Cross-order Consensus Graph Matching

Yu Xie[1] **Yiming Zhang**[1] **Xinyan Liang**[1] **Junqi Liu**[1] **Ming Li**[2,*]
[1]MoE Key Lab of Computational Intelligence and Chinese Information Processing
Shanxi University, Taiyuan 030006, China
[2]Zhejiang Key Laboratory of Intelligent Education Technology and Application
Zhejiang Normal University, Jinhua 321004, China
{yuxie,zhangyiming7,liangxinyan,liujunqi2}@sxu.edu.cn, mingli@zjnu.edu.cn

Abstract

Graph matching aims to establish node correspondences while preserving both unary appearance compatibility and pairwise structural consistency. Recent deep graph matching methods have achieved strong empirical performance, but many of them either compress edge information into node embeddings before matching or rely on surrogate optimization signals for quadratic objectives. These design choices can weaken structural discrimination and may lead to optimization directions that are not aligned with the relaxed quadratic assignment problem (QAP). We propose Cross-order Consensus Graph Matching (CCGM), a framework that explicitly couples first-order node assignment and second-order edge assignment. CCGM constructs line graphs to convert edge matching into a node matching problem, learns node and edge affinities in parallel, and refines them through a Cross-order Consensus Solver motivated by the exact gradient of the factorized relaxed QAP. We further introduce Cross-order Alignment Regularization to encourage agreement between node-induced edge assignments and edge-induced node support during training. Experiments on standard visual graph matching benchmarks show that CCGM consistently improves matching accuracy over competitive baselines.

1 Introduction

Graph matching (GM) [1] is a fundamental problem in machine learning that seeks reliable correspondences between the nodes of two graphs. It appears in a wide range of applications, including cross-domain knowledge transfer [2], unaligned multi-view clustering [3], federated graph learning [4], and egocentric task verification [5]. Classical methods usually construct an affinity matrix from hand-crafted descriptors and formulate matching as a Quadratic Assignment Problem (QAP) [6, 7]. Since QAP is NP-hard, a large body of earlier work focuses on approximate optimization, including graduated assignment, spectral matching, and projected fixed-point methods [8–11]. Deep graph matching methods extend this line of work by jointly learning visual representations and differentiable matching modules in an end-to-end pipeline [12]. Subsequent studies improve performance through stronger graph neural architectures [13–16] and more discriminative training objectives [17–19].

Despite this progress, current deep GM methods still face two closely related challenges. First, many architectures adopt a node-centric design in which edges primarily serve as message-passing links. Edge attributes or second-order structural cues are either discarded or aggregated into node embeddings before the final solver [13, 14, 20, 19, 21]. Such early aggregation is efficient, but it can discard information that is useful for disambiguating visually similar keypoints. Under the common abstraction that the solver observes only the aggregated representation $\tilde{\mathbf{X}}_N$, the inference path can be written as $(\mathbf{X}_N, \mathbf{X}_E) \to \tilde{\mathbf{X}}_N \to \mathbf{Y}$, where $\mathbf{Y}$ denotes the target correspondence. The data processing

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

inequality then gives

$$I(\tilde{\mathbf{X}}_N; \mathbf{Y}) \leq I((\mathbf{X}_N, \mathbf{X}_E); \mathbf{Y}), \tag{1}$$

which suggests that a solver operating only on compressed node features may have less information available for detecting structural inconsistencies. This limitation is particularly harmful when different keypoints have similar local appearances but different geometric neighborhoods.

Second, methods that retain quadratic structure often need to make a trade-off between faithful QAP modeling and differentiable training. Several approaches incorporate pairwise constraints by truncating continuous optimization, using association-graph message passing, or differentiating through black-box combinatorial solvers [15–17, 22–24]. These strategies are powerful, but their update directions can be biased relative to the analytical gradient of the relaxed QAP objective. When the optimization signal deviates from the objective that defines structural consistency, the learned matcher may fail to fully exploit second-order information.

We address these issues with *Cross-order Consensus Graph Matching* (CCGM). The main idea is to keep node-level and edge-level matching signals explicit and to make them mutually verifiable. CCGM first transforms each primal graph into a line graph so that edge matching can be formulated as a node matching problem in the line-graph domain. This design avoids prematurely collapsing edge information into node embeddings and enables direct learning of an edge-to-edge affinity matrix. We then derive the exact gradient of the factorized relaxed QAP and show that its second-order term naturally induces a cross-order consensus mechanism: node assignments project to edge assignments through incidence matrices, while edge compatibility projects back to node-level support. Motivated by this structure, our Cross-order Consensus Solver alternates between node-to-edge and edge-to-node projections and uses an entropy-aware gate to adaptively blend self-affinity and projected structural evidence. Finally, the Cross-order Alignment Regularization (CAR) loss penalizes disagreement between the two orders during training.

Our contributions are summarized as follows:

- We propose CCGM, a graph matching framework that explicitly maintains both node and edge affinity matrices. By constructing line graphs and path-aware edge features, CCGM preserves second-order visual and geometric information instead of relying only on aggregated node embeddings.
- We develop a Cross-order Consensus Solver that instantiates the incidence-projection structure of the relaxed QAP gradient. The solver refines node and edge affinities jointly and provides a principled mechanism for enforcing structural consistency.
- We provide a theoretical analysis of CCGM. We derive the exact cross-order gradient of the factorized relaxed QAP, formalize the incidence consistency between node and edge assignments, and establish a monotonic improvement property for projected gradient ascent under standard smoothness conditions.
- We introduce Cross-order Alignment Regularization, which encourages the node-induced edge distribution and the edge-induced node distribution to agree. Experiments on Pascal VOC, SPair-71k, and Willow Object Class demonstrate consistent improvements over strong graph matching baselines.

2 Related Work

Graph matching has long been studied through QAP-based formulations and approximate solvers, including graduated assignment, spectral matching, and projected fixed-point optimization [8–10]. Deep graph matching methods further combine representation learning with differentiable assignment modules: GMN [12] initiated an end-to-end learning paradigm, while PCA-GM [13], CIE [14], DGMC [25], GAMnet [20], ASAR [26], COMMON [19], and EQAN-U [27] improve matching through stronger node embeddings, consensus modeling, robustness mechanisms, or contrastive/objective-level designs. Another representative line keeps the quadratic structure more explicit: NGM and NGM-v2 [15] model Lawler's QAP through an association graph, QC-DGM [17] introduces differentiable quadratic-constraint optimization, and BBGM [16] differentiates through combinatorial solvers. In contrast to these approaches, CCGM explicitly maintains both node and edge assignment spaces, couples them through incidence-based cross-order projections, and regularizes their agreement during training. Due to page limitation, we provide the extended related work in **Appendix** A.

3 Preliminaries

3.1 Problem Definition

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{A})$ denote a graph, where $\mathcal{V} = \{v_1, \dots, v_n\}$ is the node set, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set, $\mathbf{X} \in \mathbb{R}^{n \times d}$ is the node feature matrix, and $\mathbf{A} \in \{0,1\}^{n \times n}$ is the adjacency matrix. Given a source graph $\mathcal{G}_1$ and a target graph $\mathcal{G}_2$ with n_1 and n_2 nodes, graph matching seeks a partial one-to-one correspondence encoded by a binary assignment matrix $\mathbf{P} \in \{0,1\}^{n_1 \times n_2}$. The feasible set is

$$\mathcal{P} = \left\{ \mathbf{P} \in \{0,1\}^{n_1 \times n_2} \mid \mathbf{P}\mathbf{1}_{n_2} \leq \mathbf{1}_{n_1},\ \mathbf{P}^\top \mathbf{1}_{n_1} \leq \mathbf{1}_{n_2} \right\}, \tag{2}$$

where $\mathbf{1}_n$ denotes the all-one vector of length n. In the Lawler QAP formulation, the vectorized assignment $\mathbf{x} = \mathrm{vec}(\mathbf{P}) \in \{0,1\}^{n_1 n_2}$ is optimized by

$$\max_{\mathbf{x}}\ J(\mathbf{x}) = \mathbf{x}^\top \mathbf{K} \mathbf{x}, \tag{3}$$

where $\mathbf{K} \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$ stores both node compatibilities and edge compatibilities. For convenience, a summary of the main mathematical notations is provided in **Appendix** B.

3.2 Factorized Relaxation

Directly optimizing Eq. (3) is difficult in neural architectures because the assignment is discrete and the full affinity matrix has $O(n_1^2 n_2^2)$ entries. Following the factorized graph matching formulation [28], we decompose the affinity into a node affinity matrix $\mathbf{K}_v \in \mathbb{R}^{n_1 \times n_2}$ and an edge affinity matrix $\mathbf{K}_e \in \mathbb{R}^{e_1 \times e_2}$. Let $\mathbf{G}_1 \in \{0,1\}^{n_1 \times e_1}$ and $\mathbf{G}_2 \in \{0,1\}^{n_2 \times e_2}$ be the node-edge incidence matrices of $\mathcal{G}_1$ and $\mathcal{G}_2$. The global affinity can be written as

$$\mathbf{K} = \mathrm{diag}(\mathrm{vec}(\mathbf{K}_v)) + (\mathbf{G}_2 \otimes \mathbf{G}_1)\, \mathrm{diag}(\mathrm{vec}(\mathbf{K}_e))(\mathbf{G}_2 \otimes \mathbf{G}_1)^\top, \tag{4}$$

where $\otimes$ denotes the Kronecker product. We relax $\mathbf{P}$ to a continuous assignment matrix $\mathbf{S} \in [0,1]^{n_1 \times n_2}$ and use the compact objective

$$\mathcal{J}(\mathbf{S}) = \mathrm{tr}(\mathbf{S}^\top \mathbf{K}_v) + \frac{1}{2}\,\mathrm{tr}\left[\mathbf{Y}^\top (\mathbf{K}_e \odot \mathbf{Y})\right], \quad \mathbf{Y} = \mathbf{G}_1^\top \mathbf{S} \mathbf{G}_2. \tag{5}$$

Here $\mathbf{Y} \in \mathbb{R}^{e_1 \times e_2}$ denotes the matrix of edge-assignment scores induced by the current node assignment, $\odot$ denotes the Hadamard product, and the factor $1/2$ removes an immaterial factor of two in the gradient. This relaxation is the mathematical basis for the solver and the theoretical analysis below.

4 Method

4.1 Overview

Figure 1 illustrates the CCGM pipeline. Given a pair of images, we extract keypoint features using a convolutional backbone and construct sparse geometric graphs by Delaunay triangulation. Each primal graph is then converted into a line graph, where every line-graph node corresponds to an edge in the primal graph. This representation allows CCGM to process node matching and edge matching as two coupled assignment problems. After attention-based feature refinement, CCGM computes initial node and edge affinities and refines them with the Cross-order Consensus Solver. The solver alternates between node-to-edge and edge-to-node projections, allowing each order to verify and correct the other. The final node and edge assignment distributions are trained with supervised matching losses and the proposed Cross-order Alignment Regularization loss.

4.2 Graph Construction and Line-graph Transformation

Primal graph construction. Given a source image I_s and a target image I_t, we use an ImageNet-pretrained VGG-16 backbone [29, 30] to extract dense visual feature maps. For each annotated keypoint, its local representation is obtained by bilinear interpolation from the `relu4_2` and `relu5_1` feature maps, followed by feature concatenation. These vectors form the initial node feature matrices

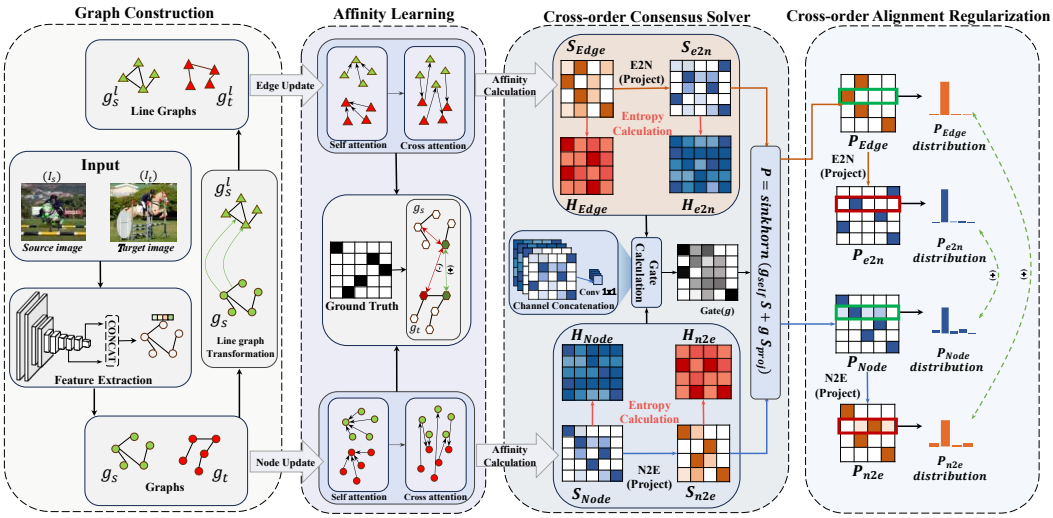

Figure 1: Overview of CCGM. Primal graphs are constructed from input images and transformed into line graphs. Node and edge features are refined through attention-based matching blocks to compute initial affinity matrices, which are then jointly updated by the Cross-order Consensus Solver. Training uses both matching supervision and Cross-order Alignment Regularization to encourage agreement between node-level and edge-level assignments.

$\mathbf{X}_s$ and $\mathbf{X}_t$. Graph connectivity is constructed using Delaunay triangulation over keypoint coordinates, providing a sparse geometric structure for pairwise reasoning.

Line-graph transformation. To retain second-order information explicitly, we transform each primal graph $\mathcal{G}$ into its line graph $\mathcal{L}(\mathcal{G}) = (\mathcal{V}_L, \mathcal{E}_L, \mathbf{X}_L, \mathbf{A}_L)$. A primal edge $e_k = (v_i, v_j)$ becomes a line-graph node v_k^L. Therefore, matching edges across the two primal graphs becomes a standard node matching problem across the two line graphs.

The feature of a line-graph node should describe both the endpoints and the visual content along the edge. Prior edge-aware graph matching methods, such as EAGM [31], aggregate visual evidence along primal edges, highlighting the value of explicit edge representations. Endpoint concatenation provides a strong topological prior, but it may miss the appearance pattern between the keypoints. Conversely, path sampling captures intermediate visual evidence but may dilute endpoint semantics. We therefore combine both sources. For edge $e_k = (v_i, v_j)$, the endpoint representation is

$$\mathbf{h}_{\text{end}}^{(k)} = \text{MLP}_{\text{end}}\left([\mathbf{x}_i \parallel \mathbf{x}_j]\right), \tag{6}$$

and the path representation is obtained by bilinear sampling along the segment from coordinate $\mathbf{p}_i$ to coordinate $\mathbf{p}_j$ on the backbone feature map $\mathbf{F}_{\text{vgg}}$:

$$\mathbf{h}_{\text{path}}^{(k)} = \text{PathSampling}(\mathbf{F}_{\text{vgg}}, \mathbf{p}_i, \mathbf{p}_j). \tag{7}$$

The final edge feature is

$$\mathbf{x}_k^L = \text{MLP}_{\text{fusion}}\left([\mathbf{h}_{\text{end}}^{(k)} \parallel \mathbf{h}_{\text{path}}^{(k)}]\right). \tag{8}$$

This design preserves the identity of the two endpoint keypoints while incorporating ordered visual evidence along the connecting segment.

Line-graph connectivity follows the incidence relation of primal edges: two line-graph nodes are adjacent if their corresponding primal edges share exactly one endpoint,

$$\mathbf{A}_L(k, q) = \mathbb{I}(|e_k \cap e_q| = 1). \tag{9}$$

Because Delaunay triangulation produces a planar graph, the number of primal edges is at most $3n - 6$ for $n \geq 3$. Thus, the number of line-graph nodes grows linearly with the number of keypoints. This use of line graphs is also consistent with classical results showing that line-graph representations can be handled efficiently in graph algorithms [32]. The number of line-graph edges depends on local degrees; in the sparse keypoint graphs used in visual matching, this remains efficient in practice.

4.3 Affinity Learning

Accurate correspondence requires both intra-graph context and inter-graph comparison. We therefore stack L Transformer-style matching blocks [33, 34]. Each block first applies self-attention within each graph to model internal structure and then applies cross-attention across the source and target graphs to exchange matching cues. For node features at layer l, the update is

$$\begin{aligned} \hat{\mathbf{X}}_s &= \mathrm{SelfAttn}(\mathbf{X}_s^{(l)}), & \hat{\mathbf{X}}_t &= \mathrm{SelfAttn}(\mathbf{X}_t^{(l)}), \\ \mathbf{X}_s^{(l+1)} &= \mathbf{X}_s^{(l)} + \mathrm{CrossAttn}(\hat{\mathbf{X}}_s, \hat{\mathbf{X}}_t), & \mathbf{X}_t^{(l+1)} &= \mathbf{X}_t^{(l)} + \mathrm{CrossAttn}(\hat{\mathbf{X}}_t, \hat{\mathbf{X}}_s). \end{aligned} \tag{10}$$

The same update is applied to the line-graph stream. After the final layer, let $\mathbf{H}_s, \mathbf{H}_t \in \mathbb{R}^{n\times d}$ denote the refined primal node embeddings and let $\mathbf{L}_s, \mathbf{L}_t \in \mathbb{R}^{m\times d}$ denote the refined line-graph node embeddings. Initial affinities are computed by inner products:

$$\mathbf{S}_{\mathrm{node}} = \mathbf{H}_s \mathbf{\Lambda} \mathbf{H}_t^\top, \qquad \mathbf{S}_{\mathrm{edge}} = \mathbf{L}_s \mathbf{\Lambda} \mathbf{L}_t^\top, \tag{11}$$

where $\mathbf{\Lambda} \in \mathbb{R}^{d\times d}$ is a dynamic diagonal affinity metric generated by a learnable projection from the global context.

4.4 Cross-order Consensus Solver

The Cross-order Consensus Solver refines $\mathbf{S}_{\mathrm{node}}$ and $\mathbf{S}_{\mathrm{edge}}$ jointly. It is motivated by the exact gradient of Eq. (5), which is formalized in Theorem 1. The key observation is that the second-order gradient has two incidence projections: node assignments are first projected to edge space, and edge evidence is then projected back to node space. Rather than treating edge information as an intermediate message that is immediately discarded, CCGM maintains edge affinities as a persistent state throughout the solver. The pseudocode for the solver is provided in **Appendix** C.

4.4.1 Cross-order Projection

At iteration t, the solver computes a node-to-edge (N2E) projection and an edge-to-node (E2N) projection.

Node-to-edge projection. The N2E projection estimates whether two primal edges are compatible under the current node affinity. For undirected edges $e_k = (u, v)$ and $e_l = (p, q)$, the two endpoint orientations are equivalent. We therefore use the orientation-invariant score

$$\mathbf{S}_{\mathrm{n2e}}^{(t)}(k, l) = \max\left(\mathbf{S}_{\mathrm{node}}^{(t)}(u, p) + \mathbf{S}_{\mathrm{node}}^{(t)}(v, q),\ \mathbf{S}_{\mathrm{node}}^{(t)}(u, q) + \mathbf{S}_{\mathrm{node}}^{(t)}(v, p)\right). \tag{12}$$

This projection is a continuous counterpart of the incidence consistency criterion in Proposition 1. It assigns a high edge score only when the two endpoint correspondences are jointly supported.

Edge-to-node projection. The E2N projection aggregates edge-level evidence back to node pairs. The exact gradient contains the term $\mathbf{G}_1(\mathbf{K}_e \odot \mathbf{Y})\mathbf{G}_2^\top$; in the network, $\mathbf{S}_{\mathrm{edge}}^{(t)}$ plays the role of learned edge compatibility modulated by the current edge state. A direct summation over incident edges can overemphasize high-degree nodes. We therefore use degree-normalized LogSumExp aggregation:

$$\mathbf{S}_{\mathrm{e2n}}^{(t)} = \tau \log\left(\mathbf{G}_1 \exp\left(\frac{\mathbf{S}_{\mathrm{edge}}^{(t)}}{\tau}\right)\mathbf{G}_2^\top\right) - \tau \log(\mathbf{D}_1 \mathbf{D}_2^\top), \tag{13}$$

where τ is a temperature and $\mathbf{D}_1, \mathbf{D}_2$ are node-degree vectors. LogSumExp smoothly interpolates between averaging and maximum selection, allowing strong edge evidence to support node pairs while reducing degree-induced scale variation.

The N2E and E2N operators are structurally motivated by the incidence-mediated gradient in Theorem 1, but they are not algebraically identical to the exact gradient. These design choices retain the cross-order information flow of the relaxed QAP while allowing a learnable approximation suited to neural graph matching.

4.4.2 Entropy-aware Gated Update

A fixed step size is often too rigid for visual graph matching because confidence varies across nodes, edges, and iterations. We therefore introduce an entropy-aware gate as an adaptive, element-wise analogue of the gradient step size. Specifically, we employ four independent lightweight gating networks, each consisting of a 1×1 convolution followed by a sigmoid activation, to adaptively balance the contributions of the current and cross-order projected affinities in the node and edge streams. To characterize matching confidence, we define the row-wise sharpness of an affinity matrix $\mathbf{S}$ as

$$\mathbf{H}_i(\mathbf{S}) = 1 + \frac{1}{\log N} \sum_j \mathbf{P}_{ij} \log \mathbf{P}_{ij}, \qquad \mathbf{P}_{ij} = \mathrm{Softmax}(\mathbf{S}_{i\cdot})_j, \tag{14}$$

where larger values indicate more concentrated matching distributions. The gating networks take the current affinity, its cross-order projection, and the sharpness map as inputs, and produce adaptive gating weights $(\mathbf{g}^v_{\text{self}}, \mathbf{g}^v_{\text{proj}})$ for nodes and $(\mathbf{g}^e_{\text{self}}, \mathbf{g}^e_{\text{proj}})$ for edges. The affinities are then updated as

$$\mathbf{S}^{(t+1)}_{\text{node}} = \mathbf{g}^v_{\text{self}} \odot \mathbf{S}^{(t)}_{\text{node}} + \mathbf{g}^v_{\text{proj}} \odot \mathbf{S}^{(t)}_{\text{e2n}}, \quad \mathbf{S}^{(t+1)}_{\text{edge}} = \mathbf{g}^e_{\text{self}} \odot \mathbf{S}^{(t)}_{\text{edge}} + \mathbf{g}^e_{\text{proj}} \odot \mathbf{S}^{(t)}_{\text{n2e}}. \tag{15}$$

After T iterations, the refined scores are normalized using Sinkhorn [35, 36]:

$$\mathbf{P}_{\text{node}} = \mathrm{Sinkhorn}(\mathbf{S}^{(T)}_{\text{node}}), \qquad \mathbf{P}_{\text{edge}} = \mathrm{Sinkhorn}(\mathbf{S}^{(T)}_{\text{edge}}). \tag{16}$$

4.5 Training Objective

The training objective contains supervised matching losses and a cross-order regularizer. Let $\pi^* : \mathcal{V}_1 \to \mathcal{V}_2$ denote the ground-truth node correspondence and let π^{*-1} denote its inverse on matched target nodes. For the node assignment distribution $\mathbf{P}_{\text{node}}$, the source-to-target InfoNCE loss is

$$\mathcal{L}^{\text{node}}_{S\to T} = -\frac{1}{|\mathcal{V}_1|} \sum_{i\in\mathcal{V}_1} \log \frac{\exp(\mathbf{P}_{\text{node}}(i, \pi^*(i))/\tau_m)}{\sum_{j\in\mathcal{V}_2} \exp(\mathbf{P}_{\text{node}}(i,j)/\tau_m)}, \tag{17}$$

where $\tau_m = 0.07$. The target-to-source loss is defined symmetrically, and the node matching loss is

$$\mathcal{L}^{\text{node}}_{\text{match}} = \frac{1}{2} \left(\mathcal{L}^{\text{node}}_{S\to T} + \mathcal{L}^{\text{node}}_{T\to S} \right). \tag{18}$$

The edge matching loss $\mathcal{L}^{\text{edge}}_{\text{match}}$ is computed in the same way using the ground-truth edge correspondence induced by π^*. The supervised matching loss is

$$\mathcal{L}_{\text{match}} = \frac{1}{2} \left(\mathcal{L}^{\text{node}}_{\text{match}} + \mathcal{L}^{\text{edge}}_{\text{match}} \right). \tag{19}$$

In practice, the matching loss $\mathcal{L}_{\text{match}}$ is computed as intermediate deep supervision across the matching blocks (with decaying weights) to facilitate stable gradient flow and faster convergence. To encourage the two orders to agree, CAR aligns each final assignment distribution with the corresponding cross-order projection:

$$\begin{aligned} \mathcal{L}^{\text{node}}_{\text{CAR}} &= -\frac{1}{2|\mathcal{V}|} \sum \mathbf{P}_{\text{node}} \odot \left[\log \sigma_{\text{row}} \left(\frac{\mathbf{S}_{\text{e2n}}}{\tau_c} \right) + \log \sigma_{\text{col}} \left(\frac{\mathbf{S}_{\text{e2n}}}{\tau_c} \right) \right], \\ \mathcal{L}^{\text{edge}}_{\text{CAR}} &= -\frac{1}{2|\mathcal{E}|} \sum \mathbf{P}_{\text{edge}} \odot \left[\log \sigma_{\text{row}} \left(\frac{\mathbf{S}_{\text{n2e}}}{\tau_c} \right) + \log \sigma_{\text{col}} \left(\frac{\mathbf{S}_{\text{n2e}}}{\tau_c} \right) \right]. \end{aligned} \tag{20}$$

The final objective is

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{match}} + \lambda \left(\mathcal{L}^{\text{node}}_{\text{CAR}} + \mathcal{L}^{\text{edge}}_{\text{CAR}} \right), \tag{21}$$

where λ balances supervised matching and cross-order regularization. The computational complexity analysis of the proposed method can be found in **Appendix** D.

5 Theoretical Analysis

This section provides formal support for the cross-order design. Proofs are deferred to **Appendix** E.

5.1 Exact Gradient of the Factorized Relaxed QAP

The relaxed objective in Eq. (5) explicitly separates first-order node affinity and second-order edge affinity. Its gradient has the same cross-order form used by the solver.

Theorem 1 (**Exact cross-order gradient**). *Let $\mathcal{J}(\mathbf{S})$ be defined as in Eq.* (5)*, with $\mathbf{Y} = \mathbf{G}_1^\top \mathbf{S} \mathbf{G}_2$. Then*

$$\nabla_{\mathbf{S}} \mathcal{J}(\mathbf{S}) = \mathbf{K}_v + \mathbf{G}_1 \left(\mathbf{K}_e \odot (\mathbf{G}_1^\top \mathbf{S} \mathbf{G}_2) \right) \mathbf{G}_2^\top. \tag{22}$$

Theorem 1 shows that a structurally faithful update cannot be computed from node scores alone. It requires a node-to-edge projection $\mathbf{G}_1^\top \mathbf{S} \mathbf{G}_2$, compatibility weighting by $\mathbf{K}_e$, and an edge-to-node projection through $\mathbf{G}_1$ and $\mathbf{G}_2^\top$. This is precisely the cross-order computation approximated by Eqs. (12) and (13).

5.2 Incidence Consistency Between Node and Edge Assignments

The next result formalizes when an edge correspondence is consistent with a node correspondence.

Proposition 1 (**Incidence consistency**). *Let $\mathbf{P} \in \mathcal{P}$ be a binary partial matching. For edges $e_k = (u, v) \in \mathcal{E}_1$ and $e_l = (p, q) \in \mathcal{E}_2$, let $\mathbf{g}_{1,k}$ and $\mathbf{g}_{2,l}$ be their incidence vectors and define*

$$c_{kl} = \mathbf{g}_{1,k}^\top \mathbf{P} \mathbf{g}_{2,l}. \tag{23}$$

Then $c_{kl} \in \{0, 1, 2\}$ and counts how many endpoints of e_k are matched by $\mathbf{P}$ to endpoints of e_l. Moreover, $c_{kl} = 2$ if and only if $\mathbf{P}$ maps the unordered endpoint set $\{u, v\}$ bijectively to $\{p, q\}$. Equivalently, the orientation-invariant score

$$q_{kl}(\mathbf{P}) = \max(P_{up} + P_{vq},\ P_{uq} + P_{vp}) \tag{24}$$

equals 2 if and only if the edge match (e_k, e_l) is induced by the node match $\mathbf{P}$, and is at most 1 otherwise.

Proposition 1 gives a discrete interpretation of the N2E projection. For binary assignments, Eq. (12) is exact up to the constant scale of two; for relaxed assignments, it provides a differentiable confidence score for whether an edge correspondence is supported by its endpoint correspondences.

5.3 Monotonicity of Exact Projected Gradient Updates

The final result clarifies why aligning the solver with the exact gradient is useful. It does not claim that the learned gate always performs a mathematically exact gradient step. Rather, it shows that exact projected gradient ascent has a monotonic improvement property under standard smoothness conditions, while proxy directions generally do not inherit this guarantee for the same objective.

Theorem 2 (**Monotonicity under smoothness**). *Let $\mathcal{D} \subset \mathbb{R}^{n_1 \times n_2}$ be a closed convex relaxation of the feasible matching set, and assume that $\mathcal{J}$ has an L-Lipschitz continuous gradient on $\mathcal{D}$. For any $\mathbf{S} \in \mathcal{D}$, define the projected gradient ascent step*

$$\mathbf{S}^+ = \Pi_{\mathcal{D}} \left(\mathbf{S} + \eta \nabla \mathcal{J}(\mathbf{S}) \right), \tag{25}$$

where $\Pi_{\mathcal{D}}$ is Euclidean projection and $0 < \eta \leq 2/L$. Then

$$\mathcal{J}(\mathbf{S}^+) \geq \mathcal{J}(\mathbf{S}) + \left(\frac{1}{\eta} - \frac{L}{2} \right) \|\mathbf{S}^+ - \mathbf{S}\|_F^2. \tag{26}$$

In particular, $\mathcal{J}(\mathbf{S}^+) \geq \mathcal{J}(\mathbf{S})$. For the objective in Eq. (5)*, one valid smoothness bound is*

$$L \leq \|\mathbf{K}_e\|_\infty \|\mathbf{G}_1\|_2^2 \|\mathbf{G}_2\|_2^2. \tag{27}$$

This theorem motivates our design choice: the solver should preserve the incidence-projection structure of the exact QAP gradient, even when the actual update is implemented by a learnable gate for better empirical flexibility.

6 Experiments

We evaluate CCGM on three standard visual graph matching benchmarks: Pascal VOC with Berkeley annotations [37, 38], Willow Object Class [39], and SPair-71k [40]. We compare against representative deep graph matching methods, including PCA [13], GMN [12], NGM and NGM-v2 [15], CIE [14], BBGM [16], QC-DGM [17], DGMC [25], GAMnet [20], ASAR [26], COMMON [19], and EQAN-U [27]. Additional experimental results can be found in **Appendix** F.

6.1 Experimental Settings

We follow the standard evaluation protocol used by prior work [15]. Images are resized and cropped to 256×256. Initial visual features are extracted by an ImageNet-pretrained VGG-16 backbone [29, 30] and further encoded using SplineCNN [41]. The affinity learner uses $L = 4$ matching blocks, and the Cross-order Consensus Solver is unrolled for $T = 4$ iterations. The sigmoid gates independently produce weights in $[0, 1]$, with the self-affinity and projection gate biases initialized to 2.0 and 0.0, respectively. The CAR weight λ is linearly increased from 0 to 0.5 during the first three epochs. We optimize the model with Adam [42] using an initial learning rate of 10^{-3}, halved four times at regular intervals. The backbone learning rate is scaled by 0.01, and the batch size is set to 8. All experiments are implemented in PyTorch 2.0.1 and conducted on a Linux server equipped with an Intel Xeon Platinum 8468V CPU, 1 TB of RAM, a 7 TB SSD, and an NVIDIA H100 PCIe GPU (80 GB VRAM).

6.2 Results and Discussion

Table 1: Experimental results on the Pascal VOC dataset. The best and second-best results are highlighted in **bold** and underlined, respectively.

Method	Aero	Bike	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table	Dog	Horse	Mbike	Person	Plant	Sheep	Sofa	Train	Tv	Mean
PCA [13]	49.8	61.9	65.3	57.2	78.8	75.6	64.7	69.7	41.6	63.4	50.7	67.1	66.7	61.6	44.5	81.2	67.8	59.2	78.5	90.4	64.8
GMN [12]	41.6	59.6	60.3	48.0	79.2	70.2	67.4	64.9	39.2	61.3	66.9	59.8	61.1	59.8	37.2	78.2	68.0	49.9	84.2	91.4	62.4
NGM [15]	50.1	63.5	57.9	53.4	79.8	77.1	73.6	68.2	41.1	66.4	40.8	60.3	61.9	63.5	45.6	77.1	69.3	65.5	79.2	88.2	64.1
CIE [14]	52.5	68.6	70.2	57.1	82.1	77.0	70.7	73.1	43.8	69.9	62.4	70.2	70.3	66.4	47.6	85.3	71.7	64.0	83.9	91.7	68.9
BBGM [16]	61.9	71.1	79.7	79.0	87.4	94.0	89.5	80.2	56.8	79.1	64.6	78.9	76.2	75.1	65.2	98.2	77.3	77.0	94.9	93.9	79.0
QC-DGM [17]	49.6	64.6	67.1	62.4	82.1	79.9	74.8	73.5	43.0	68.4	66.5	67.2	71.4	70.1	48.6	92.4	69.2	70.9	90.9	92.0	70.3
GAMnet [20]	62.4	74.9	76.8	77.1	86.6	94.2	89.7	80.4	57.8	77.6	93.2	78.5	79.2	80.7	65.0	97.6	76.0	76.2	96.9	94.2	80.7
DGMC [25]	50.4	67.6	70.7	70.5	87.2	85.2	82.5	74.3	46.2	69.4	69.9	73.9	73.8	65.4	51.6	98.0	73.2	69.6	94.3	89.6	73.2
NGM-v2 [15]	61.8	71.2	77.6	78.8	87.3	93.6	87.7	79.8	55.4	77.8	89.5	78.8	80.1	79.2	62.6	97.7	77.7	75.7	96.7	93.2	80.1
ASAR [26]	62.9	74.3	79.5	**80.1**	89.2	94.0	88.9	78.9	58.8	79.8	88.2	78.9	79.5	77.9	64.9	98.2	77.5	77.1	98.6	93.7	81.1
COMMON [19]	65.6	75.2	80.8	79.5	89.3	92.3	90.1	81.8	61.6	80.7	95.0	82.0	81.6	79.5	66.6	**98.9**	78.9	80.9	99.3	93.8	82.7
EQAN-U [27]	68.5	76.9	81.9	71.2	89.1	91.3	82.0	82.9	52.0	70.9	**100.0**	80.9	78.8	79.3	66.7	95.7	**87.4**	65.7	**100.0**	**96.8**	80.9
CCGM	**72.2**	**81.9**	**85.0**	79.4	**92.9**	**95.0**	**91.1**	**84.8**	**75.8**	**84.8**	**100.0**	**85.8**	**83.9**	**82.2**	**75.3**	97.9	83.5	**83.4**	98.5	95.4	**86.4**

Table 2: Experimental results on the SPair-71k dataset.

Method	Aero	Bike	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Dog	Horse	Mbike	Person	Plant	Sheep	Train	Tv	Mean
PCA [13]	64.7	45.7	78.1	51.3	63.8	72.7	61.2	62.8	62.6	68.2	59.1	61.2	64.9	57.7	87.4	60.4	72.5	92.8	66.0
GMN [12]	59.9	51.0	74.3	46.7	63.3	75.5	69.5	64.6	57.5	73.0	58.7	59.1	63.2	51.2	86.9	57.9	70.0	92.4	65.3
NGM [15]	66.4	52.6	77.0	49.6	67.7	78.8	67.6	68.3	59.2	73.6	63.9	60.7	70.7	60.9	87.5	63.9	79.8	91.5	68.9
CIE [14]	71.5	57.1	81.7	56.7	67.9	82.5	73.4	74.5	62.6	78.0	68.7	66.3	73.7	66.0	92.5	67.2	82.3	97.5	73.3
BBGM [16]	75.3	65.0	87.6	78.0	69.8	94.0	87.8	78.3	72.8	82.7	76.6	76.3	80.1	75.0	98.7	85.2	96.3	98.0	82.1
NGM-v2 [15]	68.8	63.3	86.8	70.1	69.7	94.7	87.4	77.4	72.1	80.7	74.3	72.5	79.5	73.4	98.9	81.2	94.3	98.7	80.2
ASAR [26]	72.4	61.8	91.8	79.1	71.2	97.4	90.4	78.3	74.2	83.1	77.3	77.0	83.1	76.4	99.5	85.2	97.8	99.5	83.1
COMMON [19]	77.3	68.2	92.0	79.5	70.4	97.5	91.6	82.5	72.2	88.0	80.0	74.1	**83.4**	82.8	99.9	84.4	**98.2**	99.8	84.5
EQAN-U [27]	65.5	65.1	81.1	**84.2**	**71.7**	97.7	**93.4**	78.1	67.7	80.1	72.8	74.3	83.2	67.3	**100**	75.7	93.9	**100**	80.7
CCGM	**79.4**	**70.1**	**92.5**	83.0	70.8	**98.2**	91.8	**82.9**	**79.3**	**88.3**	**82.1**	**79.8**	81.7	**84.2**	99.5	**86.8**	96.9	99.8	**86.0**

Evaluation on the Pascal VOC Dataset. Pascal VOC with Berkeley annotations [38, 37] contains 7,020 training and 1,682 testing images across 20 object categories. Images contain 6–23 annotated keypoints and exhibit substantial intra-class variation, pose/scale changes, and occlusion. As shown in Table 1, CCGM achieves 86.4% average accuracy, outperforming the strongest baseline, COMMON [19], by 3.7 points. Gains on structurally ambiguous categories such as Chair and Table highlight the effectiveness of explicit edge-level reasoning for structural disambiguation.

Evaluation on the SPair-71k Dataset. SPair-71k [40] is a large-scale benchmark with 70,958 image pairs across 18 categories. It is more challenging than Pascal VOC, with stronger viewpoint changes,

Table 3: Experimental results on the Willow Object dataset.

Method	Car	Duck	Face	Mbike	Wbottle	Mean
PCA [13]	87.6	83.6	**100**	77.6	88.4	87.4
GMN [12]	67.9	76.7	99.8	69.2	83.1	79.3
NGM [15]	84.2	77.6	99.4	76.8	88.3	85.3
CIE [14]	85.8	82.1	99.9	88.4	88.7	89.0
BBGM [16]	96.8	89.9	**100**	99.8	99.4	97.2
DGMC [25]	98.3	90.2	**100**	98.5	98.1	97.0
NGM-v2 [15]	97.4	93.4	**100**	98.6	98.3	97.5
QC-DGM [17]	98.0	92.8	**100**	98.8	99.0	97.7
GAMnet [20]	96.7	90.1	**100**	96.9	99.4	96.6
ASAR [26]	92.5	84.0	**100**	95.4	99.0	94.2
COMMON [19]	97.6	**98.7**	**100**	**100**	99.0	**99.1**
CCGM	**98.7**	96.6	**100**	**100**	**100.0**	**99.1**

scale variation, and occlusion. Table 2 shows that CCGM achieves 86.0% mean accuracy, outperforming COMMON (84.5%) by 1.5 points. Per-category results indicate that cross-order consensus is helpful when local appearance alone is insufficient under large geometric transformations.

Evaluation on the Willow Object Dataset. The Willow Object dataset [39] contains 256 images from 5 categories, with 10 annotated keypoints per image. As shown in Table 3, CCGM achieves 99.1% mean accuracy. Although performance on this dataset is highly saturated, CCGM remains competitive with the best methods, outperforming NGM-v2 [15] and matching COMMON [19] in mean accuracy. The strongest results appear on rigid categories such as Car and Wbottle, while Duck remains challenging because of non-rigid articulation and limited sample size.

6.3 Robustness Analysis

We evaluate robustness from two perspectives: viewpoint shifts and synthetic outliers. Table 4 reports SPair-71k results under different viewpoint difficulty levels. CCGM achieves the best overall accuracy and its largest gain on the Hard split, suggesting that explicit second-order structure is particularly beneficial when viewpoint changes make local node features unreliable. We further test outlier robustness by adding 0–10 spurious keypoints during training and evaluation. Figure 2 shows that CCGM degrades more slowly than the compared baselines as outliers increase. With 10 outliers, CCGM still maintains approximately 93% accuracy, indicating that cross-order verification suppresses structurally inconsistent matches under noisy correspondence settings.

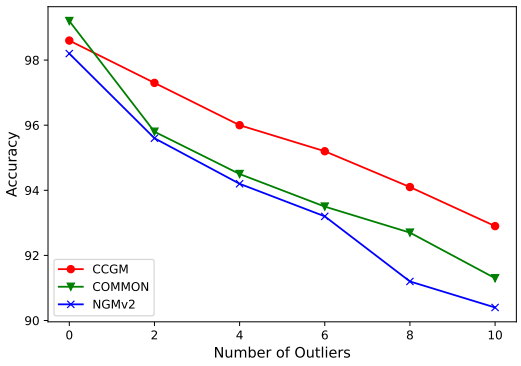

Figure 2: Performance comparison among CCGM, NGM-v2, and COMMON with varying outlier numbers.

6.4 Ablation Studies

Table 5 isolates the contribution of each component. Introducing the Cross-order Consensus Solver improves accuracy from 84.9% to 85.8%, showing that iterative node-edge verification improves the quality of the assignment. Adding line-graph construction further increases accuracy to 86.0%, indicating that explicit edge features provide useful second-order cues. The full model, including CAR, obtains 86.4%. This final gain suggests that directly regularizing agreement between node-induced and edge-induced distributions makes the learned correspondences more structurally reliable. The monotonic improvement across rows further indicates that the three modules are complementary rather than redundant. In particular, the solver refines node and edge affinities during inference, whereas CAR shapes the training objective toward cross-order consistent solutions.

Table 4: Matching accuracy on SPair-71k under different viewpoint difficulties.

Method	Viewpoint difficulty			All
	Easy	Medium	Hard	
DGMC [25]	79.4	65.2	61.3	72.2
BBGM [16]	84.8	73.1	70.6	78.9
ASAR [26]	86.5	79.1	72.5	83.1
COMMON [19]	86.6	81.4	76.4	84.5
CCGM	**87.8** (+1.2)	**82.1** (+0.7)	**80.3** (+3.9)	**86.0** (+1.5)

Table 5: Ablation study of the proposed components on Pascal VOC.

Cross-order Consensus Solver	Line Graph Construction	Cross-order Alignment Regularization	Pascal VOC
✓	✓	✓	**86.4**
✓	✓	×	86.0
✓	×	×	85.8
×	×	×	84.9

7 Conclusion

We introduced CCGM, a cross-order graph matching framework that explicitly models both node and edge assignments. By transforming primal edges into line-graph nodes, CCGM preserves second-order information that is often compressed away in node-centric pipelines. The Cross-order Consensus Solver refines node and edge affinities through incidence-based projections motivated by the exact gradient of the factorized relaxed QAP, and CAR further aligns the two assignment distributions during training. Theoretical analysis establishes the exact cross-order gradient, the discrete consistency condition between node and edge assignments, and a monotonicity property for exact projected gradient ascent. Experiments on standard visual graph matching benchmarks demonstrate consistent improvements in accuracy and robustness. Beyond aggregate performance, the ablation and robustness studies show that line-graph modeling, cross-order refinement, and CAR provide complementary benefits under structural ambiguity and noisy correspondences. These results suggest that preserving assignment structure at both the node and edge levels is an effective way to bridge efficient neural matching with classical quadratic matching principles. Future work may extend this cross-order formulation to higher-order geometric relations and multi-graph matching.

Acknowledgments and Disclosure of Funding

This work was supported in part by the National Natural Science Foundation of China (No. 62472270, 62622615, 62676230), and the "Pioneer" and "Leading Goose" R&D Program of Zhejiang (No. 2024C03262).

We declare no competing interests.

References

[1] Yu Xie, Lianhang Luo, Tianpei Cao, Bin Yu, and A. K. Qin. Contrastive learning network for unsupervised graph matching. *IEEE Transactions on Circuits and Systems for Video Technology*, 35(1):643–656, 2025.

[2] Gengzhi Zhang, Liang Feng, Xuefeng Chen, Ke Tang, and Kay Chen Tan. Enhancing reinforcement learning with cross-domain knowledge transfer via seeded graph matching. *IEEE Transactions on Neural Networks and Learning Systems*, 37(1):371–385, 2026.

[3] Zhibin Dong, Shengju Yu, Siwei Wang, Xinwang Liu, and En Zhu. Cross-view graph matching for unsupervised learning with unaligned multi-view clustering. *IEEE Transactions on Knowledge and Data Engineering*, 38(4):2221–2233, 2026.

[4] Sheng Huang, Lele Fu, Tianchi Liao, Bowen Deng, Chuanfu Zhang, and Chuan Chen. FedG2: Cross-domain federated graph learning via dual graph matching. *Pattern Recognition*, 179:113565, 2026.

[5] Xun Jiang, Xing Xu, Zheng Wang, Jingkuan Song, Fumin Shen, and Heng Tao Shen. Generalizable egocentric task verification via cross-modal hybrid hypergraph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(6):6261–6278, 2026.

[6] Eugene L Lawler. The quadratic assignment problem. *Management Science*, 9(4):586–599, 1963.

[7] Eliane Maria Loiola, Nair Maria Maia De Abreu, Paulo Oswaldo Boaventura-Netto, Peter Hahn, and Tania Querido. A survey for the quadratic assignment problem. *European Journal of Operational Research*, 176(2):657–690, 2007.

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

[9] Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1482–1489, 2005.

[10] Marius Leordeanu, Martial Hebert, and Rahul Sukthankar. An integer projected fixed point method for graph matching and MAP inference. In *Advances in Neural Information Processing Systems*, pages 1114–1122, 2009.

[11] Binrui Shen, Yuan Liang, and Shengxin Zhu. FRAM: Frobenius-regularized assignment matching with mixed-precision computing. In *Advances in Neural Information Processing Systems*, pages 112978–113005, 2025.

[12] Andrei Zanfir and Cristian Sminchisescu. Deep learning of graph matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2684–2693, 2018.

[13] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Learning combinatorial embedding networks for deep graph matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3056–3065, 2019.

[14] Tianshu Yu, Runzhong Wang, Junchi Yan, and Baoxin Li. Learning deep graph matching with channel-independent embedding and hungarian attention. In *Proceedings of the International Conference on Learning Representations*, 2020.

[15] Runzhong Wang, Junchi Yan, and Xiaokang Yang. 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*, 44(9):5261–5279, 2022.

[16] Michal Rolínek, Paul Swoboda, Dominik Zietlow, Anselm Paulus, Vít Musil, and Georg Martius. Deep graph matching via blackbox differentiation of combinatorial solvers. In *Proceedings of the European Conference on Computer Vision*, pages 407–424, 2020.

[17] Quankai Gao, Fudong Wang, Nan Xue, Jin-Gang Yu, and Gui-Song Xia. Deep graph matching under quadratic constraint. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5069–5078, 2021.

[18] Gathika Ratnayaka, Qing Wang, and Yang Li. Contrastive learning for supervised graph matching. In *Uncertainty in Artificial Intelligence*, pages 1718–1729, 2023.

[19] Yijie Lin, Mouxing Yang, Jun Yu, Peng Hu, Changqing Zhang, and Xi Peng. Graph matching with bi-level noisy correspondence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23362–23371, 2023.

[20] Bo Jiang, Pengfei Sun, Ziyan Zhang, Jin Tang, and Bin Luo. GAMNet: Robust feature matching via graph adversarial-matching network. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 5419–5426, 2021.

[21] Gathika Ratnayaka, James Nichols, and Qing Wang. Learning partial graph matching via optimal partial transport. In *Proceedings of the International Conference on Learning Representations*, 2025.

[22] Jinpei Guo, Shaofeng Zhang, Runzhong Wang, Chang Liu, and Junchi Yan. GMTR: Graph matching transformers. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6535–6539, 2024.

[23] Siddharth Tourani, Muhammad Haris Khan, Carsten Rother, and Bogdan Savchynskyy. Discrete cycle-consistency based unsupervised deep graph matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 5252–5260, 2024.

[24] Tianxiang Qin, Shikui Tu, and Lei Xu. IA-NGM: A bidirectional learning method for neural graph matching with feature fusion. *Machine Learning*, 113(4):1743–1769, 2024.

[25] Matthias Fey, Jan Eric Lenssen, Christopher Morris, Jonathan Masci, and Nils M. Kriege. Deep graph matching consensus. In *Proceedings of the International Conference on Learning Representations*, 2020.

[26] Qibing Ren, Qingquan Bao, Runzhong Wang, and Junchi Yan. Appearance and structure aware robust deep visual graph matching: Attack, defense and beyond. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15263–15272, 2022.

[27] Haoru Tan, Chuang Wang, Sitong Wu, Xu-Yao Zhang, Fei Yin, and Cheng-Lin Liu. Ensemble quadratic assignment network for graph matching. *International Journal of Computer Vision*, 132(9):3633–3655, 2024.

[28] Feng Zhou and Fernando De la Torre. Factorized graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1774–1789, 2016.

[29] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations*, 2015.

[31] Jingwei Qu, Haibin Ling, Chenrui Zhang, Xiaoqing Lyu, and Zhi Tang. Adaptive edge attention for graph matching with outliers. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 966–972, 2021.

[32] Philippe GH Lehot. An optimal algorithm to detect a line graph and output its root graph. *Journal of the ACM*, 21(4):569–575, 1974.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[34] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4938–4947, 2020.

[35] Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The Annals of Mathematical Statistics*, 35(2):876–879, 1964.

[36] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

[37] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1365–1372, 2009.

[38] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes (VOC) challenge. *International Journal of Computer Vision*, 88:303–338, 2010.

[39] Minsu Cho, Karteek Alahari, and Jean Ponce. Learning graphs to match. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 25–32, 2013.

[40] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. SPair-71k: A large-scale benchmark for semantic correspondence. *arXiv preprint arXiv:1908.10543*, 2019.

[41] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. SplineCNN: Fast geometric deep learning with continuous B-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 869–877, 2018.

[42] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015.

[43] Sushil Mahavir Varma, Irène Waldspurger, and Laurent Massoulié. Graph alignment via Birkhoff relaxation. In *Advances in Neural Information Processing Systems*, pages 52646–52667, 2025.

[44] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1735–1742, 2006.

[45] Artem Moskalev, Ivan Sosnovik, Volker Fischer, and Arnold Smeulders. Contrasting quadratic assignments for set-based representation learning. In *Proceedings of the European Conference on Computer Vision*, pages 88–104, 2022.

[46] Chang Liu, Shaofeng Zhang, Xiaokang Yang, and Junchi Yan. Self-supervised learning of visual graph matching. In *Proceedings of the European Conference on Computer Vision*, pages 370–388, 2022.

[47] Wenqi Guo, Lin Zhang, Shikui Tu, and Lei Xu. Self-supervised bidirectional learning for graph matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 7784–7792, 2023.

[48] Abtin Pourhadi and Paul Swoboda. Normalized matching transformer. In *Proceedings of the International Conference on Pattern Recognition*, pages 339–353, 2026.

[49] Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025.

Appendix

The appendix is organized as follows:

- **Appendix** A offers a more comprehensive discussion of prior research related to CCGM.
- **Appendix** B provides a summary table of the main mathematical symbols.
- **Appendix** C presents the algorithmic pseudocode of the designed CCS for reference.
- **Appendix** D analyzes the time complexity of the proposed CCGM.
- **Appendix** E contains detailed proofs for the theoretical results presented in Section 5, and further explains the specific role of each result within the proposed framework.
- **Appendix** F contains additional experimental validations.

A Related Work

This section provides a fuller discussion of work related to CCGM. We group prior studies into three themes: classical and deep graph matching, quadratic and edge-aware matching formulations, and contrastive or consistency-based objectives for correspondence learning. This organization keeps the discussion compact while preserving the distinctions most relevant to our method.

A.1 Classical and Deep Graph Matching

Graph matching has a long history in computer vision and pattern recognition. In its classical form, the task seeks a correspondence between graph nodes while preserving both unary node similarity and pairwise edge compatibility. The Lawler QAP formulation [6] offers a general model for this objective, and surveys on QAP highlight both its broad applicability and its computational difficulty [7]. Because exact QAP optimization is NP-hard, traditional methods usually rely on approximate inference. Graduated assignment [8] relaxes discrete correspondences and gradually sharpens the assignment. Spectral matching [9] uses eigenvector-based relaxation to exploit pairwise constraints. Integer projected fixed-point methods [10] alternate between continuous updates and projection-like discretization. These approaches established the central role of structural consistency in graph matching, but their dependence on hand-crafted descriptors limits their robustness under large appearance variation, occlusion, and non-rigid deformation.

Deep graph matching methods address this limitation by learning visual representations jointly with differentiable matching modules. GMN [12] introduced an early end-to-end pipeline that combines CNN feature extraction with a differentiable spectral solver. This joint learning paradigm helped motivate subsequent graph matching architectures that integrate representation learning with differentiable assignment. PCA-GM [13] further incorporated graph neural networks with cross-graph aggregation, allowing node representations to capture both intra-graph structure and inter-graph contextual information before assignment. CIE [14] improves representation learning through channel-independent embeddings and employs Hungarian attention to emphasize informative candidate correspondences. More broadly, attention-based feature interaction, inspired by Transformer architectures [33] and local feature matching systems such as SuperGlue [34], has become a standard mechanism for refining within-graph and cross-graph representations.

Many strong deep graph matching architectures follow a node-centric design. Structural cues are propagated through graph neural layers and eventually aggregated into node embeddings, after which the final correspondence is obtained by a linear assignment module or differentiable normalization such as Sinkhorn [35, 36, 43]. GAMnet [20] improves robustness through graph adversarial learning, and COMMON [19] addresses bi-level noisy correspondences through quadratic contrastive learning and momentum distillation to improve robustness against unreliable node and edge correspondences. These methods achieve strong performance without explicitly optimizing edge-to-edge assignments. However, the same design may compress second-order structure before the final matching decision. CCGM addresses this limitation by maintaining node and edge affinity streams in parallel rather than relying solely on node embeddings as the solver input.

A.2 Quadratic, Solver-aware, and Edge-aware Matching

A complementary line of research keeps the quadratic nature of graph matching more explicit. Factorized graph matching [28] is especially relevant because it represents the large Lawler affinity matrix using smaller node and edge affinity matrices together with incidence matrices. This factorization reduces memory cost and exposes the node-edge coupling that underlies the QAP objective. CCGM adopts this incidence-based perspective not only to represent the objective compactly but also to guide the design of its cross-order update rule.

Recent deep methods have explored several ways to incorporate quadratic structure into neural graph matching. NGM [15] models Lawler's QAP through an association graph and uses matching-aware message passing with Sinkhorn embeddings. QC-DGM [17] formulates pairwise graph structure as a quadratic constraint and optimizes it with a differentiable continuous procedure inspired by Frank-Wolfe optimization. BBGM [16] instead integrates discrete combinatorial solvers through black-box differentiation, enabling deep networks to train with solver outputs while avoiding the need to hand-design a differentiable relaxation. Other methods improve robustness or consistency through appearance-structure modeling, ensemble quadratic assignment, or discrete cycle-consistency constraints [26, 27, 23]. These methods demonstrate the importance of retaining structural objectives in deep graph matching. At the same time, practical differentiability often requires truncated optimization, association-graph message passing, or surrogate gradients. Such approximations can be empirically powerful, but the resulting update direction may not coincide with the analytical gradient of the relaxed QAP objective.

Explicit edge modeling is another important route for preserving second-order information. Edge-level cues are particularly useful when local node appearance is ambiguous, as they encode geometric distance, relative orientation, incidence relations, and visual evidence along the connection between keypoints. EAGM [31] is a representative edge-aware method that introduces adaptive edge attention for graph matching with outliers, showing that edge information beyond endpoint appearance can improve robustness. Line graphs provide a natural and principled way to make edges directly matchable: each edge in the primal graph is converted into a node in the line graph, while adjacency in the line graph records incidence between primal edges. Classical graph theory has studied line graphs and efficient line-graph recognition or reconstruction algorithms [32]. In our setting, the primal graph is known, so the line-graph construction is direct. This representation allows CCGM to apply node-matching operations to edge correspondences while explicitly coupling first-order node assignments with second-order edge assignments through cross-order projections.

A.3 Contrastive and Consistency-based Objectives

Training objectives are another major factor in graph matching performance. Contrastive learning is naturally suited to correspondence problems because it pulls matched pairs together and pushes unmatched pairs apart. Early contrastive representation learning [44] and later set-assignment interpretations of contrastive learning [45] provide useful foundations for viewing matching as distributional discrimination. In graph matching, self-supervised methods such as SCGM [46] learn by maximizing agreement between augmented views without requiring dense ground-truth correspondences. IA-SSGM [47] further uses graph contrastive learning to improve robustness against perturbations and noisy correspondences. In supervised settings, StableGM [18] introduces hard-negative mining guided by Sinkhorn ranking scores, while COMMON [19] connects graph matching with quadratic contrastive learning. Building upon such contrastive learning paradigms, the Normalized Matching Transformer [48] combines InfoNCE and hyperspherical losses within a normalized transformer decoder, yielding superior matching accuracy through feature-space regularization.

Consistency-based objectives complement contrastive learning by encouraging predictions to satisfy structural or cycle constraints rather than only feature-level discrimination. Such objectives are especially important in graph matching, where a visually plausible node match may still be structurally inconsistent with neighboring matches. CCGM follows this general motivation but focuses on a different form of consistency: rather than regularizing node embeddings or node-pair scores, our Cross-order Alignment Regularization encourages agreement between matching distributions and their cross-order projections. The node stream induces an edge-match distribution through incidence projection, while the edge stream provides structural support for node matches through the reverse projection. Penalizing disagreement between these two distributions encourages correspondences that are simultaneously appearance-compatible and topology-consistent.

A.4 Positioning of Our CCGM.

CCGM is positioned between efficient node-centric deep graph matching and more explicit quadratic graph matching. Compared with node-centric methods, CCGM does not collapse all structural information into node embeddings before inference; it introduces a line-graph branch so that edge correspondences remain explicit optimization variables. Compared with heavy quadratic or black-box solver approaches, CCGM avoids constructing a dense full Lawler affinity matrix and instead uses the factorized relaxed QAP to derive a transparent incidence-projection update. The Cross-order Consensus Solver therefore provides a solver-aware neural module whose design is aligned with the exact cross-order gradient structure, while the Cross-order Alignment Regularization supplies a training signal that keeps first-order and second-order assignments mutually consistent. This combination defines the main distinction of CCGM: explicit edge assignment, principled node-edge coupling, and distributional alignment between the two orders within a single differentiable graph matching framework.

B Notation

Table 6 provides a summary of the main symbols used throughout the main text.

Table 6: Summary of the main notation used in the paper.

Symbol	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{A})$	A graph with node set $\mathcal{V}$, edge set $\mathcal{E}$, features $\mathbf{X}$, adjacency matrix $\mathbf{A}$.
$\mathcal{G}_1, \mathcal{G}_2$; n_i, e_i	Source and target primal graphs, with n_i nodes and e_i primal edges in $\mathcal{G}_i$, $i \in \{1, 2\}$.
$\mathcal{L}(\mathcal{G}) = (\mathcal{V}_L, \mathcal{E}_L, \mathbf{X}_L, \mathbf{A}_L)$	Line graph of $\mathcal{G}$; each line-graph node corresponds to one primal edge.
$e_k = (v_i, v_j)$, v_k^L	The k-th primal edge and its corresponding line-graph node.
$\mathbf{G}_i$, $\mathbf{D}_i$	Node-edge incidence matrix $\mathbf{G}_i \in \{0,1\}^{n_i \times e_i}$ of $\mathcal{G}_i$ and its node-degree vector $\mathbf{D}_i$, used for cross-order projection and normalization.
$\mathbf{P}$, $\mathcal{P}$	Binary node assignment matrix $\mathbf{P} \in \{0,1\}^{n_1 \times n_2}$ and feasible set $\mathcal{P}$ of partial one-to-one assignments.
$\mathbf{S} \in [0,1]^{n_1 \times n_2}$	Continuous relaxed node assignment matrix used in the factorized relaxed QAP.
$\mathbf{K} \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$	Full Lawler affinity matrix containing first-order and second-order compatibilities.
$\mathbf{K}_v$, $\mathbf{K}_e$	First-order node affinity matrix $\mathbf{K}_v \in \mathbb{R}^{n_1 \times n_2}$ and second-order edge affinity matrix $\mathbf{K}_e \in \mathbb{R}^{e_1 \times e_2}$ in the factorized QAP objective.
$\mathbf{Y} = \mathbf{G}_1^\top \mathbf{S} \mathbf{G}_2$	Edge assignment $\mathbf{Y} \in \mathbb{R}^{e_1 \times e_2}$ induced by the current relaxed node assignment.
$\mathcal{J}(\mathbf{S})$	Factorized relaxed QAP objective optimized in the theoretical analysis.
$\nabla_{\mathbf{S}} \mathcal{J}(\mathbf{S})$	Exact gradient of the relaxed QAP objective with respect to the node assignment, with size $\mathbb{R}^{n_1 \times n_2}$.
I_s, I_t; $\mathbf{X}_s, \mathbf{X}_t$	Source/target input images and their initial node features extracted from the image backbone.
$\mathbf{h}_{\text{end}}^{(k)}$, $\mathbf{h}_{\text{path}}^{(k)}$, $\mathbf{x}_k^L$	Endpoint-based representation, path-sampled representation, and fused line-graph node feature for primal edge e_k.
$\mathbf{H}_s, \mathbf{H}_t$; $\mathbf{L}_s, \mathbf{L}_t$	Refined primal node embeddings and refined line-graph node embeddings, respectively.
$\mathbf{S}_{\text{node}}$, $\mathbf{S}_{\text{edge}}$	Node score matrix $\mathbf{S}_{\text{node}} \in \mathbb{R}^{n_1 \times n_2}$ and edge score matrix $\mathbf{S}_{\text{edge}} \in \mathbb{R}^{e_1 \times e_2}$ maintained and iteratively updated by the Cross-order Consensus Solver.
$\mathbf{S}_{\text{n2e}}$, $\mathbf{S}_{\text{e2n}}$	Node-to-edge projected support $\mathbf{S}_{\text{n2e}} \in \mathbb{R}^{e_1 \times e_2}$ and edge-to-node projected support $\mathbf{S}_{\text{e2n}} \in \mathbb{R}^{n_1 \times n_2}$ used to couple first-order and second-order assignments.
$\mathbf{g}_{\text{self}}^v, \mathbf{g}_{\text{proj}}^v$; $\mathbf{g}_{\text{self}}^e, \mathbf{g}_{\text{proj}}^e$	Entropy-aware gate weights for balancing self-affinity and projected structural evidence in the node and edge streams.
$\mathbf{P}_{\text{node}}$, $\mathbf{P}_{\text{edge}}$	Sinkhorn-normalized node assignment distribution $\mathbf{P}_{\text{node}} \in \mathbb{R}^{n_1 \times n_2}$ and edge assignment distribution $\mathbf{P}_{\text{edge}} \in \mathbb{R}^{e_1 \times e_2}$ after the unrolled solver iterations.

C Pseudocode of Cross-Order Consensus Solver

Algorithm 1 summarizes the iterative procedure of the proposed Cross-order Consensus Solver.

Algorithm 1 Cross-order Consensus Solver (CCS)

Input: Initial node affinity $\mathbf{S}_{node}^{(0)} \in \mathbb{R}^{n_1 \times n_2}$, initial edge affinity $\mathbf{S}_{edge}^{(0)} \in \mathbb{R}^{e_1 \times e_2}$, incidence matrices $\mathbf{G}_1, \mathbf{G}_2$, node degree vectors $\mathbf{D}_1, \mathbf{D}_2$, iterations T, temperature τ.
Output: Refined assignment matrices $\mathbf{P}_{node}$ and $\mathbf{P}_{edge}$.

$\mathbf{S}_{node} \leftarrow \mathbf{S}_{node}^{(0)}, \mathbf{S}_{edge} \leftarrow \mathbf{S}_{edge}^{(0)}$ ▷ Initialize affinities
for $t = 1$ **to** T **do** ▷ Iterative refinement loop
  /* 1. Cross-order Projection */
  Compute Node-to-Edge (N2E) projection:
  $S_{n2e}(k, l) \leftarrow \max(S_{node}(u, p) + S_{node}(v, q), S_{node}(u, q) + S_{node}(v, p))$
  where $e_k = (u, v), e_l = (p, q)$.
  Compute Edge-to-Node (E2N) projection using normalized LogSumExp:
  $\mathbf{S}_{e2n} \leftarrow \tau \log(\mathbf{G}_1 \exp(\mathbf{S}_{edge}/\tau)\mathbf{G}_2^\top) - \tau \log(\mathbf{D}_1\mathbf{D}_2^\top)$
  /* 2. Entropy-aware Gating */
  Calculate sharpness: $H(S) = 1 + \frac{1}{\log N} \sum_j P_{ij} \log P_{ij}$, where $\mathbf{P} = \text{Softmax}(\mathbf{S})$
  Predict node adaptive weights:
    $(\mathbf{g}_{self}^v, \mathbf{g}_{proj}^v) \leftarrow \sigma(\text{MLP}_v([\mathbf{S}_{node}; \mathbf{S}_{e2n}; H(\mathbf{S}_{node}); H(\mathbf{S}_{e2n})]))$
  Predict edge adaptive weights:
    $(\mathbf{g}_{self}^e, \mathbf{g}_{proj}^e) \leftarrow \sigma(\text{MLP}_e([\mathbf{S}_{edge}; \mathbf{S}_{n2e}; H(\mathbf{S}_{edge}); H(\mathbf{S}_{n2e})]))$
  /* 3. Gated State Update */
  $\mathbf{S}_{node} \leftarrow \mathbf{g}_{self}^v \odot \mathbf{S}_{node} + \mathbf{g}_{proj}^v \odot \mathbf{S}_{e2n}$
  $\mathbf{S}_{edge} \leftarrow \mathbf{g}_{self}^e \odot \mathbf{S}_{edge} + \mathbf{g}_{proj}^e \odot \mathbf{S}_{n2e}$
end for
Normalize via Sinkhorn:
$\mathbf{P}_{node} \leftarrow \text{Sinkhorn}(\mathbf{S}_{node}), \mathbf{P}_{edge} \leftarrow \text{Sinkhorn}(\mathbf{S}_{edge})$
return $\mathbf{P}_{node}, \mathbf{P}_{edge}$

D Computational Complexity Analysis

Our method comprises three phases: graph and line graph construction, affinity learning (attention network), and the cross-order consensus solver. To simplify notation, we assume the number of nodes $|\mathcal{V}_s| = |\mathcal{V}_t| = n$, the number of edges is m, and d denotes the dimension of representations. Since the primal graphs are constructed via Delaunay Triangulation which guarantees planarity, the number of edges is bounded by $m \leq 3n - 6$ $(n \geq 3)$. Therefore, the edge count scales linearly with the nodes, i.e., $m = O(n)$. **Graph Construction & Feature Extraction**: Calculating the initial node representations takes $O(n^2 + nd)$. The transformation to a line graph operates with linear time complexity $O(n)$, and generating the fused edge features via path sampling takes $O(md^2) = O(nd^2)$. **Affinity Learning**: The intra- and inter-graph attention mechanisms update both node and edge features. Given $m = O(n)$, the time complexity for processing both streams is $O(nd^2+n^2d+md^2+m^2d) = O(n^2d+nd^2)$. **Cross-order Consensus Solver**: When computing the affinity matrices for both nodes and edges, the time complexity is $O(nd^2+n^2d)$ and $O(md^2+m^2d) = O(n^2d + nd^2)$, respectively. The N2E and E2N cross-order projections utilize sparse incidence matrices, taking $O(n^2)$ operations. The Sinkhorn algorithm applied to both the $n \times n$ node and $m \times m$ edge matrices to obtain doubly-stochastic matrices also takes $O(n^2)$ and $O(m^2) = O(n^2)$ operations. Finally, leveraging the linear upper bound on the number of edges $m = O(n)$, the overall time complexity of CCGM can be simplified to $O(n^2d + nd^2)$. By avoiding the construction of the $O(n^4)$-sized dense pairwise affinity matrix used in conventional QAP-based graph matching formulations, CCGM enables efficient and scalable graph matching in practical applications.

E Proofs of Theoretical Results

This appendix provides detailed proofs of the theoretical results in Section 5. Beyond verifying the algebraic statements, we also explain the role of each result in the proposed framework. The goal is to make explicit which part of CCGM is justified by each result and which part remains an architectural approximation used for empirical flexibility.

E.1 Proof of Theorem 1: Exact Cross-order Gradient

Role of the theorem. Theorem 1 is the central algebraic justification for the Cross-order Consensus Solver. It shows that, for the factorized relaxed QAP objective, the gradient with respect to the node assignment matrix is not merely a node-affinity score. Instead, the second-order term first projects a node assignment to the edge domain, weights the induced edge assignment by the edge affinity matrix, and then projects the resulting edge support back to the node domain. This node-to-edge-to-node structure is the incidence-projection pattern used by the N2E and E2N operations in the proposed solver.

Proof. Recall the relaxed objective

$$\mathcal{J}(\mathbf{S}) = \operatorname{tr}(\mathbf{S}^\top \mathbf{K}_v) + \frac{1}{2} \operatorname{tr}\left[\mathbf{Y}^\top (\mathbf{K}_e \odot \mathbf{Y})\right], \qquad \mathbf{Y} = \mathbf{G}_1^\top \mathbf{S} \mathbf{G}_2. \tag{28}$$

The first term is linear in $\mathbf{S}$, and therefore

$$d \operatorname{tr}(\mathbf{S}^\top \mathbf{K}_v) = \operatorname{tr}\left(\mathbf{K}_v^\top d\mathbf{S}\right), \qquad \nabla_{\mathbf{S}} \operatorname{tr}(\mathbf{S}^\top \mathbf{K}_v) = \mathbf{K}_v. \tag{29}$$

We now consider the quadratic term. Since the Hadamard product applies elementwise, the quadratic part can be written as

$$\mathcal{J}_{\text{quad}}(\mathbf{S}) = \frac{1}{2} \sum_{k=1}^{e_1} \sum_{l=1}^{e_2} \mathbf{K}_e(k,l) \mathbf{Y}_{kl}^2. \tag{30}$$

Taking the differential gives

$$d\mathcal{J}_{\text{quad}} = \sum_{k,l} \mathbf{K}_e(k,l) \mathbf{Y}_{kl}\, d\mathbf{Y}_{kl} = \operatorname{tr}\left[(\mathbf{K}_e \odot \mathbf{Y})^\top d\mathbf{Y}\right]. \tag{31}$$

Because $\mathbf{Y} = \mathbf{G}_1^\top \mathbf{S} \mathbf{G}_2$, its differential is

$$d\mathbf{Y} = \mathbf{G}_1^\top d\mathbf{S} \mathbf{G}_2. \tag{32}$$

Substituting Eq. (32) into Eq. (31) yields

$$\begin{aligned} d\mathcal{J}_{\text{quad}} &= \operatorname{tr}\left[(\mathbf{K}_e \odot \mathbf{Y})^\top \mathbf{G}_1^\top d\mathbf{S} \mathbf{G}_2\right] \\ &= \operatorname{tr}\left[\mathbf{G}_2 (\mathbf{K}_e \odot \mathbf{Y})^\top \mathbf{G}_1^\top d\mathbf{S}\right] \\ &= \operatorname{tr}\left[\left(\mathbf{G}_1 (\mathbf{K}_e \odot \mathbf{Y}) \mathbf{G}_2^\top\right)^\top d\mathbf{S}\right], \end{aligned} \tag{33}$$

where we use the cyclic invariance of the trace. By the definition of the matrix gradient under the Frobenius inner product,

$$d\mathcal{J}_{\text{quad}} = \left\langle \nabla_{\mathbf{S}} \mathcal{J}_{\text{quad}}, d\mathbf{S} \right\rangle_F, \tag{34}$$

and thus

$$\nabla_{\mathbf{S}} \mathcal{J}_{\text{quad}} = \mathbf{G}_1 (\mathbf{K}_e \odot \mathbf{Y}) \mathbf{G}_2^\top = \mathbf{G}_1 \left(\mathbf{K}_e \odot (\mathbf{G}_1^\top \mathbf{S} \mathbf{G}_2)\right) \mathbf{G}_2^\top. \tag{35}$$

Combining Eq. (29) and Eq. (35) gives

$$\nabla_{\mathbf{S}} \mathcal{J}(\mathbf{S}) = \mathbf{K}_v + \mathbf{G}_1 \left(\mathbf{K}_e \odot (\mathbf{G}_1^\top \mathbf{S} \mathbf{G}_2)\right) \mathbf{G}_2^\top, \tag{36}$$

which is Eq. (22). □

Interpretation for the solver. The three factors in the second-order gradient have direct algorithmic meanings. The term $\mathbf{G}_1^\top \mathbf{S} \mathbf{G}_2$ computes the edge assignment induced by the current node assignment; the Hadamard product with $\mathbf{K}_e$ keeps only structurally compatible edge correspondences; and the multiplication by $\mathbf{G}_1$ and $\mathbf{G}_2^\top$ distributes the edge-level evidence back to endpoint-level node pairs. CCGM follows this structure rather than replacing it by an unconstrained learned message-passing block.

Remark 1 (Why the factor $\frac{1}{2}$ is used). The factor $\frac{1}{2}$ in Eq. (28) ensures that the derivative of the elementwise quadratic term is exactly $\mathbf{K}_e \odot \mathbf{Y}$. Without this factor, the gradient of the quadratic term would contain an additional factor of two. The factor does not change the maximizer of the objective when the quadratic term is considered alone, but it makes the stated gradient formula and the implementation-level interpretation consistent.

Remark 2 (Scope of the result). The theorem is an exact result for the relaxed factorized QAP objective in Eq. (5). It does not state that every learned update in CCGM is identical to a hand-designed numerical optimizer. Rather, it identifies the mathematically correct cross-order direction that the solver should preserve. The entropy-aware gate in CCGM is therefore best understood as a learnable, data-dependent step-size and fusion mechanism built on top of this exact incidence-projection template.

E.2 Proof of Proposition 1: Incidence Consistency

Role of the proposition. The proposition explains what it means for a node assignment and an edge assignment to agree. In graph matching, an edge correspondence should not be selected independently of its endpoints: matching edge (u, v) to edge (p, q) is structurally meaningful only when the node assignment maps $\{u, v\}$ to $\{p, q\}$. The proposition formalizes this idea using incidence vectors, and therefore justifies both the node-to-edge projection in the solver and the cross-order alignment loss used during training.

Proof. Let $e_k = (u, v) \in \mathcal{E}_1$ and $e_l = (p, q) \in \mathcal{E}_2$. The incidence vector $\mathbf{g}_{1,k}$ contains ones exactly at the endpoints u and v, and zeros elsewhere. Similarly, $\mathbf{g}_{2,l}$ contains ones exactly at p and q. Therefore,

$$\begin{aligned} c_{kl} &= \mathbf{g}_{1,k}^{\top} \mathbf{P} \mathbf{g}_{2,l} \\ &= P_{up} + P_{uq} + P_{vp} + P_{vq}. \end{aligned} \tag{37}$$

Because $\mathbf{P}$ is a binary partial one-to-one assignment, each source node can be matched to at most one target node and each target node can be matched by at most one source node. Consequently, among the four terms in Eq. (37), at most two can be equal to one. Hence $c_{kl} \in \{0, 1, 2\}$.

The value of c_{kl} has a direct counting interpretation. If $c_{kl} = 0$, neither endpoint of e_k is matched to an endpoint of e_l. If $c_{kl} = 1$, exactly one endpoint of e_k is matched to one endpoint of e_l, so the two edges are only partially compatible. If $c_{kl} = 2$, then both endpoints of e_k are matched to endpoints of e_l. The one-to-one constraints rule out the possibility that both endpoints of e_k are mapped to the same target endpoint, and also rule out the possibility that one source endpoint is mapped to both target endpoints. Thus $c_{kl} = 2$ holds exactly when either

$$u \mapsto p, \quad v \mapsto q, \tag{38}$$

or

$$u \mapsto q, \quad v \mapsto p. \tag{39}$$

These two cases are precisely the two orientation choices for matching the unordered endpoint set $\{u, v\}$ to $\{p, q\}$.

Now consider the orientation-invariant score

$$q_{kl}(\mathbf{P}) = \max(P_{up} + P_{vq}, P_{uq} + P_{vp}). \tag{40}$$

If the direct orientation $(u \mapsto p, v \mapsto q)$ holds, then $P_{up} + P_{vq} = 2$, and hence $q_{kl}(\mathbf{P}) = 2$. If the reversed orientation $(u \mapsto q, v \mapsto p)$ holds, then $P_{uq} + P_{vp} = 2$, and again $q_{kl}(\mathbf{P}) = 2$. Conversely, if $q_{kl}(\mathbf{P}) = 2$, one of these two orientation-specific sums must be equal to two, which means both endpoint matches in that orientation are present. Therefore, the edge match (e_k, e_l) is induced by $\mathbf{P}$. If neither orientation is induced, each orientation-specific sum is at most one, and thus $q_{kl}(\mathbf{P}) \leq 1$. □

Interpretation for cross-order alignment. For binary assignments, Proposition 1 gives an exact discrete criterion for deciding whether an edge correspondence is supported by node correspondences. In the relaxed setting used by neural graph matching, the entries of $\mathbf{S}$ are soft scores rather than binary indicators. The same incidence calculation then becomes a differentiable structural confidence

score: it measures how strongly the current node distribution supports an edge-level hypothesis. CAR uses this principle in both directions, encouraging edge predictions to agree with node-induced edge support and node predictions to agree with edge-induced node support.

Remark 3 (Undirected edges and orientation). Visual graph matching typically treats geometric edges as undirected. For this reason, the two endpoint orientations of an edge correspondence should be considered equivalent. The max operation in the orientation-invariant score selects the stronger of the two possible endpoint orientations. This is also why the N2E projection in CCGM uses an orientation-invariant aggregation instead of assuming a fixed ordering of edge endpoints.

Remark 4 (Why incidence consistency is stronger than local appearance agreement). Two node pairs may have high visual similarity even when their surrounding structures are incompatible. Incidence consistency introduces a structural check: an edge-level match is reliable only when both of its endpoint matches are jointly supported. This explains why explicit edge assignments can improve robustness under viewpoint changes, occlusions, and outliers.

E.3 Proof of Theorem 2: Monotonicity under Smoothness

Role of the theorem. Theorem 2 provides an optimization-level motivation for using the exact QAP gradient as a design template. It states that, under the standard smoothness assumption, a projected ascent step taken along the exact gradient of the relaxed QAP objective is guaranteed not to decrease that objective for a suitable step size. This result should be read as a reference property of the exact gradient direction. CCGM does not claim that the learned gate is identical to Euclidean projected gradient ascent; instead, the theorem explains why preserving the exact cross-order gradient structure is preferable to using an unrelated proxy direction.

Proof. Let

$$\mathbf{G} = \nabla \mathcal{J}(\mathbf{S}), \qquad \mathbf{S}^+ = \Pi_{\mathcal{D}}(\mathbf{S} + \eta \mathbf{G}), \tag{41}$$

where $\Pi_{\mathcal{D}}$ denotes Euclidean projection onto the closed convex set $\mathcal{D}$. The first-order optimality condition for Euclidean projection states that

$$\left\langle \mathbf{S} + \eta \mathbf{G} - \mathbf{S}^+, \mathbf{Z} - \mathbf{S}^+ \right\rangle \leq 0, \qquad \forall \mathbf{Z} \in \mathcal{D}. \tag{42}$$

Since $\mathbf{S} \in \mathcal{D}$, we may choose $\mathbf{Z} = \mathbf{S}$. Substituting this choice into Eq. (42) gives

$$\left\langle \mathbf{S} + \eta \mathbf{G} - \mathbf{S}^+, \mathbf{S} - \mathbf{S}^+ \right\rangle \leq 0. \tag{43}$$

Let $\Delta = \mathbf{S}^+ - \mathbf{S}$. Then the above inequality becomes

$$\langle \eta \mathbf{G} - \Delta, -\Delta \rangle \leq 0, \tag{44}$$

which is equivalent to

$$\langle \mathbf{G}, \Delta \rangle \geq \frac{1}{\eta} \|\Delta\|_F^2. \tag{45}$$

Because $\mathcal{J}$ has an L-Lipschitz continuous gradient on $\mathcal{D}$, the standard descent/ascent lemma gives, for any $\mathbf{S}, \mathbf{S}^+ \in \mathcal{D}$,

$$\mathcal{J}(\mathbf{S}^+) \geq \mathcal{J}(\mathbf{S}) + \left\langle \nabla \mathcal{J}(\mathbf{S}), \mathbf{S}^+ - \mathbf{S} \right\rangle - \frac{L}{2} \|\mathbf{S}^+ - \mathbf{S}\|_F^2. \tag{46}$$

Combining Eq. (45) and Eq. (46) yields

$$\mathcal{J}(\mathbf{S}^+) \geq \mathcal{J}(\mathbf{S}) + \left(\frac{1}{\eta} - \frac{L}{2} \right) \|\mathbf{S}^+ - \mathbf{S}\|_F^2, \tag{47}$$

which proves Eq. (26). If $0 < \eta \leq 2/L$, then $\frac{1}{\eta} - \frac{L}{2} \geq 0$, and therefore $\mathcal{J}(\mathbf{S}^+) \geq \mathcal{J}(\mathbf{S})$.

It remains to verify the Lipschitz bound in Eq. (27). Let $\mathbf{S}_1, \mathbf{S}_2 \in \mathcal{D}$ and define $\Delta = \mathbf{S}_1 - \mathbf{S}_2$. The linear term $\mathbf{K}_v$ is constant in the gradient and cancels out:

$$\nabla \mathcal{J}(\mathbf{S}_1) - \nabla \mathcal{J}(\mathbf{S}_2) = \mathbf{G}_1 \left[\mathbf{K}_e \odot \left(\mathbf{G}_1^\top \Delta \mathbf{G}_2 \right) \right] \mathbf{G}_2^\top. \tag{48}$$

Using submultiplicativity of the spectral norm and the compatibility between spectral and Frobenius norms, we obtain

$$\begin{aligned}\|\nabla\mathcal{J}(\mathbf{S}_1)-\nabla\mathcal{J}(\mathbf{S}_2)\|_F &= \left\|\mathbf{G}_1\left[\mathbf{K}_e\odot\left(\mathbf{G}_1^\top\Delta\mathbf{G}_2\right)\right]\mathbf{G}_2^\top\right\|_F\\ &\le \|\mathbf{G}_1\|_2\|\mathbf{G}_2\|_2\left\|\mathbf{K}_e\odot\left(\mathbf{G}_1^\top\Delta\mathbf{G}_2\right)\right\|_F\\ &\le \|\mathbf{K}_e\|_\infty\|\mathbf{G}_1\|_2\|\mathbf{G}_2\|_2\left\|\mathbf{G}_1^\top\Delta\mathbf{G}_2\right\|_F\\ &\le \|\mathbf{K}_e\|_\infty\|\mathbf{G}_1\|_2^2\|\mathbf{G}_2\|_2^2\|\Delta\|_F. \end{aligned} \tag{49}$$

Thus $\nabla\mathcal{J}$ is Lipschitz continuous with constant no larger than

$$L=\|\mathbf{K}_e\|_\infty\|\mathbf{G}_1\|_2^2\|\mathbf{G}_2\|_2^2, \tag{50}$$

which proves the stated smoothness bound. □

Interpretation for learned updates. The theorem separates two issues that are sometimes conflated. The first is the *direction* of structural improvement, which is given by the exact gradient in Theorem 1. The second is the *implementation* of an update step inside a neural network. A fixed projected-gradient step is mathematically clean but may be too rigid for visual graph matching, where feature quality, occlusion, and outliers vary across image pairs. CCGM therefore keeps the exact cross-order direction as the structural template and learns an entropy-aware gate to determine how strongly self-affinity and projected structural evidence should be fused at each iteration.

Remark 5 (Relation to Sinkhorn normalization). The monotonicity statement is written for Euclidean projection onto a closed convex relaxation $\mathcal{D}$. In implementation, graph matching networks often use Sinkhorn normalization to obtain approximately doubly stochastic matrices. Sinkhorn normalization is differentiable and effective for enforcing soft one-to-one constraints, but it is not identical to Euclidean projection in general. Therefore, Theorem 2 should be interpreted as a principled motivation for the exact gradient direction rather than as a literal guarantee for every normalized, gated forward pass.

Remark 6 (Why proxy directions may be less reliable). A surrogate or black-box gradient may optimize a different local objective from the relaxed QAP in Eq. (5). Such a direction can still be empirically useful, but it does not automatically inherit the monotonicity property above for the QAP objective itself. This observation motivates our emphasis on preserving the incidence-projection form of the exact gradient when designing the solver.

F Additional Experimental Results

F.1 Generalization Ability

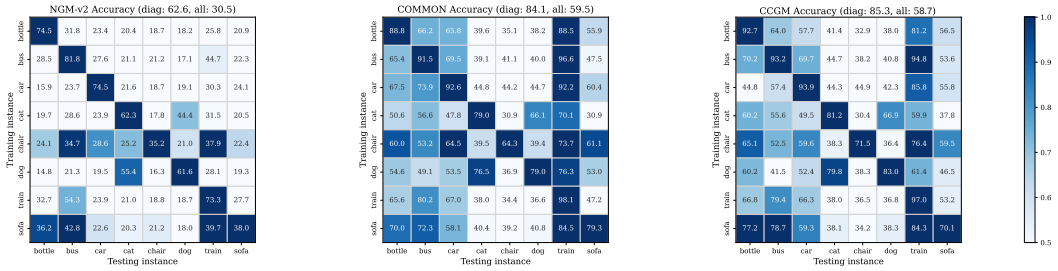

Figure 3: Models are trained on the categories indicated by the y-axis and evaluated on those indicated by the x-axis. Darker color denotes relatively better performance in the same column, and the average accuracies on both diagonal part (learned during training) and all elements (trained+generalized) of confusion matrices are reported in the brackets over confusion matrices. The train/test split follows the main experiment and eight categories are selected randomly.

To evaluate the generalization ability of our proposed framework, we conduct cross-category matching experiments where models are trained on a single specific category (y-axis) and evaluated across all other categories (x-axis). During this single-category training phase, it is worth noting that the Pascal

VOC dataset exhibits a long-tail distribution, where the number of available images varies significantly across categories. Employing a fixed number of iterations for all categories would inevitably lead to severe overfitting on minority classes and underfitting on majority classes. To ensure a fair and rigorous evaluation, we introduce a category-aware dynamic sampling strategy. Specifically, we proportionally scale the number of training iterations per epoch based on the actual sample size of the target category relative to the entire dataset. This strategy ensures that the model's expected exposure frequency to each individual image remains mathematically consistent across different categories, thereby guaranteeing that our cross-category generalization results strictly reflect the topological transferability of the model rather than training biases. Under this protocol, as illustrated in Figure 3, the baseline NGM-v2 obtains an intra-class accuracy of 62.6% and an overall accuracy of 30.5%, reflecting the inherent difficulty of directly transferring learned matching patterns to novel categories. In contrast, our proposed CCGM demonstrates exceptional cross-category generalizability. CCGM achieves the highest intra-class accuracy (diag: 85.3%) to ensure optimal matching on known classes, while concurrently maintaining a highly competitive overall generalization performance. Notably, visual inspection of the CCGM heatmap reveals that positive transfer (darker blocks) occurs predominantly among geometrically or semantically similar categories, such as vehicles (e.g., bus, car, train) and animals (e.g., cat, dog). This compellingly corroborates that CCGM successfully captures the underlying topological structures and semantic correspondences shared across different classes, rather than merely memorizing class-specific visual features.

F.2 Integration of Vision Foundation Models

Table 7: Comparison of matching accuracy on Pascal VOC dataset using different visual feature backbones (VGG-16 vs DINOv3).

Method	**Aero**	**Bike**	**Bird**	**Boat**	**Bottle**	**Bus**	**Car**	**Cat**	**Chair**	**Cow**	
CCGM+VGG-16	72.2	81.9	85.0	79.4	92.9	95.0	91.1	84.8	75.8	84.8	
CCGM+DINOv3	82.0	81.4	91.5	84.4	95.1	96.5	94.6	91.2	81.6	88.0	
Method	**Table**	**Dog**	**Horse**	**Mbike**	**Person**	**Plant**	**Sheep**	**Sofa**	**Train**	**Tv**	**Mean**
CCGM+VGG-16	100.0	85.8	83.9	82.2	75.3	97.9	83.5	83.4	98.5	95.4	86.4
CCGM+DINOv3	100.0	93.8	90.5	90.8	83.3	98.5	93.3	95.0	98.5	95.6	91.3

Conventionally, deep graph matching pipelines have relied on early convolutional architectures, such as VGG-16, to ensure fair and standardized performance evaluations. However, compared with modern vision foundation models, these backbones may provide less expressive semantic representations, particularly under substantial intra-class variation and cluttered backgrounds. To overcome these limitations and explore the performance upper bound of our framework, we investigate the integration of advanced visual foundation models for initial feature extraction.

Specifically, we replace the standard VGG-16 backbone with a pretrained DINOv3 [49] model. Instead of relying on hierarchical convolutional feature maps, we directly extract the dense patch embeddings from the terminal blocks of a pre-trained DINOv3 model. These embeddings provide highly discriminative, fine-grained semantic priors for the initial node representations in our CCGM framework. As demonstrated in Table 7, integrating DINOv3 yields a notable performance improvement. This confirms that our proposed CCGM is not restricted to specific classical CNNs. Instead, it serves as a highly flexible and model-agnostic graph matching framework that can seamlessly leverage the robust semantic priors of modern vision foundation models to further push the frontiers of matching accuracy.

F.3 Visualization

Figure 4 visualizes the matching results of CCGM on the Pascal VOC dataset. In each image pair, the blue dots denote the annotated keypoints, and the lines represent correspondences predicted by CCGM. To intuitively illustrate the matching accuracy, correct predictions (true positives) are drawn with solid green lines, while incorrect matches (false positives) are highlighted with red lines. As observed from the visualization, CCGM achieves highly accurate keypoint matching in complex real-world scenarios. The model demonstrates exceptional matching capability across both rigid objects and non-rigid entities, establishing high-precision correspondences in various challenging sample

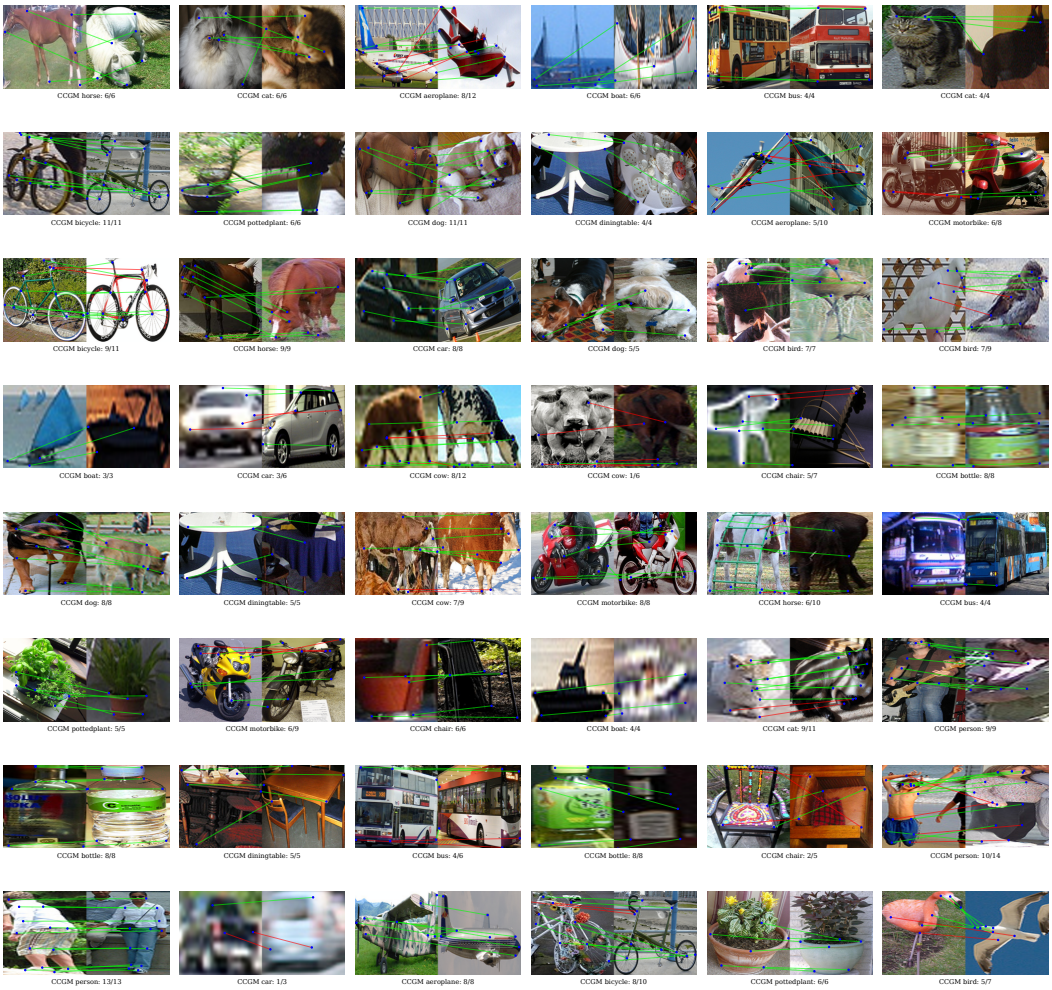

Figure 4: The visualization of the matching results achieved by CCGM on the Pascal VOC dataset. In the graphical depictions, correct predictions are indicated by green lines, while incorrect predictions are highlighted in red. The subtitle accompanying each figure provides information on the count of correctly matched keypoints out of the total keypoints.

pairs. Despite severe intra-class variations, large scale differences, and extreme viewpoint changes, CCGM consistently maintains strong graph structural consistency. Although a few false positive matches occur in categories with highly self-similar structures or severe local ambiguities, the overall topological alignment remains robust. These qualitative examples complement the quantitative results and illustrate the effectiveness of CCGM in combining visual correspondence cues with structural reasoning.

F.4 Ablation Studies

To further validate the matching ability of our proposed components under complex geometric transformations, we conduct an expanded ablation study on the SPair-71k dataset. Unlike Pascal VOC, SPair-71k exhibits severe viewpoint variations, scale changes, and non-rigid deformations, which frequently lead to profound local appearance ambiguities. Table 8 summarizes the progressive performance gains of our modules in this highly challenging scenario. Starting from a baseline of 84.5%, introducing the Cross-order Consensus Solver yields a substantial improvement of +0.8% (reaching 85.3%). This significant jump highlights the necessity of iterative node-edge verification: when local node features become visually ambiguous due to extreme perspective shifts, relying on explicit structural consensus effectively resolves local mismatches by ensuring that node correspondences are

Table 8: Ablation study of CCGM on SPair-71k.

Cross-order Consensus Solver	**Line Graph Construction**	**Cross-order Alignment Regularization**	**SPair-71k**
✓	✓	✓	**86.0**
✓	✓	×	85.6
✓	×	×	85.3
×	×	×	84.5

Table 9: Hyperparameter sensitivity analysis on Pascal VOC.

T	1	2	3	4	6	8	12
Accuracy (%)	86.19	86.37	86.33	**86.43**	86.36	86.31	86.09

strictly supported by their incident edge assignments. Subsequently, incorporating the Line Graph Construction further improves the accuracy to 85.6%. This demonstrates that maintaining explicit edge features preserves crucial higher-order geometric contexts. These explicit second-order cues are vital for distinguishing structurally similar but semantically different regions across images. Finally, the full model with Cross-order Alignment Regularization achieves the highest accuracy of 86.0%. This confirms that explicitly regularizing the distributional agreement between first-order (node) and second-order (edge) assignments during training guides the network toward more structurally reliable correspondences, aligning seamlessly with the underlying relaxed quadratic assignment objective. We further evaluate the sensitivity of CCGM to the number of solver iterations T. As shown in Table 9, the matching accuracy remains stable across different iteration counts, with the highest observed accuracy achieved at $T = 4$.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims and contributions are explicitly summarized at the end of the Introduction, and they align directly with the theoretical analysis and experimental validation presented in the subsequent sections.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses potential failure cases in Appendix F.3, and points out future extensions in the Conclusion.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The assumptions and detailed proofs for all theoretical claims (Theorem 1, Proposition 1, and Theorem 2) are detailed in Appendix E.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The experimental settings, including network architectures, hyperparameter choices, optimization strategies, and dataset processing, are comprehensively detailed in Section 6.1.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The source code is not publicly available at this stage. However, we provide comprehensive implementation details, including model architectures, hyperparameter settings, and experimental protocols in the paper. All experiments are conducted on publicly available benchmark datasets to facilitate reproducibility.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 6.1 systematically specifies the necessary training and testing details, including the optimizer, hyperparameter choices, learning rate schedule and batch size.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Error bars are not reported as we strictly follow the standard evaluation protocol established by prior deep graph matching works.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 6.1 explicitly states that all experiments were conducted on a Linux server equipped with an Intel Xeon Platinum 8468V CPU, 1 TB of RAM, a 7 TB SSD, and an NVIDIA H100 PCIe GPU (80 GB VRAM).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We focus on fundamental computer vision and graph matching algorithms using standard public datasets, which fully conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

Justification: This paper focuses on fundamental theoretical and algorithmic improvements for structural graph matching. Given its highly abstract and foundational nature, a direct discussion of specific broader societal impacts was omitted.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The proposed graph matching method does not involve high-risk assets such as generative AI, face recognition models, or newly scraped datasets that would require specific safety safeguards.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All existing assets, including the datasets (Pascal VOC, SPair-71K, Willow), compared methods and foundational backbones (VGG-16, DINOv3), are properly credited with citations.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper introduces a new algorithmic framework rather than releasing new datasets or standalone reproducible assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The research relies exclusively on existing public datasets and does not involve any crowdsourcing or human subject experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The research does not involve human subjects, making IRB approval inapplicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research is purely based on cross-order graph matching principles and does not involve Large Language Models (LLMs) as any functional component.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.