Discrete Cycle-Consistency Based Unsupervised Deep Graph Matching

Siddharth Tourani[1,2], **Muhammad Haris Khan**[2], **Carsten Rother**[1], **Bogdan Savchynskyy**[1]

[1]Computer Vision and Learning Lab, IWR, Heidelberg University,
[2]MBZUAI

Abstract

We contribute to the sparsely populated area of unsupervised deep graph matching with application to keypoint matching in images. Contrary to the standard *supervised* approach, our method does not require ground truth correspondences between keypoint pairs. Instead, it is *self-supervised* by enforcing consistency of matchings between images of the same object category. As the matching and the consistency loss are discrete, their derivatives cannot be straightforwardly used for learning. We address this issue in a principled way by building our method upon the recent results on black-box differentiation of combinatorial solvers. This makes our method exceptionally flexible, as it is compatible with arbitrary network architectures and combinatorial solvers. Our experimental evaluation suggests that our technique sets a new state-of-the-art for unsupervised deep graph matching.

1 Introduction

Graph matching (GM) is an important research topic in machine learning, computer vision, and related areas. It aims at finding an optimal node correspondence between graph-structured data. It can be applied in tasks like shape matching (Sahillioğlu 2020), activity recognition (Brendel and Todorovic 2011), point cloud registration (Fu et al. 2021), and many others. One classical application of graph matching also considered in our work is keypoint matching, as illustrated in Figure 1.

A modern, learning-based approach to this problem tries to estimate costs for the subsequent combinatorial matching algorithm. The learning is usually supervised, *i.e.*, ground truth correspondences are given as training data. However, obtaining ground truth is costly, which motivates development of unsupervised learning methods.

Our work proposes one such unsupervised technique. Instead of ground truth correspondences, our method utilizes the *cycle consistency* constraint as a supervision signal, see Fig. 1. Based on pairwise correspondences of multiple images, we iteratively update matching costs to improve consistency of the correspondences.

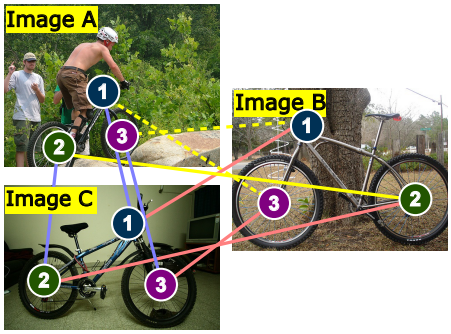

Figure 1: Illustration of cycle consistency in multi-graph matching (best viewed in color). There are three nodes in each image. They are labelled by both color (blue, green, purple) and numbers (1, 2, 3). Matches between pairs of nodes are shown by colored lines. $A \leftrightarrow B$, $B \leftrightarrow C$ and $C \leftrightarrow A$ are color coded with yellow, light purple and light pink lines. Correct matches are shown by solid, wrong matches by dotted lines. Matching of the node 2 is cycle consistent across the images, whereas nodes 1 and 3 are not.

Related Work

Supervised Deep Graph Matching methods (Fey et al. 2020a; Rolínek et al. 2020) typically consist of two parts: *Feature extraction* and *combinatorial optimization*. Whereas the first part is nowadays carried out by neural networks, the second is responsible for finding a one-to-one, possibly incomplete, matching. As shown in (Vlastelica et al. 2019; Battaglia et al. 2018), neural networks do not generalize on combinatorial tasks and cannot substitute combinatorial methods therefore.

The **architecture of neural networks** of recent deep graph matching methods such as (Rolínek et al. 2020) or (Wang, Yan, and Yang 2021) is very similar. As a backbone they use a VGG16 (Simonyan and Zisserman 2015) or a similar convolutional network for visual feature generation and a graph neural network for their refinement and combination with geometric information. Apart from specific parameters of the used networks, the key differences are the type of combinatorial solvers used and how differentiation through these solvers is dealt with.

A number of **combinatorial techniques** have been proposed to address the matching problem itself, see the re-

cent benchmark (Haller et al. 2022) and references therein. One distinguishes between linear and quadratic formulations, closely related to the *linear and quadratic assignment problems* (LAP and QAP resp.). These are deeply studied in operations research (Burkard, Dell'Amico, and Martello 2012). Whereas the optimal linear assignment minimizes the sum of node-to-node (cf. keypoint-to-keypoint in Fig. 1) matching costs, the quadratic assignment penalizes pairs of nodes (pairs of keypoints) matched to each other. Most importantly, this allows to take into account relative positions of respective keypoints.

The greater expressive power of QAPs comes at a price: In general this problem is NP-hard, whereas LAP can be exactly and efficiently solved with a number of polynomial algorithms, see a practical analysis in (Crouse 2016). Most learning pipelines, however, adopt an approximate LAP solver based on the *Sinkhorn normalization (SN)* (Bregman 1967; Peyré and Cuturi 2017) due to its inherent differentiability (Eisenberger et al. 2022).

Despite the computational difficulty of QAPs, best existing algorithms are able to provide high-quality approximate solutions for problems with hundreds of keypoints within one second or even less (Haller et al. 2022). This speed allows their direct use in end-to-end training pipelines provided there exists a way to differentiate through the solvers.

The Differentiablity Issue. When incorporating a combinatorial solver into a neural network, differentiability constitutes the principal difficulty. Such solvers take continuous inputs (matching costs in our case) and return a discrete output (an indicator vector of an optimal matching). This mapping is piecewise constant because a small change of costs typically does not affect the optimal matching. Therefore, the gradient exists almost everywhere but is equal to zero. This prohibits any gradient-based optimization.

The need for end-to-end differentiablity has given rise to various approaches that utilize differentiable relaxations of combinatorial optimization techniques (Nowak et al. 2018; Zanfir and Sminchisescu 2018; Jiang et al. 2019; Wang, Jabri, and Efros 2019; Yu et al. 2019; Fey et al. 2020b; Wang, Yan, and Yang 2021). At the end of their pipeline, all these methods use the Sinkhorn normalization. *However, such methods are either limited to approximately solving the LAP, or are solver specific, when addressing the QAP.*

The most general technique providing a *black-box differentiation of combinatorial solvers* was proposed in (Vlastelica et al. 2019). Applied to deep graph matching (Rolínek et al. 2020) it is still among the state-of-the art methods in this domain. *We also make use of this technique in our work, but in the unsupervised setting*.

Multi-Graph Matching (MGM) is a generalization of graph matching for computing correspondences between *multiple* images of the same object. From an optimization point of view, the goal is to minimize the total matching cost between all pairs of such images given the *cycle consistency* of all matchings (Swoboda et al. 2019). The problem is NP-hard even if the matching of each pair of objects is formulated as LAP (Crama and Spieksma 1992; Tang and Jebara 2017). Apart from optimization, recent works in this domain include also learning of the matching costs, see, e.g., (Wang, Yan, and Yang 2021). *Contrary to these works, we do not enforce cycle consistency during inference, and only use it as a supervision signal for training.*

Unsupervised Deep Graph Matching. The field of unsupervised deep graph matching is still under-studied. Essentially, it contains two works. The *first* one (Wang, Yan, and Yang 2020), referred to as GANN, uses cycle consistent output of an MGM solver as pseudo-ground truth for training a differentiable QAP-based matching. Costs involved in MGM as well as the QAP are updated during training, to make the output of both algorithms closer to each other in the sense of a *cross-entropy* loss. The method is restricted to specific differentiable algorithms and biased to the suboptimal solutions provided by the used MGM solver.

The *second* unsupervised training technique called SCGM (Liu et al. 2022) is based on contrastive learning with data augmentation. Specifically, in the unsupervised training stage each image and the respective keypoint graph is matched to its augmented copies. The known mapping between original and modified keypoint graphs serves as ground truth. This technique can be applied with virtually any deep graph matching method as a backbone. Moreover, it can be used as a pre-training technique for our method, as demonstrated in Section 6.

However, the augmentations are problem specific and dependent on an unknown data distribution. Also, SCGM uses two views of the same image to build its graph matching problem. It is thus biased towards complete matchings, which is a disadvantage in real world matching scenarios.

Cycle Consistency as self-supervision signal has been used in various computer vision applications such as videos, *e.g.*, (Wang, Jabri, and Efros 2019) or dense semantic matching (Kim et al. 2017). Another example is the seminal work (Zhou et al. 2016) that leverages synthetic (3D CAD) objects to obtain correct (2D image-to-image) correspondences. In all these cases, however, one considers *dense* image matching and penalizes the *Euclidean or geodesic distance* between the first and the last point in a cycle. This type of loss does not fit the discrete setting, where nothing else but the number of incorrect matches has to be minimized, *i.e.*, the Hamming distance between matches.

Another approach used, *e.g.*, in multi-shape matching, *implicitly* enforces cycle consistency by matching all objects to the *universe* (Ye et al. 2022). This, however, eliminates cycle consistency as a supervision signal, hence one has to use additional information in the unsupervised setting. This is the functional map that delivers a supervision signal in the recent multi-shape matching works (Cao and Bernard 2022; Cao, Roetzer, and Bernard 2023). However, the lack of a functional map makes this approach inapplicable to graph matching in general and in our application in particular.

Finally, the recent work (Indelman and Hazan 2023) uses a *discrete* cycle consistency as part of a loss to improve the matching results in a *supervised* setting. To differentiate the loss they use the *direct loss minimization* technique (Hazan, Keshet, and McAllester 2010) that can be seen as a limit case of the black-box-differentiation method (Vlastelica et al. 2019) utilized in our framework.

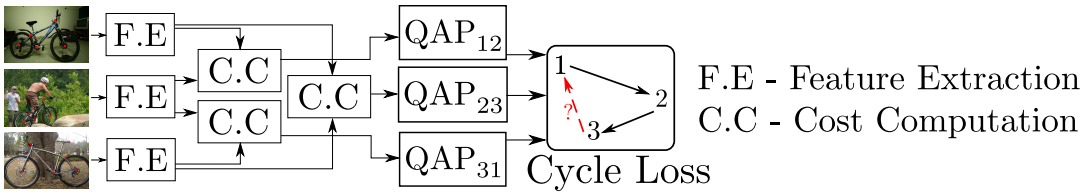

Figure 2: Overview of our framework for a batch of 3 images. Features extracted from images and keypoint positions are transformed into matching costs for each pair of images. The QAP_{ij} blocks compute the matching either as LAP or QAP. At the end the *cycle loss* counts a number of inconsistent cycles and computes a gradient for back propagation.

Contribution

We present a new principled framework for unsupervised end-to-end training of graph matching methods. It is based on a *discrete* cycle loss as a supervision signal and the black-box differentiation technique for combinatorial solvers (Vlastelica et al. 2019). Our framework can utilize *arbitrary* network architectures as well as *arbitrary* combinatorial solvers addressing LAP or QAP problems. It can handle incomplete matchings if the respective solver can.

We demonstrate flexibility of our framework by testing it with two different network architectures as well as two different QAP and one LAP solvers.

An extensive empirical evaluation suggests that our method sets a new state-of-the-art for unsupervised graph matching. Our code is available at: https://github.com/skt9/clum-semantic-correspondence.

2 Overview of the Proposed Framework

Figure 2 provides an overview of our architecture. First the features are computed for each keypoint in each image (block **Feature extraction**). These features are translated to matching costs for each pair of images (block **Cost computation**) and the respective optimization problems, QAP or LAP (blocks $\mathbf{QAP}_{12}$, $\mathbf{QAP}_{23}$, $\mathbf{QAP}_{31}$), are solved. Finally, **cycle loss** computes the *number of inconsistent cycles*.

We address all components one by one:

- Section 3 overviews the black-box differentiation technique (Vlastelica et al. 2019) that addresses the differentiability question discussed in Section 1. As the method requires a combinatorial solver to be represented in the integer linear program format, we briefly introduce this representation for LAP and QAP problems.
- Section 4 describes the key component of our framework - the *unsupervised discrete* cycle consistency loss and its differentiation.
- In Section 5 we propose a significant modification of the popular feature extraction and cost computation network of (Rolínek et al. 2020). The modified network is used in our experiments in addition to the original one.
- Finally, our experimental validation is given in Section 6. More detailed results are available in the supplement.

3 Background

Black-box differentiation of combinatorial solvers. The work (Vlastelica et al. 2019) overcomes the zero gradient problem for discrete functions in a universal way. It introduces efficient piecewise-linear approximations of the considered piecewise-constant objective functions. It has the effect of making the gradient of the approximated function informative, thus allowing for gradient-based training. Let $\mathbf{c} \in \mathbb{R}^n$ be continuous input costs and $\mathbf{x} \in \mathcal{X}$ be a discrete output taken from an arbitrary finite set $\mathcal{X}$. An important property of the method (Vlastelica et al. 2019) is that it allows to use *arbitrary* combinatorial solvers *as a black-box*, as soon as the costs $\mathbf{c}$ and the output $\mathbf{x}$ are related via an *integer linear program* (ILP):

$$\mathbf{x}(\mathbf{c}) = \underset{\mathbf{x}\in\mathcal{X}}{\arg\min} \langle \mathbf{c}, \mathbf{x} \rangle \ . \tag{1}$$

This general formulation covers a significant portion of combinatorial problems including LAPs and QAPs.

The flexibility of the black-box technique comes at the price of a somewhat higher computational cost. The used combinatorial solver must be run twice on each learning iteration: In addition to the normal execution on the forward pass, the backward pass involves another call to the solver with specially perturbed costs.

Essentially, if $L\colon \mathcal{X} \to \mathbb{R}$ denotes the final loss of the network, its gradient w.r.t. the costs $\mathbf{c}$ can be approximated as

$$\frac{dL(\mathbf{x}(\mathbf{c}))}{d\mathbf{c}} := \frac{\mathbf{x}(\mathbf{c}^\lambda) - \mathbf{x}(\mathbf{c})}{\lambda} \ . \tag{2}$$

Here $\mathbf{c}^\lambda$ is a *perturbed cost vector* computed as

$$\mathbf{c}^\lambda = \mathbf{c} + \lambda \frac{dL}{d\mathbf{x}}(\mathbf{x}(\mathbf{c})) \, . \tag{3}$$

More precisely, Equation (2) defines the gradient of a piecewise linear interpolation of L at $\mathbf{x}(\mathbf{c})$ with a hyperparameter $\lambda > 0$ controlling the interpolation accuracy. Equation (3) suggests that the loss function L is differentiable. Note that for the gradient computation (2) no explicit description of the set $\mathcal{X}$ is required.

Graph Matching Problem (QAP and LAP). For our description of the graph matching problem we mostly follow (Haller et al. 2022). Let $\mathcal{V}^1$ and $\mathcal{V}^2$ be two finite sets to be matched, *e.g.*, the sets of keypoints of two images. For each pair $i, j \in \mathcal{V}^1$, and each pair $s, l \in \mathcal{V}^2$, a cost $c_{is,jl}$ is given. Each such pair can be thought of as an edge between pairs of nodes of an underlying graph. In general, these costs are referred to as *pairwise* or *edge costs*. The *unary*, or *node-to-node* matching costs are defined by the diagonal terms of the *cost matrix* $C = (c_{is,jl})$, *i.e.*, $c_{is,is}$ is the cost for matching the node $i \in \mathcal{V}^1$ to the node $s \in \mathcal{V}^2$. For the sake of notation we further denote it as c_{is}. In turn, is will stand for (i, s) below.

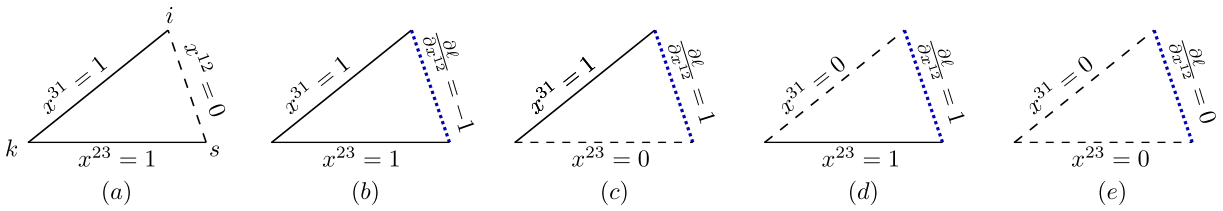

Figure 3: (a) Partial loss illustration for a triple of indices i, s, k and the respective binary variables x^{12}, x^{23}, x^{31}. The solid lines for x^{23} and x^{31} denote that these variables are equal to 1 and correspond to an actual matching between the respective points. The dashed line for x^{12} denotes that this variable is equal to 0 and therefore points indexed by i and s are *not* matched to each other. Given the values of x^{23} and x^{31} this violates cycle consistency. (b-e) Illustration of the values of the derivative $\partial \ell / \partial x^{12}$. The meaning of the solid and dashed lines as well as the position of x^{12}, x^{23} and x^{31} are the same as in (a). The thick blue dotted lines mean that x^{12} can be either 0 or 1, since $\partial \ell / \partial x^{12}$ is independent on x^{12}, see (7). So, for instance, $\partial \ell / \partial x^{12} = 1$ for $x^{23} = 0$ and $x^{31} = 1$ as illustrated by (c).

The goal of graph matching is to find a matching between elements of the sets $\mathcal{V}^1$ and $\mathcal{V}^2$ that minimizes the total cost over all pairs of assignments. It is represented as the following integer quadratic problem:

$$\min_{\mathbf{x} \in \{0,1\}^{\mathcal{V}^1 \times \mathcal{V}^2}} \sum_{is \in \mathcal{V}^1 \times \mathcal{V}^2} c_{is} x_{is} + \sum_{\substack{is,jl \in \mathcal{V}^1 \times \mathcal{V}^2 \\ is \neq jl}} c_{is,jl} x_{is} x_{jl} \tag{4}$$

$$\text{s.t.} \begin{cases} \forall i \in \mathcal{V}^1 \colon \sum_{s \in \mathcal{V}^2} x_{is} \leq 1 \,, \\ \forall s \in \mathcal{V}^2 \colon \sum_{i \in \mathcal{V}^1} x_{is} \leq 1 \,. \end{cases} \tag{5}$$

The *uniqueness constraints* (5) specify that each node of the first graph can be assigned to *at most one* node of the second graph. Due to the inequality in these constraints one speaks of *incomplete* matching contrary to the equality case termed as *complete*. The incomplete matching is much more natural for computer vision applications as it allows to account for noisy or occluded keypoints. From the computational point of view both problem variants are polynomially reducible to each other, see, *e.g.*, (Haller et al. 2022) for details. Therefore we treat them equally unless specified otherwise.

By ignoring the second, quadratic term in (4), one obtains the LAP. Note that it already has the form (1) required for black-box differentiation. As for the more general QAP case, the substitution $y_{is,jl} = x_{is} x_{jl}$ linearizes the objective of (4) and makes it amenable to black-box differentiation. The resulting ILP problem with different linearizations of the substitution $y_{is,jl} = x_{is} x_{jl}$ added as a constraint to the feasible set is addressed by a number of algorithms, see, *e.g.*, (Haller et al. 2022). We mention here only two, which we test within our framework: The LPMP solver (Swoboda et al. 2017) employed in the work (Rolínek et al. 2020), and the *fusion moves* solver (Hutschenreiter et al. 2021) showing superior results in (Haller et al. 2022).

4 Cycle Consistency Loss and Its Derivative

Let us denote the fact that a point i is matched to a point s as $s \leftrightarrow i$. We call a mutual matching of d point sets $\mathcal{V}^1, \dots, \mathcal{V}^d$ *cycle consistent*, if for any matching sequence of the form $s^{k_1} \leftrightarrow s^{k_2}$, $s^{k_2} \leftrightarrow s^{k_3}$, $\dots$, $s^{k_{m-1}} \leftrightarrow s^{k_m}$ it holds $s^{k_m} \leftrightarrow s^{k_1}$, where $k_i \in \{1, \dots, d\}$, $k_i < k_{i+1}$, $i = 1, \dots, m$.

It is well-known from the literature, see, *e.g.*, (Swoboda et al. 2019), the cycle consistency over arbitrary subsets of matched point sets is equivalent to the cycle consistency of *all triples*. We employ this in our pipeline, and define the cycle consistency loss for triples only.

Let us consider a matching of three sets $\mathcal{V}^1, \mathcal{V}^2$ and $\mathcal{V}^3$. We define the total cycle loss as the sum of *partial* losses for all possible triples of points from these sets: $L(\mathbf{x}^{12}, \mathbf{x}^{23}, \mathbf{x}^{31}) = \sum_{i \in \mathcal{V}^1} \sum_{s \in \mathcal{V}^2} \sum_{k \in \mathcal{V}^3} \ell(x^{12}_{is}, x^{23}_{sk}, x^{31}_{ki})$. Here $\mathbf{x}^{12}$ denotes a binary matching vector, *i.e.*, a vector that satisfies the uniqueness constraints (5) between $\mathcal{V}^1$ and $\mathcal{V}^2$. Vectors $\mathbf{x}^{23}$ and $\mathbf{x}^{31}$ are defined analogously.

Assume now that the triple of point indices $(i, s, k) \in \mathcal{V}^1 \times \mathcal{V}^2 \times \mathcal{V}^3$ is fixed. For the sake of notation we omit the lower indices and assume $x^{12} = x^{12}_{is}$, $x^{23} = x^{23}_{sk}$, $x^{31} = x^{31}_{ki}$ to be binary variables, see Figure 3(a) for illustration.

The partial loss ℓ penalizes cycle inconsistent configurations as the one illustrated in Figure 3(a). In particular, the partial loss is equal to 1, if $x^{12} = 0$ and $x^{23} = x^{31} = 1$. This can be achieved by, *e.g.*, the following differentiable function

$$\begin{aligned} &\ell(x^{12}, x^{23}, x^{31}) = \\ &(1 - x^{12}) x^{23} x^{31} + (1 - x^{23}) x^{12} x^{31} + (1 - x^{31}) x^{12} x^{23} \\ &\qquad = x^{12} x^{23} + x^{23} x^{31} + x^{12} x^{31} - 3 x^{12} x^{23} x^{31} \end{aligned} \tag{6}$$

where the three terms are necessary to make sure the loss function is symmetric.

The derivative of the partial loss ℓ w.r.t. x^{12} reads

$$\frac{\partial \ell}{\partial x^{12}} = x^{23} + x^{31} - 3 x^{23} x^{31} \,, \tag{7}$$

and analogously for variables x^{23} and x^{31}.

Figure 3(b-e) illustrate the values of the derivative for the four possible cases. The gradient of L is the sum of gradients of ℓ over all index triples.

Algorithm 1 summarizes our cycle-loss based unsupervised learning approach. Note that in Step 4 only unary costs c_{is} are perturbed, as the pairwise costs $c_{is,jl}$ are multiplied by the lifted variables $y_{is,jl}$ in the linearized QAP objective, and $\partial L / \partial \mathbf{y} = 0$.

5 Network Architecture

In order to show flexibility of our framework we tested it not only with the baseline network of (Rolínek et al. 2020), but

Algorithm 1: Unsupervised training algorithm

Given: Sets of keypoints to be matched $\mathbb{V} := \{\mathcal{V}^i, i \in \{1, \dots, d\}\}$; λ - the hyper-parameter from Section 3.
1. Randomly select 3 sets from $\mathbb{V}$. W.l.o.g. assume these are $\mathcal{V}^1$, $\mathcal{V}^2$ and $\mathcal{V}^3$.
2. Infer costs $\mathbf{c}^{12}, \mathbf{c}^{23}, \mathbf{c}^{31}$ for the 3 QAPs corresponding to the pairs $(\mathcal{V}^1, \mathcal{V}^2)$, $(\mathcal{V}^2, \mathcal{V}^3)$, $(\mathcal{V}^3, \mathcal{V}^1)$.
3. Solve the QAPs and obtain the respective matchings $\mathbf{x}^{12}$, $\mathbf{x}^{23}$, $\mathbf{x}^{31}$, see Figure 3.
4. Compute the perturbed costs $[\mathbf{c}^{12}]^\lambda$, $[\mathbf{c}^{23}]^\lambda$ and $[\mathbf{c}^{31}]^\lambda$ based on (3) and the loss gradient (7), *e.g.*.:

$$[\mathbf{c}^{12}_{is}]^\lambda := \mathbf{c}^{12}_{is} + \lambda \frac{\partial L}{\partial \mathbf{x}^{12}_{is}}\,,\ is \in \mathcal{V}^1 \times \mathcal{V}^2\,.$$

5. Compute the solutions $\mathbf{x}^{12}([\mathbf{c}^{12}]^\lambda)$, $\mathbf{x}^{23}([\mathbf{c}^{23}]^\lambda)$ and $\mathbf{x}^{31}([\mathbf{c}^{31}]^\lambda)$ to the QAP problems (4) with the perturbed costs.
6. Compute the gradients via (2) and backpropagate the changes to the network weights.

also with our own network, whose architecture is presented in this section.

Feature Extraction

Figure 4 shows the information flow from input, through the feature extraction block, to the construction of the matching instance. The weights for the VGG16, SplineCNN and attention layers are shared across images. Input to the feature extraction block are the image-keypoint pairs. We denote them as $(\texttt{Im}^1, \texttt{KP}^1)$ and $(\texttt{Im}^2, \texttt{KP}^2)$.

The SplineCNN layers require a graph structure for the keypoints in each image. The keypoints form the nodes of the graph. We use the Delaunay Triangulation (Delaunay 1934) of the keypoint locations to define the edge structure of this graph. We refer to the graphs of $\texttt{Im}^1$ and $\texttt{Im}^2$ as $(\mathcal{V}^1, \mathcal{E}^1)$ and $(\mathcal{V}^2, \mathcal{E}^2)$, respectively. For the sake of notation, we denote an edge (i, j) in an edge set $\mathcal{E}$ as ij.

Backbone Architecture. Following the works of (Rolínek et al. 2020; Fey et al. 2020b; Wang, Jabri, and Efros 2019) we compute the outputs of the `relu4_2` and `relu5_1` operations of the VGG16 network (Simonyan and Zisserman 2015) pre-trained on ImageNet (Deng et al. 2009). The feature vector spatially corresponding to a particular keypoint is computed via bi-linear interpolation. The set of feature vectors thus obtained are denoted as $\texttt{F}^1, \texttt{F}^2$ in Figure 4.

SplineCNN Based Feature Refinement. The keypoint features extracted from the VGG16 network are subsequently refined via SplineCNN layers (Fey et al. 2018). SplineCNNs have been shown to successfully improve feature quality in point-cloud (Li et al. 2020) and other graph structure processing applications (Verma et al. 2021).

The VGG16 keypoint features ($\texttt{F}^1, \texttt{F}^2$ in Figure 4) are input to the SplineCNN layers as node features. The input edge features to the SplineCNNs are defined as the difference of 2D coordinates of the associated nodes. We use two layers of SplineCNN with `MAX` aggregations.

The outputs of the SplineCNN layers are appended to the original VGG node features to produce the refined node features, denoted as $\mathcal{F}^1, \mathcal{F}^2$. The edge features $\mathcal{P}^k$, $k = 1, 2$, are computed as $\mathcal{P}^k_{ii'} := \mathcal{F}^k_i - \mathcal{F}^k_{i'}$, $ii' \in \mathcal{E}^k$.

Self- and Cross-Attention Layers have been recently explored in the graph matching context (Liu et al. 2023). Essentially, they implement the function

$$\mathrm{CA}(\mathbf{g}^1_i, \mathbf{g}^2, \mathbf{p}^Q, \mathbf{p}^K, \mathbf{p}^V) := \sum_{s \in \mathcal{V}_2} \mathrm{softmax}\left(\frac{(\mathbf{g}^1_i \mathbf{W}^Q + \mathbf{p}^Q_{is}) \cdot (\mathbf{g}^2_s \mathbf{W}^K + \mathbf{p}^K_{is})^\top}{\sqrt{D}}\right) \cdot (\mathbf{g}^1_i \mathbf{W}^V + \mathbf{p}^V_{is}) \quad (8)$$

$\mathbf{W}^Q$, $\mathbf{W}^K$ and $\mathbf{W}^V$ are learned projection matrices. D is the dimension of the feature vector $\mathbf{f}^1_i$. **Q**, **K**, **V** stand for Query, Key and Value respectively. Intuitively speaking, the projection matrices learn which features to take more/less notice of. The vectors $\mathbf{p}^Q, \mathbf{p}^K, \mathbf{p}^V$ are described below.

Self-Attention + RPE layer combines the self-attention mechanism (Vaswani et al. 2017) with the *relative position encoding (RPE)*. The latter has been shown to be useful for tasks where the relative ordering or distance of the elements matters (Wu et al. 2021).

The layer transforms the node features $\mathcal{F}^k = \{\mathbf{f}^k_i \mid i \in \mathcal{V}^k\}$, $k = 1, 2$ into the improved features $\mathcal{U}^k = \{\mathbf{u}^k_i \mid i \in \mathcal{V}^k\}$, $k = 1, 2$. These are computed as $\mathbf{u}^k_i = \mathrm{CA}(\mathbf{u}^k_i, \mathbf{f}^k, \mathbf{p}^Q, \mathbf{p}^K, \mathbf{p}^V)$, $k = 1, 2$. The vector $\mathbf{p}^Q = \{\mathbf{p}^Q_{is} \mid is \in \mathcal{V}_1 \times \mathcal{V}_2\}$ is computed as

$$\mathbf{p}^Q_{is} = \texttt{MLP}(\texttt{sineEmbed}(x_i - x_s))\,, \quad (9)$$

where x_i and x_s are the 2D image coordinates of the respective key points. Here sineEmbed$(\cdot)$ stands for the *sinusoidal embedding* consisting of 20 frequencies uniformly sampled in $[0, 2\pi]$, as commonly used in transformers (Vaswani et al. 2017), and MLP is a multi-layer perceptron. Vectors $\mathbf{p}^K$ and $\mathbf{p}^V$ are computed by (9) as well and differ only by learned weights of the respective MLPs.

Cross-Attention Layer incorporates feature information across graphs. It has been used in a number of applications like semantic correspondence (Yu et al. 2021) and point cloud registration (Wang and Solomon 2019) to improve feature expressivity from two different data sources.

Recall that the node features refined by the Self-Attention layer are $\mathcal{U}^k = \{\mathbf{u}^k_i \mid i \in \mathcal{V}^k\}$, $k = 1, 2$. Then the *node cross attention features for $\mathcal{U}^1$ with respect to $\mathcal{U}^2$* denoted as $\mathcal{Z}^{1/2} = \{\mathbf{z}^{1/2}_i \mid i \in \mathcal{V}^1\}$ are defined as $\mathbf{z}^{1/2}_i = \mathrm{CA}(\mathbf{u}^1_i, \mathbf{u}^2, \mathbf{0}, \mathbf{0}, \mathbf{0})$. The respective matrices W^Q, W^K and W^V defining this mapping are trained independently for self- and cross-attention layers.

The cross-attention node features for $\mathcal{U}^2$ *with respect to* $\mathcal{U}^1$ are computed analogously and denoted by $\mathcal{Z}^{2/1} = \{\mathbf{z}^{2/1}_s \mid s \in \mathcal{V}^2\}$.

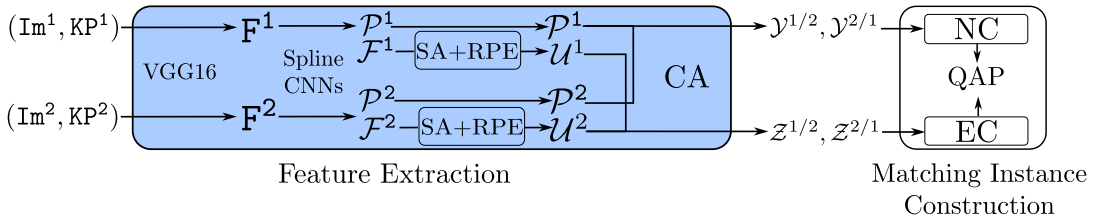

Figure 4: Information flow for feature processing and matching instance construction. The feature extraction layer is shown in the blue box. Input to the pipeline are image-keypoint pairs, $(\mathtt{Im}^1, \mathtt{KP}^1)$, $(\mathtt{Im}^2, \mathtt{KP}^2)$ in the figure. The features extracted via a pre-trained VGG16 backbone network are refined by SplineCNN layers. The outputs of the SplineCNN layers are subsequently passed through self-attention (SA) with relative position encoding (RPE) and cross-attention (CA)layers and finally used in the construction of a matching instance. NC and EC denote node and edge costs. See the detailed description in the main text.

Similarly we compute the *cross-attention edge features* $\mathcal{Y}^{1/2} = \{\mathbf{y}_{ij}^{1/2} \mid ij \in \mathcal{E}^1)\}$ and $\mathcal{Y}^{2/1} = \{\mathbf{y}_{ij}^{2/1} \mid ij \in \mathcal{E}^2)\}$ by plugging the coordinates of the edge features $\mathcal{P}^1$ and $\mathcal{P}^2$ into (8) instead of the node features $\mathcal{U}^1$ and $\mathcal{U}^2$.

Matching Instance Construction

It remains to specify how the costs for the graph matching problems in Equation (4) are computed. The unary costs c_{is} are computed as:

$$c_{is} := \left\langle \frac{\mathbf{z}_i^{1/2}}{||\mathbf{z}_i^{1/2}||}, \frac{\mathbf{z}_s^{2/1}}{||\mathbf{z}_s^{2/1}||} \right\rangle - \hat{c}\,. \tag{10}$$

The constant $\hat{c}$ regulates the number of unassigned points, *i.e.*, its larger positive values decrease this number and smaller increase. We treat $\hat{c}$ as a hyper-parameter. The edge costs $c_{is,jl}$ are given by:

$$c_{is,jl} := \left\langle \frac{\mathbf{y}_{ij}^{1/2}}{||\mathbf{y}_{ij}^{1/2}||}, \frac{\mathbf{y}_{sl}^{2/1}}{||\mathbf{y}_{sl}^{2/1}||} \right\rangle . \tag{11}$$

6 Experimental Validation

Compared Methods We evaluate our framework in four settings. As a *baseline*, we build it on top of the supervised BBGM method (Rolínek et al. 2020). That is, we reuse its network and the respective QAP solver (Swoboda et al. 2017) within our unsupervised framework. We refer to the respective algorithm as CL-BBGM. Using BBGM as a baseline allows us to compare our method directly to the competing SCGM method (Liu et al. 2022) that uses BBGM as a backbone.

Our *second* setting is a modification of CL-BBGM, referred to as CL-BBGM (SCGM), where we start with the weights learned in unsupervised fashion by SCGM (Liu et al. 2022) with BBGM as a backbone. Our *third* setup is the network described in Section 5 paired with the state-of-the-art QAP solver (Hutschenreiter et al. 2021). We term it as CLUM, which stands for ***C**ycle-**L**oss-based Unsupervised Graph **M**atching*. Our *fourth* algorithm, referred to as CLUM-L, is a variant of CLUM with a LAP solver in place of the QAP one. The edge costs generated by the network are ignored in this case.

We compare our method to the so far only existing unsupervised methods GANN (Wang, Yan, and Yang 2020) and SCGM (Liu et al. 2022). As mentioned in Section 1, SCGM is not stand-alone and requires a supervised graph matching algorithm as a backbone. Following the original SCGM paper (Liu et al. 2022), we show results with backbones BBGM and NGMv2. We also provide published results of several supervised methods for reference, see Table 1.

Experimental Setup. All experiments were run on an Nvidia-A100 GPU and a 32 core CPU. All reported results are averaged over 5 runs. The hyper-parameters are the same in all experiments. We used Adam (Kingma and Ba 2015) with an initial learning rate of 2×10^{-3} which is halved at regular intervals. The VGG16 backbone learning rate is multiplied by 0.01. We process batches of 12 image triplets. The hyper-parameter λ from (3) is set to 80. Hyper-parameter $\hat{c}$ from (10) for Pascal VOC (unfiltered) is set to 0.21 for SCGM w/BBGM, 0.257 for both CLUM and CLUM-L, 0.329 for both CL-BBGM and CL-BBGM (SCGM), respectively. Note, that $\hat{c}$ is important only for the case of incomplete assignments, *i.e.*, the Pascal VOC (Unfiltered) dataset in our experiments, see below. In other experiments a sufficiently large value of $\hat{c}$ has been used to assure complete assignments. We use image flips and rotations as augmentations.

Datasets We evaluate our proposed method on the task of keypoint matching on the following datasets: Willow Object Class (Cho, Alahari, and Ponce 2013), SPair-71K (Min et al. 2019) and Pascal VOC with Berkeley annotations (Everingham et al. 2010; Bourdev and Malik 2009). All but Pascal VOC assume *complete* matching. The consolidated results are given in Table 1. The detailed evaluation can be found in (Tourani et al. 2024).

Following (Rolínek et al. 2020) all considered methods are assumed to match pairs of images of the same category with at least three keypoints in common. We apply the same rule to select the image triples for training.

Pascal VOC with Berkeley Annotations. Following (Rolínek et al. 2020), we perform evaluations on the Pascal VOC dataset in two regimes:

- *Filtered.* Only the keypoints present in the matched images are preserved and all others are discarded as outliers. This corresponds to complete matching.
- *Unfiltered.* Original keypoints are used without any filtering. This corresponds to incomplete matching.

Willow Object Class. Similar to other methods, we pretrain our method on Pascal VOC.

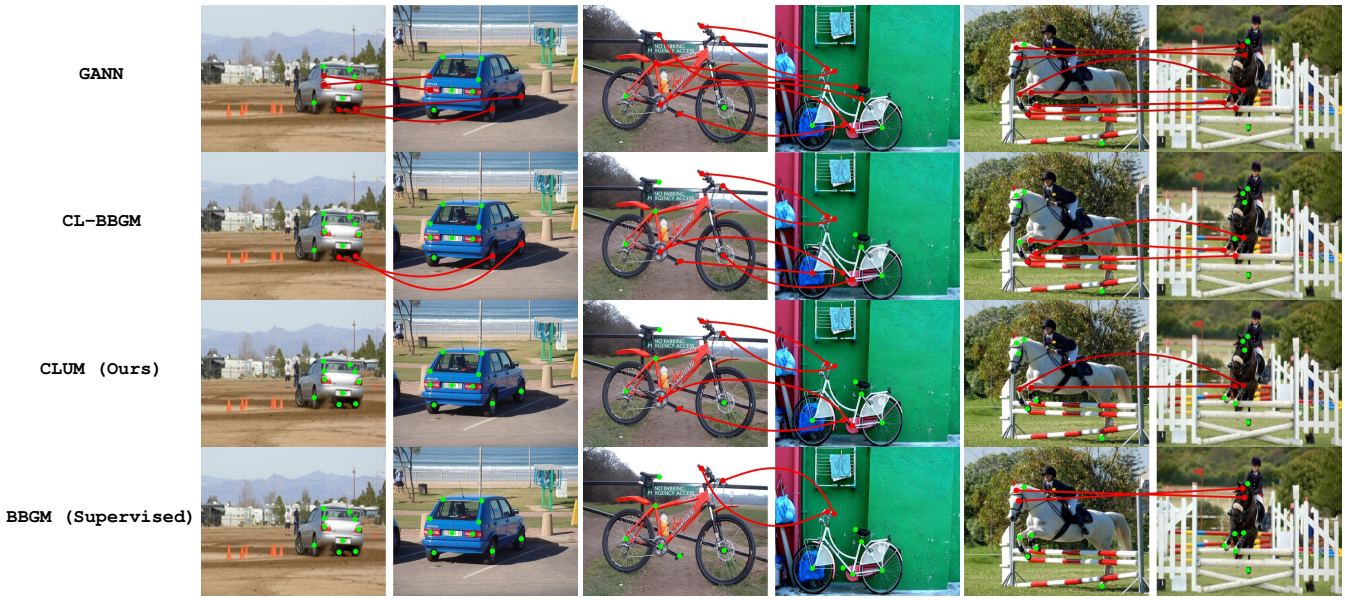

Figure 5: Visualization of matching results on the SPair-71K dataset. In addition to the unsupervised techniques GANN, CL-BBGM, CLUM we show results of the fully supervised BBGM as a baseline. Correctly matched keypoints are shown as green dots, whereas incorrect matches are represented by red lines. The matched keypoints have in general similar appearance that suggests sensible unary costs. Improving of the matching quality from top to bottom is arguably mainly due to improving the pairwise costs, with the fully supervised BBGM method showing the best results.

Dataset	**Supervised**		**Unsupervised**						
	BBGM	NGMv2	GANN	SCGM w/NGMv2	SCGM w/BBGM	CL-BBGM	CL-BBGM (SCGM)	**CLUM-L**	**CLUM (Ours)**
PascalVOC F	*79*	*80.1*	*31.5*	*54.3*	*57.1*	58.4	58.8	59.7	**62.4**
PascalVOC U	*55.4*	*54.0*	*24.3*	32.1*	33.9*	38	41.7	40.3	**43.5**
Willow	*97.2*	*97.5*	*92.0*	*91.0*	*91.3*	91.6	93.2	93.4	**95.6**
SPair-71K	*82.1*	*80.2*	31.7	36.9	38.7	40.6	41.2	41.6	**43.1**

Table 1: Consolidated results of the various deep graph matching methods on the benchmark datasets. Numbers are accuracy in percentage (higher is better) for all datasets but PascalVOC Unfiltered, (PascalVOC U) where the F1-score is used. Detailed results can be found in the supplement. Italic font is used for the values taken from original works and the ThinkMatch library. * - trained on PascalVOC Filtered (PascalVOC F).

SPair-71K is considered to have more difficult matching instances as well as higher annotation quality than PascalVOC.

Results

An **ablation study** for our method, as well as detailed results can be found in our technical report (Tourani et al. 2024).

The **evaluation results** are summarized in Table 1 and illustrated in Figure 5. In addition to the four unsupervised setups mentioned above, we trained GANN and SCGM on SPair-71K, as the respective results were missing in the original works. We also evaluated SCGM trained on PascalCOV (Filtered) on PascalVOC (Unfiltered), as SCGM is not suitable for direct training for incomplete matching. All other results are taken from the ThinkMatch (Wang, Yan, and Yang 2021) testing webpage.

Note that already our baseline algorithm CL-BBGM outperforms all existing unsupervised methods (GANN and both SCGM variants) on all datastes but Willow, where it performs slightly worse than GANN. When pre-trained with SCGM (see CL-BBGM (SCGM) in Table 1) it gets consistently better results. In turn, our high-end setup CLUM uniformly outperforms all other unsupervised techniques. The LAP variant of this method, CLUM-L, performs significantly worse than CLUM, but still better than all previously existing unsupervised methods.

7 Conclusions

We presented a new framework for unsupervised cycle-loss-based training of deep graph matching. It is extremely flexible in terms of the neural networks, as well as the combinatorial solvers it can be used with. Equipped with the best components it outperforms the state-of-the art and its flexibility suggests that its performance improves with the improvement of the components. Our framework can be adapted to other deep learning tasks like 6D pose estimation and correspondence estimation.

Acknowledgements

We thank the Center for Information Services and High Performance Computing (ZIH) at TU Dresden for its facilities for high throughput calculations. Bogdan Savchynskyy was supported by the German Research Foundation (project number 498181230).

References

Battaglia, P. W.; Hamrick, J. B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V. F.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; Gülçehre, Ç.; Song, H. F.; Ballard, A. J.; Gilmer, J.; Dahl, G. E.; Vaswani, A.; Allen, K. R.; Nash, C.; Langston, V.; Dyer, C.; Heess, N.; Wierstra, D.; Kohli, P.; Botvinick, M. M.; Vinyals, O.; Li, Y.; and Pascanu, R. 2018. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261.

Bourdev, L.; and Malik, J. 2009. Poselets: Body part detectors trained using 3d human pose annotations. In *2009 IEEE 12th International Conference on Computer Vision*, 1365–1372. IEEE.

Bregman, L. M. 1967. The Relaxation Method of Finding the Common Point of Convex Sets and its Application to the Solution of Problems in Convex Programming. *USSR Computational Mathematics and Mathematical Physics*.

Brendel, W.; and Todorovic, S. 2011. Learning spatiotemporal graphs of human activities. In *2011 International Conference on Computer Vision*, 778–785. IEEE.

Burkard, R.; Dell'Amico, M.; and Martello, S. 2012. *Assignment problems: revised reprint*.

Cao, D.; and Bernard, F. 2022. Unsupervised Deep Multi-shape Matching. In *European Conference on Computer Vision*, 55–71. Springer.

Cao, D.; Roetzer, P.; and Bernard, F. 2023. Unsupervised Learning of Robust Spectral Shape Matching. In *SIGGRAPH / ACM Transactions on Graphics (TOG)*.

Cho, M.; Alahari, K.; and Ponce, J. 2013. Learning graphs to match. In *Proceedings of the IEEE International Conference on Computer Vision*, 25–32.

Crama, Y.; and Spieksma, F. C. 1992. Approximation algorithms for three-dimensional assignment problems with triangle inequalities. *European Journal of Operational Research*, 60(3): 273–279.

Crouse, D. F. 2016. On implementing 2D rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 52(4): 1679–1696.

Delaunay, B. 1934. Sur la sphère vide. A la mémoire de Georges Vorono. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et naturelles*, 6: 793.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Eisenberger, M.; Toker, A.; Leal-Taixé, L.; Bernard, F.; and Cremers, D. 2022. A Unified Framework for Implicit Sinkhorn Differentiation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 509–518.

Everingham, M.; Van Gool, L.; Williams, C. K.; Winn, J.; and Zisserman, A. 2010. The Pascal visual object classes (VOC) challenge. *International journal of computer vision*, 88(2): 303–338.

Fey, M.; Lenssen, J. E.; Morris, C.; Masci, J.; and Kriege, N. M. 2020a. Deep Graph Matching Consensus. In *International Conference on Learning Representations (ICLR)*.

Fey, M.; Lenssen, J. E.; Morris, C.; Masci, J.; and Kriege, N. M. 2020b. Deep Graph Matching Consensus. In *8th International Conference on Learning Representations, ICLR*.

Fey, M.; Lenssen, J. E.; Weichert, F.; and Müller, H. 2018. SplineCNN: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 869–877.

Fu, K.; Liu, S.; Luo, X.; and Wang, M. 2021. Robust point cloud registration framework based on deep graph matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8893–8902.

Haller, S.; Feineis, L.; Hutschenreiter, L.; Bernard, F.; Rother, C.; Kainmüller, D.; Swoboda, P.; and Savchynskyy, B. 2022. A comparative study of graph matching algorithms in computer vision. In *European Conference on Computer Vision*, 636–653. Springer.

Hazan, T.; Keshet, J.; and McAllester, D. 2010. Direct loss minimization for structured prediction. *Advances in neural information processing systems*, 23.

Hutschenreiter, L.; Haller, S.; Feineis, L.; Rother, C.; Kainmüller, D.; and Savchynskyy, B. 2021. Fusion moves for graph matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6270–6279.

Indelman, H. C.; and Hazan, T. 2023. Learning Constrained Structured Spaces with Application to Multi-Graph Matching. In *International Conference on Artificial Intelligence and Statistics*, 2589–2602. PMLR.

Jiang, B.; Sun, P.; Tang, J.; and Luo, B. 2019. Glmnet: Graph learning-matching networks for feature matching. *arXiv preprint arXiv:1911.07681*.

Kim, S.; Min, D.; Ham, B.; Jeon, S.; Lin, S.; and Sohn, K. 2017. FCSS: Fully convolutional self-similarity for dense semantic correspondence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6560–6569.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015*.

Li, Q.; Liu, S.; Hu, L.; and Liu, X. 2020. Shape correspondence using anisotropic Chebyshev spectral CNNs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14658–14667.

Liu, C.; Zhang, S.; Yang, X.; and Yan, J. 2022. Self-supervised Learning of Visual Graph Matching. In *European Conference on Computer Vision*, 370–388. Springer.

Liu, H.; Wang, T.; Li, Y.; Lang, C.; Jin, Y.; and Ling, H. 2023. Joint graph learning and matching for semantic feature correspondence. *Pattern Recognition*, 134: 109059.

Min, J.; Lee, J.; Ponce, J.; and Cho, M. 2019. Spair-71k: A large-scale benchmark for semantic correspondence. *arXiv preprint arXiv:1908.10543*.

Nowak, A.; Villar, S.; Bandeira, A. S.; and Bruna, J. 2018. Revised note on learning quadratic assignment with graph neural networks. In *2018 IEEE Data Science Workshop (DSW)*, 1–5. IEEE.

Peyré, G.; and Cuturi, M. 2017. Computational optimal transport. *Center for Research in Economics and Statistics Working Papers*, (2017-86).

Rolínek, M.; Swoboda, P.; Zietlow, D.; Paulus, A.; Musil, V.; and Martius, G. 2020. Deep graph matching via blackbox differentiation of combinatorial solvers. In *European Conference on Computer Vision*, 407–424. Springer.

Sahillioğlu, Y. 2020. Recent advances in shape correspondence. *The Visual Computer*, 36(8): 1705–1721.

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

Swoboda, P.; Kainmüller, D.; Mokarian, A.; Theobalt, C.; and Bernard, F. 2019. A convex relaxation for multi-graph matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11156–11165.

Swoboda, P.; Rother, C.; Abu Alhaija, H.; Kainmuller, D.; and Savchynskyy, B. 2017. A Study of Lagrangean Decompositions and Dual Ascent Solvers for Graph Matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Tang, D.; and Jebara, T. 2017. Initialization and Coordinate Optimization for Multi-way Matching. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, 1385–1393.

Tourani, S.; Rother, C.; Khan, M. H.; and Savchynskyy, B. 2024. Unsupervised Deep Graph Matching Based on Cycle Consistency. *ArXiv:2307.08930*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Verma, N.; Boukhayma, A.; Boyer, E.; and Verbeek, J. 2021. Dual Mesh Convolutional Networks for Human Shape Correspondence. In *2021 International Conference on 3D Vision (3DV)*, 289–298. IEEE.

Vlastelica, M. P.; Paulus, A.; Musil, V.; Martius, G.; and Rolinek, M. 2019. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations*.

Wang, R.; Yan, J.; and Yang, X. 2020. Graduated assignment for joint multi-graph matching and clustering with application to unsupervised graph matching network learning. *Advances in Neural Information Processing Systems*, 33: 19908–19919.

Wang, R.; Yan, J.; and Yang, X. 2021. Neural graph matching network: Learning Lawler's quadratic assignment problem with extension to hypergraph and multiple-graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Wang, X.; Jabri, A.; and Efros, A. A. 2019. Learning correspondence from the cycle-consistency of time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2566–2576.

Wang, Y.; and Solomon, J. M. 2019. Deep closest point: Learning representations for point cloud registration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3523–3532.

Wu, K.; Peng, H.; Chen, M.; Fu, J.; and Chao, H. 2021. Rethinking and improving relative position encoding for vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10033–10041.

Ye, Z.; Yenamandra, T.; Bernard, F.; and Cremers, D. 2022. Joint deep multi-graph matching and 3d geometry learning from inhomogeneous 2d image collections. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 3125–3133.

Yu, H.; Li, F.; Saleh, M.; Busam, B.; and Ilic, S. 2021. Cofinet: Reliable coarse-to-fine correspondences for robust pointcloud registration. *Advances in Neural Information Processing Systems*, 34: 23872–23884.

Yu, T.; Wang, R.; Yan, J.; and Li, B. 2019. Learning deep graph matching with channel-independent embedding and Hungarian attention. In *International conference on learning representations*.

Zanfir, A.; and Sminchisescu, C. 2018. Deep learning of graph matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2684–2693.

Zhou, T.; Krahenbuhl, P.; Aubry, M.; Huang, Q.; and Efros, A. A. 2016. Learning dense correspondence via 3d-guided cycle consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 117–126.