Spectral Graph Matching, Learning, and Inference for Computer Vision

Marius Dan Leordeanu

CMU-RI-TR-09-27

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Robotics.

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

July 16, 2009

Thesis Committee:
Martial Hebert, Chair
Rahul Sukthankar,
Fernando De la Torre,
David Lowe, University of British Columbia

Contents

List of Figures

List of Tables

Abstract

Several important applications in computer vision, such as 2D and 3D object matching, object category and action recognition, object category discovery, and texture discovery and analysis, require the ability to match features efficiently in the presence of background clutter and occlusion. In order to improve matching robustness and accuracy it is important to take in consideration not only the local appearance of features but also the higher-order geometry and appearance of groups of features. In this thesis we propose several efficient algorithms for solving this task, based on a quadratic programming formulation that generalizes the classical graph matching problem. First, we introduce spectral graph matching, which is an efficient method for matching features using both local, first-order information, as well as pairwise interactions between the features. We study the theoretical properties of spectral matching in detail and show efficient ways of using it for current computer vision applications. We also propose an efficient procedure with important theoretical properties for the final step of obtaining a discrete solution from the continuous one. We show that this discretization step, which has not been studied previously in the literature, is of crucial importance for good performance. We demonstrate its efficiency by showing that it dramatically improves the performance of state-of-the art algorithms. Moreover, if used by itself, this discretization method significantly outperforms the current state-of-the-art algorithms. We also propose, for the first time, methods for learning graph matching in both supervised and unsupervised fashions. Furthermore, we study the connections between graph matching and the MAP inference problem in graphical models, for which we propose novel inference and learning algorithms. In the last part of the thesis we present an application of our matching algorithm to the problem of object category recognition, and a novel algorithm for grouping image pixels/features that can be effectively used for object category segmentation.

Funding Sources

This work was made possible in part by NSF Grants IIS0713406 and IIS0534962, Grant NBCH1030013, the Intelligent Robotics Development Program at KIST, and the Intel Graduate Fellowship program.

Acknowledgements

There are not enough words to express my gratitude to my advisor Martial Hebert. It was a true blessing and joy to have him as my mentor during my PhD years. His guidance, both technical and non-technical, as well as his care for me, are priceless. His work ethic, personality and good heart make him the raw model that will always guide my path.

I also want to thank our program manager, Suzanne Lyons Muth, who is an exceptionally kind person. She always showed me affection and care. I will never forget our coffee breaks. Her advice and kind words are invaluable to me.

To my family - my mother, Sanda Leordeanu, my father, George Leordeanu, my grandmother Florica Dunca, and my stepmother, Marieta Leordeanu - I owe my life. They gave me everything, more than I could have ever asked for. During both good and bad times, they were always there, giving me love, support and care.

Everyone has an angel, and, in my case, her name is Gina Heredea. She appeared when I needed her the most and expected her the least. The eyes of her sharp mind and clear soul gave me the light that inspired me while writing most of this thesis.

I would like to thank all my committee members. It was a great honor to have David Lowe on my thesis committee. His feedback and insightful questions were very helpful. I also want to thank Rahul Sukthankar and Fernando de la Torre for the numerous discussions we had over the years. It was with an immense pleasure that I worked with them and learned from their advice. Many times, their positivism and kindness gave me the courage to undertake problems that initially seemed unsolvable.

Last but not least, I also want to thank the people who introduced me to the world of science and research. My first Physics professor from Romania, Dragos Viorel, showed me, through his determination and passion, that the mountains are there not only to be contemplated, but also to be conquered. I share with him the love for science and truth. Later, my undergraduate mentor, Ioannis Stamos, opened the door to the fascinating world of computer vision. The moment I walked into his class I knew that I had to pursue a PhD in this field. The summer spent in his lab doing computer vision is one of the greatest experiences of my life.

Chapter 1

Introduction

Several important applications in computer vision, such as 2D or 3D object matching, object category and action recognition, object category discovery, and texture discovery and analysis, require the ability to match features efficiently in the presence of background clutter and occlusion. For good matching performance it is important to take in consideration not only the local appearance of features but also the higher-order geometry and appearance of groups of features. In this thesis we propose several efficient algorithms for this task, based on a general quadratic programming formulation that generalizes the classical graph matching problem.

Graph matching is a fundamental problem in computer vision, used mainly for 2D or 3D object/scene matching or recognition: it involves matching images, shapes, sets of image regions or other local appearance features, 2D and 3D object parts. Its importance has been recognized since the early 1970's by Barrow and Popplestone [11], and Fischler and Enschlager [68], who gave a conceptual motivation for the use of relational structures in recognition. Graph matching is related to Conditional Random Fields (CRFs) [31], [108] but, unlike most CRF models, it does not require a probabilistic formulation. In later chapters we discuss in more detail the connections between graph matching and CRFs.

Most of the recent graph matching work in computer vision considers attributed graphs that take in consideration complex unary features and pairwise relationships between them. An attributed graph is a set of nodes with associated unary features, and a set of edges between nodes, with associated second-order relationships between the features. The unary features could contain local information extracted from image regions, describing local color, texture or shape. The second-order relationships associated with the edges describe the pairwise relationships between the corresponding features, including pairwise spatial/geometric relationships, and/or pairwise appearance information. In its most general formulation, graph matching is the problem of finding correspondences between the nodes of two graphs such that both the unary information on the nodes and the pairwise information associated

with the edges are preserved as well as possible. Mathematically, graph matching can be formulated as the optimization of a certain objective/score function that incorporates both the node-to-node (or unary, first-order) as well as the edge-to-edge (or pairwise, second-order) agreements after matching two graphs. More recently, the problem of hyper-graph matching has also been studied [233], [53], by considering higher-order interactions between tuples of features (beyond pairwise). We discuss the connections between our work and hyper-graph matching, and also propose a novel method for learning in the case of higher-order matching.

Graph matching is well suited for matching objects, seen as constellations of local parts. This representation captures both the object's appearance information through the local features and the object's global shape through the higher-order geometric relationships.

1.1 Problem Formulation

We define the graph matching problem in its most general form, as follows: we want to match two attributed graphs $G^P = (V^P, E^P, A^P)$ and $G^Q = (V^Q, E^Q, A^Q)$. For each node $i \in V^P$ there is an associated feature vector $A^P_i \in A^P$. This feature usually describes the local appearance at node i. Similarly, for each node $a \in V^Q$ we have $A^Q_a \in A^Q$. For each edge $(i,j) \in E^P$ we have an associated vector $A^P_{ij} \in A^P$, usually describing the pairwise geometric relationship between nodes i and j. Similarly we have $A^Q_{ab} \in A^Q$ for each edge $(a,b) \in E^Q$. For each pair of edges $(i,j) \in E^P$ and $(a,b) \in E^Q$ there is a pairwise score function $M_{ia;jb} = f(A^P_i, A^P_{ij}, A^Q_a, A^Q_{ab})$ that measures the agreement of the first order local features (described by A^P_i and A^Q_a) and second-order relationships (defined by A^P_{ij} and A^Q_{ab}) between the pair of candidate correspondences (i,a) and (j,b). We can similarly define unary-only score functions $M_{ia;ia} = f(A^P_i, A^Q_a)$, which, in matrix form, could be stored on the diagonal of $\mathbf{M}$. Then, the graph matching problem can be formulated as an integer quadratic program (IQP), and consists of finding the indicator vector $\mathbf{x}^*$ that respects certain mapping constraints (such as one-to-one or many-to-one) and maximizes the quadratic score function:

$$\mathbf{x}^* = \text{argmax}(\mathbf{x^T M x}) \text{ s. t. } \mathbf{Ax = 1},\ \mathbf{x} \in \{\mathbf{0,1}\}^{\mathbf{n}}, \tag{1.1}$$

given the one-to-one/many-to-one constraints $\mathbf{Ax} = \mathbf{1}$, $\mathbf{x} \in \{0,1\}^n$, which require that $\mathbf{x}$ is an indicator vector such that $x_{ia} = 1$ if feature i from one image is matched to feature a from the other image and zero otherwise. Usually, one-to-one constraints are imposed on $\mathbf{x}$ such that one feature from one image can be matched to at most one other feature from the other image. The quadratic formulation is a recent, generalized definition of graph matching in computer vision, that can also accommodate the earlier, classical formulations

that focused mainly on finding the mapping between the nodes of two graphs such that the edges are preserved as well as possible. Some of the classical formulations include:

1. *Exact graph matching*: requires every node from one graph to be matched to one node from the other (bijective mapping) such that the edge structure is preserved exactly. This is also called *graph isomorphism* and the graphs matched are called *isomorphic*. The quadratic formulation above can accommodate exact graph matching when such a matching is possible, if we set $M_{ia;jb} = 1$ when we have edges between both (i, j) and (a, b), and $M_{ia;jb} = 0$ otherwise, and require $\mathbf{x}$ to obey one-to-one constraints.

2. *Inexact graph matching*: when exact graph matching is not possible, inexact matching consists of finding the mapping that best preserves the edge structure. The same quadratic formulation from exact graph matching applies to inexact graph matching.

3. *Sub-graph matching*: if the two graphs have different number of nodes, than we can only match a sub-graph of one graph to the other (or a sub-graph of the other).

4. *Weighted graph matching*: in this case, each edge from each graph has some associated weight and matching consists of finding the mapping that preserves the edge weights as well as possible, in terms of some distance function. For example, in this case we could set each pair-wise score $M_{ia;jb}$ to be inversely proportional to the absolute difference between the edge weights w_{ij} and w_{ab}.

Equation 5.1 incorporates, in a general formulation, most cases of graph matching. In this thesis we discuss both how to design and learn powerful second-order terms $M_{ia;jb}$, useful for different computer vision applications, and how to approximately optimize the matching objective function efficiently, since finding the true optimum of equation 5.1 is NP-hard. We will also discuss the connection between graph matching and MAP inference in graphical models, showing that the two problems are strongly related.

1.2 Graph Matching in Computer Vision

Graph matching has been extensively studied in computer vision since the early 70's. In vision, graph matching has varied and evolved along three avenues: 1. enriching the object model represented by the graph, by developing different unary and higher order relationships between nodes and edges/hyper-edges; 2. defining different objective functions for matching, and 3. developing efficient optimization algorithms for finding approximate solutions to the matching problem. Next we review very briefly the most relevant previous work.

1.2.1 Different Graph Matching Models

Fuzzy set theory was proposed for creating vertex and edge attributes that captured pairwise distances or relative positions between objects [20], [19], used for inexact graph matching [161] and sub-graph matching [89]. Fuzzy attributed graph models have been proposed for fingerprint verification [58] and face detection from color images [81]. Another type of graph matching representation is morphological and elastic graph matching that consists of two steps, matching with a rigid grid, followed by an elastic deformation of the grid. Applications of elastic graph matching include tracking of cyclones [113], [114], and authentication of human faces [6], [102], [101], [207]. In some applications wavelet transforms are used for creating the elastic face graph model [140], [52], [137], [226]. Related problems using morphological graph matching consist of matching facial regions [85], applying deformable spline-based models to the skeleton of non-rigid discrete objects [49], [90], [166], [206]. Similar approaches are applied to curve fitting [9] and recognition of 3D free-form objects [51]. Another graph matching representation is using generalized cylinders called geons as visual primitives to represent object models [184]. Examples of applications using geons are symmetry-based indexing of image databases [188], shape recognition from large image libraries [86], shape recognition [196],[95], [185], structural matching [224], representation and recognition of 3D free-form objects [34], [51] and hand posture recognition [212]. Conceptual graphs is another representation that has been used to model more higher level knowledge since the early 80's [197], [33] , [8], [55]. Current work in vision using graph matching usually use local appearance information for node attributes and geometrical relationships (described by distances and angles) for higher-order attributes [53], [121].

1.2.2 Different Similarity Measures of Graphs

Different similarity measures have been defined for graph matching. Some of the early work [187] compared structural representations by counting consistent subgraphs. This measure was refined in [56]. Graphs were also compared using the string edit distance [176]. The edit distance between two graphs is defined as a weighted sum of the costs of applying different edit operators, such as insert, delete and relabel nodes and edges, in order to transform one graph into the other. The edit distance was extensively used and refined in graph matching [29], [28], [208]. Early work based on more principled statistical measures include the entropy distance for structural graph matching [227] and information theoretic approaches such as [24]. Most recent work in computer vision involving graph matching [117], [43], [15] uses objective functions that fit the quadratic programming formulation given in Equation 5.1.

1.2.3 Different Optimization Algorithms for Graph Matching

Graph matching is in general NP-hard. Many different approximate algorithms have been proposed. One approach is to use genetic algorithms. In [95] the authors present a comparison between different genetic operators and compare the performance of genetic algorithms. In [191] the authors propose an evolutionary algorithm without crossover operators in order to obtain faster convergence. They also illustrate methods to parallelize their algorithm. Other work using genetic algorithms for graph matching includes [152], [224], [21], [161].

Some solutions to the graph matching problem are based on probabilistic formulations. The first pieces of work applying probability theory to graph matching [82], [98] use an iterative approach and probabilistic relaxation with second-order relations that assume a Gaussian error. Later both unary and pairwise relations were taken into account in a Bayesian formulation [36], [76], [223], [224]. A comparative study is presented in [139] between different discrete search strategies using the Bayesian consistency measure defined in [223], [224]. Probabilistic formulations can be found in other works such as [156] on modeling human mental representation of objects and [129] on recognizing Chinese characters. Probabilistic relaxation techniques for solving graph matching based on Bayesian formulations can also be found in [36], [224], [26], [185], [194], [216].

Applying the EM algorithm to graph matching is another important approach [46], [66], [135], [136]. Oher approaches use decision trees [147], neural networks [138], [171], [170], [114], [6], [102], [201], [202], different clustering techniques [31], [4], [58], [113] and [218], simulated annealing [84], tabu search [139], evolutionary game theory [160], tree search with backtracking [217], or spectral techniques [218], [181], [182], [186], [190].

1.2.4 Recent Algorithms in Computer Vision

Most previous work on graph matching proposed graph models with relatively weak unary and pairwise attributes, missing the opportunity to include as much relevant information as possible, such as discriminative local features or powerful geometric relationships beyond the simple weights or pair-wise distances on edges. The similarity measures, such as the graph edit distance or counting the number of consistent subgraphs, are computationally complex and often lack a clear, intuitive insight into geometric matching (such as matching of shapes or object models). The optimization techniques proposed were not efficient for matching large graphs, and they were usually based on a specific problem formulation, so that there was no one algorithm that could be efficiently applied to a wide range of different graph matching applications. More recent algorithms are based on the more general integer quadratic formulation presented in this thesis and can be applied to almost all types of graph matching problems. In computer vision, the first graph matching formulations based on integer quadratic programming include the work of Maciel and Costeira [141], Berg

et al [15] and our work [117]. We propose an efficient algorithm, spectral matching [117], which is specifically designed for matching using general unary and pair-wise constraints (for approximately solving Equation 5.1). This method is one of the core subjects of this thesis. We optimally solve a relaxed version of the original problem (Equation 5.1), based on the assumption that the correct assignments are likely to form strong pair-wise scores, whereas wrong assignments form such strong scores only accidentally. Later, Cour and Shi proposed an algorithm similar to ours, spectral matching with affine constraints [43], also using unary and pairwise terms. They modify the original quadratic score, while imposing the affine constraints during optimization. More recently, the problem of hyper-graph matching was also addressed, by including higher order terms (beyond pairwise), in [233] and [53], which is a generalization of our work on spectral graph matching [117] presented in this thesis.

1.3 Spectral Graph Matching

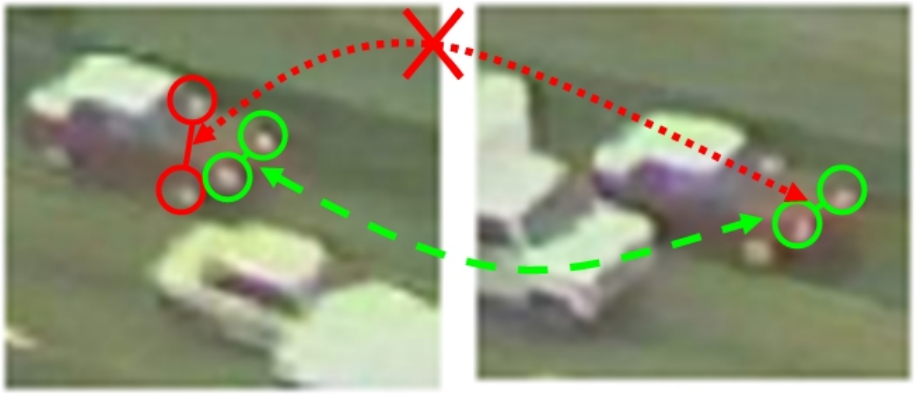

Figure 1.1: Pairs of wrong correspondences (red) are unlikely to preserve geometry, thus having a low pair-wise score. Pairs or correct assignments will preserve geometry and have a high pair-wise score. Second-order geometric relationships are often more powerful than unary terms based on local appearance. In this example correct matching is not possible by considering only local information. The pairwise geometry is preserved only by the correct matches, thus using such type of second order geometric information is encouraged for robust matching.

Here we briefly present our spectral matching algorithm designed to give an approximate solution to the graph matching problem. This algorithm can be used for a wide variety of applications because it handles the graph matching problem in its most general formulation, as an Integer Quadratic Program, and it does not impose a specific mathematical formulation on the unary and second order terms. The only constraint we require is

that the unary and pair-wise scores should be non-negative and they should increase with the quality of the match, that is, decrease with the deformation errors between the candidate correspondences. These scores can capture any type of deformations/changes/errors at the levels of both appearance as well as geometric relationships. Our key insight into the graph matching problem, as applied to computer vision, is that the correct assignments will have strong, large valued second order scores between them, while such large valued scores between incorrect ones are unlikely because accidental geometric alignments are very rare events. These probabilistic properties of the second-order scores give the match matrix $\mathbf{M}$, containing the second-order terms/scores, a very specific structure: a strong block with large values formed by the correct assignments, and mostly zero values everywhere else. This allows us to drop the integer constraints on the solution during the optimization step and impose them only after, as a binarization procedure applied to the leading eigenvector of this matrix $\mathbf{M}$. In Figure 1.1 we present an example that illustrates the intuition behind our algorithm. The images of the red car have a very low resolution. The local appearance of each feature is therefore not discriminative enough for reliable matching between two consecutive frames. However, the pairwise geometry is well preserved between frames, so it is very likely that the pairs of assignments that preserve this geometry will be correct. This is just an illustrative example, but using pair-wise geometric constraints for robust matching is suitable for a lot of computer vision applications.

We can think of $\mathbf{M}$ as the weighted adjacency matrix of the graph of candidate assignments. Each candidate assignment (or correspondence) (i, a) can be seen as a node in this graph, containing information regarding how well the local appearance between the candidate matches agrees. The links between the nodes can contain information regarding how well the pair-wise geometric information between candidate assignments is preserved. Figure 1.2 shows a possible instance of such a graph of candidate assignments. Larger nodes correspond to stronger unary scores (better agreements at the level of local appearance) and thicker edges correspond to stronger pair-wise scores reflecting stronger agreements at the second-order, geometric level. We expect that the correct assignments will form a strongly connected cluster, which can be found by analyzing the leading eigenvector of the weighted adjacency matrix $\mathbf{M}$ of the assignments graph. The elements of the eigenvector can be interpreted as confidences that each candidate assignment is correct, and the integer constraints can be applied to binarize the eigenvector and get an approximate solution to the graph matching problem.

In Figure 1.3 we show the likely structure of the matrix $\mathbf{M}$. The correct assignments will form a strong block in this matrix with large pairwise elements, while the pairwise scores between incorrect assignments will be mostly zero. This will be reflected in the leading eigenvector of $\mathbf{M}$.

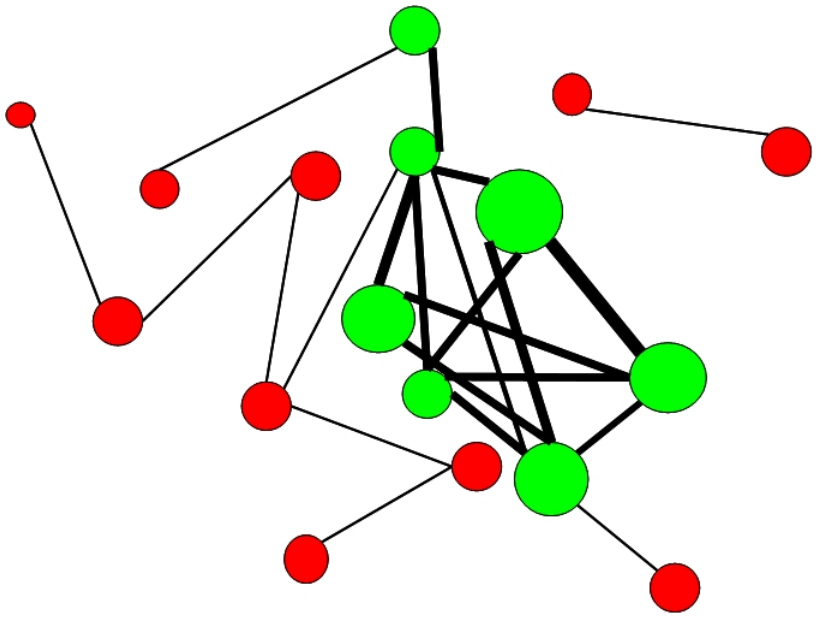

Figure 1.2: Correct assignments (shown in green) are likely to form a strong cluster by establishing stronger pairwise second-order agreements (many more thicker edges) and preserving better the local appearance of features (larger nodes).

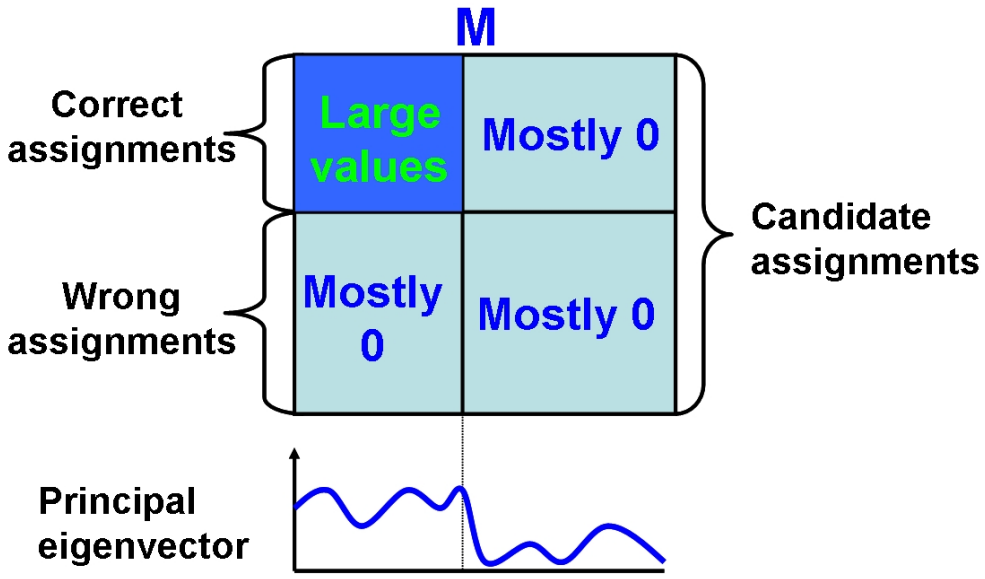

Figure 1.3: Correct assignments are likely to form a strong block with large values in the matching matrix.

1.4 Algorithmic Variations and Extensions to Spectral Matching

1.4.1 Spectral Matching with Affine Constraints

As discussed previously the graph matching problem can be formulated as an Integer Quadratic Program (IQP): maximize $\mathbf{x^T Mx}$ given the affine constraints (one-to-one map-

ping constraints) $\mathbf{Ax} = \mathbf{b}$ and the binary constraints on $\mathbf{x} = \{0, 1\}^n$, where n is the number of candidate correspondences. Cour and Shi [43] propose Spectral Matching with Affine Constraints (SMAC) that is closely related to our spectral matching algorithm. While we drop all mapping constraints during the optimization step, they propose a formulation that optimizes the modified objective function $\mathbf{x^T Mx}/\mathbf{x^T x}$, while keeping the affine constraints and dropping only the binary constraints on $\mathbf{x}$. Their proposed solution is given by the leading eigenpair of $\mathbf{P_A M P_A x} = \lambda \mathbf{x}$, where $\mathbf{x}$ is scaled so that $\mathbf{Ax} = \mathbf{b}$ exactly, $\mathbf{P_A} = \mathbf{I_n} - \mathbf{A_{eq}^T}(\mathbf{A_{eq}A_{eq}^T})^{-1}\mathbf{A_{eq}}$, $\mathbf{A_{eq}} = [\mathbf{I_{k_1}}, \mathbf{0}](\mathbf{A} - (\mathbf{1}/\mathbf{b_k})\mathbf{bA_k})$, and $\mathbf{A_k}, \mathbf{b_k}$ denote the last row of $\mathbf{A}, \mathbf{b}$ and k = number of constraints. An important aspect is that $\mathbf{P_A}$ is in general a full matrix even when $\mathbf{M}$ is sparse, so special care has to be taken in the actual implementation by using Sherman-Morrison formula (for one-to-one matching) for operations of the type $\mathbf{y} = \mathbf{P_A x}$. In practice, as shown in Chapter 5 and in [43], spectral matching and SMAC perform similarly, while SMAC, as expected, is more computationally expensive (also shown in [43]) and more difficult to implement due to the non-trivial construction of the matrix $\mathbf{P_A}$. Moreover, the learning algorithm we propose in Chapter 5 seems to be more effective for spectral matching than for SMAC. Also, the binarization algorithm (IPFP) we propose in Chapter 4 efficiently raises the performance of both algorithms to the same level.

1.4.2 Higher Order Spectral Matching

Inspired from our work, Duchenne et al. [53] go beyond second-order constraints and propose a generalization of spectral matching to higher order relationships by using a generalization of the power method for tensors. Instead of the matrix $\mathbf{M}$, they propose building a sparse tensor $\mathbf{H}$ whose elements represent the similarities of higher order relationships among tuples of features. In the case of third order relationships, for example, $H_{ia;jb;kc}$ could describe how well the geometric relationships among the features (i, j, k) is preserved after matching them to features (a, b, c). While using higher order scores is computationally more expensive than using second order ones, the payoff consists of being able to capture relationships that are invariant to similarity, affine or even perspective transformation. Our work using second-order interactions [121] can be immediately extended to perspective invariant third-order relationships (fact that was also acknowledged by [53]), which is indeed very appealing given that most deformations occurring in 2D matching are due to perspective transformations. The computational bottleneck in this case could be addressed by allowing non-zero third-order scores only among a subset of all eligible triples of candidate assignments. This subset could be chosen at random or it could be based on some proximity constraints, without necessarily damaging the matching performance. In the case of third-order potentials the matching score is defined as:

$$S = \sum_{ia;jb;kc} H_{ia;jb;kc} x_{ia} x_{jb} x_{kc} \tag{1.2}$$

The tensor power iteration for finding the leading eigenvector of $\mathbf{H}$ is an immediate extension of the matrix power iteration algorithm:

1. $v_{ia} \leftarrow \sum_{jb;kc} H_{ia;jb;kc} v_{jb} v_{kc}$
2. $\mathbf{v} \leftarrow \mathbf{v}/\|\mathbf{v}\|$

In Chapter 5, in addition to our learning method for graph matching using second-order scores, we propose an extension to the tensor method of Duchenne et al. [53] for high order graph matching.

1.4.3 Spectral MAP Inference

Efficient methods for MAP inference in graphical models are of major interest in pattern recognition, computer vision and machine learning. Similar to the graph matching problem, finding the MAP solution for MRFs and DRFs [107] often reduces to optimizing an energy function that depends on how well the labels agree with the data (first-order potentials) as well as on how well pairs of labels at connected sites agree with each other given the data (second order potentials). From this point of view the only difference between graph matching and the MAP problem are the mapping constraints. Graph matching solutions usually obey one-to-one constraints while the MAP Inference solutions many-to-one. Therefore, it is expected that the MAP and graph matching problems could be often solved by similar algorithms. In Chapter 3 we propose an approximate solution to the MAP problem that is an extension of the spectral matching algorithm to finding the approximate graph matching solution. Moreover, in Chapter 4 we present a meta-algorithm that can be efficiently applied to both graph matching and MAP inference problems. Here we briefly discuss the spectral MAP inference algorithm from Chapter 3. The method has two stages. In the first stage we follow a path similar to the power method for finding the leading eigenvector of a matrix and reach an approximate solution that obeys a certain optimality bound. In the second stage we follow a climbing path that increases the quadratic score at every step and converges to a binary solution respecting the many-to-one mapping constraints. In our experiments we show that our method significantly outperforms popular algorithms such as Loopy Belief Propagation on energy functions of arbitrary graphs with arbitrary number of labels (see Figure 1.4 and Chapter 3 for more details)

In Chapter 3, we formulate the MAP problem as an Integer Quadratic Program:

$$E = 1/2 \sum_{ia;jb} x_{ia} x_{jb} Q_{ia;jb} + \sum_{ia} x_{ia} D_{ia}, \tag{1.3}$$

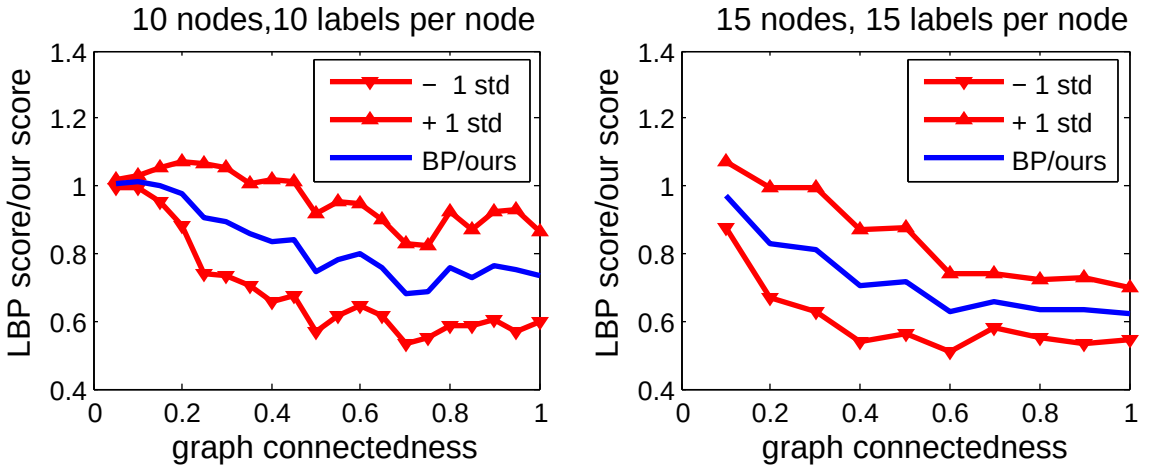

Figure 1.4: Comparison with Loopy Belief Propagation. Mean and std values for the relative scores of Loopy BP against ours E_{LBP}/E_{ours} for varying degree of connectedness, over 30 experiments.

where $Q_{ia;jb}$ corresponds to the higher order terms describing how well the label a at site i agrees with the label b at site j, given the data. For each pair of site i and its possible label a, the first order potentials are represented by D_{ia}, which in general describe how well a label a agrees with the data at site i. As in the graph matching formulation $\mathbf{x}$ is required to be an indicator vector with an entry for each pair of $(site, label)$, such that if $x_{ia} = 1$ then site i is assigned label a and $x_{ia} = 0$ otherwise.

The algorithm outline is:

1. Stage 1: find $\mathbf{x}^*$ that maximizes $1/2\mathbf{x^T Q x} + \mathbf{Dx}$, given $\sum_a x_{ia}^2 = 1$ by iterating until convergence:
   a) let $\mathbf{x} \leftarrow \mathbf{Qx} + \mathbf{D}$
   b) normalize $\mathbf{x}$ per site: $x_{ia} = \frac{x_{ia}}{\sqrt{\sum_b x_{ib}^2}}$
2. Initialize $\mathbf{x}$, such that $x_{ia} = x_{ia}^* / \sum_b x_{ib}^*$
3. Stage 2: Set $\beta = 1$ and repeat until convergence
   a) set $v_{ia} = \sum_{jb} Q_{ia;jb} x_{jb}^{(t)} + D_{ia}$
   b) $x_{ia}^{(t+1)} = \sigma_i x_{ia}^{(t)} F(v_{ia}, \beta)$, where $\sigma_i = 1/\sum_b x_{ib}^{(t)} F(v_{ib}, \beta)$
   c) increase β after updating $\mathbf{x}$ for all sites

Here $F(v, \beta)$ is any positive, monotonically increasing function of v. Regardless of the structure of the graph or the unary and pairwise terms, this algorithm converges to a

discrete solution $\mathbf{x}$ that obeys a certain optimality bound. These are desirable properties that are not met by most commonly used algorithms, such as Loopy Belief Propagation or Iterated Conditional Modes [17]. See Chapter 3 for the theoretical details and experimental evaluation.

1.4.4 Spectral MAP Inference with Affine Constraints

Cour and Shi [41] proposed a quadratic programming relaxation to the MAP problem that is closely related to ours. Instead of relaxing the constraint on the solution from L1 to L2, they keep the original L1 constraint and instead modify the objective function. Thus, they optimize

$$E = \frac{1/2\mathbf{x^T Q x} + \mathbf{D^T x}}{\mathbf{x^T x}}, \tag{1.4}$$

given the original affine constraints $\sum_{ia} \mathbf{x_{ia}} = 1$ and dropping the integer constraints. The solution can be reduced to an eigenvector computation. Intuitively the normalization $\mathbf{x^T x}$ encourages flatter solutions, but it is not clear whether that will give better solutions after discretization. They derive data independent and data dependent optimality bounds that are almost identical to ours, and obtain a discrete solution by following the same climbing procedure that we proposed. Their experiments, performed on synthetic data generated in the same way as ours [118], show that the two methods perform almost identically. As in the case of their graph matching algorithm, the implementation of their method is tricky (see [41] for details) because it involves using a full matrix the same size as $\mathbf{Q}$, similar to their SMAC algorithm for graph matching [43].

1.5 Computer Vision Tasks using Spectral Matching

Spectral matching brings a few improvements over previous methods, which already made it quite popular for solving computer vision tasks. It has wide applicability because it does not impose any constraints on the type of unary and pairwise scores. Indeed, it requires the scores to be non-negative and increase with the quality of the agreement, but that can be easily accommodated by all graph matching applications. It is very efficient, as it drops the integer constraints on the solution, so it lowers the complexity of the problem from NP-hard to a low-order polynomial complexity by computing the leading eigenvector of the match matrix $\mathbf{M}$. It is also very intuitive, easy to understand and implement. Its solution is based on an intuitive insight into the structure of the match matrix $\mathbf{M}$, based on the expected geometric alignments between correct assignments against the accidental alignments of incorrect ones. This idea inspires the appropriate design of meaningful first and second-order scores. These properties contribute to the popularity of spectral matching,

as the preferred choice for many computer vision applications. Next we present some of the most representative work using our algorithm.

1.5.1 Discovering Texture Regularity

Texture analysis is a fundamental problem in computer vision [74], [92], [122], [69], [131], [179], [215]. Understanding texture regularity in real images is a challenging computer vision task, because texture is inherently a global process and texels, as units of texture, exist only when they are repeated, often in deformed, noisy versions. In order to analyze near-regular textures and discover their local texels as repeated patterns, we need to find their global structure. We can think of discovering texture regularity as a problem of finding slightly deformed translational symmetries. In our previous work [83] we propose to formulate the discovery of lattices of near regular textures as a correspondence problem using the spectral matching algorithm to find assignments that maximize both the appearance similarity of texels as well as their symmetric geometric layout, by enforcing second order geometric consistency. We discover both the texels as well as the texture lattice at the same time. The higher-order consistency made possible by spectral matching drastically reduces the need for a priori estimates of scale and make the assignment procedure robust to outliers. The algorithm finds a plausible lattice by iteratively proposing texels and assigning neighbors between the texels. This basic framework of using higher-order matching to discover texture regularity is applicable to any set of visual features and any problem that involves the discovery of translational symmetries with distortion.

The same idea, with minimal modification, can also be used for the discovery of rotational and reflective symmetries. In Figure 1.8 we present a few results from our preliminary work with James Hays and Yanxi Liu on finding rotational symmetries. The main idea, as in the case of finding texture regularity, is that one can formulate symmetry discovery as matching a set of features to a rotated, translated or reflected version of itself. If the geometric pairwise scores are invariant to rotation, reflection or translation, such as it is the case with pairwise distances, then the same graph matching formulation can be used for finding all these types of symmetry.

1.5.2 Object Discovery

Despite a lot of recent interest, learning from unlabelled data still remains one of the most challenging problems in the fields of computer vision and machine learning. In our work [116] we present a method for learning in an unsupervised manner object models from low resolution video, as constellations of features that tend to co-occur in geometrically consistent configurations. The literature most related to our work includes Zhaozheng and Collins object modeling from tracking [229], Ramanan and Forsyth paper on learning object

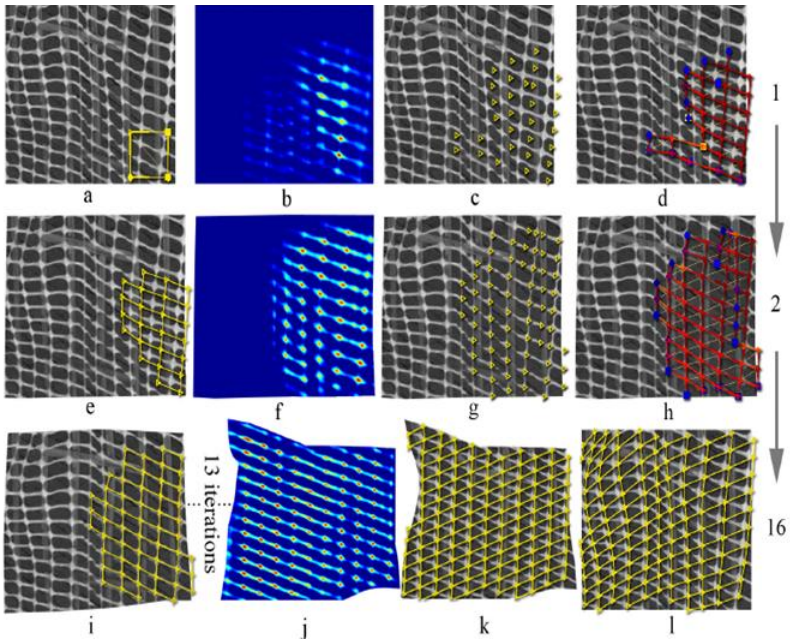

Figure 1.5: The stages of our algorithm, one iteration per row. (a) is the input image, (l) is the final result. The leftmost column shows (potentially warped) input images with the current, refined lattice overlaid. The second column is the correlation map, calculated from valid texels, used to propose the potential texels in (c) and (g). (d) and (h) show assignments made from these potential texels before they are refined into the lattices of (e) and (i), respectively. (k) shows the lattice with the highest a-score after 16 iterations, and (l) is that lattice mapped back to input image coordinates. The input image (a) is a challenge posed by David Martin at the Lotus Hill workshop, 2005. [BEST SEEN IN COLOR].

models from tracking [164], Sivic and Zisserman's work on discovering objects from video sequences [193], Kubica's work in data-mining for discovering groups of people based on co-occurrences [103], [104] and Parikh's on discovering hierarchies of objects [159].

In our work the input images typically contain single or multiple objects that change in pose, scale and degree of occlusion. Also, the objects can move significantly between consecutive frames, which is why the same method can work with collections of images that do not necessarily come from video. The content of an input sequence is unlabeled so the learner has to cluster the data based on the data's implicit coherence over time and space. Our approach takes advantage of the dependent pairwise co-occurrences of objects' features within local neighborhoods vs. the independent behavior of unrelated features. We couple or decouple pairs of features based on the probability of their co-occurrence over the sequence and extract the objects as connected components of features. In order to compute these co-occurrences we have to be able to match correctly the same object part/feature

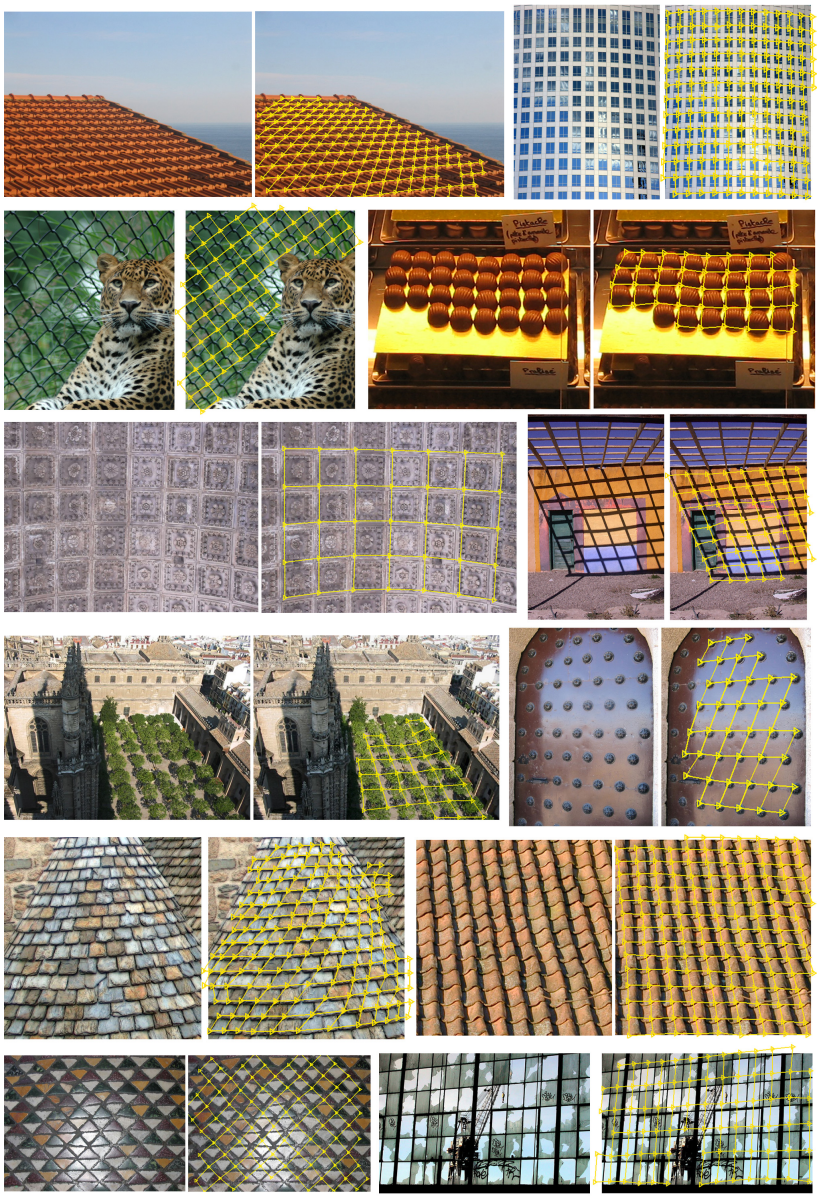

Figure 1.6: Results: pairs of original images (left) and near-regular textures found overlaid (right).

across several frames. Since these frames are from low resolution video using only local appearance is not enough for finding correct correspondences consistently. We therefore

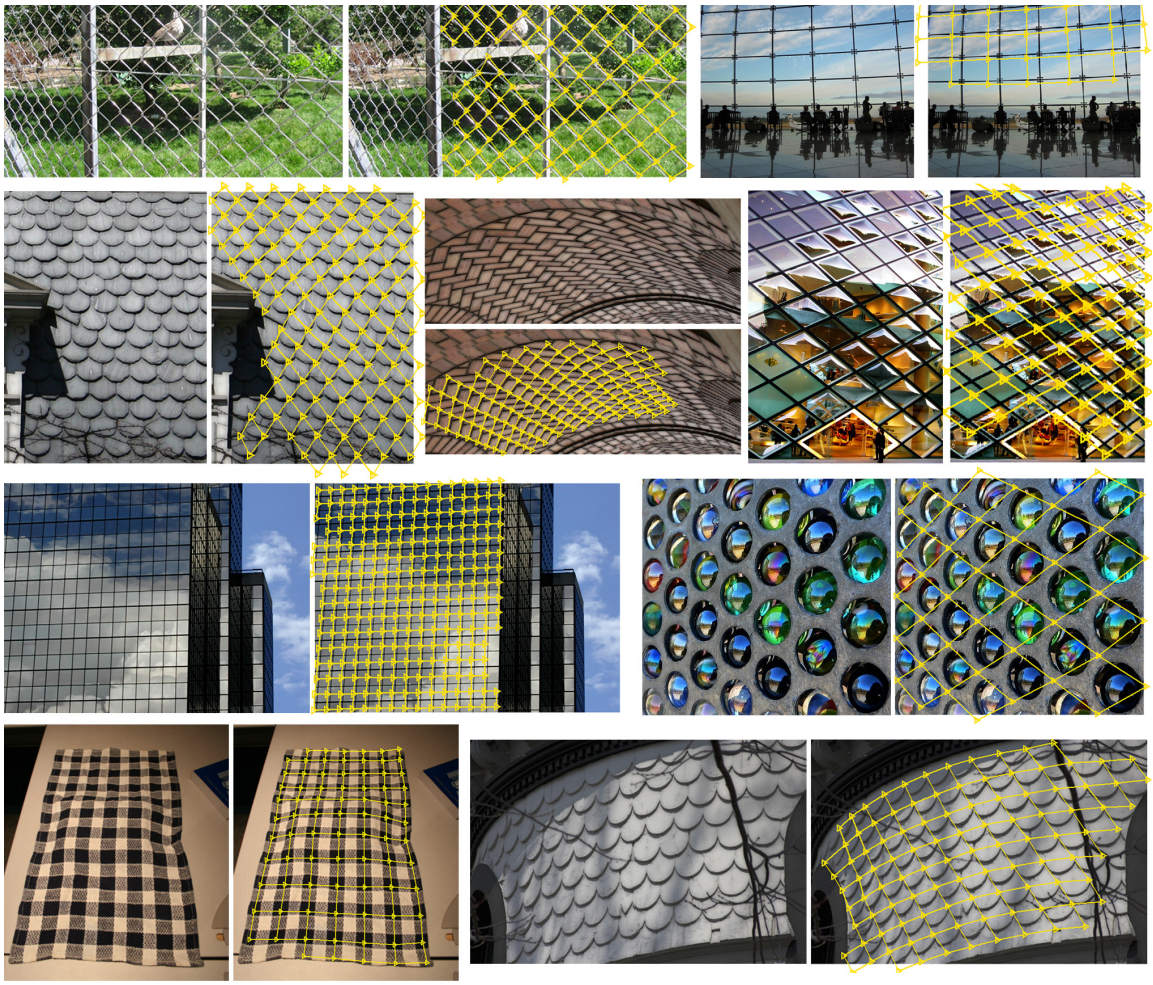

Figure 1.7: Results: pairs of original images (left) and near-regular textures found overlaid (right).

use spectral matching to preserve both local appearance (using SIFT descriptors) as well as pair-wise geometric relationships: the unary terms account for similarity in the SIFT descriptors between the features to be matched, and the pair-wise terms for the geometric deformations between pairs of features (which take in consideration the pairwise distances and the local orientations of the SIFT descriptors).

The main idea is that pairs of features belonging to the same object co-occur often and their pairwise geometry is preserved over the video sequence. In contrast, pairs of features belonging to different objects do not co-occur often and when they co-occur their pairwise geometry suffers large deformations (see Figures 1.9 and 1.10).

We build a weighted graph with each node corresponding to a part that was matched

Figure 1.8: Preliminary results on discovering rotational symmetry. The blue points/regions show the centers of potential rotational symmetries. The red lines correspond to matches of features that are rotationally symmetric. Note that even when not all features are matched correctly, the center of rotation is correctly found, even in nature (flowers) or in the presence of affine distortion.

over the video sequence and the weights on edges corresponding to the co-occurrences between pairs of parts. The co-occurrences are counted only when the pairwise geometry is also preserved and their actual value w_{AB} for any two parts A and B is $w_{AB} = \frac{n_{AB}}{n_A + n_B - n_{AB}}$, where n_A is the number of occurrences of part A, and n_{AB} is the number of times A and

Figure 1.9: Pair of unrelated parts. The circles indicate the scale and positions of the two parts; the corners of the quadrilaterals are the 4 points associated with each pair. These pair co-occurred only 3 times out of 20 but most unrelated pairs co-occur for less than 3 times. Also notice how their pairwise geometry slowly departs from their initial one.

B co-occurred together. In Figure 1.11 we show the normalized histograms, over 25 video sequences, of the co-occurrences weights of parts belonging to the same object against parts belonging to different objects. Since the histograms are significantly different, it becomes clear that these weights are appropriate for clustering the features into groups that belong to the same object. In Figure 1.12 we display some results. We show that in this manner features belonging to different aspects of the same object can be grouped together.

A very similar method for the discovery of objects was later proposed by Parikh and Chen [158], [159]. They also use spectral matching for finding consistent correspondences across image sets, and discover objects as clusters of features that co-occur in a similar geometric configuration. The main difference between their work and ours is that instead of using the pair-wise co-occurrence between features for clustering, they use the pair-wise correlation of features' positions. They claim that co-occurrences are not enough for discovering multiple objects, but in our work we actually used more sophisticated pairwise geometric relationships to account only for geometrically consistent co-occurrences, which allows us, as shown in our experiments, to discover multiple objects from the same video sequence. In contrast, the pairwise correlations between features' locations assumes a Gaussian distribution, which is in fact more limited than ours.

Figure 1.10: Pairs of parts that belong to the same object tend to co-occur most of the time and preserve their pair-wise geometry. These two parts were detected together in 16 frames out of 20.

1.5.3 Unsupervised Modeling and Recognition of Object Categories

Unsupervised learning of object categories is one of the most challenging high-level task in computer vision. It has received a lot of recent attention, along with the growing overlap between computer vision and machine learning [109], [73], [62], [78], [192], [209]. In [127] Liu and Chen propose a framework that combines the topic model of pLSA with geometrically consistent correspondences found by spectral matching. For the pairwise scores they use both local appearance and pairwise geometric relationships. The geometrically consistent correspondences are used to create a reward map $P(r/z, d)$ by counting the number of matches of each part. It is expected that the parts belonging to the foreground object have more matches. The reward map is then used to augment the graphical model of pLSA (Figure 1.13). This approach leads to a significant improvement over previous methods

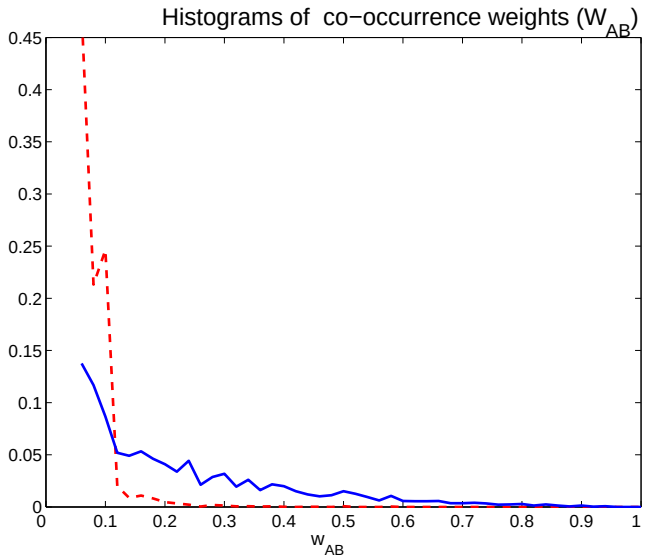

Figure 1.11: Normalized histograms of co-occurrence weights for pairs that belong to the same object (blue solid line) and pairs that belong to different objects (red dashed line) over 25 video sequences. For values above 0.15, the pair of parts belongs to the same object with very high probability.

[126], [192].

Recently, Kim et al. [97], [96] proposed two methods for this task, which combine spectral matching with link analysis techniques. Their work distinguishes itself for several important reasons: firstly, it introduces link analysis techniques usually used for ranking and clustering of web-pages to the vision problem of unsupervised learning of categories. Secondly, it uses a matching algorithm only to form soft links between the original features, thus avoiding strict, hard matchings between them that could introduce errors early on in the modeling process. Thirdly, it naturally combines geometric relationships between features with the traditional bag-of-words approaches, such as LDA and pLSA, bringing a significant improvement in performance.

Spectral matching is used for building a Visual Similarity Network, by establishing links between features, whose weights reflect the geometric consistency of the matches: strong links are formed between matched features that are geometrically consistent in a pairwise sense with other matched features. Instead of limiting itself to one-to-one matching, as in the traditional graph matching formulation, they allow many-to-many matches by iteratively repeating our original greedy binarizing procedure of the principal eigenvector. Their pairwise geometric scores are also taken from our previous work [121]. Both the use of such pairwise geometric scores as well as the many-to-many mapping constraints make

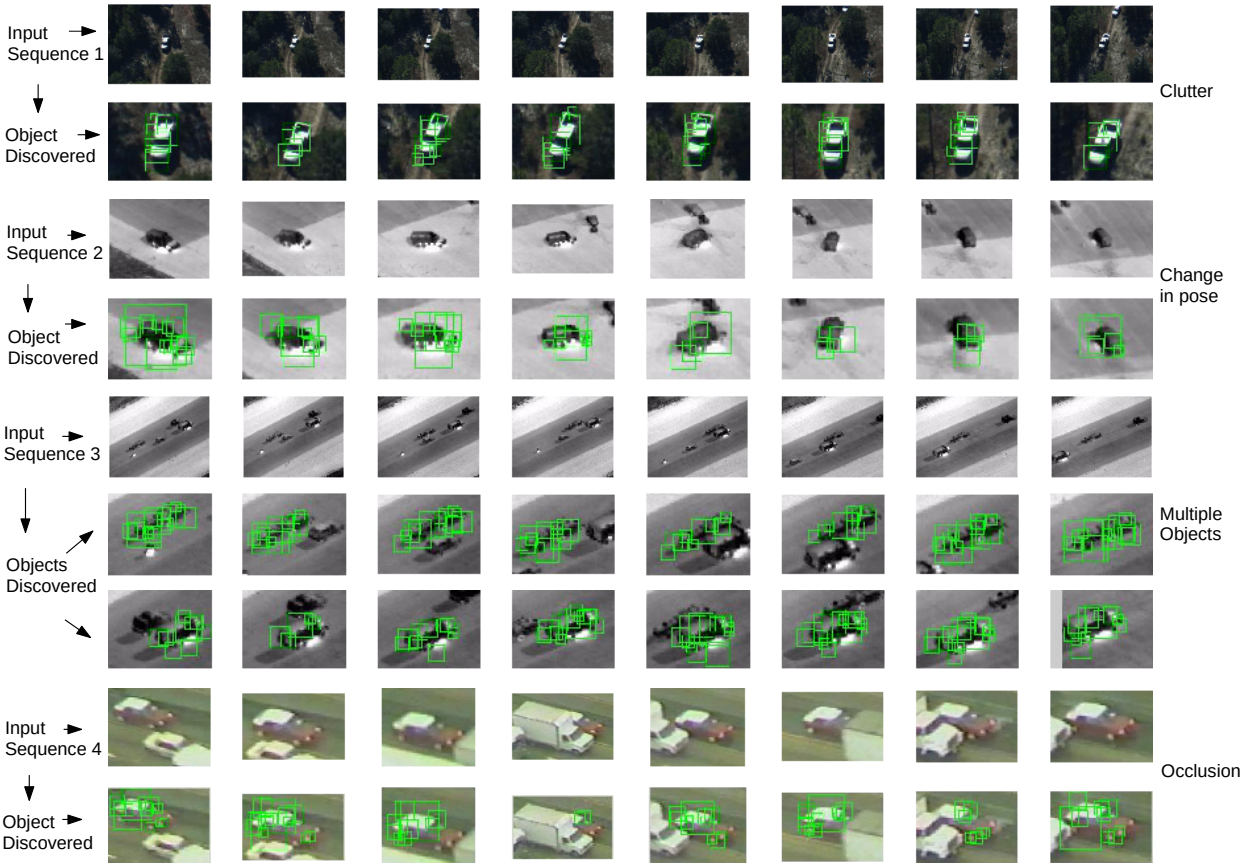

Figure 1.12: Results on different image sequences. Objects are discovered in an unsupervised manner.

our spectral matching algorithm the preferred choice for their system.

1.5.4 Recognizing Actions from Video Sequences

Link analysis techniques, previously proposed by Kim et al. [97] for unsupervised learning of object categories, were later used by Liu et. al [128] on a seemingly unrelated task: recognizing actions from unconstrained video-sequences such as the ones posted on YouTube. This is a difficult problem due to the significant intra-class variations, occlusion, and background clutter, which is why there is very little work on recognizing actions from videos captured under uncontrolled conditions. The most relevant literature on action recognition includes [222], [157], [130], [75], [228], [50], [111], [112], [154], [59], [230], [94].

In [128] the authors use Adaboost for classifying actions, using static and motion features. The static features are extracted with three interest point detectors (Harris-

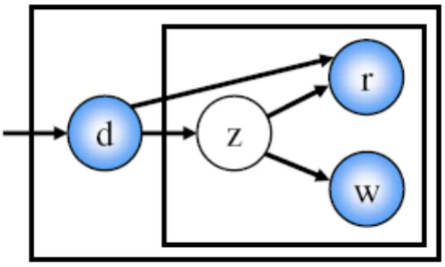

Figure 1.13: The graphical model. The outer plate indicates the graph is replicated for each image, and the inner plate indicates the replication of the graph for each patch. The topic variable z is hidden. Figure reprinted from [127].

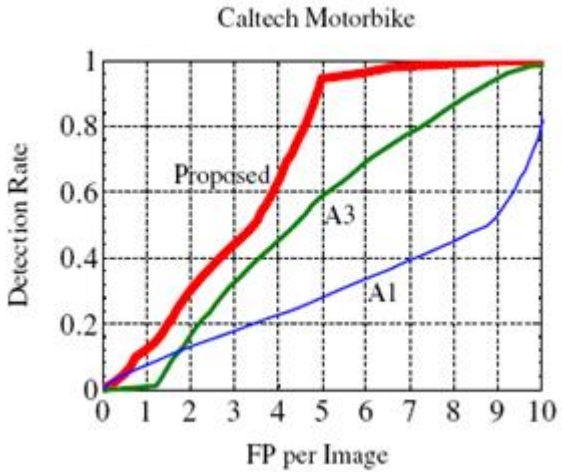

Figure 1.14: Patch-level classification results on Caltech motorbike data set. A1 is the method of [192] and A2 is [126]. For each method, its top 20% confident patches are classified as foreground versus background; the closer the posterior probability $P(z|d, r, w)$ (or $P(z|d, w)$ in A1) of a patch is to 0 or 1, the higher the confidence of the patch. Figure reprinted from [127].

Laplacian, Hessian-Laplacian and MSER). They are described by location, scale and the 128-dimensional SIFT descriptor. A feature similarity network is built from each video sequence using spectral matching, similar to the Visual Similarity Network of [97]. Spectral matching is again used to ensure that the features matched preserve both the global geometry as well as the local appearance. Links are established between matched features with

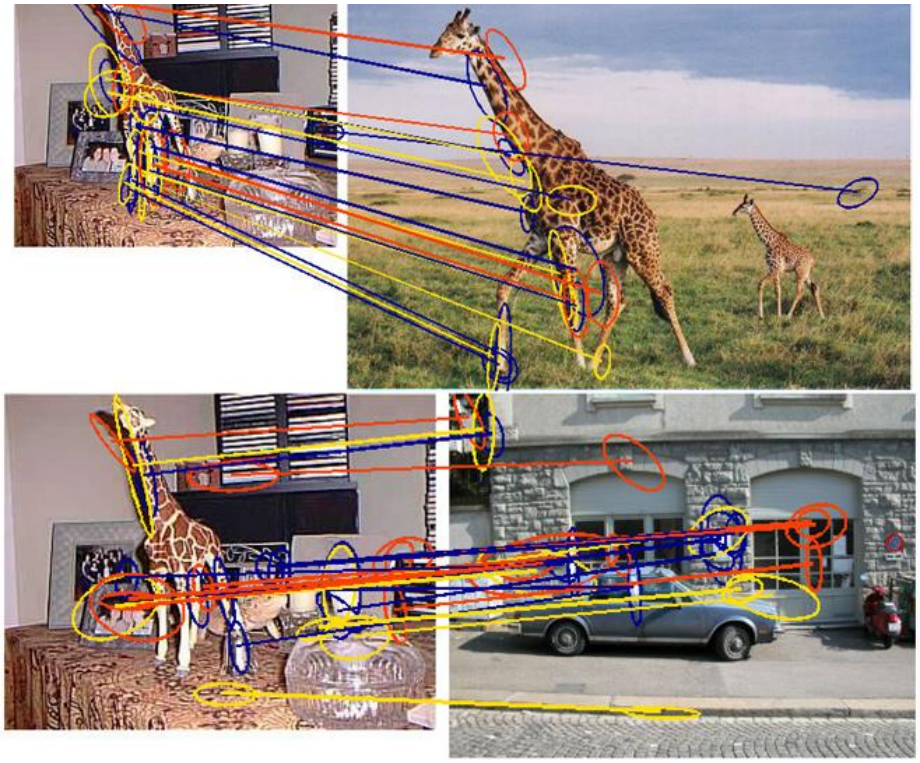

Figure 1.15: An example of link generation on two pairs of images. The features are matched by using a liberal spectral matcher. The jet colormap is used from red (strong) to blue (weak geometric consistency). Figure reprinted from [97].

weights measuring their geometric consistency, also similar to [97]. Then (again following [97]), PageRank is used for feature pruning by keeping only those with high PR rank (Figure 1.17). The surviving features are then used for building visual vocabularies, ultimately combined with AdaBoost for action recognition.

1.5.5 Matching 2D Shapes and Aspects

In our work on graph matching we emphasize the importance of second-order constraints [117], [121], [119], [120]. We show that geometric second-order constraints can be more powerful for matching than unary, appearance constraints. In tasks such as matching shapes, silhouettes, 2D aspects or actions, the second-order geometric relationships are often more discriminative than the local appearance. For example, in the work of Ren [169], aspects of people are matched using 2D lines. Lines are not discriminative individually, nevertheless, in pairs, their relative geometry can be very powerful for matching. Independently, but

Figure 1.16: Some examples of localization for the Caltech-101 and TUD/ETHZ dataset. In each image pair, the left image represents original extracted features with yellow, and the right image shows top 20% high-ranked features with color variance according to the importance weights. The jet colormap is used from red(high) to blue(low). Figure reprinted from [97].

similar to our work [121], Ren designed powerful second-order scores that captured geometric relationships between pairs of lines that considered relative distances, orientations, adjacency and parallelism. These pairwise scores were used in combination with spectral matching for recognizing 2D aspects of people (Figure 1.18).

In Chapter 7, we present our approach to shape matching using only second-order relationships, which are based on an overcomplete set of pairwise deformations considering distances and angles that describe the geometric relationships between oriented points. We show how shape matching can be used for recognition of object categories, bringing a significant improvement over bag-of-words methods that use only local appearance.

Figure 1.17: Two examples from riding (top) and cycling (bottom) demonstrate the effects of feature acquisition. The first row shows the original static features, and the second row shows the selected features. The top 10% features in PageRank values are retrieved. Figure reprinted from [128].

1.5.6 Graph Matching for 3D data

Matching 3D data is important for a lot of vision applications, such as building realistic 3D models for virtual environments used in entertainment, making computer games and CGI movies, building 3D maps of cities or cultural heritage projects. In our previous work [198], [5] we matched 3D lines (Figure 1.20) extracted from 3D scans in order to align the scans and build 3D models of buildings [199] (Figure 1.19). In that work we used a matching algorithm that performed an efficient search over all possible assignments followed by a verification step. Later we also applied spectral matching and obtained the same matching

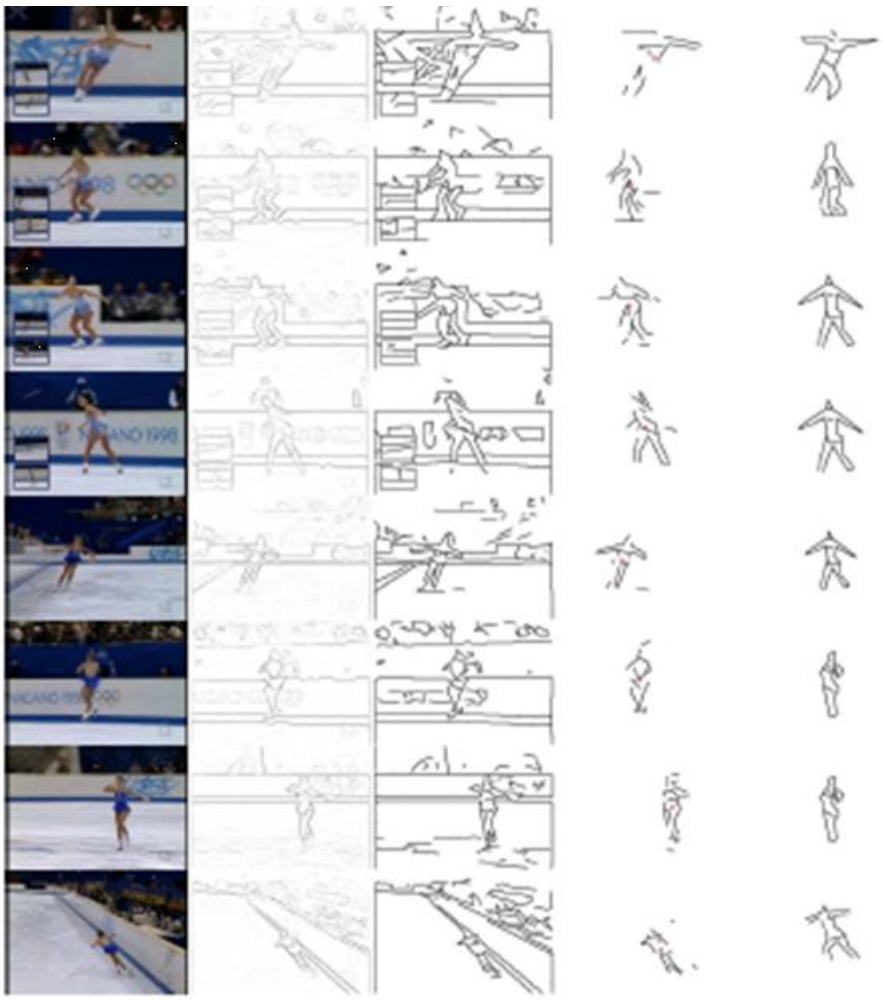

Figure 1.18: Pairwise line aspect matching results on two skating sequences. Matching is purely edge-based and done independently in each frame, no motion or temporal coherence used. In 5 columns we show the original image, the boundary map computed, the line approximation of boundaries (input to our matching algorithm), matched lines with the estimated center, and the best matched aspect. Figure reprinted from [169].

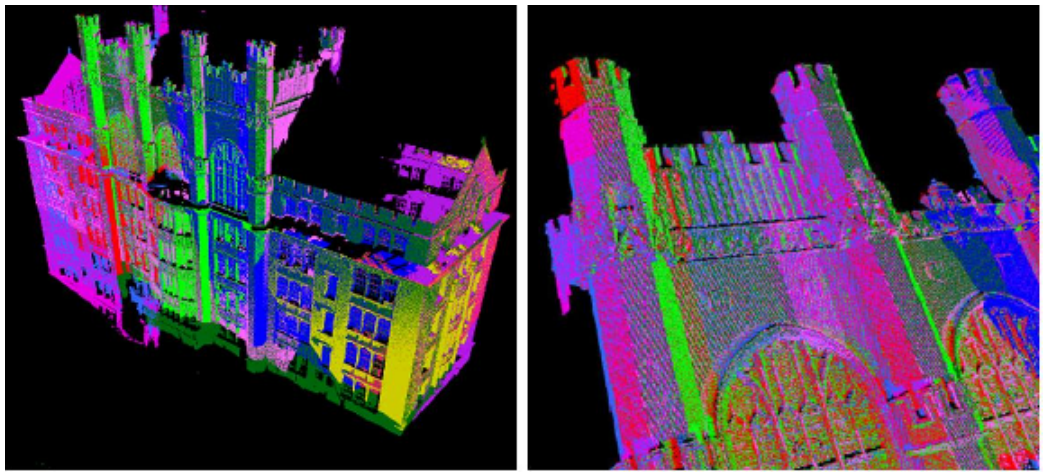

Figure 1.19: Ten different 3D scans (in different colors) of a campus building matched and shown in the same coordinate system (left). Alignment detail (right). Figure reprinted from [198].

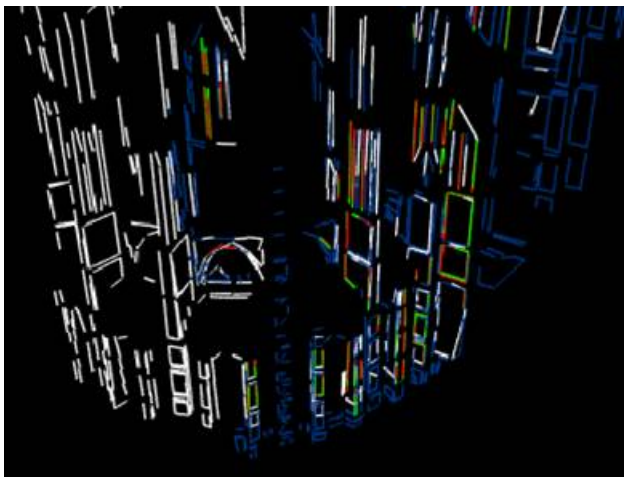

Figure 1.20: Lines extracted from two different 3D scans. The matched lines are shown in green and red. Figure reprinted from [198].

results between sets of 3D lines. The advantage of using spectral matching in this case is its reduced complexity and simpler formulation against the search method we proposed in [198]. In this case, for the quadratic assignment formulation we used no unary terms and the pairwise geometric scores considered distances and angles between pairs of 3D lines. We believe that graph matching can be more robust in 3D than 2D matching problems, because

in 2D most deformations are due to the perspective transformation between the views of the same object, while in 3D there is usually only a rigid or piecewise rigid transformation of the object.

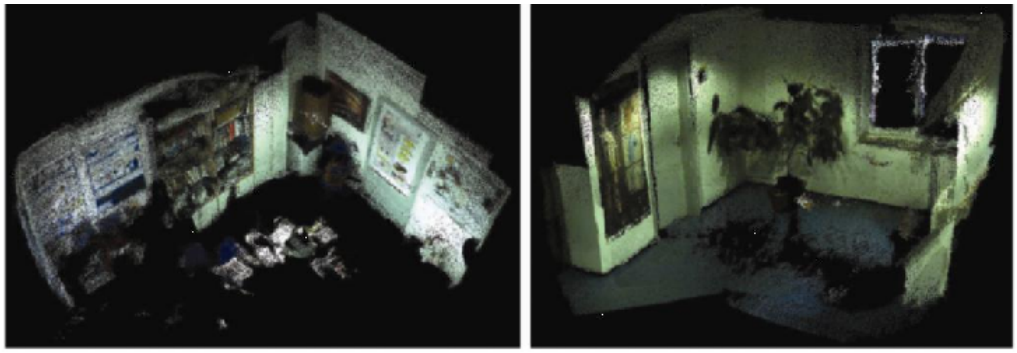

Figure 1.21: Scene acquisition by aligning different overlapping 3D scans. Spectral matching is used for finding consistent correspondences between keypoints extracted from each scan. Figure reprinted from [87].

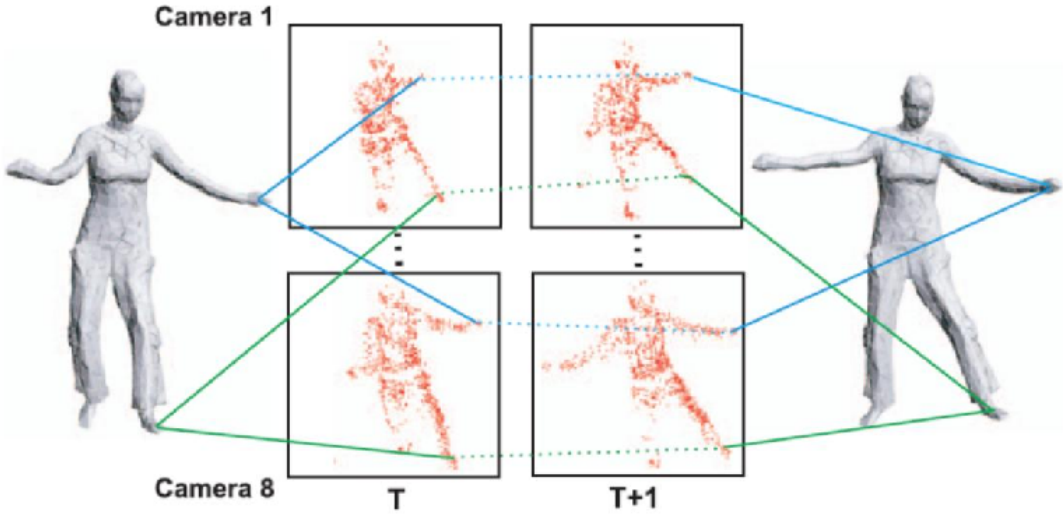

Figure 1.22: Using spectral matching to find correspondences between silhouettes from consecutive video frames. Figure reprinted from [48].

In related work [87] the authors combine the use of SIFT descriptors with pairwise distances between 3D points to match 3D scans for efficient scene acquisition (Figure 1.21).

From each scan a set of SIFT descriptors is extracted together with 3D positions, then used by spectral graph matching with both unary, appearance based and pairwise geometric constraints.

Figure 1.23: Performance capture results. A 3D model is built and deformed by tracking the performance of a human actor. Figure reprinted from [48].

Spectral matching was also successfully used for aligning human silhouettes between successive frames for capturing 3D human performance [48]. Correspondences were found between SIFT keypoints extracted from successive video frames capturing actors performing different actions. Then, a 3D model of the human was tracked by finding correspondences between the 2D silhouettes and the 3D model. This method showed increased efficiency over previous methods, with possible usage in making photo-realistic CGI movies (Figures 1.22 and 1.23).

1.6 Main Contributions

We briefly enumerate here the main contributions of this thesis:

1. Spectral Matching (Chapter 2): an efficient algorithm for graph matching, which, since its publication, has already been used successfully in several computer vision

applications such as object recognition, unsupervised discovery of object categories, action recognition, symmetry discovery and matching 3D scenes and objects. We also propose an extension of Spectral Matching to tensors, for matching tuples of graphs at the same time (instead of pairs of graphs).

2. Spectral MAP Inference (Chapter 3): an efficient MAP inference algorithm for graphical models inspired from spectral matching, which obeys certain optimality bounds and outperforms in our experiments popular methods such as Iterative Conditional Modes and Loopy Belief Propagation

3. Integer Projected Fixed Point algorithm for graph matching and MAP inference problems (Chapter 4): an efficient algorithm that can be applied to both graph matching and MAP inference. It significantly outperforms state-of-the-art algorithms. Moreover, it can be used as post-processing, discretization procedure, significantly improving the performance of other graph matching and MAP inference algorithms.

4. Unsupervised learning for graph matching (Chapter 5): we present for the first time a method for unsupervised learning for graph matching algorithms, which, in our experiments, has been successful for several state-of-the-art graph matching methods. As an extension, we also propose a similar learning method for hyper-graph matching. In Chapter 6 we present our approach to learning with Spectral MAP Inference, inspired from our learning method for graph matching.

5. Object Category Recognition (Chapter 7): a method that combines the use of spectral matching with powerful geometric and appearance-based relationships from object class specific contours. This method outperforms, on several object categories, the official results from the difficult Pascal 2007 challenge.

6. Feature Grouping (Chapter 8): An efficient method for grouping based on color. We show how to use this method for class specific object segmentation.

7. Smoothing-based Optimization (Appendix A): an algorithm for optimizing general non-negative functions. In our experiments it outperforms popular algorithms such as Simulated Annealing and Markov Chain Monte Carlo optimization

Chapter 2

Spectral Matching

There are many tasks in computer vision which require efficient techniques for finding consistent correspondences between two sets of features, such as object recognition, shape matching, wide baseline stereo, 2D and 3D registration. Here we propose an efficient technique that is suitable for such applications. Our method finds consistent correspondences between two sets of features, by taking into consideration both how well the features' descriptors match and how well pairwise geometric constraints (or any other type of pairwise relationship) are satisfied. Our formulation can accommodate different kinds of correspondence mapping constraints, such as allowing a data feature to match at most one model feature (commonly used), or allowing a feature from one set to match several features from the other set (used in shape matching [15]). This is an instance of the graph matching problem in its most general form, for which, since our spectral algorithm was introduced [117], other algorithms have been proposed, such as spectral matching with affine constraints [43], probabilistic graph and hypergraph matching [233] and tensor spectral matching [53].

The features could consist of points, lines, shape descriptors or interest points, depending on the specific application. For problems where the features are non discriminative (*e.g.* points), it is the features pairwise geometric information that helps in finding the right correspondence. When discriminative features are extracted (*e.g.* interest points) then both the geometry and the properties of each individual feature can be used.

The main difficulty of the graph matching problem, introduced in the previous Chapter (Equation 5.1), is its combinatorial complexity. Our approach avoids the combinatorial explosion by taking advantage of the spectral properties of the weighted adjacency matrix $\mathbf{M}$ of a graph, whose nodes are the potential assignments (i, a) (Section 2) and weights on edges are the agreements between pairs of potential assignments. In this thesis we use the terms *assignment* and *correspondence* interchangebly .

Our method is based on the observation that the graph associated with $\mathbf{M}$ contains:

1. A strongly connected cluster formed mainly by the correct assignments that tend to establish agreement links among each other. These agreement links are formed when pairs of assignments agree at the level of pairwise relationships (*e.g.* geometry) between the features they are putting in correspondence.

2. A lot of incorrect assignments mostly outside of that cluster or weakly connected to it, which do not form strongly connected clusters due to their small probability of establishing agreement links and random, unstructured way in which they form these links.

These statistical properties motivate our spectral approach to the problem. We start by first finding the level of *association* of each assignment with the main cluster, by inspecting the eigenvector of $\mathbf{M}$ corresponding to its largest eigenvalue (principal eigenvector). Then, different procedures for finding a discrete solutions can be applied. In this Chapter we propose a very simple and fast binarization procedure, based on a greedy algorithm, which finds an approximate solution by repeatedly rejecting the assignments of low association, until the constraints on the correspondence mapping are met (Section 3). In Chapter 4, we present a different, more accurate but slightly more expensive discretization procedure, which can greatly improve the performance of spectral matching and other state-of-the-art algorithms. Spectral methods are commonly used for finding the main clusters of a graph, in tasks such as segmentation [189], grouping [143], [182], and change detection [177]. Shapiro and Brady [186] also proposed a spectral technique for correspondence problems, later improved by Carcassoni and Hancock [31], but their formulation is different and it applies only to matchings between point sets.

In previous spectral methods [218], [181], [182], [186], [190] the rows and columns of $\mathbf{M}$ correspond to single features, and the value at M_{ij} is the affinity of feature i with feature j or a function of the pairwise distance between feature i and feature j. Different from that work, here the rows and columns of $\mathbf{M}$ correspond to candidate correspondences, which are pairs of features. Here $M_{ia;jb}$ contains the affinity of feature pair (i, a) with feature pair (j, b). Also, the diagonal terms in the typical formulation are not meaningful (they measure the affinity of a single feature i with itself). However, here the diagonal terms $M_{ia;ia}$ could be meaningful, when they represent the individual score between the two features i and a. In our experiments though, we found that integrating the unary scores $M_{ia;ia}$ and $M_{jb;jb}$ into the pairwise score $M_{ia;jb}$ and leaving zeros on the diagonal leads to better performance.

Our problem formulation, as defined in the previous Chapter and also re-iterated in the next Section, also relates to the ones given in [15] and [142]. Maciel and Costeira [142] use the fact that the integral quadratic formulation of the correspondence problem can be reduced to an equivalent concave minimization problem, without changing the optima. However, the complexity of concave minimization is still non-polynomial. Berg and Malik

[15] obtain an efficient implementation by specifically designing it to allow several features from one image to match the same feature from the second image, while approximating the quadratic problem with $n+1$ linear programming problems (where n is the number of rows of $\mathbf{M}$).

Those papers formulate the problem as an integer quadratic programming problem by embedding the mapping constraints in the general form $\mathbf{Ax} = \mathbf{b}$. Instead, we relax both the integral and the mapping constraints on $\mathbf{x}$, and use them only after the optimization step. We show that this relaxation makes sense due to the spectral properties of $\mathbf{M}$. In this way we achieve a robust performance that is several orders of magnitude faster than those methods even on small size data sets.

When imposing the constraint that a feature from one set has to match at most one feature from the other set, the linear optimization approximation [15] is several orders of magnitude slower than our spectral method, even for medium size problems (when $\mathbf{M}$ has a few hundred rows). Instead, the computational complexity of our method does not depend on whether one uses one-to-one or one-to-many constraints.

2.1 Problem formulation

Given two sets of features: P, containing n_P data features, and Q, with n_Q model features, a *correspondence mapping* is a set C of pairs (or *assignments*) (i, a), where $i \in P$ and $a \in Q$. The features in P and Q that belong to some pair from C are the *inliers*. The features for which there is no such pair in C are the *outliers*. Different problems impose different *mapping constraints* on C, such as: allowing one feature from P to match at most one feature from Q, or allowing one feature from one set to match more features from the other. Our formulation of the correspondence problem can accommodate different kinds of constraints.

For each candidate assignment (i, a) there is an associated score or *affinity* that measures how well feature $i \in P$ matches $a \in Q$. Also, for each pair of assignments $(i, a), (j, b)$, where there is an *affinity* that measures how compatible the data features (i, j) are with the model features (a, b). Given a list L of n candidate assignments, we store the affinities on every assignment $(i, a) \in L$ and every pair of assignments $(i, a), (j, b) \in L$ in the $n \times n$ matrix $\mathbf{M}$ as follows:

1. $M_{ia,ia}$ is the affinity at the level of individual assignments (i, a) from L. It measures how well the data feature i matches the model feature a. It is important to note that, in practice, assignments that are unlikely to be correct (due to a large distance between the descriptors of i and a) will be filtered out and not be included in L. Thus, each such rejection will reduce the number of rows and columns in $\mathbf{M}$ by one.

2. $M_{ia,jb}$ describes how well the relative pairwise geometry (or any other type of pairwise relationship) of two model features (i, j) is preserved after putting them in correspondence with the data features (a, b). If the two assignments do not agree (*e.g.*, the deformation between (i, j) and (a, b) is too large) or if they are incompatible based on the mapping constraints (*e.g.*, $i = j$) we set $M_{ia,jb} = 0$. We assume $M_{ia,jb} = M_{jb,ia}$ without any loss of generality.

We require these affinities to be non-negative, symmetric ($M_{ia,jb} = M_{jb,ia}$), and increasing with the quality of the match, without any loss of generality. The candidate assignments (i, a) from L can be seen as nodes forming an undirected graph, with the pairwise scores $M_{ia,jb}$ as weights on the edges and the individual scores $M_{ia;jb}$ as weights at the nodes. Then, $\mathbf{M}$ represents the adjacency matrix of this undirected weighted graph. The number of nodes in this graph (= number of elements in L), adapts based on the actual data and it depends mainly on how discriminative the features's descriptors are. If the features are highly discriminative, such as SIFT descriptors, then only a small fraction of all possible pairs (i, a) are kept as candidate matches. In this case the size of $\mathbf{M}$ and the dimension of the problem search space are considerably reduced. When the features are non-discriminative (such as 2D or 3D points) and there is no a priori information about candidate matches (*e.g.*, constraints on translation), all possible pairs (i, a) can be considered as candidate assignments. In general, $\mathbf{M}$ is an $n \times n$, sparse symmetric matrix with non-negative elements, where $n = kn_P$, and k is the average number of candidate matches for each data feature $i \in P$, representing the outliers to inliers ratio. Each feature $i \in P$ will usually have a different number of candidate correspondences (i, a), $a \in Q$. In the worst case, which happens in practice only when the unary features are not discriminative at all, the size of $\mathbf{M}$ will be $n_P n_Q \times n_P n_Q$.

The correspondence problem reduces now to finding the cluster C of assignments (i, a) that maximizes the inter-cluster score $S = \sum_{ia,jb \in C} M_{ia,jb}$ such that the mapping constraints are met. We can represent any cluster C by an indicator vector $\mathbf{x}$, such that $x_{ia} = 1$ if $(i, a) \in C$ and zero otherwise. We can rewrite the total inter-cluster score as:

$$S = \sum_{ia,jb \in C} M_{ia,jb} = \mathbf{x^T M x}. \tag{2.1}$$

The optimal solution x^* is the binary vector that maximizes the score, given the mapping constraints:

$$\mathbf{x}^* = \text{argmax } \mathbf{x^T M x}. \tag{2.2}$$

The inter-cluster score $\mathbf{x^T M x}$ depends mainly on three things: the number of assignments in the cluster, how interconnected the assignments are (number of links adjacent to

each assignment) and how well they agree (weights on the links). Previous approaches [15], [142], gave a quadratic programming formulation to this problem by embedding the mapping constraints on $\mathbf{x}$ in the general form of $\mathbf{Ax} = \mathbf{b}$. Instead, we relax both the mapping constraints and the integral constraints on $\mathbf{x}$, such that its elements can take real values in $[0, 1]$. We interpret x^*_{ia} as the *association* of (i, a) with the best cluster C^*. Since only the relative values between the elements of $\mathbf{x}$ matter, we can fix the norm of $\mathbf{x}$ to 1. Then, by the Raleigh's ratio theorem, $\mathbf{x}^*$ that maximizes the inter-cluster score $\mathbf{x^T Mx}$ is the principal eigenvector of $\mathbf{M}$. Since $\mathbf{M}$ has non-negative elements, by Perron-Frobenius theorem, the elements of $\mathbf{x}^*$ will be in the interval $[0, 1]$. In Section 3 we describe how we use the mapping constraints to binarize the eigenvector and obtain a robust approximation to the optimum solution.

One novel aspect of our approach is that we drop the mapping constraints during the optimization step, and use them only afterwards to binarize the eigenvector. The problem becomes one of finding the main cluster of the assignments graph and can be solved easily using the well known eigenvector technique. We show that this method is very robust, because the main cluster in the assignments graph is statistically formed by the correct assignments. This relaxation procedure for finding the main strongly connected cluster of a graph is a well known method, which proved to be successful in other computer vision problems such as grouping [143], segmentation [189] and detecting structural changes [177].

A key insight in the understanding of the statistics of $\mathbf{M}$ is that a pair of model features is very likely to agree (in terms of the pairwise relationship between the two features) with the correct corresponding pair of data features. The same pair is very unlikely to agree with an incorrect pair of data features. Thus, correct assignments are expected to establish links between them, while incorrect assignments are not expected to form such links, and when they do, it happens in a random, unstructured way. This suggests that the correct assignments will form a highly connected cluster with a high association score, while the wrong assignments will be weakly connected to other assignments and not form strong clusters. The larger the value in the eigenvector x^*_{ia}, the stronger the association of (i, a) with the main cluster. Since this cluster is statistically formed by correct assignments, it is natural to interpret x^*_{ia} as the confidence that (i, a) is a correct assignment.

2.2 Algorithm

We propose different ways of obtaining a discrete solution that obeys the mapping constraints from the continuous eigenvector. The best one deserves a chapter of its own, due to its excellent performance and theoretical properties (Chapter 4). In this section we restrict ourselves to the ones most commonly used in the literature. The results we present in this chapter are obtained using the simple greedy procedure that we originally proposed in [117],

which has the advantage that is very easy to implement and can accommodate all types of mapping constraints: one-to-one, many-to-one or many-to-many.

Before discretization we can use the column/row rectangular bistochastic normalization, which maps the eigenvector on the L1 constraints of the matching problem $\sum_i x_{ia} = 1$ and $\sum_a x_{ia} = 1$, also used by [233], [76]. This normalization step significantly improves the performance as shown in Chapter 5. For the actual discretization we present here two different methods. For one-to-one constraints one choice is to use the Hungarian method for finding the discrete solution $\mathbf{x}$, satisfying the mapping constraints, that maximizes the dot product with the eigenvector $\mathbf{v}$:

$$\mathbf{x}^* = \text{argmax}(\mathbf{x}^\mathbf{T}\mathbf{v}). \tag{2.3}$$

Another one is the greedy procedure that we originally proposed in [117], also described next. This procedure is much more efficient than the Hungarian method at finding the discrete solution obeying the mapping constraints. Unlike the Hungarian method, it could find discrete solutions that respect different mapping constraints such as many-to-one, one-to-many or many-to-many. In our experiments we found that it gives solutions of similar quality with the Hungarian method, while having a much lower complexity.

We next present in more detail the greedy algorithm for finding an approximate solution to the correspondence problem. As discussed earlier, we interpret the eigenvector value corresponding to a particular assignment (i, a) as the *confidence* that (i, a) is a correct assignment. We start by first accepting as correct the assignment (i, a) for which the eigenvector value x^*_{ia} is maximum, because it is the one we are most confident of being correct. Next we have to reject all other assignments that are in conflict with (i, a) , as dictated by the constraints on the correspondence mapping. In our experiments these are assignments of the form $(i, *)$ or $(*, a)$ (one feature $i \in P$ can match at most one feature $a \in Q$ and vice-versa). Note that here one could use different constraints to find the assignments that are in conflict with (i, a). We accept the next correct assignment as the one with the second highest chance of being correct that has not been rejected and thus it is not in conflict with (i, a). We repeat this procedure of accepting new assignments of next highest confidence that are not in conflict with the ones accepted already, until all assignments are either rejected or accepted. This algorithm will split the set of candidate assignments in two: the set of correct assignments C^* and rejected assignments R, having the following property: every assignment from R will be in conflict with some assignments from C^* of higher confidence. Thus, no element from R can be included in C^* without having to remove from C^* an element of higher confidence.

The overall algorithm can be summarized as follows:

1. Build the symmetric positive $n \times n$ matrix $\mathbf{M}$ as described in section 2.

2. Let $\mathbf{x}^*$ be the principal eigenvector of $\mathbf{M}$. Initialize the solution vector $\mathbf{x}$ with the $n \times 1$ zero vector. Initialize L with the set of all candidate assignments (i, a)

3. Find $(i, a)^* = \text{argmax}_{(\text{i,a}) \in \text{L}}\, x^*_{ia}$. If $x^*_{ia} = 0$ stop and return the solution $\mathbf{x}$. Otherwise set $x_{ia} = 1$ and remove $(i, a)^*$ from L.

4. Remove from L all potential assignments in conflict with $(i, a)^*$. These are assignments of the form (i, k) and (q, a) for one-to-one correspondence constraints (they will be of the form (i, k) for many-to-one constraints).

5. If L is empty return the solution $\mathbf{x}$. Otherwise go back to step 3.

We note that the outliers are found at steps 3 and 4. They belong to weak assignments incompatible with assignments of higher confidence, or to those that have a zero corresponding eigenvector value (step 3). Different kinds of constraints on the correspondence mapping can be used to remove the assignments conflicting with higher confidence assignments (step 4). Our approach takes advantage of the fact that these constraints are usually easy to check. The algorithm provides a simple way to enforce the constraints as a post-optimization step, without the need of embedding them in the general form of $\mathbf{Ax} = \mathbf{b}$, required for the more expensive quadratic optimization approach. In practice our algorithm was several orders of magnitude faster than the linear programming approximation [15] to the quadratic problem, even for medium size data-sets (matching 15-20 points). In turn, the linear optimization approximation is less computationally expensive than the optimal quadratic programming approach [142], especially as the size of $\mathbf{M}$ increases.

As mentioned earlier, the Hungarian method can also be used for finding the discrete solution that maximizes the dot-product with the eigenvector. The main disadvantage of the Hungarian method is the fact that it is applicable only in the case of one-to-one matching constraints. It is slightly more computationally expensive and harder to implement than the greedy method presented here. On the positive side, the Hungarian method performs slightly better in practice. Also, it has some interesting theoretical properties: if $\mathbf{M}$ is rank 1 then the Hungarian method gives the optimal solution to the graph matching problem. Since $\mathbf{M}$ is often expected to have a very large eigengap the rank 1 approximation is valid (see the next Section).

In Chapter 4 we present a novel algorithm for significantly improving the solution, Integer Projected Fixed Point for graph matching, which can be applied as a post-processing discretization step. We refer the reader to Chapter 4 for the theoretical details and the experimental results.

2.3 Numerical considerations

2.3.1 Stability issues

It is important to verify formally that the principal eigenvector of the ideal matrix $\mathbf{M}^*$ (in which only correct assignments form pairwise links), is stable in the presence of outliers or deformations, which accidentally cause the deletion or formation of edges in the assignments graph. Here we present an argument that shows that the statistics of $\mathbf{M}$ ensure the stability of its principal eigenvector. In general, the principal eigenvector of a symmetric matrix $\mathbf{M}$ is robust to small perturbations $\mathbf{E}$ if the difference between the first two largest eigenvalues of $\mathbf{M}$ is large [200], [153]. This difference is also known as the *eigengap* of $\mathbf{M}$. In general, our matrix $\mathbf{M}$ will be a slightly perturbed version of the ideal $\mathbf{M}^*$, for which only the correct assignments establish links among each other. In the ideal case there will be no accidental links between wrong assignments, while the correct assignments will form a clique of pairwise agreements. Thus $M^*_{ia;jb}$ will have a strong positive affinity value if (i, a) and (j, b) are correct assignments and $M^*_{ia;jb} = 0$ otherwise. Under these assumptions for $\mathbf{M}^*$, the largest eigenvalue λ^*_1 of $\mathbf{M}^*$ will be equal to the total association score of the cluster formed by correct assignments, while its second largest eigenvalue λ^*_2 will equal the largest affinity score $M_{ib;ib}$ of some wrong assignment (i, b) (since all the off-diagonal elements that include a wrong assignment are zero in the ideal case).

For a large enough number of correct assignments the eigengap ρ^* of $\mathbf{M}^*$ will be slightly smaller than λ^*_1, so we can approximate $\rho^* \approx \lambda^*_1$. Moreover, the principal eigenvalue will be much larger in absolute value than all the other eigenvalues. Therefore the Frobenius norm of the ideal $\mathbf{M}^*$, $\|\mathbf{M}^*\|_F = \sqrt{\sum_i \lambda^*_i}$ is slightly larger than λ^*_1, so we can approximate $\rho \approx \|M^*\|_F$. This approximation conforms to empirical observations. For example, we obtained the ideal matrix $\mathbf{M}^*$ experimentally, in matching sets of points (Section 5). We set a tight sensitivity on deformations, then rotate and translate arbitrarily one set of points, without deforming it, to obtain the other set. The matrices obtained are very close to the ideal $\mathbf{M}^*$ and their ratio $\frac{\rho^*}{\|\mathbf{M}^*\|_F}$ is approximately 1 even for small data sets (Figure 5.4).

Let $\mathbf{v}^*$ be the principal eigenvector of the ideal matrix $\mathbf{M}^*$ and $\mathbf{v}$ be the principal eigenvector of the actual matrix $\mathbf{M}$, which is a slightly perturbed version of $\mathbf{M}^*$: $\mathbf{M} = \mathbf{M}^* + \mathbf{E}$. By Theorems *V*.2.8 from [200] and 1 from [153], if $\sqrt{2}\|\mathbf{E}\|_F \leq \frac{\rho^*}{2}$, the perturbation of the eigenvector $\mathbf{v}^*$ satisfies the inequality:

$$\|\mathbf{v}^* - \mathbf{v}\|_2 \leq \frac{4\|\mathbf{E}\|_F}{\rho^* - \sqrt{2}\|\mathbf{E}\|_F}. \tag{2.4}$$

In our case we approximate $\rho^* \approx \|\mathbf{M}^*\|_F$. If we also use the inequality $\sqrt{2}\|\mathbf{E}\|_F \leq \frac{\rho^*}{2}$, we get a rough upper bound of the change $\|\mathbf{v}^* - \mathbf{v}\|_2$ of the principal eigenvector:

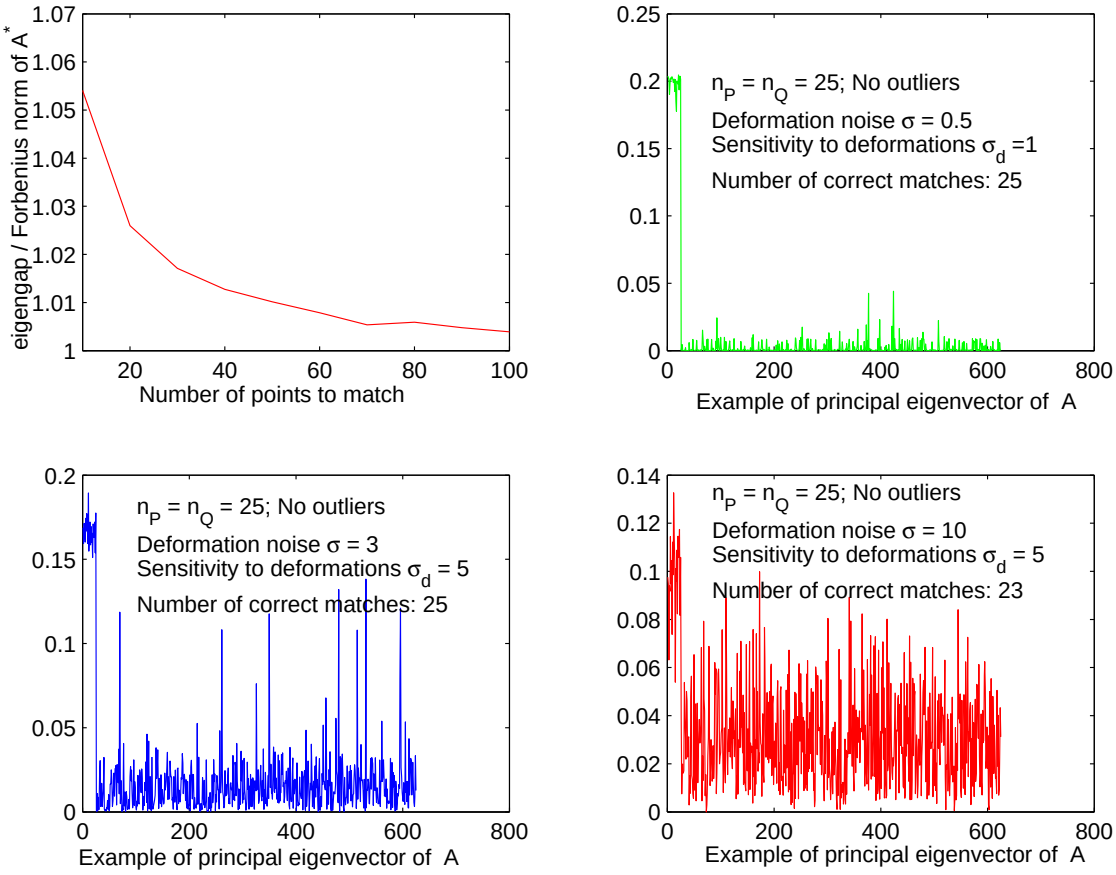

Figure 2.1: Top-right: $\rho^*/\|M^*\|_F$ vs.data set size. The other plots show principal eigenvectors (permuted such that the values of correct assignments show up on the first 25 entries).

$$\|\mathbf{v}^* - \mathbf{v}\|_2 \leq 8\frac{\|E\|_F}{\|M^*\|_F}. \tag{2.5}$$

This analysis agrees with the intuition that small perturbations $\|E\|_F$ relative to $\|M^*\|_F$ will not change significantly the direction of the principal eigenvector. In practice, even large perturbations $\|E\|_F$ that cause the formation or deletion of links in an unstructured way, will not produce higher eigenvector values for wrong assignments than for correct assignments. Only a structured perturbation, which causes wrong assignments to belong to strong clusters, can imbalance the relative difference between the eigenvector values of correct assignments vs. wrong ones. These structured accidents happen when there is a lot of symmetry in the data, the deformation noise is high, or there are a lot of outliers. Figure 5.4 shows how the principal eigenvector (obtained in experiments from Section 5) changes

smoothly as we increase the deformation noise in the problem of matching sets of points.

In a different application, on detecting structural changes, Sarkar and Boyer [177] also discuss the robustness of the principal eigenvector and eigenvalue to small unstructured perturbations of matrices with similar statistics.

2.3.2 Complexity considerations

When building the matrix **M**, one should use reasonable thresholds on the pairwise geometric deformations, tuned to the specific matching problem. For example, often in practice, we know a priori that the angular deformation or the deformation on the pairwise distances cannot be larger than some specific thresholds. When such thresholds are known (by performing the appropriate training in advance), they can be used to set the corresponding pairwise scores to zero. This gives an enormous advantage in both building and storing the matrix **M**, as well as in the complexity of computing its eigenvector, since **M** usually turns out to be very sparse.

In most cases (if not all) **M** is an $n \times n$ sparse matrix for which the efficient computation of its first eigenvector in step 2 is typically less than $O(n^{3/2})$. Variants of the Lanczos method, such as the one implemented by MATLAB function *eigs* (which we used in some of our experiments) are very efficient for finding the principal eigenvector of large symmetric sparse matrices. However, in most of our experiments we used the power iteration method for finding the principal eigenvector, which is very efficient in this case due to the large eigengap of **M**, usually converging in fewer than 20 iterations.

Since **M** is very sparse both its storage and computation can be made very efficient. Its space requirements and computation time will never reach the $O(n^2)$ complexity in practice. In our experiments (Section 5) **M** was on average about 0.1% full. After finding the principal eigenvector, one can show that in the worst case, the number of the remaining steps for the greedy binarization procedure is: $n+(n-1)...(n-m) = (n-m/2)m = O((k-1/2)m^2)$, where $m = min(n_P, n_Q)$ and $k = n/m$. For problems where the features are very discriminative (*e.g.*, SIFT descriptors, shape context), k is expected to be very small as compared to m, since a feature from one set will have only a few possible potential matches in the other set. In the worst case, every data feature could potentially match any model feature which leads to $n = n_P n_Q$.

2.3.3 Complexity Comparison with RANSAC

When the two point sets can be correctly matched by transforming one set into the other by a parametric transformation (e.g. rotation, scaling, similarity, affine), one popular algorithm to use for matching is RANSAC [67]. This is a very popular algorithm in computer vision [70] for finding correspondences, often for estimating the fundamental matrix that

relates a pair of stereo cameras. As seen in our experiments presented in this chapters as well as in Chapter 5, spectral matching is robust to matching features that undergo parameterized transformations that can be computed from two matches, such as rotation, scaling or similarity transformations. It is relatively easy to design second order scores that are not sensitive to rotation (by taking in consideration only pairwise distances), scaling (by considering only angles) or similarity (when the scale is given by the local features, such as SIFT). When the transformation can be computed from a pair of matches, RANSAC is an appealing alternative both for its simplicity and robustness. It is therefore interesting to compare the complexity of spectral matching to that of RANSAC for transformations that can be computed from two correspondences.

In the worst case (when the graphs of points/features are fully connected), the complexity of spectral matching is $O(n^2k^2)$ where n is the number of points/features and k is the number of candidate correspondences for each point. The complexity is dominated mainly by the time required to build the matrix $\mathbf{M}$, since the eigenvector computation and the binarization step have a lower complexity. This does not take into account the sparsity of the matrix: if we consider matrix fullness as a small constant fraction f of the whole matrix, then the final complexity is multiplied by the constant $f << 1$. Moreover, if instead of fully connected graphs we use sparser graph structures the complexity can be reduced anywhere from $O(fn^2k^2)$ to $O(fnk^2)$.

The expected complexity of RANSAC for transformations that require two correct correspondences and an error probability of p, is $O((\log p/\log(1-1/k^2))n^2)$ where n^2 comes from the fact that all n points have to be transformed for each candidate transformation and for each of the n points we need to find their closest neighbor.

For spectral matching, we ignore the sparsity of the matrix and consider that we have $m(n)$ number of edges for each node in the graph, where $m(n)$ could be a constant greater than 1 or a function of n, which in the worst case is $m(n) = n$ when the graph is fully connected. The complexity of spectral matching is then $O(m(n)nk^2)$. We also ignore the constant costs per iteration of the two algorithms. We want to look at the ratio of the two complexities r that is $r(k,n,p) = \frac{m(n)k^2\log(1-1/k^2)}{n\log p} = \frac{m(n)}{n\log p}\log((1-1/k^2)^{(k^2)})$. Since $p < 0.3$ (a reasonable probability of error p should in fact be much smaller than 0.3) and $m(n) \leq n$ it can be easily shown that this ratio r is a decreasing function of k, always positive and smaller than 1, which quickly converges to a constant always larger than zero. Using the well known limit $\lim_{x\to\infty}(1-1/x)^x = 1/e$ we can immediately find this constant:

$$\lim_{k\to\infty} r(k,n,p) = \frac{m(n)}{n\log p}\log((1-1/k^2)^{(k^2)}) = \frac{-m(n)}{n\log p} \tag{2.6}$$

This result basically says that if $m(n)/n$ is a constant, such as it is the case with fully connected graphs, the two algorithms have roughly the same complexity (since they are

related by a multiplicative constant factor). If the number of edges is linear in the number of nodes $m(n) = c$ (where c is a positive constant), the complexity of spectral matching is lower than that of RANSAC, because the ratio goes to zero as the number of points n increases $\lim_{n\to\infty} \frac{c}{n \log p} \log((1 - 1/k^2)^{(k^2)}) = 0$. In such cases spectral matching would be the preferred choice over RANSAC. In fact, from our experience with spectral graph matching, a constant $c \geq 10$ (that is, connecting each feature to its 10 nearest neighbors) would always give good results. Moreover, when the transformation is parametric spectral matching could be easily combined with RANSAC as follows:

1. Use a relatively small c to build the two graphs of features (from the two images to be matched), by connecting each node/feature to a few c neighbors.
2. Obtain the eigenvector $\mathbf{v}$ of $\mathbf{M}$
3. Remove the potential correspondences with eigenvector value below a certain threshold.
4. Apply RANSAC to the remaining candidate correspondences

This combination uses the strengths of each algorithm, such that each method complements the other. For a small c and a large number of features, spectral matching is significantly less expensive than RANSAC as shown earlier. Thus spectral matching would significantly speed up RANSAC by reducing the number of candidate correspondences. By finding the correct transformation, RANSAC, in turn, would output only correct correspondences, which spectral matching is not guaranteed to do by itself. Therefore, the overall complexity would be significantly lower than the original complexity of RANSAC, while the result would keep the accuracy of RANSAC as applied to the original problem by itself.

There is interesting recent work [80] that combines RANSAC with EM for simultaneous matching and finding the parameters that best fit the shape model (using Principal Component Analysis) for a specific test object. Our proposed approach is different, as it introduces a first step of using spectral matching for significantly reducing the space for RANSAC.

2.4 Experimental Analysis

We evaluate the robustness of our method first on the task of finding correspondences between $2D$ sets of points. This problem lets us evaluate the average performance of the algorithm for different levels of deformation and ratio of outliers to inliers. We study two main cases: when the deformation noise is added from a Gaussian distribution with zero mean and equal variances on the points x-y coordinates, and when sets of points are

deformed using the Thin Plate Spline model [220](TPS). The noise level is controlled by varying the variance of the gaussian distribution or the bending energy of the TPS model, and the number of outliers. We use the mapping constraint that one model feature can match at most one data feature and vice-versa. For all experiments in this section we did not use IPFP, nor the row/column normalization step proposed in Section 2.2, both of which would further improve the performance of spectral matching. For experiments with the normalization procedure and IPFP, we refer the reader to the experiments from Chapter 5. Here we limit ourselves to the basic algorithm, finding the leading eigenvector and binarizing it with the greedy algorithm proposed in Section 2.2.

2.4.1 Deformations using white noise

In the first set of experiments we generate data sets of 2D model points Q by randomly selecting n_Q^i inliers in a given region of the plane. We obtain the corresponding inliers in P by disturbing independently the n_Q^i points from Q with white Gaussian noise $N(0, \sigma)$ and then rotating and translating the whole data set Q with a random rotation and translation. Next we add n_Q^o and n_P^o outliers in Q and P, respectively, by randomly selecting points in the same region as the inliers from Q and P, respectively, from the same random uniform distribution over the x-y coordinates. The range of the $x-y$ point coordinates in Q is $256\sqrt{n_Q/10}$ to enforce an approximately constant density of 10 points over a 256 x 256 region, as the number of points varies. The total number of points in Q and P are $n_Q = n_Q^i + n_Q^o$ and $n_P = n_P^i + n_P^o$. The parameter σ controls the level of deformations between the two sets, while n_P^o and n_Q^o control the number of outliers in P and Q, respectively. This is a difficult type of problem for two reasons. First, the points are non-discriminative and they can be translated and rotated arbitrarily, so any of the n_P points from P can potentially match any of the n_Q model points from Q. This maximizes the search space (the solution vector will have $n_Q n_P$ elements) and leaves the task of finding a solution entirely to the relative geometric information between pairs of points. Secondly, picking points randomly in the plane creates homogeneous data sets with an increased level of symmetry, which increases the chance of accidental agreements between wrong correspondences or between correct correspondences and wrong ones. Also, choosing outliers from the same distribution as the inliers, within the same region, increases the chance that similar geometrical relationships will be formed among outliers or between the outliers and the clean points as among the clean points only.

Since points are non-discriminative we set the score on individual assignments $\mathbf{M_{ia;ia}}$ to zero (we left the matching score entirely to the pairwise geometric information, since there is no information on the individual assignments). For the pairwise score $M_{ia,jb}$ we use the pairwise distances between points. As mentioned, the actual positions of points are

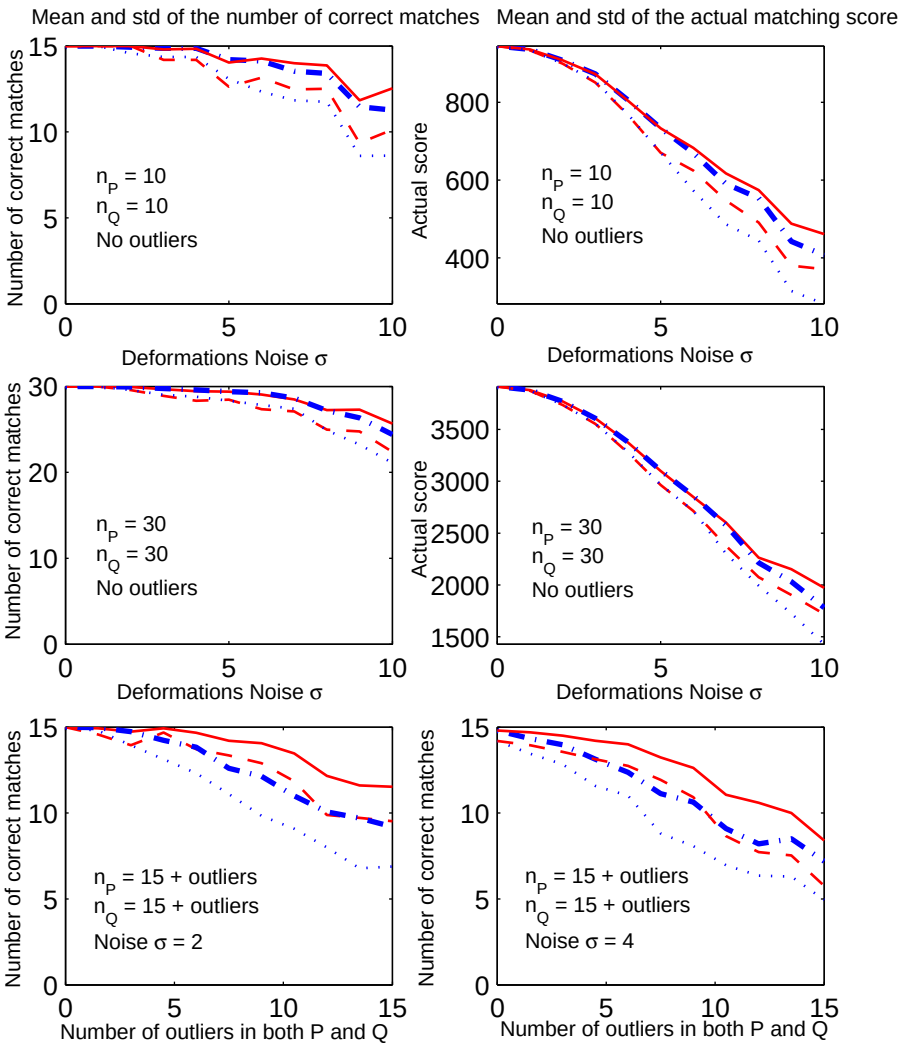

Figure 2.2: Performance curves for our method vs. *linprog* method. The mean performance is shown as a solid red line (our method) and a blue dash-dotted line (*linprog* method). One *std* below the mean: red dashed lines (our method), blue dotted lines (*linprog* method). First two rows: no outliers, varying deformation noise. The number of correct matches (left) and the actual scores (right) are plotted. Third row: the number of outliers in each P and Q is varied for two values of the deformation noise.

irrelevant because they are rotated and translated arbitrarily:

$$M_{ia;jb} = \begin{cases} 4.5 - \frac{(d_{ij}-d_{ab})^2}{2\sigma_d^2} & \text{if } |d_{ij} - d_{ab}| < 3\sigma_d \\ 0 & \text{otherwise,} \end{cases}$$

where d_{ij} and d_{ab} are the Euclidean distances between the points i and j, and between

their candidate matches a and b, respectively. The parameter σ_d controls the sensitivity of the score on deformations. For larger σ_d, larger deformations in the data are allowed, which results in more positive pairwise scores between wrong assignments. $M_{ia,jb}$ defined above is always non-negative and increases as the deformation between candidate pairs of assignments decreases. The total score $S = \mathbf{x^T M x}$ increases as the number of assignment links that are below the deformation threshold of $3\sigma_d$ increases and as the sum of squared deformations on those links decreases.

Figure 2.2 shows the performance curves of our method vs. the linear programming approximation method [15] as we vary the noise σ from 0.5 to 10 (in steps of 0.5), the number of points to be matched: from 15 up to 30, and the number of outliers in both P and Q. We score the performances of the two methods by counting how many matches agree with the ground truth. We kept the sensitivity parameter fixed $\sigma_d = 5$. For our algorithm we use MATLAB function *eigs*, which implements the Implicitly Restarted Arnoldi method [115], a variant of Lanczos method. For the linear programming method we used MATLAB function *linprog* with the LargeScale option that is based on LIPSOL (Linear Interior Point Solver, [234]). Both algorithms ran on the same problem sets over 30 trials for each value of the varying parameter (Figure 2.2). Both the mean performance curves as well as the curves one standard deviation below the mean are plotted. As expected, for large values of the deformation σ and large numbers of outliers, both algorithms start shifting smoothly from the correct matches, which indicates that some wrong assignments have established enough links to win over correct assignments. The performance of the two algorithms degrades in a similar manner, which suggests that the true optimum of the score function shifts from the ground truth as the amount of noise increases (as introduced by outliers and deformations). For lower values of the noise, both algorithms find the correct matches which indicates that the optimum of the score function coincides with the ground truth. Both algorithms prove to be robust to noise, but ours shows a slightly better robustness for larger values of noise. In the case of outliers our algorithm is clearly more robust than the *linprog* based method. The main point of this comparison is to show that our method gives results at least as reliable as an existing method, while being orders of magnitude faster: over 400 times faster on 20 points problem sets (average time of 0.03 sec. vs 13 sec) and over 650 faster on 30 points problem sets (0.25 sec. vs 165 sec.).

We further tested our method on the same problem, but with larger data sets (50, 100 and 130 points, Figure 2.3), for which *linprog* becomes too slow to make an extensive comparison. We notice that the performance of our algorithm gets better as the number of points increases. This happens because as the number of data points increases, each correct assignment establishes pairwise relationships with more correct assignments. Thus it becomes more robust to deformations and to outliers. Our method took on average less than 9 seconds in MATLAB for 130 points problem sets.

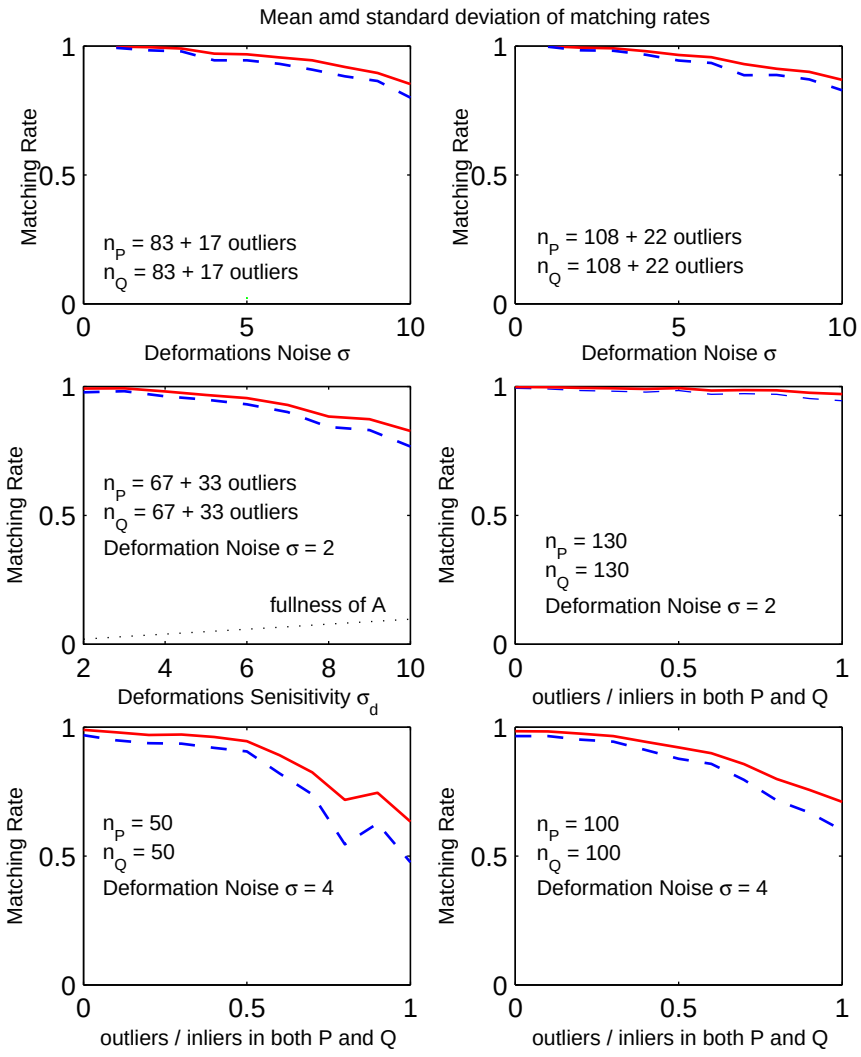

Figure 2.3: Average matching rates (=correctly matched inliers vs. total inliers) over 30 tests for each parameter value on the x axis. Middle-right: also plotted (black dash-dotted line) the ratio of non-zero values in $\mathbf{M}$ vs. total elements in $\mathbf{M}$. The more deformation we allow (σ_d) the less sparse $\mathbf{M}$ is.

We also tried to simulate real applications of matching very large number of points. Only for this case, we limited the number of candidate correspondences per point, by accepting as candidate matches (i, a) only points that were within a radius of 500 from each other. When generating the inliers in P from the inliers in Q, we limited the translation to 100 and the rotation around the center of mass of the points in Q to $[-\pi/9, \pi/9]$. This resulted in each point from P having on average around 100 candidate correspondences in Q, including

its correct correspondence. We imposed the additional constraint on the pairwise distances score that $M_{ia;jb} = 0$ if $d_{ij} > 200$ or $d_{ab} > 200$, or the angle between the directions of (i, j) and (a, b) was outside the interval $[-\pi/9, \pi/9]$. The average performances, over 30 runs, on data sets of 400, 600 and 1000 points, for a deformation noise of $\sigma = 2$ and ratio of $outliers/inliers = 0.5$ in each set P and Q, were: 97% (400 points case) and 93% (both 600 and 1000 points case). It took less than 30 sec. in MATLAB to find the solution for the 1000 points case..

2.4.2 Non rigid deformations using the Thin Plate Spline Model

In the previous experiments all the points were perturbed according to the same Gaussian distribution and then arbitrarily rotated and translated. In this Section we test our method on deformations that follow the TPS model, for which the amount of deformation is quantified by the bending energy applied to the x-y meshgrid (Figure 2.4). We generated the sets Q and P as follows: we randomly pick n_Q^i inliers in Q on a 20 by 20 x-y meshgrid. Then we randomly deform the meshgrid for a given level of the bending energy and obtain the corresponding n_P^i inliers from P. Given a ratio of $inliers/outliers$, we further add n_Q^o outliers in Q by randomly picking points in the plane in a region that contains the inliers. Similarly, we pick the n_P^o outliers in P. As in previous tests, we enforce the mapping constraint that one point from P must match at most one point from Q and vice-versa. The score function used before on pairwise deformations is not appropriate in this case, since the pairwise distance between far points is expected to be changed more than the distance between close points. Instead, we normalize the pairwise deformations by the absolute pairwise distance d_{ab}. We also penalize changes in directions. As before, we do not use any scores on individual assignments ($M_{ia;ia} = 0$). For pairwise deformations we use a score function that is similar in concept to the one from [15]: $M_{ia;jb} = (1-\gamma)c_\alpha + \gamma c_d$, if $|\alpha_{ia;jb}| < 3\sigma_\alpha$ and $|d_{ia;jb} - 1| < 3\sigma_d$, and zero otherwise. Here, $c_\alpha = 4.5 - \frac{\alpha_{ia;jb}^2}{2\sigma_\alpha^2}$, $c_d = 4.5 - \frac{(d_{ia;jb}-1)^2}{2\sigma_d^2}$, $d_{ia;jb} = \frac{d_{ij}+q}{d_{ab}+q}$, d_{ij} and d_{ab} are the distances between the model features (i, j) and between the data features (a, b), respectively, and $\alpha_{ia;jb}$ is the angle between the direction of (i, j) and that of (a, b). The term c_α penalizes changes in direction, c_d penalizes changes in the relative length, while γ weighs one term against the other. TPS is often used as a deformation model for matching shapes. Figure 2.5 shows a couple of examples of matching point sets sampled from natural images, using the scores defined here and no shape descriptors.

2.4.3 Recognizing objects from low resolution images

In this Section we show an application of our method to recognition of vehicles from aerial, low resolution images, using SIFT descriptors. In this case, because of the low resolution images, the normal voting method for retaining the correct matches [134] cannot be applied

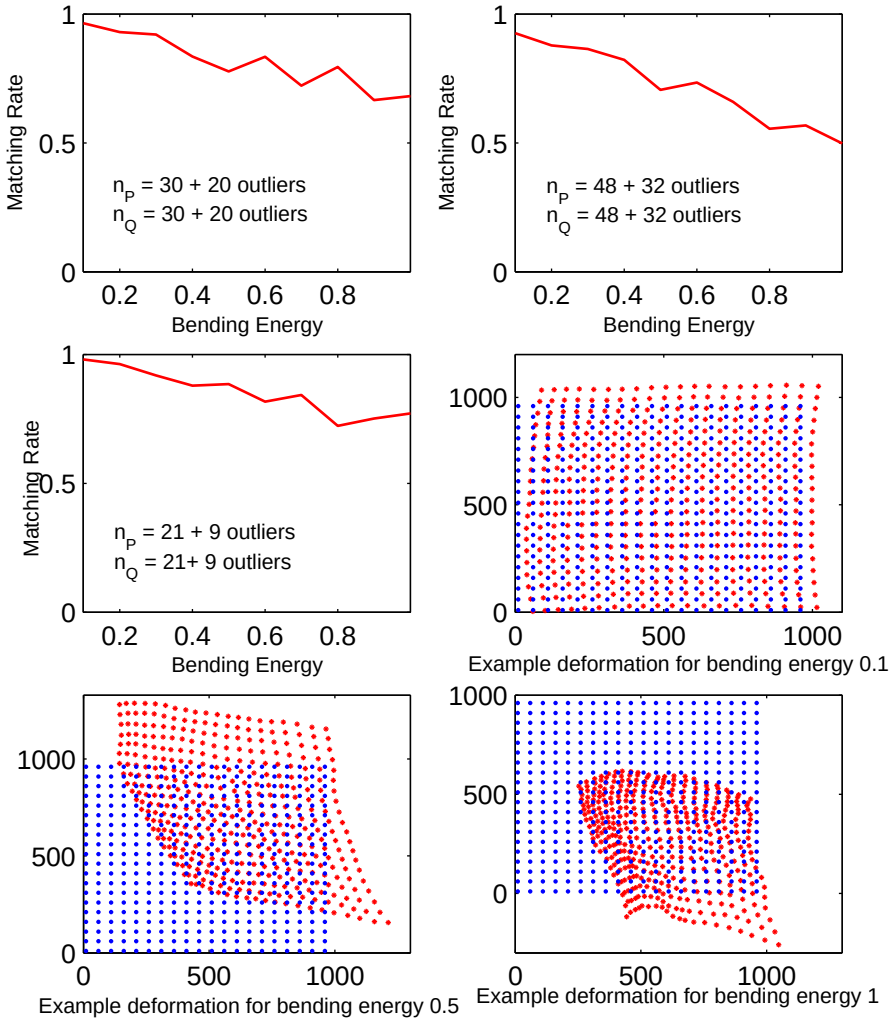

Figure 2.4: Top row: average matching rate of 50 points (top-left), 80 points (top-right) and 30 points (bottom-right) for different bending energy levels, over 30 trials for each energy level. Middle and bottom rows: examples of the x-y meshgrid deformation for bending energy levels: 0.1, 0.5 and 1.

reliably. This is due to fact that the number of features extracted for each object is very small and their location and scale is not very stable. Moreover, the SIFT descriptors are less discriminative when applied to low resolution and low texture objects. For example, a lot of these features are extracted at the cars boundaries, and it often happens that multiple features from the same or different objects are very similar to each other. Therefore, we must allow multiple candidate matches for each feature and use the pairwise relationships

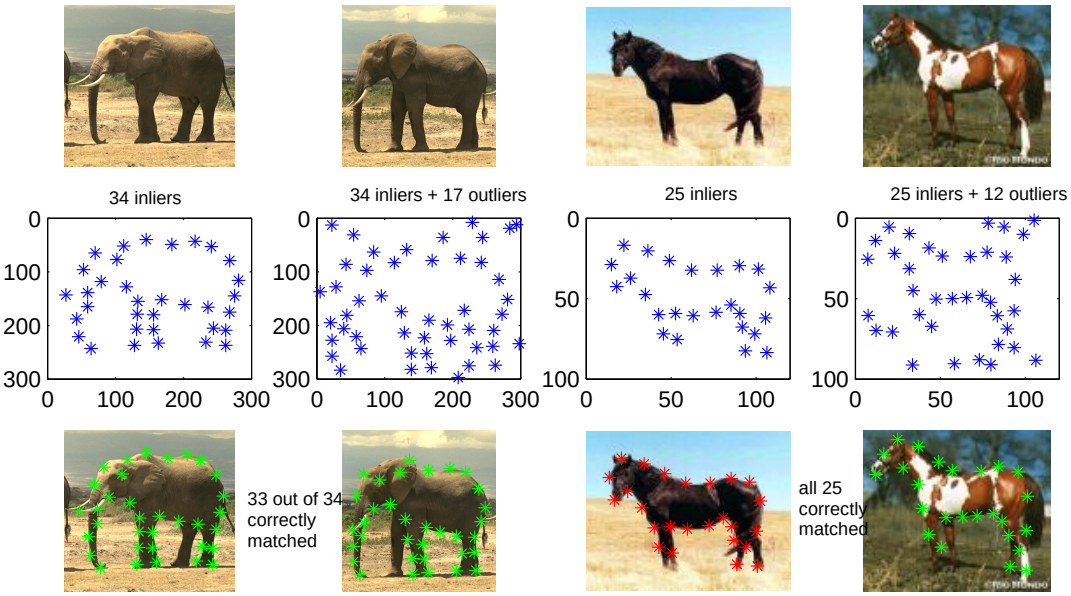

Figure 2.5: Correspondences between points from natural images.

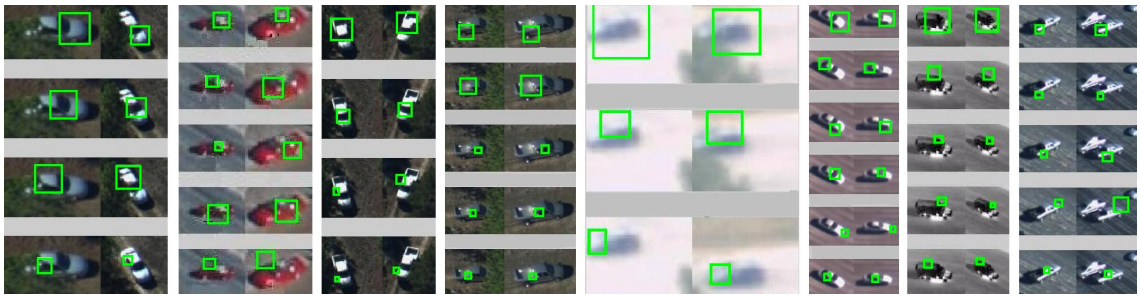

Figure 2.6: Examples of correspondences between data features and model features in recognizing cars from aerial images

between them to disambiguate the correct correspondences. We built 45 car models Q_q from 45 video sequences, following the method described in [116]. The models represent constellations of SIFT features that establish pairwise co-occurrence probabilities if they co-occur during the video sequence within a small distance from each other. Given a test image P containing an object and possible clutter, the task is to recognize it by trying to match it against a subset of the models built. The model Q_q^* that gives the highest correspondence score $\mathbf{x^T M x}$ will be retrieved as the correct match. We used a pairwise score $M_{ia,jb}$ that is high if data features i and j are within a small distance of each other

and the pairwise co-occurrence probability p_{ab} between their candidate model matches a and b is high:

$$M_{ia;jb} = \begin{cases} ln\frac{p_{ab}}{p_0} & \text{if } p_{ab} > p_0 \text{ and } d_{ij} < d \\ 0 & \text{otherwise,} \end{cases}$$

where p_{ab} is the co-occurrence probability of the model features a and b and d_{ij} is the distance between the centers of the features i and j in the image. The parameter p_{ab} is proportional to the number of co-occurrences of the model features a and b during the training sequence; p_0 is a small probability, which represents the chance that two independent features will co-occur; d defines the radius of the region in which two features must co-occur in order to consider the pair of their candidate assignments a and b. It is not important how these parameters are learned [116]. What matters is that once we have them, we can apply our correspondence method for this recognition task. For the individual assignments $M_{ia;jb}$ we used a score that is linearly decreasing with the Mahalanobis distance between the data feature i and its candidate corresponding model feature a.

Notice that the only geometrical information we used was to check whether i and j are within a distance d from each other. We did not use the actual value of d_{ij} in the pairwise score $M_{ia;jb}$ because the vehicles might undergo changes in scale and out of plane rotations. We matched 379 novel images against the correct model plus 10 other randomly selected models and we recognized correctly 371 images ($\approx 98\%$ recognition rate). Figure 2.6 shows matches between the features from the test images (left) and features from the models (right).

2.5 Extension: Matching Tuples of Graphs

So far the graph matching problem has been limited to matching pairs of graphs. There is work that uses multiple graph matching for object recognition, which consists of matching an input graph from a test image to several model graphs and retrieving the closest one in terms of some distance function [121], [15], [117]. Multiple graph matching is also used in object or category discovery [97], [96], [159], [158], [198] for which pairs of images are matched in order to track the occurrences of the same object part over a sequence of images. In all work using multiple graph matching the same basic model of matching pairs of graphs was used. To the best of our knowledge there is no previous work on matching tuples of graphs simultaneously, even though that could be very useful in applications where the same object appears in several images that need to be matched together. In Figure 2.7 we show intuitively why matching several graphs at the same time could be more powerful than breaking the problem into matching pairs of graphs. Let us assume that we have three images $\mathbf{I_1}$, $\mathbf{I_2}$ and $\mathbf{I_3}$ containing instances of the same object and we want to match the

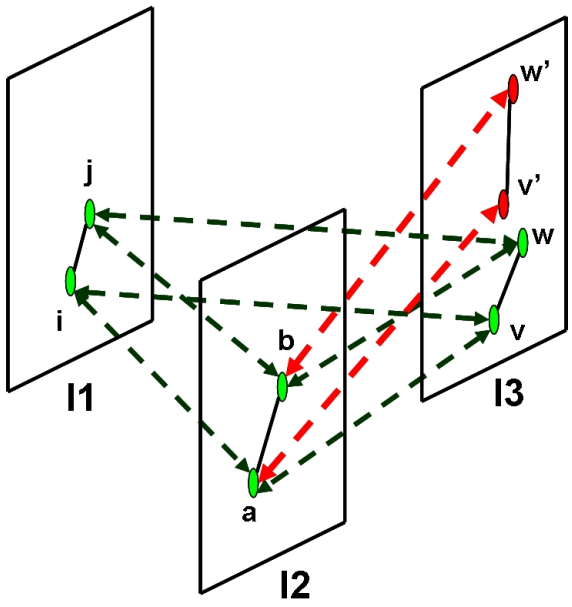

Figure 2.7: Matching three images simultaneously is more robust than matching them in sequence. Matching images in sequence has a higher chance of drifting, similar to the drifting issue in object tracking.

corresponding features across all three images. As an example, let us look at how to match a pair of features (i, j) from the first image in the other two images. If we match the images in pairs sequentially $\mathbf{I_1}$ to $\mathbf{I_2}$ and then $\mathbf{I_2}$ to $\mathbf{I_3}$ we could match (i, j) to (a, b) and (a, b) to (v', w') even though matching (a, b) to (v, w) would probably be the correct choice since the pairwise geometric relationship of (v, w) is similar to both (i, j) and (a, b) whereas the relationship of (v', w') is only similar to that of (a, b). This is a classical instance of drifting, also encountered in object tracking. Initially, one would argue that the drifting issue could be fixed in this case by also matching $\mathbf{I_1}$ to $\mathbf{I_3}$ directly. However, on closer inspection, we see that the issue is still not solved: since the pairs are matched independently, we could end up matching (i, j) to (a, b), (a, b) to (v', w') and (i, j) to (v, w), so we are left with conflicts at the mapping constraints level since i is matched to both v (directly) and v' (indirectly) and it is not clear which one to choose.

2.5.1 Matching Triplets of Graphs

Is there a way to match $\mathbf{I_1}$ to $\mathbf{I_2}$ and $\mathbf{I_2}$ to $\mathbf{I_3}$ while taking in consideration *simultaneously* the deformations among all three pairs (i, j), (a, b) and (v, w) and have no mapping conflicts? Here we propose an algorithm that addresses this question, which extends the spectral

matching ideas presented in this chapter. To the best of our knowledge, this is the first time that the problem of matching tuples of graphs (instead of pairs) is addressed in the literature. A crucial difference between the algorithm we present here and hypergraph matching [53], [233] is that, while hypergraph matching is still concerned with matching two graphs and considers higher order tuples between the nodes, here we are concerned with matching tuples of graphs, while still using pair-wise relationships (edges) between the nodes. However, the idea that we present here can be further extended, at least in theory, to hypergraph matching of tuples of graphs.

For clarity of presentation we start by discussing the case of matching triples of graphs. We first define the 4-dimensional super-symmetric tensor $\mathbf{H}$ such that only elements of the form $H_{ia;jb;av;bw}$ can be non-zero, where ia, jb, av, bw are candidate assignments and each pair of features (i,j), (a,b) and (v,w) belongs to a different image/graph. $H_{ia;jb;av;bw}$ contains the overall geometric agreement for matching the pairs of features (i,j) to (a,b), (a,b) to (v,w) and (i,j) to (v,w). Each side of the 4-dimensional cube $\mathbf{H}$ has dimension equal to the total number of assignments between **I1** and **I2** and **I2** and **I3**. For each candidate assignment (i,a) between **I1** and **I2** there is a corresponding element in the solution vector $\mathbf{x}$ such that $x_{ia} = 1$ if i is matched to a and zero otherwise and, similarly, for each candidate assignment (a,v) between **I2** and **I3** there is a corresponding element in the solution vector $\mathbf{y}$ such that $y_{av} = 1$ if a is matched to v and zero otherwise. Note that we will not consider (i,v) as a candidate assignment since the assignments from $\mathbf{I_1}$ to $\mathbf{I_3}$ are defined indirectly by the assignments through $\mathbf{I_2}$. The third order graph matching problem is then defined as follows:

$$[\mathbf{x}^*; \mathbf{y}^*] = \text{argmax} \sum_{i,j,a,b,v,w} H_{ia;jb;av;bw} x_{ia} x_{jb} y_{av} y_{bw}, \tag{2.7}$$

where $[\mathbf{x}; \mathbf{y}]$ is an indicator vector that obeys the mapping constraints. Each element in $[\mathbf{x}; \mathbf{y}]$ corresponds to a candidate assignment either from $\mathbf{I_1}$ to $\mathbf{I_2}$ or from $\mathbf{I_2}$ to $\mathbf{I_3}$. Note that the summation is not over all elements of $\mathbf{H}$ because the zero terms are skipped (since they have no influence on the total score). Only the terms $H_{ia;jb;av;bw}$ that could be non-zero are considered. Of course, in practice, most of those will also be zero as it is the case with the spectral matching sparse matrix $\mathbf{M}$.

This can be solved approximately by extending our spectral matching algorithm to tensors, also related to the work of [53], [168]. The approximate solution can be obtained by finding the leading eigenvector $\mathbf{q}$ of $\mathbf{H}$ using the extension of the power method to tensors. Note that this method will return a local optimum:

1. Repeat until convergence:

2. $q_{ia} \leftarrow \sum_{j,b,v,w} H_{ia;jb;av;bw} q_{jb} q_{av} q_{bw}$

3. $\mathbf{q} \leftarrow \mathbf{q}/\|\mathbf{q}\|$.

This formulation sounds fine in theory but it is expensive in practice and difficult to implement efficiently. Hence, next we propose a different algorithm for finding approximate solutions efficiently. First we define the tensor $\mathbf{H}$ in terms of three matrices, which reduces the storage costs and time to build the tensor and also facilitates the design of a new efficient algorithm for the optimization of Equation 2.7.

In the case of spectral matching for pairs of graphs, the most expensive step in practice is building the matching matrix $\mathbf{M}$, because, even though most of its elements are zero, each has to be visited in order to decide whether it is null or not (visiting each element is required only in the case of fully connected graphs). This would suggest that building the tensor $\mathbf{H}$ is far from being practical since its size is $(2Nk)^4$, where N is the number of nodes in each graph and k is the number of candidate assignments per node. To overcome this problem, and still capture the geometric agreements of all triplets of feature pairs $(i,j) \in \mathbf{I1}$, $(a,b) \in \mathbf{I2}$ and $(v,w) \in \mathbf{I2}$, we define each possibly non-zero element in $\mathbf{H}$ as $H_{ia;jb;av;bw} = M^{(1-2)}_{ia;jb} M^{(2-3)}_{av;bw} M^{(1-3)}_{iv;jw}$, where the matrices $\mathbf{M^{(1-2)}}$, $\mathbf{M^{(2-3)}}$, $\mathbf{M^{(1-3)}}$ are built as before for matching each pair of images $\mathbf{I_1}$ to $\mathbf{I_2}$, $\mathbf{I_2}$ to $\mathbf{I_3}$ and $\mathbf{I_1}$ to $\mathbf{I_3}$. Given the optimum $[\mathbf{x}^*; \mathbf{y}^*]$ of Equation 2.7, we can rewrite this equation in two different ways:

$$\mathbf{x}^* = \text{argmax} \sum_{ia;jb} M^{(1-2)}_{ia;jb} (\sum_{v;w} M^{(2-3)}_{av;bw} M^{(1-3)}_{iv;jw} y^*_{av} y^*_{bw}) x_{ia} x_{jb} \tag{2.8}$$

$$\mathbf{y}^* = \text{argmax} \sum_{av;bw} M^{(2-3)}_{av;bw} (\sum_{i;j} M^{(1-2)}_{ia;jb} M^{(1-3)}_{iv;jw} x^*_{ia} x^*_{jb}) y_{av} y_{bw}. \tag{2.9}$$

Here we use the fact that $H_{ia;jb;av;bw} = M^{(1-2)}_{ia;jb} M^{(2-3)}_{av;bw} M^{(1-3)}_{iv;jw}$ and enforce the one-to-one, integer constraints on $(\mathbf{x}, \mathbf{y})$. These equations are the starting point for the design of the algorithm that we propose next. The algorithm is based on an iterative procedure that, given the indicator solution $\mathbf{x^{(n)}}$ at iteration n it finds $\mathbf{y^{(n+1)}}$ as the optimum of the total matching score $S(\mathbf{x^{(n)}}, \mathbf{y})$ and vice-versa, it finds $\mathbf{x^{(n+1)}}$ as the optimum of $S(\mathbf{x}, \mathbf{y^{(n+1)}})$:

$$\mathbf{y^{(n+1)}} = \text{argmax} \sum_{av;bw} M^{(2-3)}_{av;bw} (\sum_{i;j} M^{(1-2)}_{ia;jb} M^{(1-3)}_{iv;jw} x^{(n)}_{ia} x^{(n)}_{jb}) y_{av} y_{bw} \tag{2.10}$$

$$\mathbf{x^{(n+1)}} = \text{argmax} \sum_{ia;jb} M^{(1-2)}_{ia;jb} (\sum_{v;w} M^{(2-3)}_{av;bw} M^{(1-3)}_{iv;jw} y^{(n+1)}_{av} y^{(n+1)}_{jw}) x_{ia} x_{jb}. \tag{2.11}$$

Starting from a good initialization $(\mathbf{x^1}, \mathbf{y^1})$, the algorithm consists of alternating the previous two steps until convergence. Before we present the overall algorithm in detail, we make an important observation: the sums of type $\sum_{v;w} M^{(2-3)}_{av;bw} M^{(1-3)}_{iv;jw} y^{(n+1)}_{av} y^{(n+1)}_{bw}$ contain only one element since $\mathbf{y^{(n+1)}}$ obeys the one-to-one mapping constraints. Then, for a given

n let us define $v(a)$ to be that v (there is only one) for which $\mathbf{y_{av}^{(n+1)}} = 1$. Similarly we can define $w(b)$. Also we will have $i(a)$ and $j(b)$, using $\mathbf{x^{(n)}}$ instead of $\mathbf{y^{(n+1)}}$. This notation helps next, in the actual description of the algorithm:

1. Initialize: $\mathbf{x^{(1)}} = \mathbf{argmax}(\mathbf{x^T M^{(1-2)} x})$ and $\mathbf{y^{(1)}} = \mathbf{argmax}(\mathbf{y^T M^{(2-3)} y})$ given the one-to-one mapping constraints. Set $n = 1$.

2. Build matrix $\mathbf{B}$, using $\mathbf{x^{(n)}}$, such that $B_{av;bw} = M^{(1-2)}_{i(a)a;j(b)b} M^{(2-3)}_{av;bw} M^{(1-3)}_{i(a)v;j(b)w}$

3. Solve $\mathbf{y^{(n+1)}} = \mathbf{argmax}(\mathbf{y^T B y})$, given the one-to-one mapping constraints on $\mathbf{y}$

4. Build matrix $\mathbf{A}$, using $\mathbf{y^{(n+1)}}$, such that $A_{ia;jb} = M^{(1-2)}_{ia;jb} M^{(2-3)}_{av(a);bw(b)} M^{(1-3)}_{iv(a);jw(a)}$

5. Solve $\mathbf{x^{(n+1)}} = \mathbf{argmax}(\mathbf{x^T A x})$ given the one-to-one mapping constraints on $\mathbf{x}$

6. If $(\mathbf{y^{(n+1)}})^\mathbf{T} \mathbf{B y^{(n+1)}} = (\mathbf{x^{(n+1)}})^\mathbf{T} \mathbf{A x^{(n+1)}}$ stop. Otherwise update $n \leftarrow n + 1$ and go back to step 2.

Proposition: *This algorithm increases the overall matching score at each iteration and converges in a finite number of iterations.*

Proof: These properties are easy to prove. At step n $\mathbf{y^T B y} = \mathbf{S}(\mathbf{x^{(n)}}, \mathbf{y})$ and $\mathbf{x^T B x} = \mathbf{S}(\mathbf{x}, \mathbf{y^{(n+1)}})$, where S is the total score. Since the score increases at steps 3 and 5, it follows that it increases after each iteration. For any iteration k we have:

$$\mathbf{S}(\mathbf{x^{(k)}}, \mathbf{y^{(k)}}) \leq \mathbf{S}(\mathbf{x^{(k)}}, \mathbf{y^{(k+1)}}) \leq \mathbf{S}(\mathbf{x^{(k+1)}}, \mathbf{y^{(k+1)}}). \tag{2.12}$$

The stopping condition from step 7 is met in a finite number of iterations since the score is bounded above, increasing, and the set of possible indicator (binary) solutions $\{\mathbf{x}; \mathbf{y}\}$ is finite.

We note that in each case, the argmax function corresponds to the pairwise, traditional, graph matching problem. Since graph matching is NP-hard, the argmax functions must be replaced by an approximate algorithm. We propose to replace them with the IPFP algorithm presented in Chapter 4 and reach $\mathbf{x}^{(n+1)}$ and $\mathbf{y}^{(n+1)}$ by initializing IPFP with $\mathbf{x}^{(n)}$ and $\mathbf{y}^{(n)}$. The climbing property still holds, since IPFP itself has the same climbing property. Even though there is no bound guarantee with respect to the true global optimal solution $((\mathbf{x}^*, \mathbf{y}^*))$, since step 1 offers a good initialization, we expect the solution obtained to be satisfactory in practice. Also, it is important to note that the climbing property, albeit important in practice, does not guarantee convergence to a maximum (local or global).

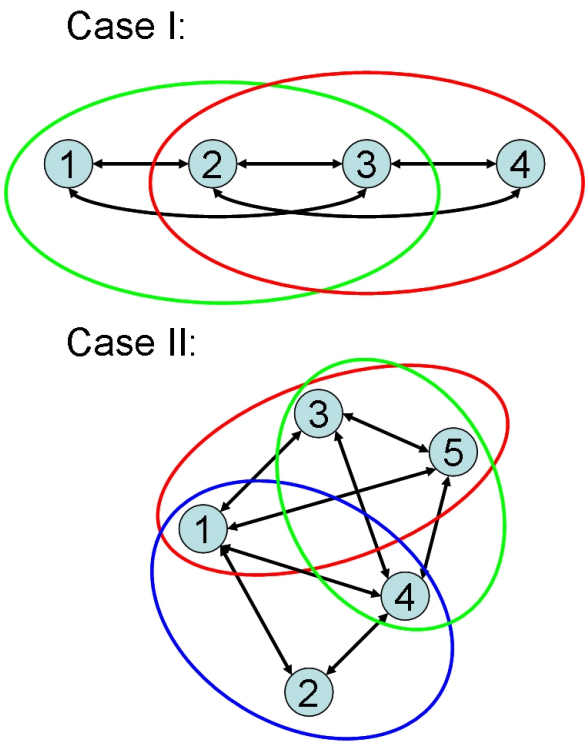

Figure 2.8: Matching tuples of images simultaneously. Case I: matching images in a sequence (e.g. object tracking). Case 2: matching images organized in an arbitrary graph structure (e.g. personal photo albums, online image databases).

2.5.2 Matching Tuples of Graphs Simultaneously

We propose a direct extension of the previous algorithm for matching tuples of graphs, by considering all possible triplets of graphs for which we would have considered pair-wise matching. We reduce the tuple graph matching problem to matching several graph triplets, in order to avoid the combinatorial explosion. We show that by repeatedly matching triplets of graphs we can propagate the assignment information to a whole set of graphs, thus achieving simultaneous multiple graph matching.

More precisely, let us consider each image as a node in a super-graph, in which each edge (i, j) corresponds to a possible pair-wise match between images $\mathbf{I}_i$ and $\mathbf{I}_j$. Then, for each ordered triangle (i, j, k) in the graph we consider one tensor $\mathbf{H}^{(i-j-k)}$. Note that for any pair of images (i, j) the pair-wise match matrix $\mathbf{M}^{(\mathbf{i-j})}$ is the same as $\mathbf{M}^{(\mathbf{j-i})}$ and the solution $\mathbf{x}_{(\mathbf{i-j})}$ is the same as $\mathbf{x}_{(\mathbf{j-i})}$. We define the total matching score for tuples of graphs as the sum of the triplet scores S_3 of over all ordered triangles:

$$S_N = \sum_{i,j,k} S_3(\mathbf{I}_1, \mathbf{I}_2, \mathbf{I}_3). \tag{2.13}$$

By applying the same ideas used in the case of 3 graphs we can extend the previous algorithm as follows. Here $\mathbf{A}_{ijk}$ are the matrices built for a triplet of graphs i, j, k in the same fashion as matrices $\mathbf{A}$ and $\mathbf{B}$ from the steps 2 and 4 of the previous algorithm for triples, where $\mathbf{A}_{123} = A$ and $\mathbf{A}_{231} = B$.

1. Initialize: for each pair of images (i, j) considered let $\mathbf{x}^{(\mathbf{1})}_{\mathbf{i-j}} = \text{argmax}\,(\mathbf{x^T M^{(i-j)} x})$ given the one-to-one mapping constraints. Set $n = 1$.

2. for each unordered pair of images (i, j) repeat until S_N stops increasing:

   a) Pick an order $i \to j$. Let $\mathbf{A}_{ij} = \sum_k \mathbf{A}_{ijk}$, for each existing triangle i, j, k using the previous $\mathbf{x}^{(\mathbf{n})}_{\mathbf{j-k}}$

   b) Let $\mathbf{x}^{(\mathbf{n+1})}_{\mathbf{i-j}} = \text{argmax}(\mathbf{x^T A_{ij} x})$.

Again, as before, the argmax function should be replaced with an efficient graph matching algorithm.

2.5.3 Complexity Considerations

We discuss the complexity of the algorithm in the case of N graphs (since it includes the case of 3). The matrices $\mathbf{A}_{ij}$ can be efficiently built in a time that is less (in complexity) than solving the graph matching problem between two graphs. Then, it immediately follows that the complexity of matching tuples is linear in the number of pairs of graphs:

$$Complexity_{tuples} = nPairs * (nIterations + 1) * Complexity_{pair}, \tag{2.14}$$

where $Complexity_{pair}$ is the complexity of IPFP or the graph matching algorithm chosen for matching two graphs, $nPairs$ is the number of edges in the graph of images and $nIterations$ is the number of iterations at step 2. Note that the complexity of matching $nPairs$ from N graphs in the pair-wise, traditional sense, is only $nIterations + 1$ times less expensive than matching the N graphs simultaneously.

Chapter 3

Spectral MAP Inference

Efficient methods for MAP inference in graphical models are of major interest in pattern recognition, computer vision and machine learning. The MAP problem often reduces to optimizing an energy function that depends on how well the labels agree with the data (first-order potentials) as well as on how well pairs of labels at connected sites agree with each other (second order potentials) given the data. From this point of view the main difference between graph matching and the MAP problem are the mapping constraints. Graph matching usually obeys one-to-one constraints while the MAP inference many-to-one. Therefore, it is expected that MAP inference and graph matching problems could be sometimes solved by similar algorithms. Our algorithm [118], which we also present in this Chapter, is among the first, together with [167], to propose a solution for the MAP inference problem based on a quadratic programming formulation, as opposed to the linear programming LP formulations [221], SDP [105] and SOCP [211]. As shown by [167], [41] and also by us, the integer quadratic program is equivalent to the quadratic optimization with continuous affine constraints, so the integer constraints on the solution can be dropped. It is interesting to look at connections between graph matching and graphical models with second-order potentials. It is also important to note that our algorithm and the one from [41] for MAP inference are inspired from their counterparts for solving graph matching [117] and [41].

Written as quadratic programs the only visible difference between the MAP inference and the combinatorial graph matching problem are the mapping constraints. Graph matching in its most general formulation in fact includes the MAP problem, because it could allow all kinds of constraints, including many-to-one or many-to-many and its unary and pairwise scores are more general since they do not have to follow any probabilistic interpretation. Graphical models on the other hand, having a probabilistic interpretation could be more powerful, but nothing prevents the scores from graph matching to be probabilistic. Another difference is the domain of applicability. While graphical models are applied to a

wider range of problems in machine learning, statistics and computer vision, graph matching is applied to problems that involve matching in a geometric sense. So, from this point of view, graph matching is in fact more restrictive and it can take advantage of more efficient solutions that are based on the unique statistical properties of geometric relationships, such as the accidental nature of the geometric alignments of incorrect correspondences. In theory at least, for most graph matching problems, one could use a graphical model representation and an algorithm for MAP inference. However, when thinking in terms of graphical models, the problem seems more complicated: the required inference and learning methods are usually more computationally expensive than, for example, spectral matching and its learning algorithm from Chapter 5. Therefore, instead of using the formalism of graphical models for problems involving geometric matching, we believe that it is much more practical to use the simpler but better suited solutions from the recent graph matching literature. Moreover, the solutions that work in graph matching with more dense graph structure and larger number of labels could potentially become, as shown by us here and by Cour and Shi [41], starting points for better solutions in graphical models with arbitrary graphs and large number of labels. While the MAP solutions related to Belief Propagation are better suited for causal models with a tree structure, the solutions inspired from graph matching are probably better for more dense, undirected graphs used for geometric reasoning.

The method we propose here for approximate MAP inference has two important properties: it does not impose any constraints on the first or second order terms and it converges to a solution that satisfies an optimality bound that is data independent.

Graph cuts have been successful in labeling tasks with regular energy functions [25], [99], [100], especially those related to low level vision. For binary labeling problems graph cuts are provably optimal. For multiple labels, the optimality bound given in [25] is data dependent and for arbitrary energy functions it could be arbitrarily far from the optimum. In contrast, our approach works with general energy functions.

Loopy Belief Propagation and its variants (*e.g.*, Tree Re-weighted Belief Propagation, closely related to Linear Programming Relaxation [221]), have also shown experimental success. The correctness and convergence of Loopy BP is not guaranteed for general graphs and energy functions and in some cases it does not converge to good approximations [151].

Other iterative optimization techniques for labeling classification problems such as deterministic annealing, self-annealing, self-annihilation [165] or relaxation labeling [88] are shown to improve the energy at each iteration [231], but there is no result as far as we know regarding their optimality properties.

Our algorithm is most related to the quadratic programming formulations from [41] and [157]. Related to our work [118] presented in this chapter, Cour and Shi [41] propose a quadratic programming relaxation of the MAP problem that follows ours very closely. They derive very similar data independent and data dependent optimality bounds and obtain a

discrete solution by following the same climbing procedure that we propose here. Their experiments, performed on synthetic data generated in the same way as ours show that the two methods perform very similarly. In [157] Parameswaran and Lafferty transform the original quadratic problem into a convex program that can efficiently be optimized globally. They make the objective function convex by adding large values to the unary potentials. Their method obeys an optimality bound that, as shown by [41], is in most cases looser than ours and [41] when the number of candidate labels per node is at least 3. It can also be shown that their bound gets worse as the density of the graph increases, since that requires larger values to be added to the unary terms. In contrast, the bounds we derive for our method get tighter as the density of the graph edges increases. In our experiments we show that, when compared to other methods, ours is the least sensitive to the graph structure, while keeping the same fast convergence rate.

Our work is also related to the optimization of polynomials with nonnegative coefficients under relaxed $L2$ constraints [10]. Our algorithm has two stages. In the first stage we follow a path similar to the one from [10] by relaxing the constraints on the solution (Section 3) and obtaining the exact optimum for the relaxed problem. In the second stage we show how to iteratively increase the energy at every step (not necessarily strictly), and how to obtain a solution respecting the original constraints, which is also guaranteed to be close to the optimum (Section 4). As a more efficient alternative to the climbing stage proposed here, the algorithm presented in the next Chapter, Integer Projected Fixed Point, can also be used. In Chapter 4 we also present an experimental comparison of the two procedures. IPFP, which could be seen as a parallel extension of Iterated Conditional Modes [17] with climbing and convergence properties, gives slightly better results than the climbing stage proposed here, while converging faster. The reader is referred to Chapter 4 for more details.

We are particularly interested in energy functions for arbitrary graphs with arbitrary number of labels. This is in contrast with recent studies of energy optimization tasks in computer vision, which are mainly focusing on weakly connected graphs such as trees or planar graphs [203]. More complex graphs with arbitrary second order potentials are important in higher level computer vision tasks, such as object recognition and scene analysis. Data dependent second order potentials are typically used in Conditional Random Fields (CRF) [106], [163]. In object recognition problems, the nodes could correspond to different object parts, while the second order potentials would describe how those parts interact given the data. At this level of representation, neither simple second-order potentials such as in the Potts model, nor simpler graphs structures such as planar graphs would be appropriate.

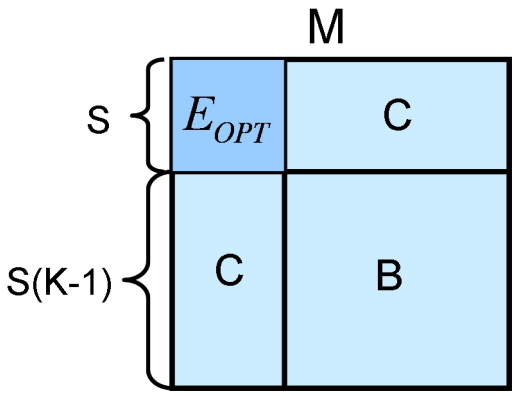

Figure 3.1: **M** has the second order potentials on its off diagonal elements and the first order elements on its diagonal.

3.1 Problem Formulation

We are addressing the problem of maximizing (minimizing) the score (energy) functions that typically arise in labeling problems. The following form of the energy function follows previous work such as deterministic annealing [165]:

$$E = 1/2 \sum_{ia;jb} x_{ia} x_{jb} Q_{ia;jb} + \sum_{ia} x_{ia} D_{ia}. \tag{3.1}$$

Here $Q_{ia;jb}$ corresponds to the higher order term describing how well the label a at site i agrees with the label b at site j, given the data. $Q_{ia;jb}$ could also be a smoothness term independent of the data (such as the Potts model) that simply encourages neighboring sites to have similar labels. We set $Q_{ia;jb} = 0$ if $i = j$ or if the sites i and j are not connected since $Q_{ia;jb}$ should describe connections between different sites. For each pair of site i and its possible label a, the first order potentials are represented by D_{ia}, which in general describes how well the label a agrees with the data at site i.

In this formulation $\mathbf{x}$ is required to be an indicator vector with an entry for each pair of $(site, label)$, such that if $x_{ia} = 1$ site i is assigned label a and $x_{ia} = 0$ otherwise. Thus, with a slight abuse of notation, we consider ia to be a unique index given to the pair site i and label a. Also, one site can be given one and only one label. These constraints are enforced by requiring $x_{ia} \in \{0, 1\}$ and $\sum_a x_{ia} = 1$. The vector $\mathbf{x}$ will be of dimension $S * L$, where S is the number of sites and L the number of labels for each site. To simplify notations we assume that for each site there is an equal number of labels, but our approach can accommodate different number of labels for each site.

It is convenient to write the expression for E in the matrix form $E = 1/2\mathbf{x^T Q x} + \mathbf{D x}$, where $Q(ia, jb) = Q_{ia;jb}$, $D(ia, ia) = D_{ia}$, and $Q(ia, ia) = 0$.

We assume that $\mathbf{Q}$ and $\mathbf{D}$ have non-negative elements, without loss of generality. That

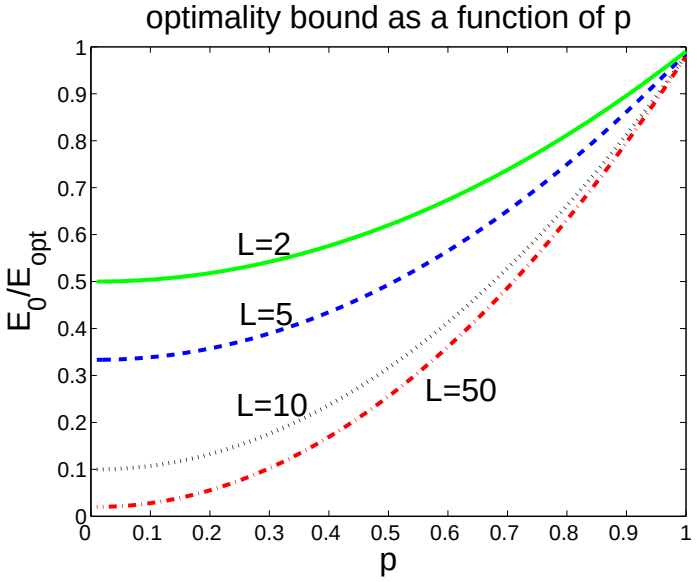

Figure 3.2: Optimality bound as we vary p, where p is the maximum $\in [0,1]$ such that $B \geq p(L-1)^2 E_{opt}$ and $C \geq p(L-1)E_{opt}$. The actual bound, with a more complicated mathematical formula, is actually tighter as shown in [41]. We chose its looser form only for simplicity.

is true because we could change them to have non-negative elements, without changing the global optimum solution of the energy under the integral constraints. More specifically: let c be the smallest element in both $\mathbf{Q}$ and $\mathbf{D}$ (we assume c is finite). Let $\mathbf{C_m}$ be a matrix of the same size as $\mathbf{Q}$, and $\mathbf{C_v}$ a vector of the same size as $\mathbf{D}$, both with constant elements equal to c, except for the diagonal of $\mathbf{C_m}$, which is set to zero . Due to the integral constraints on $\mathbf{x}$, we have $1/2\mathbf{x^T C_m x} + \mathbf{C_v x} = \frac{c*S*(S-1)}{2} + cS$, which is independent of $\mathbf{x}$. Therefore $\mathbf{x}^*$ that maximizes E also maximizes $1/2\mathbf{x^T}(\mathbf{Q} - \mathbf{C_m})\mathbf{x} + (\mathbf{D} - \mathbf{C_v})\mathbf{x}$. Next, we can redefine $\mathbf{Q}$ as $\mathbf{Q} - \mathbf{C_m}$ and $\mathbf{D}$ as $\mathbf{D} - \mathbf{C_v}$, so that both have only non-negative elements. The non-negativity of $\mathbf{Q}$ also brings a practical advantage, especially when most of its smallest elements are very close to 0. Those elements could be set to 0 at the cost of a small change in the energy, with the benefit of a significant decrease in memory cost (since $\mathbf{Q}$ should be stored as a sparse matrix).

We attack the problem in two stages. During the first step, we relax the constraints to $\sum_a x_{ia}^2 = 1$, and find the global optimum of the polynomial with non-negative coefficients given by our energy function. The procedure is extremely fast, being very similar to the iterative power method for finding the largest eigenvector of the matrix $\mathbf{Q}$ and it usually converges in a few iterations (Section 3). Then we map the relaxed global optimum on the simplex given by $\sum_a x_{ia} = 1$ and show that the energy thus obtained is close to the global

maximum (Section 3.1)

This gives us a good starting point for the second stage when we follow an iterative procedure that is guaranteed to increase the energy after every iteration (not necessarily strictly) and converge to a solution that obeys the initial integral constraints.

3.2 Global Optimum under Relaxed Constraints

We start by globally maximizing the energy function E under the relaxed constraints $\sum_a x_{ia}^2 = 1$. Introducing Lagrange multipliers we obtain the free energy as:

$$F = 1/2\mathbf{x^T Q x} + \mathbf{D x} + \sum_i \lambda_i (\sum_a x_{ia}^2 - 1). \tag{3.2}$$

Setting the partial derivatives with respect to both $\mathbf{x}$ and the parameters λ_i to zero, and solving for the Lagrange multipliers, we obtain:

$$x_{ia}^* = \frac{\sum_{jb} Q_{ia;jb} x_{jb}^* + D_{ia}}{\sqrt{\sum_{b=1}^{L} x_{ib}^{*\,2}}}. \tag{3.3}$$

This equation looks similar to the eigenvector equation $\mathbf{Qx} = \lambda\mathbf{x}$ if it were not for the vector $\mathbf{D}$ and the site-wise normalization instead of the global one which applies to eigenvectors. Starting from a vector with positive elements, the fixed point $\mathbf{x}^*$ of the above equation has positive elements, is unique and it is a global maximum of the energy E under the constraints $\sum_a x_{ia}^2 = 1$, due to the fact that $\mathbf{Q}$ and $\mathbf{D}$ have non-negative elements (Theorem 5, [10]).

3.3 Mapping the Relaxed Solution on the Simplex

Let E^* be the optimal energy $E(\mathbf{x}^*)$, E_0 the energy after bringing $\mathbf{x}^*$ on the simplex $\sum_a x_{ia} = 1$, and E_{opt} the true optimal for the original integral constraints. From vector $\mathbf{x}^*$ we can obtain a vector $\mathbf{x_0}$ that lies on the simplex $\sum_a x_{ia} = 1$, by setting $x_{0ia} = x_{ia}^* / \sum_b x_{ib}^*$. Next we show that the energy E_0 evaluated at $\mathbf{x_0}$ satisfies $E_0 \geq \frac{1}{L} E^* \geq \frac{1}{L} E_{opt}$, where L is the number of labels. This is a very loose lower bound, but it is important since it does not depend on the actual energy function.

We start by expressing E_0 in terms of $\mathbf{x}^*$:

$$E_0 = \frac{1}{2} \sum_{ia;jb} \frac{x_{ia}^*}{\sum_b x_{ib}^*} \frac{x_{jb}^*}{\sum_b x_{jb}^*} Q_{ia;jb} + \sum_{ia} \frac{x_{ia}^*}{\sum_b x_{ib}^*} D_{ia}. \tag{3.4}$$

We know that $\mathbf{x}^*$ has non-negative elements and it satisfies $\sum_b x^{*2}_{ib} = 1$, therefore we have $\sum_b x_{ib}^* \leq \sqrt{L}$. If we let $k = max_i(\sum_b x_{ib}^*)$, it immediately follows that

$$E_0 \geq \frac{1}{k^2}(\frac{1}{2}\sum_{ia;jb} x^*_{ia}x^*_{jb}Q_{ia;jb} + \sum_{ia} x^*_{ia}D_{ia}) \geq \frac{1}{k^2}E^*. \tag{3.5}$$

We must also have $E_{opt} \leq E^*$, since any solution satisfying the original constraints also satisfy the relaxed ones, and E^* is the global optimum over the relaxed constraints. Thus we obtain $E_0 \geq \frac{1}{k^2}E_{opt}$, where $k \in [1, \sqrt{L}]$. Obviously this bound is very loose since we replaced the sums over the elements of $\mathbf{x}^*$ for each site by their largest possible value. However, the bound reflects the desirable property that the more peaked the elements in $\mathbf{x}^*$ are, the lower the sums $\sum_b x^*_{ib}$, and thus the closer E_0 will be to E_{opt}. One can find tighter bounds if the actual second and first order potentials are taken in consideration, as we will show next.

3.4 Data Dependent Lower Bound

For better understanding how far E_0 is from E_{opt} it is useful to define the matrix $\mathbf{M}$ such that $\mathbf{M} = \mathbf{\frac{1}{2}Q + ID}$, where $\mathbf{ID}$ is the diagonal matrix with $ID_{ia;ia} = D_{ia}$. Then we have $M_{ia;jb} = \frac{1}{2}Q_{ia;jb}$ for any $i \neq j$ and $M_{ia;ia} = D_{ia}$. It follows that:

$$E(\mathbf{x}) = 1/2\mathbf{x^T Qx} + \mathbf{Dx} \geq \mathbf{x^T Mx}. \tag{3.6}$$

For clarity, let us assume, without loss of generality, that the elements of the optimal labeling $\mathbf{x_{opt}}$ have been permuted such that $\mathbf{x_{opt}}$ has ones on the first S elements and zeroes everywhere else, which implies the corresponding permutation of the rows and columns of $\mathbf{M}$ (Figure 3.1). Now the sum of all elements in the upper left S by S block of the matrix $\mathbf{M}$ is equal to the optimal energy E_{opt}. B and C are the sums over all elements in the corresponding sub-blocks of $\mathbf{M}$ as shown in Figure 3.1. Now let p be the maximum in $[0, 1]$, such that the average element in block B and the average element in block C are both greater or equal to p times the average element in E_{opt}. After considering the number of elements in B and C relative to E_{opt} it follows that $B \geq p(L-1)^2E_{opt}$ and $C \geq p(L-1)E_{opt}$. Such a p always exists since B and C have only non-negative elements. We will show that the larger p is, the closer the energy E_0 is to the optimal energy E_{opt}. The following inequality/bound can be made tighter as shown in [41]. We present here its looser version because it has a simpler form:

Inequality 1: For any p as defined previously $E_0 \geq \frac{1+(L-1)p^2}{L}E_{opt}$.

Proof: Let E_q be the the energy evaluated at a vector $\mathbf{x}$ constructed as a function of $q \geq 0$: $x_{ia} = \frac{1}{\sqrt{1+q^2(L-1)}}$ if a is an optimal assignment and $x_{ia} = \frac{q}{\sqrt{1+q^2(L-1)}}$ otherwise. Clearly, $\mathbf{x}$ satisfies the relaxed constraints $\sum_a x^2_{ia} = 1$. Then (using inequality 6) we have:

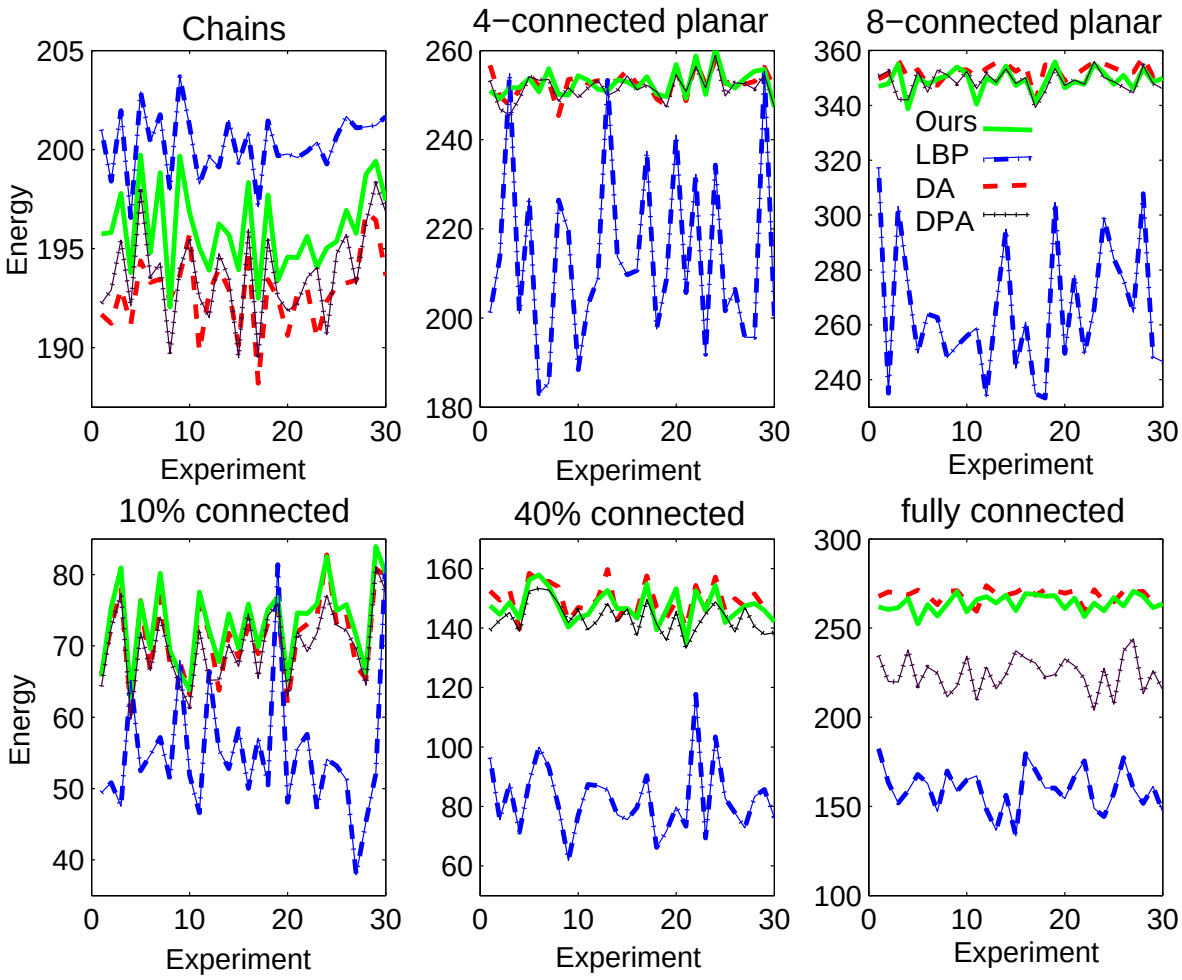

Figure 3.3: Results on 30 random experiments for different degree of graph connectedness. Top row: 100 nodes, 10 labels. Bottom row: 30 nodes, 30 labels. $p_c = 0.8$, $p_w = 0.4$, 180 iterations for each algorithm. Best viewed in color

$$E_q \geq \mathbf{x_q^T M x_q} = \frac{E_{opt} + 2qC + q^2B}{1 + q^2(L-1)}. \tag{3.7}$$

Since $B \geq p(L-1)^2 E_{opt}$ and $C \geq p(L-1)E_{opt}$ it follows that:

$$E_q \geq \frac{(1 + 2qp(L-1) + q^2p(L-1)^2)E_{opt}}{1 + q^2(L-1)}. \tag{3.8}$$

Since $E_q \leq E^*$ and $p \in [0, 1]$ we have

$$E^* \geq \frac{(1 + pq(L-1))^2}{1 + q^2(L-1)} E_{opt}. \tag{3.9}$$

The right hand side of the above inequality is maximized for $q = p$. Thus, if we use $E_0 \geq \frac{1}{L}E^*$ we have our bound:

$$E_0 \geq \frac{1 + (L-1)p^2}{L} E_{opt}. \tag{3.10}$$

The bound approaches 1 as p approaches 1 and it is always greater or equal to $1/L$. The curve becomes less and less sensitive to the number of labels as L becomes very large (Figure 3.2). Also, the lower p is the looser the bound. As future work, it would be interesting to explore tighter bounds depending on B and C relative to E_0. The less interesting result (easy to show) is that when both B and C approach zero, E_0 approaches E_{opt}.

In the next Section we show how, by starting from $\mathbf{x_0}$ we follow a coordinate ascent approach in which we are guaranteed to increase the energy at every iteration until we reach a solution that satisfies the original constraints $\sum_a x_{ia} = 1$ and $x_{ia} \in \{0, 1\}$

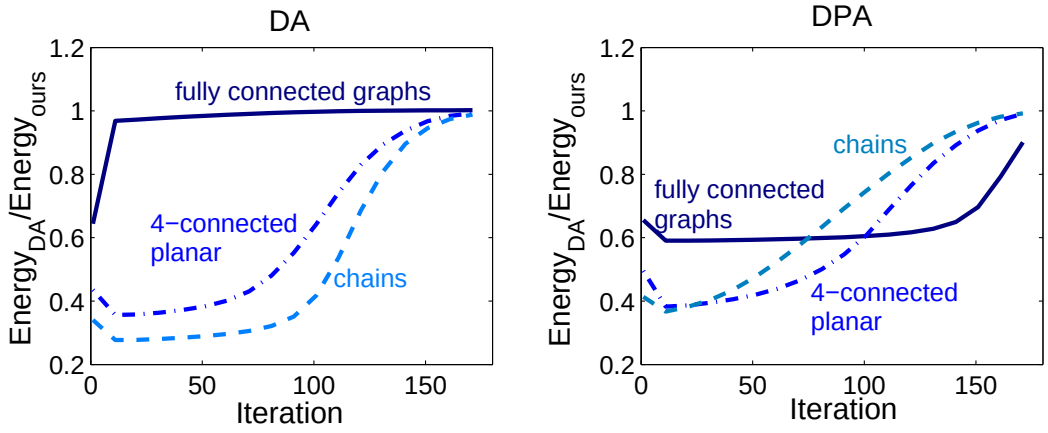

Figure 3.4: Average ratio E_{DA}/E_{ours} (left), E_{DPA}/E_{ours} (right) per iteration for 30 experiments. Standard deviation (not plotted) was always less than 0.05. nNodes =100, nLabels = 10.

3.5 Climbing Stage

Starting from an energy level E_0, which obeys an optimality bound, we continue through a special coordinate ascent procedure that guarantees both convergence and the satisfaction of the integral constraints. We show how any update rule that obeys Inequality 2 (defined later in this Section) is guaranteed to improve the energy at every step. We also give a general form of such update rules.

This stage is somehow similar to previous methods such as deterministic annealing, self-annealing, relaxation labeling and Iterative Conditional Modes (ICM). Among these only

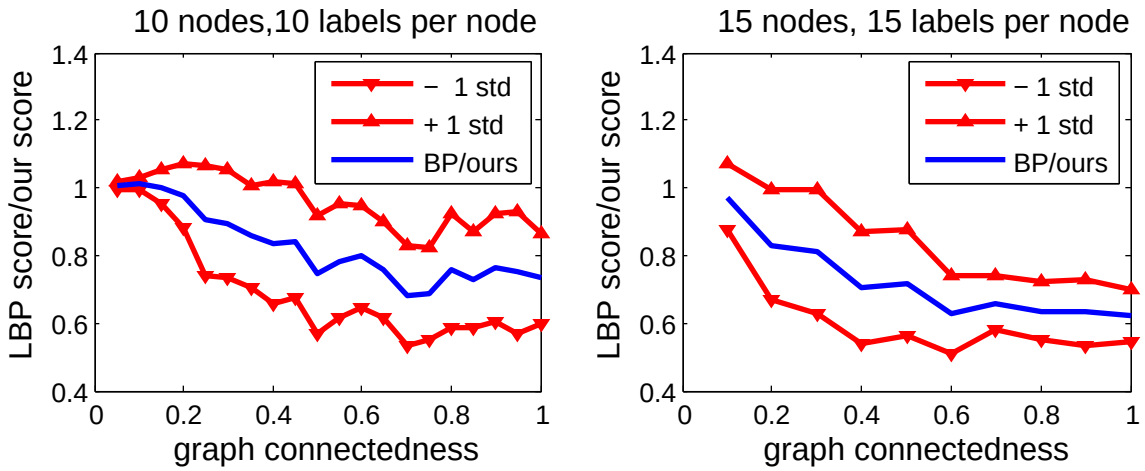

Figure 3.5: Mean and std values for E_{LBP}/E_{ours} for varying degree of connectedness, over 30 experiments.

ICM is a coordinate ascent method. While all these methods could have arbitrary starting points, we start this stage at a point that is independent of our initial conditions, since it is the unique global optimum of the relaxed problem from the previous stage. Therefore, we can regard the previous stage as an initialization procedure, which is appropriate since it respects certain optimality bounds.

The algorithm outline is:

1. Stage 1: find $\mathbf{x}^*$ that maximizes $1/2\mathbf{x^T Q x} + \mathbf{Dx}$, given $\sum_a x_{ia}^2 = 1$ by iterating until convergence:

   a) let $\mathbf{x} \leftarrow \mathbf{Qx} + \mathbf{D}$

   b) normalize $\mathbf{x}$ per site: $x_{ia} = \frac{x_{ia}}{\sqrt{\sum_b x_{ib}^2}}$

2. Normalize $\mathbf{x}$ found at step 1, such that $x_{ia} = x_{ia}^* / \sum_b x_{ib}^*$

3. Stage 2: Set $\beta = 1$ and repeat until convergence

   a) set $v_{ia} = \sum_{jb} Q_{ia;jb} x_{jb}^{(t)} + D_{ia}$

   b) $x_{ia}^{(t+1)} = \sigma_i x_{ia}^{(t)} F(v_{ia}, \beta)$, where $\sigma_i = 1/\sum_b x_{ib}^{(t)} F(v_{ib}, \beta)$

   c) increase β after updating $\mathbf{x}$ for all sites

We show that if the assignments at step 3.b are done site-wise (that is, vector $\mathbf{x}$ is updated sequentially) and $F(v, \beta)$ is any positive, monotonically increasing function of v, the energy increases at every site-wise update until convergence. If F is increasing exponentially with β then, by increasing β, we make the assignment at step 3.2 approach a

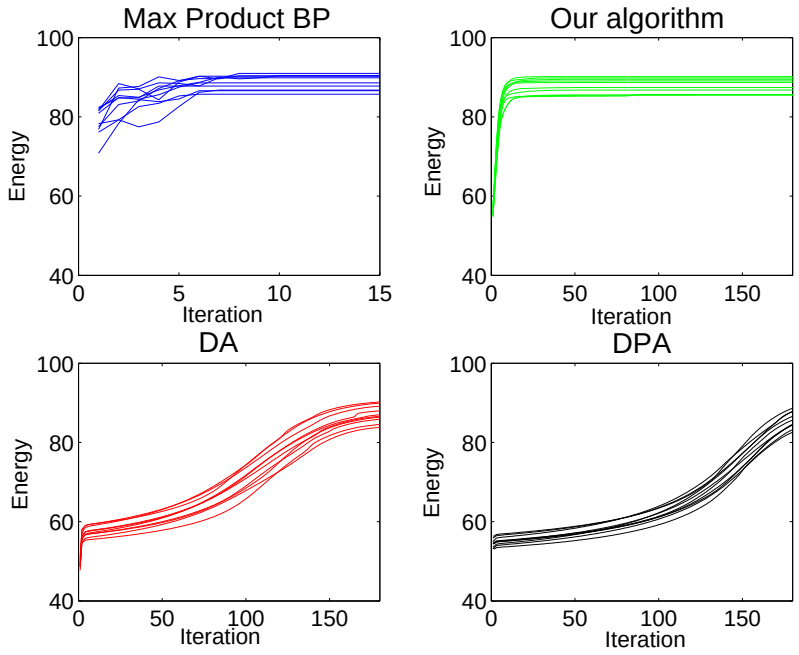

Figure 3.6: Experiments on chains, with 49 sites and 5 labels per site. The energy per iteration is plotted on the same 10 random experiments for each algorithm, with all parameters held constant.

max function such that $\mathbf{x}$ gets closer and closer to the original integral constraints. At the last step we actually set $x_{ia} = 1$ for which v_{ia} is maximum over all v_{ib}'s and 0 otherwise, thus satisfying the original integral constraints (the very last iteration is basically identical to ICM, also guaranteed to increase the energy).

Since every time we visit a site i we only update the values in vector $\mathbf{x}$ corresponding to that site, we can write the energy at that moment t as $E^{(t)} = E^{(t)}_{const} + E^{(t)}_i$, where $E^{(t)}_{const}$ is independent of x_{ia} for any label a and $E^{(t)}_i$ is a function of the variables x_{ia}. Therefore, it suffices to show that after updating the x_{ia}'s we increase the component E_i. One can show that $E^{(t)}_i$ can be expressed as:

$$E^{(t)}_i = \sum_a x^{(t)}_{ia} \Big(\sum_{jb; j \neq i} x^{(t)}_{jb} Q_{ia;jb} + D_{ia} \Big) = \sum_a x^{(t)}_{ia} v_{ia}. \tag{3.11}$$

In the formula above the v_{ia}'s are independent of the x_{ia}'s, thus they do not change after the update. Therefore it is left to show that $\sum_a x^{(t)}_{ia} v_{ia} \leq \sum_a x^{(t+1)}_{ia} v_{ia} = \sum_a \sigma_i x^{(t)}_{ia} F(v_{ia}, \beta) v_{ia}$. The proof is based on the following inequality:

Inequality 2: Given any non-negative arrays b_q, w_q, and w^*_q, with $q = 1...n$ and $\sum_q w_q > 0$, $\sum_q w^*_q > 0$, such that $\frac{w^*_q}{w^*_k} \geq \frac{w_q}{w_k}$ whenever $b_q \geq b_k$, the following inequality holds: $\frac{\sum_q w^*_q b_q}{\sum_q w^*_q} \geq \frac{\sum_q w_q b_q}{\sum_q w_q}$

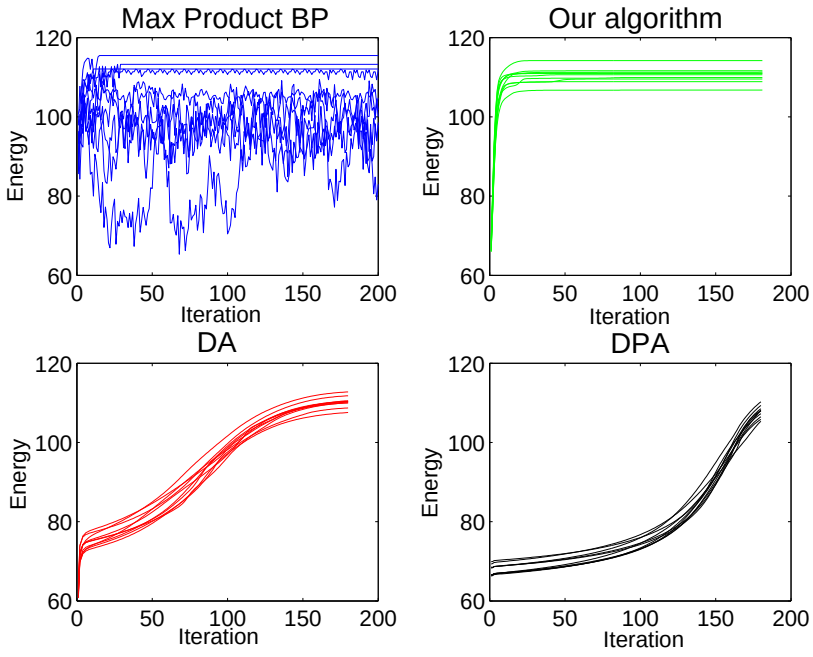

Figure 3.7: Experiments on planar graphs with 4-connected neighborhoods, with 49 sites and 5 labels per site. The energy per iteration is plotted on the same 10 random experiments for each algorithm, with all parameters held constant.

Proof: The proof relies on the fact (not proved here) that there must exists a k between 1 and n such that $\frac{w_q^*}{\sum_r w_r^*} \geq \frac{w_q}{\sum_r w_r}$ for any q such that $b_q \geq b_k$ and $\frac{w_q^*}{\sum_r w_r^*} \leq \frac{w_r}{\sum_q w_r}$ otherwise.

This inequality applies immediately to our problem if we set $w_a^* = x_{ia}^{(t)} F(v_{ia}, \beta)$, $w_a = x_{ia}^{(t)}$ and $b_a = v_{ia}$. In our experiments we used $F(v, \beta) = exp(\beta v)$. Other functions could be easily designed, such as $F(v, \beta) = \lambda + \gamma v^\beta$ with positive λ, γ (this is a generalization of the usual relaxation labeling update, also related to [12]). In experiments, both choices of F behaved very similarly.

3.6 Experimental Analysis

We compared our algorithm against other methods such as Max Product Loopy Belief Propagation (commonly used for MAP estimation), Deterministic Annealing (DA), Self-Annealing (SA), ICM, Deterministic Pseudo-Annealing (DPA) and Relaxation Labeling [165], [16], [17], [88]. In our experiments, our algorithm converges faster, while performing at least as accurately as the above mentioned methods. We show comparative results among BP, DA, DPA and ours.

For a more thorough analysis we generate synthetic energy functions. We pay more

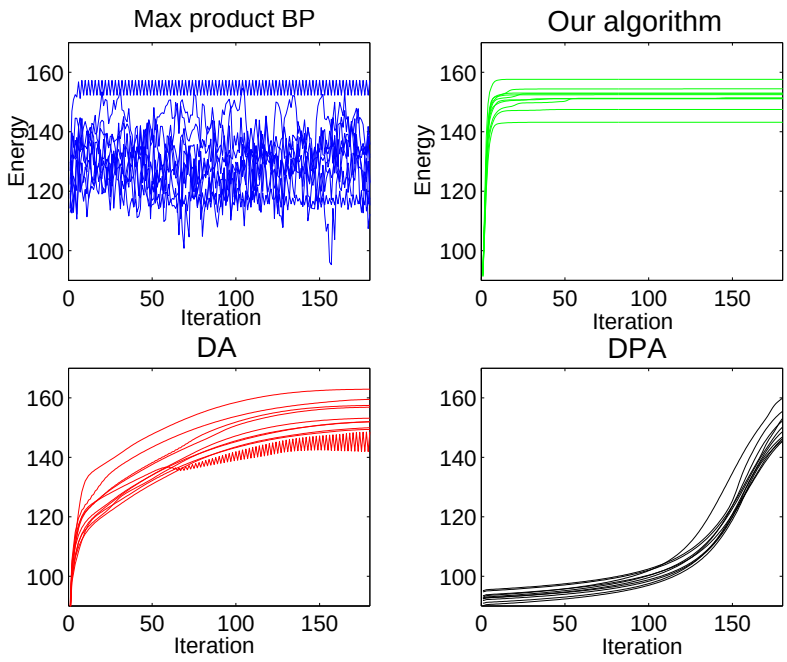

Figure 3.8: Experiments on planar graphs with 8-connected neighborhoods, with 49 sites and 5 labels per site. The energy per iteration is plotted on the same 10 random experiments for each algorithm, with all parameters held constant.

attention to the degree of connectedness and to the number of labels relative to the number of nodes. For non-planar graphs we generate their structure by picking an edge between any two sites with a certain probability p_{edge}. The first order potentials (as used in the product form of the Max-Product BP) are generated as uniform variables in the interval $[\epsilon, 1]$. Then, for each site we assume label number 1 to be the correct one, without loss of generality. Next we randomly group the sites into a number of disjoint sets (simulating the case when multiple objects or regions have to be classified simultaneously, a setup that relates to the discontinuity preserving energy functions from stereo). For pairs of connected sites (i, j), and the uniform random variable $p \in [\epsilon, 1]$, we set $Q_{ia;jb} = log(p/\epsilon)$ with probability p_0 and $Q_{ia;jb} = 0$ otherwise. If $a = b = 1$ and the pair of sites (i, j) are in the same set we have $p_0 = p_c$, otherwise we set $p_0 = p_w$ (for $p_c > p_w$). Thus we encourage second-order potentials between pairs of *correct* labels at sites in the same set to be on average larger than the rest of potentials. All experiments plotted on the same graph are generated using fixed parameters (e.g. number of nodes, labels per node, p_c, p_w, p_{edge}).

We ran experiments with different number of nodes and labels, different degree of connectedness and different values of p_c and p_w. Here we only present sets of representative results (Figure (3.3)). Our algorithm, DA, and DPA use a parameter β that increases at each iteration ($\frac{1}{\beta}$ is known as the temperature in annealing approaches, which reaches 0

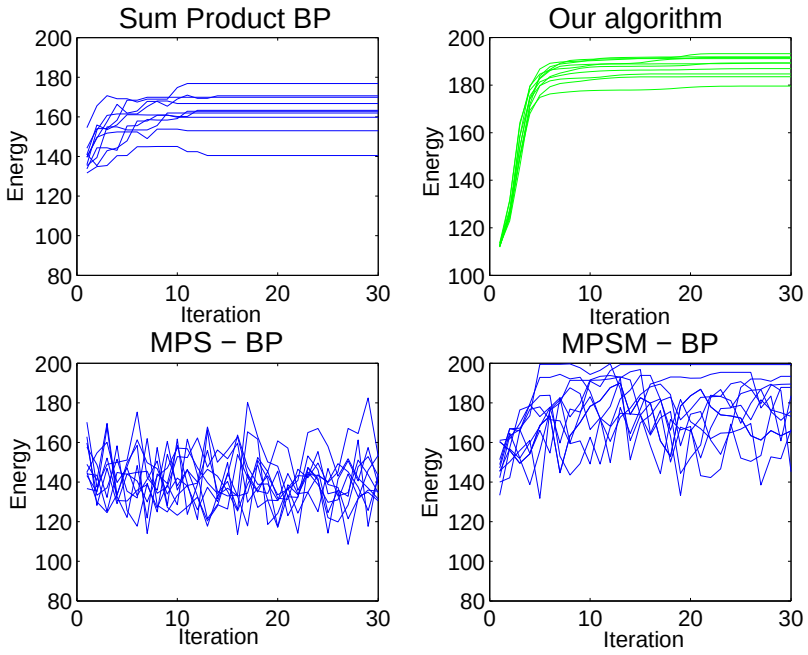

Figure 3.9: Experiments on fully connected graphs, 25 sites and 25 labels per site. The energy per iteration is plotted for the same 10 random experiments for each algorithm (all parameters held constant).

at the last iteration). For all three algorithms we use exactly the same β values at each iteration, such that $1/\beta$ decreases from 1 to 0.01 in equal steps. Also, for all algorithms we used the uniform $\mathbf{x}$ as the default initialization. Throughout the experiments our algorithm was always among the top performers along with DA, while converging much faster than both DA and DPA (Figure 3.6, 3.7, 3.8).

Belief Propagation is provably optimal for trees, but it has been applied successfully to graphs with loops. We compare our algorithm to Max Product BP and show that the performance of our algorithm is comparable with BP on trees, while consistently giving solutions of higher energies on highly connected graphs (Figures 3.5 and 3.3). When the number of labels is comparable to the number of nodes, the performance of Max Product BP starts degrading significantly as we increase the degree of connectedness of the graphs. In Figure 3.5 we plot the mean ratio and standard deviation of BP energy vs. our energy over 30 experiments for different degrees of graph connectedness. The probability of edge generation ranges from 0.1 to 1 (fully connected). Both algorithms were allowed to run for a maximum of 30 iterations. We also observed that for a given degree of connectedness, Loopy BP's output energy degrades relative to ours, as we increase the number of nodes.

If it converges, Max Product Loopy BP is guaranteed to give a labeling with a larger energy than other assignments within a large neighborhood around that labeling (Weiss and

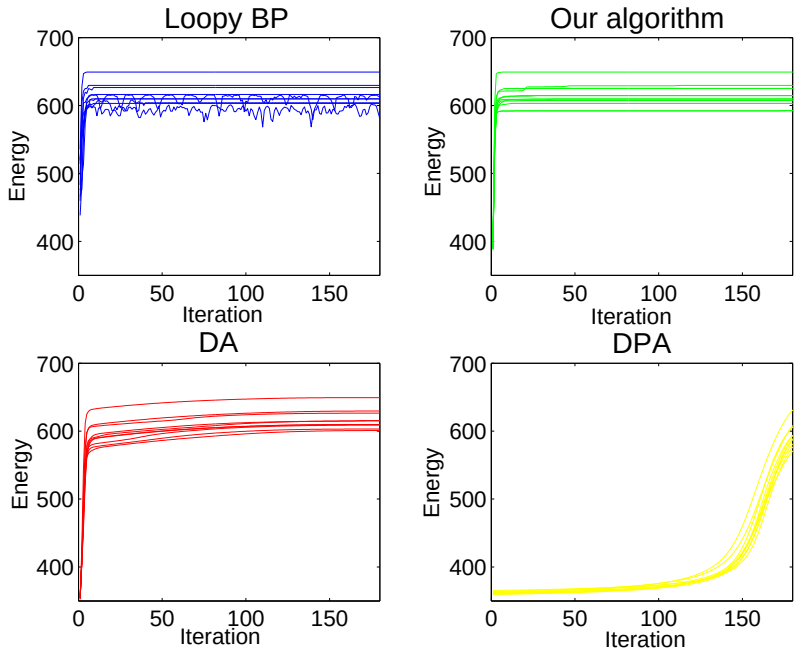

Figure 3.10: Experiments on fully connected graphs, with 49 sites and 5 labels per site. The energy per iteration is plotted on the same 10 random experiments for each algorithm, with all parameters held constant.

Table 3.1: Results for fully connected graphs over 30 experiments (maximum of 30 iterations)

$nNodes$ ($= nLabels$)	avg of E_{BP}/E_{ours}	std of E_{BP}/E_{ours}
10	0.73	0.13
15	0.62	0.08
30	0.60	0.05

Freeman, 2001). While this is an interesting theoretical result, it does not tell us anything about how often Loopy BP converges. In our experiments we actually found that Loopy BP rarely converges (Figures 3.6, 3.7, 3.8). We also noticed a very similar instability when experimenting with BP with sequential updates ($MPS-BP$) and momentum ($MPSM-BP$) for damping oscillations. The most stable version of BP was the Sum Product BP, with momentum and sequential updates ($SPSM-BP$) (Figure 3.9).

Our algorithm has an excellent average convergence performance, while seeming to be insensitive to the structure of the graph. DPA and DA also seem to monotonically increase the energy after each iteration but their convergence is more sensitive to the graph structure.

Table 3.2: Results for 8-connected planar graphs, 25 labels and 25 nodes over 30 experiments (maximum of 30 iterations)

BP $version$	avg of E_{BP}/E_{ours}	std of E_{BP}/E_{ours}
$MP-BP$	0.58	0.05
$MPS-BP$	0.71	0.09
$MPSM-BP$	0.93	0.08
$SPSM-BP$	0.75	0.05

They maintained the same behavior even when the updates were done site-wise. We believe that the performance of our algorithm is due to the combination of its two steps, which have different roles that seem to complement each other. The role of the first step is to move the solution into the right direction with respect to the global optimum, as suggested by the optimality bounds. The next step improves the energy very rapidly at every iteration.

DA, DPA and our method require the same amount of memory, and roughly the same number of operations per iteration (= one updating of the vector $\mathbf{x}$ for all sites). In practice, our method takes fewer iterations to converge, which makes it more efficient. BP needs extra memory because it has to store about $nEdges * nLabels$ messages. Also, per iteration, BP needs to update all messages, while the other methods update only the $nNodes * nLabels$ beliefs in $\mathbf{x}$. Our method took roughly the same number of iterations as BP to converge, which combined with its cheaper cost per iteration, makes it roughly $O(\frac{nEdges}{nNodes})$ times faster. For example, for a fully connected graph with 30 nodes and 30 labels per node, the $Matlab$ implementation of our algorithm took about 0.5 sec. on a laptop PC, while the BP implementation that we used (Kevin Murphy's Bayes Net Toolbox) took about 12 sec on the same problem.

3.7 Conclusions

We have presented an efficient approximate algorithm for energy optimization that has certain theoretical properties for arbitrarily structured graphs with arbitrary energy functions. Our experiments illustrate that our approach converges faster and it is less sensitive to the structure of the graphs than other existing methods, while being at least as accurate. For future work, it is worth investigating tighter theoretical bounds, that could explain better the high efficiency of our algorithm.

Chapter 4

Obtaining a Discrete Solution

In this Chapter we bring graph matching and MAP inference problems together by introducing a novel algorithm, Integer Projected Fixed Point (IPFP), that can accommodate both tasks and solve them efficiently. Moreover, this algorithm addresses a common issue of both graph matching and MAP inference in MRFs, which is that of obtaining a discrete solution, given the continuous or probabilistic solution for matching and inference. Examples of such continuous solutions are the marginal probabilities given by Belief Propagation and its variants, or the optimal solutions in the relaxed continuous domain given by graph matching algorithms such as our spectral matching [117], spectral matching with affine constraints [43], or probabilistic graph matching [233]. Graph matching with pairwise constraints and MAP inference for MRFs have recently been formulated as integer quadratic programs, where obtaining an exact solution is computationally intractable. We present a novel algorithm, Integer Projected Fixed Point (IPFP), that efficiently finds approximate solutions to such problems. In this chapter we focus on graph matching, because it is in this area that we have extensively compared our algorithm to state-of-the-art methods. Graph matching with pairwise constraints is NP-hard, and a lot of effort has been spent in finding good approximate solutions by relaxing the integer one-to-one constraints, such that the continuous global optimum of the new problem can be found efficiently. Little computational time is spent in order to binarize the solution, based on the assumption that the continuous optimum is close to the discrete global optimum of the original combinatorial problem. Spectral matching takes advantage of the particular properties of geometric matching, and, based on certain assumptions that were discussed previously, it expects the optimum of the relaxed problem to be close to the optimum of the original problem with integer constraints. In this Chapter we show experimentally that in practice the solution returned by current state-of-the-art algorithms, including spectral matching, can be significantly improved if special care is taken during the post-processing, binarization step. The results presented in this chapter strongly suggest that the optimum solution of the relaxed

problem should be used only as a initialization step, which must be followed by a careful search for the discrete solution that maximizes the quadratic score. As a consequence, we shed new light on the importance of the two stages: if until now we have seen the second, post-processing step merely as a procedure to obtain a binary solution, from now on we should give this stage equal importance after observing that this second step, if designed properly, can in fact greatly improve the performance.

Here we propose an iterative algorithm that takes as input any continuous or discrete solution, possibly given by some other graph matching method, and quickly improves it by aiming to optimize the original problem with its integer constraints. Each iteration consists of two stages: the first one optimizes in the discrete domain a linear approximation of the quadratic function around the current solution, which gives a direction along which the second stage maximizes the original quadratic score in the continuous domain. Even though this second stage might find a non-discrete solution, the optimization direction given by the first stage is always towards an integer solution, which is often the same one found in the second stage. The algorithm always improves the quadratic score in the continuous domain finally converging to a maximum. If the quadratic function is convex the solution at every iteration is always discrete and the algorithm converges in a finite number of steps. In the case of non-convex quadratic functions, the method tends to pass through/near discrete solutions and the best discrete solution encountered along the path is returned, which, in practice is either identical or very close to the point of convergence. We have performed extensive experiments with our algorithm with excellent results, the most representative of which being shown in this chapter. Our method clearly outperforms four state-of-the-art algorithms, and, when used in combination, the final solution is dramatically improved. Some recent MAP inference algorithms for Markov Random Fields [41], [157], including ours [118], formulate the problem as an integer quadratic program, for which our algorithm is also well suited, as we later explain and demonstrate in more detail.

Connections to Frank-Wolfe Algorithm IPFP is related to the Frank-Wolfe method (FW) [71], a classical optimization algorithm from 1956 most often used in operation research. Even though FW is used in a completely different area, and, when designing IPFP, we were not aware of its existence, it is important to briefly discuss the connection between the two methods, while pointing out the differences.

The Frank-Wolfe method is applied to convex programming problems with linear constraints. If the problem is convex FW is guaranteed to find the global optimum, albeit very slowly. Similarly to IPFP, FW, at each iteration, first maximizes the linear approximation of the original function, followed by a line search in which the original function is maximized. A well-known fact about FW, is that it has slow convergence rate around the optimum, which is why in practice it is stopped earlier for obtaining an approximate

solution. In contrast, in the case of IPFP (which is applied to MAP/matching problems) the local optimum is most often discrete (for MAP it is always discrete). When the solution is discrete the optimum is actually found during the optimization of the linear approximation, when the discrete point is found, so the convergence is immediate. This fact is also demonstrated in our experiments, where IPFP most often converges fast. For matching (one-to-one constraints), it could happen that the optimum is continuous, but in that case it can be stopped early without any loss, since we are interested in a discrete solution and not in the true continuous optimum. Therefore, unlike FW, IPFP finds the solution in very few iterations, which is an important advantage.

Another interesting difference is that we apply IPFP to problems that are close to integer concave minimization, and not to convex problems, as it is the case with FW. Integer concave minimization is also NP-hard, thus much more difficult than convex minimization. Also FW has been rarely used for solving the quadratic assignment (QAP) problem [27], which is related to graph matching, but usually arises in operations research.

The recent work of Zalasvskiy et al. [232] is related to the material described in this chapter. They also attempt to use the Frank-Wolfe technique to derive an efficient algorithm. At a high level, the differences between [232] and our treatment of IPFP can be summarized in two points. First, [232] is formulated in the context of traditional graph matching in which the number of overlapping edges between the two graphs is maximized, whereas we assume a general formulation of graph matching as a QAP. Using the general formulation enables us to apply the exact same algorithm to MAP inference problem, for example. Second, [232] applies a solver based on Frank-Wolfe to a series of intermediate problems which are designed such that their solutions are likely to converge to the solution of the original problem, exploiting convexity/concavity properties of the intermediate problems which are consequences of the traditional graph matching formulation, while we apply our IPFP solver directly to the original problem. A thorough comparison of the algorithms' performance and of their theoretical properties remains to be done in future work.

Matching Using Pairwise Constraints Here we review the graph matching problem (also presented in previous Chapters), in its most recent and general form, which consists of finding the indicator vector $\mathbf{x}^*$ that maximizes a certain quadratic score function:e

Problem 1:

$$\mathbf{x}^* = \text{argmax}(\mathbf{x^T M x}) \text{ s. t. } \mathbf{Ax = 1},\ \mathbf{x} \in \{\mathbf{0,1}\}^{\mathbf{n}}, \tag{4.1}$$

given the one-to-one constraints $\mathbf{Ax = 1}$, $\mathbf{x} \in \{0,1\}^n$, which require that $\mathbf{x}$ is an indicator vector such that $x_{ia} = 1$ if feature i from one image is matched to feature a from the other image and zero otherwise. Usually one-to-one constraints are imposed on $\mathbf{x}$ such that one feature from one image can be matched to at most one other feature from the other image.

In MAP inference problems, only many-to-one constraints are usually required, which can be accommodated by the same formulation, by appropriately setting the constraints matrix **A**. In graph matching, **M** is usually a symmetric matrix with positive elements containing the compatibility score functions, such that $M_{ia;jb}$ measures how similar the pair of features (i, j) from one image is in both local appearance and pair-wise geometry with the pair of their candidate matches (a, b) from the other image. The difficulty of Problem 1 depends on the structure of this matrix **M**, but in the general case it is NP-hard and no efficient algorithm exists that can guarantee optimality bounds. Previous algorithms modify Problem 1, usually by relaxing the constraints on the solution, in order to be able to find efficiently optimal solutions to the new problem. For example, spectral matching [117] (SM) drops the constraints entirely and assumes that the leading eigenvector of **M** is close to the optimal discrete solution. It then finds the discrete solution **x** by maximizing the dot-product with the leading eigenvector of **M**. The assumption is that **M** is a slightly perturbed version of an ideal matrix, with rank-1, for which maximizing this dot product gives the global optimum. Later, spectral graph matching with affine constraints was developed [43] (SMAC), which finds the optimal solution of a modified score function, with a tighter relaxation that imposes the affine constraints $\mathbf{Ax} = \mathbf{1}$ during optimization. A different, probabilistic interpretation, not based on the quadratic formulation, is given in [233] (PM), also based on the assumption that **M** is close to a rank-1 matrix that is the outer product of the vector of probabilities for each candidate assignment. An important observation is that none of the previous methods are concerned with the original integer constraints during optimization, and the final post processing step, when the continuous solution is binarized, is usually a simple procedure. It is simply assumed that the continuous solution is close to the discrete one. The algorithm we propose here optimizes the original quadratic score in the continuous domain obtained by only dropping the binary constraints, but it always moves towards discrete solutions through which it passes most of the time. Note that even in this continuous domain the quadratic optimization problem is NP-hard, so we cannot hope to get any global optimality guarantees. But we do not lose much, since, guaranteed global optimality for a relaxed problem does not require closeness to the global optimum of the original problem. A fact that is evident in most of our experiments. Our experimental results from Section 4 strongly suggest an important point: algorithms with global optimality properties in a loosely relaxed domain can often give relatively poor results in the original domain, and a well-designed procedure with local optimality properties in the original domain, such as IPFP, can have a greater impact on the final solution than the global optimality in the relaxed domain.

Our algorithm aims to optimize the following continuous problem, in which we only drop the integer constraints from Problem 1:

Problem 2:

$$\mathbf{x}^* = \text{argmax}(\mathbf{x}^\mathbf{T}\mathbf{M}\mathbf{x}) \text{ s. t. } \mathbf{A}\mathbf{x} = \mathbf{1},\ \mathbf{x} \geq \mathbf{0}. \tag{4.2}$$

Note that Problem 2 is also NP-hard, and it becomes a concave minimization problem, equivalent to Problem 1, when $\mathbf{M}$ is positive definite.

4.1 Algorithm

We introduce our novel algorithm, Integer Projected Fixed Point (IPFP), that takes as input any initial solution, continuous or discrete, and quickly finds a solution obeying the initial discrete constraints of Problem 1 with a better score, most often significantly better than the initial one (P_d from step 2 is a projection on the discrete domain, discussed below):

1. Initialize $\mathbf{x}^* = \mathbf{x_0}$, $S^* = \mathbf{x_0^T M x_0}$, $k = 0$, where $x_i \geq 0$ and $\mathbf{x} \neq \mathbf{0}$
2. Let $\mathbf{b_{k+1}} = P_d(\mathbf{M x_k})$, $C = \mathbf{x_k^T M}(\mathbf{b_{k+1}} - \mathbf{x_k})$, $D = (\mathbf{b_{k+1}} - \mathbf{x_k})^\mathbf{T}\mathbf{M}(\mathbf{b_{k+1}} - \mathbf{x_k})$
3. If $D \geq 0$ set $\mathbf{x_{k+1}} = \mathbf{b_{k+1}}$. Else let $r = \min\{-C/D, 1\}$ and set $\mathbf{x_{k+1}} = \mathbf{x_k} + \mathbf{r}(\mathbf{b_{k+1}} - \mathbf{x_k})$
4. If $\mathbf{b_{k+1}^T M b_{k+1}} \geq S^*$ then set $S^* = \mathbf{b_{k+1}^T M b_{k+1}}$ and $\mathbf{x}^* = \mathbf{b_{k+1}}$
5. If $\mathbf{x_{k+1}} = \mathbf{x_k}$ stop and return the solution $\mathbf{x}^*$
6. Set $k = k + 1$ and go back to step 2.

This algorithm is loosely related to the power method for eigenvectors, also used by spectral matching [9]: at step 2 it replaces the fixed point iteration of the power method $\mathbf{v_{k+1}} = P(\mathbf{M v_k})$, where P is the projection on the unit sphere, with a similar iteration $\mathbf{b_{k+1}} = P_d(\mathbf{M x_k})$, in which P_d is the projection on the one-to-one (for graph matching) or many-to-one (for MAP inference) discrete constraints. Since all possible discrete solutions have the same norm, P_d boils down to finding the discrete vector $\mathbf{b_{k+1}} = \text{argmax } \mathbf{b^T M x_k}$, which, for many-to-one constraints can be easily found in linear time, and for one-to-one constraints the efficient Hungarian method can be used. Note that (see Proposition 1), in both cases (one-to-one or many-to-one constraints), the discrete $\mathbf{b_{k+1}}$ is also the one maximizing the dot-product with $\mathbf{M x_k}$ in the continuous domain $\mathbf{A b} = \mathbf{1}$, $\mathbf{b} > \mathbf{0}$. IPFP is also related to Iterative Conditional Modes (ICM) [10] used for inference in graphical models. In the domain of many-to-one constraints IPFP becomes an extension of ICM for which the updates are performed in parallel without losing the climbing property and the convergence to a discrete solution. Note that the fully parallel version of ICM is IPFP without step 3: $\mathbf{x_{k+1}} = P_d(\mathbf{M x_k})$. The theoretical results that we will present shortly are valid for both one-to-one and many-to-one constraints, with a few differences that we will point out when deemed necessary.

The algorithm is a basically a sequence of linear assignment (or independent labeling) problems, in which the next solution is found by using the previous one. In practice the algorithm converges in about $5 - 10$ steps, which makes it very efficient, with basically the same complexity as the complexity of step 2. Step 3 insures that the quadratic score increases with each iteration. Step 4 guarantees that the binary solution returned is never worse than the initial solution. In practice, the algorithm significantly improves the initial binary solution, and the final continuous solution is most often discrete, and always close to the best discrete one found. In fact, in the case of MAP inference, it is guaranteed that the point of convergence is discrete, as a fixed point of P_d.

Intuition The intuition behind this algorithm is the following: at every iteration the quadratic score $\mathbf{x^T M x}$ is first approximated by the first order Taylor expansion around the current solution $\mathbf{x_k}$: $\mathbf{x^T M x} \approx \mathbf{x_k^T M x_k} + 2\mathbf{x_k^T M}(\mathbf{x} - \mathbf{x_k})$. This approximation is maximized within the discrete domain of Problem 1, at step 2, where $\mathbf{b_{k+1}}$ is found. From Proposition 1 (see next) we know that the same discrete $\mathbf{b_{k+1}}$ also maximizes the linear approximation in the continuous domain of Problem 2. The role of $\mathbf{b_{k+1}}$ is to provide a direction of largest possible increase (or ascent) in the first-order approximation, within both the continuous domain and the discrete domain simultaneously. Along this direction the original quadratic score can be further maximized in the continuous domain of Problem 2 (as long as $\mathbf{b_{k+1}} \neq \mathbf{x_k}$). At step 3 we find the optimal point along this direction, also inside the continuous domain of Problem 2. The hope, also confirmed in practice, is that the algorithm will tend to converge towards discrete solutions that are, or are close to, maxima of Problem 2.

4.2 Theoretical Analysis

Proposition 1:*For any vector* $\mathbf{x} \in R^n$ *there exists a global optimum* $\mathbf{y}^*$ *of* $\mathbf{x^T My}$ *in the domain of Problem* 2 *that has binary elements (thus it is also in the domain of Problem* 1*)*

Proof: Maximizing $\mathbf{x^T My}$ with respect to $\mathbf{y}$, subject to $\mathbf{Ay} = \mathbf{1}$ and $\mathbf{y} > 0$ is linear program for which an integer optimal solution exists because the constraints matrix $\mathbf{A}$ is totally unimodular [9]. This is true for both one-to-one and many-to-one constraints.

It follows that the maximization from step 2 $\mathbf{b_{k+1}} = \text{argmax } \mathbf{b^T M x_k}$ in the original discrete domain, also maximizes the same dot-product in the continuous domain of Problem 2, of relaxed constraints $\mathbf{Ax} = \mathbf{1}$ and $\mathbf{x} > 0$. This insures that the algorithm will always move towards some discrete solution that also maximizes the linear approximation of the quadratic function in the domain of Problem 2. Most often in practice, that discrete solution also maximizes the quadratic score, along the same direction and within the continuous domain. Therefore $\mathbf{x_k}$ is likely to be discrete at every step.

Theorem 1: *The quadratic score* $\mathbf{x_k^T M x_k}$ *increases at every step* k *and the sequence of* $\mathbf{x_k}$ *converges*

Proof: For a given step k, if $\mathbf{b_{k+1}} = \mathbf{x_k}$ we have convergence. If $\mathbf{b_{k+1}} \neq \mathbf{x_k}$, let $\mathbf{x}$ be a point on the line between $\mathbf{x_k}$ and $\mathbf{b_{k+1}}$, $\mathbf{x} = \mathbf{x_k} + t(\mathbf{b_{k+1}} - \mathbf{x_k})$. For any $0 \leq t \leq 1$, $\mathbf{x}$ is in the feasible domain of Problem 2. Let $S_k = \mathbf{x_k^T M x_k}$. Let us define the quadratic function $f(t) = \mathbf{x^T M x} = S_k + 2tC + t^2 D$, which is the original function in the domain of Problem 2 on the line between $\mathbf{x_k}$ and $\mathbf{b_{k+1}}$. Since $\mathbf{b_{k+1}}$ maximizes the dot product with $\mathbf{x_k^T M}$ in the discrete (and the continuous) domain, it follows that $C \geq 0$. We have two cases: $D \geq 0$, when $\mathbf{x_{k+1}} = \mathbf{b_{k+1}}$ (step 3) and $S_{k+1} = \mathbf{x_{k+1}^T M x_{k+1}} = f_q(1) \geq S_k = \mathbf{x_k^T M x_k}$; and $D < 0$, when the quadratic function $f_q(t)$ is convex with the maximum in the domain of Problem 2 attained at point $\mathbf{x_{k+1}} = \mathbf{x_k} + \mathbf{r}(\mathbf{b_{k+1}} - \mathbf{x_k})$. Again, it also follows that $S_{k+1} = \mathbf{x_{k+1}^T M x_{k+1}} = f_q(r) \geq S_k = \mathbf{x_k^T M x_k}$. Therefore, the algorithm is guaranteed to increase the score at every step. Since the score function is bounded above on the feasible domain, it has to converge, which happens when $C = 0$.

By always improving the quadratic score in the continuous domain, at each step the next solution moves towards discrete solutions that are better suited for solving the original Problem 1.

Theorem 2: *The algorithm converges to a maximum of Problem* 2

Proof: Let $\mathbf{x}^*$ be a point of convergence. At that point the gradient $2\mathbf{M x}^*$ is non-zero since both $\mathbf{M}$ and $\mathbf{x}^*$ have positive elements and $(\mathbf{x}^*)^\mathbf{T}\mathbf{M x}^* > \mathbf{0}$, (it is higher than the score at the first iteration, also greater than zero). Since $\mathbf{x}^*$ is a point of convergence it follows that $C = 0$, that is, for any other $\mathbf{x}$ in the continuous domain of Problem 2, $(\mathbf{x}^*)^\mathbf{T}\mathbf{M x}^* \geq (\mathbf{x}^*)^\mathbf{T}\mathbf{M x}$. This implies that for any direction vector $\mathbf{v}$ such that $\mathbf{x}^* + t\mathbf{v}$ is in the domain of Problem 2 for a small enough $t > 0$, the dot-product between $\mathbf{v}$ and the gradient of the the quadratic score is less than or equal to zero $(\mathbf{x}^*)^\mathbf{T}\mathbf{M v} \leq 0$, which further implies that $\mathbf{x}^*$ is a maximum (local or global) of the quadratic score within the continuous domain of equality constraints $\mathbf{A x}^* = \mathbf{1}, \quad \mathbf{x}^* > \mathbf{0}$.

For many-to-one constraints (MAP inference) it basically follows that the algorithm will converge to a discrete solution, since maxima of Problem 2 are in the discrete domain [11, 12, 13]. This is another similarity with ICM, which also converges to a maximum. Therefore, combining ours with ICM cannot improve the performance of ICM, and vice-versa.

Theorem 3: *If* $\mathbf{M}$ *is positive semidefinite with positive elements, then the algorithm converges in a finite number of iterations to a discrete solution, which is a maximum of Problem* 2

Proof: Since $\mathbf{M}$ is positive semidefinite we always have $D \geq 0$, thus $\mathbf{x_k}$ is always discrete for any k. Since the number of discrete solutions is finite, the algorithm must converge in a finite number of steps to a local (or global) maximum, which must be discrete. This results is obviously true for both one-to-one and many-to-one constraints.

When $\mathbf{M}$ is positive semidefinite, Problem 2 is a concave minimization problem for which it is well known that the global optimum has integer elements, so it is also a global optimum of the original Problem 1. In this case our algorithm is only guaranteed to find a local optimum in a finite number of iterations. Global optimality of concave minimization problems is a notoriously difficult task since the problem can have an exponential number of local optima. In fact, if a large enough constant is added to the diagonal elements of $\mathbf{M}$, every point in the original domain of possible solutions becomes a local optimum for one-to-one problems. Therefore adding a large constant to make the problem concave is not a good idea, even if the global optimum does not change. In practice $\mathbf{M}$ is rarely positive semidefinite, but it can be close to being one if the first eigenvalue is much larger than the others, which is the assumption made by our spectral matching algorithm, for example.

Theorem 4: *If* $\mathbf{M}$ *has non-negative elements and is rank-1, then the algorithm will converge and return the global optimum of the original problem after the first iteration*

Proof: Let $\mathbf{v}, \lambda$ be the leading eigenpair of $\mathbf{M}$. Then, since $\mathbf{M}$ has non-negative elements both $\mathbf{v}$, and λ are positive. Since $\mathbf{M}$ is also rank one, we have $\mathbf{M}\mathbf{x}_0 = \lambda(\mathbf{v}^T\mathbf{x}_0)\mathbf{v}$. Since both $\mathbf{x}_0$ and $\mathbf{v}$ have positive elements it immediately follows that $\mathbf{x}_1$ after the first iteration is the indicator solution vector that maximizes the dot-product with the leading eigenvector ($\mathbf{v}^T\mathbf{x}_0 = 0$ is a very unlikely case that never happens in practice). It is clear that this vector is the global optimum, since in the rank-1 case we have: $\mathbf{x}^T\mathbf{M}\mathbf{x} = \lambda_1(\mathbf{v}^T\mathbf{x})^2$, for any $\mathbf{x}$.

The assumption that $\mathbf{M}$ is close to being rank-1 is used by two recent algorithms, [233] and [117]. Spectral matching [117] also returns the optimal solution in this case and it assumes that the rank-1 assumption is the ideal matrix to which a small amount of noise is added. Probabilistic graph matching [233] makes the rank-1 approximation by assuming that each second-order element of $M_{ia;jb}$ is the product of the probability of feature i being matched to a and feature j being matched to b, independently. However, instead of maximizing the quadratic score function, they use this probabilistic interpretation of the pair-wise terms and find the solution by looking for the closest rank-1 matrix to $\mathbf{M}$ in terms of the KL-divergence. If the assumptions in [233] were perfectly met, then spectral matching, probabilistic graph matching and our algorithm would all return the same solution. For a comparison of all these algorithms on real world experiments please see the experiments section.

4.3 Extension of IPFP to Hyper-graph Matching

IPFP can be immediately extended to hyper-graph matching using third-order scores. We expect that its extension would outperform the algorithm of [53] in the same way IPFP outperforms the spectral matching algorithm for second-order scores. Here we outline the IPFP algorithm for hyper-graph matching with third order scores, as an immediate extension of

IPFP for graph matching:

1. Initialize $\mathbf{x}^* = \mathbf{x_0}, \quad S^* = \mathbf{H} \otimes_\mathbf{1} \mathbf{x_0} \otimes_\mathbf{2} \mathbf{x_0} \otimes_\mathbf{3} \mathbf{x_0}, \quad k = 0, \qquad \text{where } x_i \geq 0 \quad \text{and } \mathbf{x} \neq \mathbf{0}$
2. Let $\mathbf{b_{k+1}} = P_d(\mathbf{H} \otimes_\mathbf{1} \mathbf{x_k} \otimes_\mathbf{2} \mathbf{x_k})$. Let $S_b = \mathbf{H} \otimes_\mathbf{1} \mathbf{b_{k+1}} \otimes_\mathbf{2} \mathbf{b_{k+1}} \otimes_\mathbf{3} \mathbf{b_{k+1}}$
3. Let $\mathbf{v}(t) = \mathbf{x_k} + t(\mathbf{b_{k+1}} - \mathbf{x_k})$. Compute in closed-form $t^* \in [0, 1]$ that maximizes $S(t) = \mathbf{H} \otimes_\mathbf{1} \mathbf{v(t)} \otimes_\mathbf{2} \mathbf{v(t)} \otimes_\mathbf{3} \mathbf{v(t)}$.
4. Set $\mathbf{x_{k+1}} = \mathbf{v}(\mathbf{t}^*)$.
5. If $S_b \geq S^*$ then set $S^* = S_b$ and $\mathbf{x}^* = \mathbf{b_{k+1}}$
6. If $\mathbf{x_{k+1}} = \mathbf{x_k}$ stop and return the solution $\mathbf{x}^*$
7. Set $k = k + 1$ and go back to step 2.

Note that in the case of third-order scores t^* can be still computed in one step (closed-form) since $S(t) = \mathbf{H} \otimes_\mathbf{1} \mathbf{v(t)} \otimes_\mathbf{2} \mathbf{v(t)} \otimes_\mathbf{3} \mathbf{v(t)}$ is a one-parameter cubic function. This would make IPFP for third-order interactions very efficient. In the case of higher-order scores this would not work, so IPFP would loose some of its efficiency. However, in practice the use of scores of order higher than three would probably not be necessary, since third-order interactions could be designed to accommodate affine and even homographic/perspective transformations.

4.4 Experiments

We first present some representative experiments on graph matching problems. We tested IPFP by itself, as well as in conjunction with other algorithms as a post-processing step. We followed the experiments from Chapter 5 in the case of outliers: we used the same cars and motorbikes image pairs, extracted from the Pascal 2007 database, and face image pairs from Caltech-4, the same features (oriented points extracted from contours) and the same second-order potentials $M_{ia;jb} = e^{-\mathbf{w}^T \mathbf{g}_{ia;jb}}$; $\mathbf{g}_{ia;jb}$ is a five dimensional vector of deformations in pairwise distances and angles when matching features (i, j) from one image to features (a, b) from the other and $\mathbf{w}$ is the set of parameters that control the weighing of the elements of $\mathbf{g}_{ia;jb}$. We followed the setup from Chapter 5 exactly. These experiments are difficult due the large number of outliers (on average 5 times more outliers than inliers), and, in the case of cars and motorbikes, also due to the large intra-category variations in shape present in the Pascal 2007 database. By outliers we mean the features that have no ground truth correspondences in the other image, and by inliers those that have such correspondences. As in Chapter 5 we allow outliers only in one of the images in which they are present in

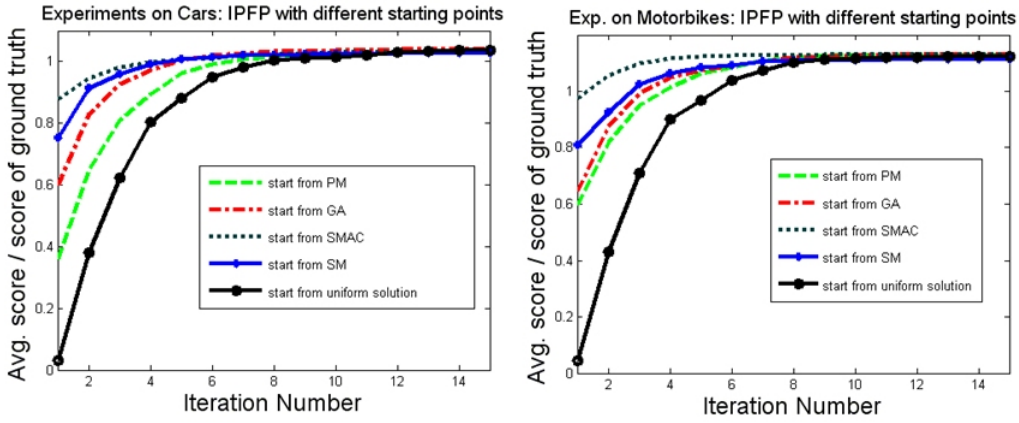

Figure 4.1: Results on motorbikes and cars averaged over 30 experiments: at each iteration the average score $\mathbf{x_k^T M x_k}$ normalized by the ground truth score is displayed. Notice how all algorithms converge to similar scores when IPFP is added, even when the starting scores are significantly different, which suggests that IPFP is robust to starting points and levels the performances of other algorithms. Even by itself IPFP achieves a similar performance. Also notice how quickly IPFP converges (fewer than 10 iterations).

large number, the ratio of outliers to inliers varying from 1.5 to over 10. The ground truth correspondences the same ones from Chapter 5, which were manually selected.

The difficulty of the matching problems is reflected by the relatively low matching scores of all algorithms, especially on the cars and motorbikes image sets (Table 4.2). In order to insure an optimal performance of all algorithms, we used the supervised version of the graph matching learning method presented in Chapter 5. Learning $\mathbf{w}$ was effective, improving the performance by more than 15% on average, for all algorithms. The algorithms we chose for comparison and also for combining with ours are among the current state-of-the-art in the literature: spectral matching with affine constraints (SMAC) [43], spectral matching (SM) [117], probabilistic graph matching (PM) [233], and graduated assignment (GA) [76]. In Tables 4.2 and 4.3 we show that in our experiments IPFP significantly outperforms other state-of-the-art algorithms.

In our experiments we focused on two aspects. First, we tested the matching rate of our algorithm against the others, and observed that it consistently outperforms them, both in the matching rate and in the final quadratic score achieved by the resulting discrete solution (see Tables 4.2, 4.3). Second, we combined our algorithm, as a post-processing step, with the others and obtained a significant improvement over the output matching rate and quadratic score of the other algorithms by themselves (see Figures 4.2, 4.3). In Figure 4.2 we show the quadratic score of our algorithm, per iteration, for several individual experiments, when it takes as initial solution the output of several other algorithms. The score at the

Table 4.1: Comparison of matching rates at testing time between our algorithm and 4 state-of-the-art graph matching algorithms following the same experimental setup with outliers as in [6]: the pairs of images are from the cars and motorbikes datasets of the Pascal07 challenge. All outlier features are allowed in the right image, no outliers in the left image. The parameters are learned using the supervised version of the algorithm from [6]. Results are averages over 30 different experiments. The same learned parameters were used by all algorithms. In the case of no learning the default parameters used by all algorithms are $\mathbf{w} = [0.2, 0.2, 0.2, 0.2, 0.2]$

Algorithm	IPFP	SM	SMAC	GA	PM
Cars: No Learning	**50.9%**	26.3%	39.1%	31.9%	20.9%
Cars: With Learning	**73.1%**	61.6%	64.8%	46.6%	33.6%
Motorbikes: No Learning	32.9%	29.7%	**39.2%**	34.2%	26.1%
Motorbikes: With Learning	**55.7%**	54.8%	52.4%	46.8%	42.2%

Table 4.2: Matching rates for the experiments with outliers on cars and motorbikes from Pascal 07, and faces from Caltech-4. Note that our algorithm by itself outperforms all the others by themselves, in most experiments. Moreover, when the binary solution of other algorithms becomes the input to our algorithm the performance is greatly improved.

Dataset	IPFP	SM	SMAC	GA	PM
Cars: alone	**73.1%**	61.6%	64.8%	46.6%	33.6%
Cars: + IPFP	**73.1%**	73.0%	72.9%	72.1%	**73.1%**
Cars: Improvement	0	+11.4%	+8.1%	+25.5%	+39.5%
Motorbikes: alone	**55.7%**	54.8%	52.4%	46.8%	42.2%
Motorbikes: + IPFP	55.7%	61.0%	59.6%	60.6%	**61.2%**
Motorbikes: Improvement	0	+6.2%	+7.2%	+13.8%	+19.0%
Faces: alone	97.1%	93.0%	**98.0%**	81.2%	65.6%
Faces: + IPFP	97.1%	98.0%	**98.5%**	97.7%	98.0%
Faces: Improvement	0	+5.0%	+0.5%	+16.5%	+32.4%

first iteration is the score of the final discrete solution returned by those algorithms and the improvement in just a few iterations is substantial, sometimes more than doubling the final quadratic score reached by the other algorithms. In Figure 4.1 we show the average scores of our algorithm, over 30 different experiments on cars and motorbikes, per iteration, normalized by the score of the solutions given by the human ground truth labeling. We notice that regardless of the starting condition, the final scores are very similar, slightly above the value of 1 (Table 4.3), which means that the solutions reached are, on average, at least as good, in terms of the matching score function, as the manually picked solutions. None of the algorithms by themselves, except only for IPFP, reach this level of quality. We also notice that a quadratic score of 1 does not correspond to a perfect matching rate, which indicates the fact that besides the ground truth solution there are other solutions with high score. This is expected, given that the large number of outliers can easily introduce wrong solutions of high score. However, increasing the quadratic score, does increase the matching rate as can be seen by comparing the results between the Tables 4.3 and 4.2.

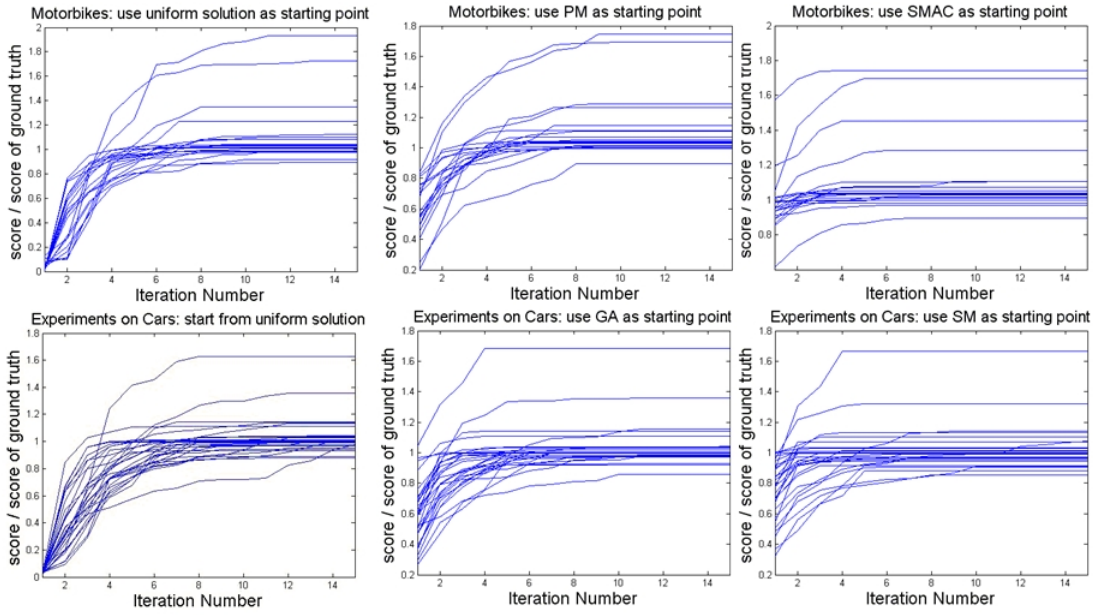

Figure 4.2: Experiments on cars and motorbikes: at each iteration the score $\mathbf{x_k^T M x_k}$ normalized by the ground truth score is displayed for 30 individual matching experiments for our algorithm starting from different solutions (uniform, or given by some other algorithm).

Experiments on MAP inference problems We believe that our algorithm can have a greater impact in graph matching problems than in MAP inference ones, due to the lack of efficient, high-quality discretization procedures in the graph matching literature. In the

Table 4.3: Quadratic scores on the Cars, Motorbikes and Faces image sets. The score of the best binary solution returned by our algorithm alone is much better than the score reached by the other algorithms. When combined with ours, all the algorithms gain a significant improvement in the score, and the final scores reached are very similar on average. The normalized score S_{max}/S^* is the score of the binary solution returned by the algorithm divided by the score of the manually picked ground truth. The "Convergence to a binary solution" row shows the average rate at which our algorithm converges to a discrete solution.

Experiments on Cars	IPFP	SM	SMAC	GA	PM
Alone, avg S_{max}/S^*	**1.038**	0.753	0.879	0.601	0.362
+ IPFP, avg S_{max}/S^*	1.038	1.027	1.033	**1.041**	1.029
Improvement $(S_{max} - S_0)/S^*$	0	+27.4%	+15.4%	+44.0%	+66.7%
Convergence to a binary solution	86.7%	93.3%	86.7%	93.3%	86.7%

Experiments on Motorbikes	IPFP	SM	SMAC	GA	PM
Alone, avg S_{max}/S^*	**1.123**	0.809	0.975	0.645	0.595
+ IPFP, avg S_{max}/S^*	1.123	1.112	**1.131**	1.130	1.130
Improvement $(S_{max} - S_0)/S^*$	0	+30.3%	+15.6%	+32.1%	+53.5%
Convergence to a binary solution	100.0%	100.0%	100.0%	85.0%	80.0%

Experiments on Faces	IPFP	SM	SMAC	GA	PM
Alone, avg S_{max}/S^*	**1.002**	0.932	0.997	0.830	0.618
+ IPFP, avg S_{max}/S^*	1.002	1.004	**1.005**	1.004	0.997
Improvement $(S_{max} - S_0)/S^*$	0	+7.2%	+0.8%	+17.4%	+ 37.9%
Convergence to a binary solution	100.0%	100.0%	96.7%	96.7%	100.0%

Table 4.4: Average scores over 30 different experiments on 4-connected and 8-connected planar graphs with 50 sites and 10 possible labels per site. As in the case of randomly generated graphs, BP by itself has the worst performance, and when combined with IPFP, the best.

Graph type	IPFP	ICM	BP	BP+IPFP	L2QP	L2QP+IPFP
4-connected planar	79.5	78.2	54.2	**82.3**	56.6	80.1
8-connected planar	126.0	123.2	75.4	**126.7**	88.1	126.5

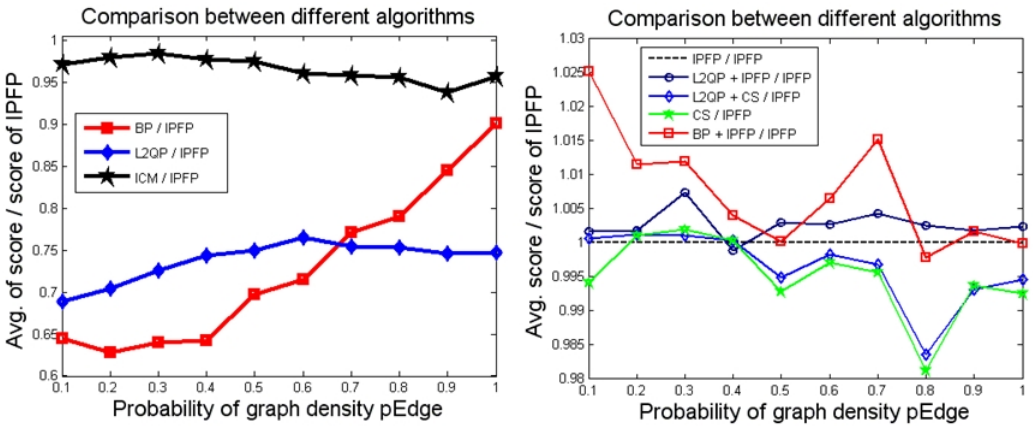

Figure 4.3: Average quadratic scores normalized by the score of IPFP, over 30 different experiments, for each probability of edge generation $pEdge \in 0.1, 0.2, ..., 1$. Note that IPFP consistently ourperforms BP (by a wide margin) and ICM. Right: CS is the climbing/discretization procedure proposed in Chapter 3, also used in [41].

domain of MAP inference for MRFs, it is important to note that IPFP is strongly related to the parallel version of Iterated Conditional Modes, but, unlike parallel ICM, it has climbing, strong convergence and local optimality properties. To see the applicability of our method to MAP inference, we tested it against the popular Max-Product BP and sequential ICM, and our algorithm L2QP [118] from Chapter 3, which together with [41] formulate the MAP inference problem as an integer quadratic program with very similar theoretical properties and performance in practice. We also measured the improvement after applying IPFP as a post processing step, to the output solution of BP and L2QP. Since ICM is guaranteed to reach a local maximum, IPFP cannot further improve its performance. For the MAP inference problems we used the same experimental setup as in Chapter 3 (same one used in [41] and [118]), on graphs with different degrees of edge density (by generating random edges with a given probability, varying from 0.1 to 1). The values of the potentials were randomly generated as in [41] and [118], favoring the correct labels vs. the wrong ones. In Figure 4.3 we show the average scores normalized by the score of IPFP over 30 different experiments, for different probabilities of edge generation *pEdge* on graphs with 50 nodes and 10 possible labels. IPFP consistently outperforms its sequential relative ICM and the popular Max-Product BP, while taking less computational time than both. Also, as in the case of graph matching, using IPFP for further post-processing of the output for methods without local optimality guarantees in the original domain, such as BP and L2QP, can greatly improve the performance.

We also compared IPFP to the algorithm of [167] which is based on a convex ap-

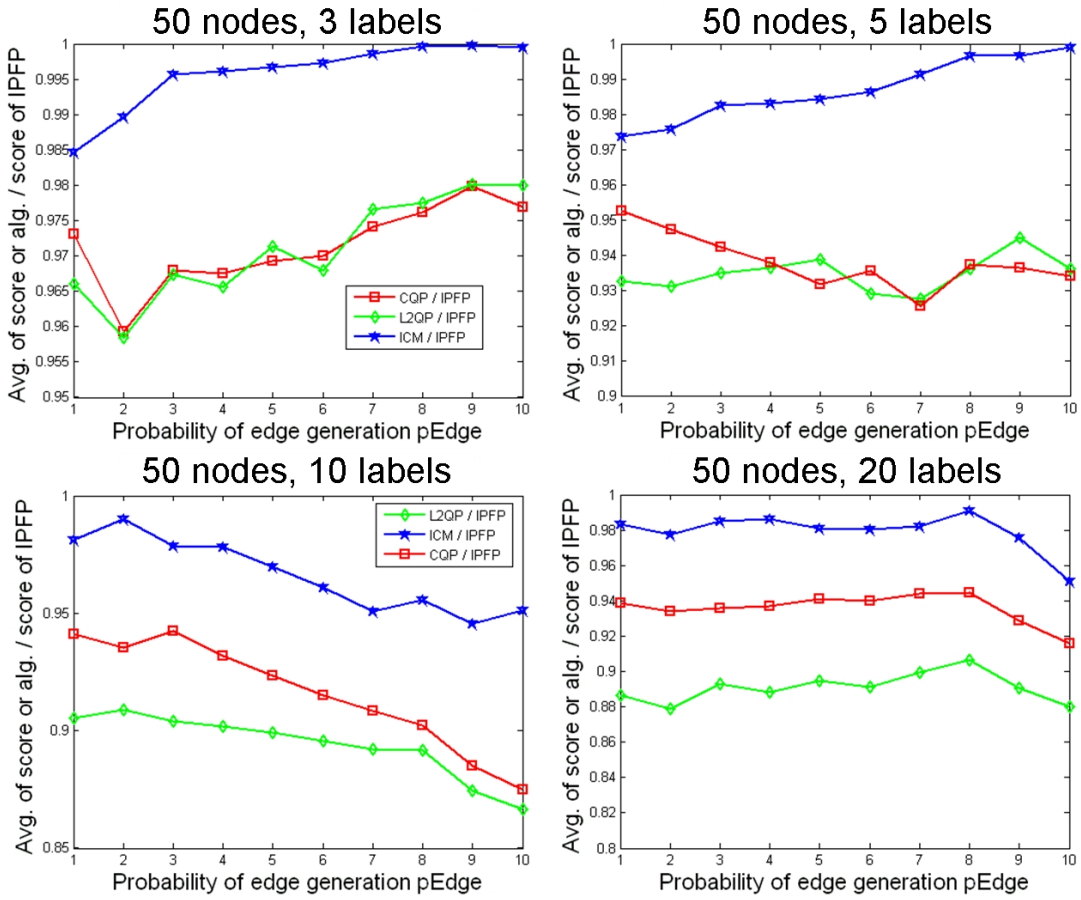

Figure 4.4: Average quadratic scores normalized by the score of IPFP, over 30 different experiments, for each probability of edge generation $pEdge \in 0.1, 0.2, ..., 1$ and different number of labels, for graphs with 50 nodes. Note that IPFP consistently ourperforms L2QP [118] and CQP [167] (by a wide margin) and ICM. Note that L2QP and CQP perform similarly for a small number of labels.

proximation. The matrix $\mathbf{M}$ is transformed into a negative-definite matrix by subtracting appropriate large values from its diagonal elements, then the modified problem is solved exactly. The solution obtained is then quickly discretized such that the mapping many-to-one constraints are imposed. Note that the authors of [167] use ICM to obtain a binary solution. However, we wanted to emphasize the quality of the methods by themselves, without a powerful discretization step, and used ICM for comparisons separately. For discretization we used one iteration of ICM for both our L2QP [118] and CQP [167] algorithms. Both ICM and IPFP used as initial condition a uniform solution as before. We tested all four algorithms on the same type of problems as before, varying the number of labels and the

degree of edge density and averaging the results over 30 different experiments (see Figure 4.4). We note that for a small number of labels L2QP and CQP perform similarly, while CQP slightly outperforming L2QP as the number of labels increases. The most important observation however is that both ICM and IPFP outperform L2QP and CQP by a wide margin on all problems without any single exception. In our experiments, on every single problem, IPFP outperformed ICM, while both IPFP and ICM outperformed both L2QP and CQP by a wide margin, which is also reflected in the averages shown in Figure 4.4. Between CQP and L2QP there was no clear winner since for a small number of labels and high density of edges L2QP seemed to perform better than CQP.

4.5 Conclusion

This chapter presents a novel and computationally efficient algorithm, Integer Projected Fixed Point (IPFP), that outperforms state-of-the-art methods for solving quadratic assignment problems in graph matching, and well-established methods in MAP inference such as BP and ICM. We analyze the theoretical properties of IPFP and show that it has strong convergence and climbing guarantees. Also, IPFP can be employed in conjunction with existing techniques, such as SMAC or SM for graph matching or BP for inference to achieve solutions that are dramatically better than the ones produced independently by those methods alone. Furthermore, IPFP is very straightforward to implement and converges in only 5–10 iterations in practice. Thus, IPFP is very well suited for addressing a broad range of real-world problems in computer vision and machine learning.

Chapter 5

Learning with Spectral Matching

In Chapter 2 we presented an algorithm for finding correspondences between two sets of features mainly based on the second-order relationships between them. The reason why this algorithm works efficiently is because the pairwise geometric scores favor correct assignments much more than incorrect ones. As we discussed earlier accidental assignments are rare, so strong pairwise scores between incorrect assignments are unlikely, while such strong scores between most correct ones are very likely. Of course, this is a qualitative, intuitive explanation that assumes that the scores are well designed and meaningful. Therefore, an important issue is how to learn the pairwise scores that will make the algorithm work at its optimal level in the task of matching a specific object type or shape. How can we find automatically the correct scores during training, without hand tuning them and even without knowing the ground truth correspondences for the training pairs of feature sets?

One would ideally like to keep the pairwise scores between correct assignments high while lowering as much as possible the pairwise scores between incorrect ones. But how can we quantify this goal, and more importantly, how can we learn these scores automatically? This is the issue that we will discuss in this chapter, showing that it is possible, without loss of quality, to learn the pairwise scores in an unsupervised way, that is, without knowing the correct assignments during training. We will demonstrate experimentally that we can learn meaningful pairwise scores even in the presence of outliers or in cases when the training set is corrupted with pairs of features for which there is no such set of correct assignments (for example, when trying to find correspondences between a motorbike and a car).

While there are many papers on solving the graph matching problem efficiently [15], [117], [43], [76], [180], [210], [233] there are only two papers published so far, to the best of our knowledge, that propose a solution for learning the optimal set of parameters for graph matching, in the context of computer vision applications: [30] and our previous work [119], also presented in the Appendix A. As shown by [30], and also by us in [119] and in this thesis, learning the parameters is important for improving the matching performance.

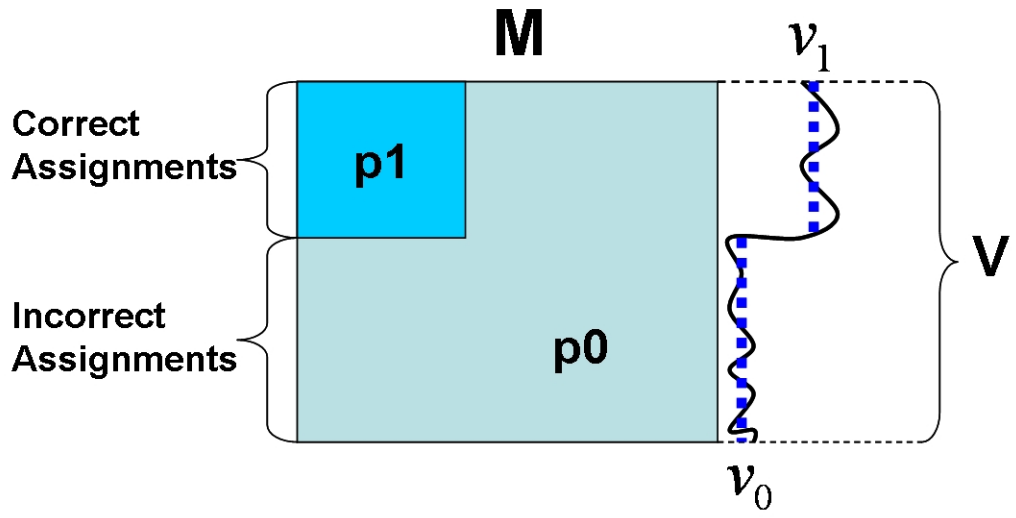

Figure 5.1: Pairwise scores (elements of the matching matrix **M**) between correct assignments have a higher expected value p_1 than elements with at least one wrong assignment, with expected value p_0. This will be reflected in the eigenvector **v** that will have higher expected value v_1 for correct assignments than v_0 for wrong ones.

The main differences between the learning method we propose here and the one described in Appendix A are the following: the algorithm in the Appendix is based on a global optimization scheme that aims at maximizing directly the number of correct matches over the training set, whereas the one presented in this Chapter improves the matching performance only indirectly, by making the leading eigenvector of the match matrix **M** more binary using a local, gradient-based optimization procedure. However, the advantage of the new method is that it is completely unsupervised, as opposed to the supervised one from the Appendix, it is much faster, while producing very similar results in practice.

In this Chapter, we show for the first time how to efficiently perform unsupervised learning for graph matching (see Section 5.2.2, [120]). Unsupervised learning for matching is important in practice, since manual labeling of correspondences can be quite time consuming. The same basic algorithm can be used in the supervised or semi-supervised cases with minimal modification, if all or some of the ground truth matches are available (Section 5.2.1). We also show empirically that our learning algorithm is robust to the presence of outliers (Sections 5.3.1, 5.3.2, 5.4). This method is inspired from the properties of spectral matching [117], but it can be successfully used for improving the performance of other state-of-the-art matching algorithms (Section 5.4).

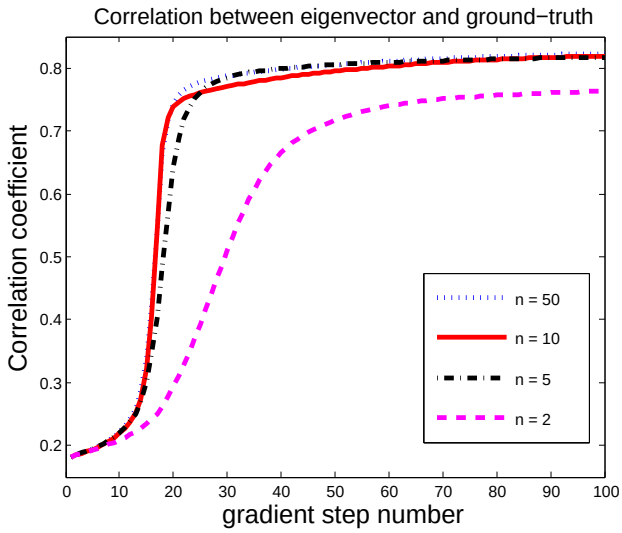

Figure 5.2: Experiments on the House sequence. The plots show the normalized correlation between the eigenvector and the ground truth solution for different numbers of recursive iterations n used to compute the approximative derivative of the eigenvector (averages over 70 experiments). Even for n as small as 5 the learning method converges in the same way, returning the same result.

Learning for Graph Matching The graph matching problem, as also presented in the previous chapters, consists of finding the indicator vector $\mathbf{x}^*$ that maximizes a quadratic score function:

$$\mathbf{x}^* = argmax(\mathbf{x}^\mathbf{T}\mathbf{M}\mathbf{x}). \tag{5.1}$$

Here $\mathbf{x}$ is an indicator vector such that $x_{ia} = 1$ if feature i from one image (or object model) is matched to feature a from the other image (or object model) and zero otherwise. Usually, one-to-one constraints are imposed on $\mathbf{x}$ such that one feature from one image can be matched to at most one other feature from the other image. In spectral matching $\mathbf{M}$ is a matrix with positive elements containing the pairwise score functions, such that $M_{ia;jb}$ measures how well the pair of features (i, j) from one image agrees in terms of geometry and appearance (e.g. difference in local appearance descriptors, pairwise distances, angles, etc) with a pair of candidate matches (a, b) from the other. The local appearance terms of candidate correspondences can be stored on the diagonal of $\mathbf{M}$; in practice we noticed that including them in the pairwise scores $M_{ia;jb}$, and leaving zeros on the diagonal gives better results; $M_{ia;jb}$ is basically a function that is defined by a certain parameter vector $\mathbf{w}$. The type of pairwise scores $M_{ia;jb}$ that we use in our experiments is:

$$M_{ia;jb} = \exp\left(-\mathbf{w}^{\mathbf{T}}\mathbf{d}_{\mathbf{ia;jb}}\right), \tag{5.2}$$

where $\mathbf{w}$ is a vector of weights/parameters (to be learned) and $\mathbf{d}_{\mathbf{ia;jb}}$ is a vector (usually containing non-negative errors/deformations) describing the changes in geometry and appearance when matching the pair of features (i, j) to the pair of features (a, b).

Then, learning for graph matching consists of finding $\mathbf{w}$ that maximizes the performance (w.r.t to the ground truth correspondences) of matching (as defined by Equation 5.1) over pairs of training images.

It is important to mention here the issue of using thresholds for the pairwise scores. In practice one would want to use thresholds in order to speed up the building of the matrix $\mathbf{M}$. Using thresholds on the individual elements of $\mathbf{d}_{\mathbf{ia;jb}}$ can achieve this speed up if these elements are computed and checked in the increasing order of their computational complexity. If an element of $\mathbf{d}$ does not pass a certain threshold then the pairwise score is immediately set to zero and there is no need of further computation. Also, during training it is important to impose thresholds on the individual elements of $\mathbf{d}$ and not on the actual pairwise scores, because this approach is independent of the parameter vector $\mathbf{w}$ that we want to learn. Loose thresholds are enough both for a significant speedup as well as for learning the correct parameters. If thresholds are too tight, the wrong parameters might be learned since it is hard to determine a priori what is the tightest threshold that does not annihilate the pairwise scores between correct assignments. In our experiments on learning, however, we did not use any thresholds, in order to emphasize the strength of our learning method. Thresholds encourage the correct assignments and help moving the parameters in the right direction and if used correctly could improve both matching and learning in practice. We did use thresholds only on our recognition experiments presented in chapter 7 and we learned them independently such that for each element in $\mathbf{d}$ 95% of the pairwise scores between correct assignments survive the check while most of the incorrect ones do not.

Our matching algorithm [117] interprets each element of the principal eigenvector $\mathbf{v}$ of $\mathbf{M}$ as the confidence that the corresponding assignment is correct. It starts by choosing the element of maximum confidence as correct, then it removes (zeroes out in $\mathbf{v}$) all the assignments in conflict (w.r.t the one-to-one mapping constraints) with the assignment chosen as correct, then it repeats this procedure until all assignments are labeled either correct or incorrect.

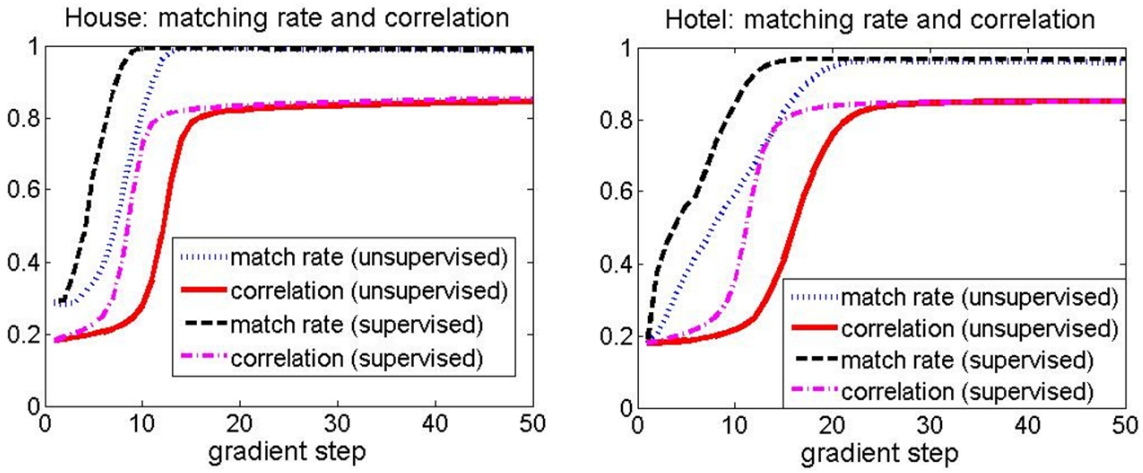

Figure 5.3: Supervised vs. unsupervised learning: Average match rate and correlation between the eigenvector and the ground truth over all training image pairs, over 70 different experiments (10 randomly chosen training images from the House (Hotel, respectively) sequence). The standard deviations are not significant.

5.1 Theoretical Analysis

Our proposed algorithm is motivated by the statistical properties of the matrix $\mathbf{M}$ and of its leading eigenvector $\mathbf{v}$, which is the continuous solution given by our spectral graph matching algorithm. In order to analyze the properties of $\mathbf{M}$ theoretically, we need a few assumptions and approximations, which we validate experimentally. Each instance of the matching problem is unique so nothing can be said with absolute certainty about $\mathbf{M}$ and its eigenvector $\mathbf{v}$, nor the quality of the solution returned. Therefore, we must be concerned with the average (or expected) properties of $\mathbf{M}$ rather than the infinitely many particular cases. We propose a model for $\mathbf{M}$ (Figure 5.1) that we validate through experiments.

Let $p_1 > 0$ be the expected value (average value over infinitely many matching experiments of the same type) of the second-order scores between correct assignments $E(M_{ia;jb})$ for any pair (ia, jb) of correct assignments. Similarly, let $p_0 = E(M_{ia;jb}) \geq 0$ if at least one of the assignments ia and jb is wrong. The assumption here is that the expected values of the second order scores do not depend on the particular assignments ia or jb, but only on whether these assignments are correct or not. p_1 should be higher than p_0, since the pairs of correct assignments are expected to agree both in appearance and geometry and have strong second-order scores, while the wrong assignments have such high pairwise scores only accidentally. We expect that the higher p_1 and the lower p_0, the higher the matching

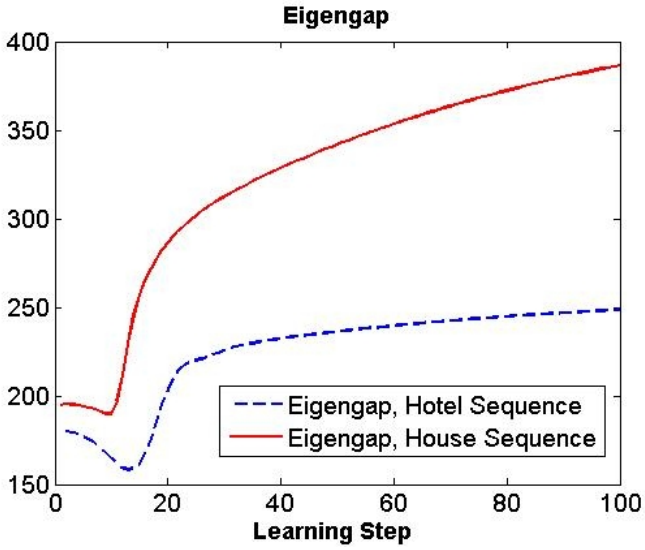

Figure 5.4: During unsupervised learning, the normalized eigengap (eigengap divided by the mean value in $\mathbf{M}$) starts increasing after a few iterations, indicating that the leading eigenvector becomes more and more stable. Results are on the House and Hotel datasets averaged over 70 random experiments.

rate. We also expect that this performance depends on their ratio $p_r = p_0/p_1$ and not on their absolute values, since multiplying $\mathbf{M}$ by a constant does not change the leading eigenvector. Since the model assumes the same expected value p_1 for all pairwise scores between correct assignments (and p_0 for all pairwise scores including a wrong assignment), and since the norm of the eigenvector does not matter, we can also assume that all correct assignments ia will have the same mean eigenvector confidence value $v_1 = E(v_{ia})$, and all wrong assignments jb will have the same $v_0 = E(v_{jb})$. The spectral matching algorithm assumes that the correct assignments will correspond to large elements of the eigenvector $\mathbf{v}$ and the wrong assignments to low values in $\mathbf{v}$, so the higher v_1 and the lower v_0 the better the matching rate. As in the case of p_r, if we could minimize during learning the average ratio $v_r = v_0/v_1$ (since the norm of the eigenvector is irrelevant) over all image pairs in a training sequence then we would expect to optimize the overall training matching rate. This model assumes fully connected graphs, but it can be verified that the results we obtain next are also valid for weakly connected graphs, as also shown in our experiments.

It is useful to investigate the relationship between v_r and p_r for a given image pair. We know that $\lambda \mathbf{v}_{ia} = \sum_{jb} M_{ia;jb} v_{jb}$. Next we assume that for each of the n features in the left image there are k candidate correspondences in the right image. We also approximate $E(\sum_{jb} M_{ia;jb} v_{jb}) \approx \sum_{jb} E(M_{ia;jb}) E(v_{jb})$, by considering that any v_{jb} is *almost* independent

of any particular $M_{ia;jb}$, since $\mathbf{M}$ is large. The approximation is actually a $\geq$ inequality, since the correlation is expected to be positive (but very small). It follows that for a correct correspondence ia, $\lambda E(v_{ia}) = \lambda v_1 \approx np_1v_1 + n(k-1)p_0v_0$. Similarly, if ia is a wrong correspondence then $\lambda E(v_{ia}) = \lambda v_0 \approx np_0v_1 + n(k-1)p_0v_0$. Dividing both equations by p_1v_1 and taking the ratio of the two we obtain:

$$v_r \approx \frac{p_r + (k-1)p_rv_r}{1 + (k-1)p_rv_r}. \tag{5.3}$$

Solving for v_r we get:

$$v_r \approx \frac{(k-1)p_r - 1 + \sqrt{1 - (k-1)p_r + 4(k-1)p_r^2}}{2(k-1)p_r}. \tag{5.4}$$

It can be verified that by this equation v_r is a monotonically increasing function of p_r. This is in fact not surprising since we expect that the smaller $p_r = p_0/p_1$, the smaller $v_r = v_0/v_1$ and the more binary the eigenvector $\mathbf{v}$ would be (and closer to the binary ground truth $\mathbf{t}$), with the elements of the wrong assignments approaching 0. This approximation turns out to be very accurate in practice, as shown by our experiments in Figures 5.6, 5.7 and 5.9. Also, the smaller v_r, the higher the expected matching rate. One way to minimize v_r is to maximize the correlation between $\mathbf{v}$ and the ground truth indicator vector $\mathbf{t}$. However, in this Chapter we want to minimize v_r in an unsupervised fashion, that is without knowing $\mathbf{t}$ during training. Our proposed solution is to maximize instead the correlation between $\mathbf{v}$ and its binary version (that is, the binary solution returned by the matching algorithm). How do we know that this procedure will ultimately give a binary version of $\mathbf{v}$ that is close to the real ground truth ? We will investigate this question next.

Let $\mathbf{b}(\mathbf{v})$ be the binary solution obtained from $\mathbf{v}$, respecting the one-to-one mapping constraints, as returned by spectral matching for a given pair of images. Let us assume for now that we know how to maximize the correlation $\mathbf{v^T b}(\mathbf{v})$. We expect that this will lead to minimizing the ratio $v_r^* = E(v_{ia}|b_{ia}(\mathbf{v}) = 0)/E(v_{ia}|b_{ia}(\mathbf{v}) = 1)$. If we let n_m be the number of misclassified assignments, n the number of true correct assignments (same as the number of features, equal in both images) and k the number of candidate assignments for each feature, we can obtain the next two equations: $E(v_{ia}|b_{ia}(\mathbf{v}) = 0) = \frac{n_mv_1+(n(k-1)-n_m)v_0}{n(k-1)}$ and $E(v_{ia}|b_{ia}(\mathbf{v}) = 1) = \frac{n_mv_0+(n-n_m)v_1}{n}$. Dividing both by v_1 and taking the ratio of the two we finally obtain:

$$v_r^* = \frac{m/(k-1) + (1 - m/(k-1))v_r}{1 - m + mv_r}, \tag{5.5}$$

where m is the matching error rate $m = n_m/n$. If we reasonably assume that $v_r < 1$ (eigenvector values slightly higher on average for correct assignments than for wrong ones) and $m < (k-1)/k$ (error rate slightly lower than random) this function of m and v_r has both

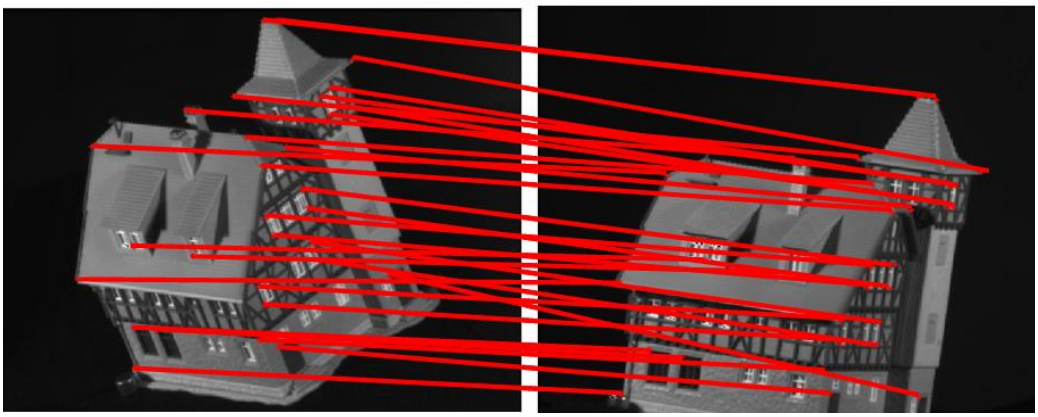

Figure 5.5: After learning the pairwise scores for matching, all the features in the first image are correctly matched to the features from the last image (House sequence).

partial derivatives strictly positive. Since m also increases with v_r, by maximizing $\mathbf{v^T b(v)}$, we minimize v_r^*, which minimizes both v_r and the true error rate m, so the unsupervised algorithm is expected to do the right thing. In all our experiments we obtained values for all p_r, v_r, v_r^* and m very close to zero, which is sufficient in practice even if we did not necessarily find the global minimum using our gradient based method (Section A.3).

The model for $\mathbf{M}$ and the equations we obtained in this section are validated experimentally in Section 5.3. By maximizing the correlation between $\mathbf{v}$ and $\mathbf{b(v)}$ over the training sequence we indeed lower the true misclassification rate m, maximize $\mathbf{v^T t}$ and also lower p_r, v_r and v_r^*.

5.2 Algorithms

5.2.1 Supervised Learning

We want to find the geometric and appearance parameters $\mathbf{w}$ that maximize (in the supervised case) the expected correlation between the principal eigenvector of $\mathbf{M}$ and the ground truth $\mathbf{t}$, which empirically is proportional to the following sum over all training image pairs:

$$J(\mathbf{w}) = \sum_{i=1}^{N} \mathbf{v}^{(i)}(\mathbf{w})^T \mathbf{t}^{(i)}, \tag{5.6}$$

where $\mathbf{t}^{(i)}$ is the ground truth indicator vector for the i-th training image pair. We maximize $J(\mathbf{w})$ by coordinate gradient ascent:

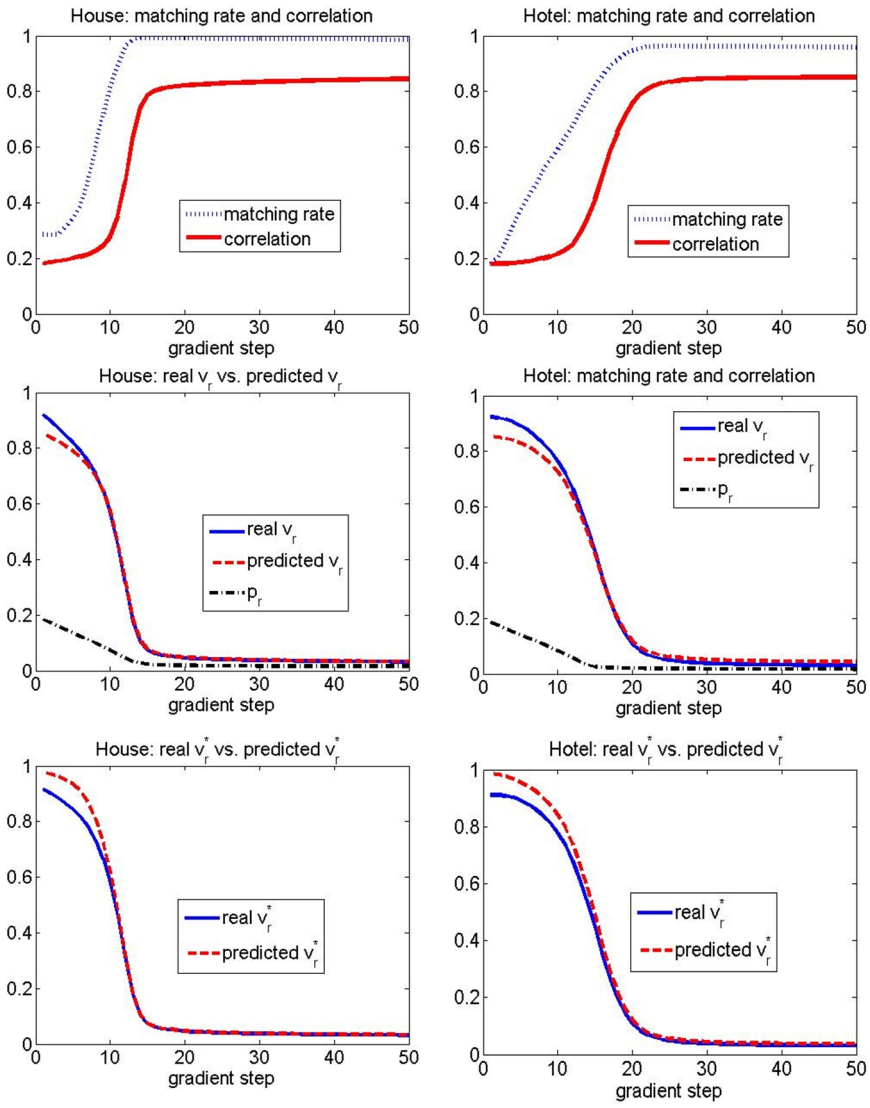

Figure 5.6: Unsupervised learning stage. First row: matching rate and correlation of eigenvector with the ground turht during training per gradient step. The rest of the plots show how the left hand side of Equations 5.4 and 5.5, that is v_r and v_r^*, estimated empirically from the eigenvectors obtained for each image pair, agree with their predicted values (right hand side of Equations 5.4 and 5.5). Results are averages over 70 different experiments, with insignificant standard deviations.

$$w_j^{(k+1)} = w_j^{(k)} + \eta \sum_{i=1}^{N} \mathbf{t_i}^T \frac{\partial \mathbf{v_i}^{(k)}(\mathbf{w})}{\partial w_j}. \tag{5.7}$$

To simplify notations throughout the rest of the thesis we use F' to denote the vector

or matrix of derivatives of any vector or matrix F with respect to some element of $\mathbf{w}$. One possible way of taking partial derivatives of an eigenvector of a symmetric matrix (when λ has order 1) is given in [42], in the context of spectral clustering:

$$\mathbf{v}' = (\lambda\mathbf{I} - \mathbf{M})^{\dagger}(\lambda'\mathbf{I} - \mathbf{M}')\mathbf{v}, \tag{5.8}$$

where

$$\lambda' = \frac{\mathbf{v}^T\mathbf{M}'\mathbf{v}}{\mathbf{v}^T\mathbf{v}}. \tag{5.9}$$

These equations are obtained by using the fact that $\mathbf{M}$ is symmetric and the equalities $\mathbf{v^T v'} = 0$ and $\mathbf{Mv} = \lambda\mathbf{v}$. However, this method is general and therefore does not take full advantage of the fact that in this case $\mathbf{v}$ is the principal eigenvector of a matrix with large eigengap. $\mathbf{M} - \lambda\mathbf{I}$ is large and also rank deficient so computing its pseudo-inverse is not efficient in practice. Instead, we use the power method to compute the partial derivatives of the approximate principal eigenvector: $\mathbf{v} = \frac{\mathbf{M}^n\mathbf{1}}{\sqrt{(\mathbf{M}^n\mathbf{1})^T(\mathbf{M}^n\mathbf{1})}}$. This seems to be related to [7], but in [7] the method is used for segmentation and as also pointed out by [42] it could be very unstable in that case, because in segmentation and typical clustering problems the eigengap between the first two eigenvalues is not large.

Here $\mathbf{M}^n\mathbf{1}$ is computed recursively by $\mathbf{M}^{k+1}\mathbf{1} = \mathbf{M}(\mathbf{M}^k\mathbf{1})$. Since the power method is the preferred choice for computing the leading eigenvector, it is justified to use the same approximation for learning. Thus the estimated derivatives are not an approximation, but actually the exact ones, given that $\mathbf{v}$ is itself an approximation based on the power method. Thus, the resulting partial derivatives of $\mathbf{v}$ are computed as follows:

$$\mathbf{v}' = \frac{(\mathbf{M}^n\mathbf{1})'(\|\mathbf{M}^n\mathbf{1}\|) - \mathbf{M}^n\mathbf{1}/\|\mathbf{M}^n\mathbf{1}\|((\mathbf{M}^n\mathbf{1})^T(\mathbf{M}^n\mathbf{1})')}{\|\mathbf{M}^n\mathbf{1}\|^2}. \tag{5.10}$$

In order to obtain the derivative of $\mathbf{v}$, we first need to compute the derivative of $\mathbf{M}^n\mathbf{1}$, which can be obtained recursively:

$$(\mathbf{M}^n\mathbf{1})' = \mathbf{M}'(\mathbf{M}^{n-1}\mathbf{1}) + \mathbf{M}(\mathbf{M}^{n-1}\mathbf{1})'. \tag{5.11}$$

Since $\mathbf{M}$ has a large eigengap, as shown in [117], this method is stable and efficient. Figure 5.2 proves this point empirically. The method is linear in the number of iterations n, but qualitatively insensitive to n, as it works equally well with n as low as 5. These results are averaged over 70 experiments (described later) on 900 by 900 matrices.

To get a better feeling of the efficiency of our method as compared to Equation 5.8, computing Equation 4 takes 1500 times longer in Matlab (using the function *pinv*) than our method for $n = 10$ on 900 by 900 matrices used in our experiments on the House and Hotel datasets.

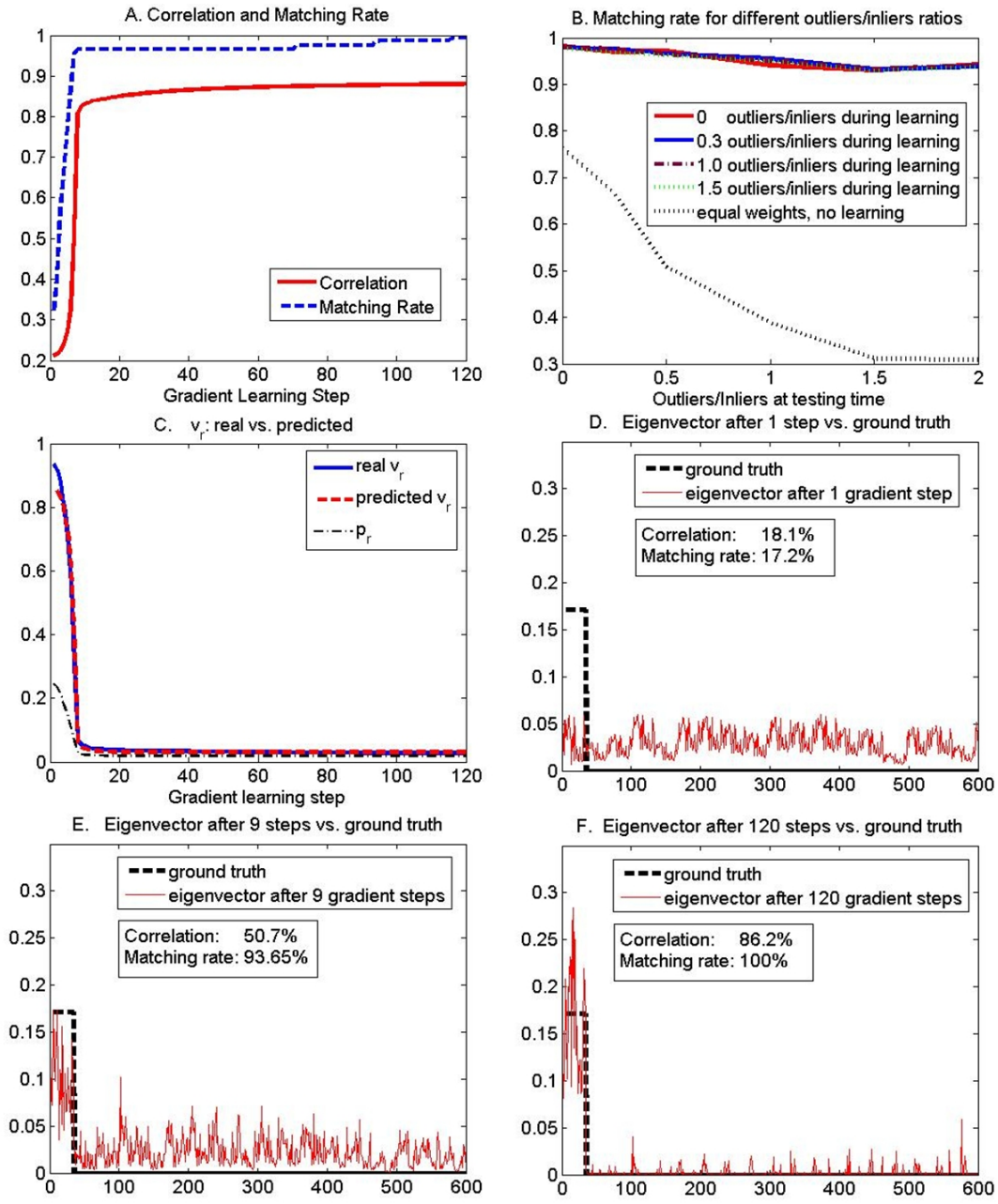

Figure 5.7: Results on faces: correlation between eigenvectors and ground truth, and matching rate during training (top left), matching rate at testing time, for different outliers/inliers ratios at both learning and test time (top-right), verifying Equation 5.4 (middle-left), example eigenvector for different learning steps. Results in the first three plots are averages over 30 different experiments.

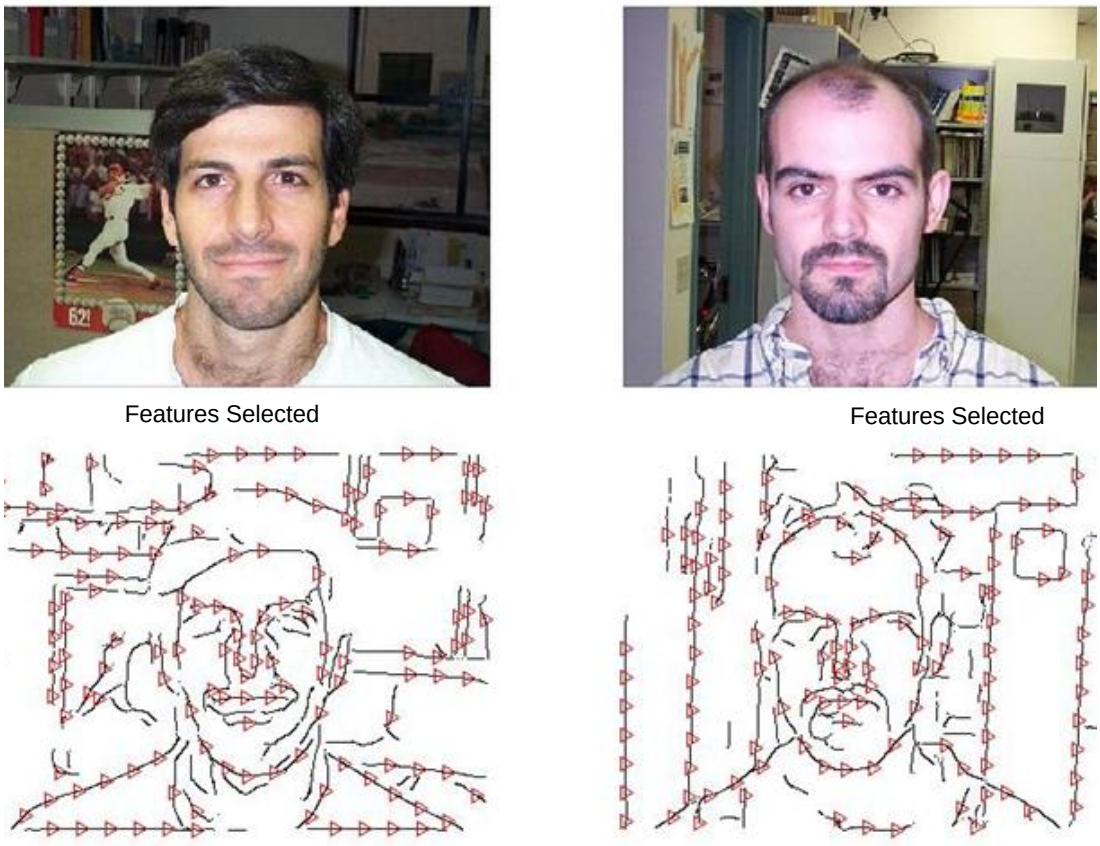

Figure 5.8: Top row: a pair of faces from Caltech-4 dataset used in our experiments. Bottom row: the contours extracted and the points selected as features.

5.2.2 Unsupervised Learning

The idea for unsupervised learning (introduced in Section 5.1), is to maximize v_r^* instead of v_r, which could be achieved either directly or through the maximization of the dot-product between the eigenvector and the binary solution obtained from the eigenvector. In practice it turns out that maximizing the dot-product is simpler and more efficient. Thus, during unsupervised training, we maximize the following function:

$$J(\mathbf{w}) = \sum_{i=1}^{N} \mathbf{v}^{(i)}(\mathbf{w})^T \mathbf{b}(\mathbf{v}^{(i)}(\mathbf{w})). \tag{5.12}$$

The difficulty here is that $\mathbf{b}(\mathbf{v}^{(i)}(\mathbf{w}))$ is not a continuous function and also it may be impossible to express in terms of $\mathbf{w}$, since $\mathbf{b}(\mathbf{v}^{(i)}(\mathbf{w}))$ is the result of the iterative greedy procedure of the spectral matching algorithm. However, it is important that $\mathbf{b}(\mathbf{v}^{(i)}(\mathbf{w}))$ is

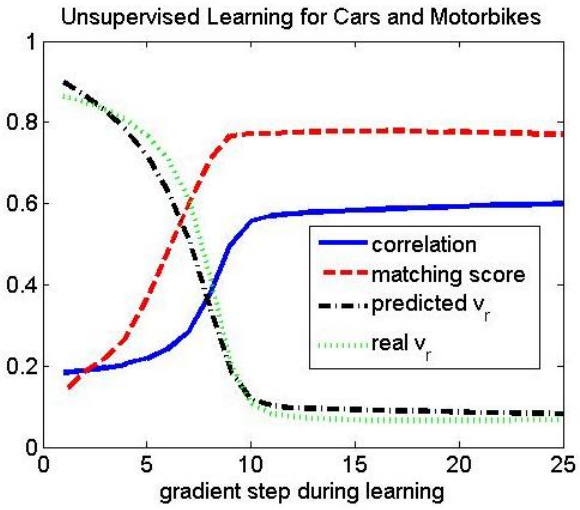

Figure 5.9: Correlation and matching rate w.r.t the ground truth during unsupervised learning for Cars and Motorbikes from Pascal 2007 challenge. Real and predicted v_r decrease as predicted by the model. Results are averaged over 30 different experiments.

piecewise constant and has zero derivatives everywhere except for a finite set of discontinuity points. We can therefore expect that we will evaluate the gradient only at points where $\mathbf{b}$ is constant, and has zero derivatives. Also, at those points, the gradient steps will lower v_r (Equation 5.5) because changes in $\mathbf{b}$ (when the gradient updates pass through discontinuity points in $\mathbf{b}$), do not affect v_r. Lowering v_r will increase $\mathbf{v^T t}$ and also decrease m, so the desired goal will be achieved without having to worry about the discontinuity points of $\mathbf{b}$. This has been verified every time in our experiments. Then, the learning step function becomes:

$$w_j^{(k+1)} = w_j^{(k)} + \eta \sum_{i=1}^{N} \mathbf{b}(\mathbf{v_i^{(k)}}(\mathbf{w}))^T \frac{\partial \mathbf{v_i}^{(k)}(\mathbf{w})}{\partial w_j}. \tag{5.13}$$

5.3 Experimental Analysis

We focus on two objectives. The first one is to validate the theoretical results from Section 5.1, especially Equation 5.4, which establishes a relationship between p_r and v_r, and Equation 5.5, which connects v_r^* to v_r and the error rate m. Each p_r is empirically estimated from each individual matrix $\mathbf{M}$ over the training sequence, and similarly each v_r^* and v_r from each individual eigenvector. Equation 5.4 is important because it shows that the

more likely the pairwise agreements between correct assignments as compared to pairwise agreements between incorrect ones (as reflected by p_r), the closer the eigenvector $\mathbf{v}$ is to the binary ground truth $\mathbf{t}$ (as reflected by v_r), and, as a direct consequence, the better the matching performance. This equation also validates our model for the matching matrix $\mathbf{M}$, which is defined by two expected values, p_0 and p_1, respectively. Equation 5.5 is important because it explains why by maximizing the correlation $\mathbf{v^T b}(\mathbf{v})$ (and implicitly minimizing v_r^*) we in fact minimize v_r and the matching error m. Equation 5.5 basically shows why the unsupervised algorithm will indeed maximize the performance with respect to the ground truth. The results that validate our theoretical claims are shown in Figures 5.6, 5.7 and 5.9 on the House, Hotel, Faces, Cars and Motorbikes experiments. The details of these experiments will be explained shortly. There are a few relevant results to consider. On all four different experiments the correlation between $\mathbf{v}$ and the ground truth $\mathbf{t}$ increases with every gradient step even though the ground truth is unknown to the learning algorithm. The matching rate improves at the same time and at a similar rate with the correlation, showing that maximizing this correlation will also maximize the final performance. In Figure 5.7 we display a representative example of the eigenvector for one pair of faces, as it becomes more and more binary during training. If after the first iteration the eigenvector is almost flat, at the last iteration it is very close to the binary ground truth, with all the correct assignments having larger confidences than any of the wrong ones. Also, on all individual experiments both approximations from Equations 5.4 and 5.5 are becoming more and more accurate with each gradient step, from less than 10% accuracy at the first iteration to less than 0.5% at the last. In all our learning experiments we started from a set of parameters $\mathbf{w}$ that does not favor any assignment ($\mathbf{w} = 0$, which means that before the very first iteration all non-zeros scores in $\mathbf{M}$ are equal to 1). These results motivate both the model proposed for $\mathbf{M}$ (Equation 5.4), but also the results (Equation 5.5) that support the unsupervised learning scheme.

The second objective of our experiments is to evaluate the matching performance, before and after learning, on new test image pairs. The goal is to show that, at testing time, the matching performance after learning is significantly better than if no learning was done.

5.3.1 Unlabeled correspondences

Matching Rigid Objects under Perspective Transformations We first perform experiments on two tasks that are the same as the ones in [30] and our previous work [119] (also presented in the Appendix A). We used exactly the same image sequences (House: 110 images and Hotel: 100 images) both for training and testing and the same features, which were manually selected by the authors of [30]. As in [119] and [30], we used 5 training images for both the House and Hotel sequences, and considered all pairs between them for training.

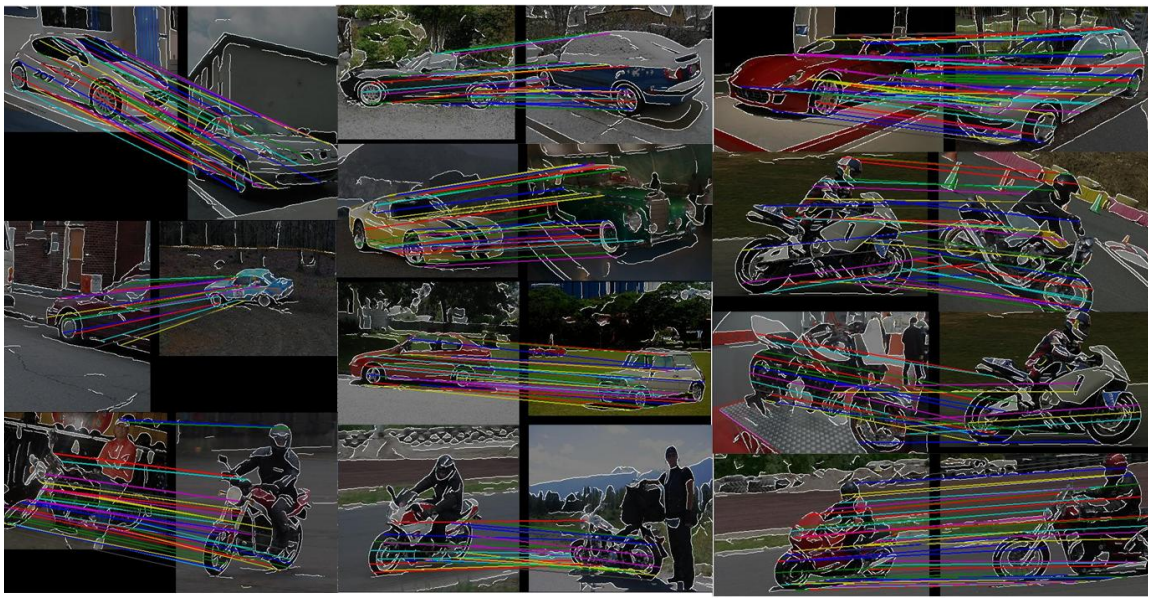

Figure 5.10: Matching results on image pairs from Pascal 2007 challenge. Best viewed in color

For testing we used all the pairs between the remaining images. The pairwise scores $M_{ia;jb}$ are the same as the ones that we previously used in [119], using shape context [14] for local appearance and pairwise distances and angles for the second-order relationships. They measure how well features (i, j) from one image agree in terms of geometry and appearance with their candidate correspondences (a, b).

More explicitly, the pairwise scores are of the type:

$$M_{ia;jb} = e^{-(w_1|s_i - s_a| + w_2|s_j - s_b| + w_3 \frac{|d_{ij} - d_{ab}|}{|d_{ij} + d_{ab}|} + w_4|\alpha_{ij} - \alpha_{ab}|)}. \tag{5.14}$$

Learning consists of finding the vector of parameters $\mathbf{w}$ that maximizes the matching performance on the training sequence. s_a is the shape context of features a, d_{ij} is the distance between features (i, j) and α_{ij} the angle between the horizontal axis and the vector $\vec{ij}$. As in both [30] and [119] we first obtain a Delaunay triangulation and allow non-zero pairwise scores $M_{ia;jb}$ if and only if both (i, j) and (a, b) are connected in their corresponding triangulation. Our previous method [119] is supervised and based on a global optimization scheme that is more likely to find the true global optimum than our unsupervised gradient based method proposed in this Chapter. Therefore it is important to

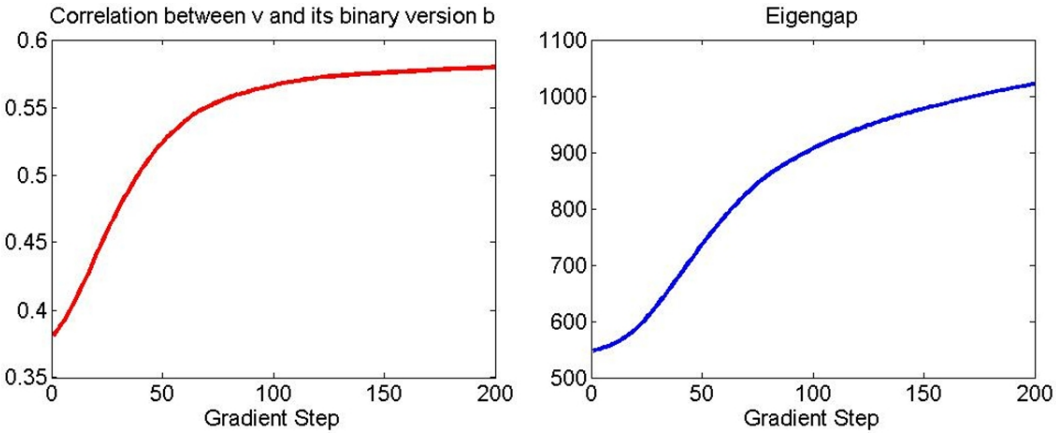

Figure 5.11: Pascal 05, learning stage: (left plot) the average correlation (Equation 5.12) increases with each gradient step, which validates the gradient update rule; (right plot) the normalized eigengap increases, indicating that the eigenvector is more and more stable and robust to noise as it becomes more binary.

Table 5.1: Matching performance on the hotel and house datasets at testing time. The same 5 training images from the House dataset and same testing images from the House and Hotel datasets were used for all three methods.

Dataset	Ours, unsup(5)	Ours, sup(5)	[30], sup(5)
House	99.8%	99.8%	$< 84\%$
Hotel	94.8%	94.8%	$< 87\%$

see that our unsupervised learning method matches our previous results, while significantly outperforming [30] (Table 5.1).

Next we investigate the performance at learning and testing stages of the unsupervised learning method vs. its supervised version (when the ground truth assignments are known). We perform 70 different experiments using both datasets, by randomly choosing 10 training images (and using all image pairs from the training set) and leaving the rest of image pairs for testing. As expected we found that the unsupervised method learns a bit slower on average than the supervised one but the parameters learned are almost identical. In Figure 5.3 we plot the average correlation (between the eigenvectors and ground truth) and matching rate at each gradient step for all training pairs and all experiments vs. each gradient step, for both the supervised and unsupervised cases. It is interesting that while the unsuper-

Table 5.2: Comparison of average matching performance at testing time on the house and hotel datasets for 70 different experiments (10 training images, the rest used for testing). We compare the case of unsupervised learning vs. no learning. First column: unsupervised learning; Second: no learning, equal default weights $\mathbf{w}$.

Datasests	Unsup. Learning	No Learning
House+Hotel	99.14%	93.24%

Figure 5.12: Cropped images (using bounding boxes) from the Pascal 05 training set, used in our classification experiments. The images in this dataset, even though they are cropped using bounding boxes, are more difficult than in the previous experiments, since the objects of the same category have sometimes very different shapes, undergo significant change in viewpoint, and contain background clutter.

vised version tends to converge slower, after several iterations their performances (and also parameters) converge to the same values. During testing the two methods performed identically in terms of matching performance (average percentage of correctly matched features over all 70 experiments). As compared to the same matching algorithm without learned parameters the two algorithms performed clearly better (Table 5.2). Without learning the default parameters (elements of $\mathbf{w}$) were chosen to be all equal.

Matching Deformable 2D Shapes with Outliers The third dataset used for evaluation consists of 30 random image pairs selected from Caltech-4 Faces dataset. The experiments on this dataset are different from the previous ones for two reasons: the images contain not only faces but also a significant amount of background clutter, and, the faces belong to different people, both women and men, with different facial expressions, so there are significant non-rigid deformations between the faces that have to be matched. The features we used are oriented points sampled along contours extracted in the image in a similar fashion as in our previous work [121], described in more detail in Chapter 7. The orientation of each point is the normal vector at that point to the contour where the point was sampled. The points on the faces that have to be matched (the inliers) were selected manually, while the outliers (features in the background) were selected randomly, while making sure that each outlier is not too close (15 pixels) to any other point. For each pair of faces we manually selected the ground truth (the correct matches) for the inliers only. The pairwise scores contain only geometric information about pairwise distances and angles:

$$M_{ia;jb} = e^{-\mathbf{w}^T \mathbf{g}_{ia;jb}}, \tag{5.15}$$

where $\mathbf{w}$ is a vector of 7 parameters (that have to be learned) and $\mathbf{g_{ia;jb}} = [|d_{ij} - d_{ab}|/d_{ij}, |\theta_i - \theta_a|, |\theta_j - \theta_b|, |\sigma_{ij} - \sigma_{ab}|, |\sigma_{ji} - \sigma_{ba}|, |\alpha_{ij} - \alpha_{ab}|, |\beta_{ij} - \beta_{ab}|]$. Here d_{ij} is the distance between the features (i, j), θ_i is the angle between the normal of feature i and the horizontal axis, σ_{ij} is the angle between the normal at point i and the vector $\vec{ij}$, α_{ij} is the angle between $\vec{ij}$ and the horizontal axis and β_{ij} is the angle between the normals of i and j.

We performed 30 random experiments (see results in Figure 5.7) by randomly picking 10 pairs for training and leaving the rest 20 for testing. The results shown in Figure 5.7 are averages over the 30 experiments. The top-left plot shows how, as in the previous experiments, both the correlation $\mathbf{v^T t}$ and the matching performance during training improves with every learning step. At both training and testing times we used different percentages of outliers to evaluate the robustness of the method (top-right plot). The learning method is robust to outliers, since the matching performance during testing does not depend on the percentage of outliers introduced during training (the percentage of outliers is always the same in the left and the right images), but only on the percentage of outliers present at testing time. Without learning (the dotted black plot), when the default parameters chosen are all equal, the performance is much worse and degrades faster as the percentage of outliers at testing time increases. This suggests that learning not only increases the matching rate, but it also makes it more robust to the presence of outliers.

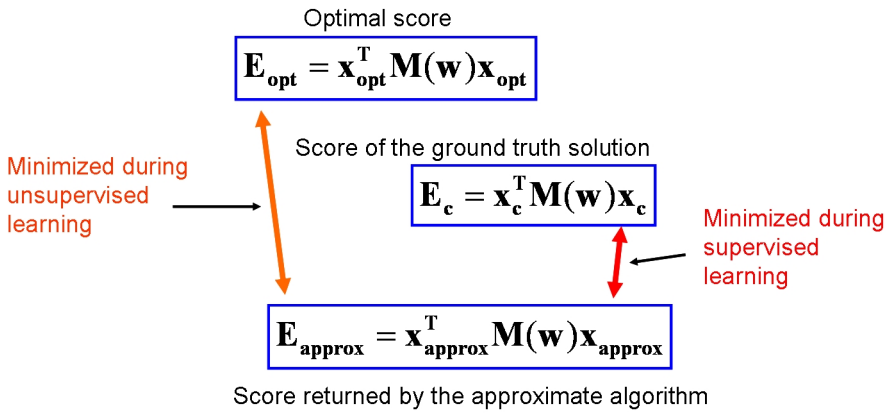

Figure 5.13: During unsupervised learning the parameters are automatically tuned such that the optimality bound on the score obtained gets tighter. The hope is that by tightening this bound the solution will get closer to the ground truth one (which is unknown), since the score of the ground truth is expected to be between the global maximum and the current score.

Table 5.3: Comparison of matching rates for 3 graph matching algorithms before and after unsupervised learning on Cars and Motorbikes from Pascal07 database, with all outliers from the right image allowed and no outliers in the left image. When no outliers were allowed all algorithms had a matching rate of over 75%, with learning moderately improving the performance.

Dataset	SM	PM	GA
Cars: No Learning	26.3%	20.9%	**31.9**%
Cars: With Learning	**62.2**%	34.2%	47.5%
Motorbikes: No Learning	29.5%	26.1%	**34.2**%
Motorbikes: With Learning	**52.7**%	41.3%	45.9%

5.3.2 Unlabeled object classes and correspondences

In our previous experiments every pair of training images contained the same object/category, so a set of inliers exists for each such pair. Next, we evaluated the algorithm on a more difficult task: the training set is corrupted such that half of the image pairs contain different object categories. In this experiment we used cars and motorbikes from Pascal 2007, a much more difficult dataset. For each class we selected 30 pairs of images and for each pair between 30 to 60 ground truth correspondences. The features and the pairwise scores were of the same type as in the experiments on faces: points and their normals selected from pieces of contours. In Figure 5.10 we show some representative results after learn-

ing, with matching rates over 80%; contours are overlaid in white. During each training experiment we randomly picked 5 pairs containing cars, 5 containing motorbikes and 10 discordant pairs: one containing a car and the other one a motorbike (a total of 20 pairs for each learning experiment). For testing we used the remaining pairs of images, such that each pair contains the same object class. The learning algorithm had no knowledge of which pairs are discordant, what classes they contain and which are the ground truth correspondences. As can be seen in Figure 5.9 at each gradient step both the matching rate and the correlation of the eigenvector w.r.t the ground truth increases (monitored only for pairs containing the same category). The model proposed is again verified as shown by the plots of the real and ideal v_r that are almost identical. Not only that the learning algorithm was not significantly influenced by the presence of discordant pairs but it was also able to find a single set of parameters that matched well both cars and motorbikes. Learning and testing results are averaged over 30 experiments.

Using the testing image pairs of cars and motorbikes, we investigated whether this learning method can improve the performance of other graph matching algorithms. We compared spectral matching (SM) using the row/column procedure from [233] during post-processing of the eigenvector, with probabilistic matching (PM) using pair-wise constraints [233], and the well-known graduated assignment algorithm [76] (GA). The same parameters and pair-wise scores were used by all algorithms. When no outliers were allowed all algorithms had similar matching rates (above 75%) with learning moderately improving the performance. When outliers were introduced in the right image (in the same fashion as in the experiments on Faces) the performance improvement after learning was much more significant for all algorithms, with spectral matching benefiting the most (Table 5.3). Spectral matching with learning outperformed the other algorithms with or without learning. This indicates that the algorithm we propose is useful for other graph matching algorithms, but it might be better suited for spectral matching.

5.3.3 Evaluation of Learning in the Context of Recognition

Next we investigate how unsupervised learning for graph matching can improve object recognition. Here we are not interested in developing a recognition algorithm, but only on demonstrating the improvement in recognition after learning for matching, while using a simple object classification scheme: the nearest neighbor algorithm. We believe that better matching should better cluster together images showing objects of the same class, while separating more images of objects from different classes. The nearest neighbor algorithm is perfectly suited for evaluating this intuition. For this experiment we used the cropped training images (using the bounding boxes provided) from Pascal '05 database (see Figure 5.12). We split the cropped images randomly in equal training and testing sets. On the

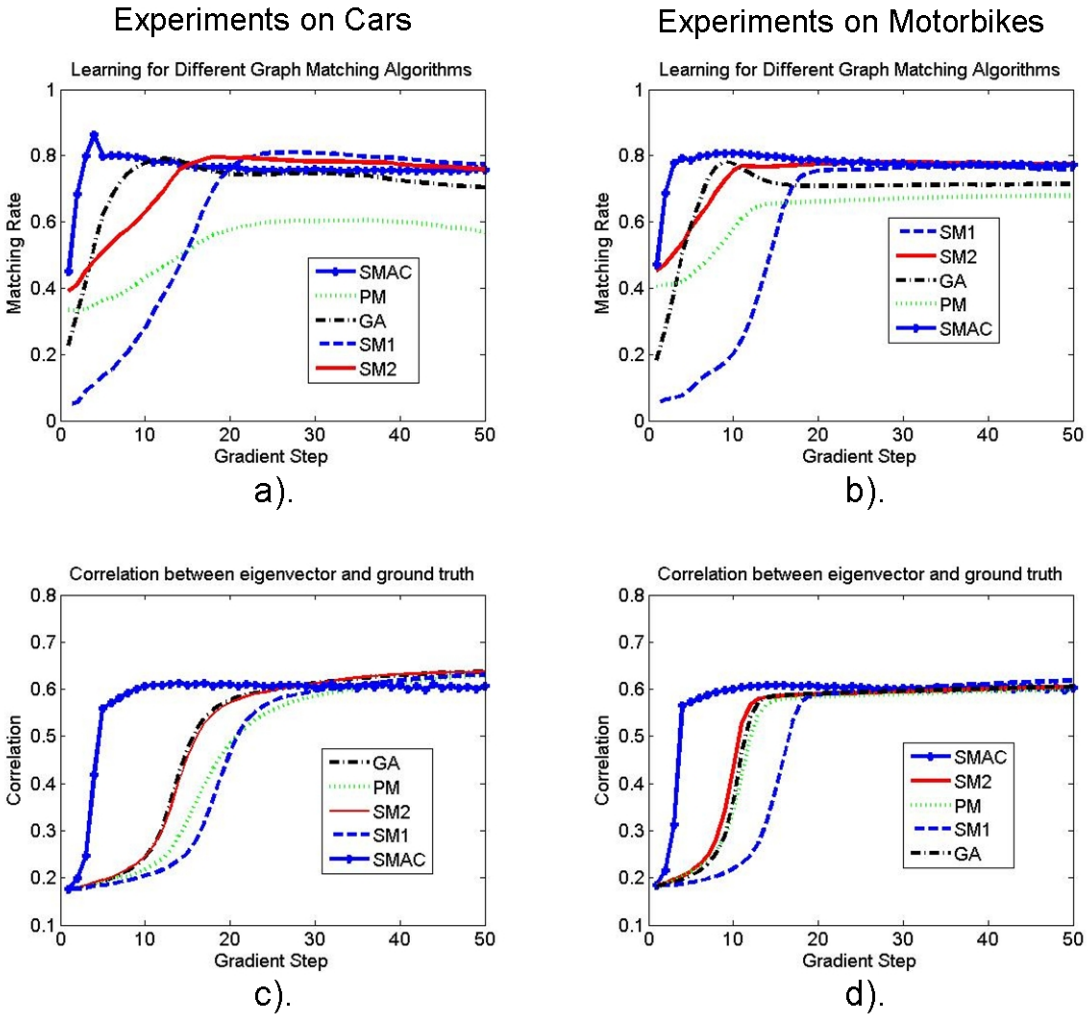

Figure 5.14: First row: the average matching rate with respect to the ground truth, during training for each algorithm at each gradient step. Second row: average correlation between the principal eigenvector and the ground truth during learning for each algorithm at each gradient step. SMAC converges faster then the other algorithms. The final parameters learned by all algorithms are similar.

training set we learn the matching parameters on pairs containing objects from the same category. The features used and the pairwise scores are the same as in the experiments on faces, except that this time we used fully connected models (about $100 - 200$ features per image). At testing time, for each test image, we return the class of the training image that returned the highest matching score $\mathbf{x^T M x}$ (Equation 5.1). We perform this classification

Table 5.4: Comparison of 4-class (bikes, cars, motorbikes and people) classification performance at testing time on the task from Section 5.3.3. Unsupervised learning for graph matching significantly reduces the classification error rate by more than 2-fold

With learning	No learning
80.8%	57.4%

task both with and without learning (with default equal $\mathbf{w}$ weights). The results (Table 5.4) suggest that unsupervised learning for graph matching can be used effectively in an object recognition system. In Figure 5.11 we see some results during learning. In this case we monitored only the eigengap and $\mathbf{b}(\mathbf{v})^{\mathbf{T}}\mathbf{v}$ because we did not have the ground truth available.

5.4 Unsupervised Learning for Other Graph Matching Algorithms

With minimal modification, the unsupervised learning scheme that we proposed can be used for other state-of-the art graph matching algorithms. In Section 5.3.2 we showed experimentally that the parameters learned for spectral matching improved the performance of other algorithms. In this Section we show that instead of using the binary solutions $\mathbf{b}$ returned by spectral matching during each learning step, we can actually use the binary solutions given by the algorithm for which we want to maximize the performance. This will produce a more efficient learning stage, better suited for that specific graph matching algorithm.

To simplify the notation, we use $\mathbf{M}$ instead of $\mathbf{M}(\mathbf{w})$, which is the correct notation since all the pairwise scores in the matrix are functions of $\mathbf{w}$. Let $\mathbf{b}(\mathbf{w})$ be the binary solution given by some graph matching algorithm, for a given $\mathbf{w}$. We first show that by maximizing $\mathbf{b}(\mathbf{w})^{\mathbf{T}}\mathbf{v}(\mathbf{w})$ with respect to $\mathbf{w}$ we maximize a lower bound of the matching score $\mathbf{b}(\mathbf{w})^{\mathbf{T}}\mathbf{M}\mathbf{b}(\mathbf{w})$ relative to the optimal score. Therefore, we expect that as $\mathbf{b}(\mathbf{w})^{\mathbf{T}}\mathbf{v}(\mathbf{w})$ increases, the solution returned by the graph matching algorithm used will get closer and closer to the optimal one. It is true that as $\mathbf{w}$ changes, the optimal solution will also change, however, it is desirable that learning will produce approximate solutions that get close to the optimal solution of the objective score function $\mathbf{x}^{\mathbf{T}}\mathbf{M}(\mathbf{w})\mathbf{x}$. Figure 5.13 illustrates this idea. If an objective function is constructed correctly then for the optimal $\mathbf{w}$ the optimal solution $\mathbf{x}^*(\mathbf{w})$ will be very close to the ground truth solution $\mathbf{t}$ and so will be the corresponding objective scores. The objective of learning is to get approximate solutions that are as close as possible to the ground truth. It follows that a good $\mathbf{w}$ must bring the score returned by

the approximate algorithm as close as possible to the optimal one, while making sure that the objective score function does not become flat and thus noninformative. While this is only a necessary condition for a good $\mathbf{w}$ and not a sufficient one, it might be the only target that can be achieved in the unsupervised scenario, since the ground truth is not available. During unsupervised learning the parameters are automatically tuned such that the global optimality bound on the score obtained gets tighter. The hope is that by tightening this bound the solution will get closer to the ground truth one (which is unknown), since the score of the ground truth is expected to be between the global maximum and the current score. Minimizing the ratio/difference between the optimal objective score and the approximate one can lead, however, to disastrous results if the learning procedure does not take into account certain properties that are problem-specific such as the model that we proposed for $\mathbf{M}$ and validated experimentally.

For any vector $\mathbf{b}$ with n elements and full rank matrix $\mathbf{M}$ of size $n \times n$, we can write $\mathbf{b}$ as:

$$\mathbf{b} = (\mathbf{b^T v_1})\mathbf{v_1} + (\mathbf{b^T v_2})\mathbf{v_2} + ... + (\mathbf{b^T v_n})\mathbf{v_n}, \tag{5.16}$$

where $\mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_n}$ are the eigenvectors of $\mathbf{M}$ ordered in the decreasing order of the magnitudes of their corresponding eigenvalues $\lambda_\mathbf{1}, \lambda_\mathbf{2}, ..., \lambda_\mathbf{n}$. Here we can consider each such $\mathbf{M}$ to be full rank due to the presence of random noise in the pairwise scores from that particular matching problem.

It follows that the quadratic score $\mathbf{b^T M b}$ can be written as:

$$\mathbf{b^T M b} = \lambda_\mathbf{1}(\mathbf{b^T v_1})^\mathbf{2} + \lambda_\mathbf{2}(\mathbf{b^T v_2})^\mathbf{2} + ... + \lambda_\mathbf{n}(\mathbf{b^T v_n})^\mathbf{2}. \tag{5.17}$$

If we consider that $\mathbf{b}$ has unit norm and that λ_1 is the eigenvalue with largest magnitude (also positive, since $\mathbf{M}$ is symmetric with non-negative elements), we immediately obtain the following inequality:

$$\mathbf{b^T M b} \geq (2(\mathbf{b^T v_1})^2 - 1)\lambda_1. \tag{5.18}$$

This inequality is very loose in practice because λ_1 is expected to be much larger than the rest of the eigenvalues. Since $\lambda_1 = \mathbf{v_1^T M v_1}$, where $\mathbf{v_1}$ is the principal eigenvector, and $\mathbf{v_1^T M v_1}$ is an upper bound to the optimal score $\mathbf{x_{opt}^T M x_{opt}}$ that obeys the mapping constraints, we finally obtain

$$\frac{E_{approx}}{E_{opt}} = \frac{\mathbf{b^T M(w) b}}{\mathbf{x_{opt}(w)^T M(w) x_{opt}(w)}} \geq 2(\mathbf{b^T v_1(w)})^2 - 1, \tag{5.19}$$

where $\mathbf{x_{opt}(w)}$ is the optimal solution of Equation 5.1 for a given $\mathbf{w}$. Therefore, by maximizing $\mathbf{b(w)^T v_1(w)}$, we maximize this lower bound and expect $\mathbf{b(w)}$ to approach

Table 5.5: Comparison of matching rates at testing time for different graph matching algorithms before and after unsupervised learning on Cars and Motorbikes from Pascal07 database, with no outliers. The algorithm used during testing was the same as the one used for learning. Results are averages over 30 different experiments. The same parameters were used by all algorithms for the case of no learning with and without outliers: all elements of $\mathbf{w}$ being equal.

Dataset	SM1	SM2	SMAC	GA	PM
Cars: No Learning	44.7%	**86.0** %	83.0%	83.6%	74.2%
Cars: With Learning	77.7%	**90.3** %	82.8%	79.9%	79.8%
Motorbikes: No Learning	49.2%	89.6 %	89.5%	**91.4** %	82.5%
Motorbikes: With Learning	79.0%	**89.1** %	84.7%	80.2%	85.5%

Table 5.6: Comparison of matching rates at testing time for different graph matching algorithms before and after unsupervised learning on Cars and Motorbikes from Pascal07 database, with outliers: all outliers allowed in the right image, no outliers in the left image. The algorithm used during testing was the same as the one used for learning. Results are averages over 30 different experiments. The same parameters were used by all algorithms for the case of no learning with and without outliers: all elements of $\mathbf{w}$ being equal

Dataset	SM1	SM2	SMAC	GA	PM
Cars: No Learning	12.6%	26.3%	**39.1**%	31.9%	20.9%
Cars: With Learning	60.3%	61.6%	**64.8**%	46.6%	33.6%
Motorbikes: No Learning	7.0%	29.7%	**39.2**%	34.2%	26.1%
Motorbikes: With Learning	50.7%	**54.8**%	52.4%	46.8%	42.2%

the optimal solution $\mathbf{x_{opt}}(\mathbf{w})$. This is true for solutions returned by any approximate graph matching algorithm if we maximize instead the correlation between the eigenvector $\mathbf{v_1}(\mathbf{w})$ and the solution $\mathbf{b}(\mathbf{w})$ returned by that specific algorithm. This suggests that the same unsupervised learning scheme may be applied to other algorithms as well, by simply replacing in the gradient step the binarized eigenvector $\mathbf{b}(\mathbf{v_1}(\mathbf{w}))$ returned by the spectral matching algorithm with the solution $\mathbf{b}(\mathbf{w})$ returned by the graph matching algorithm for which we want to learn the parameters. Therefore, the general unsupervised learning procedure for graph matching is:

$$\mathbf{w}_j^{(k+1)} = \mathbf{w}_j^{(k)} + \eta \sum_{i=1}^{N} \mathbf{b_i}(\mathbf{w^{(k)}})^T \frac{\partial \mathbf{v_i}^{(k)}(\mathbf{w})}{\partial \mathbf{w}_j}. \tag{5.20}$$

5.4.1 Experimental Analysis

We perform unsupervised learning for four state of the art graph matching algorithms on the Cars and Motorbikes data from Pascal 2007 challenge, the same as the one used in Section 5.3.2. We use the same type of features (oriented points selected from pieces of contours) as in Section 5.3.2. The algorithms we compare are: spectral matching [117] in two versions (SM1 and SM2), spectral matching with affine constraints (SMAC) [43], graduated assignment (GA) [76] and probabilistic graph matching (PM) [233]. For spectral matching we have two versions: SM1 is spectral matching with the discretization procedure that we originally proposed in [117] and SM2 for which, before the greedy discretization step from [117], we apply the column/row rectangular bistochastic normalization from [43] to the principal eigenvector. Note that, except for SM1, all algorithms use this normalization procedure to their relaxed solution before the discretization step.

For each algorithm, we perform 30 different learning and testing experiments for each class and we average the results. For each experiment we randomly pick 10 pairs of images for learning (with outliers) and leave the remaining 20 for testing (with and without outliers). During training we add outliers to one image in every pair, such that the ratio of outliers to inliers is 0.5. The other image from the pair contains no outliers. We introduce this moderate amount of outliers during training in order to test the robustness of the unsupervised learning method in real world experiments, where, especially in the unsupervised case, it is time consuming to enforce an equal number of features in both images in every pair. During testing we have two cases: we had no outliers in both images in the first case, and allowed all outliers possible in only one image in the second case. The number of outliers introduced was significant, the ratio of outliers to inliers ranging from 1.4 to 8.2 for the Cars class (average of 3.7), and from 1.8 to 10.5 for the Motorbikes class (average of 5.3). As in all our other tests, by an inlier we mean a feature for which there exists a correspondence in the other image, according to the ground truth, whereas an outlier has no such correct correspondence. The inliers were manually picked, the same as the ones used in Section 5.3.2, whereas the outliers were chosen randomly on pieces of contours such that no outlier is closer than 15 pixels to any other feature selected.

In Figure 5.14 we display the behavior of each algorithm during learning: average matching rate and average correlation of the eigenvector with the ground truth at each learning step. There are several important aspects to notice and discuss: the correlation between the eigenvector and the ground truth increases with every gradient step for all algorithms, SMAC converging much faster than the others. This is reflected also in the matching rate, that increases much faster for SMAC. All algorithms benefit from learning, as all matching rates improve significantly after several iterations. The starting set of parameters $\mathbf{w}$ were all zeros and the final $\mathbf{w}$'s learned are similar for all algorithms. GA and SMAC have a rapid

improvement in the first 10 steps, followed by a small decline and a plateau for the remaining iterations. This might suggest that for GA and SMAC learning should be performed only for a few iterations. For the other three algorithms learning constantly improves the matching rate during training, with SM1 converging the slowest. It is interesting to see how much more benefit the original spectral matching algorithm (SM1) gets from learning, having a very poor performance during the first stages of learning as compared to the other algorithms, and ending with a matching rate that sometimes exceed the others' performance. We discuss more about this differences next, when we look at their performance during testing.

In Table 5.5 we show the test results of all algorithms with and without learning, for both datasets, when no outliers are introduced. Without learning all algorithms use a parameter vector $\mathbf{w} = [0.2, 0.2, 0.2, 0.2, 0.2]$ for both cases, with and without outliers, and both datasets. In the case of no outliers (Table 5.5) all algorithms perform very well except for SM1, with a much poorer performance. Learning significantly improves the performance for SM1 and moderately for PM and SM2, but it seems to lower it for SMAC and GA. This seems to agree with the algorithms performance during learning, when SMAC and GA have a rather curious behavior, with their rates dropping slightly after a short phase of sudden improvement. As mentioned earlier, this drop could probably be fixed in the case of SMAC and GA by stopping the learning iterations early. While this seems to suggest that learning is not beneficial to SMAC and GA, its real benefit becomes evident when outliers are introduced.

In our experiments, the more outliers we introduce during testing the more beneficial learning becomes. This is also in agreement with our experiments on faces (Table 5.7). In Table 5.6 we show the results with and without learning in the presence of a significant number of outliers (no outliers in one image and all possible outliers in the other image, as explained previously). As before, in the case of no learning $\mathbf{w} = [0.2, 0.2, 0.2, 0.2, 0.2]$ for all algorithms and both datasets. It is evident that learning significantly improves the performance of all algorithms, with SM1 benefitting the most. SM1 performs very poorly when no learning is performed, but it becomes competitive after learning, outperforming GA and PM. Learning is more beneficial to SM1 probably because the learning algorithm was inspired by and designed for spectral matching. It is possible that spectral matching in its original form ($SM1$) is more sensitive to the choice of parameters and relies heavier on having high scores between pairs of incorrect assignments as seldom as possible, while keeping the high scores between correct ones as often as possible. This makes sense because learning, by minimizing $p_r = p_0/p_1$, makes the pairwise agreements between incorrect assignments more accidental relative to the likely agreements between correct ones.

The results shown in this Section strongly suggest that our unsupervised learning scheme can significantly improve the performance of other algorithms on difficult data (such as the

Cars and Motorbikes from Pascal 2007) in the presence of a large number of outliers. Among these state-of-the art algorithms, there is no clear winner in our experiments. The best performing algorithms in most cases, that also benefit the most from learning especially in the case of outliers, are our spectral matching [117] with a modified post-processing of the eigenvector, SM2, and spectral matching with affine constraints SMAC [43]. These algorithms are very close technically, with SMAC being slower than SM2 (see [43] for a comparison) and more involved technically and difficult to implement due to the introduction of the affine constraints. The authors suggest in [43] that a bistochastic normalization of the edge matrix (obtained from $\mathbf{M}$) can further improve the performance. In our experiments, however, with or without outliers or learning, we found that the normalization they suggest, applied to their algorithm SMAC, degrades its performance in all cases. In terms of computational complexity the most expensive is the graduated assignment algorithm (GA), and the least expensive, probabilistic graph matching (PM). The difference in complexity between PM and SM1 and SM2 is minimal.

5.5 Extension: Unsupervised Learning for Hypergraph Matching

In this section we present a direct extension of our learning scheme for graph matching to hypergraphs. The idea is based on the same algorithm presented in Section 5.2.2. The main difference is the extension of the derivative computation of the principal eigenvector of a matrix to principal eigenvectors of tensors, while the update step based on gradient descent remains the same.

We first show how one can compute the derivative of a tensor's leading eigenvector using the extension of the power method to tensors presented in the work of [53]. For clarity of presentation, we limit our case to third-order potentials, but the same algorithm can be immediately extended to higher order cliques. In the case of matching for computer vision, however, third order potentials are probably as far as one would want to go, given that, as the authors of [53] suggested, they are enough for matching invariant to perspective transformations.

The leading eigenvector $\mathbf{v}$ of a super-symmetric three dimensional tensor $\mathbf{H}$ can be computed by the following iterative power method:

$$\mathbf{v_{n+1}} = \frac{\mathbf{h_n}}{\sqrt{\mathbf{h_n^T h_n}}}, \tag{5.21}$$

where $h_{nia} = \sum_{jb;kc} H_{ia;jb;kc} v_{njb} v_{nkc}$

The derivative of $\mathbf{v}$ at iteration $n+1$ can be also computed recursively by an extension of the method presented in Section 5.2.1:

$$\mathbf{v'_{n+1}} = \frac{\|\mathbf{h_n}\|\mathbf{h'_n} - ((\mathbf{h_n^T h'_n})/\|\mathbf{h_n}\|)\mathbf{h_n}}{\|\mathbf{h_n}\|^{\mathbf{2}}}, \tag{5.22}$$

where the derivative of $\mathbf{h_n}$ can be computed as follows:

$$h'_{nia} == \sum_{jb;kc} H'_{ia;jb;kc} v_{njb} v_{nkc} + 2 \sum_{jb;kc} H_{ia;jb;kc} v_{njb} v'_{nkc} \tag{5.23}$$

Once we have the derivative of the principal eigenvector of $\mathbf{H}$ we can use the same gradient method for learning the parameters as in the case of learning for graph matching:

$$w_j^{k+1} = w_j^k + \eta \sum_{i=1}^{N} \mathbf{b_i} (\mathbf{w^{(k)}})^T \frac{\partial \mathbf{v_i}^{(k)}(\mathbf{w})}{\partial w_j}. \tag{5.24}$$

5.6 Conclusions

We presented an efficient way of performing both supervised and unsupervised learning for graph matching, in the context of computer vision applications. We showed that the performance of our unsupervised learning algorithm is comparable with the one in the supervised case. The algorithm significantly improves the matching performance of several state-of-the-art graph matching algorithms, which makes it widely applicable.

Chapter 6

Learning with Spectral MAP Inference

In Chapter 3 we presented an algorithm for MAP inference for Markov Random Fields, inspired from spectral graph matching. Since graph matching and MAP inference can be formulated as similar integer quadratic programs, it is not surprising that similar algorithms can be applied to both problems, as shown in the previous chapters. Here we further explore the connection between graph matching and MAP inference by introducing a method for learning the parameters for MAP inference that is inspired from the learning method we presented in Chapter 5 for graph matching. In the case of graph matching our learning method is the first to learn the parameters of the higher-order terms in both supervised and unsupervised fashions, unlike the case of graphical models where learning is a well studied problem. Most previous learning methods for MRFs and DRFs are based on probabilistic formulations, such as maximum likelihood, maximum pseudolikelihood or penalized log pseudolikelihood [106]. In contrast, here we present a method for learning the parameters that is not probabilistic and whose only goal is to find the parameters that optimize the performance of the inference algorithm, which is weakly related to [204] and Max-Margin Markov Networks [205]. Moreover, we show that for some simpler problems, such as image denoising, it is possible to learn these parameters in a completely unsupervised manner, similar to our unsupervised learning approach to graph matching. If in the case of matching our goal was to maximize the dot product between the eigenvector of $\mathbf{M}$ and the ground truth matching solution, in this case we aim to maximize the dot-product between the global optimum of the relaxed MAP inference problem with L2 constraints (presented in Chapter 3) and the ground truth labels. The details are discussed next.

6.1 Problem Formulation

We reiterate here the MAP inference formulation as an integer quadratic program, also presented in Chapter 3:

$$\mathbf{x}^* = \text{argmax } \mathbf{x^T M x} + \mathbf{D x}. \tag{6.1}$$

Here $\mathbf{x}$ is an indicator vector such that $x_{ia} = 1$ if label a is assigned to site i and zero otherwise. For MAP inference problems only many-to-one constraints are usually required, unlike the graph matching problem where one feature from one graph can match at most one other feature from the other. Here $M_{ia;jb}$ corresponds to the second-order potential (same as $1/2Q_{ia;jb}$ from Chapter 3) describing how well the label a at site i agrees with the label b at site j, given the data. $M_{ia;jb}$ could also be a smoothness term independent of the data (such as the Potts model) that simply encourages neighboring sites to have similar labels. We set $M_{ia;jb} = 0$ if $i = j$ or if the sites i and j are not connected since $M_{ia;jb}$ should describe connections between different sites. For each pair of site i and its possible label a, the first order potentials are represented by D_{ia}, which in general describes how well the label a agrees with the data at site i. Without loss of generality we require both $\mathbf{M}$ and $\mathbf{D}$ to have non-negative elements (see Chapter 3 for more details).

The pairwise terms we use here are inspired from the work of [106] on Discriminative Random Fields, having a similar formulation. Unlike that work, we drop the unary terms, and include the local, unary information into the pairwise terms, since in our experiments this approach gives better results. We consider these interaction potentials to take the form of logistic classifiers. In this Chapter we present the case of binary classification problems, which could easily be extended to multi-class problems. For binary classification problems with submodular functions graph-cuts are known to give exact solutions. In this Chapter we demonstrate that in practice we can outperform previous work using graph-cuts, not by using a better optimization algorithm (since ours is not optimal), but by learning a better set of parameters.

$$M_{ia;jb} = \sigma(\mathbf{w^T}[t_a;\ t_a\mathbf{u_i};\ t_b;\ t_b\mathbf{u_j};\ t_a t_b \mu_{\mathbf{ij}}]). \tag{6.2}$$

Here $t_a = 1$ if label $a = 1$ and $t_a = -1$ if label $a = 0$; $\mathbf{u_i}$ encodes the local/unary information (normally used in the unary potentials) and $\mu_{\mathbf{ij}}$ encodes the spatial, data dependent, interactions usually used in DRFs and CRFs. If sites i and j are not connected then $M_{ia;jb} = 0$.

6.2 Learning

The spectral inference method presented in Chapter 3 is based on a fixed point iteration similar to the power method for eigenvectors, which maximizes the quadratic score under the L2 norm constraints $\sum_{b=1}^{L} {v_{ib}^*}^2 = 1$. These constraints require that the sub-vectors corresponding to the candidate labels for each site i have norm 1.

$$v_{ia}^* = \frac{\sum_{jb} M_{ia;jb} v_{jb}^*}{\sqrt{\sum_{b=1}^{L} {v_{ib}^*}^2}}. \tag{6.3}$$

This equation looks similar to the eigenvector equation $\mathbf{Mv} = \lambda\mathbf{v}$ if it were not for the site-wise normalization instead of the global one which applies to eigenvectors. Starting from a vector with positive elements, the fixed point $\mathbf{v}^*$ of the above equation has positive elements, is unique and it is a global maximum of the quadratic score under the constraints $\sum_a v_{ia}^2 = 1$, due to the fact that $\mathbf{M}$ has non-negative elements (Theorem 5, [10]).

The learning method for the MAP problem, which we propose here, is based on gradient ascent, similar to the one for graph matching, and requires taking the derivatives of $\mathbf{v}$ with respect to the parameters $\mathbf{w}$.

Let $\mathbf{M_i}$ be the non-square submatrix of $\mathbf{M}$ of size $nLabels$ by $nLabels * nSites$, corresponding to a particular site i. Also let $\mathbf{v_i}$ be the corresponding sub-vector of $\mathbf{v}$, which is computed by the following iterative procedure. Let n be a particular iteration number:

$$\mathbf{v_i^{(n+1)}} = \frac{\mathbf{M_i v_i^{(n)}}}{\sqrt{(\mathbf{M_i v_i^{(n)}})^\mathbf{T} (\mathbf{M_i v_i^{(n)}})}}. \tag{6.4}$$

Let $\mathbf{h_i}$ be the corresponding sub-vector of an auxiliary vector $\mathbf{h}$ defined at each iteration as follows:

$$\mathbf{h_i} = \frac{\mathbf{M_i'} \mathbf{v_i^{(n)}} + \mathbf{M_i'} (\mathbf{v_i^{(n)}})'}{\sqrt{(\mathbf{M_i v_i^{(n)}})^\mathbf{T} (\mathbf{M_i v_i^{(n)}})}}. \tag{6.5}$$

Then the derivatives of $\mathbf{v_i^{(n+1)}}$ with respect to some element of $\mathbf{w}$, at step $n+1$, can be obtained recursively as a function of the derivatives of $\mathbf{v_i^{(n)}}$ at step n, by iterating the following update rule:

$$(\mathbf{v^{(n+1)}})' = \mathbf{h} - (\mathbf{h^T v^{(n+1)}}) \mathbf{v^{(n+1)}}. \tag{6.6}$$

This update rule, which can be easily verified, is similar to the one used for computing the derivatives of eigenvectors.

The partial derivatives of the individual elements of $\mathbf{M}$ with respect to the individual elements of $\mathbf{w}$ are computed from the equations that define these pairwise potentials, given

in Equation 6.8. Of course, other differential functions can also be used to define these potentials.

In both supervised and unsupervised cases, the learning update step is similar to the one used for learning graph matching. Here we present the supervised case. In the case of MAP problems we have noticed that unsupervised learning can be successfully applied only to simpler problems, as shown in the experiments Section. This is due to the fact that in MAP problems it is easily possible to find parameters that will strongly favor one label and make the solution of the relaxed problem almost perfectly discrete. The supervised learning rule for MAP is ($\mathbf{t_i}$ is the ground truth labeling for the i-th training image):

$$w_j^{k+1} = w_j^k + \eta \sum_{i=1}^{N} \mathbf{t_i}^T \frac{\partial \mathbf{v_i}^{(k)}(\mathbf{w})}{\partial w_j}. \tag{6.7}$$

6.3 Obtaining a Discrete Solution for MAP

In Chapter 4 we presented the Integer projected Fixed Point (IPFP) algorithm, while focusing more on the graph matching problem. Here we show the form of this algorithm for MAP inference problems, in the general case, when the unary potentials $\mathbf{D}$ are also used. As we mentioned previously, in our actual implementation we ignore the unary potentials and include the local appearance terms into the pairwise ones, as shown in Equation 6.8:

1. Initialize $\mathbf{x}^* = \mathbf{x_0}, \quad S^* = \mathbf{x_0^T M x_0 + D x_0}, \quad k = 0, \qquad \text{where } x_i \geq 0 \quad \text{and } \mathbf{x} \neq \mathbf{0}$

2. Let $\mathbf{b_{k+1}} = P_d(\mathbf{2Mx_k + D}), \quad C = \mathbf{2x_k^T M(b_{k+1} - x_k) + D(b_{k+1} - x_k)}, \quad D = \mathbf{(b_{k+1} - x_k)^T M(b_{k+1} - x_k)}$

3. If $D \geq 0$ set $\mathbf{x_{k+1}} = \mathbf{b_{k+1}}$. Else let $r = \min\{-C/(2D), 1\}$ and set $\mathbf{x_{k+1}} = \mathbf{x_k} + \mathbf{r(b_{k+1} - x_k)}$

4. If $\mathbf{b_{k+1}^T M b_{k+1} + D b_{k+1}} \geq S^*$ then set $S^* = \mathbf{b_{k+1}^T M b_{k+1} + D b_{k+1}}$ and $\mathbf{x}^* = \mathbf{b_{k+1}}$

5. If $\mathbf{x_{k+1}} = \mathbf{x_k}$ stop and return the solution $\mathbf{x}^*$

6. Set $k = k + 1$ and go back to step 2.

Here P_d is the projection on the domain of many-to-one integer constraints, which can be implemented easily in linear time. This algorithm, as mentioned in Chapter 4, can be seen as a parallel extension of ICM that has guaranteed climbing property and strong convergence to a discrete solution in the case of MAP inference. In practice it often converges in less than 10 iterations.

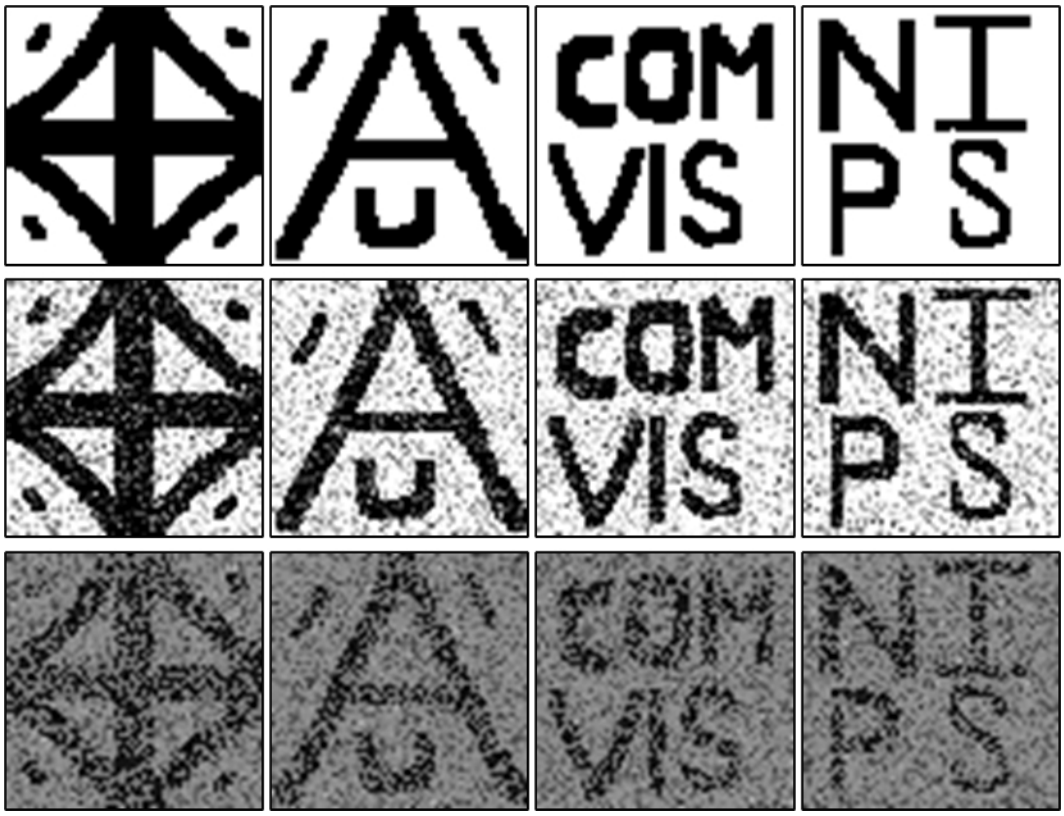

Figure 6.1: First row: original binary images (left one used for training, next three for testing). Second row: images corrupted with unimodal noise. Third row: images corrupted with bimodal noise.

6.4 Experiments

In order to compare our method to previous work on DRFs [106], we have followed exactly the experiments of [106] on image denoising. This is very easily achieved since all the necessary implementation details and test data are given in [106]. The task is to obtain denoised images from corrupted binary 64×64 images. We used the same four images and the same noise models. For the easier task the noise model is an unimodal Gaussian with mean 0 and $std = 0.3$ added to the $0 - 1$ binary images. For the more difficult task we used as in [106], for each class, a different mixture of two Gaussians with equal mixing weights yielding a bimodal noise. The model parameters (mean, std) for the two Gaussians were [(0.08, 0.03), (0.46,0.03)] for the foreground class and [(0.55, 0.02), (0.42,0.10)] for the

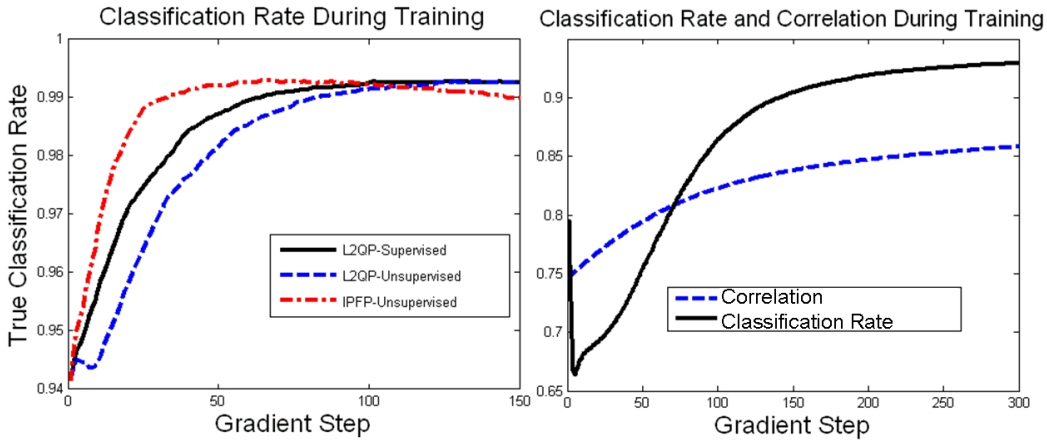

Figure 6.2: Left plot: supervised and unsupervised learning when unimodal noise is used. Right plot: supervised learning when bimodal noise is used. Results are averages over 5 training images for each learning gradient step.

background class. The original images together with examples of their noisy versions are shown in Figure 6.1.

Unlike [106] which uses 50 randomly generated noisy versions of the first image for training, we used only 5 such images. For the simpler task we also performed completely unsupervised learning getting almost identical results (Table 6.1). Our results were significantly better for the simpler noise model, while matching the results from [106] for the more difficult noise model. Also note that our learning method is easier to implement and improves the performance of IPFP, not just L2QP (our spectral MAP inference algorithm from Chapter 3 for which it was originally designed). The pairwise potentials we used are:

$$M_{ia;jb} = \sigma(\mathbf{w}^{\mathbf{T}}[t_a;\ t_a I_i;\ t_b;\ t_b I_j;\ t_a t_b |I_i - I_j|]), \tag{6.8}$$

where I_i is the value of pixel i in the image and $|I_i - I_j|$ is the absolute difference in image pixel values between connected sites i and j. We used as [106] 4-connected lattices. We also experimented with 8-connected neighborhoods with no significant difference in performance.

The start parameters for all learning experiments were $\mathbf{w} = [0.5;\ -1;\ 0.5;\ -1;\ -0.5;\ 1]$ and the learned ones were: $\mathbf{w} = [1.2673;\ -2.5523;\ 1.2673;\ -2.5523;\ -2.4977;\ 0.4673]$ (unimodal noise, unsupervised), $\mathbf{w} = [1.2619;\ -2.5467;\ 1.2619;\ -2.5467;\ -2.6321;\ 0.2591]$ (unimodal noise, supervised) and $\mathbf{w} = [1.9755;\ -5.2362;\ 1.9755;\ -5.2362;\ -2.9942;\ 0.2210]$ (bimodal, supervised).

Table 6.1: Comparisons with [106] on the same experiments. In [106] 50 noisy versions of the first image are used for training. We used only 5 noisy versions of the first image are used for training. For testing both approaches use 50 noisy versions of the remaining three images. Note that the unsupervised learning matches the performance of the supervised one. The inference method used in [106] is graph cuts and the learning methods are maximum pseudolikelihood (PL) and maximum penalized pseudolikelihood (PPL)

Algorithm	L2QP	IPFP	L2QP+IPFP	[106] (PPL)	[106] (PL)
Unimodal (sup.)	0.75%	0.73%	**0.72**%	2.3%	3.82%
Unimodal (unsup.)	0.85%	0.69%	**0.68**%	NA	NA
Bimodal (sup.)	7.15%	15.94%	8.45%	**6.21**%	17.69%

6.5 Discussion and Conclusions

In this Chapter we have presented a different approach to learning for MAP problems. Our method avoids the computational bottlenecks of most probabilistic approaches such as maximum likelihood and pseudolikelihood, which need the estimation of the normalization function Z. We also demonstrate that for simple problems such as image denoising it is sometimes possible to perform successful learning in a completely unsupervised manner. The main reason why in the case of MAP problems unsupervised learning does not work as well as in graph matching is the different structure of the matrix $\mathbf{M}$. If in the case of graph matching this matrix contains a single strong cluster formed mainly by the correct assignments in the case of MAP problems, the matrix could contain several such clusters corresponding to completely different labelings. The idea of accidental alignment is not applicable to most MAP problems, thus the learning algorithm could converge to several parameter solutions that would binarize the continuous solution, in which case supervised learning is required. Moreover, even in the case of supervised learning, training is sensitive to initialization in the case of MAP problems, a fact that was also shown by other authors. It is important that in our experiments our inference and learning methods, both supervised and unsupervised, match or even exceed some recently published approaches.

Chapter 7

Spectral Matching for Object Recognition

Object category recognition is an important part of computer vision and machine learning. The task is very challenging and the problem ill-posed, because there is no formal definition of what constitutes an object category. Philosophically, any arbitrary definition could characterize an object category (*All things with a white patch*), but such categories are not useful to us, humans. While people largely agree on common, useful categories, it is still not clear which are the objects' features that help us group them into such categories.

Our proposed approach is based on the observation that for a wide variety of common object categories, it is the shape that matters more than their local appearance. For example, it is the shape, not the color or texture, that enables a plane to fly, an animal to run or a human hand to manipulate objects. Many categories are defined by their function and it is typically the case that function dictates an object's shape rather than its low level surface appearance.

In this thesis we represent these object category models as cliques of very simple features (sparse points and their normals), and focus only on the pairwise geometric relationships between them. This differs from the popular bag of words model [47], which concentrates exclusively on local features, ignoring the geometric higher-order interactions between them. Although shape alone has previously been used for category recognition [3, 219], we demonstrate for the first time (to the best of our knowledge) that simple geometric relationships can be successful in a semi-supervised setting, without the use of any appearance features.

The importance of shape for object recognition has been emphasized by Belongie [13] and Berg [15]. Belongie introduce the shape context descriptor, which is a local, unary (one per feature) descriptor that loosely captures global shape transformations, but, by nature, is also sensitive to background clutter. Berg used second-order geometric relationships between features, but the main focus was still on the local geometric information (using the

geometric blur descriptor). In this paper we emphasize the importance of pairwise geometric relationships by omitting the local appearance information and employing a more extensive set of parameters (Section 7.3) for representing the pairwise geometric relationships; in contrast, Berg use only the pairwise distance and a single angle. Also, unlike previous work, we learn these parameters in order to better capture the space of pairwise deformations.

The idea of using geometric constraints with shape features (edges and surfaces) for object recognition dates back to the 1980s [54], [133]. For instance, Grimson and Lozano Perez [54] propose matching using an interpretation tree that enforces a global geometric transformation and requires a combinatorial search space. Our work focuses solely on pairwise geometric relationships that give us flexibility in matching deformable objects. We address the combinatorial problem by employing a fast approximate algorithm.

There are two popular trends in object recognition. The first is to formulate it as a matching problem, and to use either a nearest neighbor approach [15] or a classifier [77]. The second trend is to formulate object recognition as an inference problem and use the machinery of graphical models [63, 64, 32, 45]. Our work is influenced by ideas for matching using second-order geometric relationships [15, 117], but unlike existing approaches in matching, we build a category model that is a compact representation of the training set (similar to approaches using Conditional Random Fields). Nearest neighbor classifiers perform well but they are computationally expensive on large datasets, because each training example is considered separately. The SVM classifier is more efficient, but can be sensitive to background clutter because, while it learns which images are relevant for classification, it does not go deeper to learn the relevant information within those images (nor does it optimize the kernel function for classification). We integrate several training images into a single abstract shape model that captures both common information shared by several training images as well as useful information that is unique to each training image. We also automatically discover and remove the irrelevant clutter from training images, keeping only the features that are indeed useful for recognition. This gives us a more compact representation, which reduces the computational and memory cost, and improves generalization.

The use of second-order geometric relationships enables our algorithm to successfully overcome problems often encountered by previous methods from the literature:

1. During training our algorithm is translation invariant, robust to clutter, and does not require aligned training images. This is in contrast with previous work such as [15, 155, 149, 65].

2. We efficiently learn models consisting of hundreds of fully interconnected parts (capturing both short- and long-range dependencies). Most previous work handles models only up to 30 sparsely inter-connected parts, such as the star-shaped [64, 44], k-fan, [45] or hierarchical models [60, 23]. There has been work [32] handling hundreds of

parts, but such models are not translation invariant and each object part is connected only to its k-nearest neighbors.

3. We select features based on how well they work together as a team rather than individually (as is the case in [45, 155]). This gives us a larger pool of very useful features, which are collectively discriminative, if not necessarily on an individual basis.

We formulate the problem as follows: given a set of negative images (not containing the object) and weakly-labeled positive images (containing an object of a given category somewhere in the image), the task is to learn a category shape model that can be used both for the *localization* and *recognition* of objects from the same category in novel images (Section 7.3). This problem is challenging because we do not have any prior knowledge about the object's location in the training images. Also these images can contain a substantial amount of clutter that is irrelevant to the category that we want to model. All that is known at training time is that the object is present somewhere in the positive training images and absent in the negative ones.

7.1 Unary Features: Extracting Oriented Points

7.1.1 Grouping Edges into Contours

We first obtain the edge map by using edge detector of [146]; we obtain similar results by extracting edges using the Canny detector, indicating that our approach is robust to the quality of edge detection. Next, we remove spurious edges using a grouping technique related to [143], which will be described next. Then we select image features by evenly sampling them (at every 20 pixels in most of our experiments) along edge contours, as shown in Figure 7.1.

We obtain these contours by grouping pixels that form long and smooth curves. First, we group the edges into connected components by joining those pairs of edge pixels (i, j) that are both sufficiently close (within 5 pixels) and satisfy smoothness constraints based on collinearity and proximity, based on the assumption that non-smooth contours are probably due to noise and do not correspond to true object contours. Thus two pixels are connected iff they are within 5 pixels and meet the following conditions:

1. $(\widehat{\mathbf{n}_i, \mathbf{n}_j}) < \pi/6$, where $\mathbf{n}_i$ is the normal of edge i

2. $|\pi/2 - (\widehat{\mathbf{n}_i, \overrightarrow{(i,j)}})| < \pi/6$ and $|\pi/2 - \widehat{\mathbf{n}_j, \overrightarrow{(j,i)}})| < \pi/6$, where $\overrightarrow{(i,j)}$ is the vector from edge pixel i to j

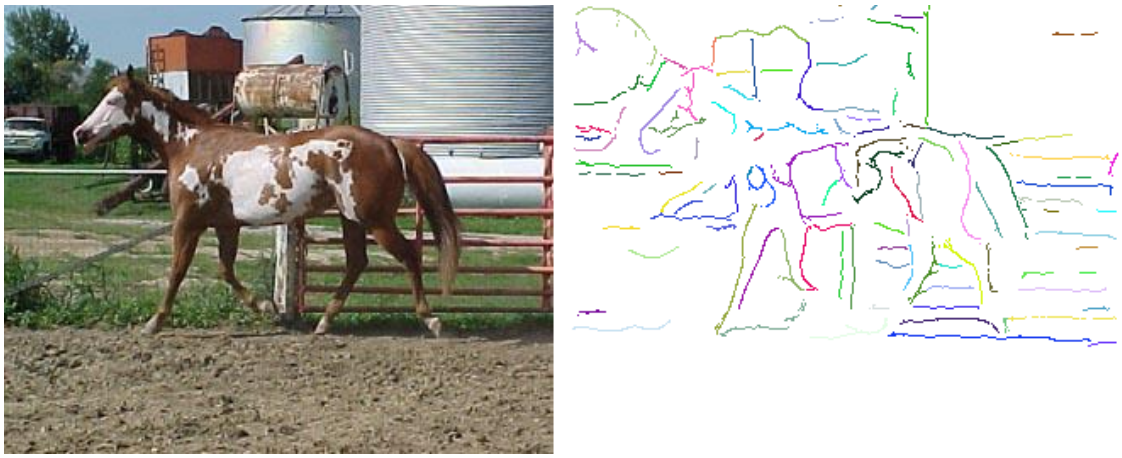

Figure 7.1: Original image (left) and extracted contours (right).Right: different colors correspond to different connected components.

For each connected component we form its weighted adjacency matrix $\mathbf{A}$ such that $\mathbf{A}_{ij}$ is positive if edge pixels (i, j) are connected and 0 otherwise. The value of $\mathbf{A}_{ij}$ increases with the smoothness between (i, j):

$$\mathbf{A}_{ij} = \frac{13.5 - (\widehat{\mathbf{n}_i, \mathbf{n}_j})^2 + |\pi/2 - (\mathbf{n}_i, \overrightarrow{(i,j)})|^2 + |\pi/2 - (\mathbf{n}_j, \overrightarrow{(j,i)})|^2}{2 * \sigma^2}, \tag{7.1}$$

where $\sigma = \pi/18$.

The principal eigenvalue λ of $\mathbf{A}$ describes the average smoothness of this component. We keep only those components that are large enough (number of pixels $> 0.01 N_{max}$) and smooth enough ($\lambda > 0.5\lambda_{max}$), where N_{max} is the size of the largest component and λ_{max} is the largest eigenvalue from the image. This step can be coded efficiently in the following way. Instead of performing it separately for each connected component one could form a sparse matrix $\mathbf{A}$ for the entire image as if all edges are connected. Of course, this matrix can be made block diagonal if the rows and columns are permuted appropriately, with each block corresponding to a connected component, but this step is not necessary as we will see next. Once this large matrix is formed we can use the power method to find its first eigenvector starting with a vector containing all ones. This vector should converge to one of those components, but in practice it will in fact contain all eigenvectors for all components, since none of its elements will actually become absolute zeros. One way to recover the eigenvalues for all those blocks is by dividing each element of $\mathbf{Av}$ by the corresponding element of $\mathbf{v}$, where $\mathbf{v}$ is the leading eigenvector of $\mathbf{A}$ as obtained by the power method starting from the vector of all ones. The resulting vector will contain the eigenvalues corresponding to the block to which that particular element belongs to. The eigenvector can also be used to recover each connected component. The first component can be obtained by keeping only

the elements of $\mathbf{v}$ that are not too close to zero. Those elements can be zeroed out next, the remaining values re-normalized and the procedure repeated. In this way, which is very easy to code, one can recover all connected components and their corresponding eigenvalues.

Once we have an edge map of the image, getting the contours is very efficient, usually taking less than a second per image, in Matlab on a 2GHz PC. The main use of these contours is to eliminate spurious edges more reliably than by simply thresholding the output of the edge detector. It also provides a better estimate of the normal at each edge pixel by considering only neighboring edges belonging to the same contour (Figure 7.1).

7.1.2 Object Category Specific Contours

The contours extracted as described earlier are generic and they do not take in consideration the specific class of which type of object the contours belong to. For object recognition it can be very useful to be able to filter out the contours that belong to distractors while keeping most contours belonging to the object category that we want to find. In an object recognition setting, if we could filter out most edges from negative images while keeping most edges from the positive ones, we would expect the recognition performance to increase significantly, a fact that is verified in our experiments. In our recent collaboration with Julien Mairal [144] we present an algorithm that is able to achieve this, based on a general patch-based discriminative framework.

Considering the edge detection task as a pixelwise classification problem, we have applied the patch-based discriminative framework to learn a local appearance model of patches centered on an edge pixel against patches centered on a non-edge pixel and therefore have a confidence value for each pixel of being on an edge or not. Then, once we have trained an edge detector, we propose to use the generic method for class-specific edge detection. Suppose we have N object classes. After finding the edges of all the images, we can then learn N classifiers to discriminate patches centered on a pixel located on an edge from an object of class A against patches centered on a pixel located on edges from the other classes. If this method should not be enough for recognizing an object by itself, we show that this local analysis can be used as a preprocessing of our global contour-based recognition framework described later in this Chapter.

7.2 Second-order Relationships: Pairwise Geometry Between Oriented Points

In the previous chapters we explained why the accidental nature of strong pairwise scores between incorrect assignments is the main reason why spectral matching works efficiently. We also discussed how we can learn these scores both in a supervised and unsupervised

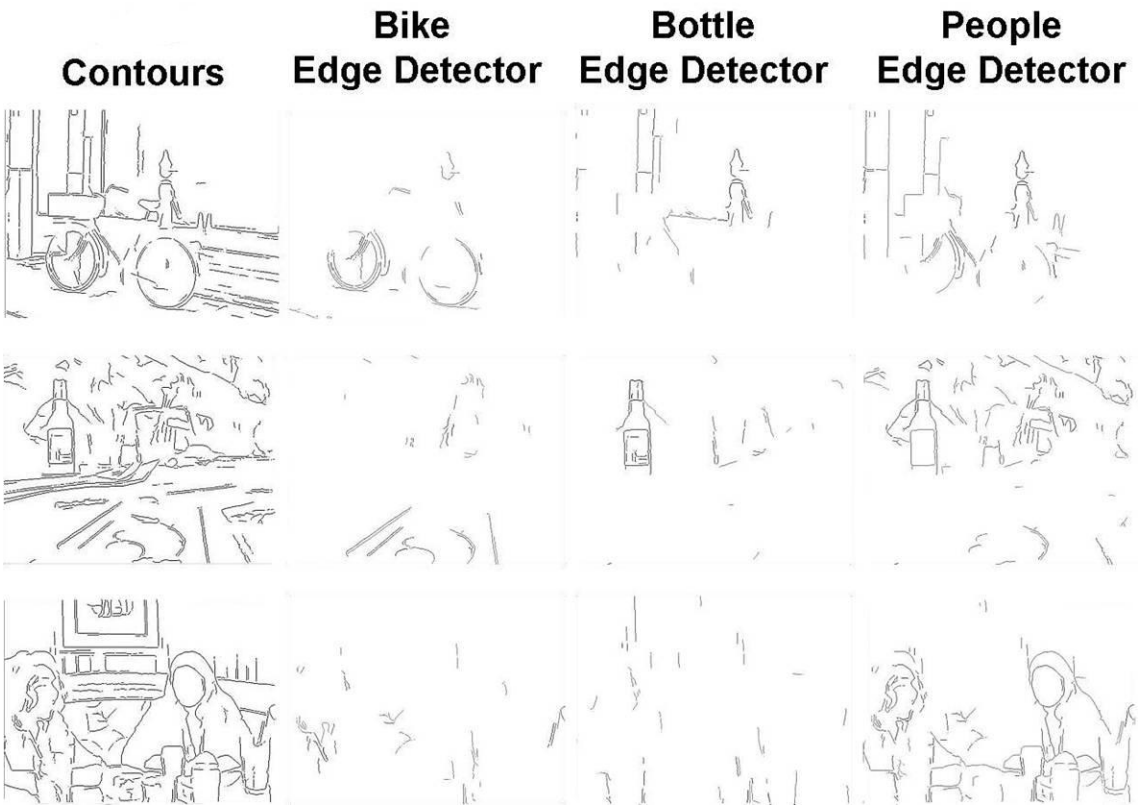

Figure 7.2: For each image several object category specific edge detectors are applied. We notice how the edge detectors are able to keep the correct edges while filtering out most edges belonging to the background/distractors. This filtering step improves significantly the object recognition performance of the shape classifier. Figure reprinted from [144].

manner. However, before learning and using this algorithm for matching, one has to first establish what pieces of information are going to be included in those pairwise scores. In this Chapter we show that geometry is one of the strongest pieces of information that should be used because it naturally establishes strong correlations between pairs of correct assignments vs. wrong ones. This notion is known in the literature as *accidental alignments*: incorrect pairs of assignments are very unlikely to form strong pairwise scores, of course, if those scores are designed in a meaningful way. In all of our experiments on recognition we used geometry as the main component of the pairwise scores. Before we describe in detail those pairwise scores we first show how to extract the unary features from pieces of contours and then what are the elements of their pair-wise geometric relationships.

We now explain in greater detail the type of features used and their pairwise relationships. Each unary feature is a point and its associated normal (with no absolute location).

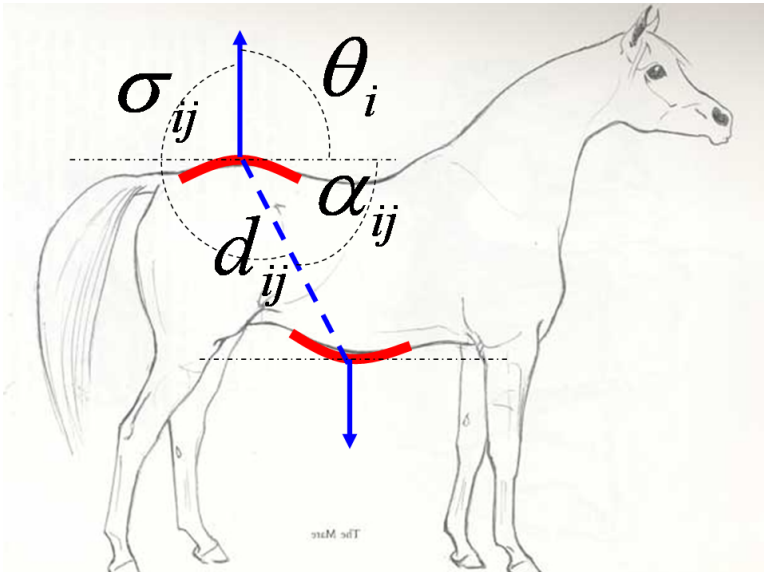

Figure 7.3: Parameters that capture the pair-wise geometric relationships between object parts.

For a pair of model parts (i, j) we capture their translation invariant relationship in the vector $\mathbf{e}_{ij} = \{\theta_i, \theta_j, \sigma_{ij}, \sigma_{ji}, \alpha_{ij}, \beta_{ij}, d_{ij}\}$, where d_{ij} represents the distance between them, β_{ij} is the angle between their normals and the rest are angles described in Figure 7.3.

The same type of information is extracted from the second image, each image feature corresponding to a point sampled from some boundary fragment extracted from that image (see Section 7.1.1). We consider a similar pairwise relationship $\mathbf{e}_{ab}$ for the pair (a, b) of image features that were matched to (i, j). Then we express the pairwise geometric deformation vector as $\mathbf{g}_{ij}(a, b) = [1, \epsilon_1^2, ..., \epsilon_7^2]$, where $\epsilon = \mathbf{e}_{ij} - \mathbf{e}_{ab}$. Notice that the geometric parameters $\mathbf{e}_{ij}$ form an overcomplete set of values, some highly dependent on each other. Considering all of them becomes very useful for geometric matching and recognition because it will make the pairwise score more robust to changes in the individual elements of $\mathbf{g}_{ij}(a, b)$.

There are different ways of forming pairwise scores once we have formed the pairwise deformation vector $\mathbf{g}_{ij}(a, b)$. The ones we prefer to use in this thesis are:

$$M_{ia;jb} = \exp\left(-\mathbf{w}^T \mathbf{g}_{ij}(a, b)\right), \tag{7.2}$$

or

$$M_{ia;jb} = \frac{1}{1 + \exp\left(-\mathbf{w}^T \mathbf{g}_{ij}(a, b)\right)}. \tag{7.3}$$

The second score function is the one we used first, historically, inspired from the interaction potentials from the graphical models literature. The sigmoid acts mostly like a thresholding function and it is not particularly sensitive to the actual deformations. Later,

we noticed that the exponential formulation, being more sensitive to such deformations, gives better matching results. Most of the results in this chapter use the sigmoid form, while the results on matching use the exponential one. If in matching the exponential pairwise score is significantly better, for recognition the difference is not significant. That is because the sigmoid score still gives matches that are nearby the ground truth ones, so, in fact, they could be considered correct if the matching rate would allow for small alignment errors.

7.2.1 Pairwise Geometry vs. Local Features

As a side-experiment we also investigate the performance of the pairwise geometric classifier $G_{ij}(x_i, x_j)$ against that of a pairwise classifier that only uses local features such as SIFT [134], shape context [13] and textons [146] extracted at the same locations. As before, positive and negative vectors for pairs of correspondences are collected — this time using changes in local feature descriptors rather than geometry. More precisely the pairwise appearance changes are represented by $\mathbf{f}_{ij}(x_i, x_j) = [1, ||s_i - s_{x_i}||^2, ||s_j - s_{x_j}||^2, ||c_i - c_{x_i}||^2, ||c_j - c_{x_j}||^2, ||t_i - t_{x_i}||^2, ||t_j - t_{x_j}||^2]$. Here s_i,c_i and t_i represent the SIFT, Shape Context and the 32 texton histogram descriptors, respectively, at the location of feature i. Using the logistic regression algorithm we train appearance-only classifiers as well as classifiers combined with geometry (using the vectors $[\mathbf{f}_{ij}(x_i, x_j), \mathbf{g}_{ij}(x_i, x_j)]$). As before, note that these classifiers are independent of the object category. Their only task is to classify a specific pair of correspondences $((i, x_i), (j, x_j))$ as correct/wrong.

Table 7.1 presents comparisons among the performances of the three types of classifiers (geometry, appearance and combined geometry+appearance). For each database we randomly split the pairs of correspondences in 6000 training (2000 positive and 4000 negative) and 18000 test (6000 positive and 10000 negative) vectors. An interesting observation is that, in each case the geometry-only classifier outperforms the one based on local features by at least 10%. This could be attributed to the fact that, while object shape remains relatively stable within the same category, object appearance varies significantly. Moreover, combining appearance and geometry only improves performance marginally $(1 - 1.5\%)$ over the geometry-only classifier. These results validate our approach of focusing on second-order relationships between simple shape features rather than on richer individual local features.

We also used the pairwise scores collected for this experiment in order to learn independent loose thresholds for each element in $\mathbf{g}_{ij}(x_i, x_j)$ that let most pairwise scores for pairs of correct assignments pass, while removing (setting to zero) most scores of that include at least one incorrect assignment. This helps tremendously in speeding up the building of the matrix $\mathbf{M}$, as previously discussed in the chapter on learning. The thresholding values

Table 7.1: The classification rates (at equal error rate) of the geometry-based pairwise classifier vs. the local feature classifier, for three different databases, averaged over 10 runs. Note that these are not category recognition results, but the results of classifiers on pairs of assignments.

Database	local only	geometry only	combined
Caltech-5	86.73%	97.42%	98.10%
INRIA horses	80.89%	95.33%	96.40%
GRAZ-02	83.30%	93.24%	94.74%

was decided by examining the histograms of these pairwise deformations in $\mathbf{g}_{ij}(x_i, x_j)$ for correct vs. incorrect pairs of assignments.

7.3 The Category Shape Model

The category shape model (Section 7.3.1) is represented as a graph of interconnected parts (nodes) whose geometric interactions are modeled using pairwise potentials inspired by Conditional Random Fields [110, 106]. The nodes in this graph are fully interconnected (they form a clique) with a single exception: there is no link between two parts that have not occurred together in the training images. These model parts have a very simple representation: they consist of sparse, abstract points together with their associated normals. Of course we could add local information in addition to their normals, but our objective is to assess the power of the geometric relationships between these simple features. We represent the pairwise relationships by an over-complete set of parameters (Section 7.3.1). The parts as well as their geometric relationships are learned (Section 7.4) from actual boundary fragments extracted from training images as described in Section 7.1.1.

Our model is a graph whose edges are abstract pairwise geometric relationships. It is a compact representation of a category shape, achieved by sharing generic geometric configurations common to most objects from a category and also by integrating specific configurations that capture different aspects or poses (Figure 7.4). In Section 7.4 we discuss and exemplify (Figures 7.6 and 7.8) the ability of our learning stage to reuse common configurations and to integrate new ones.

In order to explain in detail the pairwise geometric relationships we start with the *localization* problem, that is matching the object parts to the image features.

7.3.1 Object Localization

We define the object localization problem as finding which feature in the image best matches each model part. We formulate it as a quadratic assignment problem (QAP), due to our

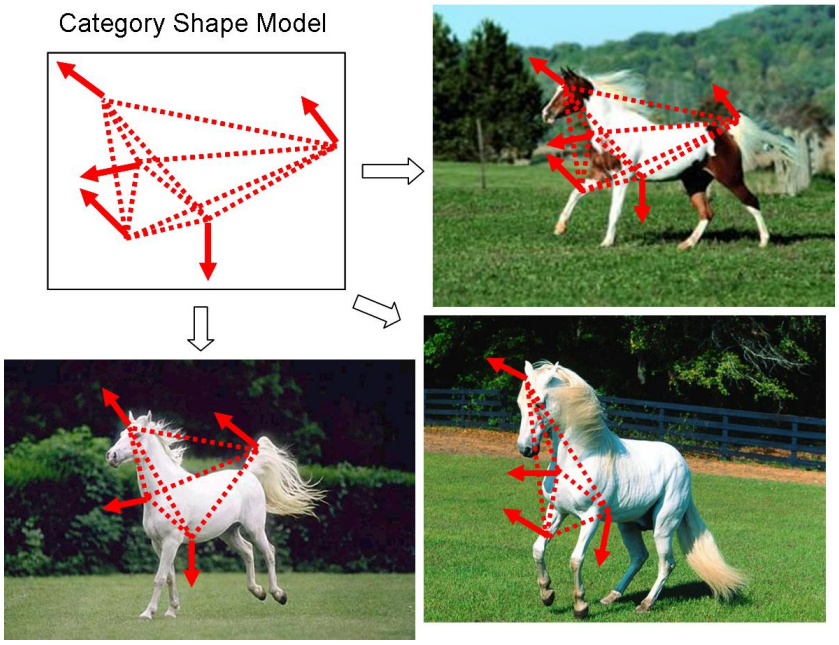

Figure 7.4: The model is a graph whose edges are abstract pairwise geometric relationships. It integrates generic configurations common to most objects from a category as well as more specific configurations that capture different poses and aspects.

use of second-order relationships. The matching score E is written as:

$$E_x = \sum_{ia;jb} x_{ia} x_{ib} G_{ia;jb}. \tag{7.4}$$

Here $\mathbf{x}$ is an indicator vector with an entry for each pair (i, a) such that $x_{ia} = 1$ if model part i is matched to image feature a and 0 otherwise. With a slight abuse of notation we consider ia to be a unique index for the pair (i, a). We also enforce the mapping constraints that one model part can match only one model feature and vice versa: $\sum_i x_{ia} = 1$ and $\sum_a x_{ia} = 1$.

The *pairwise potential* $G_{ia;jb}$ (terminology borrowed from graphical models) reflects how well the parts i and j preserve their geometric relationship when being matched to features a, b in the image. Similar to previous approaches taken in the context of CRFs [106] we model these potentials using the exponential function:

$$G_{ia;jb} = exp(-\mathbf{w}^T \mathbf{g}_{ij}(a, b)). \tag{7.5}$$

Here $\mathbf{g}_{ij}(a, b)$ is a vector describing the geometric deformations between the parts (i, j) and their matched features (a, b) as described in Chapter 7.2. The same type of information is extracted from input images, each image feature corresponding to a point sampled from

some boundary fragment extracted from that image as explained in Chapter 7.2. We consider a similar pairwise relationship $\mathbf{e}_{ab}$ for the pair (a, b) of image features that were matched to (i, j). As shown previously, we express the pairwise geometric deformation vector as $\mathbf{g}_{ij}(a, b) = [1, \epsilon_1^2, ..., \epsilon_7^2]$, where $\epsilon = \mathbf{e}_{ij} - \mathbf{e}_{ab}$.

In order to localize the object in the image, we want to find the assignment $\mathbf{x}^*$ that is likely to maximize the matching score E (written in matrix notation by setting $\mathbf{G}(ia; jb) = G_{ia;jb}$):

$$\mathbf{x}^* = argmax(\mathbf{x}^T \mathbf{G} \mathbf{x}) \tag{7.6}$$

For one-to-one constraints (each model part can match only one image feature and vice-versa) this combinatorial optimization problem is known as the quadratic assignment problem (QAP). For many-to-one constraints it is also known in the graphical models literature as MAP inference for pairwise Markov networks. In general, both problems are intractable. We enforce the one-to-one constraints and use the spectral matching algorithm [117] described in Chapter 2, which, as discussed before, is very efficient in practice, giving good approximate solutions and being able to handle hundreds of fully connected parts in a few seconds on a 2GHz desktop computer.

7.3.2 Discriminative Object Recognition

The previous section describes how we localize the object by efficiently solving a MAP inference problem. However, this does not solve the recognition problem since the matching algorithm will return an assignment even if the input image does not contain the desired object. In order to decide whether the object is present at the location $\mathbf{x}^*$ specified by our localization step, we model the posterior $P(C|\mathbf{x}^*, y)$ (where the class $C = 1$ if the object is present at location $\mathbf{x}^*$ and $C = 0$ otherwise). Modeling the true posterior would require modeling the likelihood of the data y given the background category (basically, the rest of the world), which is infeasible in practice. Instead, we take a discriminative approach and attempt to model this posterior directly, as described below.

We consider that $P(C|\mathbf{x}^*, y)$ should be a function of several factors. First, it should depend on the quality of the match (localization) as given by the pairwise potentials $G_{ij}(x_i, x_j; y)$ for the optimal solution $\mathbf{x}^*$. Second, it should depend only on those model parts that indeed belong to the category of interest and are discriminative against the negative class. It is not obvious which are those parts, since we learn the model in a semi-supervised fashion . For this reason we introduce the *relevance* parameter r_i for each part i (Section 7.4 explains how this is learned), which has a high value if part i is discriminative against the background, and low value otherwise. We approximate the posterior using the

following logistic classifier:

$$S(\mathbf{G}_o, \mathbf{r}) = \frac{1}{1 + \exp(-q_0 - q_1 \sigma(\mathbf{r})^T \mathbf{G_o} \sigma(\mathbf{r}))}. \tag{7.7}$$

The matrix $\mathbf{G_o}(i,j) = G_{ij}(x_i^*, x_j^*; y)$ contains all the pairwise potentials for the optimal localization $\mathbf{x}^*$. In Eqn. (7.7), each pairwise potential G_{ij} is weighed by the product $\sigma(r_i)\sigma(r_j)$, where $\sigma(r_i) = 1/(1+e^{-r_i})$.

The primary reason for passing the relevance parameters through a sigmoid function is that letting the relevances be unconstrained real-valued parameters would not help us conclusively establish which parts indeed belong to the category and which ones do not. What we really want is a binary relevance variable that is 1 if the model part belongs to the category model and 0 otherwise. Having a binary variable would allow us to consider only those parts that truly belong to the object category and discard the irrelevant clutter. Our intuition is that if we squash the unconstrained relevances r_i we effectively turn them into soft binary variables, and during training we force them to be either *relevant* ($\sigma(r_i) \approx 1$) or *irrelevant* ($\sigma(r_i) \approx 0$). This is exactly what happens in practice. The squashed relevances of most parts go either to 1 or 0, thus making it possible to remove the irrelevant ones ($\sigma(r_i) \approx 0$) without affecting the approximate posterior $S(\mathbf{G_o}, \mathbf{r})$. An additional benefit of squashing the relevance parameters is that it damps the effect of very large or very small negative values of r_i, reducing overfitting without the need for an explicit regularization term.

The higher the relevances r_i and r_j, the more $\mathbf{G_o}(i,j)$ contributes to the posterior. It is important to note that the relevance of one part is considered with respect to its pairwise relationships with all other parts together with their relevances. Therefore, parts are evaluated based on how well they work together as a team, rather than individually. It is important to understand that we interpret the logistic classifier $S(\mathbf{G_o}, \mathbf{r})$ not as the true posterior, which is impractical to compute, but rather as a distance function that is specifically tuned for classification.

7.4 Learning the Shape Model

The model parameters to be learned consist of the pairwise geometric relationships $\mathbf{e}_{ij}$ between all pairs of parts, the sensitivity to deformations $\mathbf{w}$ (which defines the pairwise potentials), the relevance parameters $\mathbf{r}$ and q_0, q_1 (which define the classification function S). The learning steps are summarized in Figure 7.5 and detailed below.

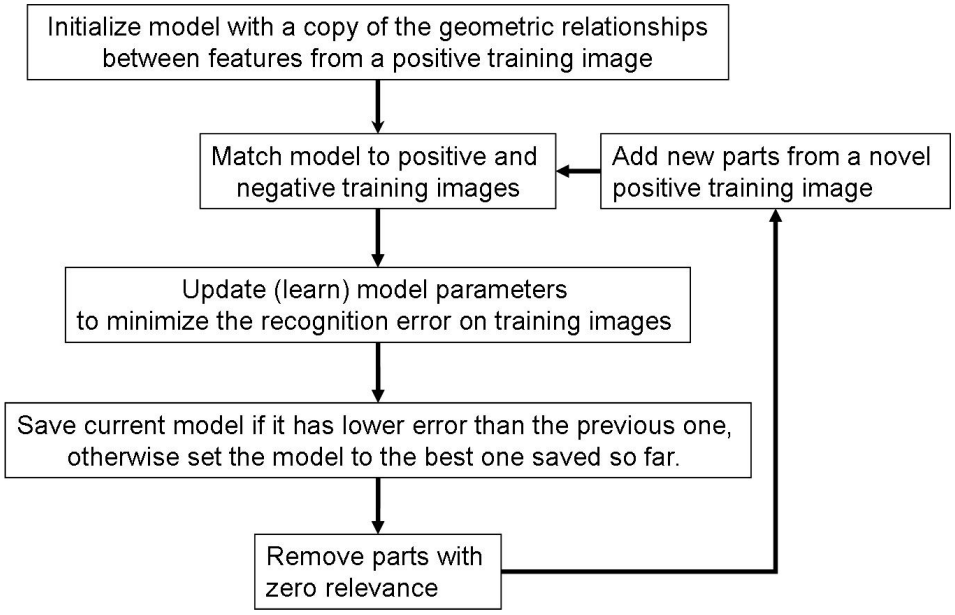

Figure 7.5: Learning algorithm overview.

7.4.1 Initialization

We initialize the pairwise geometric parameters ($\mathbf{e}_{ij}$) for each pair of model parts by simply copying them from a positive training image. Thus, our initial model will have as many parts as that training image used and the same pairwise relationships. We initialize the rest of the parameters to a set of default values. For each part i we set the default value of its r_i to 0 ($\sigma(r_i) = 0.5$). The default parameters of the pairwise potentials ($\mathbf{w}$) are learned using the unsupervised learning method proposed in Chapter 5.

7.4.2 Updating the Parameters

Starting from the previous values, we update the parameters by minimizing the familiar sum-of-squares error function (typically used for training neural networks) using sequential gradient descent. The objective function is differentiable with respect to r, q_0 and q_1 since they do not affect the optimum x^* (for the other parameters we differentiate assuming fixed x*):

$$J = \sum_{n=1}^{N} b_n (S(\mathbf{G_o}^{(n)}, r) - t^{(n)})^2. \tag{7.8}$$

Figure 7.6: The model integrates geometric configurations belonging to different aspects (view-points) within the same category. Training images (left) and the boundary fragments containing the relevant parts learned from different view-points and integrated in the same model (right). Note that the algorithm automatically determines the features that belong to the object rather than the background.

Here $t^{(n)}$ denotes the ground truth for the n^{th} image (1 if the object is present in the image, 0 otherwise). The weights b_n are fixed to m_N/m_P if $t^{(n)} = 1$ and 0 otherwise, where m_N and m_P are the number of negative and positive images, respectively. These weights balance the relative contributions to the error function between positive and negative examples. The matrix $\mathbf{G_o}^{(n)}$ contains the pairwise potentials for the optimal localization for the n^{th} image.

We update the parameters using sequential gradient descent, looping over all of the training images for a fixed number of iterations; in practice this reliably leads to convergence. The learning update for any given model parameter λ for the n^{th} example has the general form of:

$$\lambda \leftarrow \lambda - \rho b_n (S(\mathbf{G_o}^{(n)}, \mathbf{r}) - t^{(n)}) \frac{\partial S(\mathbf{G_o}^{(n)}, \mathbf{r})}{\partial \lambda}, \tag{7.9}$$

where ρ denotes the learning rate. Using this general rule we can easily write the update rules for all of the model parameters. The pairwise potentials ($\mathbf{G_o}$) do not depend on the parameters $\mathbf{r}, q_0, q_1$. It follows that the optimal labeling $\mathbf{x}^*$ of the localization problem remains constant if we only update $\mathbf{r}, q_0, q_1$, which we do in practice.

7.4.3 Removing Irrelevant Parts

As mentioned earlier, the relevance values $\sigma(r_i)$ for each part i tend to converge either towards 1 or 0. This is due to the fact that the derivative of J with respect to the free relevance parameters r_i is zero only when the output $S(\mathbf{G_o}^{(n)}, \mathbf{r})$ is either 0 or 1, or when

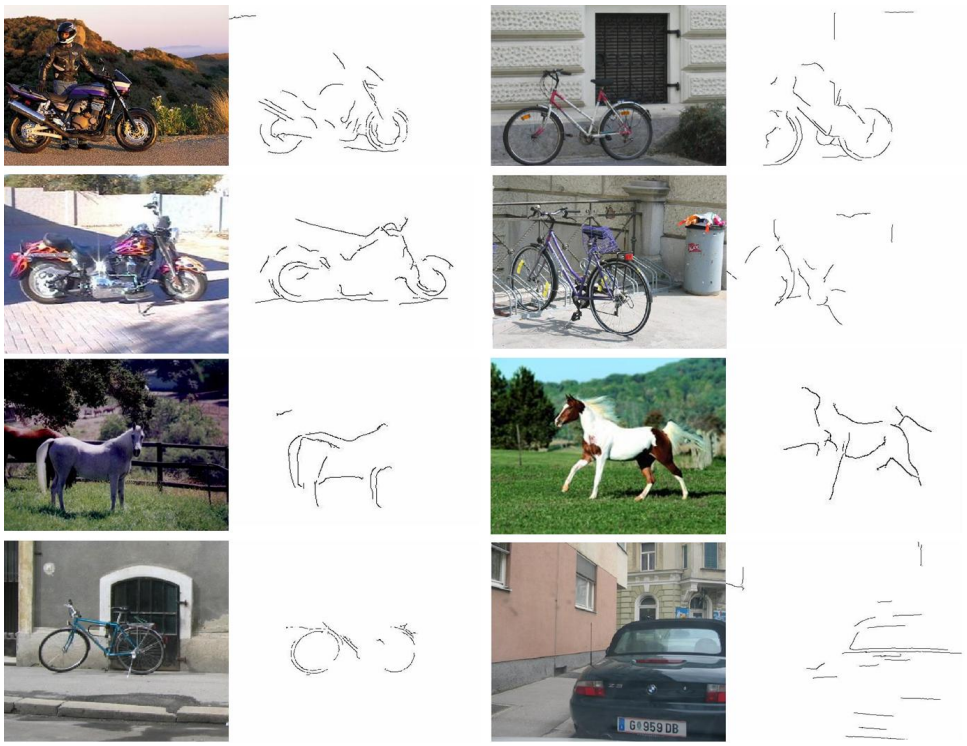

Figure 7.7: Training images (Left) and the contours on which the relevant features were found during training (Right).

the relevance $\sigma(r_i)$ is either 0 or 1, the latter being much easier to achieve. This is the key factor that allows us to discard irrelevant parts without significantly affecting the output $S(\mathbf{G_o}^{(n)}, \mathbf{r})$. Therefore, all parts with $\sigma(r_i) \approx 0$ are discarded. In our experiments we observe that the relevant features typically belong to the true object of interest (Figure 7.7).

7.4.4 Adding New Parts

We proceed by merging the current model with a newly-selected training image (randomly selected from the ones on which the recognition output is not close enough to 1): we first localize the current model in the new image, thus finding the subset of features in the image that shares similar pairwise geometric relationships with the current model. Next, we add to the model new parts corresponding to all of the image features that fail to match the current model parts. As before, we initialize all the corresponding parameters involving newly-added parts by copying the geometric relationships between the corresponding features and using default values for the rest. At this stage, different view-points or shapes of our category can be merged (Figure 7.6). The geometric configurations shared by different aspects are already in the model (that is why we first perform matching) and only the novel configurations are added (from the parts that did not match). After adding new parts we

return to the stage of updating the parameters (Figure 7.5). We continue this loop until we are satisfied with the error rate. The approach of adding training images one-by-one is related to incremental semi-supervised learning methods [172]. In our case, we later discard the information that is not useful for recognition (the parts with zero relevance). Removing and adding parts enables our model to grow or shrink dynamically, as needed for the recognition task.

7.5 Experimental Validation

7.5.1 Category Recognition without Edge Filtering

Tables 7.2 and 7.3 compare the performance of our method with Winn [225] on the Pascal challenge training dataset[1] (587 images). This is an interesting experiment because our method only employs geometry and ignors local appearance; in contrast, Winn focus on local texture information, while ignoring the geometry. We follow the same experimental setup, splitting the dataset randomly in two equal training and testing sets. The first set of experiments uses the provided bounding box (also used by Winn). We outperform the texture-based classifier by more than 10%, confirming our intuition that shape is a stronger cue than local appearance for these types of object categories. Surprisingly, bikes and motorcycles are not confused as much as one might expect despite their similar shapes. In the second set of experiments, we do not use the bounding boxes,[2] neither for training nor testing, in order to demonstrate that our method's ability to learn in a weakly-supervised setting. The performance drops by approximately 5%, which is significant, but relatively low considering that the objects of interest in this experiment frequently occupy only a small fraction of the image area. A more serious challenge is that several positive images for one class contain objects from other categories (there are people present in some of the motorcycle and car images). In our reported resuls, an image from the "motorcycle" class containing both a motorbike and a person that was classified as "person" would be treated as an error.

As mentioned earlier, the models that we learn are compact representations of the relevant features present in the positive training set. The algorithm discovers relevant parts that, in our experiments, generally belong to the true object of interest despite significant background clutter. An interesting and useful feature of our method is its ability to integrate different view-points, aspects or shapes within the same category (Figure 7.6). This happens automatically, as new parts are added from positive images.

[1] http://www.pascal-network.org/challenges/VOC/voc2005/. We ignore the gray-level UIUC car images.

[2] For the few images in which the object was too small, we select a bounding box of 4 times the area of the original bounding box.

Table 7.2: Confusion Matrix on Pascal Dataset.

Category	Bikes	Cars	Motorbikes	People
Bikes	80.7%	0%	7%	12.3%
Cars	5.7%	88.6%	5.7%	0%
Motorbikes	4.7%	0%	95.3%	0%
People	7.1%	0%	0%	92.9%

Table 7.3: Average multiclass recognition rates on Pascal.

Algorithm	Ours (bbox)	Ours (no bbox)	Winn (bbox)
Pascal Dataset	89.4%	84.8%	76.9%

The computational cost of classifying a single image does not depend on the number of training images: the model is a compact representation of the relevant features in the training images, usually containing between 40 to 100 parts. It is important to note that the size of the model is not fixed manually; it is an automatic outcome from the learning stage.

We also compare our method with Opelt [155] on the GRAZ-01 and GRAZ-02 datasets (Table 7.4). We run the experiments on the same training and test sets on full images (no bounding boxes were used). Opelt focus mainly on local appearance and select descriptors based on their individual performance and combine them using AdaBoost. This is in contrast with our approach, since our features by themselves have no discriminative power. It is their combined configuration that makes them together discriminative.

We further test our algorithm on the INRIA (168 images) and Weizmann Horse (328 images) [22] databases, using for the negative class the GRAZ-02 background images. We do not use objects masks (nor bounding boxes) and we randomly split the images in equal training and testing sets. The INRIA horse databases is particularly difficult due to significant changes in scale, pose and multiple horses present in the same image.

7.5.2 Category Recognition using Edge Filtering

Our proposed approach of combining local appearance with shape is to first learn a class specific edge detector on pieces of contours using the discriminative classifier of Mairal et al. [144]. Next this class specific edge detector is used to filter out the irrelevant edges while learning the shape-based classifier as described later. Similarly, at testing time, the shape-based algorithm is applied to the contours that survive after filtering them with our class

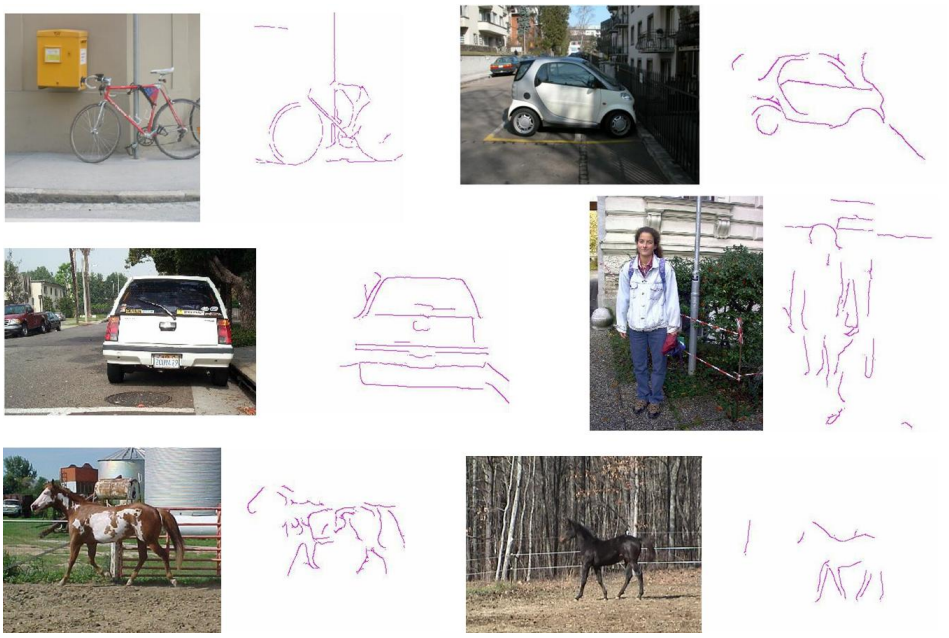

Figure 7.8: Training Images (Left) and the contours on which the relevant features were found (Right).

Table 7.4: Category recognition rates (at equal error rate) on GRAZ Dataset (People and Bikes), Shotton and INRIA horses datasets. Bounding boxes (masks) are not used.

Dataset	Ours	Opelt (1)	Opelt (2)
People (GRAZ I)	82.0%	76.5%	56.5%
Bikes (GRAZ I)	84.0%	78.0%	83.5%
People (GRAZ II)	86.0%	70.0%	74.1%
Bikes (GRAZ II)	92.0%	76.4%	74.0%
Horses (Shotton)	92.02%	-	-
Horses (INRIA)	87.14%	-	-

Table 7.5: Confusion Matrix on Pascal Dataset.

Category	Bikes	Cars	Motorbikes	People
Bikes	93.0%	3.5%	1.7%	1.8%
Cars	1.2%	97.7%	0%	1.1%
Motorbikes	1.9%	1.8%	96.3%	0%
People	0%	0%	0%	100%

dependent edge detector. The outputs of both the shape-based classifier and the real values given by our detector are later combined for the final recognition score. This framework provides a natural way of combining the lower-level, appearance based edge detection and contour filtering with the more higher level, shape-based approach. The algorithm can be summarized as follows:

1. Learn a class specific edge classifier. For each image, we apply a general edge detector, then obtain pieces of contours as described earlier. Next, we train class specific detectors on such contours belonging to the positive class vs. all other classes.

2. Use the class specific detectors to filter both training and testing images for the object category classification method based on shape described earlier in this Chapter.

In the first set of experiments with edge filtering, we use the edge detector on all the images from Pascal VOC05 and postprocess them to remove nonmeaningful edges using the countour finding method presented earlier. Then, we train our class-specific edge detector on the training set of each dataset, using the same training sets as before for VOC05. For each class a one-vs-all classifier is trained using the exact same parameters as for the edge detection, which allows us to give a confidence value for each edge as being part of a specific object type. In our recognition experiments we want to quantify the benefit of using our class-specific edge detector for object category recognition (Table 7.5, 7.6). Thus, we perform the same sets of experiments as before, on the same training and testing image sets from Pascal 2005 dataset, on a multiclass classification task. The use of edge filtering reduces the error rate more than 3-fold as compared with the shape alone classifier and more than 7-fold when compared to the appearance based method of Winn et al.

In the next set of experiments we tested the edge filtering combined with the shape classifier and the detector of [61] on the image classification task of the Pascal 2007 Challenge for 4 classes (bus, car, motorbike and TV). Note that we used the the output of the detector as a classifier (not as a detector!), by taking in consideration only its maximum output over the entire image. We followed exactly the protocol of the classification challenge, the same training and testing image sets and evaluation procedure. For this task we trained several

Table 7.6: Average multiclass classification rates on Pascal.

Algorithm	Ours (with filtering)	Ours (no filtering)	Winn (bbox)
Pascal Dataset	96.8%	89.4%	76.9%

Table 7.7: Average Precision on the Pascal 07 Classification Challenge test data for four classes.

Category	Detector only	Detector + Shape	Best of 2007 Challenge
Bus	57.9%	**63.6%**	60.3%
Car	78.9%	**80.6%**	78.0%
Motorbike	53.9%	59.6%	**64.0%**
TV	54.8%	**59.9%**	53.2%

shape models for each class, as described previously, by randomly picking different starting images for the initializations of our shape models. For each image we selected the box that returned the highest detection score using the class-specific detector. We then filtered the edges inside the box by using our class-specific edge filtering method as described earlier. For the remaining contours we applied different shape classifiers that, together with the output of the detector, were combined in a linear fashion using logistic regression. Thus, the final classifier for each class combined the output of the detector with the outputs of several shape classifiers that are selected during training in a greedy fashion as follows.

1. start by adding to the initial pool of classifiers the output of the detector
2. for each shape classifier not included in the current pool of classifiers learn a logistic classifier using its output and the outputs of the classifiers from the current pool and record the average precision performance on the training data
3. if the average precision is better than the best one obtained so far add this shape classifier to the pool of classifiers.
4. if no shape classifier improved the best average precision stop and return the logistic classifier using the outputs of the current pool of models. Otherwise return to step 2.

In Table 7.7 we show the results of the classification performance on the Pascal 07 test data. We notice that adding the shape classifiers to the output of the detector significantly improves the performance (sometimes by as much as 5%), while outperforming the best classifier in the challenge on 3 out or 4 classes. We mention that these are the only 4 classes

we experimented with. It is also interesting to mention that none of the methods competing in the official challenge used detection for classification, as we did in our experiments. One of the conclusions of the challenge was that most methods learned more the context of the objects than the actual appearance of the objects. This is in contrast to our work which is entirely based on the actual shape and appearance of the objects and uses no context information. This strongly suggests that combining our method with the best performing methods from the challenge could potentially further improve the classification performance.

Chapter 8

Relationships between Features based on Grouping

Grouping was recognized in computer vision early on as having the potential of improving both matching and recognition. We are interested in soft pairwise grouping mainly because it can potentially improve our recognition and matching approaches using pairwise relationships between features [117]. Grouping is one way of constraining the matching/recognition search space by considering only features that are likely to come from the same object. This low level process is essential for higher level tasks such as improving recognition and matching because it uses general, category independent information to prune the search space and guide the recognition process on the right path. In Figure 8.1 we show two examples that intuitively explain this idea. The images in the left column contain edges extracted from a scene. We notice that without grouping the objects are not easily distinguished (e.g. the bus, or the horse). However, after using color information for perceptual grouping we are able to retain only the edges that are likely to belong to the same object as the edge pointed out by the red circle (right column). Grouping could also bring a second benefit to the recognition process, because, without it, matching could be very expensive especially when the image contains a lot of background features/clutter. As Grimson [79] has shown, the complexity of the matching process when the search is constrained is reduced from an exponential to a low order polynomial. Therefore, it is important to be able to establish *a priori* which pairs of features are likely to be part of the same object, and discard all the other pairs. To summarize, perceptual grouping does not only improve the recognition performance but it also reduces the computational complexity.

In our work [117] we are most interested in grouping pairs of features, that is, establishing which pairs are likely to belong to the same object. This could be beneficial to our matching approach (also used in recognition) that is mostly based on pairwise relationships between features.

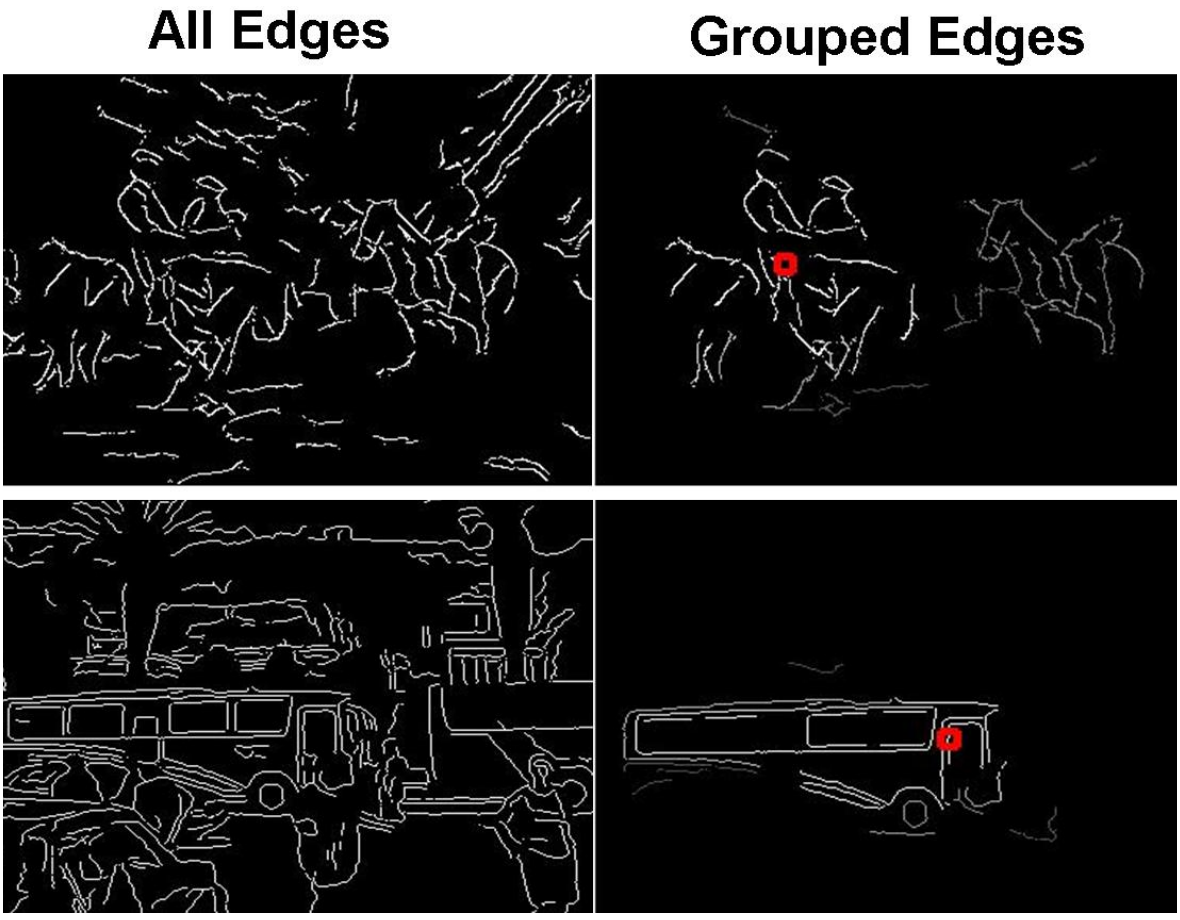

Figure 8.1: If grouping is not used it is very hard to distinguish the separate objects (left column). After grouping (right column) it is perceptually easier to distinguish them (the bus and the horse).

8.1 Soft Grouping

Grouping is the task of establishing which features in the image are likely to belong to the same object, based on cues that do not include the knowledge about the specific object or object category. In his pioneering book [145] Marr argued, based on medical human studies, that a vision system should be able to recover the 3D shape of objects without knowledge about the objects' class or about the scene. Humans are able to see a car parked in a room,

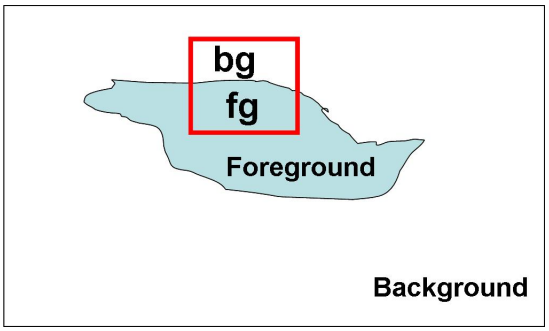

Figure 8.2: Automatically discovering the foreground mask.

even though that would be totally unexpected based on prior experience. Also, humans are able to perceive the shape of an abstract sculpture from a single image, even if they have never seen it before and have no clue about what it might represent. It seems clear that for humans both knowledge about the object class and the scene is not necessary for shape perception. But if one is able to perceive the 3D shape of the scene with respect to its own reference frame, then in most cases it would be relatively easy to figure out which features in the scene should belong together. In fact perceptual grouping most probably helps humans in the process of 3D shape recovery and not the other way around. We consider grouping as a process that happens entirely before the object recognition stage, and uses cues such as color, texture, shape, and perceptual principles that apply to most objects in general, regardless of their category or specific identity.

Unlike prior work in grouping [143], [178], [57], [91], [173], [150], [133], we do not make a hard decision about which features belong together. And that is for an important reason: it is sometimes impossible to divide features into their correct groups without the knowledge of the specific category (or the desired level of detail): for example, is the wheel of a car a separate object or is it part of the whole car? We believe that both situations can be true at the same time, depending on what we are looking for. If we are looking for whole cars, than the wheel is definitely a part of it. If we are looking just for wheels then (at least conceptually) it is not. While perceptual grouping alone should most of the time separate correctly most objects (the ones that are clearly at different depths, such as a flying plane from a close car), it sometimes does not have access to enough information to make the correct hard decisions. We immediately see why it is important to keep most the grouping information around and transmit it to the higher recognition processes (without making hard decisions, except for pruning the cases when the pairwise grouping relationship is extremely weak). Instead of being interested in forming exact feature groups based on perceptual information alone, we rather focus on the quality of

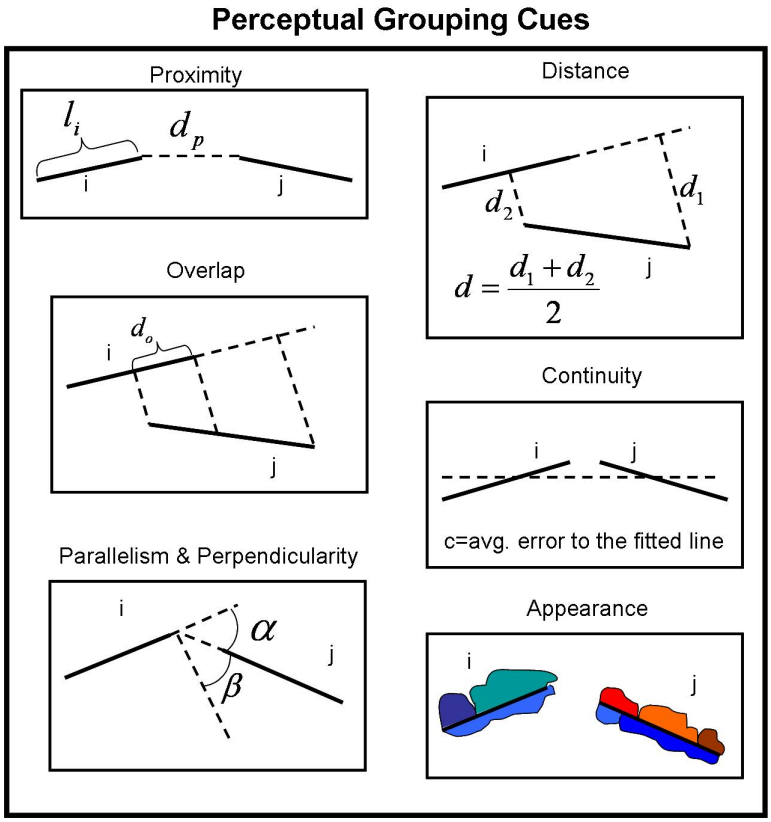

Figure 8.3: Geometric perceptual cues used for grouping pairs of line features.

pair-wise grouping relationships and how to integrate them into our recognition step. Since we use pair-wise relationships at both the grouping and the recognition levels, the two could be naturally integrated.

Our pair-wise grouping relationships are soft weights that should reflect the likelihood that the two features belong together (prior to recognition, that is before using any category specific knowledge). In this Chapter we focus only on pairwise grouping using color information: objects tend to have unique and relatively homogenous color distributions. We propose a novel and effective way of using color histograms for inferring pair-wise grouping relationships.

Color histograms are simple but powerful global statistics about objects' appearance that have been successfully used in some recognition applications. We present a novel algo-

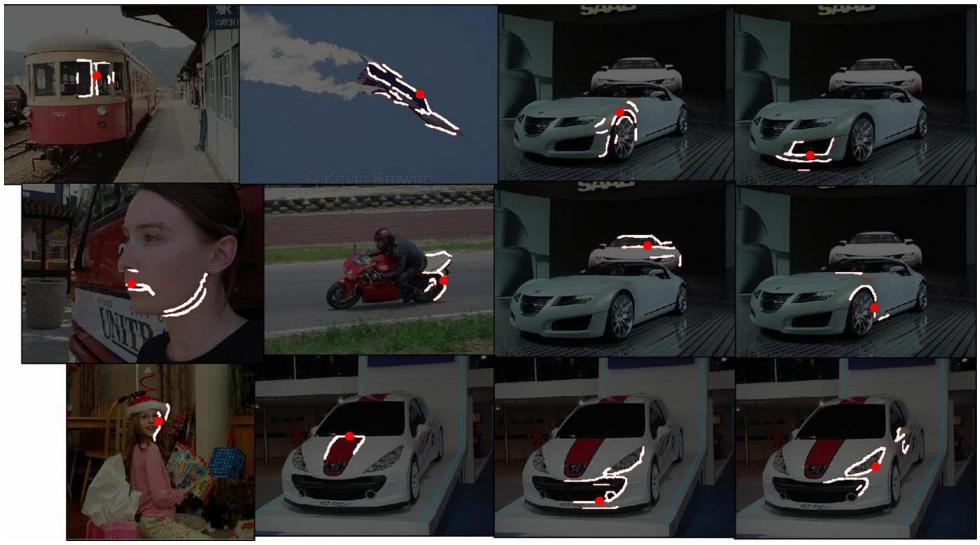

Figure 8.4: Pairwise grouping constraints based on geometry. The contours in white are the ones that establish a pairwise grouping relationship with the contour pointed out by the red circle.

rithm for automatically discovering soft object masks (based on their color distributions), without knowledge about their locations or shapes. This method becomes very useful for pairwise grouping of features, because two features that belong (in a soft way) to the same mask are more likely to belong to the same object then pairs of features that belong to different masks.

8.1.1 Pairwise Grouping using the Geometry of Line Segments

Before we discuss our approach to color grouping we first present a method for grouping using geometry. Geometric perceptual cues are particularly important because of their connection to important studies in human vision (mainly from the Gestalt school). In our experiments we found that geometric grouping is more local than grouping based on color, because faraway pixels are harder to group using only geometry, with no color or texture information. We believe that in a complete grouping system one should use as many cues as possible including both geometry and color.

Our main features for object recognition are pieces of contours extracted from the image. In the grouping stage we approximate these contours by fitted line segments. The geometric grouping cues we propose to use consist of specific relationships between pairs of such line segments (i, j): proximity, distance, overlap, continuity, parallelism and perpendicularity as shown in Figure 8.3. We also use some local appearance cues (which are different than the

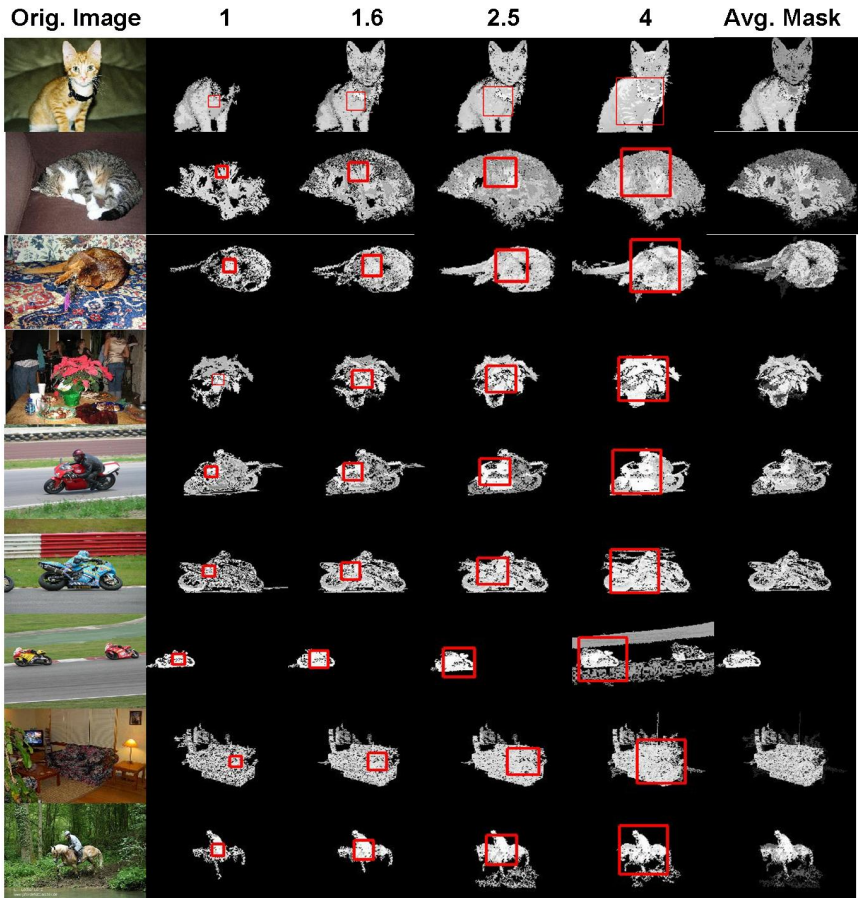

Figure 8.5: Object masks obtained from arbitrary bounding boxes, centered on the object of interest. Averaging over several fixed scales improves the result. The masks shown are computed from color likelihood histograms based on the bounding boxes (completely wrong) shown, which are centered on the object of interest. The interior of the boxes is considered to be the foreground, while their exterior is the background. The posterior color likelihood image is obtained, threshold-ed at 0.5 and the largest connected component touching the interior of the bounding box is retained. We notice the even when the bounding boxes have wrong locations and sizes the masks obtained are close to the ground truth. Different bounding boxes sizes (which are fixed for every image, starting at 8 pixels and growing at a rate of 1.6) are tried and the average mask is shown in the rightmost column.

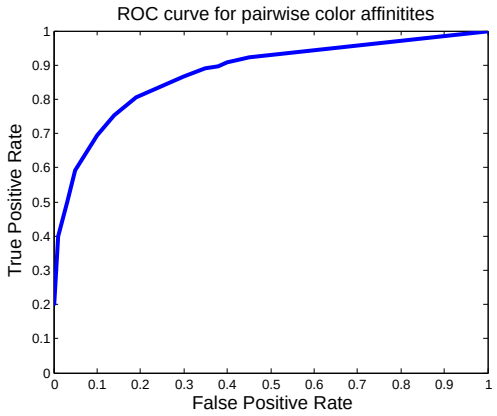

Figure 8.6: The ROC curve for the pairwise classifier on the MSR database, on about 5000 positive and 5000 negative pairs. Notice that we could eliminate almost all negative pairs ($FP = 0$) while keeping a significant part of the positive ones ($TP = 0.4$). For the purpose of recognition and matching we do not need to classify correctly all positive pairs. It is more important to remove most of the negative ones (very small false positive rate).

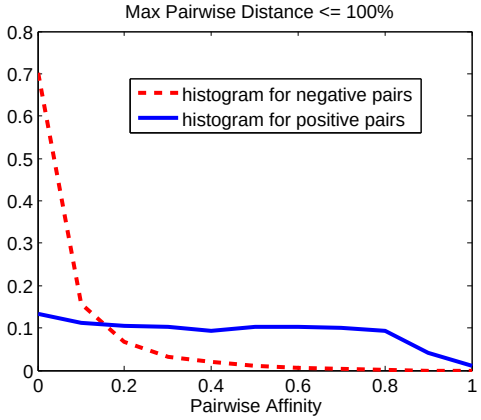

Figure 8.7: Histograms of pairwise affinities based on color. Notice that the negative pairs have very low affinities, close to zero most of the time. Thresholding at a value of 0.2 or higher would practically remove all negative pairs.

global color histograms used in the next section), which are computed over the super-pixels adjacent to the pair of lines, such as: difference between the mean colors of the super-pixels belonging to each line, as well as the differences in color histogram and color entropy. All these cues, both geometric and appearance based, form a *relation* vector $\mathbf{r}(\mathbf{i}, \mathbf{j})$ for any pair

Table 8.1: Perceptual cues used to describe the relationship between pairs of lines. Based on these cues we estimate the likelihood that pairs of lines belong to the same object or not

Cue	Description
Proximity	$\frac{d_p}{l_i+l_j}$
Distance	$\frac{d_i+d_j}{l_i+l_j}$
Overlap	$\frac{d_{oi}+d_{oj}}{l_i+l_j}$
Continuity	c
Parallelism	α
Perpendicularity	β
$Color_1$	difference in mean colors
$Color_2$	difference in color histograms
$Color_3$	difference in color entropies

of lines (i, j) whose elements are described in Table 8.1 (each row of the table corresponds to an element of $\mathbf{r}(\mathbf{i}, \mathbf{j})$). We have manually collected about 300 positive pairs of lines (lines that belong to the same object) and 1000 negative ones (pairs of lines which do not belong to the same object), and learned a binary classifier on the corresponding relation vectors $\mathbf{r}$, using the logistic regression version of Adaboost [37] with weak learners based on decision trees [72].

In Figure 8.4 we present some results. The contours shown in red belong to line segments that were classified as being part of the same object as the line segment pointed by the white circle. We notice that in general only lines that are relatively close to the white circle are positively classified. This is due to the fact that in general geometric perceptual grouping is a local process and is not able to link directly pairs of faraway lines. Such pairs could be ultimately connected indirectly thorough intermediate lines. Of course, these are only preliminary results, and more thorough experimentation is needed.

8.2 Unsupervised discovery of the foreground mask

The main focus of this Chapter is grouping based on color. The algorithm we will present for color grouping is more efficient and simpler than the one used for geometric grouping, and yet, it gives more global results, being able to group pixels (or line features) that are far away in the image. We show that color histograms alone are often powerful enough to segment the foreground vs. the background, without using any local intensity information or edges. Other work that used color histograms to separate the foreground from the background given minimal user input includes GrabCut [174] and Lazy Snapping [124]. That work required more extensive user input, such as markings on the object and outside

of it. In our case we want to discover, in a soft way, the object mask relative to a given point on the foreground, without any other information given. We want to be able to use the power of color histograms without knowing the true mask (or bounding box) of the foreground. But this seems almost impossible. How can we compute the color histogram of the foreground if we have no clue about the foreground size and rough shape?

For now, let us assume that we have the bounding box of an object in an image. Color grouping should separate the foreground vs. the background in terms of certain global statistics using the bounding box given. In this case we use color likelihoods derived from color histograms: the histogram for the foreground object is computed from the bounding box of the object and the histogram of the background is computed from the rest of the image, similar to the idea that we previously used in object tracking [38]). This should be reasonable if we had a correct bounding box. Here we show that even a completely wrong bounding box, that meets certain assumptions will give a reasonably good result. Then, at any given pixel, if we choose a bounding box that meets those assumptions, we could potentially find which other points in the image are likely to be on the same object as that particular point. As we will show later a fairly good foreground mask can be obtained based on color distributions from a bounding box centered on the object, but of completely wrong shape, size and location. This is explained theoretically if we make certain assumptions. In Figure 8.2 the foreground is shown in blue and the bounding box in red. The bounding box's center is on the foreground, but its size and location are obviously wrong. We make the following assumptions, that are easily met in practice, even without any prior knowledge about the shape and size of the foreground:

1. The area of the foreground is smaller than that of the background: this is true for most objects in images

2. The majority of pixels inside the bounding box belong to the true foreground: this is also easily met in practice since the center of the bounding box is considered to be on the foreground object by (our own) definition.

3. The color distributions of the true background and foreground are independent of position: this assumption is reasonable in practice, but harder to satisfy than the first two, since color is sometimes dependent on location (e.g. the head of a person has a different distribution than that person s clothes).

Even though the three assumptions above are not necessarily true all the time in practice, most of the time they do not need to be *perfectly true* for the following result to hold (they represent only loose sufficient conditions): let $p(c|obj)$ and $p(c|bg)$ be the true foreground and background color probabilities for a given color c, and $p(c|box)$ and $p(c|\neg box)$ the ones

computed using the (wrong) bounding box satisfying the assumptions above. We want to prove that for any color c such that $p(c|obj) > p(c|bg)$ we must also have $p(c|box) > p(c|\neg box)$ and vice-versa. This result basically shows that whenever a color c is more often found on the true object than in the background, it is also true that c will be more likely to be found inside the bounding box than outside of it, so a likelihood ratio test (> 1) would give the same result if using the bounding box instead of the true object mask. This result enables us to use color histograms as if we knew the true object mask, by using any bounding box satisfying the assumptions above. The proof is straight forward and it is based on those assumptions:

$$p(c|box) = p(c|obj, box)p(obj|box) + p(c|bg, box)p(bg|box), \tag{8.1}$$

and

$$p(c|\neg box) = p(c|obj, \neg box)p(obj|\neg box) + p(c|bg, \neg box)p(bg|\neg box). \tag{8.2}$$

Assuming that the color distribution is independent of location (third assumption) for both the object and the background, we have $p(c|obj, box) = p(c|obj)$ and $p(c|bg, box) = p(c|bg)$. Then we have: $p(c|box) = p(c|obj)p(obj|box) + p(c|bg)p(bg|box)$ and similarly $p(c|\neg box) = p(c|obj)p(obj|\neg box) + p(c|bg)p(bg|\neg box)$.

Since the object is smaller than the background (first assumption) but the main part of the bounding box is covered by the object (second assumption) we have $p(obj|box) > 0.5 > p(bg|box)$ and $p(obj|\neg box) < 0.5 < p(bg|\neg box)$. By also using $p(c|obj) > p(c|bg)$, $p(bg|box) = 1 - p(obj|box)$ and $p(bg|\neg box) = 1 - p(obj|\neg box)$, we finally get our result $p(c|box) = p(obj|box)(p(c|obj)-p(c|bg))+p(c|bg) > p(obj|\neg box)(p(c|obj)-p(c|bg))+p(c|bg) = p(c|\neg box)$ (since $p(obj|box) > p(obj|\neg box)$). The reciprocal result is obtained in the same fashion, by noticing that in order for the previous result to hold we must have $p(c|obj) - p(c|bg) > 0$, since $p(obj|box) > p(obj|\neg box)$.

In Figures 8.5, 8.10, 8.11, 8.12 we present some results using this idea. The soft masks are computed as follows: each pixel of color c in the image is given the posterior value $p(c|box)/(p(c|box) + p(c|\neg box))$ in the mask. By the result obtained previously we know that this posterior is greater than 0.5 whenever the true posterior is also greater than 0.5. Therefore we expect that the soft mask we obtain to be similar to the one we would have obtained if we had used the true mask instead of the bounding box for computing the color distributions. We compute such masks at a given location over four different bounding boxes of increasing sizes (the sizes are fixed, the same for all images in all our experiments). At each scale we zero out all pixels of value less than 0.5 and keep only the largest connected component that touches the inside of the bounding box. That is the soft mask for a given scale. Then, as a final result, we average the soft masks over all four scales. In Figure 8.9

we show that the mask obtained is robust to the location of the bounding boxes, so long as the bounding boxes' centers are on the object of interest.

8.3 Pairwise Grouping Using Color

As we mentioned already, the idea of automatically discovering foreground masks at given locations can be easily used for pairwise grouping of features. To prove our point, we present a simple approach for using such masks for color grouping. Given two features (i, j) one can compute the associated soft masks m_i and m_j by centering the bounding boxes at the features locations (x_i, y_i) and (x_j, y_j) and follow the procedure explained previously, using bounding boxes of those four fixed different sizes. The values $m_i(y_j, x_j)$ and $m_j(y_i, x_i)$ can then be used by a classifier to establish the likelihood that the two features belong to the same object. In Figure 8.13 we present some preliminary results of this idea. The red circles represent the location of some contour (feature) i. In white we show all those contours (features) j that were automatically classified as likely to belong to the object of i. Here the classifier was simply thresholding the average $(m_i(y_j, x_j) + m_j(y_i, x_i))/2$ at 0.5 (everything above 0.5 was considered positive). The images show weighted contours for the positive examples, and no contours for the negative. It is important to note that the shape and extent of the foreground is not known, and that all internal parameters are fixed (such as the four fixed bounding box sizes).

In Figure 8.14 we present some *failure* examples of the color pairwise constraints. The algorithm does fail in the sense that it connects contours from the house to contours from the car, but it also connects car contours among themselves, so it should still improve the recognition performance (because most clutter is removed). The main reason for these lower quality results is that the house and the car have similar colors (relative to the histogram binning). This issue could probably be solved to a certain extent by improving the color histogramming. We also want to point out that these *failures* are the exception and not the rule. In fact the vast majority of our results are of similar quality with those in Figure 8.13.

We measured the performance of this simple color grouping algorithm on the MSR database, for which ground truth masks are provided. For each image we randomly picked inside the true foreground object mask 100 positive pairs of points, and similarly 100 negative pairs for which one point is randomly picked inside the mask and the other outside of it. This database may not be the best choice for our algorithm because the foreground objects are sometimes very large and thus tend to violate our first assumption, but the results are encouraging. In Figures 8.10, 8.11, 8.12 we show some qualitative results on about half of the images from the database. We also measured the performance of the pairwise classifier quantitatively, Figures 8.6, 8.7. We plan on using this pairwise classifier for matching and

recognition. Grouping has the potential of considerably pruning the search space for our matching algorithm that uses pairwise constraints. Our results indicate that indeed the pairwise grouping can group most of the positive pairs, while correctly separating almost all negative ones. In cluttered scenes with large backgrounds this can be very useful.

8.4 Using Grouping for Class Specific Segmentation

Class specific segmentation is an important problem in computer vision. Here we propose an efficient method for this task, by combining grouping with an off-the-shelf object class detector. The steps of this algorithm are the following:

1. Use any class specific detector to get the approximate bounding box of objects from the category of interest
2. Reduce in half the width and length of the bounding box returned by the detector, while keeping the same box center. This step will guarantee that most pixels in the reduced bounding box belong to the object of interest
3. Obtain the weighted grouping mask using the grouping method presented in this chapter. In this case we do not need to compute masks from boxes at different scales, since we already have a good bounding box, centered on the object, of size that already has the same scale and spatial extent as the object
4. Threshold the weighted mask to obtain the final hard segmentation of the object.

We used this simple algorithm for getting segmentations of objects from the Pascal 2007 challenge test set using the detector of [61]. This dataset is very difficult for segmentation, objects of the same class varying greatly in appearance, size and shape. We present our results on four classes: cars (Figures 8.15, 8.16), motorbikes (Figures 8.17, 8.18), buses (Figures 8.19, 8.20) and TV's (Figure 8.21). We notice that the results are of high quality, comparable to the state-of-the art in the literature, despite the straight forward combination of our class independent grouping algorithm that requires no training and an off-the-shelf detector.

Figure 8.8: Results on images from the Pascal 2007 challenge database. The red mark indicates the location of the point relative to which the mask is computed. The white pixels are the ones more likely to be on the same object with the red point. The intensity indicates the strength of this likelyhood.

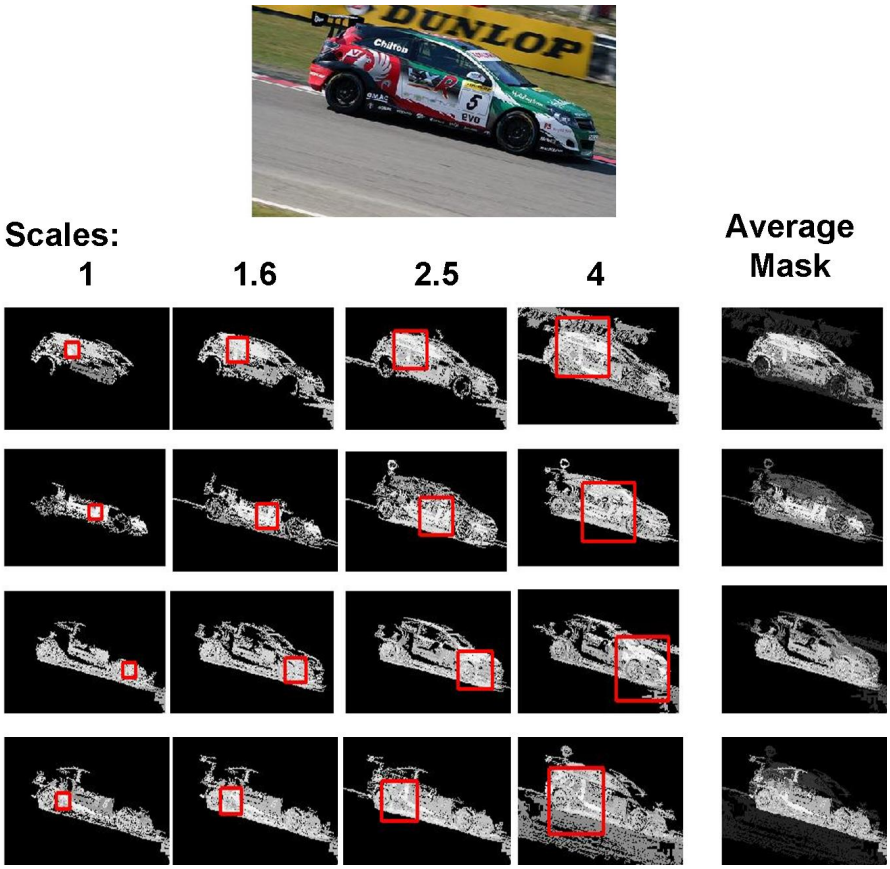

Figure 8.9: As in the previous Figure, finding reasonable object masks is robust to the location of the bounding box, even for objects with a complex color distribution.

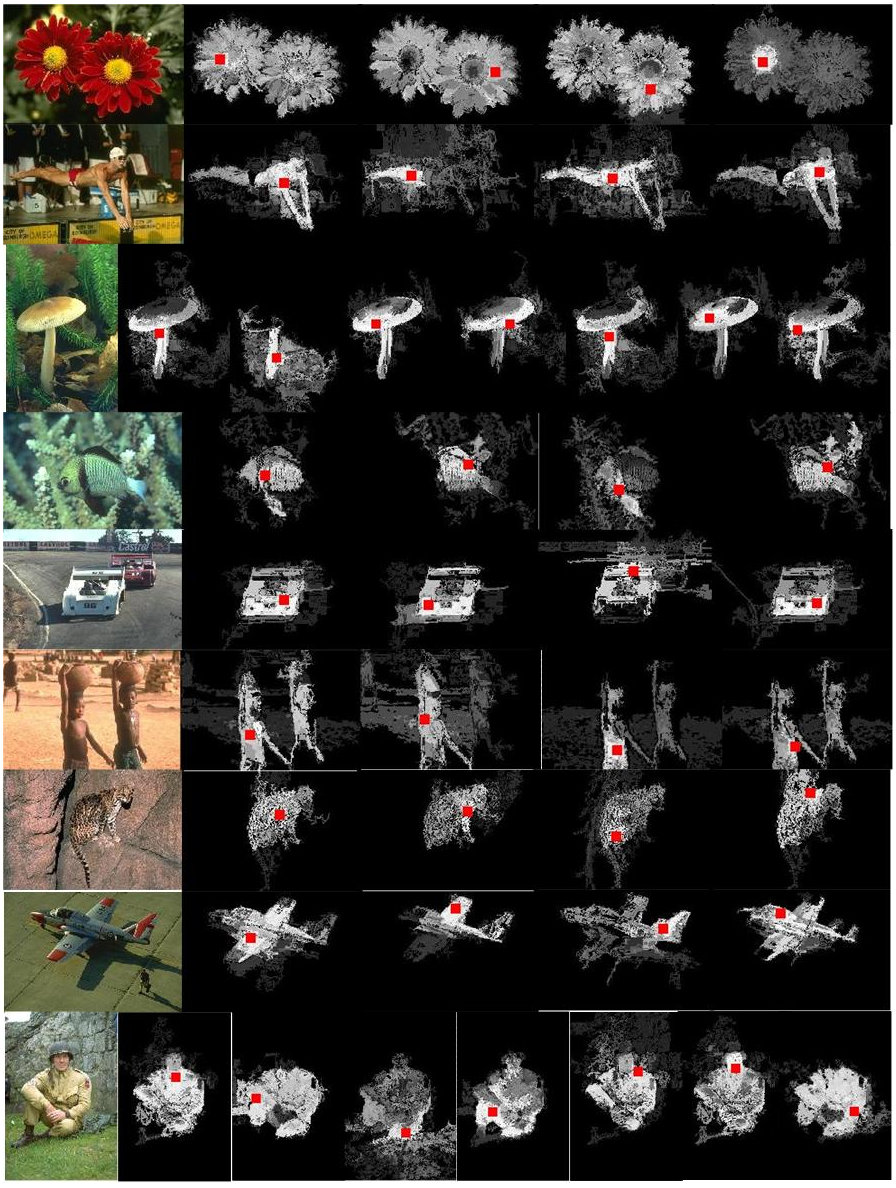

Figure 8.10: Results obtained from the MSR database. 30 points are chosen randomly on the object for each image and for each point a soft mask is computed. Here we show only a few representative results for each image.

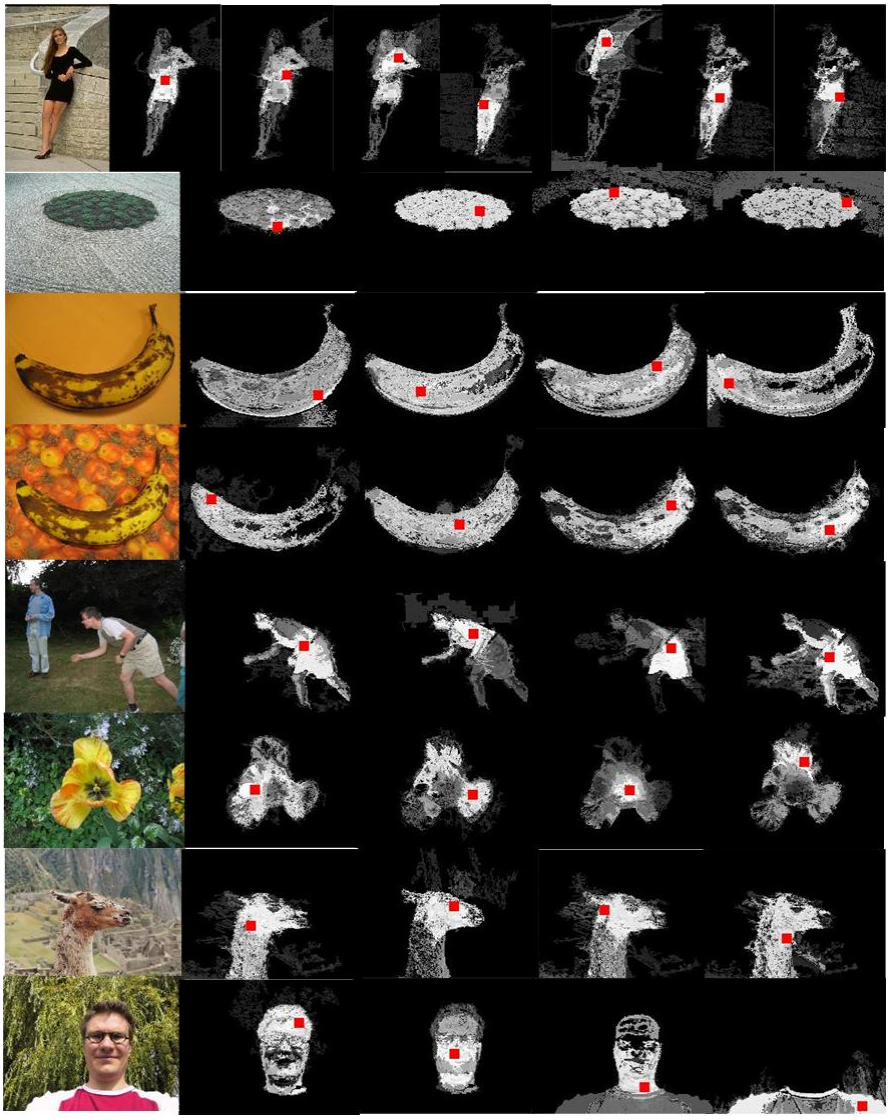

Figure 8.11: Results obtained from the MSR database. 30 points are chosen randomly on the object for each image and for each point a soft mask is computed. Here we show only a few representative results for each image.

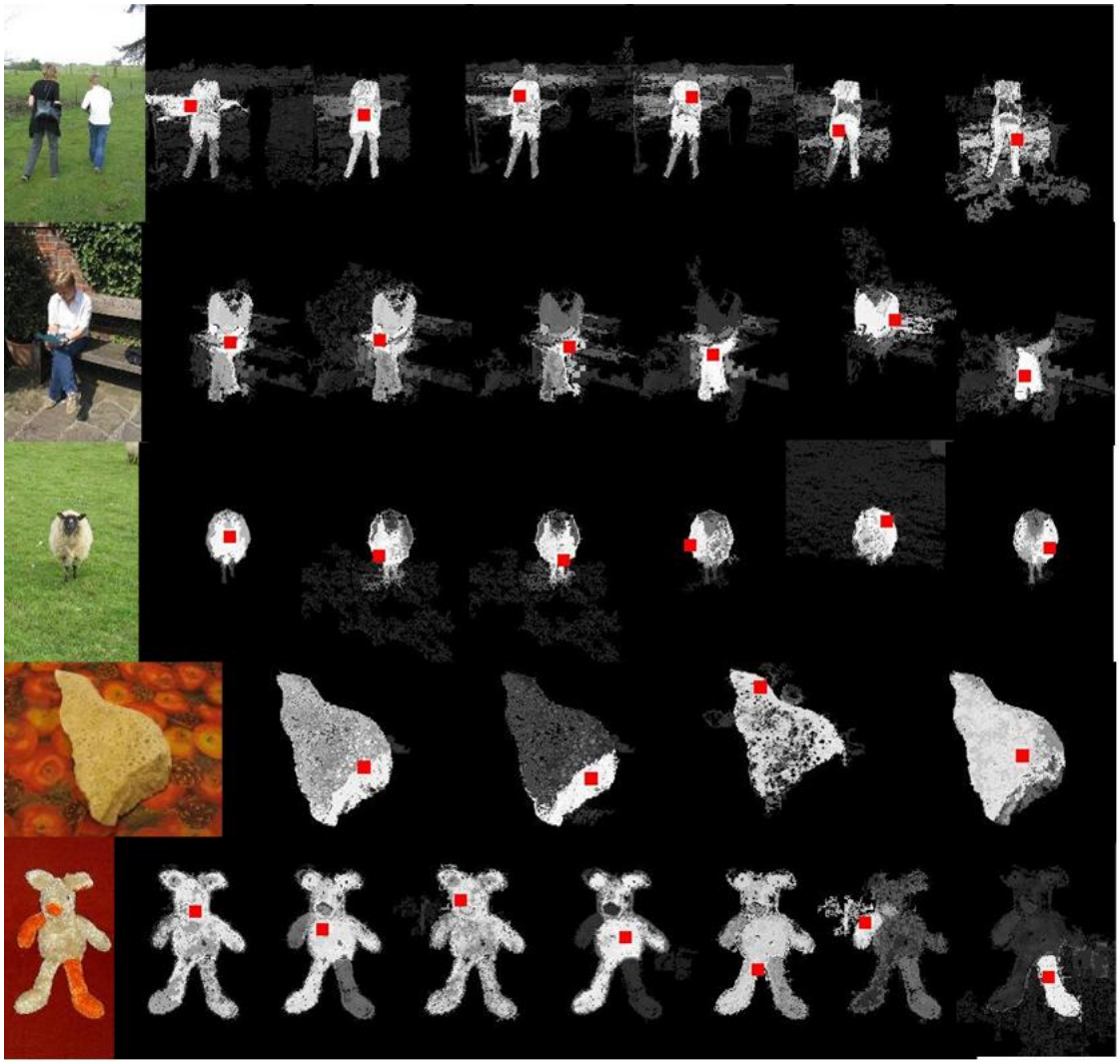

Figure 8.12: Results obtained from the MSR database. 30 points are chosen randomly on the object for each image and for each point a soft mask is computed. Here we show only a few representative results for each image.

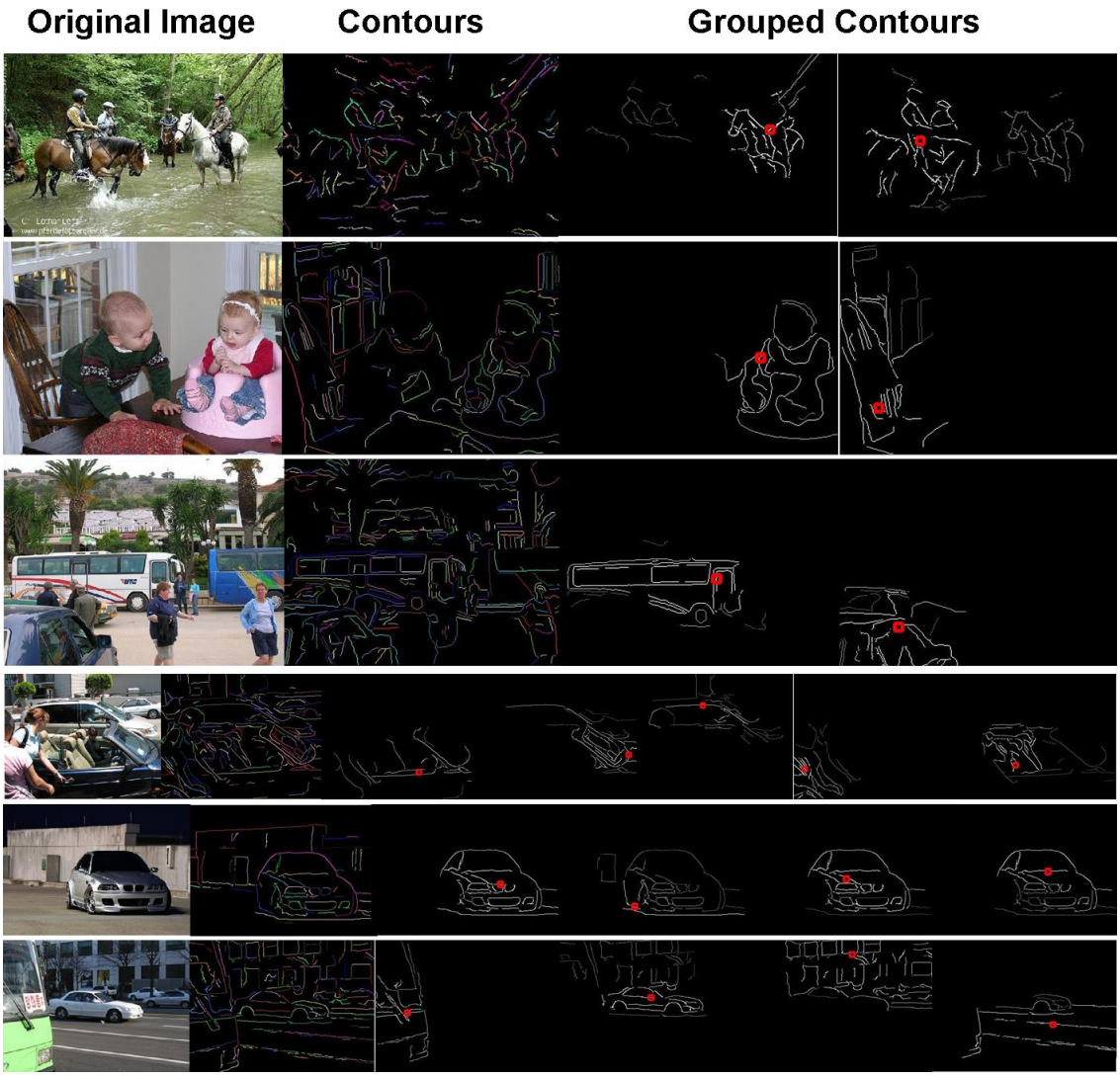

Figure 8.13: Pairwise grouping relationships (constraints) based on color distribution. The contours shown in white are the ones establishing a pairwise grouping relationship based on color with the contour pointed out by the red circle. Notice some difficult cases from very cluttered scenes. The second column (next to the original image) shows all the contours extracted, while the next images show the contours that form a positive grouping relationship with the contour shown by the red circles.

Figure 8.14: Examples when color grouping does not work so well. Upper left corner: original image. Upper middle: all the contours extracted. The rest: the results are shown in the same style as in Figure 8.13. The results are of worse quality than the ones from Figure 8.13. We can see that parts of the house are weakly connected to contours from the car. This happened mainly because the house and the car have similar colors, and the differences were lost during histogram binning.

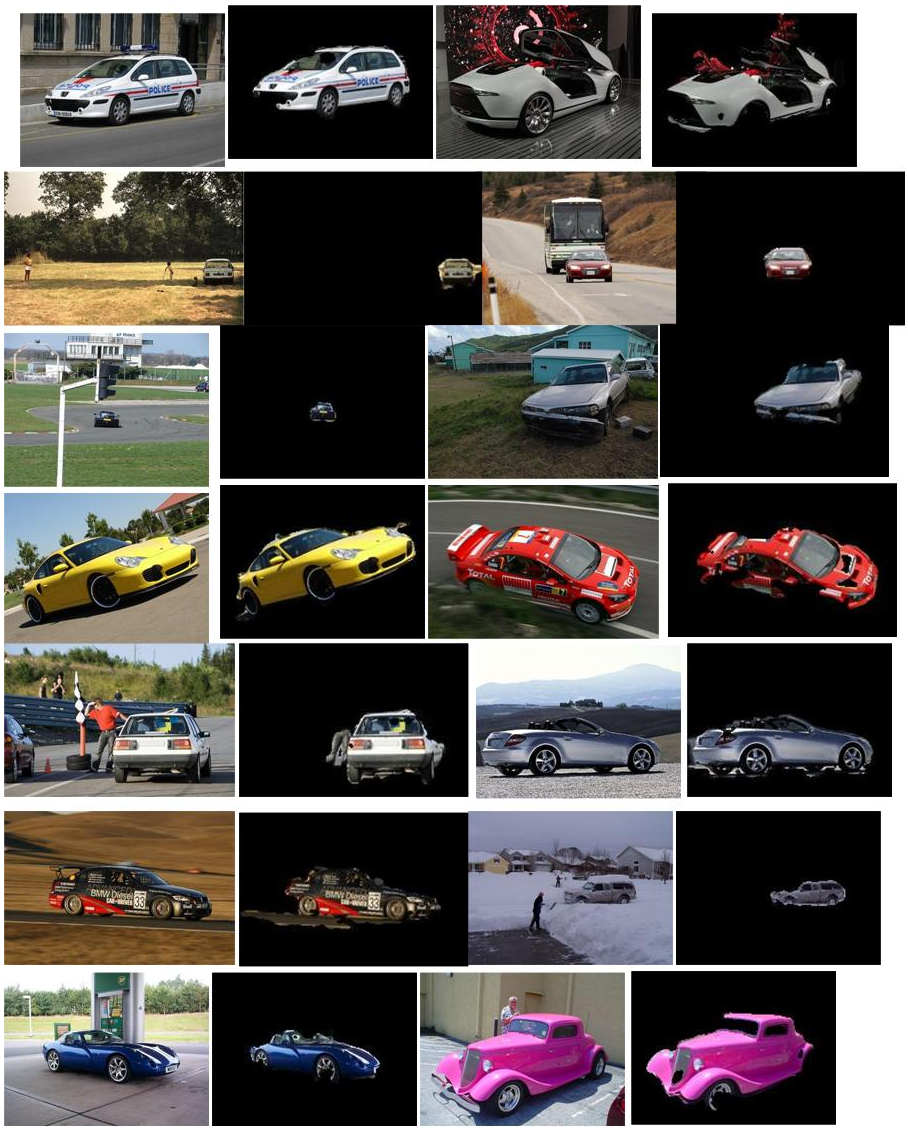

Figure 8.15: Combining grouping with detection for object class segmentation on the Pascal 2007 test set.

Figure 8.16: Combining grouping with detection for object class segmentation on the Pascal 2007 test set.

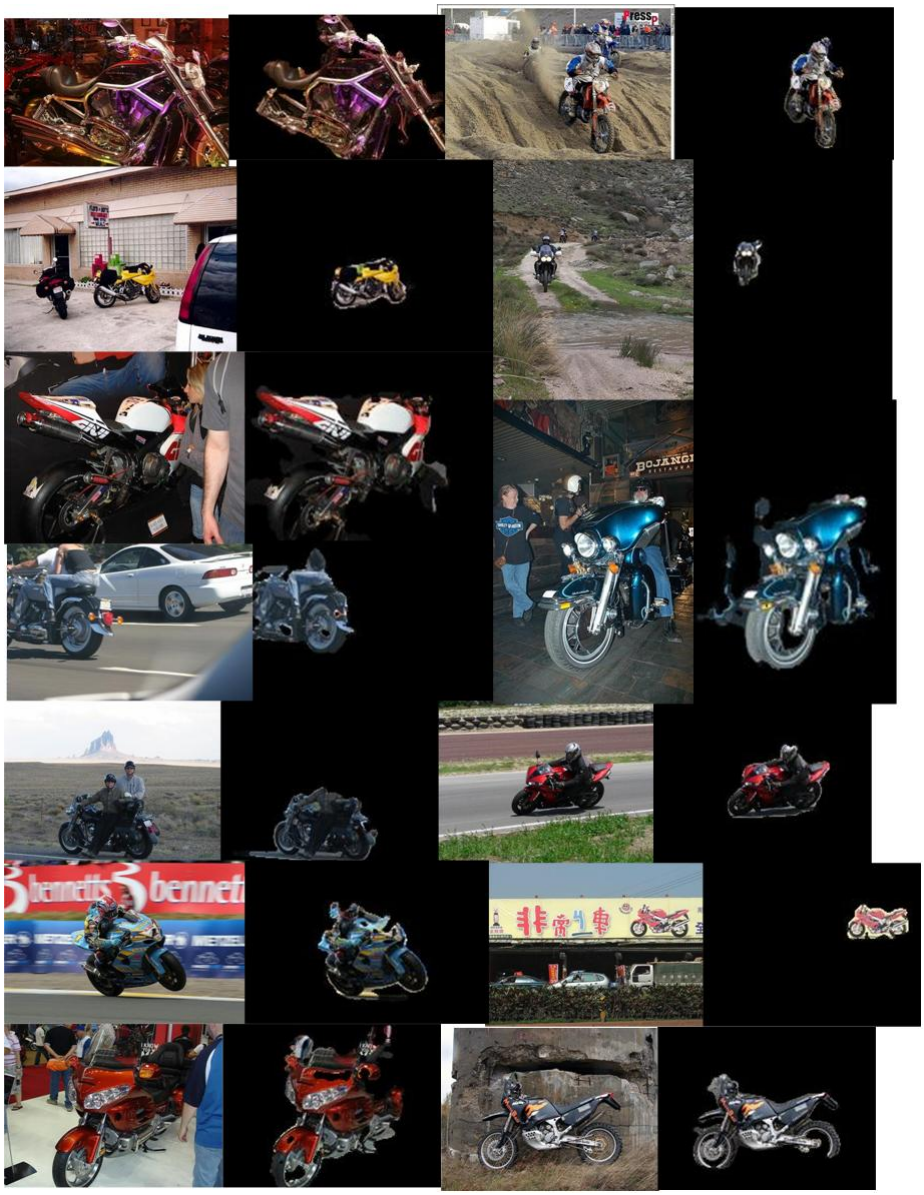

Figure 8.17: Combining grouping with detection for object class segmentation on the Pascal 2007 test set.

Figure 8.18: Combining grouping with detection for object class segmentation on the Pascal 2007 test set.

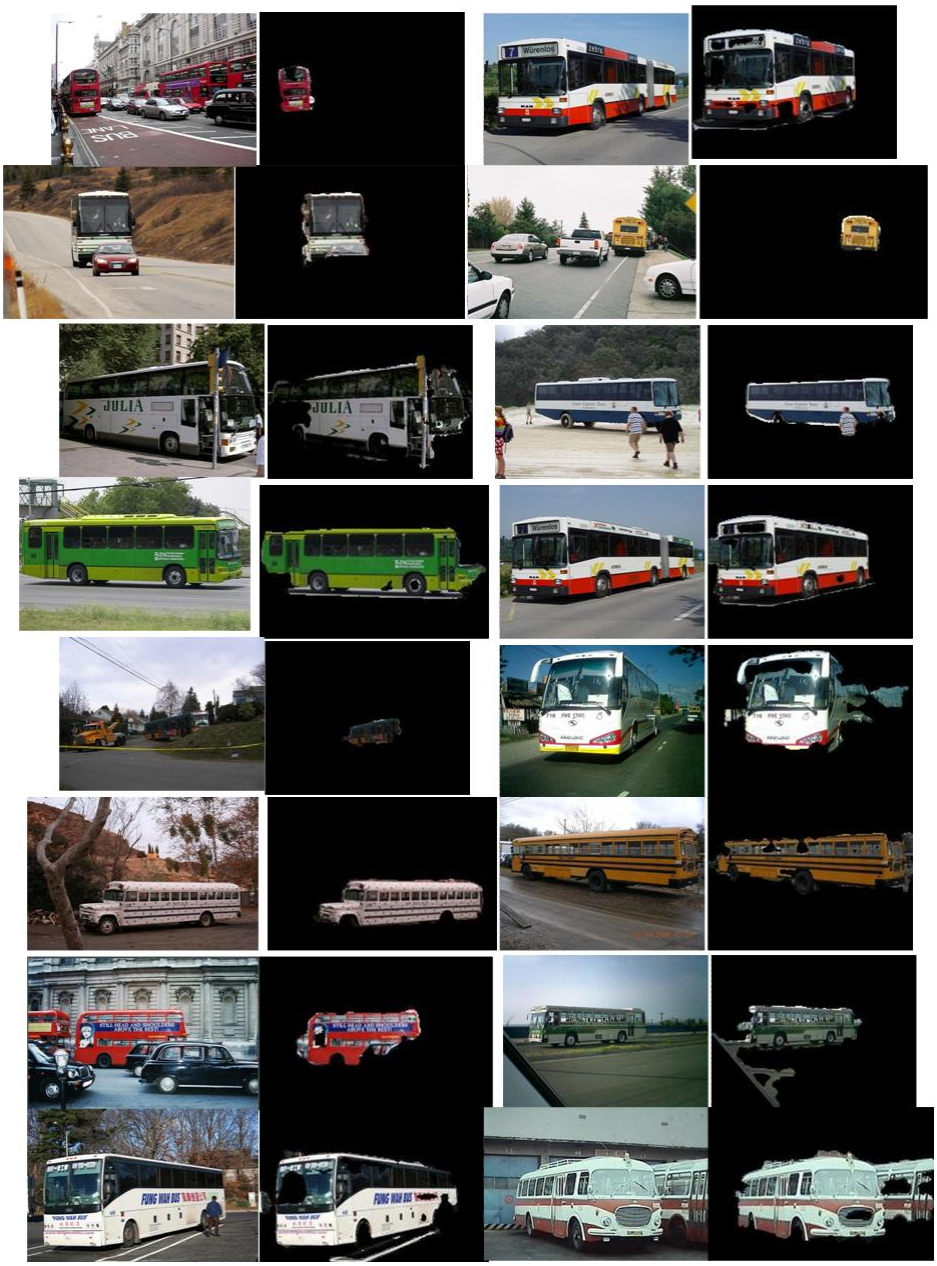

Figure 8.19: Combining grouping with detection for object class segmentation on the Pascal 2007 test set.

Figure 8.20: Combining grouping with detection for object class segmentation on the Pascal 2007 test set.

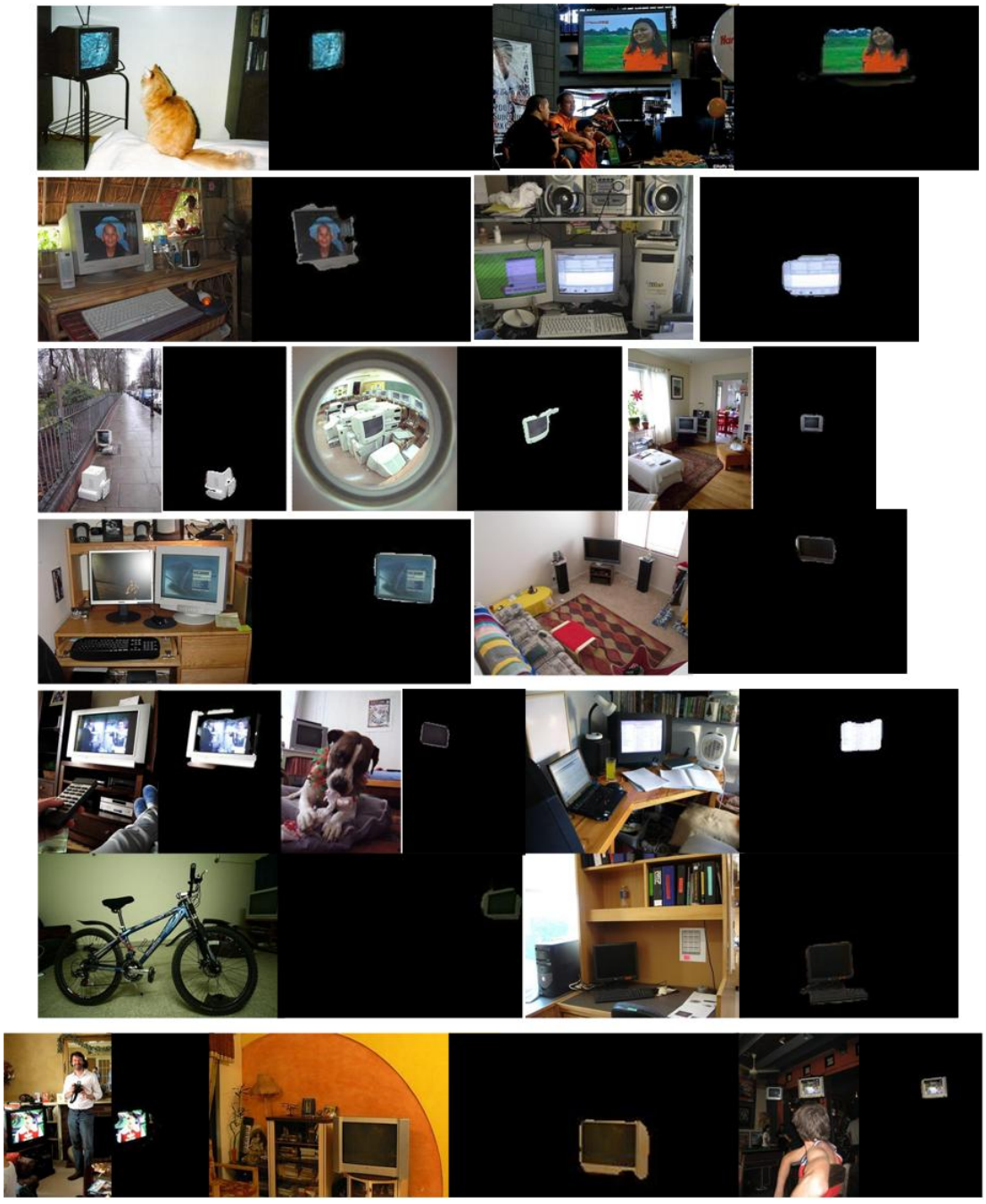

Figure 8.21: Combining grouping with detection for object class segmentation on the Pascal 2007 test set.

Chapter 9

Conclusions and Future Directions

In this thesis we have addressed the problem of matching features or object parts by taking in consideration both how well the features/parts match at the local, first-order level, and how well they preserve their higher-order geometric structure and appearance. Our main goal was to develop efficient algorithms for matching, recognition, learning, and inference, based on the integer quadratic programming formulation of the matching problem, which generalizes the classical graph matching formulations. We have also discussed the connections between our work that uses pairwise interactions between features and hypergraph matching, which considers higher-order relationships between features/object parts. Moreover, we extended our method of learning with graph matching using pairwise interactions to the case of hypergraph matching. We also discussed the connections between graph matching and MAP inference in Markov Networks, for which we proposed novel inference and learning algorithms.

9.1 Brief Summary of the Thesis

In Chapter 2 we introduced spectral graph matching, which is an efficient algorithm for graph matching. Since its publication this method has already been used successfully in several computer vision applications such as object recognition, unsupervised discovery of object categories, action recognition, symmetry discovery and matching 3D scenes and objects. In Chapter 1 we presented some of the published work using our algorithm. In Chapter 2 we also propose an extension of our algorithm to the case of matching tuples of graphs at the same time (instead of pairs of graphs).

We presented our approach to MAP inference problems (Chapter 3) based on a continuous relaxation that is inspired from our spectral matching algorithm. Our method for inference in graphical models, well suited for Markov Random Fields (MRFs) and Conditional Random Fields (CRFs), obeys certain optimality bounds and outperforms in our

experiments popular methods such as Iterative Conditional Modes and Loopy Belief Propagation.

In Chapter 4 we introduced an efficient algorithm that can be applied to both graph matching and MAP inference problems, Integer Projected Fixed Point (IPFP). It significantly outperforms state-of-the-art algorithms, and it can also be used as a post-processing, discretization procedure, significantly improving the performance of other graph matching and MAP inference algorithms.

We presented, for the first time, a method for unsupervised learning with graph matching (Chapter 5) and its extension to hyper-graph matching (Chapter 5) and MAP inference problems (Chapter 6). In our experiments, this method has been successful in improving the performance of several state-of-the-art graph matching methods.

In Chapter 7 we present our approach to object category recognition, which combines the use of spectral matching with powerful geometric relationships from object class specific contours. When combined with an off-the-shelf detector, this method outperforms, on several object categories, the official results from the difficult Pascal 2007 challenge. We also propose an efficient algorithm for grouping features/pixels based on color and show how to use it for object and class specific segmentation (Chapter 8).

In the appendix we present a new method for general optimization of non-negative functions (Smoothing-based Optimization) and show how to apply it to the problem of learning for graph matching. In our experiments it outperforms popular algorithms such as Simulated Annealing and Markov Chain Monte Carlo optimization.

9.2 Conceptual and Theoretical Contributions

On the theoretical side most algorithms proposed in our work are based on the idea of accidental alignments: pairs of correct assignments are likely to preserve the geometry (pairwise relationships) while pairs of incorrect assignments are not. Even though this is not a novel idea, its use for graph matching is novel. This concept is essential in understanding the structure of the matrix $\mathbf{M}$, which helped in the design of the efficient spectral matching algorithm. The low computational complexity of spectral matching is based on a very loose continuous L2 norm relaxation which would not work in practice if $\mathbf{M}$ did not contain a strongly connected cluster formed statistically by the correct assignments. The power iteration used for finding the main eigenvector of $\mathbf{M}$ that is the global optimal solution of the relaxed problem was later a source of inspiration for the design of the IPFP algorithm. Again we could conclude that accidental alignments are indirectly responsible for the design of IPFP. Also, the ideas used for unsupervised learning are based on the same structure of $\mathbf{M}$. In this case, also, accidental alignments are the most important concept: unsupervised learning works mainly because of the accidental nature of agreements between wrong as-

signments vs. the likely agreement between the correct ones. That is also the reason why for MAP problems unsupervised learning does not work. In that case the idea of accidental alignment does not usually apply in practice.

Understanding the structure of the matrix **M** and thinking of the quadratic assignment problem for matching as a graph clustering problem is one of the main conceptual contributions of our work. Our insight into the higher-order scores helped us understand better their importance and helped us design powerful geometric pairwise scores, thus shifting the emphasis from first order, discriminative features, to higher-order interactions. Following our initial work on graph matching, some authors (including ourselves) later designed algorithms for MAP inference, texture discovery, symmetry discovery and hyper-graph matching.

Another important theoretical contribution, not related to matching, is the design of the optimization algorithm presented in the Appendix. Unfortunately, due to time limitation, we did not explore this directions enough. However, we believe that it could become an important avenue for further research. The most relevant conceptual novelty of this algorithm is understanding the relationship between the scale-space theory from vision and image processing and the general subject of optimization of non-linear, non-smooth functions. Again, an intuitive insight from vision and natural images opened the door for an interesting optimization algorithm with important theoretical and practical properties. The results presented in the Appendix show clearly that this method is worth further studying and has the potential of a large applicability in vision, machine learning and optimization.

9.3 Practical Contributions

We took advantage of the intuition from our theoretical work and designed algorithms for important vision applications such as object category recognition, object discovery and discovery of near-regular textures.

In object category recognition, our work was the first to show that simple features such as points and their normals could be used for such a complex problem if the higher order interactions between features are carefully taken into account. We were the first to show that shape can be used for the difficult problem of category recognition, for which most methods were based on the bags of words approach. To these day, there are few methods which put an emphasis on geometry for category recognition, even though there is a general feeling in the vision community that geometry and shape are important for a lot of classes. Most work using geometry is in detection, but beyond the detection setup, of a sliding window, our work is among the first to show competitive results by using shape matching in a translation invariant manner.

In texture discovery, we were the first to demonstrate excellent results by using second-order relationships. Our use of pairwise interactions allowed the texture discovery problem

to be formulated as a matching problem. Without such pairwise interactions, the matching would fail due to the intrinsic similarity of the textels - the elements that form the texture lattice.

9.4 Future Directions

Graph matching is an important vision problem. Our algorithms, due to their efficiency, have increased its popularity in a wide variety of practical applications. We strongly believe that the use of graph matching with higher order interactions is on an ascending path, with a bright future.

Our approach to perceptual grouping has the potential of being useful for important vision tasks. For example, it could be efficiently used for object category discovery, by combining the links based on geometric matching between features from different images with perceptual grouping links between features from the same image. This would further improve the discovery of objects by establishing stronger clusters containing features from the same category. By combining the use of tensors and their eigenvectors with our greedy algorithm for discretization (which allows different mapping constraints), one could look into how to recover these clusters of features in a more efficient manner.

Graph matching could also be successful in cases of parametric transformations if combined with RANSAC in a smart way. The eigenvector values could be used for pruning the set of candidate assignments which would significantly speed up RANSAC. In turn, RANSAC could help in finding intermediate transformations that could further improve the performance of the spectral matching algorithm. A sequential method could incorporate both RANSAC and spectral matching to form a powerful matching and alignment method, with usage in practically all matching/alignment problems.

There is no published work yet, to the best of our knowledge, for finding efficiently the correct matches and the locations of the features to be matched, at the same time, using the same formulation. Matching and registration is usually a two-stage process. First, a rough alignment is obtained by solving a sparse correspondence problem, using features sparsely sampled from the two images. In the second stage, the alignment is refined locally using features that are densely sampled from the two images. Combining matching and registration in the same formulation could be achieved by using hypergraph matching with third order terms. These higher order potentials would contain geometrical information as well as appearance extracted from the interior of the corresponding triangles, defined by the locations of the three features considered for those particular third-order cliques. Matching and finding the exact feature locations will be solved simultaneously by using our ideas from unsupervised learning. As opposed to learning, in this case the parameters discovered at each iteration are the ones defining the features locations in the neighborhood of the

locations found at the previous step.

On a more theoretical side, we believe that our insight into unsupervised learning for graph matching and MAP inference could start a new direction of research in learning. We were the first to demonstrate that unsupervised learning is possible for graph matching. We also showed that in some cases it can also be applied to MAP problems. It would be interesting to investigate and describe the type of problems for which these ideas would work well.

We also believe that the optimization method presented in the Appendix has a great potential for further development and application. It could be further extended to hierarchical models, where sampling at each node could be rewarded by the oracle outputs of the nodes above, which could further refine the sampling space. The nodes higher in the hierarchy could use the sampled points of the several nodes from below and the outputs from above in order to also refine their own sampling space. This could be useful when the dimensionality of the space is very large.

The applicability of our proposed algorithms is large. Matching and optimization are essential in computer vision and almost every other application requires some sort of reliable feature matching or optimization. We expect that our work will have an important impact in the future and we are eager to stay involved in its further development.

Appendix A

Smoothing-based Optimization

Many problems in computer vision require the optimization of complex nonlinear and possibly non-differential functions. Probably the two most popular algorithms used in such cases are Markov Chain Monte Carlo (MCMC) and Simulated Annealing (and their variants). While these algorithms have global optimality properties, in practice they lack efficiency as they require a large number of samples. Variants of MCMC are commonly used in various vision applications such as segmentation [213], object recognition [214] and human body pose estimation [195]. Other difficult optimization problems such as learning graph matching [30] are approached by optimizing an upper bound of the original cost function.

We propose an efficient method for such optimization problems. One of our main ideas is that searching for the global maximum through the scale space of a function [125] is equivalent to looking for the optimum of the original function, but with the added benefit that we have to avoid fewer local optima. Our method works with any non-negative, possibly non-smooth function, and requires only the ability of evaluating the function at any specific point. In order to better explain our algorithm we first discuss the inspirations that stand behind it.

A.1 First Motivation: Smoothing for optimization

There are two main ideas that inspired the design of our algorithm. Even though at a first sight they seem unrelated, their connection becomes obvious once we explicitly present our algorithm.

Functions with many local optima have always been a problem in optimization. Most optimization algorithms are local and prone to get stuck in local optima. There are not many choices of algorithms that attempt to find the global optimum, or even an important optimum, of highly nonlinear and non-differentiable functions. Algorithms such as Graduated Non-convexity [18] address the non-convexity problem by modifying the orig-

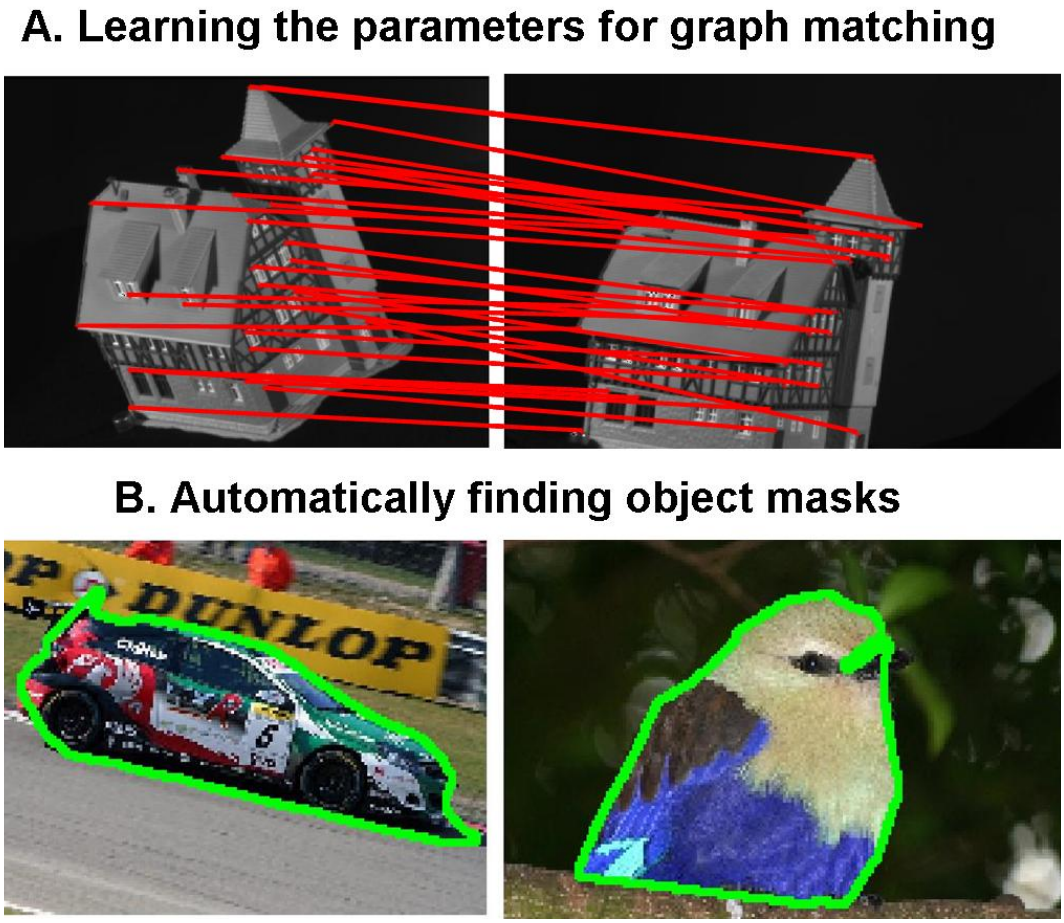

Figure A.1: Many different vision tasks involve the optimization of complex functions: A. Learning the parameters for graph matching (Quadratic Assignments Problem) B. Automatically extracting object masks, given an input bounding box.

inal function and adding to it a large convex component such that the sum will also be convex. Starting from an initial global optimum, and tracking it as the influence of the convex component is slowly reduced, the procedure hopes to finally converge to the original global optimum. Our idea of smoothing is similar: the more we smooth a function (the larger the variance of the Gaussian kernel) the less local optima the function will have. Instead of corrupting the original function by adding to it a foreign convex function such as it is the case with Graduated Non-convexity [18], blurring uses the function's own values to obtain a similar and most probably a better effect (Figure A.2). There is only one caveat

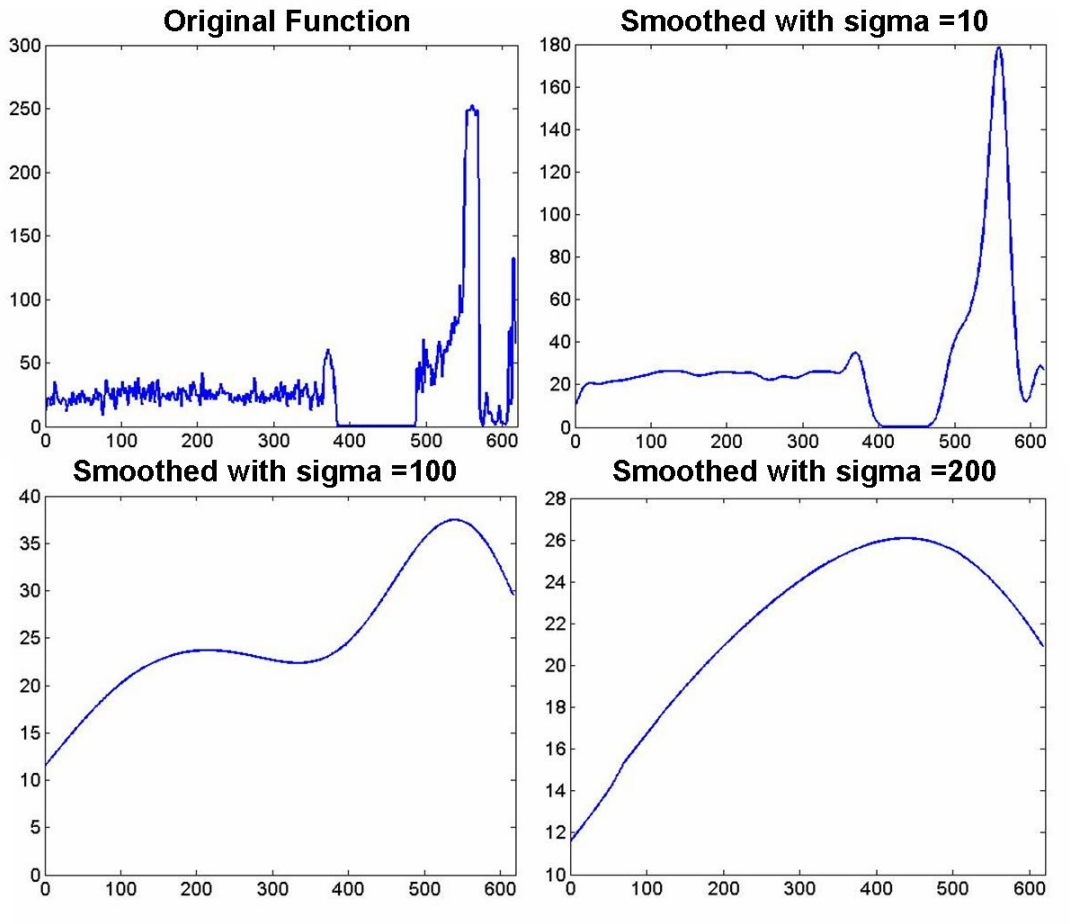

Figure A.2: At higher levels of smoothing less and less local optima survive. Even for a relatively small variance (=10), most of the noisy local optima disappear while the significant ones survive.

with the smoothing approach. Since the Gaussian kernel has infinite support, for smoothing the function at a single point one would have to visit the entire space, and would thus find the global optimum by exhaustive search! So even though the idea sounds interesting, it is in fact impossible to apply in its pure form. Fortunately, as we will show later, in practice things are not even nearly as impractical as they might seem. But before we focus on the practical aspects of our algorithm, let us first convince ourselves that we have the theory to support it.

The results from scale space theory [125] show that for most functions local optima disappear very fast as we increase the variance of the Gaussian blurring. Also, the local optima that survive at higher levels of smoothing can be usually traced back to significant local optima in the original function. Moreover, any nonnegative function with compact support will end up with a single global maximum for a large enough variance of smoothing

[132].

In Figure A.2 the original function is extremely wiggly and any gradient based technique would immediately get stuck in a local maximum. However, as soon as we blur the function with a relatively small sigma, most noisy local optima are vanished. Finally, for a large enough sigma there is only one global optimum which could be traced back to the original global optimum. The unique global maximum of a blurred function cannot always be traced back to the original global optimum, nevertheless, for most functions, it will be traced back to a *significant* local optimum, which constitutes an important progress compared to local optimization techniques. So, if we could somehow have access to the values of our smoothed function in the neighborhood of our current position, we would know where to move next to approach a more important optimum.

A.2 Second Motivation: Updating our knowledge

Our second motivation, which is seemingly unrelated to the smoothing idea, is to represent our knowledge of where the optimum of our complex function is with a multidimensional Gaussian. At any point in time, we want to evaluate the complex function at points where this Gaussian (which represents our current knowledge) has high probability mass and use those evaluations for updating our current Gaussian in a way that will get us closer to the optimum we are looking for. In Figure A.3 we present this idea by running our algorithm on a one dimensional function. The function is sampled in a region where the Gaussian has high probability. Based on those evaluation the variance can increase or decrease. At the final iteration we are indeed very close to the true optimum and our search (sampling) space is minimal (very small variance). This idea is related to the Cross Entropy Method for optimization [175]. Even though technically very different, both ideas are based on sampling from a distribution that is sequentially refined until it converges around the optimum. Other more distantly related work includes importance sampling algorithms for Bayesian Networks [35, 183], where a function, that is relatively inexpensive to draw samples from, is sampled in order to estimate marginals (expressed as integrals that are hard to compute exactly). The samples obtained are used for continuously refining the sampling function in order to obtain better and better estimates of these marginals.

A.3 Algorithm

The two motivations described above are seemingly unrelated, but the connection between them becomes clear once we look in detail at our algorithm. Before describing the algorithm, we present the following theorem on which it is based:

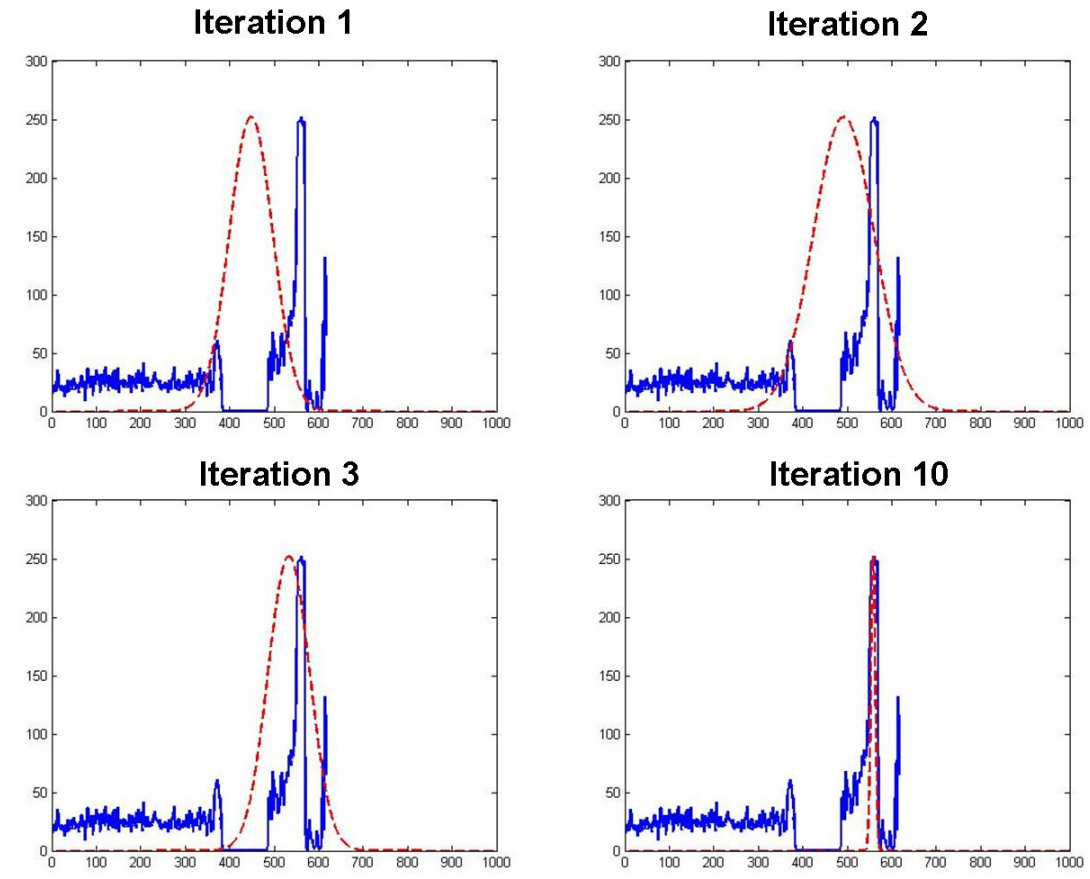

Figure A.3: Refining our knowledge (red dashed line) about the optimum of the function we want to optimize (blue line). At each iteration the mean of the Gaussian represents our current guess, while its variance the uncertainty. By the tenth iteration we are very close to the true optimum and also very certain about where it is (very small variance).

Theorem 1: Let $f : \mathbb{R}^n \rightarrow \mathbb{R}+$ be a non-negative multi-dimensional function. Let its scale space function (its smoothed version) be defined as $F(\mu, \sigma^2 I) = \int g(x; \mu, \sigma^2 I) f(x) dx$, where g is a multidimensional Gaussian (of dimension n) with mean μ and covariance matrix $\sigma^2 I$. Given the pair $(\mu^{(t)}, \sigma^{(t)})$ at time step t, we define $(\mu^{(t+1)}, \sigma^{(t+1)})$, at the next time step $t+1$, by the following update rules (i refers to dimension indices, and $g^{(t)}(x) = g(x; \mu^{(t)}, \sigma^{(t)})$):

1. $\mu^{(t+1)} = \frac{\int x g^{(t)}(x) f(x) dx}{\int g^{(t)}(x) f(x) dx}$

2. $\sigma^{(t+1)} = \sqrt{\frac{1}{n} \frac{\int (\sum_{i=1}^{n} (x_i - \mu_i)^2) g^{(t)}(x) f(x) dx}{\int g^{(t)}(x) f(x) dx}}$

Then, the following conclusions hold:

a). $F(\mu^{(t+1)}, \sigma^{(t)}) \geq F(\mu^{(t)}, \sigma^{(t)})$

b). $F(\mu^{(t)}, \sigma^{(t+1)}) \geq F(\mu^{(t)}, \sigma^{(t)})$

Proof: see the Appendix.

This theorem basically states that the update steps 1 and 2 represent growth transformations [12, 93] for the function F. They provide a specific way of updating the Gaussian which represents our knowledge about the optimum at any specific time step. This gives us the connection to our second motivation. Also, the updating steps are in the direction of the gradient of the scale space function F, and thus provide us with a way of traveling not only through the original search space but also through scale. This gives us the link to the first motivation based on smoothing.

The smaller σ the more F approaches the original function f. It is clear that the global optimum of F is the same as the global optimum of f. So optimizing f is practically equivalent to optimizing F. The difference is, as we discussed in Section A.1, that in F it is much easier to avoid the local optima. The theorem above is the basis of our method. Probably its most interesting feature (as we found in our experiments in Section A.4) is its ability of updating automatically σ, which grows if we need to escape from valleys and shrinks if we reach the top of an important mountain on the function's surface (we know from scale space theory that σ will indeed shrink once we are close to such optima). As mentioned before, the main issue of this idea is how to compute the update steps 1 and 2.

Since they cannot be computed exactly (because we can only evaluate f at a given point and do not want to search the whole space) we will resort to methods commonly used for estimating integrals. One well-known possibility is the Monte Carlo Integration method [148], the other one is by Gaussian quadrature [162] that could be more efficient in practice in spaces of lower dimensions, because it requires fewer function evaluations.

Our algorithm is an implementation of the above theorem. Below we present the version of the algorithm that uses Monte Carlo Integration sampling, but, as we mentioned above, Gaussian quadratures could also be used and are often more efficient in practice:

1. Start with initial values of $\mu^{(0)}$ and $\sigma^{(0)}$, set t = 0
2. Draw samples $s_1, s_2, ..., s_m$ from the normal distribution $N(\mu^{(t)}, (\sigma^{(t)})^2 I)$
3. Set: $\mu^{(t+1)} = \frac{\sum_{k=1}^{m} s_k f(s_k)}{\sum_{k=1}^{m} f(s_k)}$
4. Set: $\sigma^{(t+1)} = \sqrt{\frac{1}{n} \frac{\sum_{k=1}^{m} (\sum_{i=1}^{n} (s_i^{(k)} - \mu_i^{(t)})^2) f(s_k)}{\sum_{k=1}^{m} f(s_k)}}$
5. if $\sigma^{(t+1)} < \epsilon$ stop.
6. t = t+1. Go back to step 2

Notice that we apply the update steps for μ and σ at the same time, even though our theorem gives theoretical guarantees only if we apply them sequentially. Even if that is of theoretical concern, we found that in practice this does not hurt the performance, but on the contrary, it actually makes the algorithm more efficient.

Our algorithm is also related to the Mean Shift algorithm [40], [39], since both algorithms can adapt the mean and the kernel size. However, the difference between the two algorithms is substantial. Mean Shift has samples drawn from $f(x)$ and uses a kernel $k(x)$ to weight the samples. In our case, we cannot draw samples from $f(x)$, because the functions we want to optimize are very complex, so instead we draw samples from $g(x)$ and use instead $f(x)$ to weight these samples. Also in the case of Mean Shift it is not possible to evaluate $f(x)$ exactly at a specific point, whereas in our case it is. It seems that the two algorithms are complementary to each other.

A.4 Experiments on Synthetic Data

In the first set of experiments we compare the performance of our algorithm to two well established methods commonly used in complex optimization problems: Markov Chain Monte Carlo (MCMC) and Simulated Annealing (SA). While MCMC is not specifically designed for optimization, it has been successfully used for this purpose in the vision literature. SA on the other hand has guaranteed optimality properties in a statistical sense. Given enough samples, both MCMC and SA are guaranteed to find the global maximum, but often the number of samples required is very large, thus neither method is particularly efficient.

For this experiment we used synthetic data. Given a rectangle of known dimensions, known location (in 3D) of its center, we rotate it in 3D by $\theta_t = (\theta_x, \theta_y, \theta_z)$ and obtain its projection on the XY plane as a binary mask I_{θ_t}. The algorithms are provided only with this mask, their task being of finding the θ^* which maximizes the overlap between the mask given I_{θ_t} and I_{θ^*}. More precisely, the score that needs to be maximized is:

$$f(\theta^*) = \left(\frac{N(I_{\theta_t} \cap I_{\theta^*})}{N(I_{\theta_t} \cup I_{\theta^*})}\right)^{10} \tag{A.1}$$

Here $N(I_{\theta_t} \cap I_{\theta^*})$ is the area of the intersection of the two masks, and $N(I_{\theta_t} \cup I_{\theta^*})$ is the area of their union. We raised the score function to the 10^{th} power because it is too flat otherwise and the convergence becomes too slow.

This score function is periodical with infinitely many local and, of course, global maxima. The global maxima have obviously the known value of 1. The resolution of the image mask is such that an exhaustive search of the angles space in the intervals $[0, 90]$ would require around 10^{10} samples (the function is sensitive to changes in angles as little as 0.02 degrees).

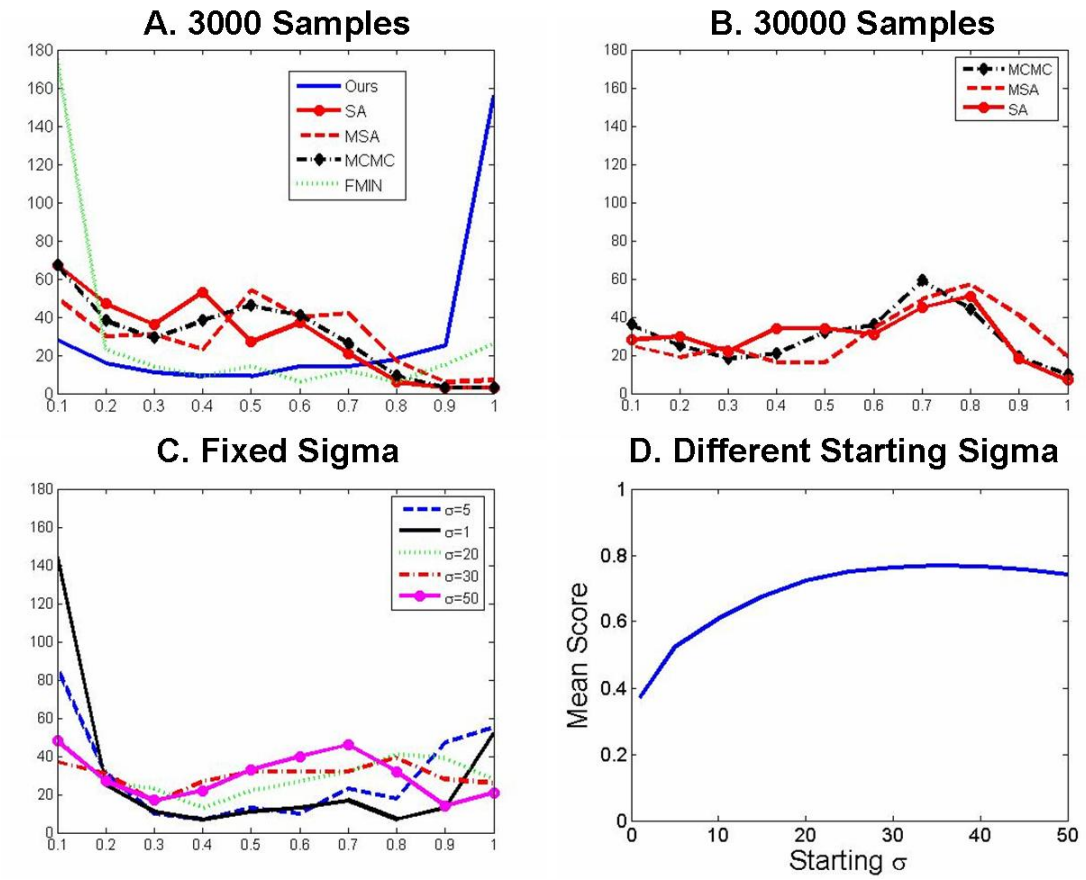

Figure A.4: A. all algorithms can run for a maximum of 3000 samples. B. the algorithms were run for a maximum of 30000 samples. C. Our algorithm, without the ability of changing its initial covariance matrix, for different initial $\sigma^{(0)}$. D. Our algorithm, with different starting $\sigma^{(0)}$ (shown on the X axis in degrees), being able to adapt it. The mean score obtained over 300 experiments is shown in plot D. The rest of the plots show histograms of the scores obtained over 300 experiments.

In Figure A.4 , plot A, we compare our method against MCMC, standard Simulated Annealing (SA), Metropolis-SA (MSA), and Nelder-Mead method as the *fminsearch* (FMIN) function from the Matlab optimization toolbox (for *fminsearch* we used as the cost function $1 - f(\theta)$ since it is a minimizing procedure, but the results showed here were only in terms of $f(\theta)$). All algorithms except *fminsearch* are limited to a maximum of 3000 function evaluations (samples). The plot shows the histogram of the maximum scores obtained over 300 experiments. Each algorithm ran on the same problems, with the same starting points and ground truth θ_t. For each experiment, both the starting point and the ground truth were chosen randomly in the degree space $[0, 360]$ (in each dimension of θ). For MCMC, SA

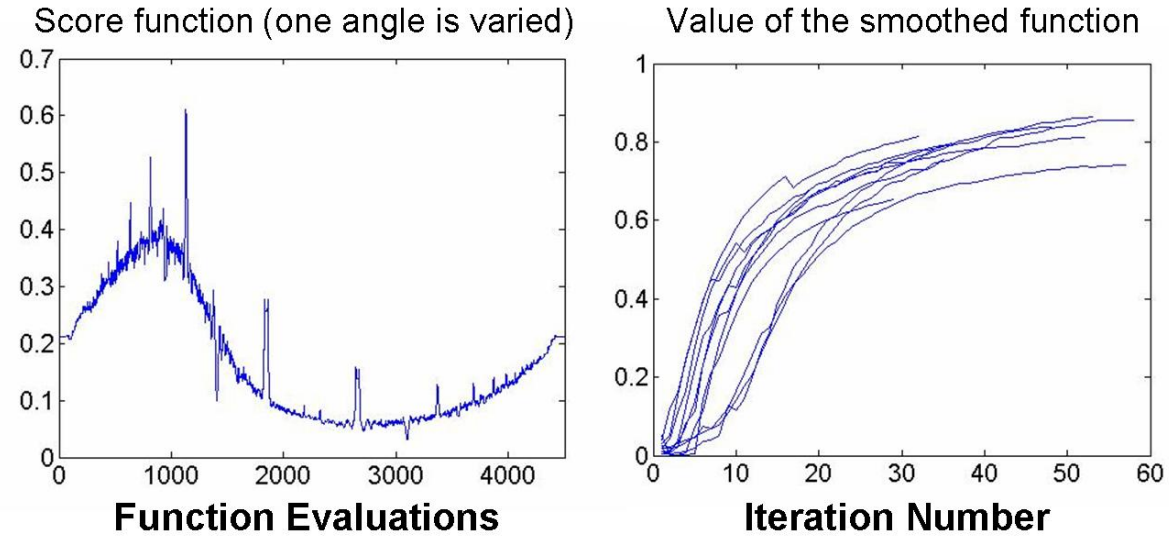

Figure A.5: Left: the score function evaluated every 0.02 degrees of θ_z in $[0, 90]$. The other angles were kept constant. Notice how wiggly the function is due to the fact that the mask is discrete in practice. Right: the value of the smoothed function for each iteration of our algorithm. Notice that it is mainly monotonic which agrees with the theory. Sometimes it fails, but this happens only because the updating steps are approximations to the ones in the theorem. The plots belong to the first 10 random experiments.

and MSA we chose the variance of the proposal distribution that gave the best performance. The worst performer was *fminsearch*, as expected, since it is a local method and the score function has a lot of local optima (see Figure A.5). Our algorithm outperformed all the others (Figure A.4. Even when we allowed MCMC, SA and MSA to run for 10 times more samples, their performance was still inferior to ours (plot B). Of course, for a sufficiently large number of samples MCMC, SA and MSA will always find the right solution, but the point of this experiment was to consider the efficiency of the different algorithms.

In the next experiment (plot C) we wanted to emphasize that one of the main strengths of our algorithm is its capacity to change the covariance of its sampling distribution. On the one hand we see that if we keep this covariance fixed its performance degrades considerably, for a wide range of σ(the starting covariance matrix was diagonal, with diagonal elements equal to σ) (Figure A.4, plot C). On the other hand, if we allow this covariance to change, the starting value of σ is not very relevant (Figure A.4, plot D). Except when the starting

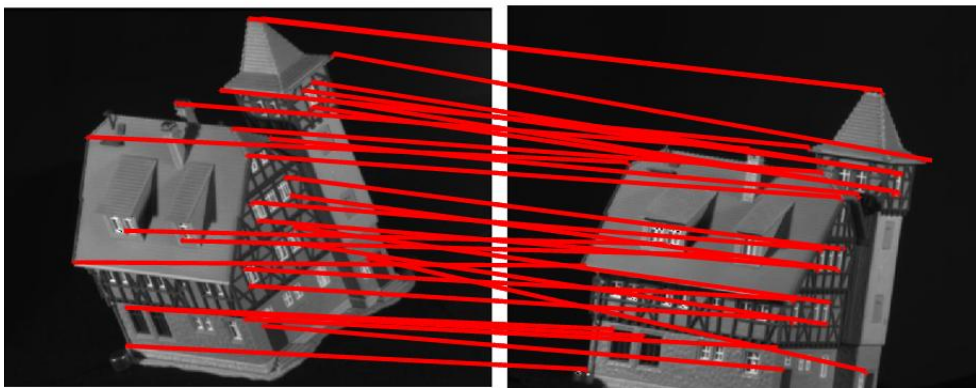

Figure A.6: All the features in the first image were correctly matched to the features from the last image (House sequence).

σ is very small (< 5 degrees), the mean score obtained over the same 300 experiments does not vary much. From this we can draw the conclusion that our algorithm is most often able to adapt its covariance correctly during the search, regardless of its starting value.

A.5 Experiments on Learning for Graph Matching

Graph matching, also known as the quadratic assignment problem (QAP) is a problem frequently encountered in computer vision. The task is formulated as an optimization problem, with the goal of finding the assignments that maximize a quadratic score, given the constraints that one feature from one image can match only one other feature from the other image, and vice-versa:

$$\mathbf{x}^* = \mathbf{argmax}(\mathbf{x}^\mathbf{T}\mathbf{M}\mathbf{x}). \tag{A.2}$$

Here $\mathbf{x}^*$ must be a binary vector such that $x^*_{ia} = 1$ if feature i from one image is matched to feature a from the other image, and $x^*_{ia} = 0$ otherwise. As stated before, each feature from one image can match only one feature from the other, and vice-versa. This problem is NP hard, so most research on this topic focused mainly on developing efficient algorithms for finding approximate solutions, such as the graduated assignment (GA) [76], the spectral matching [117] or linear approximations [15] algorithms . However, as in graphical models, it is not only important to find the optimal solution, but it is also very important to have the right function to optimize. In this case the matrix $\mathbf{M}$ contains the second order potentials,

such that $M_{ia;jb}$ measures how well the pair of features (i, j) from one image agrees in terms of geometry and/or appearance with their matched counterparts (a, b) from the other image. Using the right function $M_{ia;jb}$ is crucial for obtaining correct correspondences. Most work on this problem uses pairwise scores $M_{ia;jb}$ that are designed manually. Unlike in the graphical models literature where the learning issue is addressed abundantly, to the best of our knowledge there has only been two papers [30] [120] published on learning the quadratic assignment pair-wise potentials using the performance of the algorithm as the score function to optimize' . This is mainly because learning for graph matching is a harder problem than learning for graphical models, because the matching scores used in QAP are not normalized probability distributions.

The function we want to optimize for learning is similar to the one in [30]:

$$f(\mathbf{w}) = \sum_{i=1}^{m} n_c^{(i)}(\mathbf{w}) \tag{A.3}$$

Here $n_c^{(i)}$ is the number of correct matches for image pair i, and i iterates over the training image pairs (total of m image pairs), and $\mathbf{w}$ is the vector of parameters that define the pairwise scores. Then, the optimization problem is formulated as:

$$\mathbf{w}^* = argmax(f(\mathbf{w})) \tag{A.4}$$

We use our algorithm on two tasks that are the same as the ones in [30]. We used exactly the same image sequences both for training and testing [2, 1], and the same features, which were manually selected by [30]. For solving the quadratic assignment problem we used the spectral matching algorithm [117], instead of the the graduated assignment [76], because it is faster. The goal of these experiments was not to directly compare the two learning algorithms, but rather to show that our algorithm is suitable for this problem also. The algorithm in [30] is specifically designed for a certain class of score functions, whereas our algorithm can work with any pairwise score $M_{ia;jb}$. The algorithm in [30] optimizes a convex upper bound to the cost function, while in our case we attempt to optimize directly $f(\mathbf{w})$.

The type of pair-wise potential that we want to learn is:

$$M_{ia;jb} = \mathbf{exp}(w_0 + w_1 \frac{|d_{ij} - d_{ab}|}{|d_{ij} + d_{ab}|} + w_2 |\alpha_{ij} - \alpha_{ab}|) \tag{A.5}$$

Here d_{ij} and d_{ab} are the distances between features (i, j) and (a, b) respectively, while α_{ij} and α_{ab} are the angles between the X axis and the vectors $\vec{ij}$ and $\vec{ab}$, respectively. As in [30] we first obtain a Delaunay triangulation and allow non-zero pairwise scores $M_{ia;jb}$ if and only if both (i, j) and (a, b) are connected in their corresponding triangulation. The pair-wise scores we work with are different than the ones in [30] because we wanted to put more emphasis on the second order scores. The authors of [30] make the point that

Table A.1: Matching performance on the hotel and house datasets. In the first three columns the same 5 training images from the House dataset were used. For the fourth column 106 training images from the House sequence were used. SC stands for Shape Context [13]

Datasest	Ours No SC (5)	[30] SC (5)	[30] SC (106)
House	99.8%	$< 84\%$	$\approx 95\%$
Hotel	94.8%	$< 87\%$	$< 90\%$

after learning there is no real added benefit from using the second order potentials, and that linear assignment using only appearance terms (based on Shape Context) suffices. We make the counter argument by showing that in fact the second order terms are much stronger once distance and angle information is used (which they did not use). Our performance is significantly better even when we do not use any appearance terms (Figure A.1). With only 5 training images used, we obtain almost 100% accuracy, more than 15% better than what they achieve using the exact same training and testing pairs of images.

A.5.1 Experiments on Finding Object Masks

Next we present an application (Figures A.7, A.1) of our algorithm that is related to GrabCut [174] and Lazy Snapping [124]. The user is asked to provide the bounding box of an object, and the algorithm has to return a polygon which should be as close as possible to the true object boundary. This is just another instance of the foreground-background segmentation problem. In computer vision most segmentation algorithms approach this task from bottom up. The problem is usually formulated as a Markov Random Field [123] with unary and pairwise terms that use information only from a low, local level, and do not integrate a global view of the object, which would be needed for a better segmentation. Here we present a simple algorithm for obtaining object masks that is based on the global statistics of the foreground vs. the background. The main idea is that a good segmentation is the one that finds the best separation (in terms of certain global statistics) between the foreground and the background. In this case we use color likelihoods derived from color histograms (of the initial foreground and background defined by the bounding box given) as the global statistics of either foreground or background. Starting from the bounding box provided by the user, the algorithm has to find the polygon that best separates the color likelihood histogram computed over the interior of the polygon (foreground) from the corresponding histogram computed over its exterior (background). The function to optimize looks very simple but it is non-differentiable, highly non-linear and highly dimensional, so the task could be very difficult:

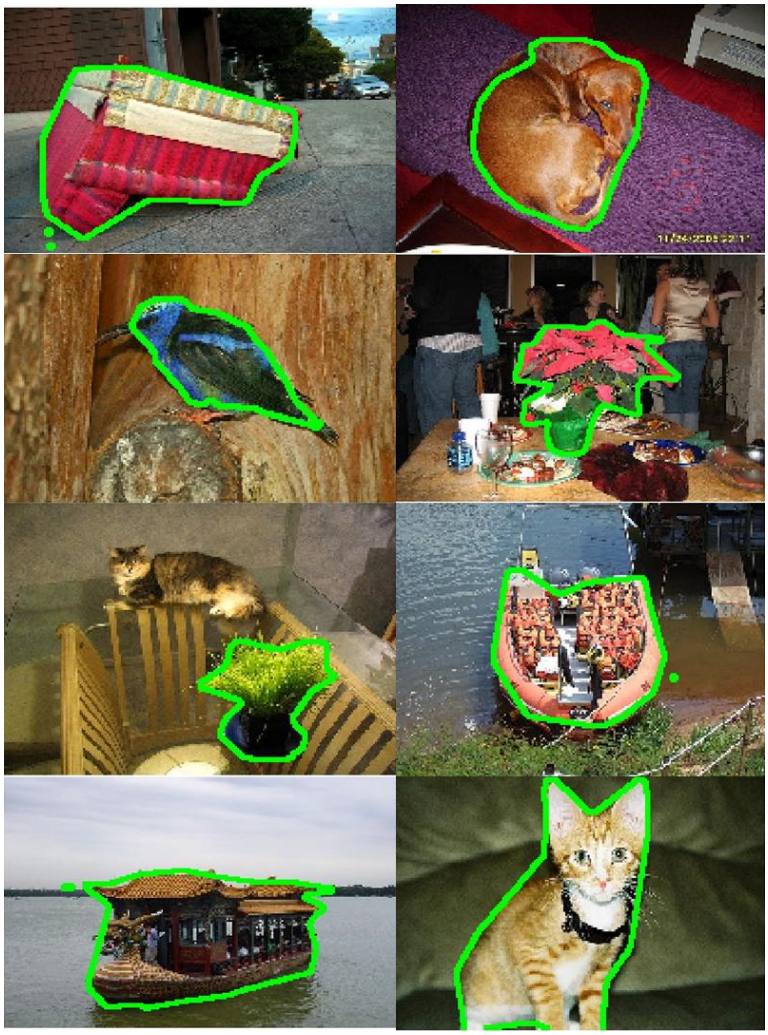

Figure A.7: The masks of objects are automatically found, given their ground truth bounding boxes.

$$f(\mathbf{x}, I) = 1 - h_f(\mathbf{x}, I)^T h_b(\mathbf{x}, I) \tag{A.6}$$

Here $\mathbf{x}$ are the vertices of the polygon, $h_f(\mathbf{x}, I)$ and $h_b(\mathbf{x}, I)$ are the foreground and background normalized color likelihood histograms, given the image I . The likelihood of color c is computed as $l(c) = \frac{N_f(c)}{N(c)}$, where $N_f(c)$ is the number of pixels of color c inside the initial bounding box and $N(c)$ is the total number of pixels of color c in the image.

Our segmentation algorithm is very simple and can be briefly described as follows:

1. initialize x with the bounding box provided by the user
2. find $x^* = argmax_x(f)$, by using the algorithm from Section A.3 without changing the number of polygon vertices
3. if x^* does not improve significantly over the previous solution stop.
4. add new vertices at the midpoints of the edges of x^*
5. go back to step 2

As long as the color distribution of the foreground is different from the background, this algorithm works well, being very robust to local variations in color or texture, unlike MRF based algorithms whose unary and pairwise terms are more sensitive to such local changes (see Figures A.7, A.1).

A.6 Conclusions

We have presented an efficient method for optimization, which could be an interesting alternative to well-established methods such as Markov Chain Monte Carlo or Simulated Annealing. The experiments we presented here were not application driven, but they are nevertheless encouraging and make us believe that this method has a lot of potential, and is worthy of more research and testing. As future work we will further explore its theoretical properties and limits as well as its practical application to different vision problems. Another exciting application of this method would be in learning the parameters of the potentials of Markov Netorks in order to maximize the actual classification performance, which would constitute a relative novelty.

A.7 Proof of Theorem 1, Conclusion a

Let the inverse covariance matrix be $\Lambda = \Sigma^{-1}$. Then we have

$$g^{(t)}(x) = e^{-\frac{1}{2}(x-\mu^{(t)})^T\Lambda(x-\mu^{(t)})}$$

To simplifying notations we dropped the normalizing constant since it does not depend on μ. We will use the following inequality which holds for any u and v:

$$e^{u+v} \geq (1+u)e^v$$

Let us define: $\delta = \mu^{(t+1)} - \mu^{(t)}$, then:

$$\begin{aligned} g^{(t+1)}(x) &= e^{-\frac{1}{2}(x-\mu^{(t+1)})^T\Lambda(x-\mu^{(t+1)})} \\ &= e^{-\frac{1}{2}(x-\mu^{(t)}-\delta)^T\Lambda(x-\mu^{(t+1)}-\delta)} \end{aligned}$$

Now using our inequality we have:

$$g^{(t+1)}(x) \geq (1 + (x - \mu^{(t)})^T \Lambda \delta - \frac{1}{2}\delta^T \Lambda \delta) g^{(t)}(x)$$

Since f is non-negative the inequality carries over to F:

$$F(\mu^{(t+1)}) \geq \int (1 + (x - \mu^{(t)})^T \Lambda \delta - \frac{1}{2}\delta^T \Lambda \delta) g^{(t)}(x) f(x) dx$$

Remembering that $\delta = \frac{\int (x-\mu^{(t)}) g^{(t)}(x) f(x) dx}{\int g^{(t)}(x) f(x) dx}$ we have:

$$\int (x - \mu^{(t)})^T \Lambda \delta g^{(t)}(x) f(x) dx =$$
$$\frac{(\int (x - \mu^{(t)}) g^{(t)}(x) f(x) dx)^T \Lambda (\int (x - \mu^{(t)}) g^{(t)}(x) f(x) dx)}{\int g^{(t)}(x) f(x) dx} =$$
$$\delta^T \Lambda \delta \int g^{(t)}(x) f(x)$$

Substituting this into the initial inequality in F we obtain:

$$F(\mu^{(t+1)}) \geq \int (1 + \frac{1}{2}\delta^T \Lambda \delta) g^{(t)}(x) f(x)$$

This concludes the proof:

$$F(\mu^{(t+1)}) \geq \int g^{(t)}(x) f(x) = F(\mu^{(t)})$$

A.8 Sketch of proof of Theorem 1, Conclusion b

It can be easily shown that the partial derivative $\frac{\partial F(\mu,\sigma)}{\partial \sigma} = 0$ when σ is a fixed point of the update step 2 of the theorem, and thus satisfies the equation $\sigma = \sqrt{\frac{1}{n} \frac{\int (\sum i=1^n (x_i - \mu_i)^2) g(x;\sigma) f(x) dx}{\int g(x;\sigma) f(x) dx}}$. Also, it is straightforward to check that the update step 2 of the theorem is taken in the direction of the gradient. Therefore, conclusion b will be satisfied if the partial derivative mentioned above is never 0 in the interval between $\sigma^{(t)}$ and $\sigma^{(t+1)}$. (Without loss of generality we can assume that $\sigma^{(t+1)} > \sigma^{(t)}$).

We give here the sketch of a proof by *reduction ad absurdum*. Let us assume that there exists $\sigma^* \in (\sigma^{(t)}, \sigma^{(t+1)})$ such that

$$\sigma^* = S(\sigma^*) = \sqrt{\frac{1}{n} \frac{\int (\sum_{i=1}^n (x_i - \mu_i)^2) g(x;\sigma^*) f(x) dx}{\int g(x;\sigma^*) f(x) dx}} \tag{A.7}$$

To simplify notations, we omit μ, which remains constant during this step. Here $S(\sigma)$ is basically the update function; it tells us which is the next sigma given σ. From the

assumption made it is clear that $S(\sigma^*) = \sigma^*$ and $S(\sigma^{(t)}) = \sigma^{(t+1)}$, while $\sigma^* \in (\sigma^{(t)}, \sigma^{(t+1)})$. It follows that:

$$\frac{S(\sigma^*) - S(\sigma^{(t)})}{\sigma^* - \sigma^{(t)}} = \frac{\sigma^* - \sigma^{(t+1)}}{\sigma^* - \sigma^{(t)}} < 0 \tag{A.8}$$

From the intermediate value theorem it follows that the derivative of S with respect to σ has to be negative somewhere inside $(\sigma^{(t)}, \sigma^{(t+1)})$. That means there exists a point σ_- where:

$$\int (\sum_{i=1}^{n} (x_i - \mu_i)^2)^2 g(x; \sigma_-) f(x) dx \int g(x; \sigma_-) f(x) dx - \\ (\int (\sum_{i=1}^{n} (x_i - \mu_i)^2) g(x; \sigma_-) f(x) dx)^2 < 0$$

But this is impossible by Cauchy-Schwarz inequality, which gives us the contradiction that concludes the proof.

References

[1] vasc.ri.cmu.edu/idb/html/motion/hotel/index.html.

[2] vasc.ri.cmu.edu/idb/html/motion/house/index.html.

[3] A. Z. A. Opelt, A. Pinz. A boundary-fragment-model for object detection. In *European Conference on Computer Vision*, 2006.

[4] F. S. A. Sanfeliu and R. Alquezar. Clustering of attributed graphs and unsupervised synthesis of function-described graphs. In *International Conference on Pattern Recognition*, 2000.

[5] P. K. Allen, A. Troccoli, B. Smith, S. Murray, I. Stamos, and M. Leordeanu. New methods for digital modeling of historic sites. *IEEE Journal of Computer Graphics and Applications*, 23(6):32–41, 2003.

[6] S. F. B. Duc and J. Bigun. Face authentication with gabor information on deformable graphs. *IEEE Transactions on Image Processing*, 8(4):504–516, 1999.

[7] F. Bach and M. Jordan. Learning spectral clustering. In *Neural Information Processing Systems*, 2003.

[8] J.-F. Baget and M.-L. Mugnier. Extensions of simple conceptual graphs: the complexity of rules and constraints. *Journal of Artificial Intelligence Research*, 16:425–465, 2001.

[9] M. Bakircioglu, U. Grenander, N. Khaneja, and M. Miller. Curve matching on brain surfaces using frenet distances. *Human Brain Mapping*, 6(5-6):329–333, 1998.

[10] L. Baratchart, M. Berthod, and L. Pottier. Optimization of positive generalized polynomials under lp constraints. *Journal of Convex Analysis*, 5(2):133, 1998.

[11] H. Barrow and R. Popplestone. Relational descriptions in picture processing. *Machine Intelligence*, 6(1):377–396, 1971.

[12] L. Baum and G. Sell. Growth transformations for functions on manifolds. *Pacific Journal of Mathematics*, 27(2):211–227, 1967.

[13] S. Belongie, J. Malik, and J. Puzicha. Shape context: A new descriptor for shape matching and object recognition. In *Neural Information Processing Systems*, 2000.

[14] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape context. *Pattern Analysis and Machine Intelligence*, 24(4):509–522, April 2002.

[15] A. Berg, T. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *International Conference on Computer Vision and Pattern Recognition*, 2005.

[16] M. Berthod, Z. Kato, S. Yu, and J. Zerubia. Bayesian image classification using markov random fields. *Image and Vision Computing*, 22(5):460–472, 1996.

[17] J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society*, 48(5):259–302, 1986.

[18] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, 1987.

[19] I. Bloch. Fuzzy relative position between objects in image processing: A morphological approach. *Pattern Analysis and Machine Intelligence*, 21(7):657–664, 1999.

[20] I. Bloch. On fuzzy distances and their use in image processing under imprecision. pattern recognition. *Pattern Recognition*, 32(11):18731895, 1999.

[21] C. Boeres. *Heursticas para Reconhecimento de Cenas por Correspondencia de Grafos*. PhD thesis, Universidade Federal do Rio de Janeiro, 2002.

[22] E. Borenstein, E. Sharon, and S. Ullman. Combining top-down and bottom-up segmentation. In *International Conference on Computer Vision and Pattern Recognition*, 2004.

[23] G. Bouchard and B. Triggs. Hierarchical part-based visual object categorization. In *International Conference on Computer Vision and Pattern Recognition*, 2005.

[24] K. Boyer and A. Kak. Structural stereopsis for 3d vision. In *Pattern Analysis and Machine Intelligence*, 1998.

[25] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence*, 23(11), 2001.

[26] A. Branca, E. Stella, and A. Distante. Qualitative scene interpretation using planar surfaces. volume 8, pages 129–139, 2000.

[27] N. Brixius and K. Anstreicher. Solving quadratic assignment problems using convex quadratic programming relaxations. *Optimization Methods and Software*, 16:49–68, 2000.

[28] H. Bunke. Error correcting graph matching: On the influence of the underlying cost function. *Pattern Analysis and Machine Intelligence*, 21(9):917–922, 1999.

[29] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19(3-4):255–259, 1998.

[30] T. Caetano, L. Cheng, Q. Le, and A. J. Smola. Learning graph matching. In *International Conference on Computer Vision*, 2007.

[31] M. Carcassoni and E. R. Hancock. Alignment using spectral clusters. In *British Machine Vision Conference*, 2002.

[32] G. Carneiro and D. Lowe. Sparse flexible models of local features. In *European Conference on Computer Vision*, 2006.

[33] H. Chen. Extracting knowledge from concept-based searching systems using conceptual graphs. Master's thesis, University of Guelph, 1996.

[34] Cheng, X. Wang, R. Collins, E. Riseman, and A. Hanson. Three-dimensional reconstruction of points and lines with unknown correspondence across images. *International Journal of Computer Vision*, 45(2):129–156, 2001.

[35] J. Cheng and M. J. Druzdzel. Ais-bn: An adaptive importance sampling algorithm for evidential reasoning in large bayesian networks. *Journal of Artificial Intelligence Research*, 13(1):155–188, 2000.

[36] W. J. Christmas, J. Kittler, and M. Petrou. Structural matching in computer vision using probabilistic relaxation. *Pattern Analysis and Machine Intelligence*, 17(8):749–764, 1995.

[37] M. Collins, R. Schapire, and Y. Singer. Logistic regression, adaboost and bregman distances. *Machine Learning*, 48(1-3):253–285, 2002.

[38] R. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *Pattern Analysis and Machine Intelligence*, 27(10):1631–1643, 2005.

[39] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[40] D. Comaniciu, V. Ramesh, and P. Meer. The variable bandwidth mean shift and data-driven scale selection. In *International Conference on Computer Vision*, 2001.

[41] T. Cour and J. Shi. Solving markov random fields with spectral relaxation. In *International Conference on Artificial Intelligence and Statistics*, 2007.

[42] T. Cour, J. Shi, and N. Gogin. Learning spectral graph segmentation. In *International Conference on Artificial Intelligence and Statistics*, 2005.

[43] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *Neural Information Processing Systems*, 2006.

[44] D. Crandall, P. Felzenszwalb, and D. Huttenlocher. Spatial priors for part-based recognition using statistical models. In *International Conference on Computer Vision and Pattern Recognition*, 2005.

[45] D. Crandall and D. Huttenlocher. Weakly supervised learning of part-based spatial models for visual object recognition. In *European Conference on Computer Vision*, 2006.

[46] A. D. J. Cross and E. R. Hancock. Graph matching with a dual-step em algorithm. *Pattern Analysis and Machine Intelligence*, 20(11):1236–1253, 1998.

[47] G. Csurka, C. Bray, C. Dance, and L. Fan. Visual categorization with bags of keypoints. In *ECCV Workshop on Statistical Learning in Computer Vision*, 2004.

[48] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H. Seidel, and S. Thrun. Performance capture from sparse multi-view video. In *SIGGRAPH*, 2008.

[49] C. di Ruberto and A. Dempster. Attributed skeleton graphs using mathematical morphology. *Electronic Letters*, 37(22):13251327, 2001.

[50] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatiotemporal features. In *International Conference on Computer Communications and Networks*, 2005.

[51] C. Dorai. *Cosmos: a Framework for Representation and Recognition of 3D Free-form Objects*. PhD thesis, Michigan State University, 1996.

[52] B. Duc, E. Bigun, J. Bigun, G. Maitre, and S. Fischer. Fusion of audio and video information for multi modal person authentication. *Pattern Recognition Letters*, 18(9):835–843, 1997.

[53] O. Duchenne, F. Bach, I. Kweon, and J. Ponce. A tensor-based algorithm for high-order graph matching. In *International Conference on Computer Vision and Pattern Recognition*, 2006.

[54] T. L. P. E. Grimson. Recognition and localization of overlapping parts from sparse data. Technical report, MIT, 1984.

[55] L. Emami. Conceptual browser: A concept-based knowledge extraction technique. Master's thesis, University of Guelph, 1997.

[56] M. Eshera and K. Fu. An image understanding system using attributed symbolic representation and inexact graph-matching. *J. Assoc. Computing Machinery*, 8(5):604–618, 1986.

[57] A. Etemadi, J.-P. Schmidt, G. Matas, J. Illingworth, , and J. Kittler. Low-level grouping of straight line segments. In *British Machine Vision Conference*, 1991.

[58] K. Fan, C. Liu, and Y. Wang. A randomized approach with geometric constraints to fingerprint verification. *Pattern Recognition*, 33(11):1793–1803, 2000.

[59] C. Fanti, L. Zelnik-Manor, and P. Perona. Hybrid models for human motion recognition. In *International Conference on Computer Vision and Pattern Recognition*, 2005.

[60] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 2004.

[61] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *International Conference on Computer Vision and Pattern Recognition*, 2008.

[62] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from googles image search. In *International Conference on Computer Vision*, 2005.

[63] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *International Conference on Computer Vision and Pattern Recognition*, 2003.

[64] R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *International Conference on Computer Vision and Pattern Recognition*, 2005.

[65] V. Ferrari, T. Tuytelaars, and L. V. Gool. Object detection by contour segment networks. In *European Conference on Computer Vision*, 2006.

[66] A. W. Finch, R. C. Wilson, and E. R. Hancock. Symbolic graph matching with the em algorithm. *Pattern Recognition*, 31(11):1777–1790, 1998.

[67] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM*, 24(6):381–395, 1981.

[68] M. Fischler and R. Elschlager. The representation and matching of pictorial structures. 22(1):67–92, 1973.

[69] D. Forsyth. Shape from texture without boundaries. In *European Conference on Computer Vision*, 2002.

[70] D. A. Forsyth and J. Ponce. *Computer Vision, a modern approach*. Prentice Hall, 2003.

[71] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

[72] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.

[73] M. Fritz and B. Schiele. Towards unsupervised discovery of visual categories. In *German Association for Pattern Recognition Symposium*, 2006.

[74] J. Gibson. *The Perception of the Visual World*. Houghton Mifflin, 1950.

[75] A. Gilbert, J. Illingworth, and R. Bowden. Scale invariant action recognition using compound features mined from dense spatiotemporal corners. In *European Conference on Computer Vision*, 2008.

[76] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. 18(4):377–388, 1996.

[77] K. Grauman and T. Darrell. Pyramid match kernels: Discriminative classification with sets of image features. In *International Conference on Computer Vision*, 2005.

[78] K. Grauman and T. Darrell. Unsupervised learning of categories from sets of partially matching image features. In *International Conference on Computer Vision and Pattern Recognition*, 2006.

[79] W. E. L. Grimson. *Object Recognition by Computer: The Role of Geometric Constraints*. MIT Press, 1990.

[80] L. Gu. *Recognizing Object Structures - A Bayesian Framework for Deformable Matching*. PhD thesis, Carnegie Mellon University, 2009.

[81] Q. C. H. Wu and M. Yachida. Face detection from color images using a fuzzy pattern matching method. *Pattern Analysis and Machine Intelligence*, 21(6):557–563, 1999.

[82] E. R. Hancock and J. Kittler. Edge-labeling using dictionary-based relaxation. *Pattern Analysis and Machine Intelligence*, 12(2):165–181, 1990.

[83] J. Hays, M. Leordeanu, A. Efros, and Y. Liu. Discovering texture regularity as a higher-order correspondence problem. In *European Conference on Computer Vision*, 2006.

[84] L. Herault, R. Horaud, F. Veillon, and J. Niez. Symbolic image matching by simulated annealing. In *British Machine Vision Conference*, 1990.

[85] R. Herpers and G. Sommer. Discrimination of facial regions based on dynamic grids of point representations. *International Journal of Pattern Recognition and Artificial Intelligence*, 12(4):381–405, 1998.

[86] B. Huet and E. R. Hancock. Shape recognition from large image libraries by inexact graph matching. *Pattern Recognition Letters*, 20(11-13):12591269, 1999.

[87] B. Huhle, P. Jenke, and W. Straer. On-the-fly scene acquisition with a handy multisensor system. *International Journal of Intelligent Systems Technologies and Applications*, 5(3-4):255–263, 2008.

[88] R. Hummel and S. Zucker. On the foundations of relaxation labeling processes. *Pattern Analysis and Machine Intelligence*, 5:267–287, 1983.

[89] J. S. Hwan. Content-based image retrieval using fuzzy multiple attribute relational graph. In *IEEE International Symposium on Industrial Electronics Proceedings*, 2001.

[90] P. Ifrah. Tree search and singular value decomposition: a comparison of two strategies for point-pattern matching. Master's thesis, McGill University, 1997.

[91] D. Jacobs. Robust and efficient detection of salient convex groups. *Pattern Analysis and Machine Intelligence*, 18(1):23–37, 1996.

[92] B. Julesz. Visual pattern discrimination. *IRE Transactions on Information Theory*, 8(2):84–92, 1962.

[93] D. Kanevsky. Extended baum transformations for general functions. In *Acoustics, Speech, and Signal Processing*, 2004.

[94] Y. Ke, R. Sukthankar, and M. Hebert. Event detection in crowded videos. In *International Conference on Computer Vision*, 2007.

[95] K. Khoo and P. Suganthan. Evaluation of genetic operators and solution representations for shape recognition by genetic algorithms. *Pattern Recognition Letters*, 23(13):1589–1597, 2002.

[96] G. Kim, C. Faloutsos, and M. Hebert. Unsupervised modeling and recognition of object categories with combination of visual contents and geometric similarity links. In *ACM International Conference on Multimedia Information Retrieval*, 2008.

[97] G. Kim, C. Faloutsos, and M. Hebert. Unsupervised modeling of object categories using link analysis techniques. In *International Conference on Computer Vision and Pattern Recognition*, 2008.

[98] J. Kittler, W. J. Christmas, and M. Petrou. Probabilistic relaxation for matching problems in computer vision. In *International Conference on Computer Vision*, 1993.

[99] P. Kohli, L. Ladicky, and P. Torr. Graph cuts for minimizing robust higher order potentials. In *International Conference on Computer Vision and Pattern Recognition*, 2008.

[100] P. Kohli and P. Torr. Effciently solving dynamic markov random fields using graph cuts. In *International Conference on Computer Vision*, 2005.

[101] C. Kotropoulos, A. Tefas, and I. Pitas. Morphological elastic graph matching applied to frontal face authentication under optimal and real conditions. In *International Conference on Multimedia Computing and Systems*, 1999.

[102] C. Kotropoulos, A. Tefas, and I. Pitas. Frontal face authentication using morphological elastic graph matching. *Transactions on Image Processing*, 9(4):555–560, 2000.

[103] J. Kubica, A. Moore, and J. Schneider. Finding underlying connections: A fast graph-based method for link analysis and collaboration querries. In *International Conference on Machine Learning*, 2003.

[104] J. Kubica, A. Moore, and J. Schneider. K-groups: Tractable group detection on large link data sets. In *International Conference on Data Mining*, 2003.

[105] M. Kumar, P. Torr, and A. Zisserman. Solving markov random fields using second order cone programming relaxations. In *International Conference on Computer Vision and Pattern Recognition*, 2006.

[106] S. Kumar. *Models for Learning Spatial Interactions in Natural Images for Context-Based Classification*. PhD thesis, The Robotics Institute, Carnegie Mellon University, 2005.

[107] S. Kumar and M. Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *International Conference on Computer Vision*, volume 2, pages 1150–1157, 2003.

[108] S. Kumar and M. Hebert. Discriminative random fields. *International Journal of Computer Vision*, 68(2):179–201, 2006.

[109] R. F. L. Fei-Fei and A. Torralba. Recognizing and learning object categories. In *Short Courses for CVPR*, 2007.

[110] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, 2001.

[111] I. Laptev and T. Lindeberg. Space-time interest points. In *International Conference on Computer Vision*, 2003.

[112] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *International Conference on Computer Vision and Pattern Recognition*, 2008.

[113] R. Lee and J. Liu. An automatic satellite interpretation of tropical cyclone patterns using elastic graph dynamic link model. *International Journal of Pattern Recognition and Artificial Intelligence*, 13(8):1251–1270, 1999.

[114] R. Lee and J. Liu. Tropical cyclone identification and tracking system using integrated neural oscillatory elastic graph matching and hybrid rbf network track mining techniques. *IEEE Transactions on Neural Networks*, 11(3):680–689, 2000.

[115] R. Lehoucq, D. Sorensen, and C. Yang. *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. SIAM Publications, 1998.

[116] M. Leordeanu, R. Collins, and M. Hebert. Unsupervised learning of object features from video sequences. In *International Conference on Computer Vision and Pattern Recognition*, 2005.

[117] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *International Conference on Computer Vision*, 2005.

[118] M. Leordeanu and M. Hebert. Efficient map approximation for dense energy functions. In *International Conference on Machine Learning*, 2006.

[119] M. Leordeanu and M. Hebert. Smoothing-based optimization. In *International Conference on Computer Vision and Pattern Recognition*, 2008.

[120] M. Leordeanu and M. Hebert. Unsupervised learning for graph matching. In *International Conference on Computer Vision and Pattern Recognition*, 2009.

[121] M. Leordeanu, M. Hebert, and R. Sukthankar. Beyond local appearance: Category recognition from pairwise interactions of simple features. In *International Conference on Computer Vision and Pattern Recognition*, 2007.

[122] T. Leung and J. Malik. Detecting, localizing and grouping repeated scene elements from an image. In *European Conference on Computer Vision*, 1996.

[123] S. Z. Li. *Markov Random Field Modeling in Computer Vision*. Springer, 1995.

[124] Y. Li, J. Sun, and C. T. H. Shum. Lazy snapping. In *SIGGRAPH*, 2004.

[125] T. Lindeberg. Scale-space behaviour of local extrema and blobs. *Journal of Mathematical Imaging and Vision*, 1(1):65–99, 1992.

[126] D. Liu and T. Chen. Semantic-shift for unsupervised object detection. In *CVPR Workshop on Beyond Patches*, 2006.

[127] D. Liu and T. Chen. Semantic-shift for unsupervised object detection. In *International Conference on Computer Vision and Pattern Recognition*, 2007.

[128] J. Liu, J. Luo, and M. Shah. Recognizing realistic actions from videos "in the wild". In *International Conference on Computer Vision and Pattern Recognition*, 2009.

[129] J. Liu, K. Ma, W. Cham, and M. Chang. Two-layer assignment method for online chinese character recognition. *IEEE Proceedings on Vision Image and Signal Processing*, 147(1):47–54, 2000.

[130] J. Liu and M. Shah. Learning human action via information maximization. In *International Conference on Computer Vision and Pattern Recognition*, 2008.

[131] A. Lobay and D. Forsyth. Recovering shape and irradiance maps from rich dense texton fields. In *International Conference on Computer Vision and Pattern Recognition*, 2004.

[132] M. Loog, J. J. Duistermaat, and L. M. J. Florack. On the behavior of spatial critical points under gaussian blurring. In *International Conference on Scale-Space and Morphology in Computer Vision*, 2001.

[133] D. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, Boston, 1985.

[134] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(4):91–110, 2004.

[135] B. Luo and E. R. Hancock. A robust eigendecomposition framework for inexact graph matching. In *International Conference on Image Analysis and Processing*, 2001.

[136] B. Luo and E. R. Hancock. Structural graph matching using the em algorithm and singular value decomposition. *Pattern Analysis and Machine Intelligence*, 23(10):1120–1136, 2001.

[137] M. Lyons, J. Budynek, and S. Akamatsu. Automatic classification of single facial images. *Pattern Analysis and Machine Intelligence*, 21(12):1357–1362, 1999.

[138] L. Y. Lyul and P. R. Hong. A surface-based approach to 3d object recognition using a mean field annealing neural network. *Pattern Recognition*, 35(2):299–316, 2002.

[139] R. C. W. M. Williams and E. R. Hancock. Deterministic search for relational graph matching. *Pattern Recognition*, 32(7):1255–1271, 1999.

[140] K. Ma and T. Xiaoou. Discrete wavelet face graph matching. In *International Conference on Image Processing*, 2001.

[141] J. Maciel and J. Costeira. A global solution to sparse correspondence problems. *Pattern Analysis and Machine Intelligence*, 25(2):187–199, 2003.

[142] J. Maciel and J. Costeira. A global solution to sparse correspondence problems. *Pattern Analysis and Machine Intelligence*, 25(2):187–199, 2003.

[143] S. Mahamud, L. R. Williams, K. K. Thornber, and K. Xu. Segmentation of multiple salient closed contours from real images. *Pattern Analysis and Machine Intelligence*, 25(4):433–444, 2003.

[144] J. Mairal, M. Leordeanu, F. Bach, M. Hebert, and J. Ponce. Discriminative sparse image models for class-specific edge detection and image interpretation. In *European Conference on Computer Vision*, 2008.

[145] D. Marr. *Vision*. Freeman, San Francisco, 1982.

[146] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *Pattern Analysis and Machine Intelligence*, 26(5):530–549, 2004.

[147] B. T. Messmer and H. Bunke. A decision tree approach to graph and subgraph isomorphism detection. *Pattern Recognition*, 32(12):1979–1998, 1999.

[148] N. Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335341, 1949.

[149] K. Mikolajczyk, B. Leibe, and B. Schiele. Multiple object class detection with a generative model, 2006.

[150] R. Mohan and R. Nevatia. Using perceptual organization to extract 3-d structures. *Pattern Analysis and Machine Intelligence*, 11(11):1121–1139, 1994.

[151] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. 1999.

[152] R. Myers and E. R. Hancock. Genetic algorithms for ambiguous labelling problems. *Pattern Recognition*, 33(4):685–704, 2000.

[153] A. Y. Ng, A. X. Zheng, and M. Jordan. Link analysis, eigenvectors and stability. In *Proc. International Joint Conference on Artificial Intelligence*, 2001.

[154] J. Niebles and L. Fei-Fei. A hierarchical model of shape and appearance for human action classification. In *International Conference on Computer Vision and Pattern Recognition*, 2007.

[155] A. Opelt, A. Pinz, M. Fussenegger, and P. Auer. Generic object recognition with boosting. *Pattern Analysis and Machine Intelligence*, 28(3):416–431, 2006.

[156] E. Osman, A. R. Pearce, M. Juettner, and I. Rentschler. Reconstructing mental object representations: A machine vision approach to human visual recognition. *Spatial Vision*, 13(2-3):277–86, 2000.

[157] V. Parameswaran and R. Chellappa. View invariance for human action recognition. *International Journal of Computer Vision*, 66(1):83–101, 2006.

[158] D. Parikh and T. Chen. Unsupervised identification of multiple objects of interest from multiple images: discover. In *Asian Conference on Computer Vision*, 2007.

[159] D. Parikh and T. Chen. Unsupervised learning of hierarchical semantics of objects (hsos). In *International Conference on Computer Vision and Pattern Recognition*, 2007.

[160] M. Pellilo. Replicator equations, maximal cliques, and graph isomorphism. In *Neural Computation*, volume 11, pages 1933–1955, 1999.

[161] A. Perchant and I. Bloch. A new definition for fuzzy attributed graph homomorphism with application to structural shape recognition in brain imaging. In *IEEE Instrumentation and Measurement Technology Conference*, 2000.

[162] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 1988.

[163] A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *Neural Information Processing Systems*, 2005.

[164] D. Ramanan and D. Forsyth. Using temporal coherence to build models of animals. In *International Conference on Computer Vision*, 2003.

[165] A. Rangarajan. Self annealing and self annihilation: Unifying deterministic annealing and relaxation labeling. *Pattern Recognition*, 33(4):635–649, 2000.

[166] A. Rangarajan, H. Chui, and E. Mjolsness. A relationship between spline-based deformable models and weighted graphs in non-rigid matching. In *International Conference on Computer Vision and Pattern Recognition*, 2001.

[167] P. Ravikumar and J. Lafferty. Quadratic programming relaxations for metric labeling and markov random field map estimation. In *International Conference on Machine Learning*, 2006.

[168] P. A. Regalia and E. Kofidis. The higher-order power method revisited: convergence proofs and effective initialization. In *ICASSP*, 2000.

[169] X. Ren. Learning and matching line aspects for articulated objects. In *International Conference on Computer Vision and Pattern Recognition*, 2007.

[170] D. Riviere, J. Mangin, D. Papadopoulos, J. Martinez, V. Frouin, and J. Regis. Automatic recognition of cortical sulci of the human brain using a congregation of neural networks. *Medical Image Analysis*, 6(2):77–92, 2002.

[171] D. Riviere, J. F. Mangin, D. Papadopoulos, J. M. Martinez, V. Frouin, and J. Regis. Automatic recognition of cortical sulci using a congregation of neural networks. In *Medical Image Computing and Computer-Assisted Intervention*, 2000.

[172] C. Rosenberg. *Semi-Supervised Training of Models for Appearance-Based Statistical Object Detection Methods*. PhD thesis, Carnegie Mellon University, 2004.

[173] G. Roth and M. Levine. Geometric primitive extraction using a genetic algorithm. *Pattern Analysis and Machine Intelligence*, 16(9):901–905, 1994.

[174] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, 2004.

[175] R. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999.

[176] A. Sanfeliu and K. S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Trans. Systems,Man, and Cybernetics*, 13(3):353–362, 1983.

[177] S. Sarkar and K. Boyer. Quantitative measures of change based on feature organization: Eigenvalues and eigenvectors. *Computer Vision and Image Understanding*, 71(1):110–136, 1998.

[178] S. Sarkar and P. Soundararajan. Supervised learning of large perceptual organization: Graph spectral partitioning and learning automata. In *Pattern Analysis and Machine Intelligence*, May 2000.

[179] F. Schaffalitzky and A. Zisserman. Geometric grouping of repeated elements within images. In *Shape, Contour and Grouping in Computer Vision*, 1999.

[180] C. Schellewald and C. Schnorr. Probabilistic subgraph matching based on convex relaxation. In *International Conference on. Energy Minimization Methods in Computer Vision and Pattern Recognition*, 2005.

[181] G. Scott and H. Longuet-Higgins. An algorithm for associating the features of 2 images. In *Royal Soc. London Series*, 1991.

[182] G. Scott and H. C. Longuet-Higgins. Feature grouping by relocalisation of eigenvectors of the proximity matrix. In *British Machine Vision Conference*, 1990.

[183] R. D. Shachter and M. A. Peot. Simulation approaches to general probabilistic inference on belief networks. In *Uncertainty in Artificial Intelligence*, 1989.

[184] L. Shams. *Development of Visual Shape Primitives*. PhD thesis, University of Southern California, 1999.

[185] Z. Shao and J. Kittler. Shape representation and recognition based on invariant unary and binary relations. *Image and Vision Computing*, 17(5-6):429–444, 1999.

[186] L. Shapiro and J. Brady. Feature-based correspondence - an eigenvector approach. *Image and Vision Computing*, 10(5):283–288, 1992.

[187] L. Shapiro and R. Haralick. A metric for comparing relational descriptions. *Pattern Analysis and Machine Intelligence*, 27(10):1845–1851, 1985.

[188] D. Sharvit, J. Chan, H. Tek, and B. Kimia. Symmetry-based indexing of image databases. *Journal of Visual Communication and Image Representation*, 9(4):366–380, 1998.

[189] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[190] A. Shokoufandeh, S. Dickinson, K. Siddiqi, and S. Zucker. Indexing using a spectral encoding of topological structure. In *International Conference on Computer Vision and Pattern Recognition*, 1999.

[191] M. Singh, A. Chatterjeeand, and S. Chaudhury. Matching structural shape descriptions using genetic algorithms. *Pattern Recognition*, 30(9):1451–1462, 2001.

[192] J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman. Discovering objects and their location in images. In *International Conference on Computer Vision*, 2005.

[193] J. Sivic and A. Zisserman. Video data mining using configurations of view-point invariant regions. In *International Conference on Computer Vision and Pattern Recognition*, 2004.

[194] M. Skomorowski. Use of random graph parsing for scene labelling by probabilistic relaxation. *Pattern Recognition Letters*, 20(9):949–956, 1999.

[195] C. Sminchisescu and B. Triggs. Fast mixing hyperdynamic sampling. *International Journal of Computer Vision*, 24(3):279–289, 2004.

[196] R. Sonthi. *The Definition and Recognition of Shape Features for Virtual Prototyping via Multiple Geometric Abstractions.* PhD thesis, The University of Wisconsin, Wisconsin, 1997.

[197] J. F. Sowa. Conceptual structures: Information processing in mind and machine. In *Addison-Wesley*, 1984.

[198] I. Stamos and M. Leordeanu. Automated feature-based range registration of urban scenes of large scale. In *International Conference on Computer Vision and Pattern Recognition*, 2003.

[199] I. Stamos and M. Leordeanu. Efficient model creation of large structures based on range segmentation. In *3D Data Processing, Visualization and Transmission*, 2004.

[200] G. Stewart and J.-G. Sun. Matrix perturbation theory. In *Academic Press*, 1990.

[201] P. Suganthan, E. Teoh, and D. Mital. Hopfield network with constraint parameter adaptation for overlapped shape recognition. *IEEE Transactions on Neural Networks*, 10(2):444–449, 1999.

[202] P. Suganthan and H. Yan. Recognition of handprinted chinese characters by constrained graph matching. *Image and Vision Computing*, 16(3):191–201, 1998.

[203] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields. In *European Conference on Computer Vision*, 2006.

[204] M. Szummer, P., Kohli, and D. Hoiem. Learning crfs using graph cuts. In *European Conference on Computer Vision*, 2008.

[205] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Neural Information Processing Systems*, 2004.

[206] A. Tefas, C. Kotropoulos, and I. Pitas. Using support vector machines to enhance the performance of elastic graph matching for frontal face authentication. volume 23, pages 735–746, 2001.

[207] A. Tefas, C. Kotropoulos, and I. Pitas. Face verification using elastic graph matching based on morphological signal decomposition. *Signal Processing*, 82(6):833–851, 2002.

[208] S. Tirthapura, D. Sharvit, P. Klein, and B. Kimia. Indexing based on edit-distance matching of shape graphs. In *Multimedia Storage and Archiving Systems*, 1998.

[209] S. Todorovic and N. Ahuja. Extracting subimages of an unknown category from a set of images. In *International Conference on Computer Vision and Pattern Recognition*, 2006.

[210] P. Torr. Solving markov random fields using semi definite programming. In *AIS*, 2003.

[211] P. Torr. Solving markov random fields using semidefinite programming. In *International Conference on Artificial Intelligence and Statistics*, 2003.

[212] J. Triesch and C. von der Malsburg. A system for person-independent hand posture recognition against complex backgrounds. *Pattern Analysis and Machine Intelligence*, 23(12):1449–1453, 2001.

[213] Tu and S. C. Zhu. Image segmentation by data driven markov chain monte carlo. *Pattern Analysis and Machine Intelligence*, 24(5):657–673, 2002.

[214] Tu and S. C. Zhu. Image parsing: unifying segmentation, detection and recognition. *International Journal of Computer Vision*, 63(2):113–140, 2005.

[215] A. Turina, T. Tuytelaars, and L. Gool. Efficient grouping under perspective skew. In *International Conference on Computer Vision and Pattern Recognition*, 2001.

[216] M. Turner and J. Austin. Graph matching by neural relaxation. *Neural Computing and Applications*, 7(3):238–248, 1998.

[217] J. Ullman. An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42, 1976.

[218] S. Umeyama. An eigen decomposition approach to weighted graph matching problems. In *Pattern Analysis and Machine Intelligence*, 1988.

[219] F. J. C. S. V. Ferrari, L. Fevrier. Groups of adjacent contour segments for object detection. Technical report, INRIA, 2006.

[220] G. Wahba. Spline models of observational data. In *SIAM Publications*, 1990.

[221] M. Wainwright, T. Jaakkola, and A. Willsky. Map estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches. In *IEEE Transactions on Information Theory*, 2005.

[222] D. Weinland, E. Boyer, and R. Ronfard. Action recognition from arbitrary views using 3d examplars. In *International Conference on Computer Vision*, 2007.

[223] R. C. Wilson and E. R. Hancock. Bayesian compatibility model for graph matching. In *Pattern Recognition Letters*, volume 17, pages 263–276, 1996.

[224] R. C. Wilson and E. R. Hancock. Structural matching by discrete relaxation. *Pattern Analysis and Machine Intelligence*, 19(6):634–648, 1997.

[225] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *International Conference on Computer Vision*, 2005.

[226] L. Wiskott. The role of topographical constraints in face recognition. *Pattern Recognition Letters*, 20(1):89–96, 1999.

[227] A. Wong and M. You. Entropy and distance of random graphs with application to structural pattern recognition. In *Pattern Analysis and Machine Intelligence*, 1985.

[228] S. Wong, T. Kim, and R. Cipolla. Learning motion categories using both semantics and structural information. In *International Conference on Computer Vision and Pattern Recognition*, 2007.

[229] Z. Yin and R. Collins. On-the-fly object modeling while tracking. In *International Conference on Computer Vision and Pattern Recognition*, 2007.

[230] J. Yuan, Z. Liu, and Y. Wu. Discriminative sub volume searches for efficient action detection. In *International Conference on Computer Vision and Pattern Recognition*, 2009.

[231] A. Yuille and A. Rangarajan. The concave-convex procedure (cccp). In *Neural Computation*, 2003.

[232] M. Zaslavskiy, F. Bach, and J.-P. Vert. A path following algorithm for graph matching. In *International Conference on Image and Signal Processing*, 2008.

[233] R. Zass and A. Shashua. Probabilistic graph and hypergraph matching. In *International Conference on Computer Vision and Pattern Recognition*, 2008.

[234] Y. Zhang. Solving large-scale linear programs by interior-point methods under the matlab environment. In *Technical Report TR96-01, Department of Mathematics and Statistics, University of Maryland*, 1995.