Available online at www.sciencedirect.com
SCIENCE DIRECT®
Pattern Recognition 39 (2006) 552–561
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/patcog

Approximating the problem, not the solution: An alternative view of point set matching

Tibério S. Caetano*, Terry Caelli

National ICT Australia, Canberra ACT 0200, Australia

Received 5 October 2005

Abstract

This work discusses the issue of approximation in point set matching. In general, one may have two classes of approximations when tackling a matching problem: (1) an algorithmic approximation which consists in using suboptimal procedures to infer the assignment, and (2), a representational approximation which involves a simplified and suboptimal model for the original data. Matching techniques have typically relied on the first approach by retaining the complete model and using suboptimal techniques to solve it. In this paper, we show how a technique based on using exact inference in simple Graphical Models, an instance of the second class, can significantly outperform instances of techniques from the first class. We experimentally compare this method with well-known Spectral and Relaxation methods, which are exemplars of the first class. We have performed experiments with synthetic and real-world data sets which reveal significant performance improvement in a wide operating range.

Keywords: Graphical models; Point pattern matching; Graph matching; Markov random fields

1. Introduction

The point set matching problem consists in finding correspondences between two point sets, which may be one-to-one or also many-to-one [1,2], and arises in a variety of real-world vision tasks such as stereo vision, registration, model-based object recognition and the like. In any real vision problem we are faced with *inexact* point set matching, or matching under structural corruption, which is known to be an NP-hard problem [3]. As a result, one must rely on approximate techniques that derive the "best" assignment in some suboptimal sense. Major representatives of proposed techniques are *spectral* [4,5] and *relaxation labeling* methods [6,9]. These families of techniques consist in encoding all the available information into a complete data model (such as the complete distance matrix) and subsequently using approximate algorithms to derive the assignment. Several limitations have been reported with respect to spectral methods when structural corruption is present and with respect to relaxation methods when matching large point sets [1,4,7,8].

This paper shows how a different approach leads to an alternative method that overcomes many of the limitations existent in some spectral and relaxation approaches. Essentially, instead of using a complete data model and an approximate algorithm, we use the opposite. By using the point matching principle introduced in Ref. [2], we model the structure of points by several sparse representations that deliberately disregard particular subsets of relational information that is actually available. In other words, we *approximate the data model*. The reason for this becomes clear in the next step: by taking advantage of this sparsity, we are able to apply *optimal algorithms* to derive the best assignment (what is not possible with the complete data model).

For performance evaluation, we have conducted experiments with synthetic and real-world data sets, where we compared the results of our approximate models with those obtained with traditional versions of spectral and relaxation methods, namely the Shapiro and Brady spectral method [5] and the Rosenfeld et al. relaxation method [9,10]. Results indicate that the accuracy obtained with the described

* Corresponding author. Tel.: +61 2 6125 1384.
E-mail address: Tiberio.Caetano@nicta.com.au (T.S. Caetano).

models significantly exceeds that obtained by the alternative techniques in a wide operating range, either under structural corruption by point position jitter or under augmentation of the point set sizes.

2. Problem definition

We consider the problem, in $\mathbb{R}^2$, of finding the subset of an S-sized point set (the *codomain pattern*) that best matches another point set (the *domain pattern*) having T points, where $T \leqslant S$. There may or may not exist distortions due to noise, but if there are, we assume no prior knowledge of the type of noise. We restrict the matching to be invariant up to isometries, so we do not consider scaling. The only constraint enforced in the mapping is that it must be a total function: every point in the domain pattern must map to one point in the codomain pattern (but the opposite may not hold).

3. Benchmarks

In order to evaluate the performance of our models, we perform experimental comparison with some standard solutions available in the literature. Specifically, we will focus on techniques for point pattern matching that take into account *exclusively* the pairwise distances between points as features. This is so because of two reasons. First, as mentioned above, we are considering the isometric version of the problem. Second, and of special importance, is the fact that in order to accomplish a fair comparison we must ensure that the different techniques are not provided with different prior information. Of course specialized techniques can be designed which take advantage of constraints that may be known to appear in specific application domains, but they are not interesting for our purposes since our aim consists in comparing different "principles" of solution in the general case, not in a specific application.

3.1. Spectral methods

One class of solution to graph matching problems in general, and point pattern matching in particular, is that based on the spectral analysis of the adjacency matrix or alternatively of the Laplacian matrix of a given point set [1,4,5,11,12]. The basic idea common to most methods consists in performing spectral analysis in each of the adjacency matrices (or Laplacian matrices) and comparing the eigenvalues/eigenvectors using some matching criteria. An early and standard representative of these techniques is the Shapiro–Brady (SB) method [5]. In this method, the adjacency matrix A is computed by exponentiating the Euclidean distance matrix so as to obtain entries that are higher for closer points:

$$A_{ij} = \exp\left(-d_{ij}/2\sigma^2\right), \tag{1}$$

where d_{ij} is the Euclidean distance between points i and j, and σ is usually heuristically chosen. Then, singular value decomposition is performed

$$A = VDV^{\mathrm{T}}, \tag{2}$$

where $V = (F_1, \ldots, F_m)^{\mathrm{T}}$. The rows of V (F_i's), which correspond to the eigenvectors of A, are seen as feature vectors. The SVD is performed both for the adjacency matrix of the first pattern (A) and that of the second pattern (A'), so that we have also $V' = (F_1', \ldots, F_m')^{\mathrm{T}}$. The final operation then consists in comparing the feature vectors for the different patterns,

$$Z_{ij} = \left\| F_i - F_j' \right\|_2^2. \tag{3}$$

Corresponding feature vectors in the two patterns may appear with opposite orientation, so a sign correction stage must be implemented, and the simplest way of doing that is using a majority rule [5].

Clearly, for different pattern sizes the spaces will have different dimensionality, and the algorithm, as originally proposed, is not suited for this case. Although very heuristic, this algorithm is extremely fast for standards of matching algorithms ($O(T^3)$, where T is the size of the domain and codomain patterns—they must be the same, otherwise the dimensions of F and F' are different and the last step, Eq. (3), cannot be accomplished). Also, it encodes *all the pairwise distances* in the point patterns. In the wording of this paper, it uses an "optimal representation", since the amount of information that is made available to this algorithm is complete—under the assumption of handling isometric invariances only. However, this approach clearly uses an "approximate algorithm" for performing the matching, since for example different matrices can have the same spectrum.

3.2. Relaxation methods

In probabilistic relaxation labeling (PRL), each point in the domain pattern is seen as a "site" and each one in the codomain pattern is seen as a possible "label" for each site. The fundamental idea of PRL is to update the probability of assigning label k to site i by taking into account the compatibility of its neighboring assignments. The global compatibility or support that the neighbors of i pass to i at iteration r is then somehow aggregated over versions of individual compatibilities c at iteration r. This is the common principle in many versions of PRL [6,9,10,13,14], with the basic difference being in how the support function is designed. In this paper we focus on a standard version of PRL described in Ref. [9], whose support function is given by

$$q_{ik}^{(r)} = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \sum_{l=1}^{S} c(i,k;j,l) p_{jl}^{(r)}, \tag{4}$$

where $\mathcal{N}_i$ is the set of neighbors of vertex i, q_{ik} is the support that the assignment $i \to k$ receives from the neighbors of i

and p_{jl} is the probability of the assignment $j \to l$. For the particular case where the graph is fully connected and all vertices are neighbors we have

$$q_{ik}^{(r)} = \frac{1}{T-1} \sum_{j \neq i} \sum_{l=1}^{S} c(i,k;j,l) p_{jl}^{(r)}. \tag{5}$$

The probability that i should be assigned to k is then updated based on the support $q_{ik}^{(r)}$

$$p_{ik}^{(r+1)} = \frac{p_{ik}^{(r)} q_{ik}^{(r)}}{\sum_{k=1}^{S} p_{ik}^{(r)} q_{ik}^{(r)}}. \tag{6}$$

The algorithm is thus an iterative one. At each iteration r, the probabilities $p_{ik}^{(r)}$, for all i and k, are updated (what can indeed be done in parallel). The algorithm stops when a fixed maximum number of iterations has been reached or when it reaches a fixed point. At this stage, we simply assign to each site i the label k^* such that $p_{ik^*} > p_{ik}, \forall k \neq k^*$.

The algorithm is clearly an heuristic optimization procedure. First, it is iterative, and there is no principled way of estimating how many iterations should be used for a particular problem size. Second, it is dependent on the initialization of the probabilities p_{ik}. If we are lucky to initialize the probabilities in such a way that the maximum of the matching similarity is close to the initialized solution, the algorithm may converge in a reasonable amount of iterations. On the other hand, if this is not the case and the complexity of the problem is high (if T and/or S are not small), the amount of iterations needed in order to find the optimal solution may be so large that it is simply not feasible to wait the required time. Third, even if the algorithm converges in a reasonable amount of iterations, the fixed point is only guaranteed to be a *local* optimum [10]. The computational complexity of the algorithm described above is $O(T^2S^3)$ per iteration.

Note that, although this is an approximate algorithm, like the SB spectral method, it also can use *all the pairwise distances* as features, so it qualifies as using an "optimal representation", in our parlance. In the next section we describe an alternative approach, which consists in inverting this reasoning: we will present suboptimal representations for the point pattern matching problem; however, in these suboptimal representations, we will see that *optimal algorithms* can run in polynomial time.

4. Approximating the problem

Here we show how the method first introduced in Ref. [2] can be used to generate, in a systematic way, a number of approximate models for the isometric point pattern matching problem. First, we review the method by describing how the problem can be written as a weighted graph matching problem. Then we present the idea of solving approximately the weighted graph matching problem by using sparse Graphical Models. In the sequel, we describe how such Graphical Models are designed (the potential functions and the edge connectivity).

4.1. Point pattern matching as weighted graph matching

Let the cardinalities of the domain and codomain pattern sets be denoted, respectively, by T and S. The relative Euclidean distance between a pair $\{d_{i_1}, d_{i_2}\}$ of points in the domain pattern is denoted as $y^d_{i_1 i_2}$. Analogously for the codomain pattern, we have that $y^c_{k_1 k_2}$ is the distance between points c_{k_1} and c_{k_2}. The first idea in this point pattern matching technique is the following: if the Euclidean distance matrices (EDMs) of two point sets are the same under some permutation, the point sets are isometric [15]. As a result, isometry can be tested by comparing the first EDM with all permutations of the second EDM. The problem of EDM comparison can be cast as one of weighted graph matching. Each point d_i in the domain is associated with a vertex of a graph G_d, and each point c_k in the codomain is associated with a vertex of a graph G_c. The weight between a pair of vertices is the entry in the EDM for the corresponding pair, so the graphs are in principle fully connected. As a result, point pattern matching can be then posed as a weighted graph matching problem, where the purpose consists in finding the map f that minimizes the following function:

$$U_T(f) = \sum_{i=1}^{T} \sum_{j=1}^{T} \mathscr{D}\left(y^d_{ij}, y^c_{f(i)f(j)}\right). \tag{7}$$

Here $U_T(f)$ is the "total" cost to be minimized (the reason for calling it "total" will be clear later), and $\mathscr{D}(\cdot,\cdot)$ is some *dissimilarity measure* between distances. Note that the arguments of $\mathscr{D}(\cdot,\cdot)$ represent the entries in the EDMs under the permutation induced by f.

Weighted graph matching for general fully connected graphs is NP-hard [8]. In the following we show how this problem can be equivalently written as one of inference over a fully connected Graphical Model. Then we introduce the idea of approximating the fully connected model with sparse models where exact inference is feasible.

4.2. Weighted graph matching as inference in graphical models

Following Ref. [2], the model formulation consists, initially, in defining each of the T vertices in G_d as a random variable that can assume S possible values (discrete states), corresponding to the vertices in G_c. Note that in this formulation the solution to the problem (the best match) corresponds to finding the most likely (the best) realization of the set of random variables (or MAP estimate of the entire random field). The challenge then consists in (i) designing a probability distribution whose MAP solution minimizes Eq. (7) and (ii) computing this MAP solution efficiently. It is not difficult to show that a fully connected pairwise

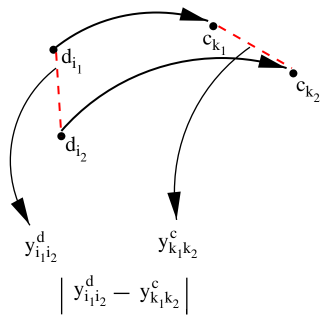

Fig. 1. An example of a pairwise mapping. An appropriate potential function should penalize more severely mappings for which $|y^d_{i_1 i_2} - y^c_{k_1 k_2}|$ is higher.

Markov random field[1] with local potential functions given by $V_{ij} = -\log \psi_{ij}(x_i, x_j) = \mathscr{D}(y^d_{ij}, y^c_{f(i)f(j)})$ does minimize $U_T(f)$, so that (i) is easy to solve:

$$p(f) = \frac{1}{Z} \prod_{(i,j)} \psi_{ij}\left(X_i = x_{f(i)}, X_j = x_{f(j)}\right) \tag{8}$$

$$= \frac{1}{Z} \exp\left(-\sum_{(i,j)} V_{ij}\left(X_i = x_{f(i)}, X_j = x_{f(j)}\right)\right)$$

$$= \frac{1}{Z} \exp\left(-\sum_{i=1}^{T}\sum_{j=1}^{T} \mathscr{D}\left(y^d_{ij}, y^c_{f(i)f(j)}\right)\right) \tag{9}$$

$$\propto \exp(-U_T(f)), \tag{10}$$

and maximizing $p(f)$ becomes equivalent to minimizing $U_T(f)$. However, in fully connected Graphical Models the problem of exact MAP computation is intractable, which gives us no hint to solve (ii). The basic idea presented in this paper consists in *approximating* the cost function U_T (thus modifying the solution for (i) actually) with simpler cost functions, defined over subsets of pairwise distances which induce tractable graphs, where *exact* inference can be performed in polynomial time.

In order to further specify such sparse Graphical Models, it is necessary to define (i) the potential functions and (ii) the edge connectivity of the models [16]. Next we describe the types of potential functions that we consider.

4.3. Potential functions

Fig. 1 illustrates a pairwise map and a possible measure which is relevant in order to construct the potential functions ($|y^d_{i_1 i_2} - y^c_{k_1 k_2}|$).

Since each node in the domain graph can map to S different nodes in the codomain graph, each pair of nodes can map to S^2 different pairs. Fig. 2 illustrates the nuclear structure of our model: a pairwise clique, where each random variable (node) represents a point in the domain graph which in turn can assume a set of S possible realizations (which themselves correspond to points in the codomain graph).

[1] In this paper, Markov random fields and Graphical Models are synonyms, since we consider exclusively *undirected* Graphical Models.

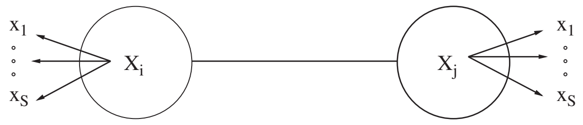

Fig. 2. The nuclear structure of the graphical model.

The sample space for this clique has S^2 elements, corresponding to all possible combinations that a pair of points in the domain graph can map to in the codomain graph. A potential function is a function that associates a positive real number to each element of the sample space. In our case, the only requirement is that the potential function must obey the general condition that edge similarity is indirectly proportional to the difference of edge lengths, as illustrated in Fig. 1.

Formally, we can specify the potential function, for each pair $\{X_i, X_j\}$ in G_d, as

$$\psi_{ij} = \psi_{ij}(X_i, X_j)$$

$$= \frac{1}{Z}\begin{pmatrix} \mathscr{S}(y^d_{ij}, y^c_{11}) & \dots & \mathscr{S}(y^d_{ij}, y^c_{1S}) \\ \vdots & \ddots & \vdots \\ \mathscr{S}(y^d_{ij}, y^c_{S1}) & \dots & \mathscr{S}(y^d_{ij}, y^c_{SS}) \end{pmatrix}, \tag{11}$$

where Z is a normalization constant that equals the sum of all elements in the matrix, in order to keep ψ_{ij} compatible with a probability distribution. $\mathscr{S}$ is a similarity function that measures the compatibility of the two arguments. We use here the Gaussian function,

$$\mathscr{S}(y^d_{ij}, y^c_{kl}) = \exp\left[-\frac{1}{2\sigma^2}\left|y^d_{ij} - y^c_{kl}\right|^2\right]. \tag{12}$$

Such a proximity measure is needed in order to model the uncertainty due to the presence of noise. Obviously, its maximal value must be attained at zero noise ($y^d_{ij} = y^c_{kl}$).

Having specified the potential functions, the edge connectivity of the graphical model remains to be determined: which nodes will be neighbors in the model?

4.4. Edge connectivity

As mentioned previously, our approach consists in approximating the original cost function U_T in such a way that the induced Graphical Model becomes tractable. By "tractable" we simply mean a model where *exact* probabilistic inference (in our case, MAP computation) can be performed in polynomial time.

In order to select such models, one must be aware of which topologies will lead to feasible inference. The algorithm for exact inference in arbitrary Graphical Models, known as the

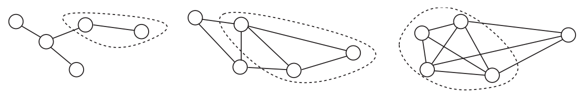

Fig. 3. Examples of k-trees. Left: a 1-tree (typically called a "tree"). Center: a 2-tree. Right: a 3-tree. The nodes in circled regions are examples of maximal cliques (which have size 2, 3 and 4, respectively).

Junction Tree algorithm, has exponential complexity which depends on the tree-width of the graph (precisely, it is equal to the tree-width plus one, which is the size of the maximal clique of an optimal triangulation[2] of the graph). If the tree-width is bounded (independent of the number of nodes), the exponent will be fixed and the computational complexity will be polynomial [16].

Given this fact, our strategy becomes a simple one: choosing cost functions U_{partial} that take into account only relative distances that conform to the edge connectivity of a graph which is *already* triangulated and has small tree-width (by choosing a graph which is already triangulated we avoid the issue of finding an optimal triangulation). In our case, we will restrict ourselves to the class of k-tree graphs, which have tree-width of size k. A k-tree is a graph that one obtains by doing the following procedure: (i), start with the complete graph with k vertices; (ii), add a new node and connect it with (and only with) k existent nodes that form a complete subgraph (a k-clique); (iii), repeat (ii) as many times as wanted.

Following this definition, we conclude that a 1-tree is what is generally simply called a "tree". Fig. 3 shows examples of a 1-tree, a 2-tree and a 3-tree. In this paper, these are the types of k-trees that will be considered, which will induce fixed exponential complexity of 2, 3 and 4, respectively.

Accordingly, the resulting cost function that will be actually minimized will be

$$U_{G_d^{kt}}(f) = \sum_{i,j|(i,j)\in\mathscr{E}_d^{kt}} \mathscr{D}\left(y_{ij}^d, y_{f(i)f(j)}^c\right), \qquad (13)$$

where $G_d^{kt} = (\mathscr{V}, \mathscr{E}_d^{kt})$ is the subgraph of G_d induced by the edge set $\mathscr{E}_d^{kt}$ (kt stands for k-tree). What differs for each model is precisely $\mathscr{E}_d^{kt}$: for a 1-tree model it consists of a set of edges forming a 1-tree, and similarly for 2-tree and 3-tree models.

Fig. 4 shows examples of Probabilistic Graphical Models for point pattern matching that have topologies given by k-trees, for different values of k.

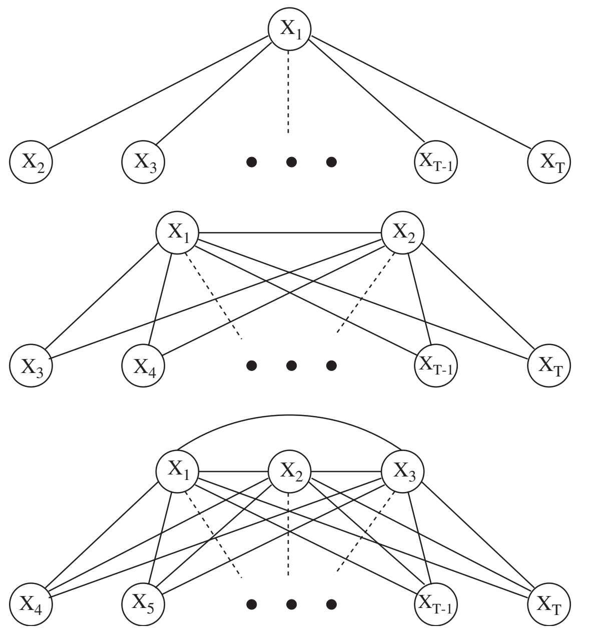

Fig. 4. Graphical Models for approximate point matching. Top: a 1-tree; Center: a 2-tree; Bottom: a 3-tree. Each of the T nodes is a random variable representing a point in the domain graph. Each variable can assume S possible realizations, corresponding to points in the codomain graph.

Note that these models deliberately disregard much of the relative distance information that is actually available. This is necessary in order to assure tractability.[3]

Given the edge connectivity and the pairwise potential functions, the model is defined. The last step then consists in inferring what is the most likely joint realization of all the random variables for the given edge connectivity and set of potentials of the model. This is precisely the MAP inference problem in this model, whose solution represents the best assignment in the point set matching task and is described in what follows.

4.5. MAP computation

The Junction Tree framework consists in a set of generalized dynamic programming algorithms for *exact* inference in arbitrary graphical models [16,17]. Here we use HUGIN, an algorithm from this framework, in order to find the optimal MAP estimate for the models in Fig. 4. A Junction Tree of a graph is another graph where (i) the nodes correspond to the maximal cliques of the former graph and (ii) the edge connectivity is such that the *running intersection property* is

[2] A triangulation of a graph is a graph obtained by adding edges to the original graph such that every cycle of length greater than 3 in the new graph has a chord (an edge connecting non-consecutive nodes in the cycle). An optimal triangulation is a triangulation whose maximal clique size is minimum among all possible triangulations. Finding an optimal triangulation for arbitrary graphs is an NP-complete problem [16].

[3] In a companion paper, we will show however that k-trees are in some sense "optimal"—their MAP solutions are the same as those of the fully connected model—for matching problems in Euclidean spaces of dimension $k-1$ when there is no jitter. For example, for matching in $\mathbb{R}^2$, a 3-tree model has the same MAP solutions as the fully connected model in the noiseless case.

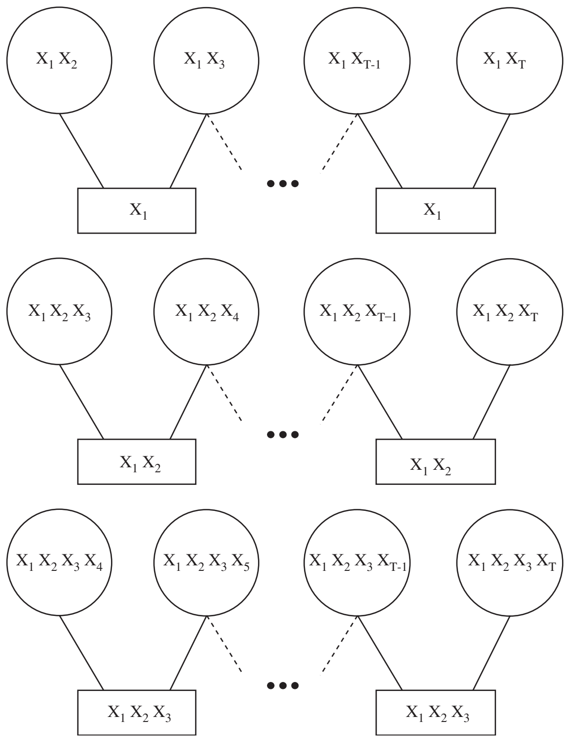

Fig. 5. The Junction Trees obtained from the models in Fig. 4.

satisfied. This property states that all the nodes in the path between any two nodes in the Junction Tree must contain the intersection of these two nodes. It is known that the condition for the existence of a Junction Tree is that the graph must be triangulated (chordal) [17].

Fig. 5 shows Junction Trees obtained from the models in Fig. 4. The nodes of the Junction Trees are denoted by circles in which are listed the variables of the original graph that correspond to the respective maximal cliques. The rectangles are the so-called *separators*, that contain the intersection of the nodes to which they are linked. Both the nodes and the separators are endowed with "clique potentials", and the optimization process consists in updating these potentials, as explained below.

The algorithm essentially works in two steps: initialization and message-passing. During initialization, the clique potential of each separator (Φ) is set to unity and the clique potential of each node (Ψ) is introduced (see Section 4.3). These last clique potentials are assembled as an element-by-element product of the pairwise potentials (see Eq. (11)) in the respective clique. For example, for the 3-tree model, $\Psi(x_i, x_j, x_k, x_l) = \psi(x_i, x_j)\psi(x_i, x_k)\psi(x_i, x_l)$. Note that in our case each potential is a table (an n-dimensional table, where n is the number of variables in the node or separator).

The second step is the message-passing scheme, which involves a transfer of information between two nodes V and W in a systematic way until every pair of nodes in the Junction Tree has participated in the process [16]. This operation is defined by the following equations:

$$\Phi_S^* = \max_{V \setminus S} \Psi_V \tag{14}$$

and

$$\Psi_W^* = \frac{\Phi_S^*}{\Phi_S} \Psi_W, \tag{15}$$

where we used standard notation for the current and updated ($*$) versions of the separator potentials (Φ) and the clique potentials (Ψ). The first equation is a maximization over all subconfigurations in Ψ_V that do not involve the variables which are common to Φ_S and Ψ_V. The second is simply a normalization step necessary to keep Ψ_W consistent with the updated version of Φ_S (division and multiplication between tables are performed element-by-element). The above potential update rules must respect the following protocol: a node V can only send a message to a node W when it has already received messages from all its other neighbors. If this protocol is respected and the equations are applied until all clique nodes have been updated, the algorithm assures that the resulting potential in each node and separator of the Junction Tree is proportional to the (global) maximum a posteriori probability distribution of the set of enclosed variables [16]. The constant of proportionality is guaranteed to be the same for every node, what implies that the mode of the local potentials will correspond to the MAP estimate. In our particular case, we need the maximum probability for each variable, what can be obtained by maximizing out the remaining three variables in each of the nodes. The indexes for which the final potentials are maximum are considered the vertices in G_c to which the corresponding vertices in G_d must be assigned. The computational complexity of the Junction Tree algorithm is $O(TS^{k+1})$, where $k+1$ is the size of the maximal clique of the graph (the number of variables in the largest node of the Junction Tree).

5. Experiments and results

We have carried out two sets of experiments, one with synthetic point sets and another with real-world data. In both of them, we compare results of the Junction Tree algorithm in 1-tree, 2-tree and 3-tree models (which we denote respectively by JT2, JT3 and JT4 due to the size of the maximal clique in each of these models) with those of probabilistic relaxation labeling, as described in Ref. [9] (denoted as PRL) and those of the spectral method of Shapiro and Brady [5] (denoted as SB). These last two methods encode all the pairwise distances in their model representation, whereas our method only encodes those distances that correspond to the edge connectivity of the given Graphical Model (1-tree, 2-tree or 3-tree). On the other hand, our approach uses a two-pass, dynamic programming procedure which is non-iterative and optimal, whereas the other two are based on approximate and heuristic algorithms. Results show how these

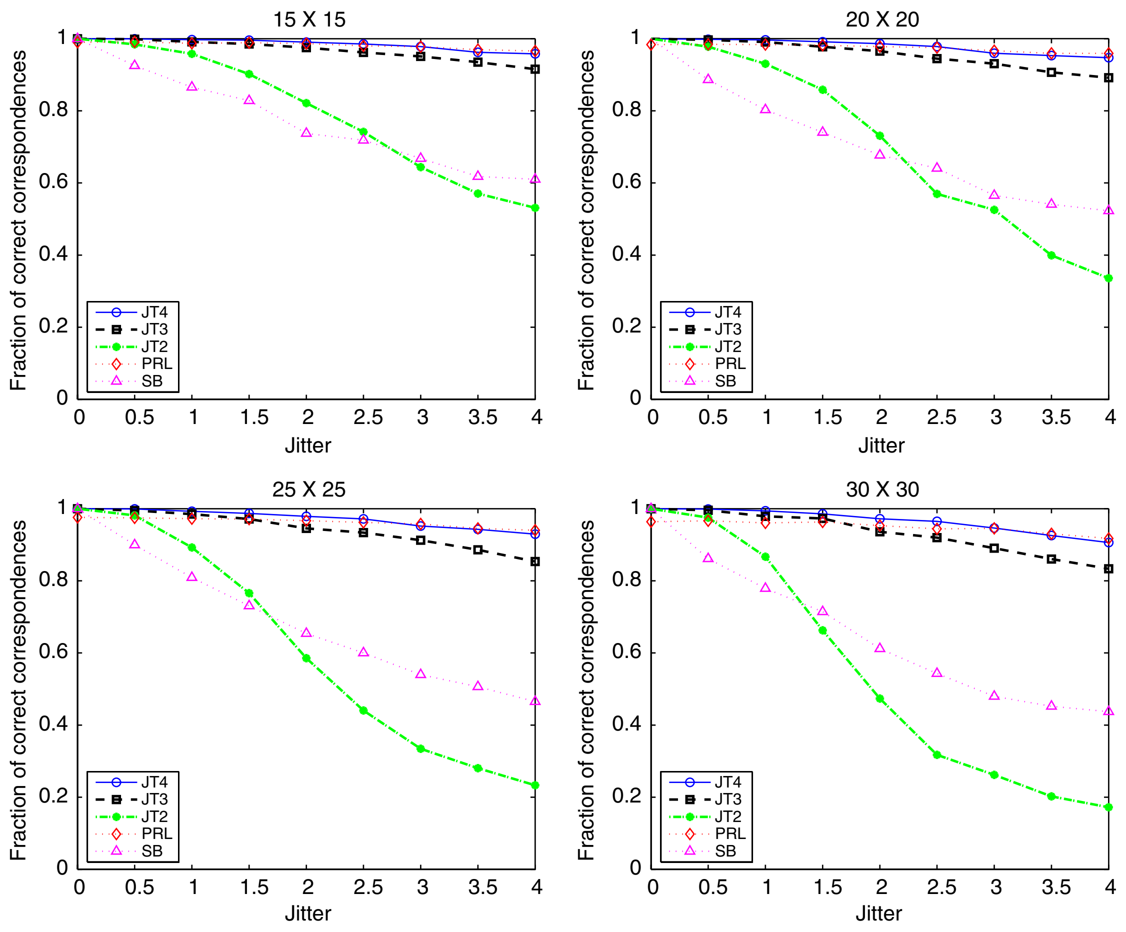

Fig. 6. Performances of JT4, JT3, JT2, PRL and SB when jitter increases, for various sizes of T ($S = T$). For JT methods and PRL, the Gaussian kernel (Eq. (12)) is used, with $\sigma = 0.4$. For the SB method, $\sigma = 0.4$.

different approximation principles affect accuracy in point set matching.

5.1. Synthetic data

In the experiments with synthetic data, we generated random points according to a bivariate uniform distribution in the interval $x = [0, 1]$, $y = [0, 1]$. The k-trees used where precisely given by the edge connectivity of the models shown in Fig. 4, but the assignment of which point in the domain pattern corresponds to which node in the model was random. The number of iterations in PRL was set to 200, which is much higher than in many typical implementations [6]. We performed two experiments, one with equal point set sizes (for various sizes) and varying jitter, and the other with fixed jitter (for various jitter levels) and increasing size of the codomain pattern.

In the first experiment, we matched point sets of sizes (T, S) equal to (15, 15), (20, 20), (25, 25) and (30, 30), and analyzed the performance robustness with respect to varying position jitter. Results over 200 trials are shown in Fig. 6.

In the second experiment, we held constant the size of the domain point set (10 nodes) and varied the size of the codomain point set (from 10 to 35 nodes in steps of 5). This was performed for several different levels of noise. In this experiment, we only compared JT4, JT3, JT2 and PRL, since SB is not suited for graphs with different sizes. Results over 200 trials are shown in Fig. 7.

5.2. Real-world data

In the real-world experiments, we performed comparisons of the algorithms using the CMU “house” sequence, available in Ref. [18]. Fig. 8 shows examples of the images contained in the dataset. Like in the synthetic experiments, the k-trees follow the edge connectivities given by Fig. 4 and the selection of which points correspond to which nodes was random. The number of iterations in PRL was also set to 200.

In total, 30 landmark points were manually marked in each of the 111 images in the database. Then we run the five algorithms (JT4, JT3, JT2, PRL and SB) in all pairs separated by 10, 20, . . . , 100 frames. This was done for the following sizes (T, S): (15, 30), (20, 30), (25, 30) and finally

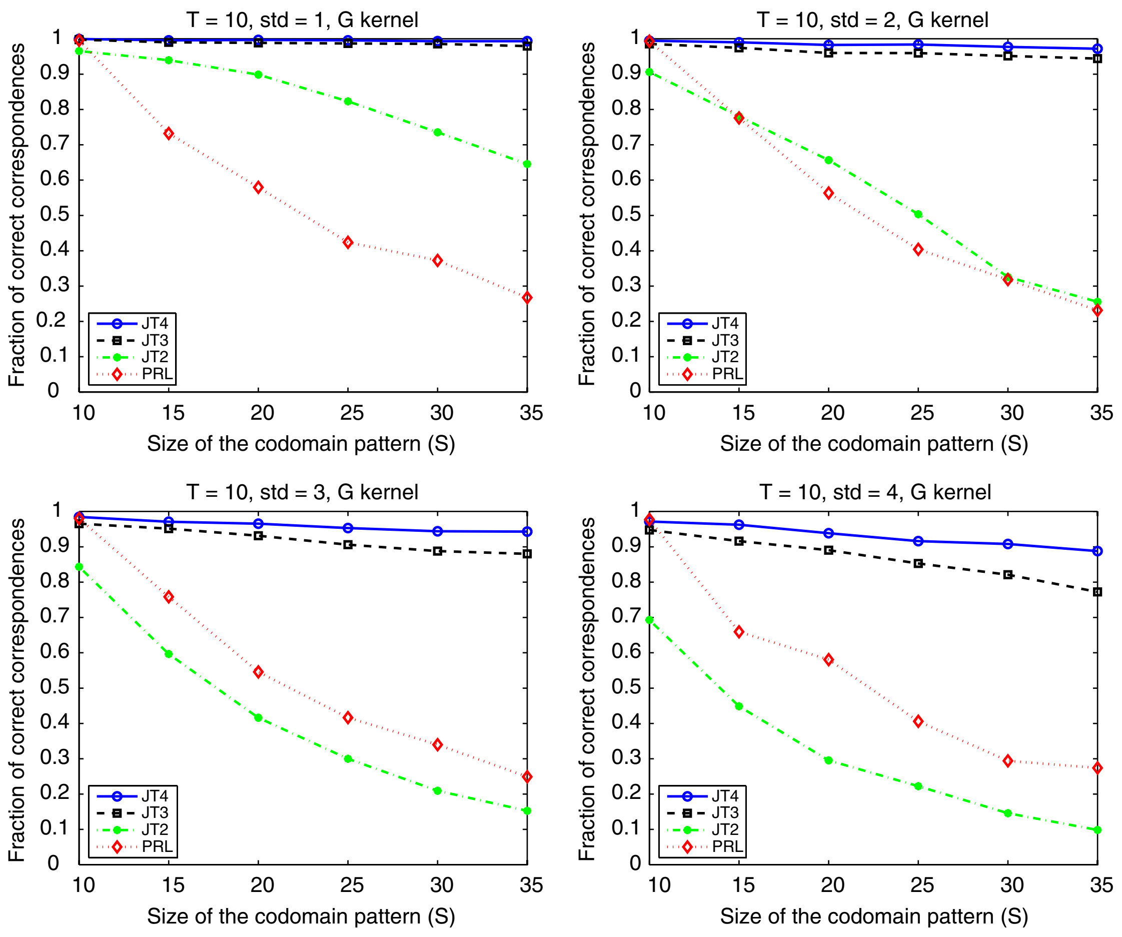

Fig. 7. Performances of JT4, JT3, JT2 and PRL when the size of the codomain pattern increases, for various levels of jitter (std). The Gaussian kernel (Eq. (12)) is used, with $\sigma = 0.4$.

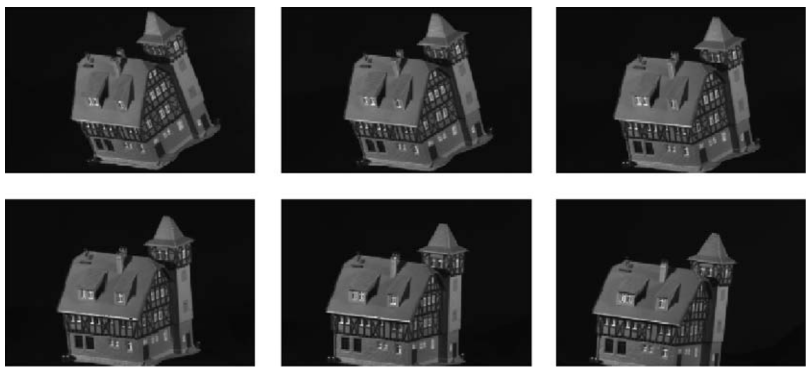

Fig. 8. Images from the CMU house data set (from left to right and top to bottom: images 1, 21, 41, 61, 81 and 101).

(30, 30) (SB was only run for (30, 30)). The average value for each separation in each of these four experiments was then recorded. Fig. 9 presents the results obtained.

6. Discussion

The synthetic experiments show that, when matching point sets of equal sizes, the Junction Tree models do not improve over PRL. However, it is interesting to note that even the simplest model (JT2), which basically takes into account only $T - 1$ relative distances in the domain pattern, instead of the whole set of $T(T-1)/2$ distances used by the SB method, improves over SB in the range of low jitter, as seen from Fig. 6. This is a clear example that a method that oversimplifies the problem *but* solves optimally the resulting problem may outperform another which tries to solve the original problem with an heuristic procedure. In Fig. 7, the

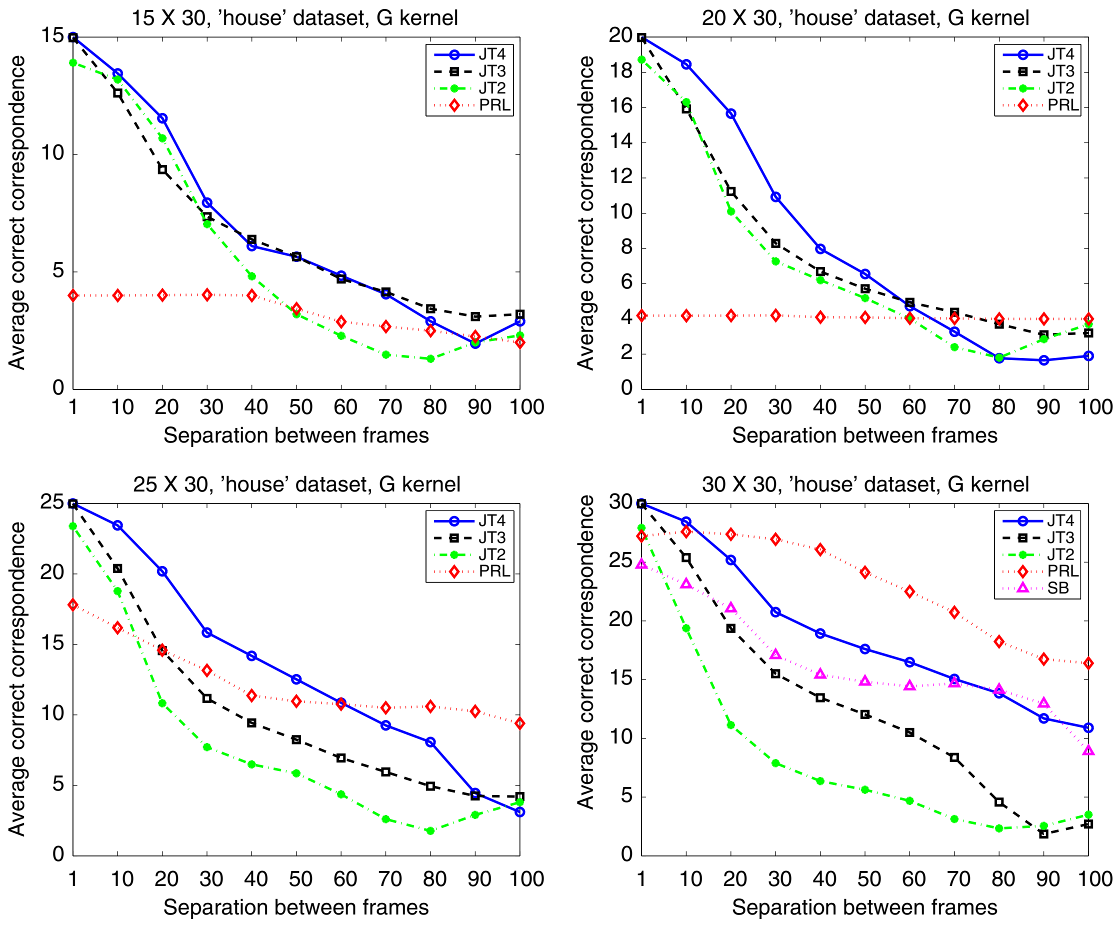

Fig. 9. Results for the "CMU house" dataset. For JT methods and PRL, the Gaussian kernel (Eq. (12)) is used, with $\sigma = 0.4$. For the SB method, $\sigma = 0.4$.

advantage of JT methods over the alternative ones becomes clear. As the size of the codomain pattern increases—for a fixed size of the domain pattern—the robustness of JT4 and JT3 are by far superior to that of PRL. It is also clear from this figure that, for very small jitter (std = 1, top left graph in Fig. 7), even JT2 outperforms PRL when the ratio between the two point set sizes increases. This is yet another example of the point made in this paper: a technique from the "second class" can be much superior than one of the "first class", *even* when the degree of problem approximation is severe. For higher jitter, however, it is clear that JT2 collapses (the problem approximation in this case becomes extreme). We believe it is worth repeating: the graphs in Fig. 7 show the relative performances between a technique that takes into account all the relative pairwise distances between the points (PRL) and techniques that use, respectively, only $T-1$, $2T-3$ and $3T-6$ (so, $O(T)$ instead of $O(T^2)$) relative distances in the domain pattern (see Fig. 4).

The real-world experiments to some extent reflect the behavior observed in the synthetic ones. However, notice that baseline separation is not properly an analogous to jitter. Overall, it is clear how PRL is sensitive to the ratio between the sizes of the patterns, whereas all JT models are almost insensitive. For patterns of equal sizes, there is basically no observable reason to choose any JT model over PRL, unless the baseline is really small. Note of course that the isometric assumption is only reasonable for small baselines, but nevertheless we decided to compute how all the methods would perform for large baseline as well (all methods rely purely on Euclidean distances, so this is supposed to be a fair comparison).

It is interesting to observe how small is the difference between JT3 and JT4 compared to that from JT2 to JT3, in general. This may give us hints as to which model to select in practical applications, when trading-off accuracy and computational burden is a recurrent issue. This is so because the complexity of the Junction Tree algorithm is $O(TS^{k+1})$, where $k+1$ is the size of the maximal clique, and as a result the cost of switching from JT2 to JT3 and to JT3 to JT4 increases by a factor of S. Also, it is clear that the approximation procedure is quite general and k-trees with greater value of k can be used. However, in our implementations we found this is not computationally attractive (the complexity for example goes to $O(TS^5)$ for a JT5 model). At this point, however, it is relevant to mention that all JT4 models for matching problems in the plane are in some sense *sufficient*. Indeed, it can be shown that this technique can be generalized to matching in any dimension, where then a "JTn" model will be sufficient for matching in $\mathbb{R}^{n-2}$. This is however the theme of another paper.

7. Conclusion

In this work, we have investigated how different sources of approximation affects the performance of point set matching methods. Usual approaches to point set matching, such as spectral and relaxation-based methods, encode all the available information in the model representation, but rely on approximate algorithms for deriving the assignment. Our method, in contrast, consists in approximating the problem itself such that the resulting representation is suitable for the use of optimal algorithms for finding the match. The method consists in modeling the relational features of a point set in a Graphical Model where the underlying graph structure is sparse and allows for optimal MAP computation in polynomial time. Experiments were performed both with synthetic and real-world data sets, which indicate that the proposed "approximate model-optimal algorithm" approach is a viable alternative to other "optimal model-approximate algorithm" approaches, over a wide operating range.

Acknowledgements

National ICT Australia is funded by the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council. Part of this work was developed when the authors were with the Department of Computing Science, University of Alberta.

References

[1] H. Wang, E.R. Hancock, A kernel view of spectral point pattern matching, International Workshops SSPR & SPR, Lecture Notes in Computer Science, vol. 3138, Springer, Berlin, 2004, pp. 361–369.

[2] T.S. Caetano, T. Caelli, D.A.C. Barone, An optimal probabilistic graphical model for point set matching, International Workshops SSPR & SPR, Lecture Notes in Computer Science, vol. 3138, 2004, pp. 162–170.

[3] T. Akutsu, K. Kanaya, A. Ohyama, A. Fujiyama, Point matching under non-uniform distortions, Discrete Appl. Math. Special Issue: Comput. Biol. Ser. (IV) (2003) 5–21.

[4] M. Carcassoni, E.R. Hancock, Spectral correspondence for point pattern matching, Pattern Recognition 36 (2003) 193–204.

[5] L. Shapiro, J. Brady, Feature-based correspondence—an eigenvector approach, Image Vis. Comput. 10 (1992) 268–281.

[6] W.J. Christmas, J. Kittler, M. Petrou, Structural matching in computer vision using probabilistic relaxation, IEEE Trans. PAMI 17 (8) (1994) 749–764.

[7] T. Caelli, S. Kosinov, An eigenspace projection clustering method for inexact graph matching, IEEE Trans. PAMI 26 (4) (2004) 515–519.

[8] S. Gold, A. Rangarajan, A graduated assignment algorithm for graph matching, IEEE Trans. PAMI 18 (4) (1996) 377–388.

[9] A. Rosenfeld, A.C. Kak, Digital Picture Processing, Academic Press, New York, NY, 1982.

[10] A. Rosenfeld, R. Hummel, S. Zucker, Scene labeling by relaxation operations, IEEE Trans. Systems, Man and Cybernetics 6 (1976) 420–433.

[11] G. Scott, H. Longuet-Higgins, An algorithm for associating the features of two patterns, Proc. R. Soc. London B 224 (1991) 21–26.

[12] A. Robles-Kelly, E.R. Hancock, Graph edit distance from spectral seriation, IEEE Trans. PAMI 27 (3) (2005) 365–378.

[13] J.V. Kittler, E.R. Hancock, Combining evidence in probabilistic relaxation, Int. J. Pattern Recognition Artificial Intelligence 3 (1989) 29–51.

[14] R.C. Wilson, E.R. Hancock, Structural matching by discrete relaxation, IEEE Trans. PAMI 19 (6) (1997) 634–648.

[15] H.-X. Huang, Z.-A. Liang, P.M. Pardalos, Some properties of the Euclidean distance matrix and positive semidefinite matrix completion problems, J. Global Optim. 25 (2003) 3–21.

[16] M.I. Jordan, An introduction to probabilistic graphical models, in preparation.

[17] S.L. Lauritzen, Graphical Models, Oxford University Press, New York, NY, 1996.

[18] CMU 'house' dataset, <http://vasc.ri.cmu.edu/idb/html/motion/house>.