Graphical Models and Point Pattern Matching

Tibério S. Caetano, Terry Caelli, *Fellow*, *IEEE*, Dale Schuurmans, and Dante A.C. Barone

Abstract—This paper describes a novel solution to the rigid point pattern matching problem in Euclidean spaces of any dimension. Although we assume rigid motion, jitter is allowed. We present a noniterative, polynomial time algorithm that is guaranteed to find an optimal solution for the noiseless case. First, we model point pattern matching as a weighted graph matching problem, where weights correspond to Euclidean distances between nodes. We then formulate graph matching as a problem of finding a maximum probability configuration in a graphical model. By using graph rigidity arguments, we prove that a sparse graphical model yields equivalent results to the fully connected model in the noiseless case. This allows us to obtain an algorithm that runs in polynomial time and is provably optimal for exact matching between noiseless point sets. For inexact matching, we can still apply the same algorithm to find approximately optimal solutions. Experimental results obtained by our approach show improvements in accuracy over current methods, particularly when matching patterns of different sizes.

Index Terms—Point pattern matching, graph matching, graphical models, Markov random fields, junction tree algorithm.

1 Introduction

Point pattern matching (or point set matching) is a basic problem in pattern recognition that is fundamental to computer vision (stereo correspondence, image registration and model-based object recognition [3], [4], [5], [6]), astronautics [7], [8], computational chemistry [9], [10] and computational biology [11], [12]. Here, we consider the (possibly noisy) rigid body case, when one pattern differs from a subset of the other by an isometry but position jitter may be present.

1.1 Problem Description and Related Problems

In general terms, the problem consists of finding a correspondence between elements of two point sets in $\mathbb{R}^2$ or $\mathbb{R}^3$ (or in $\mathbb{R}^n, n \in \mathbb{N}$, for general—not necessarily visual—patterns). In the case of *exact* matching, one point set differs from a subset of the other by an isometric transformation. In the *inexact* case, there is position jitter in one point set with respect to the other. This occurs in practical application domains like those cited above, so matching algorithms need to take this into account.

A related, but more general problem, is that of *graph* matching, which consists of finding correspondences (one-to-one [13], many-to-one [14] or many-to-many [15], [16]) between the nodes of two graphs so as to achieve some form of global consistency. In this case, nodes and edges may have vector attributes or labels. There is a vast literature addressing the graph matching problem in pattern recognition, which can be divided generally into work on search methods [14], [17], [18], [19], [20], [21], [22], [23], and work on nonsearch methods, such as probabilistic relaxation [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], spectral and least-squares methods [5], [36], [37], [38], [39], [40], graduated assignment [13], genetic optimization [41] and other principles [15], [16], [42], [43]. For a recent comprehensive review on graph matching for pattern recognition, see [44]. We have shown how ideas similar to those presented in this paper can be applied to the graph matching problem in [14] and [45]; however, in this paper we focus specifically on the point pattern matching problem.

1.2 Potential Applications

Isometric point pattern matching (allowing for jitter) is encountered in several application domains.

In computer vision, two sets of interest points extracted from two stereo images are approximately related by an isometry when the stereo pair has a narrow baseline. An accurate correspondence between the features results in an accurate depth map or the recovery of the 3D geometry of the scene [46]. This form of stereo correspondence constitutes one of the fundamental point pattern matching problems of computer vision.

In astronautics, the attitude of sounding rockets or satellites can be estimated by matching stellar images acquired from the onboard star sensor (a CCD camera) to those in an empirical star catalog [47]. Images acquired from the same region of the sky but from different viewpoints reveal sets of stars whose coordinates are related by an isometric transformation [7]. In this way, the star matching problem can be posed as a rigid point pattern matching problem.

In computational chemistry, rigid point pattern matching is a recurrent problem in drug design, specifically in the identification of pharmacophores—common subsets of molecules that systematically interact with some receptor (i.e., that perform some specific task). By matching a set of molecules (called ligands) that activate ("bind") a given receptor, one can identify whether there is a common

- *T.S. Caetano and T. Caelli are with National ICT Australia, Locked Bag 8001, Canberra ACT 2601, Australia, and the Research School of Information Sciences and Engineering, Australian National University, Canberra, ACT 0200. E-mail: {Tiberio.Caetano, Terry.Caelli}@nicta.com.au.*
- *D. Schuurmans is with the Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada. E-mail: dale@cs.ualberta.ca.*
- *D.A.C. Barone is with the Instituto de Informática, Universidade Federal do Rio Grande do Sul, Av. Bento Goncalves 9500, Porto Alegre, RS, Brazil, 91501-970. E-mail: barone@inf.ufrgs.br.*

Manuscript received 18 June 2005; revised 14 Feb. 2006; accepted 16 Feb. 2006; published online 11 Aug. 2006.
Recommended for acceptance by A. Rangarajan.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0324-0605.

subconformation among the ligands. If this is the case, the structure encountered becomes a candidate pharmacophore, which is a distillation of the functional attributes of ligands that accomplish some specific task. The pharmacophore can then be used in the design of a new drug which is expected to systematically interact with the given receptor [10].

Finally, a similar problem arises in computational biology, when the interest is to detect specific structural motifs within a family of proteins (or DNA sequences). Identification of these motifs contributes to uncovering the mechanism of the proteins' operation [12].

In all these problems, rigid point pattern matching is a reasonable assumption, but small stochastic deviations in the point positions must be accommodated (jitter). This latter condition excludes methods that only apply to exact point pattern matching problems (like [48]). The technique proposed in this paper is precisely designed for this case: We make the rigid body assumption (isometric assumption) *but* jitter is allowed.

1.3 Related Literature

Several approaches have been proposed to solve the inexact point pattern matching problem. Major classes of solutions are based on *spectral* methods [4], [5], [37], *relaxation labeling* [28], [29], [30], [33], [34], [35], and *graduated assignment* [13], [49]. The first compares the eigenstructure of proximity matrices of the point sets. The second defines a probability distribution over mappings and optimizes using a discrete relaxation algorithm. The third combines the "softassign" method [50] with Sinkhorn's method [51] to optimize the mapping. All these approaches can be seen as using optimal representations (complete data models) and approximate inference procedures. Spectral methods use the spectrum of the full adjacency matrix, but it is well-known that different graphs can be cospectral [52]; probabilistic relaxation labeling typically uses compatibility functions defined over all points, but the optimization procedure is iterative and known to be convergent to local minima [29]; graduated assignment also uses the entire set of pairwise compatibilities, being a continuous relaxation of the original combinatorial problem, which aims at tractability but is also only convergent to local minima [13]. These sources of approximation affect performance in various ways. For example, it has been frequently reported that spectral methods are not robust to structural corruption nor to matching patterns of very different sizes [4], [5]. Relaxation methods degrade with significant increases in point set sizes [13]. Graduated assignment, although extremely robust with respect to jitter, has a number of heuristic parameters that need to be tuned and, more importantly, is very sensitive to matching sets of significantly different sizes [13], [53]. All these methods are polynomial time approximations that do not guarantee global optimization.

1.4 The Proposed Technique

In this paper, we propose a conceptually different approach that overcomes many of the limitations of previous techniques. Rather than using a complete data model and an approximate inference algorithm, we do the opposite: We approximate the representation but show how optimal polynomial time algorithms can be applied to the approximated data model. However, the hallmark of this approach is that the "approximated" data model can be proven to be *equivalent* to the complete data model in the limit case of exact matching. This will ultimately allow us to obtain optimal polynomial time solutions in representations that are themselves optimal. This translates directly to optimal solutions to the original problem itself.

More specifically, our formulation is based on first posing the problem of deriving the best assignment as a graph matching problem and then solving, with an optimal algorithm, "approximate" versions of this graph matching problem, which only include a particular subset of the edge weights. This is indeed an abstraction from the original point matching task but, at the end, there will be a one-to-one correspondence between solutions of the abstracted and original problems. The motivation for this particular type of abstraction comes from the fact that such "approximations" to the graph matching problem have a strong theoretical justification. Many edge weights in the resulting graphs are in some sense "redundant," which allows us to prove—in a key part of this paper—that in the limit case of no jitter there is no approximation at all: The weights which are thrown away are completely irrelevant. It turns out that such redundancies can be naturally formulated in mathematical terms as *conditional independence assumptions* if the nodes of the first graph are seen as random variables (thus, inducing a probabilistic graphical model in the first graph [54]). If the nodes of the other graph are seen as possible realizations for these random variables, the graph matching problem becomes a problem of finding the optimal joint realization for the random variables in the graphical model, or the MAP estimate. Remarkably, the resulting "sparse" graphical model—without redundant edges—has sufficient structure to permit exact MAP computation in polynomial time—a computation that is intractable in the fully connected model. This allows us to obtain an optimal algorithm that runs in polynomial time over an optimal representation. The result is a globally optimal solution to the original problem, which is computable in polynomial time.

The resulting technique will be shown to be robust with respect to size increases in the point patterns, as well as with respect to significant differences in their sizes. It is also robust to moderate point jitter. Moreover, contrary to heuristic formulations, it is derived from first principles using Markov random field theory: The technique is noniterative and has a single parameter to be tuned (the only parameter involved being inherent to all techniques that aim to cope with jitter). To the best of our knowledge, this constitutes the first provably optimal polynomial time algorithm for exact point set matching in $\mathbb{R}^n$ that is also applicable to inexact matching (optimal algorithms which are exclusive to the unrealistic exact case do exist [48]). For the realistic problem of matching noisy point patterns, we present experimental results comparing the proposed algorithm with well-known alternative methods. Our results show that the proposed technique offers accuracy improvements, particularly when matching patterns of different sizes.

2 Point Matching as a Weighted Graph Matching Problem

We start by showing how point pattern matching can be formulated as a weighted graph matching problem. Assume we have two point sets in $\mathbb{R}^n$ ($n \in \mathbb{N}$), named $\mathcal{T}$ for "template" and $\mathcal{S}$ for "scene," with cardinalities T and S,

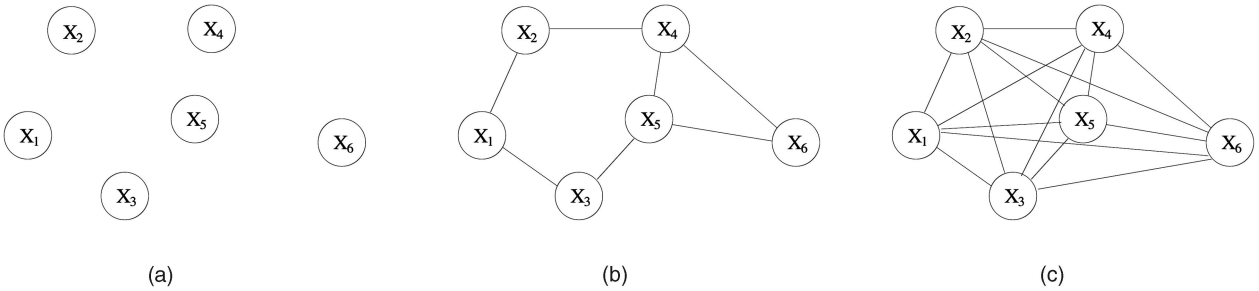

Fig. 1. Examples of three undirected graphical models. (a) Every conditional independence assumption holds. (b) Some conditional independence assumptions hold, some do not. (c) There are no conditional independence assumptions.

respectively. The idea is that some noisy instance of $\mathcal{T}$ (denoted $\mathcal{T}'$) is present in $\mathcal{S}$, up to an isometric transformation. Our goal is to find this instance $\mathcal{T}'$ in $\mathcal{S}$ and, moreover, determine a map $f : \mathcal{T} \mapsto \mathcal{S}$ that maximizes some "global similarity measure" between $\mathcal{T}$ and $\mathcal{T}'$. The *only* restriction we impose on f is that it must be a function: Every point in $\mathcal{T}$ must map to some point in $\mathcal{S}$. This is in contrast to one-to-one mapping [13], which considers a smaller class of solutions. We find this assumption of many-to-one mappings to be required in the present formulation, for reasons to be explained later in the next section. It is natural to understand $\mathcal{T}$ as the point set corresponding to a "model" and $\mathcal{S}$ as the point set obtained from a "scene" wherein we want to find some instance of the model.

Here, we will refer to the template pattern as a "domain pattern" and the scene pattern as a "codomain pattern" in analogy to their roles played with respect to the mapping function f. The ith point in the domain pattern is denoted by d_i, whereas the kth point in the codomain pattern is denoted by c_k. The Euclidean distance between d_i and d_j is denoted by y^d_{ij} and that between c_k and c_l is denoted by y^c_{kl}.

The key idea for modeling point pattern matching as a weighted graph matching problem is as follows: Recall that an isometry exists between two point sets if and only if they have the same Euclidean Distance Matrix [55] (EDM) under some permutation [56]. Consequently, an isometry can be tested by comparing all the permutations of two EDMs entrywise. In our case, we would like to handle inexact matching, which means we must also accommodate noisy situations and sets of different sizes. Thus, we define the matching problem as finding the map f that minimizes the cost

$$U_T(f) = \sum_{i=1}^{T}\sum_{j=1}^{T} \mathcal{D}(y^d_{ij}, y^c_{f(i)f(j)}), \tag{1}$$

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

This definition is equivalent (apart from f being many-to-one instead of one-to-one) to that of the weighted graph matching problem of [13], where edge weights are restricted to be relative Euclidean distances between points corresponding to the respective vertices embedded in $\mathbb{R}^n$. (Note that since all distances are taken into account, the graphs are *fully connected*.) Equation (1) actually represents an instance of the quadratic assignment problem which, in general, is known to be NP-complete [13]. Due to this graph matching formulation, we will refer to the "domain graph" G_d and the "codomain graph" G_c as the graph abstractions of the point sets. This gives the formulation of our problem as a "Euclidean" weighted graph matching problem.

3 Weighted Graph Matching as a MAP Problem in a Graphical Model

This problem can be further reformulated as finding a maximum probability (MAP) configuration in a probabilistic graphical model [57], [58], [59], [60]. Before presenting our formulation, we briefly review the main ideas about graphical models that will be required in our exposition.

3.1 Graphical Models

Graphical models are graphical representations for families of probability distributions [54], [57], [59], [61]. We will be considering exclusively *undirected* graphical models, sometimes referred to as *Markov random fields* in certain application domains. (In this paper, "graphical models" and "Markov random fields" are complete synonyms.) A graphical model is, essentially, a graph where nodes represent random variables and the edge pattern represents a set of conditional independence assumptions made among the random variables.[1] If a subset of nodes B separates (in the graph-theoretic sense) the set of nodes A from the set of nodes C, then this means, in graphical model formalism, that A and C are conditionally independent on B; that is, $p(AC|B) = p(A|B)p(C|B)$. For examples of graphical models that induce different sets of conditional independence assumptions among their variables, see Fig. 1.

Fig. 1 shows three graphical models. Each node, X_i, in a model corresponds to a random variable, which can assume a set of different realizations (in our context this set will be discrete). A fundamental result about graphical models is the Hammersley-Clifford (HC) theorem, which states that any strictly positive probability distribution that respects the set of conditional independencies implied by a graph can be written in a factored form, namely, as a product of functions over the maximal cliques[2] [57], [59]:

$$p(x) = \prod_{c\in\mathcal{C}} \psi_c(x_c)/Z, \tag{2}$$

1. All our statements about graphical models in this paper will be restricted to *discrete* random variables.

2. Recall that a clique is a complete subgraph and a maximal clique is a clique which is not a proper subset of another clique.

where c is a maximal clique, $\mathcal{C}$ is the set of all maximal cliques, and x_c is the restriction of x to the clique c. Z is the normalization constant that renders $\sum_{\mathcal{X}} p(x) = 1$. The nonnegative function $\psi_c(x_c)$ is called the *potential function*, which, in our case, will be a table with the dimensionality of x_c. From this theorem, it is clear that all we need to specify a probability distribution is a connectivity pattern for the graphical model and a set of potential functions.

The basic "query" that we are then interested in answering about a graphical model is the following: what is the most likely joint realization of all the random variables? In other words, what is the mode of the joint probability distribution defined by a graphical model and its potential functions? This is known as the MAP (maximum a posteriori) problem in a graphical model. For fully connected models, like the one in Fig. 1c, this problem is intractable (for discrete random variables). For completely independent models, like that in Fig. 1a, this problem is trivial: The joint mode can be obtained by computing each of the individual modes independently. Models that lie between these two extremes, of which the one in Fig. 1b is an example, can be either tractable or not.

At this point, it is important to state what determines the tractability of the model. The fundamental algorithm for exact inference in graphical models is the junction tree algorithm [54], [55], [56], [57]. It works by creating a hypergraph (a "junction tree") from the original graph and then running a dynamic programming algorithm on this hypergraph. However, junction trees can only be created for *triangulated*[3] (i.e., chordal) graphs [57], [58], so the effective computational complexity depends on triangulated versions of the original graph.[4] In general, there are many possible triangulations for a given graph. The exponential complexity of the MAP computation for a given graphical model will be determined by the minimum size, taken over all possible triangulations, of the maximal clique in the triangulation. If this exponent grows with the size of the graph, then the model is intractable; otherwise, it is tractable. For example, a fully connected graph is triangulated with a maximal clique size equal to the size of the graph itself, which immediately implies intractability. Naturally, in practice, one requires the exponent to be not only fixed, but also small. Notice also that, if a graph is *already* triangulated, other triangulations will only *increase* the size of the maximal clique, so the exponential complexity will be given directly by the size of the maximal clique of the graph, without any need for triangulation. Since the problem of finding a triangulation that has a minimal maximal clique size is NP-complete [59] (one calls it an "optimal triangulation"), the "ideal" scenario would be one in which the graph is already triangulated. We exploit this fact below by identifying a triangulated graphical model structure for our problem that has a small maximum clique size.

Next, we show how the point pattern matching problem can be formulated as a MAP problem in a graphical model. Although, in the initial formulation, the graphical model will be fully connected (and thus intractable), we will show afterward how we can obtain *the same* MAP solutions with a sparse, tractable model.

3.2 Formulation

The key idea for modeling weighted graph matching as a MAP problem in a graphical model is as follows: Assume that each vertex in the domain graph is a random variable X_i and that each such random variable has a finite set of possible realizations coinciding with the set of vertices in the codomain graph. This means that a particular realization x_k of a random variable X_i corresponds to a particular map between the point d_i in the domain pattern and a point c_k in the codomain pattern. Thus, a joint realization $x = \{x_k\}$ of the set of variables $X = \{X_i, \forall i | d_i \in \mathcal{T}\}$ corresponds to a particular match between the point sets $\mathcal{T}$ and $\mathcal{S}$.

In this spirit, one can define a probability distribution such that the most likely joint realization of the variables (the MAP configuration) corresponds to the minimum of (1).

In order to accomplish this, we specify a Markov random field based on edgewise potentials over the *fully connected graph*. Let ψ_{ij} denote the local potential function for edge (i, j). Then, the joint probability distribution over the pairwise Markov random field is

$$\begin{aligned} p(X = x) &= \frac{1}{Z}\prod_{(i,j)} \psi_{ij}(X_i = x_i, X_j = x_j) \\ &= \frac{1}{Z}\exp\left(-\sum_{(i,j)} V_{ij}(X_i = x_i, X_j = x_j)\right), \end{aligned} \tag{3}$$

where $V_{ij}(X_i, X_j) = -\log(\psi_{ij}(X_i, X_j))$ and Z is a global normalization constant determined by summing the product of potentials over all possible joint realizations x. For clarity, in (3), we have used standard notation where x_i denotes a *generic* realization of X_i (i.e., *any* realization, not one in particular indexed by i). In the context of this paper, we find it more convenient to modify this notation such that X_i is still the random variable, but $x_{f(i)}$ is now the *specific* realization indexed by $f(i)$.

To relate this problem to (1) (and, here, we use the new notation), all we have to do is specify appropriate potentials. In particular, define

$$\begin{aligned} V_{ij}(X_i = x_i, X_j = x_j) &= V_{ij}\big(d_i \mapsto c_{f(i)}, d_j \mapsto c_{f(j)}\big) \\ &:= \mathcal{D}\Big(y^d_{ij}, y^c_{f(i)f(j)}\Big). \end{aligned}$$

The resulting model becomes

$$\begin{aligned} p(f) &= \frac{1}{Z}\exp\left(-\sum_{i=1}^{T}\sum_{j=1}^{T}\mathcal{D}\Big(y^d_{ij}, y^c_{f(i)f(j)}\Big)\right) \\ &\propto \exp(-U_T(f)). \end{aligned} \tag{4}$$

Thus, maximizing $p(f)$ is equivalent to minimizing $U_T(f)$. Note that we now write the realization $X = x$ in the form of a map f: Each random variable X_i, which corresponds to a point d_i, will "map" to a realization $x_{f(i)}$, which corresponds to point $c_{f(i)}$ (note the new notation).

The reason why we require the whole class of many-to-one matches is that, in general, one cannot prevent two random

3. A graph becomes triangulated (or, equivalently chordal) by adding edges in such a way that all cycles of length greater than 3 have a chord. A chord is an edge that does not belong to the cycle but connects two nodes in the cycle.

4. Note that "transforming" a graph by triangulating it is not restrictive, since triangulation can only add edges and, therefore, only reduces the set of conditional independence assumptions implied by the original graph.

variables from having the same realization—unless they are in the same clique.[5] PRL-based approaches also behave in this way [33], [34]. In general, this can be seen as a limitation, because in many cases (and certainly in exact point set matching) ideal solutions are one-to-one. One exception when one-to-one approaches will fail to provide the ideal interpretation is in cases where two points in one pattern are actually *superimposed* on the second pattern. However, it is fair to acknowledge that, in general, a one-to-one version of the proposed algorithm would be not only desirable but invaluable since, in most cases, many-to-one matches may appear artificial. This is a current limitation of the present algorithm which, as of this stage, we cannot see how to overcome. Nevertheless, as will be shown, the experimental results indicate that this assumption has not prevented the proposed algorithm from improving the matching accuracy over the competing ones in many operating regions (that is, every one-to-one solution is *also* tested in our algorithm and, if any of them turns out to have a cost which is smaller than the costs over all other solutions, this will be the selected assignment).

Returning to (4), although the equivalence between finding the MAP solution and minimizing the energy function is important, it does not immediately yield a useful approach to solving the problem because MAP computations over a fully connected Markov random field are intractable. The key idea in this paper is to *approximate* $U_T(f)$ in such a way that only a subset of all the pairwise cliques in the fully connected model is taken into account. This will eventually lead us to a graphical model that is tractable. However, the hallmark of the particular model that we will obtain is that its MAP solutions can be proven to be *the same* as those of the fully connected model in the noiseless case. This makes the "approximation" exact.

4 The Model

To construct a sparse alternative to the fully connected graphical model given in (4), we need to specify 1) a set of potential functions that will define the function $\mathcal{D}$ and 2) a connectivity pattern that will define the subset $\mathcal{C}_2$ of edges on which we will define potentials.

First, to specify the potentials, consider the local "kernel" structure of our model shown in Fig. 2. Generally speaking, a potential function associates a nonnegative real number to each element of the sample space [57], [59]. In our model, potentials will be defined on edges, where each node contained in an edge (a random variable) represents one of the T vertices in G_d, which, in turn, can assume a set of S possible realizations (which correspond to vertices in G_c). Thus, the sample space for each edge has S^2 elements, and we can specify the potential function for an edge (i.e., a pair $\{X_i, X_j\}$ in G_d) by an $S \times S$ *compatibility* matrix of the edge:

$$\psi_{ij}(X_i, X_j) = \begin{pmatrix} \mathcal{H}(y_{ij}^d, y_{11}^c) & \dots & \mathcal{H}(y_{ij}^d, y_{1S}^c) \\ \vdots & \ddots & \vdots \\ \mathcal{H}(y_{ij}^d, y_{S1}^c) & \dots & \mathcal{H}(y_{ij}^d, y_{SS}^c) \end{pmatrix}, \quad (5)$$

5. If they are in the same clique, one can design a potential of zero for any joint assignment to the same realization.

Fig. 2. Local "kernel" structure of the graphical model. Each random variable can assume S possible realizations, so that the sample space for two connected variables has S^2 elements.

where y_{ij}^d denotes the edge weight between vertices with indexes i and j in graph G_d (which corresponds to the Euclidean distance between points d_i and d_j). An analogous notation holds for y_{kl}^c. $\mathcal{H}$ is a function that measures how similar these arguments are.

To measure compatibility in the exact matching case (no noise), we can simply use the indicator function

$$\mathcal{H}(y_{ij}^d, y_{kl}^c) = \mathbf{1}(y_{ij}^d = y_{kl}^c) \equiv \begin{cases} 1, & \text{if } y_{ij}^d = y_{kl}^c \\ 0, & \text{if } y_{ij}^d \neq y_{kl}^c. \end{cases} \quad (6)$$

For inexact matching, where we assume jitter in the point positions (typical in practice), we need a more general "proximity measure" to cope with uncertainty. Thus, in these cases, we measure compatibility using the Gaussian kernel[6]

$$\mathcal{H}(y_{ij}^d, y_{kl}^c) = \exp\left(-\frac{1}{2\sigma^2}\left|y_{ij}^d - y_{kl}^c\right|^2\right). \quad (7)$$

Other similarity measures could be chosen, but we do not focus on this issue here.[7] Note that *any* technique for matching noisy patterns requires some soft similarity measure, including the methods we compare to in this paper (where we use the same kernel). For example, relaxation labeling [13] and graduated assignment [13] both use a *compatibility measure* between pairs of assignments to score any putative matching. These scores use a parameter to adjust for the level of position jitter in the data. Thus, the single parameter in (7) is not itself an artifact of our method, but a necessary element in any matching model that aims to cope with noise.

Having specified the potential functions, it remains to determine the connectivity of the graphical model. Here, we will simply propose the graphical model structure shown in Fig. 3 and assert that this graphical model structure *preserves the MAP solutions of the fully connected model*, a fact we will verify in Section 5 below.

Before proceeding with the proof of its optimality, we make a few remarks about this model. First, Fig. 3 illustrates a model that is specifically constructed for matching in $\mathbb{R}^2$. For matching in $\mathbb{R}^{k-1}$, an analogous topology can be used: Instead of a three-clique in the upper layer, one simply uses a k-clique, and each of the other $T-k$ nodes is then connected to each of these k nodes. For any k (and $T > k$), this generic model topology has two important features: 1) it is *already* triangulated and 2) the size of the maximal clique is $k+1$, *independent* of both the number of nodes T and of the number of possible realizations S. As explained in Section 3, because it is triangulated, we know that this

6. In the exact matching case, the Gaussian kernel actually gives identical results to the indicator, since its maximum is attained uniquely at an exact match. However, the indicator makes the upcoming theoretical results clearer.

7. An early attempt to evaluate different measures is reported in [62].

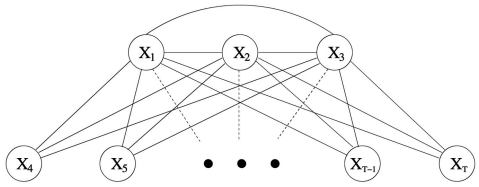

Fig. 3. A model for matching in $\mathbb{R}^2$. The topology of the model corresponds to that of a 3-tree graph, whose maximal clique size is 4 (thus independent on T, the number of nodes, and S, the number of possible realizations for each random variable).

model has a junction tree and, because it has a bounded maximal clique size, the "junction tree algorithm" has polynomial complexity in this model. That is, for models like the one in Fig. 3 the *exact* MAP solutions can be computed in polynomial time.

It might sound artificial to define the "candidate" topology as a triangulated graphical model with a fixed maximal clique size (which together form sufficient conditions for polynomial time complexity). However, in the next section, we show that this topology is not postulated, but derived from first principles, which reveals a subtle connection between exact inference in graphical models and the "global rigidity of graphs."

5 Optimality of the Model

In this section, we present theoretical results that lead to a special kind of graph: a "k-tree." The properties of this graph will allow us to draw a connection to the problem of exact inference in graphical models and will ultimately lead us to prove that the model shown in Fig. 3, although sparse and computationally tractable, yields *equivalent* results to the fully connected model in the limit case of exact matching.

5.1 A Relevant Lemma

We start by presenting a lemma that will be necessary to obtain the subsequent results:

Lemma 1. *Let $S_1, S_2, \ldots, S_{n+1}$ be $(n+1)$ spheres in $\mathbb{R}^n$ whose centers are in general position (do not lie in a $(n-1)$-dimensional vector subspace). Then, the intersection set $\cap_{i=1}^{n+1} S_i$ is either a single point or the empty set.*

Proof. See Appendix A. □

Another way to see this result is the following: if the distances from an unknown point to $n+1$ known points in $\mathbb{R}^n$ are determined, then this point is unique—provided the $n+1$ points are in general position. In order to see this fact, note first that the unknown point is clearly in the intersection of the spheres whose centers are the $n+1$ fixed points and the radii are the respective distances between their centers and the unknown point. Second, note that Lemma 1 states that the intersection is either a single point or empty (which is not the case because we have assumed the existence of this unknown point). This implies that the point is unique. This result will be used in the following in order to obtain another result concerning the "global rigidity of graphs."

5.2 Global Rigidity of k-trees

Here, we use Lemma 1 to infer a second result that will ultimately lead us to obtain the main theorem about the topology of the graphical model. The theory of graph rigidity, although mathematically rich and sophisticated [63], involves concepts that are easy to understand. Strictly speaking, we talk about the rigidity of *graph embeddings* in $\mathbb{R}^n$, where the edges are straight lines (these embeddings are called *frameworks*). Simply put, we say that a framework is globally rigid if the lengths of the edges uniquely determine the lengths of the "absent edges" (the edges of the complement graph).

To present the key result about the global rigidity of a special kind of framework—a k-tree—we start by reviewing some basic definitions from graph theory [64]. In what follows, a complete graph with n vertices is denoted as K_n and a k-clique is a clique with k vertices. Also, recall that a *framework* is a straight line embedding of a graph.

Definition 1 (k-tree, base k-clique). *A k-tree is a graph that arises from K_k by zero or more iterations of adding a new vertex to the graph and connecting it with k edges to an existing k-clique in the previous graph. The k-cliques defined by the new vertices are called base k-cliques.*

Fig. 4 shows the process of creating a k-tree, in the particular case where $k = 3$. We start with a K_3 graph. Then, we add vertex 4 and connect it to every vertex of the (so far unique) base 3-clique. Vertex 5 is then added and is connected, in this example, to the same base 3-clique. Vertex 6 is then added and connected to another base 3-clique, formed by vertices 2, 3, and 4. Note that all intermediate graphs generated in this way are themselves legitimate 3-trees. Also note that, in general, the resulting graph is sparse (the graph with five nodes is the first to present sparseness, since the edge (4-5) is absent).

A careful examination reveals that the size of the maximal clique of a k-tree with n vertices is precisely k if $n = k$ and precisely $k+1$ if $n > k$. (This is easy to see

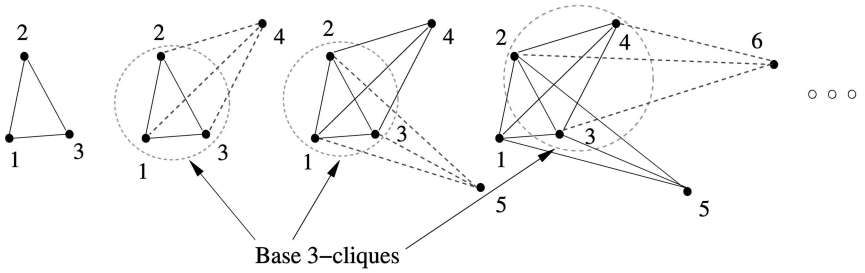

Fig. 4. The process of constructing 3-trees. At each step, a new node is added and connected to all nodes of an existent 3-clique (which is then called a "base 3-clique").

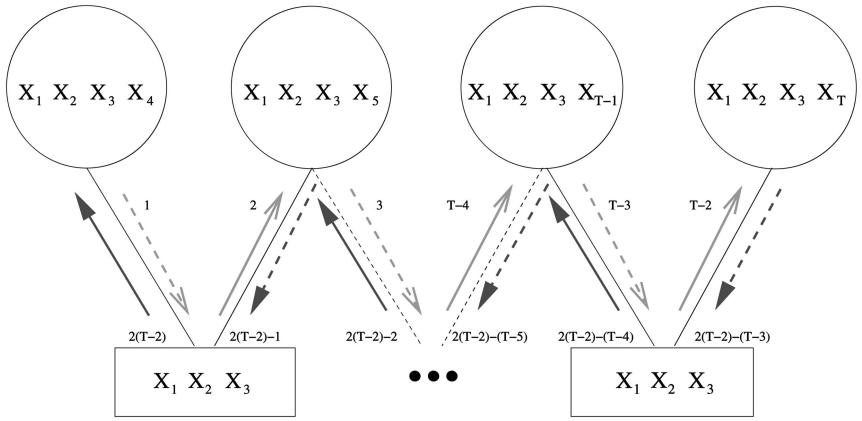

Fig. 5. The junction tree for the model in Fig. 3. Circles ("clique nodes") correspond to the maximal cliques of the original graph, whereas rectangles ("separator nodes") correspond to the intersection between adjacent clique nodes. Nonfilled arrows correspond to the first message-passing, whereas filled arrows correspond to the second. The value adjacent to an arrow denotes the order in which the corresponding message is passed. Dashed arrows correspond to (9), whereas solid arrows correspond to (10).

because every time a new vertex is added, it is connected to exactly k vertices of a k-clique, forming a $(k+1)$-clique.

We are now equipped to present the second result:

Lemma 2. *A k-tree framework with all base k-cliques in general position in $\mathbb{R}^{k-1}$ is globally rigid in $\mathbb{R}^{k-1}$.*

Proof. See Appendix A. □

The direct implication of this result is that the k-tree framework, from the perspective of pairwise distances, has *exactly* the same information content as a fully connected framework. We now show, using this fact, that our sparse model yields equivalent results to the fully connected model in the noiseless case.

5.3 Equivalence of k-tree versus Full Model

To present the main theoretical result, we add some new terminology to that established in Section 2. We specifically analyze the noiseless case, where $\mathcal{T}$ and $\mathcal{T}'$ are related by an isometry. Consider the domain and codomain graphs G_d and G_c defined in Section 2. Define $G_d^{kt} = (\mathcal{V}_d, \mathcal{E}_d^{kt})$ as a graph with the same nodes as G_d but with edge connectivity given by a k-tree whose base k-cliques are in general position in $\mathbb{R}^{k-1}$. Let $G_c^{kt} = (\mathcal{V}_c^{kt}, \mathcal{E}_c^{kt})$ be the subgraph of G_c whose nodes are those to which the nodes $\mathcal{V}_d$ map under an optimal map f. We define as $\bar{G}_d^{kt} = (\mathcal{V}_d, \bar{\mathcal{E}}_d^{kt})$ the complement graph of G_d^{kt}, while $\bar{G}_c^{kt} = (\mathcal{V}_c^{kt}, \bar{\mathcal{E}}_c^{kt})$ is the complement graph of G_c^{kt}.

Now, if we choose the edge set of the model (the set of pairwise cliques $\mathcal{C}_2$) to be a k-tree, the "approximated" optimization problem over this k-tree graph G_d^{kt} can be defined as one of minimizing the following "partial" cost function over f (as opposed to the "total" cost U_T from (1):

$$U_{G_d^{kt}}(f) = \sum_{i,j|d_{ij}\in\mathcal{E}_d^{kt}} \mathcal{D}(y_{ij}^d, y_{f(i)f(j)}^c), \tag{8}$$

where d_{ij} is the edge between vertices d_i and d_j in G_d and $\mathcal{E}_d^{kt}$ is the edge set of graph G_d^{kt}.

We can now state our main result.

Theorem 1. *In the exact matching case, a mapping function f which minimizes $U_{G_d^{kt}}(f)$ also minimizes $U_T(f)$.*

Proof. See Appendix A. □

Note now that the model shown in Fig. 3 has the topology of a k-tree (a 3-tree). As a result, the solution obtained by the junction tree algorithm over this model will minimize not only the cost function $U_{G_d^{kt}}(f)$, but also the cost function of a complete model, $U_T(f)$ (1). This is our main theoretical result.

Actually, other models can be used, as long as they have the topology of a k-tree. The specific choice of 3-tree for Fig. 3 was made simply because it has a single base 3-clique and, therefore, only requires these 3 points to be noncollinear (the points corresponding to random variables X_1, X_2, and X_3).

In the case of exact matching, as long as these points are not collinear, *any* choice can be made and Theorem 1 will still hold. However, when there is position jitter, different choices can give different results and the variance of the results over different selections of the reference points will increase with jitter (experimental evidence of this fact will be provided). A principled way of selecting the reference points in this case is still an open problem which we are currently investigating and for the purposes of the experiments presented in this paper, the selection of the reference points is made randomly.

6 Inference

Given the k-tree model, we must solve the MAP problem, i.e., determine the most likely joint realization of the random variables in the model. This is done with the junction tree algorithm. In this section, we describe how the junction tree algorithm is applied to our particular case (for details of the general case, see [59], [57], and [58]). For simplicity of exposition, we describe inference in a 3-tree (matching in $\mathbb{R}^2$), but the procedure is analogous for arbitrary k.

The junction tree for the model shown in Fig. 3 is given in Fig. 5.

The tree in this case is actually just a chain. The maximal cliques in the junction tree are denoted by circles, called "clique nodes," whereas the set of variables common to adjacent clique nodes are represented by rectangles, called "separator nodes." The junction tree algorithm is a dynamic programming procedure that systematically changes the

potentials in the clique nodes and separator nodes in a two-way "message-passing" scheme, similar to the Viterbi algorithm for MAP computation in Hidden Markov Chain models [65].

Just like the Viterbi algorithm, which after the forward and backward operations delivers the individual MAP distributions (also called "max-marginals" [58]) for each node, the junction tree algorithm delivers the MAP distribution for each clique node (up to a normalization constant Z—see Appendix B for details). The final MAP distribution for each individual node X_i can then be computed by "maximizing out" the remaining individual nodes within the clique node [57], [59]. For example, the final MAP distribution for node X_1 in Fig. 5 can be computed by

$$p(x_1) = \max_{x_2} \max_{x_3} \max_{x_4} p(x_1, x_2, x_3, x_4).$$

This operation is clearly exponential on the number of variables in the clique node, and that is one of the reasons why the junction tree algorithm is only efficient for graphs with a small maximal clique size.

The message-passing scheme works as follows. First, we initialize the potential functions for one of the clique nodes by combining the pairwise potential functions from Section 4:

$$\Psi(x_1,x_2, x_3, x_4) = \psi(x_1, x_2)\psi(x_1, x_3)\psi(x_1, x_4)\psi(x_2, x_3)\psi(x_2, x_4)\psi(x_3, x_4).$$

For the other clique nodes, the terms $\psi(x_1, x_2)$, $\psi(x_1, x_3)$ and $\psi(x_2, x_3)$ are not included, since they already have been, and the general form for $i > 4$ is

$$\Psi(x_1, x_2, x_3, x_i) = \psi(x_1, x_i)\psi(x_2, x_i)\psi(x_3, x_i).$$

The separator nodes are all then initialized to 1 [59]. After that, we perform message-passing: Starting with a clique node V that is a leaf of the chain, we compute

$$\Phi^*_S = \max_{V \setminus S} \Psi_V \tag{9}$$

$$\Psi^*_W = \frac{\Phi^*_S}{\Phi_S} \Psi_W, \tag{10}$$

where W is the clique node to which V is "sending a message." This "message" actually consists of two updates: 1) substituting the potential in the separator S by computing the MAP of clique node V with respect to the nodes that are common with the separator and 2) reweighting the potential in clique node W by the ratio between the new and the previous separator potentials. This local operation is then propagated until the other leaf is reached, after which it is repeated in the reverse direction. Once the message-passing is completed, the joint distribution has been preserved, and the marginalization property, (9), has now been established between every clique node and its separators. This ensures that we have obtained the desired MAP distribution at each clique node [59], as mentioned above.

To compute the messages, note that each potential Ψ is a 4D table, with S bins per dimension; see Fig. 5.

Thus, the maximization operation in (9) runs over the dimension of the 4D table, Ψ_V, that is not common to the 3D table, Φ_S. Similarly, the division and multiplication operations of (10) are performed entrywise in the tables.

Fig. 5 shows details of how the overall dynamic programming procedure works. As mentioned above, after the two-way message-passing is finished, local maximization yields the final MAP distributions of the singleton nodes X_i from which the mode indicates the point in the codomain pattern that matches the point in the domain corresponding to X_i.

The complexity of computing each of the messages (9), (10) is $O(S^4)$, since the largest tables (Ψs) are four-dimensional with S bins per dimension. There are, in total, $2(T-2)$ messages, so that the overall computational complexity for this model is $O(TS^4)$, which is polynomial in the size of the domain pattern (T) and in the size of the codomain pattern (S). Note that there is no iterative procedure involved, and no concept of "initialization" is present. The algorithm runs in precisely $2(T-2)$ steps and will always deliver the same result for the same input. This is because the algorithm is strictly deterministic, based solely on the dynamic programming principle [54], [57], [59]. Since the dynamic programming finds the global optimum for the given model *and* the model itself is optimal in the noiseless case, we have an algorithm for point pattern matching that has polynomial complexity and is provably optimal in the limit case of exact matching. Aiming at clarifying in detail the full inference procedure, we present a fully worked out example of the junction tree algorithm for a simple graphical model in Appendix B. Algorithm 1 shows a pseudocode for our algorithm in the general case of matching in $\mathbb{R}^{k-1}$.

Algorithm 1. Point Pattern Matching in $\mathbb{R}^{k-1}$
Data: Domain point set $\mathcal{T}$, Codomain point set $\mathcal{S}$, σ
Result: Assignment vector v
begin:
Select, from $\mathcal{T}$, k points in general position
Construct a k-tree with these k points constituting the unique base k-clique
for *every edge of the* k-tree **do**
compute compatibility matrix (5) with regard to $\mathcal{S}$ using (7) with provided σ;
Select an arbitrary maximal clique of the k-tree
for *every one of its edges* **do**
replicate its compatibility matrix across the $k-1$ remaining dimensions (thus obtaining a $(k+1)$-D compatibility array)
Assemble the $(k+1)$-D potential of the maximal clique by entrywise multiplication of the $(k+1)$-D compatibility arrays
for *every other maximal clique* **do**
for *every edge not belonging to the base k-clique* **do**
replicate its compatibility matrix across the $k-1$ remaining dimensions (thus obtaining a $(k+1)$-D compatibility array)
Assemble the $(k+1)$-D potential of the maximal clique by entrywise multiplication of the $(k+1)$-D compatibility arrays
Construct a junction tree from the maximal cliques (see Fig. 5 for $k=3$ example)
Initialize the potentials of the maximal cliques as the

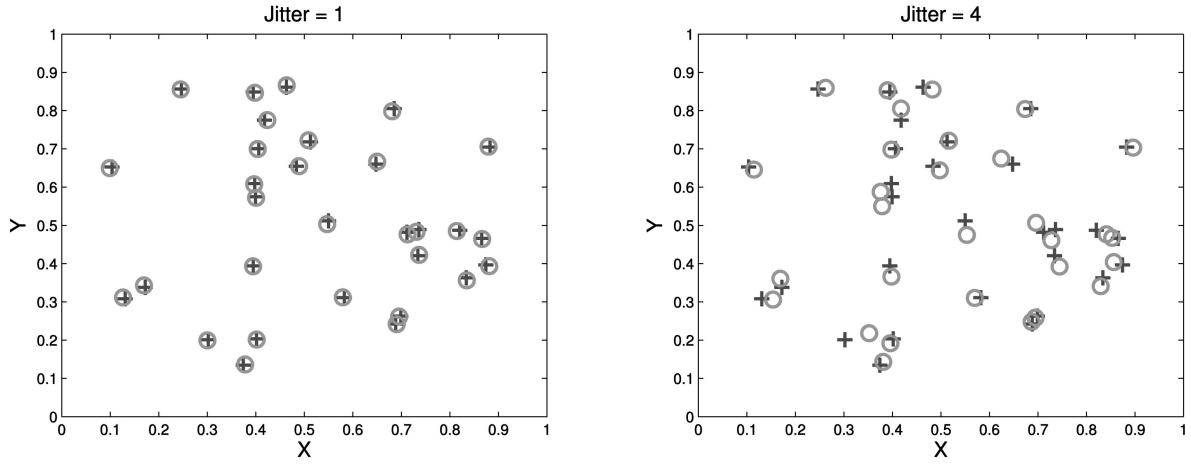

Fig. 6. Instances of patterns when different levels of jitter are introduced. Circles correspond to the jittered pattern, whereas "plus" corresponds to the original pattern. (The patterns were superimposed for the purpose of visual comparison; in practice, as should be clear from the text, they may be translated/rotated/reflected with respect to each other and may also have different cardinalities.)

assembled $(k+1)$-D potentials
Initialize the potentials of the separators to 1
Perform propagation (9), (10) from one end to the other and then back
for *every maximal clique of the junction tree* **do**
Compute the max-marginal distribution of the single variable not present in the separators by maximizing over the other k variables
for an arbitrary separator **do**
for *every variable* **do**
Compute its max-marginal distribution by maximizing over the other $k-1$ variables
for *every (ith) max-marginal distribution* **do**
Compute the argument that maximizes it, and store it in $v(i)$
end

7 Experiments and Results

One obvious shortcoming of this theory is that it only addresses the exact matching case. For inexact matching, the theoretical guarantee that the minimum of (8) equals the minimum of (1) no longer holds. However, there remains value to the framework: In the noisy case, one can still run the junction tree algorithm with the compatibility measure of (7) to cope with approximate matches, requiring the same polynomial time. The only question is: How significantly does the quality of the approximate match degrade? To evaluate this question, we conducted a number of experiments to compare our method (denoted simply as JT) to standard techniques in the literature, including probabilistic relaxation labeling (PRL), as described in [28], the spectral method (SB) presented in [37], and Graduated Assignment (GA) [49]. Note that these methods encode all pairwise distances in their objectives, whereas our method only encodes those distances that correspond to the k-tree topology. On the other hand, our approach uses an optimal noniterative algorithm, whereas the others are based on approximate heuristic algorithms. None of the standard approaches—PRL, SB, or GA—have any optimality guarantees, even in the noiseless case. The experiments involve matching tasks in $\mathbb{R}^2$, so we use the 3-tree model of Fig. 3.

7.1 Synthetic Data

To compare techniques across a range of problem conditions, we generated random points according to a bivariate uniform distribution in the interval $x = [0, 1]$, $y = [0, 1]$. We conducted two sets of synthetic experiments: 1) point sets $\mathcal{S}$ and $\mathcal{T}$ of equal sizes comparing JT, GA, PRL, and SB and 2) point sets $\mathcal{S}$ and $\mathcal{T}$ of different sizes comparing JT, GA, and PRL (SB is not suited for different graph sizes). In order to compute the similarity measure between pairwise assignments, we use the same Gaussian kernel with $\sigma = 0.4$ for all methods (see Section 4). This is the only parameter involved in our method, but is also necessary in the other ones. The problem of selecting σ is far from trivial [5], and, in this paper, we do not aim at optimizing over this parameter. See [62] for tentative experiments in this sense. We basically choose a value that we know will not underflow the kernel computation in the case of extremal differences between the argument and the mean of the Gaussian function. For the construction of the 3-tree, the three reference points were selected randomly. Also, since all algorithms exclusively use distance features (which are isometry-invariant and, as a result, do not depend on rotations, translations, or reflections in the patterns), it is not necessary to evaluate performance across several isometries since, by construction, the results will be the same up to numerical errors. Here, we simply generated random isometries for every trial.

In the first experiment, we used patterns of size $(10, 10)$, $(20, 20)$, $(30, 30)$, and $(40, 40)$ points. For each of these four instances, we perturbed the codomain pattern with progressive levels of noise, from small levels ($\mathrm{std} = 0$ to 1) to moderate levels ($\mathrm{std} = 2$), to high levels ($\mathrm{std} = 4$). The quantity "std" is 256 times the real standard deviation used (i.i.d. Gaussian noise). Fig. 6 shows typical instances of

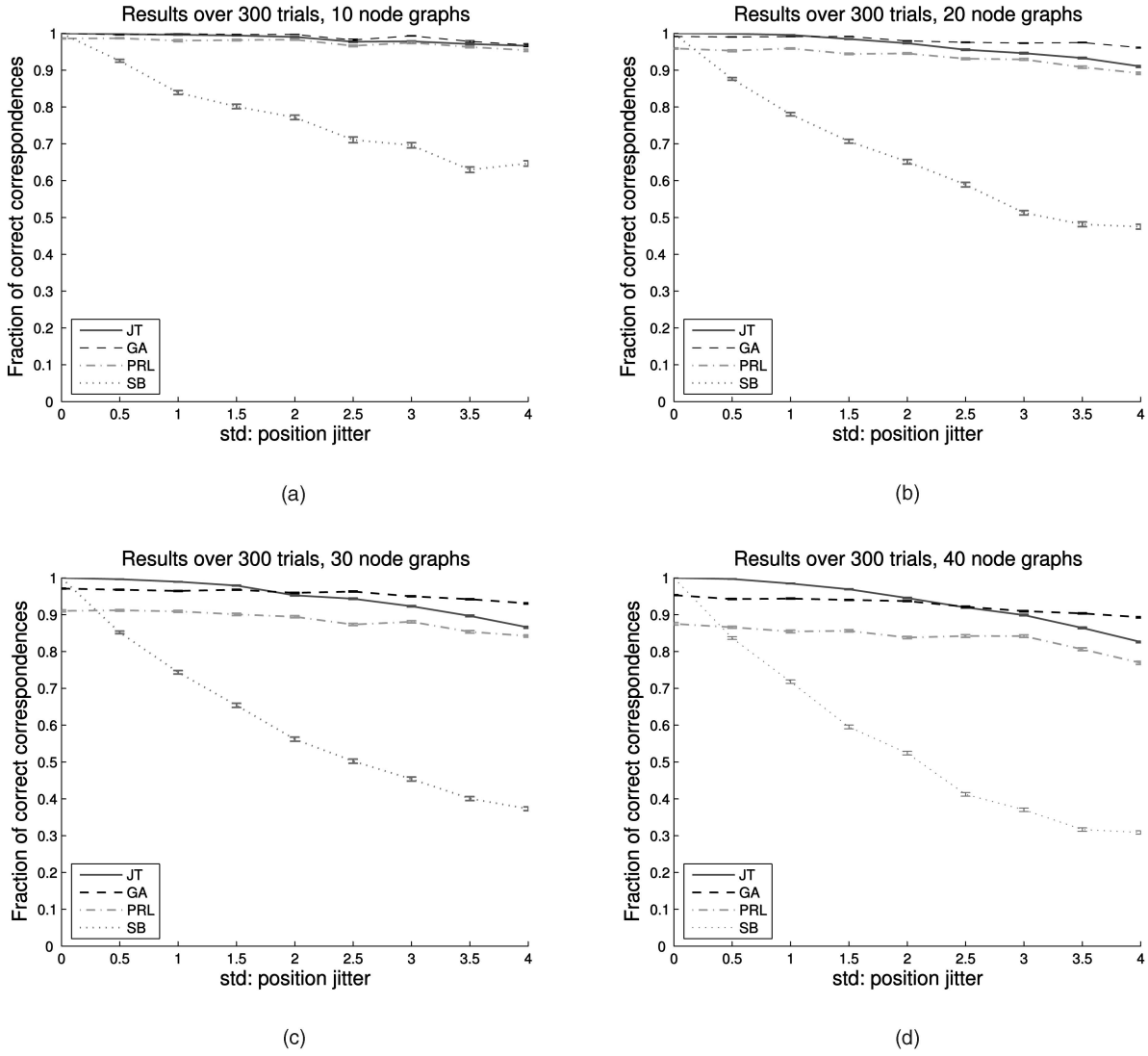

Fig. 7. Comparison of JT, GA, PRL, and SB in matching equal-sized point sets under varying jitter. Results shown for (a) 10, (b) 20, (c) 30, and (d) 40 node graphs. Error bars correspond to standard errors.

patterns perturbed with jitter of $\mathrm{std} = 1$ (Fig. 6a) and $\mathrm{std} = 4$ (Fig. 6b). Fig. 7 shows the obtained curves under these experimental conditions. Each point in a graph corresponds to the average over 300 trials.

The graphs show that JT, GA, and PRL are much more robust under jitter than SB, confirming the known fact that spectral methods are very sensitive to structural corruption (which is one of the reasons why significant research effort has been recently dedicated to alleviating this problem with spectral techniques [4], [5], [40], [53]). The graphs also show that, when the pattern sizes are increased, GA and PRL are still very robust across the whole range (the curves are almost horizontal), whereas JT is more sensitive to high jitter. However, it is clear that JT is competitive for small to moderate jitter ($\mathrm{std} = 0$-2). The curves for GA and PRL essentially just undergo a change in offset for different pattern sizes, which reveals decreasing performance in the low jitter region. PRL is particularly more sensitive than GA for large matching problems, as reported in [13].

In the second experiment, we held the size of the domain pattern $\mathcal{T}$ constant (10 nodes) and varied the size of the codomain pattern $\mathcal{S}$ (from 10 to 35 nodes in steps of 5) for various jitter levels ($\mathrm{std} = 1, 2, 3, 4$). In this experiment, we compared JT, GA, and PRL only, since SB is not suited for graphs with different sizes. Fig. 8 shows the results of this experiment. Each point in a graph corresponds to the average over 300 trials. Clearly, the accuracy of JT does not degrade significantly for larger codomain patterns, even under high jitter, whereas the performances of GA and PRL begin to fail dramatically.

Overall, it is possible to summarize the results as follows: JT always outperforms SB and PRL in all the operating regions described in the experiments. When comparing JT to GA, the only case where GA outperforms JT is for patterns of equal sizes *and* moderate to high jitter. In all other cases—when 1) the patterns have equal sizes and noise is small, or 2) the patterns have different size, regardless of jitter—JT outperforms GA. Note in particular the outstanding performance of JT for patterns of different size (Fig. 8): in this case, the advantage over all the competing techniques, including GA, is dramatic.

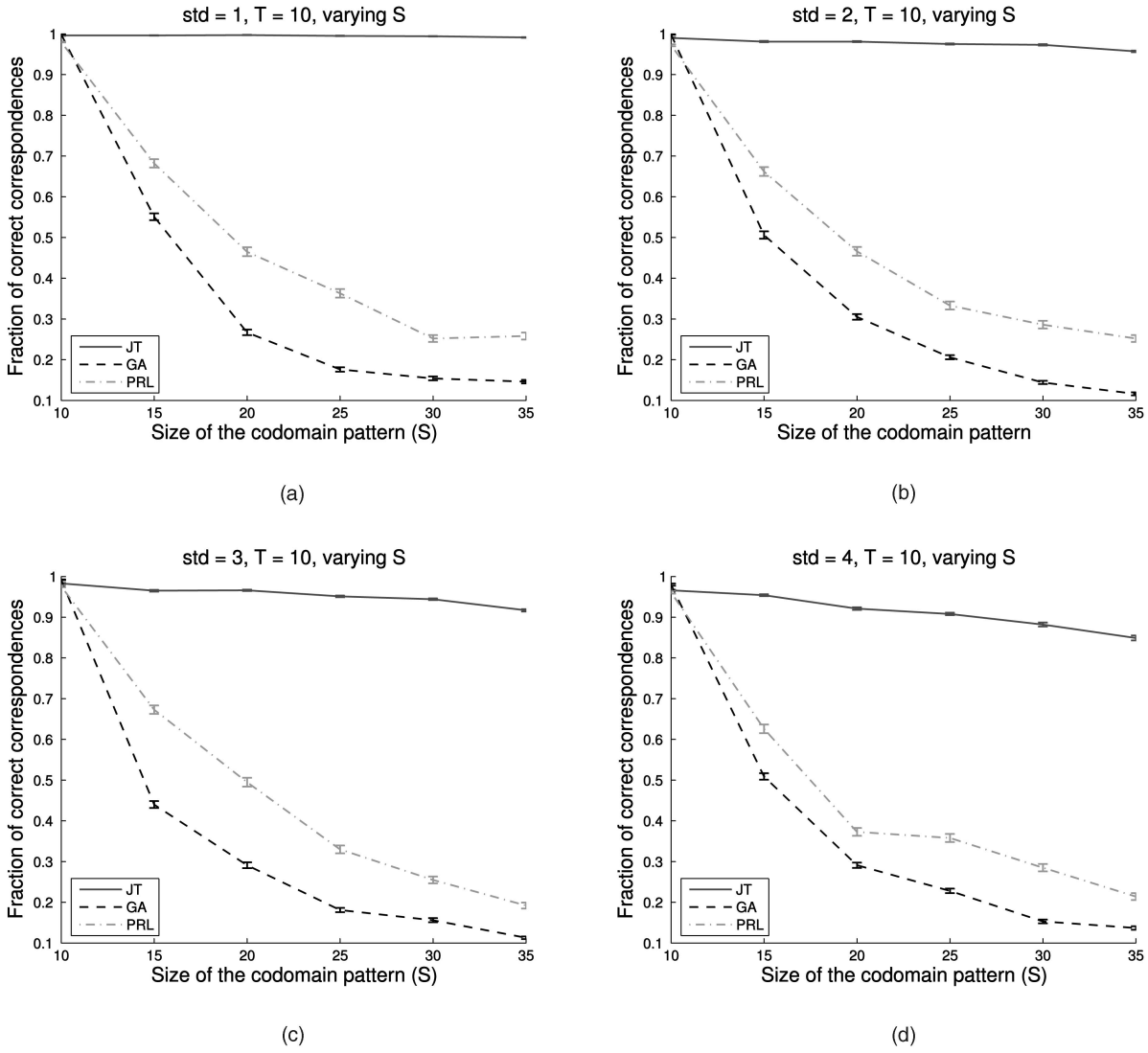

Fig. 8. Comparison of JT, GA, and PRL for matching under varying relative sizes. Results for various levels of jitter ((a) $\mathrm{std} = 1$, (b) $\mathrm{std} = 2$, (c) $\mathrm{std} = 3$, (d) $\mathrm{std} = 4$). Error bars denote standard errors.

7.2 Image Data

We also conducted experiments on image data to evaluate the techniques on a realistic computer vision problem. In the real-world experiments, we used the CMU house sequence available at http://vasc.ri.cmu.edu/idb/html/motion/house/index.html. This database consists of 111 frames of a moving sequence of a toy house.

We matched all images spaced by 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100 frames and computed the average correct correspondence. Since there are 111 frames, note that the number of image pairs spaced by these numbers of frames are, respectively, 101, 91, ... , 11.

Fig. 9 shows typical images separated by these quantities of frames.

Since all the images have size 384×576 (contrary to the synthetic experiments, where the point sets lay on $x, y = [0, 1]$), we need to use, accordingly, a large value for σ (see (7)) that does not underflow the kernel computation for very large deviations in the pairwise assignments. Here, we used $\sigma = 150$. Also, as in the synthetic experiments, the three reference points for the 3-tree were selected randomly.

In total, 30 landmark points were manually marked in each of the images. We then conducted four different experiments. We matched 15 against 30, 20 against 30, 25 against 30, and, finally, 30 against 30 points for every image pair in the experimental setting defined above. This allows us to evaluate how the techniques perform in a real problem with different point set sizes. For the 30×30 case, we ran all four techniques (JT, GA, PRL, and SB), whereas, for the remaining cases, SB was not used since it is not suitable for patterns of different sizes, as already mentioned.

Fig. 10 shows the results for these experiments. The average value is taken over different spacings between image pairs in the frame sequence.

Here, we observe that, as the relative sizes of the patterns become progressively different, the advantage of JT over the other techniques increases, which agrees with the results from the synthetic experiments. For the reported values of different pattern sizes, JT performs significantly better than the competing methods, even for a wide baseline. Although

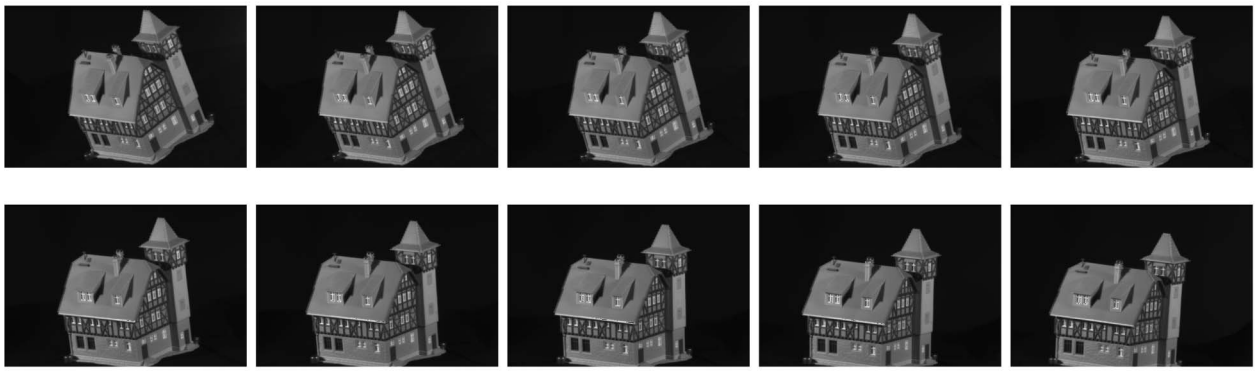

Fig. 9. Images from the CMU house sequence. (top) Frames 1, 11, . . . , 41. (bottom) Frames 51, 61, . . . , 91.

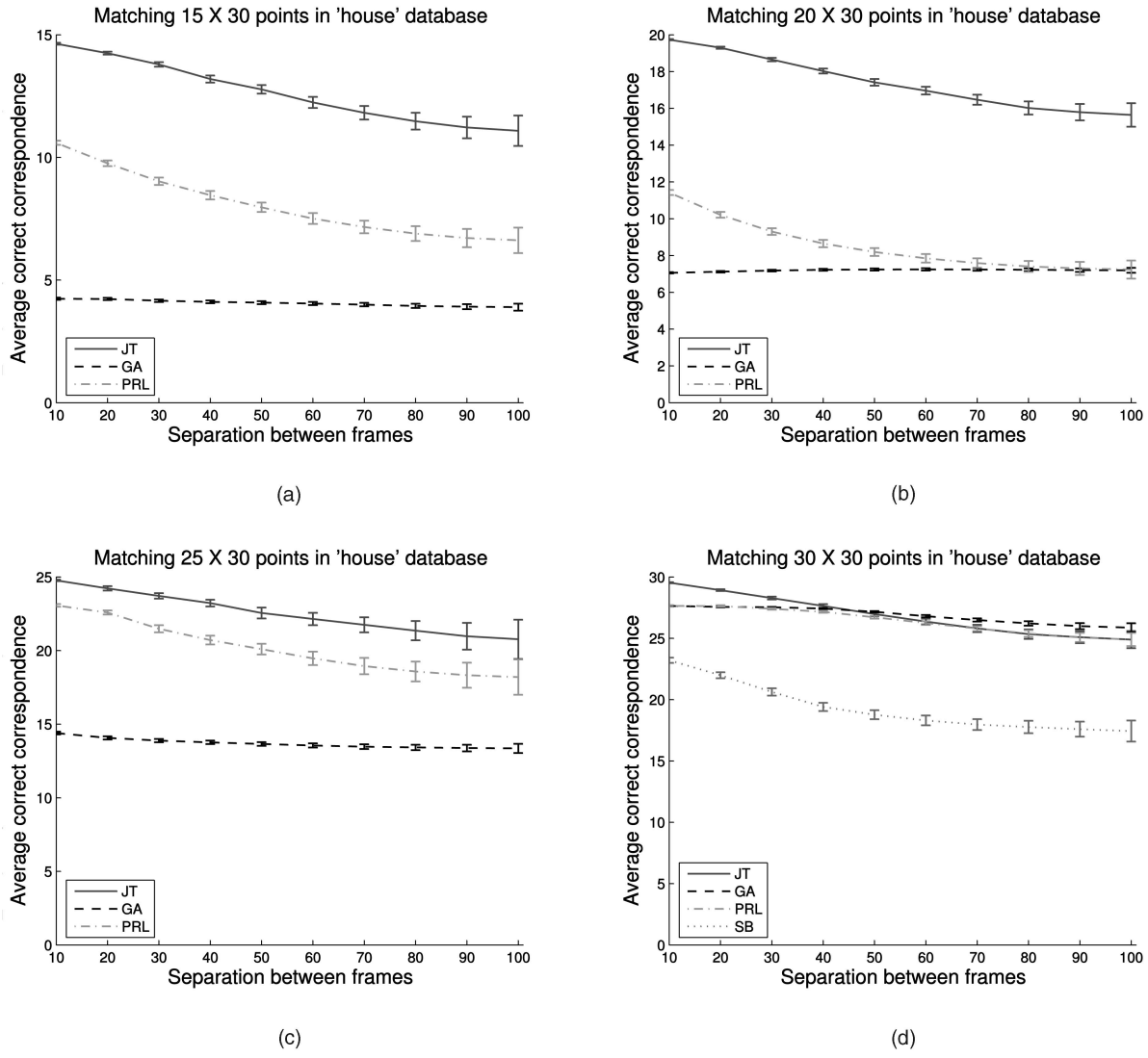

Fig. 10. Performances of JT, GA, PRL, and SB in the CMU houses sequence for increasing baselines (10 to 100 frame separation) and different domain/codomain sizes ((a) 15/30, (b) 20/30, (c) 25/30, and (d) 30/30). Error bars correspond to standard errors.

the isometric assumption clearly does not hold for a wide baseline, we note that the same pairwise distances were used as features in *every* algorithm, so we expect this to be a fair comparison.[8] For the experiment with patterns of the same size (30×30), JT has similar performance to GA and PRL when the baseline is increased. For the narrow baseline case, JT slightly outperforms the competing methods.

Our results in the real-world experiments lead us to conclude that, in the particular real-world problem of stereo correspondence, JT finds its best applicability in the narrow baseline case for patterns of different sizes where the

8. It should be clear that, in real problems, any of these approaches, since they rely exclusively on distance features, will only be competitive in the narrow baseline case.

isometric assumption holds (apart from jitter, of course). This finding agrees with the results on synthetic data.

Given the positive results on both synthetic and real-world experiments where there is a narrow baseline, we expect that, in other applications, like those involving matching of star constellations, flexible ligands, and protein motifs (where the isometric assumption also holds with good approximation), the proposed technique should perform similarly well.

7.3 Empirical Performance Evaluation for Varying k-trees

We mentioned previously that there is a theoretical problem that remains unsolved in this framework: how to select the k-tree when there is position jitter. When there is no jitter, any k-tree will find the same—optimal—solution. However, when jitter is present, different k-trees could result in different degrees of accuracy. Here, we provide empirical evidence of this fact by measuring the variance of the performance of a matching task over a range of possible choices of k-trees.

We used the same setting of the previous synthetic experiments: random point patterns in $[0,1]^2$. We randomly generated 100 domain-codomain pairs in this range, each with 20 points. For each of these 100 configurations, we randomly selected 100 3-trees for the domain pattern. Finally, the JT algorithm was run in each of these 100 3-trees, and the standard error for the fraction of correct correspondences recorded. This allows us to measure the performance variability within the same pair domain-codomain over a wide selection of 3-trees. In order to compute the aggregate over the 100 configurations, we simply averaged the standard errors. This whole procedure was performed for jitter levels of $\mathrm{std} = 0, 1, 2, 3, \text{and } 4$. The final average standard errors were, respectively, 0, 0.001, 0.003, 0.007, and 0.011 (measured in fractions of correct correspondences, which vary from 0 to 1). This indicates, as expected, that the influence on the choice of k-tree becomes more significant as jitter increases (in particular, it is zero when there is no jitter, confirming the theoretical results).

7.4 Processing Times

The computational complexity of the proposed method is higher than in the other approaches. Spectral, graduated assignment, and relaxation methods are, respectively, $O(T^3)$ $(S = T)$, $O(T^2S^2)$, and $O(T^2S^3)$, while the proposed junction tree approach is $O(TS^4)$ (for matching in $\mathbb{R}^2$). However, the graphs showing real processing times (see Fig. 11) indicate that, even for a reasonable size, like 40, the actual running time is just twice that of PRL and four times that of GA. For graphs with about 30 nodes, the technique is about as fast as relaxation labeling.

8 Discussion and Future Work

A matching algorithm should, ideally, present high robustness with respect to jitter as well as with respect to size increases of the patterns. Our experiments revealed essentially two findings. First, for matching patterns of the same size, the proposed method is very robust to small and moderate position jitter and reasonably robust to high position jitter. Second, and more important, the method is extremely robust to increasing differences in the pattern sizes. In experiments where the sizes of the two patterns are significantly different, the performance of JT is by far superior to that of the alternative methods. This is an important result because in many relevant application domains, the problem of finding a "small" model within a "large" scene is of primary concern. (A typical such scenario that arises in computer vision is model-based object recognition in cluttered scenes.) We believe that the approach presented in this paper indicates a new direction in the search for robust algorithms for subgraph matching, where the sizes of the graphs can differ significantly.

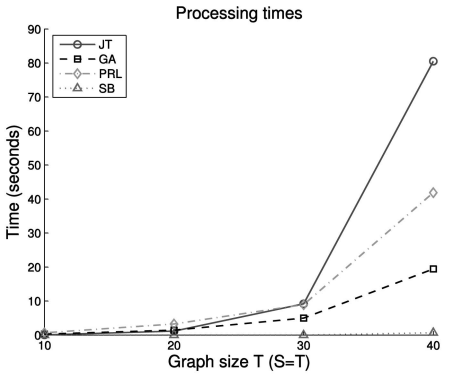

Fig. 11. Processing times for JT, GA, PRL, and SB, all in MATLAB implementations running on a 3.2 GHz Pentium with 1 GB of RAM.

There are several ways in which the current work can be extended. First, by considering higher-order potentials, one might be able to cope with more complex invariances, such as invariance under affine transformations. Second, non-rigid matching might be attainable by augmenting the clique potentials with terms that allow for some kind of nonlinear transformation. Third, theoretical results on the accuracy of the method for the noisy case can be investigated. Fourth, the framework should be extended to deal robustly with outliers. Also, a deeper understanding of the noisy case might lead to a principled technique for the k-tree selection problem.

9 Conclusion

This paper proposed a new solution to the rigid point pattern matching problem where jitter is allowed. The approach consisted of modeling the point matching task as a weighted graph matching problem and solving it using exact probabilistic inference in an appropriately designed graphical model. By using graph rigidity arguments, we showed that this graphical model, while allowing for exact MAP computation in polynomial time, still remains equivalent to the fully connected model in the noiseless limit. Contrary to many alternative heuristic approaches, the method we obtain is built from first principles, is non-iterative, obtains results independent of initialization, and provably finds a global optimum in polynomial time in the exact matching case. For inexact matching, our experiments indicate that the proposed technique is more accurate than standard methods when matching patterns of different sizes.

APPENDIX A

Proof of Lemma 1. We use induction over n. Recall that a sphere in a vector space is the set of points equidistant from a fixed point.

The Lemma obviously holds for the base case when $n = 1$.

Now, let $S_1 \cap S_2 = I_1$. Then, I_1 is an $(n-2)$-sphere lying in an $(n-1)$ vector subspace Q. (We use the convention of topology, which states that an $(n-2)$-sphere is spanned necessarily by a *complete* basis in $\mathbb{R}^{n-1}$. For example, the 3D sphere in $\mathbb{R}^3$ is a 2-sphere, not a 3-sphere.) Let $I_i = S_{i+1} \cap Q$ for $i = 2, 3, \ldots, n$. Then, $I_1, I_2, \ldots, I_n$ are n spheres in $Q \cong \mathbb{R}^{n-1}$ ($\cong$ denotes congruency) and, obviously, $\cap_{j=1}^{n} I_j = \cap_{i=1}^{n+1} S_i$. Given the above definitions, the natural induction hypothesis that arises is: If the centers of the spheres $I_1, I_2, \ldots, I_n$ do not lie in an $(n-2)$ vector subspace, then the intersection of these spheres consists of at most a single point. Since $\cap_{j=1}^{n} I_j = \cap_{i=1}^{n+1} S_i$, we have, from the hypothesis, that $\cap_{i=1}^{n+1} S_i$ consists of at most a single point. So, what is left to prove is that the centers of the spheres $S_1, S_2, \ldots, S_{n+1}$ do not lie in an $(n-1)$-dimensional vector space (i.e., are in general position). Let $(x_1, x_2, \ldots, x_n)$ be the coordinates of $\mathbb{R}^n$. Let $(a_{i1}, a_{i2}, \ldots, a_{in})$ be the center of S_i. Without loss of generality, we may assume that Q is given by $x_1 = 0$. Then, $Q \cong \mathbb{R}^{n-1}$ is parameterized by $(x_2, x_3, \ldots, x_n)$. The center of I_{j-1} has coordinates $(a_{j2}, a_{j3}, \ldots, a_{jn})$, $j \geq 2$. The centers of $I_1, I_2, \cdots, I_n$ are in general position if and only if the matrix

$$\begin{bmatrix} a_{22} & a_{23} & \ldots & a_{2n} & 1 \\ a_{32} & 33 & \ldots & a_{3n} & 1 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ a_{n+1,2} & a_{n+1,3} & \ldots & a_{n+1,n} & 1 \end{bmatrix} \tag{11}$$

is invertible, i.e., has maximal rank.

But, this matrix is precisely the $n \times n$ lower-right submatrix of the following matrix:

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} & 1 \\ a_{21} & a_{22} & \ldots & a_{2n} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n+1,1} & a_{n+1,2} & \ldots & a_{n+1,n} & 1 \end{bmatrix}, \tag{12}$$

which is the analogous matrix for the centers of $S_1, S_2, \ldots, S_{n+1}$. By subtracting the second row from the first row of (12), we obtain

$$\begin{bmatrix} a_{11} - a_{21} & 0 & \ldots & 0 & 0 \\ a_{21} & a_{22} & \ldots & a_{2n} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n+1,1} & a_{n+1,2} & \ldots & a_{n+1,n} & 1 \end{bmatrix}. \tag{13}$$

Note that $Q = \{x_1 = 0\}$ implies that

$$(a_{12}, a_{13}, \ldots, a_{1n}) = (a_{22}, a_{23}, \ldots, a_{2n}),$$

which creates the zeros in the first row. It is evident that (13) is invertible if and only if $a_{11} \neq a_{21}$ and (11) is invertible, which is the induction hypothesis. This implies that the centers of S_i do not lie in an $(n-1)$ vector subspace in $\mathbb{R}^n$, which completes the proof. □

Proof of Lemma 2. We use induction on the number of vertices n in the k-tree framework. For $n = k$, the result is obvious because the graph is simply a k-clique, which is fully connected and, by definition, is globally rigid. Now, assume the lemma is true for some $n > k$. First, choose a fixed (but arbitrary) coordinate system S. If the lemma holds for some $n > k$, then all the points in the framework are determined in S. Now, include a new vertex with given distances from all the k vertices of any existent base k-clique in general position. By drawing edges corresponding to these known distances, we generate a new framework with $n+1$ vertices. But, since the inserted vertex has determined distances from all vertices of a base k-clique, which is in general position, its position is determined in S by virtue of Lemma 1. If its position is determined in S, then the positions of all vertices in the new framework are determined in S (because the previous framework was globally rigid by the induction hypothesis). If all the positions are determined in S, all pairwise distances are determined (irrespectively of S). This guarantees that the new framework is globally rigid in $\mathbb{R}^{k-1}$, which completes the proof. □

Proof of Theorem 1. If we define a cost function over the complement graph of G_d^{kt} ($\bar{G}_d^{kt}$),

$$U_{\bar{G}_d^{kt}}(f) = \sum_{i,j | d_{ij} \in \vec{\mathcal{E}}_d^{kt}} \mathcal{D}\Big(y_{ij}^d, y_{f(i)f(j)}^c\Big), \tag{14}$$

we have

$$U_T(f) = U_{G_d^{kt}}(f) + U_{\bar{G}_d^{kt}}(f). \tag{15}$$

In the noiseless case, the dissimilarity function $\mathcal{D}(\cdot,\cdot)$ associated with a particular match is described simply in terms of an indicator function (6):

$$\mathcal{D}\Big(y_{ij}^d, y_{f(i)f(j)}^c\Big) = 1 - \mathbf{1}\Big(y_{ij}^d = y_{f(i)f(j)}^c\Big). \tag{16}$$

The optimal matching function f is such that $U_T(f) = 0$. Obviously, from (15), it holds that

$$U_T(f) = 0 \Rightarrow U_{G_d^{kt}}(f) = 0,$$

since $U_{G_d^{kt}}(f)$ and $U_{\bar{G}_d^{kt}}(f)$ are nonnegative because $\mathcal{D}(\cdot,\cdot)$ is nonnegative (8) and (14). Our purpose is to prove the converse, i.e., that $U_{G_d^{kt}}(f) = 0 \Rightarrow U_T(f) = 0$. According to (15), in order to do so, it suffices to prove that

$$U_{G_d^{kt}}(f) = 0 \Rightarrow U_{\bar{G}_d^{kt}}(f) = 0.$$

Here, we use Lemma 2, which asserts that if the distances corresponding to the edges of a k-tree framework whose base k-cliques in general position are determined, then all the remaining distances between vertices not connected by an edge are determined.

Let us write this result symbolically, for a k-tree in the domain graph, as

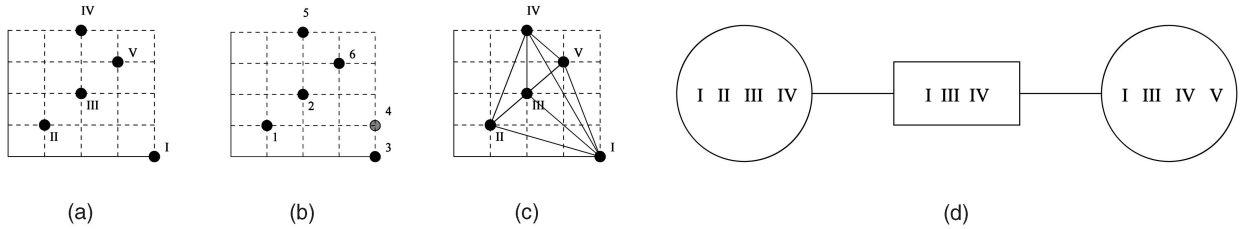

Fig. 12. A simple example of point pattern matching. (a) The domain pattern. (b) The codomain pattern. (c) A possible 3-tree in the domain pattern. (d) The resulting junction tree.

$$\{y_{ij}^d = const_{ij}^d, \forall i,j | d_{ij} \in \mathcal{E}_d^{kt}\} \Rightarrow \{y_{ij}^d = const_{ij}^d, \forall i,j | d_{ij} \in \bar{\mathcal{E}}_d^{kt}\}, \tag{17}$$

where $const_{ij}^d$ is a constant for fixed i and j.

Since $U_{G_d^{kt}}(f) = 0$, every term of the sum in (8) must be zero, since they are nonnegative:

$$\mathcal{D}(y_{ij}^d, y_{f(i)f(j)}^c) = 0, \forall i,j | d_{ij} \in \mathcal{E}_d^{kt}. \tag{18}$$

However, from the definition of $\mathcal{D}(\cdot,\cdot)$ for exact matching (16), this means that

$$y_{ij}^d = y_{f(i)f(j)}^c, \forall i,j | d_{ij} \in \mathcal{E}_d^{kt}. \tag{19}$$

Notice that the statement (17) holds for any k-tree whose base k-cliques are in general position, so it holds for G_c^{kt} in particular. (Recall that G_c^{kt} has its base k-cliques in general position in $\mathbb{R}^{k-1}$ because it is assumed isometric to G_d^{kt}, which, by assumption, has its base k-cliques in general position and, so, is globally rigid.) Therefore, we conclude

$$\{y_{f(i)f(j)}^c = const_{f(i)f(j)}^c, \forall i,j | d_{f(i)f(j)} \in \mathcal{E}_c^{kt}\} \Rightarrow \{y_{f(i)f(j)}^c = const_{f(i)f(j)}^c, \forall i,j | c_{f(i)f(j)} \in \bar{\mathcal{E}}_c^{kt}\}. \tag{20}$$

Notice that (19) implies that the left-hand sides of implications (17) and (20) are equivalent. As a result, their right-hand sides are equivalent and we obtain

$$y_{ij}^d = y_{f(i)f(j)}^c, \forall i,j | d_{ij} \in \bar{\mathcal{E}}_d^{kt}. \tag{21}$$

Substituting this into (14) yields

$$U_{\bar{G}_d^{kt}}(f) = 0, \tag{22}$$

which was what we wanted to prove. □

APPENDIX B

Fig. 12 shows a very simple instance of a point pattern matching problem, a corresponding 3-tree selection, and the associated junction tree. In this appendix, we are going to present a detailed description of the proposed algorithm when applied to this specific instance, for didactic purposes. This is one of the simplest types of examples that we can show, since five points in the domain is the minimal amount required to motivate the use of our algorithm (in the four-point case, a 3-tree is the fully connected graph and the algorithm reduces to a brute-force approach). The correct correspondence in this case is

$$\{I \mapsto 3, II \mapsto 1, III \mapsto 2, IV \mapsto 5, V \mapsto 6\}.$$

Node 4 in the codomain is the only difference between the two patterns and no point in the domain should map to it.

The first step of the algorithm consists in selecting a 3-tree for the domain graph. This is done randomly, provided that nodes II, III, and V do *not* constitute the base 3-clique (since they are collinear). A possible 3-tree is shown in Fig. 12c. The second step of the algorithm consists in constructing the junction tree and the potential functions for the graphical model induced by the 3-tree. The junction tree is shown in Fig. 12d. A potential function will associate a nonnegative real number to each possible instantiation of the four variables in a maximal clique (i.e., a "score" for the corresponding 4-wise map). This potential function is built from pairwise potential functions by combining their scores. For example, the pairwise potential function $\psi_{I,II}$ would have the following form (we use $\sigma = 1/\sqrt{2}$ in (7) for simplicity and the unit length is the side of an elementary square in the grids of Fig. 12):

$$\psi_{I,II} = \exp\left[-\begin{pmatrix} (\sqrt{10}-0)^2 & (\sqrt{10}-\sqrt{2})^2 & (\sqrt{10}-\sqrt{10})^2 & (\sqrt{10}-3)^2 & (\sqrt{10}-\sqrt{10})^2 & (\sqrt{10}-2\sqrt{2})^2 \\ (\sqrt{10}-\sqrt{2})^2 & (\sqrt{10}-0)^2 & (\sqrt{10}-2\sqrt{2})^2 & (\sqrt{10}-\sqrt{5})^2 & (\sqrt{10}-2)^2 & (\sqrt{10}-\sqrt{2})^2 \\ (\sqrt{10}-\sqrt{10})^2 & (\sqrt{10}-2\sqrt{2})^2 & (\sqrt{10}-0)^2 & (\sqrt{10}-1)^2 & (\sqrt{10}-2\sqrt{5})^2 & (\sqrt{10}-\sqrt{10})^2 \\ (\sqrt{10}-3)^2 & (\sqrt{10}-\sqrt{5})^2 & (\sqrt{10}-1)^2 & (\sqrt{10}-0)^2 & (\sqrt{10}-\sqrt{13})^2 & (\sqrt{10}-\sqrt{5})^2 \\ (\sqrt{10}-\sqrt{10})^2 & (\sqrt{10}-2)^2 & (\sqrt{10}-2\sqrt{5})^2 & (\sqrt{10}-\sqrt{13})^2 & (\sqrt{10}-0)^2 & (\sqrt{10}-\sqrt{2})^2 \\ (\sqrt{10}-2\sqrt{2})^2 & (\sqrt{10}-\sqrt{2})^2 & (\sqrt{10}-\sqrt{10})^2 & (\sqrt{10}-\sqrt{5})^2 & (\sqrt{10}-\sqrt{2})^2 & (\sqrt{10}-0)^2 \end{pmatrix}\right], \tag{23}$$

where $\exp(M)$ is the exponential of the elements of M (not the matrix exponential). For example, entry $(5,6)$ in $\psi_{I,II}$ is the "likelihood" of the pairwise map $\{I \mapsto 5, II \mapsto 6\}$. This entry must be smaller than, say, entry $(1,5)$, because the distances $d_{I,II}$ and $d_{5,6}$ are more different ($\sqrt{10}$ and $\sqrt{2}$, respectively) than the distances $d_{I,II}$ and $d_{1,5}$ ($\sqrt{10}$ and $\sqrt{10}$). In summary, $\psi_{I,II}$ has *higher* values in those entries whose correspondent points are separated by a distance which is *more* similar to the distance between I and II (i.e., $\sqrt{10}$). This reflects the idea that, in rigid point pattern matching, pairwise maps should preserve the distance between points. The same type of table is constructed for all the other pairs in the domain which correspond to an edge of the 3-tree, not only for the pair I-II. (Here, we can observe that, even for the trivial example shown, the number of numeric values involved is still very large, which prevents us from displaying all the numeric details of the computations that follow.) Once all the required pairwise tables have been constructed, one must assemble the potentials of the

maximal cliques, which are those to be used in the propagation phase. These are given by

$$\psi_{I,II,III,IV}(i,j,k,l) = \psi_{I,II}(i,j)\psi_{I,III}(i,k)\psi_{I,IV}(i,l)\psi_{II,III}(j,k)\psi_{II,IV}(j,l)\psi_{III,IV}(k,l)$$

and

$$\Psi_{I,III,IV,V}(i,j,k,l) = \psi_{I,V}(i,l)\psi_{III,V}(j,l)\psi_{IV,V}(k,l)$$

(or the other way around: the choice of which 4-wise potential has six factors is arbitrary). This operation can be understood in a vectorized form as simply replicating the 2D tables (like those in (23)) across the two lacking dimensions and then performing an entrywise multiplication of all the resulting 4D tables. Once the potentials for the maximal cliques are initialized, we initialize the potential for the separator as simply being $\Phi_{I,III,IV}(i,j,k) = 1$, for all i,j,k.

The next step in the algorithm is the propagation phase. Notice that, at this point, one has three tables: $\Psi_{I,II,III,IV}$, $\Psi_{I,III,IV,V}$, and $\Phi_{I,III,IV}$. Each Ψ indicates an initial "guess" of the likelihood of every one of the 6^4 possible instantiations that its four variables may assume. However, at this stage, the two Ψs may be inconsistent. That is why the propagation phase must be run, which will change these potentials so that they become consistent. The entire propagation algorithm for this simple example reduces to merely applying (9) and (10). For example, we may first update $\Psi_{I,III,IV,V}$ by doing

$$\Phi^*_{I,III,IV} = \max_{II} \Psi_{I,II,III,IV}, \tag{24}$$

$$\Psi^*_{I,III,IV,V} = \frac{\Phi^*_{I,III,IV}}{\Phi_{I,III,IV}} \Psi_{I,III,IV,V}, \tag{25}$$

and then update $\Psi_{I,II,III,IV}$ by doing

$$\Phi^{**}_{I,III,IV} = \max_{V} \Psi^*_{I,III,IV,V}, \tag{26}$$

$$\Psi^*_{I,II,III,IV} = \frac{\Phi^{**}_{I,III,IV}}{\Phi^*_{I,III,IV}} \Psi_{I,II,III,IV}, \tag{27}$$

where all operations are entrywise in the tables (3D Φs are replicated across the lacking dimension after the quotient is computed, so that they can be entrywise multiplied by the 4D Ψs).

After this propagation algorithm has run, one is assured that the final clique potentials $\Psi^*_{I,II,III,IV}$ and $\Psi^*_{I,III,IV,V}$ are equal to the max-marginal distributions over the four respective variables (apart from a common constant factor). What this allows us to do is to simply compute the modes of the individual max-marginals by local maximization within the cliques. For example, one can compute the correct assignment $II^\dagger$ for point II by

$$II^\dagger = \arg\max_{II} \max_{I,III,IV} \Psi^*_{I,II,III,IV} \tag{28}$$

and the correct assignment $V^\dagger$ for point V by

$$V^\dagger = \arg\max_{V} \max_{I,III,IV} \Psi^*_{I,III,IV,V}. \tag{29}$$

The correct assignment for the remaining 3 points in the domain pattern can be computed by marginalizing over either one of the Ψs, since they are present in both. This results in the final correct assignment

$$\{I \mapsto 3, II \mapsto 1, III \mapsto 2, IV \mapsto 5, V \mapsto 6\}.$$

ACKNOWLEDGMENTS

The basic ideas in this paper were first presented in [1] and [2]. National ICT Australia is funded by the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council. This work was initiated when the first three authors were with the Department of Computing Science, University of Alberta. The authors would like to thank NSERC, CFI, and the Alberta Ingenuity Centre for Machine Learning for their support. The first author thanks CAPES for financial support. Also, the authors kindly thank X. Chen for a discussion that led to the idea behind the proof of Lemma 1.

REFERENCES

[1] T.S. Caetano, T. Caelli, and D.A.C. Barone, "An Optimal Probabilistic Graphical Model for Point Set Matching," *Proc. Joint Int'l Workshops Syntactical and Structural Pattern Recognition and Statistical Pattern Recognition (SSPR & SPR),* pp. 162-170, 2004.

[2] T.S. Caetano, "Graphical Models and Point Set Matching," PhD dissertation, Universidade Federal do Rio Grande do Sul (UFRGS), 2004.

[3] R. Myers, R.C. Wilson, and E.R. Hancock, "Bayesian Graph Edit Distance," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 6, pp. 628-635, June 1999.

[4] M. Carcassoni and E.R. Hancock, "Spectral Correspondence for Point Pattern Matching," *Pattern Recognition,* vol. 36, pp. 193-204, 2003.

[5] H. Wang and E.R. Hancock, "A Kernel View of Spectral Point Pattern Matching," *Proc. Joint Int'l Workshops Syntactical and Structural Pattern Recognition and Statistical Pattern Recognition (SSPR & SPR),* pp. 361-369, 2004.

[6] M.T. Goodrich and J.S.B. Mitchell, "Approximate Geometric Pattern Matching under Rigid Motions," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 4, pp. 371-379, 1999.

[7] F. Murtagh, "A New Approach to Point-Pattern Matching," *Astronomical Soc. Pacific,* vol. 104, no. 674, pp. 301-307, 1992.

[8] G. Weber, L. Knipping, and H. Alt, "An Application of Point Pattern Matching in Astronautics," *J. Symbolic Computation,* no. 11, pp. 1-20, 1994.

[9] Y. Martin, M. Bures, E. Danaher, J. Delazzer, and I. Lico, "A Fast New Approach to Pharmacophore Mapping and Its Application to Dopaminergic and Bezodiazepine Agonists," *J. Computer-Aided Molecular Design,* vol. 7, pp. 83-102, 1993.

[10] P.W. Finn, L.E. Kavraki, J.-C. Latombe, R. Motwani, C. Shelton, S. Venkatasubramanian, and A. Yao, "Rapid: Randomized Pharmacophore Identification for Drug Design," *Computational Geometry,* vol. 10, pp. 263-272, 1998.

[11] T. Akutsu, K. Kanaya, A. Ohyama, and A. Fujiyama, "Point Matching under Non-Uniform Distortions," *Discrete Applied Math.,* special issue: computational biology series IV, pp. 5-21, 2003.

[12] R. Nussinov and H.J. Wolfson, "Efficient Detection of Three-Dimensional Structural Motifs in Biological Macromolecules by Computer Vision Techniques," *Proc. Nat'l Academy Sciences,* vol. 88, pp. 10,495-10,499, 1991.

[13] 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.

[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] Y. Keselman, A. Shokoufandeh, M.F. Demirci, and S. Dickinson, "Many-to-Many Graph Matching via Metric Embedding," *Proc. Int'l Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 18-20, 2003.

[16] M. Demirci, A. Shokoufandeh, S. Dickinson, Y. Keselman, and L. Bretzner, "Many-to-Many Feature Matching Using Spherical Coding of Directed Graphs," *Proc. European Conf. Computer Vision,* pp. 322-335, 2004.

[17] J. Ullman, "An Algorithm for Subgraph Isomorphism," *J. ACM,* vol. 23, no. 1, pp. 31-42, 1976.

[18] 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.

[19] S. Berreti, A.D. Bimbo, and E. Vicario, "Efficient Matching and Indexing of Graph Models in Content-Based Retrieval," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 10, pp. 1089-1105, Oct. 2001.

[20] K.S. Fu, "A Step Towards Unification of Syntactic and Statistical Pattern Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 5, no. 2, pp. 200-205, 1983.

[21] M. Eshera and K. Fu, "A Graph Distance Measure for Image Analysis," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 14, no. 3, pp. 353-363, 1984.

[22] W.H. Tsai and K.S. Fu, "Subgraph Error-Correcting Isomorphisms for Syntactic Pattern Recognition," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 13, no. 1, pp. 48-62, 1983.

[23] K.L. Boyer and A.C. Kak, "Structural Stereopsis for 3-D Vision," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 10, no. 2, pp. 144-166, Mar. 1988.

[24] B. Bhanu and O.D. Faugeras, "Shape Matching of Two-Dimensional Objects," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 6, no. 2, pp. 137-156, 1984.

[25] L.S. Davis, "Shape Matching Using Relaxation Techniques," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 1, no. 1, pp. 60-72, 1979.

[26] O.D. Faugeras and M. Berthod, "Improving Consistency and Reducing Ambiguity in Stochastic Labeling: An Optimization Approach," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 3, pp. 412-423, 1981.

[27] S. Ulmann, "Relaxation and Constraint Optimization by Local Process," *Computer Graphics and Image Processing,* vol. 10, pp. 115-195, 1979.

[28] A. Rosenfeld and A.C. Kak, *Digital Picture Processing.* New York: Academic Press, 1982.

[29] 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, Aug. 1994.

[30] S.Z. Li, "A Markov Random Field Model for Object Matching under Contextual Constraints," *Proc. Int'l Conf. Computer Vision and Pattern Recognition,* pp. 866-869, 1994.

[31] R.A. Hummel and S.W. Zucker, "On the Foundations of Relaxations Labeling Process," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 5, no. 3, pp. 267-286, 1983.

[32] A. Rosenfeld, R. Hummel, and S. Zucker, "Scene Labeling by Relaxation Operations," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 6, pp. 420-433, 1976.

[33] 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.

[34] E. Hancock and R.C. Wilson, "Graph-Based Methods for Vision: A Yorkist Manifesto," *Proc. Joint Int'l Workshops Syntactical and Structural Pattern Recognition and Statistical Pattern Recognition (SSPR & SPR),* pp. 31-46, 2002.

[35] 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.

[36] S. Umeyama, "An Eigen Decomposition Approach to Weighted Graph Matching Problems," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 10, pp. 695-703, 1998.

[37] L. Shapiro and J. Brady, "Feature-Based Correspondence—An Eigenvector Approach," *Image and Vision Computing,* vol. 10, pp. 283-288, 1992.

[38] B.J. van Wyk and M.A. van Wyk, "Kronecker Product Graph Matching," *Pattern Recognition,* vol. 36, no. 9, pp. 2019-2030, 2003.

[39] M.A. van Wyk, T.S. Durrani, and B.J. 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.

[40] A. Robles-Kelly and E.R. Hancock, "Graph Edit Distance From Spectral Seriation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 27, no. 3, pp. 365-378, Mar. 2005.

[41] P.N. Suganthan, "Structural Pattern Recognition Using Genetic Algorithms," *Pattern Recognition,* vol. 35, pp. 1883-1893, 2002.

[42] M. Pelillo, "Replicator Equations, Maximal Cliques, and Graph Isomorphism," *Neural Computing* vol. 11, pp. 1933-1955, 1999.

[43] M. Pelillo, K. Siddiqi, and S. Zucker, "Matching Hierarchical Structures Using Association Graphs," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 11, pp. 1105-1120, Nov. 1999.

[44] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty Years of Graph Matching in Pattern Recognition," *Int'l J. Pattern Recognition and Artificial Intelligence,* special edition on graph theory in vision, vol. 18, no. 3, pp. 265-298, 2004.

[45] T. Caelli and T. Caetano, "Graphical Models for Graph Matching: Approximate Models and Optimal Algorithms," *Pattern Recognition Letters,* vol. 26, pp. 339-346, 2005.

[46] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* Cambridge, U.K.: Cambridge Univ. Press, 2000.

[47] J. Mejía, T. Villela, and J. Braga, "The CCD Stellar Sensor of the Masco Telescope Pointing System," *Advances in Space Research,* vol. 26, no. 9, pp. 1407-1410, 2000.

[48] P.J. de Rezende and D.T. Lee, "Point Set Pattern Matching in D-Dimensions," *Algorithmica,* vol. 13, pp. 387-404, 1995.

[49] H. Chui and A. Rangarajan, "A New Algorithm for Non-Rigid Point Matching," *Proc. Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 44-51, 2000.

[50] A. Rangarajan, A. Yuille, and E. Mjolsness, "Convergence Properties of the Softassign Quadratic Assignment Algorithm," *Neural Computation,* vol. 11, pp. 1455-1474, 1999.

[51] R. Sinkhorn, "A Relationship Between Arbitrary Positive Matrices and Doubly Stochastic Matrices," *Ann. Math. Statistics,* vol. 35, pp. 876-879, 1964.

[52] T. Caelli and S. Kosinov, "An Eigenspace Projection Clustering Method for Inexact Graph Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 4, pp. 515-519, Apr. 2004.

[53] A. Robles-Kelly and E.R. Hancock, "String Edit Distance, Random Walks and Graph Matching," *Int'l J. Pattern Recognition and Artificial Intelligence,* vol. 18, no. 3, pp. 315-327, 2004.

[54] S.L. Lauritzen, *Graphical Models.* New York, NY: Oxford Univ. Press, 1996.

[55] J. Dattorro, "Euclidean Distance Matrices," PhD dissertation, Stanford Univ., 2005.

[56] H.-X. Huang, Z.-A. Liang, and P.M. Pardalos, "Some Properties of the Euclidean Distance Matrix and Positive Semidefinite Matrix Completion Problems," *J. Global Optimization,* vol. 25, pp. 3-21, 2003.

[57] M.J. Wainwright, "Stochastic Processes on Graphs with Cycles: Geometric and Variational Approaches," PhD dissertation, Dept. of Electrical Eng. and Computer Science, Massachusetts Inst. of Technology, 2002.

[58] M.J. Wainwright and M.I. Jordan, "A Variational Principle for Graphical Models," *New Directions in Statistical Signal Processing: From Systems to Brains.* MIT Press, 2006.

[59] M.I. Jordan, "An Introduction to Probabilistic Graphical Models," in preparation.

[60] M.I. Jordan, "Graphical Models," *Statistical Science,* vol. 19, no. 1, pp. 140-155, 2004.

[61] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distribution and the Bayesian Restoration of Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 6, no. 6, pp. 721-741, 1984.

[62] T.S. Caetano, T. Caelli, and D.A.C. Barone, "A Comparison of Junction Tree and Relaxation Algorithms for Point Matching Using Different Distance Metrics," *Proc. IEEE Int'l Conf. Pattern Recognition,* vol. 2, pp. 124-127, 2004.

[63] R. Connelly, "Rigidity and Energy," *Inventiones Mathemticae,* vol. 66, no. 1, pp. 11-33, 1982.

[64] D.B. West, *Introduction to Graph Theory.* Upper Saddle River, N.J.: Prentice Hall, 2001.

[65] L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. IEEE,* vol. 77, no. 2, pp. 257-286, 1989.

Tibério S. Caetano received the BSc degree in electrical engineering and the PhD degree in computer science, with highest distinction, from the Universidade Federal do Rio Grande do Sul (UFRGS), Brazil. Part of the PhD program was undertaken in 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 researcher with the Statistical Machine Learning group of National ICT Australia (NICTA) and an adjunct research fellow at the Australian National University. His main research interests are in pattern recognition and machine learning and their applications to computer vision and computational biology.

Terry Caelli received the PhD degree in pattern recognition from the University of Newcastle, New South Wales, in 1975. His interests lie in computer vision, machine learning, and artificial intelligence and their applications to environmental sensing systems. He is currently the laboratory director at the National ICT Australia (NICTA) node in Canberra. He is actively involved as an associate editor of a number of journals in the area and he is a member of several conference organizations. His specific contributions lie in texture processing, invariant pattern/object recognition, and machine learning as applied to spatio-temporal information processing. He has won a number of best paper awards for journal and conference papers and was a recipient of the University of Newcastle Convocation Medal of Excellence. He is a fellow of the International Association for Pattern Recognition (IAPR), the IEEE, and the IEEE Computer Society.

Dale Schuurmans received the BSc and MSc degrees in computing science and mathematics from the University of Alberta and the PhD degree in computer science from the University of Toronto. He is a professor of computing science and the Canada Research Chair in Machine Learning at the University of Alberta. He has previously been an associate and assistant professor of computer science at the University of Waterloo, a postdoctoral fellow at the University of Pennsylvania, a researcher at the NEC Research Institute, and a research associate at the National Research Council Canada. Prof. Schuurmans is currently an action editor for the *Journal of Machine Learning Research* and the *Machine Learning Journal*, and served as program cochair for the International Conference on Machine Learning in 2004. Prof. Schuurmans' research interests include machine learning, optimization, and search. He has authored more than 70 publications in these areas and has received outstanding paper awards at the International Joint Conference on Artificial Intelligence (IJCAI) and the National Conference on Artificial Intelligence (AAAI).

Dante A.C. Barone received the bachelor's degree in electrical engineering from the Universidade Federal do Rio Grande do Sul (UFRGS), Brazil, in 1978, the MSc degree in electrical engineering from USP-Brazil in 1981, and the PhD degree in 1984 from the Institut National Polytechnique de Grenoble (INPG), France. In the latter part of 1984, he joined, as a professor, the Computer Science Institute of UFRGS. His research interests lie mainly in image processing, robotics, and artificial intelligence. Currently, he is involved in the tracking of mobile objects. He is a member of the IAPR (International Association for Pattern Recognition) and the advisors board of the SBPC (Brazilian Society for Science Development).