle \mathcal{L}, T \rangle$
- 11: **return** T, D_{gw}

B PROOF FOR THEOREM 1

Theorem 1 Consider a sample of N_A labeled instances drawn from $\hat{\mathcal{P}}_A$ and N_B instances drawn from $\hat{\mathcal{P}}_B$, and then for all $\lambda > 0$, with $a = k\lambda$, we have with probability at least $1 - \delta$ for graph matching as follows:

$$\begin{aligned} \text{err}_B(f) &< W_1(\hat{\mathcal{P}}_A, \hat{\mathcal{P}}_B) + \sqrt{\frac{2}{c} \log\left(\frac{1}{\delta}\right) \left(\frac{1}{\sqrt{N_A}} + \frac{1}{\sqrt{N_B}}\right)} \\ &\quad + \text{err}_A(f^*) + \text{err}_B(f^*) + k\mathcal{L}_1\phi(\lambda). \end{aligned} \quad (13)$$

where N_A, N_B are the number of nodes in graphs \mathcal{G}_A and \mathcal{G}_B , respectively. c is a constant.

Proof 1 First of all, we have:

$$\begin{aligned} \text{err}_B(f) &= \mathbb{E}_{(x,y) \sim P_B} \mathcal{L}(y, f(x)) \\ &\leq \mathbb{E}_{(x,y) \sim P_B} [\mathcal{L}(y, f^*(x)) + \mathcal{L}(f^*(x), f(x))] \\ &\stackrel{(a)}{=} \mathbb{E}_{(x,y) \sim P_B} [L(f(x), f^*(x))] + \text{err}_B(f^*) \end{aligned} \quad (14)$$

$$\stackrel{(b)}{=} \mathbb{E}_{(x,y) \sim \mathcal{P}_B^f} \mathcal{L}(f(x), f^*(x)) + \text{err}_T(f^*) \quad (15)$$

$$= \text{err}_B(f^*) - \text{err}_A(f^*) + \text{err}_A(f^*) + \text{err}_B(f^*) \quad (16)$$

$$\leq |\text{err}_B(f^*) - \text{err}_A(f^*)| + \text{err}_A(f^*) + \text{err}_B(f^*) \quad (17)$$

Line (a) is due to the symmetry of the loss. Line (b) comes from the fact that: $\mathbb{E}_{(x,y) \sim P_B} \mathcal{L}(f(x), f^*(x)) = \mathbb{E}_{(x,f(x)) \sim P_B} \mathcal{L}(f(x), f^*(x)) \stackrel{\text{def}}{=} \text{err}_B(f^*(x))$.

Now, we have

$$\begin{aligned} &|\text{err}_B(f^*) - \text{err}_A(f^*)| \\ &= \left| \int_{\mathcal{X} \times \mathcal{C}} \mathcal{L}(y, f^*(x)) (P_B(X = x, Y = y) - P_A(X = x, Y = y)) dx dy \right| \\ &= \left| \int_{\mathcal{X} \times \mathcal{C}} \mathcal{L}(y, f^*(x)) d(P_B - P_A) \right| \\ &\stackrel{(a)}{\leq} \int_{(\mathcal{X} \times \mathcal{C})^2} |\mathcal{L}(y_B, f^*(x_B)) - \mathcal{L}(y_A, f^*(x_A))| d\Pi^*((x_A, y_A), (x_B, y_B)) \\ &= \int_{(\mathcal{X} \times \mathcal{C})^2} (\mathcal{L}(y_B, f^*(x_B)) - \mathcal{L}(y_A, f^*(x_A))) d\Pi^*((x_A, y_A), (x_B, y_B)) \quad (18) \\ &\stackrel{(b)}{\leq} \int_{(\mathcal{X} \times \mathcal{C})^2} k|f^*(x_B) - f^*(x_A)| (\mathcal{L}(y_B, f^*(x_A)) - \mathcal{L}(y_A, f^*(x_A))) \\ &\quad d\Pi^*((x_A, y_A), (x_B, y_B)) \\ &\stackrel{(a)}{\leq} k \cdot M \cdot \phi(\lambda) + \int_{(\mathcal{X} \times \mathcal{C})^2} k \cdot d(x_B, x_A) (\mathcal{L}(y_B, f^*(x_A)) \\ &\quad - \mathcal{L}(y_A, f^*(x_A))) d\Pi^*((x_A, y_A), (x_B, y_B)) \\ &\stackrel{(a)}{=} W_1(\mathcal{P}_A, \mathcal{P}_B) + k \cdot M \cdot \phi(\lambda). \end{aligned}$$

Line (a) is a consequence of the duality form of the Kantorovich-Rubinstein theorem saying that for any coupling $\Pi \in \Pi(\mathcal{P}_A, \mathcal{P}_B)$, we have:

$$\begin{aligned}
& \left\| \int_{\Omega \times \mathcal{C}} \mathcal{L}(y, f^*(x)) d(\mathcal{P}_B - \mathcal{P}_A) \right\| \\
&= \left\| \int_{(\Omega \times \mathcal{C})^2} [\mathcal{L}(y_B, f^*(x_B)) - \mathcal{L}(y_A, f^*(x_A))] \right. \\
&\quad \left. d\Pi((x_A, y_A), (x_B, y_B)) \right\| \\
&\leq \left\| \int_{(\Omega \times \mathcal{C})^2} [\mathcal{L}(y_B, f^*(x_B)) - L(y_A, f^*(x_A))] \right\| \\
&\quad d\Pi((x_A, y_A), (x_B, y_B)).
\end{aligned} \tag{19}$$

Since the inequality is true for any coupling, it is then also true for Π^* . Inequality (b) is due to the k -lipschitzness of the loss \mathcal{L} in its second argument. Inequality (c) uses the fact that f^* and Π^* verify the probabilistic transfer Lipschitzness property with probability $1 - \phi(\lambda)$, additionally, taking into account that the deviation between 2 instances with respect to f^* is bounded by M we have the additional term $kM\phi(\lambda)$ that covers the regions where the PTL does not hold. (c) is obtained by the symmetry of d , the use of triangle inequality on \mathcal{L} and by replacing $k\lambda$ by α . Other inequalities above are due the use of triangle inequality or properties of the absolute value. The last line (d) is due to the definition of Π^* .

Now, note that by the use of triangle inequality:

$$\begin{aligned}
& W_1(\mathcal{P}_A, \mathcal{P}'_B) \\
&\leq W_1(\mathcal{P}_A, \hat{\mathcal{P}}_s) + W_1(\hat{\mathcal{P}}_s, \hat{\mathcal{P}}'_t) + W_1(\hat{\mathcal{P}}'_t, \mathcal{P}'_B) \\
&\leq W_1(\hat{\mathcal{P}}_s, \hat{\mathcal{P}}'_t) + \sqrt{\frac{2}{c'} \log\left(\frac{2}{\delta} \left(\frac{1}{\sqrt{N_s}} + \frac{1}{\sqrt{N_t}}\right)\right)}
\end{aligned} \tag{20}$$

Indeed, the cost function

$$D((x_A, y_A), (x_B, y_B)) = \alpha d(x_1, x_2) + L(y_1, y_2)$$

defines a distance over $(\Omega \times \mathcal{C})^2$. Given that \mathcal{P}_A and \mathcal{P}'_B have bounded support and considering the bounded nature of our loss function, we can apply Theorem E.1 (presented below) to $W_1(\mathcal{P}_A, \hat{\mathcal{P}}_s)$ and $W_1(\hat{\mathcal{P}}'_t, \mathcal{P}'_B)$ (with a probability of $\delta/2$ each). It is important to note that the two settings may involve different constants N and c' . In such cases, we take the maximum value of N and the minimum value of c' that are applicable to both scenarios.

In addition to the analysis presented in the paper, a notable connection can be drawn with classic generalization bounds in the scenario where the two distributions are identical, i.e., $\mathcal{P}_A = \mathcal{P}_B$. Specifically, if we can select f^* as the true labeling function on both source and target domains such that f^* is strong ϕ -Lipschitz with respect to Π^* (meaning $\phi(\lambda)$ is almost 0), then the bound aligns with a classic generalization bound:

$$\begin{aligned}
err_B(f) &< W_1(\hat{\mathcal{P}}_A, \hat{\mathcal{P}}_B) + \sqrt{\frac{2}{c} \log\left(\frac{1}{\delta}\right) \left(\frac{1}{\sqrt{N_A}} + \frac{1}{\sqrt{N_B}}\right)} \\
&\quad + err_A(f^*) + err_B(f^*) + k\mathcal{L}_1\phi(\lambda).
\end{aligned}$$

- In this case, terms involving f^* become negligible, and using the same sample for the source and target results in $d(x_1, x_2) = 0$ with respect to the optimal alignment.
- Consequently, only the label loss remains, which aligns with a classic supervised learning loss.

This observation implies that under these conditions, our approach mirrors the familiar terrain of classic generalization in supervised learning.

Table 4: Keypoint matching accuracy (%) on SPair-71k grouped by levels of difficulty in the viewpoint of the matching pair.

Method	Easy	Medium	Hard	All
BBGM	84.7	78.9	73.6	82.1
ASAR	86.5	79.1	72.5	83.1
COMMON	86.6	81.4	76.4	84.5
OTGM	86.9	82.1	78.0	85.2

C DETAILS FOR EXPERIMENTS

Implementation details. Our method is implemented using PyTorch 1.10.0 and all evaluations are conducted on an Ubuntu 20.04 OS with an NVIDIA 3090 GPU. To ensure consistency and fairness, we use the exact same set of hyperparameters for all datasets. The encoder network in our implementation consists of an ImageNet-pretrained VGG16 Simonyan & Zisserman (2014) image encoder, a graph neural network called SplineCNN Fey et al. (2018), and a two-layer projection head Chen et al. (2020b). For more detailed network architecture information, please refer to the supplementary material. To optimize the networks during training, we utilize the Adam optimizer Kingma & Ba (2014) with default parameters. The initial learning rate is set to $3e-4$, and for fine-tuning the VGG network, the learning rate is set to $2e-5$. The batch size for training is set to 8 image pairs. To obtain the permutation matrix Y , we apply the Hungarian algorithm to the similarity matrix S obtained from the base encoder, following the approach outlined in Wang et al. (2021); Liu et al. (2021a); Wang et al. (2019; 2020b); Ren et al. (2022); Yu et al. (2021). These implementation details ensure consistency and reproducibility in our experiments, allowing for a fair comparison with existing methods.

Evaluation with Different Viewpoint Difficulty. The SPair-71k dataset provides a valuable opportunity to evaluate the performance of our method under varying levels of viewpoint difficulty. The dataset categorizes image pairs into easy, medium, and hard groups, with each group representing different degrees of viewpoint variation. In practice, we have observed that image pairs with higher viewpoint difficulty tend to exhibit more instances of noisy correspondence, such as occlusions. This implies that as the viewpoint difficulty increases, the occurrence of noisy correspondence becomes more prevalent. Table 4 presents the results of our method’s performance on different levels of viewpoint difficulty. Our proposed method consistently improves the matching results across all difficulty levels, with a notable improvement in the high viewpoint difficulty group (+1.6%). This experiment highlights the robustness of our method in handling and effectively addressing the challenges posed by noisy correspondences, particularly in image pairs with high viewpoint difficulty. By demonstrating significant improvements in these challenging scenarios, our method showcases its ability to tackle noisy correspondence and enhance overall matching performance.

Distribution of similarity scores. We investigate the similarity scores of correspondences under a noise rate $\eta = 0.3$, as demonstrated in Figs. 5. These figures depict histograms for both true pairs (free of synthetic noise) and noisy correspondences (containing synthetic noise). Our methodology effectively mitigates the influence of noisy data on network optimization, thereby reducing its adverse effects.

Running Time. The running time of the proposed algorithm depends on the size of the input graphs and the complexity of the matching scenario. In our experiments, the OTGM framework, implemented using PyTorch and evaluated on an NVIDIA RTX 3090 GPU, performs competitively with existing methods while providing improved robustness and accuracy. Specifically, we take four hours on pascal VOC dataset and 14 hours on spair-71 dataset and two hours on Willow Object.

Discussion about other graph denoiser. We notice that there is also some research about the graph denoisers such as SuperGlue Sarlin et al. (2020a), COMMON Lin et al. (2023), and LightGlue Lindenberger et al. (2023). Here we discuss the difference between our method and them. SuperGlue is inefficient in compilation and LightGlue may be heavily reliant on the quality of the local features extracted. If the initial feature detection and description are suboptimal, it could lead to a less effective matching process. For the COMMON, the momentum distillation is also predicated on the quality of initial annotations, which, if poor, could compromise the model’s ability to discern correct from noisy correspondences. Additionally, the inclusion of a momentum encoder increases the model’s

Table 5: Parameter analysis of OTGM with the increase of the distillation parameter β on Pascal VOC.

β	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
ACC	82.7	82.9	83.5	83.8	83.7	83.6	83.4	83.1	82.9	82.8	82.6

complexity and computational demands, potentially limiting its applicability in resource-constrained environments or real-time scenarios. Different from them, our model can perform graph denoising in a self-supervised manner with fast computation, and can also be used as a plug-in for other models without incurring excessive computational overhead.

D BROADER IMPACTS

The OTGM model significantly advances graph matching techniques by addressing distributional alignment issues and enhancing applications in network security, social network analysis, and computer vision. Its capability to refine and denoise graph data also holds the potential for improving data integrity in scientific and commercial analyses. However, the sophistication of OTGM could lead to potential risks, such as over-reliance on automated decision-making in sensitive areas, necessitating further research into safeguards and efficiency optimizations for larger datasets to ensure responsible use and broader applicability.

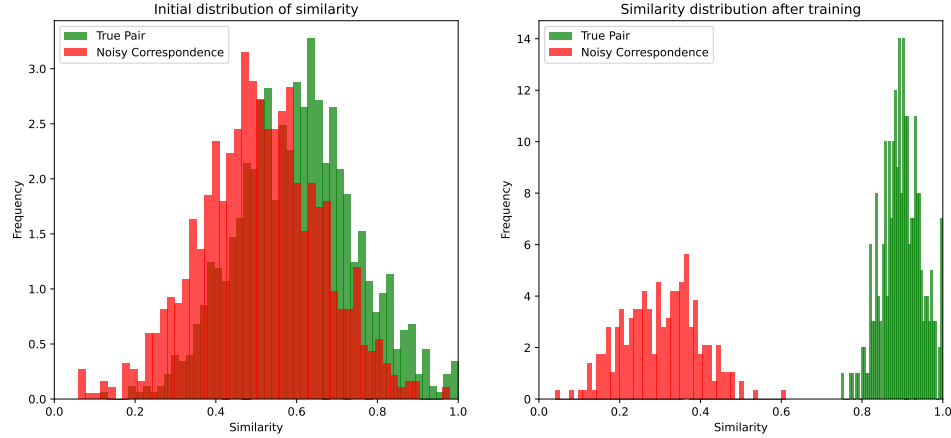


Figure 5: The similarity distribution. The left is the initial distribution of similarity while the right is the similarity distribution after denoising by our method.