UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

TIBÉRIO SILVA CAETANO

Graphical Models and Point Set Matching

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Prof. Dr. Dante Augusto Couto Barone
Advisor

Prof. Dr. Terry Caelli
Coadvisor

Porto Alegre, July 2004

CIP – CATALOGING-IN-PUBLICATION

Caetano, Tibério Silva

Graphical Models and Point Set Matching / Tibério Silva Caetano. – Porto Alegre: PPGC da UFRGS, 2004.

87 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Curso de Pós-Graduação em Ciência da Computação, Porto Alegre, BR–RS, 2004. Advisor: Dante Augusto Couto Barone; Coadvisor: Terry Caelli.

1. Point pattern matching. 2. Weighted graph matching. 3. Probabilistic graphical models. 4. Hidden Markov random fields. 5. Pattern recognition. I. Barone, Dante Augusto Couto. II. Caelli, Terry. III. Title.

À Camila

ACKNOWLEDGMENTS

First of all, I thank my supervisor Dante Augusto Couto Barone and my co-supervisor Terry Caelli. Dante has provided me continuous encouragement, support and, specially, has made me realize that science can be better made if one has freedom to think. In all circumstances where I needed his approval for taking important decisions, he was supportive. This freedom with which I have undertaken my Ph.D. program is - I'm very aware of this now - one of the main elements that lead me to obtain the results presented in this Thesis. Terry was not only an outstanding supervisor during my stay at the University of Alberta (UofA), where most of the technical work documented in this Thesis was developed, but also a mentor, an example as a scientist, and a friend. While working with him, I always had the most genuine sense of freedom in pursuing research in my own way. Yet, he was always there to solve my technical questions, as well as to give me wise advices about which path to choose in the uncountable bifurcations that one faces during the course of a Ph.D. Thesis gestation. For all this I believe I was most fortunate to have Dante and Terry as my advisors, and here I acknowledge that.

Many Professors have somehow been important during the course of my academic career, either during the Ph.D. or before it. At the Universidade Federal do Rio Grande do Sul (UFRGS), during my undergraduate studies, Felipe Rizzato was the first person to give me an opportunity to be involved with science, and much of my scientific posture has been inherited from him. Already in the course of my graduate studies, several Professors have influenced me significantly at UFRGS: Sílvia Olabarriaga, Jacob Scharcanski, Vitor Haertel and Robin Thomas Clarke. Sílvia deserves a special mention because she was of fundamental importance for having established formal contact with the University of Alberta in order to help me to undertake my sandwich program with Terry Caelli. Jacob, Vitor and Clarke prepared me technically so that I could attack the Thesis problem with a solid background. During my stay in Canada, I have benefited from the interactions with several Professors, but one had a direct impact on the development of this Thesis: Dale Schuurmans. Dale's influence cannot be easily quantified. All the Graphical Model apparatus that I have used to solve the problem of this Thesis was acquired in a course where Dale was the instructor. He has helped me so much by giving several suggestions and by raising fundamental questions (that made me think a lot!) that I realized he should necessarily take part in my Thesis committee, for no one else was so aware of the technical intricacies involved in the work.

Important technical discussions with many people were helpful while I was maturing the ideas. Discussions with X. Chen, Robert Connelly, Jon Dattorro, Ted Lewis and with my brother Gregório Silva Caetano and his wife Maria Carolina Nizarala Caetano were helpful in order to obtain the theoretical results.

Some people have diligently revised the first version of this Thesis. My brother

Gregório and my brother-in-law Adriano Petry have both provided thorough and impressively detailed revisions by correcting many errors and making critical observations that have improved significantly the quality of the original version of the manuscript. My great friend and guru for technological inquires Marcelo Boeira de Barcelos introduced many important ideas and observations about the presentation of the work, and I believe it became more readable for a broader audience after that.

From all the colleagues with whom I have professionally interacted during the course of my graduate studies, I would like to mention four in particular: Luís Renato Erpen and Siovani Felipussi at UFRGS and Li Cheng and John Arnold at UofA. Renato was my best friend at UFRGS during the years where we coexisted there. He was very helpful in many technical senses, in particular in his ability to optimizing my codes. He was very helpful as well when it came to listen patiently to crazy ideas from his friends. The relaxing and funny conversations with Siovani will never be forgotten. They were important to improve my mood in many circumstances. At UofA, Li Cheng was the colleague with whom I have interacted the most. John Arnold shared the office with me, and he was my Mac guru.

I am deeply grateful to the Universidade Federal do Rio Grande do Sul (UFRGS), in which I was a student both during undergraduate and graduate programs. In particular I thank the Instituto de Informática and all its members with whom I have interacted during these years. I also thank sincerely the University of Alberta (UofA), and specially the Department of Computing Science and its members, where part of the Ph.D. program was undertaken. The dedication needed to accomplish full-time research was only possible due to generous grants that I have received from the Brazilian research agencies CNPq and CAPES. CNPq has funded me while in Brazil, whereas CAPES funded the sandwich program in Canada.

I owe a special acknowledgment to the members of my committee. They have pointed out several errors and have made important suggestions in order to improve the quality of the final Thesis document. Felipe, Jacob, Vitor and Dale: thanks.

To all the friends from Cardionuclear, my sincere and deep thanks. Friendship is something that takes time to be established, and we certainly made it. We have interacted during most part of my graduate program, and their confidence in myself will never be forgotten.

A very important person was Adriano Petry, my wife's brother. Adriano was the person who convinced me that I should get back to school and do a Ph.D.. I am sincerely grateful for this attitude. This Thesis would not exist if he were not so committed in his attempts to convince me to return to Academia. He has also carefully and competently revised many of my papers prior to submission, as well as the first version of this Thesis.

I would like to thank all my Brazilian friends from Edmonton. They have provided a social environment that allowed me to relax from my hard moments. They have also provided important emotional support, specially when one is away from the great family and important friends. I would like to mention specially the fundamental role that Stanley Oliveira played in all the stages of our lives in Canada. From the moment we arrived in Edmonton to the moment we left, he was helpful in practically all matters: from providing us a place to stay in the first days to making us laugh with his keen sense of humor. This not to mention many other ways in which he has definitely made our lives easier in Edmonton. To me, he is now as another brother.

My parents-in-law, Claudio and Elisabeth, as well as Elisabeth's mother, Ethel, have made a significant effort in order to provide the appropriate environment for the writing of the Thesis. A significant part of this document was typed in their residence, and they were always very patient, demonstrating willingness in every matter that could in any sense make my life easier.

A very special acknowledgment goes to three very important people. My father, Reginaldo, was my first Professor, from whom I found inspiration to follow the scientific career by observing his indications that nature is outside there waiting to be understood. My mother Ione is the person who is always teaching me that without a synergic equilibrium between intellectual and physical developments one cannot succeed. Both have, since my early days, provided an education full of love, wisdom, well-defined values and constant support in all senses. In this respect, this Thesis is a result of their work as well. I am deeply grateful for all that they are and for everything that they have done and still do. My brother Gregório was and is an example of human being, and many of my attitudes both in professional and personal life have in him an inspiration. He is a great counselor for any matter, and has a special gift when it comes to clarify in a simple picture apparently complex psychological interactions among people. Certainly great part of my way of thinking today is a result of my interactions with him.

Finally, the most present person in all this endeavor was my wife, Camila. Her love, patience, continuous encouragement and support were present during all stages. The emotional stability resultant from a healthy relationship has been the major element in tackling the most varied issues and difficulties which are typical of the stressful period which is that of a Ph.D. program. The successful completion of this Thesis was only possible due to her commitment, optimism and unconditional support in all senses. I owe her my most profound acknowledgments. *Obrigado, Cami.*

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

AGM	Attributed Graph Matching
ARG	Attributed Relational Graph
ARS	Attributed Relational Structure
CPU	Central Processing Unit
CSP	Constraint Satisfaction Problem
EDM	Euclidean Distance Matrix
GAGM	Generalized Attributed Graph Matching
GPS	Global Positioning System
HC	Hammersley-Clifford
JT	Junction Tree technique (the technique proposed in this work)
MAP	Maximum a Posteriori
MBOR	Model-Based Object Recognition
MRF	Markov Random Field
PCA	Principal Component Analysis
PPM	Point Pattern Matching
PRL	Probabilistic Relaxation Labeling technique
PSM	Point Set Matching
RAM	Random Access Memory
rv	Random Variable
SGM	Structural Graph Matching
std	Standard Deviation
WGM	Weighted Graph Matching

LIST OF SYMBOLS

G	graph
$\mathcal{V}$	vertex set of graph G
$\mathcal{E}$	edge set of graph G
G_d	domain graph
G_c	codomain graph
$\mathcal{V}_d$	vertex set of graph G_d
$\mathcal{E}_d$	edge set of graph G_d
$\mathcal{V}_c$	vertex set of graph G_c
$\mathcal{E}_c$	edge set of graph G_c
$y(\mathcal{V})$	set of attributes of vertex set $\mathcal{V}$
$y(\mathcal{E})$	set of weights/attributes of edge set $\mathcal{E}$
$y(\mathcal{V}_d)$	set of attributes of vertex set $\mathcal{V}_d$
$y(\mathcal{E}_d)$	set of weights/attributes of edge set $\mathcal{E}_d$
$y(\mathcal{V}_c)$	set of attributes of vertex set $\mathcal{V}_c$
$y(\mathcal{E}_c)$	set of weights/attributes of edge set $\mathcal{E}_c$
$\mathcal{F}$	factorization property
$\mathcal{G}$	global Markov property
$\mathcal{L}$	local Markov property
$\mathcal{P}$	pairwise Markov property
$\mathbb{G}(\cdot)$	Gaussian similarity function
$\mathbb{H}(\cdot)$	Hyperbolic tangent similarity function
$\mathbb{I}(\cdot)$	Increasing weighting similarity function
$P(\cdot)$	Similarity function
d	dimensionality of a maximal clique/dimensionality of Euclidean Space
C	a maximal clique of a graph
$\mathcal{C}$	set of all C’s

C_Ω	a Ω-clique of a graph
$\mathcal{C}_\Omega$	set of all C_Ω's
ψ_C	potential function of clique C
ϕ_S	potential function of separator S
V_C	modified potential function of clique C
$\perp\!\!\!\perp$	notation for conditional independence
$\lvert . \rvert$	absolute value of a number/cardinality of a set
$\lVert . \rVert$	Euclidean (L^2) norm
K	minimum possible value for a similarity function in the experiments
N_i	nodes belonging to clique node i
$\mathcal{N}_i$	set of neighbors of node i in a graph
$c(\cdot)$	compatibility function in PRL
K_n	the complete graph with n vertices
Z	partition function
T	cardinality of the domain vertex set $\mathcal{V}_d$
S	cardinality of the codomain vertex set $\mathcal{V}_c$
d_i	i^{th} vertex of the domain vertex set $\mathcal{V}_d$
c_k	k^{th} vertex of the codomain vertex set $\mathcal{V}_c$
$d_{i_1 i_2}$	edge connecting vertices d_{i_1} and d_{i_2}
$c_{k_1 k_2}$	edge connecting vertices c_{k_1} and c_{k_2}
X_i	random variable associated to vertex d_i in G_d
x_k	realization associated to vertex c_k in G_c
$y^d_{i_1}$	attribute of vertex d_{i_1}
$y^c_{k_1}$	attribute of vertex c_{k_1}
$y^d_{i_1 i_2}$	weight/attribute of edge $d_{i_1 i_2}$
$y^c_{k_1 k_2}$	weight/attribute of edge $c_{k_1 k_2}$
y^d	whole set of weights/attributes in graph G_d
y^c	whole set of weights/attributes in graph G_c
G_m	model graph
X_A	set of random variables indexed by set A
X	entire set of random variables (the complete random field)
x	a particular realization of the entire random field X
x_C	realization of random variables in clique C
$\mathcal{X}_A$	sample space of random field X_A

$\mathcal{X}$	sample space of random field X
$p(x)$	probability distribution on $\mathcal{X}$
$U(x)$	energy function on $\mathcal{X}$
$C(\cdot)$	compatibility function
$C_\Omega(\cdot)$	compatibility function between attributes of a Ω-clique
$D(\cdot)$	some distance function
$D_\Omega(\cdot)$	distance function between attributes of a Ω-clique
$\mathcal{J}$	subset of the cartesian product $\mathcal{V}_d \times \mathcal{V}_c$
$S(\cdot)$	similarity measure
M	matching matrix
$\mathcal{K}_{ij}$	set of clique nodes between clique nodes i and j
$\mathbf{p}$	a configuration in $\mathbb{R}^d$ (a set of labeled points)/a realization of an *EDM*
$\mathbf{p}_i$	i^{th} realization of an *EDM*
$G(\mathbf{p})$	a framework defined by graph G and configuration $\mathbf{p}$
p_i	i^{th} point in a configuration $\mathbf{p}$
S_i	i^{th} sphere in $\mathbb{R}^d$
Q	a vector subspace
I_i	i^{th} sphere lying in Q
σ	parameter that controls the width of the similarity functions $\mathbb{G}$, $\mathbb{H}$ and $\mathbb{I}$
$\square$	symbol for end of proof
$\mathbb{J}$	indicator function
$\mathcal{B}$	set of clique nodes in a clique tree

LIST OF FIGURES

ABSTRACT

Point pattern matching in Euclidean Spaces is one of the fundamental problems in Pattern Recognition, having applications ranging from Computer Vision to Computational Chemistry. Whenever two complex patterns are encoded by two sets of points identifying their key features, their comparison can be seen as a point pattern matching problem. This work proposes a single approach to both exact and inexact point set matching in Euclidean Spaces of arbitrary dimension. In the case of exact matching, it is assured to find an optimal solution. For inexact matching (when noise is involved), experimental results confirm the validity of the approach. We start by regarding point pattern matching as a weighted graph matching problem. We then formulate the weighted graph matching problem as one of Bayesian inference in a probabilistic graphical model. By exploiting the existence of fundamental constraints in patterns embedded in Euclidean Spaces, we prove that for exact point set matching a simple graphical model is equivalent to the full model. It is possible to show that exact probabilistic inference in this simple model has polynomial time complexity with respect to the number of elements in the patterns to be matched. This gives rise to a technique that for exact matching provably finds a global optimum in polynomial time for any dimensionality of the underlying Euclidean Space. Computational experiments comparing this technique with well-known probabilistic relaxation labeling show significant performance improvement for inexact matching. The proposed approach is significantly more robust under augmentation of the sizes of the involved patterns. In the absence of noise, the results are always perfect.

Keywords: Point pattern matching, weighted graph matching, probabilistic graphical models, hidden Markov random fields, pattern recognition.

Modelos Gráficos e Casamento de Padrões de Pontos

RESUMO

Casamento de padrões de pontos em Espaços Euclidianos é um dos problemas fundamentais em reconhecimento de padrões, tendo aplicações que vão desde Visão Computacional até Química Computacional. Sempre que dois padrões complexos estão codificados em termos de dois conjuntos de pontos que identificam suas características fundamentais, sua comparação pode ser vista como um problema de casamento de padrões de pontos. Este trabalho propõe uma abordagem unificada para os problemas de casamento exato e inexato de padrões de pontos em Espaços Euclidianos de dimensão arbitrária. No caso de casamento exato, é garantida a obtenção de uma solução ótima. Para casamento inexato (quando ruído está presente), resultados experimentais confirmam a validade da abordagem. Inicialmente, considera-se o problema de casamento de padrões de pontos como um problema de casamento de grafos ponderados. O problema de casamento de grafos ponderados é então formulado como um problema de inferência Bayesiana em um modelo gráfico probabilístico. Ao explorar certos vínculos fundamentais existentes em padrões de pontos imersos em Espaços Euclidianos, provamos que, para o casamento exato de padrões de pontos, um modelo gráfico simples é equivalente ao modelo completo. É possível mostrar que inferência probabilística exata neste modelo simples tem complexidade polinomial para qualquer dimensionalidade do Espaço Euclidiano em consideração. Experimentos computacionais comparando esta técnica com a bem conhecida baseada em relaxamento probabilístico evidenciam uma melhora significativa de desempenho para casamento inexato de padrões de pontos. A abordagem proposta é significativamente mais robusta diante do aumento do tamanho dos padrões envolvidos. Na ausência de ruído, os resultados são sempre perfeitos.

Palavras-chave: casamento de padrões de pontos, casamento de grafos ponderados, modelos gráficos probabilísticos, campos aleatórios de Markov, reconhecimento de padrões.

CONTRIBUIÇÕES

Este trabalho apresenta algumas contribuições relativas ao problema de casamento de padrões de pontos em Espaços Euclidianos de dimensão qualquer.

A primeira contribuição é formular o problema de casamento de padrões de pontos como um problema de casamento de grafos ponderados onde nodos e pesos dos grafos correspondem, respectivamente, a pontos e a distâncias entre pontos em um Espaço Euclidiano.

A segunda contribuição consiste na reformulação do problema de casamento de grafos ponderados como um problema de inferência probabilística Bayesiana em um modelo gráfico probabilístico. Neste modelo, as variáveis aleatórias correspondem a pontos em um padrão de pontos e as realizações correspondem a pontos no outro padrão de pontos. Neste problema de inferência, uma solução máxima a posteriori (MAP) é procurada.

A terceira contribuição consiste na demonstração de que um simples modelo gráfico esparso é equivalente ao modelo completo no caso de casamento exato. Ocorre que inferência probabilística exata neste modelo simples é factível em tempo polinomial, o que nos permite obter uma técnica eficiente tanto para casamento exato quanto inexato de padrões de pontos em Espaços Euclidianos de dimensão arbitrária, que além disso ainda é ótima para o caso de casamento exato.

A quarta contribuição consiste na extensão da formulação através da demonstração de que qualquer tipo de problema de casamento de padrões pode ser visto como um problema de inferência probabilística em um modelo gráfico.

A quinta e última contribuição consiste na implementação em software do algoritmo resultante e sua comparação com uma técnica padrão na literatura através de uma série de numerosos experimentos controlados. Os experimentos evidenciam uma melhora significativa de desempenho, o qual tende a ser progressivamente melhor à medida em que a quantidade de pontos envolvidos aumenta.

1 INTRODUCTION

A fundamental problem in Pattern Recognition is that of *matching* two sets of points that represent relevant features of associated objects or entities. Matching simply means finding a mapping, or a correspondence, from one set of points to another set of points. By representing objects with point sets depicting their fundamental structure, an abstract representation is obtained through which the comparison between a pair of objects can be made in practice via point set matching.

This issue arises in a large variety of fields. In Computer Vision, researchers are frequently interested in model-based object recognition, for example. In this problem domain, a visual object is frequently modeled as a set of spatially distributed points that summarizes its structural content (LEUNG; BURL; PERONA, 1995; BESL; JAIN, 1985). As a result, the problem of finding an object in a visual scene (which is a set of visual objects) turns out to be a point pattern matching problem. Also in Computer Vision one finds the stereo matching problem, which consists in finding the correspondence between points of a pair of slightly different 2D images acquired by a binocular camera setting (REIMANN; HAKEN, 1994). In Image Processing, there is the problem of Image Registration, which consists of aligning two images which are supposed to correspond to the same scene but were acquired under different circumstances (TON; JAIN, 1989). In order to find a proper alignment, it is necessary to know which points in one image correspond to which points in the other. If the mapping is known, a transformation can be obtained, which brings the misaligned image to match the reference one. This issue also appears in fields like Computational Chemistry and Biology, where problems like that of detecting pharmacophores, crucial in drug design, can be regarded as a point set matching task, as well as molecular database screening and comparative molecular field analysis (MARTIN et al., 1993; AKUTSU et al., 2003). In Astronomy, constellation pattern search can be also modeled as a point pattern matching problem (MURTAGH, 1992). Many other application domains involve some sort of matching problems, including Data Mining, Biometrics and Information Retrieval, among others (CONTE et al., 2004).

Although the theoretical results presented in this work are clearly independent on the type of application considered, the main focus lies on the Computer Vision and Image Analysis domain, in which the computational experiments were performed. In Computer Vision, the point pattern matching problem has been almost entirely approached as a *graph matching* problem since the 1970's when the pioneering studies in this area were published (CONTE et al., 2004). In graph matching, point sets are abstracted as being graphs where the points correspond to nodes and some edge adjacency structure is used to model the spatial configuration of the distribution of points. Purely structural relations (i.e. presence or absence of edges) may be used, that results in the *structural* graph matching problem. Alternatively, *attributed* relations may be used, where the edges and maybe nodes are associated with attribute vectors. In this last case, it is said that we have an *attributed* graph matching problem, which has as an important particular case the *weighted* graph matching problem. Once the patterns have been encoded as graphs, point set matching becomes a graph matching problem, that consists in finding the mapping from the set of nodes in one graph to the set of nodes in the other graph such that some global similarity measure (which will depend on the adjacency structure or attribute set of both graphs) is maximized.

Over the past 30 years, several contributions have been made aiming at developing efficient algorithms for graph matching in visual pattern recognition. The initial enthusiasm of the late 1970's and early 1980's, however, was shifted into a period of minor interest from mid 1980's to early 1990's. Recently, it has been observed a significant growth in the attention devoted by the scientific community to this problem, probably due to the fact that the computational cost of graph matching algorithms is now becoming compatible with modern hardware facilities (CONTE et al., 2004). Among the existent approaches for tackling the problem, it is possible to identify Tree Search algorithms, Spectral methods, Continuous Optimization techniques and many others (CONTE et al., 2004). A common feature of all approaches for graph matching in Computer Vision is that they do not assure global optimality in polynomial time (CONTE et al., 2004). For arbitrary graphs, the worst case scenario for finding the optimal match has always exponential time complexity, what has encouraged researchers to develop efficient approximations that run in polynomial time.

The main contribution of this work is to show how it is possible to obtain remarkably good solutions (optimal solutions in the case of exact matching) to the graph matching problem in Euclidean Spaces in polynomial time while handling invariance to translations, rotations and reflections of the involved patterns. This is not a polynomial time optimal solution to the general graph matching problem, which is known to be NP-complete, but to the particular case when the graphs are embedded in Euclidean spaces of arbitrary dimension, which includes Computer Vision and many other domains. Computational experiments comparing the proposed technique with the well-known and widely used probabilistic relaxation labeling approach confirm a significant improvement in performance. In particular, the suggested method exhibits robustness under augmentation of the problem size which is unparalleled in the alternative technique.

We also make a contribution when we show that the proposed principle of modeling matching via inference in graphical models is naturally extendable to *any* form of matching. This is done by showing that the formulation naturally gives rise to the most general form of attributed graph matching, which itself is the most general form of matching.

This Thesis is organized as follows. Chapter 2 describes the graph matching problem and its several instances, as well as the specific instance considered in the present work. Chapter 3 presents an overview of several of the important contributions in the related literature spread over the last 30 years. Chapter 4 provides the necessary background for the subsequent chapters: the theory of probabilistic graphical models. In Chapters 5 and 6, we present the original contributions of this work, which include the theoretical foundations of a new point set matching procedure as well as a set of computational experiments aiming at evaluating its performance. Chapters 7 and 8 present final discussion and conclusions.

2 THE PROBLEM

Point Pattern Matching in Computer Vision has been eminently modeled as a graph matching problem, and in this work we also take this perspective. The expression “graph matching problem” actually involves not just a single problem, but a series of similar but different problems that have been studied in the Pattern Recognition research community over the past 30 years. In essence, these problems are important because graphs are powerful mathematical representations that can model real patterns, and the pattern matching task is of course the core of Pattern Recognition.

In this chapter, the point pattern matching problem is described, both in its exact and inexact versions. It is then explained that this problem is almost always formulated as a graph matching task, and the attention is then turned to graph matching per se. The different “classes” of graph matching problems are described in further detail. In particular, we introduce our first contribution by formulating point pattern matching as a graph matching problem (a weighted graph matching problem) based on the concept of Euclidean Distance Matrices.

2.1 The Point Pattern Matching Problem

Point Pattern Matching (PPM), or, equivalently, Point Set Matching (PSM), is one of the most fundamental problems in Structural Pattern Recognition (CONTE et al., 2004). It arises in several fields like 2D and 3D Image Analysis, Document Processing, Biometric Identification, Image Databases, Video Analysis and Biological and Biomedical applications (CONTE et al., 2004). In a loose description, it consists in assigning each point in one point set to a point in another point set, such that some constraint or set of constraints is enforced, and some global similarity measure is optimized. The types of constraints and similarity measure determine the type of the matching problem. Figure 2.1 shows a pictorial description of a point set matching task where the matched patterns are rotated with respect to each other.

PPM can be either *exact* or *inexact*. In exact matching, the “query” point set is identical, up to some linear transformation such as an isometry, for example, to the “database” point set, or to some subset of it. In inexact matching, some type of stochastic noise may alter the intrinsic structure of the pattern, such that its instance contained in the database may be significantly deformed. In most applications, particularly in Computer Vision, virtually any matching problem falls into this second category, due to the unavoidable interference of imperfect acquisition systems and non-stationary acquisition conditions.

The fact that inexact matching is the rule, not the exception, means that matching algorithms must be able to cope with some degree of uncertainty, as well as be able to show some robustness with respect to the introduction of noisy artifacts and structural

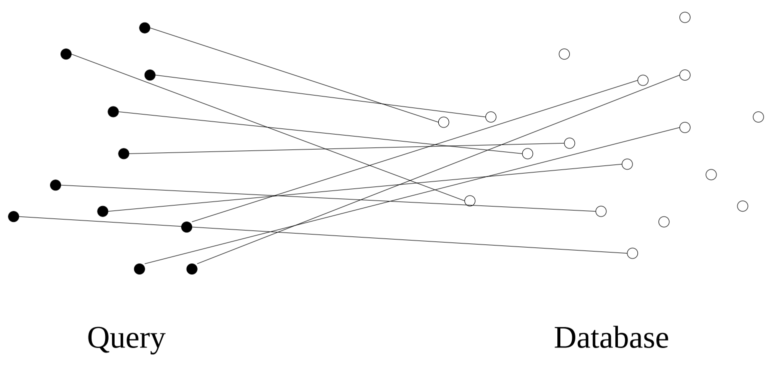


Figure 2.1: A matching between two point sets.

corruption of the involved patterns. This necessity for a representation somewhat robust to error deviations, among other things, has encouraged researchers - at least in Computer Vision and related domains - to model point pattern matching as a graph matching problem (HANCOCK; WILSON, 2002). Graphs are entities that preserve their topological properties even under a large class of node position deformations, and this fact has caught the attention of algorithm designers over the years, who have exploited these invariance properties to develop efficient invariant matching algorithms (LUO; HANCOCK, 2001). Alternatively, graphs can be generalized by including node and edge *attributes*, and invariant matching can be accomplished by properly choosing invariant features as attributes (LI, 1992). In summary, inexact point pattern matching (which can include the exact case as a particular instance when the noise is zero) can be effectively modeled as a graph matching problem. In the following section a classification of graph matching problems is introduced.

2.2 Graph Matching Problems

Although it is possible to differentiate a few types of graph matching problems, it is also true that they have a common feature: a *correspondence* between vertices of two graphs must be found. Assume that a *domain* graph with vertex set $\mathcal{V}_d$ and a *codomain* graph with vertex set $\mathcal{V}_c$ are given. A correspondence simply means that a particular subset $\mathcal{J}$ of the cartesian product between the two vertex sets $\mathcal{V}_d$ and $\mathcal{V}_c$ must be selected:

$$\mathcal{J} \subset \mathcal{V}_d \times \mathcal{V}_c. \tag{2.1}$$

However, it is clear that this definition is too general to be useful. What really specifies the type of graph matching problem are the requirements, constraints or specificities involved in the selection of this subset $\mathcal{J}$. And this has to do with the characteristics of the particular graphs being considered. In this section it is presented a classification of graph matching problems which aims to cover all types of studied problems.

2.2.1 Structural graph matching

The type of graph matching problem is directly dependent on the *types of graphs* being considered. If the graphs are defined as usual, being a pair $G = (\mathcal{V}, \mathcal{E})$ composed of the vertex set $\mathcal{V}$ and the edge set $\mathcal{E}$, the resulting problem is called a *Structural Graph Matching* (SGM) problem.

In the SGM problem, the only information available in order to determine the "correct" correspondence is the *adjacency matrix* of the involved graphs. For a given vertex ordering, the adjacency matrix has an element that equals 0 when there is no edge between the vertices associated to the indexes of the corresponding row and column, and that equals 1 otherwise. The "correct" correspondence in this case may have several different meanings, depending on the type of constraints enforced in the mapping. For example, there is the *graph isomorphism* problem, which consists in finding a mapping $\mathcal{J}$ such that the adjacency matrices of both graphs (when the vertices of the second are indexed isomorphically with respect to those of the first, according to $\mathcal{J}$) are equal. From this definition it is possible to conclude that graph isomorphism applies only to graphs with the same number of vertices. Another example is *subgraph isomorphism*, which simply consists in finding a graph isomorphism between a subset of the first graph and the second graph, where "first graph" means the graph with greater (or equal) amount of vertices.

In the strict versions of these two problems, the interest lies in *exact* matching: a perfect correspondence (one which preserves adjacency) is sought. In many real applications, however, this is not the case. Consider, for example, Computer Vision where the relevant point features in two objects to be matched are invariably extracted under error measurements or noisy conditions. Points extracted in one object may be missed in the other, and strict graph or subgraph isomorphism is no longer useful. Due to these limitations presented by exact graph matching methods, researchers have also created what is called *inexact* graph matching, or *error-tolerant* graph matching. In these techniques, models for structural errors are postulated, and the "optimal" match is the one that minimizes the global error measure. SGM problems can also be of this kind (HANCOCK; WILSON, 2002; LUO; HANCOCK, 2001), in which case one is interested in finding a correspondence that in some sense locally or globally maximizes some compatibility measure between the graphs. The types of constraints enforced in the correspondence mapping are in this case a key aspect of the designed algorithm. For example, one may be interested only in injective mappings (each element of the domain graph maps to a unique element of the codomain graph, and two elements never map to the same element), or in total mappings (each element of the domain graph maps to a unique element of the codomain graph, but different elements may map to the same element). These concepts will be addressed in the other types of graph matching problems discussed in the following.

2.2.2 Weighted graph matching

Weighted graph matching (WGM) is the problem of matching *weighted graphs*. A weighted graph is a triple $G = (\mathcal{V}, \mathcal{E}, y(\mathcal{E}))$ where $\mathcal{V}$ and $\mathcal{E}$ are defined as before and $y(\mathcal{E})$ is a set of real values, each one associated to an element of $\mathcal{E}$ (an edge). These real values are called weights. Consider a domain graph G_d and a codomain graph G_c. The i^{th} vertex in G_d is denoted by d_i, while the k^{th} vertex in G_c is denoted by c_k. The particular weight associated to the edge that connects vertices d_{i_1} and d_{i_2} in G_d is denoted as $y^d_{i_1 i_2}$, and similarly for the weight of the edge that connects c_{k_1} and c_{k_2} in G_c, which is denoted as $y^c_{k_1 k_2}$. The complete sets of weights in G_d and G_c are simply denoted by y^d and y^c, respectively. It is possible to see the SGM problem as a particular case of the WGM problem, where the weights only take values in $\{0, 1\}$.

Given two weighted graphs $G_d = (\mathcal{V}_d, \mathcal{E}_d, y(\mathcal{E}_d))$ and $G_c = (\mathcal{V}_c, \mathcal{E}_c, y(\mathcal{E}_c))$, the WGM problem can be posed as one of maximizing with respect to M the following function (GOLD; RANGARAJAN, 1996):

$$P(M) = \sum_{i_1=1}^{T}\sum_{i_2=1}^{T}\sum_{k_1=1}^{S}\sum_{k_2=1}^{S} M_{i_1k_1}M_{i_2k_2}C(y^d_{i_1i_2}, y^c_{k_1k_2}) \tag{2.2}$$

where T and S are the cardinalities of $\mathcal{V}_d$ and $\mathcal{V}_c$, respectively. $C(\cdot,\cdot)$ is a compatibility measure between the weights of the two edges associated with the corresponding vertex indices. M is a *matching matrix* ($M_{i_jk_j}$), which plays the role of an indicator function such that

$$M_{i_jk_j} = \begin{cases} 0, & \text{if } d_{i_j} \text{ does not map to } c_{k_j}, j = 1,2 \\ 1, & \text{if } d_{i_j} \text{ does map to } c_{k_j}, j = 1,2 \end{cases} \tag{2.3}$$

where different constraints can be enforced in M, such as the requirement that the map must be *injective* (GOLD; RANGARAJAN, 1996)

$$\begin{aligned} \sum_{k_j=1}^{S} M_{i_jk_j} &= 1, \forall i_j; j = 1,2 \\ \sum_{i_j=1}^{S} M_{i_jk_j} &= 1, \forall k_j; j = 1,2 \end{aligned} \tag{2.4}$$

or more generally a total function

$$\sum_{k_j=1}^{S} M_{i_jk_j} = 1, \forall i_j; j = 1,2. \tag{2.5}$$

Equivalently, the WGM problem can be also formulated as one of minimizing an "energy" function $U(M)$, as described in (GOLD; RANGARAJAN, 1996):

$$U(M) = \sum_{i_1=1}^{T}\sum_{i_2=1}^{T}\sum_{k_1=1}^{S}\sum_{k_2=1}^{S} M_{i_1k_1}M_{i_2k_2}D(y^d_{i_1i_2}, y^c_{k_1k_2}) \tag{2.6}$$

where $D(\cdot)$ in Eq. (2.6), as opposed to $C(\cdot)$ in Eq. (2.2), is a *dissimilarity* measure (instead of a similarity or compatibility measure).

In words, this optimization problem involves finding the optimal mapping M, which is the one that maximizes the sum over all compatibility measures over pairwise mappings. This problem is known to be NP-complete for many definitions of the compatibility measure $C(\cdot,\cdot)$ (GOLD; RANGARAJAN, 1996).

It is possible to infer from its definition that this problem is a model suited for describing the matching of patterns whose parts are related not in a discrete, but in a more flexible way such as a gradual representation over a certain range of values. This may be the case, as we will show in this work, when mutual distances of point sets need to be encoded in a graph.

2.2.3 Attributed graph matching

A natural generalization of a weighted graph is an *attributed* graph, which is a 4-tuple $G = (\mathcal{V}, \mathcal{E}, y(\mathcal{V}), y(\mathcal{E}))$, where $\mathcal{V}$ and $\mathcal{E}$ are defined as before, $y(\mathcal{V})$ is a set of vector attributes associated to the vertices of G and $y(\mathcal{E})$ is a set of edge vector attributes associated to the edges of G. It generalizes the weighted graph in two senses: first, the edge weights (scalars) are generalized to edge *vector* attributes, also called *binary attributes*; second, the vertices are also endowed with vector attributes, also called *unary attributes*. A particular attribute vector associated to vertex d_{i_1} is denoted as $y^d_{i_1}$, and a particular attribute vector associated to an edge $d_{i_1 i_2}$ as $y^d_{i_1 i_2}$. Analogously, a vector associated to vertex c_{k_1} is denoted as $y^c_{k_1}$, and a vector associated to an edge $c_{k_1 k_2}$ as $y^c_{k_1 k_2}$.

Given a domain attributed graph $G_d = (\mathcal{V}_d, \mathcal{E}_d, y(\mathcal{V}_d), y(\mathcal{E}_d))$ and a codomain attributed graph $G_c = (\mathcal{V}_c, \mathcal{E}_c, y(\mathcal{V}_c), y(\mathcal{E}_c))$, the *attributed graph matching* (AGM) problem can be directly generalized as one of maximizing, with respect to M, the following function:

$$P(M) = \sum_{i_1=1}^{T} \sum_{k_1=1}^{S} M_{i_1 k_1} C_1(y^d_{i_1}, y^c_{k_1}) + \\ + \sum_{i_1=1}^{T} \sum_{i_2=1}^{T} \sum_{k_1=1}^{S} \sum_{k_2=1}^{S} M_{i_1 k_1} M_{i_2 k_2} C_2(y^d_{i_1 i_2}, y^c_{k_1 k_2}) \tag{2.7}$$

under some of the constraints in the spirit of 2.4 or 2.5. $C_1(\cdot,\cdot)$ and $C_2(\cdot,\cdot)$ denote respectively the "unary" and "binary" compatibility measure functions. It is evident that this problem is a generalization of the WGM problem, which can be obtained by choosing C_1 to be null and restricting binary attributes to be 1-dimensional.

Several relevant problems in Computer Vision can be modeled as AGM problems (CHRISTMAS; KITTLER; PETROU, 1994; LI, 1992).

2.3 Point Pattern Matching as a Weighted Graph Matching Problem

As said above, the problem studied in this work - exact and inexact point pattern matching - can be formulated as a graph matching problem. Here we present the first contribution of this work by showing how to formulate this problem specifically as a *weighted* graph matching problem.

2.3.1 Euclidean distance matrices

A helpful concept to introduce our way of formulating the PPM problem is that of an *Euclidean Distance Matrix* (*EDM*) (DATTORRO, 2004; HUANG; LIANG; PARDALOS, 2003). A matrix $A = (a_{ij}) \in \mathbb{R}^{m \times m}$ is an *EDM* if there exist vectors $p_1, \ldots, p_m \in \mathbb{R}^n$ (for some $n \geq 1$) such that $\|p_i - p_j\| = a_{ij}$ for all $i, j \in N = \{1, \ldots, m\}$, where $\|.\|$ denotes the Euclidean norm in $\mathbb{R}^n$. Note that the ordering of the vectors is essential: different orderings will result in general in different *EDM*s, even if all the relative distances are the same. Below are two different *EDM*s that have the same set of elements.

$$EDM_1 = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & \sqrt{2} \\ 1 & \sqrt{2} & 0 \end{pmatrix} \qquad EDM_2 = \begin{pmatrix} 0 & 1 & \sqrt{2} \\ 1 & 0 & 1 \\ \sqrt{2} & 1 & 0 \end{pmatrix} \tag{2.8}$$

Note that these different *EDM*s can represent a different ordering in the indexing of the underlying vectors: the order of the first and second vectors have been switched.

The set of vectors $\mathbf{p} = \{p_i, i \in N\}$ is called a *realization* of A (which has nothing to do with a realization of a random variable). Obviously, there are infinite realizations for the same *EDM*. Two possible realizations in $\mathbb{R}^2$ for the above EDM_1 are

$$p_1 = (0,0),\ p_2 = (0,1),\ p_3 = (1,0) \qquad \text{and} \qquad p_1 = (1,0),\ p_2 = (1,1),\ p_3 = (2,0)$$

and two possible realizations for EDM_2 are

$$p_1 = (0,0),\ p_2 = (1,0),\ p_3 = (0,1) \qquad \text{and} \qquad p_1 = (1,0),\ p_2 = (1,1),\ p_3 = (0,2)$$

There is a formal result which confirms the intuitive notion that any linear combination of translations, rotations and reflections (an *isometry* in general) of a realization $\mathbf{p}_1$ of an *EDM* gives another realization $\mathbf{p}_2$ of the *same EDM*, such that the *conformation* (i.e., the rigid spatial structure related to points in a realization) of the realizations remain the same (HUANG; LIANG; PARDALOS, 2003). This is the starting point for our formulation of the PPM problem: an *EDM uniquely* determines the conformation of the corresponding point set. Since an *EDM* uniquely determines the conformation, the equality of two *EDM*s implies the equality of their conformations, what justifies the comparison of conformations in terms of the comparison of their respective *EDM*s. This is precisely the idea of our formulation: measuring the similarity of two conformations (two point patterns) by the similarity of their respective *EDM*s. In the following we present the description of the problem formulation.

2.3.2 Problem formulation by comparing *EDM*s

Assume the existence of two sets of points (vectors) in $\mathbb{R}^n$, $\mathbf{p}_1$ and $\mathbf{p}_2$. Assume for convenience that the cardinality of $\mathbf{p}_2$ is greater than or equal to that of $\mathbf{p}_1$: $|\mathbf{p}_2| \geq |\mathbf{p}_1|$. The interest here is to find a subset of $\mathbf{p}_2$ that has the same *EDM* than $\mathbf{p}_1$. The equality of two *EDM*s can be tested by simply making an element-by-element equality comparison. However, in the present work we are interested in handling noisy situations, where the conformation sought is possibly deformed. This encourages us to model a *similarity* measure between two *EDM*s as follows:

$$S(EDM_1, EDM_2) = \sum_{i=1}^{|\mathbf{p}_1|} \sum_{j=1}^{|\mathbf{p}_1|} C(EDM_1(i,j), EDM_2(i,j)) \tag{2.9}$$

where $C(\cdot,\cdot)$ is some *compatibility* or proximity measure between the corresponding elements (distances) of the two matrices.

In the following we show that this problem can be recast into a WGM problem.

2.3.3 Problem formulation by comparing graph embeddings in $\mathbb{R}^n$

It is useful in the context of this work to make an analogy between realizations of an *EDM* and *graph embeddings*. Graphs are abstract entities that, in principle, have no relation with the concept of vector spaces in linear algebra. It is possible to define a new object, however, if it is considered a graph whose vertices represent points in some vector space, such as $\mathbb{R}^n$, and whose edges represent curves in this vector space. It is said that this relevant object is a *graph embedding* in $\mathbb{R}^n$ (WEST, 2001). Figure 2.2 shows two examples of different embeddings of the same graph in the plane of this page (in $\mathbb{R}^2$, the

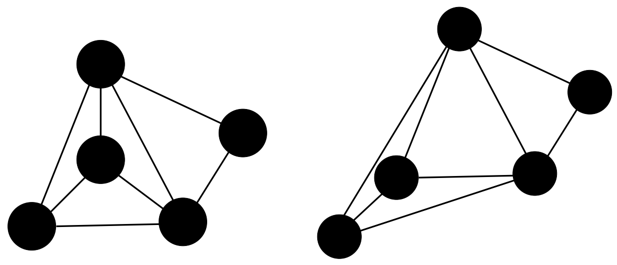


Figure 2.2: Different straight line embeddings of the same graph.

concept of embedding is analogous to the concept of “drawing”). These embeddings are called “straight line embeddings”, in the sense that the curves that join neighbor nodes are straight line segments.

In particular, if the edges are restricted to be straight line segments, their length will coincide with the Euclidean distance between the corresponding points (nodes in the graph embedding). This leads to a complete analogy between an *EDM* and a weighted graph: if the edge weights are precisely the Euclidean distances between the corresponding points, an *EDM* can be seen as a table which indicates the correspondent weight between each pair of vertices in the graph. A particular realization of an *EDM* will then correspond to a particular embedding of the graph (always assuming straight line edges). Under this perspective, we can remap the problem of comparing two *EDM*s to that of comparing two weighted graphs where the weights correspond to the Euclidean distances between the corresponding vertices.

The final function to be maximized will be

$$P(M) = \sum_{i_1=1}^{|\mathbf{p}_1|} \sum_{i_2=1}^{|\mathbf{p}_1|} \sum_{k_1=1}^{|\mathbf{p}_2|} \sum_{k_2=1}^{|\mathbf{p}_2|} M_{i_1 k_1} M_{i_2 k_2} C(y^d_{i_1 i_2}, y^c_{k_1 k_2}), \tag{2.10}$$

where we introduce explicitly the notation in terms of the WGM problem: $C(\cdot,\cdot)$ measures the similarity of the two distances $y^d_{i_1 i_2}$ and $y^c_{k_1 k_2}$, while $M_{i_j k_j}$ indicates if d_{i_j} maps to c_{k_j} ($M_{i_j k_j} = 1$) or not ($M_{i_j k_j} = 0$), for $j = 1, 2$. Note that the elements of EDM_1 and EDM_2 are denoted now as $y^d_{i_1 i_2}$ and $y^c_{k_1 k_2}$, the edge weights. Also, the comparison between two *EDM*s as shown in Eq. (2.9) now is explicitly done for all possible EDM_2s, what is represented in the sums over the indices of the elements in $\mathbf{p}_2$.

In summary, for a given global mapping M (analogously, for given EDM_1 and EDM_2), the above expression computes the sum of the pairwise compatibility measures over all pairwise mappings. By spanning over the whole range of possible mappings M (by spanning over all possible EDM_2s), it is possible to find which map M (which combination of EDM_1 and EDM_2) maximizes $P(M)$.

In this work, we choose to enforce only the constraint that the mapping must be total:

$$\sum_{k_j=1}^{|\mathbf{p}_2|} M_{i_j k_j} = 1, \forall i_j, j = 1, 2 \tag{2.11}$$

such that every element in $\mathbf{p}_1$ must map to one and only one element in $\mathbf{p}_2$, but there is no enforcement that different elements in $\mathbf{p}_1$ must map to different elements in $\mathbf{p}_2$, which is the case in injective mappings (GOLD; RANGARAJAN, 1996). There are two reasons for this choice: first, as we will see, the technique we are going to propose to solve the problem is naturally suited for the case of total mappings; second, and most relevant, is the fact that in many practical applications injective mappings are too restrictive to capture the variety of possible settings that may arise. For example, in Computer Vision it is often the case when two points closely apart in one image are superimposed in the second image due to noise influences: in this situation an injective-enforcing algorithm will not allow these two points to be mapped to the same point, although this is the correct interpretation.

It is clear that a naive solution to the problem is to compute the expression 2.10 for each one of the possible EDM_2s and choose the one for which the computed value for P is maximal. In detail, this expression must be calculated for all the $|\mathbf{p}_1|!$ orderings of each of the $|\mathbf{p}_2|!/(|\mathbf{p}_1|!(|\mathbf{p}_2| - |\mathbf{p}_1|)!)$ unordered $|\mathbf{p}_1|$-sized subsets of $\mathbf{p}_2$. Although this may look like an NP-hard problem, in this work we will show how to globally optimize the function 2.10 in polynomial time for the exact matching case.

2.4 Application Domains

We mentioned in the introduction that there is a large set of application domains in which the point pattern matching problem arises. In this section, a brief description of some relevant applications related to Computer Vision, which is the main focus of this work, will be presented.

2.4.1 Model-based object recognition

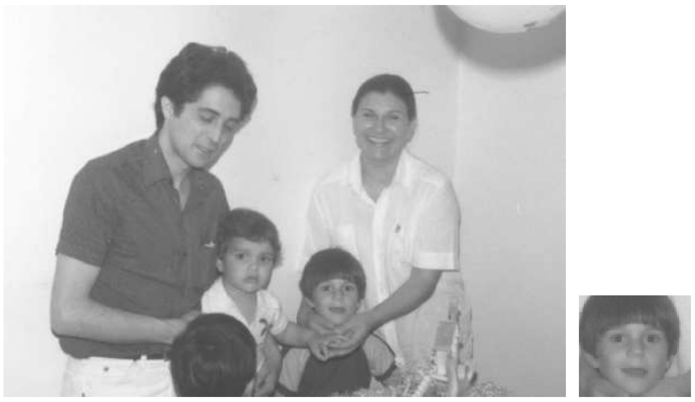
In Computer Vision, researchers are frequently interested in designing algorithms to recognize objects in visual scenes (LEUNG; BURL; PERONA, 1995). In order to accomplish that, they may use model-based object recognition (MBOR). MBOR consists in representing both the scene and a particular object that is expected to be found in the scene by some mathematical model. The problem of finding the object in the scene then reduces to a problem of comparison or matching between the scene and object models. This problem is probably one of the most important in Computer Vision (CONTE et al., 2004).

Point set matching is a standard way of performing MBOR. Figure 2.3 shows a scene and a particular instance that one is interested in finding in the scene. Point feature detectors are applied to both images in order to select representative landmarks and finally a point set matching algorithm is applied in order to find which point in the scene correspond to each point in the template. In this way the face of the boy is expected to be found.

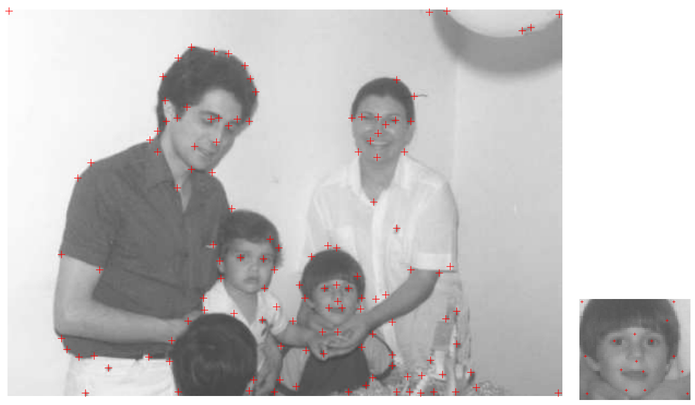
2.4.2 Stereo matching

Another important application for point set matching is in finding the correspondence between a pair of images acquired by a stereo camera apparatus (BOYER; KAK, 1988). Figure 2.4 shows the first pair of images acquired from Mars exploration rover Spirit in early 2004.

In this particular application, the robot is expected to exhibit stereoscopic vision, what is only possible through a continuous matching of both images via some point set matching algorithm. From the disparity measured in the matching process, the robot can infer the depth of each point in the image, what is the main characteristic of stereoscopic vision.



(a)



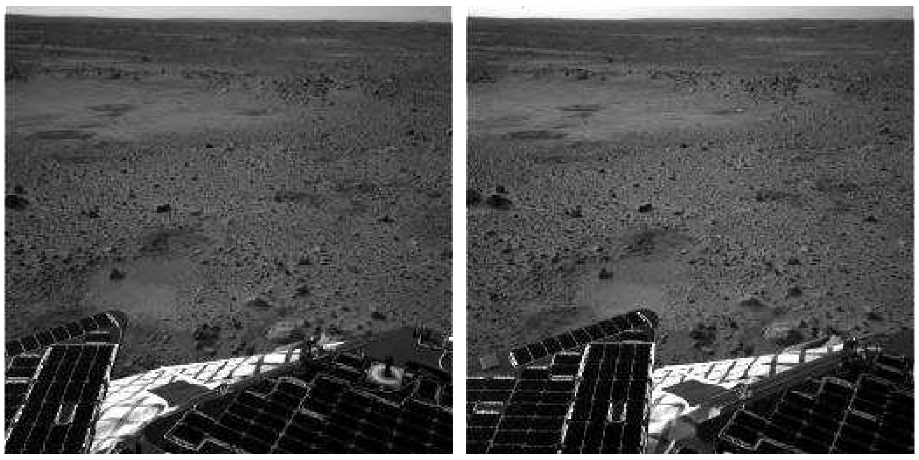
(b)

Figure 2.3: Point set matching applied to model-based object recognition: (a) a scene and a template; (b) point features detected in both images in order to apply point set matching to find where is the template in the scene.

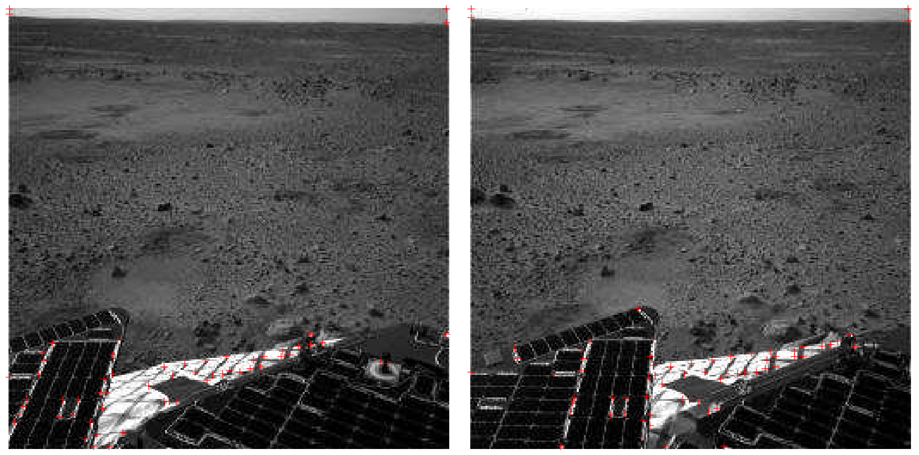
2.4.3 Image registration

A very important application in Image Processing is “Image Registration”. It consists in aligning a pair of misaligned images that have been acquired from the same scene but from different camera positions, and possibly different cameras. Figure 2.5 shows an example of images that are expected to be registered against each other. Using some point feature detector, the relevant points describing both images can be selected and a point set matching algorithm is run. From the resultant matching, a set of parameters can be estimated, which will recover the position and orientations of one image with respect to the other (TON; JAIN, 1989).

In the next chapter, it is presented a general overview of the literature addressing graph matching problems.

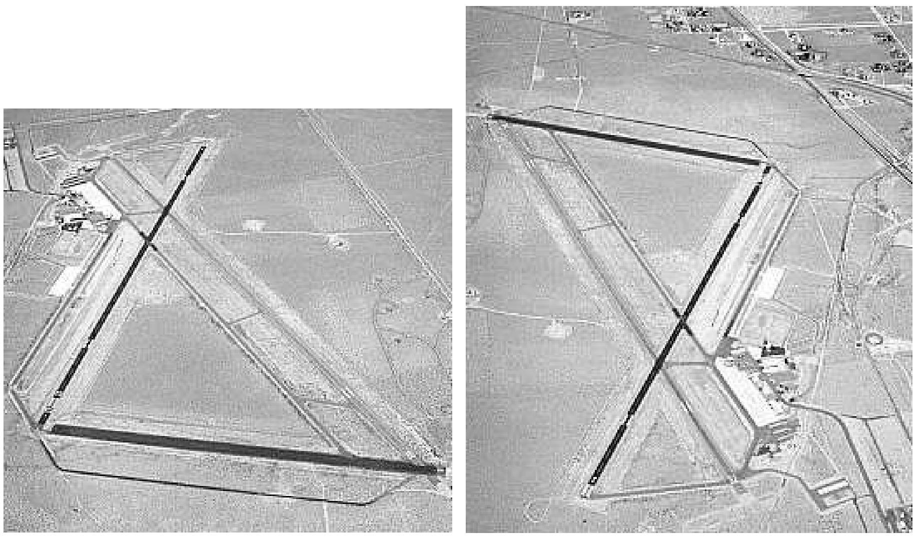


(a)

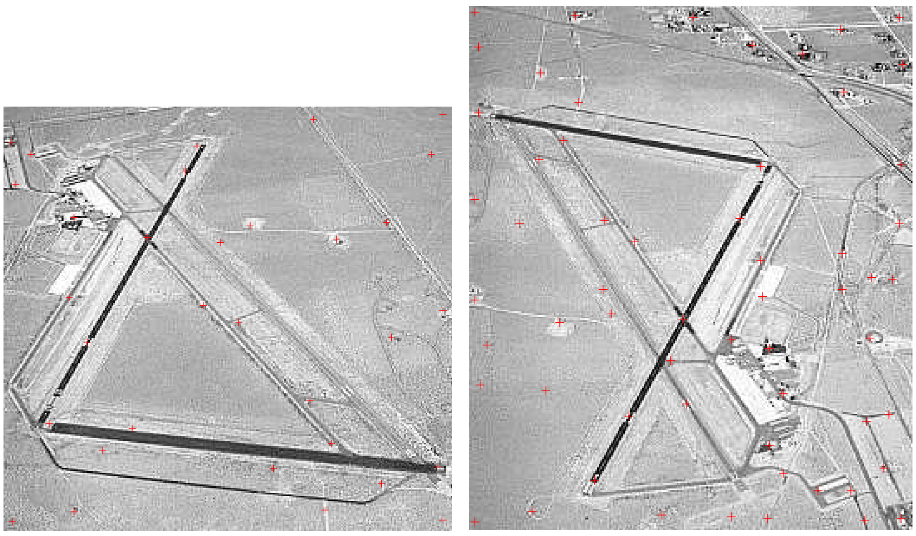


(b)

Figure 2.4: Point set matching applied to stereo matching: (a) left and right images; (b) left and right images with the landmark points detected in order to apply point set matching to derive the correspondence.



(a)



(b)

Figure 2.5: Point set matching applied to image registration: (a) two significantly different images acquired from the same scene; (b) the two images with the landmark points detected in order to apply point set matching, from which parameters can be extracted to perform image registration.

3 RELATED LITERATURE

In the early 1970's, it was already clear that powerful mathematical representations for complex structures could be obtained by using *graphs* (BARROW; POPPLESTONE, 1971; FISCHLER; ELSCHLAGER, 1973). This is due to the fact that graphs are abstract objects characterized by nodes (which play the role of structure parts) and edges (which represent the relations eventually present between subsets of parts). Since then, the problem of matching structural entities has been almost exclusively approached as a *graph matching problem*, which essentially consists in assigning each node of a graph to some node of a second graph aiming at achieving some "global consistency". In particular, different classes of the graph matching problem appeared, mainly represented by the *structural* and the *attributed* approaches. In the structural approach, a structure is represented by a "standard" graph: it is made up by a vertex set and an edge set denoting vertices that are in some sense related. In the attributed approach, a structure is represented by an *attributed graph*, which has attribute vectors associated to vertices and edges. Representing a graph solely by its adjacency structure (the structural approach) has the advantage of providing abstractions that convey important visual invariances (LUO; HANCOCK, 2001). However, attributed graphs are suitable for encoding complex dependencies in a straightforward way, and powerful representations are obtained when appropriate features are selected such that a certain degree of invariance is achieved (CHRISTMAS; KITTLER; PETROU, 1994; LI, 1992). Apart from this classification into structural or attributed approaches, matching can also be *exact* or *inexact*.

Not only these two different classes of graph matching problems have been identified, but also two different classes of techniques or algorithms. Gold and Rangarajan (GOLD; RANGARAJAN, 1996) divided graph matching algorithms into search-based, which rely on the construction of a state-space and have worst case exponential complexity, and non-search methods, which are based on nonlinear optimization techniques, or heuristic approximations, and are in general of polynomial complexity. The first class of algorithms finds the global optimum of the matching task, while in the second this is not guaranteed. Several of the early attempts to graph matching fall into the first category. Typical instances of the first class are graph decomposition methods and tree search (ULLMAN, 1976; MESSMER; BUNKE, 1998; BERRETI; BIMBO; VICARIO, 2001), but several other techniques also exist (FU, 1983; ESHERA; FU, 1984; TSAI; FU, 1983; BOYER; KAK, 1988). In the second class, which has gained increasing popularity since its appearance in the late 1970's, the most popular and widely used approach is based on probabilistic relaxation labeling (ROSENFELD; HUMMEL; ZUCKER, 1976; BHANU, 1984; BHANU; FAUGERAS, 1984; DAVIS, 1979; FAUGERAS; BERTHOD, 1981; ULMANN, 1979; ROSENFELD; KAK, 1982; CHRISTMAS; KITTLER; PETROU, 1994; LI, 1994; HUMMEL; ZUCKER, 1983). Techniques based on spectral analysis and leastsquares methods (UMEYAMA, 1998; SHAPIRO; BRADY, 1992; WYK; WYK, 2003; WYK; DURRANI; WYK, 2002), graduated assignment (GOLD; RANGARAJAN, 1996), genetic optimization (SUGANTHAN, 2002) and other principles (SIMIC, 1991; SUGANTHAN; TEOH; MITAL, 1995; PELILLO, 1999; PELILLO; SIDDIQI; ZUCKER, 1999) have also been proposed in recent years.

In this chapter, a discussion of some of the most important approaches for matching developed along the years is presented. Most of what follows is based on (CONTE et al., 2004) and (HANCOCK; WILSON, 2002).

3.1 Exact Matching

We first address briefly the exact matching problem. Although real problems in most application domains involve predominantly inexact matching, it is elucidating to start with the idealized problem of exact matching since it provides some basic terminology and concepts to be further explored in the description of the inexact matching problem.

Exact matching, as seen in the previous chapter, is characterized by the fact that the mapping between the vertices of the two graphs must be edge-preserving. The most stringent form of exact matching is *graph isomorphism*, which consists in a bijective mapping between graphs with the same number of vertices. *Subgraph isomorphism* consists in a graph isomorphism between a graph and a subset of the other graph. A *monomorphism* is an injective mapping, such that different nodes in the first graph are mapped to distinct nodes in the second graph. Finally, the most general form of correspondence is named *homomorphism*, which consists in a total function: the only requirement is that each vertex in the first graph must map to a unique vertex in the second graph.

All the above forms of graph matching are NP-complete, except for graph isomorphism, which has not yet been proven to be NP-hard or not.

3.1.1 Search-based methods

Most exact matching algorithms involve some sort of tree searching and backtracking. The first relevant algorithm of this kind is due to Ullman (ULLMAN, 1976). This algorithm tackles graph isomorphism, subgraph isomorphism and monomorphism, and is still very popular. It is essentially a branch and bound algorithm that progressively expands partial matchings until a global matching has been reached or, alternatively, a point is reached where no additional matchings are possible given the constraints. If this is the case, the algorithm backtracks until it finds a partial matching that meets the constraints.

A more recent version of exact search-based exact matching algorithm is due to Cordella et al. (CORDELLA et al., 1998). In this work, a heuristic is introduced which is based on the analysis of the sets of nodes adjacent to those already taken into account in the partial mapping. It turns out that this heuristic is fast to compute, and significant improvements over the Ullman algorithm have been observed.

Another recent tree search method for isomorphism was proposed by Larrosa and Valiente (LARROSA; VALIENTE, 2002), where the authors pose the problem of isomorphism as a constraint satisfaction problem (CSP). They use heuristics developed in the CSP literature in order to solve the isomorphism problem.

3.1.2 Miscellaneous methods

One of the most effective algorithms for graph isomorphism is not based on three search, but on group theory. It was developed by Mckay in 1981 (MCKAY, 1981), and is called *Nauty*. The automorphism group of each graph is constructed and from it a canonical labeling of each graph is obtained. In this way, isomorphism between two graphs can be checked by comparing the adjacency matrix of their canonical form. The equality can be done in $O(N^2)$ time, but the construction of the canonical labeling has worst case exponential time complexity. On average, however, the algorithm is very effective and is considered by many authors to be the fastest isomorphism algorithm available (CONTE et al., 2004).

Another important algorithm specifically designed to match an input graph against a large library of graphs is due to Messmer (MESSMER, 1995). The approach is based on the decomposition of each graph in the library into smaller subgraphs. The matching process then exploits the fact that some parts of the graphs are similar and redundant comparison is avoided. Some other recent papers have proposed the use of decision trees to speed up the matching against a large database of graphs (IRNIGER; BUNKE, 2001; LAZARESCU; BUNKE; VENKATESH, 2000).

3.2 Inexact Matching

In many real applications, the observed graphs are subject to a number of factors that may deform their original structure: noisy conditions in the acquisition process, intrinsic variability of features within a given pattern and nonlinear deformations of sensing apparatus are some examples.

These unavoidable factors make exact graph matching not very realistic, since exact similarities may never be encountered in practice. Aiming at dealing with these uncertainties, researchers have developed what is known as *inexact* matching algorithms, or *error-tolerant* matching algorithms. Instead of forbidding matches that are not edge-preserving, a *cost* is associated to these imperfect assignments. As a result, the purpose of inexact matching algorithms is to find a particular mapping that minimizes some global cost measure. Note that exact matching may be considered in this formulation as a special case where the involved cost is zero.

Optimal inexact matching algorithms are those that find the global minimum of the cost function. All existent algorithms of this class have worst case exponential time complexity. *Approximate* inexact matching algorithms only guarantee that a local minimum of the cost function will be found. On the other hand, many algorithms of this class have polynomial time complexity.

3.2.1 Search-based methods

The idea of searching in trees with backtracking, used in exact matching, is also present in the development of inexact matching algorithms. The first of these algorithms proposed in the literature is due to Tsai and Fu (TSAI; FU, 1979). This work defines attributed relational graphs (ARGs) and a possible error measure in the matching of two ARGs. The algorithm suggests operations of node and edge substitution, and was later extended to incorporate other operations like node insertion and deletion (TSAI; FU, 1983).

Sanfeliu and Fu further developed these ideas by introducing the definition of a graph edit distance measure (SANFELIU; FU, 1983). This is obtained by considering a canonical set of operations defined by node and edge substitution as well as node splitting and merging. In the sequel, Eshera and Fu (ESHERA; FU, 1984) proposed a method for distance computation, which is based on defining appropriate simple subgraphs and approximating the matching by finding an optimal matching of these subgraphs via dynamic programming.

Shapiro and Haralick proposed in 1981 an algorithm for optimal error-correcting homomorphism, which is based on branch and bound with the use of heuristics (SHAPIRO; HARALICK, 1981). In 1985, they proposed a distance measure between relational descriptions that satisfies the requirements of a metric (SHAPIRO; HARALICK, 1985).

Some recent contributions using tree search - specifically the A^* algorithm - are due to Berreti et al. (BERRETI; BIMBO; VICARIO, 2001) and Gregory and Kittler (GREGORY; KITTLER, 2002). These works use heuristics that take into account estimates of the cost of future partial matchings.

3.2.2 Continuous optimization

A different approach to inexact matching consists in formulating the problem as one of a continuous function optimization, instead of purely discrete search. This allows for the use of continuous optimization techniques to solve the matching problem. These algorithms do not assure global optimality, but very often they provide fast and accurate solutions.

The most important representatives of this class are methods based on Probabilistic Relaxation Labeling (PRL). The first forms of relaxation labeling appeared in early 1970's with the work of Fischler and Elschlager (FISCHLER; ELSCHLAGER, 1973). In 1976, Rosenfeld et al. introduced a formalism for relaxation applied to scene labeling (ROSENFELD; HUMMEL; ZUCKER, 1976). The fundamental idea of this approach consists in assigning *labels* to vertices in the first graph that correspond to the vertices in the second graph. This is done in parallel such that the likelihood that a given vertex is assigned to a given label is related to the support that neighbors of this vertex provide to that label. It is essentially an iterative heuristic algorithm that updates the likelihood of assignment according to evidence provided by neighboring sites.

The initially heuristic formulation of relaxation labeling was along the years changed into a more principled probabilistic formulation. In subsequent works, Kittler and Hancock (KITTLER; HANCOCK, 1989) and Christmas et al. (CHRISTMAS; KITTLER; PETROU, 1994) further developed a Bayesian probabilistic relaxation framework that has proven to be very successful for attributed graph matching problems. In a similar path, Hancock and associates (HANCOCK; WILSON, 2002) have explored several formulations of graph matching in terms of probabilistic relaxation. They have introduced principled models of matching errors and used Bayesian probabilistic relaxation in several of their contributions. It is fair to say that their work has established a new view of graph matching in terms of probabilistic relaxation labeling that has had a significant impact in the research community.

In 1996, Gold and Rangarajan introduced the Graduated Assignment algorithm for graph matching (GOLD; RANGARAJAN, 1996). In this algorithm, a technique called *graduated nonconvexity* is used to avoid local minima solutions. The algorithm presents impressive results when compared to standard probabilistic relaxation labeling and has low order polynomial time complexity. However, as all the other algorithms based on continuous optimization, it does not assure reaching the global minimum of the matching cost.

3.2.3 Spectral methods

Alternatively to search-based and continuous optimization techniques, graph matching has also been approached using spectral methods. The key idea underlying this approach is that the eigenvalues and eigenvectors of the adjacency matrix of a graph are invariant with respect to node permutations. As a result, isomorphic graphs will have adjacency matrices with the same set of eigenvalues and eigenvectors. The opposite does not hold. If the eigenvalues and eigenvectors are the same, it is not possible to infer that the graphs are isomorphic. However, since the computational complexity of spectral decomposition is attractive, researchers have been interested in using the spectral signature of graphs to measure their similarities.

The pioneering study in this area was performed by Umeyama in 1988 (UMEYAMA, 1998). This work studies the weighted graph matching problem in the isomorphism case. There are important limitations such as the restriction that the number of nodes in both graphs must be the same and the matching matrix must be a permutation matrix. The author derives in closed form the orthogonal matrix that optimizes the objective function, assuming that the graphs are isomorphic. If it is not known that the graphs are isomorphic, the method can produce very unsatisfactory results.

A recent alternative method was proposed by Xu and King (XU; KING, 2001), who used Principal Component Analysis (PCA) and gradient descent to find the optimum of the cost function. The resulting method is claimed to outperform Umeyama's both in speed and accuracy.

Shapiro and Brady (SHAPIRO; BRADY, 1992) proposed a method in which point sets are matched by comparing the eigenvectors of the point proximity function, where the proximity function is build using the Gaussian function. The similarity of point patterns can then be evaluated by comparing the pattern of eigenvectors.

Carcassoni and Hancock (CARCASSONI; HANCOCK, 2003) also developed an algorithm for spectral correspondence of point sets. The method consists in clustering nodes that are likely to be matched. Then the algorithm exploits this hierarchy and first matches the clusters and just after that matches the nodes within clusters. Differently from Umeyama's method, this technique can be applied to graphs of different size.

Kosinov and Caelli also introduced a method that combines clustering and spectral decomposition (KOSINOV; CAELLI, 2002). In this method, the eigenvectors of the adjacency matrix are used to form a vector space denoted “graph eigenspace”. The nodes of the graphs are then projected on this space, and a clustering algorithm is responsible for finding the nodes that should match. The method is very robust to graph distortions.

3.2.4 Miscellaneous methods

Several miscellaneous algorithms have been proposed for graph matching. There are methods based on Neural Networks and Genetic Algorithms, like those proposed by Suganthan et al. (SUGANTHAN; TEOH; MITAL, 1995), Suganthan and Yan (SUGANTHAN; YAN, 1998) and Suganthan (SUGANTHAN, 2002). There are also methods based on evolutionary game theory (PELILLO, 1999), which have been used to match shock graphs (PELILLO; SIDDIQI; ZUCKER, 1999).

The next chapter presents the background knowledge needed in order to introduce the matching technique proposed in this work.

4 BACKGROUND: GRAPHICAL MODELS AND EXACT INFERENCE

This chapter presents the background knowledge necessary to understand the technical work of this Thesis. Essentially, a brief description of the theory of probabilistic graphical models is introduced, emphasizing undirected graphical models. Although the expression "graphical models" may represent several different concepts, depending on the research domain that one considers, its meaning here is the one used by the statistical machine learning research community, and consists of a model defined by a graph whose nodes represent random variables and whose edges represent possible dependencies between those random variables. The classical reference for a formal description of Graphical Models is (LAURITZEN, 1996).

4.1 Probabilistic Inference

Consider a set of random variables (rv's) $\{X_1, X_2, \ldots, X_n\}$, such that x_i is the realization of a rv X_i. These variables may be scalar-valued or vector-valued, continuous or discrete. In this work, they will always be scalar-valued and discrete, so that from now on this will be implicit wherever random variables are mentioned. Given this collection of random variables, it may be of interest to perform "queries" about subsets of them. For example, it may be of interest the computation of the marginal probability of one subset of these variables. It may be either the case that one is interested in computing the conditional probability of a particular subset of variables given another particular subset. Another important and relevant problem is that of evaluating what is the most likely realization of a subset of random variables if the realization of the complement set is known. These "query" problems are called *probabilistic inference* problems.

In principle, such queries can be answered if one has available the joint probability distribution $P(X_1, X_2, \ldots, X_n)$. For example, if X_A denotes the set of random variables indexed by an index set A and U is the complete index set, it is possible to obtain the marginal probability distribution over the set X_A simply by summing the joint distribution over the complement of X_A:

$$P(X_A) = \sum_{X_U \setminus X_A} P(X_1, X_2, \ldots, X_n). \tag{4.1}$$

Similarly, for calculating the most likely completion (or Maximum a Posteriori - MAP - estimate) with respect to a subset X_A it is necessary to maximize the joint distribution over the complement variables:

$$MAP(X_A) = \max_{X_U \setminus X_A} P(X_1, X_2, \ldots, X_n), \tag{4.2}$$

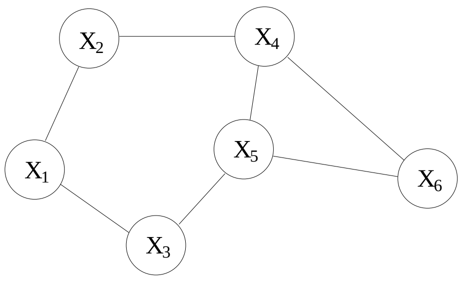


Figure 4.1: A particular undirected graphical model.

where $MAP(X_A)$ denotes the most likely estimate of the complement variables with respect to variables in X_A.

Conditional probabilities, in their turn, can be computed as a normalization of marginals:

$$P(X_A|X_B) = \frac{P(X_A, X_B)}{P(X_B)}. \tag{4.3}$$

In summary, probabilistic inference problems can be solved by computational manipulation of the joint distribution.

However, a detailed observation of the type of a generic representation for a non-parametric joint probability distribution over discrete random variables should convince us that the above naive solutions are infeasible for most real-world situations. Consider for example the most general representation for the joint distribution of a set of n discrete random variables that can assume r possible realizations: it can be seen as an n-dimensional table where each cell contains the probability $p(x_1, x_2, \ldots, x_n)$ for a specific realization $(x_1, x_2, \ldots, x_n)$ of the set of random variables $X_1, X_2, \ldots, X_n$. This representation needs a storage capacity of r^n numbers, which is exponential on n. Moreover, the computation of a marginal distribution, for example, involves in the worst case a sum which is also exponential on n (for calculating the marginal distribution of a singleton, there are $r.r^{n-1} = r^n$ elements to be summed). Useful models in most areas involve n in the dozens or hundreds. As a result, this naive tabular representation is not feasible.

Graphical Models provide a way of representing joint probability distributions in a more parsimonious manner. The key idea is to capture conditional independence relationships among variables and use them in order to factorize the joint distribution for facilitating the inference procedure.

4.2 Undirected Graphical Models

Graphical Models can be either *directed* or *undirected*. Directed graphical models are those where each edge has a particular direction, such that there is an asymmetric relation between neighbor nodes in the graph. In undirected graphical models, the edges have no direction (JORDAN, Forthcoming, 2004). The present work considers exclusively the undirected case. Undirected graphical models are also known as *Markov Random Fields* (MRFs). An undirected graphical model is a graph $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of nodes that have a bijective relation with a set of random variables and $\mathcal{E}$ is a set of edges. Although nodes and random variables are different objects, from now on we will simply say that a node of a graphical model *is* a random variable. Figure 4.1 shows an example of a graphical model with a particular connectivity.

4.2.1 Conditional independence

The first fundamental concept necessary to understand graphical models is that of conditional independence. It is said that a subset of variables X_A is conditionally independent on the subset X_C *given* X_B if the knowledge about X_C does not add any information to the knowledge of the distribution of X_A *if* it is known in advance the value of X_B. It is important to stress that this must hold for all possible realizations of X_B. We use the following notation to express this fact:

$$X_A \perp\!\!\!\perp X_C \mid X_B. \tag{4.4}$$

In terms of conditional probabilities, this fact can be described in the following manner:

$$P(X_A|X_B, X_C) = P(X_A|X_B). \tag{4.5}$$

In other words, the conditional probability of X_A given X_B is the same regardless if X_C is given or not. A common real-life example of conditional independence is the conditional independence of past and future given the present. All the past events that could affect the future are collapsed in the present, in the sense that there is no information from the past that can "be transfered" to the future without passing through the present:

$$\text{future} \perp\!\!\!\perp \text{past} \mid \text{present}. \tag{4.6}$$

4.2.2 Markov properties

In this subsection we consider conditional independence in the case when there is a set of random variables $\{X_i\}_{i\in\mathcal{V}}$ which take values in sample spaces $\{\mathcal{X}_i\}_{i\in\mathcal{V}}$. These variables correspond to the nodes of a graphical model, whose vertex set is $\mathcal{V}$. If A, B and C are sets of numbers that index variables, we will for simplicity of notation denote the assertion $X_A \perp\!\!\!\perp X_C \mid X_B$ as $A \perp\!\!\!\perp C \mid B$. For a given graph $G = (\mathcal{V}, \mathcal{E})$ and a set of random variables $\{X_i\}_{i\in\mathcal{V}}$, it is possible to define a series of properties (LAURITZEN, 1996):

Definition 4.1 (Pairwise Markov property ($\mathcal{P}$)) *A probability distribution p on $\mathcal{X}$ is said to be* pairwise Markov *with respect to an undirected graph G if, for any non-adjacent pair of nodes α and β in G, α is independent of β given the states of all the remaining nodes:*

$$\alpha \perp\!\!\!\perp \beta \mid \mathcal{V}\backslash\{\alpha, \beta\}. \tag{4.7}$$

Definition 4.2 (Local Markov property ($\mathcal{L}$)) *A probability distribution p on $\mathcal{X}$ is said to be* local Markov *with respect to an undirected graph G if, for any node α in G and given its boundary $bd(\alpha)$ (which is the set of nodes connected to α), α is independent of the remaining nodes:*

$$\alpha \perp\!\!\!\perp V\backslash\{bd(\alpha) \cup \alpha\} \mid bd(\alpha). \tag{4.8}$$

Definition 4.3 (Global Markov property ($\mathcal{G}$)) *A probability distribution p on $\mathcal{X}$ is said to be* global Markov *with respect to an undirected graph G if, for any triple (A, B, C) of disjoint subsets such that B separates A from C in G, A and C are conditionally independent given B:*

$$A \perp\!\!\!\perp C \mid B. \tag{4.9}$$

These Markov properties are tied together via the following result (LAURITZEN, 1996):

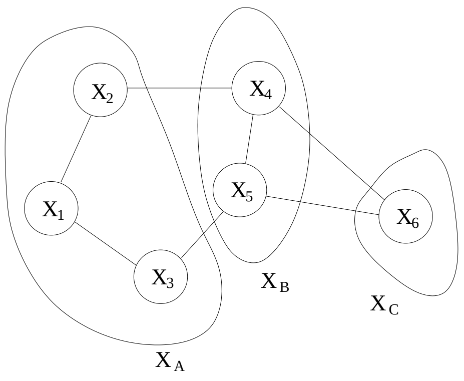


Figure 4.2: The concept of conditional independence in undirected graphical models.

Proposition 4.1 *($\mathcal{G} \Rightarrow \mathcal{L} \Rightarrow \mathcal{P}$) For any undirected graph G and any probability distribution on $\mathcal{X}$ it holds that*

$$(\mathcal{G}) \Rightarrow (\mathcal{L}) \Rightarrow (\mathcal{P}). \qquad (4.10)$$

In particular, if p is strictly positive ($p(x) > 0, \forall x \in \mathcal{X}$), the following theorem holds:

Theorem 4.1 (Pearl and Paz ($\mathcal{G} \Leftrightarrow \mathcal{L} \Leftrightarrow \mathcal{P}$)) *If the probability distribution p is strictly positive on $\mathcal{X}$, the pairwise, local and global Markov properties are equivalent:*

$$(\mathcal{G}) \Leftrightarrow (\mathcal{L}) \Leftrightarrow (\mathcal{P}). \qquad (4.11)$$

The importance of the global Markov property lies in the introduction of a general criterion for inferring when the sets of variables A and C are conditionally independent given a third set of variables B. Figure 4.2 shows, for the graphical model in Figure 4.1, a separator set X_B and two sets (X_A and X_C) who are conditionally independent given X_B.

Conditional independence statements can be read directly from an undirected graph. It is said that X_A is independent of X_C given X_B if the set of nodes X_B separates the sets of nodes X_A and X_C, in the graph-theoretic sense. This means that if every path from any node in X_A to any node in X_C includes at least one node in X_B, then $X_A \perp\!\!\!\perp X_C \mid X_B$ holds. Otherwise, $X_A \perp\!\!\!\perp X_C \mid X_B$ does not hold. Figure 4.2 illustrates this fact.

4.2.3 Cliques, potentials and factorization

It is important to introduce some additional key concepts regarding graphical models. The first basic concept is that of a *clique*. A clique in a graphical model is a set of nodes in which every pair of nodes is connected by an edge. It is also necessary to define a trivial clique: a single node is itself a clique. A *maximal clique* is a clique which is not a proper subset of another clique. Figure 4.3 shows examples of cliques and maximal cliques.

A *potential function* ψ_C of a particular clique C is defined as a function that associates to each joint realization of the random variables in C a positive real number (which is called “potential”):

$$\psi_C : \mathcal{X}_C \rightarrow \mathbb{R}^+, \qquad (4.12)$$

where $\mathcal{X}_C$ denotes the probability subspace involving exclusively the variables in clique C. In practice, potential functions are seen as measures of likelihood for a given configuration. Note an important detail about potential functions in face of the definitions

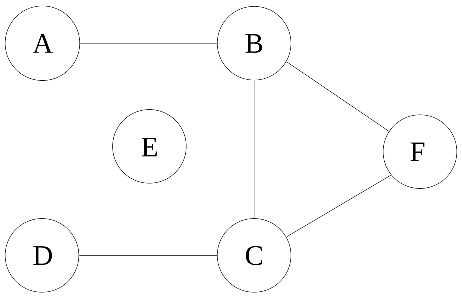


Figure 4.3: Cliques and Maximal cliques. Every node is a clique (A,B,C,D,E,F), every connected pair of nodes is a clique (AB,BC,CD,AD,BF,CF) and the triple BCF is a clique. The Maximal cliques are AB,CD,AD,BCF and E.

regarding cliques: since every clique is a maximal clique or a proper subset of it, potential functions of non-maximal cliques may be embedded into the potential function of their correspondent maximal cliques. Consider the example in Figure 4.3. The clique BC is a proper subset of the clique BCF. So, the potential ψ_{BCF} may actually include in its expression a factor ψ_{BC}.

A formal definition of factorization in graphical models is now introduced:

Definition 4.4 (Factorization Property ($\mathcal{F}$)) *A probability distribution p on $\mathcal{X}$ is said to* factorize *with respect to an undirected graph G if it can be expressed as the product of potential functions over the set $\mathcal{C}$ of maximal cliques in G:*

$$p(x) = \prod_{C \in \mathcal{C}} \psi_C(x_C). \tag{4.13}$$

The potentials ψ_C are nonnegative and such that $\sum_x p(x) = 1$. Apart from this they are arbitrary. They may or may not factorize into products of sub-potentials over smaller cliques.

Now it is possible to present important results on the relations between Markov properties and factorization.

4.2.4 Factorization implies Markovianity

Two results connect Markov properties and the Factorization property. Here the first result is shown and in the next subsection the most important result is introduced.

The factorization property is related formally to the Markov properties via the following result (LAURITZEN, 1996):

Proposition 4.2 *(Gibbs $\Rightarrow$ Markov) For any undirected graph G and any probability distribution on $\mathcal{X}$ it holds that*

$$\mathcal{F} \Rightarrow \mathcal{G} \Rightarrow \mathcal{L} \Rightarrow \mathcal{P}. \tag{4.14}$$

We denote this important result as the “Gibbs ⇒ Markov” result, as it is usually known in Statistical Image Processing and Vision. The “Gibbs” label refers to the fact that the factorized form of the joint distribution is equivalent to a Gibbs distribution, as it is known in Statistical Physics.

In summary, the following implication diagram involving these properties arises:

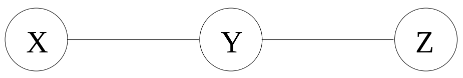


Figure 4.4: A graph with conditional independence given by $X \perp\!\!\!\perp Z \mid Y$.

- For general distributions p:

$$\mathcal{F} \Rightarrow \mathcal{G} \Rightarrow \mathcal{L} \Rightarrow \mathcal{P}. \tag{4.15}$$

- For strictly positive distributions p:

$$\mathcal{F} \Rightarrow \mathcal{G} \Leftrightarrow \mathcal{L} \Leftrightarrow \mathcal{P}. \tag{4.16}$$

Markov Random Fields, as they are understood in Image Processing and related areas, are defined as undirected graphical models where the positivity condition and the Markov properties hold. Note, however, that these two facts do *not* imply a factorized form for the joint distribution from the above. Only the converse was shown to be true: factorization implies Markovianity. In the following the presentation is completed by the introduction of the result which is lacking. The result implies that factorization is not only a sufficient condition for Markovianity, but also a necessary condition. This will allow us to faithfully represent Markov Random Fields via a factorized distribution.

4.2.5 Markovianity implies factorization: The Hammersley-Clifford Theorem

We can think of a particular graphical model as a “filter” (JORDAN, Forthcoming, 2004) in which only joint distributions that satisfy the conditional independency assumptions implied by the graph are accepted. What is the most general form for the joint distribution such that all conditional independency assumptions are strictly respected? This is a fundamental question, and before presenting the theorem that answers it, a brief intuitive investigation into the question is presented.

Let us start with an example. Consider the graphical model in Figure 4.4.

This graphical model has an associated conditional independence assumption given by $X \perp\!\!\!\perp Z \mid Y$. The joint distribution over the variables can be then factorized as follows:

$$\begin{aligned} p(x,y,z) &= p(x|y,z)p(y,z) && (4.17)\\ &= p(x|y)p(y,z) && (4.18)\\ &= \frac{p(x,y)}{p(y)}p(y,z) && (4.19)\\ &\equiv \psi_1(x,y)\psi_2(y,z) && (4.20) \end{aligned}$$

where we have defined the functions $\psi_1(x,y) \equiv p(x,y)/p(y)$ and $\psi_2(y,z) \equiv p(y,z)$. Note that it was possible to factorize the joint distribution over three variables into a product of functions over only two variables. This is important because, as seen in chapter 2, the complexity of both storing and manipulating the numbers in a given distribution is exponential on the number of variables in the distribution table. This apparently simple procedure is the core of graphical models: to exploit systematically the conditional independence assumptions implied by the graph in order to factorize the joint distribution into a product over *local* sets of variables. In our example, these local sets of variables are $\{X, Y\}$ and $\{Y, Z\}$. The fact that the resulting local variables are connected by an edge in the graph is not a coincidence (note that X is connected to Y and Y is connected to Z). Neither it is a coincidence that unconnected variables (X and Z in our example) do *not* appear as arguments of a same factor function ψ. It will be seen in a moment that these facts emerge from a very important theorem.

An important observation is that, in order to construct a factorization in terms of the variables in separate cliques, it is necessary to introduce the factor functions ψ_1 and ψ_2 which in general are not simply conditional distributions. In our particular example, ψ_1 is a conditional, but ψ_2 is a marginal distribution. So, the functions ψ that allow for the factorization of the joint distribution in terms of the maximal cliques are more general objects. Indeed, they correspond to what was defined as being potential functions and, in general, they convey the intuitive meaning of likelihood of a particular realization.

This simple but important example reveals a fact that is derived indeed from a deep result, which is valid for arbitrary graphs, and was first proven in 1971 by Hammersley and Clifford (HAMMERSLEY; CLIFFORD, 1971), although published versions are due to other authors (BESAG, 1974). The result bears their names, being known as the Hammersley-Clifford (HC) theorem. It can be formalized as follows:

Theorem 4.2 (Hammersley-Clifford ($P \Rightarrow F$)) *Any strictly positive distribution which satisfies the pairwise Markov property with respect to a particular arbitrary graph factorizes with respect to this graph.*

This Theorem, when tied to Theorem 4.1, gives us a particularly suitable form for this result:

Theorem 4.3 (Hammersley-Clifford ($P \Rightarrow F$, $L \Rightarrow F$, $G \Rightarrow F$)) *Any strictly positive distribution which satisfies any type of Markov property with respect to a particular arbitrary graph factorizes with respect to this graph.*

which from now on will be denoted as the “HC theorem”. Note that, in the example in Figure 4.4, the joint distribution was decomposed into a product of potentials over pairwise cliques (in this case the maximal cliques). This is precisely what the theorem says: it tells us that any strictly positive joint distribution which respects the Markov properties in some graph can be factorized as a product over functions whose arguments involve only nodes in the maximal cliques of this graph. In other words, if the maximal cliques have a moderate size one may be able to deal with low dimensional tables, what possibly will render inference problems feasible, as seen in the previous section.

A real problem still persists. It is necessary to choose the potential functions as being such that

$$\sum_x \prod_{C \in \mathcal{C}} \psi_C(x_C) = 1 \tag{4.21}$$

in order to guarantee that the factorization is a probability distribution. It turns out that this is extremely inconvenient, since in most applications a high degree of flexibility in choosing the potential functions is required. If one needs this flexibility in choosing arbitrary non-negative potential functions, it is necessary to pay the price of normalizing the factorized form such that the entire expression still sums to one:

$$p(x) = \frac{\prod_{C \in \mathcal{C}} \psi_C(x_C)}{Z} \tag{4.22}$$

where Z,

$$Z \equiv \sum_{x} \prod_{C \in C} \psi_C(x_C), \tag{4.23}$$

is a sum that runs over all configurations in the sample space $\mathcal{X}$ - an infeasible calculation. However, there is a particular type of inference problem where the knowledge of Z is irrelevant to the problems' solution (the calculation of most likely completions or Maximum a Posteriori probabilities). Fortunately, we have managed to formulate the problem of this Thesis precisely in this particular instance form, as will be seen.

The HC theorem allows us to change the "implication diagram" into an "equivalence diagram". For strictly positive distributions p, now the following holds:

$$\mathcal{F} \Leftrightarrow \mathcal{G} \Leftrightarrow \mathcal{L} \Leftrightarrow \mathcal{P} \tag{4.24}$$

In other words, if every configuration in a graphical model has strictly positive probability of occurrence, then all Markov properties are equivalent and, more importantly, they imply that the joint distribution can be factorized into a product of potential functions over the maximal cliques of the graph.

We are now able to exploit this factorization property of Markov Random Fields in order to describe efficient algorithms for probabilistic inference.

4.3 Exact Inference and the Junction Tree Framework

So far it has been shown that the joint distribution in Markov Random Fields is a factorized expression over functions of variables in the maximal cliques of the graph. How can probabilistic inference be realized effectively with this simplified form of the joint distribution? This section presents an answer to this question, which involves what is known as the *Junction Tree* framework for probabilistic inference.

During the past decades, a few efficient algorithms for inference in graphical models without cycles were developed (PEARL, 1988; LAURITZEN, 1996; SHAFER; SHENOY, 1990). All of these algorithms involve some form of dynamic programming. Examples are the Viterbi and forward-backward algorithms for chain graphs and belief propagation (or sum-product) for tree graphs. In general, what has been discovered is that in graphs with no cycles the dynamic programming principle can be applied in order to efficiently accomplish exact inference.

However, in general graphs, which may involve cycles, these algorithms fail to give exact solutions. This limitation encouraged researchers to rethink the graph structure over which the algorithms should run, not the algorithms themselves. The idea was to transform an arbitrary graph in a "hypergraph" where nodes consist of the maximal cliques of the triangulated graph, and derive a way to guarantee that this general graph would be a tree (even if the original one had cycles). Then the same efficient algorithms for trees could be used on these hypergraphs, and the only difference would be that the final marginals would not be associated to singleton nodes, but with an entire clique. Individual marginals then could be obtained by marginalizing out the remaining variables within the same clique.

The above is a very short primer to what was later formally developed as the *Junction Tree* Framework for exact inference, which will now be described in detail.

4.3.1 The Junction Tree framework

The Junction Tree framework consists in a set of algorithms for systematic exact inference on arbitrary graphical models. Several different algorithms for performing inference are available (e.g. the Hugin, Shafer-Shenoy and Lauritzen-Spiegelhalter algorithms (VASILICA; PRAKASH, 1998)) in this framework, but essentially they lie on the same principle: to accomplish local computations systematically in a special data structure - a "Junction Tree".

We will be concerned with a particular algorithm called the "Hugin algorithm" (LAURITZEN, 1996). However, the same results would be obtained by using either the Shafer-Shenoy or the Lauritzen-Spiegelhalter algorithm, since all of them perform exact inference. For a comparison among the three algorithms, see (VASILICA; PRAKASH, 1998). The presentation that follows is mainly based on (JORDAN, Forthcoming, 2004).

The Hugin algorithm is a "message-passing" algorithm, where "messages" are seen as local computations involving two maximal cliques of the graph. Its general view for undirected graphical models can be described as follows:

- **Triangulation.** The first step is to *triangulate* the graph, or equivalently to make it a *chordal graph* by proper insertion of edges.

- **Junction Tree construction.** A hypergraph (called a *Junction Tree*), where the nodes are the maximal cliques of the original graph, is constructed.

- **Initialization of potentials.** The nodes of this Junction Tree are initialized with potentials in a proper manner.

- **Propagation.** A "message-passing" or "probability propagation" algorithm is run on this Junction Tree in order to systematically update the potentials.

Once the algorithm finishes, each node of the Junction Tree is guaranteed to have a potential which is proportional to the marginal distribution (or proportional to the MAP distribution, depending on the version of Hugin used) for the node. Each of these steps will be detailed in what follows.

4.3.2 Graph triangulation

Graph triangulation is the first step in the algorithm. Recall from graph theory that a triangulated (or chordal) graph is a graph with no chordless cycles (WEST, 2001). A chord in a cycle is an edge between non-consecutive nodes in that cycle. Figure 4.5 shows an example of graph triangulation. The graph on the left is not triangulated because there are two cycles that have no chord (ABDCA and CDFEC). In order to triangulate this graph, it is necessary to create at least one chord in each of these cycles, by inserting edges BC and DE, for example, as shown in the graph on the right.

This triangulation step is necessary for the following reason: there is a result which states that a graph has a Junction Tree if and only if it is triangulated (JORDAN, Forthcoming, 2004; LAURITZEN, 1996). Although a Junction Tree has not yet been defined, it was already written that the inference algorithm runs over such a structure, what forces the accomplishment of the triangulation step.

The same graph can be triangulated in different ways. It will be seen that there are triangulations which are "optimal", in the sense that the Junction Tree that arises from

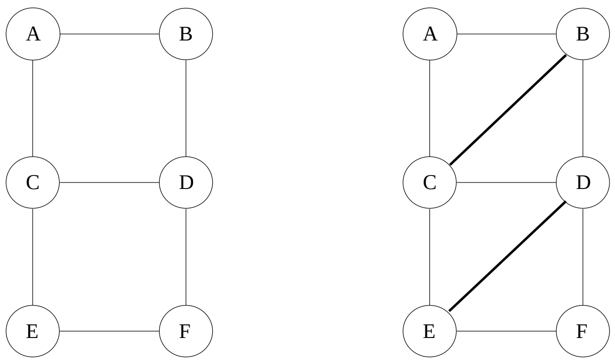


Figure 4.5: Left: a non-triangulated graph. Right: a possible triangulation for the graph on the left.

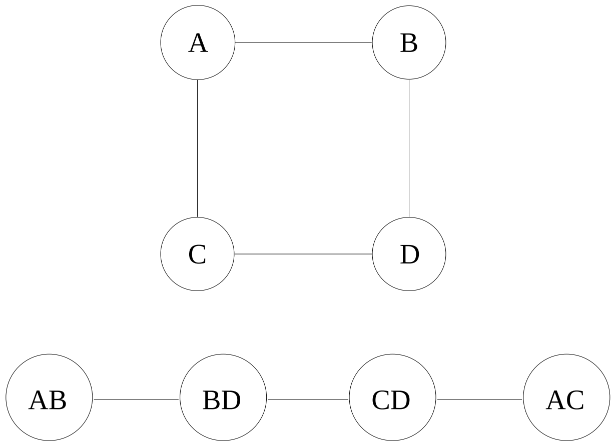


Figure 4.6: A graph on the top and a corresponding clique tree on the bottom.

this triangulation has a maximal clique size which is minimum among all possible triangulations (what impacts on the complexity of the propagation phase, as will be shown). As a result, if one is concerned with optimal models, an optimal triangulation should be computed. The problem of computing an optimal triangulation for arbitrary graphs is, however, known to be NP-hard (JORDAN, Forthcoming, 2004). This would be a significant problem if triangulation were not an "off-line" process in the whole algorithm: note that for a given model it is necessary to triangulate just once the graph and obtain its Junction Tree. From this moment on, the model can be applied to different inference problems by only executing the initialization and propagation phases. Moreover, in many problem domains the graph is *already* triangulated (as in the problem tackled in the present work), what makes vacuous this step of the algorithm.

4.3.3 Junction Tree construction

The first key concept for understanding the construction of a Junction Tree is that of a *clique tree*. A clique tree of a graph is another graph (a "hypergraph") where the nodes (which are called *clique nodes*) correspond to the maximal cliques of the original graph, and the connectivity is such that there are no loops in the hypergraph. Figure 4.6 shows a graph (on the top) and a possible clique tree for this graph (on the bottom).

The Junction Tree algorithm for inference in a graphical model runs over a clique tree of this graphical model. However, not all clique trees are appropriate for the purpose of the algorithm. The associated clique tree must have what is called the *running intersection*

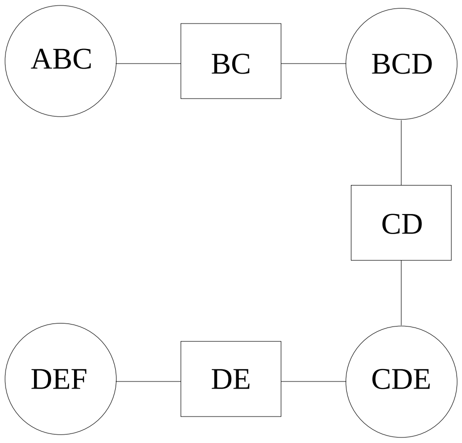


Figure 4.7: A Junction Tree for the triangulated graph in Figure 4.5.

property. The running intersection property asserts that all clique nodes (k) in the path between any two nodes (i, j) must contain their intersection. If $\mathcal{B}$ is the set of clique nodes, N_i is the set of nodes in clique node i and $\mathcal{K}_{ij}$ is the set of clique nodes in the path from i to j, this property can be formalized as

$$N_i \cap N_j \subset N_k, \forall\, i, j \in \mathcal{B}, \forall\, k \in \mathcal{K}_{ij}. \tag{4.25}$$

If a clique tree possesses the running intersection property, it is called a Junction Tree. Note that the clique tree in Figure 4.6 is *not* a Junction Tree, because node A, which appears in the clique nodes AB and AC, does not appear in clique nodes in the path from AB to AC. Indeed, this is not a surprise because the graph in Figure 4.6 is not triangulated, so all clique trees associated to this graph will not present the running intersection property. The equivalence between the existence of a Junction Tree and triangularity of the graph is what forces the triangulation of the graph in the first step of the algorithm, since a Junction Tree is needed in order to run the propagation step.

Consider again the triangulated graph in Figure 4.5. Since it is triangulated, it has a Junction Tree representation. In Figure 4.7, it is shown a possible Junction Tree for this graph. In this Junction Tree, it is introduced explicitly what are called the *separator nodes*. They are denoted in rectangles and represent the intersection set of the neighboring clique nodes. This is done as a matter of convenience: as we will see, the propagation phase of the algorithm involves a transfer of information between two clique nodes in such a way that their separator plays a prominent role. By inserting the separators, the joint distribution can be written as

$$p(x) = \frac{\prod_C \psi_C(x_C)}{\prod_S \phi_S(x_S)} \tag{4.26}$$

where $\phi_S(x_S)$ is a separator potential function. It is not difficult to see that this representation is equivalent to the one given by the HC theorem:

$$p(x) = \prod_{C \in C} \psi_C(x_C). \tag{4.27}$$

This is because the separators are subsets of the maximal cliques and thus the joint distribution remains unaltered in this new parameterization (JORDAN, Forthcoming, 2004).

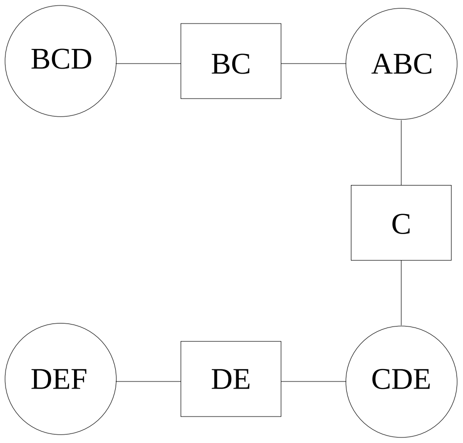


Figure 4.8: A clique tree for the triangulated graph in Figure 4.5 which is not a Junction Tree.

But a question remains: how are the clique nodes connected in order to form a Junction Tree? One solution is given by Kruskal's algorithm (JORDAN, Forthcoming, 2004), which consists in beginning with no edges and, at each step, add an edge that has maximal separator cardinality (without creating loops in the graph). Once there is a path between any pair of cliques (once the graph is connected), the resulting tree will be a Junction Tree (JORDAN, Forthcoming, 2004). Figure 4.8 shows a clique tree for the triangulated graph in Figure 4.5 which is *not* a Junction Tree. Note that the sum of the cardinalities of the separator sets is $|\{BC\}| + |\{C\}| + |\{DE\}| = 5$. This sum for the Junction Tree in Figure 4.7 is 6. It is possible to prove that a clique tree of a triangulated graph is a Junction Tree if and only if this number is maximal among all possible clique trees (JORDAN, Forthcoming, 2004). There may exist, however, more than one Junction Tree.

4.3.4 Initialization of potentials

Once the graph has been triangulated and a Junction Tree has been constructed, it is necessary to "initialize the clique potentials". What is meant by that is the following: the potential functions of the underlying graph must be somehow encoded in the new hypergraph.

There are situations in which the potentials of the underlying graph are given in terms of its maximal cliques. In these cases, the initialization procedure is trivial: simply assign the potential of each maximal clique to the correspondent node of the Junction Tree (since the Junction Tree is made up by nodes representing the maximal cliques). However, this is in general not the case. It is common that the potentials in the underlying graph are given in terms of proper subsets of the maximal cliques. In this situation, the potentials in the nodes of the Junction Tree are given by the product of the potentials in the proper subsets (JORDAN, Forthcoming, 2004).

Consider, for example, the graphical model shown in Figure 4.9. In this figure, it is assumed that the graphical model is parameterized in terms of pairwise potentials. However, the maximal cliques of the model are triples. In this case, the potentials of the triples ABC and BCD would be factorized as

$$\psi_{ABC} = \psi_{AB}\psi_{BC}\psi_{AC} \text{ and } \psi_{BCD} = \psi_{CD}\psi_{BD} \tag{4.28}$$

Note that the pair BC has been arbitrarily assigned to the clique ABC. It could also have been assigned to the clique BCD. What is important is that each pairwise clique is

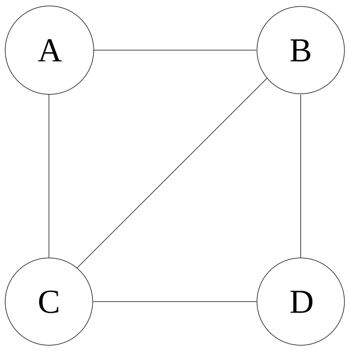


Figure 4.9: A model with maximal clique of size 3 which can be parameterized with pairwise potentials: ψ_{AB}, ψ_{AC}, ψ_{BC}, ψ_{BD} and ψ_{CD}.

inserted just once in the representation of the joint distribution. If the clique potentials are mounted in this way, it is guaranteed that the factorized form

$$p(x) = \frac{\prod_{C \in \mathcal{C}} \psi_C(x_C)}{Z} \tag{4.29}$$

does represent faithfully the joint distribution (JORDAN, Forthcoming, 2004).

We must also initialize the separator potentials. These are initialized to unity (JORDAN, Forthcoming, 2004).

Once the clique potentials have been initialized, the last step of the algorithm provides a systematic way of updating them. This procedure is described in what follows.

4.3.5 Propagation

The final stage of the Hugin algorithm consists in a probability propagation or message-passing phase. This propagation phase can be seen as an algorithm that performs local operations between neighboring clique nodes so as to manipulate the clique potentials by "transforming" them into maximum a posteriori clique distributions (or marginal clique distributions, depending on how the computations are performed, as will be seen). After the algorithm runs, it is guaranteed that every clique potential ψ_C will be proportional to the MAP distribution $MAP(C)$ (or proportional to the marginal distribution $p(C)$).

In order to understand the propagation algorithm, it is important to describe its core procedure, which consists in a computation involving two neighbor clique nodes and their separator (JORDAN, Forthcoming, 2004). Assume the existence of two clique nodes, V and W, with separator node S. The basic operation of the propagation algorithm involves updating the potentials ψ_V, ψ_W and ϕ_S so as to enforce the consistency of both MAPs (marginals) with respect to the separator. This operation can be understood as a transfer of information between V and W, with S being a conduit.

The first step is to update W from V:

$$\phi_S^* = \sum_{V \setminus S} \psi_V \tag{4.30}$$

$$\psi_W^* = \frac{\phi_S^*}{\phi_S} \psi_W. \tag{4.31}$$

Eq. (4.30) is simply a summation in order to obtain the marginal for the separator with respect to the potential ψ_V. If the purpose is to obtain MAPs, not marginals, Eq. (4.30) should be changed to

$$\phi_S^* = \max_{V \setminus S} \psi_V. \tag{4.32}$$

As a result, there is a “marginal version” and a “MAP version” of the algorithm.

Eq. (4.31) is an update in ψ_W so as to retain the same joint distribution. Note that the joint distribution prior to this update is the same after it has been computed:

$$\frac{\psi_V \psi_W}{\phi_S} = \frac{\psi_V \psi_W^* \phi_S}{\phi_S \phi_S^*} = \frac{\psi_V \psi_W^*}{\phi_S^*}. \tag{4.33}$$

Once the “message” has been passed from V to W, it is necessary to send back to V a new message from W, which is the second step in the propagation algorithm:

$$\phi_S^{**} = \sum_{W \setminus S} \psi_W^* \tag{4.34}$$

$$\psi_V^{**} = \frac{\phi_S^{**}}{\phi_S^*} \psi_W^*. \tag{4.35}$$

Again, for calculating MAPs (for running the MAP version instead of the marginal version), Eq. (4.34) should be changed to

$$\phi_S^{**} = \max_{W \setminus S} \psi_W^*. \tag{4.36}$$

Since the only potential function to be updated twice is that of the separator (ϕ_S), the following correspondences are defined: $\psi_V^* \equiv \psi_V^{**}$ and $\psi_W^* \equiv \psi_W^{**}$. Note that, for the same reason as described above, this second step of the propagation algorithm does not alter the joint distribution. Moreover, it is possible to show that once the forward and backward messages have been passed, the marginal of ϕ_S with respect to ψ_V is consistent with the marginal of ϕ_S with respect to ψ_W:

$$\begin{aligned}
\sum_{V \setminus S} \psi_V^{**} &= \sum_{V \setminus S} \frac{\phi_S^{**}}{\phi_S^*} \psi_V^* && (4.37)\\
&= \frac{\phi_S^{**}}{\phi_S^*} \sum_{V \setminus S} \psi_V^* && (4.38)\\
&= \frac{\phi_S^{**}}{\phi_S^*} \phi_S^* && (4.39)\\
&= \phi_S^{**} && (4.40)\\
&= \sum_{W \setminus S} \psi_W^{**}, && (4.41)
\end{aligned}$$

what is also true for the MAPs, where it is obtained that $\max_{V \setminus S} \psi_V^{**} = \max_{W \setminus S} \psi_W^{**}$. This fact is called *local consistency*.

The above message-passing operation is the fundamental element of the propagation algorithm. Real graphs will have in general Junction Trees that have more than two clique nodes. In this case, the message-passing scheme must[1] respect the following protocol:

[1]Actually, even if the protocol is not respected, eventually all messages will be passed and the algorithm will succeed. However, in order to avoid unnecessary operations the protocol should be respected.

- A node V can only send a message to a neighbor W if it has already received messages from all its other neighbors.

This protocol can be implemented using different algorithms, but it is clear from it that the first messages should come from the leafs of the Junction Tree, since they have no other neighbors than the one they should pass the message to. The algorithm is said to be finished when all clique nodes have received messages from all their neighbors.

Note that there is an interesting scenario in which we need to keep multiple copies of separator potentials. This is the case when a given set of nodes is the separator for more than one pair of clique nodes. Consider the example in Figure (4.10).

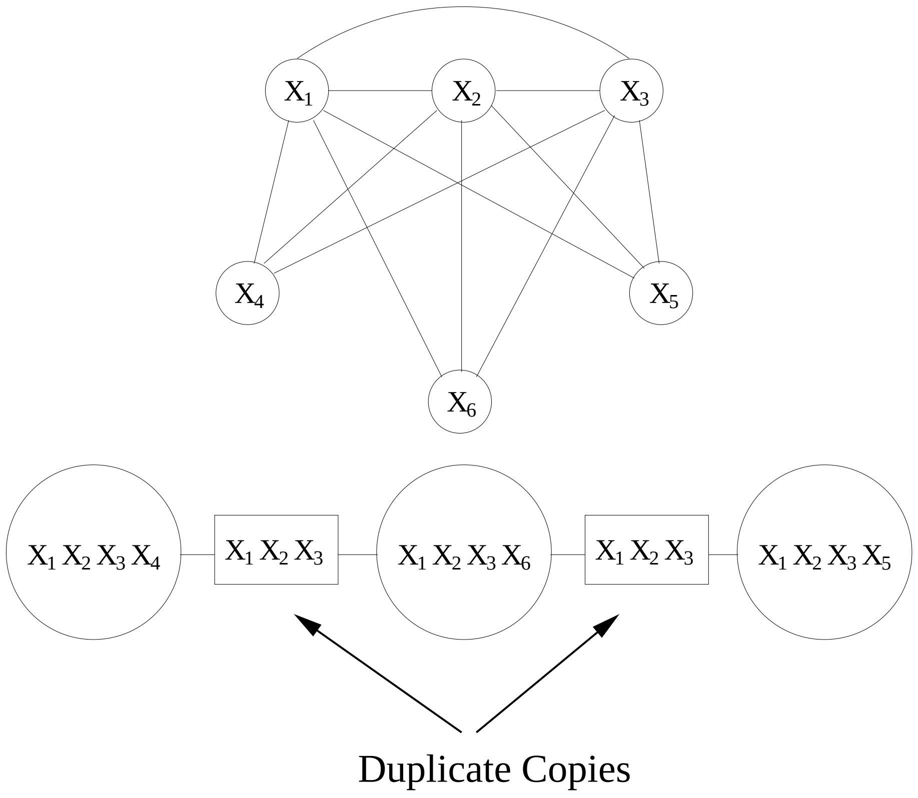


Figure 4.10: A situation in which the same set of nodes ($X_1X_2X_3$) appears as separating different pairs of clique nodes. In this case, different copies of the potentials must be stored. Top: the model. Bottom: a possible Junction Tree.

In this situation, when the potential in one rectangle is updated, the potential in the other rectangle keeps its value until the moment specified by the protocol, when then its value will be updated. As a result, there must be as many copies of a separator potential as is the number of its occurrences in a Junction Tree.

We will encounter in this Thesis a graphical model with this property.

It is not difficult to see that after the algorithm has finished it will be obtained a Junction Tree which is "globally" consistent (in the sense that every separator is consistent with respect to its neighbors), because (a) the local consistency is guaranteed by the updating computations and (b) the running intersection property implies that local consistency, in a Junction Tree, is equivalent to global consistency. It is certainly true that we have a consistent Junction Tree with the same joint distribution than before the algorithm was run. The question that arises at this stage is: have we achieved something useful? The answer comes in the following theorem (JORDAN, Forthcoming, 2004):

Theorem 4.4 *Let $p(x_C)$ and $p(x_S)$ be the marginal probability distributions for a clique node C and a separator node S of a Junction Tree. Let ψ_C and ϕ_S be, respectively, the final potentials for clique node C and separator node S after the marginal version of the Hugin algorithm has finished. For every clique node C and separator node S, it holds that*

$$\psi_C = p(x_C)Z \tag{4.42}$$

$$\phi_S = p(x_S)Z. \tag{4.43}$$

where Z is a constant (which is the same for every clique node). Similarly, if $MAP(x_C)$ and $MAP(x_S)$ are the Maximum a Posteriori distributions for the clique node C and separator node S, after the MAP version of Hugin has finished it holds that

$$\psi_C = MAP(x_C)Z \tag{4.44}$$

$$\phi_S = MAP(x_S)Z \tag{4.45}$$

for every clique node C and separator S.

In other words, the final potentials in each clique and separator are proportional to their marginals (or proportional to their MAPs, if the MAP version is run). The constant of proportionality is the *same* for all clique nodes, though. This is what Junction Tree algorithms (in particular the Hugin instance described here) give us: a way of obtaining the marginal (MAP) distribution for each maximal clique and separator of the underlying graph. If the interest is to compute marginals (MAPs) for subsets of the maximal cliques, it is necessary to perform as post-processing a marginalization (maximization) within those cliques. For example, if after running Hugin we obtain the MAP for the maximal clique ABC but we want the MAP for the singleton A, it is necessary to compute $p(x_A) = \max_{x_B x_C} p(x_A x_B x_C)$, which is a local computation with cost determined by the size of the clique node (in this case the dimension of the clique node is 3).

As a summary, this is the set of procedures for realizing probabilistic inference in arbitrary undirected graphical models: (1) first triangulate the graph and construct a Junction Tree, then (2) initialize the potentials by direct plug-in if they are parameterized with respect to the maximal cliques, or by multiplication of sub-potentials if they are parameterized with respect to proper subsets of the maximal cliques; (3) run the propagation algorithm respecting the protocol and finally (4) perform local marginalization (maximization) if marginals (MAPs) for the subsets of maximal cliques are needed.

It turns out that the computational complexity of this algorithm is dominated by an exponential term on the dimensionality of the maximal clique, as will be shown in what follows. This is an important result, because for models where the maximal cliques have a moderate size exact inference may be feasible.

4.3.6 Computational complexity

As seen above, the Hugin algorithm has a series of steps, each of them having its own computational complexity.

Finding an optimal triangulation is known to be NP-hard (JORDAN, Forthcoming, 2004). However, as was mentioned earlier this is an off-line procedure, and we may accept to pay the price of running an algorithm for a reasonable amount of time based on heuristics in order to find a good triangulation[2]. Moreover, in many cases it happens that the underlying graph is already triangulated, which is precisely the case in the problem studied in this Thesis. As a result, triangulation will not be a concern in the context of this work.

The determination of a Junction Tree from a triangulated graph can be obtained by greedy algorithms such as Kruskal's or Primm's (JORDAN, Forthcoming, 2004). The run time of both these algorithms is $O(N^2)$ - being N the number of nodes in the tree - which is feasible.

Initialization is a straightforward procedure once the parameterized potentials are available and the Junction Tree is constructed. It is clearly linear on the number of nodes, since there is no dependency between the assignment of different potentials. The construction of the potentials, however, may be a problem because the number of entries in the table defining a potential is exponential on the dimensionality of this potential. The complexity of this task depends on the way that the potentials are parameterized. In the problem studied in this work, we will see an instance of the issue of constructing the potentials.

The above phases of the algorithm - triangulation, Junction Tree construction and initialization - are *preprocessing* phases. In usual situations, one is interested in answering probabilistic inference questions about an already given model. As a result, these tasks are usually performed off-line in real problems. The on-line phase of the algorithm is the propagation one. A tree with N nodes has $N - 1$ edges. Two messages are passed through each edge - one in each direction. So the total number of messages to be passed is $2(N - 1)$. Each message has a marginalization (maximization) step and a normalization step, what results in $4(N - 1)$ steps. Both the marginalization (maximization) and the normalization steps involve operating all entries in the tables of the involved potentials. Since the number of entries in a table is exponential on its dimensionality and the dimensionality of a table is the size of the correspondent maximal clique, the number of computations involved in the marginalization (maximization) and normalization steps is exponential on the size of the largest clique involved[3], $O(S^d)$, where S is the number of possible realizations for each random variable and d is the dimensionality of the largest clique in the Junction Tree. As a result, the clique which has maximal size is the one who dominates the computational complexity of the propagation phase. The propagation phase overall complexity is obtained by merging the complexity of local message-passing and that of computing all messages. This results in $O(NS^d)$.

We will see that, in the problem studied in this work, a particular off-line phase is actually performed on-line. This is precisely the phase of construction of the potentials. This fact results in a dual on-line processing: the construction and initialization of potentials and the propagation phase.

The next chapter introduces the original contributions of this work by establishing a formulation for point set matching in terms of inference in graphical models as well as some important theoretical results.

---

[2]Note that by "optimal triangulation" we mean a triangulation that makes the size of the maximal clique minimum. "Good triangulation" would be a triangulation that generates a maximal clique of acceptable size for the given application.

[3]Note that, if a message is to be passed between clique nodes of different dimensionality, the computational complexity will effectively be dominated by the largest clique. This occurs because in order to perform the element-by-element computation it is necessary to replicate the smaller table over the dimension(s) that is (are) present in the larger clique but not in the smaller.

5 THESIS CONTRIBUTIONS: THEORY

This chapter presents three contributions: (1) a principled way of formulating the point pattern matching problem as one of inference in graphical models, (2) the introduction of a graphical model for point set matching which is optimal in the limit case of exact matching and where optimal inference is feasible in polynomial time and (3) a generalization of the formulation for arbitrary attributed graph matching problems. The original publications describing most of these contributions are (CAETANO; CAELLI; BARONE, 2004a,b; CAELLI; CAETANO, 2003, 2004).

We start by formulating point pattern matching as a problem of probabilistic inference over a graphical model, where the purpose consists in finding a Maximum a Posteriori (MAP) estimate. In the following, we present a fundamental result regarding the global rigidity of straight line graph embeddings, which will allow us to prove that, in the limit of exact matching, a particularly simple graphical model is equivalent to the fully connected model for this problem. This result lays the foundation of the present work since exact inference in the resulting graphical model is feasible. It turns out that we obtain a point set matching technique that (1) is based on an optimal probabilistic inference algorithm over a graphical model, such that (2) this graphical model is optimal in the limit of exact matching, in the sense that it is equivalent to the full model and (3) this graphical model is sufficiently simple so that exact inference runs in polynomial time. The final result is a polynomial time procedure for point set matching in arbitrary dimensions invariant to translations, rotations and reflections, which in addition is optimal in the noise free scenario. Finally, we present a straightforward generalization of the formulation for the case of arbitrary attributed graph matching problems.

5.1 Point Pattern Matching as Inference in Graphical Models

The fundamental idea driving the formulation of our model is that the matching problem between two point sets can be seen as a Maximum a Posteriori (MAP) inference problem in a Graphical Model. It was seen in chapter 2 that the point set matching problem can be seen as one of Weighted Graph Matching. The idea is to formulate WGM itself as an inference problem in a Graphical Model. First we present the modeling idea and then the formal development of the Markov Random Field formulation.

5.1.1 The modeling idea

In order to introduce and describe the modeling idea, let us recall the definition of one point pattern as the *domain* pattern and the second point pattern as the *codomain* pattern. This notation is in accordance with the formal notions of domain and codomain

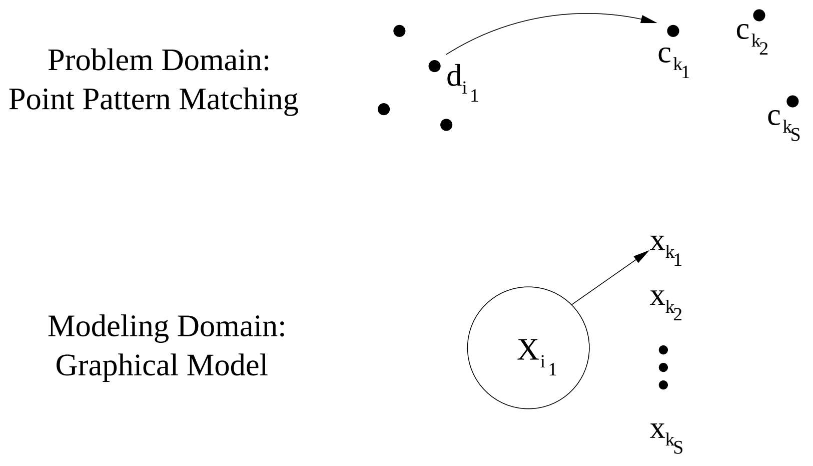


Figure 5.1: Pictorial representation of the modeling idea.

of functions, such that we assume that each point in the domain pattern will necessarily map to one and only one point in the codomain pattern. Now, assume that to each point d_i in the domain pattern a random variable X_i is assigned. Formally, there is an isomorphism between the set of points and a set of random variables, but for simplicity we say that each point in the domain pattern "is" a random variable. This is the first step of the construction of our model: to understand each point in the domain pattern as being a random variable. The second step is quite logical: each random variable has a finite set of possible realizations which coincides with the set of points in the codomain pattern. As a result, a particular realization x_k of a random variable X_i corresponds to a particular map between point d_i in the domain pattern and point c_k in the codomain pattern. Consequently, a particular joint realization $\{x_k, \forall k \in \mathcal{V}_c\}$ of the complete random field $\{X_i, \forall i \in \mathcal{V}_d\}$ corresponds to a particular match between the point sets $\{d_i\}$ and $\{c_k\}$. Figure 5.1 shows a pictorial description of this modeling strategy. In this figure, we illustrate the notion that a particular unary map from an element d_{i_1} in one pattern to an element c_{k_1} in the other pattern is represented, in our modeling scheme, as the fact that the random variable X_{i_1} has realization x_{k_1}.

In this sense a joint realization of the whole set of random variables corresponds to a particular match between the two patterns. The purpose then is to find an appropriate graphical model (i.e. connectivity structure and potential functions) in order to make the likelihood of a particular joint realization of the entire random field representative of the global similarity between the two point patterns.

Note that the formulation in terms of random variables and their realizations naturally poses the problem as one of optimization under the constraint of a total (many-to-one) mapping, as described in chapter 2, section 2.2:

$$\sum_{k_1=1}^{S} M_{i_1 k_1} = 1, \forall i_1 \tag{5.1}$$

since every random variable (an element of the domain pattern) must assume one and only one value (an element of the codomain pattern).

The next section describes formally how we derive the problem from first principles using MRF theory.

5.1.2 Formulation

Here we present a contribution by showing how the weighted graph matching problem, usually posed as the minimization of an heuristic energy function (GOLD; RANGARAJAN, 1996), can be formally derived from first principles using the theory of graphical models. We will make assertions concerning probability distributions defined over random variables indexed by nodes of a graph (JORDAN, Forthcoming, 2004). We follow here the notation used in the previous chapter: X is a set of random variables indexed by the nodes of a graph, x is a particular realization of this set and $\mathcal{X}$ is the set of all possible realizations.

We saw in the last chapter that the HC theorem tells us that the probability distribution of a MRF can be expressed as a product over "clique potential functions". This is important because the problem reduces to simply specifying these potential functions. This result is well-known and is usually presented as the "MRF-Gibbs equivalence", due to the fact that the factorized distribution can be expressed as a Gibbs (or Boltzmann) distribution from Statistical Physics (GEMAN; GEMAN, 1984).

Taking this result into account, we are equipped to derive the weighted graph matching problem. Assume the existence of a graph G_m, where m represents the model. Having chosen a particular graph G_m, the HC Theorem states that any strictly positive probability distribution which is Markov over the nodes of this graph can be expressed as (see the previous chapter)

$$p(x) = \prod_{C \in \mathcal{C}} \psi_C(x_C), \tag{5.2}$$

where C is a maximal clique in G_m, $\mathcal{C}$ is the set of all C's and $\psi_{x_C}(\cdot)$ is the potential function for clique C. Similarly, we will denote a clique of size Ω as C_Ω, and the set of all C_Ω's as $\mathcal{C}_\Omega$. x_C is the joint realization of the set of variables X_C comprising clique C. Now, we define a MRF $(G_m, p(X = x))$ over the graph G_m, such that each random variable X_i (a node in G_m) corresponds to a node of the graph G_d and each possible realization x_k corresponds to a node in the graph G_c. Thus, the joint realization $X = x$ represents a particular match from the domain graph G_d to the codomain graph G_c. Note that there is a conceptual difference between the graph G_m and the graphs G_d and G_c. While G_d and G_c are the full attributed graphs to be matched, G_m is a possibly sparse graph. The fact that the nodes of G_m represent the random variables associated to nodes in G_d implies that G_m and G_d have the same number of nodes. It is possible to see G_m as a subgraph of G_d, since it has the same number of nodes but possibly a sparse edge constitution (while G_d is a fully connected graph).

Being $(G_m, p(x))$ a MRF, the HC Theorem states that $p(x)$ is given by Eq. (5.2), which can be rewritten as a Gibbs distribution:

$$p(x) = \exp\left(-\sum_{C \in \mathcal{C}} V_C(x_C)\right) \tag{5.3}$$

where $V_C = -\log \psi_C(x_C)$. Observe that, although the potentials ψ_C must be non-negative, the modified potentials[1] V_C are arbitrary - apart from being such that $\sum_x p(x) = 1$, since $\log(x)$ maps from the positives to the entire real line. Actually, we can decouple from V_C the part responsible for normalizing $p(x)$ and make it totally arbitrary:

[1]We call both the function over a clique and its negative logarithm as "clique potential function" or simply "potential function". This is for simplicity and will hopefully not harm the clarity of the presentation.

$$p(x) = \frac{\exp\left(-\sum_{C \in \mathcal{C}} V_C(x_C)\right)}{\sum_x \exp\left(-\sum_{C \in \mathcal{C}} V_C(x_C)\right)} \tag{5.4}$$

where the denominator is a normalization constant (called partition function) that renders $\sum_x p(x) = 1$.

Note that we are interested in finding the most likely joint realization of the MRF, given all the V_C's. This is precisely the solution to the assignment problem, since the realization $X = x$ represents a particular match in this formulation. It is obvious that the V_C's are functions of the attributes of G_d and G_c. So, for a given set of V_C's, our purpose is to find x^* that maximizes Eq. (5.4):

$$x^* = arg \max_x p(x|\{V_C\}) \tag{5.5}$$

where $\{V_C\}$ denotes the set of all V_C's. Here, x^* is known as the Maximum a Posteriori (MAP) estimate, because it is the argument that maximizes the posterior distribution $p(x|\{V_C\})$.

In this way, the problem is one of finding a MAP estimate for $p(x)$ given all the potentials V_C, which are themselves functions of the attributes in both graphs.

Due to the fact that the negative exponential is a strictly decreasing function, maximizing Eq. (5.4) is equivalent to minimizing:

$$U(x) = \sum_{C \in \mathcal{C}} V_C(x_C). \tag{5.6}$$

In words, the MAP problem over the MRF is equivalent to one of minimizing an "energy" function $U(x)$ made up of a sum over local clique potentials. This fact will lead us to a definition of the problem in terms of the minimization of a cost function, as it is usually posed (GOLD; RANGARAJAN, 1996). We recall that the local clique potentials V_C's are in principle functions of the maximal cliques of G_m, but can be decomposed into sums of smaller clique potentials. In particular, the standard definition of the weighted graph matching problem can be obtained by decomposing the maximal clique potentials into a sum of pairwise clique potentials:

$$U(x) = \sum_{C_2 \in \mathcal{C}_2} V_{C_2}(x_{C_2}). \tag{5.7}$$

Since a 2-clique is an edge in G_m, these potentials must simply measure the dissimilarity of these attributes for a given pairwise mapping:

$$V_{C_2}(X_{i_1} = x_{k_1}; X_{i_2} = x_{k_2}) = D_2(y^d_{i_1 i_2}, y^c_{k_1 k_2}), \tag{5.8}$$

where $D_2(y^d_{ij}, y^c_{kl})$ is some distance measure between the weights of edges d_{ij} in G_d and c_{kl} in G_c.

What we finally obtain is the following optimization problem:

$$U(M) = \sum_{C_2 \in \mathcal{C}_2} \sum_{k_1=1}^{S} \sum_{k_2=1}^{S} M_{i_1 k_1} M_{i_2 k_2} D_2(y^d_{i_1 i_2}, y^c_{k_1 k_2}) \tag{5.9}$$

subject to

$$\forall i_1, \sum_{k_1=1}^{S} M_{i_1 k_1} = 1 \tag{5.10}$$

where M_{ik} denotes an indicator function with respect to the match $X = x$:

$$M_{i_1 k_1} = \begin{cases} 0, & X_{i_1} \neq x_{k_1} \\ 1, & X_{i_1} = x_{k_1} \end{cases} \tag{5.11}$$

This condition simply states that the only constraint enforced in the mapping is that it is a total function, so that every node in G_d must map to a single node in G_c. The opposite is not necessarily true, because we do not require that $\sum_{i_1} M_{i_1 k_1} = 1$. This is in contrast, for example, with (GOLD; RANGARAJAN, 1996), where both constraints were imposed and so generating an injective function.

We note that Eq. (5.9) involves a summation only over the edges (2-cliques) of G_d that correspond to edges of the model graph G_m. This is indeed an approximation to the complete problem, and lies at the core of our MRF formulation[2]. The complete problem is obtained directly from Eq. (5.9) by considering G_m a full graph, and resembles well-known definitions of the weighted graph matching problem (GOLD; RANGARAJAN, 1996):

$$U(M) = \sum_{i_1=1}^{T} \sum_{i_2=1}^{T} \sum_{k_1=1}^{S} \sum_{k_2=1}^{S} M_{i_1 k_1} M_{i_2 k_2} D_2(y^d_{i_1 i_2}, y^c_{k_1 k_2}) \tag{5.12}$$

under the same constraint (5.10). Note that Eq. (5.12) has the same form of Eq. (2.6), which is the heuristic definition of the WGM problem existent in the literature (GOLD; RANGARAJAN, 1996) . In this way, we notice that the usual formulation of the WGM problem can be derived from first principles using the theory of probabilistic graphical models.

In order to fully specify the model, one needs to define the potential functions and the connectivity structure of the graph. This is done in the next two sections.

5.2 The Model: Potential Functions

Recall that the basic idea of the modeling strategy is to consider a graphical model where the nodes are points in the domain pattern and their possible realizations are points in the codomain pattern. In order to fully specify the graphical model, it is necessary to define (1) the potential functions and (2) the connectivity of the model. We start by formally specifying the model and introducing the potential functions.

We continue with a notation coherent with the descriptions of the problem presented in chapter 2. The cardinalities of the domain and codomain pattern sets are denoted, respectively, by T and S. Each point in the domain is associated with a vertex of a graph G_d, and each point in the codomain is associated with a vertex of a graph G_c. The relative distance between a pair $\{i, j\}$ of points in a pattern is seen as an *edge weight* of the edge that connects vertices i and j in the respective graph. In this formulation, point pattern matching turns out to be a weighted graph matching problem, as seen in chapter 2.

[2]Although this is in principle an approximation, we will show that there is a particular graph G_m that has the same representational power than the full graph in the limit of exact matching.

The model formulation consists, initially, in defining each of the T vertices in G_d as a random variable that can assume S possible values (discrete states), corresponding to the vertices in G_c. Note that in this formulation the solution to the problem (the best match) corresponds to finding the most likely (the best) realization of the set of rv's.

According to the last section, one needs to define the distance measures $D_2(y^d_{i_1 i_2}, y^c_{k_1 k_2})$ that represent the potential functions of our graphical model (the potential functions are the negative exponentials of the distance measures, see eq. (5.3)). Recall that the fundamental feature of this measure is that is must penalize more severely pairwise mappings for which the difference of the lengths of mapped edges ($y^d_{i_1 i_2}$ and $y^c_{k_1 k_2}$) is higher. Figure (5.2) illustrates a pairwise map and a possible measure which is relevant in order to construct the potential functions ($|y^d_{i_1 i_2} - y^c_{k_1 k_2}|$).

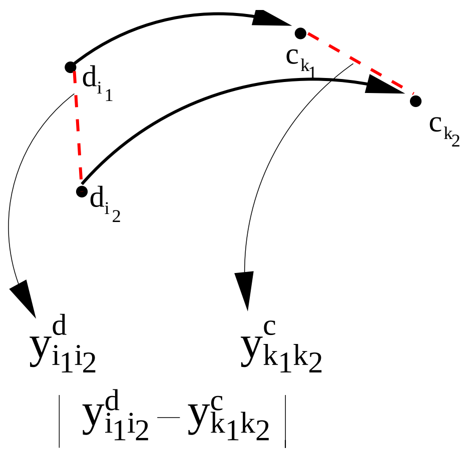


Figure 5.2: An example of a pairwise mapping. An appropriate potential function should penalize more severely mappings for which $|y^d_{i_1 i_2} - y^c_{k_1 k_2}|$ is higher.

Since each node in the domain graph can map to S different nodes in the codomain graph, each pair of nodes can map to S^2 different pairs in the codomain graph. Figure (5.3) illustrates the kernel structure of our model: a pairwise clique, where each random variable represents a point in the domain graph which in turn can assume a set of S possible realizations (which themselves correspond to points in the codomain graph).

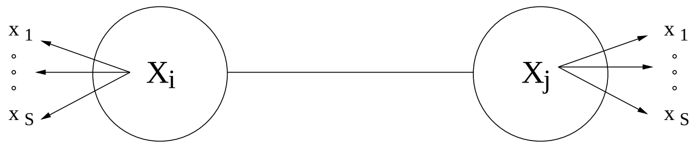


Figure 5.3: The kernel structure of the graphical model.

The sample space for this clique has S^2 elements, corresponding to all possible combinations that a pair of points in the domain graph can map to in the codomain graph. A potential function - recall from chapter (4) - is a function that associates to each element of the sample space a positive real number. In our case, the only requirement that the potential function must obey is that its value must be as higher as more similar are the distances of the mapped edges, as illustrated in Figure (5.2).

Formally, we can specify the potential function by

$$\psi_{ij;kl} = p(X_i = x_k | X_j = x_l) \tag{5.13}$$

or, in matrix form, for each pair $\{X_i, X_j\}$ in G_d, we define

$$\psi_{ij} = \psi_{ij}(X_i, X_j) = \frac{1}{Z}\begin{pmatrix} \mathcal{S}(y^d_{ij}, y^c_{11}) & \dots & \mathcal{S}(y^d_{ij}, y^c_{1S}) \\ \vdots & \ddots & \vdots \\ \mathcal{S}(y^d_{ij}, y^c_{S1}) & \dots & \mathcal{S}(y^d_{ij}, y^c_{SS}) \end{pmatrix} \tag{5.14}$$

where y^a_{bc} denotes the edge weight between vertices with indexes b and c in graph G_a. Z is a normalization constant that equals the sum of all elements in the matrix, in order to keep ψ_{ij} compatible with a probability distribution (it is known as partition function in Physics). $\mathcal{S}$ is a similarity function that measures the compatibility of the two arguments. Several options are available for $\mathcal{S}$ (CARCASSONI; HANCOCK, 2003), such as the Gaussian,

$$\mathbb{G}(y^d_{ij}, y^c_{kl}) = \exp\left(-\frac{1}{2\sigma^2}|y^d_{ij} - y^c_{kl}|^2\right) \tag{5.15}$$

the Hyperbolic Tangent,

$$\mathbb{H}(y^d_{ij}, y^c_{kl}) = 1 - \tanh\left[\frac{|y^d_{ij} - y^c_{kl}|}{\sigma}\right] \tag{5.16}$$

or an Increasing Weighting function,

$$\mathbb{I}(y^d_{ij}, y^c_{kl}) = \left[1 + \frac{|y^d_{ij} - y^c_{kl}|}{\sigma}\right]^{-1} . \tag{5.17}$$

These proximity measures are needed in order to model the uncertainty due to the presence of noise. Obviously, their maximal value must be reached when there is no noise ($y^d_{ij} = y^c_{kl}$). In the particular case of exact matching, one may use simply the indicator function,

$$\mathbb{J}(y^d_{ij}, y^c_{kl}) = \mathbf{1}(y^d_{ij}, y^c_{kl}), \tag{5.18}$$

which is defined as

$$\mathbf{1}(y^d_{ij}, y^c_{kl}) = \begin{cases} 1, \text{ if } & y^d_{ij} = y^c_{kl} \\ 0, \text{ if } & y^d_{ij} \neq y^c_{kl} \end{cases} \tag{5.19}$$

As a result we can now define a Markov Random Field (MRF) graphical model (i.e. an *undirected* graphical model) over the model graph G_m (as described in chapter 2). The nodes in the model correspond to the vertices in G_d whose states are defined by the set of vertices in G_c, and joint realizations of neighboring random variables have an associated potential.

Having specified the potential functions, it remains to be determined the connectivity of the graphical model: which nodes will be neighbors in the model? Next section shows theoretical results that lead us to an answer to this question.

5.3 The Model: Connectivity

In this section we present a contribution by introducing some theoretical results that will ultimately lead us to a proper *connectivity* structure for the graphical model to be considered.

5.3.1 A relevant lemma

To start our considerations, we now formulate a relevant lemma which turns out to be essential in order to obtain the upcoming results:

Lemma 5.1 *Let $S_1, S_2, \dots, S_{n+1}$ 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 null set.*

Proof. Induction over n.

Recall that a sphere in a vector space is the set of points equidistant to a fixed point in that vector space.

The Lemma obviously holds for the trivial case when $n = 1$. See Figure (5.4).

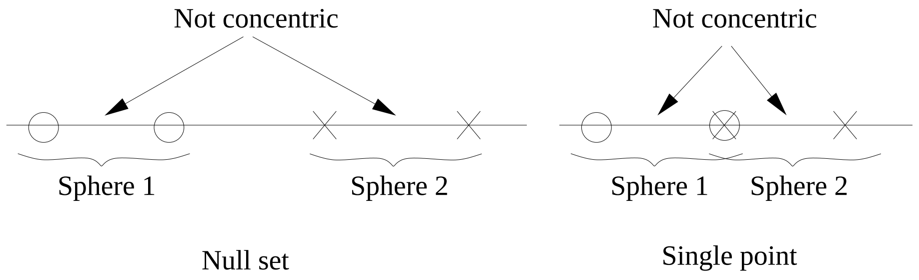


Figure 5.4: $n = 1$: 2 spheres in $\mathbb{R}^1$ whose centers do not lie in a 0-dimensional vector space (a point); Left: null intersection. Right: intersection is a single point.

Let $S_1 \cap S_2 = I_1$. Then I_1 is a $(n-2)$-sphere lying in a $(n-1)$ vector subspace Q (we use the convention of topology, which denotes a n-sphere as a sphere whose surface is n-dimensional). Let $I_i = S_{i+1} \cap Q$ for $i = 2, 3, \dots, n$. Then $I_1, I_2, \dots, I_n$ are n spheres in $Q \cong \mathbb{R}^{n-1}$ and, obviously, $\cap_{i=1}^{n+1} S_i = \cap_{j=1}^{n} I_j$. We can now apply the induction hypothesis to the set of spheres $I_1, I_2, \dots, I_n$. In order to prove it, it is necessary to show that the centers of I_j do not lie in a $(n-2)$ vector subspace.

Let $(x_1, x_2, \dots, x_n)$ be the coordinates of $\mathbb{R}^n$. Let $(a_{i1}, a_{i2}, \dots, a_{in})$ be the center of S_i. It is easy to see that the centers of S_i are in general position if and only if the matrix

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} & 1 \\ a_{21} & a_{22} & \dots & a_{2n} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n+1,1} & a_{n+1,2} & \dots & a_{n+1,n} & 1 \end{bmatrix} \tag{5.20}$$

is invertible, i.e. has maximal rank.

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 $(x_2, x_3, \dots, x_n)$. The center of I_{j-1} has coordinate $(a_{j2}, a_{j3}, \dots, a_{jn})$. If we subtract the second row from the first row of matrix 5.20 we obtain:

$$\begin{bmatrix} a_{11} - a_{21} & 0 & \dots & 0 & 0 \\ a_{21} & a_{22} & \dots & a_{2n} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n+1,1} & a_{n+1,2} & \dots & a_{n+1,n} & 1 \end{bmatrix} \tag{5.21}$$

Note that $Q = \{x_1 = 0\}$ implies that $(a_{12}, a_{13}, \dots, a_{1n}) = (a_{22}, a_{23}, \dots, a_{2n})$. If we eliminate the first row and column of matrix (5.21), we obtain

$$\begin{bmatrix} a_{22} & a_{23} & \dots & a_{2n} & 1 \\ a_{32} & a_{33} & \dots & a_{3n} & 1 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ a_{n+1,2} & a_{n+1,3} & \dots & a_{n+1,n} & 1 \end{bmatrix} \tag{5.22}$$

It is evident that matrix 5.21 is invertible if and only if $a_{11} \neq a_{21}$ and matrix 5.22 is invertible. This implies that the centers of I_j do not lie in a $(n-2)$ vector subspace in $Q \cong \mathbb{R}^{n-1}$, what completes the proof. □

Note that this result assures that there is a unique point with fixed distances from k fixed points in $\mathbb{R}^{k-1}$ (the intersection point of the spheres). Actually there is a widely known application of a particular case of this result when $k = 4$ (in $\mathbb{R}^3$), which is based on the "Global Positioning System (GPS) principle": if the distances from a point to 4 known points in $\mathbb{R}^3$ is available, then the position of this point is known (the four points should not lie in a plane).

Figure (5.5) illustrates the particular case of this result in $\mathbb{R}^2$.

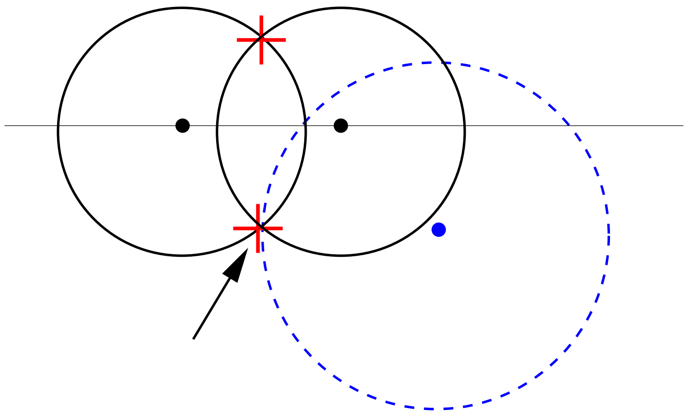


Figure 5.5: An illustration of Lemma 5.1 in $\mathbb{R}^2$.

In Figure (5.5), if the center of the dashed circle does not lie in the same straight line defined by the centers of the other two circles, it can intersect the intersection set of the other circles in *at most* one point (the one pointed by the arrow).

5.3.2 Global rigidity: basic definitions

The Lemma introduced above will enable us to obtain an important result regarding the redundancy of pairwise distances in a point set. Here we present some basic definitions of the global rigidity of graphs (GRAVER; SERVATIUS; SERVATIUS, 1993; CONNELLY, 1982, 2003), which will lay the basic concepts needed in order to introduce this result. A *configuration* is a finite set of n labeled points, $\mathbf{p} = (p_1, \cdots, p_n)$, such that each $p_i \in \mathbb{R}^d$. A *framework* in $\mathbb{R}^d$ consists of a straight line embedding of a graph G with n vertices with configuration $\mathbf{p} = (p_1, \cdots, p_n)$, and is denoted by $G(\mathbf{p})$. In this representation the lengths of the edges correspond to the Euclidean distances between the corresponding vertices. A configuration in *general position* (also called a *general configuration*) in $\mathbb{R}^d$ is such that no $(d+1)$ points lie in a $(d$-$1)$-dimensional vector subspace. In $\mathbb{R}^2$, this means that no 3 points are collinear.

Two frameworks $G(\mathbf{p})$ and $G(\mathbf{q})$ are said to be *equivalent*, and are denoted by $G(\mathbf{p}) \equiv G(\mathbf{q})$, if when $\{i, j\}$ is an edge of G, then $||p_i - p_j|| = ||q_i - q_j||$, where $||.||$ is the Euclidean norm. It is said that a configuration $\mathbf{p} = (p_1, \cdots, p_n)$ is *congruent* to $\mathbf{q} = (q_1, \cdots, q_n)$, and are denoted by $\mathbf{p} \equiv \mathbf{q}$, if, for all $\{i, j\} \in \{1, \cdots, n\}$, $\|p_i - p_j\| = \|q_i - q_j\|$. This is equivalent to saying that congruent configurations are those related by an *isometry*, or a transformation that preserves distances. A framework $G(\mathbf{p})$ is called *globally rigid* in $\mathbb{R}^d$ if $G(\mathbf{p}) \equiv G(\mathbf{q})$ implies $\mathbf{p} \equiv \mathbf{q}$. In other words, a framework is globally rigid when the specification of the edge lengths uniquely specifies the remaining pairwise distances between vertices that are not joined by an edge. Note that under this definition a framework with a complete set of edges is necessarily globally rigid. In the following we present a key fact about the global rigidity of a special kind of framework, which turns out to allow for the development of an effective technique for point set matching.

5.3.3 Global rigidity of k-trees

In order to present the basic result about the global rigidity of a special kind of framework - a k-tree -, we start by reviewing some basic definitions from graph theory (WEST, 2001). In what follows a complete graph with n vertices is denoted as K_n. We recall that a *framework* is a straight line embedding of a graph.

Definition 5.1 (k-clique) *A* k-clique *of a graph is a complete subgraph with k vertices.*

Definition 5.2 (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 adjacent to each vertex of a k-clique in the older graph and nonadjacent to the remaining vertices. The k-cliques adjacent to the new vertices are called* base k-cliques.

Figure 5.6 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 new vertices by connecting them to 3 vertices of any existing base 3-clique. Note that all intermediate graphs generated in this way are themselves legitimate 3-trees.

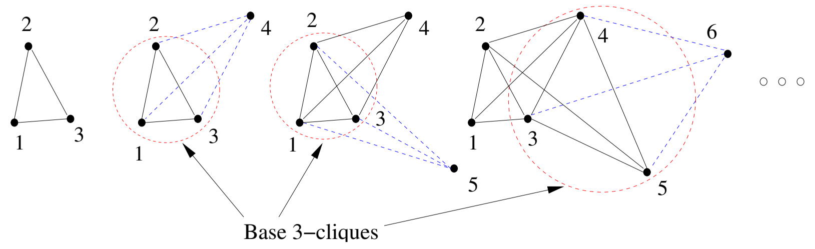


Figure 5.6: The process of constructing 3-trees.

We are now equipped to prove the second result:

Proposition 5.1 *Any k-tree framework having each of its base k-cliques in general position in $\mathbb{R}^{k-1}$ is globally rigid in $\mathbb{R}^{k-1}$.*

Proof. 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 a fully connected graph and as a result has an associated framework which is globally rigid. Now let us assume that the result is true for some $n > k$ and check if under this assumption the result is also true for $n + 1$. First, choose a fixed (but arbitrary) coordinate system S. If it is true for some $n > k$, then all the points in the framework are determined in S.

Now let us include a new vertex such that we know its distances from all the k vertices of any existent base k-clique in general position of the previous framework. 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 5.1. 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. Since the pairwise distances do not depend on any particular choice of S, the result is valid for any coordinate system, what confirms that the initial choice of S is indeed arbitrary. This guarantees that the new framework is globally rigid in $\mathbb{R}^{k-1}$, what completes the proof. □

The direct implication of the result is that the k-tree framework has *exactly* the same informational content than a fully connected framework (since the absent edges have uniquely determined lengths, given the present edges). We can now exploit this result in order to derive the fundamental result of this section.

5.3.4 The main result

In order to introduce our main result, we recall for convenience some already defined notation and also introduce some new terminology.

$G_d = (\mathcal{V}_d, \mathcal{E}_d)$ is the *domain graph* and $G_c = (\mathcal{V}_c, \mathcal{E}_c)$ is the *codomain graph*, which are both complete graphs (the edge sets are complete) embedded in $\mathbb{R}^{k-1}$, for some integer $k > 1$. They are "straight line embeddings", in the sense that their edges are the straight line segments that join the corresponding points in the vector space. The name for these straight line embeddings of graphs are *frameworks*, but we will for simplicity of terminology use "framework" and "graph" interchangeably, and always meaning by that a straight line embedding of a graph in a vector space.

The vertex in the codomain graph to which the vertex d_i in the domain graph maps is $c_{f(i)}$, where f is a "mapping function" such that $M_{if(i)} = 1, \forall i$, where M is defined as in eq. (2.3). Let us also define as $G_d^{kt} = (\mathcal{V}_d, \mathcal{E}_d^{kt})$ a model graph with the same nodes than G_d but with an edge connectivity given by a k-tree whose base k-cliques are in general position in $\mathbb{R}^{k-1}$. $G_c^{kt} = (\mathcal{V}_c^{kt}, \mathcal{E}_c^{kt})$ is the subgraph of G_c whose nodes are those to which the nodes $\mathcal{V}_d$ map under an optimal map f and whose edges have the same length as those in $\mathcal{E}_d^{kt}$. In other words, G_c^{kt} is the subgraph of G_c which is isomorphic and isometric (mapped edges under f have the same length) to the domain graph G_d, and the map f is said to be the optimal map between G_d^{kt} and G_c^{kt}. 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}. Figure (5.7) shows a pictorial description of an optimal match f, where the aforementioned graphs are depicted.

According to the above notation and definitions, the complete optimization problem of eq. (5.12) can be rewritten as one of minimizing with respect to f the "total" cost function

$$U_T(f) = \sum_{i=1}^{T} \sum_{j=1}^{T} D(y_{ij}^d, y_{f(i)f(j)}^c), \tag{5.23}$$

where T is the cardinality of $\mathcal{V}_d$, y_{ij}^d is the Euclidean distance between vertices d_i and d_j in the domain graph and $y_{f(i)f(j)}^c$ is the Euclidean distance between vertices $c_{f(i)}$ and $c_{f(j)}$ in

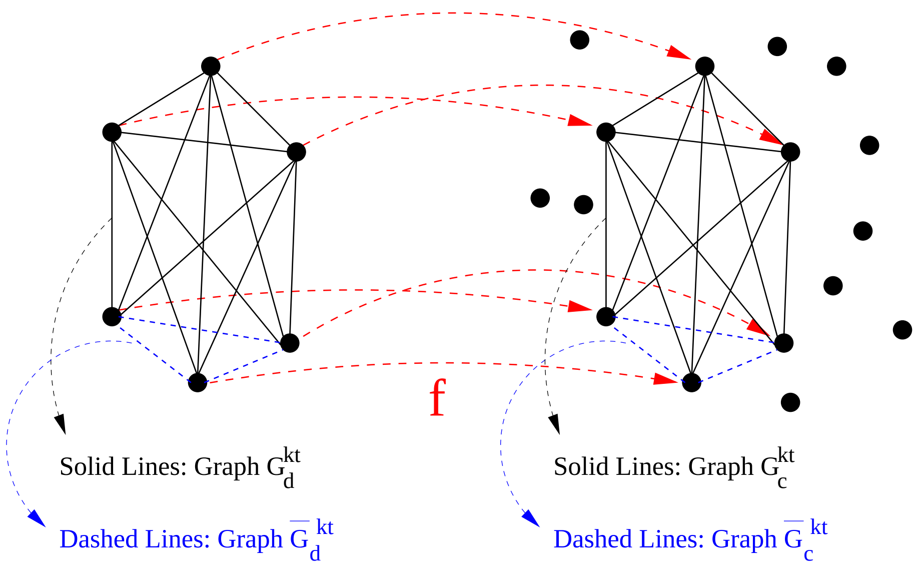

f
Solid Lines: Graph G_d^{kt}
Dashed Lines: Graph $\bar{G}_d^{kt}$
Solid Lines: Graph G_c^{kt}
Dashed Lines: Graph $\bar{G}_c^{kt}$

Figure 5.7: Left: the domain pattern with the graphs G_d^{kt} and $\bar{G}_d^{kt}$ depicted; Right: the codomain pattern with the graphs G_c^{kt} and $\bar{G}_c^{kt}$ depicted. An optimal map f is shown.

the codomain graph. $D(\cdot)$ is simply some distance measure that satisfies $D(a,b) \geq 0, \forall a,b$ and $D(a,b) = 0$ iff $a = b$.

The optimization problem over a k-tree graph G_d^{kt} can be redefined from eq. (5.9) as one of minimizing with respect to f the cost function

$$U_{G_d^{kt}}(f) = \sum_{i,j|d_{ij}\in\mathcal{E}_d^{kt}} D(y_{ij}^d, y_{f(i)f(j)}^c), \tag{5.24}$$

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

We now have all the background to introduce our main result.

Theorem 5.1 *In the exact matching case, a mapping function f which minimizes $U_{G_d^{kt}}(f)$ also minimizes $U_T(f)$.*

Proof: If we define a cost function over the graph which is the complement of G_d^{kt} ($\bar{G}_d^{kt}$):

$$U_{\bar{G}_d^{kt}}(f) = \sum_{i,j|d_{ij}\in\bar{\mathcal{E}}_d^{kt}} D(y_{ij}^d, y_{f(i)f(j)}^c), \tag{5.25}$$

we naturally have that

$$U_T(f) = U_{G_d^{kt}}(f) + U_{\bar{G}_d^{kt}}(f). \tag{5.26}$$

In the noiseless case, the dissimilarity function D associated to a particular match is described simply in terms of an indicator function $\mathbf{1}(.)$:

$$D(y_{ij}^d, y_{f(i)f(j)}^c) = 1 - \mathbf{1}(y_{ij}^d, y_{f(i)f(j)}^c) = \begin{cases} 0, & \text{if } y_{ij}^d = y_{f(i)f(j)}^c \\ 1, & \text{if } y_{ij}^d \neq y_{f(i)f(j)}^c \end{cases} \tag{5.27}$$

The optimal matching function f is such that $U_T(f) = 0$. Obviously, from eq. (5.26) 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 non-negative because $D(.)$ is non-negative (eqs. (5.24) and (5.25)). Our purpose is to prove the converse, i.e. that $U_{G_d^{kt}}(f) = 0 \Rightarrow U_T(f) = 0$. According to eq. (5.26), in order to do that it suffices to prove that $U_{G_d^{kt}}(f) = 0 \Rightarrow U_{\bar{G}_d^{kt}}(f) = 0$.

We use here Proposition 5.1, which means, in words, that if the distances corresponding to the edges of a k-tree framework whose base k-cliques are 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

$$\{y_{ij}^d = const_{ij}^d, \forall i, j | d_{ij} \in \mathcal{E}_d^{kt}\} \Rightarrow \{y_{ij}^d = const_{ij}^d, \forall i, j | d_{ij} \in \bar{\mathcal{E}}_d^{kt}\} \tag{5.28}$$

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 eq. (5.24) must be zero, since they are non-negative:

$$D(y_{ij}^d, y_{f(i)f(j)}^c) = 0, \forall i, j | d_{ij} \in \mathcal{E}_d^{kt}. \tag{5.29}$$

However, from the definition of $D(.)$ for exact matching (eq. (5.27)), this means that

$$y_{ij}^d = y_{f(i)f(j)}^c, \forall i, j | d_{ij} \in \mathcal{E}_d^{kt}. \tag{5.30}$$

Notice that the proposition (eq. (5.28)) holds for any k-tree whose base k-cliques are in general position, in particular for G_c^{kt} (G_c^{kt} also has its base k-cliques in general position in $\mathbb{R}^{k-1}$ because it is isometric to G_d^{kt}, which by assumption has its base k-cliques in general position and so is globally rigid):

$$\begin{aligned} &\{y_{f(i)f(j)}^c = const_{f(i)f(j)}^c, \forall i, j | d_{f(i)f(j)} \in \mathcal{E}_c^{kt}\} \Rightarrow \\ &\Rightarrow \{y_{f(i)f(j)}^c = const_{f(i)f(j)}^c, \forall i, j | c_{f(i)f(j)} \in \bar{\mathcal{E}}_c^{kt}\} \end{aligned} \tag{5.31}$$

Notice that eq. (5.30) implies that the left hand side of implications (5.28) and (5.31) are equivalent. As a result, their right hand side is equivalent and the following holds:

$$y_{ij}^d = y_{f(i)f(j)}^c, \forall i, j | d_{ij} \in \bar{\mathcal{E}}_d^{kt}, \tag{5.32}$$

that, when substituted in eq. (5.25), yields

$$U_{\bar{G}_d^{kt}}(f) = 0, \tag{5.33}$$

that was what we wanted to prove. □

In the next section we show how, by taking advantage of this result, we can formulate a MRF model that in the limit of exact matching is equivalent to the full model and where exact probabilistic inference is feasible in polynomial time.

5.4 The Complete Graphical Model

In this section we show how the results obtained in the previous section can be exploited in order to create a suitable connectivity structure for the graphical model.

5.4.1 The model

The MAP solution found by the Hugin algorithm over a particular sparse MRF model G_m, we recall eq. (5.9), is such that minimizes

$$U(M) = \sum_{C_2 \in \mathcal{C}_2} \sum_{k_1=1}^{S} \sum_{k_2=1}^{S} M_{i_1k_1} M_{i_2k_2} D_2(y^d_{i_1i_2}, y^c_{k_1k_2}) \tag{5.34}$$

where $\mathcal{C}_2$ is the set of all edges in G_m and C_2 is a particular edge. Equivalently, according to the notation of the mapping function f, it is the one that minimizes

$$U_{G_m}(f) = \sum_{i,j|d_{ij}\in\mathcal{E}_m} D(y^d_{ij}, y^c_{f(i)f(j)}), \tag{5.35}$$

where $\mathcal{E}_m$ is the edge set of G_m. The results of the previous subsection imply that, in the limit of exact matching, if G_m is a k-tree with its base k-cliques in general position in $\mathbb{R}^{k-1}$ (and we recall that it is denoted by G^{kt}_d), then the argument f that minimizes eq. (5.35) *also* minimizes the cost function associated to a *complete model*:

$$U_T(f) = \sum_{i=1}^{T} \sum_{j=1}^{T} D(y^d_{ij}, y^c_{f(i)f(j)}). \tag{5.36}$$

As a result, if our purpose is to minimize $U_T(f)$, it suffices to minimize $U_{G^{kt}_d}(f)$, and a sparse graphical model given by a k-tree topology can be used. The MAP estimate obtained (the best mapping function f) will be then an estimate that is optimal for both the k-tree model and the complete model.

Figure 5.8 shows an example of a particular 3-tree graphical model. The results of subsection 5.3.3 imply that, in the limit of exact matching, this model is equivalent to a complete model for matching tasks in $\mathbb{R}^2$, where our experiments will take place. Each connection represents the interaction between the corresponding random variables, which is given by the associated potential function ψ_{ij}.

It is important to emphasize that any model that has the topology of a 3-tree *and* has all the base k-cliques in general position will be equivalent to the model in Figure 5.8 (in the limit of exact matching). However, the particular model drawn in Figure (5.8) has a very attractive property: it has a *single* base k-clique given by nodes $X_1X_2X_3$. This means that, in order to meet the requirements for an optimal model as explained in the previous subsection, only the points corresponding to variables X_1, X_2 and X_3 will be required not to lie in a 1-dimensional vector space (a straight line). In other words, for the construction of models for matching tasks in $\mathbb{R}^2$, it suffices to find 3 non-collinear points and define them as being X_1, X_2 and X_3, and the algorithm will run properly irrespectively of the position of the remaining points.

The model shown in Figure (5.8) has a maximal clique of fixed size: 4. In general, note that k-tree structured models will have maximal cliques of size fixed in $k + 1$ and will be used for matching tasks in $\mathbb{R}^{k-1}$. Since the maximal cliques for a fixed dimension $d = k-1$ have a fixed size equal to $k + 1$, the exponential complexity of the inference algorithm (as seen in chapter 4) will be fixed and the complexity on the number of elements T and S in the graphs will be polynomial.

In what follows we describe the inference procedure in the k-tree graphical model.

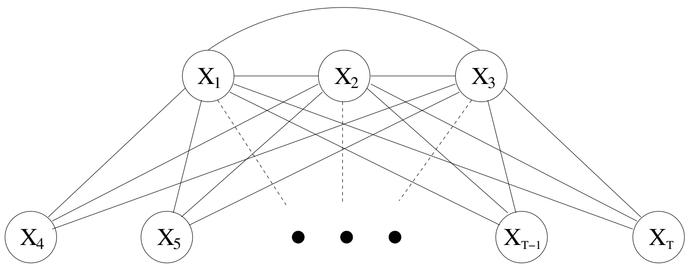


Figure 5.8: A possible k-tree model for $k = 3$.

5.4.2 Optimization

Inference in MRFs typically capitalizes on the Gibbs distribution to employ simulated annealing or Monte Carlo methods to derive the assignment (GEMAN; GEMAN, 1984). However, these inexact inference procedures are needed only in models where exact inference is infeasible (models where the underlying triangulated graph has a maximal clique size that grows with the problem size, as image models for example). The key fact about our model is that it both (1) has a fixed and small maximal clique size and (2) has, in the limit of exact matching, an encoding capacity equivalent to that of the full model, which has maximal clique size equal to the graph size. As a result, we may be able to use effectively the exact inference methods described in chapter 4.

Following the material presented in chapter 4, we can design a Junction Tree for our model and run an optimal inference algorithm in this Junction Tree. Recall that the Junction Tree framework encompasses a set of deterministic algorithms for *exact* inference in arbitrary graphical models (JORDAN, Forthcoming, 2004; LAURITZEN, 1996), and is defined as a graph where the nodes correspond to the maximal cliques of the original graph such that the *running intersection property* is satisfied. This property states that all the nodes in the path between any two nodes in the Junction Tree must contain the intersection of these two nodes. It is known that the condition for the existence of a Junction Tree is that the graph must be chordal or triangulated (LAURITZEN, 1996). A k-tree is a chordal graph, and this allows us to use the Junction Tree framework to perform optimization over the model.

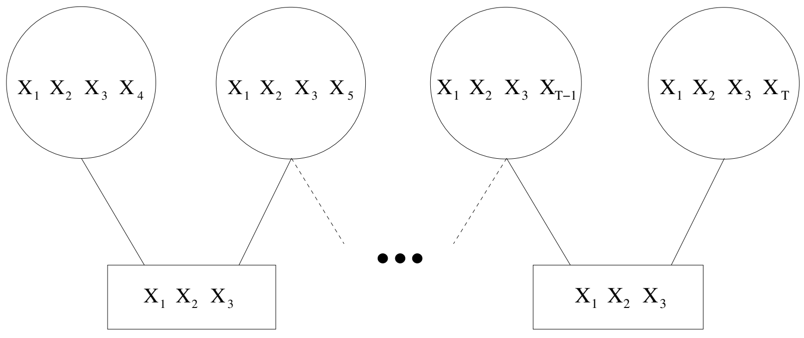


Figure 5.9: The Junction Tree obtained from the model in Figure 5.8.

Figure 5.9 shows a Junction Tree obtained from the model in Figure 5.8. The nodes of the Junction Tree are denoted by circles in which are listed the nodes of the original graph that correspond to the respective maximal cliques. The rectangles are the so-called *separators*, that contain the intersection of the nodes to which they are linked. Both the nodes and the separators are endowed with “clique potentials”, and the optimization process consists in updating these potentials, as explained in chapter 4. Note that the Junction Tree in Figure (5.9) has multiple instances of the triple $X_1X_2X_3$ as separator nodes. Recall from chapter 4 that, in this case, we must keep several copies of the potentials, one for each separator, so as to guarantee that the updating of a particular instance will not affect the content of another instance.

We can apply the Hugin algorithm, introduced in chapter 4, to accomplish exact inference in the k-tree model (in particular in the 3-tree model shown in Figure 5.8). From the analysis of the computational complexity in chapter 4, we conclude that the complexity of Hugin in our k-tree model is $O(S^{k+1}T)$ (or equivalently $O(S^{d+2}T)$, where d is the dimension of the Euclidean Space). As a result, the complexity on S and T is polynomial.

Our model is parameterized with pairwise clique potentials determined by similarity functions between distances of the two point patterns (see the previous subsection). The number of edges in a k-tree - which is the number of pairwise potential functions - with T vertices can be easily shown to be $k[T-(k+1)/2]$. Each pairwise potential function has S^2 elements, being S the number of points in the codomain pattern. Assuming $k << T$, which is always the case in problems of interest, the computational complexity of assembling all the pairwise potential functions is $O(TS^2)$. As a result, the computational complexity of the propagation phase is still the bottleneck of the overall algorithm.

Once the pairwise potential functions are available, we must initialize the potentials of the maximal cliques in the Junction Tree. The initialization of the clique potentials involves a subtlety in our case. Note that the potentials in our model are *pairwise* potentials, while the clique nodes of the corresponding Junction Tree have four elements, meaning that 4D potentials would be required instead of pairwise potentials. This is indeed a common situation in many modeling scenarios, where the available parameterization of an underlying undirected graph is restricted to cliques that are proper subsets of its maximal cliques. This issue was discussed in detail in chapter 4.

The final result in our model is that the clique potentials of a clique node are assembled as an element-by-element product of the pairwise potentials (see Eq. (5.14)) in the respective clique node. For example, $\psi(x_i, x_j, x_k, x_l) = \psi(x_i, x_j)\psi(x_i, x_k)\psi(x_i, x_l)$ for the 3-tree model, assuming that $\psi(x_j, x_k)$, $\psi(x_j, x_l)$ and $\psi(x_k, x_l)$ will belong to the "next" maximal clique.

Once the initialization process is finished, the second step of the Hugin algorithm starts. It is the message-passing scheme, which involves a transfer of information between two nodes V and W (JORDAN, Forthcoming, 2004). In our problem, we are interested in deriving the optimal assignment of the MRF, so we apply the maximization version of Hugin:

$$\phi_S^* = \max_{V \setminus S} \psi_V \tag{5.37}$$

$$\psi_W^* = \frac{\phi_S^*}{\phi_S} \psi_W. \tag{5.38}$$

The above potential update rules must respect the following protocol: a node V can only send a message to a node W when it has already received messages from all its other neighbors. If this protocol is respected and the equations are applied until all clique nodes have been updated, the algorithm assures that the resulting potential in each node and separator of the Junction Tree is equal to the maximum a posteriori probability distribution of the set of enclosed singleton nodes (JORDAN, Forthcoming, 2004). In our particular case, we need the maximum probability for each singleton, what can be obtained by maximizing out the remaining k singletons in each of the nodes. The indexes for which the final potentials are maximum are considered the vertices in G_c to which the corresponding vertices in G_d must be assigned.

5.5 General Matching with Graphical Models

Here we show that the theory of graphical models allows for the formulation of a more general form of matching problem: the *attributed* graph matching problem, described in chapter 2. Actually, we even show that the formalism of graphical models reveals that the attributed graph matching problem, as it is usually understood, is only a particular instance of an even more general matching problem, which we denote here as the *generalized* attributed graph matching (GAGM) problem. The notation used here follows that introduced in section 5.1.

Although these more general problems can be nicely modeled as inference in graphical models, exact inference may not be feasible in these cases because the inter-relations may not present redundancy and conditional independence assumptions cannot be postulated without introducing errors. The k-tree model is an optimal model for matching tasks in Euclidean Spaces, but, in the most general formulation of AGM, the binary attributes in the graph are arbitrary and may not present the redundancy that is present in the set of relative distances in the Euclidean Space.

5.5.1 Attributed graph matching

Recall from chapter 2 that the AGM problem involves measuring unary and binary compatibilities between the graphs. In a development similar to that shown for the WGM problem, we can formulate the AGM problem as inference in a graphical model by decomposing the modified factorized expression over the maximal cliques (eq. 5.6):

$$U(x) = \sum_{C \in \mathcal{C}} V_C(x_C) \tag{5.39}$$

into sums of unary and pairwise clique potentials:

$$U(x) = \sum_{C_1 \in \mathcal{C}_1} V_{C_1}(x_{C_1}) + \sum_{C_2 \in \mathcal{C}_2} V_{C_2}(x_{C_2}) \tag{5.40}$$

Since a 1-clique is a node and a 2-clique is an edge in G_m, these potentials must simply measure the dissimilarity of these attributes for a given unary or pairwise mapping, respectively:

$$V_{C_1}(X_{i_1} = x_{k_1}) = D_1(y^d_{i_1}, y^c_{k_1}) \tag{5.41}$$

and

$$V_{C_2}(X_{i_1} = x_{k_1}; X_{i_2} = x_{k_2}) = D_2(y^d_{i_1 i_2}, y^c_{k_1 k_2}), \tag{5.42}$$

where $D_1(y^d_i, y^c_k)$ is a distance measure between the unary vector attributes of nodes d_i in G_d and c_k in G_c. Analogously, $D_2(y^d_{ij}, y^c_{kl})$ is a distance measure between the binary vector attributes of edges d_{ij} in G_d and c_{kl} in G_c.

What we finally obtain is the optimization of:

$$U(M) = \sum_{i_1=1}^{T}\sum_{k_1=1}^{S} M_{i_1k_1} D_1(y^d_{i_1}, y^c_{k_1}) +$$
$$+ \sum_{C_2 \in \mathcal{C}_2}\sum_{k_1=1}^{S}\sum_{k_2=1}^{S} M_{i_1k_1}M_{i_2k_2}D_2(y^d_{i_1i_2}, y^c_{k_1k_2}) \tag{5.43}$$

with respect to M, subject to the same condition than the WGM problem (5.10).

The complete version of the problem is obtained by setting G_m as being the full graph:

$$U(M) = \sum_{i_1=1}^{T}\sum_{k_1=1}^{S} M_{i_1k_1} D_1(y^d_{i_1}, y^c_{k_1}) +$$
$$+ \sum_{i_1=1}^{T}\sum_{i_2=1}^{T}\sum_{k_1=1}^{S}\sum_{k_2=1}^{S} M_{i_1k_1}M_{i_2k_2}D_2(y^d_{i_1i_2}, y^c_{k_1k_2}). \tag{5.44}$$

Note that the approximated problem defined by Eq. (5.43) is now *really* an approximation, in the sense that the missing information is not redundant as in the point pattern matching problem.

5.5.2 Generalized attributed graph matching

Note that we have described the attributed graph matching problem in accordance with the usual formulation, where only nodes and edges have attributes. However, this is not necessarily so. Li has argued in (LI, 1992) that higher-order attributes may be involved, and suggested the term attributed relational structure (ARS), instead of attributed relational graph (ARG), to denote graphs where also ternary relations are present. In the present work, we take this view into attributed graph matching. The extension of the optimization problem for this case is straightforward from the MRF formulation: instead of considering just singleton and pairwise cliques in the factorization, we may include any higher-order clique, so that ternary and other possibly complex attributes involving several nodes can also be explicitly considered. In this way Eq. (5.40) is generalized to

$$U(x) = \sum_{\Omega=1}^{T}\left(\sum_{C_\Omega \in \mathcal{C}_\Omega} V_{C_\Omega}(x_{C_\Omega})\right), \tag{5.45}$$

and using analogous notation for higher-order distance measures $D_\Omega(.,.)$, we have a generalization for the approximated optimization problem of Eq. (5.43):

$$U(M) = \sum_{\Omega=1}^{T}\left(\sum_{i_1 \in \mathcal{C}_1} \cdots \sum_{i_\Omega \in \mathcal{C}_\Omega}\sum_{k_1=1}^{S} \cdots \sum_{k_\Omega=1}^{S} \mathcal{D}\right) \tag{5.46}$$

where

$$\mathcal{D} = M_{i_1k_1} \cdots M_{i_\Omega k_\Omega} D_\Omega(y^d_{i_1\cdots i_\Omega}, y^c_{k_1\cdots k_\Omega}) \tag{5.47}$$

and for the generalized complete problem:

$$U(M) = \sum_{\Omega=1}^{T} \left(\sum_{i_1=1}^{T} \cdots \sum_{i_\Omega=1}^{T} \sum_{k_1=1}^{S} \cdots \sum_{k_\Omega=1}^{S} \mathcal{D} \right) \tag{5.48}$$

subject to the same constraint (5.10). In general, of course, D_n can incorporate D_m for $m < n$. However, we explicitly denote each D_n as having only terms which cannot be split into lower-order ones.

These derivations show that, starting from the MAP formulation in the MRF, we can obtain the standard definition of the attributed graph matching problem (Eq. (5.44)), as well as approximated versions of the problem (Eq. (5.43)), which are dependent on the chosen graph G_m. Moreover, the derivations lead us to a general definition of the problem, both in its complete (Eq. (5.48)) and approximated (Eq. (5.46)) versions.

6 THESIS CONTRIBUTIONS: EXPERIMENTS

In this chapter we present a contribution by providing experimental support for the technique proposed in this work. In a series of experiments we compare the accuracy of our technique (which we denote as JT) and that of Probabilistic Relaxation Labeling (PRL). PRL is often used as a benchmark to compare new algorithms for matching because (1) it is the most widely used technique for this purpose, (2) it is easy to implement and (3) it gives state-of-the-art results (GOLD; RANGARAJAN, 1996). There is an additional reason to compare our method with PRL. Both methods have a similar principle: they are "message-passing" algorithms that update probabilities according to evidence provided by neighbor sites. The original source of the results presented here is (CAETANO; CAELLI; BARONE, 2004c).

6.1 The Benchmark

Probabilistic Relaxation Labeling has its roots in the 1970's (ROSENFELD; HUMMEL; ZUCKER, 1976; DAVIS, 1979; ULMANN, 1979), when it was first conceived as an heuristic method for classification where the probability of classifying a "site" into a given label is a function of the support that its neighbors give to that label. The concepts of "site" and "neighbor" naturally fit into a graph structure, and PRL has been used ever since in domains like image processing, analysis and classification (where the graph is a grid) and graph matching (where the graph is in principle arbitrary).

Several variations on the relaxation labeling theme appeared over the years (ROSENFELD; HUMMEL; ZUCKER, 1976; DAVIS, 1979; FAUGERAS; BERTHOD, 1981; ULMANN, 1979; ROSENFELD; KAK, 1982; CHRISTMAS; KITTLER; PETROU, 1994; HUMMEL; ZUCKER, 1983). We decided to use as a benchmark the algorithm described in (ROSENFELD; KAK, 1982), which is a standard form of relaxation labeling that has been used for purposes of comparison when introducing new algorithms (GOLD; RANGARAJAN, 1996).

6.1.1 The algorithm

In order to introduce the algorithm, assume the existence of a graph with a particular connectivity and our purpose is to assign a label to each vertex. Assume that there are S labels and T vertices. Assume also that there is a *compatibility function* $c(i,k;j,l)$ that measures the strength of the likelihood with which vertex i receives label k *and* vertex j receives label l. The fundamental idea of PRL is to update the probability of assigning label k to vertex i by taking into account its neighbors "opinions" about this assignment. The global "opinion" or support that the neighbors of i pass to i at iteration r is then an average over the individual opinions at iteration r:

$$q_{ik}^{(r)} = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \sum_{l=1}^{S} c(i,k;j,l) p_{jl}^{(r)} \tag{6.1}$$

where $\mathcal{N}_i$ is the set of neighbors of vertex i, q_{ik} is the support that the assignment $i \rightarrow k$ receives from the neighbors of i and p_{jl} is the probability of the assignment $j \rightarrow l$. For the particular case where the graph is fully connected and all vertices are neighbors we have

$$q_{ik}^{(r)} = \frac{1}{T-1} \sum_{j \neq i} \sum_{l=1}^{S} c(i,k;j,l) p_{jl}^{(r)}. \tag{6.2}$$

The probability that i should be assigned to k is then updated based on the support:

$$p_{ik}^{(r+1)} = \frac{p_{ik}^{(r)} q_{ik}^{(r)}}{\sum_{k=1}^{S} p_{ik}^{(r)} q_{ik}^{(r)}}. \tag{6.3}$$

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

The performance of the algorithm is dependent on the initialization of the probabilities p_{ik}. If we are lucky to initialize the probabilities in such a way that the maximum of the matching similarity is close to the first solution, the algorithm may converge in a reasonable amount of iterations. On the other hand, if this is not the case and the complexity of the problem is high (if T and/or S are not small), the amount of iterations needed in order to find the optimal solution may be so large that it is simply not feasible to wait the required time. These two different behaviors are clearly observed in the experiments described in this chapter.

6.1.2 Computational complexity

A careful analysis of the steps involved in the PRL algorithm described above leads to the conclusion that each iteration has computational complexity $O(S^3T^2)$. From Eq. (6.2), the calculation of each q_{ik} is easily seen to be $O(ST)$, assuming that the neighborhood of node i consists of all the other nodes. The computation of each p_{ik}, as seen from Eq. (6.3), requires S terms involving q_{ik}, what results in a complexity of $O(S^2T)$ for the computation of each p_{ik}. Since there are ST p_{ik}s, the total computational complexity for a single iteration of all sites results in $O(S^3T^2)$.

However, notice that the number of iterations is not considered (simply because it is not determined). In practical applications, the number of iterations is a parameter that must be tuned properly, according to the specific complexity of the particular problem. In the experiments performed in this chapter, it was set to 200 (what still gives poor results when compared to the proposed technique, as will be shown).

6.2 Experiments

All the performed experiments consist in generating artificial point sets in digital images of size 256x256 pixels. The *codomain* point set is created first in one image by

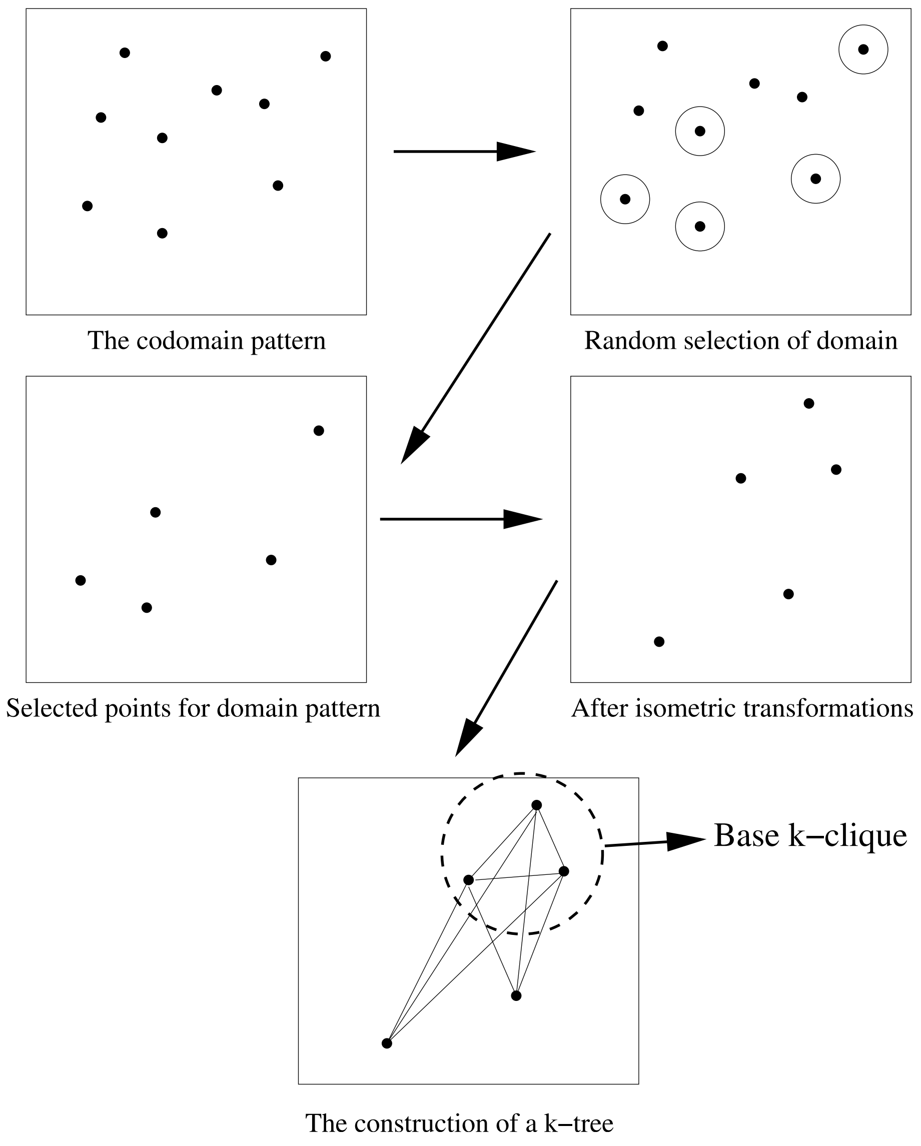

The codomain pattern
Random selection of domain
Selected points for domain pattern
After isometric transformations
Base k−clique
The construction of a k−tree

Figure 6.1: The process of creating domain and codomain point sets for experimentation: the k-tree model is created from the final version of the domain pattern by fully connecting a base k-clique in general position and by connecting to its vertices all the remaining nodes.

randomly selecting, according to a uniform distribution, a fixed number S of pixels as representing the points. The implementation simply generates the cartesian coordinates of the selected points ensuring that no coincident points are produced. The *domain* point set is then created *from* the codomain point set by selecting from it a subset of T points according to a uniform distribution. Finally, the domain point set is rotated, translated and reflected randomly. Figure 6.1 illustrates this process. Since the experiments were performed in $\mathbb{R}^2$, the computational complexity of the proposed algorithm is $O(S^4T)$ (recall from chapter 5 that the complexity of the algorithm is $O(S^{d+2}T)$, where d is the dimension of the Euclidean Space considered). The computational complexity of PRL, as described above, is $O(S^3T^2)$. However, all the experiments presented here were performed using 200 iterations for PRL, what represents a factor of $2T^2$ in the time required to perform each iteration (since all experiments use $T = 10$).

Three types of series of experiments were performed using this setting as a starting point. The first type was done in the absence of noise. The theoretical predictions indicate that in this situation the result found by our technique will be optimal, and this is indeed the case. In the second type, the sizes of the domain and codomain patterns are fixed and noise is progressively introduced in order to corrupt the codomain pattern. This simulation evaluates the robustness of both techniques with respect to the introduction of Gaussian noise. In the other type of experiments, a fixed noise level is set and the size of the codomain pattern is increased. The experiments reveal how the performance of the different techniques scale with the problem size.

6.2.1 Exact matching

In the exact matching case, there is no noise and the domain pattern has a perfect instance embedded in the codomain pattern, apart from isometric transformations. Theory expects that in this situation any k-tree model whose base k-cliques correspond to points that are in general position is equivalent to the complete model, and thus the global optimal solution to the complete problem must be found.

This is precisely what was observed in practice. We have performed 50 experiments following the setup described above but without introducing noise, keeping $T = 10$ and varying S from $S = 20$ to $S = 40$ in steps of 5. In each experiment, different positions of the points where generated and a random 3-tree was created ensuring that the base 3-cliques did not lie in a straight line (we used the model depicted in chapter 5, which has a single 3-tree).

In all experiments, the correct assignment rate of our technique was 1, meaning that no incorrect assignments were detected. PRL however exhibited a non-perfect behavior. Figure (6.2) shows the results. Each point in a curve is the average result over 50 runs of the corresponding algorithm.

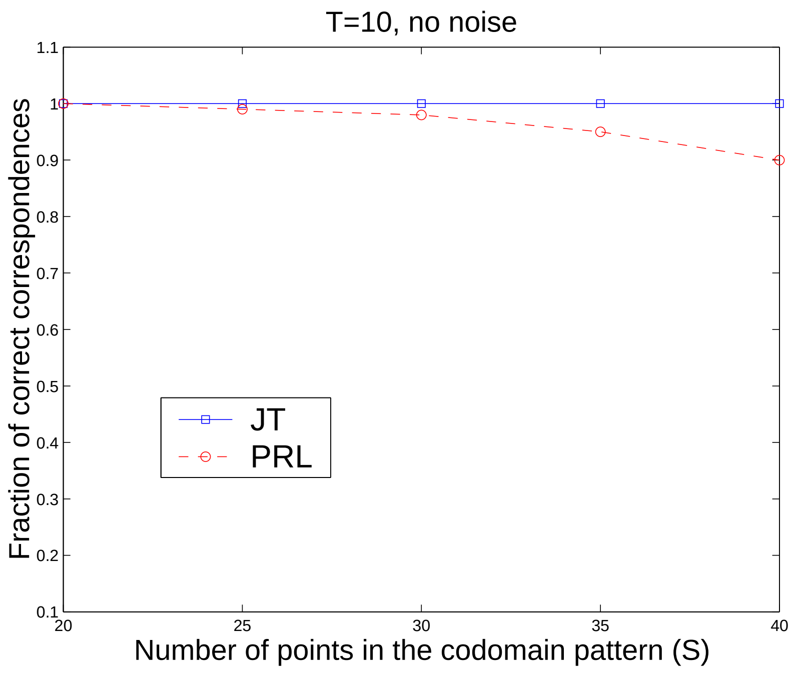


Figure 6.2: The results in the absence of noise. Experiments confirm the theoretical prediction that in this case our technique should always return the global optimum, that for exact matching means cost function equals to zero or equivalently similarity function equals to one.

6.2.2 Inexact matching: sensitivity to noise

In the second series of experiments, we fix the sizes of the domain and codomain patterns (T was set to 10 and S to 20) and progressively introducing noise in the codomain pattern. First the domain and codomain patterns are created according to the procedure described in the beginning of this section. Then the codomain pattern is perturbed by the introduction of Gaussian noise. All x and y point coordinates in the codomain pattern were shifted by numbers drawn independently from a normal distribution with zero mean and varying standard deviation. The standard deviation (std) was progressively set to $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ pixels. In this way, we generated 10 experiments: each with a different level of noise.

Each of these 10 experiments was run 1000 times. In each of these runs, 20 random points were generated for the codomain pattern and a random selection of 10 points was done for building the domain pattern, followed by random rotation, translation and reflection. Finally, noise was introduced in the codomain pattern as explained above.

The results are shown in Figure 6.3. Solid lines correspond to the proposed method (the JT method), while dashed ones correspond to PRL. Different curves were produced for JT and PRL, one for each type of similarity function described in chapter 5 (Gaussian, Hyperbolic and Increasing Weighting). The four different graphs correspond to four different values for the parameter σ in the similarity functions (note that the three similarity functions involve a parameter that essentially determine the width of the function). σ is such that the minimum possible value for the similarity functions is normalized to K, whose values are in the top of the images in the figure. This normalization prevents underflow in the computation of the similarity functions.

Note that *every point* in the curves of Figure 6.3 represents the average result over 1000 runs of the algorithm for the correspondent noise level and similarity function. Since each figure has 6 curves each with 10 levels of noise, the total number of experiments run in order to generate a single graph of Figure 6.3 is 60,000. A total of 240,000 experiments were performed in order to generate the four graphs in this figure. The amount of computational processing needed in order to generate the four graphs of Figure 6.3 was nearly 108 hours (4.5 days) using a MATLAB code running over Linux on a computer with Intel Pentium 4 CPU at 2.4 GHz and 1Gb of RAM.

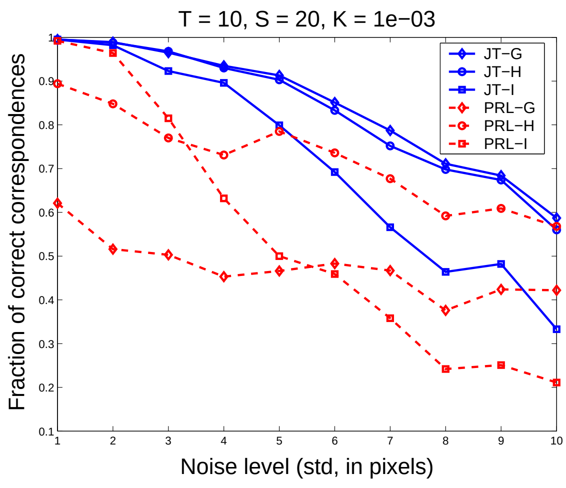
Essentially what we can infer from these results is that the performance of PRL is roughly upper bounded by that of JT. Clearly the Hyperbolic version of PRL performs equivalently to the Gaussian and Hyperbolic versions of JT for small K. We may in fact ask what is the real advantage of using JT. The answer comes in what follows.

6.2.3 Inexact matching: sensitivity to problem size

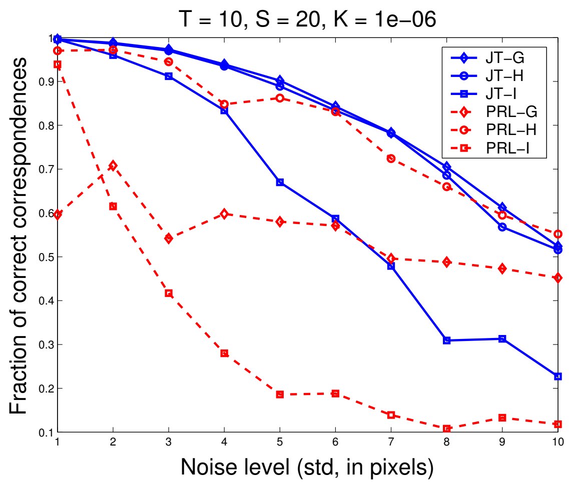
In the above set of experiments, modest values for the sizes of domain and codomain patterns were used. How will the performances of both techniques scale with problem size? In order to answer this question, we prepared a series of experiments in which the size of the codomain pattern is progressively increased while the noise level introduced is kept constant.

A set of 8 increasingly complex matching tasks were carried out, where graphs of size (T,S) = (10,15), (10,20), (10,25), (10,30), (10,35), (10,40), (10,45) and (10,50) were matched using both JT and PRL[1]. Results are shown in Figure 6.4. The noise level was

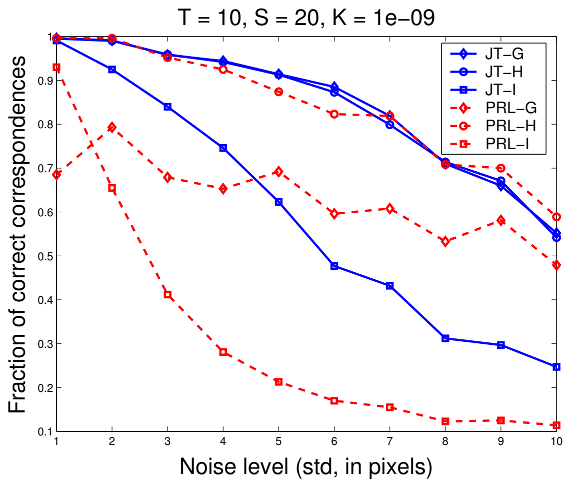
[1]Note that our interest is to increase the size of the “database” point set (the codomain pattern), because it can be done without changing the size of the domain point set. We could of course increase the size of



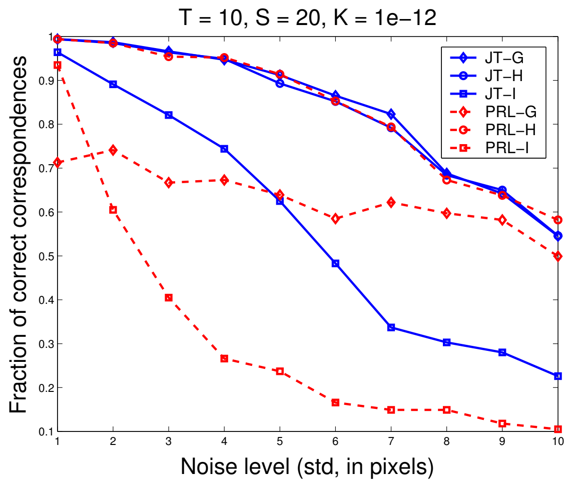
(a)



(b)



(c)



(d)

Figure 6.3: Robustness with respect to the noise level in the codomain pattern. Results for both the proposed Junction Tree technique (JT) and Probabilistic Relaxation Labeling (PRL), using the Gaussian (G), Hyperbolic Tangent (H) and Increasing Weighting (I) similarity functions.

set to a standard deviation of 2 pixels. As in the previous experiments, the three similarity functions (Gaussian, Hyperbolic Tangent and Increasing Weighting) were considered as well as the four values for K, resulting in the four graphs in the figure. Also, *each point* in a curve of Figure 6.4 corresponds to the result over 1000 runs of the algorithms for the correspondent size of codomain pattern and type of similarity function. Since there were 6 curves and 8 operating regions, the total amount of experiments run in a single graph was 48,000 (making up a total of 192,000 in the four graphs of Figure 6.4). The total amount of time needed to generate the four graphs (with the same hardware and software used in the experiment of varying noise) in this figure was about 1100 hours: more than

T and reduce the amount of points in the codomain pattern that do not correspond to the domain in order to keep S constant. However, increasing T results in an exponential growth in the search space (which has S^T configurations), what will severely affect the performance of PRL, which depends on the initialization as will be seen. JT, on the other hand, will always find the global maximum in a determined number of messages. The overall conclusion is that increasing T but not S will be much worse to PRL than the opposite, and we decided to choose the experimental setup where PRL performs best.

45 days.

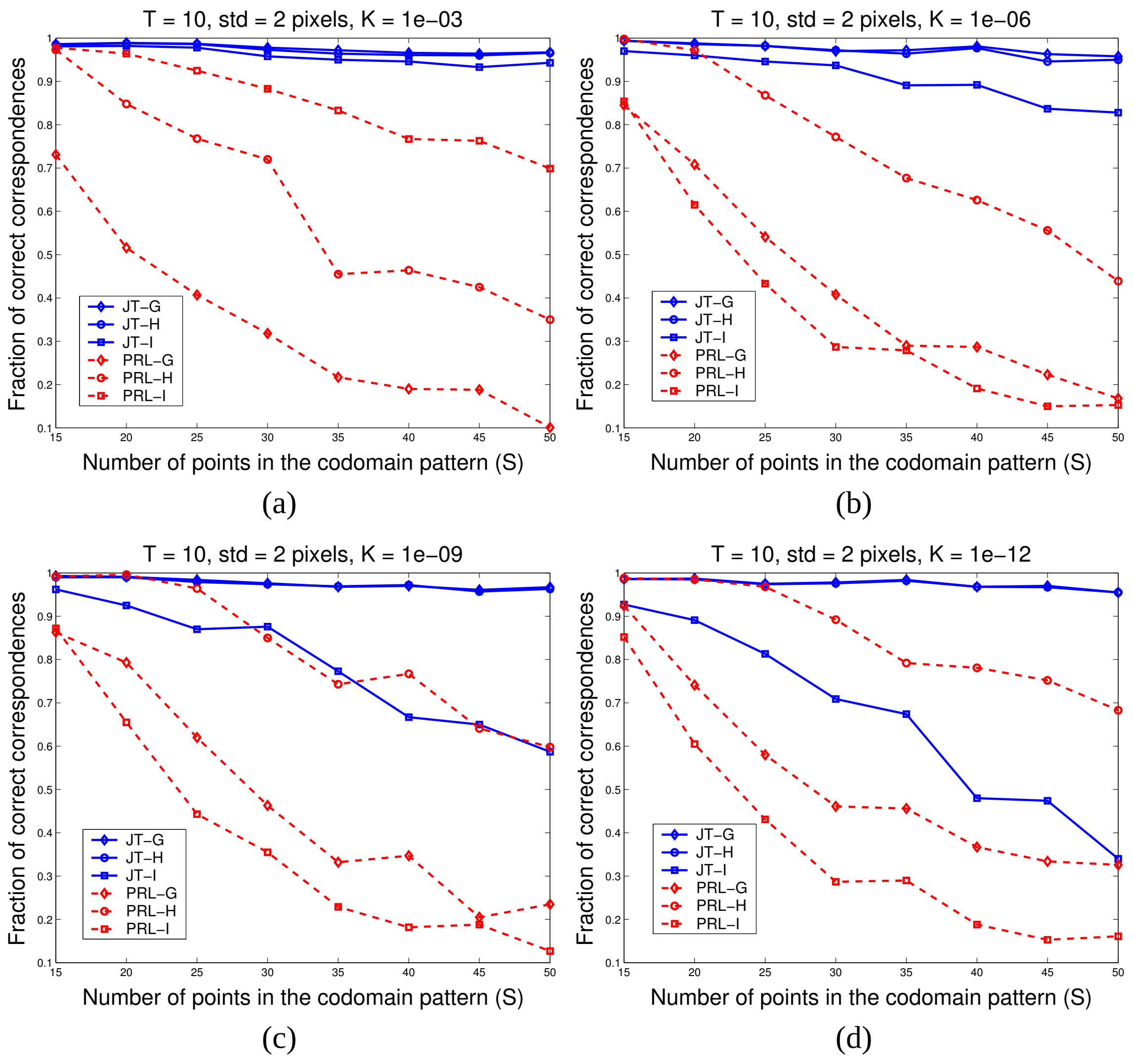


Figure 6.4: Robustness with respect to the size of the codomain pattern. Results for both the proposed Junction Tree technique (JT) and Probabilistic Relaxation Labeling (PRL), using the Gaussian (G), Hyperbolic Tangent (H) and Increasing Weighting (I) similarity functions.

The difference in performance between the techniques is clearly seen to increase with problem size. Indeed, this second set of experiments enlighten the understanding of the results in the first series of experiments. Notice that, in the first set of experiments, we have $T = 10$ and $S = 20$, what corresponds in Figure 6.4 to operating points where the Hyperbolic version of PRL performs similarly to at least the Gaussian and the Hyperbolic versions of JT. When S is increased, however, a significant divergence is observed between the evolutions of performances for the different techniques: while JT retains practically the same performance for the Gaussian and Hyperbolic functions, PRL has a severe drop for all three functions. Note that despite the fact that the Increasing Weighting function for JT loses performance quickly, the same happens with PRL in a more severe scenario.

This analysis lead us to conclude that the similar results between PRL and JT in the first series of experiments is mainly due to the fact that the operating region studied in those experiments ($T = 10$ and $S = 20$) is *not* representative of the circumstances in which JT outperforms PRL. The general conclusion is that JT is only significantly better than PRL for complex matching problems involving large point sets.

7 DISCUSSION

The theoretical results presented in chapter 5 assure that the final match using the proposed technique (JT) corresponds to the *global* maximum of the similarity function over the simplified k-tree graph. Moreover, in the limit when there is no noise (exact point set matching), the optimization problem over a k-tree is equivalent to the complete optimization problem, and the obtained result as a consequence is optimal. This is done with a predictable number of computations and in a feasible amount of time (the whole algorithm has polynomial time complexity). The algorithm PRL, used as a benchmark for comparison via experimental performance evaluation, also has polynomial time complexity for each iteration, but (a) it only converges to a *local* maximum of the complete similarity function and (b) the number of iterations needed in order to reach this local maximum is not determined. Moreover, the lack of ability to reach the global optimum in polynomial time for this problem is a property not restricted to PRL, but to all algorithms known until now.

In spite of the fact that our technique in principle presents these two advantages with respect to PRL, in this work we have *not* exploited the fact that our technique finds the global optimum while PRL finds a local one. This is because the function to be optimized (Eq. (2.10)) is *strictly quasi-concave* (it has a unique local maximum which happens to be the global maximum), since it is a sum of strictly quasi-concave functions, what can be shown to be also strictly quasi-concave[1] (LIMA, 2000). As a result, any local update in the direction of the gradient will ultimately reach the global maximum, and in principle local optimization techniques such as PRL would be able to reach the global maximum after a sufficient number of iterative steps. However, the experimental evidence provided in the previous chapter clearly indicates that our technique tends to progressively outperform PRL when the task complexity is increased. What causes this gain in performance if we are not taking advantage of the global x local optimization issue?

The answer arises if we pay attention to a fundamental difference between the two techniques. PRL is an iterative process where the number of iterations needed in order to reach the maximum is not specified. For simple problems where the search space is not huge, PRL may converge quickly to a reasonable solution which is close to the maximum, even if the initialization is not close to the solution. This is indeed evidenced in the experiments, where we observe that there is no significant difference between our technique and PRL for matching problems where T and S are small (10-20). However, in cases such as those experimented and shown in Figure 6.4, where huge search spaces are involved (50^{10} configurations!), the number of iterations needed to make PRL reach a

[1] We mean here a sum of functions which are strictly quasi-concave with respect to *different* variables. If the sum is over strictly quasi-concave functions of the same variable, the resulting function is not necessarily strictly quasi-concave.

satisfactory solution is inconceivable if we assume random initialization. For example, the experiments described in Figures 6.3 and 6.4 were done with 200 iterations and random initialization, what already made PRL 2 times slower than our technique. We tried to increase the number of iterations in PRL to 500, 1000 and even 2000 without having any significant improvement in the graphs[2].

The key difference of the proposed approach is that the number of "iterations"[3] involved is absolutely determined *and* feasible. It is linear on the number of maximal cliques in the graph (remember from chapter 5 that this number is $2(N-1)$ where N is the number of maximal cliques). As a result, the solution is found in a predictable and feasible amount of time.

Of course that, if the similarity function were not strictly quasi-concave, we would take advantage of the fact that the proposed technique finds the global optimum while others don't. In this case, even if we could wait indefinitely the convergence of alternative techniques, they would possibly produce unsatisfactory results due to their inability of performing global optimization.

Therefore, we may as a general recommendation assert that the proposed technique is a serious alternative to existent approaches for the point set matching problem, since it assures both polynomial time complexity and optimality in the limit of exact matching. In particular, the experiments have shown that it is a serious alternative to PRL for inexact matching, and it is as more effective as larger are the sizes of the involved point patterns.

[2]However, due to limited availability of computational resources we only run 100 times each experiment of 500, 1000 and 2000 iterations.

[3]We use quotes because they are not iterations in the sense of refining previous estimates, but computations ("messages") that are optimal at every step.

8 CONCLUSION

This work presented a new way of performing both exact and inexact point set matching in Euclidean Spaces of arbitrary dimension. The first contribution is to formulate point set matching as a graph matching problem where nodes and weights of the graphs correspond to, respectively, points and their relative distances in a Euclidean Space. The second contribution consists in reformulating this graph matching problem as one of exact probabilistic inference in a graphical model where the random variables are the nodes in one point pattern and the realizations are the nodes in the other point pattern. In this inference problem, a Maximum a Posteriori (MAP) optimal solution is sought. The third contribution consists in proving that a particular sparse edge constitution for the graphical model results in a simple model that, in the limit case of exact matching, is equivalent to the fully connected model. It turns out that the particular form of this model (a k-tree) is chordal and has a fixed maximal clique size, what allows us to obtain a polynomial time algorithm for both inexact and exact point set matching in arbitrary dimensions which in addition is provably optimal in the exact matching case. The fourth contribution of this work consists in extending the formulation by showing that any matching problem can be considered an inference problem in a graphical model. The fifth and last contribution is the software implementation of the resulting algorithm and its comparison with a standard benchmark in the literature through a series of controlled and time-consuming experiments. These experiments evidence that the proposed approach presents significant performance improvement for large graph sizes.

REFERENCES

AKUTSU, T.; KANAYA, K.; OHYAMA, A.; FUJIYAMA, A. Point matching under non-uniform distortions. **Discrete Applied Mathematics. Special Issue: Computational biology series issue IV**, [S.l.], v.127, n.1, p.5–21, 2003.

BARROW, H. G.; POPPLESTONE, R. Relational Descriptions in Picture Processing. **Machine Intelligence**, [S.l.], v.6, p.377–396, 1971.

BERRETI, S.; BIMBO, A. D.; VICARIO, E. Efficient matching and indexing of graph models in content-based retrieval. **IEEE Trans. PAMI**, [S.l.], v.23, n.10, p.1089–1105, 2001.

BESAG, J. Spatial interaction and the statistical analysis of lattice systems. **J. Royal Statistical Soc., Series B**, [S.l.], v.36, p.192–236, 1974.

BESL, P. J.; JAIN, R. C. Three-Dimensional Object Recognition. **Artificial Intelligence**, [S.l.], v.17, n.1, p.75–145, 1985.

BHANU, B. Representation and shape matching of 3-D objects. **IEEE Trans. PAMI**, [S.l.], v.6, n.3, p.340–351, 1984.

BHANU, B.; FAUGERAS, O. D. Shape matching of two-dimensional objects. **IEEE Trans. PAMI**, [S.l.], v.6, n.2, p.137–156, 1984.

BOYER, K. L.; KAK, A. C. Structural stereopsis for 3-D vision. **IEEE Trans. on PAMI**, [S.l.], v.10, n.2, p.144–166, 1988.

CAELLI, T.; CAETANO, T. Recent developments in the extraction and matching of image structure and syntax: from relaxation to junction tree models. In: PATTERN RECOGNITION ASSOCIATION OF SOUTH AFRICA IAPR CONFERENCE, 2003. **Proceedings. . .** [S.l.: s.n.], 2003. p.1–8.

CAELLI, T.; CAETANO, T. Graphical models for graph matching: approximate models and optimal algorithms. **Pattern Recognition Letters**, [S.l.], 2004. Invited paper in honour of Azriel Rozenfeld, forthcoming.

CAETANO, T. S.; CAELLI, T.; BARONE, D. A. C. Graphical models for graph matching. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2004, Washington, DC. **Proceedings. . .** [S.l.: s.n.], 2004. p.466–473.

CAETANO, T. S.; CAELLI, T.; BARONE, D. A. C. An optimal probabilistic graphical model for point set matching. In: INTERNATIONAL WORKSHOPS SSPR & SPR, 2004, Lisbon. **Proceedings...** [S.l.: s.n.], 2004. p.162–170.

CAETANO, T. S.; CAELLI, T.; BARONE, D. A. C. A comparison of Junction Tree and Relaxation Algorithms for point matching using different distance metrics. In: IEEE INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION, 2004, Cambridge, UK. **Proceedings...** [S.l.: s.n.], 2004. Accepted.

CARCASSONI, M.; HANCOCK, E. R. Spectral correspondence for point pattern matching. **Pattern Recognition**, [S.l.], v.36, p.193–204, 2003.

CHRISTMAS, W. J.; KITTLER, J.; PETROU, M. Structural Matching in Computer Vision Using probabilistic Relaxation. **IEEE Trans. PAMI**, [S.l.], v.17, n.8, p.749–764, 1994.

CONNELLY, R. Rigidity and energy. **Invent. Math.**, [S.l.], v.66, n.1, p.11–33, 1982.

CONNELLY, R. **Generic global rigidity**. Preprint privately provided.

CONTE, D.; FOGGIA, P.; SANSONE, C.; VENTO, M. Thirty years of graph matching in pattern recognition. **Special Edition of the International Journal of Pattern Recognition and Artificial Intelligence on Graph Theory in Vision**, [S.l.], v.18, n.3, p.265–298, 2004.

CORDELLA, L. P.; FOGGIA, P.; SANSONE, C.; VENTO, M. Graph matching: a fast algorithm and its evaluation. In: IEEE INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION, 1998. **Proceedings...** [S.l.: s.n.], 1998. p.1582–1584.

DATTORRO, J. **Euclidean Distance Matrices**. Ph.D. Thesis, Stanford University.

DAVIS, L. S. Shape matching using relaxation techniques. **IEEE Trans. PAMI**, [S.l.], v.1, n.1, p.60–72, 1979.

ESHERA, M.; FU, K. A Graph Distance Measure for Image Analysis. **IEEE Transactions on Systems Man and Cybernetics**, [S.l.], v.14, n.3, p.353–363, 1984.

FAUGERAS, O. D.; BERTHOD, M. Improving consistency and reducing ambiguity in stochastic labeling: an optimization approach. **IEEE Trans. PAMI**, [S.l.], v.3, p.412–423, 1981.

FISCHLER, M. A.; ELSCHLAGER, R. A. The representation and matching of pictorial structures. **IEEE Trans. on Computers**, [S.l.], v.22, n.1, p.67–92, 1973.

FU, K. S. A step towards unification of syntatic and statistical pattern recognition. **IEEE Trans. PAMI**, [S.l.], v.5, n.2, p.200–205, 1983.

GEMAN, S.; GEMAN, D. Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images. **IEEE Transactions on PAMI**, [S.l.], v.6, n.6, p.721–741, 1984.

GOLD, S.; RANGARAJAN, A. A graduated assignment algorithm for graph matching. **IEEE Trans. PAMI**, [S.l.], v.18, n.4, p.377–388, 1996.

GRAVER, J.; SERVATIUS, B.; SERVATIUS, H. **Combinatorial rigidity**. Providence, RI: American Mathematical Society, 1993.

GREGORY, L.; KITTLER, J. Using graph search techniques for contextual color retrieval. In: INTERNATIONAL WORKSHOPS SSPR & SPR, 2002. **Proceedings...** [S.l.: s.n.], 2002. p.133–142.

HAMMERSLEY, J. M.; CLIFFORD, P. Markov fields on finite graphs and lattices. (unpublished), [S.l.], 1971.

HANCOCK, E.; WILSON, R. C. Graph-Based Methods for Vision: a yorkist manifesto. In: INTERNATIONAL WORKSHOPS SSPR & SPR, 2002. **Proceedings...** [S.l.: s.n.], 2002. p.31–46.

HUANG, H.-X.; LIANG, Z.-A.; PARDALOS, P. M. Some properties of the Euclidean distance matrix and positive semidefinite matrix completion problems. **Journal of Global Optimization**, [S.l.], v.25, p.3–21, 2003.

HUMMEL, R. A.; ZUCKER, S. W. On the foundations of the relaxation labeling process. **IEEE Trans. on PAMI**, [S.l.], v.5, n.3, p.267–286, 1983.

IRNIGER, C.; BUNKE, H. Graph matching: filtering a large database of graphs using decision trees. In: WORKSHOP ON GRAPH-BASED REPRESENTATIONS IN PATTERN RECOGNITION, 2001. **Proceedings...** [S.l.: s.n.], 2001. p.239–249.

JORDAN, M. I. **An Introduction to Probabilistic Graphical Models**. [S.l.: s.n.], Forthcoming, 2004.

KITTLER, J. V.; HANCOCK, E. R. Combining Evidence in Probabilistic Relaxation. **Int. Journal of Pattern Recognition and Artificial Intelligence**, [S.l.], v.3, p.29–51, 1989.

KOSINOV, S.; CAELLI, T. Inexact Multisubgraph matching using Graph Eigenspace and Clustering Models. In: INTERNATIONAL WORKSHOPS SSPR & SPR, 2002. **Proceedings...** [S.l.: s.n.], 2002. p.133–142.

LARROSA, J.; VALIENTE, G. Constraint satisfaction algorithms for graph pattern matching. **Mathematical structures in computer science**, [S.l.], v.12, p.403–422, 2002.

LAURITZEN, S. L. **Graphical Models**. New York, NY: Oxford University Press, 1996.

LAZARESCU, M.; BUNKE, H.; VENKATESH, S. Graph matching: fast candidate elimination using machine learning techniques. In: INTERNATIONAL WORKSHOPS SSPR & SPR, 2000. **Proceedings...** [S.l.: s.n.], 2000. p.236–245.

LEUNG, T. K.; BURL, M. C.; PERONA, P. Finding faces in cluttered scenes using random labeled graph matching. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, 1995. **Proceedings...** [S.l.: s.n.], 1995. p.637–644.

LI, S. Z. Matching: invariant to translations, rotations and scale changes. **Pattern Recognition**, [S.l.], v.25, n.6, p.583–594, 1992.

LI, S. Z. A Markov random field model for object matching under contextual constraints. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 1994. **Proceedings...** [S.l.: s.n.], 1994. p.866–869.

LIMA, E. L. **Curso de Análise**. Rio de Janeiro, RJ: IMPA, 2000. v. 2.

LUO, B.; HANCOCK, E. Structural graph matching using the EM algorithm and singular value decomposition. **IEEE Trans. PAMI**, [S.l.], v.23, n.10, p.1120–1136, October 2001.

MARTIN, Y.; BURES, M.; DANAHER, E.; DELAZZER, J.; LICO, I. A fast new approach to pharmacophore mapping and its application to dopaminergic and bezodiazepine agonists. **J. of Computer-Aided Molecular Design**, [S.l.], v.7, p.83–102, 1993.

MCKAY, B. D. Practical graph isomorphism. **Congressus numerantium**, [S.l.], v.30, p.45–87, 1981.

MESSMER, B. **Efficient graph matching algorithms for preprocessed model graphs**. Ph.D. Thesis, University of Bern.

MESSMER, B. T.; BUNKE, H. A new algorithm for error-tolerant subgraph isomorphism detection. **IEEE Trans. PAMI**, [S.l.], v.20, n.5, p.493–503, 1998.

MURTAGH, F. A new approach to point-pattern matching. **Astronomical Society of the Pacific**, [S.l.], v.104, n.674, p.301–307, 1992.

PEARL, J. **Probabilistic Reasoning in Intelligent Systems**: networks of plausible inference. San Mateo, CA: Morgan Kaufmann Publishers, 1988.

PELILLO, M. Replicator equations, maximal cliques, and graph isomorphism. **Neural Computation**, [S.l.], v.11, p.1933–1955, 1999.

PELILLO, M.; SIDDIQI, K.; ZUCKER, S. Matching hierarchical structures using association graphs. **IEEE Trans. PAMI**, [S.l.], v.21, n.11, p.1105–1120, 1999.

REIMANN, D.; HAKEN, H. Stereo Vision by Self-Organization. **Biological Cybernetics**, [S.l.], v.71, n.1, p.17–26, 1994.

ROSENFELD, A.; HUMMEL, R. A.; ZUCKER, S. W. Scene labelling by relaxation operations. **IEEE Transactions on Systems, Man and Cybernetics**, [S.l.], v.6, n.6, p.420–433, 1976.

ROSENFELD, A.; KAK, A. C. **Digital Picture Processing**. New York, NY: Academic Press, 1982.

SANFELIU, A.; FU, K. A distance measure between attributed relational graphs for pattern recognition. **IEEE Transactions on Systems, Man and Cybernetics**, [S.l.], v.13, n.3, p.353–362, 1983.

SHAFER, G.; SHENOY, P. Probability propagation. **Annals of Mathematics and Artificial Intelligence**, [S.l.], v.2, p.327–352, 1990.

SHAPIRO, L.; BRADY, J. Feature-based Correspondence - An Eigenvector Approach. **Image and Vision Computing**, [S.l.], v.10, p.283–288, 1992.

SHAPIRO, L. G.; HARALICK, R. M. Structural descriptions and inexact matching. **IEEE Trans. PAMI**, [S.l.], v.3, n.5, p.504–519, 1981.

SHAPIRO, L. G.; HARALICK, R. M. A metric for comparing relational descriptions. **IEEE Trans. PAMI**, [S.l.], v.7, n.1, p.90–94, 1985.

SIMIC, P. D. Constrained nets for graph matching and other quadratic assignment problems. **Neural Computation**, [S.l.], v.3, n.2, p.268–281, 1991.

SUGANTHAN, P. N. Structural pattern recognition using genetic algorithms. **Pattern Recognition**, [S.l.], v.35, p.1883–1893, 2002.

SUGANTHAN, P. N.; YAN, H. Recognition of handprinted Chinese characters by constrained graph matching. **Image and Vision Computing**, [S.l.], v.16, n.3, p.191–201, 1998.

SUGANTHAN, P.; TEOH, E.; MITAL, D. Pattern recognition by graph matching using potts mft networks. **Pattern Recognition**, [S.l.], v.28, p.997–1009, 1995.

TON, J.; JAIN, A. K. Registering Landsat images by point matching. **IEEE Trans. on Geoscience and Remote Sensing**, [S.l.], v.27, n.5, p.642–651, 1989.

TSAI, W. H.; FU, K. S. Error-correcting isomorphism of attributed relational graphs for pattern analysis. **IEEE Trans. on Systems, Man and Cybernetics**, [S.l.], v.9, n.2, p.757–768, 1979.

TSAI, W. H.; FU, K. S. Subgraph error-correcting isomorphisms for syntactic pattern recognition. **IEEE Trans. on Systems, Man and Cybernetics**, [S.l.], v.13, n.1, p.48–62, 1983.

ULLMAN, J. An algorithm for subgraph isomorphism. **Journal of the ACM**, [S.l.], v.23, n.1, p.31–42, 1976.

ULMANN, S. Relaxation and constraint optimization by local process. **Computer Graphics and Image Processing**, [S.l.], v.10, p.115–195, 1979.

UMEYAMA, S. An eigen decomposition approach to weighted graph matching problems. **IEEE Trans. PAMI**, [S.l.], v.10, p.695–703, 1998.

VASILICA, L.; PRAKASH, S. A Comparison of Lauritzen-Spiegelhalter, Hugin, and Shenoy-Shafer Architectures for Computing Marginals of Probability Distributions. In: UNCERTAINTY IN ARTIFICIAL INTELLIGENCE, 1998, San Francisco, CA. **Proceedings. . .** Morgan Kaufmann Publishers, 1998. p.328–337.

WEST, D. B. **Introduction to Graph Theory**. Upper Saddle River, NJ: Prentice Hall, 2001.

WYK, B. J. van; WYK, M. A. van. Kronecker Product graph matching. **Pattern Recognition**, [S.l.], v.36, n.9, p.2019–2030, 2003.

WYK, M. A. van; DURRANI, T. S.; WYK, B. J. van. A RKHS Interpolator-Based Graph Matching Algorithm. **IEEE Trans. on PAMI**, [S.l.], v.24, n.7, p.988–995, 2002.

XU, L.; KING, I. A PCA approach for fast retrieval of structural patterns in attributed graphs. **IEEE Trans. on Systems, Man and Cybernetics**, [S.l.], v.31, n.5, p.812–817, 2001.