Learning Graph Matching

Tibério S. Caetano, Julian J. McAuley, *Student Member*, *IEEE*,
Li Cheng, *Member*, *IEEE*, Quoc V. Le, and Alex J. Smola

Abstract—As a fundamental problem in pattern recognition, graph matching has applications in a variety of fields, from computer vision to computational biology. In graph matching, patterns are modeled as graphs and pattern recognition amounts to finding a correspondence between the nodes of different graphs. Many formulations of this problem can be cast in general as a quadratic assignment problem, where a linear term in the objective function encodes node compatibility and a quadratic term encodes edge compatibility. The main research focus in this theme is about designing efficient algorithms for approximately solving the quadratic assignment problem since it is NP-hard. In this paper, we turn our attention to a different question: how to *estimate compatibility functions* such that the solution of the resulting graph matching problem best matches the expected solution that a human would manually provide. We present a method for *learning graph matching*: The training examples are pairs of graphs and the "labels" are matches between them. Our experimental results reveal that learning can substantially improve the performance of standard graph matching algorithms. In particular, we find that simple linear assignment with such a learning scheme outperforms Graduated Assignment with bistochastic normalization, a state-of-the-art quadratic assignment relaxation algorithm.

Index Terms—Graph matching, learning, support vector machines, structured estimation, optimization.

1 Introduction

Graphs are commonly used as abstract representations for complex structures, including DNA sequences, documents, text, and images. In particular they are extensively used in the field of computer vision, where many problems can be formulated as an attributed graph matching problem. Here the nodes of the graphs correspond to local features of the image and edges correspond to relational aspects between features (both nodes and edges can be attributed, i.e., they can encode feature vectors). Graph matching then consists of finding a correspondence between nodes of the two graphs such that they "look most similar" when the vertices are labeled according to such a correspondence.

Typically, the problem is mathematically formulated as a quadratic assignment problem, which consists of finding the assignment that maximizes an objective function encoding local compatibilities (a linear term) and structural compatibilities (a quadratic term). The main body of research in graph matching has then been focused on devising more accurate and/or faster algorithms to solve the problem approximately (since it is NP-hard); the compatibility functions used in graph matching are typically handcrafted.

An interesting question arises in this context: If we are given two attributed graphs to match, G and G', should the optimal match be uniquely determined? For example, assume first that G and G' come from two images acquired by a surveillance camera in an airport's lounge; now, assume that the same G and G' instead come from two images in a photographer's image database; should the optimal match be the same in both situations? If the algorithm takes into account exclusively the graphs to be matched, the optimal solutions will be the same[1] since the graph pair is the same in both cases. This is the standard way graph matching is approached today.

In this paper, we address what we believe to be a limitation of this approach. We argue that if we know the "conditions" under which a pair of graphs has been extracted, then we should take into account *how graphs arising in those conditions are typically matched*. However, we do not take the information on the conditions explicitly into account, since this would obviously be impractical. Instead, we approach the problem purely from a statistical inference perspective. First, we extract graphs from a number of images acquired under the same conditions as those for which we want to solve, whatever the word "conditions" means (e.g., from the surveillance camera or the photographer's database). We then *manually* provide what we understand to be the optimal matches between the resulting graphs. This information is then used in a *learning* algorithm which learns a map from the space of pairs of graphs to the space of matches.

In terms of the quadratic assignment problem, this learning algorithm amounts to (in loose language) adjusting

- *T.S. Caetano and J.J. McAuley are with the Statistical Machine Learning Group, NICTA, Locked Bag 8001, Canberra, ACT 2601, Australia, and the Research School of Information Sciences and Engineering, Australian National University, ACT 0200, Australia. E-mail: {tiberio.caetano, julian.mcauley}@nicta.com.au.*
- *Li Cheng is with TTI-Chicago, 1427 East 60th Street, Chicago, IL 60637. E-mail: licheng@tti-c.org.*
- *Q.V. Le is with the Department of Computer Science, Stanford University, Room 112, Gates Building 1A, Stanford, CA 94305. E-mail: quocle@stanford.edu.*
- *A.J. Smola is with Yahoo! Research, 2821 Mission College Blvd., 1MC8332, Santa Clara, CA 95051. E-mail: alex@smola.org.*

Manuscript received 16 June 2008; revised 2 Nov. 2008; accepted 16 Jan. 2009; published online 23 Jan. 2009.
Recommended for acceptance by M. Pelillo.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2008-06-0361.
Digital Object Identifier no. 10.1109/TPAMI.2009.28.

1. Assuming there is a single optimal solution and that the algorithm finds it.

the node and edge compatibility functions such that the expected optimal match in a test pair of graphs agrees with the expected match they would have had had they been in the training set. In this formulation, the learning problem consists of a convex, quadratic program which is readily solvable by means of a column generation procedure.

We provide experimental evidence that applying learning to standard graph matching algorithms significantly improves their performance. In fact, we show that learning improves upon nonlearning results so dramatically that linear assignment *with learning* outperforms Graduated Assignment with bistochastic normalization, a state-of-the-art quadratic assignment relaxation algorithm. Also, by introducing learning in Graduated Assignment itself, we obtain results that improve both in accuracy and speed over the state-of-the-art relaxation.

A preliminary version of this paper appeared in [1].

TABLE 1
Definitions and Notation

G - generic graph (similarly, G');
G_i - attribute of node i in G (similarly, $G'_{i'}$ for G');
G_{ij} - attribute of edge ij in G (similarly, $G'_{i'j'}$ for G');
$\mathcal{G}$ - space of graphs ($\mathcal{G} \times \mathcal{G}$ - space of pairs of graphs);
x - generic observation: graph pair (G, G'); $x \in \mathcal{X}$, space of observations;
y - generic label: matching matrix; $y \in \mathcal{Y}$, space of labels;
n - index for training instance; N - number of training instances;
x^n - n^{th} training observation: graph pair (G^n, G'^n);
y^n - n^{th} training label: matching matrix;
g - predictor function;
y^w - optimal prediction for g under w;
f - discriminant function;
Δ - loss function;
Φ - joint feature map;
ϕ_1 - node feature map;
ϕ_2 - edge feature map;
S_n - constraint set for training instance n;
y^* - solution of the quadratic assignment problem;
$\hat{y}$ - most violated constraint in column generation;
$y_{ii'}$ - i^{th} row and i'^{th} column element of y;
$c_{ii'}$ - value of compatibility function for map $i \mapsto i'$;
$d_{ii'jj'}$ - value of compatibility function for map $ij \mapsto i'j'$;
ϵ - tolerance for column generation;
w_1 - node parameter vector; w_2 - edge parameter vector; $w := [w_1\ w_2]$ - joint parameter vector; $w \in \mathcal{W}$;
ξ_n - slack variable for training instance n;
Ω - regularization function; λ - regularization parameter;
δ - convergence monitoring threshold in bistochastic normalization.

2 Literature Review

2.1 Learning with Graphs

For completeness, we briefly touch on a *related* body of literature, which, although clearly distinct from graph matching, does involve the concept of learning in data structures represented as graphs. We stress that the work in this literature is essentially concerned with problems of classifying and/or clustering graphs, but *not* learning a matching criterion per se.

Since graphs are eminently nonvectorial data structures, a substantial part of this literature has been focused on Kernel Methods [2], [3], which comprise a principled framework for dealing with structured data using standard tools from linear analysis. We refer the reader to the recent unified treatment on these methods as applied to graphs [4], as well as the references therein. Another line of work has been the use of generative models for graphs in the structural pattern recognition community, such as [5], [6] and [7]. Also, learning the graph edit distance for purposes of graph classification has been introduced in [8].

2.2 Graph Matching

The graph matching literature is extensive, and many different types of approaches have been proposed, which mainly focus on approximations and heuristics for the quadratic assignment problem. An incomplete list includes *spectral* methods [9], [10], [11], [12], [13], *relaxation labeling* and *probabilistic approaches* [14], [15], [16], [17], [18], [19], [20], *semidefinite relaxations* [21], *replicator equations* [22], *tree search* [23], *graduated assignment* [24], and *RKHS* methods [25]. Spectral methods consist of studying the similarities between the spectra of the adjacency or Laplacian matrices of the graphs and using them for matching. Relaxation and probabilistic methods define a probability distribution over mappings, and optimize using discrete relaxation algorithms or variants of belief propagation. Semidefinite relaxations solve a convex relaxation of the original combinatorial problem. Replicator equations draw an analogy with models from biology where an equilibrium state is sought, which solves a system of differential equations on the nodes of the graphs. Tree-search techniques in general have worst-case exponential complexity and work via sequential tests of compatibility of local parts of the graphs. Graduated Assignment combines the "softassign" method [26] with Sinkhorn's method [27] and essentially consists of a series of first-order approximations to the quadratic assignment objective function. This method is particularly popular in computer vision since it produces accurate results while scaling reasonably in the size of the graph.

The above literature strictly focuses on trying better *algorithms* for approximating a solution for the graph matching problem, but does not address the issue of how to determine the compatibility functions in a principled way.

In [28] , the authors learn compatibility functions for the relaxation labeling process; this is, however, a different problem than graph matching and the "compatibility functions" have a different meaning. Nevertheless, it does provide an initial motivation for learning in the context of matching tasks. In terms of methodology, the paper most closely related to ours is possibly [29], which uses structured estimation tools in a quadratic assignment setting for word alignment. A recent paper of interest shows that *very* significant improvements on the performance of graph matching can be obtained by an appropriate *normalization* of the compatibility functions [30]; however, no learning is involved.

3 The Graph Matching Problem

The notation used in this paper is summarized in Table 1. In the following, we denote a graph by G. We will often refer to a *pair* of graphs, and the second graph in the pair will be denoted by G'. We study the general case of *attributed* graph matching, and attributes of the vertex i and the edge ij in G are denoted by G_i and G_{ij}, respectively. Standard graphs

are obtained if the node attributes are empty and the edge attributes $G_{ij} \in \{0,1\}$ are binary, denoting the absence or presence of an edge, in which case we get the so-called *exact* graph matching problem.

Define a *matching matrix* y by $y_{ii'} \in \{0,1\}$ such that $y_{ii'} = 1$ if node i in the first graph maps to node i' in the second graph ($i \mapsto i'$) and $y_{ii'} = 0$ otherwise. Define by $c_{ii'}$ the value of the compatibility function for the unary assignment $i \mapsto i'$ and by $d_{ii'jj'}$ the value of the compatibility function for the pairwise assignment $ij \mapsto i'j'$. Then, a generic formulation of the graph matching problem consists of finding the optimal matching matrix y^* given by the solution of the following (NP-hard) *quadratic assignment problem* [31],

$$y^* = \underset{y}{\operatorname{argmax}} \left[\sum_{ii'} c_{ii'} y_{ii'} + \sum_{ii'jj'} d_{ii'jj'} y_{ii'} y_{jj'} \right], \tag{1}$$

typically subject to either the injectivity constraint (one-to-one, that is, $\sum_i y_{ii'} \leq 1$ for all i', $\sum_{i'} y_{ii'} \leq 1$ for all i) or simply the constraint that the map should be a function (many-to-one, that is, $\sum_{i'} y_{ii'} = 1$ for all i). If $d_{ii'jj'} = 0$ for all $ii'jj'$, then (1) becomes a *linear assignment problem*, exactly solvable in worst-case cubic time [32]. Although the compatibility functions c and d obviously depend on the attributes $\{G_i, G'_{i'}\}$ and $\{G_{ij}, G'_{i'j'}\}$, the functional form of this dependency is typically assumed to be fixed in graph matching. This is precisely the restriction we are going to relax in this paper: Both the functions c and d will be parametrized by vectors whose coefficients will be learned within a convex optimization framework. In a way, instead of proposing yet another algorithm for determining *how* to approximate the solution for (1), we are aiming at finding a way to determine *what* should be maximized in (1) since different c and d will produce different criteria to be maximized.

4 Learning Graph Matching

4.1 General Problem Setting

We approach the problem of learning the compatibility functions for graph matching as a supervised learning problem [33]. The training set is comprised of N observations x from an input set $\mathcal{X}$ and N corresponding labels y from an output set $\mathcal{Y}$ and can be represented by $\{(x^1; y^1), \ldots, (x^N; y^N)\}$. Critical in our setting is the fact that the observations and labels are *structured objects*. In typical supervised learning scenarios, observations are vectors and labels are elements from some discrete set of small cardinality, for example, $y^n \in \{-1, 1\}$ in the case of binary classification. However, in our case, an observation x^n is a *pair of graphs*, i.e., $x^n = (G^n, G'^n)$, and the label y^n is a *match* between graphs, represented by a matching matrix, as defined in Section 3.

If $\mathcal{X} = \mathcal{G} \times \mathcal{G}$ is the space of pairs of graphs and $\mathcal{Y}$ the space of matching matrices, then learning graph matching amounts to finding a w-parametrized function $g_w : \mathcal{G} \times \mathcal{G} \mapsto \mathcal{Y}$ which minimizes the prediction loss on the test set. Since the test set here is assumed not to be available at training time, we use the standard approach of minimizing the empirical risk (average loss in the training set) plus a regularization term in order to avoid overfitting. The optimal predictor will then be the one which minimizes an expression of the following type:

$$\underbrace{\frac{1}{N} \sum_{n=1}^{N} \Delta(g_w(G^n, G'^n), y^n)}_{\text{empirical risk}} + \underbrace{\lambda \Omega(w)}_{\text{regularization term}}, \tag{2}$$

where $\Delta(g_w(G^n, G'^n), y^n)$ is the loss incurred by the predictor g when predicting, for training input (G^n, G'^n), the output $g_w(G^n, G'^n)$ instead of the training output y^n. The function $\Omega(w)$ penalizes "complex" vectors w, and λ is a parameter that trades off data fitting against generalization ability, which is, in practice, determined using a validation set. In order to completely specify such an optimization problem, we need to define the parametrized class of predictors $g_w(G, G')$, whose parameters w we will optimize over the loss function Δ, and the regularization term $\Omega(w)$. In the following, we will focus on setting up the optimization problem by addressing each of these points.

4.2 The Model

We start by specifying a w-parametrized class of predictors $g_w(G, G')$. We use the standard approach of discriminant functions, which consists of picking as our optimal estimate the one for which the discriminant function $f(G, G', y; w)$ is maximal, i.e., $g_w(G, G') = \operatorname{argmax}_y f(G, G', y; w)$. We assume linear discriminant functions $f(G, G', y; w) = \langle w, \Phi(G, G', y) \rangle$ so that our predictor has the form

$$g_w(G, G') = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \langle w, \Phi(G, G', y) \rangle. \tag{3}$$

Effectively, we are solving an inverse optimization problem, as described in [33], [34], that is, we are trying to find f such that g has desirable properties. Further specification of $g_w(G, G')$ requires determining the joint feature map $\Phi(G, G', y)$, which has to encode the properties of both graphs as well as the properties of a match y between these graphs. The key observation here is that we can relate the quadratic assignment formulation of graph matching, given by (1), with the predictor given by (3), and interpret the solution of the graph matching problem as being the estimate of g, i.e., $y^w = g_w(G, G')$. This allows us to interpret the discriminant function in (3) as the objective function to be maximized in (1):

$$\langle \Phi(G, G', y), w \rangle = \sum_{ii'} c_{ii'} y_{ii'} + \sum_{ii'jj'} d_{ii'jj'} y_{ii'} y_{jj'}. \tag{4}$$

This clearly reveals that the graphs and the parameters must be encoded in the compatibility functions. The last step before obtaining Φ consists of choosing a parameterization for the compatibility functions. We assume a simple linear parameterization

$$c_{ii'} = \langle \phi_1(G_i, G'_{i'}), w_1 \rangle, \tag{5a}$$

$$d_{ii'jj'} = \langle \phi_2(G_{ij}, G'_{i'j'}), w_2 \rangle, \tag{5b}$$

i.e., the compatibility functions are linearly dependent on the parameters, and on new feature maps ϕ_1 and ϕ_2 that only involve the graphs (Section 5 specifies the feature maps ϕ_1 and ϕ_2). As already defined, G_i is the attribute of

node i and G_{ij} is the attribute of edge ij (similarly for G'). However, we stress here that these are not necessarily *local* attributes, but are arbitrary features *simply indexed* by the nodes and edges.[2] For instance, we will see in Section 5 an example where G_i encodes the graph structure of G as "seen" from node i, or from the "perspective" of node i.

Note that the traditional way in which graph matching is approached arises as a particular case of (5): If w_1 and w_2 are constants, then $c_{ii'}$ and $d_{ii'jj'}$ depend only on the features of the graphs. By defining $w := [w_1 \;\; w_2]$, we arrive at the final form for $\Phi(G, G', y)$ from (4) and (5):

$$\Phi(G, G', y) = \left[\sum_{ii'} y_{ii'} \phi_1(G_i, G'_{i'}), \sum_{ii'jj'} y_{ii'} y_{jj'} \phi_2(G_{ij}, G'_{i'j'}) \right]. \tag{6}$$

Naturally, the final specification of the predictor g depends on the choices of ϕ_1 and ϕ_2. Since our experiments are concentrated on the computer vision domain, we use typical computer vision features (e.g., Shape Context) for constructing ϕ_1 and a simple edge-match criterion for constructing ϕ_2 (details follow in Section 5).

4.3 The Loss

Next we define the loss $\Delta(y, y^n)$ incurred by estimating the matching matrix y instead of the correct one, y^n. When both graphs have large sizes, we define this as the fraction of mismatches between matrices y and y^n (i.e., a normalized Hamming loss),

$$\Delta(y, y^n) = 1 - \frac{1}{\|y^n\|_F^2} \sum_{ii'} y_{ii'} y^n_{ii'}, \tag{7}$$

where $\|\cdot\|_F$ is the Frobenius norm. If one of the graphs has a small size, this measure may be too rough. In our experiments, we will encounter such a situation in the context of matching in images. In this case, we instead use the loss

$$\Delta(G, G', \pi, \pi^n) = 1 - \frac{1}{|\pi|} \sum_i \left[\frac{d(G'_{\pi(i)}, G'_{\pi^n(i)})}{\sigma} \right]. \tag{8}$$

Here, graph nodes correspond to point sets in the images, G corresponds to the smaller, "query" graph and G' is the larger, "target" graph (in this expression, G_i and G'_j are particular points in G and G'; $\pi(i)$ is the index of the point in G' to which the ith point in G is mapped, $\pi^n(i)$ is the index of the "correct" mapping; and d is simply the euclidean distance and is scaled by σ, which is simply the width of the image in question). Hence, we are penalizing matches based on how distant they are from the correct match; this is commonly referred to as the "endpoint error."

Finally, we specify a quadratic regularizer $\Omega(w) = \frac{1}{2}\|w\|^2$.

4.4 The Optimization Problem

Here, we combine the elements discussed in Section 4.2 in order to formally set up a mathematical optimization problem that corresponds to the learning procedure, following the large-margin approach of [33]. The expression that arises from (2) by incorporating the specifics discussed in Sections 4.2 and 4.3 still consists of a very difficult (in particular nonconvex) optimization problem. Although the regularization term is convex in the parameters w, the empirical risk, i.e., the first term in (2), is not. Note that there are finitely many matches y and, therefore, a finite number of possible values for the loss Δ; however, the space of parameters $\mathcal{W}$ is continuous. What this means is that there are large equivalence classes of w (an equivalence class in this case is a given set of ws each of which produces the same loss). Therefore, the loss is piecewise constant on w and, as a result, certainly not amenable to any type of smooth optimization.

One approach to render the problem of minimizing (2) more tractable is to replace the empirical risk by a convex upper bound on the empirical risk, an idea that has been exploited in machine learning in recent years [33], [35], [36]. By minimizing this convex upper bound, we hope to decrease the empirical risk as well. It is easy to show that the convex (in particular, linear) function $\frac{1}{N}\sum_n \xi_n$ is an upper bound for $\frac{1}{N}\sum_n \Delta(g_w(G^n, G'^n), y^n)$ for the solution of (2) with appropriately chosen constraints:

$$\underset{w,\xi}{\text{minimize}} \; \frac{1}{N}\sum_{n=1}^{N} \xi_n + \frac{\lambda}{2}\|w\|^2 \tag{9a}$$

$$\text{subject to } \langle w, \Psi^n(y) \rangle \geq \Delta(y, y^n) - \xi_n \quad \text{for all } n \text{ and } y \in \mathcal{Y}. \tag{9b}$$

Here, we define $\Psi^n(y) := \Phi(G^n, G'^n, y^n) - \Phi(G^n, G'^n, y)$. Formally, we have the following lemma:

Lemma 1. *For any feasible (ξ, w) of (9), the inequality $\xi_n \geq \Delta(g_w(G^n, G'^n), y^n)$ holds for all n. In particular, for the optimal solution (ξ^*, w^*), we have $\frac{1}{N}\sum_n \xi^*_n \geq \frac{1}{N}\sum_n \Delta(g_{w^*}(G^n, G'^n), y^n)$.*

Proof. The constraint (9-b) needs to hold for all y, hence in particular for $y^{w^*} = g_{w^*}(G^n, G'^n)$. By construction, y^{w^*} satisfies $\langle w, \Psi^n(y^{w^*}) \rangle \leq 0$. Consequently, $\xi_n \geq \Delta(y^{w^*}, y^n)$. The second part of the claim follows immediately. □

The constraints (9b) mean that the margin $f(G^n, G'^n, y^n; w) - f(G^n, G'^n, y; w)$, i.e., the gap between the discriminant functions for y^n and y should exceed the loss induced by estimating y instead of the training matching matrix y^n. This is highly intuitive since it reflects the fact that we want to safeguard ourselves most against mispredictions y, which incur a large loss (i.e., the smaller the loss, the less we should care about making a misprediction, so we can enforce a smaller margin). The presence of ξ_n in the constraints and in the objective function means that we allow the hard inequality (without ξ_n) to be violated, but we penalize violations for a given n by adding to the objective function the cost $\frac{1}{N}\xi_n$.

Despite the fact that (9) has exponentially many constraints (every possible matching y is a constraint), we will see in what follows that there is an efficient way of finding an ϵ-approximation to the optimal solution of (9) by finding the worst violators of the constrained optimization problem.

2. As a result, in our general setting, "node" compatibilities and "edge" compatibilities become somewhat misnomers, being more appropriately described as unary and binary compatibilities. We, however, stick to the standard terminology for simplicity of exposition.

4.5 The Algorithm

Instead of using the formulation in (9), which has n slack variables (used in [1] and [33]), we here use the (equivalent) formulation given in [37], in which there is only a single slack variable:

$$\underset{w,\xi}{\text{minimize}}\ \xi + \frac{\lambda}{2}\|w\|^2 \tag{10a}$$

$$\text{subject to}\ \frac{1}{N}\sum_n \langle w, \Psi^n(y)\rangle \geq \frac{1}{N}\sum_n \Delta(y, y^n) - \xi \quad \text{for all } y \in \mathcal{Y}. \tag{10b}$$

Note that the number of constraints in (10) is given by the number of *possible* matching matrices $\|\mathcal{Y}\|$ times the number of training instances N. In graph matching, the number of possible matches between two graphs grows factorially with their size. In this case, it is infeasible to solve (9) exactly.

There is, however, a way out of this problem by using an optimization technique known as *column generation* [32]. Instead of solving (10) directly, one computes the most violated constraint in (10) iteratively for the current solution and adds this constraint to the optimization problem. In order to do so, we need to solve

$$\hat{y}_n = \underset{y}{\text{argmax}}[\langle w, \Phi(G^n, G'^n, y)\rangle + \Delta(y, y^n)], \tag{11}$$

as this is the term for which the constraint (10b) is tightest (i.e., the constraint that maximizes ξ). Substituting into (10b) we obtain

$$\xi = \Delta(\hat{y}_n, y^n) - \langle w, \Psi^n(\hat{y}_n)\rangle. \tag{12}$$

Thus, in (10a), we obtain

$$\frac{1}{N}\sum_n \Delta(\hat{y}_n, y^n) - \langle w, \Psi^n(\hat{y}_n)\rangle + \frac{\lambda}{2}\|w\|^2, \tag{13}$$

whose gradient (with respect to w) is

$$\lambda w - \frac{1}{N}\sum_n \Psi^n(\hat{y}_n). \tag{14}$$

Equations (13) and (14) define the new constraint to be added to the optimization problem. Pseudocode for this algorithm is described in Algorithm 1. See [38] for more details.

Let us investigate the complexity of solving (11). Using the joint feature map Φ as in (6) and the loss as in (7), the argument in (11) becomes

$$\langle \Phi(G, G', y), w\rangle + \Delta(y, y^n) = \sum_{ii'} y_{ii'}\bar{c}_{ii'} + \sum_{ii'jj'} y_{ii'}y_{jj'}d_{ii'jj'} + \text{constant}, \tag{15}$$

where $\bar{c}_{ii'} = \langle \phi_1(G_i, G'_{i'})w_1\rangle + \|y^n_{ii'}/{y^n}^2_F\|$ and $d_{ii'jj'}$ is defined as in (5b).

The maximization of (15), which needs to be carried out at *training* time, is a quadratic assignment problem, as is the problem to be solved at test time. In the particular case where $d_{ii'jj'} = 0$ throughout, both the problems at training and at test time are linear assignment problems, which can be solved efficiently in worst-case cubic time.

In our experiments, we solve the linear assignment problem with the efficient solver from [39] ("house" sequence) and the Hungarian algorithm (video/bikes data set). For quadratic assignment, we developed a C++ implementation of the well-known Graduated Assignment algorithm [24]. However, the learning scheme discussed here is independent of which algorithm we use for solving either linear or quadratic assignment. Note that the estimator is only an approximation in the case of quadratic assignment: Since we are not guaranteed to find *precisely* the *most* violated constraint of (11), we cannot be sure that the duality gap is properly minimized in the constrained optimization problem.[3]

Algorithm 1. Bundle Method

Define:
$\Psi^n(y) := \Phi(G^n, G'^n, y^n) - \Phi(G^n, G'^n, y)$
$H^n(y) := \langle w, \Phi(G^n, G'^n, y)\rangle + \Delta(y, y^n)$
Input: training graph pairs $\{G^n\}$,$\{G'^n\}$, training matching matrices $\{y^n\}$, sample size N, tolerance ϵ
Initialize $i = 1$, $w_1 = 0$
repeat
  for $n = 1$ **to** N **do**
    $\hat{y}_n = \text{argmax}_{y\in\mathcal{Y}} H^n(y)$
  end for
  Compute 'gradient' a_i (equation 14)
  Compute 'offset' b_i (equation 13)
  $w_{i+1} := \text{argmin}_w \frac{\lambda}{2}\|w\|^2 + \max(0, \max_{j\leq i}\langle a_j, w\rangle + b_j)$
  $i \leftarrow i + 1$
until converged (see [38])

5 Features for the Compatibility Functions

The joint feature map $\Phi(G, G', y)$ has been derived in its full generality (6), but, in order to have a working model, we need to choose a specific form for $\phi_1(G_i, G'_{i'})$ and $\phi_2(G_{ij}, G'_{i'j'})$, as mentioned in Section 4. We first discuss the linear features ϕ_1 and then proceed to the quadratic terms ϕ_2. For concreteness, here we only discuss options actually used in our experiments.

5.1 Node Features

We construct $\phi_1(G_i, G'_{i'})$ using the squared difference $\phi_1(G_i, G'_{i'}) = (\ldots, -|G_i(r) - G'_{i'}(r)|^2, \ldots)$. This differs from what is shown in [1], in which an exponential decay is used (i.e., $\exp(-|G_i(r) - G'_{i'}(r)|^2)$); we found that using the squared difference resulted in much better performance after learning. Here, $G_i(r)$ and $G'_{i'}(r)$ denote the rth coordinates of the corresponding attribute vectors. Note that, in standard graph matching without learning, we typically have $c_{ii'} = \exp(-\|G_i - G'_{i'}\|^2)$, which can be

3. Recent work has been done on structured learning when exact inference is not feasible [40]. In that paper, the authors analyze both theoretically and empirically the class of models represented by a fully connected Markov random field. This sheds some light on structured prediction problems such as multilabel classification, clustering, and image segmentation. Unfortunately, the analysis does not apply to settings with hard assignment constraints, such as quadratic assignment, which makes it difficult for us to assess to what extent (and if) their insights extend to our setting.

seen as the particular case of (5a) for both ϕ_1 and w_1 flat, given by $\phi_1(G_i, G'_{i'}) = (\ldots, \exp(-\|G_i - G'_{i'}\|^2), \ldots)$ and $w_1 = (\ldots, 1, \ldots)$ [30]. Here, instead, we have $c_{ii'} = \langle \phi_1(G_i, G'_{i'}), w_1 \rangle$, where w_1 is learned from training data. In this way, by tuning the rth coordinate of w_1 accordingly, the learning process finds the relevance of the rth feature of ϕ_1. In our experiments (to be described in the next section), we use the well-known 60-dimensional Shape Context features [41]. They encode how each node "sees" the other nodes. It is an instance of what we called in Section 4 a feature that captures the node "perspective" with respect to the graph. We use 12 angular bins (for angles in $[0, \frac{\pi}{6}) \ldots [\frac{11\pi}{6}, 2\pi)$) and 5 radial bins (for radii in $(0, 0.125), [0.125, 0.25) \ldots [1, 2)$, where the radius is scaled by the average of all distances in the scene) to obtain our 60 features. This is similar to the setting described in [41].

5.2 Edge Features

For the edge features G_{ij} ($G'_{i'j'}$), we use standard graphs, i.e., G_{ij} ($G'_{i'j'}$) is 1 if there is an edge between i and j and 0 otherwise. In this case, we set $\phi_2(G_{ij}, G'_{i'j'}) = G_{ij}G'_{i'j'}$ (so that w_2 is a scalar).

6 Experiments

6.1 House/Hotel Sequence

For our first experiment, we consider the CMU "house" sequence—a data set consisting of 111 frames of a toy house [42]. Each frame in this sequence has been hand-labeled, with the same 30 landmarks identified in each frame [43]. We explore the performance of our method as the baseline (separation between frames) varies. We assess the quality of a match with the normalized Hamming loss (7).

For each baseline (from 0 to 90, by 10), we identified all pairs of images separated by exactly this many frames. We then split these pairs into three sets, for training, validation, and testing. In order to determine the adjacency matrix for our edge features, we triangulated the set of landmarks using the Delaunay triangulation (see Fig. 1).

Fig. 1a shows the performance of our method as the baseline increases, for both linear and quadratic assignment (for quadratic assignment, we use the Graduated Assignment algorithm, as mentioned previously). The values shown report the normalized Hamming loss (i.e., the proportion of points incorrectly matched); for each baseline, the regularization constant resulting in the best performance of its validation set is used for testing. Graduated assignment using bistochastic normalization (with a normalization constant of $\delta = 0.00001$), which, to the best of our knowledge, is the state-of-the-art relaxation, is shown for comparison (quadratic normalization $\delta = 0.00001$); the spectral matching implementation of [30] is also shown (SMAC).[4]

For both linear and quadratic assignment, Fig. 1 shows that learning significantly outperforms nonlearning in terms of accuracy. Interestingly, quadratic assignment performs *worse* than linear assignment before learning is applied—this is likely because the relative scale of the linear and quadratic features is badly tuned before learning. The line "quadratic" (simple scaling) shows that we can address this problem somewhat by scaling the linear features to be in the range $[0, 1]$ (so that both the linear and quadratic features have the same scale), in which case quadratic

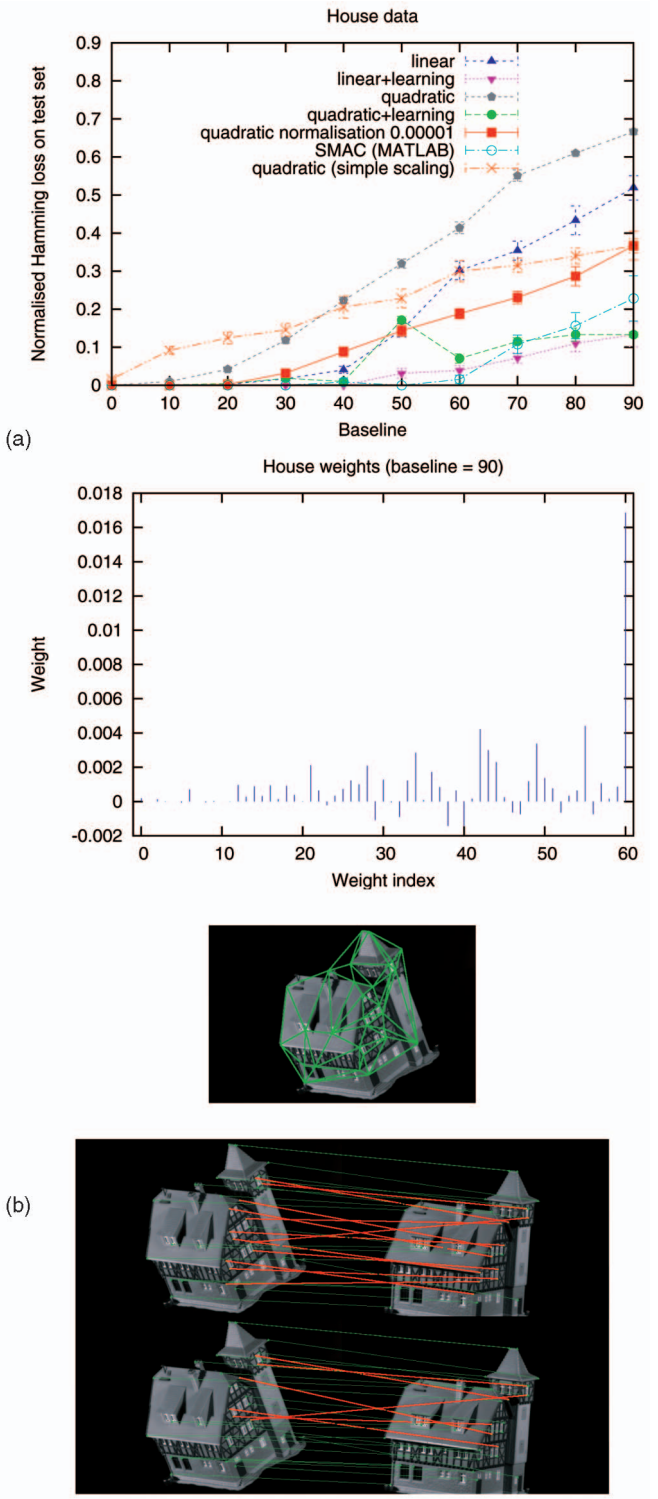


Fig. 1. (a) Performance on the "house" sequence as the baseline (separation between frames) varies (the normalized Hamming loss on all testing examples is reported, with error bars indicating the standard error). (b) The weights learned for the quadratic model ($\text{baseline} = 90$, $\lambda = 0.1$). (c) A frame from the sequence, together with its landmarks and triangulation; the 12th and the 102nd frames, matched using linear assignment (without learning, $\text{loss} = 14/30$), and the same match after learning ($\lambda = 1$, $\text{loss} = 6/30$). Mismatches are shown in red.

4. Exponential decay on the node features *was* beneficial when using the method of [30], and has hence been maintained in this case (see Section 5.1).

assignment does better than linear assignment for large baselines. It is also worth noting that linear assignment *with* learning performs similarly to quadratic assignment with bistochastic normalization (without learning)—this is an important result since quadratic assignment via Graduated Assignment is significantly more computationally intensive. After learning, linear and quadratic assignment perform similarly, whereas we might expect quadratic assignment to do better; we expect that this is simply due to the inexact nature of the learning scheme when quadratic assignment is applied.

Fig. 1b shows the weight vector learned using quadratic assignment (for a baseline of 90 frames, with $\lambda = 1$). Note that the first 60 points show the weights of the Shape Context features, whereas the final point corresponds to the edge features. The final point is given a very high score after learning, indicating that the edge features are important in this model.[5] Here, the first 12 features correspond to the first radial bin (as described in Section 5), etc. The first radial bin appears to be more important than the last, for example. Fig. 1c also shows an example match, using the 12th and the 102nd frames of the sequence for linear assignment, before and after learning.

Fig. 3 shows similar results, for another CMU data set (the 101 frame "hotel" sequence). Exponential decay was found to work better than linear decay for the quadratic assignment method in this experiment (see Section 5.1). After learning, quadratic assignment outperforms linear assignment in this experiment.

Finally, Fig. 4 shows the running time of our method compared to its accuracy. First, it should be noted that the use of learning has no effect on running time; since learning outperforms nonlearning in all cases, this presents a very strong case for learning.[6] Graduated assignment with bistochastic normalization gives the best nonlearning performance; however, it is still worse than either linear or quadratic assignment *with* learning and it is significantly slower. The implementation of [30] is also shown, though this code is implemented in MATLAB (whereas the others are implemented in C++), so direct comparison of running times is not possible. Note that the timing for the "hotel" sequence is identical and is not shown.

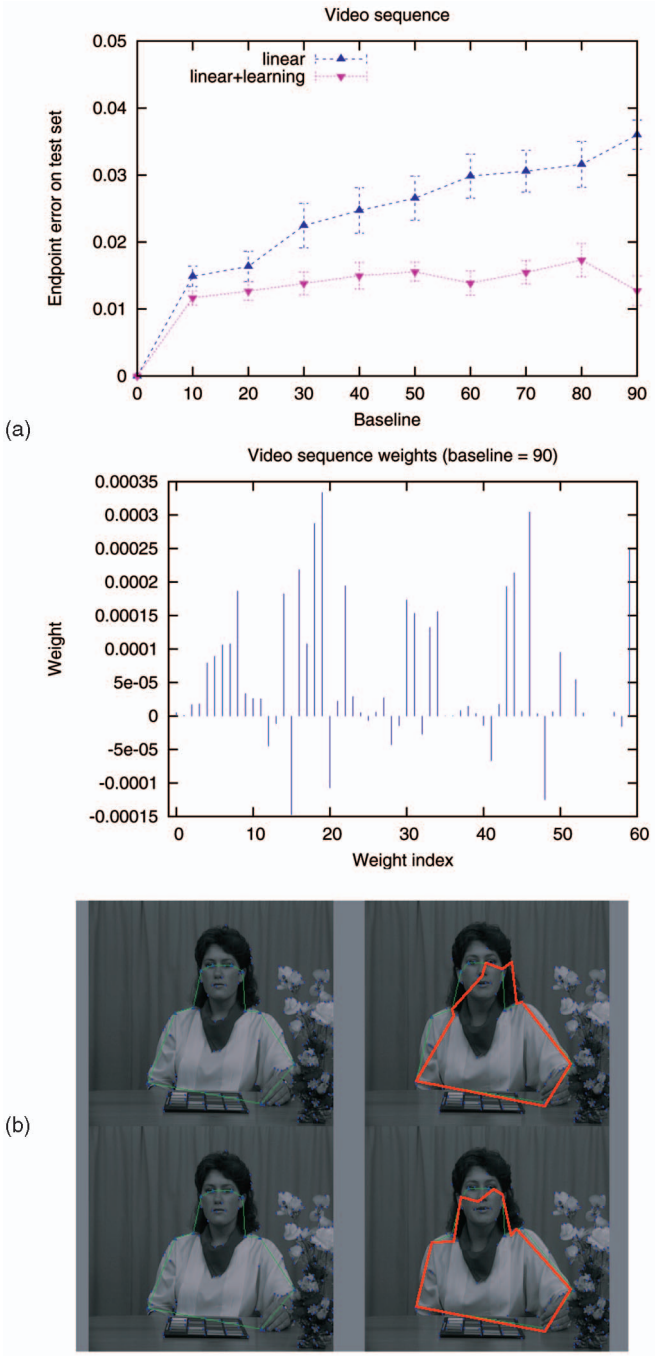


Fig. 2. (a) Performance on the video sequence as the baseline (separation between frames) varies (the endpoint error on all testing examples is reported, with error bars indicating the standard error). (b) The weights learned for the model ($\text{baseline} = 90, \lambda = 10{,}000$). (c) The 1st and the 91st frames, matched using linear assignment (loss = 0.035), and the same match after learning ($\lambda = 10{,}000, \text{loss} = 0.015$). The outline of the points to be matched (left) and the correct match (right) are shown in green; the inferred match is outlined in red; the match after learning is much closer to the correct match.

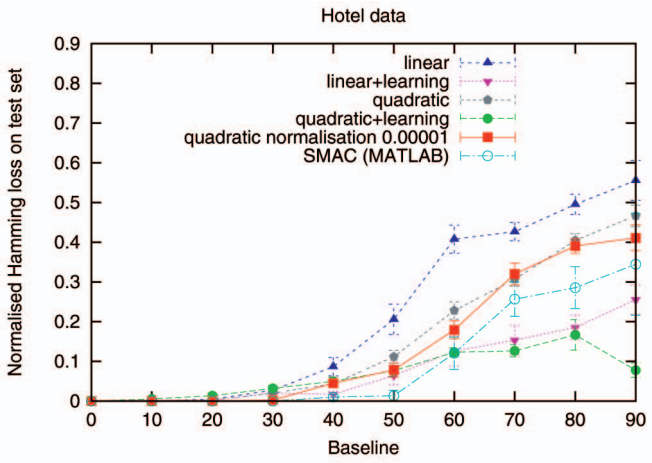
6.2 Synthetic Transformations

For our second experiment, we consider an artificially generated sequence of points and apply a series of common transformations to it. Again we use the Shape Context features, which are not invariant to these transformations; thus, the matching problem becomes more difficult as the transformations become more extreme. This experiment will assess how well learning is able to choose those features which remain useful under these transformations.

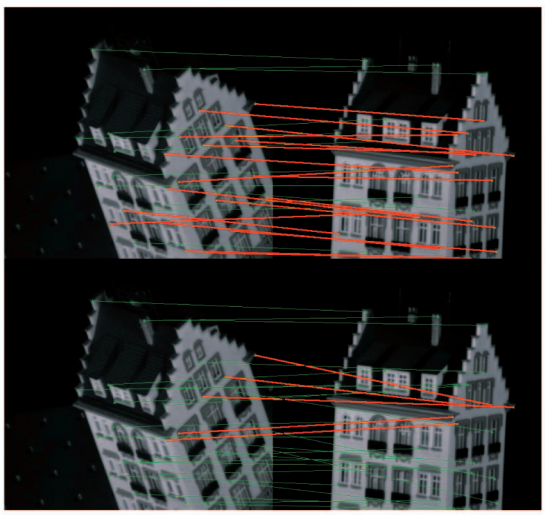
Our setup is similar to the previous experiments: We begin with the point set in Fig. 6a (image taken from [44], [45], [46], with 35 landmarks identified using code provided by Longbin Chen); we then rotate the point set by 90 degrees (for the "rotation" sequence), shear it horizontally to twice its width (for the "shear" sequence), and apply noise with standard deviation 20 pixels (for the "noise" sequence). These transformations are applied gradually over a series of 200 frames. We compute the Shape Context features and

5. This should be interpreted with some caution: The features have different scales, meaning that their importances cannot be compared directly. However, from the point of view of the regularizer, assigning this feature a high weight bears a high cost, implying that it is an important feature.

6. The time taken to learn the model is not included as it is typically an offline step. This amount can vary greatly, depending on the number of training samples, the value of λ, the matching algorithm being used, and various parameters of the learning algorithm; typically, less than 1 minute was required to learn each model.



(a)



(b)

Fig. 3. (a) Performance on the "hotel" sequence as the baseline (separation between frames) varies. The normalized Hamming loss on all testing examples is reported, with error bars indicating the standard error. (b) A frame from the sequence, together with its landmarks and triangulation; the 3rd and the 93rd frames, matched using linear assignment (without learning, $\text{loss} = 18/30$), and the same match after learning ($\lambda = 100, \text{loss} = 5/30$). Mismatches are shown in red.

Delaunay triangulation for each frame, and conduct experiments for fixed baselines as before.

Results for this experiment are shown in Fig. 5. First note that learning outperforms nonlearning in all cases, for both linear and quadratic assignments. Graduated assignment with bistochastic normalization and the SMAC method of [30] are also shown for comparison. Again, exponential decay was found to work better than linear decay for the quadratic assignment method in this experiment.

For the "rotation" sequence (Fig. 5a), the Shape Context features become gradually less useful as the baseline increases (as they are not invariant to rotation), meaning that the loss for linear assignment approaches 1 for large baselines. Alternately, the adjacency features are invariant to rotation; after learning, quadratic assignment achieves approximately zero error for all baselines.

For the "shear" sequence (Fig. 5b), quadratic assignment does much better than linear assignment before learning, though linear assignment does better after learning (as

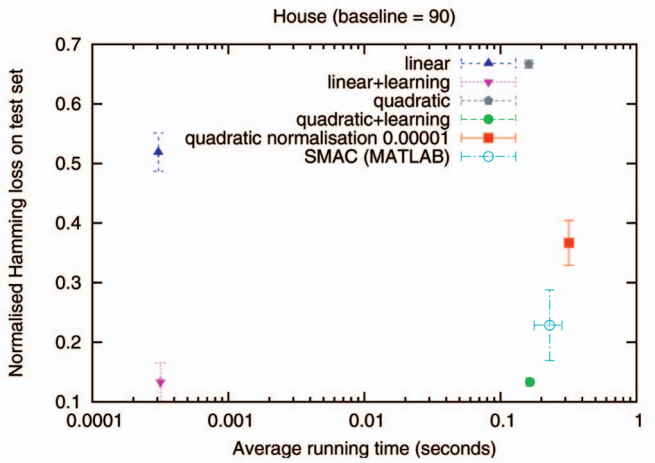


Fig. 4. Running time versus accuracy on the "house" data set, for a baseline of 90. Standard errors of both running time and performance are shown (the standard error for the running time is almost zero). Note that linear assignment is around three orders of magnitude faster than quadratic assignment. Note that the SMAC code of [30] is implemented in MATLAB, and is not directly comparable.

with the previous experiment, this is probably due to the inexact nature of the learning scheme when doing quadratic assignment).

For the "noise" sequence (Fig. 5c), linear assignment again does better than quadratic assignment—this is perhaps due to the fact that the Delaunay triangulation is very sensitive to noise, rendering the adjacency features useless for large baselines.

Fig. 6 shows an example match, before and after learning.

6.3 Video Sequence

For our third experiment, we consider matching features of a human in a video sequence. We used a video sequence from the SAMPL data set [47]—a 108 frame sequence of a human face (see Fig. 2c). To identify landmarks for these scenes, we used the SUSAN corner detector [48], [49]. This detector essentially identifies points as corners if their neighbors within a small radius are dissimilar. This detector was tuned such that no more than 200 landmarks were identified in each scene.

In this setting, we are no longer interested in matching *all* of the landmarks in both images, but rather those that correspond to important parts of the human figure. We identified the same 11 points in each image (Fig. 2c). It is assumed that these points are known in advance for the template scene (G) and are to be found in the target scene (G'). Clearly, since the correct match corresponds to only a tiny proportion of the scene, using the normalized Hamming loss is no longer appropriate—we wish to penalize incorrect matches less if they are "close to" the correct match. Hence, we use the loss function (as introduced in Section 4.2):

$$\Delta(G, G', \pi, \pi^n) = 1 - \frac{1}{|\pi|} \sum_i \left[\frac{d(G'_{\pi(i)}, G'_{\pi^n(i)})}{\sigma} \right]. \quad (16)$$

Here, the loss is small if the distance between the chosen match and the correct match is small.

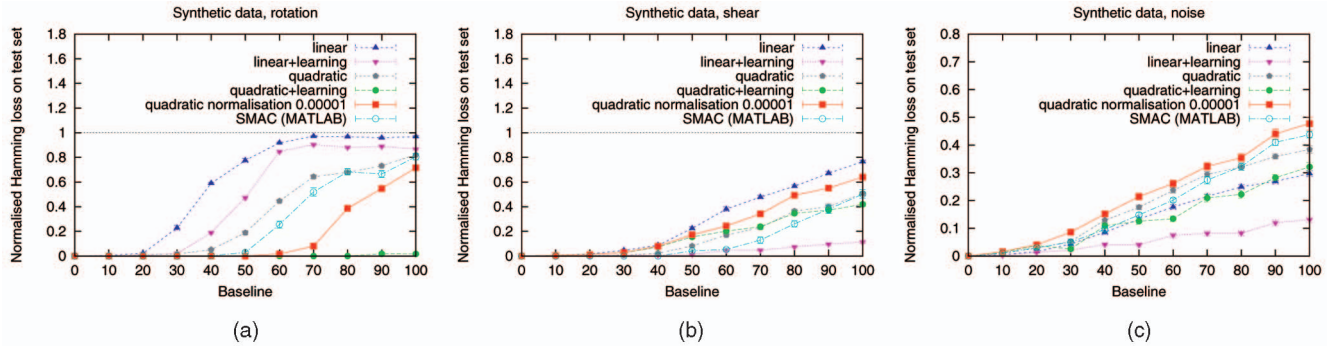


Fig. 5. Performance of linear and quadratic assignment under various transformations, both before and after learning: (a) rotation, (b) shear, and (c) noise.

Since we are interested in only a few of our landmarks, triangulating the graph is no longer meaningful. Hence, we present results only for linear assignment.

Fig. 2a shows the performance of our method as the baseline increases. In this case, the performance is non-monotonic as the subject moves in and out of view throughout the sequence. This sequence presents additional difficulties over the "house" data set, as we are subject to noise in the detected landmarks, and possibly in their labeling also. Nevertheless, learning outperforms nonlearning for all baselines. The weight vector (Fig. 2b) is heavily peaked about particular angular bins.

6.4 Bikes

For our final experiment, we used images from the Caltech 256 data set [50]. We chose to match images in the "touring bike" class, which contains 110 images of bicycles. Since the Shape Context features we are using are robust to only a small amount of rotation (and not to reflection), we only included images in this data set that were taken "side-on." Some of these were then reflected to ensure that each image had a consistent orientation (in total, 78 images remained). Again, the SUSAN corner detector was used to identify the landmarks in each scene; six points corresponding to the frame of the bicycle were identified in each frame (see Fig. 7a).

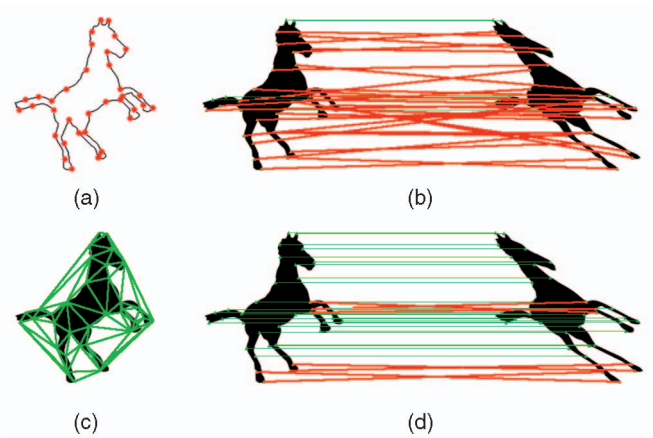


Fig. 6. An example match on the "shear" sequence, with a horizontal shear of 150 percent. (a) The point set used for our synthetic experiments. (b) The adjacency matrix for our template set. (c) Two frames matched using linear assignment (without learning, $\text{loss} = 29/35$). (d) The same match after learning ($\lambda = 0.1$, $\text{loss} = 7/35$).

Rather than matching all *pairs* of bicycles, we used a fixed template (G), and only varied the target. This is an easier problem than matching all pairs, but is realistic in many scenarios, such as image retrieval.

Table 2 shows the endpoint error of our method and gives further evidence of the improvement of learning over nonlearning. Fig. 7 shows a selection of data from our training set, as well as an example matching, with and without learning.

7 Conclusions and Discussion

We have shown how the compatibility functions for the graph matching problem can be estimated from labeled training examples, where a training input is a pair of graphs and a training output is a matching matrix. We use large-margin structured estimation techniques with column generation in order to solve the learning problem efficiently, despite the huge number of constraints in the optimization problem. We presented experimental results in three different settings, each of which revealed that the graph matching problem can be significantly improved by means of learning.

An interesting finding in this work has been that *linear* assignment with learning performs similarly to Graduated Assignment with bistochastic normalization, a state-of-the-art *quadratic* assignment relaxation algorithm. This suggests that in situations where speed is a major issue, linear assignment may be resurrected as a means for graph matching. In addition to that, if learning is introduced to Graduated Assignment itself, then the performance of graph matching improves significantly in accuracy and slightly in speed when compared to the best existing quadratic assignment relaxation [30].

There are many other situations in which learning a matching criterion can be useful. In multicamera settings, for example, when different cameras may be of different types and have different calibrations and viewpoints, it is reasonable to expect that the optimal compatibility functions will be different depending on which camera pair we consider. In surveillance applications, we should take advantage of the fact that much of the context does not change: the camera and the viewpoint are typically the same.

To summarize, by learning a matching criterion from previously labeled data, we are able to substantially improve the accuracy of graph matching algorithms.

Fig. 7. (a) Some of our training scenes. (b) A match from our test set. The top frame shows the points as matched without learning ($\text{loss} = 0.122$) and the bottom frame shows the match with learning ($\text{loss} = 0.060$). The points to be matched (left) and the correct match (right) are outlined in green; the inferred match is outlined in red.

ACKNOWLEDGMENTS

The authors thank Gideon Dror, James Petterson, and Choon Hui Teo for comments on the paper. They also thank Longbin Chen and Choon Hui Teo for code. NICTA is funded by the Australian Government's *Backing Australia's Ability* initiative and the Australian Research Council's *ICT Centre of Excellence* program.

REFERENCES

[1] T.S. Caetano, L. Cheng, Q.V. Le, and A.J. Smola, "Learning Graph Matching," *Proc. Int'l Conf. Computer Vision*, 2007.
[2] B. Schölkopf and A. Smola, *Learning with Kernels*. MIT Press, 2002.
[3] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge Univ. Press, 2004.

TABLE 2
Performance on the 'Bikes' Data Set

	Training	Validation	Testing
Loss	0.108 (0.006)	0.043 (0.007)	0.114 (0.004)
Loss (learning)	0.068 (0.004)	0.039 (0.006)	**0.077 (0.004)**

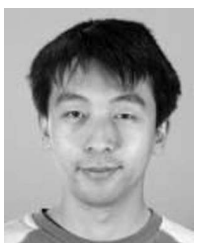
Results for the minimizer of the validation loss ($\lambda = 10{,}000$) are reported. Standard errors are in parentheses.

[4] S. Vishwanathan, K.M. Borgwardt, N. Schraudolph, and I.R. Kondor, "On Graph Kernels," *J. Machine Learning Research*, submitted, 2008.
[5] A. Torsello and E.R. Hancock, "Learning Shape-Classes Using a Mixture of Tree-Unions," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 6, pp. 954-967, June 2006.
[6] D. White and R.C. Wilson, "Spectral Generative Models for Graphs," *Proc. Int'l Conf. Image Analysis and Processing*, pp. 35-42, 2007.
[7] B. Bonev, F. Escolano, M. Lozano, P. Suau, M. Cazorla, and W. Aguilar, "Constellations and the Unsupervised Learning of Graphs," *Proc. Graph-Based Representations in Pattern Recognition*, pp. 340-350, 2007.
[8] M. Neuhaus and H. Bunke, "Automatic Learning of Cost Functions for Graph Edit Distance," *Information Sciences*, vol. 177, no. 1, pp. 239-247, 2007.
[9] M. Leordeanu and M. Hebert, "A Spectral Technique for Correspondence Problems Using Pairwise Constraints," *Proc. Int'l Conf. Computer Vision*, 2005.
[10] H. Wang and E.R. Hancock, "A Kernel View of Spectral Point Pattern Matching," *Proc. Int'l Workshops Advances in Structural and Syntactic Pattern Recognition and Statistical Techniques in Pattern Recognition*, pp. 361-369, 2004.
[11] L. Shapiro and J. Brady, "Feature-Based Correspondence—An Eigenvector Approach," *Image and Vision Computing*, vol. 10, pp. 283-288, 1992.
[12] M. Carcassoni and E.R. Hancock, "Spectral Correspondence for Point Pattern Matching," *Pattern Recognition*, vol. 36, pp. 193-204, 2003.
[13] T. Caelli and S. Kosinov, "An Eigenspace Projection Clustering Method for Inexact Graph Matching," *IEEE Trans. Pattern Analayis and Machine Intelligence*, vol. 26, no. 4, pp. 515-519, Apr. 2004.
[14] T.S. Caetano, T. Caelli, and D.A.C. Barone, "Graphical Models for Graph Matching," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, pp. 466-473, 2004.
[15] A. Rosenfeld and A.C. Kak, *Digital Picture Processing*. Academic Press, 1982.
[16] R.C. Wilson and E.R. Hancock, "Structural Matching by Discrete Relaxation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 6, pp. 634-648, June 1997.
[17] E. Hancock and R.C. Wilson, "Graph-Based Methods for Vision: A Yorkist Manifesto," *Proc. Int'l Workshops Advances in Structural and Syntactic Pattern Recognition and Statistical Techniques in Pattern Recognition*, pp. 31-46, 2002.
[18] W.J. Christmas, J. Kittler, and M. Petrou, "Structural Matching in Computer Vision Using Probabilistic Relaxation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 749-764, 1995.
[19] J.V. Kittler and E.R. Hancock, "Combining Evidence in Probabilistic Relaxation," *Int'l J. Pattern Recognition and Artificial Intelligence*, vol. 3, pp. 29-51, 1989.
[20] S.Z. Li, "A Markov Random Field Model for Object Matching Under Contextual Constraints," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, pp. 866-869, 1994.
[21] C. Schellewald, "Convex Mathematical Programs for Relational Matching of Object Views," PhD dissertation, Univ. of Mannhein, 2004.
[22] M. Pelillo, "Replicator Equations, Maximal Cliques, and Graph Isomorphism," *Neural Computation*, vol. 11, pp. 1933-1955, 1999.
[23] B.T. Messmer and H. Bunke, "A New Algorithm for Error-Tolerant Subgraph Isomorphism Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 5, pp. 493-503, May 1998.

[24] S. Gold and A. Rangarajan, "A Graduated Assignment Algorithm for Graph Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 4, pp. 377-388, Apr. 1996.
[25] M. van Wyk, T. Durrani, and B. van Wyk, "A RKHS Interpolator-Based Graph Matching Algorithm," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 7, pp. 988-995, July 2002.
[26] A. Rangarajan, A. Yuille, and E. Mjolsness, "Convergence Properties of the Softassign Quadratic Assignment Algorithm," *Neural Computation,* vol. 11, pp. 1455-1474, 1999.
[27] R. Sinkhorn, "A Relationship between Arbitrary Positive Matrices and Doubly Stochastic Matrices," *Annals Math. and Statistics,* vol. 35, pp. 876-879, 1964.
[28] M. Pelillo and M. Refice, "Learning Compatibility Coefficients for Relaxation Labeling Processes," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 16, no. 9, pp. 933-945, Sept. 1994.
[29] S. Lacoste-Julien, B. Taskar, D. Klein, and M. Jordan, "Word Alignment Via Quadratic Assignment," *Proc. North Am. Chapter Assoc. for Computational Linguistics-Human Language Technologies,* 2006.
[30] T. Cour, P. Srinivasan, and J. Shi, "Balanced Graph Matching," *Proc. Conf. Neural Information Processing Systems,* 2006.
[31] K. Anstreicher, "Recent Advances in the Solution of Quadratic Assignment Problems," *Math. Programming, Ser. B,* vol. 97, pp. 27-42, 2003.
[32] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity.* Dover Publications, July 1998.
[33] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large Margin Methods for Structured and Interdependent Output Variables," *J. Machine Learning Research,* vol. 6, pp. 1453-1484, 2005.
[34] R.K. Ahuja and J.B. Orlin, "Inverse Optimization," *Operations Research,* vol. 49, no. 5, pp. 771-783, 2001.
[35] A. Smola, S.V.N. Vishwanathan, and Q. Le, "Bundle Methods for Machine Learning," *Proc. Advances in Neural Information Processing Systems 20,* J. Platt, D. Koller, Y. Singer, and S. Roweis, eds., pp. 1377-1384, 2008.
[36] B. Taskar, C. Guestrin, and D. Koller, "Max-Margin Markov Networks" *Proc. Advances in Neural Information Processing Systems 16,* S. Thrun, L. Saul, and B. Schölkopf, eds., 2004.
[37] T. Joachims, "Training Linear SVMs in Linear Time," *Proc. Knowledge Discovery and Data Mining,* 2006.
[38] C. Teo, Q. Le, A. Smola, and S. Vishwanathan, "A Scalable Modular Convex Solver for Regularized Risk Minimization," *Proc. Knowledge Discovery and Data Mining,* 2007.
[39] R. Jonker and A. Volgenant, "A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems," *Computing,* vol. 38, no. 4, pp. 325-340, 1987.
[40] T. Finley and T. Joachims, "Training Structural SVMs when Exact Inference Is Intractable," *Proc. Int'l Conf. Machine Learning,* 2008.
[41] S. Belongie, J. Malik, and J. Puzicha, "Shape Matching and Object Recognition Using Shape Contexts," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 4, pp. 509-521, Apr. 2002.
[42] CMU "house" data set, http://vasc.ri.cmu.edu/idb/html/motion/house/index.html, 2009.
[43] T.S. Caetano, T. Caelli, D. Schuurmans, and D.A.C. Barone, "Graphical Models and Point Pattern Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 10, pp. 1646-1663, Oct. 2006.
[44] A.M. Bronstein, M.M. Bronstein, and R. Kimmel, *Numerical Geometry of Non-Rigid Shapes.* Springer, 2007.
[45] A.M. Bronstein, M.M. Bronstein, A.M. Bruckstein, and R. Kimmel, "Analysis of Two-Dimensional Non-Rigid Shapes," *Int'l J. Computer Vision,* 2007.
[46] Mythological Creatures 2D database, http://tosca.cs.technion.ac.il, 2009.
[47] SAMPL motion data set, http://sampl.ece.ohio-state.edu/database.htm, 2009.
[48] S. Smith, "A New Class of Corner Finder," *Proc. British Machine Vision Conf.,* pp. 139-148, 1992.
[49] S. Smith, "Flexible Filter Neighbourhood Designation," *Proc. Int'l Conf. Pattern Recognition,* pp. 206-212, 1996.
[50] G. Griffin, A. Holub, and P. Perona, "Caltech-256 Object Category Data Set," Technical Report 7694, California Institute of Technology, http://authors.library.caltech.edu/7694, 2007.

Tibério S. Caetano received the BSc degree in electrical engineering (with research in physics) and the PhD degree in computer science, (with highest distinction) from the Universidade Federal do Rio Grande do Sul (UFRGS), Brazil. The research part of the PhD program was undertaken at the Computing Science Department at the University of Alberta, Canada. He held a postdoctoral research position at the Alberta Ingenuity Centre for Machine Learning and is currently a senior researcher with the Statistical Machine Learning Group at NICTA. He is also an adjunct research fellow at the Research School of Information Sciences and Engineering, Australian National University. His research interests include pattern recognition, machine learning, and computer vision.

Julian J. McAuley received the BSc degree in mathematics and the BEng degree in software engineering (with first-class honors and the university medal) from the University of New South Wales in 2007. He is currently undertaking a PhD at the Australian National University, under the supervision of Tibério Caetano. He is a student member of the IEEE.



Li Cheng received the PhD degree from the Department of Computing Science, University of Alberta, Canada, in 2004. He worked as a research associate in the same department at the University of Alberta, and then worked as a researcher with the Machine Learning group, NICTA, Australia. He is now with TTI-Chicago. His research interests are mainly on image and video understanding, computer vision, and machine learning. He is a member of the IEEE.

Quoc V. Le is a PhD student at Stanford University's AI Lab, under the supervision of Andrew Ng. He has also studied with the Statistical Machine Learning Group at NICTA, and the Max Planck Institute for Biological Cybernetics.

Alex J. Smola received the master's degree at the University of Technology, Munich, and the doctoral degree in computer science at the University of Technology Berlin. Until 1999, he was a researcher at the IDA Group of the GMD Institute for Software Engineering and Computer Architecture in Berlin (now part of the Fraunhofer Geselschaft). He worked as a researcher and group leader at the Research School for Information Sciences and Engineering of the Australian National University. From 2004 until 2008, he worked as a senior principal researcher and the program leader of the Statistical Machine Learning Group at NICTA. He is now at Yahoo! Research.