CLAP: Concave Linear APproximation for Quadratic Graph Matching

Yongqing Liang[1[0000−0002−7282−0476]], Huijun Han[1[0009−0009−1360−4886]], and Xin Li[1[0000−0002−0144−9489]]

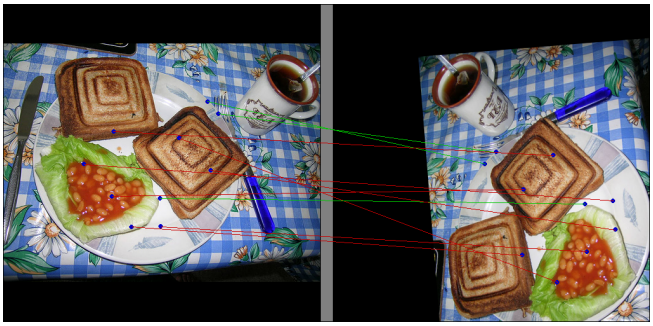
Texas A&M University, College Station TX 77840, USA
{lyq, hazelhan, xinli}@tamu.edu

Abstract. Solving point-wise feature correspondence in visual data is a fundamental problem in computer vision. A powerful model that addresses this challenge is to formulate it as graph matching, which entails solving a Quadratic Assignment Problem (QAP) with node-wise and edge-wise constraints. However, solving such a QAP can be both expensive and difficult due to numerous local extreme points. In this work, we introduce a novel linear model and solver designed to accelerate the computation of graph matching. Specifically, we employ a positive semi-definite matrix approximation to establish the structural attribute constraint. We then transform the original QAP into a linear model that is concave for maximization. This model can subsequently be solved using the Sinkhorn optimal transport algorithm, known for its enhanced efficiency and numerical stability compared to existing approaches. Experimental results on the widely used benchmark PascalVOC showcase that our algorithm achieves state-of-the-art performance with significantly improved efficiency. We plan to release our code for public access.

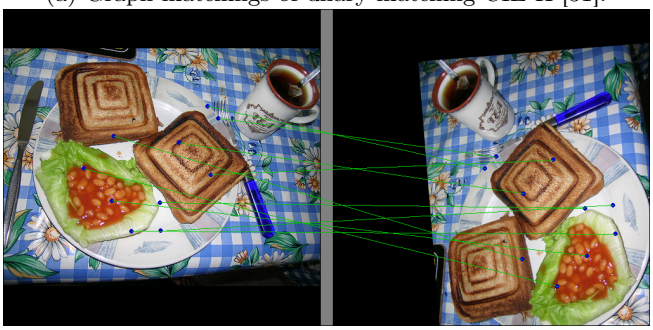
Keywords: Image Feature Matching · Quadratic Assignment Problem · Quadratic Graph Matching.

1 Introduction

Graph is a natural structure to encode real-world data. Graph matching is to find node-to-node correspondences between two given graphs. Hence, as a general and powerful tool to discover correlation or detect similar structural pattern between graphs, graph matching has been widely used in many computer vision tasks including keypoints matching [21], multi-object tracking [6], and scene flow estimation [18]. A graph matching pipeline starts from extracting keypoints and their descriptors. Then, each image/frame/object is modeled using a graph, whose nodes correspond to keypoints or regions and are associated with their descriptors (i.e., *attributes*) and edges encode relationship between these nodes. And then, the matching is solved through certain optimization procedure that finds a node-to-node map that minimizes content/structural matching cost. Recent graph matching methods can be divided into two categories [27], *Unary Matching* and *Pairwise Matching*.



(a) Graph matchings of unary matching CIE-H [31].



(b) Graph matchings of our pairwise matching model.

Fig. 1. Graph matching results by unary matching and pairwise matching. The image undergoes an affine transformation. Green/red lines indicate correct/wrong matchings.

Unary Matching methods [26,21,19] formulate graph matching as node matching. Deep neural networks are often used to learn discriminating descriptors on keypoints (nodes); and these descriptors can be designed to encode both local context surrounding nodes and inter-node relationship. Then, subsequent matching can be computed using similarity between these descriptors (namely, *pointwise affinity*). Unary matching methods can run fast and achieve competitive results on common benchmarks. However, as pointed out in recent studies [5], when keypoints share similar descriptors but different structural contexts, unary methods often fail to produce reliable matching. Fig. 1 (a) also illustrates such an example: these two images undergo an affine transformation, a unary matching method such as CIE-H [31] matches two graphs by node attributes that are extracted from a deep neural network. Unary matching methods often fail to find the correct correspondence when node attributes become indistinctive (because (1) the point's neighboring textures are not unique, and (2) transformations involving significant rotations make deep features less reliable).

Pairwise Matching methods [33,24,34,5] construct the matching model using not only *pointwise affinity* information but also *pairwise structure* constraints and formulate graph matching as a Koopmans-Beckman Quadratic Assignment

Problem (KB-QAP) [10]. People integrate local features from adjacent nodes to compose *edge attributes* to encode the inter-node structure information. Then these methods develop pairwise structure constraint to penalize edge attributes discrepancy according to node correspondence.

Pairwise structure constraints make the matching more robust against global (camera-wise) and local (object-related) geometric transformations [5,24]. As shown in Fig. 1 (b), with the help of structure constraint, a pairwise matching model (*e.g.*, our model) could find correct correspondence. However, The objective function of KB-QAP is nonconvex-nonconcave because the Hessian matrix of its quadratic term is indefinite. The solution of the KB-QAP has many local maximums or minimums. These local extremes make the solver sensitive to initial poses; and the slow convergence makes the solver computationally expensive. This remains as the bottleneck of pairwise matching models, and limit their applications on real-time tasks such as multi-object tracking and others.

In this paper, we propose a linear model, named *CLAP*, to convert the pairwise graph matching to a concave maximization problem. We follow the objective function of KB-QAP but formulate the pairwise structure constraint into a linear model under L1 norm. To build such a linear model, we leverage the decomposition property of positive semi-definite matrix. We analyze the widely-used edge attributes (*e.g.*, Euclidean distance [24] and inner-product distance [21,26]), and convert them to be positive semi-definite and construct a linear structure constraint.

We showed that our new objective function is concave, whose maximization is easy and results in a global maximum. Using the Sinkhorn algorithm [1], our CLAP model can be efficiently solved by the Lagrangian multiplier method. Experiments showed that our method achieves similar accuracy with other state-of-the-art methods but runs significantly faster.

2 Related Work

2.1 Unary Matching

Unary matching based methods [31,26,27,2,19,21] formulates the graph matching as node matching. They first extract local features for each node as node attributes, then iteratively enhance the node attributes by various feature refinement modules, such as GNN [19,26,31] and Transformer [21]. The assumption is that graph structure can be sufficiently encoded into node attributes. Therefore they can just use the node-to-node affinity matrix by inner-product or metric learning [27,26,31] to solve graph matching. With the node-to-node affinity matrix, the matching can be computed by using nearest neighbor search [30,20], dual-softmax [21], or optimal transport [19,27,26].

Despite their simplicity, the unary matching models have two limitations: (1) Feature embedding modules are often adopted to enhance the node attributes/descriptors, but they also slow down the overall running time. (2) More importantly, the assumption that node attributes are distinctive enough to encode graph structure sometimes doesn't hold. For example, when the region of

interest in the images do not contain rich texture information, their node attributes can be indistinguishable and ambiguous. Without the effectively modeling structure information, unary matching models often fail to produce reliable matching results.

2.2 Pairwise Matching

Pairwise matching methods [32,24,5] formulate the graph matching as quadratic assignment problems (QAP) [15], such as Lawler QAP [12] and Koopmans-Beckman QAP (KB-QAP) [10]. The objective function uses node and edge similarity constraints to build affinity matrix. When the affinity matrix of Lawler's QAP can be decomposed by inverse Kronecker product, KB-QAP is a special case of Lawler's QAP with much lower space complexity [15,24]. Since this condition is true in general cases, recent papers chose KB-QAP as their objective functions. Because the discrete QAP is NP-complete [9], researchers relax the feasible field into a continuous domain to find approximate solutions in polynomial time.

The bottlenecks in developing effective QAP solvers are on the problem of many local extreme points and slow convergence because the Hessian matrix of the QAP objective function is often nonconvex-nonconcave. Various approximate algorithms have been proposed to solve the QAP function. Umeyama [22] used the absolute values of eigenvectors of the edge attributes to construct the structure constraint, but such approximation changes the physical meaning of the edge attributes and leads to large matching errors [33]. Lu *et al.* [16] proposed a fast projected fixed point scheme. FGM [36] factorized the affinity matrix of Lawler's QAP into small matrices. PATH [33], Gao *et al.* [5] and Wang *et al.* [24] used Frank-Wolfe method [7] to obtain an approximate solution. AFAT [25] suggested the slow convergence rate could be addressed by partial graph matching. Thus AFAT ruled out the unpaired outliers within the detected keypoints using attention-fused prediction for the number of reliable inliers in a data-driven manner. The noisy nature of data contaminates the robostness of deep neural networks for graph matching. As a solution, momentum distillation is exploited by COMMON [14] to emphasize the graph consistency by gradually lowering the supervision from the ground truth correspondence.

Existing methods are generally time consuming to find the optimal correspondences because they often require dozen iterations to convergence. The long computation time limits the application of pairwise matching methods for real-time tasks such as multi-object tracking. Our method belongs to pairwise matching and is based on KB-QAP objective function.

3 Algorithm

3.1 Problem Definition

Graph matching solves node-to-node correspondence between two given graphs $G_A = \{V_A, E_A\}$ and $G_B = \{V_B, E_B\}$, where V and E denote the node and edge

sets. The node sets V_A and V_B contain n and m feature keypoints extracted from I_A and I_B respectively.

The similarity between two descriptors can be measured in the feature space and represented as $U \in \mathbb{R}^{n\times m}$. Here U_{ij} represents node similarity between $(v_A)_i \in V_A$ and $(v_B)_j \in V_B$.

The edge set E encodes spatial (either in the positional space or feature space) correlation between two nodes in a graph. The definition of edge attributes depends on the graph matching models, *e.g.*, Lawler QAP [12,29] and Koopmans-Beckman QAP (KB-QAP) [10]. We adopt the widely used KB-QAP model. The KB-QAP model uses adjacency weight matrices $D_A \in \mathbb{R}^{n\times n}$ (and $D_B \in \mathbb{R}^{m\times m}$) to store the structure information of graph G_A (and G_B).

The graph matching problem is to find an *optimal node-to-node assignment* $P \in \{0,1\}^{n\times m}$, where $P_{ij} = 1$ indicates that nodes $(v_A)_i \in V_A$ and $(v_B)_j \in V_B$ are matched and $P_{ij} = 0$ otherwise. Following the graph matching setting, the feasible field of $\mathcal{P}$ is

$$\mathcal{P} \triangleq \left\{P \in \{0,1\}^{n\times m};\ P\mathbf{1_m} = \mathbf{1_n},\ P^T\mathbf{1_n} \leq \mathbf{1_m}\right\}, \tag{1}$$

where $\mathbf{1_m}$ is a vector of m ones. The KB-QAP formulates graph matching as maximizing the sum of the node and the edge similarities,

$$\max_{P\in\mathcal{P}} \left(\sum_{i,j} P_{ij}U_{ij} - \lambda||D_A - PD_BP^T||_F^2 \right), \tag{2}$$

where $\lambda > 0$ is a balancing weight, and $||\cdot||_F^2$ is the square of the Frobenius norm. The columns of $P \in \mathcal{P}$ are orthogonal. Rewriting the second term of Eq. (2) with the trace of that matrix, we have

$$\begin{aligned} &- ||D_A - PD_BP^T||_F^2 = -tr((D_A - PD_BP^T)^T(D_A - PD_BP^T)) \\ &= 2tr(P^TD_A^TPD_B) - tr(D_A^TD_A) - tr(D_B^TD_B). \end{aligned}$$

After removing the constant items, maximizing $-||D_A - PD_BP^T||_F^2$ is equivalent to maximizing $tr(P^TD_A^TPD_B)$. Thus, we can rewrite the objective function Eq. (2) as

$$\max_{P\in\mathcal{P}} \left(\sum_{i,j} P_{ij}U_{ij} + \lambda tr(P^TD_A^TPD_B) \right). \tag{3}$$

The objective function Eq. (3) is defined on discrete space and is known as an NP problem. Many recent methods [5,24,7] first relax the feasible field $\mathcal{P}$ to a continuous field $\mathcal{P}'$,

$$\mathcal{P}' \triangleq \left\{P \in [0,1]^{n\times m};\ P\mathbf{1_m} = \mathbf{1_n},\ P^T\mathbf{1_n} \leq \mathbf{1_m}\right\}, \tag{4}$$

then adopt gradient descend based methods to solve it.

However, the second term in Eq. (3) is quadratic and its Hessian matrix is indefinite. Hence, this objective function is nonconcave and has many local maximums. Existing solvers often require long computing time to converge.

3.2 Our Model

Symmetric Edge Matrices. We first analyze the properties of edge attribute matrices D_A and D_B. Recent graph matching models construct them using pairwise structure information (*e.g.*, Euclidean distance [24], adjacency [5,26], Mahalanobis distance [5], and inner-product distance [19,21]) between feature points. In most of these models, D_A and D_B are symmetric matrices. However, they are usually not *positive semi-definite*.

If we can make the edge weight matrix D *positive semi-definite* (Sec. 3.3 will discuss the conversion of given D_A and D_B into positive semi-definite matrices.), then it can be decomposed as $D = HH^T$. Based on this, we can design a fast linear graph matching model.

Linear Matching Model. If D_A and D_B are positive semi-definite, then we can decompose them as $D_A = H_A H_A^T$ and $D_B = H_B H_B^T$. The quadratic term in Eq. (3) can be written as

$$\begin{aligned} & tr(P^T D_A^T P D_B) \\ =& tr(P^T D_A P D_B) = tr(P^T H_A H_A^T P H_B H_B^T) \\ =& tr(H_B^T P^T H_A H_A^T P H_B) = tr((H_A^T P H_B)^T (H_A^T P H_B)) \\ =& ||H_A^T P H_B||_F^2 = \sum_{i,j} (|H_A^T P H_B|_{ij})^2. \end{aligned} \tag{5}$$

This term Eq. (5) is quadratic and is in the form of the sum of squares (*i.e.*, L2 norm). This converts the matching model Eq. (3) from a *nonconvex-nonconcave* function to a *convex* function. Nevertheless, maximizing a convex function still results in multiple local maximum and is difficult to solve. Our idea is to change this sum-of-squares form into a sum of absolute values (*i.e.*, L1 norm). This modification has two main benefits: (1) Compared with L2 norm, optimizing L1 norm promotes sparsity and is more robust against outliers. L1 norm has been used in PCA [11,3] and kernel discriminant analysis [35], and demonstrated its better robustness when outliers exist. (2) Numerically, combining with the Entropy regularization, the objective function can be made concave. Maximizing a concave objective function is much easier and it quickly converges to a global maximum.

Thus, we propose the **linear objective function** for graph matching,

$$\max_{P \in \mathcal{P}'} \left(\sum_{i,j} P_{ij} U_{ij} + \lambda \sum_{i,j} |H_A^T P H_B|_{ij} \right). \tag{6}$$

Inspired by the recent successful usage of Sinkhorn algorithm [1] in Optimal Transport problem, in Eq. (6), we add an *Entropy Regularization* $h(P) = -\sum_{i,j} P_{ij} \log P_{ij}$. Hence, finally, we have

$$\max_{P \in \mathcal{P}'} \left(\sum_{i,j} P_{ij} U_{ij} + \lambda \sum_{i,j} |H_A^T P H_B|_{ij} + \epsilon h(P) \right), \tag{7}$$

where $\epsilon > 0$ balances the weight of $h(P)$.

This objective function Eq. (7) has negative semi-definite Hessian matrix $\mathcal{H}$, whose diagonal elements $\mathcal{H}_{ij,ij} = -\epsilon/P_{ij}$ and non-diagonal elements are all zeros. Because $P_{ij} \geq 0$, the $\mathcal{H}$ is negative semi-definite. Thus, this objective function Eq. (7) is concave and has a global maximum. And it can be solved efficiently using Lagrange multipliers (optimization details elaborated in Sec. 3.4).

3.3 Graph Attribute Construction

When formulating the feature matching problem on a graph, both *node similarity* (that aligns keypoints with similar descriptors) and *structure similarity* (that matches relative positional or contextual correlation between keypoints) should be considered. These similarities are encoded in the matching of nodes and edges of the graph using the *node similarity term* and *edge similarity term*, respectively.

Node Similarity. Recent matching models first use neural networks to extract keypoints and their local feature descriptors, then compute the node similarity by a direct inner-product or a learnable metric [19,26,5,21]. We chose Gao *et al*. [5]'s model as our baseline as it has the state-of-the-art accuracy. Our main design is to speedup the more expensive edge similarity term. Hence, for a fair comparison, on the node similarity term, we followed the same setting of [5] and used VGG16 to extract descriptors and get the same node similarity matrix.

Let $\psi(V_A) \in \mathbb{R}^{n\times d}$ and $\psi(V_B) \in \mathbb{R}^{m\times d}$ represent the feature maps of node V_A and V_B. The ith row of the feature map is the descriptor of the ith node. The node similarity matrix U_{ij} is computed like the Mahalanobis distance,

$$U_{ij} = \psi(V_A)\Sigma_U\psi(V_B)^T, \tag{8}$$

where $\Sigma_U \in \mathbb{R}^{n\times m}$ is a learnable matrix. After the training stage, Σ_U is fixed during the inference stage.

Edge Similarity. Edge attribute matrices D_A and D_B are designed to characterize the structure of the matching model. As we discussed in Sec. 3.2, we want to develop a scheme to convert any given symmetric edge attribute matrix D to a positive semi-definite matrix $\hat{D}$.

In graph matching models, because edges are usually undirected and unordered, most edge attributes that represent the inter-node relationship in graphs, such as Euclidean distance [24], adjacency [5,26], Mahalanobis distance [5], and inner-product distance [19,21], inherently form symmetric attribute matrices. Let D be an edge attribute matrix, where D_{ij} represents the edge attribute between node v_i and v_j. D usually has the following form: (1) When $i \neq j$, we have $D_{ij} = D_{ji}$. (2) When $i = j$, a diagonal entry D_{ii} is undefined and is set to 0 in most existing models.

Given the two edge attributes D_A and D_B, we can convert them to positive semi-definite matrices $\hat{D_A}$ and $\hat{D_B}$ by just modifying the diagonal entries,

$$d_x = \max\left\{R_i = \sum_{k\neq i} |(D_x)_{ik}|, i \in \{1, \cdots, n\}\right\}, d_{max} = \max\{d_A, d_B\}, \tag{9}$$

where $x = \{A, B\}$, d_A and d_B are the maximums of the sum of the absolute entries in each rows, and d_{max} is the maximum of d_A and d_B. We use d_{max} to modify both edge attributes to make sure they have the same diagonal entries,

$$(\hat{D}_x)_{ij} = \begin{cases} d_{max} & i = j \\ (D_x)_{ij} & i \neq j \end{cases}, \ x = \{A, B\}. \tag{10}$$

We use $\hat{D}_A$ and $\hat{D}_B$ as the new edge attribute matrices to replace the original D_A and D_B in graph matching computation.

$\hat{D}_A$ and $\hat{D}_B$ have two important properties: (1) *Similar structural constraint.* The non-diagonal entries of $\hat{D}_x$ and the original one D_x are the same, which characterize the structure and the edge attributes of the graph. Recall the structure constraints in the objective function Eq. (2), $||\hat{D}_A - P\hat{D}_B P^T||_F^2$. By definition, P and P^T can be treated as permutation matrices that switch the rows and columns of $\hat{D}_B$. Hence, $P\hat{D}_B P^T$ has the same diagonal entries as D_B. Then, since $\hat{D}_A$ and $\hat{D}_B$ have the same diagonal entries d_{max}, optimizing $||D_A - PD_B P^T||_F^2$ and $||\hat{D}_A - P\hat{D}_B P^T||_F^2$ is equivalent. Modifying D to $\hat{D}$ does not affect the optimal P. (2) *Positive Semi-definiteness.* They are both positive semi-definite and can be decomposed to construct the linear matching model. We prove the positive semi-definiteness property by Gershgorin circle theorem [23], every eigenvalue of a matrix M lies within at least one of the Gershgorin discs $r(\hat{D}_{ii}, R_i)$.

3.4 CLAP Solver

With positive semi-definite edge attribute matrices and their decomposition, our matching model Eq. (7) is concave and has a global maximum. This model can be solved efficiently using a Lagrangian multiplier method.

Let $\mathcal{L}(P, \mu_1, \mu_2)$ be the Lagrangian of Eq. (7) with dual variables $\mu_1 \in \mathbb{R}^n, \mu_2 \in \mathbb{R}^m$,

$$\begin{aligned} \mathcal{L}(P, \mu_1, \mu_2) = &\sum_{i,j} P_{ij} U_{ij} + \lambda \sum_{i,j} |H_A^T P H_B|_{ij} - \epsilon \sum_{i,j} P_{ij} log P_{ij} \\ &+ \mu_1^T (P\mathbf{1_m} - \mathbf{1_n}) + \mu_2^T (P^T \mathbf{1_n} - \mathbf{1_m}). \end{aligned} \tag{11}$$

For any couple (i, j), let the first derivative of Eq. (11) be zero, we have

$$0 = U_{ij} - \epsilon - \epsilon log P_{ij} + (\mu_1)_i + (\mu_2)_j + \lambda \sum_{k=1}^{k_1} \sum_{l=1}^{k_2} \delta\left((H_A^T P H_B)_{kl}\right) (H_A)_{ik} (H_B)_{jl}, \tag{12}$$

where $H_A \in \mathbb{R}^{n \times k_1}$, $H_B \in \mathbb{R}^{m \times k_2}$, and $\delta(\cdot) = \{-1, 1\}$ is the sign function. The solution for Eq. (12) is,

$$P_{ij} = \exp\left(\frac{(\mu_1)_i}{\epsilon} - \frac{1}{2}\right) \exp\left(\frac{m_{ij}}{\epsilon}\right) \exp\left(\frac{(\mu_2)_j}{\epsilon} - \frac{1}{2}\right), P\mathbf{1_m} = \mathbf{1_n}, P^T \mathbf{1_n} = \mathbf{1_m}, \tag{13}$$

Table 1. Comparing the baseline and the revised CLIP model on synthetic image pairs. Three types of edge attributes (*i.e.*, learning-based, adjacency matrix, and edge length distance) are tested. The baseline model is qc-DGM [5].

Method	Acc. (%)	Time (ms)	FPS
Learning : qc-DGM	42.6	228.4	4.4
Learning : Ours	**44.3**	**159.8**	**6.3**
Adjacency : qc-DGM	32.8	21.6	46.3
Adjacency : Ours	**79.2**	**9.0**	**111.1**
Length : qc-DGM	63.8	22.8	43.9
Length : Ours	**98.1**	**8.7**	**114.9**

Table 2. Comparisons on synthetic image pairs.

	Method	Acc. (%)	Time (ms)	FPS
Unary	IPCA [26]	23.7	207.1	4.83
	PCA [27]	30.4	204.2	4.90
	CIE-H [31]	37.3	202.5	4.93
Pairwise	qc-DGM [5]	42.6	228.4	4.4
	Ours	**44.3**	**159.8**	**6.3**

where $m_{ij} = U_{ij} + \lambda \sum_{k,l} \delta((H_A^T P H_B)_{kl})(H_A)_{ik}(H_B)_{jl}$. Note that since δ is a sign function, m_{ij} is a scalar that does not contain the variable P_{ij}. We substitute the P_{ij} in Eq. (13) back to the constraints of feasible field $\mathcal{P}'$. By Cuturi's algorithm [1], $\mathcal{L}(P, \mu_1, \mu_2)$ has maximum point and can be computed with Sinkhorn's fixed point iteration. For numerical stability, we follow Cuturi and Peyre [17] to solve the Eq. (13) in log-domain. Finally, we follow [5,31] to use the Hungarian algorithm for a discrete solution.

4 Experiments

We conducted experiments to evaluate the proposed CLAP model. We evaluated our matching model's performance under various edge attributes on image pairs that differ by randomly synthesized geometric transformations. We also compared our model with the SOTA graph matching algorithms.

Evaluation Metrics. We compared *matching accuracy* and *matching efficiency*. A widely adopted *matching accuracy* metric is the accuracy score *Acc*. Given an assignment matrix P of n nodes, *Acc* is defined as $Acc = \frac{1}{n}\sum_{ij}(P^* \circ P)_{ij}$, where $P^* \in \{0,1\}^{n\times n}$ is the groundtruth correspondence, and operator $\circ$ is the element-wise production. To evaluate *matching efficiency*, we measured the average per graph matching time: *Time* and *FPS* (frame-per-second).

We compared our model with state-of-the-art matching methods, including **unary matching** models: PCA [27], IPCA [26], and CIE-H [31], and **pairwise matching** models: GMN [32], NGM [28], HNN-HM [13], and qc-DGM [5]. Among these methods, since no codes nor pretrained models of HNN-HM [13],

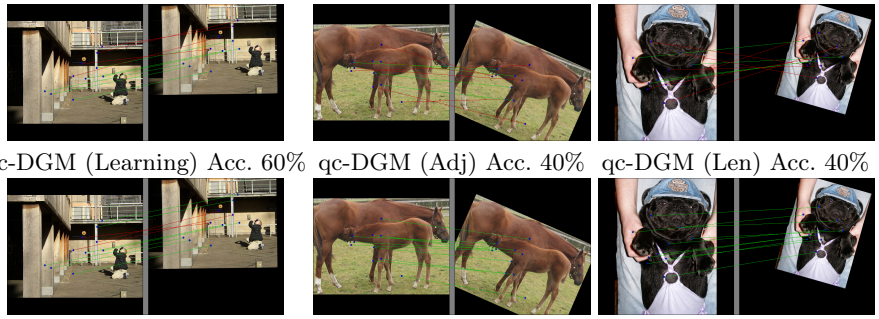


Fig. 2. Qualitative comparisons on synthetic transformed images, where green/red lines indicate correct/wrong matchings, respectively. 'Adj', 'Len', and 'Learning' represent the three types of edge attributes adopted in qc-DGM and our models, respectively.

LCSGM [29], and GLMNet [8] is available, we use the evaluation scores from their papers directly. For other methods, we ran their pretrained models. We set $\lambda = 0.1$ and $\epsilon = 1$. All the experiments were done on an Intel Xeon(R) CPU E5-2630 with one NVIDIA GeForce 1080Ti graphic card. Source code: `https://github.com/xmlyqing00/clap`.

4.1 Results on Synthetic Transformations

Dataset Construction. We synthesized $1,000$ pairs of images to evaluate the robustness of graph matching models under random affine transformations. The backgrounds are randomly selected from Pascal VOC dataset [4]. For each pair of images (I_A, I_B), image I_A contains 10 nodes with random 2D-coordinates. Then we synthesized an affine transformation with randomly generated scaling (range: $[0.5, 1)$), rotation (range: $[-\pi, \pi)$), and translation (range: $[-w/4, w/4)$ on x-axis and $[-h/4, h/4)$ on y-axis). This affine transformation was applied on I_A (and its node positions) to get I_B (and the new node positions).

Different Edge Attribute Constructions. Our proposed CLAP model is compatible with various edge attributes that are used in existing pairwise matching methods. We tested on three types of commonly used edge attributes: (1) learning based edge descriptors [5], (2) 0/1 adjacency matrix [36,5], and (3) edge length structure [24]. We denote them as "Learning", "Adjacency", and "Length" attributes in the following.

As shown in Tab. 1, our linear model not only preserves but actually surpasses the quadratic baseline model qc-DGM [5] in matching accuracy, on all the three edge attribute settings. And our model is also significantly faster.

One main reason that our linear model often leads to more accurate results in this experiment is that the quadratic baseline model is nonconcave and has multiple local maximums, especially when the image transformation is significant.

Table 3. Quantitative comparisons on Pascal VOC benchmark. We used the pretrained models that were trained on the same benchmark. Our baseline is qc-DGM [5]. $^+$ indicates an additional post-processing. The accuracy scores are in percentage. Red numbers indicate the best performance, and Blue numbers indicate the second best.

Method	GMN	PCA	NGM	IPCA	GLM	HNN-HM	LCSGM	CIE-H	qc-DGM	qc-DGM$^+$	Ours
areo	40.8	49.0	52.5	54.0	52.0	39.6	46.9	51.7	49.3	49.8	49.2
bike	58.0	60.8	62.0	64.9	67.3	55.7	58.0	67.6	65.6	66.9	65.7
bird	59.8	65.1	62.5	64.8	63.2	60.7	63.6	70.0	60.8	62.0	61.1
boat	50.5	58.2	59.2	61.5	57.4	76.4	69.9	61.1	56.4	56.9	56.2
bottle	78.5	77.3	78.4	78.9	80.3	87.3	87.8	82.4	82.5	82.6	81.9
bus	69.5	73.9	77.1	73.8	74.6	86.2	79.8	76.0	78.9	78.9	78.8
car	65.9	65.7	73.8	71.7	70.0	77.6	71.8	70.6	71.9	72.3	72.1
cat	64.7	68.4	68.1	70.9	72.6	54.2	60.3	71.7	71.3	71.6	71.5
chair	40.3	42.9	43.8	46.6	38.9	50.0	44.8	43.5	41.7	42.8	42.1
cow	61.8	63.9	66.6	66.2	66.3	60.7	64.3	70.5	67.7	67.9	67.3
table	66.8	45.2	48.5	40.3	77.3	78.8	79.4	63.5	73.4	77.5	77.5
dog	62.3	68.2	63.5	68.3	65.7	51.2	57.5	71.3	64.4	65.3	64.0
horse	62.4	66.6	65.3	67.1	67.9	55.8	64.4	70.9	70.7	71.5	69.7
mbike	58.9	61.5	63.0	65.1	64.2	60.2	57.6	66.8	65.8	66.3	65.9
person	37.2	44.2	47.6	49.3	44.8	52.5	52.4	47.3	48.2	48.8	47.4
plant	79.1	83.0	83.0	85.7	86.3	96.5	96.1	85.7	91.5	93.0	92.5
sheep	66.8	66.7	67.3	68.9	69.0	58.7	62.9	69.0	68.4	69.5	69.0
sofa	49.9	57.4	62.3	60.0	61.9	68.4	65.8	61.3	66.1	65.7	63.8
train	85.5	78.3	80.3	82.4	79.3	96.2	94.4	83.5	88.1	88.1	88.0
tv	91.0	89.1	90.0	88.6	91.3	92.8	92.0	89.7	92.0	92.1	91.8
Mean (%)	62.5	64.3	65.8	66.5	67.5	68.0	68.5	68.7	68.7	69.5	68.8
Time (ms)	89.4	89.2	105.2	94.9	-	-	-	83.3	116.4	145.5	73.7
FPS	11.19	11.21	9.51	10.54	-	-	-	12.00	8.59	6.87	13.57

In contrast, our linear model has a global maximum and is easy to be solved. This experiment demonstrates that our model (using L1 norm) is better than the baseline (using L2 norm). Note that here the learning-based edge attributes were constructed using the pre-trained descriptors learned from the large benchmark dataset Pascal VOC. Their matching accuracy is low, showing that such learned descriptors don't generalize well for these synthesized transformations (probably much more significant than those in the benchmark). Qualitative comparisons are shown in Fig. 2.

Structure Info Encoding: Unary vs Pairwise Matching. We also compared the matching computed by unary models and pairwise models in Tab. 2. Three representative unary models, IPCA [26], PCA [27] and CIE-H [31], and two pairwise models, qc-DGM [5] and our CLAP, are compared. Unary matching models learn to encode structure information by aggregating node descriptors. For all these five methods, we used node and edge attributes from the models pretrained on the Pascal VOC benchmark [4]. Tab. 2 shows the matching results on synthesized data. In this synthetic experiment, due to the significant rotation components involved, these learned/aggregated descriptors tend to be not discriminating enough to support reliable unary matching. Pairwise matching, in contrast, uses the explicit structure constraint, and turns out to be more reliable. In terms of

efficiency, although unary matching methods have simpler matching model to solve, they need to conduct (expensive) aggregation to encode structure information from node descriptors. This makes their overlap matching speed slower than our CLAP model.

4.2 Results of Pascal VOC Keypoints

Pascal VOC dataset [4] with Berkeley annotations of keypoints has 20 classes of instance images with keypoints. The training set includes 7,020 annotated images and the testing set includes 1,682 images. Our baseline qc-DGM model [5] has two settings: with and without a post-processing refinement, respectively denoted as qc-DGM and qc-DGM$^+$. Our baseline is based on qc-DGM without post-processing. For fair comparisons, we kept all the feature extraction and aggregation unchanged with the pretrained weights, and just replaced the KB-QAP objective function and its solver with our proposed algorithm.

We report the average runtime and accuracies in Tab. 3. Unary matching methods, such as LCSGM [29] and CIE-H [31], are highly reliant on the node attribute learning. They have to use extra embedding network to enhance the local features. Therefore, although without pairwise constraints, their matching is faster. Their feature learning component is slower. Lawler's QAP methods GMN [32] and NGM [28] have lower accuracy scores. Our CLAP model formulate the graph matching by L1 norm that is linear to solve. Therefore, it is both accurate and efficient.

Compared with the baseline qc-DGM, the proposed model achieves matching accuracy of 68.8% (baseline 68.7%) and 73.7ms computation time (baseline 116.4ms). Our model achieves similar accuracy but significantly improves the runtime efficiency. Note that here our model does not include a post-processing. Although qc-DGM$^+$, with post-processing refinement added to the baseline, achieves slightly better matching score, its computation speed is much slower. The results show that: (1) our positive semi-definite edge attribute matrices can successfully model structure information; (2) our CLAP model can greatly improve matching efficiency without losing accuracy.

5 Conclusion

This paper presents a new linear model for fast graph matching. We reformulated the pairwise graph matching as a concave maximization problem, which has a global maximum and can be solved efficiently. Specifically, we converted the pairwise structure constraint of KB-QAP into an L1 norm linear model. We showed that a common symmetric edge attribute matrix can be refined to become positive semi-definite to construct a linear structure constraint. Then, the problem can be solved using the Sinkhorn algorithm. Experiments showed that our method can achieve state-of-the-art performance and can greatly improve the computation speed of pairwise graph matching.

Limitations. Pointwise affinity depends on descriptor learning. When images undergo big global transformations or local deformation/transformation, descriptors can become unreliable and this could negatively impact matching accuracy. In the future, we will explore learning mechanisms to estimate confidence of local feature descriptors (*i.e.*, pointwise affinity), and increase structure constraints when necessary. Such a refined adaptive matching model could potentially improve the matching robustness in these challenging scenarios.

Acknowledgments. This research was partially supported by NSF CBET-2115405 and the Texas A&M University ASCEND Research Leadership Fellows Program. Part of the experiments were conducted using the computing resources provided by Texas A&M High Performance Research Computing.

References

1. Cuturi, M.: Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In: Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 26 (2013)
2. DeTone, D., Malisiewicz, T., Rabinovich, A.: Superpoint: Self-supervised interest point detection and description. In: CVPR workshops. pp. 224–236 (2018)
3. Ding, C., Zhou, D., He, X., Zha, H.: R 1-pca: rotational invariant l 1-norm principal component analysis for robust subspace factorization. In: ICML (2006)
4. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. IJCV **88**(2), 303–338 (2010)
5. Gao, Q., Wang, F., Xue, N., Yu, J., Xia, G.: Deep graph matching under quadratic constraint. In: CVPR. pp. 5067–5074 (2021)
6. He, J., Huang, Z., Wang, N., Zhang, Z.: Learnable graph matching: Incorporating graph partitioning with deep feature learning for multiple object tracking. In: CVPR. pp. 5299–5309 (2021)
7. Jaggi, M., Lacoste-Julien, S.: On the global linear convergence of frank-wolfe optimization variants. NeurIPS **28** (2015)
8. Jiang, B., Sun, P., Tang, J., Luo, B.: Glmnet: Graph learning-matching networks for feature matching. arXiv preprint arXiv:1911.07681 (2019)
9. Johnson, D.S., Garey, M.R.: Computers and intractability: A guide to the theory of NP-completeness. WH Freeman (1979)
10. Koopmans, T.C., Beckmann, M.: Assignment problems and the location of economic activities. Econometrica: journal of the Econometric Society (1957)
11. Kwak, N.: Principal component analysis based on l1-norm maximization. IEEE T-PAMI **30**(9), 1672–1680 (2008)
12. Lawler, E.L.: The quadratic assignment problem. Management science **9**(4) (1963)
13. Liao, X., Xu, Y., Ling, H.: Hypergraph neural networks for hypergraph matching. In: ICCV. pp. 1266–1275 (2021)
14. Lin, Y., Yang, M., Yu, J., Hu, P., Zhang, C., Peng, X.: Graph matching with bi-level noisy correspondence. In: ICCV. pp. 23362–23371 (2023)
15. Loiola, E.M., de Abreu, N.M.M., Boaventura-Netto, P.O., Hahn, P., Querido, T.: A survey for the quadratic assignment problem. European journal of operational research **176**(2), 657–690 (2007)
16. Lu, Y., Huang, K., Liu, C.L.: A fast projected fixed-point algorithm for large graph matching. Pattern Recognition **60**, 971–982 (2016)

17. Peyré, G., Cuturi, M., et al.: Computational optimal transport: With applications to data science. Foundations and Trends in Machine Learning **11**(5-6) (2019)
18. Puy, G., Boulch, A., Marlet, R.: Flot: Scene flow on point clouds guided by optimal transport. In: ECCV. pp. 527–544. Springer (2020)
19. Sarlin, P.E., DeTone, D., Malisiewicz, T., Rabinovich, A.: Superglue: Learning feature matching with graph neural networks. In: CVPR. pp. 4938–4947 (2020)
20. Shen, X., Wang, C., Li, X., Cheng, M., Yu, Z., Li, J., Wen, C., Cheng, M., He, Z.: Rf-net: An end-to-end image matching network based on receptive field. In: CVPR. pp. 8132–8140 (2019)
21. Sun, J., Shen, Z., Wang, Y., Bao, H., Zhou, X.: Loftr: Detector-free local feature matching with transformers. In: CVPR. pp. 8922–8931 (2021)
22. Umeyama, S.: An eigendecomposition approach to weighted graph matching problems. IEEE T-PAMI **10**(5), 695–703 (1988)
23. Varga, R.S.: Geršgorin and his circles. Springer Science & Business Media (2010)
24. Wang, F.D., Xue, N., Zhang, Y., Xia, G.S., Pelillo, M.: A Functional Representation for Graph Matching. IEEE T-PAMI **42**(11), 2737–2754 (Nov 2020)
25. Wang, R., Guo, Z., Jiang, S., Yang, X., Yan, J.: Deep learning of partial graph matching via differentiable top-k. In: CVPR. pp. 6272–6281 (2023)
26. Wang, R., Yan, J., Yang, X.: Learning combinatorial embedding networks for deep graph matching. In: ICCV. pp. 3056–3065 (2019)
27. Wang, R., Yan, J., Yang, X.: Combinatorial learning of robust deep graph matching: an embedding based approach. IEEE T-PAMI (2020)
28. Wang, R., Yan, J., Yang, X.: Neural graph matching network: Learning lawler's quadratic assignment problem with extension to hypergraph and multiple-graph matching. IEEE T-PAMI (2021)
29. Wang, T., Liu, H., Li, Y., Jin, Y., Hou, X., Ling, H.: Learning combinatorial solver for graph matching. In: CVPR. pp. 7568–7577 (2020)
30. Yi, K.M., Trulls, E., Lepetit, V., Fua, P.: Lift: Learned invariant feature transform. In: ECCV. pp. 467–483. Springer (2016)
31. Yu, T., Wang, R., Yan, J., Li, B.: Learning deep graph matching with channel-independent embedding and hungarian attention. In: ICLR (2019)
32. Zanfir, A., Sminchisescu, C.: Deep learning of graph matching. In: CVPR. pp. 2684–2693 (2018)
33. Zaslavskiy, M., Bach, F., Vert, J.P.: A path following algorithm for the graph matching problem. IEEE T-PAMI **31**(12), 2227–2242 (2008)
34. Zhang, Z., Xiang, Y., Wu, L., Xue, B., Nehorai, A.: Kergm: Kernelized graph matching. NeurIPS **32**, 3335–3346 (2019)
35. Zheng, W., Lin, Z., Wang, H.: L1-norm kernel discriminant analysis via bayes error bound optimization for robust feature extraction. IEEE transactions on neural networks and learning systems **25**(4), 793–805 (2013)
36. Zhou, F., De la Torre, F.: Factorized graph matching. In: CVPR. IEEE (2012)