Finding Matches in a Haystack: A Max-Pooling Strategy for Graph Matching in the Presence of Outliers

Minsu Cho[1,*] Jian Sun[2,1,*] Olivier Duchenne[3] Jean Ponce[4,*]

[1]Inria [2]Xi'an Jiaotong Univ. [3]Intel Korea [4]École Normale Supérieure

Abstract

A major challenge in real-world feature matching problems is to tolerate the numerous outliers arising in typical visual tasks. Variations in object appearance, shape, and structure within the same object class make it harder to distinguish inliers from outliers due to clutters. In this paper, we propose a max-pooling approach to graph matching, which is not only resilient to deformations but also remarkably tolerant to outliers. The proposed algorithm evaluates each candidate match using its most promising neighbors, and gradually propagates the corresponding scores to update the neighbors. As final output, it assigns a reliable score to each match together with its supporting neighbors, thus providing contextual information for further verification. We demonstrate the robustness and utility of our method with synthetic and real image experiments.

1. Introduction

Feature matching lies at the heart of computer vision research, and most vision problems involve correspondence tasks in different forms [13]. A major challenge in real-world matching problems is to tolerate the numerous outliers arising in typical situations. In images and videos, objects of interest often appear small against dominant backgrounds, and also involve variations in appearance, shape, and structure. These variations in cluttered scenes make it harder to distinguish inlier features from outliers. To tackle real-world vision tasks, thus, feature matching should be robust to a large number of outlier features while effectively using mutual relations among features.

This paper addresses the issues with a max-pooling approach to graph matching [8]. Let us suppose we have a large set of candidate matches from the features that make up a target object with numerous false candidates. In evaluating a candidate match, the nearby matches are useful contextual information, and graph matching provides a powerful framework for exploiting these relational similarities. We propose to adopt a feature-pooling perspective [3] into this graph matching framework. The proposed method computes the score of each candidate match using maximal support from nearby matches, which corresponds to max-pooling of nearby features. As both the max-pooled matches and the scores improve each other in our optimization process, the method efficiently avoids the adverse effect of false matches or outliers. Our experiments demonstrate its robustness to outliers, and significant improvement over recent state of the art methods.

(a) A cluttered scene and its extracted features

(b) Feature matching between two images

Figure 1. Feature matching in the presence of outliers. (a) In real-world scenes, background clutter often produces numerous outlier features, making it hard to find correspondences. (b) We address the issue with a max-pooling approach to graph matching. The proposed method is not only resilient to deformations but also remarkably tolerant to outliers. Each node on the left image corresponds to one with the same color on the right image, where bigger nodes represent more similar nodes. (best viewed in color.)

*WILLOW project-team, Département d'Informatique de l'Ecole Normale Supérieure, ENS/Inria/CNRS UMR 8548.

1.1. Related work

Graph matching is widely used in computer vision problems such as finding feature correspondences [7, 10, 25], shape matching [22, 30], object recognition [2, 11], and video analysis [4]. Since graph matching is mathematically expressed as a quadratic assignment problem [8], which is NP-hard, most research has long focused on developing accurate and efficient approximate algorithms [14, 27]. In feature matching and object recognition, both a reference and a test scene are represented as graphs using visual features, and graph matching finds correspondences by minimizing the structural distortions between the two graphs. Contrary to rigid geometric constraints used in popular methods such as RANSAC and the Hough transform, *e.g.*, planar assumptions or epipolar geometry [13], graph matching allows non-rigid deformations and provides greater robustness in matching and recognition [2, 11, 19]. As observed in recent evaluations [6, 20, 32], however, current methods are still vulnerable to the large amount of outliers, that typically arise in highly cluttered scenes. Our approach tackles this issue based on a max-pooling scheme, and improves outlier-tolerance over the current state of the arts.

Feature pooling aims to transform feature representation into a more compact one that preserves important information while discarding irrelevant details [3]. The pooling operation typically performs a max, average, or sum over the feature responses in a local or global spatial region. Many recent recognition methods adopt it to compute local or global bags of features, *e.g.*, vector-quantizing feature descriptors, computing the codeword frequencies over local or global areas [16, 23, 29], and learning the invariant image features in convolutional neural networks [15, 17]. As witnessed by the success of the spatial pyramid model [16] and deep learning methods [15, 17], appropriate pooling strategies achieve invariance to image variations, more compact representations, and better robustness to noise. Our approach will show how these advantages can also be achieved in a graph matching framework.

2. Graph matching and max-pooling

In this section we revisit the standard formulation of graph matching, and describe our novel max-pooling formulation and its optimization in the graph matching framework.

2.1. Standard formulation

We are given two attributed graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$, where $\mathcal{V}$ represents a set of nodes and $\mathcal{E}$ represents a set of edges. Here, the graph $\mathcal{G}$ represents a target object model with its features as nodes and their relations as edges, and graph $\mathcal{G}'$ represents a scene. A solution of matching is defined as a subset of possible correspondences $\mathcal{X} \subset \mathcal{V} \times \mathcal{V}'$, represented by a binary assignment matrix $\mathbf{X} \in \{0,1\}^{n \times n'}$, where n and n' denote the number of nodes in $\mathcal{G}$ and $\mathcal{G}'$, respectively. If $v_i \in \mathcal{V}$ matches $v'_a \in \mathcal{V}'$, then $\mathbf{X}_{ia} = 1$, and $\mathbf{X}_{ia} = 0$ otherwise. We denote by $\mathbf{x} \in \{0,1\}^{nn'}$, a column-wise vectorized replica of $\mathbf{X}$.

The objective function $f(\mathbf{x})$ measures the mutual similarity of graph attributes, and is typically decomposed into a unary similarity function $\mathbf{s}_V(v_i, v'_a)$ for a node pair of v_i in $\mathcal{V}$ and v'_a in $\mathcal{V}'$, and a pairwise similarity function $\mathbf{s}_E(e_{ij}, e'_{ab})$ for an edge pair of e_{ij} in $\mathcal{E}$ and e'_{ab} in $\mathcal{E}'$. The similarity functions are usually represented by a symmetric affinity matrix $\mathbf{A}$, where non-diagonal elements are assigned as $\mathbf{A}_{ia;jb} = \mathbf{s}_E(e_{ij}, e'_{ab})$, and diagonal terms as $\mathbf{A}_{ia;ia} = \mathbf{s}_V(v_i, v'_a)$. In general, the standard objective function of graph matching is defined as:

$$\begin{aligned} f(\mathbf{x}) &= \sum_{\substack{\mathbf{x}_{ia}=1 \\ \mathbf{x}_{jb}=1}} \mathbf{s}_E(e_{ij}, e'_{ab}) + \sum_{\mathbf{x}_{ia}=1} \mathbf{s}_V(v_i, v'_a) \\ &= \mathbf{x}^\intercal \mathbf{A} \mathbf{x}. \end{aligned} \tag{1}$$

Finally, the graph matching problem can be expressed as finding the assignment vector $\mathbf{x}^*$ that maximizes the objective function $f(\mathbf{x})$ as follows:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \mathbf{x}^\intercal \mathbf{A} \mathbf{x} \tag{2a}$$

$$s.t. \begin{cases} \mathbf{x} \in \{0,1\}^{nn'} \\ \sum_{i=1}^{n} \mathbf{x}_{ia} \leq 1, \;\; \sum_{a=1}^{n'} \mathbf{x}_{ia} \leq 1, \end{cases} \tag{2b}$$

where Eq. (2b) induces the matching constraints, thus making $\mathbf{x}$ an *assignment* vector [6, 9, 14, 18]. In essence, the objective function accumulates all the affinity values relevant to the assignment. The formulation in Eq. (2) is an integer quadratic programming (IQP) problem. More precisely, it is the quadratic assignment problem, which is known to be NP-hard. Due to its generality and flexibility, this formulation and its extensions has been favored in recent graph matching research [6, 7, 10, 18, 20, 24, 31, 32].

2.2. Max-pooling for matching

The goal of feature pooling is to transform a joint feature representation into a more compact one that preserves important information while discarding irrelevant details [3, 16, 23, 29]. Here we introduce a feature pooling perspective in the graph matching framework. Previous algorithms modify the original problem of Eq. (2), usually by relaxing the constraints of Eq.(2b) on the solution, in order to alleviate the NP-hard property in the original problem. A common approach is to relax the integer and matching constraints of Eq.(2b) from the standard objective, and aim at maximizing the quadratic function $f(\mathbf{x}) = \mathbf{x}^\intercal \mathbf{A} \mathbf{x}$ in a continuous domain. The function can be approximated by the first order Taylor expansion around the current solution $\mathbf{x}_k$:

$$f(\mathbf{x}) \approx f(\mathbf{x}_k) + (\mathbf{x} - \mathbf{x}_k)^\intercal \mathbf{A} \mathbf{x}_\mathbf{k}. \tag{3}$$

In the iterative first-order optimization framework, maximizing the second term $\mathbf{x}^\intercal \mathbf{A}\mathbf{x}_\mathbf{k}$ with a unit l^2-norm constraint boils down to the following update at each iteration:

$$\mathbf{x}_{k+1} \leftarrow \frac{1}{\|\mathbf{A}\mathbf{x}_k\|_2}\mathbf{A}\mathbf{x}_k, \tag{4}$$

which is equivalent to the power method of spectral matching [18]. Methods of [6, 10, 14, 20] combine an additional projection step with this to guide the solution closer to a feasible region consistent with matching constraints. All these iterative optimization methods, however, share the similar form of Eq.(4) at their heart. In the context of graph matching, given two graphs $\mathcal{G}$ and $\mathcal{G}'$, each element $\mathbf{x}_{ia}$ represents the assignment confidence associated with a candidate match (v_i, v'_a). The confidence $\mathbf{x}_{ia}$ is updated by Eq.(4) in which $\mathbf{A}\mathbf{x}$ can be written as:

$$\begin{aligned}(\mathbf{A}\mathbf{x})_{ia} &= \mathbf{x}_{ia}\mathbf{A}_{ia;ia} + \sum_{j\in N_i}\sum_{b\in N_a}\mathbf{x}_{jb}\mathbf{A}_{ia;jb} \\ &= \mathbf{x}_{ia}\mathbf{s}_V(v_i, v'_a) + \sum_{j\in\mathcal{N}_i}\sum_{b\in\mathcal{N}_a}\mathbf{x}_{jb}\mathbf{s}_E(e_{ij}, e'_{ab}).\end{aligned} \tag{5}$$

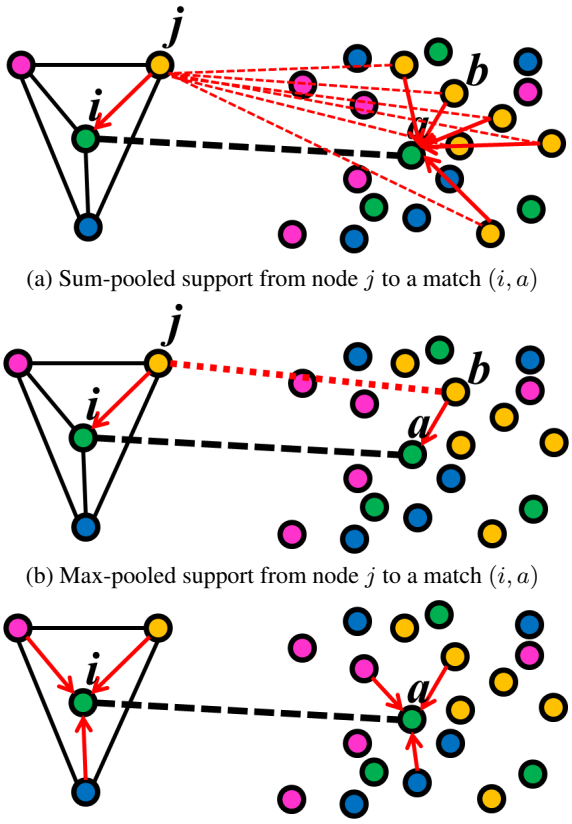
The second term can be seen as accumulating scores from $\mathcal{N}_i$, each of which is the sum of weighted pairwise affinities: $\sum_{b\in N_a}\mathbf{A}_{ia;jb}\mathbf{x}_{jb}$. Figure 2(a) shows how this works, *i.e.*, it sums over all the affinities, no matter whether they really match to node a. In this perspective, the product $\mathbf{A}\mathbf{x}$ used in the standard formulation of Eq.(2a) can be viewed as performing *sum-pooling* or *average-pooling* [3], which accumulates all the weighted affinities with respect to each match to measure the confidence of how compatible the match is with the others. A problem of sum-pooling, commonly used in other matching methods, is that it is strongly influenced by uninformative or irrelevant elements, often resulting in bad local minimum.

To address this issue, we instead take the following max-pooling product:

$$(\mathbf{A}\circledast\mathbf{x})_{ia} = \mathbf{x}_{ia}\mathbf{A}_{ia;ia} + \sum_{j\in\mathcal{N}_i}\max_{b\in\mathcal{N}_a}\mathbf{x}_{jb}\mathbf{A}_{ia;jb} \tag{6}$$

which collects the best pairwise affinity from each neighbor. Unlike the sum-pooled confidence of $\mathbf{A}\mathbf{x}$, which sums up all weighted affinities of candidate matches for each feature, the max-pooled confidence of $\mathbf{A}\circledast\mathbf{x}$ corresponds to the maximum possible matching score for each feature. Each neighborhood can be seen as a spatial bin for pooling [16].

There are two clear advantages in using $\mathbf{A}\circledast\mathbf{x}$: (i) While $\mathbf{A}\mathbf{x}$ in the relaxed continuous domain inevitably contains a large amount of noisy scores from outliers, max-pooling selection in $\mathbf{A}\circledast\mathbf{x}$ effectively suppresses the noisy scores from numerous outlier features by ignoring most of them. This effect is similar to that of median filtering [1, 28] in signal processing, which removes noise by taking the median value of neighboring signals rather than the average value. (ii) Max pooling in $\mathbf{A}\circledast\mathbf{x}$ is very likely to provide higher scores only on true matches because the highest affinities from them are always selected and accidental high affinities occur rarely with outliers. These advantages are critical in practice because we usually need to collect a large number of candidate matches to avoid losing true matches among them, which leads to facing numerous outlier matches.



(a) Sum-pooled support from node j to a match (i, a)

(b) Max-pooled support from node j to a match (i, a)

(c) All max-pooled supports to a match (i, a)

Figure 2. Feature-pooling perspective for a match (i, a). Features of a target object are shown as nodes on the left domain, and features of candidate matches, represented by nodes in the same color, are detected on the right domain. (a) Sum-pooling in Eq.(5) collects the weighted affinities of all possible matches (denoted by yellow nodes) from each neighbor j. (b) Max-pooling in Eq.(6), a candidate match (j, b) with the highest weighted affinity $\mathbf{x}_{ia;jb}\mathbf{s}_E(e_{ij}, e'_{ab})$ is selected among all possible matches as a support of j for (i, a). (c) In max-pooling, each candidate match has its own max-pooled support from all neighbors and a max-pooled score as a sum of their weighted affinities. $\mathbf{A}\circledast\mathbf{x}$ of Eq.(6) computes such max-pooled confidences for all candidate matches. (Best viewed in color.)

2.3. Proposed algorithm

Using the max-pooling product, we propose to address graph matching as follows. From the first-order approximation of Eq.(3), we take the max-pooling product $\mathbf{A}\circledast\mathbf{x}$,

Algorithm 1: Max-pooling matching (MPM)

Input: affinity matrix $\mathbf{A}$
Output: soft-assignment $\mathbf{x}$

Initialize the starting assignment $\mathbf{x}$ as uniform;
repeat
  for each candidate match (i, a) **do**
    $\mathbf{x}_{ia} \leftarrow$
    $\mathbf{x}_{ia}\mathbf{A}_{ia;ia} + \sum_{j \in \mathcal{N}_i} \max_{b \in \mathcal{N}_a} \mathbf{x}_{jb}\mathbf{A}_{ia;jb}$;
  $\mathbf{x} \leftarrow \frac{1}{\|\mathbf{x}\|_2}\mathbf{x}$;
until $\mathbf{x}$ *converges*;
Discretize the solution $\mathbf{x}$ if needed;

instead of the sum-pooling product $\mathbf{Ax}$. Then, we have the local objective at each iteration: $\mathbf{x}_{k+1} = \arg\max_{\mathbf{x}} \mathbf{x}^\intercal(\mathbf{A} \circledast \mathbf{x}_k)$ under the unit l^2-norm of $\mathbf{x}$. Finally, we obtain the following iteration:

$$\mathbf{x}_{k+1} = \underset{||\mathbf{x}||_2=1}{\arg\max}\, \mathbf{x}^\intercal(\mathbf{A} \circledast \mathbf{x}_k) = \frac{1}{\|\mathbf{A} \circledast \mathbf{x}_k\|_2}\mathbf{A} \circledast \mathbf{x}_k. \tag{7}$$

This method, dubbed max-pooling matching (MPM), is described in Algorithm 1. Since $\mathbf{x}^\intercal\mathbf{A}\mathbf{x}_k \geq \mathbf{x}^\intercal(\mathbf{A} \circledast \mathbf{x}_k)$[1], the proposed algorithm only maximizes a lower bound of the first-order approximation of Eq.(3) at each iteration. In this algorithm, max-pooling implicitly involves surjective constraints, where each candidate correspondence is assigned the best supporting matches as shown in Fig. 2(b). These are "soft" constraints (*e.g.*, as opposed to ones in [20, 24]), the consistent elements of $\mathbf{x}$ being driven toward a solution satisfying these constraints (*e.g.*, as similar to ones in [6, 10, 14]). MPM itself does not prevent one-to-many nor many-to-one matches if they are well supported by max-pooled neighboring matches. It is the final discretization step that enforces any matching constraints explicitly.

In general, MPM iterations do not monotonically increase the objective function of Eq.(2a). As can be seen in the experiments of Sec. 3.1, however, MPM turns out to be remarkably robust to outliers, and effectively optimizes the objective function after the final discretization, compared to recent state of the arts [6, 9, 18, 20]. In all our experiments, convergence has been obtained in 10 to 50 steps, but we do not have theoretical guarantees in general yet. A theoretical understanding of the convergence of our approach is clearly desirable, but left for future work.

Complexity. The computational complexity of MPM is $O(nmkk')$ per iteration, where n and m represent the number of nodes in $\mathcal{G}$ and the number of candidate matches for each node, and k and k' denotes the size of neighborhood in $\mathcal{G}$ and $\mathcal{G}'$, respectively. In practice, this is comparable or faster in speed than the recent state-of-the-art algorithms [6, 20, 25, 32]. Restricting the neighbors $\mathcal{N}_i$ of node v_i boils down to different kinds of spatial max-pooling centered on each feature i, and further reduces computational complexity.

[1]Elements in $\mathbf{A}$, $\mathbf{x}_k$, $\mathbf{x}$ are all non-negative, and the elements summed in $\mathbf{A} \circledast \mathbf{x_k}$ correspond to a max-pooled subset of elements summed in $\mathbf{A}\mathbf{x_k}$. Therefore, it is obvious that $\mathbf{x}^\intercal\mathbf{A}\mathbf{x}_k \geq \mathbf{x}^\intercal(\mathbf{A} \circledast \mathbf{x}_k)$.

Relation with other algorithms. MPM can be compared with other graph matching algorithms, which adopt a similar iterative optimization strategy [6, 9, 14, 18]. The iterative update in MPM resembles a gradient descent step in GA [14], a power method in SM [14], and a random walk step in RRWM [6]. Unlike those, however, the update step of MPM performs pooling of best features instead of using all observations, which make it robust to outliers. In the view of random walks for graph matching [6], this corresponds to a constrained random walking, which takes a random walk to one of max-pooled matches at each step.

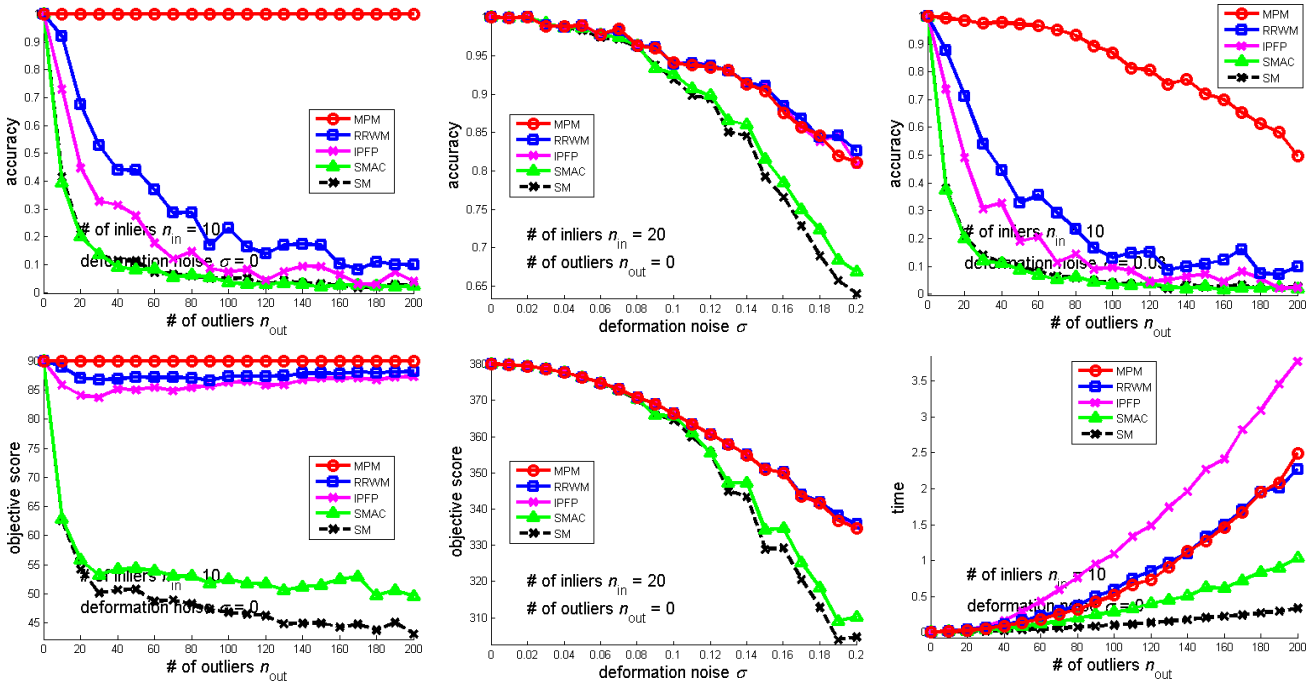
2.4. Practical advantages in vision problems

The proposed method is adequate for practical matching problems in computer vision where the following critical issues frequently arise in many applications.

Clutter and deformations. In real-world images, target objects appear as small regions whereas clutters dominate larger regions. Even object regions also generate distracting features as feature detections are not perfectly stable nor repeatable [26]. These numerous outliers become unsurmountable obstacles to matching. Furthermore, variations in appearance and shape make it harder to discern true matches from outliers. MPM provides strong robustness against outliers using adaptive max pooling, and addresses object deformations by minimizing distortion in both shape and appearance.

Contextual support. MPM provides reliable soft-assignments $\mathbf{x}$ by gradually updating max-pooled neighbor matches. In the end, each candidate match (i, a) is assigned not only a score $\mathbf{x}_{ia}$ but also its max-pooled neighbor matches $\mathcal{P}(i, a) = \{(j, b) \mid \exists j \in \mathcal{N}_i, b^* = \arg\max_{b \in \mathcal{N}_a} \mathbf{x}_{jb}\mathbf{A}_{ia;jb}\}$. These max-pooled neighbors are useful for further tasks. For example, they function as contextual information for verifying the presence of the match they support. They also reveal explicit relations between each match and its neighbors, providing better evidence for its correctness than relative scores of the soft-assignment $\mathbf{x}$. Such better verified matches also could lead to more accurate localization of objects [12].

Structural flexibility. Despite surjective constraints used in max pooling, MPM itself does not necessarily force the result to be one-to-one correspondence. Any candidate match with strong max-pooled supports keeps a high score in $\mathbf{x}$. In other words, MPM does not prevent one-to-many



(a) Increasing outliers w/o deformation (b) Increasing deformation w/o outliers (c) Increasing outliers with deformation

Figure 3. Comparative experiments on synthetic point sets varying the amount of outliers and deformation. To better show the effects of max-pooling, our algorithm (MPM) is compared to other recent algorithms, RRWM [6], IPFP [20], SMAC [9], and SM [18]. The top row presents accuracy while the first and the second plots in the bottom row show the objective score. The third plot in the bottom represents the average computation time with increasing outliers. (a) The number of outliers n_{out} was varied from 0 to 200 by intervals of 10 without deformation. (b) The deformation noise σ was varied from 0 to 0.2 by intervals of 0.01 without any outliers. (c) The number of outliers n_{out} was varied with deformation $\sigma = 0.03$. Our method shows remarkable tolerance to outliers. In the aspect of deformation, our method is comparable to the best methods, IPFP and RRWM. In a realistic situation with both of deformations and outliers, shown in (c), our algorithm significantly outperforms all the other state of the arts. (Best viewed in color.)

nor many-to-one matches if they are well supported by max-pooled neighboring matches. This enables to address structural flexibility in matching, which is important for many generic objects [21]. For example, a clover may have three or four leaves, and an airplane may have four, two or no visible engines. The number of windows in a house are highly variable. There are also many kinds of repetitive patterns in nature as well as man-made objects. MPM is particularly adequate for dealing with such flexible structures without strict matching constraints.

3. Experimental evaluation

We evaluate the proposed algorithms on standard synthetic benchmarks and real image datasets. For comparison to the state of the art, reweighted random-walk matching (RRWM) [6], integer projected fixed-point matching (IPFP) [20], and spectral matching with affine constraint (SMAC) [9] are evaluated in the same setting[2]. To better measure the effect of max-pooling, spectral matching (SM) [18] is also compared as a baseline, since it uses a comparable optimization strategy with sum pooling ($\mathbf{Ax}$) instead of the max pooling ($\mathbf{A} \circledast \mathbf{x}$).

[2]For the algorithms, the original implementations from the author's websites were used in all our experiments.

3.1. Synthetic point set matching

This section presents comparative evaluations on the task of random point set matching, which is widely used as a benchmark test for graph matching [6, 9, 18]. Synthetic point sets P and P' are built as follows. For the point set P, n_{in} inlier points are randomly generated on $\mathbb{R}^2$ using Gaussian distribution $\mathcal{N}(0, 1)$. Each point in P is then copied to the point set P' with additional Gaussian noise $\mathcal{N}(0, \sigma^2)$. Further outlier noise is created in P' by adding n_{out} random points using $\mathcal{N}(0, 1)$. In this setup, we consider the point sets P and P' as the graphs $\mathcal{G}$ and $\mathcal{G}'$, and test all the methods on them to quantitatively evaluate the matching accuracy. The number of inliers is fixed as $n_{\text{in}} = 10$ or 20. For the pairwise similarity, the difference between the

Table 1. Performance on the object class dataset [5]. (learn ×: results without learning; learn ○: results with learning.)

	Face		Motorbike		Car		Duck		Wine bottle	
Method	learn ×	learn ○	learn ×	learn ○	learn ×	learn ○	learn ×	learn ○	learn ×	learn ○
SM [18]	20.3	35.0	28.7	51.3	30.0	46.3	31.3	46.3	54.4	65.2
IPFP [20]	33.7	83.3	39.5	56.0	40.8	55.2	42.1	59.2	72.7	85.3
RRWM [6]	47.1	84.1	40.7	**63.7**	40.3	54.8	44.7	59.3	70.8	84.4
MPM (ours)	68.9	**92.2**	41.7	61.3	36.7	**61.7**	46.0	**65.2**	68.6	**87.8**

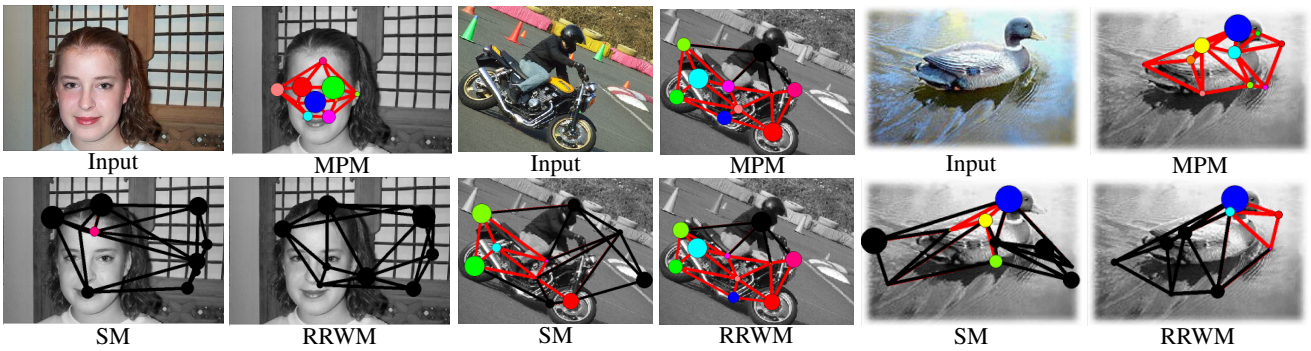


Figure 4. Comparative matching examples on the object class dataset. Given an input image, features of each learned object model are matched and localized as circles, where bigger circles represent more similar features. Correctly matched features are connected by red lines, and otherwise by black lines. See Table 1 for the quantitative result on the entire dataset. (Best viewed in color.)

Euclidean distances of two point pairs is employed:

$$\mathbf{s}_E(e_{ij}, e'_{ab}) = \exp(-(\|\mathbf{p}_i - \mathbf{p}_j\| - \|\mathbf{p}'_a - \mathbf{p}'_b\|)^2/\sigma_s^2). \quad (8)$$

where $\sigma_s^2 = 0.5$ in this experiment. No unary similarity is used: $\mathbf{s}_V(v_i, v'_a) = 0$. This distance-based measure has been used in other papers [6, 9, 18]. For all the methods, the same linear assignment of the Hungarian algorithm is performed on the resultant assignment vector as a post-processing scheme enforcing one-to-one constraints.

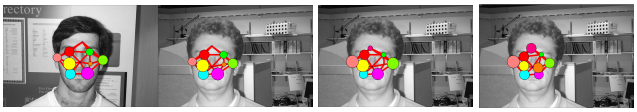
The experimental results are shown in Fig.3. The proposed method shows remarkable tolerance to the number of outliers, while all the other methods drop shapely with the number of outliers increases. Note that our experiment uses a large number of outliers (up to 20 times more than inliers) compared to the experiments in [6, 9, 18, 32]. In the case without deformation, as shown in Fig.3(a), the proposed method performs perfectly without being distracted by outliers, because our max-pooling provides accurate neighboring supports when there is no deformation. Under deformation only, IPFP, RRWM, and our method perform comparable each other. For high deformation levels, MPM's robustness to deformation becomes slightly worse than IPFP and RRWM. In a more realistic situation with both of deformations and outliers, shown in Fig.3(c), our method clearly outperforms the current state of the art. In the presence of both much larger deformations and more outliers together, the performance gap between MPM and other methods becomes modest as the problem itself starts making less and less sense. In practice, however, MPM strongly benefits from its robustness to outliers as shown in the following real image experiments.

3.2. Object class learning and matching

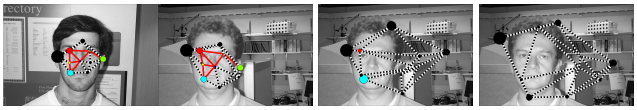
We experimented on real object class images in this section. In order to test the proposed method in the context of learning, we adopted the recent method of [5] to learn a graph model for matching, and perform a quantitative comparison. This learning framework is particularly useful for evaluation because any graph matching algorithm can be used as a graph matching module. For the experiment, we use the publicly available dataset used in [5], which includes annotated images on 5 object classes[3]. These dataset consists of images from Caltech-256 and PASCAL VOC2007. For each image, 10 distinctive features are annotated for the target object. We split the image set for each object class into two partitions, and report accuracy and error of matching, averaged over the 10 random splits. The scale-invariant Hessian detector is used to detect local regions as nodes. As unary and pairwise affinity functions, we adopt the histogram-based affinity measures used in [5].

To observe the robustness to outliers, we use a larger set of candidate matches than that of [5]. Given a test image, we select 200 nearest neighbor features for each node of the model graph, which is four time more than that used in the original experiment. In this setting, both learning and matching become more challenging due to a larger number of distracting outliers. We compare the proposed method with SM, IPFP, and RRWM in the framework. Matching performance is evaluated without learning as well as with learning. The results for all the methods are summarized

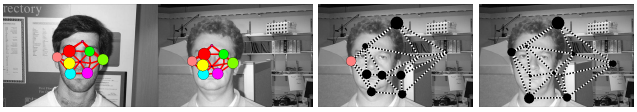
[3]For the detailed information, refer to the project site: http://www.di.ens.fr/willow/research/graphlearning/



(a) MPM with 500, 1000, and 2000 candidate matches.



(b) SM with 500, 1000, and 2000 candidate matches.



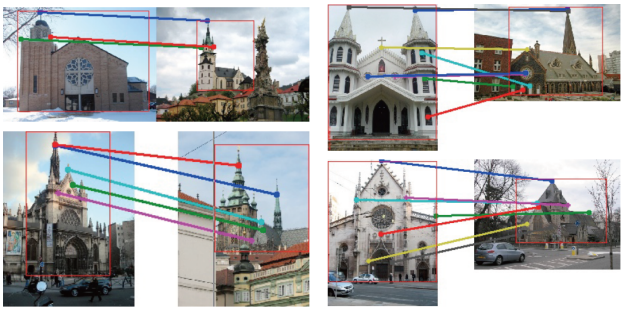
(c) RRWM with 500, 1000, and 2000 candidate matches.

Figure 5. Image matching varying the amount of outliers. For each feature on the reference image (leftmost), the best k candidate matches are selected on the input image according to the SIFT distance. From the second to the fourth column, k = 50, 100, 200 are used, respectively. More candidates involve more outliers, making the problem harder. (Best viewed in color.)

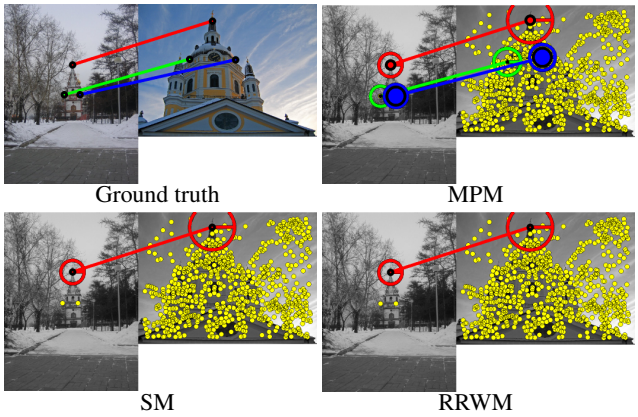
in Table 1, and some comparative examples are shown in Fig. 4. This experiment demonstrates that our approach successfully handles a large number of outliers in practical learning and matching. Interestingly, without learning, we can see significantly better performance of MPM on Face, which is relatively more rigid than other classes. In the other classes, MPM is comparable to IPFP and RRWM. With learning, however, MPM clearly outperforms others in most classes. It means that the proposed method provides better performance given an adequate graph model. Figure 5 presents a comparative example where MPM outperform the others with increasing outliers in the Face class.

3.3. Building landmark matching

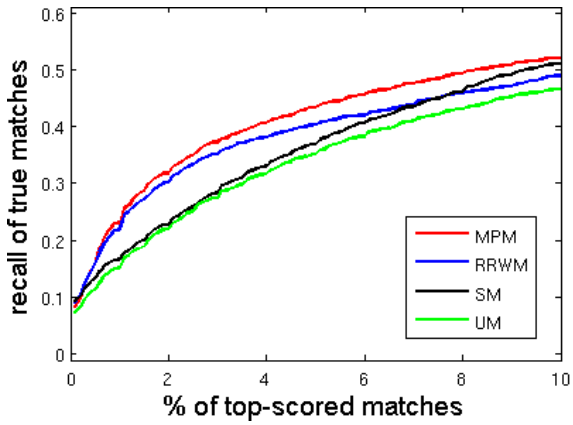
The datasets used in the previous experiments deal with deformable but isomorphic objects, where one-to-one correspondences usually hold between matching features. In this experiment we evaluate flexible matching performance using the partial correspondence dataset [21]. The dataset contains correspondence annotations for 1000 image pairs among 288 images of church buildings downloaded from Flickr. Those annotations were obtained on Amazon Mechanical Turk by asking subjects to click on pairs of matching landmark points in two instances of the category. The annotated pairs, a few examples of which are shown in Fig. 6(a), contain a variety of semantic matches on identical structural elements of buildings, and allows one-to-many or many-to-one relations among them. Finding these correspondences, which have high variability in both appearance and structure, is a challenging problem. We run each method on this dataset, and evaluate a matching performance by measuring the average recall ratio of the top k percentage matches with increasing k. Those top matches



(a) Landmark annotation examples from the dataset of [21]



(b) Comparative matching results of different algorithms



(c) Recall of true matches among top-k percentage matches.

Figure 6. Matching performance on the partial correspondence dataset [21]. (a) Landmark annotations include a variety of semantic matches between flexible structures allowing one-to-many or many-to-one relations, such as windows, spires, corners, and gables. (b) Comparative results are shown for an example, where only true matches are visualized. (c) Matching performance is measured as the ratio of true matches among the top k percentage of matches. While increasing k, we compare ours to RRWM [6], SM [18], and a simple appearance-based matching (UM) as a baseline. (Best viewed in color.)

are chosen based on the confidence score obtained by each method. We use the soft-assignment value, which is not

discretized, of each method as the confidence score. MPM is compared to RRWM and SM. As a baseline, we add a simple appearance-based method, dubbed UM, which only uses the similarity of SIFT descriptors without any pairwise affinity. The results are reported in Fig. 6(c). While the two other graph matching methods, RRWM and SM, show better performance than the appearance-based method, MPM outperforms all of them in recall, taking more true matches to the top ranks. As mentioned earlier, it implies that our method is robust to structural flexibility because any match with strong max-pooled support keeps a high score.

4. Conclusion

We proposed a novel matching method based on a max-pooling strategy within the graph matching framework. The proposed method is well suited for practical matching problems in computer vision where numerous outlier features occur from clutter. In our synthetic and real image experiments, we demonstrated its robustness to outliers and its utility in real applications. We believe the feature-pooling perspective introduced in this paper is worth further research. More theoretical understandings of our algorithm are also left for future work.

Acknowledgments. The authors would like to thank Francis Bach for helpful discussions, and Subhransu Maji for providing his data. This work was supported in part by the ERC grant VideoWorld and the Institut Universitaire de France.

References

[1] G. Arce. *Nonlinear signal processing: a statistical approach*. Wiley:New Jersey, USA, 2005.

[2] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *CVPR*, 2005.

[3] L. Bourdev, S. Maji, and J. Malik. Describing people: a poselet-based approach to attribute classification. *ICCV*, 2011.

[4] W. Brendel and S. Todorovic. Learning spatiotemporal graphs of human activities. In *ICCV*, 2011.

[5] M. Cho, K. Alahari, and J. Ponce. Learning graphs to match. In *ICCV*, 2013.

[6] M. Cho, J. Lee, and K. M. Lee. Reweighted random walks for graph matching. In *ECCV*, 2010.

[7] M. Cho and K. M. Lee. Progressive graph matching: Making a move of graphs via probabilistic voting. In *CVPR*, 2012.

[8] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *IJPRAI*, 2004.

[9] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *NIPS*, 2007.

[10] O. Duchenne, F. Bach, I. Kweon, and J. Ponce. A tensor-based algorithm for high-order graph matching. In *CVPR*, pages 1980–1987, 2009.

[11] O. Duchenne, A. Joulin, and J. Ponce. A graph-matching kernel for object categorization. In *ICCV*, 2011.

[12] I. Endres, K. J. Shih, J. Jiaa, and D. Hoiem. Learning collections of part models for object recognition. In *CVPR*, 2013.

[13] D. Forsyth and J. Ponce. Computer vision: A modern approach. *Pearson Education Inc.*, 2011.

[14] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *Trans. PAMI*, 1996.

[15] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[16] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[17] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, 2009.

[18] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, 2005.

[19] M. Leordeanu, M. Hebert, and R. Sukthankar. Beyond local appearance: category recognition from pairwise interactions of simple features. In *CVPR*, 2007.

[20] M. Leordeanu and M. Herbert. An integer projected fixed point method for graph matching and map inference. In *NIPS*, 2009.

[21] S. Maji and G. Shakhnarovich. Part discovery from partial correspondence. In *CVPR*, 2013.

[22] A. Sharma, R. Horaud, J. Cech, and E. Boyer. Topologically-robust 3d shape matching based on diffusion geometry and seed growing. In *CVPR*, 2011.

[23] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *CVPR*, 2003.

[24] Y. Suh, M. Cho, and K. M. Lee. Graph matching via sequential monte carlo. In *ECCV*, 2012.

[25] L. Torresani, V. Kolmogorov, and C. Rother. Feature correspondence via graph matching: Models and global optimization. In *ECCV*, 2008.

[26] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: a survey. *Foundations and Trends® in Computer Graphics and Vision*, 2007.

[27] S. Umeyama. An eigen decomposition approach to weighted graph matching problems. *IEEE Trans. PAMI*, 1988.

[28] Z. Wang and D. Zhang. Progressive switching median filter for the removal of impulse noise from highly corrupted images. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 46(1):78–80, 1999.

[29] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[30] M. Zaslavskiy, F. Bach, and J. Vert. A path following algorithm for the graph matching problem. *PAMI*, 2009.

[31] R. Zass and A. Shashua. Probabilistic graph and hypergraph matching. In *CVPR*, 2008.

[32] F. Zhou and F. D. la Torre. Deformable graph matching. In *CVPR*, 2013.