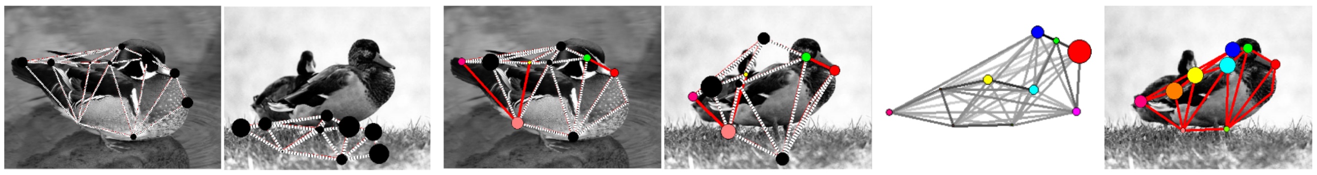
Learning Graphs to Match

Minsu Cho[1,*] Karteek Alahari[1,*] Jean Ponce[2,*]

[1]Inria [2]École Normale Supérieure



(a) graph matching without learning (b) with a learned matching function (c) a learned graph model and its matching

Figure 1: Graph learning for matching. Our approach learns a graph model from labeled data to provide the best match to instances of a target class. It shows significant improvement over previous approaches for matching. (Best viewed in color.)

Abstract

Many tasks in computer vision are formulated as graph matching problems. Despite the NP-hard nature of the problem, fast and accurate approximations have led to significant progress in a wide range of applications. Learning graph models from observed data, however, still remains a challenging issue. This paper presents an effective scheme to parameterize a graph model, and learn its structural attributes for visual object matching. For this, we propose a graph representation with histogram-based attributes, and optimize them to increase the matching accuracy. Experimental evaluations on synthetic and real image datasets demonstrate the effectiveness of our approach, and show significant improvement in matching accuracy over graphs with pre-defined structures.

1. Introduction

Graphs are widely used as a general and powerful representation in a variety of scientific fields, including computer vision, and many problems can be formulated as attributed graph matching. Since graph matching is mathematically expressed as a quadratic assignment problem, which is NP-hard, most research has long focused on developing accurate and efficient approximate algorithms [8, 14, 32]. Much progress has been achieved recently in various applications of graph matching, such as shape analysis [27], image matching [12, 30], action recognition [33], and object categorization [3, 13].

For many tasks, however, a natural question arises: How can we obtain a good graph model for a target object to match? Recent studies have revealed that simple graphs with hand-crafted structures and similarity functions, typically used in graph matching, are insufficient to capture the inherent structure underlying the problem at hand. As a consequence, a better optimization of the graph matching objective does not guarantee better correspondence accuracy [5, 6]. Previous learning methods for graph matching tackle this issue by learning a set of parameters in the objective function [5, 21, 26, 30]. Although it is useful to learn such *a matching function* for two graphs of a certain class, a more apt goal would be to learn *a graph model to match*, which provides an optimal matching to all instances of the class. Such a learned graph would better model the inherent structure in the target class, thus resulting in better performance for matching.

In this paper, we propose to learn a graph model based on a particular, yet rather general, graph representation with histogram-based attributes for nodes and edges. To this end, we present a generalized formulation for graph matching, which is an extension of previous learning approaches (Sec. 2). We show that all attributes of the graph can be learned in a max-margin framework [31] (Sec. 3). The proposed method reconstructs a graph model inherent in a target class, and provides impressive matching performance, as demonstrated in our experiments on synthetic and real datasets (Sec. 4). Figure 1 illustrates the effectiveness of the learning approach presented in the rest of this paper on an example. Here, the learned graph model finds better correspondences than a graph with a learned matching function as well as a hand-crafted graph.

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

The problem of learning graphical models has been addressed in other contexts. Several methods learn the parameters in clique functions defined on Markov random fields [25, 28]. Other works learn the connectivity structure of the Markov networks by introducing sparsity on clique functions [11, 17]. In the context of certain graph matching applications, an iterative method that alternates between estimating parameters and punning some of the nodes and edges has been proposed [4, 19]. Our approach differs from these methods in the sense that it learns attributes for all nodes and edges in a max-margin framework, not limited to global parameters in clique functions or sparse selection of clique functions. The learned attributes, combined with weight parameters, turn out to be critical for matching.

2. Graph matching revisited

We begin by reviewing the standard graph matching formulation and elaborate on methods to learn its parameters. In this context, we generalize the standard formulation and also highlight related work.

2.1. Problem formulation

The objective of graph matching is to find correspondences between two attributed graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ and $\mathcal{G}' = (\mathcal{V}', \mathcal{E}', \mathcal{A}')$, where $\mathcal{V}$ represents a set of nodes, $\mathcal{E}$ a set of edges, and $\mathcal{A}$ a set of attributes of the nodes and edges. A solution of graph matching is defined as a subset of possible correspondences $\mathcal{Y} \subset \mathcal{V} \times \mathcal{V}'$, represented by a binary assignment matrix $\mathbf{Y} \in \{0,1\}^{n \times n'}$, 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 $\mathbf{Y}_{i,a} = 1$, and $\mathbf{Y}_{i,a} = 0$ otherwise. We denote by $\mathbf{y} \in \{0,1\}^{nn'}$, a column-wise vectorized replica of $\mathbf{Y}$. With this notation, graph matching problems can be expressed as finding the assignment vector $\mathbf{y}^*$ that maximizes a score function $S(\mathcal{G}, \mathcal{G}', \mathbf{y})$ as follows:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} S(\mathcal{G}, \mathcal{G}', \mathbf{y}), \tag{1a}$$

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

where Eq. (1b) induces the matching constraints, thus making $\mathbf{y}$ an *assignment* vector [6, 9, 14, 18].

The score function $S(\mathcal{G}, \mathcal{G}', \mathbf{y})$ measures the similarity of graph attributes, and is typically decomposed into a first-order similarity function $\mathbf{s}_V(\mathbf{a}_i, \mathbf{a}'_a)$ for a node pair v_i in $\mathcal{V}$ and v'_a in $\mathcal{V}'$, and a second-order similarity function $\mathbf{s}_E(\mathbf{a}_{ij}, \mathbf{a}'_{ab})$ for an edge pair e_{ij} in $\mathcal{E}$ and e'_{ab} in $\mathcal{E}'$. Similarity functions are usually represented by a symmetric similarity (or affinity) matrix $\mathbf{A}$, where a non-diagonal element $\mathbf{A}_{ia;jb} = \mathbf{s}_E(\mathbf{a}_{ij}, \mathbf{a}'_{ab})$ contains the *edge similarity* of two correspondences (v_i, v'_a) and (v_j, v'_b), and a diagonal term $\mathbf{A}_{ia;ia} = \mathbf{s}_V(\mathbf{a}_i, \mathbf{a}'_a)$ represents the *node similarity* of a correspondence (v_i, v'_a). Thus, the score function of graph matching is defined as:

$$\begin{aligned} S(\mathcal{G}, \mathcal{G}', \mathbf{y}) &= \sum_{\mathbf{y}_{ia}=1} \mathbf{s}_V(\mathbf{a}_i, \mathbf{a}'_a) + \sum_{\substack{\mathbf{y}_{ia}=1 \\ \mathbf{y}_{jb}=1}} \mathbf{s}_E(\mathbf{a}_{ij}, \mathbf{a}'_{ab}) \\ &= \mathbf{y}^T \mathbf{A} \mathbf{y}. \end{aligned} \tag{2}$$

In essence, the score accumulates all the similarity values relevant to the assignment.

The formulation in Eq. (1) is referred to as an integer quadratic programming. More precisely, it is the quadratic assignment problem, which is known to be NP-hard. Due to its generality and flexibility, this formulation has been favored in recent graph matching research. Many efficient approximate algorithms have been proposed for the formulation [6, 9, 14, 20, 30] and its extensions [7, 12].

2.2. Learning parameters

In the context of scoring functions defined in Eq. (2), an interesting question is what can be learned to improve graph matching. To address this, we parameterize Eq. (2) as follows. Let $\pi(i) = a$ denote an assignment of node v_i in $\mathcal{G}$ to node v'_a in $\mathcal{G}'$, *i.e.* $\mathbf{y}_{ia} = 1$. A joint feature map $\Phi(\mathcal{G}, \mathcal{G}', \mathbf{y})$ is defined by aligning the relevant similarity values of Eq. (2) into a vectorial form as:

$$\begin{aligned} &\Phi(\mathcal{G}, \mathcal{G}', \mathbf{y}) \\ &= [\cdots; \mathbf{s}_V(\mathbf{a}_i, \mathbf{a}'_{\pi(i)}); \cdots; \mathbf{s}_E(\mathbf{a}_{ij}, \mathbf{a}'_{\pi(i)\pi(j)}); \cdots]. \end{aligned} \tag{3}$$

By introducing weights on all elements of this feature map, we obtain a discriminative score function:

$$S(\mathcal{G}, \mathcal{G}', \mathbf{y}; \beta) = \beta \cdot \Phi(\mathcal{G}, \mathcal{G}', \mathbf{y}), \tag{4}$$

where β is a weight vector encoding the importance of node and edge similarity values. In the case of uniform weights, *i.e.* $\beta = \mathbf{1}$, it reduces to the conventional graph matching score function of Eq. (2): $S(\mathcal{G}, \mathcal{G}', \mathbf{y}) = S(\mathcal{G}, \mathcal{G}', \mathbf{y}; \mathbf{1})$. We refer to it as the *discriminative weight* formulation for graph matching. Note that the similarity functions $\mathbf{s}_V$ and $\mathbf{s}_E$ can take not only scalar-valued, but also vector-valued functions. Despite its apparent simplicity, this formulation covers a wide range of parameter learning approaches proposed for graph matching. They are equivalent to special cases of this formulation, which restrict β, and assign the same vector for all $\mathbf{s}_V(\mathbf{a}_i, \mathbf{a}'_a)$ and $\mathbf{s}_E(\mathbf{a}_{ij}, \mathbf{a}'_{ab})$. Caetano *et al.* [5] use a 60-dimensional similarity function $\mathbf{s}_V$ for appearance similarity and a simple binary similarity $\mathbf{s}_E$ for edges. Leordeanu *et al.* [21] do not use $\mathbf{s}_V$, and instead employ a multi-dimensional $\mathbf{s}_E$ for similarity of appearance, angle, and distance. The work of Torresani *et al.* [30] can be viewed as adopting 2-dimensional $\mathbf{s}_V$ and $\mathbf{s}_E$ functions for measuring appearance similarity, geometric compatibility, and occlusion likelihood. While the optimization methods for learning these functions are different, all of them are essentially aimed at learning common weights for all the edge and node similarity functions. The discriminative weight formulation is more general in the sense that it can assign different parameters for individual nodes and edges. We will highlight this advantage with appropriate evidence in the experimental section. However, like previous approaches, it does not learn a graph model underlying the feature map Φ, and requires a reference graph $\mathcal{G}'$ at query time, whose attributes cannot be modified in the learning phase. We overcome this drawback by proposing an optimization framework to learn the reference graph.

3. Graph learning

To address the problem of learning graphs, we start with the discriminative weight formulation of Eq. (4). Instead of a reference graph used in the previous section, we consider a class-specific model graph $\mathcal{G}^*$. Our aim is to infer this graph, such that it produces the best matches with other instances of the class. Let $\hat{\mathbf{y}}$ denote the optimal matching between the model graph $\mathcal{G}^*$ and an input graph $\mathcal{G}$, given by:

$$\hat{\mathbf{y}}(\mathcal{G};\mathcal{G}^*,\beta) = \arg\max_{\mathbf{y}\in\mathcal{Y}(\mathcal{G})} S(\mathcal{G}^*,\mathcal{G},\mathbf{y};\beta), \tag{5}$$

where β is a weight vector, $\mathcal{Y}(\mathcal{G})$ defines the set of possible assignment vectors for the input graph $\mathcal{G}$. Inspired by the max-margin framework [31], we learn the model graph $\mathcal{G}^*$ and its weights β from labeled examples $D = (\langle\mathcal{G}_1,\mathbf{y}_1\rangle,\dots,\langle\mathcal{G}_n,\mathbf{y}_n\rangle)$, where $\mathbf{y}_i \in \mathcal{Y}(\mathcal{G}_i)$, by minimizing the following objective function:

$$L_D(\mathcal{G}^*,\beta) = r(\mathcal{G}^*,\beta) + \frac{C}{n}\sum_{i=1}^{n}\Delta(\mathbf{y}_i,\hat{\mathbf{y}}(\mathcal{G}_i;\mathcal{G}^*,\beta)), \tag{6}$$

In this objective function r is a regularization function, Δ a loss function, and $\mathbf{y}_i$ denotes the ground truth assignment vector. The parameter C controls the relative importance of the loss term.

A critical question to address here is how to parametrize $\mathcal{G}^*$ and β such that the optimization of Eq. (6) is feasible. We propose to parameterize both $\mathcal{G}^*$ and β in a vectorial form, which then enables us to optimize the objective function in Eq. (6) efficiently, as shown below. We first separate the graph model $\mathcal{G}^*$ from the joint feature map $\Phi(\mathcal{G}^*,\mathcal{G},\mathbf{y})$, so as to parameterize it separately. We assume that the similarity functions $\mathbf{s}_V$ and $\mathbf{s}_E$ are dot products of two attribute vectors:

$$\mathbf{s}_V(\mathbf{a}^*_i,\mathbf{a}_a) = \mathbf{a}^*_i\cdot\mathbf{a}_a, \quad \mathbf{s}_E(\mathbf{a}^*_{ij},\mathbf{a}_{ab}) = \mathbf{a}^*_{ij}\cdot\mathbf{a}_{ab}, \tag{7}$$

where $\mathbf{a}^*_i$ and $\mathbf{a}^*_{ij}$ correspond to the node and edge attributes of the model graph respectively. Further, we define the attribute vector $\Theta(\mathcal{G}^*)$ and the feature map $\Psi(\mathcal{G},\mathbf{y})$ as:

$$\Theta(\mathcal{G}^*) = [\cdots;\mathbf{a}^*_i;\cdots;\mathbf{a}^*_{ij};\cdots], \tag{8}$$
$$\Psi(\mathcal{G},\mathbf{y}) = [\cdots;\mathbf{a}_{\pi(i)};\cdots;\mathbf{a}_{\pi(i)\pi(j)};\cdots], \tag{9}$$

where $\Theta(\mathcal{G}^*)$ describes all the attributes of $\mathcal{G}^*$ and $\Psi(\mathcal{G},\mathbf{y})$ represents the corresponding attributes of $\mathcal{G}$, according to the assignment $\mathbf{y}$. This enables the attributes of $\Phi(\mathcal{G}^*,\mathcal{G},\mathbf{y})$ to be factorized into $\Theta(\mathcal{G}^*)$ and $\Psi(\mathcal{G},\mathbf{y})$, and thus to rewrite the score function as:

$$\begin{aligned} S(\mathcal{G}^*,\mathcal{G},\mathbf{y};\beta) &= \beta\cdot\Phi(\mathcal{G}^*,\mathcal{G},\mathbf{y}) \\ &= \beta\cdot(\Theta(\mathcal{G}^*)\odot\Psi(\mathcal{G},\mathbf{y})) \\ &= (\beta\odot\Theta(\mathcal{G}^*))\cdot\Psi(\mathcal{G},\mathbf{y}), \end{aligned} \tag{10}$$

where $\odot$ denotes the Hadamard (element-wise) product. Note that both terms of the attributes $\Theta(\mathcal{G}^*)$ and their weights β are now combined into a single vector $(\beta\odot\Theta(\mathcal{G}^*))$. Thus, when the similarity functions are dot products, both the graph model attributes and their weights can be jointly expressed by a single vector. By substituting $\mathbf{w} = \beta\odot\Theta(\mathcal{G}^*)$ into Eq. (5), we obtain a linear form for the optimal assignment:

$$\hat{\mathbf{y}}(\mathcal{G};\mathbf{w}) = \arg\max_{\mathbf{y}\in\mathcal{Y}(\mathcal{G})} \mathbf{w}\cdot\Psi(\mathcal{G},\mathbf{y}). \tag{11}$$

In turn, this transforms the learning objective in Eq. (6) into a standard formulation of the structured support vector machine (SSVM):

$$L_D(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{n}\sum_{i=1}^{n}\Delta(\mathbf{y}_i,\hat{\mathbf{y}}(\mathcal{G}_i;\mathbf{w})), \tag{12}$$

where all the graph model attributes and their weights, to be learned, are represented by $\mathbf{w}$. This function can be minimized by various optimization approaches to estimate the parameters $\mathbf{w}$ [16, 29, 31]. Unlike other learning methods for graph matching [5, 21, 26, 30], this formulation allows us to combine graph learning, and learning a matching function into a coherent structured output framework. A related approach has been proposed to learn homography estimation in keypoint matching and tracking [15].

3.1. Histogram-attributed relational graph

In general, any graph representation satisfying the condition of dot product similarity of Eq. (7), which leads to the linearization in Eq. (10), can be learned with our approach. However, not all potential representations are effective in representing the data in the context of graph learning and matching performance. In this work, we propose a new *histogram-attributed relational graph* (HARG), wherein all node and edge attributes are represented by histogram distributions. The similarity value between two attributes in this graph is then computed as their dot product. The histogram attributes in this framework can be composed of a variety of features. In this work we chose to build them using angle and length for edge attributes, and local appearance for node attributes.

The *histogram of log-polar bins* edge attribute describes the geometric relationship between two interest points as illustrated in Fig. 2. As widely done in computer vision [22, 23], we assume that each interest point can be assigned a characteristic scale and orientation.[1] Consider an edge e_{ij} from node v_i (represented by point $\mathbf{x}_i$ in Fig. 2) to node v_j ($\mathbf{x}_j$ in the figure). The vector from $\mathbf{x}_i$ to $\mathbf{x}_j$ can be expressed in polar coordinates as (ρ_{ij}, θ_{ij}). We transform this into a histogram-based attribute, which is invariant to the characteristic scale and orientation of v_i. Two histograms – one for length and another for angle – are built and concatenated to quantize the edge vectors. For length, we use uniform bins of size n_L in the log space with respect to the position and scale of v_i, making the histogram more sensitive to the position of nearby points. The log-distance histogram L_{ij} is constructed on the bins by a discrete Gaussian histogram centered on the bin for ρ_{ij}:

$$\begin{aligned} & L_{ij}(k) = f_L(k-m), \\ \text{s.t.} \quad & f_L(x) = \mathcal{N}(0, \sigma_L)\,,\ \rho_{ij} \in \text{bin}_\rho(m), \end{aligned} \tag{13}$$

where $\mathcal{N}(\mu, \sigma)$ represents a discrete Gaussian window[2] of size σ centered on μ, and $\text{bin}_\rho(k)$ denotes the kth log-distance bin from the center of v_i. For angle, we use uniform bins of size $2\pi/n_P$. The polar-angle histogram P_{ij} is constructed on it in a similar way, except that a circular Gaussian histogram centered on the bin for θ_{ij} with respect to the characteristic orientation of v_i, is used:

$$\begin{aligned} & P_{ij}(k) = f_P(k-m), \\ \text{s.t.} \quad & f_P(x) = \mathcal{N}(0, \sigma_P) + \mathcal{N}(\pm n_P, \sigma_P)\,,\ \theta_{ij} \in \text{bin}_\theta(m), \end{aligned} \tag{14}$$

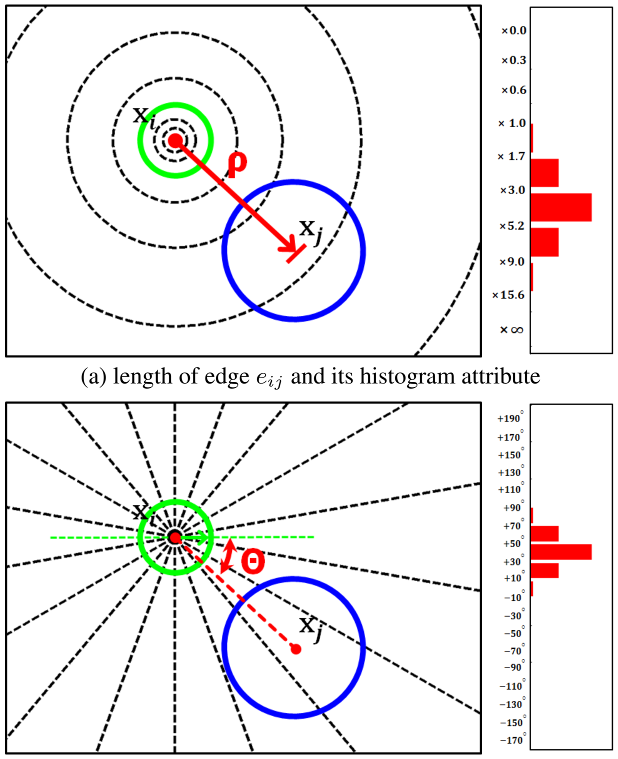
where additional Gaussian terms in $f_P(x)$ induce the circular bins for angle. The final histogram composed by concatenating the log-distance L_{ij}, and the polar-angle P_{ij}, histograms is defined as the attribute for edge e_{ij}: $\mathbf{a}_{ij} = [L_{ij}; P_{ij}]$, which is asymmetric ($\mathbf{a}_{ij} \neq \mathbf{a}_{ji}$).[3] In this work, we used $n_L = 9$, $n_P = 18$. Our representation has several advantages for visual matching. When used with local invariant features, it becomes geometrically invariant in scale and orientation. From the viewpoint of learning, notably, the nonparametric nature of histograms allows us to represent multi-modal distributions of distance and angle through the learning process.

For node attributes $\mathbf{a}_i$, describing the local appearance of node v_i, we could adopt the histogram of gradient bins such as SIFT [22], HOG [10], and their variants, given their effectiveness. In our experiments, we used the SIFT descriptor.

[1]When the interest points do not have characteristic scales and orientations, we fix them to 1 and 0 respectively.

[2]We used a window size $\sigma_L = \sigma_P = 5$ so that $\mathcal{N}(0,5) = 1.0$, $\mathcal{N}(\pm 1, 5) = 0.4578$, $\mathcal{N}(\pm 2, 5) = 0.0439$, and 0 otherwise.

[3] In the context of feature descriptors, shape-context [2] uses a histogram to represent the distribution of points in a two-dimensional log-polar space. In contrast, our histogram for edge attributes consists of two separate log-distance and polar-angle ones.



(a) length of edge e_{ij} and its histogram attribute

(b) angle of edge e_{ij} and its histogram attribute

Figure 2: Histogram of log-polar bins for edge attributes. This attribute is a concatenation of log-distance and polar-angle histograms. Each histogram is represented by a discrete Gaussian window centered at a bin. (a) Log-distance ρ_{ij} (left) and its histogram with 9 bins (right). The log-distance ρ_{ij} of edge e_{ij} is measured relative to the scale of v_i. (b) Polar-angle θ_{ij} (left) and its histogram with 18 bins (right). The polar-angle θ_{ij} of edge e_{ij} is measured from the characteristic orientation of v_i, or from the horizontal line through v_i (shown as a green line), when there is no such orientation.

3.2. Loss functions

Another ingredient in the objective function Eq. (12) is the loss function $\Delta(\mathbf{y}, \hat{\mathbf{y}})$. It drives the learning process by measuring the quality of a predicted matching $\hat{\mathbf{y}}$ against its ground truth $\mathbf{y}$. We use the normalized Hamming loss, similar to [5], which is the fraction of mismatches between assignment vectors $\mathbf{y}$ and $\hat{\mathbf{y}}$,

$$\Delta(\mathbf{y}, \hat{\mathbf{y}}) = 1 - \frac{1}{||\mathbf{y}||_F^2} \mathbf{y} \cdot \hat{\mathbf{y}}, \tag{15}$$

where $||\cdot||_F$ is the Frobenius norm.

3.3. Optimization

Many approaches have been proposed to train SSVMs [16, 29, 31]. This problem amounts to solving a convex quadratic program with an exponentially large number of constraints. Solutions for this optimization problem either: (i) reduce it to an equivalent polynomial-size reformulation (for certain decomposable loss functions), and use methods like SMO [29] or general-purpose solvers; or (ii) work with the original problem by considering a subset of constraints, and employing cutting plane [31] or stochastic subgradient methods. For solving the problem in Eq. (12), we use the efficient cutting plane method proposed by Joachims *et al.* [16]. This method differs from most other SVM training approaches by considering individual data points as well as their linear combinations as potential support vectors. This leads to a smaller set of cutting plane models, and thus more efficient training.

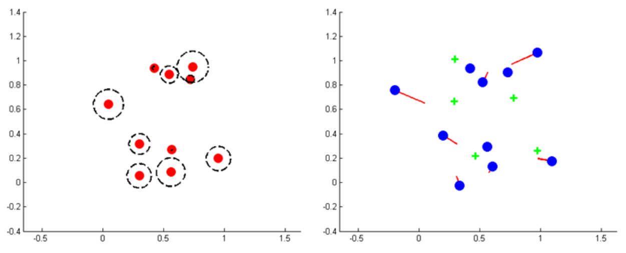
4. Experimental evaluation

In this section we conduct comparative evaluations on synthetic and real data. We observed that our histogram-based similarity function showed better or comparable matching performance than other similarity measures used in [6, 9, 21]. Hence, we chose to focus on the performance of learning using these attributes in all our experiments. A fully-connected graph is used as the initial graph for learning. We evaluate four methods: 'w/o learning', 'SW-SSVM', 'SW-SPEC', 'DW-SSVM', and our method 'HARG-SSVM'. For w/o learning, we use a conventional graph matching method with uniform weighting. For SW-SSVM, SSVM learning is applied to shared weights on nodes and edges. Here, we learn 2 (angle and distance) parameters for edges, and 128 (SIFT) parameters for nodes. Although the similarity functions and the optimization algorithm are different, SW-SSVM is closely related to the method of [5]. SW-SPEC is the learning method of [21] for shared weights. DW-SSVM represents a discriminative weight learning approach based on the formulation discussed in Sec. 2.2, which learns individual weights for nodes and edges (2 parameters for each edge and 128 parameters for each node). HARG-SSVM is our graph learning approach proposed in Sec. 3. The SSVM objective is optimized with the same method [16] in all the experiments.

It should be noted that the methods [5, 21] were originally proposed to learn the weights of a graph matching function for two graphs in the same class. Our approach (HARG-SSVM), on the other hand, learns the graph model as well. The approaches are evaluated on three datasets, including a synthetic dataset, the CMU House/Hotel sequences, and an object class dataset.

4.1. Synthetic point sets

The goal of this experiment is to evaluate and compare the performance in a controlled setting. Here, we build on the widely used point set matching problem protocol [9, 18]. We define a source set $\mathcal{P}$ by n Gaussian distributions, each of which is centered at a random point in the 2-dimensional


(a) a source set and its sample with noise and outliers

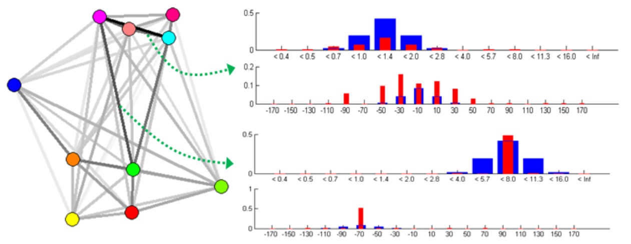

(b) a learned graph model from 100 point set samples

Figure 3: Graph learning from synthetic point sets. (a) A source set of points is defined by 10 Gaussian distributions, each centered at a random point (red dots in the left image) and has a random variance (denoted by a black circle around each red dot). A sample point set is obtained by generating one point from each distribution (blue dots in the right image) and adding 5 random outliers (shown as green crosses). (b) We learn angle and distance attributes from these samples. Darker lines represent edges learned as being more important than others. Two examples of edge attributes are shown on the right, where the upper histogram represents distance and the lower histogram describes angle. The learned attributes (red) not only recover the attributes of the source (blue), but also adjust its weights and variance. The first example (the edge on the top) has lower weights and a larger variance than the second example (the edge in the middle) because the two nodes in the former are closer each other and have larger variances in their position. (Best viewed in color.)

domain $[0,1]^2$, with a random variance in $[0, 0.15]$. As shown in Fig. 3(a), an observable sample from the source set $\mathcal{P}$ consists of n inlier points, which come from $\mathcal{N}(p_i, \sigma_i)$, and n_o random outliers, generated from a uniform distribution in $[0,1]^2$. Visually, this simulates deformation and clutter in the observations. We consider 100 sample point sets from the source set, where each point in the sample set has an assignment label to one of the n distributions in $\mathcal{P}$. Our task is to assign labels to new samples with graph matching. This problem setting resembles many real-world applications, and is equivalent to experiments in previous works [5, 21]. Since there is no unary information in the points, graph matching in this case relies solely on pairwise similarity. From each point set we construct a graph with our histogram-based attributes.

Table 1: Performance on synthetic point sets. Several learning approaches (shown in each row) are evaluated with the state-of-the-art graph matching algorithms (in columns). See text for details of the learning methods. ‘Sample’ or ‘Source’ refer to the type of reference graphs used for learning and matching. Given that in real problems it is unlikely to get observations without deformations and noise, the result with ‘Source’ corresponds to the upper bound for the methods we compare to. In contrast, our graph learning approach (HARG-SSVM) does not require such a reference graph, and consistently outperforms all the other learning approaches, including those with the source reference graphs. The error is measured by the average distance from true matching points.

		SM [18]		GAGM [14]		IPFP [20]		RRWM [6]	
Reference	Methods	accuracy (%)	error	accuracy (%)	error	accuracy (%)	error	accuracy (%)	error
Sample	w/o learning	60.4	0.079	62.4	0.070	61.4	0.067	62.2	0.072
	SW-SSVM	59.7	0.073	53.9	0.114	61.2	0.070	67.5	0.057
	SW-SPEC	62.2	0.068	58.8	0.103	64.3	0.054	66.2	0.063
	DW-SSVM	66.2	0.054	70.4	0.050	69.0	0.053	75.6	0.040
Source	w/o learning	70.0	0.044	76.1	0.034	75.9	0.033	75.4	0.037
	SW-SSVM	70.9	0.042	68.1	0.061	70.8	0.043	79.0	0.030
	SW-SPEC	71.8	0.040	68.6	0.053	69.1	0.049	76.0	0.037
	DW-SSVM	73.9	0.035	77.7	0.030	77.0	0.030	78.6	0.032
HARG-SSVM		**79.7**	**0.026**	**79.5**	**0.027**	**79.7**	**0.029**	**81.7**	**0.025**

For our method (HARG-SSVM), we directly learn the model and use it to match with a test set. For all the other methods, since they do not learn a model, we take a reference point set, and use it to match with the test set. Here, two types of references are used: ‘Sample’ or ‘Source’. For ‘Sample’, we randomly select one of the training sample point sets, and for ‘Source’, the points from the source set are directly used. In other words, ‘source’ corresponds to an ideal reference graph without deformations and noise, *i.e.*, a graph formed by red dots on the left image of Fig. 3(a).

We performed learning with 10 inliers ($n = 10$), 5 outliers ($n_o = 5$), and maximum variance of 0.15. A comparison of matching performance on 1000 test samples from 10 source sets is shown in Table 1. We also use different graph matching algorithms to account for dependency on the matching algorithm used. Note that our method HARG-SSVM consistently outperforms all the other methods, when a sample set is used as the reference graph. Furthermore, HARG-SSVM provides better results even when other methods use the source set as the reference graph. This is because our graph model additionally captures both the variation and the importance in edge attributes as shown in Fig. 3(b). The two methods, SW-SSVM and SW-SPEC, do not show a notable improvement compared to w/o learning because of the limitations of using shared weights. Given the fact that in realistic situations we cannot directly access the source of information, and can only sample from the source, this synthetic experiment suggests that our learning approach leads to improvement in such problems.

4.2. House/Hotel dataset

The CMU House/Hotel sequence is one of the most popular benchmark datasets for graph matching. We used the image sequences (House: 111 images and Hotel: 101 images) and the feature points from [5, 21]. In one of their experiments, they used 5 training images for learning, and tested on all the remaining pairs. Here, we learn a graph model using only 3 training images (#0, #50, #100) from the 5 images they used, and match it to all the other test images. Unlike [5, 21], we only rely on the edge attributes without any appearance descriptor. As shown in Table 2, learned HARG achieves perfect results without any mismatch, outperforming all the other methods. Figure 5 shows the learned graphs and some example results of matching to other frames. For images of identical objects, such as these House/Hotel sequences, only a few number of images are sufficient for our method to learn the model graph. In Table 2, we also compared with other learning approaches (SW-SSVM, DW-SSVM) using a reference graph (#0). In this case, DW-SSVM also provides near-perfect results, which suggests that learning individual parameters of edges is important for matching.

4.3. Object class datasets

We tested our method on learning graphs of visual object classes. This experiment is different from the previous ones for two reasons: First, the image contains not only background clutter but also significant intra-category, and view-point changes. Second, contrary to many related works [5, 21, 26, 30], we use a local feature detector without any additional manual pruning on cluttered images. In this case, a significant number of outliers exists, and thus graph matching becomes challenging.

We use the scale-invariant Hessian detector [24] to detect local regions as nodes. We then describe the edge and node attributes, as detailed in Sec. 3.1. Given a test image, we select kNN features ($k = 50$) for each node of the model graph based on dot product similarity, and construct

Table 2: Matching performance comparison on the CMU House/Hotel sequences. The frame #0 was used as a reference graph in w/o learning, SW-SSVM, and DW-SSVM. The numbers in parentheses denote the number of training images used for each method. In the last three rows, we also report the published results for [5, 21] on 5 and 106 training images. Note that while we learn a graph model and match it to all the other images, they learned the parameters for matching, and applied them to match all possible pairs among all the other images.

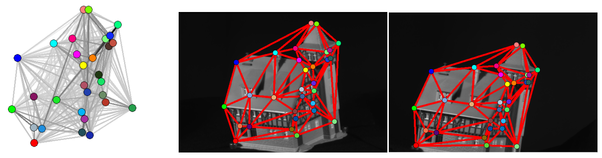
	House		Hotel	
Method	Acc.(%)	Err.(px)	Acc.(%)	Err.(px)
w/o learning	99.6	0.06	78.7	11.32
SW-SSVM (3)	99.6	0.06	94.6	3.02
DW-SSVM (3)	99.8	0.01	**100.0**	**0.00**
HARG-SSVM (3)	**100.0**	**0.000**	**100.0**	**0.00**
SW-SPEC [21] (5)	99.8	n/a	94.8	n/a
SW-SSVM [5] (5)	< 84	n/a	< 87	n/a
SW-SSVM [5] (106)	< 96	n/a	< 90	n/a

a fully-connected graph. The error measure is defined as the distance between a detected point and its ground truth, normalized by the maximum distance between ground truth points in the image, *i.e.* an estimate of the size of the target object. Since we use a feature detector, it is virtually impossible to make an exact correspondence for each point, due to localization errors. Hence, we define a margin for error t_e, and consider a correspondence to be correct if its error is less than t_e. We set $t_e = 0.15$ in this experiment.

We constructed an object class dataset with images from Caltech-256 (Face, Duck, and Wine bottle) and PASCAL VOC2007 (Motorbike and Car) datasets. The images of these classes are selected such that each class contains at least 40 images with different instances.[4] Some of these were then reflected laterally to ensure that each object has a consistent orientation. We manually selected 10 distinctive points on the target object, and identified them in each image. In order to ease the labelling process, we used the following scheme. We first take an image, identify 10 points, and set the image as a reference for the class. Then, we apply graph matching to find correspondence from the reference to the other images, and refine them by fixing incorrect matches. This scheme allows us to build the ground truth data more easily than other methods for the same purpose [5, 21].

We split the image set for each object class into two random partitions (20 images for training and the rest for testing), and report accuracy and error of matching, averaged over the 20 random splits. Table 3 summarizes the results and Figure 4 shows the learned model graphs and their matching results. Our approach (HARG-SSVM) shows significantly better results than other approaches. It suggests that the detailed attributes learned by our approach enable more accurate matching to other instances in the same class. This experiment demonstrates that our approach successfully handles the practical challenges in matching by constructing a robust graph model. For more results, refer to our project website [1].

[4]The dataset consists of 109 Face, 50 Duck, 66 Wine bottle, 40 Motorbike, and 40 Car images, and is available from our project website [1].


(a) a learned House graph and its matching examples

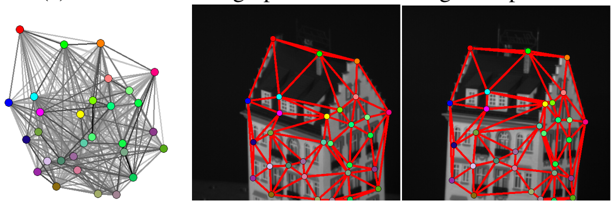

(b) a learned Hotel graph and its matching examples

Figure 5: Results on the CMU House/Hotel sequence. For each sequence, a graph model is learned using 3 images, and tested on all the other images (108 for House, 98 for Hotel). From left to right, the learned graph and two matching examples are shown. In the graph models, darker lines denote edges with higher weights. For comparison with other methods, see Table 2. (Best viewed in color.)

5. Conclusion

We presented a novel graph learning approach with a histogram-based representation and an SSVM framework. In synthetic and real data experiments, we demonstrated that the proposed method effectively learns an inherent model graph from a training set, and provides good generalization to unseen instances for matching. In future work, we plan to explore sparse graph representation for better efficiency.

Acknowledgments. The authors would like to thank Olivier Duchenne and Francis Bach for helpful discussions, and Marius Leordeanu for providing his code and data. This research was supported in part by the ERC advanced grant VideoWorld, Institut Universitaire de France, and the Quaero programme funded by the OSEO.

References

[1] http://www.di.ens.fr/willow/research/graphlearning/.
[2] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *TPAMI*, 2002.
[3] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *CVPR*, 2005.
[4] W. Brendel and S. Todorovic. Learning spatiotemporal graphs of human activities. In *ICCV*, pages 778–785, 2011.

Table 3: Matching performance on 5 object classes from Caltech-256 and PASCAL VOC2007 datasets. For each class, the average performance on 20 random splits of the data is reported. Our method (HARG-SSVM) shows the best matching results on all the 5 classes. For more details, see text and our project website [1].

	Face		Motorbike		Car		Duck		Wine bottle	
Method	Acc. (%)	error	Acc. (%)	error	Acc. (%)	error	Acc. (%)	error	Acc. (%)	error
w/o learning	66.6	0.205	44.1	0.226	34.1	0.301	39.0	0.228	70.5	0.129
SW-SSVM	75.3	0.142	48.6	0.211	40.3	0.259	42.2	0.216	73.3	0.122
SW-SPEC	78.7	0.133	47.2	0.212	42.1	0.253	44.2	0.211	72.4	0.124
DW-SSVM	84.3	0.102	54.2	0.189	50.8	0.244	52.1	0.186	75.5	0.120
HARG-SSVM	**93.9**	**0.070**	**71.4**	**0.134**	**71.9**	**0.158**	**72.2**	**0.126**	**86.1**	**0.090**

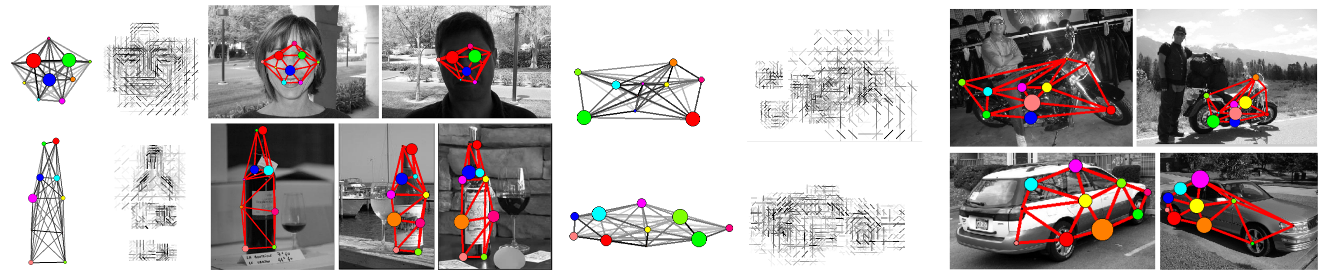


Figure 4: Learned graph models and their matching results. For each class, the learned graph model, its learned appearance in node attributes, and matching results are shown. In the graph model, bigger circles represent stronger nodes, and darker lines denote stronger edges. In the second and the fifth columns, to better visualize node attributes, we show the edge responses based on the learned SIFT attributes. For each model, some matching examples with high scores are shown. The results show that the learned graph model enables robust matching in spite of deformation and appearance changes. (Best viewed in pdf.)

[5] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, and A. J. Smola. Learning graph matching. *TPAMI*, 2009.

[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] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[11] J. Davis and P. Domingos. Bottom-up learning of Markov network structure. In *ICML*, 2010.

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

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

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

[15] S. Hare, A. Saffari, and P. H. S. Torr. Efficient Online Structured Output Learning for Keypoint-Based Object Tracking. In *CVPR*, 2012.

[16] T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane training of structural svms. *Machine Learning*, 2009.

[17] S.-I. Lee, V. Ganapahthi, and D. Koller. Efficient Structure Learning of Markov Networks using L1-Regularization. In *NIPS*, 2006.

[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, M. Hebert, and R. Sukthankar. An integer projected fixed point method for graph matching and map inference. In *NIPS*, 2009.

[21] M. Leordeanu, R. Sukthankar, and M. Hebert. Unsupervised learning for graph matching. *IJCV*, 2012.

[22] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

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

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

[25] S. Nowozin, P. Gehler, and C. Lampert. On parameter learning in CRF-based approaches to object class image segmentation. *ECCV*, 2010.

[26] D. Pachauri, M. Collins, V. SIngh, R. Kondor, and V. S. Deepti Pachauri, Maxwell Collins, Risi Kondor. Incorporating domain knowledge in matching problems via harmonic analysis. In *ICML*, 2012.

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

[28] M. Szummer, P. Kohli, and D. Hoiem. Learning CRFs using graph cuts. In *ECCV*, 2008.

[29] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *NIPS*, 2003.

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

[31] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 2005.

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

[33] B. Yao and L. Fei-Fei. Action recognition with exemplar based 2.5d graph matching. In *ECCV*, 2012.