Progressive Graph Matching:
Making a Move of Graphs via Probabilistic Voting

Minsu Cho and Kyoung Mu Lee
Dept. of EECS, ASRI, Seoul National University, Korea
chominsu@gmail.com kyoungmu@snu.ac.kr

Abstract

Graph matching is widely used in a variety of scientific fields, including computer vision, due to its powerful performance, robustness, and generality. Its computational complexity, however, limits the permissible size of input graphs in practice. Therefore, in real-world applications, the initial construction of graphs to match becomes a critical factor for the matching performance, and often leads to unsatisfactory results. In this paper, to resolve the issue, we propose a novel progressive framework which combines probabilistic progression of graphs with matching of graphs. The algorithm efficiently re-estimates in a Bayesian manner the most plausible target graphs based on the current matching result, and guarantees to boost the matching objective at the subsequent graph matching. Experimental evaluation demonstrates that our approach effectively handles the limits of conventional graph matching and achieves significant improvement in challenging image matching problems.

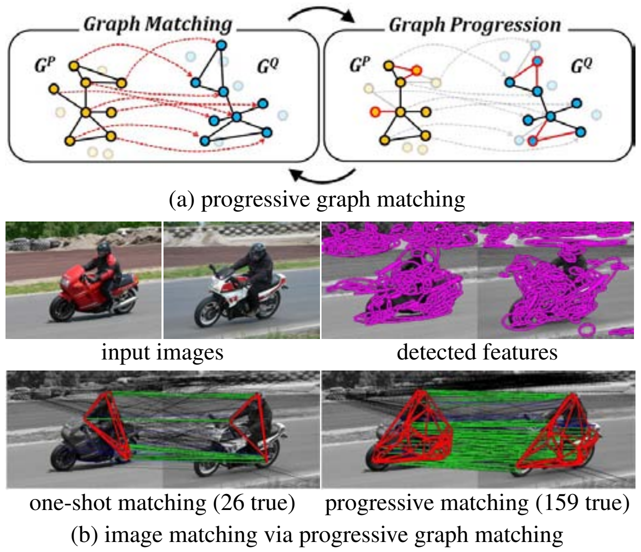

Graph Matching
Graph Progression
G^P
G^Q
G^P
G^Q
(a) progressive graph matching
input images
detected features
one-shot matching (26 true)
progressive matching (159 true)
(b) image matching via progressive graph matching

Figure 1. Our progressive graph matching framework. (a) *Graph matching* finds the best matches between two current graphs, and *graph progression* updates the graphs for the next step of graph matching. (b) It significantly improves feature correspondence between images beyond conventional graph matching. True matches are shown in green with red triangulations of feature centers.

1. Introduction

Graph matching is a powerful and robust technique in a variety of scientific fields, and is widely used in computer vision problems, such as feature correspondence, shape matching, object recognition, and video analysis [4, 18, 23, 24, 10, 7, 2, 11]. In shape matching and object recognition, both a model object and a test scene are represented as graphs using salient visual features, and graph matching finds the object and its corresponding features by minimizing the distortions of the two matching graphs. Unlike popular geometric constraints in vision (e.g. planar or rigid body assumptions), graph matching provides greater flexibility to the object model and promises robust matching and recognition [4, 18, 11]. The drawback of graph matching lies in its NP-hard nature. Although recent research has proposed various approximations [17, 9, 24, 10, 7], the computational costs in time and memory still limit the permissible sizes of input graphs to match. For instance, the full computation of edge attribute similarities is not tractable for large graphs due to their combinatorial nature. Thus, the similarity data for graph matching usually need to be reduced or sub-decimated at the cost of better performance. This is the main reason why most graph matching methods deal with problems where a relatively sparse set of discriminative features can be selected or pre-defined [4, 9, 24, 23, 10, 7]. In more realistic applications, such a limited construction of graph data essentially restricts its resultant performance.

As illustrated in Fig. 1, we propose a novel progressive framework for resolving the issue by combining probabilistic progression of graphs with matching of graphs. At each step, it efficiently re-estimates in a Bayesian manner the most plausible target graphs based on the current graph matching result. Exploring the space of graphs beyond the current graphs, this graph progression guarantees to boost the matching objective at the next step of graph matching. This framework can be viewed as a move-making algorithm for graph matching. In a move-making algorithm, the solution is updated within a certain move space for each iteration; for instance, move-making algorithms of graph-cuts [5, 16] generate a binary label space on MRFs to update the solution for multi-label problems. In the similar way, progressive graph matching creates adaptive move spaces of graphs to obtain better graph matching.

Although our approach provides a general framework for graph matching, the current work focuses on image matching applications. In this regard, it is worth noting that our progressive graph matching essentially differs from recent patch-based match-growing methods [14, 12, 8]. The methods are commonly driven by gradually duplicating the individual matches into their neighboring regions based on local appearances; thus, they are largely dependent on the initial matching and restricted to the case of identical objects with almost the same appearance. In contrast, our progressive matching optimizes geometric distortions of feature matches in a global sense, not by duplicating individual matches but by considering other potential matches in a Bayesian manner. Hence, our method becomes robust to appearance variation as well as outliers, and generally applicable to generic objects with intra-class variation. Experimental evaluation attests to this claim and shows that our approach effectively handles the limits of conventional one-shot graph matching and achieves significant improvement in challenging image matching problems.

2. Graph Matching Revisited

The objective of graph matching is to find correspondences between two attributed graphs $G^P = (\mathcal{V}^P, \mathcal{E}^P, \mathcal{A}^P)$ and $G^Q = (\mathcal{V}^Q, \mathcal{E}^Q, \mathcal{A}^Q)$, where $\mathcal{V}$ represents a set of nodes, $\mathcal{E}$, edges, and $\mathcal{A}$, attributes. A solution of graph matching $\mathcal{X}$ is defined as a subset of possible correspondences $\mathcal{X} \subset \mathcal{V}^P \times \mathcal{V}^Q$, and represented by a binary assignment matrix $\mathbf{X} \in \{0,1\}^{n^P \times n^Q}$, where n^P and n^Q denote the numbers of nodes in G^P and G^Q, respectively. If $v_i^P \in \mathcal{V}^P$ matches $v_a^Q \in \mathcal{V}^Q$, then $\mathbf{X}_{i,a} = 1$, or $\mathbf{X}_{i,a} = 0$. In this paper, we denote by $\mathbf{x} \in \{0,1\}^{n^P n^Q}$ a column-wise vectorized replica of $\mathbf{X}$. Then, graph matching problems can be generally expressed as finding the indicator vector $\mathbf{x}^*$ that maximizes a score function $S(\mathbf{x})$ as follows:

$$\begin{aligned} &\mathbf{x}^* = \arg\max_{\mathbf{x}} S(\mathbf{x}) \\ s.t. \quad &\begin{cases} \mathbf{x} \in \{0,1\}^{n^P n^Q} \\ \mathbf{X}\mathbf{1}_{n^Q \times 1} \preceq \mathbf{1}_{n^P \times 1}, \quad \mathbf{X}^T \mathbf{1}_{n^P \times 1} \preceq \mathbf{1}_{n^Q \times 1}, \end{cases} \end{aligned} \tag{1}$$

where the two-way constraints of Eq. (1) represent the one-to-one matching from G^P to G^Q, thus making $\mathbf{X}$ an *assignment matrix*. $\mathbf{1}_{n \times 1}$ denotes an all-ones vector with size n and the less-than-or-equal-to $\preceq$ holds for every element. The graph matching objective of Eq. (1) is directly related to progressive graph matching, as will be explained in Sec. 3.

For the score function $S(\mathbf{x})$, we adopt a quadratic assignment formulation [4, 18, 17, 9, 19, 7]. It assumes similarity functions that measure the mutual similarity of the graph attributes; the first-order similarity function $\Omega_1(\mathbf{a}_i^P, \mathbf{a}_a^Q)$ for a node pair of $v_i^P \in \mathcal{V}^P$ and $v_a^Q \in \mathcal{V}^Q$, and the second-order similarity function $\Omega_2(\mathbf{a}_{ij}^P, \mathbf{a}_{ab}^Q)$ for an edge pair of $e_{ij}^P \in \mathcal{E}^P$ and $e_{ab}^Q \in \mathcal{E}^Q$. The similarity functions can be encoded in a symmetric similarity matrix $\mathbf{W}$. A non-diagonal element $\mathbf{W}_{ia;jb} = \Omega_2(\mathbf{a}_{ij}^P, \mathbf{a}_{ab}^Q)$ contains a pairwise similarity of two correspondences (v_i^P, v_a^Q) and (v_j^P, v_b^Q), and a diagonal term $\mathbf{W}_{ia;ia} = \Omega_1(\mathbf{a}_i^P, \mathbf{a}_a^Q)$ represents a unary similarity of a correspondence (v_i^P, v_a^Q). Using the assignment vector $\mathbf{x}$ introduced earlier, a score function of the accumulation of all the relevant similarities is defined as

$$S(\mathbf{x}) = \mathbf{x}^T \mathbf{W} \mathbf{x}. \tag{2}$$

The formulation of Eq. (2) under Eq. (1) is called the integer quadratic programming (IQP), which is proven to be NP-complete. Recently, IQP has been favored in graph matching research due to its generality and strong robustness, and several efficient approximate algorithms have been proposed [7, 19, 9, 23]. Notably, our progressive graph matching framework is orthogonal to specific graph matching formulations and algorithms; thus, any of them or any high-order algorithm [24, 10, 15] can be adopted as the graph matching module in the framework.

2.1. Graph Construction for Real Applications

The issues of graph construction need to be discussed here in advance before we proceed to progressive graph matching. In the application of graph matching to real-world problems, the first step is to construct graphs based on given observations. As graphs are usually not given as raw data, nodes and edges should be defined based on a specific type of the given observations. For instance, salient features and their proper relations from an image constitute a scene graph used for image matching [4, 18]. In most real applications, however, graphs made of all possible features and their relations are not only intractable but

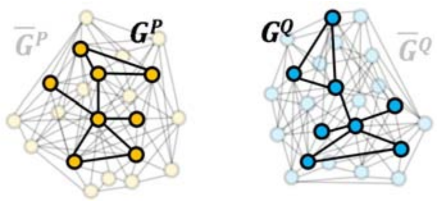


Figure 2. Illustration of active graphs within maximal graphs. In real applications, active graphs G^P and G^Q (solid line), which are subgraphs of maximal graphs $\bar{G}^P$ and $\bar{G}^Q$ (faded line), are used.

also inappropriate because such graphs are usually highly complex and contain too much clutter information, which distracts graph matching. Therefore, as depicted in Fig. 2, a reduced set of features and their relations is selectively used to construct a graph. We term such a graph used for practical graph matching an "*active graph*", whereas we call the largest complete graph covering all the given features a "*maximal graph*". Given two *active graphs*, similarity values of edge (and node) attributes between the two graphs should be computed. As repeatedly used in the graph matching process, the similarity values are commonly precomputed and stored in the form of a similarity matrix (or tensor) [13, 17, 9, 23, 24, 10, 7, 15], which corresponds to $\mathbf{W}$ of Eq. (2) in our case. The size of non-zero elements in the similarity matrix is another critical factor to determine the complexity of the graph matching process.

Therefore, if a graph matching method is given, the ways of reducing complexity in practice are (1) to reduce the sizes of active graphs and (2) to sparsify the similarity matrix (reduce the number of similarity values to be considered in graph matching). An effective way to achieve both (1) and (2) is to establish a set of candidate matches. Once the candidate matches $\mathcal{C}$ are selected from the given possible features, the active graphs are determined by the nodes and edges involved in $\mathcal{C}$, and their similarity matrix is sparsified by restricting the edge similarity comparison to the pairs from $\mathcal{C}$. In many applications, the candidate matches can be readily established using unary descriptors of the features at a relatively low cost. Such methods are widely adopted in the practical problems of feature matching and object recognition in the literature [4, 18, 7, 10]. In many cases, however, all these methods for graph construction lead to unsatisfactory matching results due to the loss of useful information hidden in the *maximal graphs*.

3. Our Progressive Framework

We propose a progressive framework for graph matching to explore the space of *maximal graphs* beyond the *active graphs* in an efficient way. As sketched in Fig. 1, our framework basically consists of two alternating processes: *graph matching* and *graph progression*. *Graph matching* finds the best matches between current active graphs, whereas *graph progression* updates the active graphs and their similarity matrix to boost the score in the next graph matching. The objective of progressive graph matching is to further maximize the graph matching score in Eq. (1) by adaptively re-organizing graphs G^P and G^Q. To formalize this approach at time step t, we assume two active graphs $G^P_t = (\mathcal{V}^P_t, \mathcal{E}^P_t, \mathcal{A}^P_t)$ and $G^Q_t = (\mathcal{V}^Q_t, \mathcal{E}^Q_t, \mathcal{A}^Q_t)$, and candidate matches $\mathcal{C}_t \subset \mathcal{V}^P_t \times \mathcal{V}^Q_t$. The matching solution between G^P_t and G^Q_t is denoted by $\mathcal{M}_t (\subset \mathcal{C}_t)$ under the constraints of Eq. (1), and its score is denoted by s^{GM}_t. On the other hand, maximal graphs are represented by $\bar{G}^P = (\bar{\mathcal{V}}^P, \bar{\mathcal{E}}^P, \bar{\mathcal{A}}^P)$ and $\bar{G}^Q = (\bar{\mathcal{V}}^Q, \bar{\mathcal{E}}^Q, \bar{\mathcal{A}}^Q)$, where $\bar{G}^P \supset G^P_t$ and $\bar{G}^Q \supset G^Q_t$ for any t.

For graph progression, we consider a conditional joint probability distribution $p(V^{\mathrm{P}}, V^{\mathrm{Q}} | \mathcal{M}_t)$ of $V^{\mathrm{P}} \in \bar{\mathcal{V}}^{\mathrm{P}}$ and $V^{\mathrm{Q}} \in \bar{\mathcal{V}}^{\mathrm{Q}}$, the domain of which represents possible node matches of two maximal graphs; $p(v^{\mathrm{P}}_j, v^{\mathrm{Q}}_b | \mathcal{M}_t)$ denotes the conditional probability of a match $(v^{\mathrm{P}}_j, v^{\mathrm{Q}}_b)$ between the two maximal graphs. In graph progression, given the current graph matching $\mathcal{M}_t$, the distribution of $p(V^{\mathrm{P}}, V^{\mathrm{Q}} | \mathcal{M}_t)$ is estimated, and then the new set of candidate matches $\mathcal{C}_{t+1}$ is selected based on the distribution to construct the new active graphs G^P_{t+1} and G^Q_{t+1} and their similarity matrix. The new active graphs consist of the nodes and edges involved in $\mathcal{C}_{t+1}$ and their similarity matrix considers only the similarity values within $\mathcal{C}_{t+1}$. Under a managable size of $\mathcal{C}_{t+1}$, this graph progression effectively explores the space of maximal graphs beyond the current active graphs and thus enables better graph matching. In the progressive graph matching, the use of the inclusion constraint $\mathcal{C}_{t+1} \supset \mathcal{M}_t$ guarantees the non-decreasing score $s^{GM}_{t+1} \geq s^{GM}_t$ by the optimal graph matching of each step. The iterations continue until the score no longer increases.

3.1. Bayesian Formulation for Graph Progression

In a Bayesian framework, the conditional joint distribution $p(V^{\mathrm{P}}, V^{\mathrm{Q}} | \mathcal{M})$ for graph progression can be analyzed as follows. Suppose a current graph matching solution $\mathcal{M}_t = \{\mathbf{m}_1, \cdots, \mathbf{m}_{|\mathcal{M}_t|}\}$ where $\mathbf{m}_i = (v^{\mathrm{P}}_{p_i}, v^{\mathrm{Q}}_{q_i})$. To facilitate a generative model of $p(V^{\mathrm{P}}, V^{\mathrm{Q}} | \mathcal{M})$, we introduce an auxiliary variable of an anchor match $M \in \mathcal{M}_t$ so that $p(V^{\mathrm{P}}, V^{\mathrm{Q}} | \mathcal{M}_t)$ can be obtained by marginalizing $p(V^{\mathrm{P}}, V^{\mathrm{Q}}, M | \mathcal{M}_t)$ over M. Then, by the chain rule, it can be further decomposed as

$$\begin{aligned} p(V^{\mathrm{P}}, V^{\mathrm{Q}} | \mathcal{M}_t) &= \sum_{\mathbf{m}_i \in \mathcal{M}_t} p(V^{\mathrm{P}}, V^{\mathrm{Q}}, M = \mathbf{m}_i | \mathcal{M}_t) \\ &= \sum_{\mathbf{m}_i \in \mathcal{M}_t} p(V^{\mathrm{Q}} | V^{\mathrm{P}}, M = \mathbf{m}_i, \mathcal{M}_t) \\ &\qquad p(V^{\mathrm{P}} | M = \mathbf{m}_i, \mathcal{M}_t)\, p(M = \mathbf{m}_i | \mathcal{M}_t), \end{aligned} \tag{3}$$

where $p(M = \mathbf{m}_i | \mathcal{M}_t)$ expresses a conditional prior for choosing $\mathbf{m}_i$ as an anchor among $\mathcal{M}_t$. $p(V^{\mathrm{P}} | M = \mathbf{m}_i, \mathcal{M}_t)$ describes the probability of V^{P} relating to the anchor $\mathbf{m}_i$ among $\mathcal{M}_t$. $p(V^{\mathrm{Q}} | V^{\mathrm{P}}, M = \mathbf{m}_i, \mathcal{M}_t)$ expresses the probability of V^{Q} relating to the node V^{P} and the anchor $\mathbf{m}_i$ among $\mathcal{M}_t$. Interestingly, the marginalizing formulation of Eq. (3) can be interpreted as probabilistic voting, where the voting elements are individual anchor matches $\mathbf{m}_i \in \mathcal{M}_t$. Unlike the pseudo-probabilistic formulation of the Hough transform[1], our formulation is naturally fitted to probabilis-

[1] For the analysis on the Hough transform, refer to [3].

Algorithm 1: Progressive Graph Matching

```
Input: maximal graphs $\bar{G}^{\mathrm{P}}$, $\bar{G}^{\mathrm{Q}}$, # of candidates $N_c$
Output: graph matching $\mathcal{M}^*$ with the highest score
( Initial Active Graphs )
$\mathcal{C}_0 \leftarrow$ FindInitialCandidates($\bar{\mathcal{V}}^{\mathrm{P}}, \bar{\mathcal{V}}^{\mathrm{Q}}, N_c$);
( Progressive Loop )
$t \leftarrow 0, s_0^{\mathrm{GM}} \leftarrow 0$;
while score $s_{\mathrm{GM}}$ increases do
    ( Graph Matching )
    $(\mathcal{M}_t, s_t^{\mathrm{GM}}) \leftarrow$ GraphMatching($\mathcal{C}_t$);
    ( Graph Progression )
    $p(v_p^{\mathrm{P}}, v_q^{\mathrm{Q}} | \mathcal{M}_t) \leftarrow 0 \;\; \forall v_p^{\mathrm{P}} \in \bar{\mathcal{V}}^{\mathrm{P}}, v_q^{\mathrm{Q}} \in \bar{\mathcal{V}}^{\mathrm{Q}}$;
    foreach $\mathbf{m}_i = (v_p^{\mathrm{P}}, v_q^{\mathrm{Q}}) \in \mathcal{M}_t$ do
        $\mathcal{N}_{\mathrm{A}} \leftarrow \{v^{\mathrm{P}} \in \bar{\mathcal{V}}^{\mathrm{P}} | p(v^{\mathrm{P}} | \mathbf{m}_i, \mathcal{M}_t) > \epsilon\}$;
        foreach $v_j^{\mathrm{P}} \in \mathcal{N}_{\mathrm{A}}$ do
            $\mathcal{N}_{\mathrm{B}} \leftarrow \{v^{\mathrm{Q}} \in \bar{\mathcal{V}}^{\mathrm{Q}} | p(v^{\mathrm{Q}} | v_j^{\mathrm{P}}, \mathbf{m}_i, \mathcal{M}_t) > \epsilon\}$;
            foreach $v_b^{\mathrm{Q}} \in \mathcal{N}_{\mathrm{B}}$ do
                $p(v_j^{\mathrm{P}}, v_b^{\mathrm{Q}} | \mathcal{M}_t) \leftarrow p(v_j^{\mathrm{P}}, v_b^{\mathrm{Q}} | \mathcal{M}_t) +$
                $p(v_b^{\mathrm{Q}} | v_j^{\mathrm{P}}, \mathbf{m}_i, \mathcal{M}_t) p(v_j^{\mathrm{P}} | \mathbf{m}_i, \mathcal{M}_t) p(\mathbf{m}_i | \mathcal{M}_t)$;
            end
        end
    end
    ( Updated Active Graphs )
    $\mathcal{C}_{t+1} \leftarrow N_c$ best matches based on $p(V^{\mathrm{P}}, V^{\mathrm{Q}} | \mathcal{M}_t)$,
    which contains $\mathcal{M}_t$;
    $t \leftarrow t + 1$;
end
```

tic interpretation in the Bayesian manner. After the distribution of $p(V^{\mathrm{P}}, V^{\mathrm{Q}}|\mathcal{M}_t)$ is obtained via the probabilistic voting, the most frequently voted pairs in the joint space are selected as potential candidates for the next stage of graph matching. Non-maximal suppression is not necessary to this step because the purpose here is to collect the plausible candidates. The set of candidate matches $\mathcal{C}_{t+1}$ is assigned as N_c matches consisting of the matches in $\mathcal{M}_t$ and $N_c - |\mathcal{M}_t|$ best matches based on $p(V^{\mathrm{P}}, V^{\mathrm{Q}}|\mathcal{M}_t)$. The progressive framework is summarized in Algorithm 1. The detailed design of three terms in Eq. (3) depends on specific application domains.

3.2. Application to Image Matching

In this section, we deal with feature correspondence between two images, which is one of the most common and fundamental problems in computer vision. To construct scene graphs, the popular features of local (affine- or scale-) invariant regions such as SIFT [20], MSER [21], Harris-Affine [22] is adopted. We focus on the case of complex real-world images with more than hundreds or thousands of detected features. The graphs are constructed with the features as nodes and their geometric relations as edges. In this case, graph matching on *maximal graphs* is virtually intractable or impractical due to time and memory.

Graph Matching Given two images $\mathcal{I}_{\mathrm{P}}$ and $\mathcal{I}_{\mathrm{Q}}$, nodes $\mathcal{V}^{\mathrm{P}}$ and $\mathcal{V}^{\mathrm{Q}}$ represent features detected in $\mathcal{I}_{\mathrm{P}}$ and $\mathcal{I}_{\mathrm{Q}}$, respectively. Each feature usually has a unary appearance information of its local patch, and thus the initial candidate matches in Algorithm 1 can be collected by comparing their descriptors. This approach to *active graphs* is widely adopted in conventional graph matching [17, 4, 23, 7]. For a pairwise edge similarity $\mathbf{W}_{ia;jb} = \Omega_2(\mathbf{a}_{ij}^P, \mathbf{a}_{ab}^Q)$ of Eq. (2) between two matches (i, a) and (j, b), we adopt the Symmetric Transfer Error (STE) used in [12, 6, 7]. As depicted in Fig. 4, an affine region feature i centered on point $\mathbf{x}_i^P$ is represented by an elliptical region, and its orientation is estimated by a dominant orientation of the gradient histogram at the local region [20]. Using the characteristics, the affine homography transformation $\mathcal{T}_{ia}(\cdot)$ from feature i in P to another feature a in Q can be derived, so that the neighborhoods $\mathbf{x}^P$ and $\mathbf{x}^Q$ of two points $\mathbf{x}_i^P$ and $\mathbf{x}_a^Q$ are related by $\mathbf{x}^Q = \mathcal{T}_{ia}(\mathbf{x}^P)$ [22, 6]. Then, given two matches (i, a) and (j, b) as shown in Fig. 4, the transfer error of (j, b) with respect to (i, a) is denoted by $d_{jb|ia}$, and formulated as

$$d_{jb|ia} = \|\mathbf{x}_b^Q - \mathcal{T}_{ia}(\mathbf{x}_j^P)\|. \tag{4}$$

The better the homography $\mathbf{T}_{ia}$ transfers the center points of feature v_j to that of feature v_b, the smaller the value of $d_{jb|ia}$. Its symmetric version with respect to (i, a) is simply represented by $(d_{jb|ia} + d_{bj|ai})/2$. Based on this, the STE similarity measure of $\mathbf{W}_{ia;jb} = \Omega_2(\mathbf{a}_{ij}^P, \mathbf{a}_{ab}^Q)$ is defined as

$$\Omega_2(\mathbf{a}_{ij}^P, \mathbf{a}_{ab}^Q) = \max\left(0, \alpha - \frac{d_{jb|ia} + d_{bj|ai} + d_{ia|jb} + d_{ai|bj}}{4}\right), \tag{5}$$

which is invariant to the scale and affine changes of the images. Because the locally planar surfaces modeled by the affine regions approximate the deformable patterns well, the STE similarity provides a robust measure for deformation. As previously mentioned, any of the algorithms properly

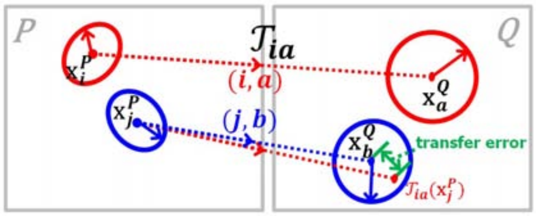
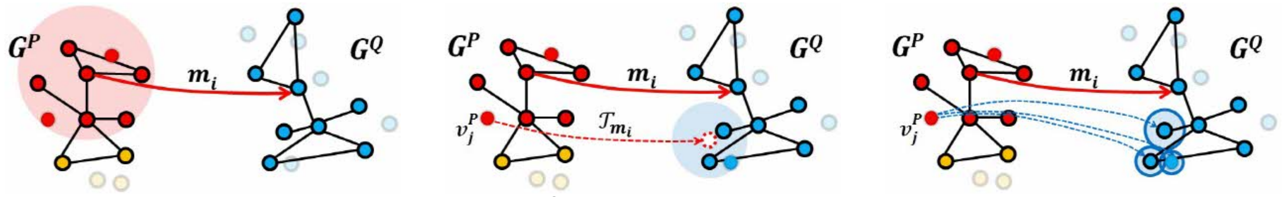


Figure 4. Affine feature matches and a transfer error. Given two matches (i, a) and (j, b), a transfer error of $\mathbf{x}_j^P$ with respect to (i, a) is computed as the length of the green arrow based on the homography transformation $\mathcal{T}_{ia}$ of match (i, a).



(a) $p(V^{\mathrm{P}}|M\!=\!\mathbf{m}_i, \mathcal{M}_t)$ (b) $p(V^{\mathrm{Q}}|V^{\mathrm{P}}\!=\!v_j^{\mathrm{P}}, M\!=\!\mathbf{m}_i, \mathcal{M}_t)$ (c) probabilistic votes to potential matches

Figure 3. Graph progression for feature correspondence. (a) k_1 nearest neighbors (red nodes) of $\mathbf{m}_i$ are selected in P. (b) k_2 nearest neighbors (blue nodes) of $\mathcal{T}_{\mathbf{m}_i}(v_j^{\mathrm{P}})$ are selected in P. (c) The probabilistic votes are given according to their geometric compatabilities with $\mathbf{m}_i$. In this illustration, $k_1 = 7$ and $k_2 = 3$.

maximizing the score of Eq. (2) can be adopted as the graph matching module.

Graph Progression The detailed design of three terms in Eq. (3) is explained in the following and illustrated in Fig. 3. The conditional prior $p(M=\mathbf{m}_i|\mathcal{M}_t)$ is measured by the relative confidence of each match in current graph matching:

$$p(M\!=\!\mathbf{m}_i|\mathcal{M}_t) = \mathrm{score}(\mathbf{m}_i)/\sum_i \mathrm{score}(\mathbf{m}_i). \qquad (6)$$

where $\mathrm{score}(\mathbf{m}_i)$ denotes the confidence score function of each match. If the graph matching method does not provide it, the prior becomes uniform: $p(M\!=\!\mathbf{m}_i|\mathcal{M}_t) = 1/|\mathcal{M}_t|$.

The probability of $p(V^{\mathrm{P}} = v_j^{\mathrm{P}}|M = \mathbf{m}_i, \mathcal{M}_t)$, implying the correlation of feature v_j^{P} to the achor match $\mathbf{m}_i$ among $\mathcal{M}_t$, is evaluated based on the proximity of the features. Denoting the k-nearest neighbor function by $\mathrm{kNN}(\cdot, k)$ and supposing $\mathbf{m}_i = (v_{p_i}^{\mathrm{P}}, v_{q_i}^{\mathrm{Q}})$, it is defined as

$$p(V^{\mathrm{P}}\!=\!v_j^{\mathrm{P}}|M\!=\!\mathbf{m}_i, \mathcal{M}_t) = \begin{cases} \frac{1}{k_1} & \text{if } v_j^{\mathrm{P}} \in \mathrm{kNN}(v_{p_i}^{\mathrm{P}}, k_1) \\ 0 & \text{otherwise} \end{cases}$$

which select k nearest nodes (features) in domain P from the anchor match $\mathbf{m}_i$.

The probability of $p(V^{\mathrm{Q}}\!=\!v_b^{\mathrm{Q}}|V^{\mathrm{P}}\!=\!v_j^{\mathrm{P}}, M\!=\!\mathbf{m}_i, \mathcal{M}_t)$ is evaluated using geometric properties of local affine features. Given feature v_j^{P} and match $\mathbf{m}_i$, the transfer error to v_b^{Q} is measured by $d_{jb|\mathbf{m}_i}$ using Eq. (4). Based on the transfer errors, we evaluate the potential matching probabilities of the neighboring features. Denoting the nearest neighbor feature as $\mathrm{NN}(\cdot)$, we define

$$p(V^{\mathrm{Q}}\!=\!v_b^{\mathrm{Q}}|V^{\mathrm{P}}\!=\!v_j^{\mathrm{P}}, M\!=\!\mathbf{m}_i, \mathcal{M}_t) = \begin{cases} 1 & \text{if } v_b^{\mathrm{Q}} = \mathrm{NN}(\mathcal{T}_{\mathbf{m}_i}(v_j^{\mathrm{P}})) \\ & \text{and } (v_j^{\mathrm{P}}, v_b^{\mathrm{Q}}) \in \mathcal{M}_t \\ \frac{\exp(-d_{jb|\mathbf{m}_i})}{Z} & \text{if } v_b^{\mathrm{Q}} \in \mathrm{kNN}(\mathcal{T}_{\mathbf{m}_i}(v_j^{\mathrm{P}}), k_2) \\ & \text{and} \big(v_j^{\mathrm{P}}, \mathrm{NN}(\mathcal{T}_{\mathbf{m}_i}(v_j^{\mathrm{P}}))\big) \notin \mathcal{M}_t \\ 0 & \text{otherwise} \end{cases}$$

where Z denotes a normalizing constant defined as $Z = \sum_{v_b^{\mathrm{Q}} \in \mathrm{kNN}(\mathcal{T}_{\mathbf{m}_i}(v_j^{\mathrm{P}}), k_2)} \exp(-d_{jb|\mathbf{m}_i})$. If the match of v_j^{P} to the nearest neighbor of $\mathcal{T}_{\mathbf{m}_i}(v_j^{\mathrm{P}})$ is already present in the current graph matching $\mathcal{M}_t$, the match $\big(v_j^{\mathrm{P}}, \mathrm{NN}(\mathcal{T}_{\mathbf{m}_i}(v_j^{\mathrm{P}}))\big)$ is considered highly reliable and receives all votes. Otherwise (i.e. if $\big(v^{\mathrm{P}}, \mathrm{NN}(\mathcal{T}_{\mathbf{m}_i}(v_j^{\mathrm{P}}))\big) \notin \mathcal{M}_t$), the k nearest neighbors of $\mathcal{T}_{\mathbf{m}_i}(v_j^{\mathrm{P}})$ obtain votes according to the transfer error.

By employing these formulations for graph progression, Algorithm 1 performs progressive image matching.

Complexity Using the approximate nearest neighbor (ANN) search [1], this graph progression is efficiently performed in $\mathcal{O}(k_1 k_2 |\mathcal{M}_t| \log(|\bar{\mathcal{V}}^P|) \log(|\bar{\mathcal{V}}^Q|))$ for each iteration. It mainly depends on the number of features (maximal nodes), but is much faster than graph matching in usual cases. Furthermore, by considering the common matches between $\mathcal{M}_t$ and $\mathcal{M}_{t-1}$, the computation can be reduced even greater at each iteration. As will be shown in the next section, more than 80% of performance increase is usually achieved in early five iterations in our experiments.

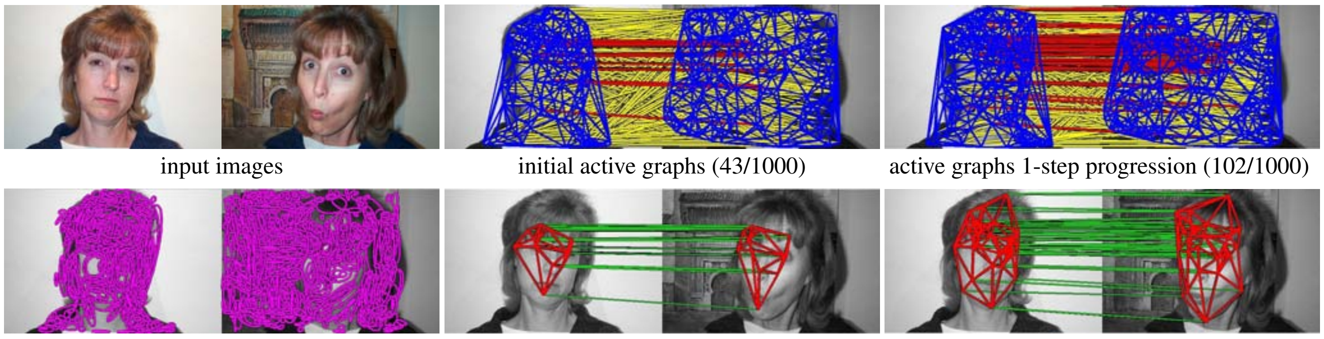
4. Experimental Evaluation

The progressive graph matching was applied to image matching problems based on the design of graph progression in Sec. 3.2. As its graph matching module, we tried several state-of-the-art methods [13, 17, 9, 24, 19, 7][2] and found that the reweighted random walk matching (RRWM) [7] performs best in the experiments. Therefore, we employed RRWM as the primary graph matching module and, when necessary, compare it with others. For the similarity matrix $\mathbf{W}_{ia;jb} = \Omega_2(\mathbf{a}_{ij}^P \mathbf{a}_{ab}^Q)$ of Eq. (5), we set $\alpha = 50$. The parameters of graph progression are fixed as $k_1 = 25$ and $k_2 = 5$.

4.1. Progressive vs. One-Shot Graph Matching

In this experiment, the proposed method is tested on image pairs with deformation and intra-category variation, and compared to those of conventional one-shot graph matching to observe the effect of the progressive framework. To construct initial active graphs, N_c best matches were collected based on the distance of the feature pair in the SIFT descriptor space [20], allowing multiple correspondences for

[2]All the codes were available at the authors' websites or on the request.



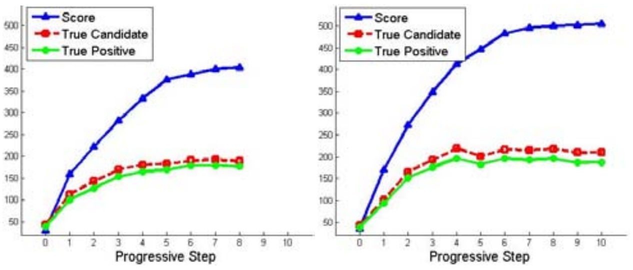
input images — initial active graphs (43/1000) — active graphs 1-step progression (102/1000)

detected features (MSER: 486 × 921) — one-shot graph matching (39/43) — 1-step progressive graph matching (94/102)

Figure 5. Boosting effects of progressive graph matching. At the top row, to visualize active graphs, the candidate matches are drawn by yellow (false) and red (true) lines, and their related features are represented by blue triangulations. At the bottom row, the true matches obtained by graph matching are shown with green lines with red triangulations. As the graph matching module, RRWM is used.

each feature; the number N_c remained the same in all graph progressions. To quantitatively evaluate the performance, the ground truth feature pairs were manually constructed for each image pairs[3].

Figure 5 compares the result of one-shot graph matching with that of its one-step progressive matching. Among possible 486 × 921 possible matches, $N_c = 1000$ candidate matches are used for active graphs. The top middle represents the initial active graphs while the top right shows the active graphs after graph progression based on one-shot graph matching at the bottom middle. The result clearly demonstrates the effect of progressive graph matching; graph progression significantly increases the number of true candidate matches in active graphs, thus boosting the performance in the subsequent graph matching. Despite the use of the same number of candidate matches and the same size of similarity matrix, the result shows a significant improvement even after one step of progression. The effect continues in the subsequent iterations as plotted in Fig. 6, where we compared two versions of progressive matching (SM and RRWM) on the example. In both of the plots, the number of true candidates in active graphs (red dashed line) increases with the progressive steps, and the number of true matches in graph matching (green line) follows it with a small gap. The scores (blue line) are monotonically increasing, and the boosting effect is more intensive in early steps. The progressive SM and RRWM achieved 337% and 425% growth rates in true matches, and 1388% and 1461% growth rates in score, respectively. The comparison reveals that the use of the better matching module provides faster convergence and higher performance; the RRWM version outperformed the SM version by 17% in true matches and by 22% in score. The similar improvement was consistently observed in all our experiments on various image pairs. Figure 7 shows some representative results on



(a) Progressive SM (b) Progressive RRWM

Figure 6. Performance growth on the image pair of Fig. 5. The plots show the graph matching score and the number of true matches w.r.t the increasing steps. Note that the step 0 denotes conventional one-shot graph matching.

the image pairs with deformation and intra-category variation. About two thousand features are extracted from the images, and multiple types of features (MSER [21], Harris-Affine, and Hessian-Affine [22]) are adopted in the second and third test pairs. Here, we set $N_c = 3000$ for active graphs. The results of Fig. 7 show the impressive improvement in both the true matches and their object coverage over those of one-shot graph matching. Furthermore, the results demonstrates that unlike patch-based match-growing methods [14, 12, 8] heavily dependent on object appearances, the proposed method by progressive graph matching is applicable to generic objects with intra-category variations. More results are available from our project site: http://cv.snu.ac.kr/research/~ProgGM/

4.2. Plug-in Comparison on a Benchmark Dataset

In this experiment, we performed quantitative comparison on a benchmark image dataset[4] used in [7]. The dataset consists of 30 image pairs containing photos of various objects most of which are collected from Caltech-

[3]The ground truths are labeled by hands on landmark points, and then extrapolated to neighboring features in the vicinity of 10∼20 pixels.

[4]The dataset with ground truths is available from http://cv.snu.ac.kr/research/~RRWM/

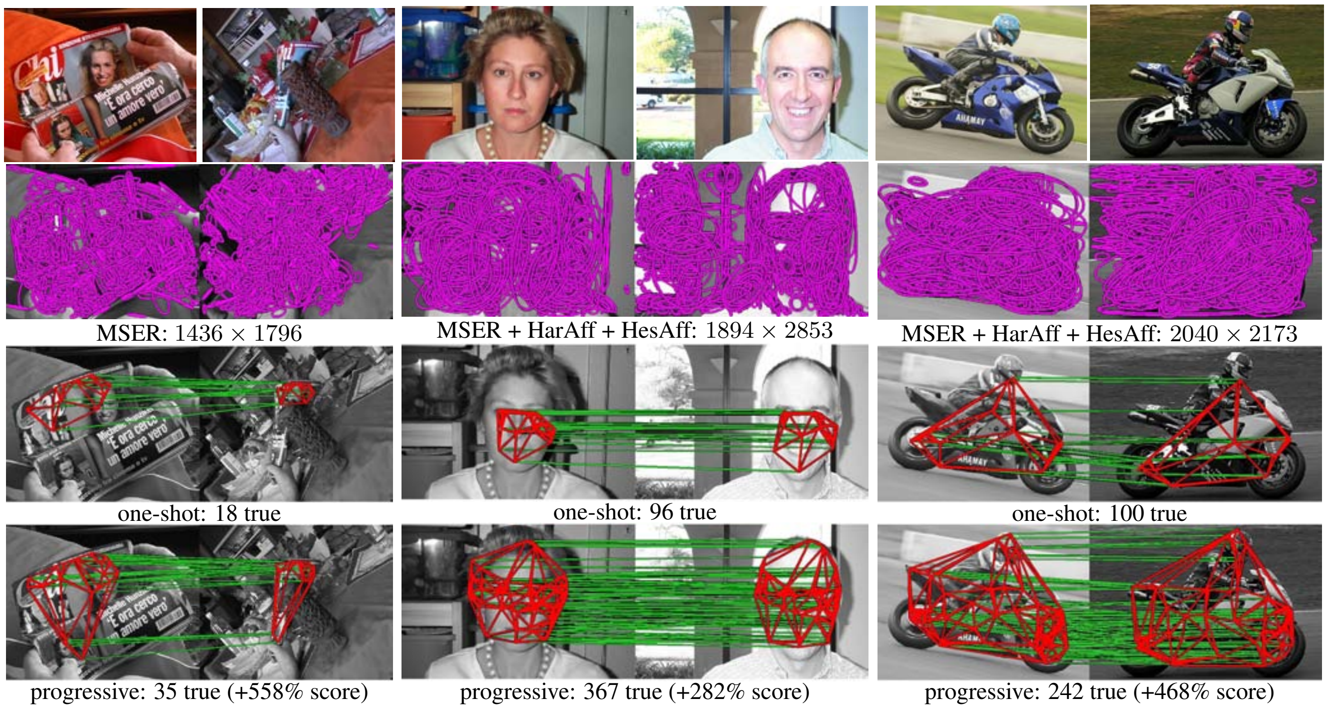

MSER: 1436×1796
MSER + HarAff + HesAff: 1894×2853
MSER + HarAff + HesAff: 2040×2173
one-shot: 18 true
one-shot: 96 true
one-shot: 100 true
progressive: 35 true (+558% score)
progressive: 367 true (+282% score)
progressive: 242 true (+468% score)

Figure 7. Progressive graph matching for feature correspondence. From the top row, input images, detected features, one-shot graph matching, and progressive graph matching are shown. In the matching results, the true positive matches are shown in green lines with red triangulations. At the bottom, the number of true positive matches and the growth rate of score are captioned.

101 and MSRC datasets. For each image pair, it provides detected MSER features, initial matches, and manually-labeled groundtruth feature pairs. We used the given initial matches for constructing initial active graphs, and applied the progressive graph matching on the dataset while employing as its graph matching module several state-of-the-arts of graph matching methods: SM, SMAC[9], PM[24], RRWM and IPFP[19]. In this setting, we intended to assess the performance increase over the conventional graph matching as well as the influence of the graph matching modules in use. The number of candidate matches in the progressive graph matching was always fixed to the same as the number of the given initial matches. Thus, the performance increase in this experiment is not due to the increase in the sizes of active graphs, but to the progressive improvement in the quality of active graphs. Based on the ground truths, we define the accuracy of matching as the number of true positive matches over the number of all true candidates matches contained in the current active graphs: (# of true positives) / (# of true candidates) [17, 9, 7].

Table 1. Performance on the benchmark dataset of 30 image pairs

	Graph Matching Module				
One-Shot	SM	SMAC	PM	RRWM	IPFP
Accuracy (%)	62.6	57.6	63.7	73.6	71.9
Progressive	SM	SMAC	PM	RRWM	IPFP
Accuracy (%)	68.2	63.6	66.7	81.2	78.2
Prog. vs. One-Shot	SM	SMAC	PM	RRWM	IPFP
Score Growth (%)	+65.0	+38.7	+92.1	+65.7	+63.8
Inlier Growth (%)	+59.6	+17.0	+85.1	+65.6	+69.7

The overall results are summarized in Table 1, and some examples are shown in Fig. 8. In this dataset, many of images have a small number of features (in some cases, even tens of features); thus, the progressive effects were less significant than those in the previous experiment. However, the consistent improvement is clearly observed for all graph matching modules; the progressive algorithms boost the accuracy by $3\% \sim 8\%$, the score by $39\% \sim 92\%$, and the number of inliers by $17\% \sim 85\%$. In general, the better algorithm plugged into our progressive framework, the better performance it obtains. For SM and RRWM, the average increase of accuracy and score at each progressive step is plotted in Fig. 9. Within five steps, each progressive algorithm rapidly reached to almost its maximum performance.

For more information, refer to our project site: `http://cv.snu.ac.kr/research/~ProgGM/`

5. Conclusion and Future Work

We introduced a progressive graph matching framework, which effectively resolves the limitations of conventional graph matching and achieves impressive performance in im-

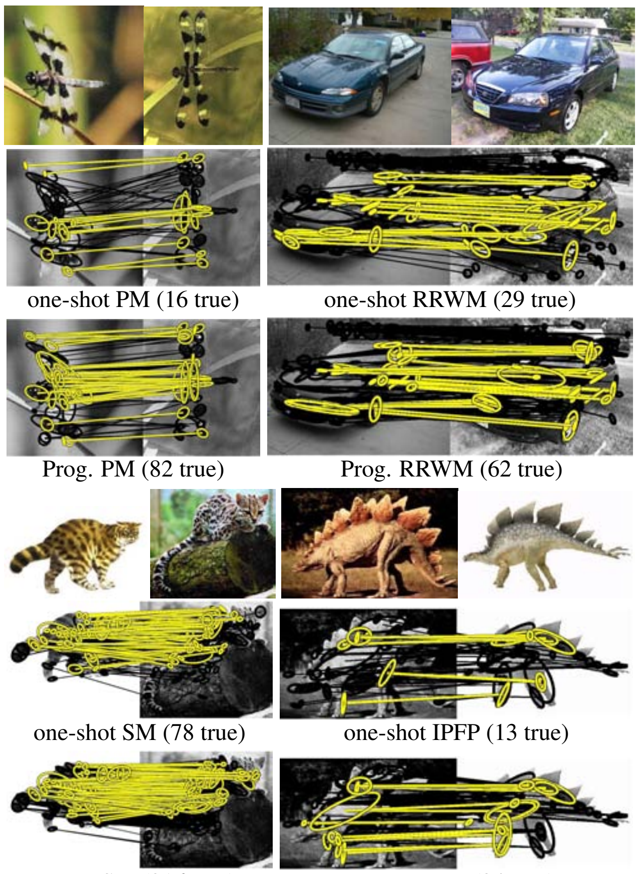
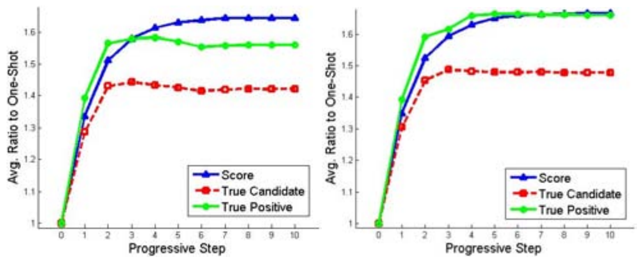


Figure 8. Example results on the benchmark dataset of 30 pairs.



(a) Progressive SM (b) Progressive RRWM

Figure 9. Average performance increase on the benchmark dataset of 30 pairs. The plots show the average growth rates of accuracies and scores obtained by the progressive algorithm over those by the conventional one-shot algorithm. Note that unlike Fig. 6, the plots show the average rates, not the absolute values.

age matching experiments. The proposed approach is orthogonal to specific graph matching modules and also extendable to general graph and hyper-graph matching problems. In the view of progressive graph matching as a move-making algorithm for graph matching, further related issues can be investigated for better graph progression in future, e.g. the interaction between multiple types of features, the incremental or hierarchical approach, and the optimal size of the evolving active graphs. We believe that the proposed framework will contribute to a wide range of graph-based approaches and applications, handling real-world problems with large data including significant clutter and outliers.

References

[1] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for approximate nearest neighbor searching. *ACM*, 1994. 5

[2] J. C. Avinash Sharma, Radu Horaud and E. Boyer. Topologically-robust 3d shape matching based on diffusion geometry and seed growing. *CVPR*, 2011. 1

[3] O. Barinova, V. Lempitsky, and P. Kohli. On detection of multiple object instances using hough transforms. *CVPR*, 2010. 3

[4] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. *CVPR*, 2005. 1, 2, 3, 4

[5] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *TPAMI*, 2001. 2

[6] M. Cho, J. Lee, and K. M. Lee. Feature correspondence and deformable object matching via agglomerative correspondence clustering. *ICCV*, 2009. 4

[7] M. Cho, J. Lee, and K. M. Lee. Reweighted random walks for graph matching. *ECCV*, 2010. 1, 2, 3, 4, 5, 6, 7

[8] M. Cho, Y. M. Shin, and K. M. Lee. Unsupervised detection and segmentation of identical objects. *CVPR*, 2010. 2, 6

[9] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. *NIPS*, 2007. 1, 2, 3, 5, 7

[10] O. Duchenne, F. Bach, I. Kweon, and J. Ponce. A tensor-based algorithm for high-order graph matching. *CVPR*, 2009. 1, 2, 3

[11] O. Duchenne, A. Joulin, and J. Ponce. A graph-matching kernel for object categorization. *ICCV*, 2011. 1

[12] V. Ferrari, T. Tuytelaars, and L. Gool. Simultaneous object recognition and segmentation from single or multiple model views. *IJCV*, 2006. 2, 4, 6

[13] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *TPAMI*, 1996. 3, 5

[14] J. Kannala, E. Rahtu, S. Brandt, and J. Heikkila. Object recognition and segmentation by non-rigid quasi-dense matching. *CVPR*, 2008. 2, 6

[15] J. Lee, M. Cho, and K. M. Lee. Hyper-graph matching via reweighted random walks. *CVPR*, 2011. 2, 3

[16] V. Lempitsky, C. Rother, S. Roth, and A. Blake. Fusion moves for markov random field optimization. *TPAMI*, 2009. 2

[17] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. *ICCV*, 2005. 1, 2, 3, 4, 5, 7

[18] M. Leordeanu, M. Hebert, and R. Sukthankar. Beyond local appearance: Category recognition from pairwise interactions of simple features. *CVPR*, 2007. 1, 2, 3

[19] M. Leordeanu and M. Herbert. An integer projected fixed point method for graph matching and map inference. *NIPS*, 2009. 2, 5, 7

[20] D. G. Lowe. Object recognition from local scale-invariant features. *ICCV*, 1999. 4, 5

[21] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. *BMVC*, 2002. 4, 6

[22] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *IJCV*, 2004. 4, 6

[23] L. Torresani, V. Kolmogorov, and C. Rother. Feature correspondence via graph matching: Models and global optimization. *ECCV*, 2008. 1, 2, 3, 4

[24] R. Zass and A. Shashua. Probabilistic graph and hypergraph matching. *CVPR*, 2008. 1, 2, 3, 5, 7