CLAPS: RELAXATION-FREE PERMUTATION SYNCHRONIZATION WITH EXACT PER-OBJECT ASSIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

In permutation synchronization, the goal is to recover cycle-consistent correspondences among the keypoints of a collection of objects, such as images, from noisy pairwise matchings. In this work, we propose **CAPS**, a solver that keeps the problem in its original combinatorial form throughout, unlike methods that relax it into a continuous domain, losing tightness, and then round the solution back. We show that, once the assignments of all other objects are fixed, the objective is *exactly affine* in the assignment of the remaining object, because the feasibility constraints alone remove every interaction between its keypoints. Each update of CAPS is therefore a linear assignment problem with a totally unimodular constraint matrix, solved exactly and without rounding. We prove that CAPS increases the objective monotonically, terminates after finitely many passes at a block-wise optimal assignment, and costs time linear in the number of observed matches per pass. We further unroll CAPS into a lightweight network, **CLAPS**, that learns per-layer constants and a per-edge trust function end-to-end, which allows it to decline outliers, while every assignment step remains exact. On the SPair-71k and Willow Object benchmarks, CLAPS outperforms the state of the art in both precision and F-score.

1 INTRODUCTION

Establishing correspondences between visual observations is a fundamental task in computer vision, with downstream impact on shape deformation modeling [11, 13], object tracking [2, 37], 3D reconstruction from imagery [28, 29], graph matching [19, 20], and image registration [33, 39]. Even for a single pair of images, matching keypoints while respecting their geometric relations is itself a quadratic assignment problem, which is NP-hard in general [26]. Real-world pipelines, moreover, need to match dozens or hundreds of images at once. In this multi-object setting, every pairwise match is linked to the others through transitivity, so an error in one match propagates through the collection, and a solver that treats each pair separately cannot correct it.

Permutation synchronization addresses this difficulty by exploiting the over-determined structure of a multi-image collection [14, 25]. The core requirement is *cycle consistency*, under which composing the matchings around any closed path of images yields the identity. This consistency condition has proven useful across multi-alignment [1, 3, 17], multi-shape analysis [12, 15, 16], feature correspondence [6, 8, 9, 38], and multi-graph alignment [4, 34, 36]. By aggregating the pairwise matchings into a single coherent assignment over the whole collection, synchronization returns matchings that respect transitivity throughout, which no pairwise matcher guarantees.

The standard mathematical formulation of synchronization is a trace maximization over permutation matrices subject to integrality and row/column-sum constraints. Spectral approaches opened this direction by embedding permutations into the orthogonal group, which turns the combinatorial program into a tractable continuous one [25, 31]. This tractability comes at a cost in scope, since these approaches presume full bijective correspondences and cannot express a partial match. Later methods admit partial matches while keeping cycle consistency, but they too work on a continuous version of the problem. A scheme based on non-negative matrix factorization computes a continuous solution and has to project it back at the end, so its output is determined by a step that the optimized

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

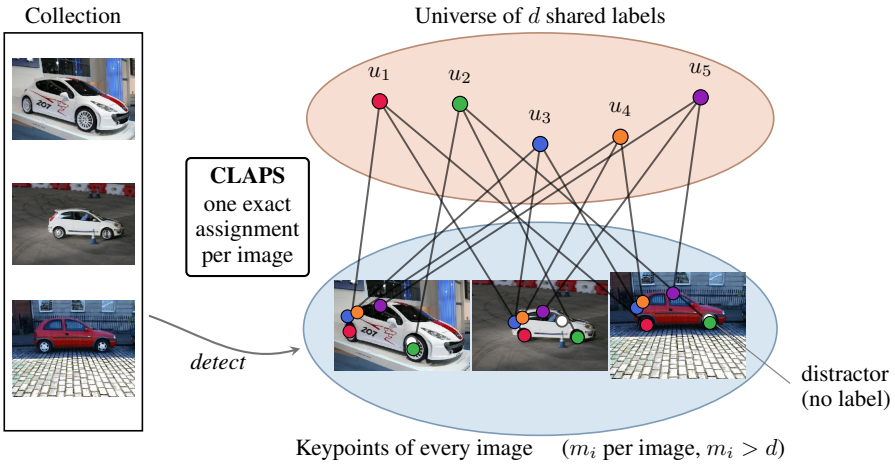

Figure 1: **Permutation synchronization, and what CLAPS returns.** Image i contributes m_i keypoints (blue); the *distractors* among them (hollow) have no correct partner, so m_i exceeds the number d of shared universe labels (orange). CLAPS assigns each keypoint to at most one label by one exactly solved linear assignment problem per image, giving image i its assignment $\mathbf{X}_i$ (black lines), and reads off every pairwise matching as $\mathbf{X}_{ij} = \mathbf{X}_i\mathbf{X}_j^T$, so cycle consistency holds by construction.

criterion does not take into account [5]. The Stiefel-manifold formulation does not address the combinatorial problem either [7]. It replaces the binary constraints by an orthogonality constraint, a relaxation that loses tightness, and then promotes sparsity to push the solution towards binary values. Because the problem it solves is this loose relaxation and not the original one, the sparsity term can guarantee neither that the result satisfies the combinatorial constraints nor that it is an optimal solution of the original problem. Other methods score each observed match by its consistency with the rest of the collection and obtain a matching only after thresholding or projecting these scores [21, 32].

A second route abandons the continuous embedding and searches over matchings directly. A construction-then-refinement pipeline pairs a sequential or a parallel construction with a graph-matching local search, giving SEQ-C + SEQ-LS and PAR-C + PAR-LS [19]. GREEDA carries the same recipe to sparse and incomplete instances behind an added swap-based refinement [20]. None of them carries a convergence guarantee, since each improves a heuristic objective by trial exchanges with no formal assurance of reaching even a local optimum of the synchronization criterion, and the returned matching depends on where the exchanges terminate.

This paper works on the combinatorial problem itself, one object at a time. Our key observation is that, once the assignments of all other objects are fixed, the constraints on the remaining object make the objective exactly affine in its assignment, for any observed matchings. Each update is therefore a linear assignment problem that is solved exactly, and every iterate is a partial permutation whose pairwise matchings are cycle-consistent by construction, as Figure 1 illustrates. We call the resulting solver **CAPS** (Combinatorial Assignment Permutation Solver). Its iteration unrolls into a network with a small number of learned parameters, **CLAPS** (Combinatorial Learned Assignment Permutation Solver), which keeps every assignment step exact and contains CAPS as a special case, so training begins from the exact solver.

Our contribution

We introduce a combinatorial alternating optimization algorithm for permutation synchronization that works on the original problem throughout, never on a relaxation that loses tightness, together with a deep-unrolled counterpart. The main contributions of this work are summarized as follows.

- **A direct combinatorial approach.** We work directly in the space of binary partial permutation matrices and never pass through a loose relaxation of the problem, so no tightness is lost at any stage. Every iterate is a valid partial permutation, and the pairwise matchings read off from it are cycle-consistent by construction.

- **An exactly affine objective for each object.** Feasibility alone requires the keypoints of one object to occupy distinct universe labels, and this requirement removes every interaction between two keypoints of the same object, for any observed matchings. With the other objects fixed, the objective is therefore exactly affine in the assignment of the remaining object, although as a whole it is an indefinite quadratic in general.
- **Each update is solved exactly, by assignment and without rounding.** The constraint matrix of each per-object update is totally unimodular, so the linear program of that update has an integral optimum, which the Jonker–Volgenant algorithm computes exactly as a rectangular linear assignment problem. Because the feasible set is a *product* over objects, updating one object never affects the feasibility of the others. No thresholding, no greedy pass, and no projection onto binary values is performed at any point.
- **A deep-unrolled realization that remains relaxation-free.** Each iteration is a cost assembly followed by an exact assignment. Truncating the iteration to a fixed number of layers and making the per-layer constants learnable — together with a small per-edge trust function of cycle-support features — gives a network trained end-to-end against the annotation. The learned unary term admits an opt-out, which lets the model *decline* an outlier and leave it unlabeled. The assignment step remains an exact linear assignment at training and test time alike, so every iterate is a partial permutation and no tightness is lost at any point. With its initial constants and a uniform trust, the network reproduces the update of the iterative solver exactly.
- **Guarantees stated for a finite feasible set, and a complexity analysis.** Because each step maximizes the true objective, the objective is non-decreasing, and it is integer-valued whenever the observations are, which bounds the number of improving passes. Under a strict-improvement rule the *iterate sequence* becomes constant after finitely many passes, and its limit is *block-wise optimal*, meaning no change confined to one object can improve the objective, with each conditional problem solved exactly. One pass is linear in the number of *observed* matches.
- **Evaluation on the standard benchmarks of this literature.** We evaluate on SPair-71k and Willow Object, the standard benchmarks for this problem. Ground truth is taken from each dataset's own semantic annotations, keypoints are ordered so that a row index carries no information about its universe label, unannotated outliers are left in place, and the observed matchings come from a real feature matcher. Cycle consistency is treated as a requirement of the design, so every method we compare against returns a synchronized matching. Our method outperforms the state of the art in both precision and F-score.

Notation. For the notation used throughout this paper, please refer to Appendix A.

2 Problem Formulation

We consider a collection of k objects, where the i-th object contains m_i keypoints. Throughout, $\mathbb{1}_p$ denotes the all-ones vector of length p, and all vector inequalities are interpreted entry-wise. For each ordered pair (i, j), the pairwise correspondence is represented by a partial permutation matrix $\mathbf{X}_{ij} \in \mathbb{P}_{m_i m_j} := \{\mathbf{X} \in \{0,1\}^{m_i \times m_j} : \mathbf{X}\mathbb{1}_{m_j} \leq \mathbb{1}_{m_i},\ \mathbf{X}^T\mathbb{1}_{m_i} \leq \mathbb{1}_{m_j}\}$, which encodes which keypoints of object i match which keypoints of object j. In the bijective (full-correspondence) case, the family $\mathcal{P} = \{\mathbf{X}_{ij}\}_{i,j=1}^{k}$ is *cycle-consistent* if for every triplet of indices (i, ℓ, j) the composition law $\mathbf{X}_{i\ell}\,\mathbf{X}_{\ell j} = \mathbf{X}_{ij}$ holds without exception.

A convenient way to parameterize all cycle-consistent families is through *universe matchings*. We define $\overline{\mathbb{P}}_{m_i d}$ as the set of full-rank partial permutations, $\overline{\mathbb{P}}_{m_i d} = \{\mathbf{X} \in \mathbb{P}_{m_i d} : \mathbb{1}^T\mathbf{X}\,\mathbb{1} = \min(m_i, d)\}$, the *maximal* partial permutations, where d is the number of shared universe labels, that is, of distinct keypoint identities across all objects. Writing the row constraint as the equality $\mathbf{X}\mathbb{1}_d = \mathbb{1}_{m_i}$ instead would be infeasible whenever $m_i > d$ — the regime that motivates the partial setting — since each label can be used only once. Maximality reduces to that equality when $m_i \leq d$, so the complete setting is unaffected. Cycle consistency among the pairwise matrices $\{\mathbf{X}_{ij}\}$ holds if and only if there exist object-to-universe assignments $\mathcal{U} = \{\mathbf{X}_i \in \overline{\mathbb{P}}_{m_i d}\}_{i=1}^{k}$ satisfying $\mathbf{X}_{ij} = \mathbf{X}_i\,\mathbf{X}_j^T$ for every pair (i, j), an equivalence that extends to partial (non-bijective) maps [5, 35].

Given a noisy observed set $\mathcal{P} = \{\mathbf{X}_{ij}\}_{i,j=1}^{k}$, permutation synchronization seeks the universe assignment family that best fits the observations, and the standard least-squares trace-matching criterion $\underset{\{\mathbf{X}_i \in \overline{\mathbb{P}}_{m_i d}\}}{\arg\max} \sum_{i,j} \operatorname{tr}(\mathbf{X}_{ij}^T \mathbf{X}_i \mathbf{X}_j^T)$ is equivalent to a single trace maximization [25]. Setting $m := \sum_{i=1}^{k} m_i$ and stacking the object-to-universe matrices as

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_k \end{bmatrix} \in \mathbb{R}^{m \times d}, \qquad \mathbf{W} := [\mathbf{X}_{ij}]_{i,j=1}^{k} \in \mathbb{R}^{m \times m},$$
$$\mathbb{U} := \overline{\mathbb{P}}_{m_1 d} \times \cdots \times \overline{\mathbb{P}}_{m_k d} \subset \mathbb{R}^{m \times d},$$

that maximization reads

$$\underset{\mathbf{X} \in \mathbb{U}}{\arg\max} \ \operatorname{tr}(\mathbf{X}^T \mathbf{W} \mathbf{X}). \tag{1}$$

Writing out the constraints explicitly yields the canonical integer program

$$\begin{aligned} \underset{\mathbf{X} \in \{0,1\}^{m \times d}}{\text{maximize}} \quad & \operatorname{tr}\left(\mathbf{X}^T \mathbf{W} \mathbf{X}\right) \\ \text{subject to} \quad & \mathbf{X}_i \mathbb{1}_d \le \mathbb{1}_{m_i} \\ & \mathbf{X}_i^T \mathbb{1}_{m_i} \le \mathbb{1}_d \\ & \mathbb{1}^T \mathbf{X}_i \mathbb{1} = \min(m_i, d), \quad \text{for } i = 1, 2, \ldots, k. \end{aligned} \tag{2}$$

Two features of (2) make it hard. First, $\operatorname{tr}(\mathbf{X}^T \mathbf{W} \mathbf{X})$ is a non-concave quadratic form whenever $\mathbf{W}$ is indefinite, so a direction that improves the objective at one point carries no guarantee about the optimum. Second, the binary row- and column-sum constraints make the problem an NP-hard combinatorial search. The method developed in the next section addresses both difficulties without leaving the permutation domain.

3 CAPS: Exact Block-Coordinate Synchronization

Problem (2) is a quadratic program over a finite, combinatorial set, and $\mathbf{W}$ is indefinite in general. CAPS rests on a single observation, namely that the difficulty arises only when all k objects are considered jointly. Restricted to one object, the objective is exactly affine for any observed matchings, and it can therefore be maximized exactly over the binary feasible set by a linear assignment solver. This section establishes both facts and builds the solver on them.

3.1 The per-object restriction is exactly affine

We regard the objective as a function of the assignment $\mathbf{X}_i$ of a single object, with the assignments of all other objects held fixed. Let $\mathbf{E}_i = \sum_{j \neq i} \mathbf{W}_{ij} \mathbf{X}_j$ denote the *cross-object score* of object i, whose entry (p, ℓ) counts the observed matches between keypoint p of object i and the keypoints of other objects that currently carry label ℓ. The objective then splits as

$$\operatorname{tr}(\mathbf{X}^T \mathbf{W} \mathbf{X}) = \operatorname{tr}(\mathbf{X}_i^T \mathbf{W}_{ii} \mathbf{X}_i) + 2 \operatorname{tr}(\mathbf{X}_i^T \mathbf{E}_i) + \text{const}. \tag{3}$$

The second term is linear in $\mathbf{X}_i$. The first is quadratic in form, but the constraints make it linear, as the following lemma shows.

Lemma 3.1. *For any* $\mathbf{X}_i \in \mathbb{P}_{m_i d}$,

$$\mathbf{X}_i \mathbf{X}_i^T = \operatorname{Diag}\left(\mathbf{X}_i \mathbb{1}_d\right), \tag{4}$$

and consequently, for any *symmetric* $\mathbf{W}_{ii}$,

$$\operatorname{tr}(\mathbf{X}_i^T \mathbf{W}_{ii} \mathbf{X}_i) = \left\langle \operatorname{diag}(\mathbf{W}_{ii}),\, \mathbf{X}_i \mathbb{1}_d \right\rangle, \tag{5}$$

which is linear in $\mathbf{X}_i$.

Proof. Each row of $\mathbf{X}_i$ has at most one nonzero, and the column constraint $\mathbf{X}_i^T \mathbb{1}_{m_i} \le \mathbb{1}_d$ forbids two rows from sharing a column; distinct rows therefore have disjoint support and $(\mathbf{X}_i \mathbf{X}_i^T)_{rs} = 0$ for $r \neq s$. The diagonal entry is $\sum_c (\mathbf{X}_i)_{rc}^2 = \sum_c (\mathbf{X}_i)_{rc} = (\mathbf{X}_i \mathbb{1}_d)_r$, the indicator that row r is assigned. Equation (5) follows from $\operatorname{tr}(\mathbf{X}_i^T \mathbf{W}_{ii} \mathbf{X}_i) = \operatorname{tr}(\mathbf{W}_{ii} \mathbf{X}_i \mathbf{X}_i^T)$. ■

Every off-diagonal entry of $\mathbf{W}_{ii}$ is thus eliminated from the objective by the constraints alone, however dense or indefinite $\mathbf{W}_{ii}$ may be, since two keypoints of the same object never carry the same label. Substituting (5) into (3),

$$\operatorname{tr}(\mathbf{X}^T\mathbf{W}\mathbf{X}) = \langle \mathbf{A}_i,\ \mathbf{X}_i \rangle + \text{const}, \qquad \mathbf{A}_i = 2\,\mathbf{E}_i + \operatorname{diag}(\mathbf{W}_{ii})\mathbb{1}_d^T. \tag{6}$$

The restriction of the objective to one object is therefore affine in $\mathbf{X}_i$, and it remains to show that this affine function can be maximized exactly over the binary feasible set.

3.2 The subproblem is an exact linear assignment

Maximizing (6) over $\overline{\mathbb{P}}_{m_i d}$ is an integer linear program. We show that its linear programming relaxation, in which the binary constraint on $\mathbf{X}_i$ is replaced by $\mathbf{X}_i \geq \mathbf{0}$, is tight, so the solution never leaves the permutation domain. The argument rests on two classical facts.

Definition 3.2 (Total Unimodularity)**.** An integer matrix $\mathbf{A}$ is *totally unimodular* (TU) if the determinant of every square submatrix belongs to $\{-1, 0, 1\}$.

Theorem 3.3 (Total unimodularity and integrality [30])**.** *(i) The incidence matrix of a bipartite graph is totally unimodular. (ii) If* $\mathbf{D}$ *is totally unimodular and* $\mathbf{b}$ *is integral, then every vertex of the polyhedron* $\{\mathbf{x} \geq \mathbf{0} : \mathbf{D}\mathbf{x} \leq \mathbf{b}\}$ *is integral.*

Lemma 3.4. *For every* $\mathbf{A}_i \in \mathbb{R}^{m_i \times d}$*, the linear program*

$$\max \left\{ \langle \mathbf{A}_i, \mathbf{X}_i \rangle :\ \mathbf{X}_i \geq \mathbf{0},\ \mathbf{X}_i \mathbb{1}_d \leq \mathbb{1}_{m_i},\ \mathbf{X}_i^T \mathbb{1}_{m_i} \leq \mathbb{1}_d,\ \mathbb{1}^T \mathbf{X}_i \mathbb{1} = \min(m_i, d) \right\}$$

attains its maximum at a point of $\overline{\mathbb{P}}_{m_i d}$*.*

Proof. Let $\mathbf{x} = \operatorname{vec}(\mathbf{X}_i)$. The row and column constraints read $(\mathbf{I}_{m_i} \otimes \mathbb{1}_d^T)\mathbf{x} \leq \mathbb{1}_{m_i}$ and $(\mathbb{1}_{m_i}^T \otimes \mathbf{I}_d)\mathbf{x} \leq \mathbb{1}_d$, and their stacked matrix $\mathbf{D}$ is the incidence matrix of the complete bipartite graph between the m_i keypoints and the d labels. By Theorem 3.3, $\mathbf{D}$ is totally unimodular and the polytope $Q = \{\mathbf{x} \geq \mathbf{0} : \mathbf{D}\mathbf{x} \leq \mathbb{1}\}$ has integral vertices. Every $\mathbf{x} \in Q$ satisfies $\mathbb{1}^T\mathbf{x} \leq \min(m_i, d)$, so the equality $\mathbb{1}^T\mathbf{x} = \min(m_i, d)$ defines a nonempty face F of Q, whose vertices are vertices of Q and hence integral. A linear function attains its maximum over the polytope F at a vertex, which is a binary matrix with at most one unit entry in each row and each column and exactly $\min(m_i, d)$ unit entries, that is, an element of $\overline{\mathbb{P}}_{m_i d}$. ■

The integral maximizer of Lemma 3.4 is computed directly as a *rectangular linear assignment problem* (LAP), solved exactly by the Jonker–Volgenant algorithm [18]. This gives the CAPS update, which for $i = 1, \ldots, k$ sets

$$\mathbf{X}_i \ \leftarrow\ \arg \max_{\mathbf{X}_i \in \overline{\mathbb{P}}_{m_i d}} \ \left\langle\, 2\,\mathbf{E}_i + \operatorname{diag}(\mathbf{W}_{ii})\mathbb{1}_d^T,\ \mathbf{X}_i \,\right\rangle. \tag{7}$$

Every iterate is an exact partial permutation, and **no thresholding, greedy pass, projection or rounding occurs at any point**. Because the feasible set is the product $\mathbb{U} = \prod_i \overline{\mathbb{P}}_{m_i d}$, changing one block alone always leaves the iterate feasible.

3.3 CLAPS: unrolling the solver into a learned one

Every CAPS iteration has the same two-step shape — it assembles a cost from the other blocks, then solves one exact assignment. Truncating to T layers and replacing every constant by a per-layer parameter gives a finite-depth network. Algorithm 1 states the result; layer t computes

$$\begin{aligned}
&\mathbf{G} = \sigma\big(\phi_\theta(\mathcal{F})\big), \qquad \widetilde{\mathbf{W}} = \mathbf{W} \odot \mathbf{G}, \\
&\mathbf{E}_i = \textstyle\sum_{j \neq i} \widetilde{\mathbf{W}}_{ij}\mathbf{X}_j, \\
&\mathbf{A}_i = a_t\,\mathbf{E}_i / \|\mathbf{E}_i\|_{\max} + b_t\,\mathbf{X}_i + c_t, \\
&\mathbf{X}_i = \arg\max\nolimits_{\mathbf{X}_i} \langle \mathbf{A}_i, \mathbf{X}_i \rangle \quad \text{(LAP with opt-out, solved exactly)},
\end{aligned} \tag{8}$$

where ϕ_θ is a small multilayer perceptron applied to four per-edge features $\mathcal{F}$ — the observation itself, its two- and three-hop cycle support, and the degree imbalance of the pair — producing a learned *trust* $\mathbf{G} \in (0, 1)^{m \times m}$.

Algorithm 1 CLAPS — the unrolled solver, T layers

Require: $\mathbf{W}$; cluster sizes $\{m_i\}$; universe size d; depth T; parameters $\{a_t, b_t, c_t\}_{t=1}^{T}$ and gate weights θ.
Ensure: $\mathbf{X} \in \{0,1\}^{m \times d}$, a partial permutation.

Edge features: $\mathcal{F} \leftarrow [\,\mathbf{W},\ \widehat{\mathbf{W}^2},\ \widehat{\mathbf{W}^3},\ \text{degree imbalance}\,]$ ▷ computed once
Learned trust: $\mathbf{G} \leftarrow \sigma(\phi_\theta(\mathcal{F})) \in (0,1)^{m \times m}$, $\widetilde{\mathbf{W}} \leftarrow \mathbf{W} \odot \mathbf{G}$
Warm start: $\mathbf{X}^{(0)} \leftarrow$ block projection of the Stiefel solution
for $t = 1, \ldots, T$ **do**
  for each object i **do**
    $\mathbf{E}_i \leftarrow \sum_{j \neq i} \widetilde{\mathbf{W}}_{ij} \mathbf{X}_j$ ▷ cross-object scores
    $\mathbf{A}_i \leftarrow a_t\, \mathbf{E}_i / \|\mathbf{E}_i\|_{\max} + b_t\, \mathbf{X}_i + c_t$
    $\mathbf{X}_i \leftarrow \arg\max_{\mathbf{X}_i} \langle \mathbf{A}_i, \mathbf{X}_i \rangle$ ▷ exact LAP with opt-out
  end for
end for
return $\mathbf{X}^{(T)}$

The layer departs from (7) in three ways. First, b_t rewards keeping the current label; CAPS sets it to 0 because the affine restriction (6) contains no such term, and learning it lets training decide how much weight the current label receives. Second, c_t adds the linear term $\langle \mathbf{C}, \mathbf{X} \rangle$ with $\mathbf{C} = c_t \mathbb{1}\mathbb{1}^T$ to the objective, so a negative c_t places a price on every assignment and allows the model to *decline* an outlier. For this purpose the assignment step is solved as a maximum-weight matching in which a keypoint may remain unassigned. It is implemented by appending m_i auxiliary columns of value $-\varepsilon$, for a small $\varepsilon > 0$, so that each keypoint is matched either to a label or to an auxiliary column. The augmented problem is still a bipartite assignment problem, so its optimum is integral and is computed exactly. If $\mathbf{A}_i$ is entrywise nonnegative, no keypoint gains from being declined, and the step maximizes $\langle \mathbf{A}_i, \mathbf{X}_i \rangle$ over $\overline{\mathbb{P}}_{m_i d}$, exactly as the CAPS update (7) does.

Third, the network is **relaxation-free at both training and test time**, in the sense that tightness is never lost. Gradients are carried by an entropically regularized transport plan, whose cost matrix is augmented with one extra row and one extra column that hold a learned opt-out score and absorb the keypoints left unassigned and the labels left unused. This plan enters the backward pass only. The forward value is always the exact assignment (straight-through), so every iterate, in training and inference alike, solves the assignment problem to optimality. Setting $a_t = 1$, $b_t = c_t = 0$ and $\mathbf{G} \equiv \mathbb{1}\mathbb{1}^T$ recovers (7) exactly (Appendix E). Training minimizes a margin loss on the cost matrix $\mathbf{A}_i$ — the quantity the assignment step consumes — together with a binary cross-entropy on $\mathbf{G}$ against whether each observed match is correct.

Guarantees and cost. Because the per-object restriction is *exactly* affine (Lemma 3.1), each step of CAPS maximizes the true objective of problem (2), and Appendix C derives the resulting guarantees. The objective ascends monotonically, and under the strict-improvement rule of that appendix the iterate is constant after finitely many passes and the limit is block-wise optimal with every conditional subproblem solved to its *exact* optimum. This combinatorial 1-opt guarantee holds for problem (2) itself. Relaxation-based methods, by contrast, establish their properties for a looser problem, and these do not carry over to the solution they return after rounding. One pass costs $O(\mathrm{nnz}(\mathbf{W}) + d^2 m)$, linear in the number of *observed* matches, and CLAPS, with fixed depth T, costs T passes plus the evaluation of its trust function, as derived in Appendix C.3.

4 Experiments

4.1 Benchmarks and evaluation protocol

We evaluate on the SPair-71k and Willow Object benchmarks [10, 24]. Multi-image matching datasets are commonly assembled by storing one keypoint slot per semantic annotation index, in the same order for every image. That convention makes the ground-truth correspondence the *block identity* $\mathbf{P}_{\mathrm{gt}} = \mathbb{1}_k \mathbb{1}_k^T \otimes \mathbf{I}_d$, so a solver returning the identity attains a perfect score without ever consulting $\mathbf{W}$. The instances used here are free of this flaw and of three related ones.

Ground truth is read from the datasets' own annotations, with no correspondence aligned by hand. On SPair-71k, two keypoints correspond exactly when the dataset assigns them the same semantic identity, and on Willow Object, exactly when they occupy the same position in the dataset's fixed

annotation order. The keypoint list of each image is then randomly permuted, so that a row index carries no information about its universe label and the identity no longer coincides with the ground truth. Each image also receives $\mathrm{round}(d/k)$ *distractors*, the strongest SIFT detections inside the object box that lie at least $12\,\mathrm{px}$ from every annotated keypoint. These carry no universe label and have no correspondent in any other image, so $m_i > d$ and the setting is partial. Finally, the observations $\mathbf{W}_{ij}$ come from a real matcher, namely mutual nearest-neighbor matching under a cost that combines a SIFT descriptor at each keypoint with a shape context computed over the *entire* keypoint set of the image. The matcher sees no label and is not told which keypoints are co-visible, no synthetic perturbation is injected, and the diagonal blocks are $\mathbf{W}_{ii} = \mathbf{I}$.

Images are split into train, validation and test sets *before* any instance is assembled, so no image is shared across splits and the unrolled model never sees a test image during training. We report 140 SPair-71k test instances over 14 categories and 40 Willow Object test instances over 4 classes.

4.2 Competing methods and metrics

We compare against eight baselines — Spectral [25], NmfSync [5], Stiefel [7], FCC [32], MatchFAME [21], and the local-search solvers SEQ-C + SEQ-LS and PAR-C + PAR-LS [19] together with GREEDA [20]. What each of them does is described in Appendix B. To make the comparison a fair one, each is run from the implementation its authors have published and at the parameter settings their own paper specifies. FCC's thresholded output assigns one keypoint to as many as five partners and is not a matching, so we report *FCC + exact rounding*, its scores put through the same exact block LAP applied to every other baseline. Every method's output passes through that same rounding, which leaves ours unchanged, since every CLAPS iterate is already a partial permutation. Our solvers start from the Stiefel solution projected onto partial permutations, while each baseline runs with the initialization its own authors specify. The guarantees hold from any feasible starting point, and the Stiefel solution is the one under which the reported results were obtained. Appendix F.1 reports an ablation on the sensitivity to this choice, repeating the evaluation from a spectral, an identity and a random start, with the Stiefel solution the strongest of the four in F-score.

We report F-score and precision against the annotation, together with the cycle agreement defined below and the runtime.

Cycle consistency as a design requirement. It is worth mentioning that cycle consistency is a property that every valid solution must satisfy, as also stressed in [20]. A family of pairwise matchings that violates it is not a synchronization, so we require it of every method that enters the comparison. Methods that do not enforce it, among them MatchEig [23] as well as the pairwise feature matchers SuperGlue [27] and LightGlue [22], are therefore not comparable in this paper; the measured cycle agreement of MatchEig on SPair-71k, for instance, is only 50%. Every method we report is cycle-consistent by construction, ours included, because each returns $\mathbf{P} = \mathbf{X}\mathbf{X}^T$ for a single universe assignment $\mathbf{X}$, and we checked this on every test instance without finding a contradiction.

In the partial setting, two natural ways of quantifying cycle consistency behave differently. Since $\mathbf{P}_{ij}\mathbf{P}_{jk} = \mathbf{X}_i \,\mathrm{Diag}(\mathbf{s}_j)\mathbf{X}_k^T \leq \mathbf{P}_{ik}$, where $\mathbf{s}_j$ indicates the labels object j carries, composition through j can *drop* a correspondence whose label j lacks, but can never assert a different one. An intersection-over-union score between $\mathbf{P}_{ik}$ and $\mathbf{P}_{ij}\mathbf{P}_{jk}$ therefore captures incompleteness, whereas the agreement, the fraction of features that agree among those for which the composed path and the direct match both exist, reflects consistency. Both quantities are recorded, per instance, in the code and data submitted with this paper.

4.3 Quantitative results

Tables 1 and 2 report means over the test instances; per-category tables appear in Appendix G.

Accuracy. As we can see, CLAPS attains the best F-score and the best precision on both benchmarks, $63.9/68.5\%$ on Willow against $62.7/62.7\%$ for the strongest baseline, and $40.0/43.2\%$ on SPair-71k against $39.1/36.1\%$. Against the Stiefel solution it begins from, the refinement adds 2.8 points of F-score and 7.3 of precision on Willow, and 1.5 and 7.6 on SPair-71k, so the margin is due to the update and not to the starting point. The advantage holds category by category. CLAPS takes the best precision in every class of both benchmarks, and its F-score is competitive with the best in

Table 1: Synchronization on **Willow Object**, mean over 40 test instances. **Best** and second best per column; our row is shaded.

Method	F-score (%) ↑	Precision (%) ↑	Cyclic (%) ↑	Time (s) ↓
Spectral (2013) [25]	61.6	61.6	100.0	**0.001**
NmfSync (2019) [5]	62.1	62.1	100.0	0.022
Stiefel (2021) [7]	61.1	61.2	100.0	0.011
FCC + exact rounding (2021) [32]	62.7	62.7	100.0	0.006
MatchFAME (2022) [21]	45.6	45.6	100.0	0.053
SEQ-C + SEQ-LS (2024) [19]	59.4	54.4	100.0	0.070
PAR-C + PAR-LS (2024) [19]	59.3	54.3	100.0	0.057
GREEDA (2025) [20]	59.6	54.6	100.0	0.064
CLAPS (ours)	**63.9**	**68.5**	100.0	0.019

Table 2: Synchronization on **SPair-71k**, mean over 140 test instances. **Best** and second best per column; our row is shaded.

Method	F-score (%) ↑	Precision (%) ↑	Cyclic (%) ↑	Time (s) ↓
Spectral (2013) [25]	38.1	35.2	100.0	**0.003**
NmfSync (2019) [5]	38.5	35.6	100.0	0.062
Stiefel (2021) [7]	38.5	35.6	100.0	0.028
FCC + exact rounding (2021) [32]	39.1	36.1	100.0	0.011
MatchFAME (2022) [21]	31.0	28.4	100.0	0.029
SEQ-C + SEQ-LS (2024) [19]	38.3	34.1	100.0	0.073
PAR-C + PAR-LS (2024) [19]	38.2	34.0	100.0	0.072
GREEDA (2025) [20]	38.6	34.4	100.0	0.088
CLAPS (ours)	**40.0**	**43.2**	100.0	0.038

every class and ahead of every baseline on each Willow Object class (Appendix G). The local-search solvers SEQ-C + SEQ-LS, PAR-C + PAR-LS and GREEDA sit below it on both counts on SPair, at 38.2–38.6% of F-score and a precision of only 34.0–34.4%, because they assign every keypoint including the distractors that have no correct partner anywhere.

Qualitative comparison. Figure 2 shows the annotation ground truth alongside the correspondences each method recovers on a single image pair. The pair is the one on which our margin over the strongest displayed baseline is largest; per-category panels chosen by the same rule are given in Appendix H. Our panel carries fewer red lines terminating on hollow dots, since the learned unary price declines a distractor and leaves it unassigned.

The learned constants. Training leaves the cost of CAPS nearly intact, with $a_t \approx 0.98$ and $b_t \approx -0.02$ on Willow and $a_t \approx 0.93$ and $b_t \approx -0.068$ on SPair, which is what one expects if the cost of the exact solver is already close to the one the data prefers. What training adds is a price on assignment, $c_t < 0$ in every layer, near -0.014 on Willow and growing with depth from -0.027 to -0.054 on SPair. That price lets the model decline outliers and is the source of the precision gain.

Efficiency. The solve itself takes 8–10 ms, so refining the Stiefel start into the reported matching costs about that much on top of it. End to end a single instance takes 19–38 ms, below NmfSync (22–62 ms) and well below the local-search solvers (57–88 ms). The cheaper baselines — Spectral, FCC and Stiefel, at 1–28 ms, and MatchFAME on SPair-71k — each reach a lower F-score and a markedly lower precision, so the additional time is what buys the accuracy the tables report. The learned component is small, 113 parameters in all — twelve per-layer constants, four opt-out scores and a 97-parameter trust function — and it costs nothing appreciable at run time, since its layers perform the same assembly-plus-assignment as the iterative solver, truncated to $T = 4$.

5 Limitations

Problem (2) is NP-hard, so no polynomial-time algorithm is known to return an optimizer of it as a whole. What CAPS establishes is monotone ascent, finite termination and block-wise optimality of the limit, with each per-object subproblem solved to its exact optimum.

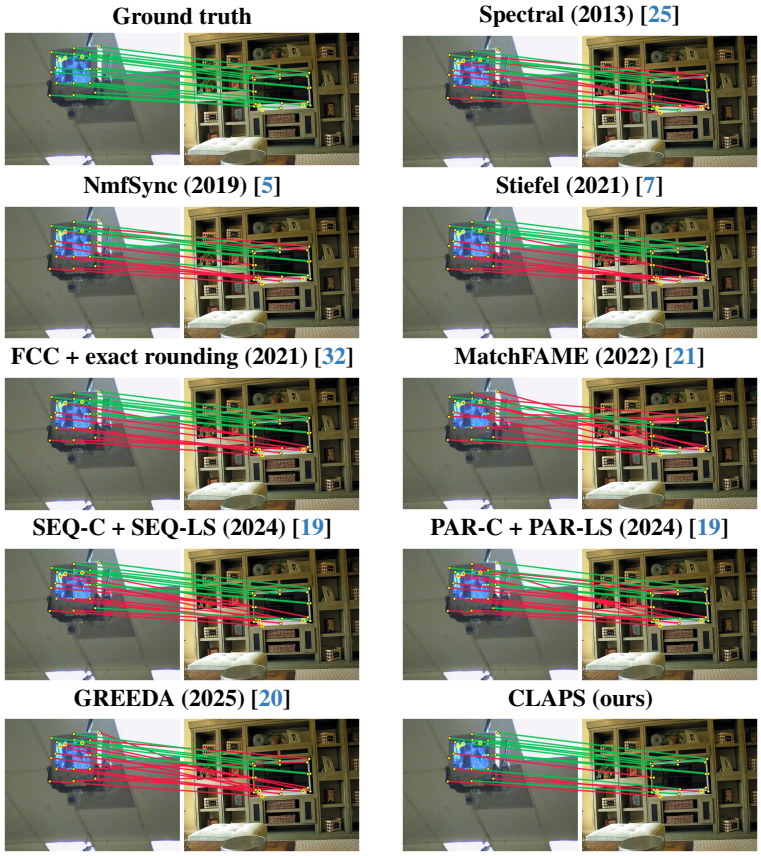

Figure 2: Correspondences on an SPair-71k *tvmonitor* pair, one panel per method. Green means correct and red wrong; filled dots are annotated keypoints, hollow dots distractors.

6 Conclusion

We have presented CAPS, a direct combinatorial solver for permutation synchronization, which never passes through a relaxation that loses tightness. We showed that feasibility alone forces $\mathbf{X}_i\mathbf{X}_i^T$ to be diagonal, so the off-diagonal entries of every self-block drop out of the objective, and its per-object restriction is *exactly affine* for any symmetric $\mathbf{W}$, with no further assumption. Each update is therefore a rectangular linear assignment problem with a totally unimodular constraint matrix, which is solved exactly, so no rounding step is ever needed. Every iterate is a partial permutation, and cycle consistency is built into the parameterization, with no constraint imposed afterwards. We proved that CAPS increases the objective monotonically and terminates after finitely many passes at a block-wise optimal assignment, and that each pass costs time linear in the number of observed matches. We further unrolled CAPS into a lightweight network, CLAPS, which learns per-layer constants and a per-edge trust function end-to-end and can decline outliers, while every assignment step remains exact. Because CLAPS contains CAPS as a special case, training begins from that solver and improves upon it. On the SPair-71k and Willow Object benchmarks, CLAPS outperforms the state of the art in both precision and F-score.

AI use statement

AI-assisted editing was used for language editing and proofreading of the main paper and supplementary materials. The authors take full responsibility for the final content of this paper.

Reproducibility statement

Every number this paper reports is produced by the code and the data that accompany the submission as supplementary material. This material contains CAPS and CLAPS as Section 3 and Subsection 3.3

describe them, the benchmark construction of Subsection 4.1, all 540 instances the experiments are run on, the trained models, and the scripts that emit the tables and the figures the paper typesets. A single command per dataset reproduces our rows of Tables 1 and 2, of the per-category tables of Appendix G and of the ablation of Appendix F, and a further script reproduces the initialization ablation of Appendix F.1. All of this runs from the shipped instances alone, on a single desktop CPU, so neither SPair-71k nor Willow Object has to be downloaded. Every runtime this paper reports was measured on one machine, an Intel Core i7-13700K with 16 cores and 24 threads; no GPU is used at any stage, training included. The warm start of every instance is stored beside it, so a run reproduces the reported values without depending on the state of any other solver. How the instances themselves are built is specified in Subsection 4.1 and implemented in the submitted code. That code fixes how the ground truth is read from each dataset's own annotation, how the keypoint order is randomized, how the distractors are detected, how the observed matchings are produced, and how the images are divided into splits before any instance is assembled. The scores of the compared methods on these instances are included with the results. Appendix C proves Propositions 1–4 and derives the complexity, stating in each case the assumption the result rests on. Appendices D and E give CAPS and CLAPS line by line, together with the test, included in the code, that the second reduces to the first. The pair shown in each qualitative figure and the rule that selects it are recorded in the supplementary material.

REFERENCES

[1] Federica Arrigoni, Beatrice Rossi, and Andrea Fusiello. Spectral synchronization of multiple views in SE(3). *SIAM Journal on Imaging Sciences*, 9(4):1963–1990, 2016. doi: 10.1137/16M1060248. URL https://doi.org/10.1137/16M1060248. 1

[2] Jérôme Berclaz, François Fleuret, Engin Türetken, and Pascal Fua. Multiple object tracking using K-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1806–1819, 2011. 1

[3] Florian Bernard, Johan Thunberg, Peter Gemmar, Frank Hertel, Andreas Husch, and Jorge Goncalves. A solution for multi-alignment by transformation synchronisation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2161–2169, June 2015. doi: 10.1109/CVPR.2015.7298828. 1

[4] Florian Bernard, Christian Theobalt, and Michael Moeller. DS*: Tighter lifting-free convex relaxations for quadratic matching problems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4310–4319, June 2018. doi: 10.1109/CVPR.2018.00453. 1

[5] Florian Bernard, Johan Thunberg, Jorge Goncalves, and Christian Theobalt. Synchronisation of partial multi-matchings via non-negative factorisations. *Pattern Recognition*, 92:146–155, 2019. 2, 3, 7, 8, 9, 13, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34

[6] Florian Bernard, Johan Thunberg, Paul Swoboda, and Christian Theobalt. HiPPI: Higher-order projected power iterations for scalable multi-matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10283–10292, 2019. doi: 10.1109/ICCV.2019.01038. 1

[7] Florian Bernard, Daniel Cremers, and Johan Thunberg. Sparse quadratic optimisation over the Stiefel manifold with application to permutation synchronisation. *Advances in Neural Information Processing Systems*, 34:25256–25266, 2021. 2, 7, 8, 9, 13, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34

[8] Tolga Birdal and Umut Simsekli. Probabilistic permutation synchronization using the Riemannian structure of the Birkhoff polytope. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11097–11108, 2019. doi: 10.1109/CVPR.2019.01136. 1

[9] Tolga Birdal, Vladislav Golyanik, Christian Theobalt, and Leonidas J Guibas. Quantum permutation synchronization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13117–13128, 2021. doi: 10.1109/CVPR46437.2021.01292. 1

[10] Minsu Cho, Karteek Alahari, and Jean Ponce. Learning graphs to match. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 25–32, 2013. doi: 10.1109/ICCV.2013.11. 6

[11] Timothy F Cootes and Christopher J Taylor. Active shape models—'smart snakes'. In *BMVC92: Proceedings of the British Machine Vision Conference, organised by the British Machine Vision Association 22–24 September 1992 Leeds*, pages 266–275. Springer, 1992. 1

[12] Maolin Gao, Zorah Lahner, Johan Thunberg, Daniel Cremers, and Florian Bernard. Isometric multi-shape matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14178–14188, 2021. doi: 10.1109/CVPR46437.2021.01396. 1

[13] Tobias Heimann and Hans-Peter Meinzer. Statistical shape models for 3D medical image segmentation: a review. *Medical Image Analysis*, 13(4):543–563, 2009. 1

[14] Qi-Xing Huang and Leonidas Guibas. Consistent shape maps via semidefinite programming. In *Computer Graphics Forum*, volume 32, pages 177–186. Wiley Online Library, 2013. 1

[15] Qixing Huang, Zhenxiao Liang, Haoyun Wang, Simiao Zuo, and Chandrajit Bajaj. Tensor maps for synchronizing heterogeneous shape collections. *ACM Transactions on Graphics (TOG)*, 38 (4):1–18, 2019. 1

[16] Ruqi Huang, Jing Ren, Peter Wonka, and Maks Ovsjanikov. Consistent ZoomOut: Efficient spectral map synchronization. In *Computer Graphics Forum*, volume 39, pages 265–278. Wiley Online Library, 2020. 1

[17] Xiangru Huang, Zhenxiao Liang, Xiaowei Zhou, Yao Xie, Leonidas J Guibas, and Qixing Huang. Learning transformation synchronization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8074–8083, 2019. doi: 10.1109/CVPR.2019.00827. 1

[18] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340, 1987. doi: 10.1007/BF02278710. 5

[19] Max Kahl, Sebastian Stricker, Lisa Hutschenreiter, Florian Bernard, and Bogdan Savchynskyy. Unlocking the potential of operations research for multi-graph matching. arXiv preprint arXiv:2406.18215v1, 2024. 1, 2, 7, 8, 9, 14, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34

[20] Max Kahl, Sebastian Stricker, Lisa Hutschenreiter, Florian Bernard, Carsten Rother, and Bogdan Savchynskyy. Towards optimizing large-scale multi-graph matching in bioimaging. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11569–11578, 2025. doi: 10.1109/CVPR52734.2025.01080. 1, 2, 7, 8, 9, 14, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34

[21] Shaohan Li, Yunpeng Shi, and Gilad Lerman. Fast, accurate and memory-efficient partial permutation synchronization. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15714–15722, 2022. doi: 10.1109/CVPR52688.2022.01528. 2, 7, 8, 9, 14, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34

[22] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. LightGlue: Local feature matching at light speed. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17581–17592, 2023. doi: 10.1109/ICCV51070.2023.01616. 7

[23] Eleonora Maset, Federica Arrigoni, and Andrea Fusiello. Practical and efficient multi-view matching. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4578–4586, Oct 2017. doi: 10.1109/ICCV.2017.489. 7

[24] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. SPair-71k: A large-scale benchmark for semantic correspondence. *arXiv preprint arXiv:1908.10543*, 2019. 6

[25] Deepti Pachauri, Risi Kondor, and Vikas Singh. Solving the multi-way matching problem by permutation synchronization. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/3df1d4b96d8976ff5986393e8767f5b2-Paper.pdf. 1, 4, 7, 8, 9, 13, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34

[26] Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. *Journal of the ACM (JACM)*, 23(3):555–565, 1976. 1

[27] 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 (CVPR)*, pages 4937–4946, 2020. doi: 10.1109/CVPR42600.2020.00499. 7

[28] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2538–2547, 2017. doi: 10.1109/CVPR.2017.272. 1

[29] Thomas Schöps, Torsten Sattler, and Marc Pollefeys. BAD SLAM: Bundle adjusted direct RGB-D SLAM. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 134–144, 2019. doi: 10.1109/CVPR.2019.00022. 1

[30] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., USA, 1986. ISBN 0471908541. 5

[31] Yanyao Shen, Qixing Huang, Nati Srebro, and Sujay Sanghavi. Normalized spectral map synchronization. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/bb03e43ffe34eeb242a2ee4a4f125e56-Paper.pdf. 1, 13

[32] Yunpeng Shi, Shaohan Li, Tyler Maunu, and Gilad Lerman. Scalable cluster-consistency statistics for robust multi-object matching. In *2021 International Conference on 3D Vision (3DV)*, pages 352–360, 2021. doi: 10.1109/3DV53792.2021.00045. 2, 7, 8, 9, 13, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34

[33] Aristeidis Sotiras, Christos Davatzikos, and Nikos Paragios. Deformable medical image registration: A survey. *IEEE Transactions on Medical Imaging*, 32(7):1153–1190, 2013. 1

[34] Paul Swoboda, Dagmar Kainmüller, Ashkan Mokarian, Christian Theobalt, and Florian Bernard. A convex relaxation for multi-graph matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11148–11157, 2019. doi: 10.1109/CVPR.2019.01141. 1

[35] Roberto Tron, Xiaowei Zhou, Carlos Esteves, and Kostas Daniilidis. Fast multi-image matching via density-based clustering. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4077–4086, 2017. doi: 10.1109/ICCV.2017.437. 3

[36] Junchi Yan, Xu-Cheng Yin, Weiyao Lin, Cheng Deng, Hongyuan Zha, and Xiaokang Yang. A short survey of recent advances in graph matching. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, pages 167–174, 2016. 1

[37] Li Zhang, Yuan Li, and Ramakant Nevatia. Global data association for multi-object tracking using network flows. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008. doi: 10.1109/CVPR.2008.4587584. 1

[38] Xiaowei Zhou, Menglong Zhu, and Kostas Daniilidis. Multi-image matching via fast alternating minimization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4032–4040, 2015. 1

[39] Barbara Zitová and Jan Flusser. Image registration methods: A survey. *Image and Vision Computing*, 21(11):977–1000, 2003. 1

Supplementary Material

A Notation

Bold uppercase letters stand for matrices and bold lowercase letters for vectors, $\mathbf{A}^T$ is the transpose of $\mathbf{A}$, and $\langle \mathbf{A}, \mathbf{B} \rangle = \operatorname{tr}(\mathbf{A}^T\mathbf{B})$ is the trace inner product. We write $\mathbb{1}_p$ for the all-ones vector of length p and $\mathbf{I}_p$ for the $p \times p$ identity, dropping the subscript wherever the size is clear from the context, and every vector inequality is read entry-wise. For a square $\mathbf{A}$, $\operatorname{diag}(\mathbf{A})$ collects the diagonal entries into a vector and $\operatorname{Diag}(\mathbf{a})$ places $\mathbf{a}$ on the diagonal of a matrix; $\odot$ denotes the entry-wise product and $\otimes$ the Kronecker product. The number of nonzeros of $\mathbf{W}$ is $\operatorname{nnz}(\mathbf{W})$, and $\|\mathbf{A}\|_{\max}$ is the largest absolute entry of $\mathbf{A}$.

The collection holds k objects, object i contributing m_i keypoints, and we set $m = \sum_{i=1}^{k} m_i$, while d is the number of shared universe labels. Partial permutations of size $p \times q$ form $\mathbb{P}_{pq}$, their maximal members form $\overline{\mathbb{P}}_{pq}$, and $\mathbb{U} = \overline{\mathbb{P}}_{m_1 d} \times \cdots \times \overline{\mathbb{P}}_{m_k d}$ is the feasible set the solver never leaves. The observed matching between objects i and j is $\mathbf{X}_{ij}$, these blocks assemble into $\mathbf{W} = [\mathbf{X}_{ij}]_{i,j=1}^{k}$, the assignment of object i to the universe is $\mathbf{X}_i$, and the blocks $\mathbf{X}_i$ stack into $\mathbf{X}$. The objective is $f(\mathbf{X}) = \operatorname{tr}(\mathbf{X}^T\mathbf{W}\mathbf{X})$, $\mathbf{E}_i$ is the cross-object score of object i and $\mathbf{A}_i$ the cost matrix its assignment step consumes.

For the unrolled solver, T is the depth, a_t, b_t and c_t are the constants of layer t, θ collects the weights of the trust function ϕ_θ, σ is the logistic function, and $\mathbf{G}$ is the learned per-edge trust that gates $\mathbf{W}$.

B Related work and the methods compared

Two routes dominate the treatment of permutation synchronization, and the methods the main paper compares against divide between them. One leaves the permutation domain for a continuous surrogate, optimizes there and returns by rounding; the other keeps matchings throughout and improves them by trial exchanges. This section describes what each of them does and where its difficulty lies, so that Tables 1 and 2 can be read without recourse to the original papers. The methods appear in the order of their publication.

Spectral (2013) [25] and its normalized form (2016) [31]. The universe assignments are read off the d leading eigenvectors of $\mathbf{W}$, which converts (1) into an eigenvalue problem, and the later variant normalizes the blocks before the same step is taken. The difficulty is the embedding itself. The eigenvectors satisfy an orthogonality condition and none of the row and column constraints that define $\mathbb{U}$, so the matching that is finally reported comes out of a rounding step the objective never sees. The surrogate is also posed over full bijections, so it cannot represent an object carrying keypoints with no correspondent anywhere, the regime both benchmarks exhibit.

NmfSync (2019) [5]. A non-negative factorization of $\mathbf{W}$ supplies the universe assignments, and unlike the spectral treatments it is posed for partial matchings from the start. The factorization is nevertheless continuous, so a projection onto partial permutations closes the method, and that projection, independent of the criterion being optimized, decides which correspondences survive.

Stiefel (2021) [7]. The problem is posed over the Stiefel manifold and driven by a sparsity-promoting variant of orthogonal iteration. Its output is a continuous soft assignment whose entries approach zero without reaching it. The sparsity subproblem it relies on is itself non-convex, and no binary solution is guaranteed.

FCC (2021) [32]. A cluster-consistency statistic scores how well each observed match agrees with the rest of the collection, and the low-scoring matches are removed. The threshold returns a filtered set of observations and no assignment, and on these instances it carries one keypoint to as many as five partners at once, so it is not a matching. The main paper therefore reports it after the same exact block assignment every other baseline is put through.

MatchFAME (2022) [21]. An iterative scheme estimates cycle consistency on large sparse collections and synchronizes the partial permutations from that estimate. The estimate is again formed outside the permutation domain and reaches it only through a final projection, so the guarantee that the reported matching maximizes anything is lost at the last step.

SEQ-C + SEQ-LS and PAR-C + PAR-LS (2024) [19], GREEDA (2025) [20]. These construct a matching and then refine it by local search over exchanges, the construction sequential in the first case and parallel in the second, while GREEDA takes the same recipe to sparse and incomplete instances behind an added swap-based refinement. None of them carries a convergence guarantee, since a heuristic objective is improved by trial exchanges with no formal assurance that even a local optimum of the synchronization criterion is reached, and the returned matching depends on where the exchanges terminate. They also assign every keypoint, distractors included, which is what the precision columns of Tables 1 and 2 record.

C Convergence and Complexity

This section proves that CAPS, summarized in Algorithm S1, increases the objective of problem (2) monotonically, that its iterates become constant after finitely many passes, and that the assignment at which they stop is block-wise optimal. It then derives the computational complexity of CAPS and of CLAPS. Throughout, $f(\mathbf{X}) = \operatorname{tr}(\mathbf{X}^T\mathbf{W}\mathbf{X})$ is the objective and $\mathbb{U}$ the feasible set of problem (2), which is finite. We write $\mathbf{X}^{(t)}$ for the iterate after pass t, where one pass updates the objects $i = 1, \dots, k$ in turn through (7).

A block may admit several maximizers of the same value. For the finiteness results, we therefore adopt the *strict-improvement rule*, under which the assignment of an object is replaced only when the replacement strictly increases the objective. Propositions 2 and 3 rely on this rule. Proposition 1 does not, and the property stated in Proposition 4 holds at any iterate that a complete pass leaves unchanged. The runs reported in Appendix F accept every exact maximizer, as each layer of CLAPS does, so that the two solvers are compared there under the same update.

C.1 The objective

Proposition 1 (Monotone ascent)**.** *For every t, $f(\mathbf{X}^{(t+1)}) \geq f(\mathbf{X}^{(t)})$. Under the strict-improvement rule, the inequality is strict whenever the pass changes a block.*

Proof. We consider the update of a single object i, with all other blocks held fixed. By Lemma 3.1, $\operatorname{tr}(\mathbf{X}_i^T\mathbf{W}_{ii}\mathbf{X}_i) = \langle \operatorname{diag}(\mathbf{W}_{ii}),\, \mathbf{X}_i\mathbb{1}_d\rangle = \langle \operatorname{diag}(\mathbf{W}_{ii})\mathbb{1}_d^T,\, \mathbf{X}_i\rangle$, so, collecting into a constant every term that does not involve object i,

$$f(\mathbf{X}) \;=\; \big\langle \mathbf{A}_i,\; \mathbf{X}_i \big\rangle + \text{const}, \qquad \mathbf{A}_i \;=\; 2\,\mathbf{E}_i + \operatorname{diag}(\mathbf{W}_{ii})\mathbb{1}_d^T, \tag{S1}$$

which is (6). Step (7) returns an *exact* maximizer of $\langle \mathbf{A}_i, \cdot\rangle$ over $\overline{\mathbb{P}}_{m_i d}$, by Lemma 3.4 and the Jonker–Volgenant algorithm. Since the current assignment of object i is itself one of the candidates, no step of the pass can decrease f. Under the strict-improvement rule, an assignment is replaced only when the new value exceeds the current one, so each replacement increases f strictly. ■

Monotone ascent therefore requires neither convexity of f nor positive semidefiniteness of $\mathbf{W}$, and it makes no assumption on the diagonal blocks $\mathbf{W}_{ii}$, since the argument uses feasibility alone.

Proposition 2 (Finite termination and an iteration bound)**.** *Under the strict-improvement rule, CAPS terminates after finitely many passes. Moreover, when $\mathbf{W}$ has integer entries, as it does whenever it is assembled from observed matchings, the number of passes that change a block is at most*

$$f^\star - f(\mathbf{X}^{(0)}), \tag{S2}$$

where $f^\star$ is the value at termination.

Proof. By Proposition 1, every pass that changes a block increases f strictly, so the iterates that such passes produce are pairwise distinct. As $\mathbb{U}$ is finite, only finitely many passes can therefore change a block, and the first pass that changes none terminates the algorithm. When $\mathbf{W}$ is integral, f takes

integer values on $\mathbb{U}$, because $\mathbf{X}$ is binary, so every pass that changes a block increases f by at least 1. Since f is nondecreasing along the iterates and bounded above by its value $f^\star$ at termination, at most $f^\star - f(\mathbf{X}^{(0)})$ such passes can occur. ■

C.2 The sequence of iterates

Convergence of the objective values does not by itself imply convergence of the iterates. If moves that leave the objective unchanged were accepted, the iterates could alternate indefinitely between distinct maximizers of the same value. The strict-improvement rule excludes this behavior, and the next proposition shows that the sequence of iterates itself becomes constant after finitely many passes.

Proposition 3 (The iterate sequence is eventually constant)**.** *Under the strict-improvement rule, there exists a finite $T^\star$ such that $\mathbf{X}^{(t)} = \mathbf{X}^{(T^\star)}$ for all $t \geq T^\star$. The sequence of iterates therefore converges, and it coincides with its limit after finitely many passes.*

Proof. By Proposition 2, only finitely many passes change a block. Let $T^\star$ be the index of the last such pass, with $T^\star = 0$ if there is none. Every subsequent pass leaves all k blocks unchanged, hence $\mathbf{X}^{(t+1)} = \mathbf{X}^{(t)}$ for all $t \geq T^\star$. ■

Proposition 4 (Block-wise optimality of the limit)**.** $\mathbf{X}^\star := \mathbf{X}^{(T^\star)}$ *is* block-wise optimal*, in that for every object i,*

$$\mathbf{X}_i^\star \in \arg\max_{\mathbf{X}_i \in \mathbb{P}_{m_i d}} f\big(\mathbf{X}_1^\star, \ldots, \mathbf{X}_i, \ldots, \mathbf{X}_k^\star\big). \tag{S3}$$

Equivalently, no change confined to a single object can improve the objective, and each such conditional problem is solved to its exact *optimum.*

Proof. No block is replaced in the pass that follows $T^\star$. Each update of that pass computes an exact maximizer of the affine restriction (6) (proof of Proposition 1), and under the strict-improvement rule it keeps the current block only when no candidate attains a larger value. Hence $\mathbf{X}_i^\star$ attains the maximum for every object i. ■

C.3 Computational complexity

We now analyze the computational complexity of one pass of CAPS. Let $\mathrm{nnz}(\mathbf{W})$ denote the number of observed matches, $m = \sum_i m_i$ the total number of keypoints and d the universe size. For every object, a pass assembles the cross-object score $\mathbf{E}_i$ and solves one assignment problem.

First, consider the assembly of the scores. Since each row of $\mathbf{X}$ carries at most one nonzero entry, $\mathbf{W}_{pq}$ contributes to the single cell $(\mathbf{WX})_{p,\,\ell(q)}$ of $\mathbf{WX}$, where $\ell(q)$ denotes the label assigned to keypoint q. This product can therefore be accumulated by a single scan over the nonzero entries of $\mathbf{W}$, at a cost of $O(\mathrm{nnz}(\mathbf{W}))$, well below the $O(m^2 d)$ of a dense multiplication. As in Algorithm S1, the product $\mathbf{S} = \mathbf{WX}$ is kept throughout the pass and updated after each block, so that one scan suffices. Next, each per-object assignment problem is solved by the Jonker–Volgenant algorithm, whose cost on an $m_i \times d$ instance is $O(\min(m_i, d)^2 \max(m_i, d))$. This cost is at most $O(d^2 m_i)$, whether or not $d \leq m_i$, and summing over the k objects gives $O(d^2 m)$. In sum, the computational complexity of one pass of CAPS is

$$\underbrace{O\big(\mathrm{nnz}(\mathbf{W})\big)}_{\text{scores } \mathbf{E}_i} + \underbrace{\textstyle\sum_{i=1}^{k} O\big(\min(m_i, d)^2 \max(m_i, d)\big)}_{k \text{ rectangular LAPs}} = O\big(\mathrm{nnz}(\mathbf{W}) + d^2 m\big), \tag{S4}$$

which is linear in the number of observed matches.

Each of the T layers of CLAPS performs the same two steps and, in addition, evaluates the trust function, a multilayer perceptron of hidden width h, on the observed matches, at a cost of $O(h\,\mathrm{nnz}(\mathbf{W}))$. The four features on which the trust function acts are computed once per instance. Since each keypoint has at most one observed partner in every other object, each row of $\mathbf{W}$ has at most k nonzero entries, so the two- and three-hop supports of all observed matches can be accumulated in $O(k\,\mathrm{nnz}(\mathbf{W}))$ and $O(k^2\,\mathrm{nnz}(\mathbf{W}))$ operations, respectively. The computational complexity of CLAPS is therefore

$$O\big(k^2\,\mathrm{nnz}(\mathbf{W}) + T\,\big(h\,\mathrm{nnz}(\mathbf{W}) + d^2 m\big)\big), \tag{S5}$$

Algorithm S1 CAPS — Combinatorial Assignment Permutation Solver

Require: Observed pairwise matchings $\{\mathbf{X}_{ij}\}_{i,j=1}^{k}$; sizes $\{m_i\}_{i=1}^{k}$; universe size d.
Ensure: $\mathbf{X} \in \mathbb{U}$, an exact partial permutation per object.

Assemble $\mathbf{W} \leftarrow [\mathbf{X}_{ij}]_{i,j=1}^{k} \in \mathbb{R}^{m \times m}$.
Warm start $\mathbf{X}^{(0)} \in \mathbb{U}$ ▷ any feasible point; we use the projected Stiefel solution
$\mathbf{S} \leftarrow \mathbf{W}\mathbf{X}^{(0)}$ ▷ maintained incrementally; $O(\mathrm{nnz}(\mathbf{W}))$
for $t = 1, 2, \ldots$ **do**
  for $i = 1, \ldots, k$ **do**
    $\mathbf{E}_i \leftarrow \mathbf{S}_{[i]} - \mathbf{W}_{ii}\mathbf{X}_i$ ▷ cross-object scores $\sum_{j \neq i} \mathbf{W}_{ij}\mathbf{X}_j$
    $\mathbf{A}_i \leftarrow 2\,\mathbf{E}_i + \mathrm{diag}(\mathbf{W}_{ii})\mathbb{1}_d^T$ ▷ the affine restriction (6)
    $\mathbf{X}_i^{\text{new}} \leftarrow \arg\max_{\mathbf{X}_i \in \overline{\mathbb{P}}_{m_i d}} \langle \mathbf{A}_i, \mathbf{X}_i \rangle$ ▷ rectangular LAP, Jonker–Volgenant
    if $\langle \mathbf{A}_i, \mathbf{X}_i^{\text{new}} \rangle > \langle \mathbf{A}_i, \mathbf{X}_i \rangle$ **then** ▷ strict-improvement rule
      $\mathbf{S} \leftarrow \mathbf{S} + \mathbf{W}_{[:,i]}(\mathbf{X}_i^{\text{new}} - \mathbf{X}_i)$; $\mathbf{X}_i \leftarrow \mathbf{X}_i^{\text{new}}$
    end if
  end for
  if no block changed in this pass **then**
    break ▷ Proposition 3
  end if
end for
return $\mathbf{X}$ ▷ binary; every $\mathbf{X}_{ij} = \mathbf{X}_i\mathbf{X}_j^T$ cycle-consistent by construction

which, for a fixed number of objects, is again linear in $\mathrm{nnz}(\mathbf{W})$. On the benchmarks, the instances have $m = 53$–124 keypoints and $d = 10$–25 labels, and $\mathbf{W}$ has 150–360 nonzero entries, which corresponds to a density between 2.1% and 6.4%. The resulting run times are compared with those of the baselines in Section 4.

D CAPS IN PSEUDOCODE

Algorithm S1 states the iterative solver of Section 3. It contains no auxiliary variable and no relaxation that loses tightness. By Lemma 3.1 the per-object restriction is already affine, so each step is a linear assignment problem on the cross-object scores, solved exactly. Its guarantees are Propositions 1–4; its cost is analyzed in Appendix C.3.

No line of Algorithm S1 rounds, thresholds or projects. The only non-trivial operation is an exactly solved assignment, whose output is integral by Lemma 3.4.

Table S1: What training adds. Mean test F-score (%); **best** and second best per column.

Variant	Willow	SPair-71k
CAPS (7), iterative, nothing learned	62.6	38.7
CLAPS (ours), unrolled and learned	**63.9**	**40.0**

E THE UNROLLED SOLVER

CLAPS, stated as Algorithm 1 in Subsection 3.3, runs the update of Algorithm S1 for a fixed number of layers and accepts every exact maximizer, and it differs from that update in three further places. The observations are gated by a learned per-edge trust, the three constants of the cost become per-layer parameters in place of the fixed values of (6), and the assignment admits an opt-out. The assignment remains an exact rectangular LAP, augmented for the opt-out, so every iterate remains a partial permutation at training time as well as at inference.

Opt-out. A keypoint may remain unassigned. This is implemented by appending m_i auxiliary columns of value $-\varepsilon$, so that each keypoint is matched either to a label or to an auxiliary column; the augmented problem is still a bipartite assignment problem, so its optimum is integral and is computed exactly. If $\mathbf{A}_i$ is entrywise nonnegative, no keypoint gains from being declined, and the step coincides with the rectangular LAP of Algorithm S1.

Gradients. Training carries gradients through an entropically regularized transport plan, whose cost matrix has one extra row and one extra column for the keypoints left unassigned and the labels left unused, while the forward value is always the exact assignment (straight-through). Training and inference therefore compute the *same* function, and no relaxed iterate is ever produced, so no tightness is lost in training either.

Reduction. With $a_t = 1$, $b_t = c_t = 0$ and $\mathbf{G} \equiv \mathbb{1}\mathbb{1}^T$ the network reproduces the update (7) as CAPS applies it in Appendix F. We verified this numerically on test instances of both datasets, where the maximum entrywise deviation between the CLAPS and CAPS iterates is zero in every check.

Training. The loss acts on the cost matrix $\mathbf{A}_i$, the quantity the assignment step consumes. A margin term requires the true column of each annotated row to exceed every other column and the opt-out threshold, and every column of a distractor row to fall below that threshold, so the loss is aligned with the decisions the exact assignment takes. A binary cross-entropy on $\mathbf{G}$ against the correctness of each observed match completes it.

F THE CONTRIBUTION OF THE LEARNING

This section examines the respective contributions of the combinatorial solver and of the learning to the accuracy of CLAPS. Table S1 compares CAPS, which applies step (7) and contains no learned quantity, with CLAPS, which uses the same form of update and the same exact assignment but learns the per-layer constants and the per-edge trust function. As a purely combinatorial solver, CAPS is already competitive with the strongest baselines of Tables 1 and 2, and training adds a further 1.3 points of mean F-score on both benchmarks, which places CLAPS ahead of every baseline. The two contributions therefore build on one another. The combinatorial solver provides a strong result by itself, and training, which starts from it because CLAPS contains CAPS as a special case, improves upon it.

F.1 SENSITIVITY TO THE INITIALIZATION

The guarantees of Propositions 1–4 hold for CAPS from any feasible starting point $\mathbf{X}^{(0)} \in \mathbb{U}$. Since problem (2) is non-convex, the initialization also matters. To measure its effect, we repeat the evaluation from three alternatives to the Stiefel start, under which the accuracy reported in Section 4 was obtained. The first alternative takes the d leading eigenvectors of $\mathbf{W}$ and projects them in the same way as the Stiefel solution, so it is also informed by the observed matchings. The other two use no such information. One assigns to object i the labels $1, \ldots, \min(m_i, d)$ in order, and the other

Table S2: Sensitivity to the initialization. Mean F-score and precision (%) on the test splits; **best** and second best per column.

	CAPS		CLAPS	
Start	**F**	**Precision**	**F**	**Precision**
Willow Object				
Stiefel (reported)	**62.6**	**62.6**	**63.9**	68.5
Spectral	61.2	61.2	62.2	68.4
Identity	58.1	58.1	62.8	**68.9**
Random	60.8	60.8	62.6	68.5
SPair-71k				
Stiefel (reported)	**38.7**	**35.7**	**40.0**	**43.2**
Spectral	38.5	35.5	39.3	43.1
Identity	38.0	35.1	39.0	43.1
Random	38.2	35.2	38.8	42.8

projects a seeded random score matrix in the same way. Table S2 reports all four starts, the random one averaged over three seeds.

The Stiefel start leads every F-score column of Table S2 and both precision columns of CAPS. On Willow Object, its F-score exceeds that of the nearest alternative by 1.4 points for CAPS and 1.1 for CLAPS. In the precision of CLAPS, the four starts are on a par, all within 0.5 points of one another, at 68.4–68.9% on Willow Object and 42.8–43.2% on SPair-71k. CLAPS is also markedly less sensitive to the initialization than CAPS, since replacing the Stiefel start by the identity changes its F-score on Willow Object by only 1.1 points, against 4.5 for CAPS. Even from the identity, for which no warm start is computed, CLAPS retains its precision lead over the strongest baseline, 6.2 points on Willow Object and 7.0 on SPair-71k, and remains on a par with it in F-score on both benchmarks.

Whereas the identity and the random start use no information from $\mathbf{W}$, the two informed starts lead because they already encode the observed matchings. Neither of them is a solution in itself. Each relaxes the trace maximization to a continuous domain, losing tightness, and becomes a partial permutation only through the projection. It is the refinement by CAPS and CLAPS that turns such a start into an accurate matching, as the gain over the Stiefel solution reported in Section 4 shows. Of the two relaxations, only the Stiefel one is posed over partial permutations, whereas the eigenvector one assumes full bijections and cannot represent the unmatched keypoints present in these instances. The F-scores of CAPS follow this ordering on both benchmarks, with the two informed starts ahead of the two uninformed ones and the Stiefel start leading the spectral one, which is why we adopt it for the reported experiments.

G Per-category results

Tables S3 and S4–S6 give the per-category breakdown of Tables 1 and 2 in the main paper, under the protocol of Subsection 4.1. As Subsection 4.2 requires of every method reported here, each is cycle-consistent in every category, with no contradiction anywhere.

In every class of both benchmarks, CLAPS attains the highest precision, ahead of the strongest baseline by 4.0 to 11.5 points. Its F-score is the best on every Willow Object class and on *bicycle*, *boat*, *bottle*, *cat*, *chair*, *train* and *tvmonitor*. On *aeroplane*, *car*, *horse* and *motorbike*, its F-score is on a par with the best, and on *bird*, *dog* and *sheep* it remains competitive, while its precision on these three classes exceeds that of every baseline by 4.0 to 5.9 points.

Table S3: Per-category results on **Willow Object**. **Best** and second best per category and metric; our row is shaded.

Cat.	Method	F-score (%) ↑	Precision (%) ↑	Cyclic (%) ↑	Time (s) ↓
Car	Spectral (2013) [25]	65.7	65.7	100.0	**0.001**
	NmfSync (2019) [5]	64.9	64.9	100.0	0.023
	Stiefel (2021) [7]	65.4	65.4	100.0	0.012
	FCC + exact rounding (2021) [32]	68.0	68.0	100.0	0.007
	MatchFAME (2022) [21]	46.4	46.4	100.0	0.145
	SEQ-C + SEQ-LS (2024) [19]	64.7	59.3	100.0	0.088
	PAR-C + PAR-LS (2024) [19]	63.6	58.3	100.0	0.059
	GREEDA (2025) [20]	64.9	59.5	100.0	0.066
	CLAPS (ours)	**69.1**	**73.0**	100.0	0.021
Duck	Spectral (2013) [25]	44.6	44.6	100.0	**0.001**
	NmfSync (2019) [5]	44.4	44.4	100.0	0.021
	Stiefel (2021) [7]	45.1	45.1	100.0	0.011
	FCC + exact rounding (2021) [32]	45.8	45.8	100.0	0.006
	MatchFAME (2022) [21]	35.0	35.0	100.0	0.022
	SEQ-C + SEQ-LS (2024) [19]	45.3	41.5	100.0	0.067
	PAR-C + PAR-LS (2024) [19]	45.8	42.0	100.0	0.059
	GREEDA (2025) [20]	44.5	40.8	100.0	0.065
	CLAPS (ours)	**46.5**	**53.0**	100.0	0.019
Motorbike	Spectral (2013) [25]	61.4	61.4	100.0	**0.001**
	NmfSync (2019) [5]	63.6	63.6	100.0	0.022
	Stiefel (2021) [7]	60.7	60.7	100.0	0.011
	FCC + exact rounding (2021) [32]	61.9	61.9	100.0	0.006
	MatchFAME (2022) [21]	47.9	47.9	100.0	0.023
	SEQ-C + SEQ-LS (2024) [19]	59.7	54.7	100.0	0.063
	PAR-C + PAR-LS (2024) [19]	60.4	55.4	100.0	0.056
	GREEDA (2025) [20]	60.0	55.0	100.0	0.062
	CLAPS (ours)	**64.0**	**67.7**	100.0	0.019
Winebottle	Spectral (2013) [25]	74.6	74.6	100.0	**0.001**
	NmfSync (2019) [5]	75.4	75.4	100.0	0.021
	Stiefel (2021) [7]	73.4	73.4	100.0	0.010
	FCC + exact rounding (2021) [32]	74.9	74.9	100.0	0.006
	MatchFAME (2022) [21]	53.1	53.1	100.0	0.022
	SEQ-C + SEQ-LS (2024) [19]	67.9	62.2	100.0	0.060
	PAR-C + PAR-LS (2024) [19]	67.3	61.7	100.0	0.054
	GREEDA (2025) [20]	69.0	63.3	100.0	0.062
	CLAPS (ours)	**76.0**	**80.3**	100.0	0.018

Table S4: Per-category results on **SPair-71k**, categories *aeroplane* to *bottle*. **Best** and second best per category and metric; our row is shaded.

Cat.	Method	F-score (%) ↑	Precision (%) ↑	Cyclic (%) ↑	Time (s) ↓
aeroplane	Spectral (2013) [25]	27.1	23.1	100.0	**0.006**
	NmfSync (2019) [5]	26.7	22.8	100.0	0.160
	Stiefel (2021) [7]	**28.2**	24.1	100.0	0.085
	FCC + exact rounding (2021) [32]	28.1	24.0	100.0	0.023
	MatchFAME (2022) [21]	24.7	21.1	100.0	0.089
	SEQ-C + SEQ-LS (2024) [19]	27.3	23.3	100.0	0.093
	PAR-C + PAR-LS (2024) [19]	27.9	23.7	100.0	0.102
	GREEDA (2025) [20]	27.5	23.5	100.0	0.125
	CLAPS (ours)	27.9	**28.9**	100.0	0.097
bicycle	Spectral (2013) [25]	44.1	40.5	100.0	**0.001**
	NmfSync (2019) [5]	43.3	39.8	100.0	0.025
	Stiefel (2021) [7]	42.4	38.9	100.0	0.013
	FCC + exact rounding (2021) [32]	44.3	40.7	100.0	0.006
	MatchFAME (2022) [21]	37.6	34.5	100.0	0.022
	SEQ-C + SEQ-LS (2024) [19]	44.2	40.0	100.0	0.061
	PAR-C + PAR-LS (2024) [19]	44.0	39.8	100.0	0.058
	GREEDA (2025) [20]	45.1	40.8	100.0	0.061
	CLAPS (ours)	**45.9**	**47.6**	100.0	0.021
bird	Spectral (2013) [25]	28.0	24.9	100.0	**0.002**
	NmfSync (2019) [5]	27.7	24.6	100.0	0.035
	Stiefel (2021) [7]	28.1	25.0	100.0	0.018
	FCC + exact rounding (2021) [32]	28.6	25.5	100.0	0.007
	MatchFAME (2022) [21]	25.6	22.7	100.0	0.024
	SEQ-C + SEQ-LS (2024) [19]	29.4	25.8	100.0	0.064
	PAR-C + PAR-LS (2024) [19]	29.7	26.0	100.0	0.062
	GREEDA (2025) [20]	**30.2**	26.6	100.0	0.070
	CLAPS (ours)	28.4	**32.5**	100.0	0.026
boat	Spectral (2013) [25]	37.2	33.7	100.0	**0.001**
	NmfSync (2019) [5]	38.5	34.9	100.0	0.028
	Stiefel (2021) [7]	36.5	33.1	100.0	0.018
	FCC + exact rounding (2021) [32]	39.2	35.6	100.0	0.006
	MatchFAME (2022) [21]	28.8	26.0	100.0	0.024
	SEQ-C + SEQ-LS (2024) [19]	38.8	35.1	100.0	0.068
	PAR-C + PAR-LS (2024) [19]	38.7	35.0	100.0	0.061
	GREEDA (2025) [20]	39.1	35.3	100.0	0.075
	CLAPS (ours)	**40.7**	**45.4**	100.0	0.026
bottle	Spectral (2013) [25]	60.0	60.0	100.0	**0.001**
	NmfSync (2019) [5]	63.5	63.5	100.0	0.022
	Stiefel (2021) [7]	64.0	64.0	100.0	0.011
	FCC + exact rounding (2021) [32]	65.3	65.3	100.0	0.006
	MatchFAME (2022) [21]	45.2	45.2	100.0	0.023
	SEQ-C + SEQ-LS (2024) [19]	60.7	55.7	100.0	0.059
	PAR-C + PAR-LS (2024) [19]	60.3	55.3	100.0	0.055
	GREEDA (2025) [20]	59.3	54.4	100.0	0.058
	CLAPS (ours)	**65.7**	**72.1**	100.0	0.018

Table S5: Per-category results on **SPair-71k**, categories *car* to *horse*. **Best** and second best per category and metric; our row is shaded.

Cat.	Method	F-score (%) ↑	Precision (%) ↑	Cyclic (%) ↑	Time (s) ↓
car	Spectral (2013) [25]	34.9	31.7	100.0	**0.004**
	NmfSync (2019) [5]	36.1	32.8	100.0	0.091
	Stiefel (2021) [7]	36.4	33.1	100.0	0.035
	FCC + exact rounding (2021) [32]	36.3	33.0	100.0	0.013
	MatchFAME (2022) [21]	29.5	26.7	100.0	0.027
	SEQ-C + SEQ-LS (2024) [19]	**37.7**	33.2	100.0	0.077
	PAR-C + PAR-LS (2024) [19]	37.6	33.1	100.0	0.077
	GREEDA (2025) [20]	37.2	32.8	100.0	0.102
	CLAPS (ours)	37.6	**39.9**	100.0	0.047
cat	Spectral (2013) [25]	40.6	37.8	100.0	**0.003**
	NmfSync (2019) [5]	39.1	36.4	100.0	0.057
	Stiefel (2021) [7]	40.1	37.3	100.0	0.032
	FCC + exact rounding (2021) [32]	39.5	36.8	100.0	0.011
	MatchFAME (2022) [21]	29.6	27.6	100.0	0.024
	SEQ-C + SEQ-LS (2024) [19]	38.4	34.7	100.0	0.073
	PAR-C + PAR-LS (2024) [19]	38.8	35.0	100.0	0.071
	GREEDA (2025) [20]	39.6	35.7	100.0	0.096
	CLAPS (ours)	**43.2**	**49.3**	100.0	0.043
chair	Spectral (2013) [25]	32.6	29.0	100.0	**0.002**
	NmfSync (2019) [5]	33.1	29.5	100.0	0.047
	Stiefel (2021) [7]	31.8	28.3	100.0	0.025
	FCC + exact rounding (2021) [32]	33.2	29.6	100.0	0.009
	MatchFAME (2022) [21]	28.1	25.0	100.0	0.025
	SEQ-C + SEQ-LS (2024) [19]	34.3	30.1	100.0	0.077
	PAR-C + PAR-LS (2024) [19]	33.5	29.3	100.0	0.077
	GREEDA (2025) [20]	34.3	30.0	100.0	0.088
	CLAPS (ours)	**34.9**	**36.5**	100.0	0.036
dog	Spectral (2013) [25]	25.7	23.0	100.0	**0.003**
	NmfSync (2019) [5]	25.5	22.9	100.0	0.063
	Stiefel (2021) [7]	25.3	22.7	100.0	0.033
	FCC + exact rounding (2021) [32]	25.1	22.6	100.0	0.013
	MatchFAME (2022) [21]	24.9	22.3	100.0	0.029
	SEQ-C + SEQ-LS (2024) [19]	25.8	22.8	100.0	0.091
	PAR-C + PAR-LS (2024) [19]	**26.5**	23.4	100.0	0.082
	GREEDA (2025) [20]	26.3	23.3	100.0	0.097
	CLAPS (ours)	24.4	**28.0**	100.0	0.045
horse	Spectral (2013) [25]	26.7	23.2	100.0	**0.004**
	NmfSync (2019) [5]	27.0	23.4	100.0	0.097
	Stiefel (2021) [7]	27.8	24.1	100.0	0.037
	FCC + exact rounding (2021) [32]	27.6	23.9	100.0	0.013
	MatchFAME (2022) [21]	23.7	20.5	100.0	0.026
	SEQ-C + SEQ-LS (2024) [19]	27.8	24.0	100.0	0.082
	PAR-C + PAR-LS (2024) [19]	27.6	23.8	100.0	0.083
	GREEDA (2025) [20]	**28.1**	24.2	100.0	0.108
	CLAPS (ours)	27.7	**31.4**	100.0	0.049

Table S6: Per-category results on **SPair-71k**, categories *motorbike* to *tvmonitor*. **Best** and second best per category and metric; our row is shaded.

Cat.	Method	F-score (%) ↑	Precision (%) ↑	Cyclic (%) ↑	Time (s) ↓
motorbike	Spectral (2013) [25]	33.0	28.4	100.0	**0.001**
	NmfSync (2019) [5]	33.6	28.9	100.0	0.022
	Stiefel (2021) [7]	32.7	28.1	100.0	0.013
	FCC + exact rounding (2021) [32]	33.5	28.8	100.0	0.005
	MatchFAME (2022) [21]	30.0	25.8	100.0	0.025
	SEQ-C + SEQ-LS (2024) [19]	**34.4**	29.6	100.0	0.065
	PAR-C + PAR-LS (2024) [19]	32.4	28.0	100.0	0.056
	GREEDA (2025) [20]	**34.4**	29.7	100.0	0.067
	CLAPS (ours)	34.2	**34.4**	100.0	0.021
sheep	Spectral (2013) [25]	26.8	22.1	100.0	**0.003**
	NmfSync (2019) [5]	26.6	21.9	100.0	0.075
	Stiefel (2021) [7]	**27.7**	23.0	100.0	0.024
	FCC + exact rounding (2021) [32]	27.5	22.7	100.0	0.010
	MatchFAME (2022) [21]	25.6	21.0	100.0	0.022
	SEQ-C + SEQ-LS (2024) [19]	26.5	21.8	100.0	0.067
	PAR-C + PAR-LS (2024) [19]	26.2	21.5	100.0	0.071
	GREEDA (2025) [20]	26.3	21.6	100.0	0.089
	CLAPS (ours)	26.2	**27.0**	100.0	0.036
train	Spectral (2013) [25]	39.4	37.6	100.0	**0.004**
	NmfSync (2019) [5]	37.3	35.7	100.0	0.084
	Stiefel (2021) [7]	38.8	37.1	100.0	0.031
	FCC + exact rounding (2021) [32]	39.5	37.7	100.0	0.013
	MatchFAME (2022) [21]	28.9	27.4	100.0	0.022
	SEQ-C + SEQ-LS (2024) [19]	38.6	34.7	100.0	0.076
	PAR-C + PAR-LS (2024) [19]	39.5	35.5	100.0	0.075
	GREEDA (2025) [20]	38.7	34.8	100.0	0.102
	CLAPS (ours)	**41.4**	**46.8**	100.0	0.041
tvmonitor	Spectral (2013) [25]	77.7	77.7	100.0	**0.003**
	NmfSync (2019) [5]	80.6	80.6	100.0	0.070
	Stiefel (2021) [7]	79.2	79.2	100.0	0.018
	FCC + exact rounding (2021) [32]	79.1	79.1	100.0	0.013
	MatchFAME (2022) [21]	51.3	51.3	100.0	0.023
	SEQ-C + SEQ-LS (2024) [19]	72.4	66.8	100.0	0.074
	PAR-C + PAR-LS (2024) [19]	72.6	66.9	100.0	0.075
	GREEDA (2025) [20]	74.0	68.2	100.0	0.090
	CLAPS (ours)	**82.0**	**85.2**	100.0	0.029

H PER-CATEGORY QUALITATIVE RESULTS

Figure 2 of the main paper shows one category, and the figures below show further ones. For each, the panel gives the annotation ground truth and the correspondences recovered by every method on one image pair. Green lines agree with the annotation and red lines are errors; filled yellow dots are annotated keypoints and hollow ones are distractors, so a line terminating on a hollow dot is visibly a match to a keypoint that has no correct partner anywhere.

The pair displayed for each category is the one on which our margin over the *strongest* baseline shown in that figure is largest. The panels exhibit, in every category, the behavior discussed for Figure 2. In each figure, our panel contains fewer red lines than that of any baseline. The local-search solvers SEQ-C + SEQ-LS and PAR-C + PAR-LS [19] and GREEDA [20] assign every keypoint, so lines also leave the distractors in their panels, and each such line is an error. Spectral [25], NmfSync [5], Stiefel [7], FCC [32] and MatchFAME [21] reach their matching only through a final rounding, projection or thresholding, which the criterion they optimize does not take into account. Among their panels, those of MatchFAME on *Car* (Figure S1) and of NmfSync on *cat* (Figure S8) are dominated by red lines. CLAPS differs from all of them in two respects. Its iterates never leave the set of partial permutations, since every assignment step is solved exactly, and each observed match enters its cost weighted by its learned trust, so an observation with little cycle support has less influence on the result. The learned price on assignment, moreover, lets it decline a distractor and leave it unmatched.

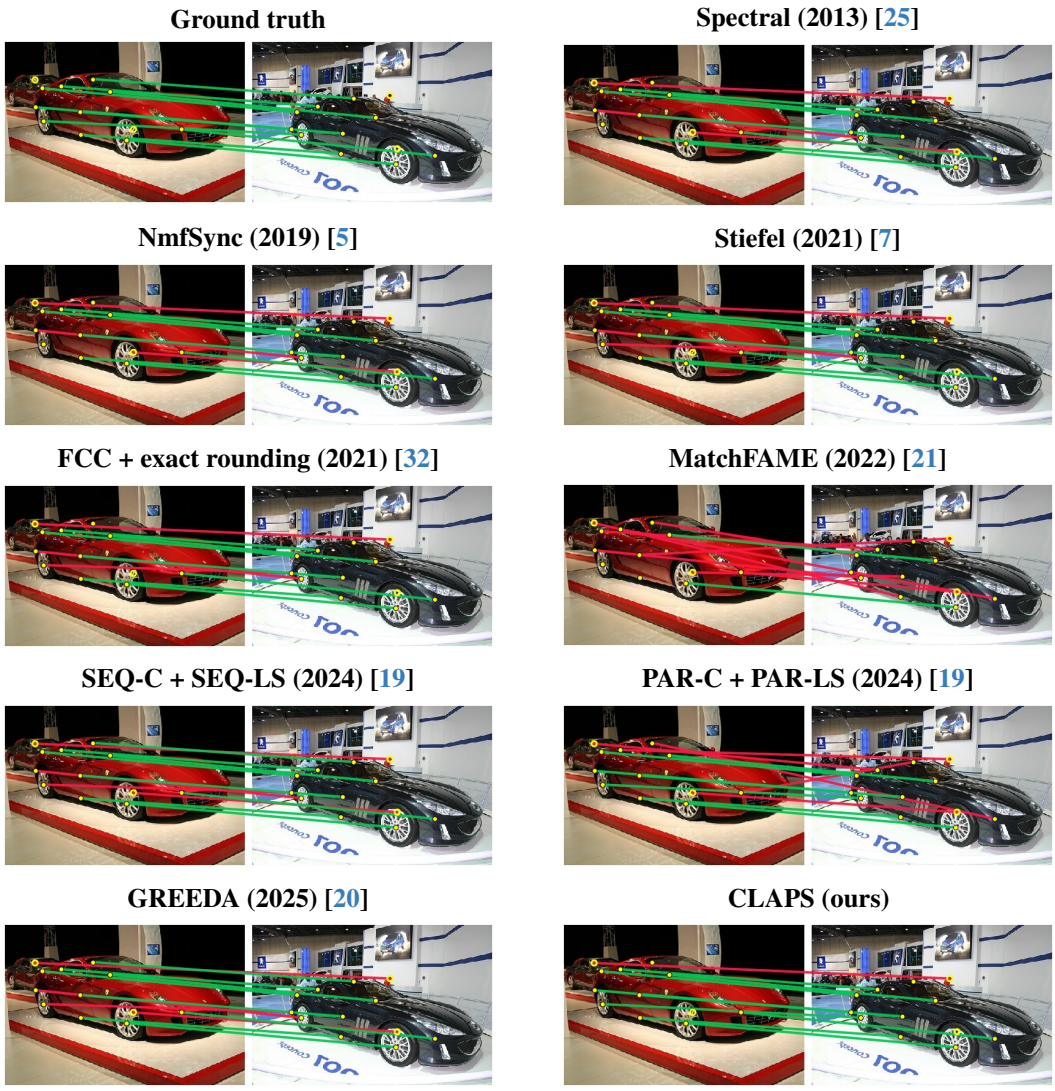

Figure S1: Correspondences on an image pair of the Willow Object category *Car*. Each panel shows the correspondences recovered by one method. Green lines agree with the annotation and red lines are wrong; filled yellow dots are annotated keypoints and hollow dots are distractors, which carry no universe label.

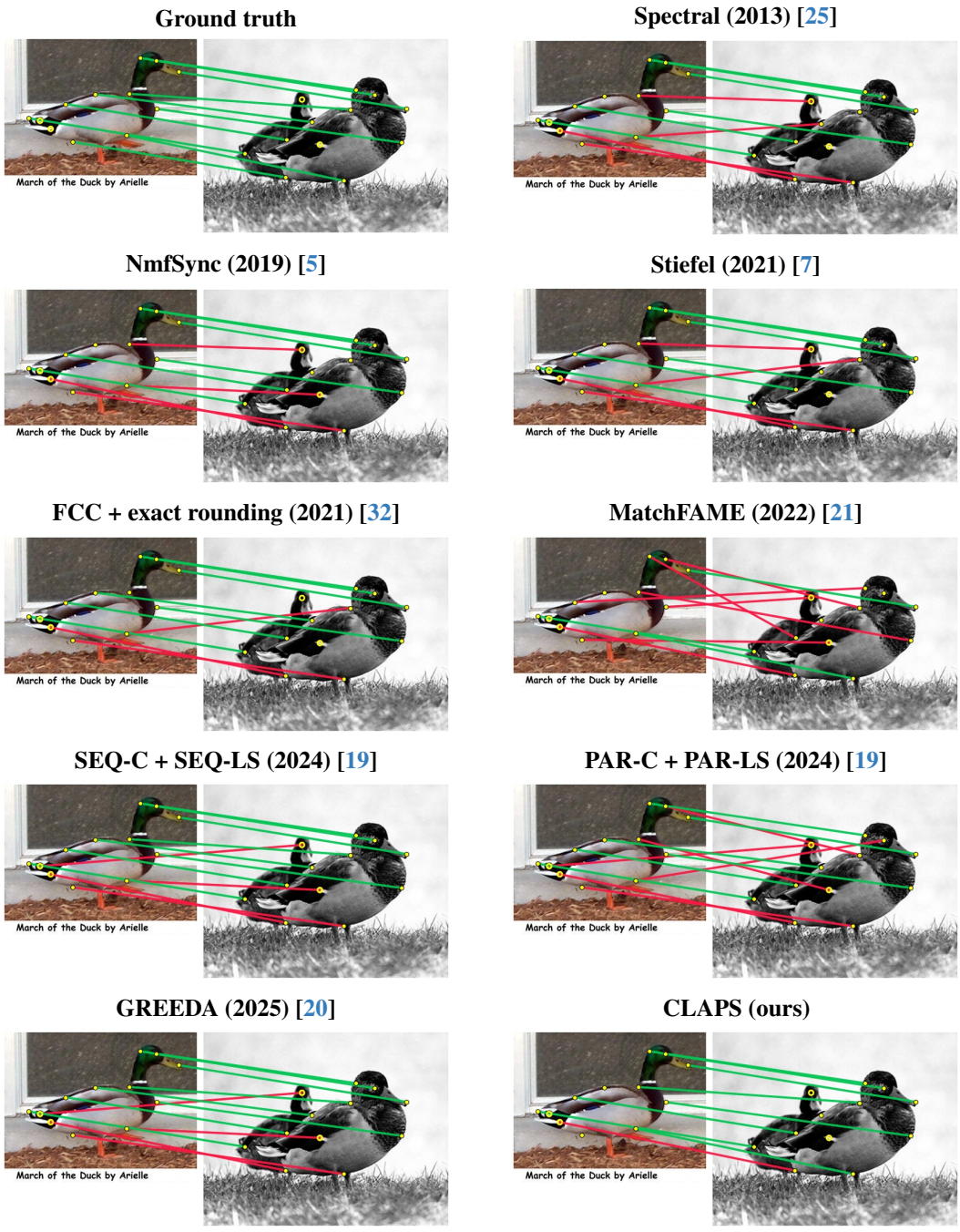

Figure S2: Correspondences on an image pair of the Willow Object category *Duck*. Each panel shows the correspondences recovered by one method. Green lines agree with the annotation and red lines are wrong; filled yellow dots are annotated keypoints and hollow dots are distractors, which carry no universe label.

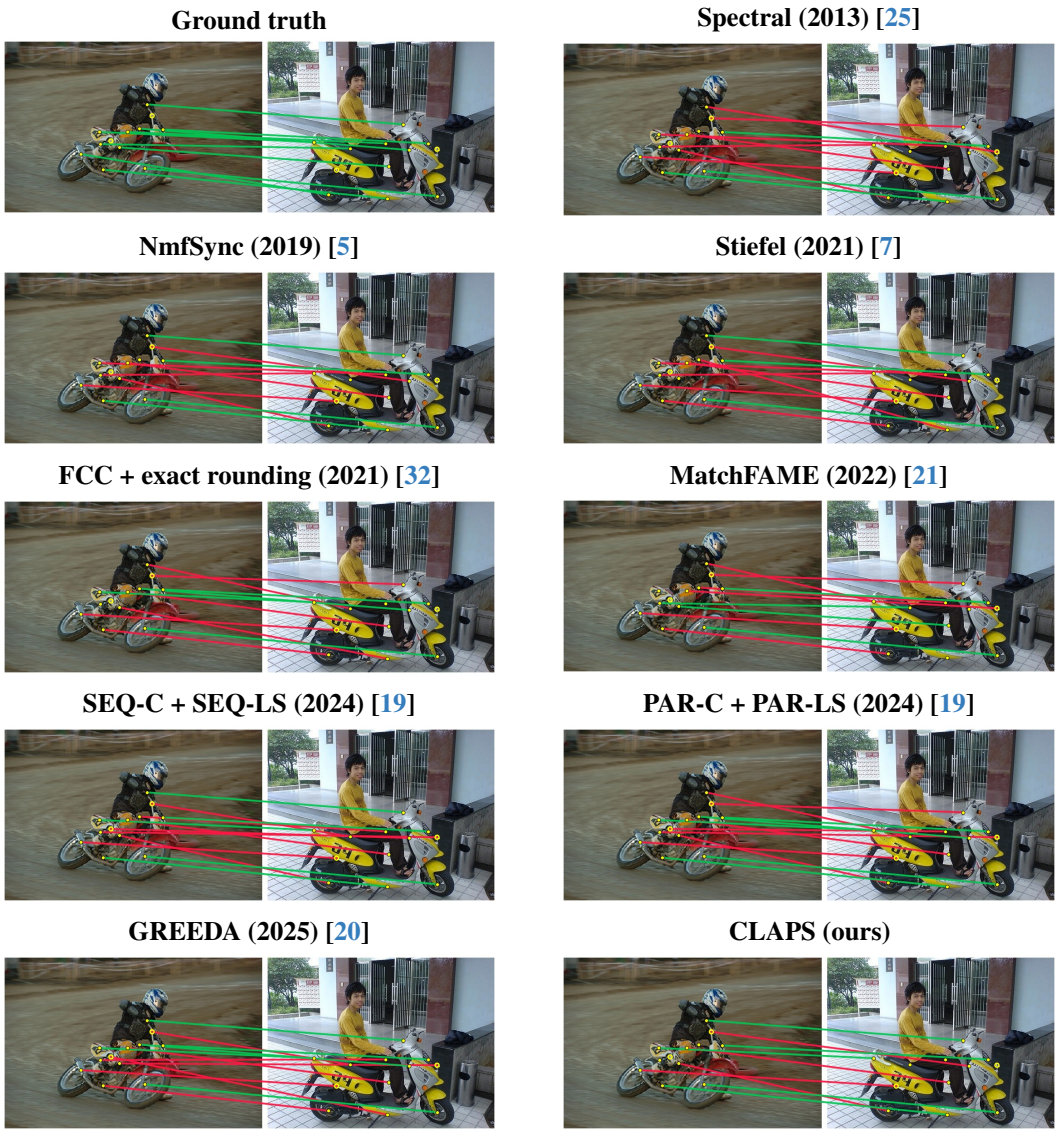

Figure S3: Correspondences on an image pair of the Willow Object category *Motorbike*. Each panel shows the correspondences recovered by one method. Green lines agree with the annotation and red lines are wrong; filled yellow dots are annotated keypoints and hollow dots are distractors, which carry no universe label.

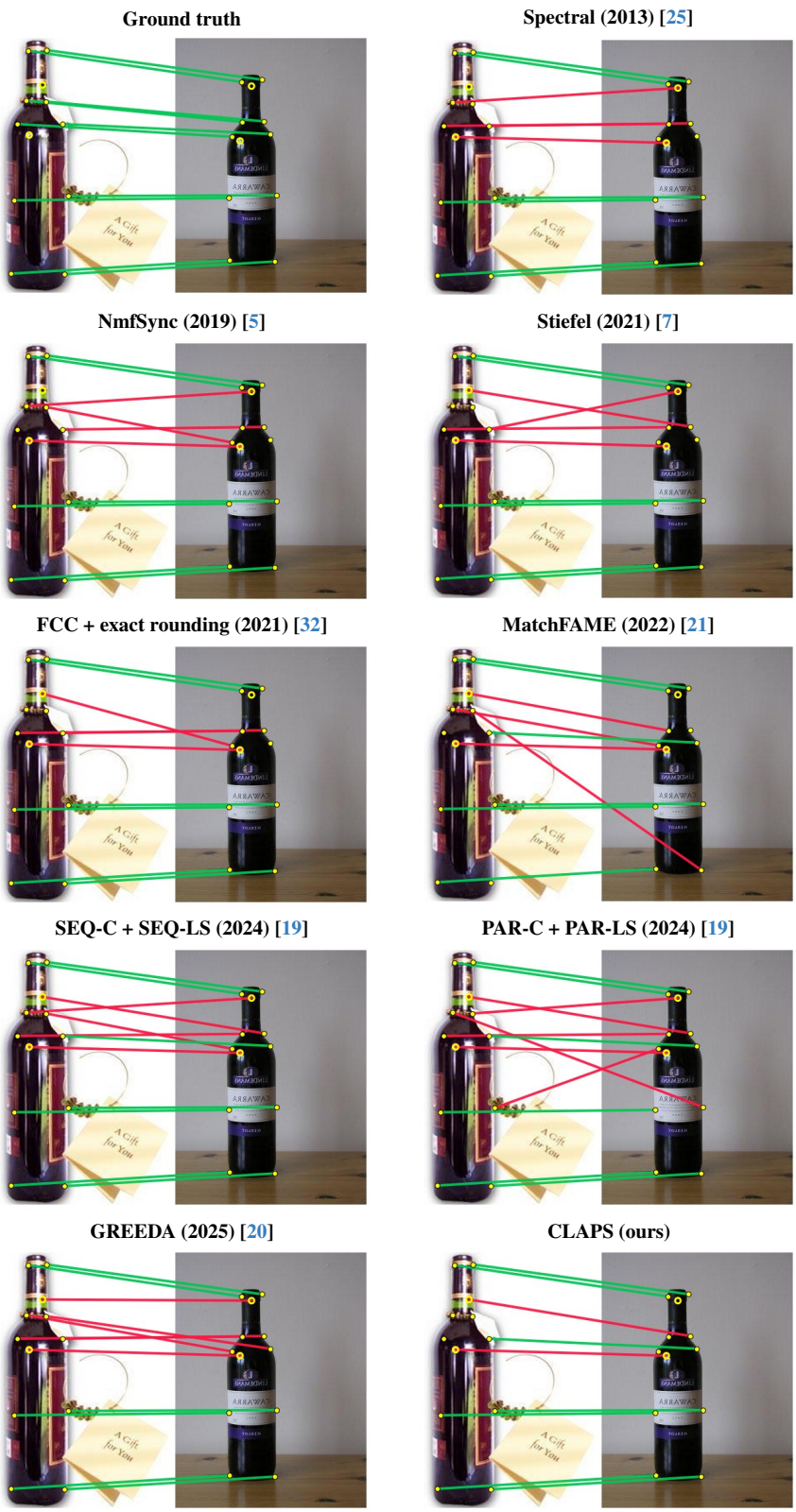

Figure S4: Correspondences on an image pair of the Willow Object category *Winebottle*. Each panel shows the correspondences recovered by one method. Green lines agree with the annotation and red lines are wrong; filled yellow dots are annotated keypoints and hollow dots are distractors, which carry no universe label.

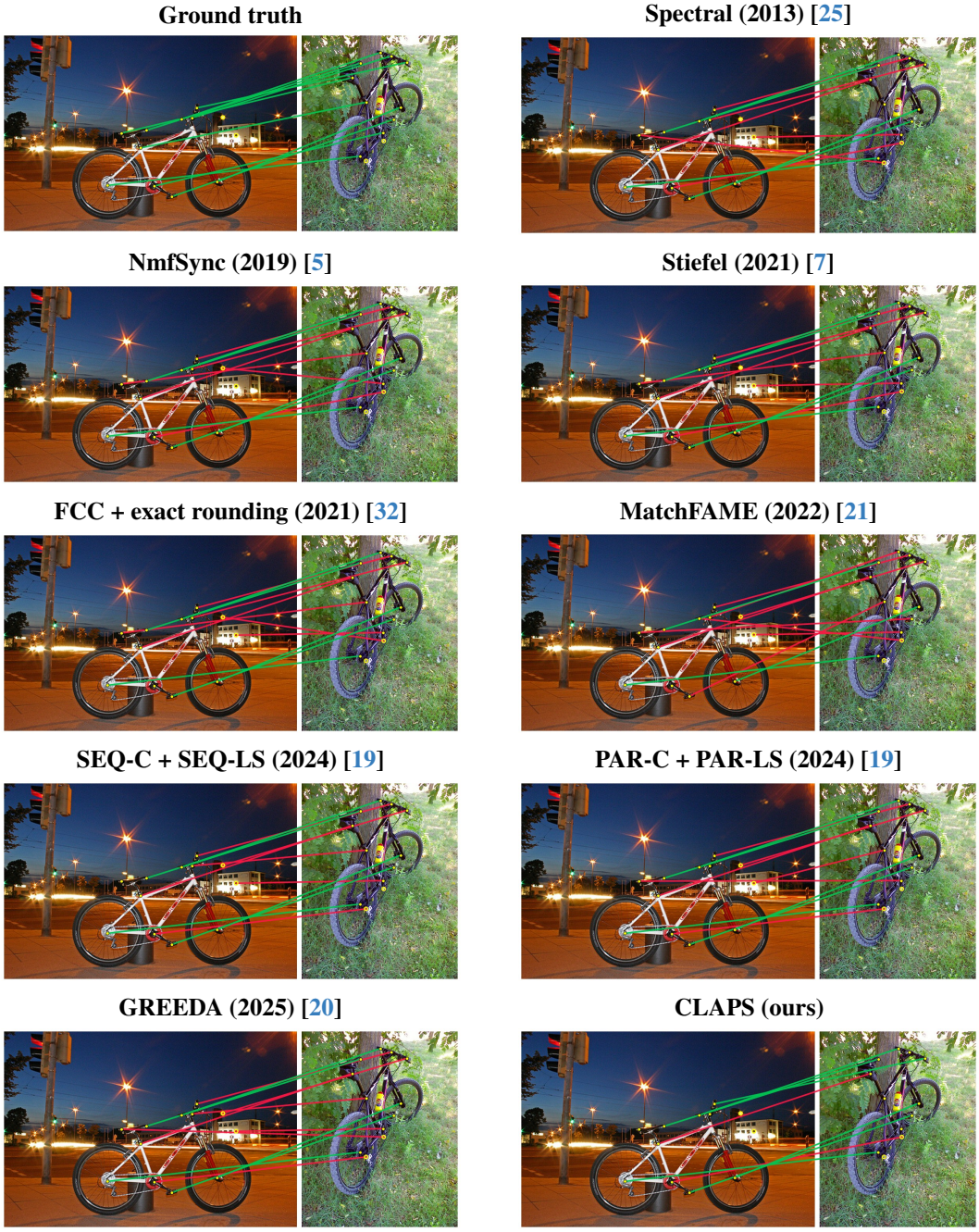

Figure S5: Correspondences on an image pair of the SPair-71k category *bicycle*. Each panel shows the correspondences recovered by one method. Green lines agree with the annotation and red lines are wrong; filled yellow dots are annotated keypoints and hollow dots are distractors, which carry no universe label.

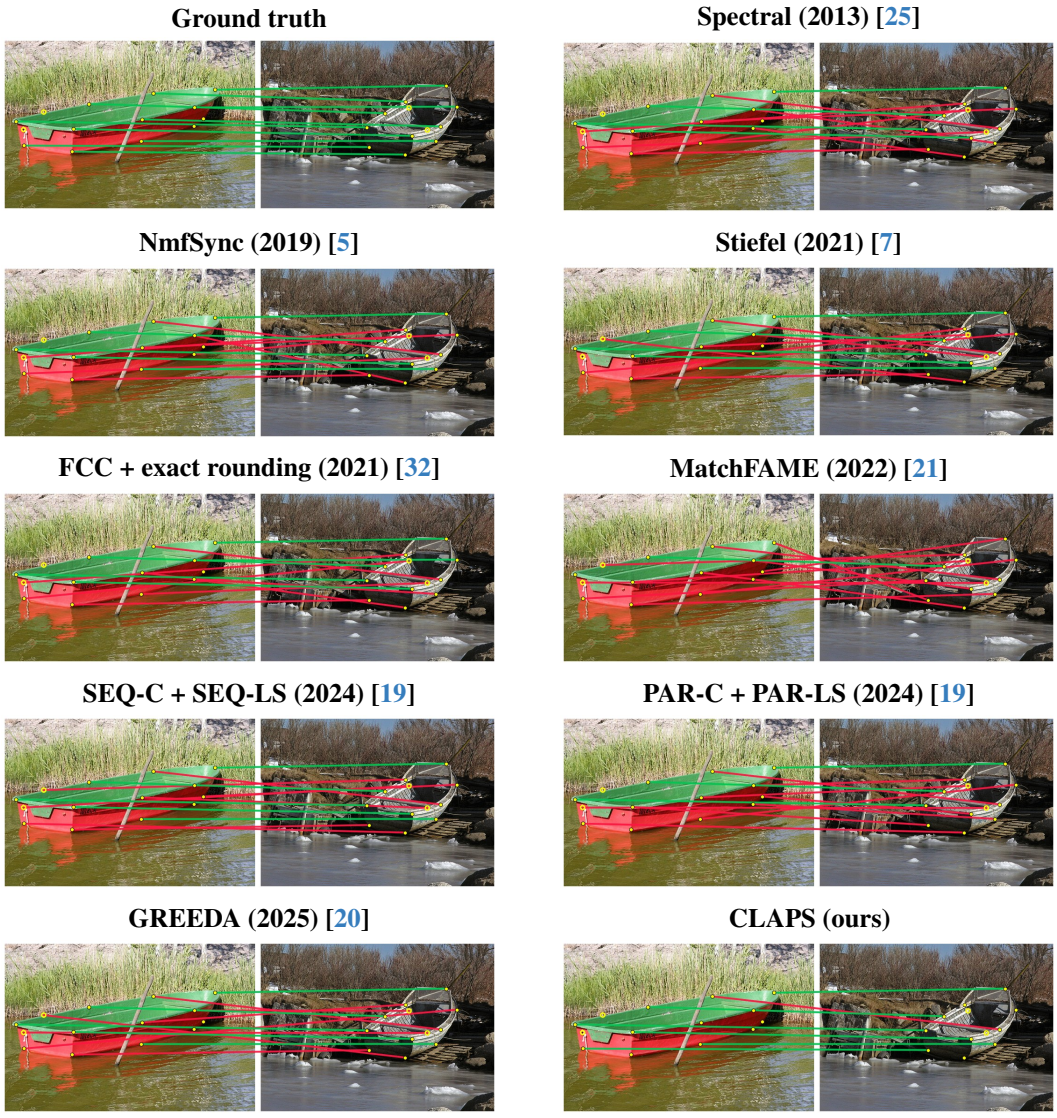

Figure S6: Correspondences on an image pair of the SPair-71k category *boat*. Each panel shows the correspondences recovered by one method. Green lines agree with the annotation and red lines are wrong; filled yellow dots are annotated keypoints and hollow dots are distractors, which carry no universe label.

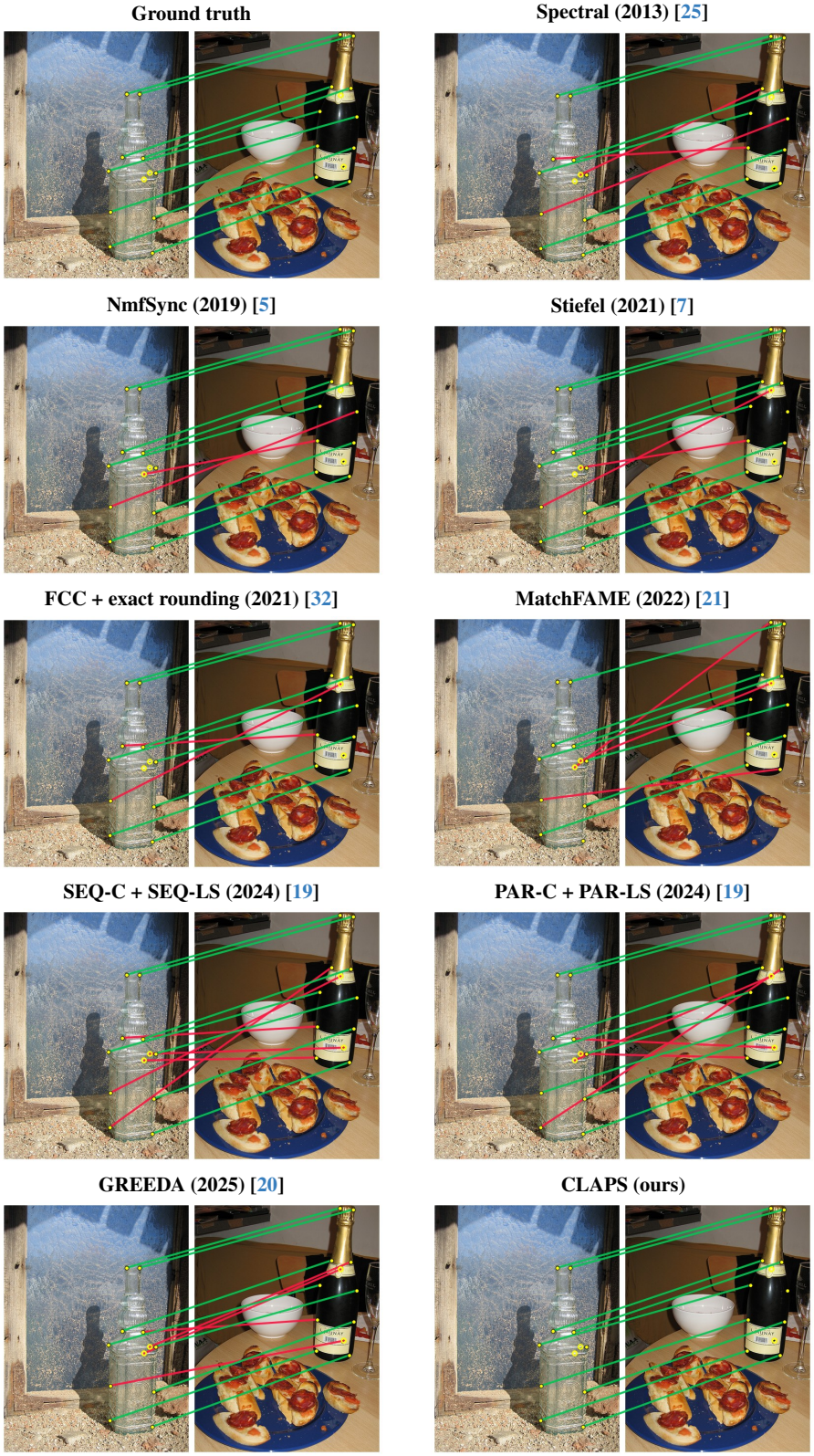

Figure S7: Correspondences on an image pair of the SPair-71k category *bottle*. Each panel shows the correspondences recovered by one method. Green lines agree with the annotation and red lines are wrong; filled yellow dots are annotated keypoints and hollow dots are distractors, which carry no universe label.

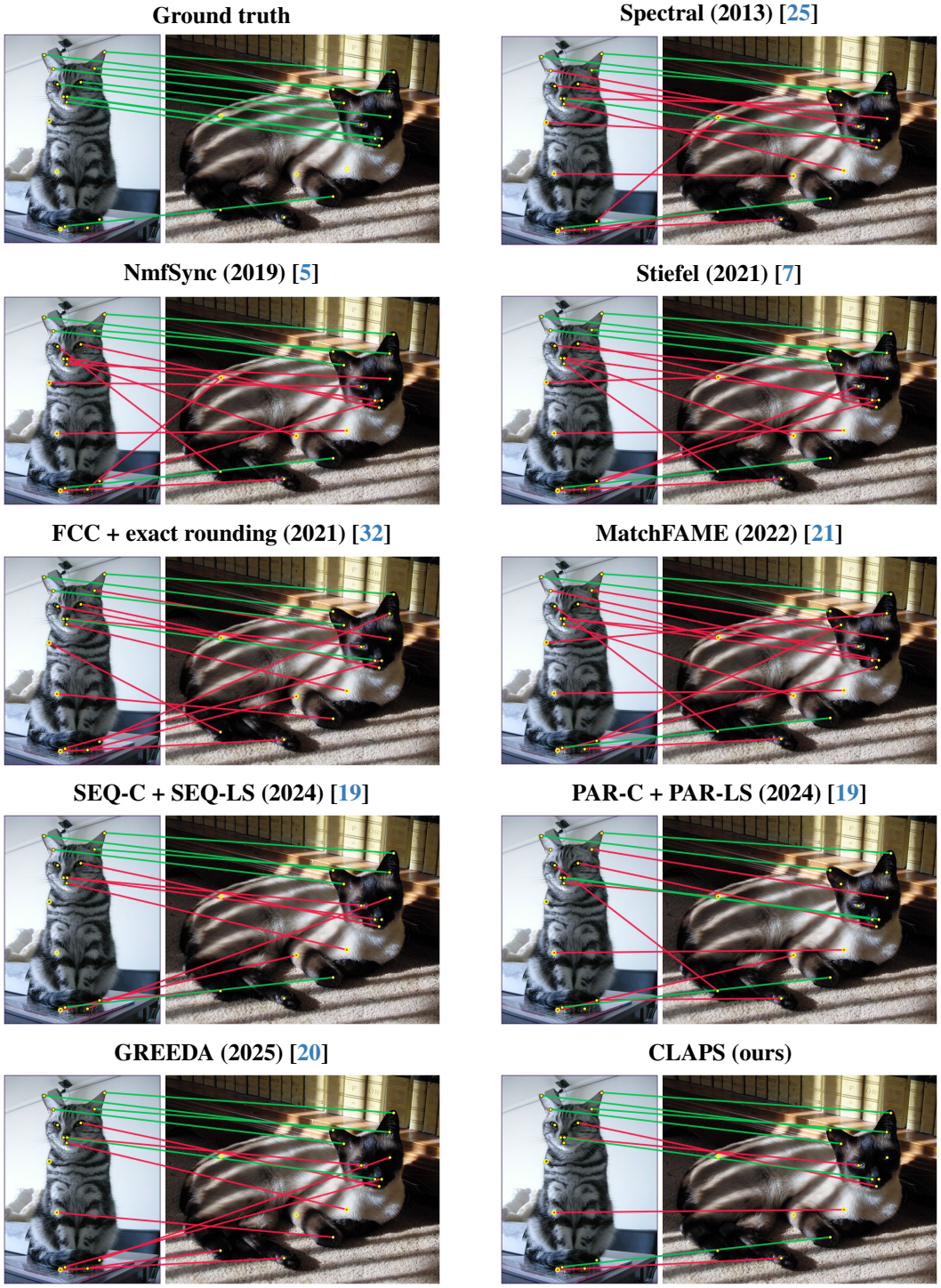

Figure S8: Correspondences on an image pair of the SPair-71k category *cat*. Each panel shows the correspondences recovered by one method. Green lines agree with the annotation and red lines are wrong; filled yellow dots are annotated keypoints and hollow dots are distractors, which carry no universe label.



Figure S9: Correspondences on an image pair of the SPair-71k category *chair*. Each panel shows the correspondences recovered by one method. Green lines agree with the annotation and red lines are wrong; filled yellow dots are annotated keypoints and hollow dots are distractors, which carry no universe label.

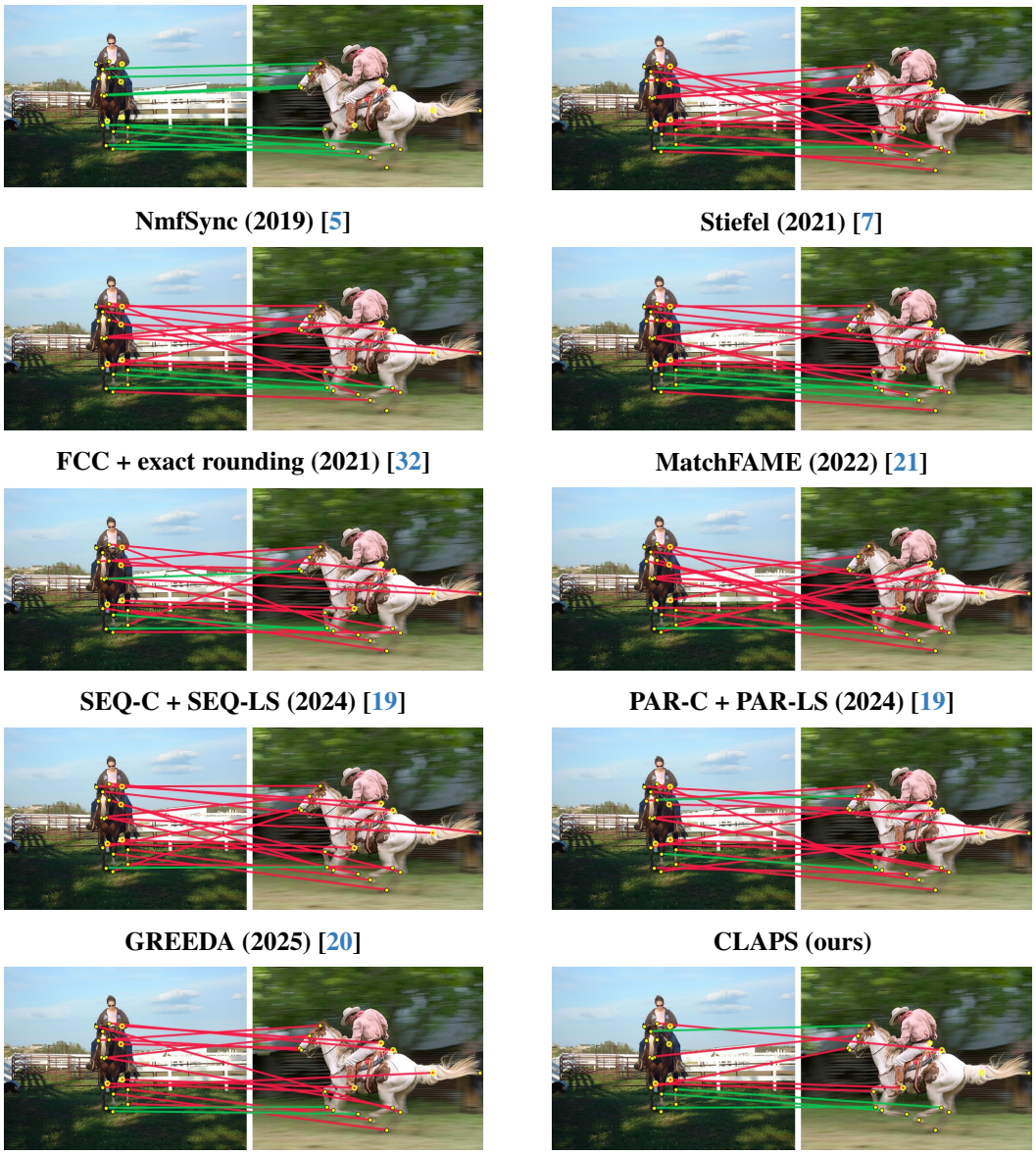

Figure S10: Correspondences on an image pair of the SPair-71k category *horse*. Each panel shows the correspondences recovered by one method. Green lines agree with the annotation and red lines are wrong; filled yellow dots are annotated keypoints and hollow dots are distractors, which carry no universe label.

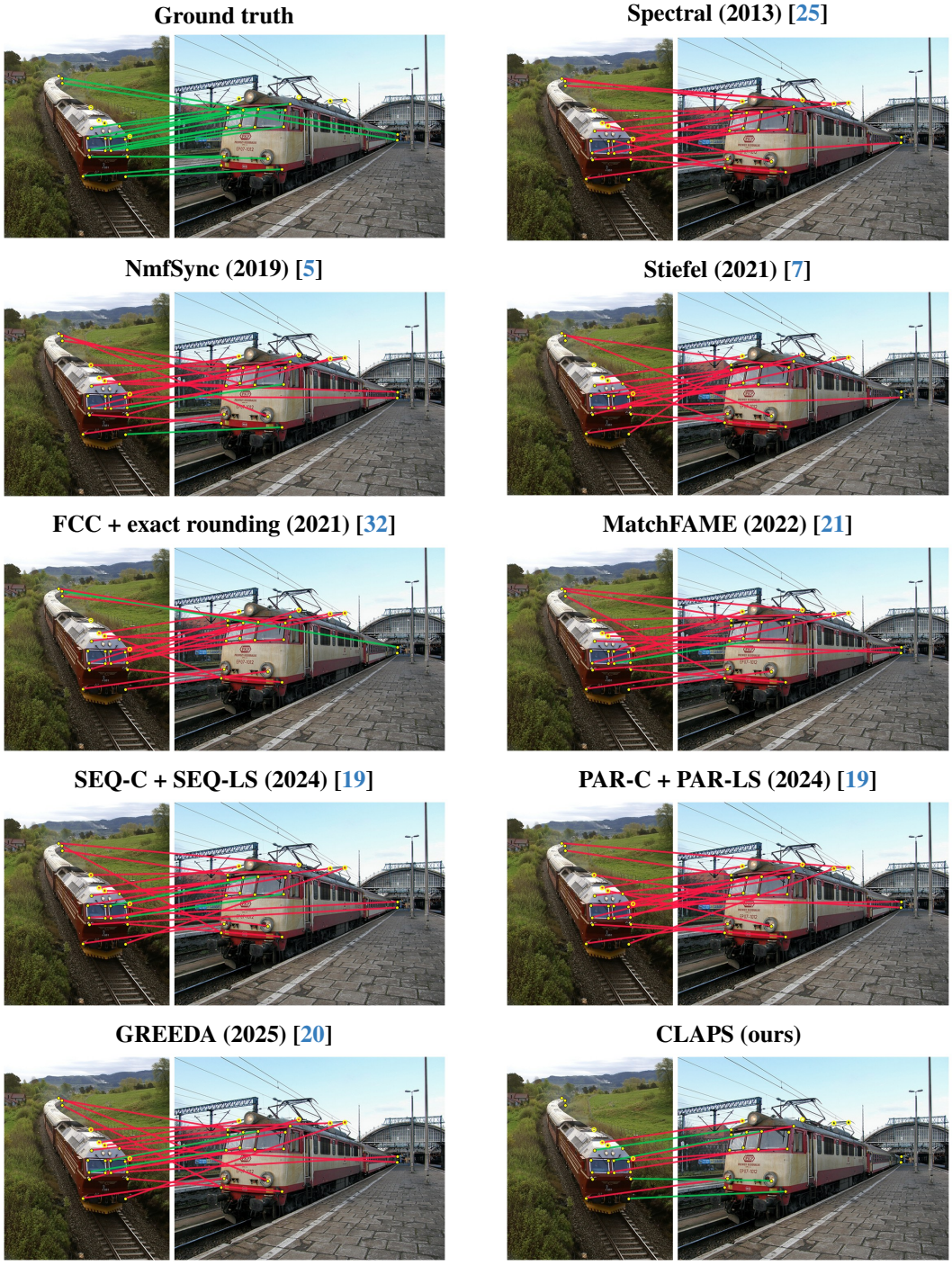

Figure S11: Correspondences on an image pair of the SPair-71k category *train*. Each panel shows the correspondences recovered by one method. Green lines agree with the annotation and red lines are wrong; filled yellow dots are annotated keypoints and hollow dots are distractors, which carry no universe label.