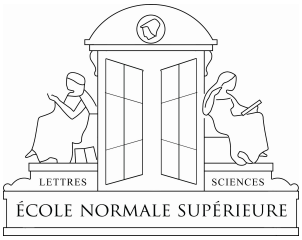


THÈSE DE DOCTORAT

présentée par **OLIVIER DUCHENNE**

pour obtenir le grade de
DOCTEUR DE L'ÉCOLE NORMALE SUPÉRIEURE

Domaine: **MATHÉMATIQUES APPLIQUÉES**

Sujet de la thèse:

Alignement élastique d'images pour la reconnaissance d'objet

—

Non-rigid image alignment for object recognition

Thèse présentée et soutenue à l'ENS Ulm le
29 Novembre 2012 devant le jury composé de:

Jean Ponce	Professeur, Directeur du DI, ENS Ulm	Directeur de thèse
Pedro Felzenszwalb	Professeur, Brown University	Rapporteur
Martial Hebert	Professeur, Carnegie Mellon University	Rapporteur
Francis Bach	Directeur de recherche, ENS Ulm	Éxaminateur
Jitendra Malik	Professeur, University of Berkeley	Éxaminateur
Cordelia Schmid	Professeur, INP Grenoble	Éxaminateur
Andrew Zisserman	Professeur, University of Oxford	Éxaminateur

---

Thèse préparée au sein de l' équipe WILLOW du département d'informatique de l'École Normale Supérieure, Ulm. (INRIA/ENS/CNRS UMR 8548).

Contents

Acknowledgment

My first thanks and thoughts go to my wife Soowon. She is my life and happiness. She has been supporting me during my years of Ph.D. Her smile gives me strength in down times.

I thank my parents Véronique and François Duchenne for giving me life, love, education, and my wonderful little sister Cécile.

I thank my Ph.D. advisor Prof. Jean Ponce. He has done an amazing achievement, building the Willow lab from scratch in a few years, and attracting to ENS half-a-dozen world first-class researchers. He has introduced me to computer vision, which has convinced me to switch from the undergrad Physics department to do Computer Science. He provides very nice research environment to the lab members. He has given me many great advices.

I thank the members of my thesis committee Dr Francis Bach, Prof. Pedro Felzenszwalb, Prof. Martial Hebert, Prof. Jitendra Malik, Prof. Jean Ponce, Prof. Cordelia Schmid, and Prof. Andrew Zisserman for their great expertise and precious time that they use to evaluate my work.

I thank Prof. Alyosha Efros, our part-time Willow lab member and my future boss, as a post-doc. He has transmitted to me his excitement and passion for solving vision. He has shared with me his unusual, original and unexpected points of view and ideas which encouraged me to think differently about my research.

I thank all the members of the Willow Lab and Sierra Team. They have all influenced me deeply and helped me improve my way of doing research. Especially Dr Ivan Laptev, Dr Josef Sivic and Dr Francis Bach, Bryan Russell for his communicative passion and enthusiasm for vision, my co-author Armand Joulin, Y-Lan Boureau, Warith Harchaoui, Mikel Rodriguez, Julien Mairal, Rodolph Jenatton, Guillaume Seguin, Florent Couzinie, Neva Cherniavsky, Guillaume Obozinski, Karteek Alahari, Vincent Delaitre.

I also thank Prof. Jitendra Malik from UC Bekeley and his lab members Chunhui Gu, Subhransu Maji, Chetan Nandakumar, and Pablo Arbelaez. I have spent 7 months with them, and learned what research is. Thanks to them through an intensive reading group, I learned very quickly a big chunk of the literature of our field. I got to know the American way of doing research: their passion for science

deeply impressed me and influenced me.

I thank Prof. Kweon Inso from KAIST in Korea who has welcomed me 8 months in his lab, and his hard-working lab members, Hwang YoungBae, Choi Ouk, Ju Hanbyeol, Jean-Charles Bazin, and all others.

I thank Kai Yu, Timothée Cour and the NEC Labs engineers and interns. There, I have learned a more pragmatic way of doing science.

Of course, solving Computer Vision is a collaborative work, and I thank all the great professors and students around the world, especially those whom my work is based on.

Part of this work has been supported by the ERC Video World grant.

Chapter 1

Introduction

1.1 Background

Seeing allows animals and people alike to gather information from a distance, often with high spatial and temporal resolution. In the animal world, many species use vision as their primary way of finding preys and food, localizing predators and potential dangers, and recognizing familiar places and other animals. Since the first seeing animals appeared 300 million years ago [Carroll, 1988], the fierce competition for survival has been so unfair to the other species that few of them remain today, and that seeing animals now represent 96% of all species [Land and Fernald, 1992], from mammals to crustaceans and spiders. As a result, the visual processing capabilities of animals have evolved tremendously to maximize the information that they can extract from the world surrounding them.

The motivation for studying computer vision is similar: endowing machines with the capability of visual perception to help them interact with the physical world and gather information to make decisions. By now, machines have shown an extraordinary ability to quickly and flawlessly handle well-organized and formatted data. However, they remain mostly incapable of understanding real-world noisy data including images and videos. Therefore autonomous robots mainly stay in factories where everything happens in a carefully predefined manner. In order to interact with people and objects in a real-world environment, they need vision.

However, vision is complicated: macaques use 60 per cent of their brain for visual perception [Vanduffel et al., 2002], and people use even more in terms of absolute size. This makes vision essentially effortless for us; over millions of years we have developed unconscious, fast and accurate networks of neurons to perform this primordial task. Therefore, people can see without even noticing how complex and difficult this task is, and unlike many other computer science fields, computer vision is still far behind human-level performance.

1.2 Motivation and aim of study

Computer vision includes many sub-fields, such as 3D reconstruction, action recognition, scene understanding, image retrieval, and so on. In this thesis, we are more specifically interested in finding a way to compare two images despite variations in the position of local parts, with the goal of building a category-level object recognition and detection system. We want to develop algorithms that can find out whether or not an object of a given category (e.g., car, person, shoe) is present in a given image (categorization) and where it is located (detection). Although this task has already received a lot of attention from the scientific community, it still remains very challenging. Our approach is based on image alignment, as explained later in section 1.2.4.

1.2.1 Building a category-level object detector

Suppose we could store *all possible images* of a given category (for instance, cars). Let us call these images the prototypes. Once a test image is given, we could automatically compare this image to all our prototypes. If one prototype is *exactly* equal to the test image, the algorithm outputs that the test image contains an object of the same category than the prototype.

Of course, the number of possible images of a given category is enormous, and it is not even nearly possible to collect them all, to store them all, and to compare a test image with all of them.

One more feasible way to build such a detector is to store a reasonable amount of prototypes. In this case, no test image is *exactly* equal to any prototype. So the

algorithm needs to measure the *similarity* between the test image and the prototype. If, according to this measure, the test image is similar enough to one of the prototypes, they represent the same category.

One might decompose such a method in three main tasks:

- Picking or making good prototypes that span the set of possible objects in the category, and are dissimilar to objects outside of the category.
- Building a similarity measure that scores high to pairs of images of the same category, and that scores low to other pairs.
- Deciding from the similarity measurements between the test image and the prototypes whether or not it contains an object of the given category.

This thesis focuses on the second point: the construction of a similarity measure.

1.2.2 Aligning to build a similarity measure

The misalignment issue. We would like to build a similarity measure to help us determine whether or not two given images are from the same category. As an example, let us consider the two toy images of cars in Figure 1.1. They correspond to different viewpoints, and their shapes are also dissimilar (intra-class variation). They have, however, many common features (headlights, side mirrors, wheels, doors, ...) that we might be able to rely on to assess their similarity. Unfortunately, we do not know, a priori, which part of the first image corresponds to which part in the second image. This is the problem of misalignment.

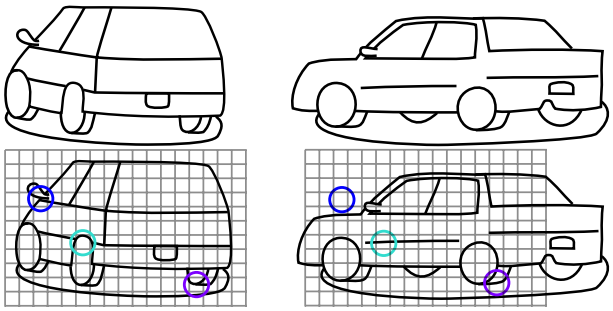


Figure 1.1: Two unaligned cars. Circles at fixed positions do not match.

In this section, we try to keep a generic definition of he word "part", which just have to be locally delimited in the image.

Feature matching. One common approach to handle that misalignment problem is to match each part of the first image to the part in the second image which has the most similar local appearance. In addition, in the bag-of-word approach [Sivic and Zisserman, 2003; Lazebnik et al., 2003; Csurka et al., 2004], the local appearances are discretized in a finite set, allowing this type of similarity measures to be computed via a simple histogram comparison, and therefore making it really fast to compute.

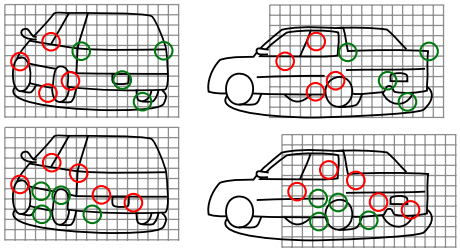
The drawback of such techniques is that they do not use the *position* of each part: a part in the first image can be matched anywhere in the second one. In other word, they waste the geometric information, such that any shuffled versions of two images would have the same similarity measure.

Regular grids. One usual approach [Dalal and Triggs, 2005a; Lazebnik et al., 2006a] to take advantage of the geometric information is to hypothesize that corresponding parts are roughly at the same position in both images. One way to visualize that technique is to draw a grid on top of the two cars (Figure 1.1). The algorithm compares each cell of the grid of the first image with the cell of the grid of the second image at the same position. However, the same problem arises at a smaller scale: we still do not know how to align elements inside each cell. To tackle that, one can typically use the feature matching approach, described in the previous paragraph. The loss of geometric information is just local, and so is not as serious as in the pure bag-of-word approach.

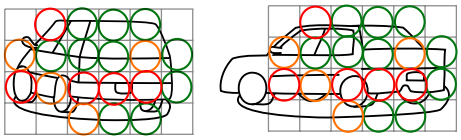
In the example of Figure 1.1, one can see that the cells at the same position on both images (colored circles) do not match. As a consequence, a similarity measure based on the sum of the local similarities of cells at fixed positions would not work in this example.

However, two commonly used techniques can make it practical:

(1) Using the sliding window approach (Figure 1.2(a)) that consists in comparing the prototypes with translated (and scaled) versions of the test image. It is equivalent to moving the grid in the test image. We can see in Figure 1.2(a) that, for specific choices of translation, it is possible to match many cells (green circles), but unfortunately there is no translation for which all cells match at the same time.



(a) Sliding the right grid with different translations allows to match part of the cells (green) but not all simulatenously (red).



(b) Increasing the cell size improves tolerance to miss-alignment.

(2) Increasing the size of the cells [Lazebnik et al., 2006a]. We can see on figure 1.2(b) that this may allow for many good matches. One extreme example of such a technique is the bag-of-visual-words approach [Sivic and Zisserman, 2003; Lazebnik et al., 2003; Csurka et al., 2004] described in the previous paragraph, where each grid has only one cell. Unfortunately, this kind of method loses all geometric information inside each cell. So the bigger the cells are, the most information is lost.

1.2.3 A direct approach to the misalignment problem

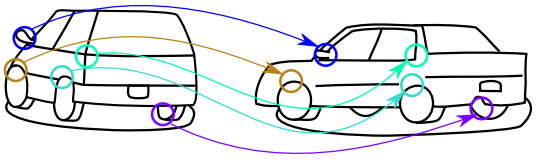


Figure 1.2: In this work, we try to directly find correspondances.

In this thesis, we use a much more direct approach to address the misalignment issue. Our algorithms try to find correspondences between parts in the first and second image (Figure 1.2), while keeping the geometric relationship between them. This is a hard task that requires more sophisticated techniques.

We design a measure of alignment quality, and we define the similarity of the images as the quality of the best possible alignment between them. To compute this quality, we rely on two types of clues: (1) matching parts should have similar local appearance, and (2) the set of correspondences (matches) should satisfy some geometric constraints:

Local appearance. Matching parts should have similar local appearance. This can be easier to estimate than global similarity because, usually, local deformations are less severe than global ones (Figure 1.3). So, we can use a method that assumes local rigidity (or local affine rigidity, since perspective transformation can be assumed locally affine). One can also use local deformation-invariant features. This loses much less exploitable information than global deformation-invariant features.

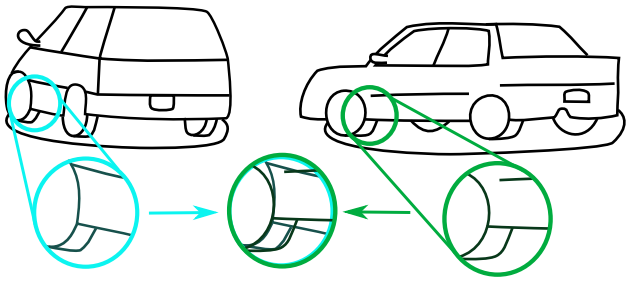


Figure 1.3: Even though these two cars are impossible to align globally and rigidly, it is possible to do so locally. For instance the two circled wheels are superimposed in this figure to show that they align well.

Geometric constraints. Using local appearance only, one would just match each part of the first image to the most similar part in the second image. This is not satisfying because the rich geometrical relationship between parts is ignored. In other words, any spatially shuffled version of the images would give the same similarity measurement. So one usually applies geometric constraints to the matching process. As explained in the two next sections, one can hypothesize a parametric rigid transform, or a deformable alignment.

1.2.4 Aligning rigid objects

Alignment methods were proposed in the 1980s to detect *exact replicates* of a prototype [Faugeras and Hebert, 1983; Grimson and Lozano-Perez, 1984; Ayache and Faugeras, 1986]. For instance, one can see the results of [Huttenlocher and Ullman, 1987a] in Figure 1.4.

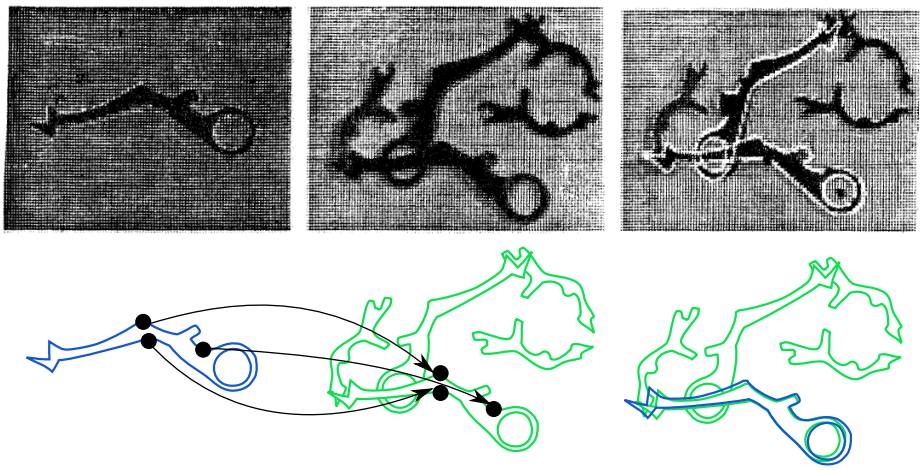


Figure 1.4: The algorithm of [Huttenlocher and Ullman, 1987a] searches for the prototype (left) in the test image (center) by looking for correspondences, generating transformation hypotheses. Then, each hypothesized transformation is verified (right) by transforming the prototype in the test image (right), and checking that they match well, for each.

In such a situation, the alignment problem is reduced to finding the right rotation and translation from the 3D or 2D prototype to the 2D test image. Such transformations are parametric (typically with up to 6 parameters). So one just has to find *a few correct correspondences* between points in the prototype and test image to estimate the parameters, and, thus, get the transformation *for all points*. Of course it is not easy to find even a few accurate correspondences.

Finding good correspondences. More recently, the Scale Invariant Feature Transform [Lowe, 2004b] (also known as SIFT) has become the standard tool to find accurate correspondence candidates in the rigid transformation case. First, it extracts repeatable points in both images (i.e. points which are likely to be detected on the same position of objects independently of their viewpoint, sacle,...). Then it computes scale-and-rotation-invariant descriptors of them, and finally generates correspondence candidates between feature points which have similar descriptors and have no other ambiguous possible match.

Selecting coherent correspondences. However the generated correspondence candidates typically contain many outliers. To sort them out, the popular Random sample consensus [Fischler and Bolles, 1981] (also known as RANSAC) randomly samples a few of them which are used to estimate the parameters of the transformation. Then, it checks how many other correspondence candidates agree with this transformation (inliers). This is iterated many times, and RANSAC picks the transformation which contains the most inliers.

Those techniques combined have been very successful at matching exact same objects, especially when they are flat and highly-textured, and also at matching images of a scene seen from different viewpoints. This has led to many commercial systems (for instance, for 3D reconstruction, dvd/cd/book cover recognition, logo detection, etc.) (e.g. Google Goggles[1] , Acute3D[2],LTU[3],...).

1.2.5 Aligning deformable objects.

However in this thesis we are interested in recognizing object categories (such as cars, dogs, houses). In that case, there is no obvious parametrization of the transformation between two images of the same category (due to intra-class variation).

In this situation, it is not possible to compute the global alignment from the prototype to the test image based on *a few* correct correspondences only. Therefore, we cannot apply the techniques described in the previous section.

[1] `http://www.google.com/mobile/goggles/`
[2] `http://www.acute3d.com/`
[3] `http://www.ltutech.com`

Constraining possible alignments. As seen in the previous section, when aligning rigid objects, the set of possible alignments is rather small / low-dimensional. On the contrary, here, each part of the prototype can potentially match anywhere in the test image. But still not all the alignments are equally desirable. As an example, if the test image is a shuffled version of the prototype, every part of prototype does have a match in the test image, but, the alignment field is highly discontinuous. In contrast, we want to encourage more natural alignments with smooth deformation fields, where two *nearby* parts of the prototype have *nearby* matching parts in the test image.

From prototypes to models. Another problem compared to the rigid case is that it is possible that, due to severe intra-class and viewpoint variations, two images from the same category look very different, and therefore we cannot find a good alignment between them. We can only hope that a test image would be similar to at least one of the stored prototypes. So, as explained in section 1.2.1, we need a set of prototypes with the following ideal properties: (1) All the possible images of the given category have a high similarity with at least one prototype. (2) All images not from this category have low similarity with all prototypes. (3) The number of prototypes is small to reduce the computation time.

One can consider that actual images are not the best fit to satisfy those properties. Many works use *models* rather than prototypes which are more abstract representations of objects from the same category. For instance, in the field of psychology, according to the theory of [Rosch, 1973], the human brain, in order to recognize dogs, do not actually store many images of dogs that it has seen by the past. Rather, it stores a model of an "average dog", and hierarchically, other average models for each dog race, and so on. Examples of model-based implementations are described in the next paragraph.

Influential past works. The early work Pictorial Structure [Fischler and Elschlager, 1973a] uses a hand-designed model (see Figure 1.5) in which a face is decomposed into several supposedly rigid parts (eyes, mouth, ears,...) connected by stretchable springs. Their algorithm tries to align this model to a given test image, while the springs penalize deformation of the relative positions between parts. This work has introduced the concept of an abstract deformable model that can be aligned to test images. It has been a long source of inspiration: for instance, [Yuille, 1991] extends it by replacing the rigid parts by hand-designed deformable templates (see Figure 1.6).

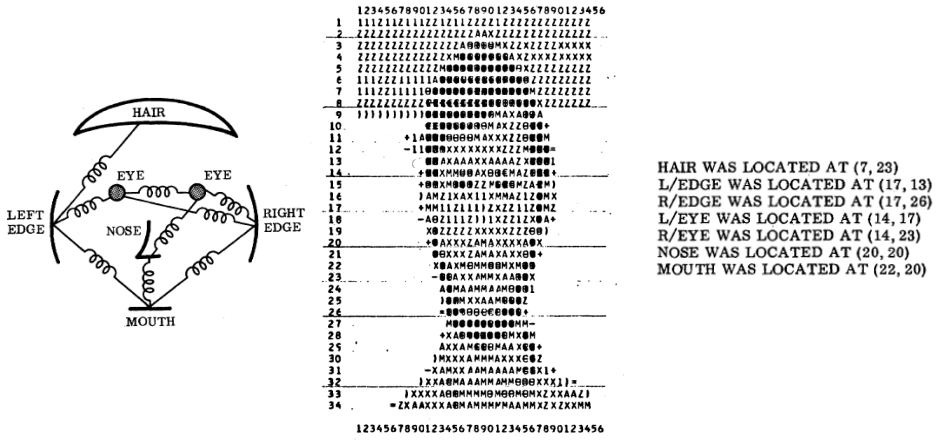


Figure 1.5: [Fischler and Elschlager, 1973a] match their hand-designed model (left) to the test image (center). The result (right) is the positions of the sub-parts.

In their work, [Lades et al., 1993] extract parts on a regular grid and match prototypes of faces or office objects to the test images to detect these categories (see Figure 1.7). Although it is intellectually satisfying, in theory, to have a few "name-able" parts (ears,mouth,...), [Lades et al., 1993] have shown that rising the number of parts leads to better performance, even though they lose their meaning.

Following that trend, and after the success of the key point detectors [Schmid and Mohr, 1997; Lowe, 2004b], which, in theory, localize interesting and repeatable points in images, several authors [Berg et al., 2005a; Leordeanu and Hebert, 2005a] have successfully developed algorithms that match hundreds of interest points from prototypes to test images in order to measure their similarity. Then,

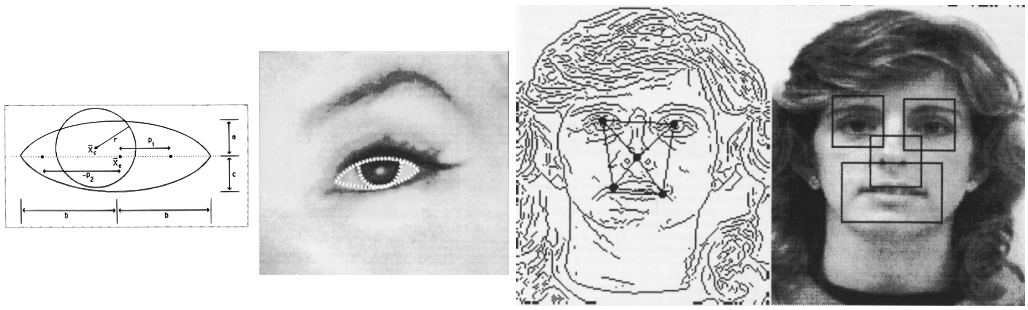


Figure 1.6: [Yuille, 1991] Hand-designed eye model (deftemplates91). Eye model to eye image alignment (center-left). Face model to face image alignment (center-right and right).

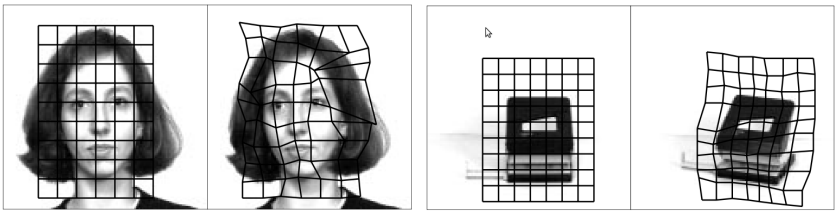


Figure 1.7: [Lades et al., 1993] align grids on prototypes to test images in order to detect object categories.

each test image is assigned to the category of its closest prototype.

In parallel, researchers (e.g. [Fergus et al., 2007; Felzenszwalb et al., 2008b; Leordeanu et al., 2007]) have used machine learning techniques to learn deformable models from training data. These algorithms try to produce models which are similar to training images of the same category and dissimilar to others. The most successful approach is [Felzenszwalb et al., 2008b] (see Figure 1.8) that nowadays produce state-of-the art results on standard detection datasets.

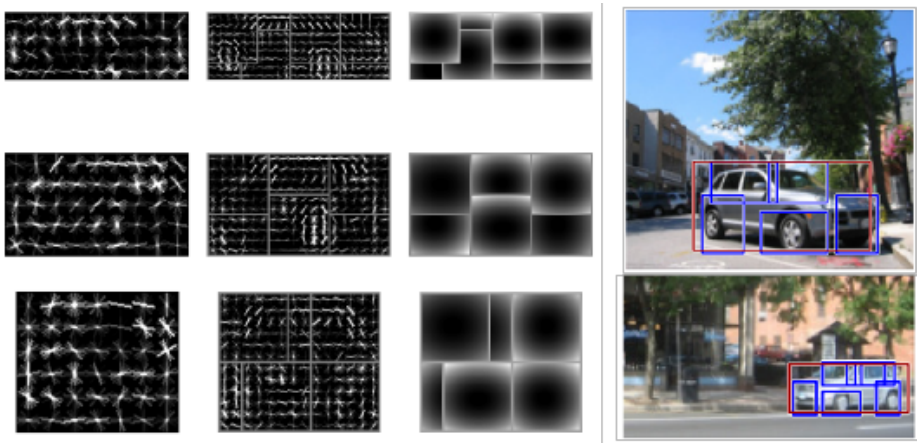


Figure 1.8: [Felzenszwalb et al., 2008b] have produced a state-of-the-art object detector based on a deformable models trained from data. First column: learned HOG root model. Second column: learned parts. Third row: learned deformation field for each part. Last column: detection example, red rectangle for root, blue for parts.

1.3 Graph matching

In the previous section, we have described why we want to align images. In this section, we review graph matching, which is a practical method to perform alignment, and which is the main theme of this thesis.

Graph-matching techniques represent an image with a graph. Typically, the nodes of this graph represent local regions of the image, and its edges represent the geometric relationships between nodes (more concrete examples are described later).

The alignment of two images is reduced to the matching of two graphs. Let us define a node match as a pair of nodes one from each graph, and a graph match as a set of node matches. So instead of finding a generic alignment, one "just" has to find a graph match. Since the graph is a simplified representation of the image, this reduces the complexity of the problem.

A graph matching algorithm tries to find the graph match which maximizes a predefined quality measure (or objective function). This quality measure has typically two parts:

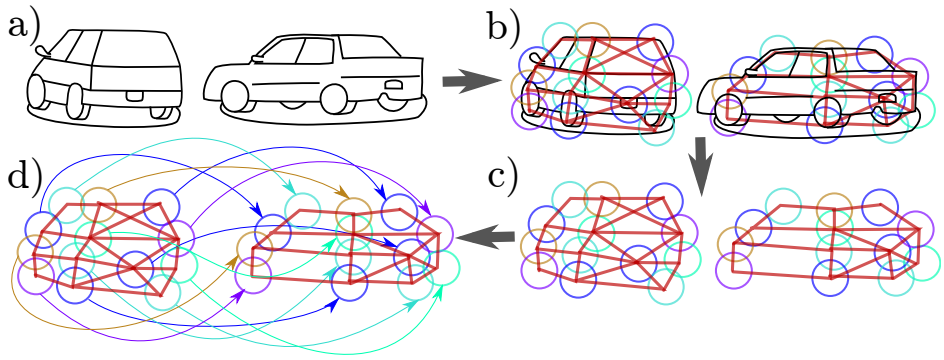


Figure 1.9: Given two images to be compared (a), the graph matching approach extracts one graph for each image (b), and forgets about the image except the node and edge descriptors (c), then it matches the two graphs (d), the arrows represent node matches.

- A unary part, which encourages nodes of the first graph to be matched to similar nodes, e.g. nodes corresponding to similar local regions.
- A binary part, which encourages pairs of node of the first graph to be matched to pairs of node of the second graph, with similar geometrical relationship (see Figure 1.10 and caption).

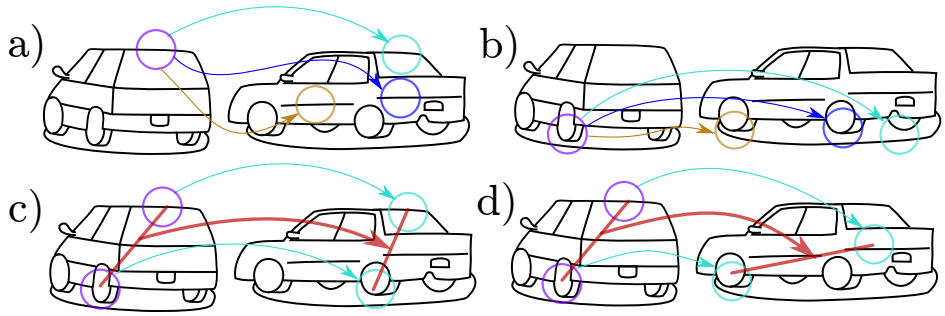


Figure 1.10: Graph matching encourages the matching of nodes with similar local descriptors (e.g. (a) and (b)). However, one node in the first graph might have several good matching nodes in the second graph (e.g. (a) and (b)). To cope with that kind of local ambiguities, graph matching encourages that edges (in red) in the first graph match to edges with similar descriptors in the second graph. This is the case in (c), but not in (d).

One concrete method to go graph matching is described in Section 2.2.2.

1.4 Objective and contributions of the thesis

Until now, we have seen some of the motivations to do computer vision. We have described the task of category-level object recognition/detection, and have explained how the design of a good similarity measure can be useful to perform that task, and how we can use image alignment to build this similarity measure. We have also described some previous work doing alignment. Finally, we have described graph matching, which is a concrete technique to perform alignment (see Figure 1.11).

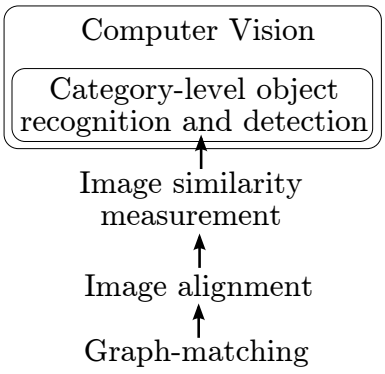


Figure 1.11: Summary of the introduction. The arrow means "can be an appropriate tool for".

The objective of this thesis is to explore the use of graph matching in object recognition systems. In the continuity of the previously described articles, rather than using descriptors invariant to misalignment, this work directly tries to find explicit correspondences between prototypes and test images, in order to build a robust similarity measure and infer the class of the test images.

In chapter 2, we will present a method that given interest points in two images tries to find correspondences between them. It extends previous graph matching approaches [Leordeanu and Hebert, 2005a] to handle interactions between more than two feature correspondences. This allows us to build a more discriminative and/or more invariant matching method. The main contributions of this chapter are:

- The introduction of an high-order objective function for hyper-graph matching (Section 2.3.1).

- The application of the tensor power iteration method to the high-order matching task, combined with a relaxation based on constraints on the row norms of assignment matrices, which is tighter than previous methods (Section 2.3.1),

- An ℓ^1-norm instead of the classical ℓ^2-norm relaxation, that provides solutions that are more interpretable but still allows an efficient power iteration algorithm (Section 2.3.5).

- The design of appropriate similarity measures that can be chosen either to improve the invariance of matching, or to improve the expressivity of the model (Section 2.3.6).

- The proposed approach has been implemented, and it is compared to state-of-the-art algorithms on both synthetic and real data. As shown by our experiments (Section 2.5), our implementation is, overall, as fast as these methods in spite of the higher complexity of the model, with better accuracy on standard databases.

In chapter 3, we build a graph-matching method for object categorization. The main contributions of this chapter are:

- Generalizing [Caputo and Jie, 2009; Wallraven et al., 2003], we propose in Section 3.3 to use the optimum value of the graph-matching problem associated with two images as a (non positive definite) kernel, suitable for SVM classification.

- We propose in Section 3.4 a novel extension of Ishikawa's method [Ishikawa, 2003] for optimizing MRFs which is orders of magnitude faster than competing algorithms (e.g., [Kim and Grauman, 2010; Kolmogorov and Zabih, 2004; Leordeanu and Hebert, 2005a]) for the grids with a few hundred nodes considered in this article). In turn, this allows us to combine our kernel with SVMs in image classification tasks.

- We demonstrate in Section 3.5 through experiments with standard benchmarks (Caltech 101, Caltech 256, and Scenes datasets) that our method matches and in some cases exceeds the state of the art for methods using a single type of features.

In chapter 4, we introduce our work about object detection that perform fast image alignment. The main contributions of this chapter are:

- We propose a novel image similarity measure that allows for arbitrary deformations of the image pattern within some given disparity range and can be evaluated very efficiently [Lemire, 2006], with a cost equal to a small constant times that of correlation in a sliding-window mode.

- Our similarity measure relies on a hierarchical notion of parts based on simple rectangular image primitives and HOG cells [Dalal and Triggs, 2005a], and does not require manual part specification [Felzenszwalb and Huttenlocher, 2005b; Bourdev and Malik, 2009; Felzenszwalb et al., 2010] or automated discovery [Lazebnik et al., 2005; Kushal et al., 2007].

Chapter 2

A tensorial formulation for hyper-graph matching

2.1 Introduction

2.1.1 Motivation and goals

We have reviewed in Chapter 1, some motivations behind graph matching. In this chapter, we present a pure graph-matching contribution, with a few experiments to motivate the use of this method. Applications of graph matching to object detection and recognition are presented in the following chapters.

The motivation behind the work presented here is that the type of geometric constraints that one can use in graph-matching might be restricted. To tackle this, we propose an hyper-graph matching technique, that can handle relationship with more than two points at a time. Typically, this allows our matching to be invariant to more classes of transformation and/or to have a richer and more discriminative description of the geometrical relationship between nodes.

As noted earlier, finding correspondences between visual features (such as interest points, edges, or even raw pixels) is a key problem in many computer vision tasks. The simplest approach to this problem is to define some measure of similarity between two features (e.g., the Euclidean distance between SIFT descriptors of small image patches [Lowe, 2004a]), and match each feature in the first image

Method name	order	number of nodes per descriptor
Feature matching	1^{st}	1
Graph Matching	2^{nd}	$1-2$
Hypergraph matching	$3^{rd}+$	any

Figure 2.1: Matching methods

to its nearest neighbor in the second one, and finally sum all the local similarities to compute the global one. This naive first-order approach is called feature matching. It will fail in the presence of ambiguities such as repeated patterns, textures or non-discriminative local appearance. To handle this difficulty, second-order methods (graph-matching) try to enforce geometric consistency between pairs of feature correspondences. The basic idea is shown on Figure 2.2: if the points p_1 and p'_1 of image 1 are matched to points p_2 and p'_2 of image 2, then the geometric relation between p_1 and p'_1, and the one between p_2 and p'_2 should be similar.

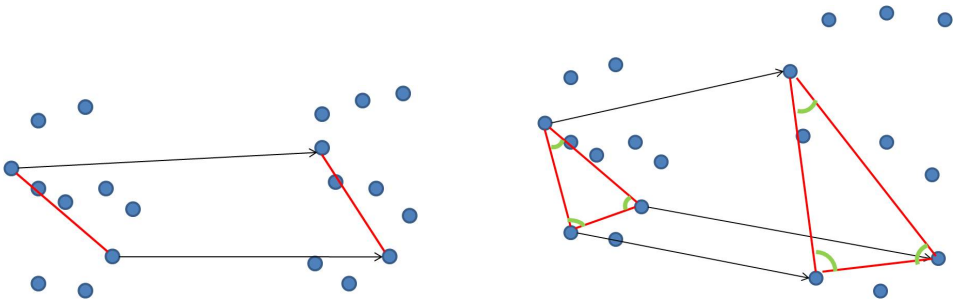


Figure 2.2: Left: second-order potentials can be made rotation-invariant by comparing distances between matched points. Right: Third-order potentials can be made similarity-invariant by comparing the angles of triangles.

Several pairwise geometric relations have been used. [Leordeanu and Hebert, 2005b] use only the distance between two points, leading to a matching criterion which is invariant to rotation. In their objective function, [Berg et al., 2005a] use a combination of potentials based on distances (rotation-invariant) and angles (scale-invariant), to find a trade-off between rotation and scale invariance. Some other methods (e.g., [Schmid and Mohr, 1997; Zheng and Doermann, 2006]) use proximity, only assuming that two adjacent points should be matched in the other image to two points which are also close to each other. One difficulty here is to

define an appropriate notion of neighborhood.

Recently, the computer vision community has put much effort in increasing the order of complexity of the models used: For example,[Kohli et al., 2007] introduce a high-order clique potential for segmentation, but the type of energy is limited to specific types of functions, using the alpha-expansion framework. [Zass and Shashua, 2008] formulate the search for higher-order feature correspondences as a hypergraph matching problem. However, they use independence assumptions in order to reduce the hyper-graph matching to a first-order matching (feature matching), where nodes are matched independently (except for the one-to-one macthing constraint).

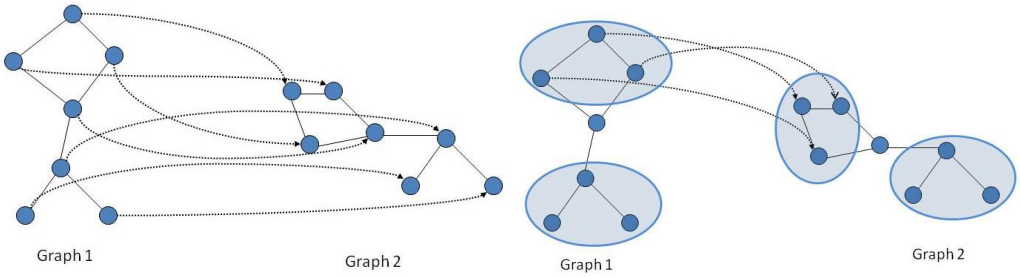


Figure 2.3: Left: Given two graphs, the matching problem is to find node correspondences which preserve their topology. Right: In a hypergraph, one hyperedge can link more than two nodes. As an example, in this figure four hyper-edges are represented by circles. Each of them regroups three nodes that they link together. The three matches drawn on the right figure induce a matching between two hyperedges.

The method described in this chapter also deals with hypergraph matching (see Figure 2.3). In addition, unlike these authors, we will refrain from using independence assumptions (that may or may not be justified depending on the situation). The method presented in this chapter generalizes the spectral matching method of [Leordeanu and Hebert, 2005b] to higher-order potentials: The corresponding hypergraph matching problem is formulated as the maximization of a multilinear objective function over all permutations of the features. This function is defined by a tensor representing the affinity between feature tuples. It is maximized by first using a multi-dimensional power method to solve a relaxed version of the problem, whose solution is then projected onto the closest assignment matrix. As will be shown in the comparative experiments of Section 2.5, explicitly

maintaining higher-order interactions in the optimization process leads to superior performance.

2.1.2 Problem statement

We consider two images, and assume that we have extracted N_1 points from image 1, and N_2 from image 2. We do not assume that $N_1 = N_2$, i.e., there may be different numbers of points in the two images to be matched. Moreover, instead of points, we could use any other type of visual features such as edges, raw pixels, etc. Throughout this paper, for $s = 1, 2$, all indices i_s, j_s, k_s will be assumed to vary from 1 to N_s. We will also note $i = (i_1, i_2)$, $j = (j_1, j_2)$, $k = (k_1, k_2)$ pairs of potentially matched points.

Let P_n^s be the n^{th} point of image s. The problem of matching points from image 1 to points from image 2 is equivalent to looking for an $N_1 \times N_2$ *assignment matrix* X such that X_{i_1,i_2} a.k.a. X_i is equal to 1 when $P_{i_1}^1$ is matched to $P_{i_2}^2$, and to 0 otherwise. In this paper, we assume that a point in the first image is matched to *exactly one* point in the second image, but that one point in the second image may be matched to an arbitrary number of points in the first image, i.e., we assume that the sums of each row of X is equal to one, but put no constraints on the column sums[1]. Thus, we consider the set $\mathcal{X}$ of assignment matrices:

$$\mathcal{X} = \{X \in \{0,1\}^{N_1 \times N_2},\ \forall i_1, \textstyle\sum_{i_2} X_{i_1,i_2} = 1\}.$$

Note that our definition is not symmetric (i.e., if we switch the two images, we obtain different correspondences). It can be made symmetric by considering the two possible matchings (image 1 to image 2 and image 2 to image 1) and combining them (in a mostly application-dependent way), e.g., by taking the union or intersection of matchings.

[1]This framework can easily be extended to allow matching points from the first image to no point in the second image adding a dummy node to the second image as in [Berg et al., 2005a] (if a point of the first image is matched to this dummy node, it means that it is matched to no point).

2.1.3 Organization and Contributions

First, in Section 2.2.2 and 2.2.3, we explain the spectral-matching algorithm. Then, in Section 2.3.1, we introduce our method, which extends spectral matching to higher order matching problems.

The three main contributions of this work are (1) the introduction of an high-order objective function for hyper-graph matching (Section 2.3.1), (2) the application of the tensor power iteration method to the high-order matching task, combined with a relaxation based on constraints on the row norms of assignment matrices, which is tighter than previous methods (Section 2.3.1), (3) an ℓ^1-norm instead of the classical ℓ^2-norm relaxation, that provides solutions that are more interpretable but still allows an efficient power iteration algorithm (Section 2.3.5), and (4) the design of appropriate similarity measures that can be chosen either to improve the invariance of matching, or to improve the expressivity of the model (Section 2.3.6).

The proposed approach has been implemented (Section 2.4), and it is compared to state-of-the-art algorithms on both synthetic and real data. As shown by our experiments (Section 2.5), our implementation is, overall, as fast as these methods in spite of the higher complexity of the underlying model, with better accuracy on standard databases.

Preliminary versions of this work appear in [Duchenne et al., 2009a] and [Duchenne et al., 2011b]. The source code of our software is available on line at `http://www.di.ens.fr/~duchenne`.

2.2 Previous work

2.2.1 Historical review of literature

Establishing correspondences between two sets of visual features is a key problem in computer vision tasks as diverse as feature tracking [Birchfield, 1998], image classification [Lazebnik et al., 2006a] or retrieval [Schmid and Mohr, 1997], object detection [Berg et al., 2005a], shape matching [Leordeanu and Hebert, 2005b; Zheng and Doermann, 2006], or wide-baseline stereo fusion [Pritchett

and Zisserman, 1998]. Different image cues may lead to very different matching strategies. At one end of the spectrum, geometric matching techniques such as RANSAC [Fischler and Bolles, 1981], interpretation trees [Grimson and Lozano-Pérez, 1987], or alignment [Huttenlocher and Ullman, 1987b] can be used to efficiently explore consistent correspondence hypotheses when the mapping between image features is assumed to have some parametric form (e.g., a planar affine transformation), or obey some parametric constraints (e.g., epipolar ones). At the other end of the spectrum, visual appearance alone can be used to find matching features when such an assumption does not hold: For example, bag-of-features methods that discard *all* spatial information to build some invariance to intra-class variations and viewpoint changes have been applied quite successfully in image classification tasks [Zhang et al., 2006, 2007a]. Modern methods for image matching now tend to mix both geometric and appearance cues to guide the search for correspondences (see, for example, [Lazebnik et al., 2006a; Lowe, 2004a]).

Many matching algorithms proposed in the 80s and 90s have an iterative form but are not explicitly aimed as optimizing a well-defined objective function (this is the case for RANSAC and alignment methods for example). The situation has changed in the past few years, with the advent of combinatorial or mixed continuous/combinatorial optimization approaches to feature matching (see, for example [Berg et al., 2005a; Leordeanu and Hebert, 2005b; Maciel and Costeira, 2003; Oliveira et al., 2006; Zheng and Doermann, 2006])[2]. This paper builds on this work in a framework that can accommodate both (mostly local) geometric invariants and image descriptors. Concretely, the search for correspondences is cast as a hypergraph matching problem using higher-order constraints instead of the unary or pairwise ones used by previous methods: First-order methods based (for example) on local image descriptions are susceptible to image ambiguities due to repeated patterns, textures or non-discriminative local appearance for example. Geometric consistency is normally enforced using pairwise relationships between image features. In contrast, we propose in this paper to use higher-order (mostly third-order) constraints to enforce feature matching consistency (Figure 2.2).

[2]To be fair, it should be noted that optimization-based approaches to graph matching were considered a key component of object recognition and scene analysis strategies in the 70s and 80s, see for example the classical text by Ballard and Brown [Ballard and Brown, 1982].

2.2.2 Spectral matching

In [Berg et al., 2005a; Cour et al., 2007; Leordeanu and Hebert, 2005b], the matching problem is formulated as the maximization of the following score over $\mathcal{X}$:

$$\text{score}(X) = \sum_{i_1,i_2,j_1,j_2} H_{i_1,i_2,j_1,j_2} X_{i_1,i_2} X_{j_1,j_2},$$

where H_{i_1,i_2,j_1,j_2} (which is equal to $H_{i,j}$ with our notations for pairs) is a binary potential corresponding to the pairs of feature nodes (P_{i_1}, P_{j_1}) of image 1, and (P_{i_2}, P_{j_2}) of image 2. H is a *positive* similarity measure, such that high values of H correspond to similar pairs.

As described in Section 2.3.6, in this paper, we compute for each pair of nodes from the same image a feature vector f, and we compute H as follow:

$$\forall i_1, i_2, j_1, j_2, H_{i_1,i_2,j_1,j_2} = \exp(-\gamma \| f_{i_1,j_1} - f_{i_2,j_2} \|^2).$$

Many other similarity measures are of course possible.

This graph matching problem is actually an integer quadratic programming problem, with no known polynomial-time algorithm for solving it. Approximate methods may be divided into two groups. The first one is composed of methods that use spectral representations of adjacency matrices (e.g., [Umeyama, 1988]). The second group is composed of algorithms that work directly with the graph adjacency matrices, and typically involve a relaxation of the discrete optimization problem (e.g., [Leordeanu and Hebert, 2005b; Almohamad and Duffuaa, 1993; Zaslavskiy et al., 2009]). In this paper, we focus on improvements of the second group of methods.

In [Cour et al., 2007; Leordeanu and Hebert, 2005b], the set of binary matrices over which the optimization is performed is thus relaxed to the set of real matrices with Frobenius norm equal to $\sqrt{N_1}$, leading to the simpler problem:

$$\max_{\|X\|_F = \sqrt{N_1}} \sum_{i_1,i_2,j_1,j_2} H_{i_1,i_2,j_1,j_2} X_{i_1,i_2} X_{j_1,j_2}. \tag{2.1}$$

Note that all the matrices in $\mathcal{X}$ have only N_1 non-zeros coefficients, which are

equal to one, therefore they indeed all have their Frobenius norm equal to $\sqrt{N_1}$. In turn, Eq. (2.1) can be rewritten as $\max_{\|\tilde{X}\|_2=\sqrt{N_1}} \tilde{X}^T \tilde{H} \tilde{X}$, where $\tilde{X}$ denotes the vector in $\mathbb{R}^{N_1 N_2}$ obtained by concatenating the columns of X and, likewise, $\tilde{H}$ the $N_1 N_2 \times N_1 N_2$ symmetric matrix obtained by unfolding the tensor H. This is a classical Rayleigh quotient problem, whose solution $\tilde{X}^*$ is equal to $\sqrt{N_1}$ times the eigenvector associated with the largest eigenvalue (which we refer to as the *main eigenvector* V) of the matrix $\tilde{H}$ [Golub and Loan, 1996], and can be computed efficiently by the power iteration method described in the next section.

An important constraint that H must satisfy is that it is pointwise non-negative. This is the main hypothesis of the Perron-Frobenius theorem [Frobenius, 1912] that ensures that $\tilde{X}^*$ only has non-negative coefficients, which simplifies the interpretation of the result (see [Leordeanu and Hebert, 2005b]).

In order to obtain an assignment matrix in $\mathcal{X}$, i.e., a matrix with elements in $\{0, 1\}$ and proper row sums, the authors of [Leordeanu and Hebert, 2005b] discretize the eigenvector $\tilde{X}^*$ using a greedy algorithm (see [Leordeanu and Hebert, 2005b] for more details). One could also use the Hungarian algorithm with cost matrix $\tilde{X}^*$ to obtain a permutation matrix.

2.2.3 Power iterations for eigenvalue problems

The power iteration method is a very simple algorithm for computing the main eigenvector of a matrix, which is needed for matching.

Input: matrix $\tilde{H}$
Output: V main eigenvector of $\tilde{H}$
initialize V randomly ;
repeat
    $V \leftarrow \tilde{H} V$;
    $V \leftarrow \frac{1}{\|V\|_2} V$;
until convergence ;

Algorithm 1: Power iterations for eigenvalue problems.

This algorithm is guaranteed to converge geometrically to the main eigenvector of the input matrix [Golub and Loan, 1996]. As explained in Section 2.4, in our situation, H is very sparse and we want to take advantage of this. Indeed,

each step of the power iteration algorithm requires only $O(m)$ operations, where m is the number of non-zero elements of H. Also, typically, in our situation, the algorithm converges in a few dozen steps.

Thanks to this algorithm description, it becomes easy to see one reason why the output V will have only non-negative values as described in the Perron-Frobenius theorem [Frobenius, 1912]. Let us assume we initialize V with only non-negative values (since the algorithm converges to a global optimum, this will not change the output of the algorithm). In our case H is also point-wise non-negative. Therefore at each iteration, each coordinate of V will be replaced by a sum of products of non-negative values, which is also non-negative. So this property remains true until convergence. We will see that this nice property will be conserved in our higher-order algorithm.

2.3 Proposed approach

2.3.1 Tensor formulation of hypergraph matching

We propose to use tensors to solve the high-order feature matching problem. Indeed, using tensors is quite natural to generalize the spectral matching [Leordeanu and Hebert, 2005b] introduced in the previous section which deal with a matrix. Previous work except [Zass and Shashua, 2008] only uses one-to-one and pair-to-pair comparisons for matching. In this paper, we want to compare tuples of points. We denote by d the number of points per tuple, and add higher-order terms to the score function defined in Eq. (2.1). For simplicity, we will focus from now on third-order interactions ($d = 3$). Generalizations to higher-order potentials are (in theory at least) straightforward. However, in practice, it could lead to an exponential growth of the computational complexity.

We define a new high-order score:

$$\text{score}(X) = \sum_{i_1,i_2,j_1,j_2,k_1,k_2} H_{i_1,i_2,j_1,j_2,k_1,k_2} X_{i_1,i_2} X_{j_1,j_2} X_{k_1,k_2}, \tag{2.2}$$

where we assume that H is a 6-dimensional super-symmetric tensor, i.e., invariant under permutations of indices in $\{i_1, j_1, k_1\}$ or $\{i_2, j_2, k_2\}$.

Here, the product $X_{i_1,i_2}X_{j_1,j_2}X_{k_1,k_2}$ will be equal to 1 if and only if the points $\{i_1, j_1, k_1\}$ are respectively matched to the points $\{i_2, j_2, k_2\}$. In this case, it will add $H_{i_1,i_2,j_1,j_2,k_1,k_2}$ to the total score function and 0 otherwise.

As described in Section 2.3.6, H represents a similarity measure, which will be high if the set of features $\{i_1, j_1, k_1\}$ is similar to the set $\{i_2, j_2, k_2\}$. In our experiments, we compute for each triplet of nodes in the same image a feature vector f, and we compute H as follow:

$$\forall i, j, k, H_{i_1,i_2,j_1,j_2,k_1,k_2} = \exp(-\gamma \| f_{i_1,j_1,k_1} - f_{i_2,j_2,k_2} \|^2).$$

More details are provided in latter sections.

As explained in the next section, we can rewrite the score compactly using tensor notation as:

$$\text{score}(\tilde{X}) = \tilde{H} \otimes_3 \tilde{X} \otimes_2 \tilde{X} \otimes_1 \tilde{X}, \tag{2.3}$$

with the same notation as in the matrix case: $\tilde{X} = vec(X)$ and H is rewritten as a tensor $\tilde{H}$ of size $(N_1 N_2)^d$.

This score can be interpreted as a hypergraph matching score. In a hypergraph, an edge can link more than two vertices together (Figure 2.3). In this framework, any element of H is a matching score between two hyper-edges.

In Section 2.3.6, we will explain how higher-orders potentials can be used to have more invariant or more expressive features.

A short introduction to tensors

A tensor is the n-dimensional generalization of a matrix: a matrix can be represented as 2-D rectangular table, and tensors can be viewed as n-dimensional hyper-rectangular tables. Each of the elements of such a tensor is indexed by n numbers: $H = \{H_{i_1,i_2,\ldots,i_n}\}$.

A tensor and a vector can be multiplied in different ways. In this work, we use the following notation:

$$B = A \otimes_k V,$$

$$B_{i_1,\ldots,i_{k-1},i_{k+1},\ldots,i_n} = \sum_{i_k} A_{i_1,\ldots,i_k,\ldots,i_n} V_{i_k},$$

where V is vector and A a n-dimensional tensor. Like a matrix multiplied by a vector produces a vector, an n-dimensional tensor multiplied by a vector is $(n-1)$-dimensional. Also, like the matrix-vector multiplication that can be done in two ways (on the left or on the right), the tensor-vector multiplication can be done in n different ways. The index k in the notation $\otimes_k$ indicates that we multiply on the k^{th} dimension.

In Eq. (2.3), we use the following calculus:

$$\begin{aligned}
\text{score}(\tilde{X}) &= \tilde{H} \otimes_1 \tilde{X} \otimes_2 \tilde{X} \otimes_3 \tilde{X} \\
&= (((\tilde{H} \otimes_1 \tilde{X}) \otimes_2 \tilde{X}) \otimes_3 \tilde{X}) \\
&= (((\sum_k H_{i,j,k} X_k)_{i,j} \otimes_2 \tilde{X}) \otimes_3 \tilde{X}) \\
&= \sum_{i,j,k} H_{i,j,k} X_i X_j X_k.
\end{aligned}$$

So the two expressions of the score in Eq. (2.2) and (2.3) are equivalent.

2.3.2 Tensor power iterations

To find the optimum of the high-order score of Eq. (2.2), we use a generalization of the previously mentioned power iterations, as proposed in [Lathauwer et al., 2000]. The algorithm presented below extends Algorithm 1.

This method is not guaranteed to reach a global optimum. However, it converges to a stationary point for tensors that lead to convex functions of X [Regalia and Kofidis, 2000]. In our experiments, it converges almost always to a very satisfactory solution. Also, the authors of [Regalia and Kofidis, 2000] propose a smart way to initialize it, to lead to a quantifiable proximity to the optimal solution.

We can see that, as in the matrix case, if we initialize V with only non-negative values, the resulting vector will have only non-negative values. This is required to have a meaningful result. Indeed, if negative values of X in the score in Eq. (2.2) were allowed, some product of negative values could have a positive value. Therefore even coordinates of X with a low value could increase the final score, pre-

Input: supersymmetric tensor $\tilde{H}$
Output: V main eigenvector of $\tilde{H}$
initialize V randomly ;
repeat
    $V \leftarrow \tilde{H} \otimes_1 V \otimes_2 V$;
    (i.e. $\forall i,\ V_i \leftarrow \sum_{j,k} H_{i,j,k} V_j V_k$)
    $V \leftarrow \frac{1}{\|V\|_2} V$;
until convergence ;

Algorithm 2: Supersymmetric tensor power iteration (third order).

venting us from interpreting the coordinates of H as a similarity potential activated only when all corresponding pairs are matched.

2.3.3 Tensor power iterations for unit-norm rows

In our context, we want to constrain the norm of each row of X to 1, which is a tighter relaxation of $\mathcal{X}$ than matrices of fixed Frobenius norm. In addition, this corresponds to a many-to-one matching setting: all nodes of the first images are matched to exactly one node in the second image, but several nodes in the first images can be matched to the same one in the second image. We denote by $\mathcal{C}_2$ the set of matrices such that all theirs rows have unit Euclidean norm.

We can extend the previous algorithm to this new set of matrices (Algorithm 3).

Input: supersymmetric tensor $\tilde{H}$
Output: $V = [v_{1,1}, \ldots, v_{N_1,N_2}]^T$ stationary point
initialize V randomly ;
repeat
    $V \leftarrow \tilde{H} \otimes_1 V \otimes_2 V$;
    (i.e. $\forall i,\ V_i \leftarrow \sum_{j,k} H_{i,j,k} V_j V_k$)
    $\forall i_1,\ V(i_1,:) \leftarrow \frac{1}{\|V(i_1,:)\|_2} V(i_1,:)$;
until convergence ;

Algorithm 3: Supersymmetric tensor power iteration (third order) with unit norm constraints. $V(i,:)$ denotes the vector $(V_{i,1}, V_{i,2}, \ldots, V_{i,N_2})^T$.

As shown in the appendix, we have extended the proof of [Regalia and Kofidis,

2000] to handle those new constraints. In particular, we have shown that this algorithm has the same nice properties of the previous one: if the score is a convex function of X, then Algorithm 3 converges to a stationary point V. Note that we can always make the score convex by adding to it a multiple of the function $\tilde{X}^\top \tilde{X}$. Since the $\tilde{X}$ vectors in $\mathcal{C}_2$ all have the same norm, this change the value of the score function only by a constant and thus does not change its optima on $\mathcal{C}_2$ (the set of matrices whose Euclidean norms of each of the N_1 rows are equal to one).

Finally, we want to obtain correspondences and need to compute a binary matrix X from V. We obtain a natural projection step here on the set $\mathcal{X}$: For each row, the coordinate with maximum value in V is set to 1 in X, and the other coordinates of X are set to 0.

2.3.4 Merging potentials of different orders

It could be interesting to include in the matching process, at the same time, information about different potential orders (e.g., considering at the same time pair similarities and triplet similarities). To do this, a first solution is to include the low-order information into the tensor of the highest-order potential H. Cour and Shi [Cour et al., 2007] present a method to do this, combining second and first-order potential. The generalization to our setting is straightforward. However, in our power iteration framework, it is equivalent to use the simple following algorithm (which could also be extended to constrain rows to have unit norms):

Input: several supersymmetric tensors $\tilde{H}^d$ of order d
Output: V main eigenvector of H
initialize V randomly ;
repeat
    $V \leftarrow \tilde{H}^4 \otimes_1 V \otimes_2 V \otimes_3 V +$
    $\tilde{H}^3 \otimes_1 V \otimes_2 V + \tilde{H}^2 \otimes_1 V + \tilde{H}^1$;
    (i.e.$\forall i, V_i \leftarrow \sum_{j,k,l} H^4_{i,j,k,l} V_j V_k V_l + \sum_{j,k} H^3_{i,j,k} V_j V_k + \sum_j H^2_{i,j} V_j + H^1_i$)
    $V \leftarrow \frac{1}{\|V\|_2} V$;
until convergence ;

Algorithm 4: Multiple order supersymmetric tensor power iteration (fourth order).

2.3.5 ℓ^1-norm constraint for rows

One of the main problems of spectral relaxations is that the solution is often nearly uniform, which means that it is hard to extract from it an assignment matrix with values in $\{0, 1\}$. This is due in part to the relaxation of the set $\mathcal{X}$ of assignment matrices to matrices in $\mathcal{C}_2$ with unit ℓ^2-norm rows, which does not lead to sparsity. In fact, we can also relax the set $\mathcal{X}$ to the matrices in $\mathcal{C}_1$ with rows having unit ℓ^1-norm (i.e., sum of absolute values). As shown in Figure 2.4, this leads to results that are more easily interpretable.

In the context of second-order interactions, solving the ℓ^1-norm problem cannot be done by power iterations. However, in our higher-order context, this can be done seamlessly. Indeed, solving the following problem on $\mathcal{C}_1$:

$$\max_{X \in \mathcal{C}_1, X \geqslant 0} \sum_{i,j} H_{i,j} X_i X_j$$

is equivalent to solving (on $\mathcal{C}_2$):

$$\max_{Y \in \mathcal{C}_2} \sum_{i,j} H_{i,j} Y_i^2 Y_j^2$$

with the change of variable: $Y_i^2 = X_i$. The order of this new problem is 4 when using the tensor power iteration algorithm, but the complexity is still as low as second-order problem (see Algorithm 5). Using this algorithm we usually obtain in practice an almost completely binary solution, as shown on a particular example in Figure 2.4. This method is easily extended to solve any high-order matching problem.

2.3.6 Building tensors for computer vision

We can use higher-order potentials to increase either the geometric invariance of image features, or the expressivity of the models (see Figure 2.5). We describe here a few possible potentials. They are all based on computing a Gaussian kernel between appropriate invariant features. Clearly, many other potentials are possible.

In this section we will only consider third-order potentials. As illustrated by

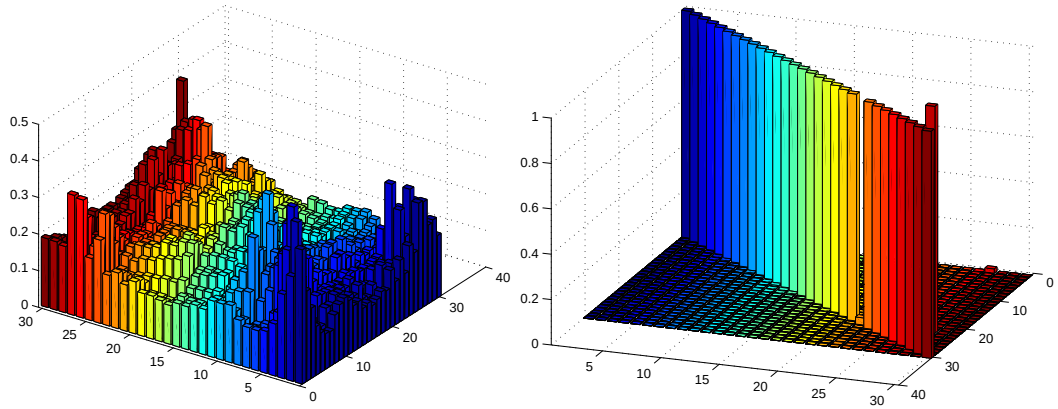


Figure 2.4: We run the spectral algorithm to match a random point cloud to a randomly perturbated copy of itself. We show here the resulting assignment matrices by using ℓ^2 or ℓ^1-norm constraints (respectively Algorithm 1 and 5). The indexing of the points is made such that for all i, the point i of the first cloud corresponds to the point i of the second cloud. So the perfect assignment matrix should be the identity matrix. The horizontal axes correspond to the coordinates of the assignment matrix and the vertical one to its values. Left: Values of the assignment matrix when the ℓ^2-norm is used. They are hard to project to a matrix in $\mathcal{X}$, i.e., with values in $\{0, 1\}$. Right: When using the ℓ^1-norm, we obtain directly a very clear assignment matrix with minor adjustments. Indeed, its values are nearly boolean.

Input: matrix $\tilde{H}$
Output: V stationary point
initialize V randomly ;
repeat
    $V \leftarrow (\tilde{H}(V \circ V)) \circ V$;
    (i.e. $\forall i,\ V_i \leftarrow V_i \sum_j H_{i,j} V_j^2$)
    $\forall i_1,\ V(i_1,:) \leftarrow \frac{1}{\|V(i_1,:)\|_2} V(i_1,:)$;
until convergence ;

Algorithm 5: Tensor power iteration for the ℓ^1-norm relaxation. Here, $\circ$ represents the Hadamard product (or pointwise product). $V(i,:)$ denotes the vector $(V_{i,1}, V_{i,2}, \ldots, V_{i,N_2})^T$.

Figure 2.2, classical methods try to remove ambiguities by looking for matches that preserve some properties of point pairs. Here, we will try to preserve properties of point triplets. In particular, in most of the cases, we will use the properties of the triangle formed by three points. Basically, if the points (P_1^1, P_2^1, P_3^1) are matched to the points (P_1^2, P_2^2, P_3^2), the corresponding triangles should be similar.

In [Leordeanu and Hebert, 2005b], rotation and translation-invariant potentials based on edge lengths and angles are used since it is impossible to build invariants to larger classes of transformations from feature pairs alone. Here, we propose using potentials based on triplets of points, which can be made invariant to richer classes of transformations, including (planar) similarities, affine transformations, and projective ones.

Similarity-invariant potentials

The angles of a triangle are invariant under similarities. Thus we can describe each triangle by its three angles (Figure 2.2). However, in our implementation, we rather use the sines of the angles to speed-up the computation.

Affine-invariant potentials

When the camera is moving, in the general case, the transformation of the image is perspective. However, this transformation can also be affine when the seen object is planar or when looking locally at the image. Therefore affine invariance can be a good approximation of perspective invariance.

Concretely, we normalize each triangle into an equilateral one, and then compare the intensity patterns of normalized triangles by normalized cross correlation. This description of the triangle is of course invariant under affine transformation.

Projective-invariant potentials

Inspired by [Leordeanu et al., 2007], we can also develop higher-order potentials invariant to planar projective transforms with no parametric form when the scene shape is unknown. If we sample only feature points on the contours in the image, we can use the edge direction as an additional feature, and focus on properties of three points and three directions that are conserved under projective transforms. The main property conserved by a projective transform is the cross-ratio. So if we suppose that the object surface in the triangle we are looking at is flat, we can build three lines with four points on each. We compute the descriptor of the three points P_1, P_2, P_3 shown in Figure 2.6. We also show the three vectors N_1, N_2, N_3 which are orthogonal to the image gradient at each point. We draw a line from the point P_1 in the direction of vector N_1 which intersects the other lines (P_2, N_2), (P_3, N_3), (P_2, P_3) in the points z_2, z_3, z_4. And we add $z_1 = P_1$. If the z_i are written with complex numbers, their cross ratio is:

$$(z_1, z_2; z_3, z_4) = \frac{(z_1 - z_3)(z_2 - z_4)}{(z_2 - z_3)(z_1 - z_4)},$$

and this formula is computed for each of the three points. We will use the three cross-ratios defined by those points to make a perspective-invariant descriptor.

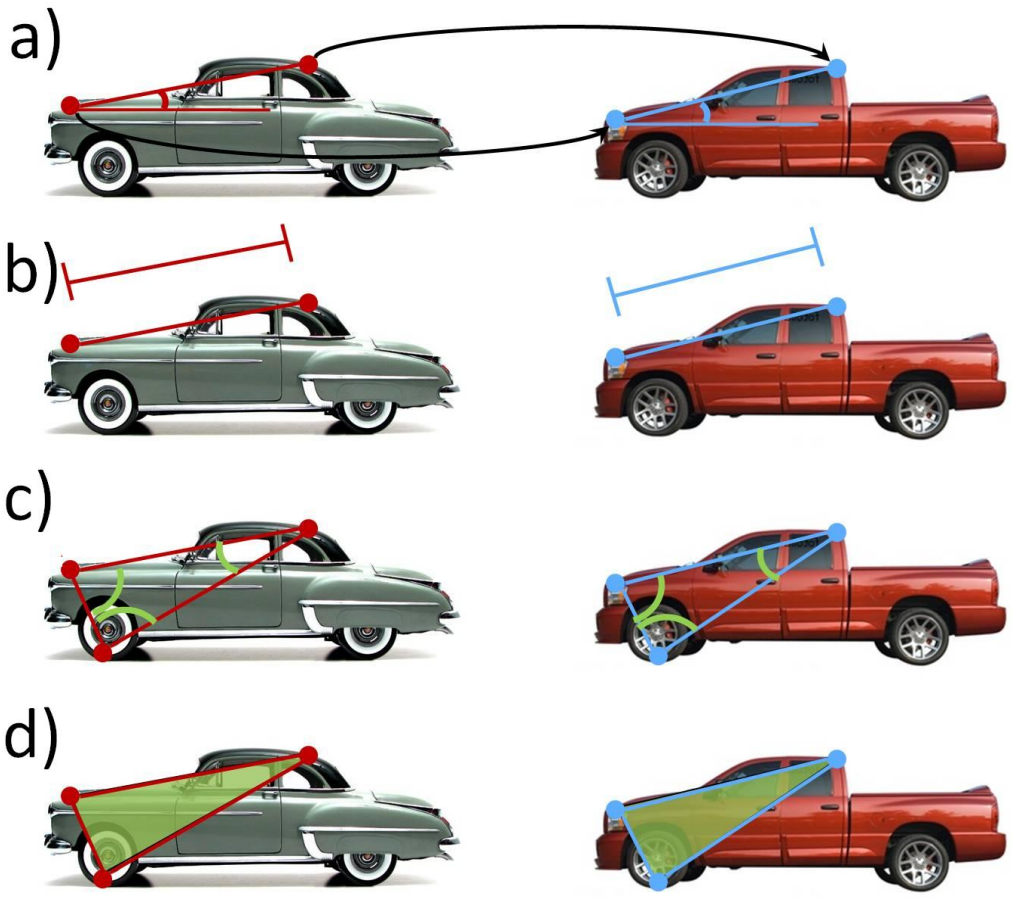
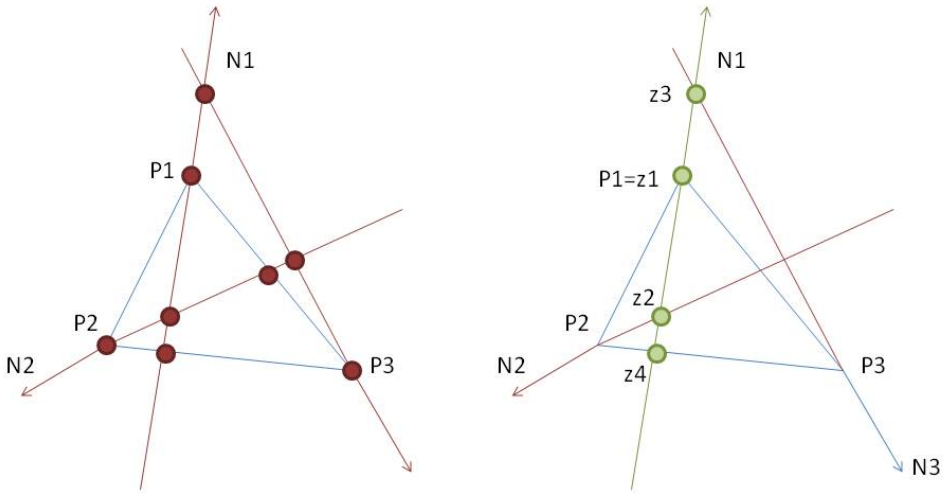


Figure 2.5: (a) a scale-invariant pairwise potential, the angle with respect to the horizontal axis. (b) a rotation-invariant pairwise potential, the distance between the two points. (c) a scale and rotation invariant triplet potential, the 3 angles of the triangle. (d) triplets allow the description of the interior of the triangle, which is a much richer description, and can be affine invariant.

More expressive potentials

Most previous approaches focus on using rather simple geometric relationships between points. Typically the descriptor of their pairwise relationship is a scalar or a low-dimensional vector. As a consequence, such descriptors have low discriminative power, and many different pairs of points have similar descriptors. Therefore matching becomes ambiguous. We believe that our higher-dimensional framework makes it possible to build more expressive features. Triangles, unlike line segments, have an interior. So, it should be possible to have image-based fea-



$$(z_1, z_2; z_3, z_4) = \frac{(z_1 - z_3)(z_2 - z_4)}{(z_2 - z_3)(z_1 - z_4)}$$

Figure 2.6: Left: diagram illustrating the features used in the proposed projective-invariant potential. In order to describe the blue triangle, we build three red lines with four points on each. Right: in order to describe one of these lines (the green one), we denote the four points by z_1 to z_4 (complex numbers). Bottom: we use the cross ratio formula to compute one descriptor for each of the three lines.

tures to describe this interior. To do this, we can use the affine-invariant method explained in section 2.3.6, or, for instance, a histogram of gradient features. Obviously, many other types of features are possible (e.g., bags of words.). These new features would be more specific, and would have higher discriminative power. Therefore a triplet in one image would have fewer similar triplets in the other. In this situation, the matching would become less ambiguous and easier to compute.

3D potential

We now present a high-order potential to match 3D point clouds. We assume that the vertical axis is known and design a potential which is invariant to scale and rotation about this axis. Clearly, many other potentials are possible depending on the assumptions made (e.g., scale/vertical axis known or unknown). We use a 6-dimensional feature to describe the 3D point triplets (Figure 2.7): The first three features are the angles between the three edges of the triangle and the vertical. For the three other features we use the angles of the triangle (as in the 2D case).

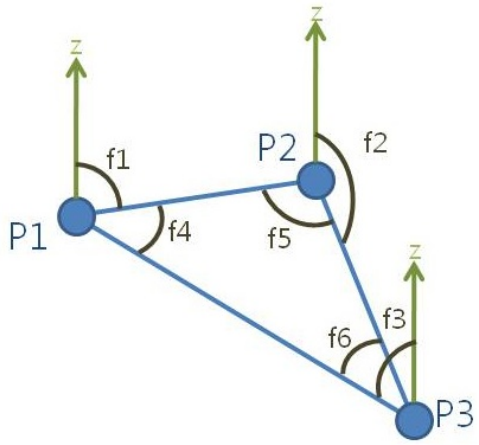


Figure 2.7: Diagram explaining the 3D invariant construction. The 3 points described (P_1 to P_3) and their triangle are in blue. f_1 to f_6 are the angles used as features. The 3 vertical green arrows represent the vertical axis.

2.4 Implementation

In the case of d-th order potentials, the brute force algorithm (see Algorithm 2) has a complexity $O(n^{2d})$ per iteration, where $n = \max\{N_1, N_2\}$. However, Leordeanu and Hebert [Leordeanu and Hebert, 2005b] argue that, in their case, the matrix $\tilde{H}$ has approximately $O(n^3)$ non-zero values. Therefore, the complexity of their algorithm is around $O(n^3)$ operations per iteration for second-order potentials.

As we increase the order of the potential, the number of elements of the tensor

H increases exponentially, and the computation time of the standard algorithm would be the same. Therefore an efficient algorithm is required.

The first step of our algorithm is to build the tensor. This step is often neglected in the literature, but actually requires as much computation as the matching itself, even in the case of conventional graph matching. The tensor size $(N_1N_2)^d$ is huge . If we computed it completely, it would require $O((N_1N_2)^d n_f)$ operations, where n_f is the size of the descriptor of a tuple. However, if we use a truncated similarity measure (i.e., with a compact support), the tensor H can be very sparse. Moreover, in practice, it is not necessary to compute H completely. Therefore in our algorithm, we compute only a small part of H.

In our experiment, we use a truncated Gaussian kernel: $H_{i,j,k} = \exp(-\gamma\|f_{i_1,j_1,k_1} - f_{i_2,j_2,k_2}\|^2)$ if $\|f_{i_1,j_1,k_1} - f_{i_2,j_2,k_2}\| \leq \sigma$ otherwise 0, where f_{i_1,j_1,k_1} is the feature vector describing the tuple (i_1, j_1, k_1).

So, for each tuple i of the first image, we need to find the features of image 2 in a neighborhood of size σ. In practice, we only take the k nearest neighbors with k fixed. This allows us to use a standard implementation of approximate nearest neighbors [Mount and Arya, 2000], which in practice is very efficient. This ANN search implementation is based on kd-trees. In our experiments using this approximate algorithm does not bring any significant change to the results.

However, doing this for all tuples in image 1 would be very time consuming, and forcing all the tuples to be matched correctly is very redundant. So, as in [Zass and Shashua, 2008], we only sample tN_1 triangles in image 1, with fixed t.

Then, we sample all the possible triangles of image 2, and compute their descriptors. We store them in a kd-tree to allow an efficiently search. For each of the selected triangles of image 1, we find the k approximate nearest neighbors of image 2. Then we compute the tensor values: $H_{i_1,j_1,k_1,i_2,j_2,k_2} = \exp(-\gamma\|\mathrm{triangle}(i_1, j_1, k_1) - \mathrm{triangle}(i_2, j_2, k_2)\|^2)$ if $\mathrm{triangle}(i_2, j_2, k_2)$ is among the k nearest neighbors of $\mathrm{triangle}(i_1, j_1, k_1)$, and 0 otherwise. Then we start the power iteration.

The total complexity of the algorithm is $O(n^d \log(n) + ntk \log(n))$ per iteration. The final algorithm typically takes one second for 80 points, $t = 20$ and $k = 500$. The complete setup is summarized in Algorithm 6.

Input: images I_1, I_2 and point sets $\mathcal{P}_1$,$\mathcal{P}_2$
Output: tensor H

$H \leftarrow$ empty tensor ;
foreach $t \in$ *set of all tuples in* $\mathcal{P}_2$ **do**
    $f \leftarrow$computetupleFeature(t, I_2) ;
    $\mathcal{F} \leftarrow \mathcal{F} \cup \{f\}$;
end
$\mathcal{T} \leftarrow$computeANNtree$(\mathcal{F})$;
$\mathcal{S} \leftarrow$select some tuples in I_1 ;
foreach $s \in \mathcal{S}$ **do**
    $\mathcal{N} \leftarrow$search for k nearest-neighbors$(\mathcal{T}, s)$;
    foreach $n \in \mathcal{N}$ **do**
        $H(\text{index}(s), \text{index}(n)) \leftarrow \text{similarity}(\text{descriptor}(s), \text{descriptor}(n))$;
    end
end

Algorithm 6: Efficient ANN-based algorithm for computing the tensor.

Smart selections of triangles

There are several strategies for selecting triangles depending of the final goal. If one wants to match and allow deformations, the triangle should be selected at small scales. On the other hand, if one wants to capture the global property of a shape, one should select big triangles.

2.4.1 Separable similarity measure

One important problem with spectral methods (high-order or not) is that H can be huge. But in some cases, it is possible to avoid computing it. $H_{i,j,k}$ should be a similarity measure between the tuples (i_1, j_1, k_1) and (i_2, j_2, k_2). In this section, we explain that if we can decompose this measure as the inner product of two descriptors $< f_{i_1,j_1,k_1}, f_{i_2,j_2,k_2} >$, we do not have to compute the whole H. When the similarity function is a positive definite kernel, it is always possible to write it as an inner product. However, we also need to have a finite representation of f.

In this situation, we can write (for the second order case):

$$\tilde{H} = \sum_d F_1^d \otimes_{kro} F_2^d,$$

where $\otimes_{kro}$ is the Kronecker product, and $F^m_{I,i,j}$ is the m^{th} descriptor of the pair (i,j) of image I. These two feature matrices can also be very sparse if one considers only the relationship between certain pairs (e.g., pairs of close points).

In this situation we can simplify the computation of the power iteration step:

$$\begin{aligned}
\tilde{X} &\leftarrow \tilde{H}\tilde{X} \\
\tilde{H}\tilde{X} &= (\sum_m F^m_1 \otimes_{kro} F^m_2)\tilde{X} \\
&= \sum_m vec(F^m_2 X (F^m_1)^T) \\
\forall i_1, i_2, X_{i_1,i_2} &\leftarrow \sum_{j_1,m} F^m_{1,i_1,j_1} \sum_{j_2} X_{j_1,j_2} F^m_{2,i_2,j_2} \\
&(= \sum_{j_2,m} F^m_{2,i_2,j_2} \sum_{j1} X_{j_1,j_2} F^m_{1,i_1,j_1}),
\end{aligned}$$

by using the formula $(B^T \otimes_{kro} A)vec(X) = vec(AXB)$.

This decomposition of the similarity measure decreases the amount of memory required by the program. Its time complexity is only $O(nnz(F_1)n + nnz(F_2)n)$ per iteration and, unlike the method described in the previous section, it is an exact algorithm.

In the higher-order case, we can also use this decomposition and the new power iteration step can be written as follows (for the third order case) :

$$\begin{aligned}
&\forall i_1, i_2, \\
&X_{i_1,i_2} \leftarrow \sum_{j_1,k_1,j_2,k_2} H_{i_1,j_1,k_1,i_2,j_2,k_2} X_{j_1,j_2} X_{k_1,k_2} \\
&= \sum_{j_1,k_1,j_2,k_2,m} F^m_{1,i_1,j_1,k_1} F^m_{2,i_2,j_2,k_2} X_{j_1,j_2} X_{k_1,k_2} \\
&= \sum_{j_1,k1,m} F^m_{1,i_1,j_1,k_1} \sum_{j_2} X_{j_1,j_2} \sum_{k_2} F^m_{2,i_2,j_2,k_2} X_{k_1,k_2}.
\end{aligned}$$

Here, the complexity is $O(nnz(F_1) \cdot n + \|F_1\|_{\inf,0,0} \cdot \|F_2\|_{0,0,\inf} \cdot d + nnz(F_2) \cdot n)$ per iteration, where $\|F_1\|_{\inf,0,0}$ is the number of doublets (j_1, k_1) such that

$F_{1,(\cdot),j_1,k_1}$ is not null, and $\|F_2\|_{0,0,\inf}$ is the number of doublets (i_2, j_2) such that $F_{2,i_2,j_2,(\cdot)}$ is not null.

2.5 Experiments

In the experiments presented here (with some exceptions detailed in the subsections), we use Algorithm 6 to compute the tensor. Then, we use the tensor power iteration with unit-norm row constraints described in Algorithm 3, and the ℓ^1-variant of Algorithm 5. The three first experiments use the simple similarity-invariant potential presented in section 2.3.6. The smart selection of triangles is not used in those experiments. As explained in section 2.4, we compute H using the following formula: $H_{i,j,k} = \exp(-\gamma \| f_{i_1,j_1,k_1} - f_{i_2,j_2,k_2} \|^2)$. We set the parameter γ as follows: we compute all the ℓ^2 distances between the tuples of image 1 and their nearest neighbors in image 2 as described in Algorithm 6, then we set γ to the inverse of the average of all the squares of the computed distances. More details are given in the following subsections.

Running time

Even though our tensor power iteration is slower than the probabilistic hypergraph matching method proposed in [Zass and Shashua, 2008], since both methods have to first compute H, their total running times are similar (on the order of one second for 80 nodes).

2.5.1 Synthetic data

Following [Leordeanu and Hebert, 2005b; Zass and Shashua, 2008], we first use synthetic data in order to quantitatively compare our algorithm to the state of the art. We sample randomly and uniformly $n = 25$ points in the 2D plane. We create a second set of points by perturbing the first one with Gaussian noise on their positions. Then, we compare different algorithms to match those two sets. The algorithm provides n matches. The accuracy of the algorithm is computed as the number of good matches divided by n.

In order to have a fair comparison between our method and probabilistic hypergraph matching [Zass and Shashua, 2008], we first compute the tensor as described earlier. Then, we marginalize it as explained in [Zass and Shashua, 2008], and we use the resulting vector with the algorithm they provide on line. We also compare our result and [Zass and Shashua, 2008] to spectral matching [Leordeanu and Hebert, 2005b] to show the improvement of using higher-order potentials.

First, we add Gaussian noise to the position of the second point set, apply a global rotation, and add outliers. The results are shown in Figure 2.8 (top). We can see that our method outperforms the other two. Our interpretation is that when many outliers are added, the ambiguity of pairwise methods [Leordeanu and Hebert, 2005b] increases, because many pairs become similar, whereas triplets are less likely to become similar. Moreover, probabilistic hypergraph matching [Zass and Shashua, 2008] reduces the high-order problem to a first-order one, so that it is likely to match points which have the same neighborhoods. Such a method thus becomes ambiguous when there are many outliers.

Second, we add Gaussian noise, rotation, *and* rescaling. Indeed, low-order matching techniques, such as the spectral method, cannot handle those transformations (rotation and rescaling at the same time). In Figure 2.8 (bottom), we can see that our method and the one of [Zass and Shashua, 2008] are indifferent to those transformations, but the performance of [Leordeanu and Hebert, 2005b] drops to 50% after a scaling of only 1.1 or 0.9, and quickly reaches chance level at 1.2 or 0.8.

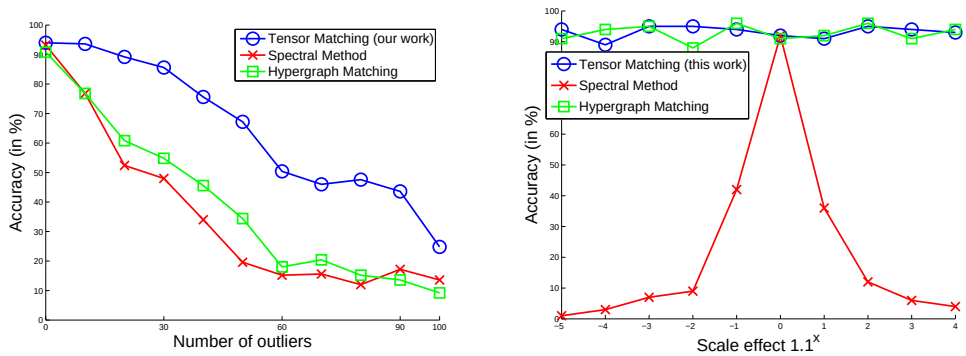


Figure 2.8: Left: Accuracy as a function of the number of added outliers. Right: Accuracy, as a function of the rescaling. (e.g., $x = 2$, correspond to a scaling of 1.1^2).

2.5.2 House dataset

The House dataset [CMU, 2005] is commonly used to test the performance of matching algorithms. Some objects are taken from different viewpoints and $n = 30$ key points, which are present in every frame, are labeled. The scale is always roughly the same, but the transformation is now perspective (although the experiment has been designed such that it is almost orthographic). Since the ground truth is provided, it is also easy to compute the accuracy of the algorithm. The algorithm provides n matches, and the accuracy is the number of good matches among them divided by n. In Figure 2.9, we can see that the low-order algorithms cannot handle the fact that in perspective transforms, the relative positions of points change in a complex way.

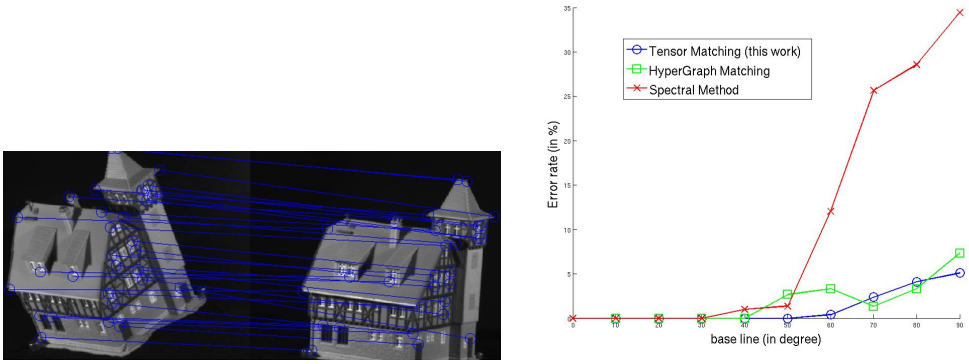


Figure 2.9: House data. Left: Correspondences found by the proposed method in the house dataset. Right: Error Rate on this dataset depending on the base line angle, for different methods.

2.5.3 Natural images

We take images from the Caltech-256 image database [Griffin et al., 2007b] which depict objects on a clear background. We extract their silhouettes and subsample points on them. We can then match images from the same class using our algorithm; results are presented in Figure 2.10. Our tensor-based algorithm is able to match objects with different visual appearances in the presence of strong deformations.

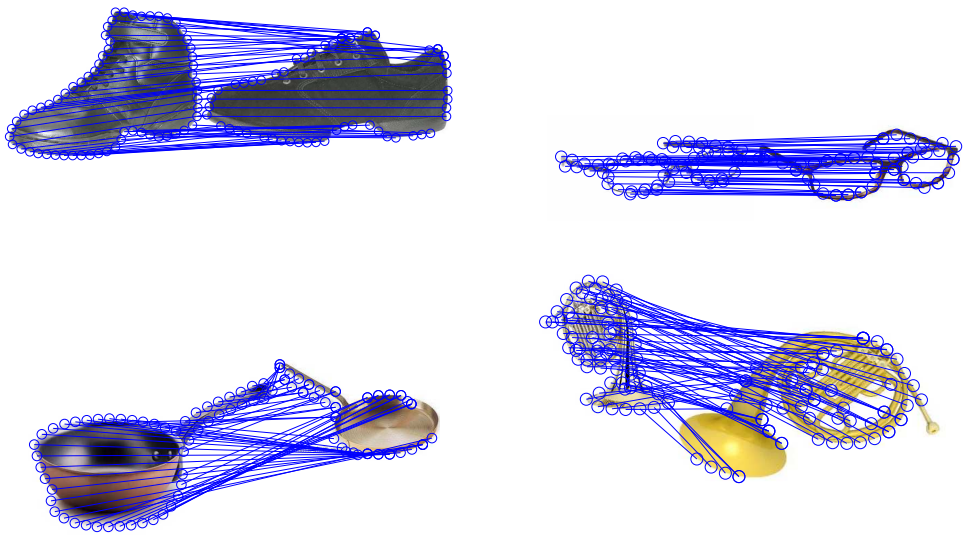


Figure 2.10: Matching silhouettes from the Caltech-256 database.

2.5.4 3D object matching

We have also experimented with the 3D point descriptor described in Section 2.3.6. We have downloaded some freely available 3D models [archive3d, 2008], have manually extracted some points at key positions. After that, we execute our algorithm with only our third-order potential (no local description of the shape). Points are not always matched perfectly, but the results is almost always visually good. Some results are shown in Figure 2.11.

For more completeness, we also use some 3D models from SHREC 2009 dataset [Pratikakis et al., 2009], randomly select 70 points on each of them, and match them. The results can be seen on Figure 2.12. The two sets of experiments have been performed with the same set of parameters.

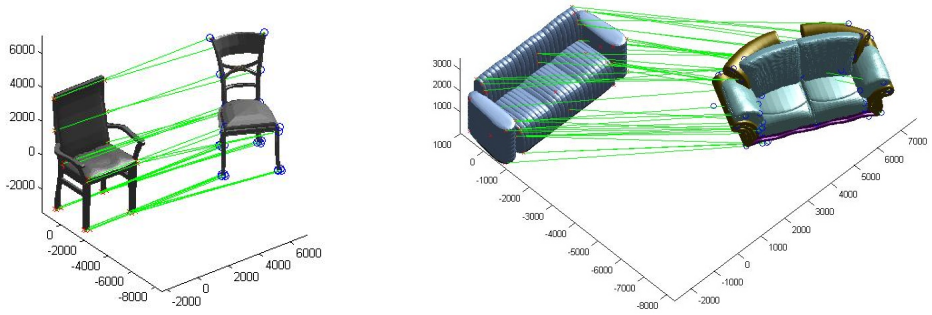


Figure 2.11: Two pairs of 3D models matched by our algorithm.

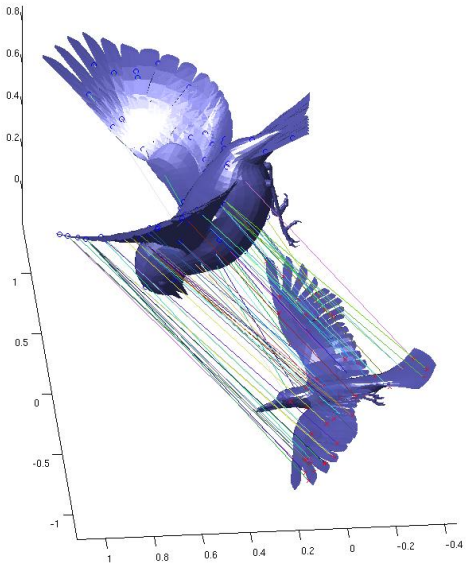


Figure 2.12: Random points are selected on each of the two 3D models, and then matched.

2.5.5 Potentials of different orders

As explained in Section 2.3.4, we can also simultaneously use potentials of different orders. We have chosen an example (Figure 2.13) which is both hard for first-order matching based on SIFT-descriptors, and for tensor matching based on triplets only. We have taken two pictures of the same person with changes in both

viewing angle and face expression. Since the face is deformable, local descriptors tend to be unreliable. In addition, algorithms based on the assumption that the transformation is parametric (such as RANSAC) are not applicable.

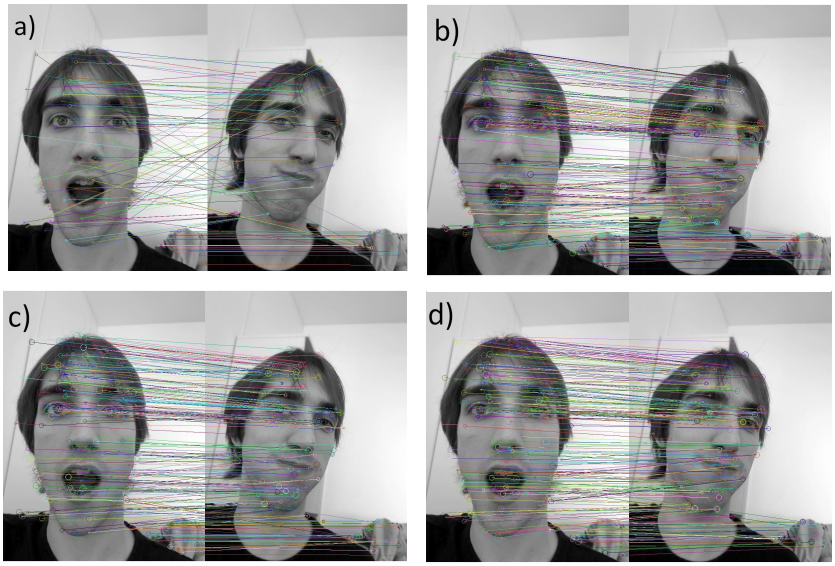


Figure 2.13: (Best seen in color.) Matching of two different pictures with a) SIFT only, b) tensor matching only, c) combined, and d) combined with dummy nodes

In both pictures, we automatically extract around 300 interest points, and for each of them compute its SIFT descriptor (using the implementation of [Vedaldi, 2008]). Then we match each interest point to the one with the closest descriptor in the other image, when the match is unambiguous, as described in [Lowe, 2004b]. The result (Figure 2.13.a) is not satisfying. We believe that this is due to the non-parametric deformation of the face, occlusion, relatively textureless images, and ambiguities due to the symmetry.

We use also triplet information only, without SIFT descriptors. Here too, the graph nodes are the SIFT interest points, and are automatically extracted. This leads to ambiguity in the matching process: not all nodes in one image have similar nodes in the other image. Moreover, matching with only scale- and rotation-invariant features (see Section 2.3.6) is too ambiguous. So we use as triplet descriptor the concatenation of the sines of the three angles and the image coordi-

nates (x-y) of the three edges. This descriptor is only translation invariant but can be robust to small scale or rotation changes. In Figure 2.13.b, one can see that the resulting matching is globally good, but some details are wrong. Since the descriptors are only based on geometry (and not on the image), the algorithm matches the left (resp. right) part of face 1 to the left (resp. right) part of face 2. However, since the face has turned in the second image, the corresponding parts no longer keep their original geometric location, resulting in wrong matches.

As explained in section 2.3.4, we can also combine both cues. The algorithm can use both image-based cues from the first-order potential and geometry-based cues from the third-order potential. The result is very satisfying (Figure 2.13.c). We also show in figure 2.13.d a result with dummy nodes which is slightly improved.

2.6 Conclusion

In this chapter, we have proposed a tensor-based algorithm for high-order graph matching in computer vision applications. We have reached state-of-the-art performance with simple potentials that are invariant to rigid, affine or projective image transformations. This work can be extended in a number of ways, for example by considering more complex features based on three, four or even more point or line features to be fully invariant to richer classes of transformations. It would also be natural to follow the approach of [Caetano et al., 2007] and learn potentials automatically from labeled or partially labeled data.

Chapter 3

Graph matching for image categorization

3.1 Introduction

In this chapter, we address the problem of category-level image classification. The method presented here models an image by a graph whose nodes correspond to a dense set of regions, and edges reflect the underlying grid structure of the image and act as springs to guarantee the geometric consistency of nearby regions during matching (see Figure 3.1). A fast approximate algorithm for matching the graphs associated with two images is presented. This algorithm is used to construct a kernel appropriate for SVM-based image classification, and experiments with standard categorization datasets demonstrate performance that matches or exceeds the state of the art for methods using a single type of features.

3.1.1 Motivation and goals

As explained in the introduction chapter, the advantage of graph-matching methods is that they use the spatial relationship between visual features, on the contrary of other methods, such as bag-of-words, which drop this important information. On the downside, they require more computation time and are hard to optimize. Although graph-matching methods have recorded some success in image categorization (e.g. [Berg et al., 2005a]), they have been soon after outperformed by

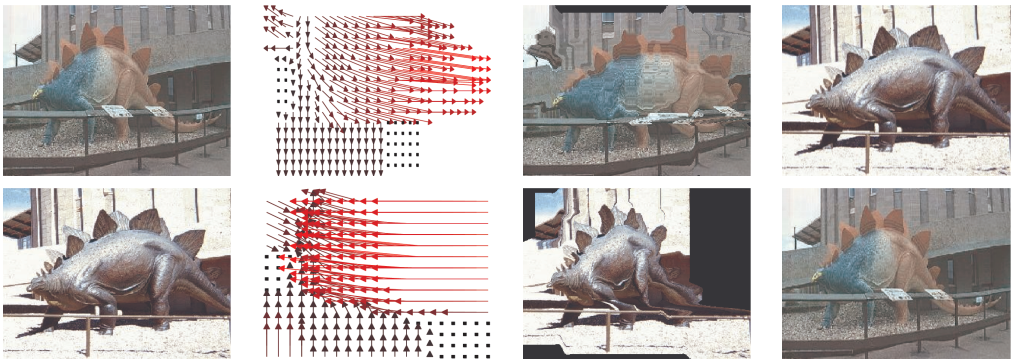


Figure 3.1: The leftmost picture in each row is matched to the rightmost one. The second panel shows the deformation field (displacements) computed by our matching procedure, and the third panel shows the leftmost image after it has been deformed according to that field. Since the matching process is asymmetric, our kernel is the average of the two matching scores (best seen in color).

simpler and faster methods such as spatial pyramids [Lazebnik et al., 2006a] (as shown by [Kim and Grauman, 2010]). At the beginning of this work, we have tried to look for the reasons behind the performance gap between graph-matching approaches and the state of the art, and we have identified three main ones:

- The smaller number of visual features used in graph-matching methods. Typically, [Berg et al., 2005a; Leordeanu and Hebert, 2005b] use around one hundred features, while [Lazebnik et al., 2006a] use thousands of SIFT features extracted on a dense grid.

- The use of a simpler learning machinery: [Berg et al., 2005a] uses the nearest-neighbor approach with a short-listing approximation strategy, while most bag-of-word approaches use non-linear SVMs.

- The heavier computation time requirement of graph matching. We believe that this is a crucial point since it is the main problem behind the two previous points: the slowness of graph matching forces to use fewer visual features and simpler machine learning techniques.

As a result of this analysis, our main challenge has been build a very fast algorithm. Although still much slower than pyramid matching kernel, our method

is fast enough to allow us to use more visual features and the SVM machinery. Thanks to these points, this work shows better performance than spatial pyramids in similar conditions [Boureau et al., 2010] (see Section 3.5).

3.1.2 Organization and contributions

As in the previous chapter, after this introduction, we first do a general review of literature (Section 3.2.1), before describing an highly related work: [Caputo and Jie, 2009; Wallraven et al., 2003] which uses a feature-matching technique to build a kernel appropriate for SVM-based classification (Section 3.2.2).

Then, we present our three main contributions:

1. Generalizing [Caputo and Jie, 2009; Wallraven et al., 2003] to graphs, we propose in Section 3.3 to use the optimum value of the MRF associated with two images as a (non positive definite) kernel, suitable for SVM classification.

2. We propose in Section 3.4 a novel extension of Ishikawa's method [Ishikawa, 2003] for optimizing MRFs which is orders of magnitude faster than competing algorithms (e.g., [Kim and Grauman, 2010; Kolmogorov and Zabih, 2004; Leordeanu and Hebert, 2005a]) for the grids with a few hundred nodes considered in this chapter). In turn, this allows us to combine our kernel with SVMs in image classification tasks.

3. We demonstrate in Section 3.5 through experiments with standard benchmarks (Caltech 101, Caltech 256, and Scenes datasets) that our method matches and in some cases exceeds the state of the art for methods using a single type of features.

This work as been made in collaboration with Armand Joulin. A previous version appears in [Duchenne et al., 2011a].

3.2 Previous work

3.2.1 Review of literature

Early "appearance-based" approaches to image retrieval and object recognition, such as color histograms, eigenfaces or appearance manifolds, used global image descriptors to match images. Schmid and Mohr [Schmid and Mohr, 1997] pro-

posed instead to formulate image retrieval as a correspondence problem where local and semi-local image descriptors (jets and geometric configurations of image neighbors) are used to match individual (or groups of) interest points, and these correspondences vote for the corresponding images. A related technique was proposed by Lowe [Lowe, 2004c] to detect particular object instances using correspondences established between SIFT images descriptors, which have proven very effective for this task. Following Sivic and Zisserman [Sivic and Zisserman, 2003], many modern approaches to image retrieval use SIFT and SIFT-like features, but abandon the correspondence formulation in favor of an approach inspired by text retrieval, where features are quantized using k-means to form a *bag of features* (or *BOF*)—that is, a histogram of quantized features. Pictures similar to a query image are then retrieved by comparing the corresponding histograms, a process that can be sped up by the use of inverted file systems and various indexing schemes. As noted by Jegou et al. [Jegou et al., 2010], image retrieval methods based on bags of features can be seen as voting schemes between local features where the Voronoi cells associated with the k-means clusters are used to approximate the inter-feature distances. In turn, this suggests exploring alternative approximation schemes that retain the efficiency of bags of features in terms of memory and speed, yet afford a retrieval performance comparable to that of correspondence-based methods ([Jegou et al., 2010] is an example among many others of such a scheme).

This also suggests that explicit correspondences between features may provide a good measure of image similarity in image categorization tasks. Variants of this approach can be found in quite different guises in the part-based constellation model of [Fergus et al., 2006], the naive Bayes nearest-neighbor algorithm of [Boiman et al., 2008], and the pyramid matching kernel of [Grauman and Darrell, 2007]. Yet, although these techniques may give state-of-the-art results (e.g., [Boiman et al., 2008]), it is probably fair to say that methods using bags of features and their variants [Boureau et al., 2010; Csurka et al., 2004; Dalal and Triggs, 2005b; Felzenszwalb et al., 2008c; Lazebnik et al., 2006b; Yang et al., 2010; Zhang et al., 2007b] to train sophisticated classifiers such as support vector machines (SVMs) are dominant today in image classification and object detection tasks. This may be due, in part, to the simplicity and efficiency of the BOF model,

but one should keep in mind that, as in the image retrieval domain, BOF-based approaches can be seen as approximations of their correspondence-based counterparts and, indeed, Caputo and Jie [Caputo and Jie, 2009] have shown that feature correspondences can be used to construct an image comparison kernel [Wallraven et al., 2003] that, although not positive definite, is appropriate for SVM-based classification, and often outperforms BOFs on standard datasets such as Caltech 101 in terms of classification rates if not run time.

Bags of features discard all spatial information. There is always a trade-off between viewpoint invariance and discriminative power, and retaining at least a coarse approximation of an image layout makes sense for many object classes, at least when they are observed from a limited range of viewpoints. Indeed, image representations that enforce some degree of spatial consistency –such as HOG models [Dalal and Triggs, 2005b], spatial pyramids [Lazebnik et al., 2006b], and their variants, e.g. [Boureau et al., 2010; Yang et al., 2010]– typically perform better in image classification tasks than pure bags of features. As noted in the introduction, this suggests adding spatial constraints to correspondence-based approaches to object categorization. In this context, several authors [Berg et al., 2005b; Felzenszwalb and Huttenlocher, 2005a; Fergus et al., 2005, 2006; Fischler and Elschlager, 1973b; Kim and Grauman, 2010; Leordeanu and Hebert, 2005a; Liu et al., 2008; Shekhovtsov et al., 2008] have proposed using graph-matching techniques to minimize pairwise geometric distortions while establishing correspondences between object parts, interest points, or small image regions. The problem of matching two images is formulated as the optimization of an energy akin to a first-order multi-label MRF, defined on the corresponding graphs, the labels corresponding to node assignments or, equivalently, to a set of discrete two-dimensional image translations. This optimization problem is unfortunately intractable for general graphs [Boykov et al., 2001], prompting the use of restricted graph structures (e.g., very small graphs [Fergus et al., 2006], trees [Felzenszwalb and Huttenlocher, 2005a], stars [Fergus et al., 2005], or strings [Kim and Grauman, 2010]) and/or approximate optimization algorithms (e.g., greedy approaches [Fischler and Elschlager, 1973b], spectral matching [Leordeanu and Hebert, 2005a], alpha expansion [Boykov et al., 2001], or tree-reweighted message passing, aka TRW-S [Kolmogorov, 2006; Wainwright et al., 2002]).

3.2.2 Caputo's method

In this section, we present the work of [Caputo and Jie, 2009; Wallraven et al., 2003]. This method uses a feature-matching technique to build a kernel matrix appropriate for SVM classification. Our method extends it by using graph matching instead.

Let us denote the set of images (train and test) by $\mathcal{I} = \{I_i\}_{i=1}^{m}$. Each image contains a set of local features $\mathcal{L} = \{L_i\}_{i=1}^{m}$, with $L_i = \{l_j(I_i)\}_{j=1}^{m_i}$. Let us denote K_l a mercer kernel that computes a similarity measure between local features. When comparing two images, this technique matches each local feature of the first image to its nearest neighbor in the second image (feature matching). Thus, they define a global kernel K_G:

$$K_G(I_h, I_k) = \frac{1}{n_h} \sum_{i=1}^{n_h} \max_{j=1..n_k} K_l(l_i(I_h), l_j(I_k)).$$

Then, they symmetrize it, by defining K_S:

$$K_S(I_h, I_k) = \frac{1}{2}(K_G(I_h, I_k) + K_G(I_k, I_h)).$$

They show in there article that, for a given set of images, there always exist a range of parameters for which K_S is definite positive.

Finally, they use this kernel for standard one-versus-rest SVM classification.

3.3 Proposed approach

We propose in this paper to represent images by graphs whose nodes and edges represent the regions associated with a coarse image grid (about 500 regions) and their adjacency relationships. The regions are represented by the mid-level sparse features proposed in [Boureau et al., 2010], and the unary potential used in our MRF is used to select matching features, while the binary one encourages nearby features in one image to match nearby features in the second one while discouraging matching nearby features to cross each other (the matching process is illustrated in Figure 3.1). The optimum MRF value is then used to

construct a (non positive definite) kernel for comparing images (Section 3.3.3). We formulate the optimization of our MRF as a graph cuts problem, and propose as an alternative to alpha expansion [Boykov et al., 2001] an algorithm that extends Ishikawa's technique [Ishikawa, 2003] for optimizing one-dimensional multi-label problems to our two-dimensional setting (Section 3.4). This algorithm is particularly well suited to the grids of moderate size considered here: Our algorithm yields an image matching method that is empirically much faster (by several orders of magnitude) than alternatives based on alpha expansion [Boykov et al., 2001], TRW-S [Felzenszwalb and Huttenlocher, 2006; Liu et al., 2008; Shekhovtsov et al., 2008], or the approximate string matching algorithm of [Kim and Grauman, 2010], for our grid size at least. Speed is particularly important in kernel-based approaches to object categorization, since computing the kernel requires comparing all pairs of images in the training set. In turn, speed issues often force graph-matching techniques [Berg et al., 2005b; Kim and Grauman, 2010] to rely on nearest-neighbor classification. In contrast, we use our kernel to train a support vector machine, and demonstrate in Section 3.5 classification results that match or exceed the state of the art for methods using a single type of features on standard benchmarks (Caltech 101, Caltech 256, and Scenes datasets).

3.3.1 Image representation

An image is represented in this paper by a graph $\mathcal{G}$ whose nodes represent the N image regions associated with a coarse image grid (Figure 3.2), and each node is connected with its four neighbors. The nodes are indexed by their position on the grid, defined as the corresponding couple of row and column indices. It should thus be clear that, when we talk of the "position" of a node n, we mean the couple $d_n = (x_n, y_n)$ formed by these indices. The corresponding "units" are not pixels but the region extents in the x (horizontal) and y (vertical) directions. For each node n in $\mathcal{G}$, we also define the feature vector F_n associated with the corresponding image region.

SIFT local image descriptors [Lowe, 2004c] are often used as low-level features in object categorization tasks. In [Boureau et al., 2010], Boureau et al. propose new features which lead in general to better classification performance

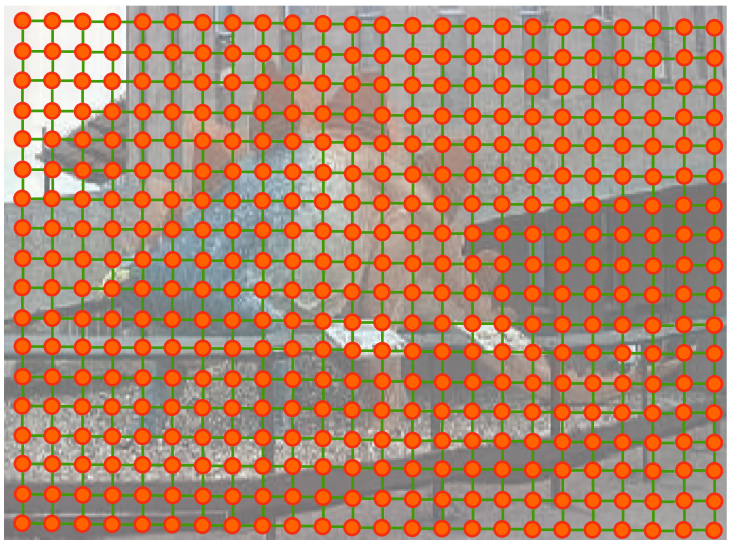


Figure 3.2: We represent images by a grid-shaped graph. The actual number of nodes is 18 by 24.

than SIFT. They are based on sparse coding and max pooling: Briefly, the image is divided into overlapping regions of 32×32 pixels. In each region, four 128-dimensional SIFT descriptors are extracted and concatenated. The resulting 512-dimensional vector is decomposed as a sparse linear combination of atoms of a learned dictionary. The vectors of the coefficients of this sparse decomposition are used as local sparse features. These local sparse features are then summarized over larger image regions by taking, for each dimension of the vector of coefficients, the maximum value over the region (max pooling) [Boureau et al., 2010]. We use the result of max pooling over our graph regions as image features in this paper.

3.3.2 Matching two images

To match two images, we distort the graph $\mathcal{G}$ representing the first one to the graph $\mathcal{G}'$ associated with the second one while enforcing spatial consistency across adjacent nodes. Concretely, correspondences are defined in terms of displacements within the graph grid: Given a node n in $\mathcal{G}$, and some displacement d_n, n is matched to the node n' in $\mathcal{G}'$ such that $p_{n'} = p_n + d_n$, and we maximize the energy function

$$E_{\rightarrow}(d) = \sum_{n \in \mathcal{V}} U_n(d_n) + \sum_{(m,n) \in \mathcal{E}} B_{m,n}(d_m, d_n), \tag{3.1}$$

where $\mathcal{V}$ and $\mathcal{E}$ respectively denote the set of nodes and edges of $\mathcal{G}$, d is the vector formed by the displacements associated with all the elements of $\mathcal{V}$, and U_n and $B_{m,n}$ respectively denote unary and binary potentials that will be defined below. Note that we fix a maximum displacement in each direction, K, leading to a total of $(2K+1)^2$ possible displacements d_n for each node n. The energy function defined by Eq. (3.1) is thus a multi-label Markov random field (MRF) where the labels are the displacements. Typically, we set $K = 5$, which leads to $(2 * 5 + 1)^2 = 121$ possible displacements.

Unary potential

The unary potential is simply the correlation (dot product) between F_n and $F_{n'}$. This simple similarity measure brings several interesting properties:

- Robustness to outliers: In the worst case, completely different features have a similarity score of zero, so they are not too heavily penalized.

- Likely zero-valued on non-similar descriptors: Our descriptors are sparse, they have few non-zero values. In practice, visually different features have few chances to have the same non-zero coefficients, and therefore their dot product is likely to be equal to zero. This limits what we call the "similarity noise", when two visually different features can have a sizable (but small) similarity value. When they add up on the whole image, their values can significantly change the global similarity measure, even though they are

insignificant. In some case, they can overflow the really informative signal coming from the measure between visually similar features.

Global normalization. In this work, at test time, we compare the similarity values between training and test images. But we have noticed that some images have almost systematically higher similarity measures with all test images. This had a high negative impact on performances. That is why we have normalized the descriptors of the images such that the similarity to themselves have a constant value. We assume that when matching an image to itself, there is no deformation, so the binary cost explained below is equal to zero. In this situation, normalizing comes down to dividing the descriptors of an image by their ℓ_2 norm. So, we concatenate the descriptors of all the nodes in the image. We compute its ℓ_2 norm, and we divide all the descriptors by this value.

Binary potential.

The binary potential enforces spatial consistency and is decomposed into two terms.

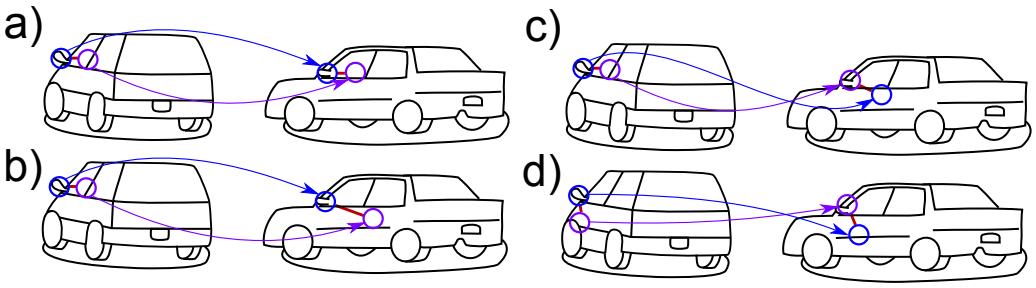


Figure 3.3: Our binary potential encourages two neighboring nodes to be matched to nodes with the same relative position (a). It penalizes distortion of this relative position (b). It highly penalizes inversions (e.g. when the left node is matched on the right (c), or the top node is matched on the bottom (d)).

Binary potential I: stretching The first one acts as a spring. It penalizes stretching (see Figure 3.3.b):

$$u_{m,n}(d_m, d_n) = -\lambda \|d_m - d_n\|_1, \tag{3.2}$$

where λ is the positive spring constant. We use the ℓ_1 distance to be robust to sparse distortion differences.

Binary potential II: crossing We focus on categorizing objects (as opposed to more general scenes) such that, for some range of viewpoints, shape variability can be represented by image displacements varying smoothly over the image, and object fragments typically cannot cross each other (see Figure 3.3.c-d). We thus penalize crossing by adding a binary potential between nearby nodes such that:

$$v_{m,n}(d_m, d_n) = \begin{cases} -\mu[dx_n - dx_m]_+ & \text{if } x_n = x_m + 1 \\ & \text{and } y_n = y_m, \\ -\mu[dy_n - dy_m]_+ & \text{if } x_n = x_m \\ & \text{and } y_n = y_m + 1, \\ 0 & \text{otherwise,} \end{cases}$$

where μ a positive constant and $[z]_+ = \max(0, z)$.
The overall binary potential is thus:

$$B_{m,n}(d_m, d_n) = u_{m,n}(d_m, d_n) + v_{m,n}(d_m, d_n). \tag{3.3}$$

3.3.3 A kernel for image comparison

The two graphs $\mathcal{G}$ and $\mathcal{G}'$ play asymmetric roles in the objective function $E_{\rightarrow}(d)$. One can define a second objective function $E_{\leftarrow}(d)$ by reversing the roles of $\mathcal{G}$ and $\mathcal{G}'$. Optimizing both functions allows us to define a kernel for measuring the similarity of two images, whose value is $\frac{1}{2}(\max_{d_1} E_{\rightarrow}(d_1) + \max_{d_2} E_{\leftarrow}(d_2))$. This kernel does not satisfy the positive definitiveness criterion (this is because of the maximization of Eq. (3.1), see [Caputo and Jie, 2009] for the corresponding argument in a related situation).

The machine learning community has developed several simple techniques to cope with this situation, and to obtain a definitive positive kernel out of a non positive one. [Wu et al., 2005] review and compare some of them, including thresholding or soft thresholding of negative eigenvalues, and shifting of all eigenvalues. We have simply chosen to threshold the negative eigenvalues to 0, in order to construct a valid kernel matrix S. This matrix is then used to train a support vector machine classifier (SVM) in a standard one-vs-all fashion.

3.4 Implementation

Maximizing Eq. (3.1) over all possible deformations is NP hard for general graphs [Boykov et al., 2001]. Many algorithms have been developed for finding approximate solutions of this problem [Boykov et al., 2001; Ishikawa, 2003; Kolmogorov, 2006; Wainwright et al., 2002]. Alpha expansion [Boykov et al., 2001] is a greedy algorithm with strong optimality properties. TRW-S [Kolmogorov, 2006; Wainwright et al., 2002] has even stronger optimality properties for very general energy functions, but it is also known to be slower than alpha expansion [Jung et al., 2008]. Ishikawa's method [Ishikawa, 2003] is a fast alternative that finds the global maximum for a well-defined family of energy functions. Since our energy function defined in Eq. (3.1) possesses properties very close to those of this family, we focus here on Ishikawa's method, and propose extensions to handle the specificities of our energy. Note that Ishikawa's method is one of the rare algorithms capable of solving exactly a multi-label MRF.

3.4.1 Ishikawa's method

[Ishikawa, 2003] has proposed a min-cut/max-flow method for finding the global maximum of a multi-label MRF whose binary potentials verify:

$$B_{mn}(\lambda_m, \lambda_n) = g(\lambda_m - \lambda_n), \tag{3.4}$$

where g is a concave function and λ_m (resp. λ_n) is the label of the node m (resp. n). The set of labels has to be linearly ordered. The Ishikawa method relies on build-

ing a graph with one node n^λ for each pair of node n of $\mathcal{G}$ and label λ, see Figure 3.4 from details. A min-cut/max-flow algorithm is performed on this graph. If it cuts the edge from $n^{\lambda-1}$ to n^λ, we assign the node n to the label λ. [Ishikawa, 2003] proves that this assignment is optimal.

Unfortunately, our set of labels is two-dimensional and there is no linear ordering of $\mathbb{N}^2$ which keeps the induced binary potential concave. A simple argument is that for any label $d_n = (dx_n, dy_n)$, its 4-neighbor labels d_m (e.g., $d_m = (dx_n + 1, dy_n)$, $d_m = (dx_n, dy_n - 1)$...) are equally distant to d_n,i.e. $B(d_n, d_m) = c_n < 0$ for all its neighbors. Any concave function which has the same value c for 3 different points is necessarily always below c. This contradicts $B(d_n, d_n) = 0$.

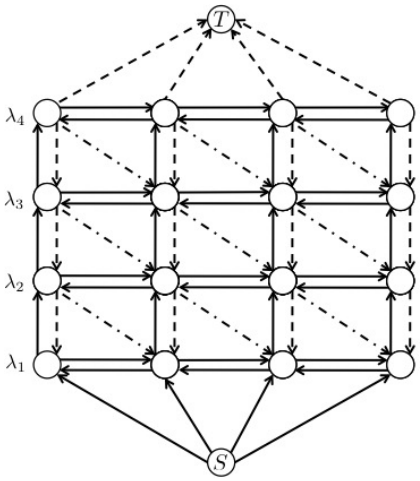


Figure 3.4: A graph associated with Ishikawa's method. On the horizontal axis are the nodes $n \in \{1, \ldots, 4\}$ and on the vertical axis the labels $(\lambda_j)_{1 \leq j \leq 4}$. Each Ishikawa node corresponds to a node n and a label λ_j. The plain vertical arrows represent the unary potentials $U_n(\lambda_j)$. The dashed vertical arrows represent the infinite edges. The plain horizontal arrows correspond to the binary potentials $u_{mn}(\lambda_j, \lambda_j)$ while the dash-dot arrows correspond to the non-crossing binary potentials $v_{mn}(\lambda_j, \lambda_j - 1)$.

3.4.2 Proposed method: Curve expansion

We propose in this section a generalization of Ishikawa's method capable of solving problems with two-dimensional labels.

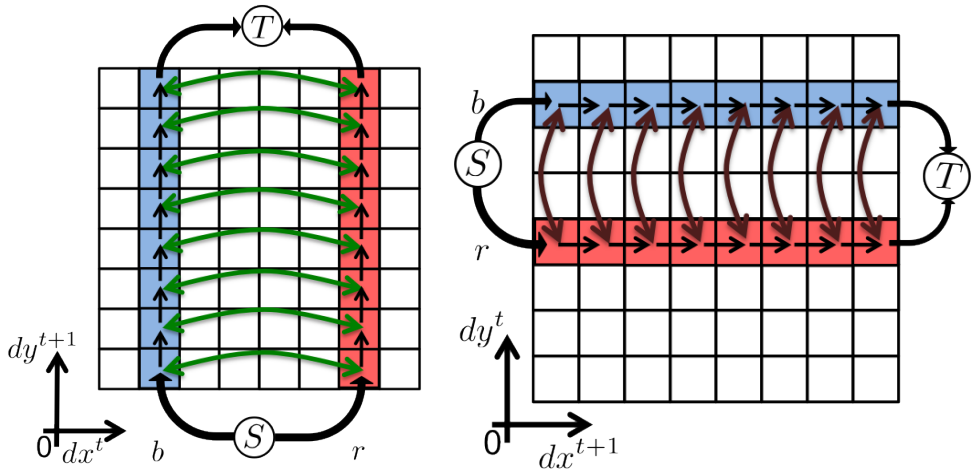


Figure 3.5: Vertical curve expansion (left) and horizontal curve expansion (right) with two nodes, blue ("b") and red ("r"). The grid corresponds to all possible distortions d. The blue (red) squares represent the allowed d for the curve expansion of the blue (red) node. The arrows explain the construction of Ishikawa graph: The black arrows are the unary potentials $U_n(d_n)$ and the green arrows (resp. red) are the binary potentials $B_{mn}(d_m, d_n)$ for vertical (resp. horizontal) moves, i.e., $dx^{t+1} = dx^t$ (resp. $dy^{t+1} = dy^t$). The infinite edges are omitted for clarity (best seen in color).

Two-step curve expansion

The binary part of our MRF can readily be rewritten as:

$$B(d) = \sum_{(m,n)\in\mathcal{E}} g_x(dx_m - dx_n) + g_y(dy_m - dy_n), \tag{3.5}$$

where g_x and g_y are negative concave functions. For a fixed value of $dx = (dx_1, \ldots, dx_N)$, the potentials in $B(d)$ verify condition (3.4), and Ishikawa's method can be used to find the optimal distortion $dy = (dy_1, \ldots, dy_N)$ given dx. We thus alternate between optimizing over dy given dx ("vertical move") and optimizing

over dx given dy ("horizontal move"). Figure 3.5 shows an example of a vertical move (left) and a horizontal move for two nodes.

More precisely, we first initialize d by computing the following upper-bound of $E_{1\to 2}$:

$$\max_{d} \sum_{n\in\mathcal{V}} U_n(d_n) + \sum_{(m,n)\in\mathcal{E}} g_x(dx_m - dx_n).$$

Since $d_n = (dx_n, dy_n)$, this can be rewritten as:

$$\max_{dx} \sum_{n\in\mathcal{V}} \max_{dy_n} \left(U_n(dx_n, dy_n)\right) + \sum_{(m,n)\in\mathcal{E}} g_x(dx_m - dx_n),$$

which can be solved optimally using Ishikawa's method. We then solve a sequence of vertical and horizontal moves:

$$\begin{aligned} dy^{t+1} &\leftarrow \operatorname*{argmax}_{dy} \sum_{n\in\mathcal{V}} U_n(dx_n^t, dy_n) + \sum_{(m,n)\in\mathcal{E}} g_y(dy_m - dy_n), \\ dx^{t+1} &\leftarrow \operatorname*{argmax}_{dx} \sum_{n\in\mathcal{V}} U_n(dx_n, dy_n^t) + \sum_{(m,n)\in\mathcal{E}} g_x(dx_m - dx_n). \end{aligned} \tag{3.6}$$

The local minimum obtained by this procedure is lower than $2\left(\sqrt{N_l}\right)^{N_n}$ configurations, where N_l is the number of labels. By comparison, the minimum obtained by alpha expansion is only guaranteed to be lower than $N_l 2^{N_n}$ other configurations [Boykov et al., 2001].

Multi-step curve expansion

The procedure proposed in the previous section only allows vertical and horizontal moves. Let us now show how to extend it to allow more complicated moves. Ishikawa's method reaches the global minimum of functions verifying condition (3.4). It can be extended to more general binary terms by replacing (3.4) by:

$$\forall \lambda, \mu,\ B(\lambda, \mu) + B(\lambda + 1, \mu + 1) \leq B(\lambda + 1, \mu) + B(\lambda, \mu + 1).$$

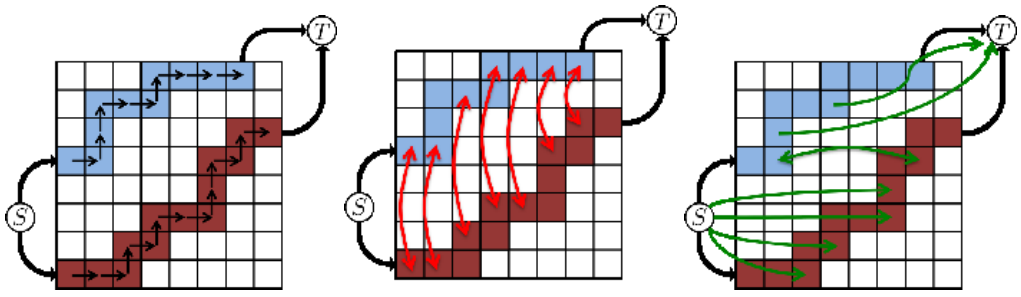


Figure 3.6: An example of curve expansion move for two nodes, b and r. The blue curve corresponds to the nodes $(n_b^\lambda)_{1\leq\lambda\leq N_l}$ obtained by applying the labels λ to the node b. On the left, we show the arrows between nodes n^λ and $n^{\lambda+1}$ representing the unary potential $U_n^{\lambda+1}$ (infinite arrows in the opposite direction have been omitted for clarity). The center (resp. right) panel represents binary potential edges between nodes b and r with labels λ_b and λ_r corresponding to vertical (resp. horizontal) displacements. A displacement which cannot be connected to the corresponding displacements of another node, is either connected to the source S or the sink T (best seen in color).

This is a direct consequence of the proof in [Ishikawa, 2003]. With this condition, we can handle binary functions which do not only depend on pairwise label differences. This allows us to use more complicated moves than horizontal or vertical displacements. We thus propose the following algorithm: At each step t, we consider an ordered list of P_t possible distortions $\mathcal{D}^t = [\tilde{d}_1, \tilde{d}_2, ..., \tilde{d}_{P_t}]$. Given nodes n with current distortion d_n^t, we update these distortions by solving the following problem:

$$\max_{\tilde{d}_t \in \mathcal{D}^t} \sum_{n\in\mathcal{V}} U_n(d_n^t + \tilde{d}_n^t) - \lambda \sum_{(m,n)\in\mathcal{E}} \|d_n^t + \tilde{d}_m^t - (d_n^t + \tilde{d}_n^t)\|_1,$$

where $\tilde{d}_t = (\tilde{d}_1^t, \ldots, \tilde{d}_N^t)$. Then the updated distortion of node n is $d_n^{t+1} \leftarrow d_n^t + \tilde{d}_n^t$.

For example the vertical move, Eq. (3.6), consists of distortions $\tilde{d}$ of the form $(d\tilde{x}, d\tilde{y}) = (0, k)$ for $k \in \{-K, \ldots, K\}$. In practice, we construct a graph inspired by the one constructed for Ishikawa's method. An example is shown in Figure 3.6 with two nodes (for proof details, see the supplementary material).

Note that the set of all possible distortions $\mathcal{D}$ can be different for each node, which gives N different $\mathcal{D}_n$'s. The only constraint is that all the $(\mathcal{D}_n)_{1\leq n\leq N}$ should

be increasing (or decreasing) in y and increasing (or decreasing) in x.

3.5 Experiments

The proposed approach to graph matching has been implemented in C++. In this section we compare its running time to competing algorithms, before presenting image matching results and a comparative evaluation with the state of the art in image classification tasks on standard benchmarks (Caltech 101, Caltech 256 and Scenes).

3.5.1 Running time

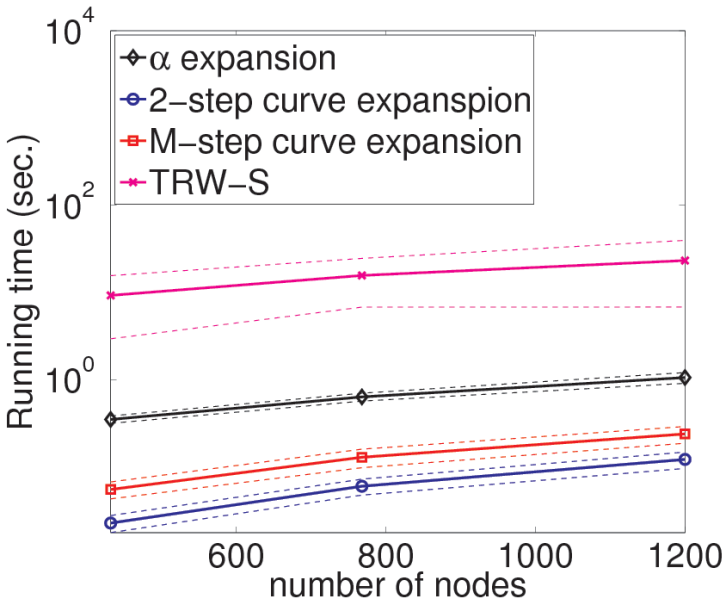


Figure 3.7: Comparison between the running times of TRW-S (purple), alpha expansion (black), 2-step (blue) and multi-step (red) curve expansions, for an increasing number of nodes in the grid (best seen in color, running time in log scale).

We compare here the running times of our curve expansion algorithm to the alpha expansion and TRW-S. For the alpha expansion, we use the C++ implementation of Boykov et al. [Boykov et al., 2001; Kolmogorov and Zabih, 2004]. For

TRW-S, we use the C++ implementation of Kolmogorov [Kolmogorov, 2006]. For the multi-step curve expansion we use four different moves (horizontal, vertical and diagonal moves). All experiments are performed on a single 2.4 gHz processor with 4 gB of RAM. We take 100 random pairs of images from Caltech 101 and run the four algorithms on increasing grid sizes. The results of our comparison are shown in Figure 3.7. The 2-step and multi-step curve expansions are much faster than the alpha expansion and TRW-S for grids with up to 1000 nodes or so. However, empirically, their complexity in the number of nodes is higher than alpha expansion's, which makes them slower for graphs with more than 4000 nodes.

In terms of average minimization performance, 2-step curve expansion is similar to alpha expansion, whereas the multi-step curve expansion with 4 moves, and TRW-S improve the results by respectively, 2% and 5%. However, these improvements have empirically little influence on the overall process. Indeed, for categorization, a coarse matching seems to be enough to obtain high categorization performance. Thus, the real issue in this context is time and we prefer to use the 2-step curve expansion which matches two images in less than **0.04 seconds** for 500 nodes.

Graph-matching based Method	Number of nodes	Matching time
[Berg et al., 2005b]	50	5s
[Leordeanu and Hebert, 2005a]	130	9s
[Kim and Grauman, 2010]	4800	10s
[Kim and Grauman, 2010]	500	1s
Alpha Expansion	500	1s
TRWS	500	10s
Ours 18×24	500	**0.04s**

Table 3.1: Speed of several graph-matching algorithms.

As shown on Table 3.1, other methods have been developed for approximate graph matching [Berg et al., 2005b; Kim and Grauman, 2010; Leordeanu and Hebert, 2005a], but their running time is prohibitive for our application. Berg et al. [Berg et al., 2005b] match two graphs with 50 nodes in 5 seconds and Leordeanu et al. [Leordeanu and Hebert, 2005a] match 130 points in 9 seconds. Kim and Grauman [Kim and Grauman, 2010] propose a string matching

algorithm which takes around 10 seconds for 4800 nodes and around 1 second to match 500 nodes.

3.5.2 Image matching

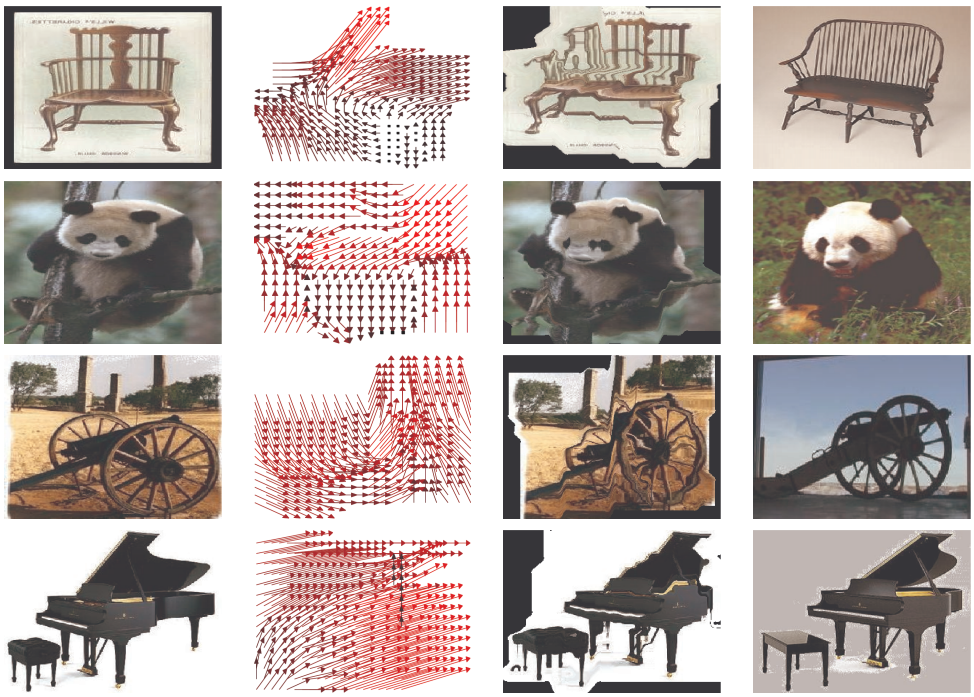


Figure 3.8: Additional examples of image matching on the Caltech 101 dataset. The format of the figure is the same as that of Figure 3.1.

To illustrate image matching, we use a finer grid than in our actual image classification experiments to show the level of localization accuracy that can be achieved. We fix 30×40 grid with maximum allowed displacement $K = 15$. In Figure 3.9, we show the influence of the parameter μ which penalizes the crossing between matches. On the left panel of the figure, where crossings are not penalized, some parts of the image are duplicated, whereas, when crossings are forbidden (right panel), the deformed picture retains the original image structure yet still matches well the model. For our categorization experiments, we choose a value of this parameter in between (middle panel). Figure 3.8 shows some matching results for images of the same category, similar to Figure 3.1.

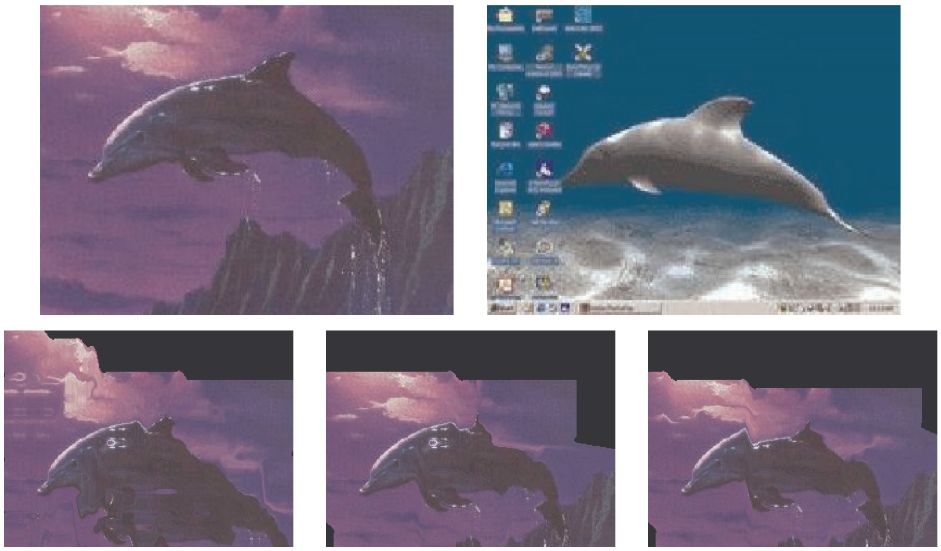


Figure 3.9: We match the top-left image to the top-right image with different value of the crossing constant μ (on the bottom). From left to right, $\mu = 0$, $.5$ and 1.5.

3.5.3 Image classification

We test our algorithm on the publicly available Caltech 101, Caltech 256, and Scenes datasets. We fix $\lambda = 0.1$, $\mu = 0.5$ and $K = 5$ for all our experiments on all the databases. We have tried two grid sizes: 18×24, and 24×32, and have consistently obtained better results (by 1 to 2%) using the coarser grid, so only show the corresponding results in this section. Our algorithm is robust to the choice of K (as long as K is at least 5). The value for λ and μ have been chosen by looking at the resulting matching on a single pair of images. Obviously using parameters adapted to each database and selected by cross-validation would have lead to better performance. Since time is a major issue when dealing with large databases, we use the 2-step curve expansion instead of the M-step version.

Caltech 101. Like others, we report results for two different training set sizes (15 or 30 images), and report the average performance over 20 random splits. Our results are compared to those of other methods based on graph matching [Berg, 2005; Berg et al., 2005b; Kim and Grauman, 2010; Varma and Ray, 2007] in Table 3.2, which shows that we obtain classification rates that are better by more than 12%. We also compare our results to the state of art on Caltech 101 in Ta-

ble 3.3. Our algorithm outperforms all competing methods for 15 training examples, and is the third performer overall, behind Yang et al. [Yang et al., 2009a] and, Todorovic and Ahuja [Todorovic and Ahuja, 2008] for 30 examples. Note that our method is the top performer among algorithms using a single type of feature for both 15 and 30 training examples.

Caltech 256. Our results are compared with the state of the art for this dataset in Table 3.4. They are similar to those obtained by methods using a single feature [Boiman et al., 2008; Kim and Grauman, 2010], but not as good as those using multiple features ([Boiman et al., 2008] with 5 descriptors,[Todorovic and Ahuja, 2008]).

Scenes. A comparison with the state of the art on this dataset is given in Table 3.5. Our method is the second top performer below Boureau et al. [Boureau et al., 2010]. This result is expected since it is designed to recognize objects with a fairly consistent spatial layout (at least for some range of viewpoints). In contrast, scenes are composed of many different elements that move freely in space.

Caltech101 (%)	
Graph-matching based Method	15 examples
GBDist [Varma and Ray, 2007]	45.2
BergMatching [Berg et al., 2005b]	48.0
GBVote [Berg, 2005]	52.0
Kim and Grauman [Kim and Grauman, 2010]	61.5
Ours 18×24	**75.3** $\pm$ **0.7**

Table 3.2: Average recognition rates of methods based on graph matching for Caltech 101 using 15 training examples. In this table as in the following ones, the top performance is shown in bold.

3.6 Conclusion

We have presented a new approach to object categorization that formulates image matching as an energy optimization problem defined over graphs associated with a coarse image grid, presented an efficient algorithm for optimizing this energy function and constructing the corresponding image comparison kernel, and demonstrated results that match or exceed the state of the art for methods using a

Caltech101 (%)			
Feature	Method	15 examples	30 examples
Single	NBNN (1 Desc) [Boiman et al., 2008]	65.0 ± 1.1	-
	Boureau et al. [Boureau et al., 2010]	69.0 ± 1.2	75.7 ± 1.1
	Ours 18×24	$\mathbf{75.3 \pm 0.7}$	80.3 ± 0.8
Multiple	Gu et al.[Gu et al., 2009]	-	77.5
	Gehler et al. [Gehler and Nowozin, 2009]	-	77.7
	NBNN (5 Desc)[Boiman et al., 2008]	72.8 ± 0.4	-
	Todorovic et al.[Todorovic and Ahuja, 2008]	72.0	83.0
	Yang et al. [Yang et al., 2009a]	73.3	**84.3**

Table 3.3: Average recognition rates of state-of-the-art methods for Caltech 101.

Caltech 256 (%)		
Feature	Method	30 examples
Single	SPM+SVM [Griffin et al., 2007a]	34.1
	Kim et al.[Kim and Grauman, 2010]	36.3
	NBNN (1 desc) [Boiman et al., 2008]	37.0
	Ours 18×24	$36.5 \pm .9$
Multiple	NBNN (5 desc) [Boiman et al., 2008]	42.0
	Todorovic et al. [Todorovic and Ahuja, 2008]	**49.5**

Table 3.4: Average recognition rates of state-of-the-art methods for the Caltech 256 database.

Scenes database (%)	
Method	100 examples
Yang et al. [Yang et al., 2009b]	80.3 ± 0.9
Lazebnik et al. [Lazebnik et al., 2006b]	81.4 ± 0.5
Ours 18×24	82.1 ± 1.1
Boureau et al. [Boureau et al., 2010]	$\mathbf{84.3 \pm 0.5}$

Table 3.5: Average recognition rates of state-of-the-art methods for the Scenes database.

single type of features on standard benchmarks. Future interesting research directions are to combine this method with sliding windows to perform object detection or to abandon sliding windows altogether in detection tasks, by matching bounding boxes available in training images to test scenes containing instances of the corresponding objects.

Chapter 4

Image alignment for object detection.

4.1 Introduction

As in the previous chapter, we present here a novel method for comparing two images despite nonrigid transformations. This time, however, we omit pairwise terms in our objective function, which allows us to efficiently compute its global optimum. The corresponding similarity measure is used in object detection tasks.

In the object detection task, the position of the object is unknown. This leads to an increased difficulty, especially for graph-matching techniques. Indeed, the number of possible matches grows with the size of the test image. The size of the search space increases, making it harder and more time-consuming to find a good local optimum. For this reason, in this work, we present a model easier to optimize. This allows us to speed up the matching process and to be guaranteed to get a global optimum.
The method proposed here can be used in two ways:

- It can measure the similarity (up to a deformation) between a given prototype and all the bounding boxes of the same size included in a larger test image. This method can be seen as a "deformable" correlation. As shown on Figure 4.1, we use it to find the part of a test image, which is the most similar to a given template.

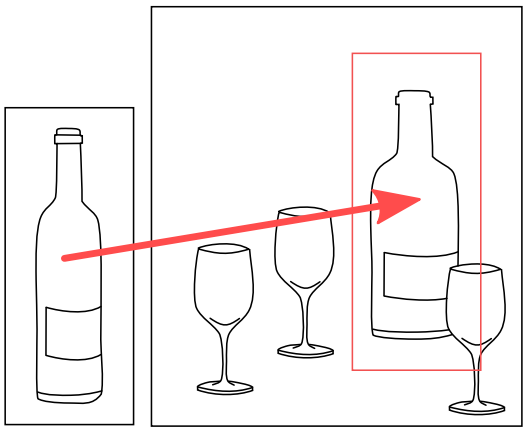


Figure 4.1: We can use our similarity measure to find the part of a test image (right), which is the most similar to a given template (left). The red rectangle is the output bounding box.

- Given two images, it can find the two most similar bounding boxes (one from each image). This can be used: (1) For the co-detection problem (explained in Section 4.3). (2) Or for the partial matching problem. Indeed images often contain background clutter that might induce error in the matching. Our method can be used to match only two relevant bounding boxes, one from each image (see Figure 4.2).

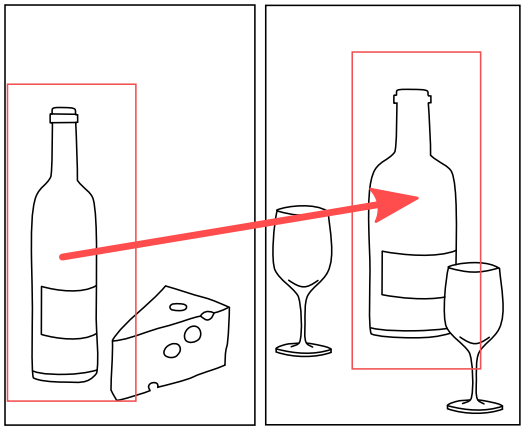


Figure 4.2: Often, only a part of each image represents the same object, the rest is clutter. We can use our similarity measure to quickly find the most similar pair of bounding boxes, one from each image.

For a given sliding window, the method considers all possible parts of a fixed size which are contained in it (see Figure 4.3). All those parts are allowed to move around their anchor position (with a maximum distance). The method is hierarchical such that a part can itself contained subparts. For visibility reason, only the first row and the first column of parts are displayed in Figure 4.3.

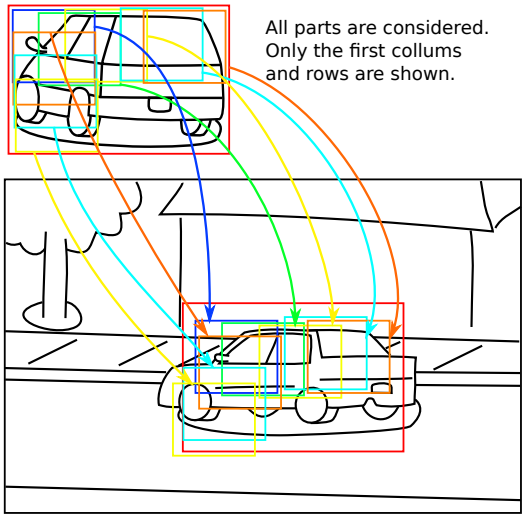


Figure 4.3: (best seen in color) Our method considers all the parts of a given size included in the prototype. For a given position of the sliding window in the test image, it finds the best position of each part, which can move around its anchor position.

4.1.1 Motivation and goals

Most modern approaches to visual image interpretation and scene analysis are composed of three main components: an image model, a measure of similarity between instances of this model, and a classifier based on these. For example, the pedestrian detector of [Dalal and Triggs, 2005a] uses HOG features as its image model, their inner product as its similarity measure, and a linear SVM in sliding-window mode to find people in images.

Constructing appropriate image models and similarity measures is thus fundamental. Ideally, the models should account for the wide variability in appearance of common-day objects due to changes in viewpoint and illumination, and the

variations in shape, color and texture inherent to each category. Likewise, the similarity measure should be invariant to these factors, but capable of discriminating among different object classes.

This work proposes a step in this direction with a very simple, yet highly flexible image model consisting of a hierarchy of rectangular regions allowed to shift arbitrarily within some disparity range to maximize the average correlation of the HOG cells they contain, which is in turn used as a similarity measure suitable for discriminative classifiers such as support vector machines or logistic regression. Our work is inspired by the deformable image templates of [Felzenszwalb et al., 2010], but unlike these, (1) we use all possible parts of the model (with a given size), instead of selecting a few of them, and (2) it allows direct image-to-image comparison in addition to model-to-image comparison.

4.2 Proposed approach

In this work, we propose a very simple but flexible model with only two primitive objects: fixed-size HOG cells [Dalal and Triggs, 2005a] sampled on a coarse image grid, and variable-size rectangular image regions. In this setting, parts are defined at a given resolution as *all* the rectangular image regions that fit within a larger region, which itself may be a part of an even larger region, etc. In addition, parts are allowed to arbitrarily shift within some rectangular disparity area during matching to maximize the mean similarity of the corresponding HOG cells (Figure 4.4), which is used in turn as the similarity measure for the corresponding image patterns. We show that a four-dimensional variant of integral images [Viola and Jones, 2004] and fast streaming maximum filters [Lemire, 2006] can be used to compute this similarity measure efficiently with a computational cost within a small constant factor of the optimal one.

This chapter makes two main original contributions:

1. We propose a novel image similarity measure that allows for arbitrary deformations of the image pattern within some given disparity range and can be evaluated very efficiently [Lemire, 2006], with a cost equal to a small constant times that of correlation in a sliding-window mode.

2. Our similarity measure relies on a hierarchical notion of parts based on simple

Figure 4.4: A target image, a query image, and an overlay of the best match of the deformed query and target images.

rectangular image primitives and HOG cells [Dalal and Triggs, 2005a], and does not require manual part specification [Bourdev and Malik, 2009; Felzenszwalb et al., 2010] or automated discovery [Felzenszwalb and Huttenlocher, 2005b; Lazebnik et al., 2005; Kushal et al., 2007].

Preliminary experiments using the PASCAL VOC 2007 benchmark demonstrate the promise of the proposed approach.

4.2.1 Image Model

The two primitive objects considered in this presentation are HOG cells [Dalal and Triggs, 2005a] of fixed size, sampled on a coarse image grid, and rectangular image regions of different sizes. We assume that the HOG cells have been properly normalized so their similarity can be measured by the inner product of the corresponding feature vectors. Note that we focus on HOG cells since they have proven effective in various object detection tasks [Dalal and Triggs, 2005a; Felzenszwalb et al., 2010], but any other local image descriptor sampled on a regular grid would do.

We limit our attention for the time being to three types of such regions, namely *images*, (bounding) *boxes*, and *parts* (Figure 4.5). As will be explained later, our actual implementation uses a deeper hierarchy (four levels of image regions besides the images themselves), but we stick with boxes and parts in this section for the sake of clarity.

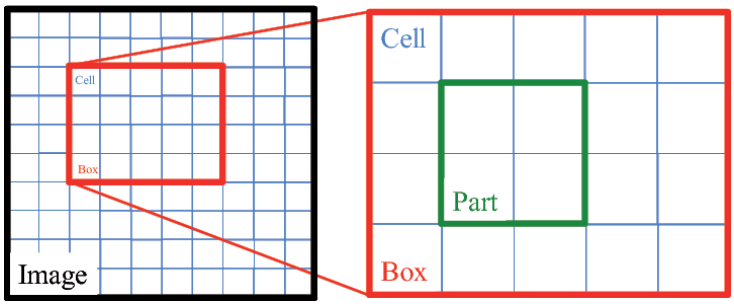


Figure 4.5: Images, boxes, parts and cells.

Let us also assume for the time being that we only have two images, and identify every HOG cell with its position (row plus column) in the corresponding image. Likewise, we identify each rectangular image region with the position of its lower left corner (Figure 4.6). Given cells $\boldsymbol{c}$ and $\boldsymbol{c}'$ in the first and second image, we define $\iota(\boldsymbol{c}, \boldsymbol{c}')$ as the similarity of the corresponding image descriptors, as measured by their dot products.

The similarity between parts respectively located in $\boldsymbol{p}$ and $\boldsymbol{p}'$ in the two images can now be defined as

$$\mu(\boldsymbol{p}, \boldsymbol{p}') = \sum_{\boldsymbol{c} \in \mathbb{P}} \iota(\boldsymbol{p} + \boldsymbol{c}, \boldsymbol{p}' + \boldsymbol{c}), \tag{4.1}$$

where $\mathbb{P}$ is the range of cell locations corresponding to a part.

If we allow the second part to shift within some disparity range $\mathbb{R}$, we obtain a second similarity measure, namely

$$\nu(\boldsymbol{p}, \boldsymbol{p}') = \max_{\boldsymbol{d} \in \mathbb{D}} \mu(\boldsymbol{p}, \boldsymbol{p}' + \boldsymbol{d}). \tag{4.2}$$

Finally we can measure the similarity of two bounding boxes $\boldsymbol{b}$ and $\boldsymbol{b}'$ as the sum of the similarities of their parts, that is

$$\tau(\boldsymbol{b}, \boldsymbol{b}') = \sum_{\boldsymbol{p} \in \mathbb{B}} \nu(\boldsymbol{b} + \boldsymbol{p}, \boldsymbol{b}' + \boldsymbol{p}), \tag{4.3}$$

where $\mathbb{B}$ is the range of part positions within a box.

The indexing of the various image regions is illustrated in Figure 4.6. Clearly,

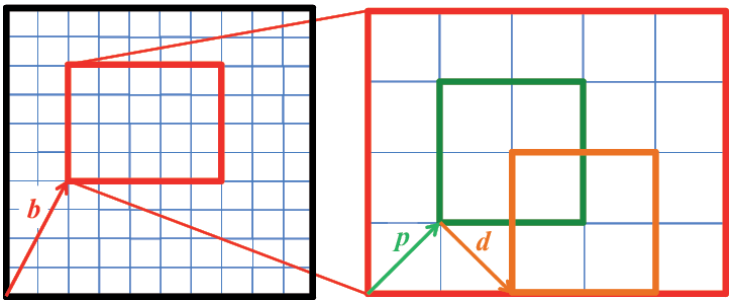


Figure 4.6: An image box positioned in $\boldsymbol{b}$ with a part in position $\boldsymbol{p}$ relative to this box (or absolute position $\boldsymbol{b}+\boldsymbol{p}$), and a shifted part in position $\boldsymbol{d}$ relative to the first one (or absolute position $\boldsymbol{b}+\boldsymbol{p}+\boldsymbol{d}$).

the definitions of Eqs. (4.1)–(4.3) are easily extended to deeper hierarchies of image regions using a recursive notation. We abstain from this worthy exercise here. The main point is that they involve a succession of alternating *sum* and *max* operators computed over rectangular windows. As will be shown in the next section, this is the key to very efficient similarity computations, whatever the depth of the hierarchy may be.

Note that the function ν is not, in general, symmetric (and thus τ is not symmetric either), even when the range of possible negative shifts is the same as the range of positive ones, which is the setting adopted in this paper.

Finally, this similarity measure can be used for two different problems:

- Given an image A, an image B and a box size s, we can find the most similar pair of boxes (one from each image) of size s (see Figure 4.7):

$$[\boldsymbol{b}_A^*, \boldsymbol{b}_B^*] = \arg\max_{\boldsymbol{b}_A, \boldsymbol{b}_B} \tau(\boldsymbol{b}_A, \boldsymbol{b}_B)$$

- Given a template T (an image with the same size than a box) and a test image I, we can find the box in I which is the most similar to T (see Figure 4.4):

$$\boldsymbol{b}_I^* = \arg\max_{\boldsymbol{b}_I} \tau(0, \boldsymbol{b}_I)$$

Figure 4.7: Our method can find the most similar pair of bounding boxes (in red).

In the rest of the chapter, we denote by s the similarity function:

$$s(T, I) = \max_{\boldsymbol{b}_I} \tau(0, \boldsymbol{b}_I), \tag{4.4}$$

where τ is computed based on the features of the template T and the test image I.

4.2.2 Efficient Similarity Computation

We show in this section how to compare efficiently a box $\boldsymbol{b}$ associated with some image $\boldsymbol{i}$ with *all* the bounding boxes $\boldsymbol{b}'$ associated with a second image $\boldsymbol{i}'$. This is useful at training time, when positive pairs of training examples are obtained from two boxes $\boldsymbol{b}$ and $\boldsymbol{b}'$ bounding different instances of the same object, and many positive-negative pairs are obtained by comparing $\boldsymbol{b}$ to all bounding boxes $\boldsymbol{b}'$ not intersecting the object in the second image (or to a subset of these). This is also useful at test time to avoid repeated computations in a sliding window mode.

Let us represent individual HOG descriptors by vectors of fixed dimension h, and denote by q (resp. r) the number of HOG cells in an image (resp. a box), with $q \geq r$. We can compute and tabulate in a $q \times r$ matrix $\mathcal{A}$ the inner products of all pairs of cells in the box $\boldsymbol{b}$ and the image $\boldsymbol{i}'$ in $2hqr$ *flops* (for *floating point operations*). Note that this can be implemented as the product of an $r \times h$ matrix (HOG cells of the box) and the transpose of a $q \times h$ matrix, which is *very* efficient in MATLAB for example.

Let us assume for simplicity that there are r part placements in a bounding box,

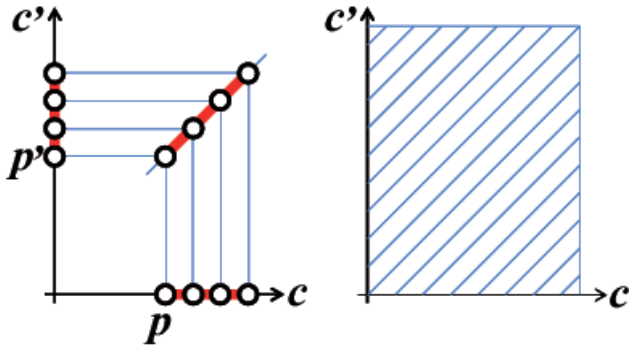


Figure 4.8: Efficient computation of part similarity using modified integral images. Left: computing the similarity of two parts $\boldsymbol{p}$ and $\boldsymbol{p}'$. Right: the integral image is computing by summing up similarities along parallel diagonal lines. See text for details.

and q such placements in an image.[1] Once $\mathcal{A}$ has been computed, the $r \times q$ matrix of similarities $\mathcal{B}$ with entries $\nu(\boldsymbol{p}, \boldsymbol{p}')$ can be calculated efficiently using a slight modification of integral images [Viola and Jones, 2004]. Figure 4.8 illustrates the process for one-dimensional boxes and images. It shows the matrix $\mathcal{A}$ and two parts located in $\boldsymbol{p}$ and $\boldsymbol{p}'$ respectively. The similarity score associated with these parts is obtained by summing the scores along the diagonal line segment with unit slope shown in the figure. Changing the values of $\boldsymbol{p}$ and $\boldsymbol{p}'$ changes where the line segment is located, but not its slope, and if follows that the similarities of any pair of parts can be computed in constant time using 2D integral images summing up similarities along diagonal lines. Two-dimensional boxes and images are easily handled by using 4D integral images represented as multiple 2D ones, and it is easily shown that the total cost of constructing the $r \times q$ matrix $\mathcal{B}$ is $6qr$ flops, including $2qr$ flops for computing the integral images.

Let us ignore boundary effects (again, for simplicity) and assume that all parts in $\boldsymbol{b}$ and $\boldsymbol{i}'$ can be shifted anywhere in the range $\mathbb{D}$. The $r \times q$ matrix $\mathcal{C}$ of shifted part similarities $\tau(\boldsymbol{b}, \boldsymbol{b}')$ can now be computed in a naive manner in qrs flops, where s is the size of $\mathbb{D}$. Using a fast streaming maximum filter [Lemire, 2006], this can be improved to $6qr$ flops by computing the running maxima, first row per row, and then vertically among rows. The total computation thus takes $6(h+1)qr$

[1] There are slightly fewer placements in practice of course. This requires a bit of bookkeeping but does not significantly change the cost estimates of this section.

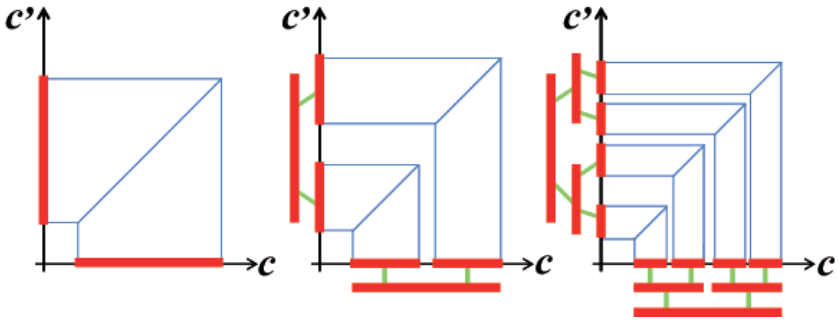


Figure 4.9: A one-dimensional illustration of (one-dimensional) hierarchical deformable part models with one to three layers.

floating point operations.

As mentioned in Section 4.2.1, deeper hierarchies involving several layers of moving parts can easily be accommodated by a simple generalization of the approach presented so far: Similarities between boxes and images can be computed according to the same scheme, alternating *sum* and *max* operations over two-dimensional arrays, and the overall cost remains a small constant time the optimal qr cost (see Figure 4.9 for a one-dimensional example). We have implemented a four-layer hierarchy in our experiments.

One interesting feature is that, like special pyramids, our method can compute a similarity score based on matching, without actually computing the matching. Indeed on Equation (4.2), we compute a $\max$ without keeping the $\arg\max$. So, more work and computing time are needed to actually retrieve the matching parts, especially when we use a deep hierarchy.

4.3 Proof of concept: automated object discovery

The co-segmentation problem is defined as follow: given several images containing a common object, the goal is to segment this common object in each image. Most co-segmentation articles (e.g. [Joulin et al., 2012]) do not take full advantage of the structure of the object in order to be robust to heavy deformations.

Similarly, in this section, we use our method to automatically detect the most similar set of objects from a collection of images. We do not aim to do segmentation but just to find a bounding box around the object. Since our similarity

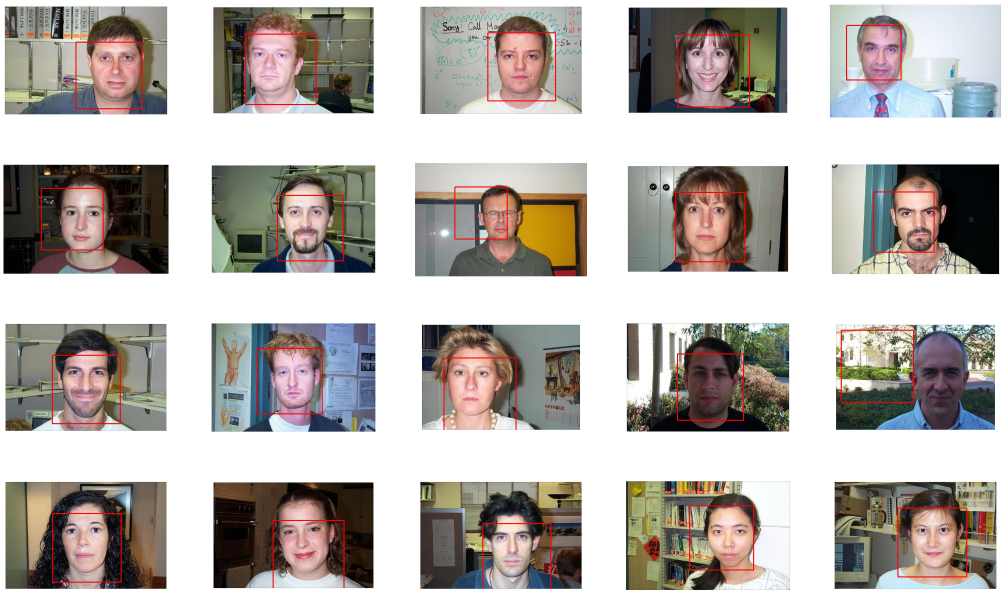


Figure 4.10: A proof-of-concept co-segmentation example on a subset of 20 images from the Caltech face dataset: The last image is used as a query, and the bounding box minimizing the mean similarity to a subset of 19 random images in the dataset is returned, along with the matching bounding boxes in these images. Not that the faces are found reliably in all but one image, even if the bounding boxes are a bit off due to uncompensated deformations.

measure keeps the structural information of the object, we believe that it is more appropriate to call it "co-detection".

As a simple proof of concept (see Figure 4.10), we take some faces from the Caltech face dataset[2]. For a given size of bounding box, we take one reference image, and measure its similarity with others for all positions of the bounding box. We set the reference bounding box where it maximizes the sum of the similarities with other images. Then, we set the other bounding boxes where they maximize their similarity to the reference bounding box.

We believe that this kind of method can also be used to automatically find common parts of an object class. Those parts could afterward be used to perform object detection.

[2]See `http://www.vision.caltech.edu/html-files/archive.html`.

4.4 Learning a detector

We work in the classical setting of (linear) binary supervised classification and recall here its original formulation as well as the primal version of [Chapelle, 2007]. We are given at training time n positive and negative examples $\boldsymbol{x}_i$ in some space $\mathcal{X}$ ($i = 1, \ldots, n$) with labels y_i in $\{-1, +1\}$. Let us consider some Hilbert feature space $\mathcal{H}$, and denote by $\boldsymbol{h} \cdot \boldsymbol{h}'$ the inner product of two elements $\boldsymbol{h}$ and $\boldsymbol{h}'$ of $\mathcal{H}$. This allows us to define the features associated with points in $\mathcal{X}$ by some mapping $\varphi : \mathcal{X} \to \mathcal{H}$, and construct the energy function $E : \mathcal{H} \to \mathbb{R}$ defined by

$$E(\boldsymbol{w}) = \sum_{i=1}^{n} \ell[y_i, \boldsymbol{w} \cdot \varphi(\boldsymbol{x}_i)] + \frac{\lambda}{2}||\boldsymbol{w}||^2, \tag{4.5}$$

where $\ell : \mathbb{R}^2 \to \mathbb{R}$ is some loss function (the hinge loss in the case of support vector machines for example).[3] The classifier is trained by minimizing E with respect to $\boldsymbol{w}$. Once trained, any new point $\boldsymbol{x}$ can be classified according to the sign of the decision function $f : \mathcal{H} \times \mathcal{X} \to \mathbb{R}$ defined by $f(\boldsymbol{w}, \boldsymbol{x}) = \boldsymbol{w} \cdot \varphi(\boldsymbol{x})$. Now, let us define the kernel function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$

4.4.1 Latent SVMs

The latent SVM model of [Felzenszwalb et al., 2010] is learned by minimizing with respect to $\boldsymbol{w}$ the objective function of Eq. (4.5) where, this time, the decision function is defined as the maximum of linear terms over some latent variable $\boldsymbol{d}$ in some space $\mathcal{D}$, that is

$$f(\boldsymbol{w}, \boldsymbol{x}) = \max_{\boldsymbol{d} \in \mathcal{D}} \boldsymbol{w} \cdot \varphi(\boldsymbol{d}, \boldsymbol{D}) = \boldsymbol{w} \cdot \varphi[\boldsymbol{x}, \boldsymbol{d}(\boldsymbol{w}, \boldsymbol{x})], \tag{4.6}$$

where $\varphi : \mathcal{X} \times \mathcal{Z} \to \mathcal{H}$ is a new feature function. Our model naturally fit withing this framework, the latent variable d corresponding to part placements.

[3] An additional bias term may be added, but this does not change our setting.

4.4.2 Hybrid method: Latent SVM and exemplar SVM

We have tried several methods to build an object detector based on our similarity measure s. For instance, we have selected some positive exemplars $\{I_p\}_{p\in\tilde{\mathcal{P}}}$, we have described each sample I by a feature $f = \{s(I_p, I)\}_{p\in\tilde{\mathcal{P}}}$ which contains the similarity measure of I with each of the selected positives, and then we have used a linear SVM or an intersection-kernel SVM on top of those features. However, the results have been disappointing.

In this section, we describe the most promising method so far to perform category-level detection based on our similarity measure. We believe that many of previous detection methods can be modified to benefit from our deformation model.

We combine our similarity measure with a variant of the exemplar SVM method [Malisiewicz et al., 2011] and the Latent SVM [Felzenszwalb et al., 2008a].

Like [Malisiewicz et al., 2011], we train one classifier per exemplar. We train an SVM with this exemplar as positive and all the rest as negatives.

Since the function ι used in Eq. (4.1) is the dot product, if we fix the deformation (the argmax d of Eq. (4.2)), our similarity measure s (Eq. 4.4) can be expressed as a *linear function* of the descriptors of the template T. Therefore we can define a function φ that computes the linear coefficients of this function for a given deformation d, such that:

$$s(T, I) = \max_d < \boldsymbol{w}, \varphi(I, d) >, \tag{4.7}$$

where $< , >$ is the dot product, and $\boldsymbol{w}$ is the descriptor of the template T.

In this equation the deformation d is a hidden latent variable. Similar to the latent SVMs of [Felzenszwalb et al., 2008a], we train our (latent) expamplar-based SVM by solving, for all positive examples p, the following problem:

$$T_p = \arg\min_{T_p} C\left(l(s(T_p, I_p), 1) + \sum_{n\in\mathcal{N}} l(s(T_p, I_n), -1)\right) + \Omega(T) \quad (4.8)$$

$$= \arg\min_{\boldsymbol{w}_p} C(l(\max_d < \boldsymbol{w}_p, \varphi(I_p, d) >, 1)$$

$$+ \sum_{n\in\mathcal{N}} l(\max_d < \boldsymbol{w}_p, \varphi(I_n, d) >, -1)) + \Omega(\boldsymbol{w}_p) \quad (4.9)$$

$$= \arg\min_{\boldsymbol{w}_p} C(l(< \boldsymbol{w}_p, \varphi(I_p, d(\boldsymbol{w}_p, I_p)) >, 1)$$

$$+ \sum_{n\in\mathcal{N}} l(< \boldsymbol{w}_p, \varphi(I_n, d(\boldsymbol{w}_p, I_n)) >, -1)) + \Omega(\boldsymbol{w}_p), \quad (4.10)$$

where l is the hinge loss, Ω is the ℓ_2 normalization on the HOG features, and C is the SVM constant, $\mathcal{P}$ and $\mathcal{N}$ are the sets of positive and negative examplars, the T_p's are the templates learned for each positive exemplar, the $\boldsymbol{w}_p$'s are their descriptors, and $d(\boldsymbol{w}, I) = \arg\max_d(\boldsymbol{w}, \varphi(I, d))$.

We use a very simple optimization scheme: we initialize the template with one positive exemplar, and then we alternate between (1) computing (for each sample I) $\varphi(I, d(\boldsymbol{w}_p, I))$, (2) optimizing on $\boldsymbol{w}_p$ (with libSVM [Chang and Lin, 2001]).

Some details:

- We compute the similarity measure with both the original image, and the mirrored one. Then, we keep the one with the highest similarity. This allows us to be invariant to that transformation.

- The positive exemplars I_p are images extracted from the positive training bounding boxes plus a large margin. This improves robustness to mis-positioned annotations. Our similarity measure finds out the sub image of I_p which is the most similar to T.

- Each template is initialized with one positive exemplars. Except that, we do not use the ones which are too small because their resolution is too small. We resize the others at a constant area. Their aspect ratios are kept.

At test time, we have one template per positive exemplars. Given a test image, for each template, we compute its similarity at all positions of the test image. This is done quickly with the method explained in 4.2.2. To cope with the scale issue, we compare the templates with several rescaled versions of the test image. Then, we use a standard non-max suppression scheme (local-max finding), where the detected bounding boxes have the same aspect ratio than the most similar template.

4.5 Implementation and Results

We have implemented our method and conducted preliminary object detection experiments using the well known PASCAL VOC 2007 benchmark. But unlike [Malisiewicz et al., 2011] and [Felzenszwalb et al., 2010], we do not mine (yet) for hard negative training samples.

In practice, our model may be *too flexible*, and sometimes have difficulties learning appropriate negative weights: For example, if the learned model has strong negative weights along some vertical line, our model will allow parts to shift at test time to avoid this line. Thus, we add to piffies a rigid HOG model at half the template resolution to avoid this problem (Figure 4.13). Empirically, on the other hand, PIF templates appear to select more discriminative features than their HOG counterparts for many object categories (Figure 4.11).

Our implementation uses a fixed template size of 100 9×9 HOG cells and a varying aspect ratio. Comparing a template to a 350×500 image takes 0.01s. At test time (object detection), the classifier is run at 40 different scales. Figure 4.12 shows a few qualitative examples of successful detections on the PASCAL VOC 2007 object detection dataset.

Table 4.1 shows a preliminary quantitative comparison of our results with those of [Malisiewicz et al., 2011] and [Felzenszwalb et al., 2010] on the same dataset. The first three rows of the table show results of three variants of our method obtained on 1000 images randomly picked from this benchmark's test set:[4] Using a plain HOG (or equivalently, using our model with zero layer), using

[4]Lack of time has unfortunately prevented us from using the full test set. Complete results will of course be included in the final version of our paper if it is accepted.

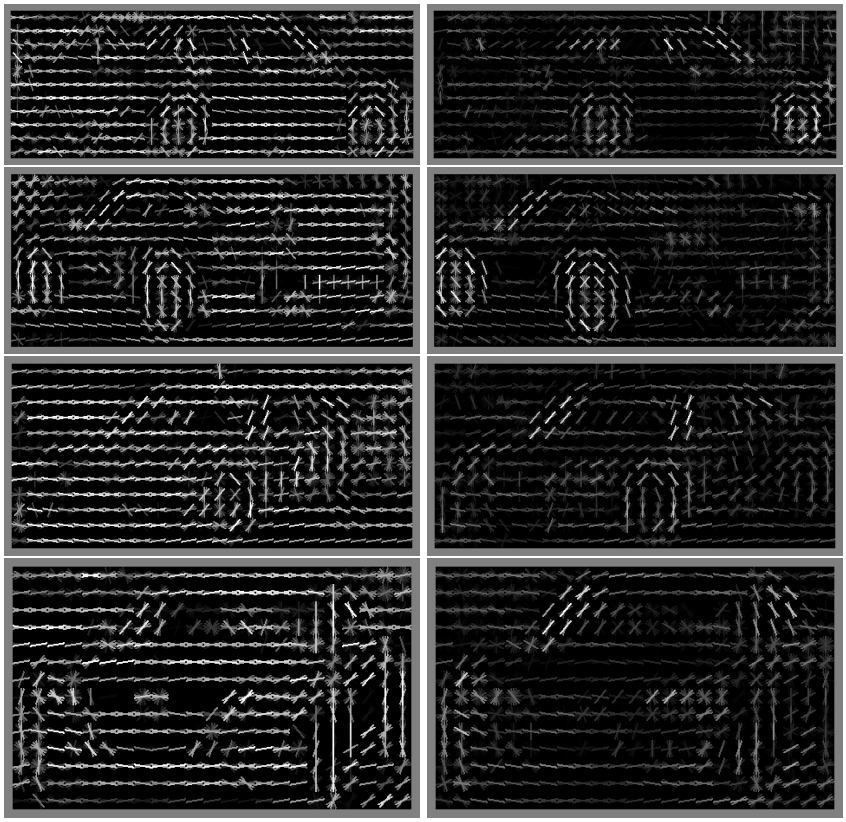


Figure 4.11: Left: HOG descriptors of cars from the Pascal VOC 2007 dataset. Right: The corresponding models learned by our method. In both cases we only show the positive weights. Notice that discriminative parts such as frame, headlights, and wheels are emphasized.

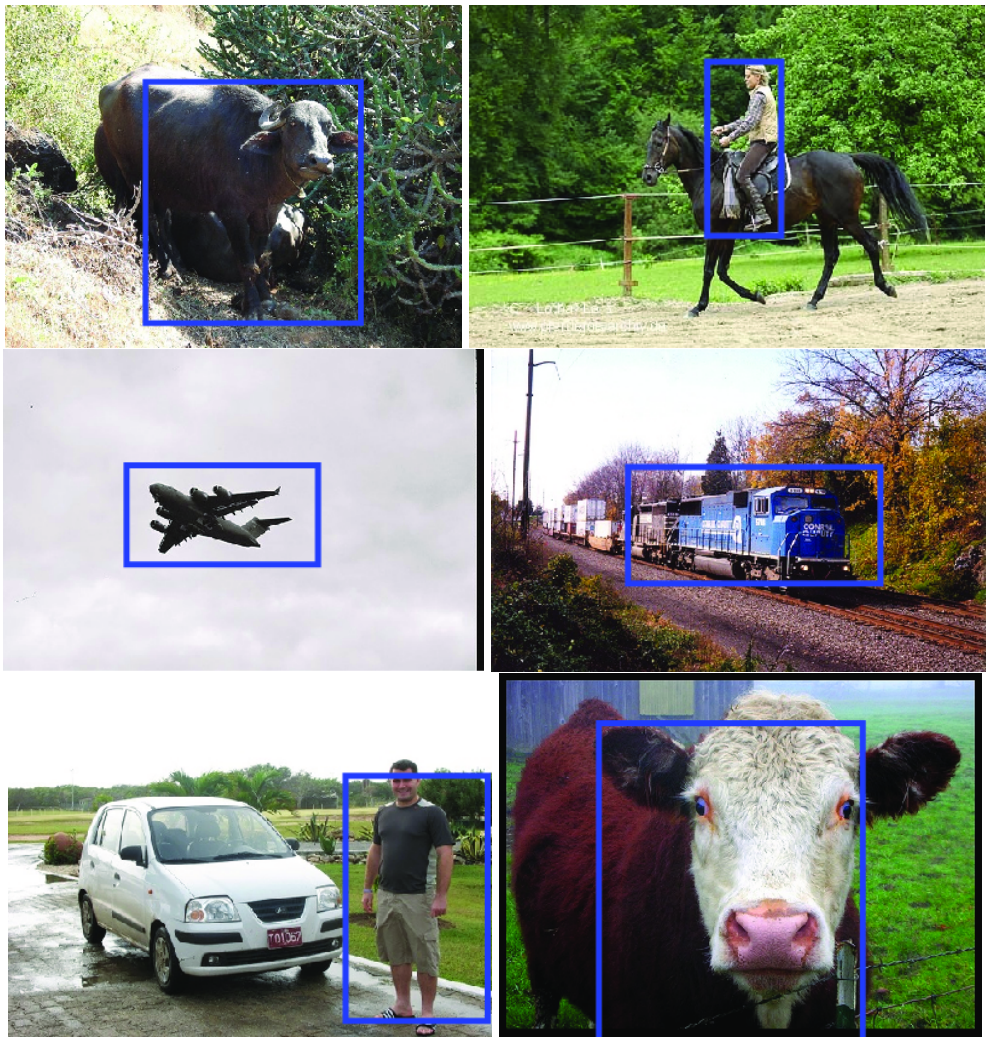


Figure 4.12: Some qualitative detection results for the cow, person, aeroplane, and train classes.

two layers, and using a simple notion of context based on co-occurrence, and similar to [Malisiewicz et al., 2011], to improve the results. The next four rows show, for comparison, the results obtained by [Malisiewicz et al., 2011] and [Felzenszwalb et al., 2010] without and with contextual information. Note that the contextual model of [Felzenszwalb et al., 2010] is much more powerful than the one used in [Malisiewicz et al., 2011] and our work, since it involves the co-occurrence of multiple classes.

Several points are worth noting: First, within our common implementation, the deformations afforded by our model clearly improve over the rigid HOG model of [Dalal and Triggs, 2005a]. Second, contextual information also clearly improves the overall performance. Comparing our results with [Malisiewicz et al., 2011] and [Felzenszwalb et al., 2010] is a bit more difficult since we only use a random subset of the data, which may bias things a bit. It should also be noted that, as it stands, our method is not appropriate for small object detection since it requires a high enough resolution (roughly 60×60 pixels) to account for our hierarchy of parts. Overall, [Felzenszwalb et al., 2010] dominates the other two methods in our experiments, which may be due to its more sophisticated use of aspect and contextual information. Both our methods and [Malisiewicz et al., 2011] give comparable results (22.3% vs 22.7%) when using context, despite the fact that, unlike [Felzenszwalb et al., 2010; Malisiewicz et al., 2011], we do not mine for hard negative examples, which is known to improve results.

4.6 Conclusion

In this chapter, we have described a novel similarity measurement, which combine several interesting advantages: speed, deformation of all possible parts, hierarchy, partial matching. We have a fast implementation based on the integral image method and the streaming maximum filters. We have combined it with exemplar SVM in order to perform object detection. Our experiments are an initial demonstration of the potential of our method. Further experiments are of course needed to assess its full power. Our near future work will be to complete them, as explained in the following chapter.

	aeroplane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motorbike	person	plant	sheep	sofa	train	tv	mAP
HOG	12.3	38.2	9.1	3.8	0.1	7.2	34.1	9.1	0.2	10.9	0.7	1.6	30.6	28.5	9.8	1.5	9.6	0.2	14.1	28.0	12.5
Ours	13.6	52.9	9.3	15.1	0.2	29.2	41.7	10.5	2.6	15.1	1.0	3.2	38.9	38.8	12.9	3.5	10.4	1.5	27.5	27.4	17.8
Ours CO	20.3	50.5	1.5	15.8	10.6	37.4	41.9	11.3	10.9	27.3	8.3	7.6	39.9	36.1	16.4	16.5	17.9	13.5	26.2	35.7	22.3
ESVM	20.4	40.7	9.3	10.0	10.3	31.0	40.1	9.6	10.4	14.7	2.3	9.7	38.4	32.0	19.2	9.6	16.7	11.0	29.1	31.5	19.8
ESVM CO	20.8	48.0	7.7	14.3	13.1	39.7	41.1	5.2	11.6	18.6	11.1	3.1	44.7	39.4	16.9	11.2	22.6	17.0	36.9	30.0	22.7
LDPM	28.7	55.1	0.6	14.5	26.5	39.7	50.2	16.3	16.5	16.6	24.5	5.0	45.2	38.3	36.2	9.0	17.4	22.8	34.1	38.4	26.7
LDPM CO	32.8	56.8	2.5	16.8	28.5	39.7	51.6	21.3	17.9	18.5	25.9	8.8	49.2	41.2	36.8	14.6	16.2	24.4	39.2	39.1	29.1

Table 4.1: A comparison on the PASCAL VOC 2007 object detection benchmark of our implementation of Dalal and Triggs [Dalal and Triggs, 2005a] (HOG), the proposed method without (Ours) and with (Ours CO) contextual information, and the methods of Malisiewicz *et al* [Malisiewicz et al., 2011] (ESVM and ESVM CO) and Felzenszwalb *et al.* [Felzenszwalb et al., 2010] (LDPM and LDPM CO), again without and with contextual information. All results are given in average precision percentage.

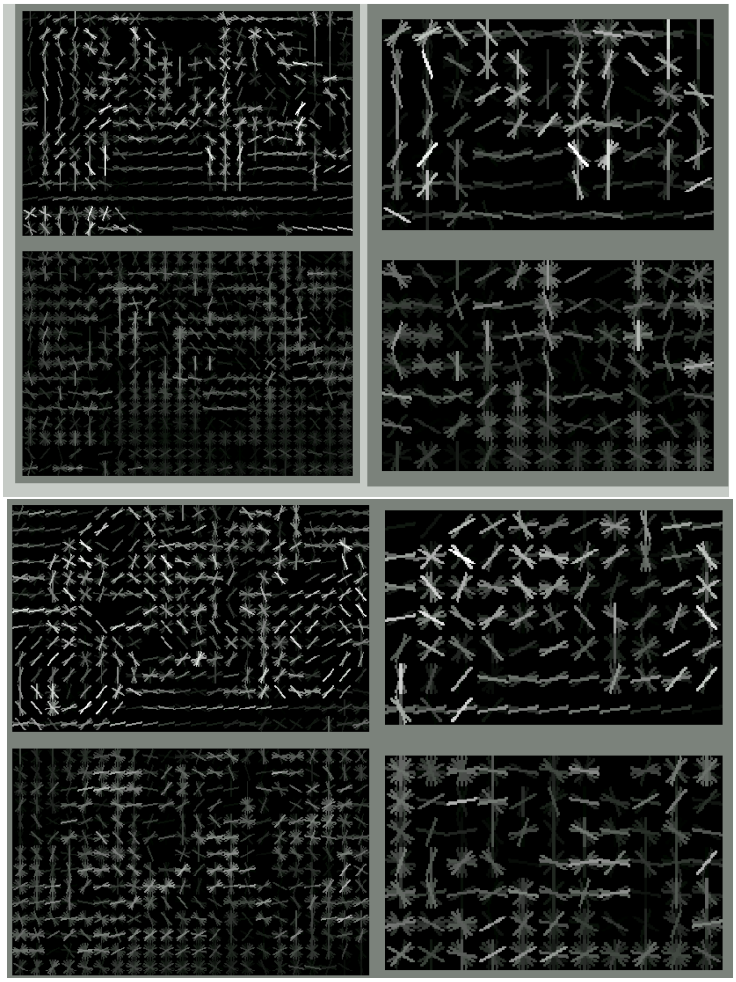


Figure 4.13: Cow and motorbike models using our model contrasting high-resolution, deformable template with low-resolution, rigid HOG templates. Top: positive weights. Bottom: negative weights.

Chapter 5

Conclusion and perspectives

5.1 Contributions

In this thesis, we have presented several nonrigid alignment methods for visual recognition. In chapter 2, we have shown a novel way to use graph-matching for a richer set of objective functions that can handle relationship between more than two nodes, and how to optimize it. In chapter 3, we have demonstrated that the combination of a fast matching technique on dense features and an SVM can achieve state-of-the-art results on object categorization. Finally, in chapter 4, we have presented a fast non rigid alignment method to perform object detection, with promising early results.

5.2 Future work

5.2.1 Detection experiments

Next on our agenda is to conduct a full assessment of our object detection framework. In section 4.4.2, we have demonstrated that –within the exemplar SVM framework– using our deformation-aware similarity measure increases the performance compared to the standard "rigid" convolution. But our implementation of exemplar-SVM with rigid templates does not perform as well as the one of [Malisiewicz et al., 2011]. Therefore we will put effort into matching their results.

This will hopefully also automatically increase the performance of our deformable model.

Moreover, in the exemplar-SVM framework, we have one template per positive exemplar. Since each of them needs to be compared to each test image, this leads to a considerable slow-down of our method. Therefore, we will also implement a combination of our similarity measure with the aspects of the latent SVM [Felzenszwalb et al., 2008a], using the deformation d as the latent variable. This method clusters the positive samples into "aspects". Each cluster is used as the positive training set for a template. This way, we would only have to compare a few templates (one per aspect) with each test image, leading to a substantial speed-up. Moreover, each template will use several positive exemplars for training, which could leads to a better generalization. However, the clustering is a non-convex task, which needs a good initialization and a well-engineered implementation.

5.2.2 Aspects and full object model

This thesis mainly focuses on the design of efficient alignment method. But it addresses only superficially the problem of constructing an object model. We have used the exemplar SVM that separately deals with each positive exemplar. So, we do not take advantage of the commonalities between them. On the other hand, [Felzenszwalb et al., 2008a] clusters the exemplars into "aspects", such that each template is learned on several similar positive exemplars. Like them, we believe that a linear SVM can learn a concept only if it is trained on similar samples.

In their case, each training sample is either completely used for the training of a concept, or not at all. However some objects may only partially match, such that they cannot be clustered in the same group. For instance, many of [Felzenszwalb et al., 2008a]'s aspects correspond to different viewpoints, and objects seen from close viewpoints often have large common parts. Our alignment methods can be used to find these commonalities. This would allow positive exemplars to be used for the training of several aspects: all those who share a common part with them. It would also allows aspects to share parts. We believe that our sub-window similarity measure (Figure 4.2) can help to go further, and automatically find these

common parts between positive exemplars.

5.2.3 Joint Alignment of multiple images

In this thesis, we have only worked on comparing two images. However, being able to align many images simultaneously could lead to many interesting applications.

For instance, in our co-detection experiments (Section 4.3), we have presented a proof-of-concept method to find out the common parts of many images (all containing faces). But it works around the problem by computing the pair-wise alignment to a reference image. If we could really align many images at the same time, the method would be more robust, and we could discover commonalities between images which are more complicated than faces.

Moreover, if we want to find the template T (with feature $\boldsymbol{w}$ of a given norm) that has the best sum of similarity s with the positive exemplars I_p, we can do the following derivation (using the notations of the section 4.4.2):

$$\begin{aligned}
\max_T \sum_{p\in\mathcal{P}} s(T, I_p) &= \max_{\|\boldsymbol{w}\|_2=1} \sum_{p\in\mathcal{P}} \max_{\boldsymbol{w}} < \boldsymbol{w}, \varphi(I_p, d_p) > & (5.1)\\
&= \max_d \max_{\|\boldsymbol{w}\|_2=1} < \boldsymbol{w}, \sum_{p\in\mathcal{P}} \varphi(I_p, d_p) > & (5.2)\\
&= \max_d || \sum_{p\in\mathcal{P}} \varphi(I_p, d_p)||_2, & (5.3)
\end{aligned}$$

where $\varphi(I, d)$ is the set of linear coefficients of the function $\boldsymbol{w} \rightarrow s(T, I)$ when the deformation is fixed at d.

We can see in the last line of this calculus that the original problem is equivalent to optimizing a new objective function on all deformations simultaneously. Therefore it would be interesting to develop new algorithms that perform joint alignment.

5.3 Other work

During this phD, we have also worked on action detection and localization [Duchenne et al., 2009b]. It is not related to deformation, so we did not mention it during the main chapters of this thesis. However, it is highly related to the previously mentioned points of part discovery and joint alignment.

The problem addressed in this article is as follow: Given videos with scripts, we are able to use text processing techniques to localize some actions. However, we obtain a very low temporal precision. So, we obtain many large temporal windows containing a common action. The proposed algorithm uses a simple discriminative clustering technique to find out the precise location of this action in each window. Then, we show that this helps to train a more accurate action detector.

An interesting direction for future work would be to combine this method with a deformable model (both spatially and temporally). Indeed, the same type of action can occur with different pace, different viewpoints, and different people. Explicitly using deformation could help to cope with this high intra-class variability.

Appendix A

Power iterations for unit norm rows

In this appendix, we consider the problem of maximizing a convex function on V on a product of ℓ_2-spheres in dimension N_2, i.e., on the set $\mathcal{C}_2$. The following proposition extends the result of Regalia and Kofidis [2000] from spheres to products of spheres. We consider the general algorithm:

Input: Convex function f
Output: $V = [v_1^T; \ldots; v_{N_1}^T]$ stationary point of f in $\mathcal{C}_2$.
initialize V randomly ;
repeat
    $V \leftarrow \nabla f(V)$;
    $V \leftarrow [\frac{1}{\|v_1\|_2} v_1^T; \ldots; \frac{1}{\|v_{N_1}\|_2} v_{N_1}^T]$;
until convergence ;

Algorithm 7: Power iteration for maximizing a convex function f over $X \in C_2$.

Proposition A.0.1 *If the function f is differentiable on $\mathbb{R}^{N_1 \times N_2}$ and strictly convex, Algorithm 5 is an ascent method and converges to a stationary point of f on $\mathcal{C}_2$.*

Proof

We refer to the i_1-th row of any matrix v as v_{i_1}. Given v^0 in $\mathcal{C}_2$, one iteration of Algorithm 5 applied to v^0 leads to the matrix v^1 with i_1-th row equal to $v^1_{i_1} = \nabla f(v^0)_{i_1} / \|\nabla f(v^0)_{i_1}\|_2$. Since f is strictly convex, for all w in $\mathbb{R}^{N_1 \times N_2}$, $f(w) \geqslant f(v^0) + \sum_{i_1} \nabla f(v^0)_{i_1}^\top (w_{i_1} - v^0_{i_1})$, with equality if and only if $w = v^0$. We thus have:

$$\begin{aligned} f(v^1) &\geqslant f(v^0) + \textstyle\sum_{i_1} \nabla f(v^0)_{i_1}^\top (v^1_{i_1} - v^0_{i_1}) \\ &\geqslant f(v^0) + \textstyle\sum_{i_1} \nabla f(v^0)_{i_1}^\top (v^0_{i_1} - v^0_{i_1}) = f(v^0), \end{aligned}$$

because for each i_2, $\nabla f(v^0)_{i_1}^\top v^1_{i_1} = \|\nabla f(v^0)_{i_1}\|_2 \geqslant \nabla f(v^0)_{i_1}^\top v^0_{i_1}$. We have equalities above if and only if $v^1 = v^0$ and, for each i_1, $\nabla f(v^0)_{i_1}$ is equal to a positive constant times $v^0_{i_1}$. This shows that each iteration is increasing the cost function. Since f is continuous and $\mathcal{C}_2$ is compact, if we denote by v^t the sequence of iterations, the sequence $f(v^t)$ is non-decreasing and bounded, hence convergent. Since having $f(v^0) = f(v^1)$ implies $v^1 = v^0$, the sequence v^t is also converging, and its limit v^∞ is such that for each i_1, $\nabla f(v^\infty)_{i_1}$ is equal to a positive constant times $v^\infty_{i_1}$, i.e., v^∞ is a stationary point of f on the product of spheres $\mathcal{C}_2$ [Absil et al., 2008].

Bibliography

P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton Univ. Press, 2008.

H. A. Almohamad and S. O. Duffuaa. A linear programming approach for the weighted graph matching problem. *IEEE PAMI*, 15, 1993.

archive3d. http://archive3d.net/, 2008.

Nicholas Ayache and Olivier D. Faugeras. Hyper: A new approach for the recognition and positioning of two-dimensional objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:44–54, 1986. ISSN 0162-8828.

D.H. Ballard and C.M. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

A. Berg. *Shape Matching and Object Recognition*. PhD thesis, UC Berkeley, 2005.

A. Berg, T. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondence. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, volume II, pages 435–439, 2005a.

Alexander C. Berg, Tamara L. Berg, and Jitendra Malik. Shape matching and object recognition using low distortion correspondence. In *CVPR*, 2005b.

S. Birchfield. KLT: An implementation of the Kanade-Lucas-Tomasi feature tracker, 1998. URL `http://vision.stanford.edu/~birch/klt`.

O. Boiman, E. Schechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008.

L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *Proc. Int. Conf. Comp. Vision*, 2009.

Y-Lan Boureau, Francis Bach, Yann LeCun, and Jean Ponce. Learning mid-level features for recognition. In *CVPR*, 2010.

Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 23:1222–1239, 2001.

T. Caetano, L. Cheng, Q. V. Le, and A. J. Smola. Learning graph matching. In *ICCV*, 2007.

B. Caputo and L. Jie. A performance evaluation of exact and approximate match kernels for object recognition. *ELCVIA*, 8:15–26, 2009.

Robert L. Carroll. *Vertebrate Paleontology and Evolution*. W. H. Freeman and Company, 1988.

Chih C. Chang and Chih J. Lin. *LIBSVM: a library for support vector machines*, 2001. http://www.csie.ntu.edu.tw/ ∼cjlin/libsvm.

O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5), 2007.

CMU. http://vasc.ri.cmu.edu//idb/html/motion/house/index.html, 2005.

T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *NIPS 19*, 2007.

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.

N. Dalal and B Triggs. Histogram of oriented gradients for human detection. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, 2005a.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005b.

O. Duchenne, A. Joulin, and J. Ponce. A graph-matching kernel for object categorization. In *ICCV*, 2011a.

Olivier Duchenne, Francis Bach, InSo Kweon, , and Jean Ponce. A tensor-based algorithm for high-order graph matching. In *CVPR*, 2009a.

Olivier Duchenne, Ivan Laptev, Josef Sivic, Francis Bach, and Jean Ponce. Automatic annotation of human actions in video. In *ICCV*, pages 1491–1498, 2009b.

Olivier Duchenne, Francis Bach, InSo Kweon, , and Jean Ponce. A tensor-based algorithm for high-order graph matching. 2011b.

O. D. Faugeras and M. Hebert. A 3-d recognition and positioning algorithm using geometrical matching between primitive surfaces. In *Proceedings of the Eighth international joint conference on Artificial intelligence - Volume 2*, IJCAI'83, pages 996–1002. Morgan Kaufmann Publishers Inc., 1983.

P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61:55–79, 2005a.

P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *Int. J. of Comp. Vision*, 61(1):55–79, 2005b.

P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, 2008a.

P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Trans. Patt. Anal. Mach. Intell.*, 32(9), 2010.

Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008b.

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient belief propagation for early vision. *IJCV*, 70:41–54, 2006.

Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part based models. In *CVPR*, 2008c.

R. Fergus, Fei-Fei L., P. Perona, and A. Zisserman. Learning object categories from google's image search. In *Proc. Int. Conf. Comp. Vision*, 2005.

R. Fergus, P. Perona, and A. Zisserman. Weakly supervised scale-invariant learning of models for visual recognition. *IJCV*, 71:273–303, 2006.

R. Fergus, P. Perona, and A. Zisserman. Weakly supervised scale-invariant learning of models for visual recognition. *Int. J. of Comp. Vision*, 71(3):273–303, 2007.

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.

M. A. Fischler and R. A. Elschlager. The Representation and Matching of Pictorial Structures. *IEEE Transactions on Computers*, C-22:67–92, 1973a.

M.A. Fischler and R.A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, C-22:67–92, 1973b.

G. Frobenius. Ueber matrizen aus nicht negativen elementen. *Sitzungsber. Königl. Preuss. Akad. Wiss.*, page 456–477, 1912.

Peter Gehler and Sebastian Nowozin. On feature combination for multiclass object classication. In *ICCV*, 2009.

G. H. Golub and C. F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, 1996.

K. Grauman and T. Darrell. Pyramid Match Hashing: Sub-Linear Time Indexing Over Partial Correspondences. In *CVPR*, 2007.

G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical report, Caltech, 2007a.

G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007b. URL `http://authors.library.caltech.edu/7694`.

W.E.L. Grimson and T. Lozano-Perez. Model-based recognition and localization from sparse range or tactile data. *International Journal of Robotics Research*, 3(3):3–35, 1984.

W.E.L. Grimson and T. Lozano-Pérez. Localizing overlapping parts by searching the interpretation tree. *PAMI*, 9(4):469–482, 1987.

C. Gu, J. Lim, P. Arbelaez, and J.Malik. Recognition using regions. In *CVPR*, 2009.

Daniel P. Huttenlocher and Shimon Ullman. Object recognition using alignment. In *ICCV*, 1987a.

D.P. Huttenlocher and S. Ullman. Object recognition using alignment. In *ICCV*, 1987b.

Hiroshi Ishikawa. Exact optimization for markov random fields with convex priors. *PAMI*, 25:1333–1336, 2003.

H. Jegou, M. Douze, and C. Schmid. Improving bag-of-features for large scale image search. *IJCV*, 87:313–336, 2010.

A. Joulin, F. Bach, and J. Ponce. Multi-class cosegmentation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Ho Yub Jung, Kyoung Mu Lee, and Sang Uk Lee. Toward global minimum through combined local minima. In *ECCV*, 2008.

J. Kim and K. Grauman. Asymmetric region-to-image matching for comparing images with generic object categories. In *CVPR*, 2010.

P. Kohli, P. Mudigonda, and P. Torr. p^3 and beyond: Solving energies with higher order cliques. *CVPR*, 2007.

Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *PAMI*, 28:1568–1583, 2006.

Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts? *PAMI*, 26:147–159, 2004.

A. Kushal, C. Schmid, and J. Ponce. Flexible object models for category-level 3d object recognition. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, 2007.

M. Lades, J.C. Vorbruggen, J. Buhmann, J. Lange, C. von der Malsburg, R.P. Wurtz, and W. Konen. Distortion invariant object recognition in the dynamic link architecture. *Computers, IEEE Transactions on*, 42(3):300 –311, mar 1993. ISSN 0018-9340. doi: 10.1109/12.210173.

M. F. Land and R. D. Fernald. The Evolution of Eyes. *Annual Review of Neuroscience*, 15(1):1–29, 1992.

L. De Lathauwer, B. De Moor, and J. Vandewalle. On the best rank-1 and rank-(r1,r2,. . .,rn) approximation of higher-order tensors. *SIAM J. Matrix Anal. Appl.*, 21(4):1324–1342, 2000. ISSN 0895-4798. doi: http://dx.doi.org/10.1137/S0895479898346995.

S. Lazebnik, C. Schmid, and J. Ponce. A maximum entropy framework for part-based texture and object recognition. In *Proc. Int. Conf. Comp. Vision*, volume I, pages 832–838, 2005.

S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006a.

S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006b.

Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. A sparse texture representation using affine-invariant regions. In *CVPR*, pages 319–326, 2003.

D. Lemire. Streaming maximum-minimum filter using no more than three comparisons per element. *Nordic Journal of Computing*, 13(4):328–339, 2006.

M. Leordeanu and M. Hebert. A spectral technique for for correspondance problems using pairwise constraints. In *ICCV*, 2005a.

M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, 2005b.

M. Leordeanu, M. Hebert, and R. Sukthankar. Beyond local appearance: Category recognition from pairwise interactions of simple features. In *CVPR*, 2007.

Ce Liu, Jenny Yuen, Antonio Torralba, Josef Sivic, and William T. Freeman. Sift flow: Dense correspondence across different scenes. In *ECCV*, 2008.

D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(4): 91–110, 2004a.

David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004b.

D.G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60: 91–110, 2004c.

J. Maciel and J. Costeira. A global solution to sparse correspondence problems. *PAMI*, 25(2), 2003.

Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.

David M. Mount and Sunil Arya. http://www.cs.umd.edu/ mount/ann/, 2000.

R. Oliveira, R. Ferreira, and J. Costeira. Optimal multi-frame correspondence with assignment tensors. In *Proc. European Conf. Comp. Vision*, 2006.

I. Pratikakis, M. Spagnuolo, T. Theoharis, R. Veltkamp (editors, A. Godil, H. Dutagaci, C. Akgül, A. Axenopoulos, B. Bustos, M. Chaouch, P. Daras, T. Furuya, S. Kreft, Z. Lian, T. Napoléon, A. Mademlis, R. Ohbuchi, P. L. Rosin,

B. Sankur, T. Schreck, X. Sun, M. Tezuka, A. Verroust-blondet, M. Walter, and Y. Yemez. Shrec'09 track: Generic shape retrieval, 2009.

P. Pritchett and A. Zisserman. Wide baseline stereo matching. In *ICCV*, 1998.

P. A. Regalia and E. Kofidis. The higher-order power method revisited: convergence proofs and effective initialization. *ICASSP*, 2000.

Eleanor H. Rosch. Natural categories. *Cognitive Psychology*, 4(3):328–350, May 1973. ISSN 00100285. doi: 10.1016/0010-0285(73)90017-0.

C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *PAMI*, 19 (5):530–535, 1997.

Alexander Shekhovtsov, Ivan Kovtun, and Vaclav Hlavac. Efficient MRF deformation model for non-rigid image matching. *CVIU*, 112:91–99, 2008.

J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. Int. Conf. Comp. Vision*, volume 2, pages 1470–1477, 2003.

S. Todorovic and N. Ahuja. Learning subcategory relevances for category recognition. In *CVPR*, 2008.

S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *PAMI*, 10(5):695–703, 1988.

Wim Vanduffel, Roger B.H. Tootell, Aniek A. Schoups, and Guy A. Orban. The organization of orientation selectivity throughout macaque visual cortex. *Cerebral Cortex*, 12(6):647–662, 2002. doi: 10.1093/cercor/12.6.647. URL `http://cercor.oxfordjournals.org/content/12/6/647.abstract`.

M. Varma and D. Ray. Learning The Discriminative Power-Invariance Trade-Off. In *ICCV*, 2007.

A. Vedaldi. A matlab implementation of sift, 2008. URL `http://www.vlfeat.org/~vedaldi/code/sift.html`.

P. Viola and M.J. Jones. Robust real-time face detection. *Int. J. of Comp. Vision*, 57(2):137–154, 2004.

Martin Wainwright, Tommi Jaakkola, and Alan Willsky. Exact map estimates by (hyper)tree agreement. In *NIPS*, 2002.

C. Wallraven, B. Caputo, and A. Graf. Recognition with local features: the kernel recipe. In *ICCV*, 2003.

Gang Wu, Edward Y. Chang, and Zhihua Zhang. An analysis of transformation on non-positive semidefinite similarity matrix for kernel machines. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

J. Yang, Y. Li, Y. Tian, L. Duan, and W. Gao. Group-sensitive multiple kernel learning for object categorization. In *ICCV*, 2009a.

J. Yang, K. Yu, and T. Huang. Efficient highly over-complete sparse coding using a mixture model. In *ECCV*, 2010.

Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009b.

A. L. Yuille. Deformable Templates for Face Recognition. *Journal of Cognitive Neuroscience*, 3:59–70, 1991.

M. Zaslavskiy, F. Bach, and J.P. Vert. A path following algorithm for the graph matching problem. *IEEE Trans. Patt. Anal. Mach. Intell.*, 31(12):2227–2242, December 2009.

R. Zass and A. Shashua. Probabilistic graph and hypergraph matching. *CVPR*, 2008.

H. Zhang, A.C. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *CVPR*, 2006.

J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classifcation of texture and object categories: An in-depth study. *Int. J. of Comp. Vision*, 73(2):213–238, 2007a.

Jianguo Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *IJCV*, 73:213–238, 2007b.

Y. Zheng and D. Doermann. Robust point matching for nonrigid shapes by preserving local neighborhood structures. *PAMI*, 28(4):643, 2006. ISSN 0162-8828. doi: http://dx.doi.org/10.1109/TPAMI.2006.81.