Probabilistic, Features-based Object Recognition

Thesis by

Pierre Moreels

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2008

(Defended October 12, 2007)

To my mother,

Your memory will ever be with me.

Acknowledgements

First, I wish to thank my thesis advisor, Professor Pietro Perona, for all his help and advice during my stay at Caltech. Thanks to Pietro, I have enjoyed all these years of PhD research, and hope to continue working in the field of computer vision.

I wish to thank Pr. Yaser Abu-Mostafa, Pr. Christof Koch, Dr. Larry Matthies and Dr. Baback Moghaddam, for accepting to be on my thesis defense committee.

I wish to thank Francois Fleuret, Donald Geman, David Lowe, Mario Munich, Michel Paindavoine, Jean Pallo, Josiane Zerubia and Andrew Zissermann for very fruitful discussions and useful suggestions for my research.

I wish to thank all my lab-mates from the Vision lab. Anelia, very nice behind her apparent bad mood. Claudio, always ready to help for computer-related issues. Marco, the reference for any math- or manga- related question. Fei Fei and Rob, always full of energy. Seigo, with whom I enjoyed countless japanese dinners and surfing days. Lihi, whose ideas were always right. The new lab-members, Greg, Merrielle and Ryan, full of enthusiasm. I started recently to collaborate with my office-mate and friend Alex Holub, working with him has been very enjoyable, I hope we will keep doing research together.

I wish to thank my friend Christophe Basset, with whom I spent many days on a bicycle. Sunny

days, windy days, rainy days, all of these are good memories.

I wish to thank Jean-Yves Bouguet, with whom I worked at Intel during one happy summer, Susan Smrekar, who gave me the chance to work with her at JPL, and Chris Assad, Jay Hanan and Michael Maire, for collaborating with me on various projects.

I wish to thank my parents and sisters. My parents' advice was the reason I came to California, initially to work for one-year at JPL. My parents were always extremely supportive and of good advice. Sadly, the illness took my mother before she could come to my graduation.

Last, I wish to thank my girlfriend Chihiro. During all these years she waited patiently, enjoying the few moments of happiness when we could see each other, but living far apart most of the year. The day we met was so many years ago that I can barely remember, and I hope we will spend many more years together.

Abstract

Object recognition is of fundamental importance in computer vision. In a few years, pedestrian detection, car detection, and more generally scene recognition will likely be reliable enough to allow fully-automated car navigation, and the human driver will be relegated to the back seat to sip his coffee.

In this thesis we are interested in recognizing individual objects and categories. In order to reduce the volume of information one has to process, images are characterized by sets of features. These features, also called interest points, are targeted at image locations with high local information content. Various systems for detecting interest points and for describing the local image appearance near these points, have been proposed in the last two decades. We investigate which combinations from this plethora of detectors and descriptors, are most suited for object recognition tasks.

On to the problem of object recognition, we are first interested in recognizing individual objects. In a few years, one can imagine that customers in shops, will take with their cell phone a picture of a product that looks interesting, send it to a remote server with a huge database of individual objects, and get back information about that specific product. We propose a system for individual object recognition, inspired from previous work on coarse-to-fine recognition. All steps

of the recognition process are translated into principled probabilistic terms, which allows us to outperform a state-of-the-art commercial system for individual recognition.

Regarding categories, faces are probably the category that has received the most attention in computer vision literature. Here we propose a system to recognize images of the same individual in large databases of images. This can be of high interest when looking for images of a given person over the internet. Our method's advantage is that it works on real-world images, as opposed to the face databases from the literature, collected in laboratories with controlled lighting, pose and background conditions.

Finally, we are interested in recognition of object categories in general. Using support vector machines for the classification task, we propose a features-based kernel that improves recognition performance on object categories.

Contents

Chapter 1

Introduction

Learning from examples of seen objects, and being able to recognize these objects in a new environment, is one great capability of the human visual system and the human brain. Despite tremendous progress in the recent computer vision literature, the performance of state-of-the-art software is still far behind human performance.

Several types of tasks are of interest in object recognition. The first one, *individual object recognition*, consists of identifying the same exact object in training and test images. One might be interested in finding e.g. the same exact brand logo, the same exact building, or the same exact person. In the past, this problem has been addressed in terms of registering a query image to the best-matching image from the database. Other groups viewed this as a stereo vision problem with wide base-line. In recent studies, the query scenes deviate from these studies as they are allowed to contain more than one object from the database. Besides, image scenes do not only contain objects of interest (*foreground*), but also a *background* part. One needs to discriminate between both, and not only register the foreground part between pairs of images, but also be able to discard the background. Some applications of individual object recognition available on a commercial basis include fingerprint or iris identification systems, used as access code as an alternative to passwords.

The second task in object recognition is category recognition, i.e. recognizing that an image contains a person without identifying which specific person, or a plant without identifying exactly what type of plant. This task has received a lot of interest in recent years, with constant progress of the recognition performance on data-sets that have become standard, like the "Caltech-101". The category that has seen the highest number of studies is the Face category, so that recently, commercial digital cameras capable of face recognition in order to improve focus accuracy, or printers with face recognition in order to improve color rendering of skin tones, have started to

appear on the market.

A transversal, but fundamental issue, is how to represent images. While many systems use raw image information and feed it to the recognition system (e.g. neural-network based applications), a recently popular approach - which we will adopt in this thesis - is to represent the image via *points of interest*, or *features*. These points of interest characterize the image by a set of privileged locations. The local image appearance or geometry around these locations is then encoded in a descriptor, and the set of descriptors is used to represent the image. The advantage of this representation, is to reduce the amount of information that needs to be processed in an image, by focusing only on a subset of most salient regions. Besides, depending on the encoding used to represent the local appearance or geometry, one might gain invariance with respect to nuisances like illumination change, rotation or scaling, affine or projective transformations... One limitation of this approach is that one has to be careful with regard to the locations selected as points of interest. For example, while a uniform sky has no salient points, it might be useful to still assign features to it, when addressing the problem of scene recognition. Ultimately, if infinite computational power was available, it would be best to characterize images using dense grids of points of interest.

This thesis is organized as follows:

The second chapter investigates which combinations of features detectors and features descriptors are best suited for use in object recognition tasks. A number of detectors and descriptors have been proposed in the past two decades. The literature on features contains comparison studies, but they have all focused on flat scenes, as ground truth regarding pairs of matching features is easy to obtain. As a result, features were reported to have an unrealistically high stability across images. In contrast, we propose a system to establish automatically ground truth matches between

triplets of images of 3D objects. The features obtained from various combinations of detectors and descriptors are matched using this system, to evaluate how stable features are across viewpoints.

The third chapter focuses on individual object recognition. We address the detection problem, which aims at identifying the objects from the database that are present in a query scene, along with their location and pose. Motivated by recent studies on coarse-to-fine recognition systems, we use a cascade of simple detectors, organized from coarse resolution to fine resolution, that filter quickly through the space of possible hypotheses. Each step is interpreted with a probabilistic model - this allows us to select parameters automatically rather than tuning them manually, and provides hypotheses scores in order to decide if hypotheses should be accepted or rejected.

The fourth chapter is also interested in individual objects, for the specific case of faces. We want to identify the images of a same person, in sets of images that contain numerous irrelevant faces as well. We characterize face images by sets of features, and investigate which features are most relevant for the retrieval task. We also learn a distance metric in face-space, that optimizes the quantity of interest: distances between images of a same person need to be as small as possible, while distances between images of unrelated persons should be large.

Finally, the fifth chapter looks at the classification problem for object categories. We build upon the successful pyramid-match kernel - a features-based histogram-type of kernel that was recently designed and combined with support vector machines, for the task of object classification. We propose a principled, broader family of kernels, and show that they improve performance compared to the original kernel.

Chapter 2

Features detectors and descriptors

2.1 Abstract

We explore the performance of a number of popular feature detectors and descriptors in matching 3D object features across viewpoints and lighting conditions. To this end we design a method, based on intersecting epipolar constraints, for providing ground truth correspondence automatically. These correspondences are based purely on geometric information, and do not rely on the choice of a specific feature appearance descriptor. We test detector-descriptor combinations on a database of 100 objects viewed from 144 calibrated viewpoints under three different lighting conditions. We find that the combination of Hessian-affine feature finder and SIFT features is most robust to viewpoint change. Harris-affine combined with SIFT and Hessian-affine combined with shape context descriptors were best respectively for lighting change and change in camera focal length. We also find that no detector-descriptor combination performs well with viewpoint changes of more than 25–30°.

2.2 Introduction

Detecting and matching specific features across different images has been shown to be useful for a diverse set of visual tasks including stereoscopic vision [TG00, ea02b], vision-based simultaneous localization and mapping (SLAM) for autonomous vehicles [SLL02, Low04], mosaicking images [BL03], and recognizing objects [SM97, Low04]. This operation typically involves three distinct steps. First a 'feature detector' identifies a set of image locations presenting rich visual information and whose spatial location is well defined. The spatial extent or 'scale' of the feature may also be identified in this first step, as well as the local shape near the detected location [MS02, ea02b, TG00, TG04]. The second step is 'description': a vector characterizing local visual appearance is computed from the image near the nominal location of the feature. 'Matching' is the third step: a given feature is associated with one or more features in other images. Important aspects of matching are metrics and criteria to decide whether two features should be associated, and data structures and algorithms for matching efficiently.

The ideal system will be able to detect a large number of meaningful features in the typical

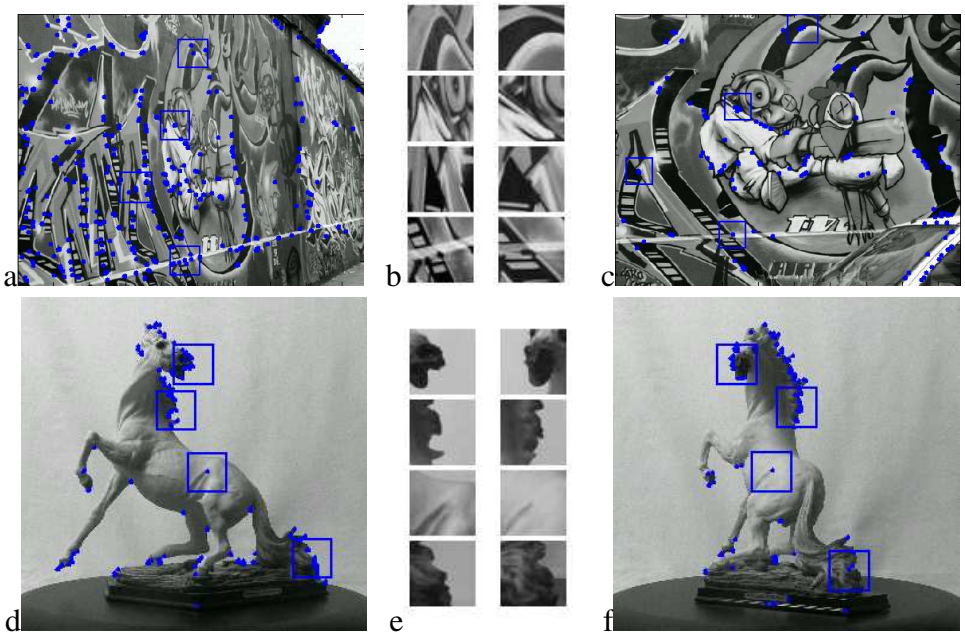


Figure 2.1: (Top row) Large ($\approx 50°$) viewpoint change for a flat scene. Many interest points can be matched after the transformation. The appearance change is modeled by an affine transformation. b) shows four 40×40 patches before and after viewpoint change — images courtesy of K.Mikolajczyk. (Bottom row) Similar $50°$ viewpoint change for a 3D scene. Many visually salient features are associated with locations where the 3D surface is irregular or near boundaries. The local geometric structure of the image around these features varies rapidly with viewing direction changes, which makes matching features more challenging because of occlusion and changes in appearance. In particular, the appearance of the patches shown in e) varies significantly with the change in viewpoint. This change is difficult to model.

image, and will match them reliably across different views of the same scene/object. Critical issues in detection, description, and matching are robustness with respect to viewpoint and lighting changes, the number of features detected in a typical image, the frequency of false alarms and mismatches, and the computational cost of each step. Different applications weigh these requirements differently. For example, viewpoint changes more significantly in object recognition, SLAM, and wide-baseline stereo than in image mosaicking, while the frequency of false matches may be more critical in object recognition, where thousands of potentially matching images are considered, rather than in wide-baseline stereo and mosaicking where only few images are present.

A number of feature detectors [ea02b, HS88, Bea78, KZB04, MS02, CP84], feature descriptors [Low04, FA91, BMP02, KS04], and feature matchers [SM97, Low04, CJ04, MP04] have been proposed in the literature. They can be variously combined and concatenated to produce different

systems. Which combination should be used in a given application? A couple of studies explore this question. Schmid [SM97] characterized and compared the performance of several features detectors. Recently, Mikolajczik and Schmid [MS05] focused primarily on the descriptor stage. For a chosen detector, the performance of a number of descriptors was assessed. These evaluations of interest point operators and feature descriptors, have relied on the use of images of flat scenes, or in some cases synthetic images. The reason is that in these special cases the transformation between pairs of images can be computed easily, which is convenient to establish ground truth.

However, the relative performance of various detectors can change when switching from planar scenes to 3D images (see Figures 2.1, 2.17, and [FB04]). Features detected in an image are generated in part by surface markings, and in part by the geometric shape of the object. The former are often associated with smooth surfaces, they are usually located far from object boundaries and have been shown to have a high stability across viewpoints [SM97, MS05]. Their deformation may be modeled by an affine transformation, hence the development of affine-invariant detectors [LG97, MS02, SZ01, TG00, TG04]. The latter are associated with high surface curvature and are located near edges, corners, and folds of the object. Due to self-occlusion and complexity of local shape, these features have a much lower stability with respect to viewpoint change. It is difficult to model their deformation without a full 3D model of the shape.

The present study generalizes the analyses in [SM97, KS04, MS05] to 3D scenes.[1] We evaluate the performance of feature detectors and descriptors for images of 3D objects viewed under different viewpoint, lighting, and scale conditions. To this effect, we collected a database of 100 objects viewed from 144 different calibrated viewpoints under 3 lighting conditions. We also developed a practical and accurate method for establishing automatically ground truth in images of 3D scenes. Unlike [FB04], ground truth is established using geometric constraints only, so that the feature/descriptor evaluation is not biased by the choice of a specific descriptor and appearance-based matches. Besides, our method is fully automated, so that the evaluation can be performed on a large-scale database, rather than on a handful of images as in [MS05, FB04].

Another novel aspect is the use of a metric for accepting/rejecting feature matches due to Lowe [Low04]; it is based on the ratio of the distance of a given feature from its best match

[1] An early version of this work was presented in [MP05].

vs. the distance to the second-best match. This metric has been shown to perform better than the traditional 'distance-to-best-match'.

Section 2.3 presents the previous work on evaluation of features detectors and descriptors. In Section 2.4 we describe the geometrical considerations which allow us to construct automatically a ground truth for our experiments. Section 2.5 presents our laboratory setup and the database of images we collected. Section 2.6 describes the decision process used in order to assess performances of detectors and descriptors. Section 2.7 presents the experiments. Section 2.8 contains our conclusions.

2.3 Previous work

The first extensive study of features stability depending on the feature detector being used, was performed by Schmid and Mohr [SMB00]. The database consisted of images of drawings and paintings photographed from a number of viewpoints. The authors extracted and matched interest points across pairs of views. The different views were generated by rotating and moving the camera as well as by varying the illumination. Since all scenes were planar, the transformation between two images taken from different viewpoints was a homography. Ground truth, i.e., the homography between pairs of views, was computed from a grid of artificial points projected onto the paintings. The authors measured the performance by the repeatability rate, i.e., the percentage of locations selected as features in two images.

Mikolajczyk et al. [ea05c] performed a similar study of affine-invariant features detectors. This time, most images of the database consisted of natural scenes. However, the scenes were either planar (e.g., graffiti on a wall), or viewed from a large distance, such that the scene appeared flat. Therefore the authors could model the ground truth transformation between a pair of views with a homography as was previously done in [SMB00]. This ground truth homography was computed using manually selected correspondences, followed by an automatic computation of the residual homography.

Note that the performance criterion used in both of these studies is well defined only when a small number of features is detected in each image. If the number of interest points is arbitrary,

one could indeed consider a trivial interest point operator that selects every point in the image to be a new feature. The performance of this detector would be excellent in terms of stability of the features location. In particular for planar images such as considered by [ea05c, SMB00], this detector would reach 100% stability. This perfect stability still holds if the detector selects a dense grid of points in the image. This argument illustrates the necessity of including the descriptor stage in performance evaluation.

Fraundorfer and Bischof [FB04] compared local detectors on real-world scenes. Ground truth was established in triplets of views. Correspondences were first identified between grids of points sampled densely in two close views: matches were obtained by nearest neighbor search in appearance space. The coordinates of pairs of matching points in the first two images, were transferred on the third image via the trifocal tensor. The test scenes used for detector evaluation were piecewise flat (building, office space).

Mikolajczyk and Schmid [MS05] provided a complementary study where the focus was not anymore on the detector stage but on the descriptor, i.e., a vector characterizing the local appearance at each detected location. Two interest points were considered a good match if their appearance descriptors were closer than a threshold t in appearance space. Matches that were accepted were compared to ground truth to determine if they were true matches or false alarms. Ground truth was computed as in their previous study [ea05c]. By varying the acceptance threshold t, the authors generated recall-precision curves to compare the descriptors. If the value of t is small, the user is very strict in accepting a match based on appearance, which leads to a high precision but a poor recall. If t is high, all candidate correspondences are accepted regardless of their appearance. Correct matches are accepted (high recall), as well as lots of false positives, leading to lower precision.

Ke and Sukthankar [KS04] used a similar setup to test their PCA-SIFT descriptor against SIFT. Test features were indexed into a database, the resulting matches were accepted based on a threshold t on quality of the appearance match. Ground truth was provided by labeled images, or by using synthetic data. The threshold t was varied to obtain recall-precision curves.

A recent study by Mikolajczyk et al. [MLS05] compared detectors and descriptors when they are integrated in the framework of the full recognition system from [LSS05]. They assessed the

performance from the performance of the overall system. The integration within a complete recognition method has the advantage of computing directly the bottom line performance in recognition. However, the scores might depend heavily on the architecture of the recognition system and may not be generalized to other applications such as large baseline stereo, SLAM, and mosaicking.

2.4 Ground truth

In order to evaluate a particular detector-descriptor combination we need to calculate the probability that a feature extracted in a given image can be matched to the corresponding feature in an image of the same object/scene viewed from a different viewpoint. For this to succeed, the feature's physical location must be visible in both images, the feature detector must detect it in both cases with minimal positional variation, and the descriptor of the features must be sufficiently close. To compute this probability we must have a ground truth telling us if any tentative match between two features is correct or not. Conversely, whenever a feature is detected in one image, we must be able to tell whether in the corresponding location in another image a feature was detected and matched.

We establish ground truth by using epipolar constraints between triplets of calibrated views of the objects. The motivation comes from stereoscopic imagery: if the position of a point is identified in two calibrated images of a same scene, the position in 3D space of the physical point may be computed, and its location may be predicted in any additional calibrated image of the same scene.

We distinguish between a *reference* view (A in figure 2.2 and figure 2.3), a *test* view C, and an *auxiliary* view B. Given one *reference feature* f^A in the reference image, any feature in C that matches the reference feature must satisfy the constraint of belonging to the corresponding reference epipolar line l^{AC}. This excludes most potential matches but not all of them (in our experiments, typically 5–10 features remain out of 500–1000 features in image C). We make the test more stringent by imposing a second constraint. In the auxiliary image B, an epipolar line l^{AB} is associated to the reference feature f^A. Again, f^A has typically 5–10 potential matches along l^{AB}, each of which in turn generates an 'auxiliary' epipolar line $l^{BC}_{1...10}$ in C. The intersection of the primary (l^{AC}) and auxiliary ($l^{BC}_{1...10}$) epipolar lines in C identify a number of small matching

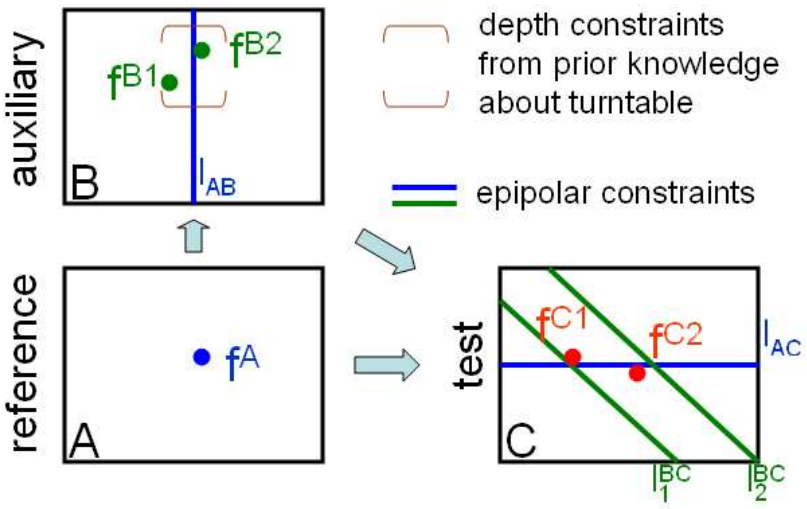


Figure 2.2: Diagram explaining the geometry of our three-cameras arrangement and of the triple epipolar constraint.

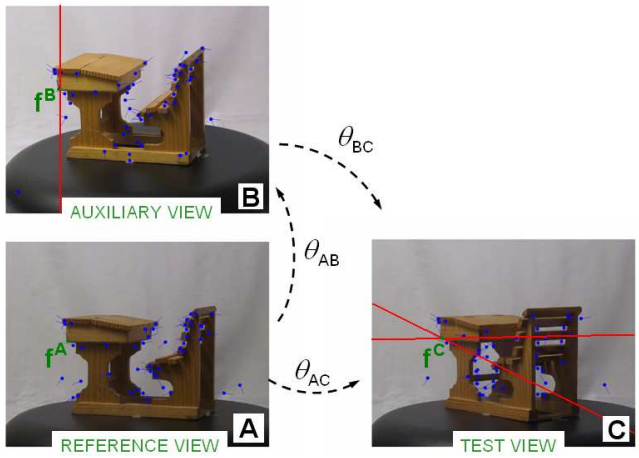


Figure 2.3: Example of matching process for one feature.

regions, in which only zero or one feature is typically detected. As we will make clear later, when a matching feature is found, this indicates with overwhelming probability that it is the correct match.

Note that the geometry of our acquisition system (figures 2.2 and 2.3) does not allow the degenerate case where the reference point is on the trifocal plane. In this case, the triangle (reference camera/auxiliary camera/test camera) would be a degenerate triangle and the epipolar transfer would fail [HZ00]. An alternative would be to use the trifocal tensor to perform the point transfer

Figure 2.4: Photograph of our laboratory setup. Each object was placed on a computer-controlled turntable which can be rotated with $1/50$ degree resolution and 10^{-5} degree accuracy. Two computer-controlled cameras imaged the object. The cameras were located 10° apart with respect to the object. The resolution of each camera is 3 M pixels. In addition to a neon tube on the ceiling, two photographic spotlights with diffusers are alternatively used to create 3 lighting conditions.

[SW95, HZ00] (transfer using the trifocal tensor avoids the degeneracy of epipolar transfer).

The benefit of using the double epipolar constraint in the test image is that any correspondence — or lack thereof — may be validated with extremely low error margins. The cost is that only a fraction (50–70%) of the reference features have a correspondence in the auxiliary image, thus limiting the number of features triplets that can be formed.

2.5 Experimental setup

2.5.1 Photographic setup and database

Our acquisition system consists of 2 cameras taking images of objects on a motorized turntable (see figure 2.4). We used inexpensive off-the-shelf Canon Powershot G1 cameras with a 3 MPixel resolution. The highest focal length available on the cameras — 14.6 mm — was used in order to minimize distortion (0.5% pincushion distortion with the 14.6 mm focal length). A change in viewpoint is performed by the rotation of the turntable. The lower camera takes the reference view and the top camera the auxiliary view, then the turntable is rotated and the same lower camera takes the test view. Each acquisition was repeated with 3 different lighting conditions obtained with a combination of photographic spotlights and diffusers. The images were converted to gray-scale using Matlab's command rgb2gray (keeps luminance, eliminates hue and saturation).

The baseline of our stereo rig, or distance between the reference camera and the auxiliary camera, is a trade-off parameter between repeatability and accuracy. On one hand, we would like to set these cameras very close to each other, in order to have a high feature stability (also called repeatability rate) between the reference view and the auxiliary view. On the other hand, if the baseline is small the epipolar lines intersect in the test view C with a very shallow angle, which lowers the accuracy in the computation of the intersection. We chose an angle of 10° between reference camera and auxiliary camera; with this choice, the intersection angle between both epipolar lines varied between 65° and 6° when the rotation of the test view varied between 5° and 60°.

The database consisted of 100 different objects. Figures 2.5 – 2.7 show some examples from this database. The objects were chosen to include both heavily textured objects (pineapple, globe) and objects with a more homogenous surface (bananas, horse). The only constraint on the objects' identity concerned their size. They had to be small enough to fit on the turntable (40 cm diameter), but needed to be large enough so that their image would generate a significant number of features. Aside from these constraints, the objects were selected randomly. Most objects were 3-dimensional, with folds and self-occlusions, which are a major cause of feature instability in real-world scenes. A few piecewise-flat objects (e.g., box of cereal, bottle of motor oil) were also present. The database is available at http://www.vision.caltech.edu.

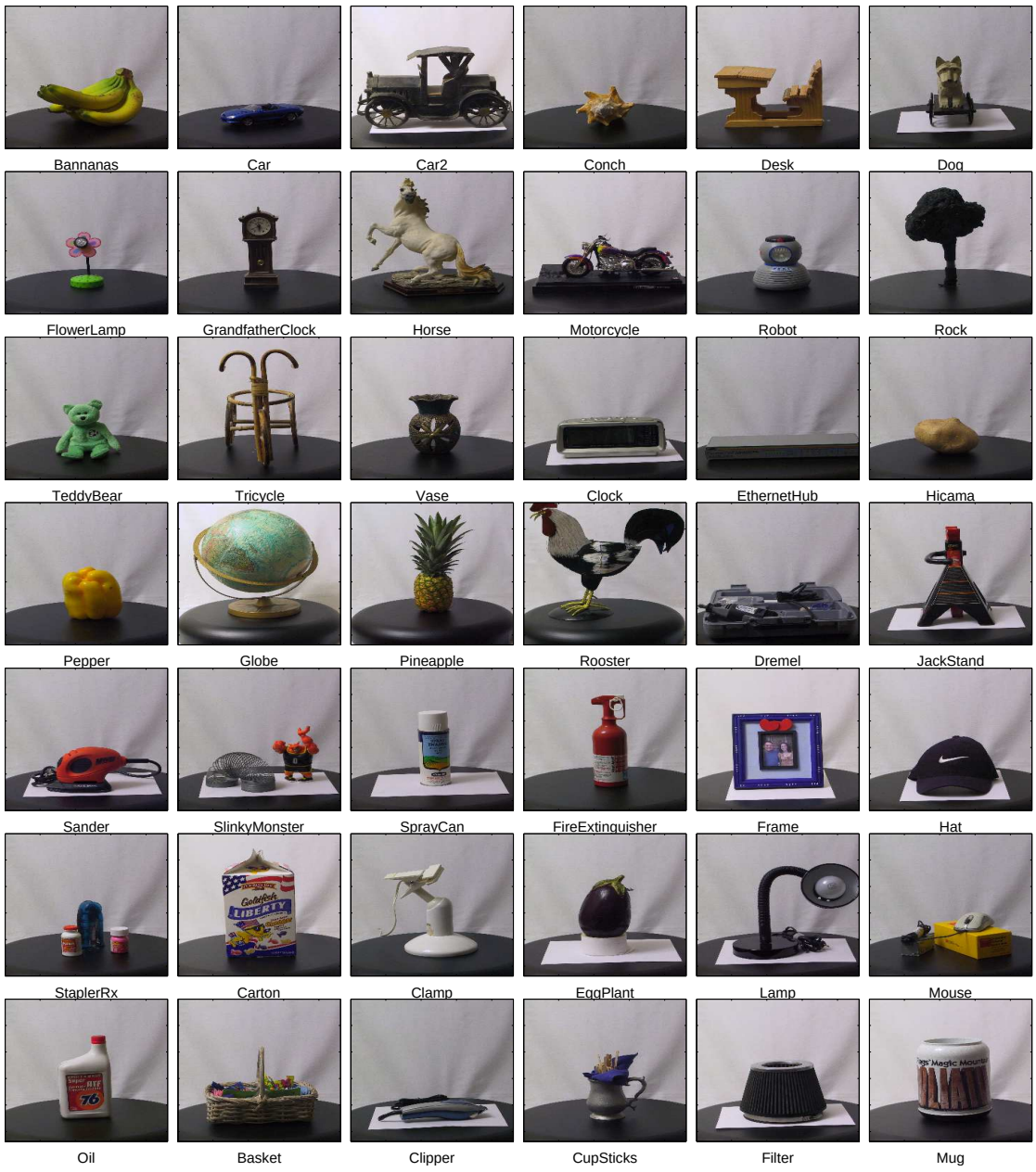


Figure 2.5: Our calibrated database consists of photographs of 100 objects which were imaged in three lighting conditions: diffuse lighting, light from left, and light from right. Two people chose objects from the set they met in their daily life. The objects had to fit on the turntable and within the camera's field of view. The range of shape statistics was explored, ranging from wireframe-type objects (Tricycle) to irregular 3D objects (Car2, Desk). Textured objects (Pineapple, Globe) were included as well as homogenous ones (Hicama, Pepper). A few piecewise flat objects were imaged as well (Carton, Oil). Each object was photographed by two cameras located above each over, 10° apart. 42 objects from the database are displayed.

Figure 2.6: Each object was rotated with 5° increments and photographed at each orientation with both cameras and three lighting conditions for a total of $72 \times 2 \times 3 = 432$ photographs per object. Eight such photographs (taken every 45°) are shown for one of our objects.

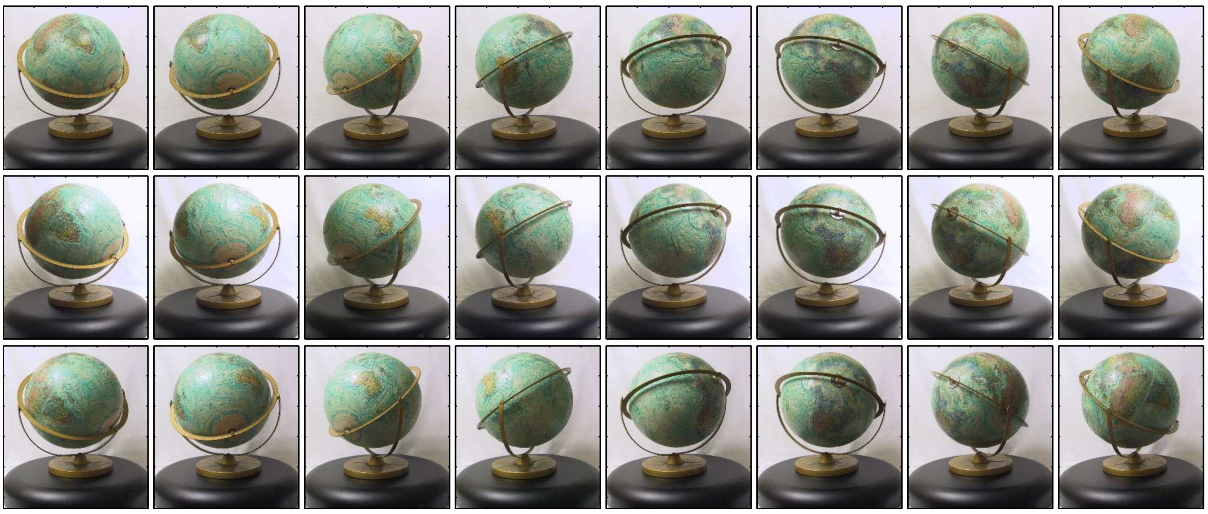


Figure 2.7: Three lighting conditions were generated by turning on a spotlight (with diffuser) located on the left hand side of the object, then a spotlight located on the right hand side, then both. This figure shows 8 photographs for each lighting condition.

2.5.2 Calibration

The calibration images were acquired using a checkerboard pattern. The corners of the checkerboard were first identified by the Harris interest point operator, then both cameras were automatically calibrated using the calibration routines in Intel's Open CV library, including the estimation of the radial distortion [Bou99], which was used to map features locations to their exact perspective projection.

The uncertainty on the position of the epipolar lines was estimated by Monte Carlo perturbations of the calibration patterns. Hartley and Zisserman [HZ00] showed that the envelope of the epipolar lines obtained when the fundamental matrix varies around its mean value is a hyperbola.

Rather than computing this curve analytically, we computed it by Monte Carlo simulation. The calibration patterns were perturbed by a random amount between 0 and 3 pixels (uniform distribution). This perturbation was performed by shifting the position of the corners in the checkerboard pattern after detection. This quantity was chosen so that it would produce a reprojection error on the grid's corners that was comparable to the one observed during calibration. The perturbation was followed by the calibration optimization.

For each point f^A of the reference image, the Monte-Carlo process leads to a bundle of epipolar lines in the test image, whose envelope is the hyperbola of interest. From our Monte Carlo simulation we found that the width between the two branches of the hyperbola varied between 3 and 5 pixels. The area inside the hyperbola defines the region allowed for detection of a match to f^A (a similar condition holds between reference and auxiliary images, and between auxiliary and test images).

2.5.3 Detectors and descriptors

2.5.3.1 Detectors

A large number of the traditional feature detectors follow the same general scheme. In a first step a *saliency map* is computed, which is a local function of the image. The saliency is a measure of the local contrast or local information content in the image. Patches with a high contrast (typically corners or highly textured areas) are expected to be detected and localized reliably between different images of the scene, therefore the local maxima of the saliency map are selected as features. This process is repeated after subsampling the image iteratively, to provide a multi-scale detector. In order to provide some invariance to noise, only local maxima that exceed a given threshold are selected.

— The Forstner detector [For86] relies on first-order derivatives of the image intensities. It is based on the second-order moment matrix (also called squared gradient matrix)

$$\mu = \begin{bmatrix} L_x^2 & L_x \cdot L_y \\ L_x \cdot L_y & L_y^2 \end{bmatrix} \text{ where } L_x = \frac{\partial I}{\partial x} \text{ and } L_y = \frac{\partial I}{\partial y} \tag{2.1}$$

and selects as features the local maxima of the function $det(\mu)/tr(\mu)$. The second-order moment matrix is a local measure of the variation of the gradient image. It is usually integrated over a small window in order to obtain robustness to noise and to make it a matrix of rank 2 (our implementations used a small 5×5 window)

Several other feature detectors use the second-order moment matrix as well. The popular Harris detector [HS88] selects as features the extrema of the saliency map defined by $det(\mu) - 0.04 \cdot tr^2(\mu)$. The Lucas-Tomasi-Kanade feature detector [LK81, ST94] averages μ over a small window around each pixel, and selects as features the points that maximize the smallest eigenvalue of the resulting matrix. The motivation for these three detectors is to select points where the image intensity has a high variability both in the x and the y directions.

— The Hessian detector [Bea78] is a second-order filter. The saliency measure is here the negative determinant of the matrix of second-order derivatives.

— When the interest point detection is performed at multiple scales, one can combine the locations obtained at all scales.

— The difference-of-Gaussians detector [CP84, Lin94] selects scale-space extrema of the image filtered by a difference of Gaussians. Note that the difference-of-Gaussians filter can be considered as an approximation of a Laplacian filter, i.e. a second-order derivative-based filter.

— The Kadir-Brady detector [KZB04] selects locations where the local entropy has a maximum over scale and where the intensity probability density function varies fastest.

— MSER features [ea02b] are based on a watershed flooding [VS91] process performed on the image intensities. The authors look at the rate of expansion of the segmented regions, as the flooding process is performed. Features are selected at locations of slowest expansion of the catchment basins. This carries the idea of stability to perturbations, since the regions are virtually unchanged over a range of values of the 'flooding level.'

Characteristic scale — Interest point detectors are typically used at multiple scales obtained by down-sampling the initial image. The scale associated to the feature is the scale at which it was detected. A refinement for scale selection consists of computing the response of the image to a function. The local extremum over scale of the response at the detected location is then selected as the characteristic scale [Lin98]. Functions used to filter the image include the square gradient,

the Laplacian and the difference-of-Gaussians. In [Low04], Lowe uses the same difference-of-Gaussians function to detect the interest point locations, i.e., the detector is used to perform a search over scale-space.

Affine invariance — Processes that warp the image locally around the points of interest have been developed and used by [BG98, LG97, MS02, SZ01] in order to obtain a patch invariant to affine transformations prior to computation of the descriptor. The second-order moment matrix is used as an estimation of the parameters of the local shape around the detected point, i.e., a measure of the local anisotropy of the image. The goal is to deform the shape of the detected region so that it is invariant to affine transformations. The affine rectification process is an iterative warping method that reduces the image's local second-order moment matrix at the detected feature location to have identical eigenvalues.

Tuytelaars and Van Gool [TG00, TG04] adopt an intensity-based approach. The candidate interest points are local extrema of the intensity. The affine-invariant region around such a point is bounded by the points that are local extrema of a contrast measure along rays emanating from the interest point. In [TG04] they propose another method based on geometry, where the affine-invariant region is extracted by following the edges next to the interest point.

Regarding speed, the detectors based on Gaussian filters and their derivatives (Harris, Hessian, difference-of-Gaussians) are fastest, they can easily be implemented very efficiently using the recursive filters introduced in [VYV98]. The MSER detector [ea02b] has a comparable running time. The detection process typically takes one second or less for a 3 GHz machine on a 1024×768 image. If one uses the affine rectification process, computation is more expensive, a similar detection takes of the order of 10 seconds. The most expensive detector is the Kadir-Brady detector, which takes of the order of 1 minute on a 800x600 image.

2.5.3.2 Descriptors

The role of the descriptor is to characterize the local image appearance around the location identified by the feature detector. Invariance to noise is usually obtained by low-pass filtering. Partial invariance to lighting conditions is obtained by considering image derivatives instead of the raw greylevels.

— SIFT features [Low04] are computed from gradient information. Invariance to orientation is obtained by evaluating a main orientation for each feature and rotating the local image according to this orientation prior to the computation of the descriptor. Local appearance is then described by histograms of gradients, which provides a degree of robustness to translation errors.
— PCA-SIFT [KS04] computes a primary orientation similarly to SIFT. Local patches are then projected onto a lower-dimensional space by using PCA analysis.
— Steerable filters [FA91] steer derivatives in a particular direction given the components of the local jet, e.g., steering derivatives in the direction of the gradient makes them invariant to rotation. Scale invariance is achieved by using various filter sizes.
— Differential invariants [SM97] combine local derivatives of the intensity image (up to third-order derivative) into quantities which are invariant with respect to rotation.
- The shape context descriptor [BMP02] is comparable to SIFT, but based on edges. Edges are extracted with the Canny filter, their location and orientation are then quantized into histograms using log-polar coordinates.

The implementations used for the experiments in Section 2.7 were our own for the derivative-based detectors, Lowe's for the difference-of-Gaussians (includes scale selection in scale-space), and the respective authors' for MSER and the Kadir detector. Mikolajzyk's version of the affine rectification process was used. Lowe's code was used for SIFT, Ke's for PCA-SIFT, and Mikolajczyk's for steerable filters, differential invariants, and shape context.

2.6 Performance evaluation

2.6.1 Matching criteria

The performance of the different combinations of detectors and descriptors was evaluated on a feature matching problem. Each feature f^C from a test image C was appearance-matched against a large database of features. The nearest neighbor in this database was selected and tentatively matched to the feature. The database contained both features from a reference image A of the same object (10^2-10^3 features depending on the detector and on the image), as well as a significantly

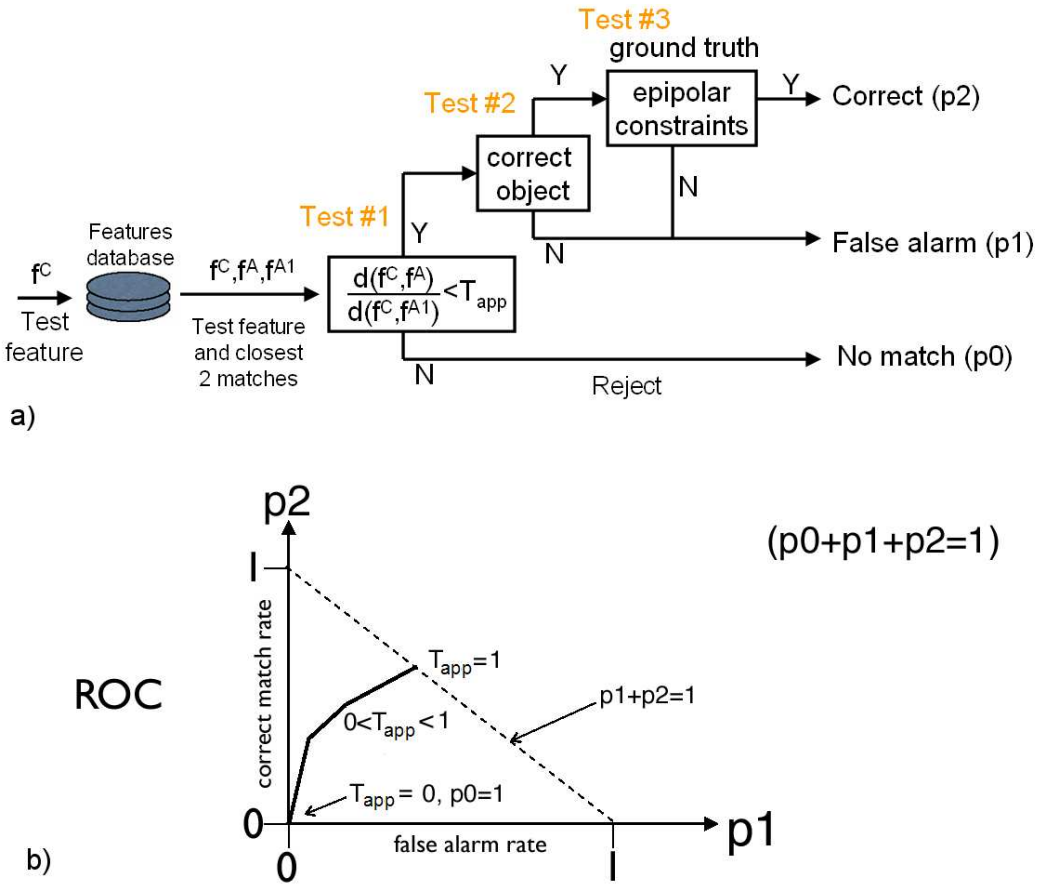


Figure 2.8: (a) Diagram showing the process used to classify feature triplets. (b) Conceptual shape of the ROC trading off false alarm rate with detection rate. The threshold T_{app} on distance ratios (Section 2.6.2) is bounded by $[0, 1]$, and the ROC is bounded by the curve $p1 + p2 = 1$.

larger number (10^5) of features from unrelated images. Using this large database replicates the matching process in object/class recognition applications, where incorrect pairs can arise from matching features to wrong images.

The diagram in figure 2.8-(a) shows the decision strategy. Starting from feature f^C from the test image C, a candidate match to f^C is proposed by selecting the most similar amongst the whole database of features. The search is performed in appearance space. The feature returned by the search is accepted or rejected ($Test\#1$) based on the distance metric ratio that will be described in Section 2.6.2. The candidate match is accepted only if the ratio lies below a user-defined threshold T_{app}.

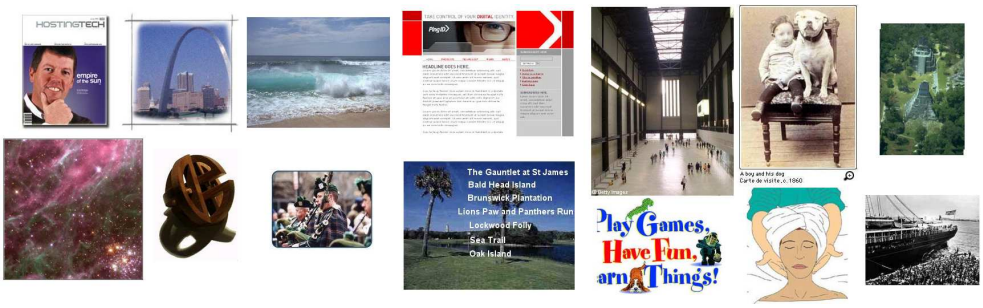


Figure 2.9: A few examples of the 535 irrelevant images that were used to load the feature database. They were obtained from Google by typing 'things.' 10^5 features detected in these images were selected at random and included in our database.

If the candidate match is accepted based on this appearance test, the next stages aim at validating this match. $Test\#2$ checks the identity of the image from which the proposed match is coming. If it comes from the image of an unrelated object, the proposed match cannot correspond to the same physical point. The match is rejected as a false alarm.

$Test\#3$ validates the proposed match based on geometry. The test starts from the proposed match f^A in the reference image, it uses the epipolar constraints described in Section 2.4 and tries to build a triplet (initial feature — auxiliary feature — proposed match) that verifies all epipolar conditions (one constraint in the auxiliary image and two constraints in the test image). As mentioned in Section 2.4, typically only zero or one feature from the test image verifies all epipolar constraints generated by a given feature from the reference image. If this feature from the test image is precisely our test feature f^C, the proposed match is declared validated and is accepted. In the alternative this is a false alarm.

In case no feature was found along the epipolar line in the auxiliary image B, the initial point f^C is discarded and doesn't contribute to any statistics, since our inability to establish a triple match is not caused by a poor performance of the detector on the target image C.

Note that this method doesn't guarantee the absence of false alarms. False alarms can arise if an incorrect auxiliary feature is used during the geometric validation — as we will see, they are very few. However, our method offers the important advantage of being purely geometric. Any system involving appearance vectors as an additional constraint would be dependent on the underlying

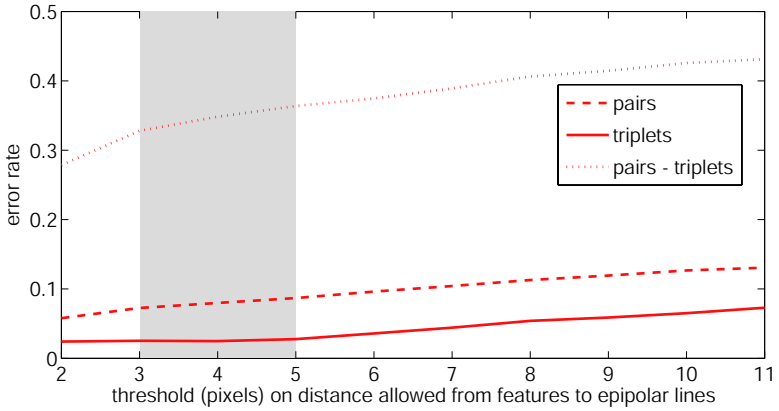


Figure 2.10: Operator-assisted validation of our automated ground truth. A sample of 3000 pairs and triplets was randomly selected from the set of automatically computed tentative feature matches. Two experts classified each pair and triplet by hand as to whether it was correct or not. The fraction of wrong triplets is displayed as a function of the maximum distance allowed to the epipolar line (curve 'triplets'). Our experiments were conducted using adaptive thresholds of 3–5 pixels (gray-shaded zone, see Section 2.5.2), which as the plot shows yields 2% of incorrect triplets. A method based on a single epipolar line constraint ('pairs') would have entailed a rate of wrong correspondences three times higher. In particular, the rate of wrong correspondences is very high for features that could be matched in two images but not in all 3 images ('pairs $-$ triplets').

descriptor and bias our evaluation.

In order to evaluate the fraction of incorrect correspondences established and accepted by our geometric system, 2 experts examined visually the triplets accepted by the system and classified them into correct and incorrect matches. 3000 matches were selected randomly from the accepted triplets and were visually classified; results are reported in figure 2.10. The users also classified matches obtained by a simpler method that would use only two images of the object (reference and test view) and a single epipolar constraint: in this case the geometric validation consists of checking whether or not the test feature lies on the epipolar line generated by the proposed match in the test view. The fraction of incorrect matches is displayed as a function of the threshold on the maximum distance in pixels allowed between features and epipolar lines. We also display the error rate for features that were successfully matched according to the 2-views method, but failed according to the 3-views method. The method using 3 views yields a significantly better performance: when the threshold on acceptable distances to epipolar lines varies between 3 and 5 pixels (see Section 2.5.2), the error rate of the 3-views method is 2%, while the error rate of the

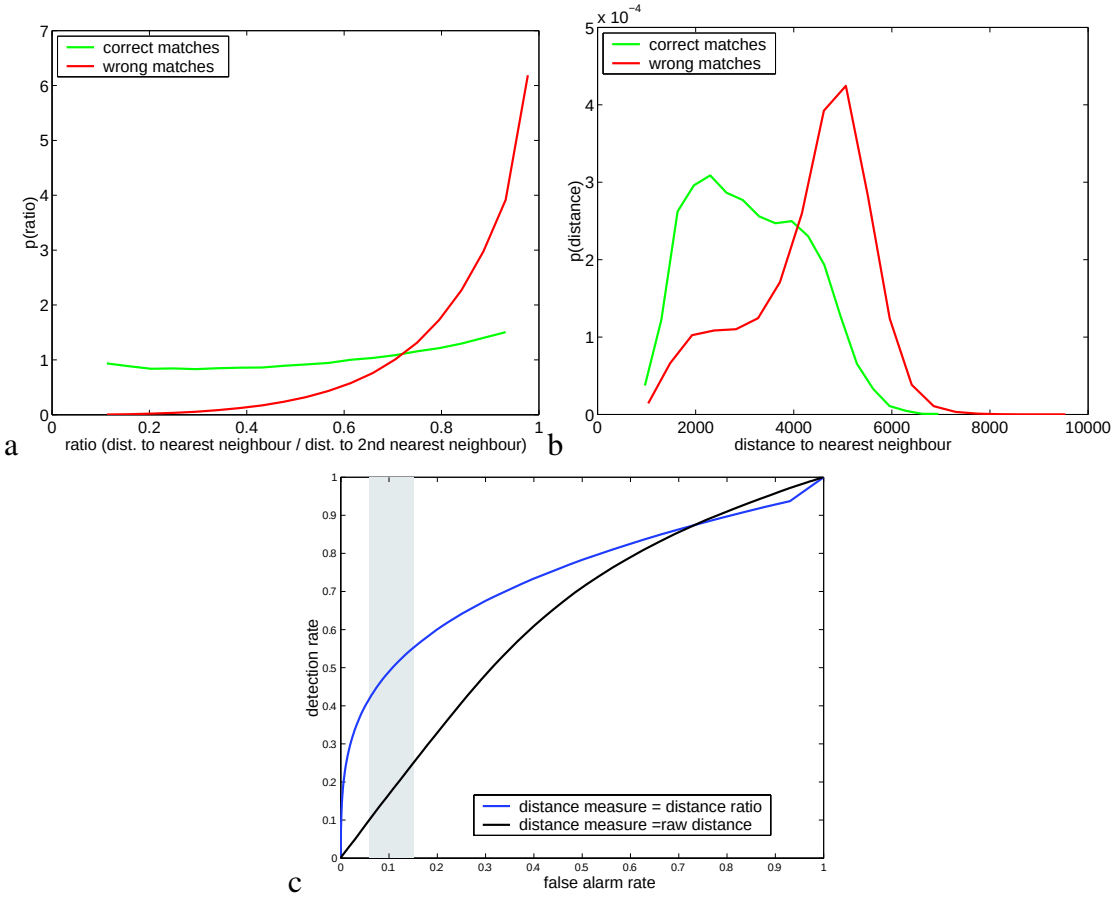


Figure 2.11: a) Sample pdf of the distance ratio between best and second-best match for correct correspondences (green) and false alarms (red). These curves are analogous to the ones in figure 11 of [Low04]. The SIFT descriptor is used here to have results comparable to [Low04]. Lowe's correct-match density is peaked around 0.4 while ours is flat — this may be due to the fact that we use 3D objects, while Lowe uses flat images with added noise. b) Distributions obtained using the distance to best match. c) comparative ROC curves obtained from the distance ratio distributions in panel a and the raw distance distributions in panel b. The threshold varied to generate the ROCs is the threshold T_{app} on quality of the appearance match (see figure 2.8). The distance ratio clearly performs better.

2-views method is three times higher at 6%.

2.6.2 Distance measure in appearance space

In order to decide on acceptance or rejection of a candidate match ($Test\#1$ in figure 2.8), we need a metric on appearance space. Instead of using directly the Euclidean or Mahalanobis distance in appearance as in [MS05, KS04], we use the distance ratio introduced by Lowe [Low04].

The proposed measure compares the distances in appearance of the query point to its best and second-best matches. In figure 2.8 the query feature and its best and second-best matches are denoted by f^C, f^A, and f^{A1}, respectively. The criterion used is the ratio of these two distances, i.e. $\frac{d(f^C, f^A)}{d(f^C, f^{A1})}$. This ratio characterizes how distinctive a given feature is, and avoids ambiguous matches. A low value means that the best match performs significantly better than its best contender, and is thus likely to be a reliable match. A high value of the distance ratio is obtained when the feature points are clustered in a tight group in appearance space. Those features are not distinctive enough relative to each other. In order to avoid a false alarm it is safer to reject the match.

The distance ratio is a convenient measure for our study, since the range of values it can take is always $[0, 1]$ no matter what the choice of descriptor is.

Figure 2.11-(a) shows the resulting distribution of distance ratios conditioning on correct or incorrect matches. The distance ratios statistics were collected during the experiments in Section 2.7. Correct matches and false alarms were identified using the process described in 2.6.1. Figure 2.11-(b) shows the distributions of 'raw distance to nearest neighbor' conditioning on correct or incorrect matches. Since distances depend on the chosen descriptor, the descriptor chosen here was SIFT.

Figure 2.11-(c) motivates further the use of the distance ratio by comparing it to raw distance on a classification task. We computed ROC curves on the classification problem 'correct vs. incorrect match,' based on the conditional distributions from figures 2.11-(a) and 2.11-(b). The parameter being varied to generate the ROC is the threshold T_{app}, which decides if a match is correct or incorrect. Figure 2.11-(c) displays the results. Depending on the combination detector/descriptor, the operating point chosen for the comparisons in Section 2.7 leads to values of T_{app} between 0.56 and 0.70. In the ROC curves from figure 2.11-(c), these values are highlighted by a shaded area. In this operating region, the distance ratios clearly outperform raw distances.

2.6.3 Detection and false alarm rates

As seen in the previous section and figure 2.8, the system can have 3 outcomes. In the first case, the match is rejected based on appearance (probability p_0). In the second case, the match is accepted based on distance in appearance space, but the geometry constraints are not verified and ground truth rules the match as incorrect: this is a false alarm (probability p_1). In the third alternative, the match verifies both appearance and geometric conditions, this is a correct detection (probability p_2). These probabilities verify $p_0 + p_1 + p_2 = 1$. The false alarm rate is further normalized by the number of database features (10^5). This additional normalization was an arbitrary choice, motivated by the dependency of the false alarm rate on the size of the database: the larger the database, the higher the risk of obtaining an incorrect match during the appearance-based indexing described in Section 2.6.1. Detection rate and false alarm rate can be written as

$$false\_alarm\_rate = \frac{\#false\,alarms}{\#attempted\,matches \cdot \#database} \tag{2.2}$$

$$detection\_rate = \frac{\#detections}{\#attempted\,matches}. \tag{2.3}$$

By varying the threshold T_{app} on the quality of the appearance match, we obtain a ROC curve (figure 2.8-(b)). Note that the detection rate does not necessarily reach 1 when T_{app} is lowered to zero since some features will fail Test$\#2$ and Test$\#3$ on object identity and on geometry.

2.6.4 Number of detected features

For the detectors based on extrema of a saliency map, the threshold T_{det} that determines the minimum saliency necessary for a region to be considered as a feature, is an important parameter. If many features are accepted, the distinctiveness of each of them might be reduced, as the appearance descriptor of one feature will be similar to the appearance of a feature located only a few pixels away. This causes false alarms during appearance-based indexing of features in the database. Conversely, if T_{det} is set to a high value and only few highly salient regions are accepted as features, missed detections will occur when a region has been detected in one image but didn't make it to the threshold level in the second image. In order to use each detector/descriptor combination

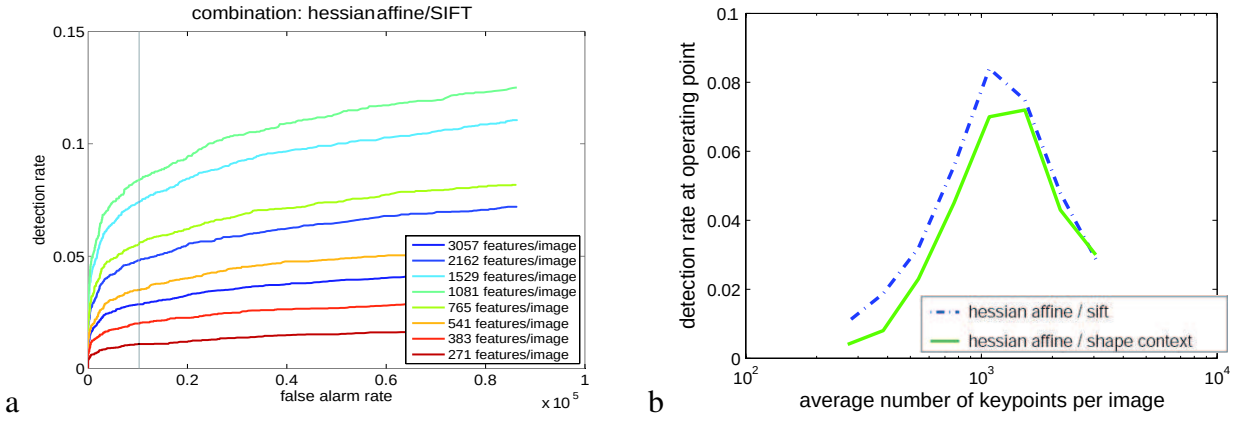


Figure 2.12: a) ROCs obtained when varying the threshold T_{det} on minimum saliency that a region has to satisfy in order to be declared a feature. The legend shows the average number of features detected per image. The chosen operating point is displayed by a vertical line. ROCs are displayed for the Hessian-affine/SIFT combination. b) Another representation of the data from (a) taken at the operating point. Here the horizontal axis is the average number of features per image, which was displayed in the legend in (a). Results are displayed for the two combinations that performed best in Section 2.7.1, while (a) shows results only for one combination.

at its optimal performance level, we performed the matching process described in Section 2.6.1 with a range of values of T_{det}. These values were chosen such that the number of features would vary from $\approx$ 270 features (one feature every 2200 pixels on the objects), up to $\approx$ 3000 features (one feature every 200 pixels on the object), with increments by a factor of $\sqrt{2}$ in the number of features. Similarly to Section 2.7, we choose the operating point at the false alarm rate 10^{-6}. As expected, the detection rate for this operating point first increases, then decreases when the number of features is increased. Figure 2.12-(a) shows the ROC curves obtained for the combination Hessian-affine/SIFT. The operating point is indicated by a vertical line. Figure 2.12-(b) shows at this operating point the detection rate as a function of the number of features detected per image for the two combinations that performed best in Section 2.7: Hessian-affine/SIFT and Hessian-affine/shape context. In the experiments from Section 2.7, the value of T_{det} corresponding to the highest detection rate was chosen for the various detectors/descriptors.

2.7 Results and discussion

2.7.1 Viewpoint change

Figure 2.13 shows the detection results when the viewing angle was varied and lighting/scale was held constant. (a)–(h) display results when varying the feature detector for a given image descriptor. (a)-(d) display the ROC curves obtained by varying the threshold T_{app} in the first step of the matching process (threshold on distinctiveness of the features' appearance). The number of features tested is displayed in the legend. (e)-(h) show the detection rate as a function of the viewing angle for a fixed false alarm rate of 10^{-6} was chosen (one false alarm every 10 attempts — this is displayed by a gray line in the ROC curves from figures 2.13-2.16). This false alarm rate corresponds to different distance ratio thresholds for each detector/descriptor combination. Those thresholds varied between 0.56 and 0.70 (a bit lower than the 0.8 value chosen by Lowe in [Low04]). Figure 2.14(a)–(b) summarize for each descriptor the detector that performed best.

The Hessian-affine and difference-of-Gaussians detectors performed best consistently with all descriptors. While the absolute performance of the various detectors varies when they are coupled with different descriptors, their rankings vary very little. The combination of Hessian-affine with SIFT and shape context obtained the best overall score, with SIFT slightly ahead. In our graphs the false alarm rate was normalized by the size of the database (10^5) so that the maximum false alarm rate was 10^{-5}. The PCA-SIFT descriptor is only combined with difference-of-Gaussians, as was done in [KS04]. PCA-SIFT didn't seem to outperform SIFT as would be expected from [KS04].

Note that the difference-of-Gaussians detector performed consistently almost as well as Hessian-affine. The difference-of-Gaussians is simpler and faster, this motivates its use in fast recognition systems such as [Low04].

In the stability curves, the fraction of stable features doesn't reach 1 when $\theta = 0°$. This is due to several factors: first, triplets can be identified only when the match to the auxiliary image succeeds (see Section 2.4). The $10°$ viewpoint change between reference and auxiliary image prevents a number of features from being identified in both images.

Another reason lies in the tree search. The use of a tree that contains both the correct image and a large number of unrelated images replicates the matching process used in recognition applica-

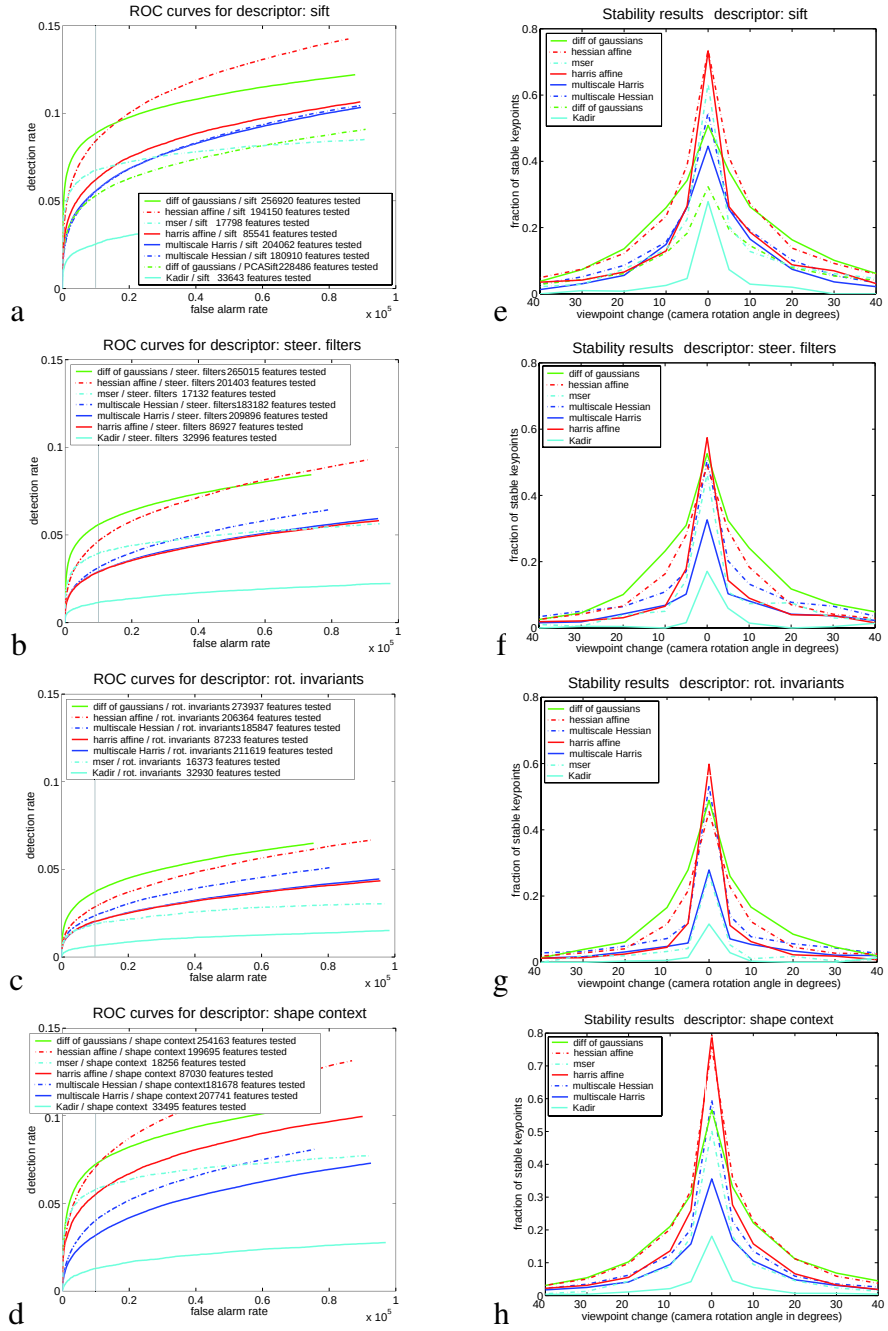


Figure 2.13: Performance for viewpoint change — each panel (a)-(d) shows the ROC curves for a given descriptor when varying the detector. (e)-(h) show the corresponding stability rates as a function of the rotation angle. The 0° value is computed by matching features extracted from different images taken from the same location. The operating point chosen for the stability curves on the right hand side is highlighted by a vertical line in the ROCs.

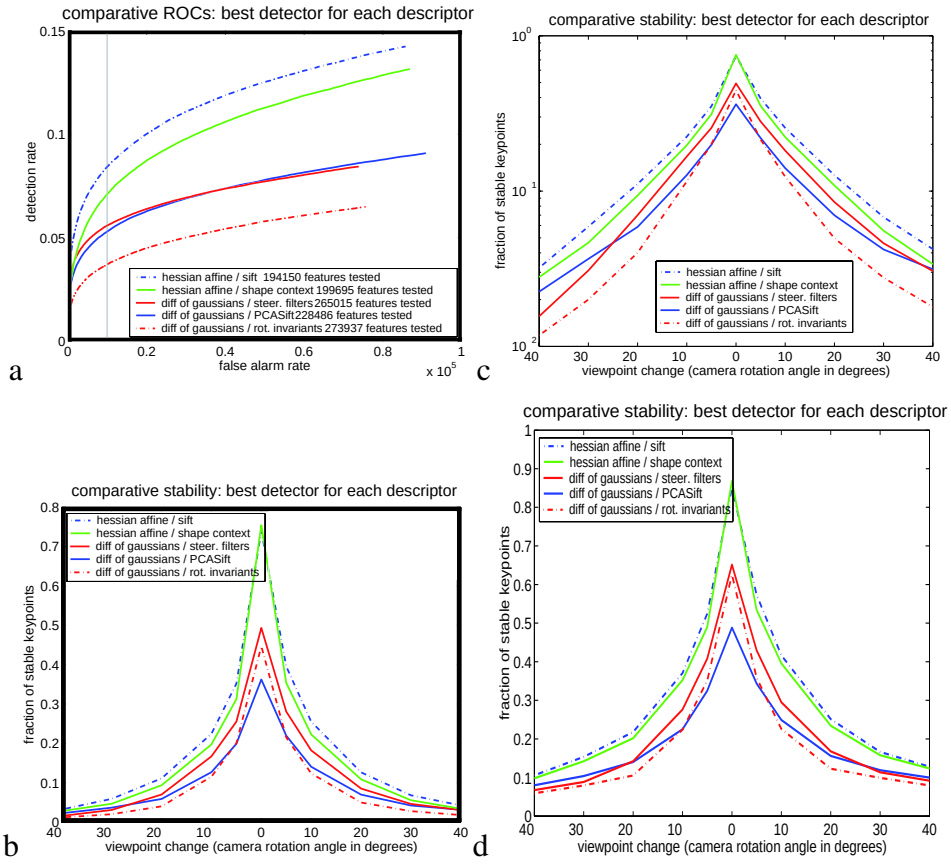


Figure 2.14: Summary of performance for viewpoint change - (a)-(b) show the combination of each descriptor with the detector that performed best for that descriptor. (c) displays the stability results on a semi-log scale. (d) is similar to (b), but the database used for the search tree contained only the features extracted from the correct image (easier task which mimicks wide-baseline stereo).

tions. However, since some features have low distinctiveness, the correct image doesn't collect all the matches. In order to evaluate the detection drop due to the search tree, the experiment was run again with a search tree that contained only the features from the correct image. Figure 2.14-(d) shows the stability results, the performance is 10–15% higher.

A third reason is the noise present in the camera. On repeated images taken from the same viewpoint, this noise causes 5–10% of the features to be unstable.

Another observation concerns the dramatic drop in number of matched features with viewpoint change. For a viewpoint change of 30° the detection rate was below 5%.

Figure 2.15 investigates the effect of different levels of invariance applied to the Harris and

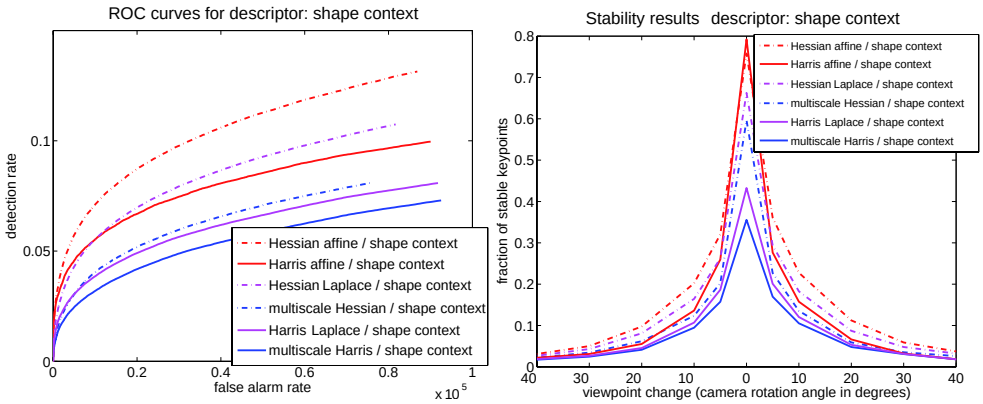


Figure 2.15: Results with several levels of invariance applied to the Harris and Hessian detectors. The shape context descriptor was used.

Hessian detectors. The Harris-Laplace detector was used in [MS02]. In this case the interest points are found at several scales with the Harris detector, then the characteristic scale is selected as the local maximum over scale of the Laplacian function. The same process is applied to the Hessian detector. As mentioned in Section 2.5.3.1, the affine-invariant rectification warps the local area around an interest point so that its second-order moment matrix becomes isotropic. Figure 2.15 shows that the performance improves when using the Laplacian scale selection and when using the affine rectification. The shape context descriptor was chosen for this experiment as this is the descriptor that leads to the largest performance increase when adding affine invariance, as seen in figure 2.13.

Figure 2.16 shows the results ('summary' panel only) when the Euclidean distance on appearance descriptors is replaced by the Mahalanobis distance. The covariance matrix used to normalize the Mahalanobis distance was generated from distances between descriptors generated by a large number $(300,000)$ of background features. Although the Mahalanobis distance is more general than the Euclidean distance, most relative performances were not modified. This might be related to our use of the distance ratio (cf., Section 2.6.2) instead of raw distances. Hessian-affine, again, performed best, while shape context and SIFT were the best descriptors. In this case, shape context outperformed SIFT.

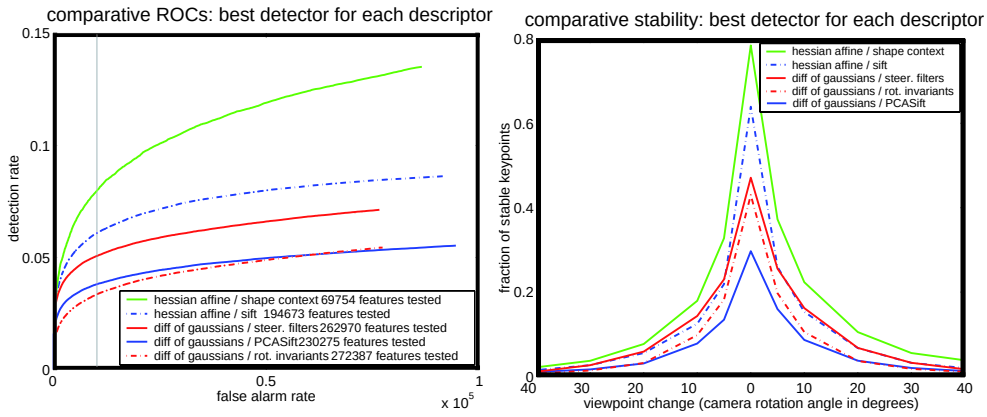


Figure 2.16: Results for viewpoint change, using the Mahalanobis distance instead of the Euclidean distance on appearance vectors.

2.7.2 Normalization

As mentioned above, the matching performance between images A and C is affected by the inability to find a match in the auxiliary image B. One could want to normalize out this loss in order to get 'pure' stability results between A and C.

Let's denote by $p(\theta)$ and $p(\theta_1, \theta_2)$ the probabilities that, given a reference feature, a match will respectively exist in one view of the same scene taken from a viewpoint θ degrees apart (for pairs), and in two views taken from viewpoints θ_1 and θ_2 apart from the reference image (for triplets). If we assume independence between the matching processes from A to B and from A to C, we can decompose $p(\theta_{AB}, \theta_{AC})$ into $p(\theta_{AB}, \theta_{AC}) = p(\theta_{AB}) p_{f^A}(\theta_{AC})$ and normalize by $p(\theta_{AB}) = p(10^\circ)$ to obtain absolute performance figures between A and C.

Unfortunately, it seems that the matching processes from A to B and from A to C cannot be considered to be independent. First, figure 2.10 shows a different behavior between features that were successfully matched between A, B, and C, and the features that were matched between A and C, but for which the match A–B failed. In the latter case, the fraction of incorrect matches is much higher. Another hint comes from the stability results from figure 2.14-(c). Note that all combinations detectors/descriptors show a comparable performance of 6–10% when the rotation is 40°. If we were to normalize by $p(10^\circ)$, the combination that performs worst at 0° (i.e., difference-of-Gaussians/PCA-SIFT) would by far perform best at 40°. It seems very unlikely that a combination

that performs poorly in easy conditions, would outperform all others when matching becomes more difficult. Therefore we believe that matches between A and B and between A and C are not independent. In order to avoid any inconsistency, we did not normalize the stability results. Our system is only collecting the most stable features, those that were not only stable between A and C, but were successfully matched into triplets.

2.7.3 Flat vs. 3D scenes

As mentioned in Section 2.2, one important motivation for the present study is the difference in terms of stability between texture-generated features extracted from images of flat scenes, and geometry-generated features from 3D scenes. In order to illustrate this stability difference, we performed the same study as in Section 2.7.1, on one hand with 2 images of piecewise flat objects (box of cookies, can of motor oil), on the other hand on two objects with a more irregular surface (toy car, dog). Results are displayed in figure 2.17. As expected, the stability is significantly higher for features extracted from the flat scenes. Note that the stability curves are not as symmetrical with respect to the 0° value as the curves in figures 2.13–2.14. This is due to the fact that here the results are only averaged over a small number of objects.

One interesting result was that the relative performance of the various combinations detector/descriptor was modified between flat and 3D objects. (e)–(f) display stability results respectively for rotations of 10° and 40°. The fractions of stable features from flat scenes are displayed on the x axis, for 3D scenes they are on the y axis. All combinations lie below the diagonal $x = y$ since stability is lower for 3D scenes. Some changes in relative performances are highlighted. For example, for flat scenes MSER/SIFT and MSER/shape context performed best, while their performance was only average for 3D scenes. Conversely, difference-of-Gaussians/SIFT, difference-of-Gaussians/shape context, and Hessian-affine/shape context, which were the best combinations for 3D scenes, were outperformed on 2D objects.

The dependency of the features stability on the object identity is investigated in figure 2.18. A viewpoint change of 10° was chosen. For each object, the highest fraction of stable features across all investigated detector/descriptor combinations was recorded. Two users separated the objects

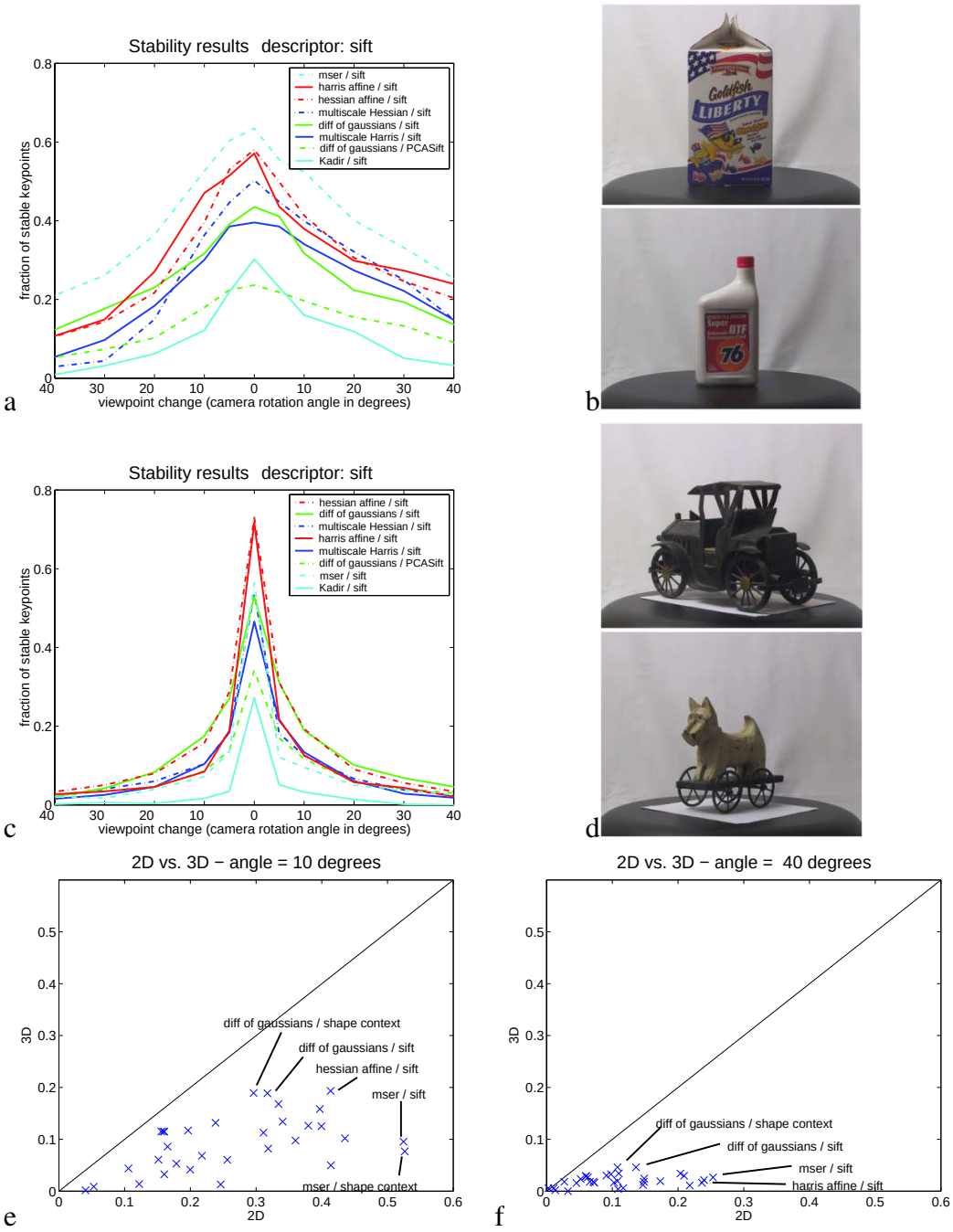


Figure 2.17: Flat vs. 3D objects — (a) shows the stability curves obtained for SIFT for the two piecewise flat objects in (b). Similarly, (c) shows the SIFT stability curves for the two 3D objects in (d). 'Flat' features are significantly more robust to viewpoint change. (e)-(f) show the fractions of stable features for the same piecewise 2D objects versus the same 3D objects, for all combinations of detectors / descriptors in this study. Scatter plots are displayed for rotations of $10°$ and $40°$. A few combinations whose relative performance changes significantly are highlighted.

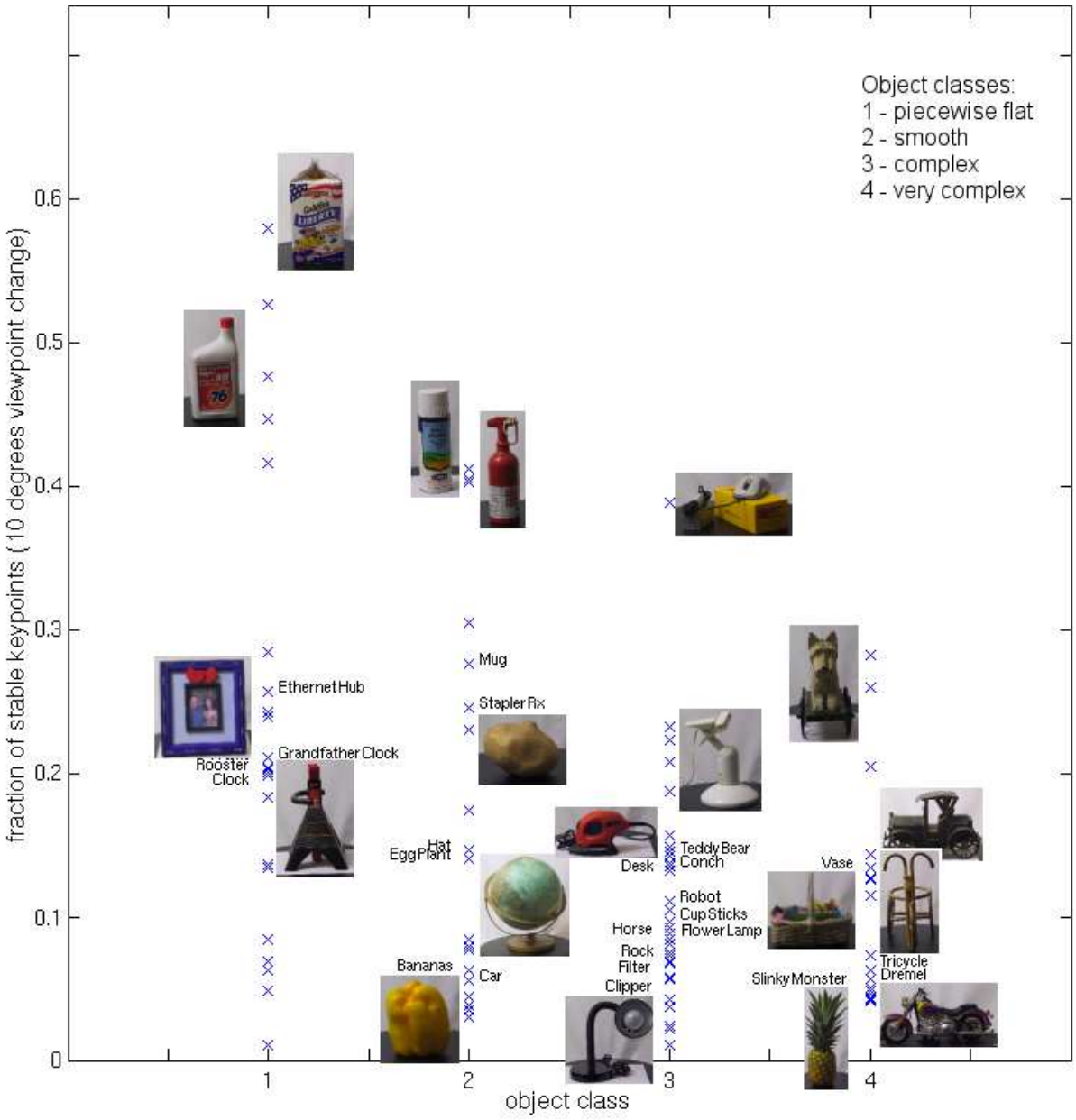


Figure 2.18: Fraction of stable features for each object of the database, under a fixed viewpoint change of 10°. For each object, the figure shows the highest fraction of stable features across all investigated combinations of detectors/descriptors. The objects were user-separated into: 1. piecewise flat 2. smooth 3. complex 4. very complex. Names of objects and thumbnails representing them are displayed only for the 42 objects showed in figure 2.5 (objects displayed by thumbnails were randomly selected). The performance on the other objects is displayed by a cross only.

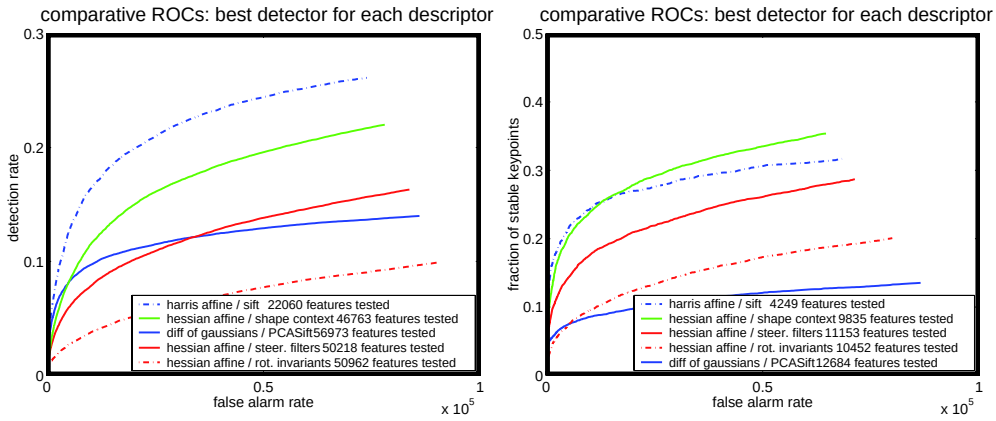


Figure 2.19: (Left panel) ROCs for variations in lighting conditions. Results are averaged over 3 lighting conditions. (Right panel) ROCs for variations in scale.

into 4 classes: 'piecewise flat,' 'smooth,' 'complex,' and 'very complex.' Figure 2.18 displays the fraction of stable features for each object. We observe that the average stability for piecewise flat objects is higher than for the other classes, which confirms the results from figure 2.17. The most stable objects in all classes are those which bear printed text. Characters have well contrasted boundaries which are suitable for generating features, besides text is usually printed on a flat or smooth surface which ensures a high feature stability. Objects with smooth surfaces do not perform so well, as most features generated by these objects are close to the boundary and will be sensitive to self-occlusion. Objects with a specular surface (globe, lamp, motorcycle) also perform poorly, as the reflection will not follow the change of viewpoint or object movement.

2.7.4 Lighting and scale change

Figure 2.19(left) shows the results obtained when changing lighting conditions and keeping the viewpoint unchanged. This task is easier: since the position of the features shouldn't change, we don't need to introduce the auxiliary image B. As a result, the detection rates reported in the ROC curves are significantly higher than in the study of viewpoint changes. Only the 'summary' panels with the best detector for each descriptor are displayed. This time, the combination which achieved best performance was Harris-affine combined with SIFT.

Figure 2.19(right) displays the results for a change of scale. The scale change was performed

by switching the camera's focal length from 14.6 mm to 7.0 mm. Again, the figure displays only the 'summary' panel. Hessian-affine combined with shape context and Harris-affine combined with SIFT obtained the best results.

2.8 Discussion and conclusions

We compared the most popular feature detectors and descriptors on a benchmark designed to assess their performance in recognition of 3D objects. In a nutshell: we find that the best overall choice is using an affine-rectified detector [MS02] followed by a SIFT [Low04] or shape-context descriptor [BMP02]. These detectors and descriptor were the best when tested for robustness to change in viewpoint, change in lighting, and change in scale. Amongst detectors, runner-ups are the Hessian-affine detector [MS02], which performed well for viewpoint change and scale change, and the Harris-affine detector [MS02], which performed well for lighting change and scale change. However, the performance of the difference-of-Gaussians detector is close to the affine-rectified detectors, while its implementation is simpler and its computation time shorter; therefore we believe it is a good compromise between performance and speed.

Our benchmark differs from previous work from Mikolajczyk and Schmid in that we use a large and heterogeneous collection of 100 3D objects, rather than a set of flat scenes. We also use Lowe's ratio criterion, rather than absolute distance, in order to establish correspondence in appearance space. This is a more realistic approximation of object recognition. A major difference with their findings is a significantly lower stability of 3D features. Only a small fraction of all features (less than 3%) can be matched for viewpoint changes beyond 30°. The situation is a bit better when the goal is stereo-vision or mosaicking (figure 2.14-(c)), where features are matched across a small number of images. Our results on descriptors favor SIFT and shape context descriptors, and are in agreement with [MS05]. However, regarding detectors, not all affine-invariant methods are equivalent as suggested in [ea05c] — e.g., MSER performs poorly on 3D objects, while it is very stable on flat surfaces.

We find significant differences in performance with respect to a previous study on 3D scenes [FB04]. One possible reason for these differences is the particular statistics of their scenes, which

appear to contain a high proportion of highly textured quasi-flat surfaces (boxes, desktops, building facades, see figure 6 in [FB04]). This hypothesis is supported by the fact that our measurements on piecewise flat objects (figure 2.17) are more consistent with their findings. Another difference with their study is that we establish ground truth correspondence purely geometrically, while they use appearance matching as well, which may bias the evaluation.

An additional contribution of this paper is a new method for establishing geometrical features matches in different views of 3D objects. Using epipolar constraints, we are able to extract with high reliability (2% wrong matches) ground truth matches from 3D images. This allowed us to step up detector-descriptor evaluations from 2D scenes to 3D objects. Comparing to other 3D benchmarks, the ability to rely on an automatic method, rather than painfully acquired manual ground truth, allowed us to work with a large number of heterogeneous 3D objects. Our setup is inexpensive and easy to reproduce for collecting statistics on correct matches between 3D images. In particular, those statistics will be helpful for tuning recognition algorithms such as [Low04, CJ04, MP04, MP07a]. Our database of 100 objects viewed from 72 positions with three lighting conditions will be available on our web site.

Chapter 3

Coarse-to-fi ne recognition of individual objects

3.1 Abstract

A coarse-to-fine probabilistic model for detecting objects in images is presented. Objects are composed of constellations of features, and features from a same object share the common reference frame of the image in which they are detected. Features' appearance and pose are modeled by probabilistic distributions, the parameters of which are shared across features in order to allow training from few examples.

In order to avoid an expensive combinatorial search, we propose a coarse-to-fine strategy, inspired by the work of Lowe [Low99, Low04] and of Geman and collaborators [AGF04, FG01]. Well-established, simple, and inexpensive steps are used as successive refinement blocks that discard incorrect matching hypotheses in a cascade. The candidate hypotheses output by our algorithm are evaluated by a generative probabilistic model that takes into account each stage of the matching process.

We apply our ideas to the problem of individual object recognition and test our method on several data-sets. We compare with Lowe's algorithm and demonstrate significantly better performance.

3.2 Introduction

Recognizing objects in images is perhaps the most challenging problem currently facing machine vision researchers. Much progress has been made in the recent past both in recognizing individual objects [FTG04, Low04], as well as in recognizing object categories [BBM05, FPZ03, LSP06, OFPA04, WWP00]. Other groups have interpreted the problem of individual object recognition and detection as a wide-baseline problem; their work aims at registering pairs of images with each other [FTVG05, ea02b, PZ98, Rot04]. A number of ideas have proven to be key to recent progress. First of all: objects and categories may be represented as collections of *features*, each one of which encodes the distinctive appearance of a portion of the object. The mutual position of these features, as well as their appearance, contains information as to the identity of the object. Second: a (small) subset of the object's features is often sufficient signal to infer its presence

in the image — if one enforces both mutual position and appearance constraints one may rule out false alarms arising from features detected in random clutter. This provides robustness to occlusion and poor feature detection — even if many model features are missing in the test image, recognition can proceed with the remaining features. Third: very efficient approximate algorithms for matching features in the high dimensional space where the features' appearance is represented have been discovered [BL97]. Similarly, there exist efficient algorithms for enforcing geometrical constraints [FB81, FG01, Low85]. The combination of these ideas and techniques has given us very efficient and robust object recognition algorithms.

However, much progress still needs to be made to reach levels of performance comparable to those of the human visual system. Error rates are still large on fairly benign benchmark image sets [KSP07, LSP06, MP04, WZFF06]. Furthermore, many classes of objects are still difficult to recognize or discriminate: e.g., objects containing repeated textures (furry teddy bears) and objects with smooth featureless surfaces (plain coffee mugs). In order to overtake some of the current challenges we need to be creative and generate novel image analysis ideas, e.g., new feature types. Another way to make progress is to place our recent discoveries on a firm theoretical footing. Much of what we know is still a 'bag of tricks' — we need to understand better the underlying principles in order to improve our designs and take full advantage of what we can learn from the statistics of images.

In this study we focus on recognition of individual objects in complex images. Our goal is to produce a consistent probabilistic interpretation of the recognition system from Lowe [Low99, Low04], one of the most effective techniques we know for individual object recognition. The techniques we use are inspired by the work of Burl [BMW98], Weber [WWP00], and Fergus [FPZ03] on the probabilistic 'constellation' model for object categories. We are also inspired by Amit [AGF04] and Fleuret [FG01] and their work on coarse-to-fine searching.

We started an exploration of this topic in [MMP04, MP04] and build upon this work. The current study makes three novel contributions. First: we incorporate in our recognition algorithm a number of well-established, deterministic modules or 'atomic operations,' arranged in a cascade in order to pursue the search for the best interpretation of a given image in a coarse-to-fine fashion. We start with 'statistical' global measurements and eliminate a great number of interpretations of

the test image with very little effort, before analyzing the remaining regions of hypothesis space in greater detail.

Second novel contribution: we introduce a generative probabilistic model that evaluates the hypotheses taking into account each stage of their formation. Regarding parameters estimation, we benefit from a recent study measuring the variation in position and appearance of features in 3D objects imaged under different viewpoints and lighting conditions – various conditional probabilities in our algorithm are therefore based on careful empirical measurements [MP07b].

Third: although this was not the main goal of our study, our experiments show that our new algorithm and probabilistic scoring model, perform substantially better than a state-of-the-art detection system developed independently by Lowe [Low99, Low04].

Section 3.3 introduces our generative model. Section 3.4 describes the coarse-to-fine process used to generate hypotheses and sets of features assignments. Sections 3.5 and 3.6 explain in detail the probabilistic model and parameters estimation used to assign a score to the hypotheses. Section 3.7 presents and discusses results, and Section 3.8 contains our conclusions.

3.3 Generative model

3.3.1 Object recognition scenario

Our target scenario consists of recognizing individual objects in complex images [Low99, Low04]. We assume that a number of *known objects* have been gathered. We collect images of these objects, the images collected for each object form the *model* of this object. The set of models form our *database*. In the experiments of Section 3.7 we consider one or few training images per known object. On the other hand, we are given a query image, which is the photograph of a complex *test composition* containing some of the known objects — we call this image the *test image*. In addition to some of the known objects, the test image might contain unknown objects (i.e. objects not included in our training set), as well as background clutter. Our goal is to identify the known objects present in the test composition, along with their pose — i.e., position, orientation, and scale.

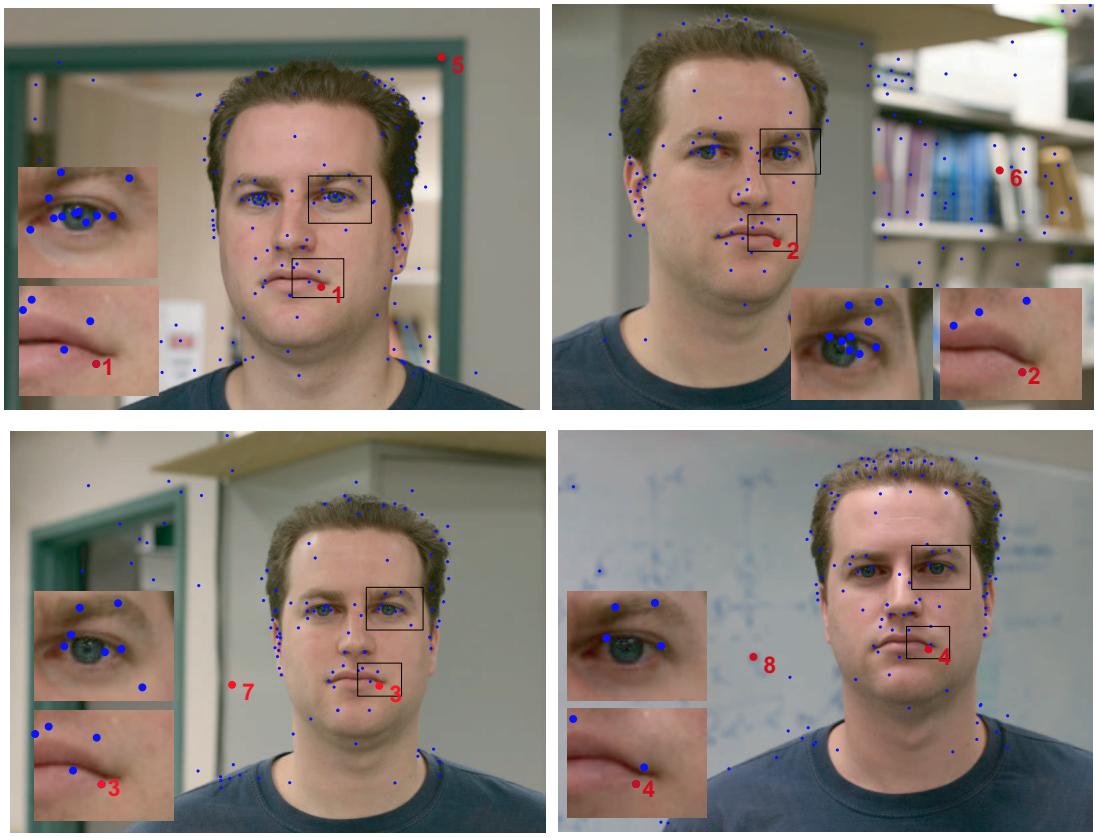
3.3.2 Modeling object images as collections of features

The physical objects photographed in the database and in the test image are represented as a spatially deformable collection of *parts* [BMW98, FPZ03, FB81, ea93, WWP00, WFKvdM97]. In images of the objects, these parts are associated to visually distinctive *features*. In probabilistic terms, one may model this as the object parts being the cause of visual features, whose location and appearance is a random variable centered on a nominal location and appearance. Figure 3.1-(a) displays the features identified by the commonly used difference-of-Gaussians detector [CP84, Lin94, Low04] on several images of the same face. Some features and groups of features are geometrically consistent with each other across different views of the face, since they are detected repeatedly near the same physical location on the face. On the other hand, since the background is different from one image to the next, the locations of the background features detections are unrelated between any two images. Features-based object recognition methods based only on such considerations on pose, have been developed successfully, e.g., by Lowe [Low85] and more recently by Fleuret and Geman [FG01]. In both cases the authors use perceptual grouping of simple features to form objects.

In general, building a recognition system based only on such considerations on pose, is a hard task: one challenge, when we want to put into correspondence features from the database and from the test image, is due to the different number of features generated by the same object in separate images containing it. These different numbers of features are due to differences in lighting conditions and in viewpoint, imaging noise, resolution, or occlusions. For example, an object imaged successively from far and from a close viewpoint, shows more detail in the second picture, thus generating more features.

While pose information is a very useful tool for recognition, additional information is provided by image descriptors which represent the local image texture or *appearance* near these points of interest. Figure 3.1-(b) shows the pairwise distances between the SIFT descriptors [Low04] associated to a few features from the four views of the face in figure 3.1-(a). In general, pairs of features corresponding to the same physical object part often look similar, while features corresponding to different parts almost always look different. Some recognition applications [DS04, GD05] have

a)



b)

		Foreground				Background			
		1	2	3	4	5	6	7	8
Frgrnd	1	0	0.1	0.1	0.1	0.7	0.5	0.5	0.8
	2	0.1	0	0.1	0.2	0.7	0.6	0.5	0.9
	3	0.1	0.1	0	0.0	0.5	0.4	0.5	0.8
	4	0.1	0.2	0.0	0	0.5	0.4	0.5	0.9
Bkgnd	5	0.7	0.7	0.5	0.5	0	0.5	0.8	1.0
	6	0.5	0.6	0.4	0.4	0.5	0	0.4	0.9
	7	0.5	0.5	0.5	0.5	0.8	0.4	0	1.0
	8	0.8	0.9	0.8	0.9	1.0	0.9	1.0	0

Figure 3.1: Foreground features vs. background features. (a) A significant fraction of the features generated by the face are found again at the same face location in separate views — see enlarged image areas, the corner of the eye and the corner of the mouth always generate features. On the other hand, since the background changes, background features are generated at different locations in each picture. Some 'unreliable' features are also generated by the face. We consider these to be 'clutter' as well. (b) Pairwise Euclidean distances in appearance space between a few features. Here a feature's appearance is encoded by the SIFT descriptor [Low99, Low04]. The indices attributed to the features are indicated on the images. Correct correspondences are displayed in green, incorrect correspondences in black. Note that correct correspondences are associated to smaller distances.

had good success with a characterization of features that considers appearance only and discards pose information ('bag of features' approaches).

In this paper, we will consider both pose and appearance. A feature will be called *stable* when it is both found repeatedly near the same physical object part in separate images, and generates descriptors that are very close in appearance space [MP07b]. The features' pose (location, scale, orientation), along with their appearance description will be all the knowledge that we learn from training images of known objects. In the test image, features' pose and appearance will be our evidence regarding presence or absence of some of the known objects in the test image.

In a model, the features are caused by parts of the known object or by random background events. In the test image, the features are either caused by a known object present in the test composition, or they originate from objects not present in the training set and from random clutter. The former type of features will be called *foreground features*, the latter *background features*. Clutter features may appear everywhere in the test image, in particular on the footprint of the known objects, these features cannot be matched to the database models. In order to recognize objects in the test image, we need to establish correspondences between test image and database, i.e., identify subsets of the test features as well as subsets of the database features that are both consistent in terms of geometrical structure, and match each other in terms of appearance. Large subsets of corresponding features, and good consistency in pose and appearance, are indicators of the presence of the object in the test image. We will discuss how this may be interpreted in probabilistic terms in Sections 3.5 and 3.6. During the matching process, a fraction p_{det} of the database features associated to the objects present in the test image can be associated successfully to corresponding features in the database. Similarly, a fraction p_{stray} of the test image's background features are extremely similar in appearance to database images, and form spurious matches that we need to reject.

3.3.3 Probabilistic interpretation of the test image

The previous section depicts the generation of the database models and of the test image starting from the known objects. The solution of our recognition problem may be found by searching

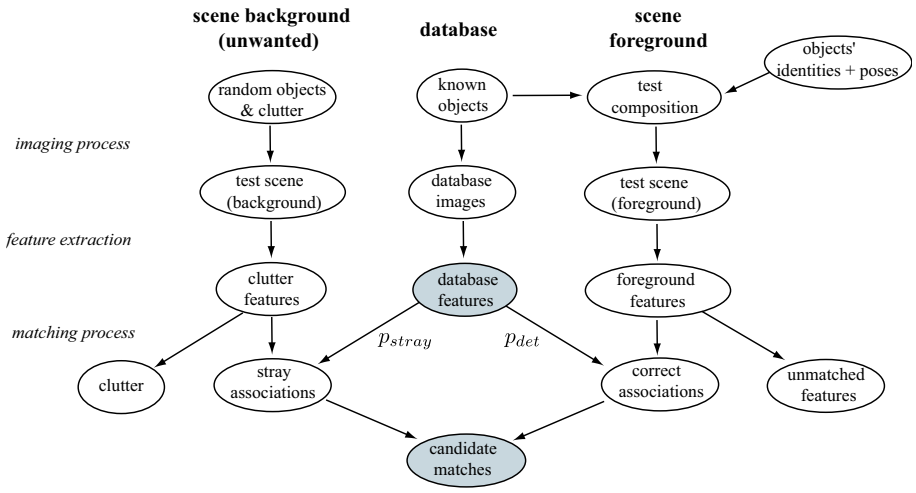


Figure 3.2: Generative model for database and test image. The grey-shaded nodes indicate variables from the recognition system that are directly observable.

for the most likely *interpretation* of the test image. With the terms used in the next sections, an interpretation is the combination of a *hypothesis* (identity and pose of the known objects present in the test image) and an *assignment vector* (set of correspondences between test features and database features). Using the maximum likelihood framework, we look for

$$\underset{interpretation}{argmax}\ P(test\_image|interpretation, database). \tag{3.1}$$

One expressive model that interprets combinations of features in probabilistic terms consists of having a joint distribution on pose and appearance as in the 'constellation model' developed by [BMW98, FPZ03, WWP00]. Unfortunately, the number of parameters that have to be estimated is very large and requires many training examples. Here we wish to train on one or very few images. To reduce the number of parameters, we assume that if we condition on an object pose in the test image, the test features attributed to this object are independent of each other. This assumption will be discussed in Section 3.6.5.

Our model is summarized in the plate diagram shown in figure 3.2. The known objects generate features that may be observed both in the models and in the test image. Similarly, the known objects that are present in the test image generate foreground features. Additionally, clutter and unknown objects that were not included in the training set lead to unwanted features. The features generated by unknown objects are locally indistinguishable from clutter features caused by random texture

in the test image.

3.4 Hypothesis generation

A *hypothesis* contains the identity of the known objects that are believed to be present in the test image, along with their pose. In this section we describe the algorithms that are used to generate likely hypotheses, in Sections 3.5–3.6 we will present the probabilistic scoring method used to evaluate hypotheses.

3.4.1 Test image and models

As mentioned in Section 3.3.2, all test compositions and known objects, are represented by collections of distinctive features. In this work we use the popular combination of multi-scale difference-of-Gaussians detector and SIFT descriptor proposed by Lowe [Low04], although a few other options are equally good [ea05c, MS05, MP07b]. We call *database of features* and denote by M the set of features extracted in images of known objects, and denote by F the set of features extracted from the test image.

Known objects, in number $\mathcal{M}$, are indexed by k and denoted by m_k (k will appear both as a subscript and a superscript, but will always denote a known object). The indices i and j are used respectively for test features and database features: f_i denotes the i-th test feature, while f_j^k denotes the j-th feature from the k-th object. The number of features detected in images of object m_k is denoted by n_k. For the $\mathcal{M}$ known objects, these cardinalities form the vector $\mathbf{n} = \{n_k\}_k = (n_1 ... n_{\mathcal{M}})$. Therefore, M is a set of sets of features: $M = \left\{\{f_j^k\}_{j=1...n_k}\right\}_{k=1...\mathcal{M}}$. Note: throughout this paper, bold notation will denote vectors.

A special object, denoted by m_0, represents the background clutter which generates unwanted detections. The number of features in the 'background object' is not known in advance.

Each feature is described by its *pose information* and its *appearance*: $f_i = (\mathcal{X}_i, \mathcal{A}_i)$ for a test feature, $f_j^k = (\mathcal{X}_j^k, \mathcal{A}_j^k)$ for an object feature (see figure 3.3). The pose information is composed of position information x, y, scale s, and orientation θ in the image. We have $\mathcal{X}_i = (x_i, y_i, s_i, \theta_i)$ for test features, and similarly $\mathcal{X}_j^k = (x_j^k, y_j^k, s_j^k, \theta_j^k)$ for database features. This pose informa-

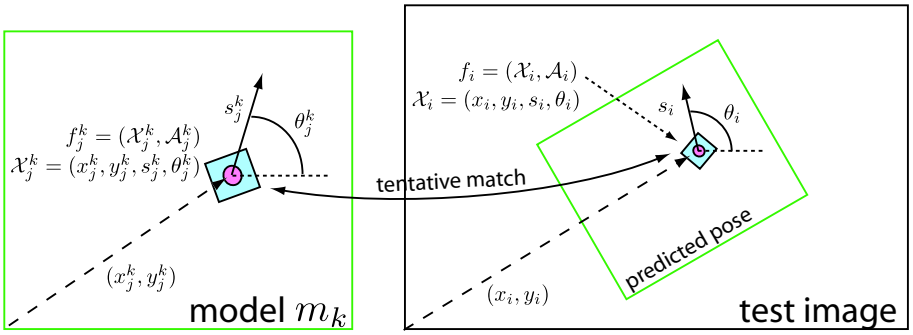


Figure 3.3: This figure illustrates the notation related to features. The pink points represent the location where the feature was detected, the blue squares represent the image area that is sampled to compute the appearance descriptors. Model feature f_j^k matches test image feature f_i when its appearance $\mathcal{A}_j^k$ is closest to $\mathcal{A}_i$ in the database. The transformation between the two features' poses $\mathcal{X}_j^k$ and $\mathcal{X}_i$ predicts the model's pose in the test image. This pose prediction is the building block of the Hough transform described in Section 3.4.2.2.

tion is measured relative to the standard *reference frame* of the image in which the feature has been detected. All features extracted from the same image share the same reference frame. The $appearance$ information associated to a feature is a descriptor characterizing the local image appearance, or texture, in a region of the image centered at this location. It is denoted by $\mathcal{A}_i$ for test feature f_i and $\mathcal{A}_j^k$ for database feature f_j^k.

A *hypothesis* H is an interpretation of a test image, i.e., a subset of the known objects $\mathbf{m} = \{m_k\}_k$ together with their poses $\mathbf{\Theta} = \{\Theta_k\}_k$. Note that repetitions are allowed since two similar objects can be present in the test image (e.g., two cans of Pepsi), therefore this is a *multiset*. In order to keep a simple terminology, we will nevertheless call this a set. The objects specified by the hypothesis are believed to be present in the test composition, thus 'causing' the test. A $pose$ is a set of parameters that map a known object onto a corresponding object in the test composition. In this paper we consider affine transformations. Thus, an object's pose is the affine transformation that maps a database model onto a corresponding object in the test image (see figure 3.3). The number of objects specified present by the hypothesis is denoted by $\mathcal{H}$. A hypothesis can be a *no-object* ($\mathcal{H} = 0$), *single-object* ($\mathcal{H} = 1$), or *multi-objects* hypothesis ($\mathcal{H} > 1$). We will consider first single-object hypotheses; in Section 3.4.4 we will investigate how to combine them into multiple-objects hypotheses. We denote by H_0 the special hypothesis that considers no object to be present (all features are clutter detections).

We assume that in each test image there is an *underlying ground truth* as to which objects are

present with which pose, i.e., a true hypothesis H_{true}.

An *assignment vector* $\mathbf{V}$ carries complementary information to a hypothesis: it assigns each feature from the test image to a database feature (in which case we call it a *foreground feature*) or to clutter (*background feature*). The $i\text{-}th$ component $V(i) = (k, j)$ denotes that the test feature f_i is matched to f_j^k, j-th feature from the k-th object m_k. $V(i) = 0$ denotes the case when f_i is attributed to clutter.

We also assume that in each test image there is an underlying ground truth regarding the assignment vector. Features detections in images are triggered by informative parts on the physical objects, both in models and in the test image. The correspondences between parts on the physical objects, and features detected respectively in models and in the test image, are a hidden variable. These correspondences induce a ground truth set of correspondences $\mathbf{V}_{true}$ between features in models and features in the test image.

3.4.2 Coarse-to-fine hypotheses filtering

Given a test image, our goal is to come up quickly with a likely explanation H. Before we see the test image, all hypotheses are possible explanations. Since there are very many hypotheses to be considered, our strategy will be to exclude as many as possible from consideration at an early stage. We use a coarse-to-fine strategy inspired by Amit [AGF04] and Fleuret [FG01]. In these lines of work, a same object detector is run several times from a coarse resolution to a fine resolution. At coarse resolution, the detector can exclude quickly large, irrelevant regions of the hypothesis space without exploring them in detail. The disadvantage of using a coarse detector is that it is not able to output a precise hypothesis, but can only tell which large regions of the hypothesis space might contain an object. Next, the regions that triggered this coarse detector are explored again at a finer resolution. This second detection stage rejects more regions of hypothesis space, and focuses more precisely on the relevant locations. The process is repeated several times until the target resolution is reached.

Our recognition algorithm follows the same idea of discarding large regions of the hypothesis space at an early stage of the recognition process, without exploring these regions in detail. How-

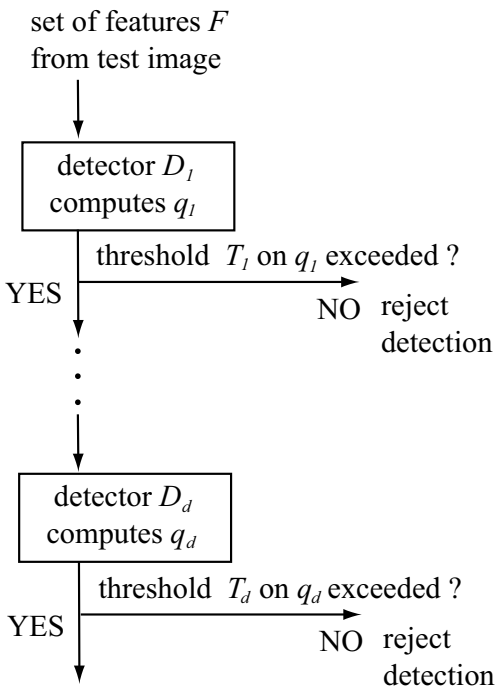


Figure 3.4: General sequence of steps for coarse-to-fine object recognition. Each step focuses more precisely on the regions of hypothesis space that were retained by the previous step. Each detector makes its decision based on a threshold on a quantity that it computes. The thresholds are set based on probabilistic measurements (see Section 3.6.7).

ever, we do not limit ourselves to using the same detector at all resolutions. We choose a sequence of d detectors, cascaded from coarse to fine resolution, that use inexpensive tests on the image features. Each detector narrows down the set of possible explanations to a smaller number, which are explored in greater detail by the next detector. This architecture is illustrated in figure 3.4. Each detector $D_1 ... D_d$ computes a quantity $q_1 ... q_d$ for each region of hypothesis space that is explored, this region is accepted for consideration at the next level of resolution if the quantity exceeds a given threshold. All thresholds are set automatically based on probabilistic measurements (see Section 3.6.7).

This coarse-to-fine approach is in contrast with [Low04], where Lowe's recognition system is less complex, as it uses only a Hough transform and a simple outlier rejection stage.

One of the main goals of this paper is to provide a principled, probabilistic interpretation of the heuristics used by Lowe. With the coarse-to-fine framework, after each module we update the probabilities of the possible explanations of the test image. We describe here the basic steps in the hypotheses filtering process; the probabilistic interpretation will be introduced in Section 3.5 and

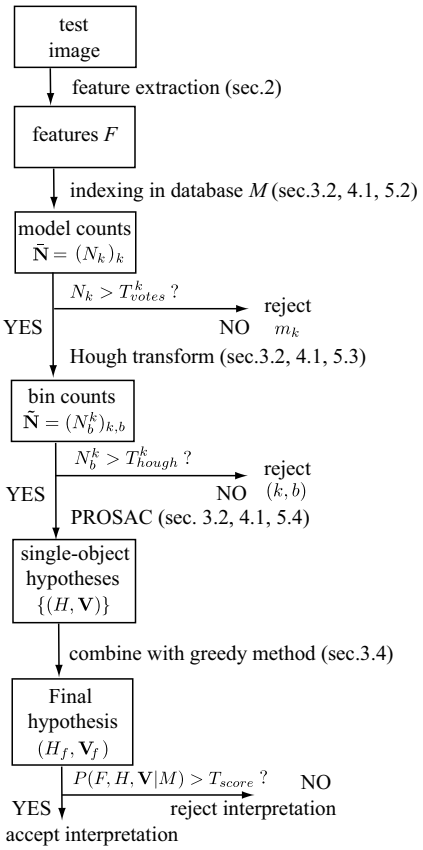


Figure 3.5: Sequence of steps for the recognition process — see notation in table 3.1.

Section 3.6.

3.4.2.1 Model voting

The first screening of the hypothesis space is done by indexing the test features into the database, i.e., searching in the database for candidate matches to the features extracted from the test image. This indexing is based on appearance only, and performed by a kd-tree structure [FBF77], with backtracking to improve its accuracy [BL97]. Either one or more closest neighbors to the test feature are selected (one neighbor only in the experiments described in Section 3.7), to form *candidate matches* (f_i, f^k_j). Using Lowe's criterion [Low04] on the ratio between the distance to nearest neighbor and distance to second-nearest neighbor, some test image features might not be associated to any database feature and automatically be considered as background.

The observed variable $\tilde{\mathbf{N}} = \{N_k\}_k = (N_1 ... N_{\mathcal{M}})$ indicates how many test features were associated to database features associated with each object. Only the objects that collected a sufficient

number of matches — defined by a threshold T^k_{votes} — are considered in the subsequent steps. We will see in Section 3.6.7 that this threshold is set automatically based on probabilistic measurements on each object.

3.4.2.2 Coarse Hough transform

We use the Hough transform [Bal81, Low04] as a means to enforce pose consistency amongst candidate feature matches. The purpose of this Hough transform is to create clusters of candidate matches that support similar model-to-test_image transformations.

The features encode location, orientation, and scale, thus a single candidate match (f_i, f^k_j) between a test feature and a database feature, provides enough information to characterize a similarity transform from model to test image (see figure 3.3). For this reason, at this stage we restrict ourselves to the space of similarity transforms [Low99, Low04]. For each known object, the Hough space of transform parameters is discretized into coarse bins (one table of bins for each known object), and the candidate matches are hashed into these bins. Each bin in Hough space can be considered to be a single-object hypothesis with a pose defined coarsely. This bin corresponds to a particular object, and specifies a set of possible poses that this object can take in the composition. In other words, the bins identify clusters of candidate matches in pose agreement with each other. Another possibility for finding clusters in Hough space would be to use Mean-Shift Mode Estimation [Che95] (as, e.g., in [LLS04].

The size of the bins is discussed in Section 3.6.8. The main point is that we choose very large bins, in particular they are significantly larger than the bins chosen in [Low04].

The choice of a coarse discretization — as opposed to choosing bins with small dimensions — makes the exploration of the Hough space a fast process. Besides, the coarse discretization causes the boundary-related hashing issues to be less evident than in [Low04]. These boundary issues occur whenever a cluster is split in two by a bin boundary. Partitions with smaller bins are more frequently affected by this problem since boundaries are in higher number. The disadvantage of having large bins, is that this Hough transform step is not accurate enough to fully characterize fine-scale hypotheses.

The number of candidate matches that index into each bin is a first estimate of the supporting

evidence for the coarse hypothesis that it represents. The variable $\tilde{\mathbf{N}}$ denotes the number of candidate matches falling in each $(object, pose)$ bin. $\tilde{\mathbf{N}} = \{N_k^b\}_{k,b}$ where k indexes the object and b the pose bin. Similarly to the model voting stage, only the combinations of object and pose that collected a sufficient number of matches — defined by a threshold T_{hough}^k, see Section 3.6.7 for the automatic determination of this threshold — are considered in the subsequent steps.

3.4.2.3 Generation of single-object hypotheses using PROSAC

The next step of our coarse-to-fine refinement process generates fine-scale single-object hypotheses. Each bin from the Hough transform space is considered in turn, starting from the most populated bins. As mentioned above in the description of the Hough transform, the bins are very large, so that a bin that contains a correct hypothesis might contain a large number of incorrect candidate features matches. These outliers hinder the recovery of the model pose in the test image and might lead to false alarms. They must be rejected.

Our outlier rejection stage uses the PROSAC algorithm [CM05]. PROSAC is similar to the popular RANSAC algorithm [FB81]. Each Hough bin that passed the previous tests is considered independently. For a given bin, random subsets of 4 matches (in our case) are repeatedly sampled randomly from the set of candidate matches in this bin. A global pose is computed from each sample, and the consistency of all tentative correspondences with this pose is measured. The winning pose is the pose that obtains the largest consensus set, i.e., the highest number of consistent correspondences. Unlike RANSAC, the samples used by PROSAC are not uniformly distributed. They are drawn from progressively larger sets of tentative correspondences (*sampling sets*). A higher priority is given to the tentative correspondences with the highest quality in terms of similarity between descriptors. PROSAC thus relies on the assumption that ordering tentative correspondences by decreasing similarity of the descriptors converges faster to a good solution than random ordering.

The function that specifies the growth of the sampling set is a trade-off, it indicates how much we rely on the descriptor similarity being a good indicator of true matches. If the sampling set grows slowly, we will sample only from the tentative correspondences with highest similarity ('bag of words' model). If the sampling set grows fast, we are back to RANSAC, where the

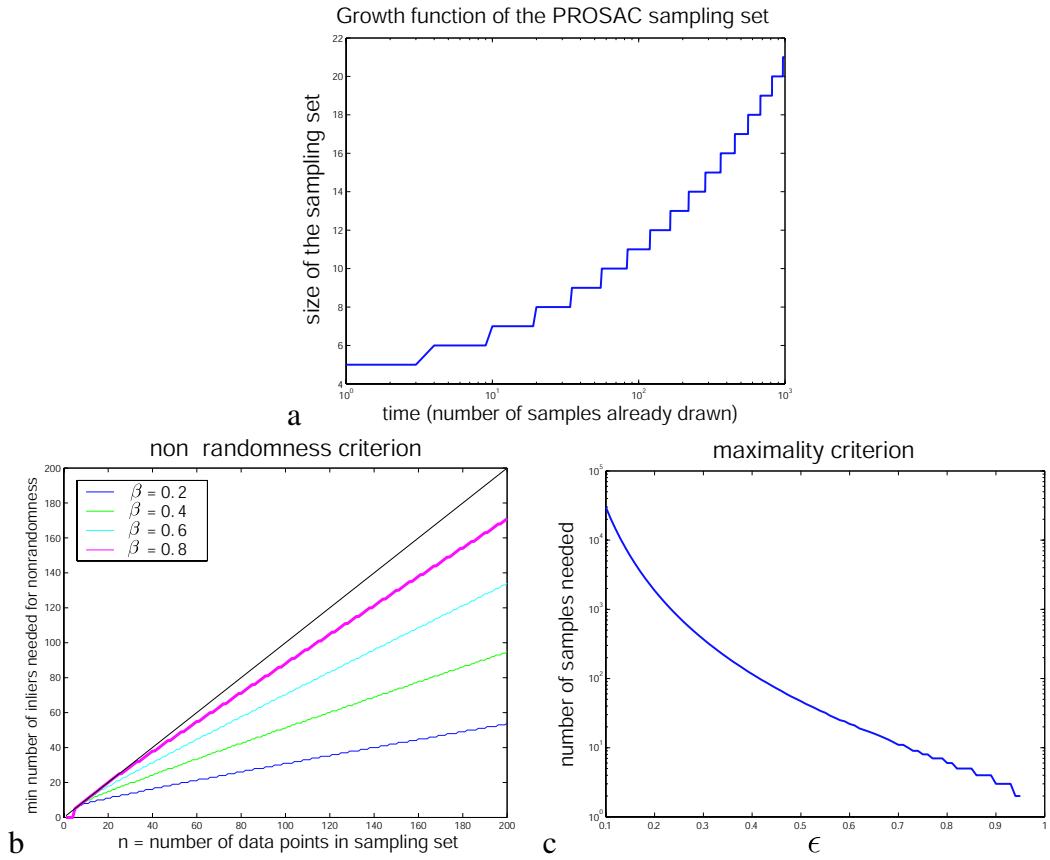


Figure 3.6: (a) 'Growth function' for PROSAC [CM05]. It specifies the size of the sampling set as a function of time. Each time step corresponds to one sample being drawn. The sampling set is the set of candidate matches with best quality of appearance match, from which PROSAC samples are drawn. This curve is computed for our situation with samples consisting of 4 data points. (b)-(c): Stopping criteria for PROSAC. (b) Number of inliers required in the consensus set, as a function of the size of the sampling set. The parameter β is the probability that an incorrect pose calculated from a random sample containing an outlier is supported by a correspondence not included in the sample. (c) Number of PROSAC iterations needed. ϵ is the fraction of inliers among the sampling set, i.e., the percentage of tentative correspondences from the sampling set that end up in the consensus set (it is computed 'on the fly').

samples are drawn uniformly from the full set of tentative correspondences. The optimal growth function [CM05] for our situation with 4 matches per sample (see below) is displayed in figure 3.6-(a).

In our application, we use an affine transformation between models and test image. Using the (x, y) location information from the features, 3 candidate matches are needed to compute the 6 degrees of freedom of an affine transformation, using least-squares fit [Low99, Low04]. A method for incorporating information from the scale and orientation data is discussed in the appendix. Note

that if the samples contain 3 candidate matches, the residual error of these matches is always zero and these matches are systematically part of the consensus set. To reduce this discrepancy between the role of the sample and the rest of the consensus set, we chose a sampling size of 4 matches per sample. As a consequence, many incorrect bins have a best consensus set that is empty (see figure 3.7-(i)), they can be discarded immediately.

The stopping criteria for PROSAC consist of two conditions: first, the consensus set needs to be large enough so that the probability of the pose being incorrect is negligible (*non-randomness criterion*). This criterion specifies the size of the consensus set needed to stop sampling repeatedly (figure 3.6-(b)). It depends on one parameter called β, which is the probability that an incorrect pose computed from a sample is supported by a data-point not included in the sample. The value of this parameter is difficult to compute; we chose a conservative value of 0.8 (curve with thicker line in figure 3.6-(b)). The non-randomness threshold is computed only once, before running PROSAC.

Second, we need to draw enough samples so that the probability of finding a better consensus set than what has been already explored, is also negligible (*maximality criterion*). This criterion specifies the number of sampling iterations needed to stop (figure 3.6-(c)). The parameter for this criterion is ϵ, the fraction of inliers among the sampling set (it is computed 'on-the-fly' while iterating). Through ϵ, the maximality criterion depends on the size of the population in the bin of interest. ϵ is computed 'on-the-fly' at each iteration.

We use the following measure of pose consistency of a candidate correspondence (f_i, f_j^k) with respect to the pose H computed from a sample: the score contributions $p_{fg}(f_i|f_j^k, H)$ and $p_{bg}(f_i)$ (see equations (3.30) and (3.33)) are computed for both alternatives that would accept the candidate correspondence as a true match, or reject it and assign the test feature to clutter. In the bin under consideration, the candidate matches that verify $p_{fg}(f_i|f_j^k, H) > p_{bg}(f_i)$ are accepted; these candidate matches form the consensus set. The largest consensus set is accepted, and its matches are used to update its geometric pose H via a new least-squares fit.

The output of PROSAC is twofold. On one hand, we obtain a partial assignment vector $\mathbf{V}$, namely the winning consensus set. On the other hand, we obtain a single-object hypothesis H, i.e., the combination of the object in the largest consensus set and the geometric pose computed from it.

We will see in Section 3.7.4 that PROSAC is very efficient at reducing the number of hypotheses to be considered. We use it in conjunction with the Hough transform rather than as a single filtering stage for two reasons. First, we need a tool to select clusters of candidate matches with good consistency. The Hough transform is an efficient tool for this purpose. Second, as mentioned in [Low04], algorithms derived from RANSAC do not perform well when the fraction of outliers in the data is high, as the number of iterations and computation time become prohibitive.

After completing PROSAC, we have a set of single-object hypotheses. We will discuss in Section 3.4.4 how to combine multiple single-object hypotheses detected in a same test image, into a single multi-object hypothesis. This will help reduce the rate of false alarms. Single-object and multi-objects hypotheses can then be rated using the probabilistic score developed in Sections 3.5–3.6.

3.4.3 An example

The matching process explained in the previous sections is illustrated in figure 3.7. We use a small data-set that contains 31 images of 31 known objects. (a) displays a test image, (b) the models of the three objects from the data-set that are present in the test image.

(c) shows the result of the model voting stage. The object index is on the horizontal axis, the vertical axis is the number of votes. Values for correct objects are displayed in different shades of green and blue, for models not present they are shown in red. A black horizontal bar shows the threshold T^k_{votes} (see figure 3.5, Sections 3.4.2.1 and 3.6.7) on the minimum number of votes needed for the object to be considered in further steps — when the object's bar is above the red bar, the object did not pass this test. In this example, 6 models can be rejected at this stage.

(d) shows the output of the Hough transform stage. Here the horizontal axis indexes the bins with more than T^k_{hough} candidate matches (in number 52). The vertical axis is the number of candidate correspondences falling in each of these bins. Similarly to (c), the value of the threshold T^k_{hough} is displayed by a black horizontal bar (see figure 3.5, Sections 3.4.2.2 and 3.6.7). Correct sets of matches not only need to address an object present in the test image, but also the candidate matches contained in them have to specify a pose consistent with the true pose. We use the same

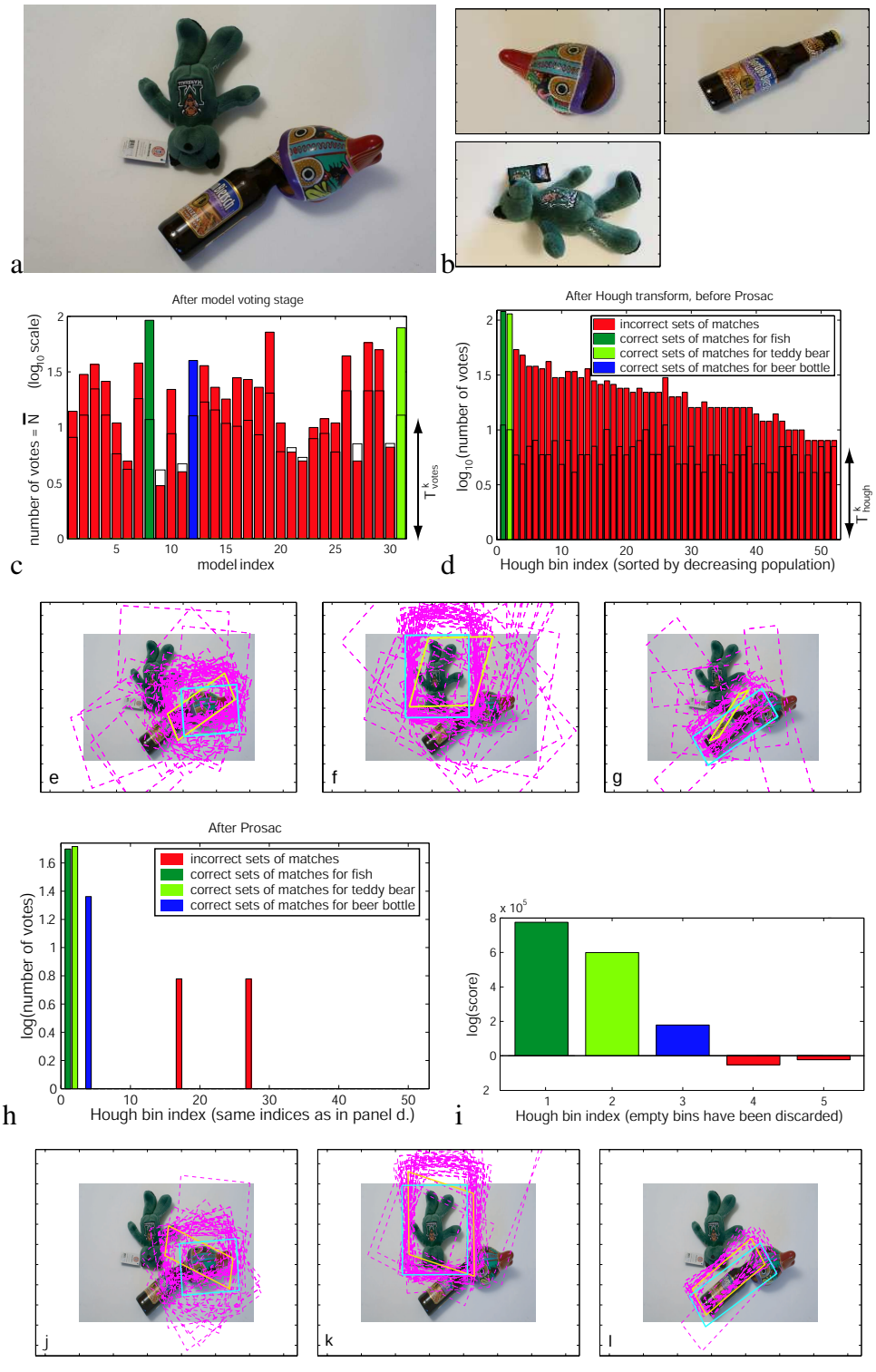


Figure 3.7: Example of matching process. See Section 3.4.3 for details about the various panels.

consistency measure as in [ea06a]: a hypothesis is considered consistent with the true pose if the ratio of overlap a_0 between the bounding box B_p predicted by this hypothesis and the ground truth bounding box B_{gt}, exceeds 50% by the formula:

$$a_o = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}. \tag{3.2}$$

The ground truth bounding box B_{gt} is labeled manually. The predicted bounding box B_p is computed by using a least-square error fit to estimate affine parameters for the object pose in the test image, using all candidate correspondences collected in the bin of interest. Note that for the beer bottle, none of the sets of matches from the various bins passes the consistency test from equation 3.2. (g) shows the bin corresponding to the ground truth pose and the candidate matches in it. Too many outlier candidate matches pollute this bin, and the transformation computed from these matches (in yellow) does not meet the consistency criterion. We will see in (h) and (i) that after the PROSAC outlier rejection, the transformation computed from the remaining candidate matches passes the consistency test successfully. Note that if no outlier rejection was applied after the Hough transform stage, the test image would lead to numerous false alarms.

(e)–(f)–(g) show the ground truth bounding box (blue) and the predicted bounding box (yellow) for the most populated bins corresponding to the 3 models present in the test image (bins $\#1, 2, 4$ in (d) correspond to the fish, teddy bear and beer bottle). As mentioned in Section 3.4.2.2, each single candidate match provides enough information to characterize a similarity transform between the corresponding model and the test image. The bounding boxes predicted by the matches in the bin under consideration are displayed in magenta.

Note that (d) still contains 52 single-object hypothesis. Using the results of the recognition process after this stage, without further filtering, would entail an enormous false-positive rate.

(h) displays the same information as (d) after applying outliers rejection with PROSAC. As expected, most incorrect bins have been decimated since the algorithm was not able to find an object that obtained any significant pose agreement. On the other hand, in correct bins a high fraction of candidate matches survive as inliers. The predicted bounding box B_p is recomputed using the same least-squares error fit as in (d), and displayed in panels (j)–(k)–(l) for the same bins

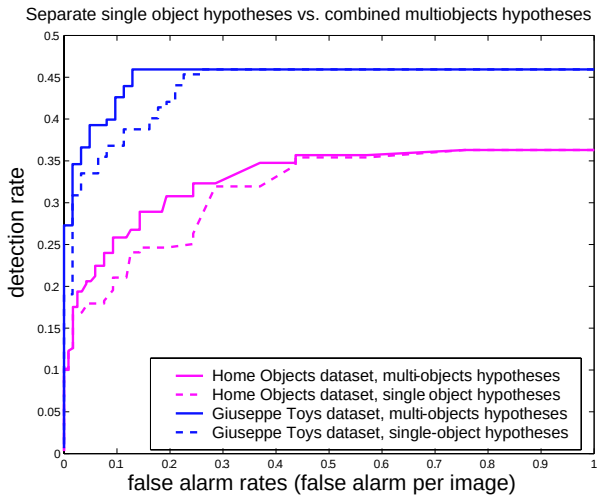


Figure 3.8: Performance gain from combining single-object hypotheses into multi-objects hypotheses. Data-sets described in Section 3.7.

as (e)–(f)–(g). Note: in (d)–(e)–(f)–(g) all the candidate matches that indexed to the bin of interest were used in the least-squares fit. Here only the candidate matches that survived the PROSAC step are used to compute the geometric pose. Another observation is that the fourth bin (beer bottle), now verifies the criterion from equation 3.2 (see (l))

(i) shows the probabilistic scores (see Sections 3.5–3.6) obtained by each bin considered as a separate single-object hypothesis (test features not included in the bin are assigned to clutter in order to obtain a 'full' hypothesis). The bins which obtained an empty consensus set during the PROSAC stage have been discarded in this graph.

3.4.4 From single-object to multiple-object hypotheses

The previous sections were concerned with building separate single-object hypotheses: each of these hypotheses considers one object to be present in the test image, and all features that were not assigned to this object are declared to be background detections.

In order to combine these single-object hypotheses into a final hypothesis H_f (and assignment vector $\mathbf{V}_f$) containing multiple objects, we follow a greedy approach. $(H_f, \mathbf{V}_f)$ are initialized with the single-object hypothesis $(H_1, \mathbf{V}_1)$ that obtained the highest score. This hypothesis corresponds to the object that we believe most to be present in the test image.

We consider next the single-object hypothesis $(H_2, \mathbf{V}_2)$ that obtained the next highest score,

Algorithm 1 Recognition algorithm — see notation in table 3.1

Compute features $\{f_i\}$ from test image.
Index test features into database. Compute model counts in $\bar{\mathbf{N}}$. Discard objects that collect less than T^k_{votes}.
Form candidate matches, discretize Hough space into bins, index candidate matches into bins. Record bin counts in $\tilde{\mathbf{N}}$. Discard bins that collect less than T^k_{hough} candidate matches.
Index features from objects that survived previous steps into the test image. Add these new matches to Hough table.
for each remaining bin **do**
  PROSAC: Sort candidate matches by decreasing quality of underlying appearance match. Create new empty single-object hypothesis H and empty assignment vector $\mathbf{V}$.
  while iterated less than n_{iter} times and PROSAC's non-randomness and maximality criteria are not satisfied **do**
    Choose four candidate matches, compute parameters of affine transformation H between model and test image via least-squared error fit.
    for each candidate match (f_i, f^k_j) in current bin **do**
      if likelihood ratio $R_i = \frac{p_{fg}(f_i|f^k_j,H)}{p_{bg}(f_i)} > 1$ **then**
        insert candidate match in assignment vector $\mathbf{V}$
      else
        assign test feature to clutter
      end if
    end for
  end while
  Assign remaining test features to clutter and compute score of $(H, \mathbf{V})$
end for
Initialize multi-object hypothesis $(H_f, \mathbf{V_f})$ with single-object hypothesis which obtained highest score.
while not all single-object hypotheses have been considered **do**
  Consider single-object hypothesis $(H, \mathbf{V})$ with next highest score.
  if $score(H_f, H) > score(H_f)$ **then**
    Include matches from $(H, \mathbf{V})$ into $(H_f, \mathbf{V_f})$.
    Discard these matches from all remaining single-object hypotheses.
  end if
end while
if $score(H_f, \mathbf{V_f}) > T_{score}$ **then**
  output $(H_f, \mathbf{V_f})$ as the solution
else
  output null hypothesis
end if
Output

Table 3.1: Notation

	Features
f	Generic notation for a feature, whether it was generated by a known object or by a part in the composition. f_i is the i-th feature from the test image, f_j^k is the j-th feature from k-th object. $f = (\mathcal{X}, \mathcal{A})$ is composed of pose $\mathcal{X}$ and appearance $\mathcal{A}$
$\mathcal{X}$	Pose information in a feature (4 dimensions: x & y location, scale s, orientation θ). $\mathcal{X}_i$ (resp. $\mathcal{X}_j^k$) is the pose associated to f_i (resp. f_j^k)
$\mathcal{A}$	Similarly, appearance information included in a feature
F	Set of features observed in the test image
M	Database of features generated by the images of known objects
$\mathcal{M}$	Number of known objects in the database
$\mathbf{n}$	Number of features in each known object, arranged in a vector of size $\mathcal{M}$
	Hypotheses and correspondences
H	Hypothesis. Includes the identity of the object identified in the test image (indicated by a vector of objects $\mathbf{m}$), along with their pose, indicated by a vector of pose parameters $\mathbf{\Theta}$. ($\mathbf{\Theta}$ points to as many poses as there are objects in H). For a feature f from an object that H depicts in the composition, $H(f)$ denotes the image of f by the model-to-test_image transformation specified by H. $\mathcal{H}$ is the number of objects specified by H to be in the test image.
$\mathbf{V}$	Vector of assignments. $V(i) = (k, j)$ means that f_i is associated to f_j^k, $V(i) = 0$ means that f_i is considered a clutter detection. $\mathbf{V}_k^b$ is the restriction of $\mathbf{V}$ to object k and Hough space bin b. $N_{(k,b)}^V$ is the number of features assigned by $\mathbf{V}_k^b$ to database features (as opposed to clutter).
	Intermediate variables
$\bar{\mathbf{N}}$	Votes counts per object. Vector recording how many times the initial indexing of test features in the database returned features from each object. $\bar{\mathbf{N}} = (N_1 ... N_{\mathcal{M}})$. T_{votes}^k is the minimum number of votes that object m_k must collect to be considered in subsequent recognition stages.
$\tilde{\mathbf{N}}$	Bin counts. Vector recording how many votes each bin from Hough space collected after the Hough transform. $\tilde{\mathbf{N}} = (N_k^b)_{k,b}$ where k indexes the object and b indexes the pose bin. T_{hough}^k is the minimum number of votes that a bin must collect to be considered in subsequent recognition stages.
T_{score}	Threshold on final score, depending on the operating point chosen by the user
	Parameters
λ	Intensity of the Poisson process characterizing the frequency of clutter detections in the test image. λ is a number of features per unit image area.
p_{det}	Probability of detecting a given database feature in the test image
p_{stray}	During initial indexing process, probability of accidentally matching a feature from the test image to a feature from the database

and combine it with $(H_1, \mathbf{V}_1)$ by adding to H_1 the object and pose specified by H_2, and adding to $\mathbf{V}_1$ the foreground features specified by $\mathbf{V}_2$, i.e., the features associated by $\mathbf{V}_2$ to a known object and not clutter. In the case when H_1 and H_2 describe the same object and poses with footprints in the test image that overlap by more than 50%, the two hypotheses are considered one and the same detection. In this case, no new object instance is created, and the matches from $\mathbf{V}_2$ are simply added to those from $\mathbf{V}_1$. We assume the known objects to be opaque, therefore the foreground features from $\mathbf{V}_1$ are discarded from $\mathbf{V}_2$. This corresponds to the assumption, also made in [LLS04], that the object with the highest likelihood is also the one that appears in front. The score (cf., Sections 3.5–3.6) of the combined hypothesis and assignment vector is computed. If this score is higher than the score of $(H_1, \mathbf{V}_1)$ alone, the object instance specified by H_2 is added to our interpretation of the test image, and the combined hypothesis and assignment vector are accepted as new value for $(H_f, \mathbf{V}_f)$. Otherwise, $(H_2, \mathbf{V}_2)$ is rejected and $(H_f, \mathbf{V}_f)$ stays unmodified. This greedy approach is in agreement with our maximum-likelihood framework. The process is repeated with the other single-object hypotheses sorted by order of decreasing score, until either all single-object hypotheses have been considered, or all test features are matched to features from known objects.

Note that this greedy iterative process is only an approximation. In theory, one should first consider all combinations of two single-object hypotheses, then all combinations of three single-object hypotheses, and so on. The computation requirements entailed by this combinatorial explosion would be prohibitive, therefore we adopted the greedy approach described above. A greedy approach was also used in the systems for detection of object categories proposed by [LLS04, MLS06]. In [Low99, Low01, Low04], Lowe considers and aggregates all identified objects independently into a global hypothesis. This approach is very simple, but can lead to false alarms, in particular when a same feature can vote for several object poses (in [Low04], a same feature can vote for 16 adjacent bins in Hough transform space).

The performance gain obtained from combining single objects hypotheses into multi-object hypotheses is illustrated in figure 3.8. The probabilistic scores used to compute these ROC curves will be described in detail in the next sections. Depending on the operating point chosen by the user, a threshold T_{score} on the score of the combined hypotheses determines if the hypotheses is

output as interpretation of the test image or if we believe that no known object is present in the test image.

The performance gain is comparable to that reported by Leibe in [LLS04] using his MDL verification stage. A majority of the false positive cases is due to secondary hypotheses that overlap partially with an object's detection. Combining separate hypotheses helps discard these secondary hypotheses or merge them with the primary detection (what we call primary detection is the detection with highest score, i.e., the detection that collected the best evidence with respect to the presence of the object in the test image).

One must note that the detection performance on the right-hand side of the curve (regime with high false-alarm rate) is not modified. Combining single-object hypotheses into a multiple-object one does not create new detections, it only helps deciding if hypotheses should be accepted or rejected by incorporating more accurately information from the whole test image into the probabilistic score.

3.5 Probabilistic interpretation of the coarse-to-fine search

Our probabilistic treatment reflects the coarse-to-fine strategy described in Section 3.4.2 and illustrated in figure 3.5 and figure 3.7. We want to rate the hypotheses and assignment vectors output by our recognition algorithm in order to decide which detections should be accepted or rejected. We develop a principled, probabilistic approach in order to improve upon 'ad hoc' tuning of the detection system used, e.g., in [Low99, Low04]. The thresholds used in our algorithms will also be chosen based on our probabilistic model (see Section 3.6.7).

3.5.1 Probabilistic decomposition

In this section, we want to rate combinations of hypotheses and assignment vectors, in the light of the features detected in the test image and in the database, i.e., we want to evaluate $P(H = H_{true}, \mathbf{V} = \mathbf{V}_{true}|F, M)$, also written $P(H, \mathbf{V}|F, M)$. This is the joint probability that hypothesis H describes the set of objects truly present in the test image with their correct pose, and that the assignment vector $\mathbf{V}$ describes the true correspondences between model and test image, conditioning

on the observations F from the test image and M from the database.

From the definition of conditional probabilities,

$$P(H, \mathbf{V}|F, M) = \frac{P(F, H, \mathbf{V}|M)}{P(F|M)} \tag{3.3}$$

where $P(F|M)$ is a prior on features observations which does not depend on H or $\mathbf{V}$. Our goal is maximizing $P(H, \mathbf{V}|F, M)$ with respect to H and $\mathbf{V}$, therefore $P(F|M)$ has no influence on the solution and we can omit it. The database of features M is acquired off-line and is not a variable, therefore we can omit the condition on M in further equations. We now examine $P(F, H, \mathbf{V})$, which we will call *score*, as it rates how close H is to H_{true} and $\mathbf{V}$ to $\mathbf{V}_{true}$. Using the additional variables $\bar{\mathbf{N}}$ and $\tilde{\mathbf{N}}$ defined in Section 3.4.2 (these variables are deterministic functions of $F, H, \mathbf{V}, M$), we obtain

$$\begin{aligned} &P(F, H, \mathbf{V}) = P(F, H, \mathbf{V}, \tilde{\mathbf{N}}, \bar{\mathbf{N}}) = \\ &P(F|\mathbf{V}, \tilde{\mathbf{N}}, \bar{\mathbf{N}}, H) \cdot P(\mathbf{V}|\tilde{\mathbf{N}}, \bar{\mathbf{N}}, H) \cdot P(\tilde{\mathbf{N}}|\bar{\mathbf{N}}, H) \cdot P(\bar{\mathbf{N}}|H) \cdot P(H). \end{aligned} \tag{3.4}$$

This decomposition reflects the algorithmic steps detailed in the previous section. We first describe the meaning of each one of these terms. In Section 3.6 we will explain in detail how each term is computed.

— $P(H)$ is a prior on all possible coarse hypotheses. It contains information on which objects are most likely present, together with their most probable pose. In principle, $P(H)$ may also encode 'context,' i.e., which objects are likely to show up in combination within a test image, and what is their likely mutual position. Our model for $P(H)$ is described in Section 3.6.1.

— $P(\bar{\mathbf{N}}|H)$ predicts the number of test features that will be associated to each object during the initial search phase, so this is a "bag of features" model. The features associated to each object come from two separate contributions. First, if the object is present in the test image ($m \in H$), some test features will be correctly associated to database features from this model. Secondly, the test image will contain background features unrelated to the objects that are present. During the matching process, a fraction of these background features will be erroneously associated to known objects.

The term $P(\bar{\mathbf{N}}|H)$ is used to discard unpromising objects after the initial model voting step described in Section 3.4.2. Our model of $P(\bar{\mathbf{N}}|H)$ is described in Section 3.6.2.

— $P(\tilde{\mathbf{N}}|\bar{\mathbf{N}}, H)$ models the spread of candidate matches in the Hough space. If all features were detected with exact position, scale, and orientation, all correct matches should fall in the same bin, namely the bin that contains the pose parameters specified by H. Errors in the measurement of features' location, orientation, and scale, cause these matches to spread to adjacent bins as well. Note that for a given object, the sum of the values of $\tilde{\mathbf{N}}$ over all bins, is equal to the value of $\bar{\mathbf{N}}$ on that object. Thus $\bar{\mathbf{N}}$ provides a constraint on the values that $\tilde{\mathbf{N}}$ can take. The term $P(\tilde{\mathbf{N}}|\bar{\mathbf{N}}, H)$ is computed after the Hough transform step described in Section 3.4.2. Our model of $P(\tilde{\mathbf{N}}|\bar{\mathbf{N}}, H)$ is described in Section 3.6.3.

— $P(\mathbf{V}|\tilde{\mathbf{N}}, \bar{\mathbf{N}}, H)$. The assignment vector $\mathbf{V}$ specifies for each image feature, whether it is associated to a database feature or considered a clutter detection. Note that pose and appearance information on the test features (variable F) is not present in this term, it will be taken into account in the next term. Similarly to $P(\bar{\mathbf{N}}|H)$, only counts of features matches are taken into account, thus this term is purely a combinatorial probabilistic expression. The term $P(\mathbf{V}|\tilde{\mathbf{N}}, \bar{\mathbf{N}}, H)$ can be interpreted as a prediction of how many features from the test image will be eventually put in correspondence with database features, given the initial counts provided by $\bar{\mathbf{N}}$ and $\tilde{\mathbf{N}}$. Our model of $P(\mathbf{V}|\tilde{\mathbf{N}}, \bar{\mathbf{N}}, H)$ is described in Section 3.6.4.

— $P(F|\mathbf{V}, \tilde{\mathbf{N}}, \bar{\mathbf{N}}, H) = P(F|\mathbf{V}, H)$. This term compares features values predicted in the test image by $(H, \mathbf{V})$ with the values actually observed. According to $\mathbf{V}$, the test feature f_i is associated to the database feature $f_{\mathbf{V}(i)}$. This feature is projected in the test image according to the transformation specified by H. The predicted feature, i.e. the image by the transformation in H of $f_{\mathbf{V}(i)}$ (denoted by $H(f_{\mathbf{V}(i)})$), is compared with the feature actually observed f_i. The discrepancy in feature pose and appearance between f_i and $H(f_{\mathbf{V}(i)})$ is attributed to a combination of measurement noise and modeling error.

The term $P(F|\mathbf{V}, H)$ and the term $P(\mathbf{V}|\tilde{\mathbf{N}}, \bar{\mathbf{N}}, H)$ are computed at each iteration of the PROSAC process described in Section 3.4.2. Our model of $P(F|\mathbf{V}, \tilde{\mathbf{N}}, \bar{\mathbf{N}}, H) = P(F|\mathbf{V}, H)$ is described in Section 3.6.5.

3.6 Models

We now present detailed models for the components of the hypothesis score described in Section 3.5.1.

3.6.1 Prior $P(H)$

Let $H = ((m_1, \Theta_1)...(m_{\mathcal{H}}, \Theta_{\mathcal{H}}))$ be a hypothesis. Conditioning on the number $\mathcal{H}$ of known objects present in this hypothesis, we decompose $P(H)$ into

$$P(H) = P\big((m_1, \Theta_1)...(m_{\mathcal{H}}, \Theta_{\mathcal{H}}), \mathcal{H}\big) \tag{3.5}$$

$$= P\big((m_1, \Theta_1)...(m_{\mathcal{H}}, \Theta_{\mathcal{H}})|\mathcal{H}\big) \cdot P(\mathcal{H}) \tag{3.6}$$

$$= P(\Theta_1....\Theta_{\mathcal{H}}|m_1...m_{\mathcal{H}}) \cdot P(m_1...m_{\mathcal{H}}|\mathcal{H}) \cdot P(\mathcal{H}). \tag{3.7}$$

The first term in equation 3.7 expresses context information on relative poses of objects in the composition. Given that we believe a keyboard, a mouse, and a screen to be present in the test image, the keyboard is likely to be located below the screen with the mouse to its right or left. The second term is the probability of having a certain set of $\mathcal{H}$ objects among the $\mathcal{M}$ objects available in the database. The third term is simply a probability density on the cardinality of hypotheses.

For the sake of simplicity, in the experiments presented in Section 3.7 we will use the simplest setting with mutually independent poses of the objects present in the composition, thus

$$P(H) = \prod_{1 \leq i \leq \mathcal{H}} P(\Theta_i|m_i) \cdot P(m_1...m_{\mathcal{H}}|\mathcal{H}) \cdot P(\mathcal{H}). \tag{3.8}$$

In the absence of prior knowledge regarding poses, $P(\Theta_i|m_i)$ is modeled by the product of a uniform density over the test image area $\mathcal{A}$ for translation, a uniform density over $[0, 2\pi]$ for orientation, and a uniform density in log-scale. The range of values chosen for the log-scale in our experiments is $S = [2^{-4}, 2^4]$. If all objects have equal probability to be photographed in the test image, $P(m_1...m_{\mathcal{H}}|\mathcal{H}) = 1/\binom{\mathcal{M}}{\mathcal{H}}$ ($\binom{\mathcal{M}}{\mathcal{H}}$ is the number of ways of picking $\mathcal{H}$ models among the $\mathcal{M}$ available in the database). $P(\mathcal{H})$ is modeled by a Poisson distribution. The parameter $\lambda_{\mathcal{H}}$ of this

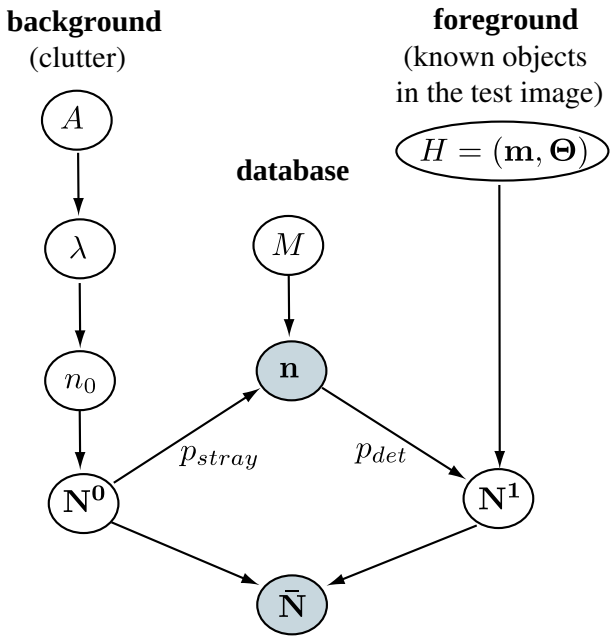


Figure 3.9: Diagram for the model voting counts observed in the first stage — see table 3.1 for the notation. The grey-shaded nodes indicate variables from the recognition system that are directly observable.

distribution should be the average number of objects observed per test image. Therefore,

$$P(H) = (\mathcal{A} \cdot 2\pi \cdot S)^{-\mathcal{H}} \cdot \binom{\mathcal{M}}{\mathcal{H}}^{-1} \cdot pois_{\lambda_{\mathcal{H}}}(\mathcal{H}). \tag{3.9}$$

3.6.2 Model votes $P(\bar{\mathbf{N}}|H)$

The model used to explain the number of candidate matches voting for each object is illustrated in figure 3.9.

The database contains $\mathbf{n} = (n_1 ... n_{\mathcal{M}})$ features. The database features associated to objects specified by H are expected to be detected in the test image with probability p_{det}. These 'stable' features, in number $\mathbf{N^1}$, account for the correct correspondences. Here $\mathbf{N^1}$ is a vector with as many entries as objects in the database, but only entries corresponding to objects in H are different from zero.

On the other hand, the background generates unwanted clutter features in the test image. Other sources of clutter features include imaging noise, and texture from the known objects that was

not captured in the training models. The number n_0 of clutter detections is modeled by a Poisson distribution, the mean $\lambda(A)$ of this distribution is proportional to the test image area A. We use the full area of the image instead of discounting the footprint of the known objects, because clutter detections occur on the known objects as well. These are features that are 'produced' by the surface of the objects, but are not modeled in the database (e.g., due to the difference in viewpoint between the image used in the database and the test image). Some of these clutter detections will match to database features during the search for candidate correspondences. This results from the indexing of test features into the database, so the arrow in the diagram points at the database (in the diagram from figure 3.2 we represented it pointing from the database to the background features). The fraction of database features that will match to background features is characterized by the probability p_{stray}. The vector of observed model counts corresponding to these unwanted matches is denoted by $\mathbf{N^0}$. Finally, the total number of features that index to each object, i.e., the number of features counted in the model voting stage, is $\bar{\mathbf{N}} = \mathbf{N^1} + \mathbf{N^0}$. Note that $\bar{\mathbf{N}}$ is observable, while $\mathbf{N^1}$ and $\mathbf{N^0}$ are not.

We make the assumption, already used in the previous section, that known objects are independent of each other, and independent of the background, so that the counts of features observed for known objects and background are also independent of each other. Under this assumption, we can factor $P(\bar{\mathbf{N}}|H)$ into

$$P(\bar{\mathbf{N}}|H) = \prod_{k=0}^{\mathcal{M}} P(N_k|H). \tag{3.10}$$

Let's consider a known object m_k that the hypothesis H specifies to be present in the image. The count of features N_k associated to this object is composed of correct matches in number N_k^1, and stray matches to clutter features, in number N_k^0 (N_k^1 and N_k^0 are hidden variables).

Regarding correct matches, the database features from m_k, in number n_k, are observed in the test image with probability p_{det}. We use a constant value of p_{det} equal to 0.1. This is consistent with the observations on stability of features from [MP07b], and with results observed when running Lowe's system [Low99, Low04]. A more elaborate model would consider a value of p_{det} that decreases when the change Θ_k in viewpoint between models and test image is increasing. A

natural model for the count of correct matches is a binomial distribution :

$$P(N_k^1|H) = B(N_k^1|n_k, p_{det}). \tag{3.11}$$

On the other hand the incorrect matches, in number N_k^0, originate from a 'pool of background features,' in number n_0. These stray associations occur with probability p_{stray} (we take $p_{stray} = 0.8$, this is consistent with the fraction of outliers observed in our experiments). We model the probability of observing N_k^0 stray matches by a binomial distribution. The models that have the highest number of features in the database are expected to lead to the highest number of stray matches. Therefore, this binomial distribution is biased towards the models with the highest number of features:

$$P(N_k^0|H) = B(N_k^0|n_0, p_{stray} \cdot \frac{n_k}{\sum_k n_k}) \tag{3.12}$$

where $\frac{n_k}{\sum_k n_k}$ is the fraction of the database features that are generated by object m_k. In order to determine n_0, we assume that the background has on average as much texture as the known objects, so that the background generates features with the same density as do known objects. In other words, the total number of detections is unchanged whether the test image is composed only of background or of known objects. Therefore, n_0 is approximated by the number of features detected in the test image, i.e., $n_0 \approx |\bar{\mathbf{N}}| = \sum_{i=1}^{\mathcal{M}} N_i$.

Finally, N_k is the sum of N_k^0 and N_k^1. All possible combinations of N_k^0 and N_k^1 leading to the same sum should be considered, i.e.,

$$P(N_k|H) = \sum_{N_k^0+N_k^1=N_k} P(N_k^0|H) \cdot P(N_k^1|H). \tag{3.13}$$

In the alternative when the known object m_k is not present in the hypothesis H, equation 3.13 reduces to the 'background term' $P(N_k|H) = P(N_k^0 = N_k|H)$. In this case, according to the hypothesis H, no feature from the known object m_k should be detected in the test image, all candidate matches are spurious, unwanted matches.

The complete expression for $P(N_k|H)$ becomes:

$$\sum_{m_k \in H, N_k^0 + N_k^1 = N_k} P(N_k^0|H) \cdot P(N_k^1|H) + \sum_{m_k \notin H} P(N_k^0 = N_k|H). \tag{3.14}$$

3.6.3 Hough votes on pose: $P(\tilde{\mathbf{N}}|\bar{\mathbf{N}}, H)$

We make the simplifying approximation that bin counts are independent of each other, i.e., that the clusters of candidate matches determined by dividing the Hough transform space into bins are independent of each other. This approximation carries the idea that what happens in one part of the test image is independent of what happens far away. Note that strictly speaking, this approximation is always incorrect, since we always have the constraint $\forall k, \sum_b N_k^b = N_k$.

Under this approximation,

$$P(\tilde{\mathbf{N}}|\bar{\mathbf{N}}, H) = \prod_{k,b} P(N_k^b|\bar{\mathbf{N}}, H) = \prod_{k,b} P(N_k^b|N_k, H). \tag{3.15}$$

• If $m_k \notin H$, i.e., H believes that the known object m_k is not present, all candidate matches counted in N_k are spurious. For a given candidate match, the probability of hashing into any specific bin is uniform over the set of possible bins, this uniform probability is $p_{m_k \notin H} = 1/\mathcal{B}$ where $\mathcal{B}$ is the number of bins in the discretized Hough space. The resulting distribution $P(N_k^b|N_k, H)$ is a Bernoulli distribution

$$P(N_k^b|N_k, H) = B(N_k^b|N_k, \frac{1}{\mathcal{B}}). \tag{3.16}$$

• When $m_k \in H$ we need to take into account the contribution to the bin from two sources. As mentioned in Section 3.6.2, N_k is the sum of correct votes in number N_k^1, and spurious correspondences in number N_k^0.

The correct votes get distributed in the various bins to form the set of Hough votes $\{{}^bN_k^1\}_b$. For a given candidate match, there is a probability p_H^b to hash into bin b. We obtain a Bernoulli distribution on ${}^bN_k^1$ given N_k^1:

$$P({}^bN_k^1|N_k^1) = B({}^bN_k^1|N_k^1, p_H^b). \tag{3.17}$$

Unlike the above case for $m_k \notin H$, p_H^b does not have the same value for all bins. Naturally, the bin with the highest p_H^b is in all likelihood the bin that contains the pose specified by H, we denote it by $b(H)$. If the observations and measurements did not contain any error, all votes from N_k^1 would index to $b(H)$. In practice, we determined p_H^b statistically using the setup described in [MP07b], where ground truth regarding the transformation between a pair of views of a same object is known, and ground truth regarding features correspondences is known as well. We obtained $p_H^b = 0.48$ for the privileged bin $b(H)$, and $p_H^b = 0.06$ for its nearest neighbors. Too few candidate matches indexed into second-order neighbors and farther bins to obtain statistically significant data, for these bins we set a fixed value $p_H^b = 0.001$.

The spurious votes, in number N_k^0, get distributed in all bins with equal probability as in the above case when $m_k \notin H$:

$$P({}^bN_k^0) = B({}^bN_k^0 | N_k^0, \frac{1}{\mathcal{B}}). \tag{3.18}$$

We obtain

$$P(N_k^b | N_k, H) = \sum_{N_k^0 + N_k^1 = N_k} P(N_k^b, N_k^0, N_k^1 | N_k, H) \tag{3.19}$$

$$= \sum_{N_k^0 + N_k^1 = N_k} P(N_k^b | N_k^0, N_k^1, N_k, H) \cdot P(N_k^0, N_k^1 | N_k, H) \tag{3.20}$$

$$= \sum_{N_k^0 + N_k^1 = N_k} P(N_k^b | N_k^0, N_k^1, N_k, H) \cdot P(N_k^0 | N_k, H) P(N_k^1 | N_k, H) \tag{3.21}$$

$$= \sum_{N_k^0 + N_k^1 = N_k} \Big[\sum_{{}^bN_k^0 + {}^bN_k^1 = N_k^b} P({}^bN_k^1 | N_k^1, H) \cdot P({}^bN_k^0 | N_k^0) \Big] \tag{3.22}$$

$$\cdot P(N_k^0 | N_k, H) \cdot P(N_k^1 | N_k, H) \tag{3.23}$$

$$= \sum_{N_k^0 + N_k^1 = N_k} \Big[\sum_{{}^bN_k^0 + {}^bN_k^1 = N_k^b} B({}^bN_k^1 | N_k^1, p_H^b) \cdot B({}^bN_k^0 | N_k^0, 1/\mathcal{B}) \Big] \tag{3.24}$$

$$\cdot B(N_k | n_0 \cdot \frac{n_k}{\sum_k n_k}, p_{stray}) \cdot B(N_k^1 | n_k, p_{det}). \tag{3.25}$$

Equations 3.19 and 3.20 introduce the auxiliary variables N_k^0 and N_k^1 and condition on them; equation 3.21 uses the approximation of independence between foreground and background similarly to the treatment in Section 3.6.2 ; equations 3.22–3.23 use this approximation again for ${}^bN_k^0$ and

${}^bN_k^1$; equations 3.24–3.25 use the density models hypothesized in this section and Section 3.6.2.

3.6.4 Probability of specific assignments $P(\mathbf{V}|\tilde{\mathbf{N}}, \bar{\mathbf{N}}, H)$

This term does not take into account the precise pose and appearance information contained in the features (this information is included in F). Therefore, the probability $P(\mathbf{V}|\tilde{\mathbf{N}}, \bar{\mathbf{N}}, H)$ depends only on the number of features associated by $\mathbf{V}$ to each known object and to clutter.

We denote by $\mathbf{V_b^k}$ the restriction of $\mathbf{V}$ to the bin (k, b), and $\mathcal{V}_b^k$ the number of foreground features in $\mathbf{V_b^k}$ ($\mathbf{V_b^k}$ assigns the remaining $N_b^k - \mathcal{V}_b^k$ features to clutter).

Using the same approximation of objects independence and bin independence as in Section 3.6.3, we decompose $P(\mathbf{V}|\tilde{\mathbf{N}}, \bar{\mathbf{N}}, H)$ into

$$P(\mathbf{V}|\tilde{\mathbf{N}}, \bar{\mathbf{N}}, H) = \prod_{k,b} P(\mathbf{V_k^b}|N_k^b, N_k, H). \tag{3.26}$$

We then obtain, for each model k and each bin b,

$$\begin{aligned} P(\mathbf{V_k^b}|N_k^b, N_k, H) &= P(\mathbf{V_k^b}, \mathcal{V}_k^b|N_k^b, H) && (3.27)\\ &= P(\mathbf{V_k^b}|\mathcal{V}_k^b, N_k^b, H) \cdot P(\mathcal{V}_k^b|N_k^b, H) && (3.28)\\ &= 1/\binom{N_k^b}{\mathcal{V}_k^b} \cdot B(\mathcal{V}_k^b|n_k, p_{det}). && (3.29) \end{aligned}$$

Line 3.27 is due to the relevant information in N_k being already included in N_k^b, and the information from $\mathcal{V}_k^b$ is already included in $\mathbf{V}_k^b$. Line 3.28 conditions on $\mathcal{V}_k^b$. In line 3.29, the first term is due to all $\mathbf{V_k^b}$ with the same number of candidate matches $\mathcal{V}_k^b$ having the same probability. These vectors are in number $\binom{N_k^b}{\mathcal{V}_k^b}$. The expression for the second term is identical to equation (3.11), with $\mathcal{V}_k^b$ used instead of N_k^1. Note that equations 3.27–3.29 describe a proper probability distribution that sums to 1.

3.6.5 Pose and appearance consistency $P(F|\mathbf{V}, \tilde{\mathbf{N}}, \bar{\mathbf{N}}, H)$

We make the assumption that if we condition on the reference frame defined by an object pose in the test image, the test features attributed to this object are independent of each other. This is a

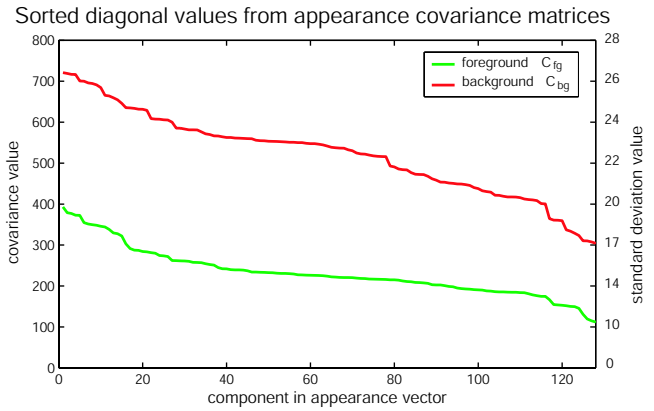


Figure 3.10: Diagonal terms of the appearance covariance matrices for foreground and background, sorted by decreasing values. The appearance components vary in $[0, 255]$. For reference, the corresponding standard deviations are indicated on the right side

'star model' where the center of the star is a hidden variable, namely the reference frame of the object. This assumption was also made in [FH00, FPZ05, MP04], and is implicitly used in [Low99, Low01, Low04]. Compared to [BMW98, FPZ03, WWP00] which learn a joint distribution on the object parts, our assumption of conditional independence dramatically reduces the number of parameters one has to learn, which grows linearly with the number of object parts instead of quadratically. Besides, we do not require a reference landmark part to be present as in the star model used by [FPZ05]. Note that this conditional independence assumption is a simplification ; this approximation does not hold, e.g., if the object is articulated with known segments length, and the features are selected at the vertices. In this case, even if we condition on a known object pose, the position of one part influences its neighbors'.

From the conditional feature independence we obtain

$$P(F|\mathbf{V}, \tilde{\mathbf{N}}, \bar{\mathbf{N}}, H) = \prod_{i|V(i)\neq 0} p_{fg}(f_i|f_{V(i)}, H) \cdot \prod_{i|V(i)=0} p_{bg}(f_i) \tag{3.30}$$

where p_{fg} is the probability of the observed feature's appearance and pose if the candidate match is correct, whereas p_{bg} is the probability of its appearance and pose if the test feature was actually a clutter detection. We call these densities *'foreground'* and *'background'* densities respectively.

If $V(i) \neq 0$, f_i and $f_{V(i)}$ are believed to be caused by the same object part, respectively, in the test image and in a model. Differences measured between f_i and the value predicted by H (i.e., the image of $f_{V(i)}$ after the transformation from H), are an observation noise due to the imaging system

as well as distortions caused by viewpoint or lighting conditions changes. The 'foreground' probability p_{fg} encodes differences in appearance of the descriptors, but also in pose, i.e., location, scale, orientation (pose is omitted, e.g., in [BMP02] due to the use of a richer descriptor that captures the object shape). Assuming independence between pose and appearance, denoted respectively by $\mathcal{A}$ and $\mathcal{X}$, we have

$$p_{fg}(f_i|f_{V(i)}, H) = p_{fg,\mathcal{A}}(\mathcal{A}_i|\mathcal{A}_{V(i)}, H) \cdot p_{fg,\mathcal{X}}(\mathcal{X}_i|\mathcal{X}_{V(i)}, H). \tag{3.31}$$

The error in appearance is measured by comparing the appearance descriptors of the test and database features within candidate matches. We model the density $p_{fg,\mathcal{A}}(\mathcal{A}_i|\mathcal{A}_{V(i)}, H)$ by a Gaussian distribution with diagonal covariance matrix $C_{fg,\mathcal{A}}$. In order to learn the parameters of this distribution, ground truth matches between features in separate views are formed using the same experiments as in [MP07b] (see Section 3.6.6). The vector-valued differences in feature appearance between pairs of views are computed, and the foreground covariance matrix is taken as the covariance of these differences. This covariance matrix characterizes the 'typical' change of appearance between pairs of instances of a same feature, and the density $p_{fg,\mathcal{A}}(\mathcal{A}_i|\mathcal{A}_{V(i)}, H)$ is assumed to be the same for all features in all objects.

The error in feature pose is measured by comparing the position observed in the test image, with the predicted value that would be observed if the database feature was to be transformed according to the pose parameters specified by H. The deviation between predicted values and observed values is the error that we are rating. We model the density $p_{fg,\mathcal{X}}(\mathcal{X}_i|\mathcal{X}_{V(i)}, H)$ on this error by the product of Gaussian distributions for position, orientation and log-scale (regarding orientation, this is actually a Gaussian truncated over $[mean - \pi, mean + \pi]$ and renormalized). The parameters of these densities are also learned from statistics on ground truth matches. For a given set of ground truth matches, a corresponding transformation is computed with least-squares fit. This transformation predicts a location, scale, and orientation in the test image for each feature of the object under consideration. The quantities that interest us are the deviations between predicted values and values actually observed in the test image, for location, scale, orientation. We obtained a standard deviation of $13 * scale$ pixels for x and y location, 19 degrees for orientation, and 0.2 in

log-scale (log in base 2). The value for the x and y location is taken in reference to the scale of the model in the test image ; this is why we denote it by $13 * scale$.

If $V(i) = 0$, f_i is believed to be a clutter detection. Similarly to the 'foreground' treatment, we assume independence between pose and appearance, and decompose $p_{bg}(f_i)$ into

$$p_{bg}(f_i) = p_{bg,\mathcal{A}}(\mathcal{A}_i) \cdot p_{bg,\mathcal{X}}(\mathcal{X}_i). \tag{3.32}$$

The density $p_{bg,\mathcal{A}}$ on appearance of background features is also modeled by a Gaussian density with diagonal covariance matrix. As for the foreground density, this background density is centered for each features on the reference constituted by its nearest neighbor. Therefore this is, similarly to the foreground appearance density, a probability density on *differences in appearance* between a test image feature and its nearest neighbor in the database. In order to compute the parameters of this density, we collect random images and collect interest points from these images. The covariance matrix of the appearance differences obtained with these features, is taken as covariance of the background density $p_{bg,\mathcal{A}}$. The motivation for this choice is that background features are very general — any feature from any image could be a background feature.

The diagonal values of the foreground and background covariance matrices for appearance are showed in figure 3.10 in sorted decreasing order. As one could expect, the values for the background covariance matrix are significantly larger than for the foreground matrix, which corresponds to the intuitive idea that distances for incorrect matches are higher than for correct matches. This observation was also made in [Low04, MP07b].

Similarly to the foreground density on pose, the background density $p_{bg,\mathcal{X}}$ is modeled by the product of independent densities for location, orientation, and log-scale. The location density is modeled by a uniform distribution over the test image, the orientation density by a uniform distribution over $[0, 2\pi]$, and the log-scale density by a uniform distribution over $[-4, 4]$ (log in base 2).

As mentioned in Section 3.5, we accept or reject matches in a candidate assignment vector

based on the likelihood ratio of the match being correct, versus the feature being a clutter detection.

$$R_i = \frac{p_{fg}(f_i|f_j^k, H)}{p_{bg}(f_i)} = \frac{p_{fg,\mathcal{A}}(\mathcal{A}_i|\mathcal{A}_j^k, H) \cdot p_{fg,\mathcal{X}}(\mathcal{X}_i|\mathcal{X}_j^k, H))}{p_{bg,\mathcal{A}}(\mathcal{A}_i) \cdot p_{bg,\mathcal{X}}(\mathcal{X}_i)} \tag{3.33}$$

where $V(i) = (k, j)$. The match is accepted if the value of R_i is above 1, and rejected in favor of a clutter detection otherwise.

It is important to note that the parameters for the densities $p_{fg,\mathcal{A}}$, $p_{fg,\mathcal{X}}$, $p_{bg,\mathcal{A}}$, $p_{bg,\mathcal{X}}$ are shared across features, instead of having one set of parameters for each feature, as in [BMW98, FPZ03, WWP00]. This results in an important decrease of the number of parameters that have to be learned. The trade-off cost is a reduced model expressiveness.

3.6.6 Ground truth matches

In order to learn the parameters of the foreground densities, we need sets of ground truth matches. These measurements can be performed by a user clicking on features in pairs of images, but this is tedious.

In order to automate the ground truth identification process, we used a stereo rig and a computer-controlled turntable [MP07b]. Three-dimensional objects were photographed from calibrated viewpoints obtained by rotating the turntable (image set available from http://www.vision.caltech.edu/archive.html). For a triplet of views of a same object, correspondences between features were established using epipolar constraints. First, these ground truth features matches provide statistics on the number of stable features. Second, they also provide information on changes of appearance between two features representing the same object part in two different images. A third measurement is the inaccuracy in feature position, due to the feature detector and the viewpoint change: with the epipolar constraints, for a given feature in a reference view, we can predict its position in other views. The pose inaccuracy is the deviation from this predicted position, to the actual observed location of the corresponding feature (reprojection error).

Regarding lighting conditions, these measurements were carried under three different lighting conditions created by different combinations of photographic spotlights with diffusers, and neon

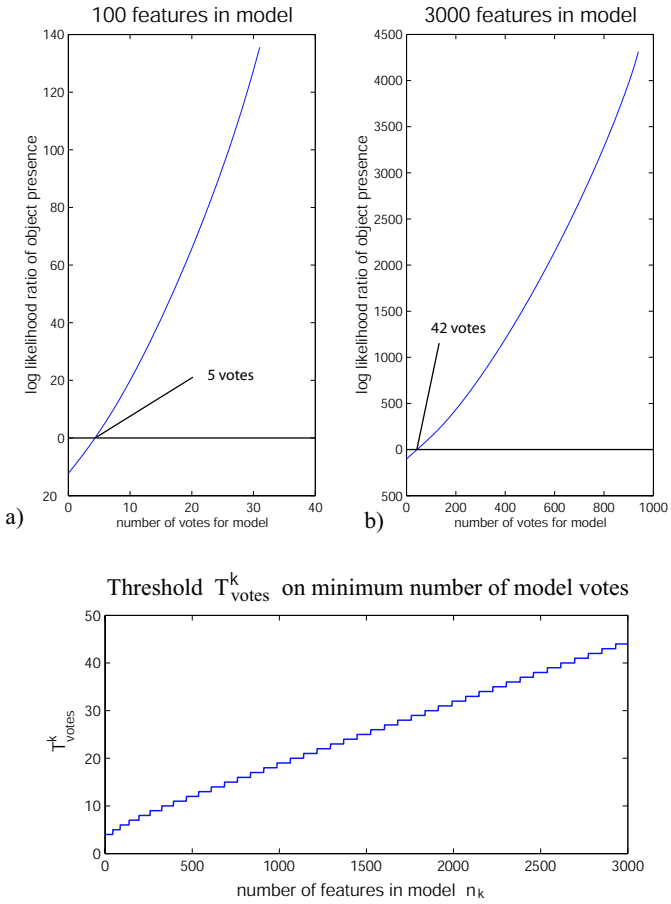


Figure 3.11: (top) log likelihood ratio R^k_{votes} as a function of the number of votes collected by the object under consideration. The black line indicates zero and is only drawn as a visualization aid. (a) 100 database features for the model under consideration. (b) case when the object generates 3000 features. (bottom) Resulting value of T^k_{votes} as a function of the number of features n_k in model. The curve has a staircase appearance because the value of the threshold is always rounded.

lights on the ceiling. Therefore we believe that the results are useful as well when applied to images taken in outdoors environment.

3.6.7 Choice of T^k_{votes} and T^k_{hough}

T^k_{votes} is the minimum number of votes that the known object m_k must collect to be possibly included in a hypothesis. This threshold is chosen based on the ratio (cf., equations (3.12) and

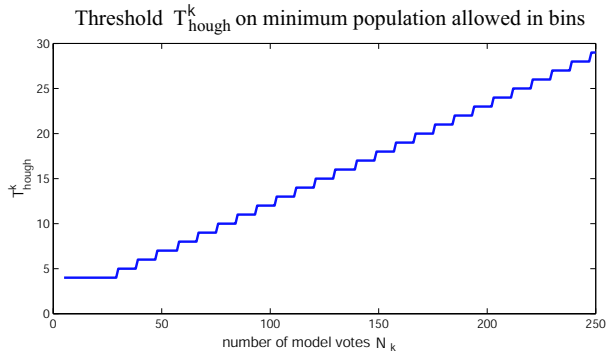


Figure 3.12: Variation of the threshold T^k_{hough} on minimum population of Hough space bin in order to continue to the PROSAC stage, as a function of the number of votes N_k for model k. This curve appears to be approximately linear. The staircase appearance is due to T^k_{hough} being necessarily an integer, so the value of T^k_{hough} is rounded.

(3.13))

$$R^k_{votes} = \frac{P(N_k|H)}{P(N_k|H_0)} \tag{3.34}$$

The variation of $log(R^k_{votes})$ with N_k is represented in figure 3.11 (top panels) for both cases when the object under consideration generates 100 or 3000 features. Note that $log(R^k_{votes})$ is increasing with N_k. This corresponds to the intuitive idea that a higher number of votes for a given object constitute more evidence towards the presence of this object in the image.

T^k_{votes} is chosen as the minimum value of N_k for which $log(R^k_{votes}) > 0$ (see figure 3.11 (bottom)). Note that the threshold is higher when the object generates 3000 features than for 100 features — objects with a large number of database features are expected to generate more detections in the test image, than objects that generated few database features (similarly to [Low01]).

T^k_{hough} is the minimum number of candidates matches that a bin from Hough space must collect to be considered in further stages.

Similarly to equation 3.34, we consider the ratio

$$R^k_{hough} = \frac{P(N^b_k|N_k, m_k \in b)}{P(N^b_k|N_k, H_0}. \tag{3.35}$$

where the notation $m_k \in b$ denotes that the known object is present in the test image and the transformation that takes m_k into its pose in the test image, corresponds to the bin b. This ratio

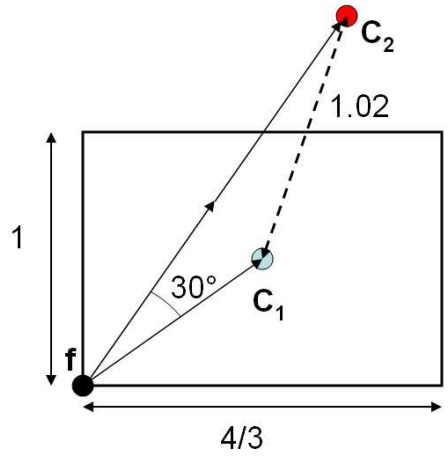


Figure 3.13: Worst-case scenario for the prediction of the location of the image center, due to an inaccurate measurement of feature orientation and scale. The feature under consideration is f. C_1 is the true center of the image. C_2 is the location predicted due to an error of 30° in orientation and a factor 2 in scale.

expands into

$$R^k_{hough} = \frac{\sum_{^0N^b_k + {}^1N^b_k} B({}^0N^b_k | N_k, 1/\mathcal{B}) \cdot B({}^1N^b_k | N_k, 0.48)}{B(N^b_k | N_k, 1/\mathcal{B})} \tag{3.36}$$

A bin is considered in the next stage of the detection process if it verifies $R^k_{hough} > 1$ and has a population of more than 4 candidate matches (as mentioned in Section 3.4.2.3, we require 4 matches to run PROSAC), and rejected in favor of the null hypothesis otherwise. The variation of the resulting T^k_{hough} with N_k is displayed in figure 3.12. As for T^k_{votes}, the value of T^k_{hough} increases with N_k.

3.6.8 Size of bins in Hough transform space

The size of the bins used during the Hough transform stage is a trade-off. If small bins are used, candidate matches are consistent with each other inside each bin, and the precision of the predicted transformation is high. However, clusters might be split by bin boundaries, and each bin collects few matches, i.e., a weak evidence for the predicted transformation. On the other hand, large bins collect strong evidence for the predicted transformation, however these candidate matches are polluted by numerous outliers, and the predicted transformation is not as accurate.

Consistency errors between correct correspondences cause the clusters to be spread in Hough space, even if no outlier is involved. These errors have two main causes:

— Each candidate match predicts a similarity transform between a model and the test image.

This is only an approximation of the true transformation, which can be an affine or even non-rigid transformation.

— Inaccuracies in features measurements inside correct matches. For true matches, the inaccuracy in location is typically no more than a few pixels. On the other hand, the uncertainty on orientation and scale can be large. Lowe describes in [Low04] bins that accommodate 30° inaccuracy in orientation and a factor of 2 in scale. The effect of this inaccuracy in orientation and scale is illustrated in figure 3.13. The worst-case scenario is considered, where feature f is as far as possible from the image center C_1. C_2 is the center predicted due to an error of 30° in orientation and a factor 2 in scale. For a typical image $height/width$ ratio of $3/4$, the uncertainty on the predicted position of the image center, is equal to the whole height of the image.

This last argument suggests that larger bins should be used than those used by Lowe. To confirm this, we ran our full recognition system on the 'Giuseppe toys' data-set (see Section 3.7), using various bin sizes. The ROC curves obtained are displayed in figure 3.14. The legend displays the bin sizes: for x and y, the bin size is a fraction of the model's largest dimension ; for the orientation θ it is indicated in degrees ; for the scale s it is a scale factor allowed in a same bin. The best results were achieved when we allow for an error of 0.5 times the model size in x and y, 60° in orientation and a factor of 2.8 for scale. These parameters were used in further experiments.

Note that if two database models represent the same object imaged successively at low and high resolution, the bins corresponding to these two models have different sizes. This is very reasonable: given the images only with no additional information, one cannot know if the quantity that changed is the image resolution or the size of the physical object.

How many bins do we need to explore? For a database containing 100 objects, the number of bins is on the order of $60,000$. However, the number of non-empty bins is at most equal to the number of features in the test image (typically a few thousands), therefore only a small fraction of the Hough space needs to be explored in order to find all bins that correspond potentially to correct transformations.

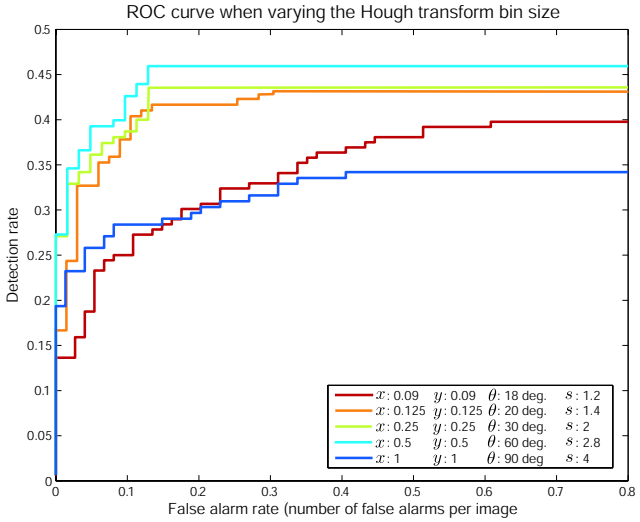


Figure 3.14: Experimental determination of the best bin size for the Hough transform stage

3.6.9 Single vs. multiple Hough table entry for candidate matches

As mentioned in Section 3.4.2.2, one drawback associated to binning the Hough space occurs when clusters of matches are cut by bin boundaries. To alleviate this problem, [Low99] proposes to add extra hash entries for points close to boundaries, by having each candidate match vote for the two closest bins in each dimension. The performance for both the case when each candidate match hashes to a single bin, and when each candidate match hashes to the two closest bins in each dimension, is illustrated in figure 3.15 for the two data-sets described in Section 3.7.1. Adding entries in adjacent bins improves the detection rate slightly, while it increases the false alarm rate. The performance gain is not significant compared to the increase in system complexity — on average three times more bins need to be examined and filtered through with PROSAC. Therefore, we used only single-bin hashing in further experiments.

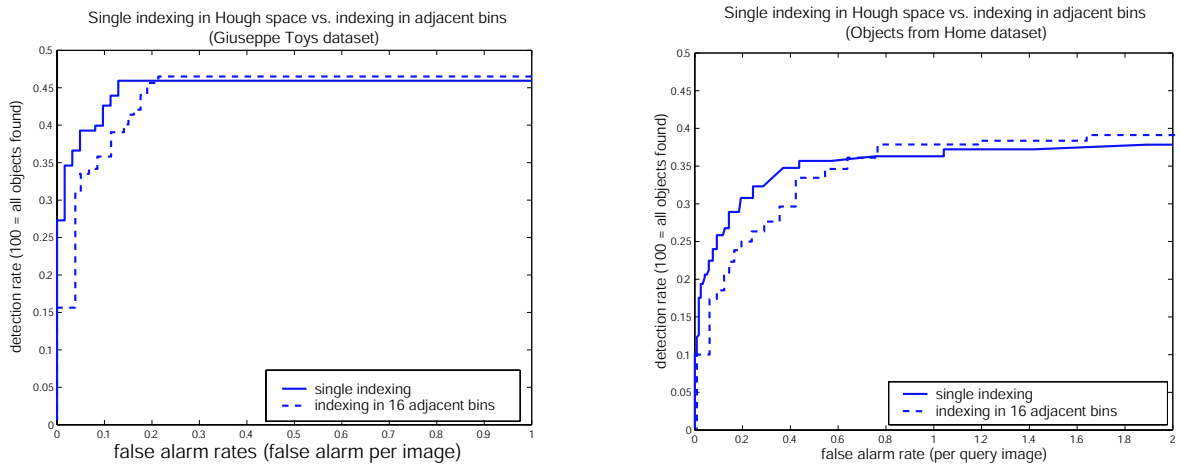


Figure 3.15: Single Hough table entry for each candidate match vs. multiple entries in adjacent bins

3.7 Experimental results

3.7.1 Setting

The recognition method presented above was tested on two data-sets. Both data-sets are available on http://www.vision.caltech.edu/archive.html. The first data-set, labeled 'Giuseppe Toys,' consists of 61 training and 141 test images of toys belonging to a 3-year old child. Most objects are stuffed animals and toy vehicles. The typical image size, both for models and test images, is 800×600 pixels. In most test images, the objects are photographed against a background of grass, which has lots of texture and generates countless clutter detections. The test images consist either of multiple toys thrown together (52 images and 394 objects to be detected), images that contained a single toy (63 images) or dummy images that do not contain any toy (26 images).

The second data-set, labeled 'Objects from home' consists of 49 training and 101 test images of kitchen items and objects of everyday use. Most test images use a background of concrete. The training set contains either one or few images of each object. In the test images, the background is very grainy and generates lots of clutter detections (more than two thirds of the features detections are background detections).

3.7.2 Results

The method presented here was compared against Lowe's voting approach [Low04], which is a state-of-the-art method for detection of individual objects. The implementation of Lowe's system was provided by Evolution Robotics (http://www.evolution.com/products/ersp/). We did not have access to the source code. The output of the software is for each single-object hypothesis: the object identity, a set of transform parameters between object and composition, the number of matches in the considered hypothesis, and a probability score (based on the number of matches, see [Low01]) characterizing the belief that the hypothesis is correct.

Figure 3.18 displays the ROC curves obtained with the two methods, both when the probabilistic score (equation 3.4) is used as threshold for the ROC, and when the threshold is simply the number of matches in the hypothesis. In the latter case, the more matches survive all steps of the matching process, the more we believe in this hypothesis. The thresholds T^k_{votes} and T^k_{hough} are set manually to two fixed values, instead of the adaptive probabilistic thresholds described in Section 3.6.7. The values chosen for T^k_{votes} and T^k_{hough} are the median of the adaptive values from Section 3.6.7.

On the 'Objects from Home' data-set, the detection rates were comparable for both method. Our method is less prone to false alarms, we believe that this is due partly to the efficiency of the PROSAC algorithm at rejecting hypotheses that are inconsistent in terms of pose, and partly to our probabilistic model which checks systematically the appearance and pose consistencies of each candidate match with respect to the hypothesis being tested.

For the 'Giuseppe Toys' data-set, the detection rate of the Evolution Robotics software was lower than our method. In this data-set, most images contain grass, soil, and concrete, which generate a very high number of irrelevant features (several thousand features per test image). The grass features lead to numerous outlier candidate matches, in particular in Hough space bins corresponding to correct objects and poses. These incorrect matches cause the least-squares fit to yield a wrong pose. In our case, although numerous incorrect matches will be included in the Hough transform stage as well, the PROSAC stage is a more robust estimation method than the straight least-squares fit used by Lowe, and outliers are rejected more reliably, leading in particular to a

better detection rate. We will see in Section 3.7.8 that the performance of both recognition systems on cluttered images is in agreement with these conjectures.

Note that in figure 3.18 the ROC for the Evolution Robotics system seems to be lower when using the probabilistic score than when using the number of matches as threshold. In fact, the probabilistic score was either extremely close to zero or extremely close to one, so that the ROC curve consists essentially of only the two points $(0, 0)$ and $(2, 0.15)$. A straight line was drawn to connect these two points but there are no data points along that line, and the performance of the probabilistic scoring is no worse than when using the number of matches in the hypothesis as ROC threshold.

On the ROC curve for the 'Giuseppe Toys' data-set, one can observe that in the regime of high false-alarm rates, the detection rates for both cases when probabilistic scores are used or not used, are very similar. The situation is identical for the 'Home Objects' data-set (the regime with very high false-alarm rates is not showed in figure 3.18). The reason is that in the regime with high false-alarm rate, the only difference is in the adaptive vs. non-adaptive thresholds — some detections are missed by fixing the thresholds to an arbitrary value. The probabilistic scoring process does not create new hypotheses. Its role is to help the decision process 'I have a hypothesis — should it be accepted or rejected?' by rating hypotheses in a principled manner free of user intervention. As a consequence, the probabilistic rating of hypotheses does not improve the system's detection rate, but only the rate of false alarms.

3.7.3 Failure mode

The limitations of our recognition system are twofold. First, many features have poor distinctiveness and index to a wrong model. This is the case, e.g., in figure 3.7-(c), where most objects collect a significant number of votes. Besides, smooth, textureless objects yield very few features and are thus hard to match. In some cases the use of color information in the descriptor would probably be useful in identifying objects present in the composition. Unfortunately, features incorporating color information are usually sensitive to changes in lighting conditions [dWS06].

Secondly, the global affine transformation model used here is only an approximation. The

advantage of modeling a global transformation is that information from the whole image can be used to provide evidence for a transformation. In some cases, a model using only local information [LSP04, Rot04, Sch99] might produce better results, e.g., with highly deformable objects, like cloth or articulated objects.

Figure 3.19 shows an example of the non-detection process that is observed most frequently. Panel (a) shows the test image, (b) the model of the corresponding object. The large blue part is flexible, its structure changes completely between both shots, which makes the task difficult for any detection system based on pose. On the other hand, the white handle and the colored bells are rigid parts which should be matched reliably between views. However, their smooth surface triggers only very few features detections with poor localization. Most features detected in this picture are generated by the grainy background of concrete, which create random candidate matches with all models as can be seen in panel (c). The textureless and specular object surface make the orientation and scale information in the SIFT descriptor unreliable. As a result, the candidate matches are spread in the whole Hough space, and the Hough transform stage identifies only one cluster with enough matches to continue to the next detection stage. This cluster is in fact incorrect, and rejected by the PROSAC stage. One might object that the values of the thresholds T^k_{votes} and T^k_{hough} might be set too high and lead to false rejections, however the same phenomenon was observed when these thresholds are manually tuned down to a minimum value of 4.

Similarly, almost all failures to detect objects were caused by insufficient distinctiveness of the descriptors and incorrect characterization of orientation and scale. This leads to candidate matches indexing to wrong objects, and too much spread in Hough space.

3.7.4 Reduction of number of hypotheses with the coarse-to-fine process

The evolution of the number of possible hypotheses as the coarse-to-fine process is performed, is illustrated in figure 3.20 for the 'Giuseppe Toys' data-set. Note that the vertical axis is a logarithmic scale. For the first three stages (prior knowledge, model voting, Hough transform), the number of remaining hypotheses is displayed as a number of bins in Hough space (see Sections 3.4.2.2 and 3.6.8). In other terms, these hypotheses can be considered as *coarse* hypotheses. After the

PROSAC stage, each hypothesis is associated to a detailed set of transformation parameters and to an assignment vector, it becomes a *fine* hypothesis.

One can observe that the model voting stage has only a small effect on the pruning process. We still keep this stage, as its computational cost is extremely low. The most effective stages are the Hough transform stage and the PROSAC outlier rejection process, which reduce the number of hypotheses by more than an order of magnitude.

The right panel in figure 3.20 displays computation times for each step. We used a Matlab implementation on a Pentium4 running at 3 GHz. The slowest step of our method is the PROSAC stage, in particular in cases when many bins contain a large number of votes. It makes sense to use PROSAC as the last stage of the recognition process, once a large number of branches of the hypotheses tree have been pruned. Lowe's system is faster (several frames per second with an optimized implementation in C), as his system is simpler with only a Hough transform and a least-squares fit in each bin.

3.7.5 Influence of training set used to learn the foreground appearance density

We investigated how performance is affected by the choice of the covariance matrix $C_{fg,\mathcal{A}}$ used in the density $p_{fg,\mathcal{A}}$. This matrix defines a norm used in appearance space to characterize the difference between two instances of the same feature. We compared the performance obtained with three different choices for this norm. The simplest choice is the identity matrix, which corresponds to the Euclidean distance in appearance space. A second, more elaborate choice, consists of using the covariance of features appearance differences from flat objects (Section 3.7.6). In this case, ground truth correspondences between pairs of views can be easily identified since the object transformation is a homography: one only needs to define four point correspondences to compute the object transformation [ea05c, MS05]. The third choice uses the covariance matrix obtained with the ground truth matches described in Section 3.6.6.

Figure 3.21 compares the results obtained with these three covariance matrices, on the 'Giuseppe Toys' data-set (Section 3.7.1). The best results were obtained with the 3D objects, the covariance

matrix obtained from these images expresses better the features variations in real-world images than with the identity matrix or flat objects. Note that the 3D objects are a different set from the 'Giuseppe Toys,' in order to avoid contaminating the experiment.

3.7.6 Performance on objects with text and graphics

In order to obtain a more fine-grained evaluation, we compared both systems on smaller data-sets targeted at specific environments. First, we investigated performance on a smaller data-set of objects with printed text and graphics (see figure 3.22). Seven objects were used for training; the testing set contained 33 test images of compositions with several objects, simpler images with a single object, and clutter images with no known object.

The ROC curve showing the performance of both systems is displayed in figure 3.22-(right). Both the Evolution Robotics system and ours performed extremely well on this data-set. The text and graphics on the objects have a high contrast, and generate features at the same physical locations on different views of a same object. Besides, the uniqueness of each pattern leads to SIFT features with good distinctiveness. These conditions yield excellent results for the appearance and pose matching process of both our algorithm and Lowe's.

3.7.7 Performance on textureless objects

Second, we tested both systems on a small data-set containing objects with very little texture and with shiny surfaces. Six objects were used for training; the test set contained 36 images of test images with multiple instances of these objects, images with one object, and clutter images with no known object.

These textureless objects generate very few features, and the location of these features varies substantially between different views of a same object. Regarding appearance, the local appearance on homogenous surfaces varies very little with the location, which leads to features with poor distinctiveness. This creates extremely difficult conditions for both our system and Lowe's, and the performance was very poor, as can be seen with the ROC curves in figure 3.23-(right).

3.7.8 Performance on cluttered images

Third, both systems were compared in a 'clutter-only' environment. We collected images of 39 different textures, with two images per texture — one to be used as a training image, the other as test image. The training and test images were taken at separate locations, so that in theory no match should be identified between the training set and the testing set. In other words, in these conditions any detection is a false alarm. The similarity of texture and features appearance between training and test images might lead to confusion and makes this setup a challenging environment. However, pose consistency, used both by the ERSP system and ours, should be able to solve this confusion.

On this data-set our system performed significantly better (see table 3.24-(d/left)). Lowe's method lead to 111 false alarms, versus 14 false alarms for our system. (Since there is no detection to be made, we cannot generate ROC curves for this experiment). Of these hypotheses, 63 false alarm were high-confidence hypotheses with more than 30 matches in Lowe's case, while our system had only 3 hypotheses with more than 30 matches. Besides, in 11 cases Lowe's system matched the test image to a training image with an unrelated texture — this happened only 4 times with our system.

Last, both systems were compared on this same test, but with the training set consisting of the 'Home Objects' database. This is an easier task as the same texture is never present both in models and in the test images. Again, our system obtained significantly fewer false alarms than ERSP's, with 12 false alarms for our system versus 30 for ERSP's (Table 3.24-d / right).

3.8 Conclusion

We presented a consistent probabilistic framework for individual object recognition. The search for the best interpretation of a given image is performed with a coarse-to-fine strategy. The early stages take into account only global counting variables that are inexpensive to compute. We benefit here from Kd-tree search and Hough transform, which result in first estimates of the objects likely contained in the test image and their pose. A large fraction of irrelevant hypotheses are discarded at a very low computational cost. Further steps refine the hypotheses and specify individual features

assignments. The pose consistency is efficiently enforced by the PROSAC estimator. The search procedure results in a small set of hypotheses whose probability is computed. The use of our probabilistic model allows us to further reduce the rate of false alarms. Besides, the conditional densities used here are estimated using extensive measurements on ground truth matches between images from real 3D objects.

We tested this recognition method against a state-of-the-art system on multiple data-sets. Our method performed consistently better than Lowe's, but the performance was especially encouraging when the test images contained lots of clutter and texture, a frequent source of confusion for recognition systems.

Figure 3.16: Samples from 'Giuseppe Toys' data-set. Detections from our system are overlaid in green and yellow on the test images. The yellow boxes denote objects identified by our system but missed by ERSP's, while the green boxes denote objects identified by both systems. The child, present in some test images, was not part of the training set.

Figure 3.17: Samples from 'Home Objects' data-set. Detections from our system are overlaid in green on the test images. The yellow boxes denote objects identified by our system but missed by ERSP's, while the green boxes denote objects identified by both systems.

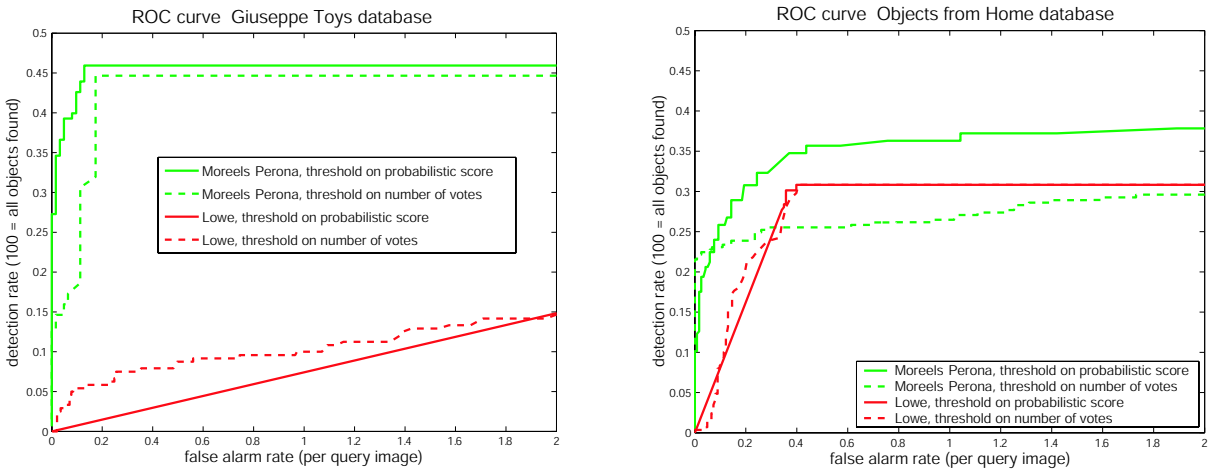


Figure 3.18: ROCs for the 'Giuseppe Toys' and the 'Home Objects' data-sets. The performance of our system is shown in green and compared with the performance of Lowe's system as implemented by Evolution Robotics (in red). The figure displays the ROCs obtained when using the probabilistic hypothesis score (equation 3.4) as threshold (solid line) and when using the number of foreground matches in the hypothesis (dashed lines). Data-sets available from *www.vision.caltech.edu/archive.html*.

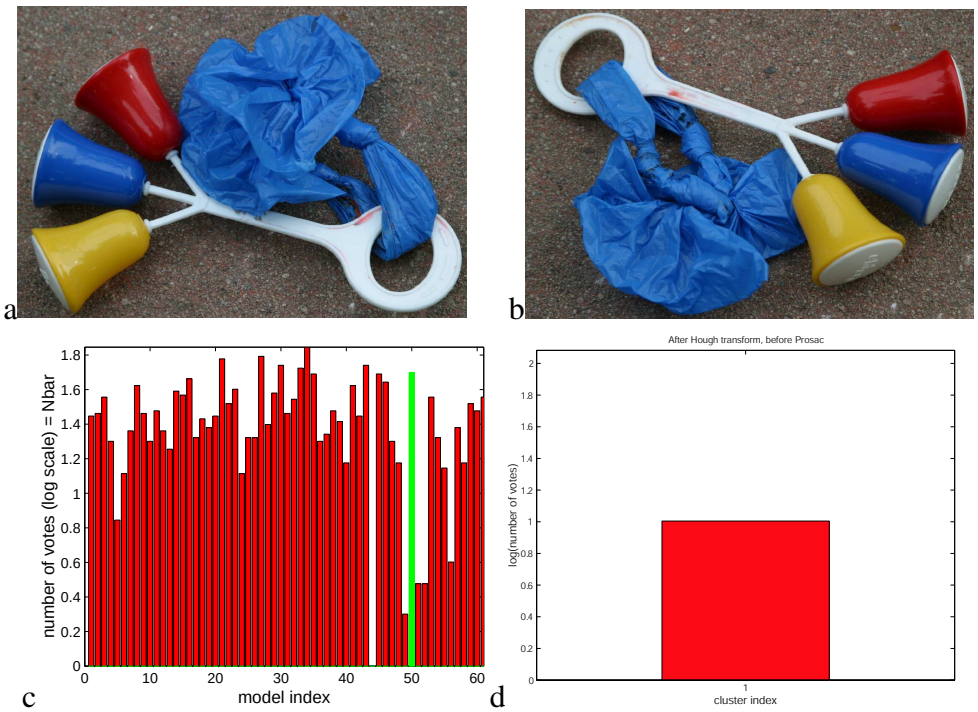


Figure 3.19: Example of failure to recognize. (a) Test image. (b) Model of corresponding known object. (c) Candidate matches are spread among most models, indicating insufficient descriptor distinctiveness. (d) The pose information from candidate matches is excessively spread through the Hough space, only one (incorrect) bin collects more than 4 candidate matches.

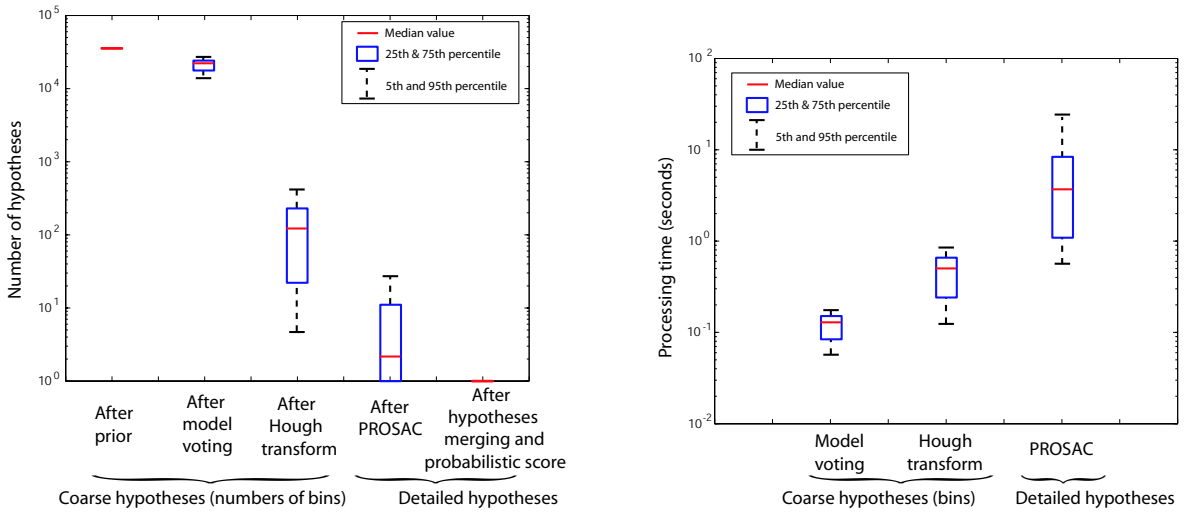


Figure 3.20: (left panel) Number of relevant hypotheses after each stage of the coarse-to-fine process. Percentiles are displayed when applicable. (right panel) Computation time required by each stage of the recognition process. Both panels are for the 'Giuseppe Toys' data-set.

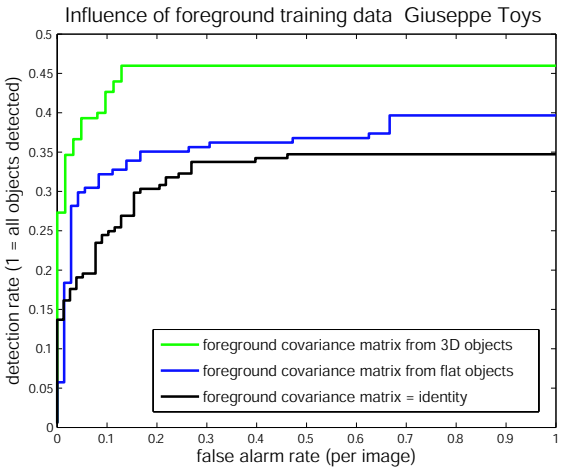


Figure 3.21: Influence on performance of the foreground appearance covariance matrix

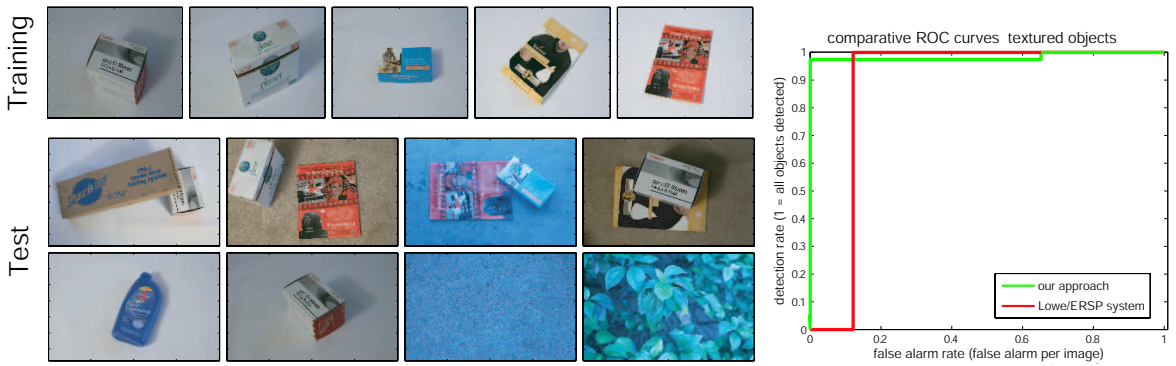


Figure 3.22: Evaluation on objects with text and graphics

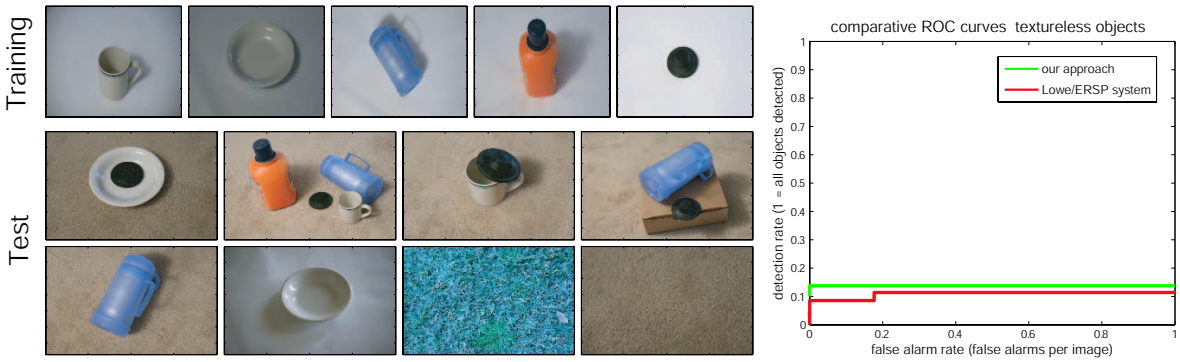
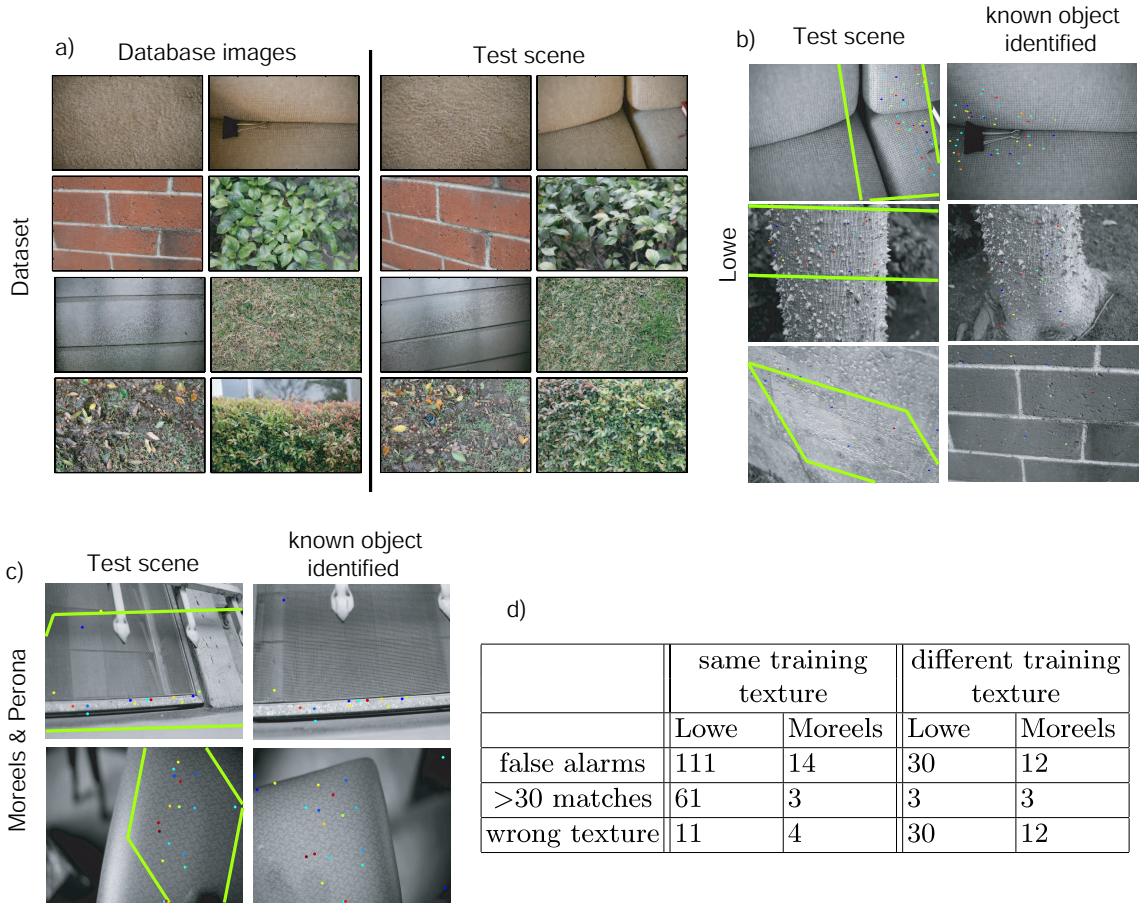


Figure 3.23: Evaluation on texture-less objects



	same training texture		different training texture	
	Lowe	Moreels	Lowe	Moreels
false alarms	111	14	30	12
>30 matches	61	3	3	3
wrong texture	11	4	30	12

Figure 3.24: Comparison on cluttered images: (a) Some examples of images from training and test set. b) Examples of false alarms identified by the ERSP system. The outline of the prediction of the object pose in the composition according to the hypothesis is displayed in green. Candidate features matches identified by the hypothesis are displayed in color. (c) Examples of false alarms for our system. (d) Performance comparison summary for both testing conditions when the training set contains the same textures as the test set and when the training set consists of the 'Home Objects' images.

Chapter 4

Face identity

4.1 Abstract

How do we identify images of the same person in photo albums? How can we find images of a particular celebrity via Google Image searches? Both of these tasks require solving numerous challenging issues in computer vision. Among them are (1) finding the location of individuals in images, (2) maintaining robustness to variability in pose, image quality, lighting, occlusion, and scale, and (3) using an appropriate distance metric in order to compare the detected individuals. Many of these issues have been worked on in isolation within the computer vision community, however there exist few systems which combine all components together into a complete system capable of being effective when confronted with the variability of *real images*. In this work we aim to create a visual recognition system, that, given a target image, is able to find all other images of that individual within a typical photo collection or web search. Each individual is represented by a feature vector composed of extracted patches around automatically detected facial key-points. We use a training set of over 1000 images to generate a distance metric which combines and weighs different components of the feature vector to drastically increase recall performance. We analyze different components of our system to provide insight into such as issues as: Which facial features are most important for recognition? What representation for facial features is most useful? Finally, we demonstrate our system on a large image-set of 99 individuals which we collected from the web and show impressive ability to **automatically** detect images of the same person.

4.2 Introduction

The most common image searches on the web are celebrity pictures. This is, at the moment, implemented by searching keywords, and it is thus susceptible to errors: many images containing our favorite heroes are missed because they are not properly indexed. If we could search images for known faces, no celebrity picture would be missed any longer. This technology could, more in general, be useful for (1) searching the Web for an image of a particular indivudal (e.g., searching for Brad Pitt on Google Image search), and (2) finding all images of an individual within a personal photo collection (e.g., asking for all images of Grandpa in your personal photo collection or

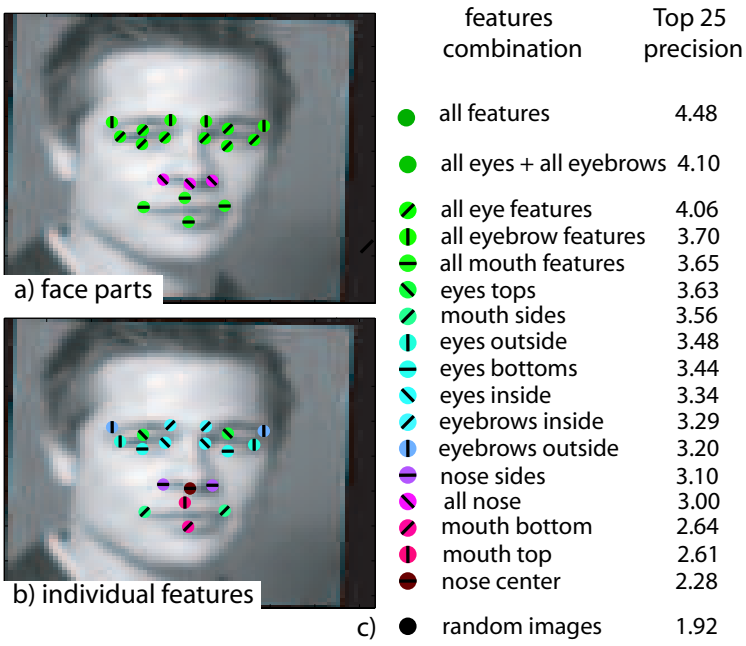


Figure 4.1: Which facial key-points are most useful? We rank the ability of different facial key-points *by themselves* to recall images of the same person within a large image data-set (see Section 4.6.1 for a more thorough description of the performance metric) (a) The performance of different features on recall experiments. Corresponding number and color-code in (c) indicates the precision within the top 25 images, higher number is better. Recall performed only with: eyebrows structure, eyes structure, nose structure, and mouth structure. (b) Performance when a face is characterized by a single individual patch (two in case of symmetry, e.g., both sides of the mouth are included together). The same color scale is used for (a) and (b). (c) Scores of parts and individual features. The set consisting of all parts performs best, followed by the eyes and eyebrows structures. Overall all features related to the eyes and eyebrows perform well. 'Random images' is the baseline method that draws randomly 25 images and indicates the precision. This experiment used a set of images of 20 individuals, with 10 images per individual. We used 7×7 patches and raw intensity values, the ranking did not change significantly when using 13×13 patches and image gradients. Note that we used an L1 distance to measure similarity and did not apply our learned distance metric for this comparison.

MySpace pages).

There has been significant previous work involving various aspects of facial recognition and detection, and face recognition is one of the most studied fields in computer vision. Many algorithms exist for detecting the location of a face in images including Rowley et al. [RBK98], Schneiderman et al. [SK00], and the popular Viola and Jones [VJ01] detector (which is both fast and reliable). Once the face is detected, a feature representation must be created in order to compare different

faces. Both global, in which the entire face is segmented and made into a feature vector, and local, in which only specific key-points are used to create a feature vector, have been suggested. Global approaches, such as the well-known Eigenface [TP91] technique, which maps global representations for a face onto a eigenbasis, tend to suffer from slight variations in alignment which can cause large differences in feature representation. Local representations tend to have more success and have been used by numerous authors including [EZ06]. In particular, the Everingham facial feature detector [EZ06], which was trained on a large data-base of facial features seems to work very well in practice.

Although there has been much previous work on facial representations and recognition, most of the work has focused on controlled environment, well-segmented and/or aligned images of faces. This includes the CMU Pie data-set and the Yale face data-set. There has been relatively little work on faces in real scenes, although there are exceptions, most notably [SSZ04, ESZ06] who work on finding images of actors in scenes and [ea04a] who use both text and images to automatically associate names to faces from news articles.

Furthermore, there is existing work on learning distance metrics and/or mapping functions to place facial features in appropriate spaces to perform recall on. The most prominent is the work using FisherFaces [BHK97] which uses a combination of Eigenfaces and Linear Discriminant analysis to learn a linear mapping.

Finally, the psychological literature has made numerous interesting contributions regarding the cues which humans use to recognize faces. See [ea06b] for an excellent review, which, among other observations, states that the eye-regions of faces are very useful for facial recognition.

In this work, we utilize and extend some of this existing work, including the Viola-Jones [VJ01] face-finder and the Everingham facial feature finder [ESZ06], and combine it with novel methods for learning distance metrics in face space, to create a complete system capable of accurately retrieving images of the same person in a challenging data-set of 99 individuals obtained from the Web. We report performance results on numerous different tasks and compare our performance to other existing methods.

Section 4.3 describes the general algorithm we employ. In Section 4.4 we describe the face detector, the facial feature finder, and facial feature representations used. In Section 4.6.1 we show

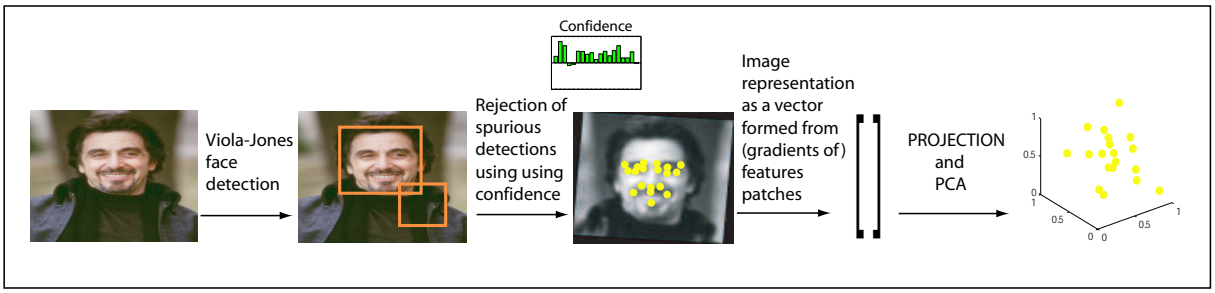


Figure 4.2: From photo album to space of representation — cascade of steps

how to create an effective distance metric. Section 4.8 shows and compares results of our complete system. We conclude in Section 4.9.

4.3 Overview

Figure 4.2 gives a schematic of our facial recognition system. First we find the face using Viola and Jones. Next we remove spurious detections and extract a feature representation. Finally we project the face into a new space using a learned mapping function. The resulting feature representations should reflect facial identity, i.e., images of the same person will be closer to one another than images of different people.

4.3.1 Performance metrics

Here we introduce our 3 performance metrics which mimic typical facial recognition tasks. The three performance tasks which we consider are:

1. Given a target image of a particular identity, how often is the nearest neighbor to this target, actually the same individual? This task would be useful in such applications as finding the most similar-looking celebrity to a person. The rank of the best performing image of the target individual among nearest neighbors, is termed *Best Rank Distance*.

2. Given a target individual, our goal is to find K images, among which images of the target individual are as frequent as possible. Think of, for instance, a Google Image Search in

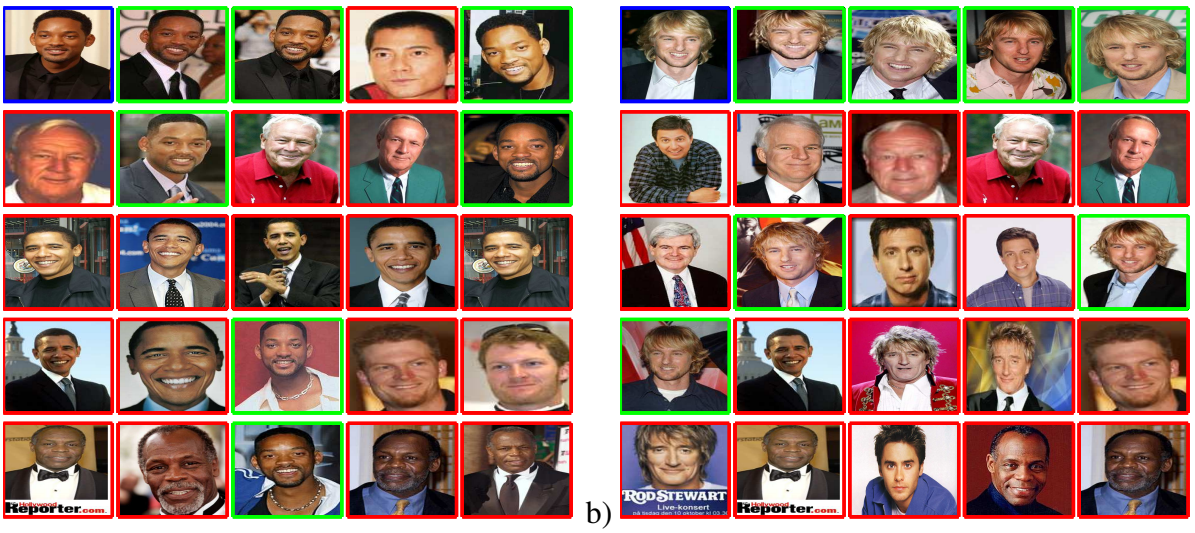


Figure 4.3: Two examples for the 'Top 25 Precision' retrieval task. We queried for Will Smith (a) and for Owen Wilson (b) — the query images are showed in the top left with a blue outline. In both cases the query returned 7 correct results (green outline) out of the 25 closest matches after projection and PCA. It is important to remember that the category 'Will Smith' and the category 'Owen Wilson' contain ONLY 10 EXAMPLES each, therefore a perfect answer would return 10 samples from the target category out of 25 results. This is to contrast with the 'Google-images' engine, where the target category typically contains hundreds of images for each celebrity.

which we would desire a high number of the target individual within the top 25 returned images. This is the detection performance with a recall of 25, we term it *Top 25 Precision*. Note that in our experiments we limit the number of images of each single individual to 10, such that the highest value achievable by *Top 25 Precision* would be 9.

3. Given a query image of an individual, the final metric is the mean of the rank distances to all other images of the same individual. We normalize this metric by the total number of images and call it *Average Rank Distance*. It measures how tight the cluster of images of the individual of interest is, when compared to the whole set of images.

These metrics will be used in further experiments to evaluate the performance of different representations and learned distance functions.

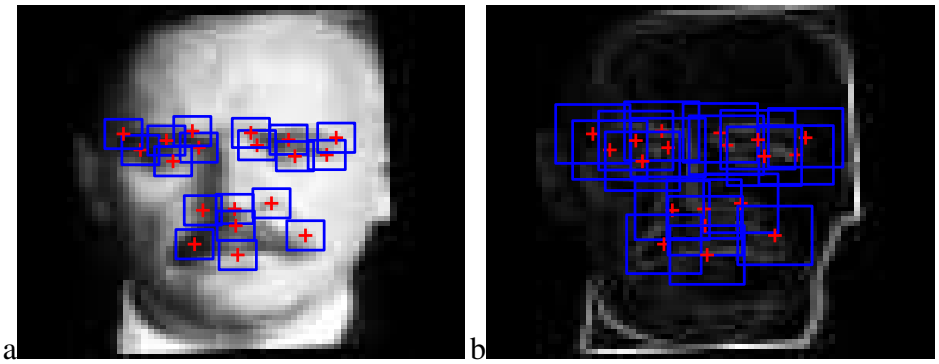


Figure 4.4: Example of the relative size of each patch. (a) Examples of 7×7 patches on a raw intensity figure (this yielded the optimal performance for raw patch representations, see figure 4.6). (b) Example of gradient image and the size of extracted patches, 13×13 patches performed best when using the gradient as a feature.

Image Set	Total #img.	VJ Mult. det.	False al.
99 Celebs	1066	68	4
BG Celebs	200	15	0

Table 4.1: Table showing the size of our data-sets as well as the performance of the Viola and Jones detections. (First Column) The two data-sets, both a set of 99 individual celebrities downloaded from the web and a set of 200 other images also downloaded from the web. (Second Column) The total number of images in each data-set. (Third Column) Total number of images for which the Viola and Jones algorithm generated multiple detections. (Last Column) Number of remaining false alarms after heuristic based to remove spurious detections (see Section 4.4).

4.4 Feature extraction and representation

In order to detect faces we use the OpenCV [Int] implementation of the Viola and Jones [VJ01] face detector. The output from the face detector is a set of bounding boxes which identify the position of the face in the image. Most images from the '99-Celebrities' data-set (see Section 4.5) yield a single detection, however in 68 cases (out of 1266) it mistakenly finds two or more faces.

Next, these bounding boxes are used as input for the impressive Everingham facial-feature detectors [ESZ06]. This detector identifies the position of 19 features facial features as well as the confidence it has with these detections. The facial-feature identified are shown in figures 4.4–4.5.

We used the following heuristic in order to discard spurious face detections: let CF represent the 19-dimension confidence vector for a face, $C^+ = max(CF, 0)$ its positive component and

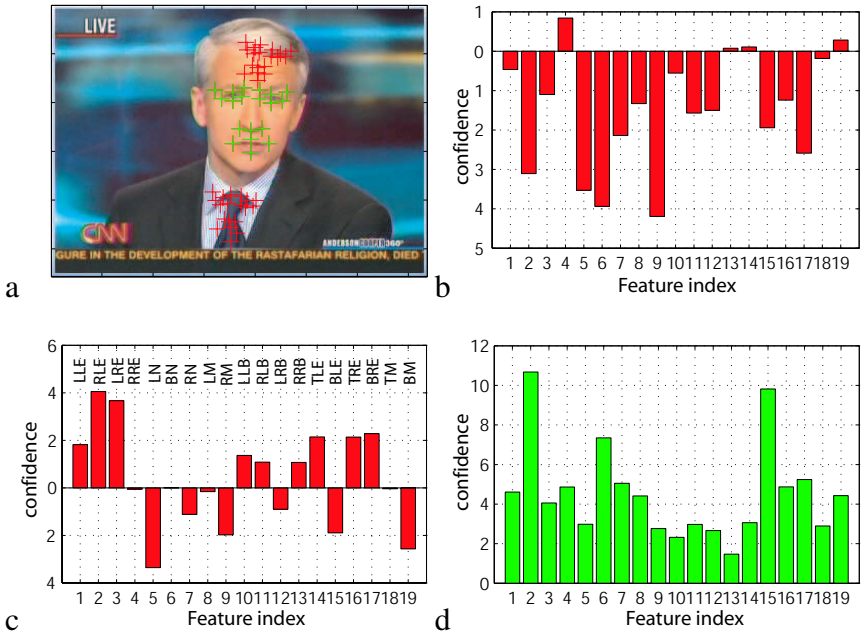


Figure 4.5: Example of Viola and Jones detection and Everingham feature detectors. (a) The Viola-Jones face detector found 3 'faces' in the image. (b)–(d) confidences associated to each feature from each of the 3 detections. The following abbreviations are used: L=left, R=right, T=top, B=bottom, B=eyebrow, E=eye, N=nose, M=mouth. Using our heuristic based on parts' confidence, the incorrect detections are rejected (in red — the detection on the forehead corresponds to confidence scores in (b), the detection on the tie corresponds to confidence scores in (c)). The correct face is accepted (shown in green).

$C^- = -min(CF, 0)$ its negative component. We accept a face detection if $\sum_{k=1}^{1} 9C^+ > 4 * \sum_{k=1}^{1} 9C^-$, and reject it otherwise. Using this method, we successfully rejected all but 4 false face detections, and introduced only 4 new false rejects. Figure 4.5 shows an example of successful rejection of spurious matches on an image that generated 3 detections.

4.4.1 Facial feature representation and size

For each detection accepted by the previous steps, we normalize the Viola-Jones bounding box to a fixed size of 80×80 pixels. We rectify variations in orientation by aligning the eye corners to a same position for all images. We characterize each feature in a face by a patch of variable size extracted around the feature location. The set of patches describing the facial features detected in each face are converted to a vector $\vec{x}_i$ corresponding to image i.

The choice of the size of the patches for feature representation is a trade-off. If patches are

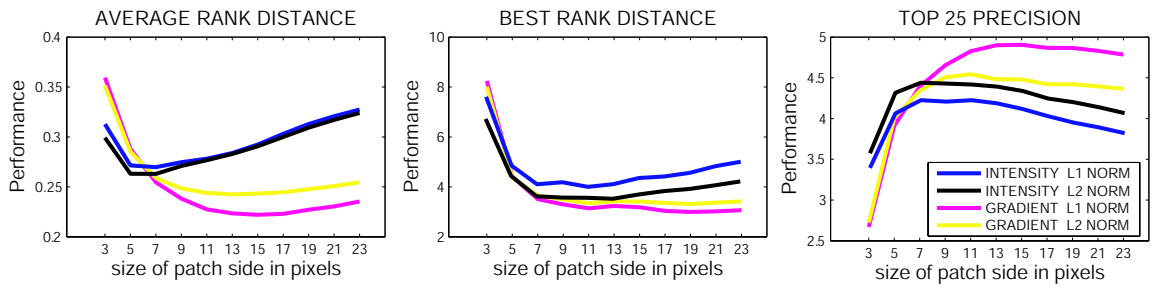


Figure 4.6: Variation of our three performance criteria with size of the patches used for face representation. We plot results for a simplified experiment for which no projection was performed, L_1 and L_2 distances were used on the raw patches. We display results of the simplified experiment both using image intensities and image gradients. X-axis: the size of patches extracted. Y-axis: performance using various metrics. All experiments averaged over the set of 99 Celebrities (see Section 4.8 for more details). (Left) Comparison using *Average Rank Distance* metric. (Center) Comparison using *Best Rank Distance*. (Right) Comparison using *Top 25 Precision*. The best performance is consistently obtained when using 7×7 patches in the case of raw intensity, and 13×13 patches when using image gradients.

too small, the representation is sensitive to artifacts in the image, and not discriminative enough as the patch fails to capture enough of the local texture around the feature of interest. Conversely, if the patches are too large, features include too much detail specific to a particular image and have poor generalization properties. In this section, we investigate the influence of the patch size on the recognition performance.

For this section and Section 4.4.2 only, for the sake of speed we did not optimize the system with respect to a distance metric Φ nor reduce dimensionality with PCA. Rather, we used the raw L_1 and L_2 distances between patches. As a consequence, the performance reported in this experiment is lower than the results in Section 4.8. Figure 4.6 describes the results of our experiments. The performance of patch sizes from 3×3 pixels up to 23×23 was computed for the three score measures defined in Section 4.3.1. We computed the score with L_1 and L_2 distances, both when sampling patches from the raw intensity image and when sampling them from the gradient image. Overall, the best patch sizes when using raw image intensity were 7×7 and 9×9 pixels, while larger patches performed better when using gradients (13×13 and 15×15 performed best). Note that these patches are always extracted after the image has been rectified for orientation and resampled to 80×80 pixels. The overall best performance was obtained with the L_1 norm and gradient values. The superior performance of gradient images is no surprise — our images show a

huge variability in lighting conditions and gradient images show some invariance with respect to lighting.

4.4.2 Evaluation of face subparts

One question that arises naturally is: Which features in the face are most important for recognizing a specific person? In an attempt to answer this question, we investigated the performance of various subparts of the face on a simplified experiment (see figure 4.1).

Features were extracted using 7×7 patches and intensity values. Faces were characterized by various combinations of features. One experiment focused on the sets of features that form face parts: eyebrows, eyes, nose and mouth. The other experiment focused on individual features: outer corner of eyebrows, inner corner of eyebrows, top of the eyes, outer corner of eyes, inner corner of eyes, bottom of the eyes, sides of the nose, tip of the nose, top of the mouth, sides of the mouth and bottom of the mouth. For features which occur symmetrically on both sides of the face, both left and right feature were included together. For example, the left side of the mouth and the right side of the mouth were included together.

Figure 4.1 shows the performance for the 'Top 25 detection' (see Section 4.3.1) criterion, color-coded by decreasing performance. (a) and (c) indicate that, as expected, face structures perform better than individual features. In particular, the most successful face structures are the eyebrows and the eyes. This was the case both when considering face parts and when considering individual features. Interestingly, this is consistent with the human performance results from [ea06b], where Sinha reports that eyebrows are among the most important features for the human face recognition task.

4.5 Data-sets

In creating our data-sets we attempted to mimic the actual statistics and variability encountered in real-world images, contrary to more controlled data-set such as the CMU PIE face data-set or the Yale Face data-set. In particular our faces contained the typical variability found on the Web: large variations in lighting, not aligned, varying resolution (our resolution varied from about 100×130

to about 500×800). One caveat is that we attempted to limit the amount of pose variability to roughly frontal images as can be seen in figure 4.3.

We collected a data-set of 99 individuals from the Web which were mostly celebrities, as they tended to have a large number of images available. Hence we call this data-set the 99-Celebrity data-set. We ensured that each individual had a minimum of 10 images (although some had as many as 16). We also collected a background set of 200 images which were used to assess the performance of our system (see Table 4.1) for a description of the size of the data-sets. Figure 4.3 shows the typical variability of our faces.

4.6 Learning a distance metric

In the previous section we have described how to automatically detect a face and extract features of the face. We now take an additional step in suggesting that each component of our feature vector should not be weighted equally, i.e., certain parts of the face may be more important for recognition than others (figure 4.1 suggests this is a reasonable intuition, as different facial features are better or worse for recognition). The natural question arises, how do we weight these features? We proceed by *learning* a metric which increases performance on recognition tasks.

4.6.1 The Relative Rank Distance Metric

Our goal is to learn a distance metric between any two images such that images of the same person are deemed close to one another, while images of different people are farther from one another. More formally consider each celebrity indexed by c and all feature representations $\vec{x}_i^c$ for an image i in class c. The following cost function follows naturally from the criteria just mentioned:

$$\mathcal{C} = \sum_c \sum_{i,j \in c} \sum_{k \notin c} \text{Dist}(\vec{x}_i, \vec{x}_j) - \text{Dist}(\vec{x}_i, \vec{x}_k). \tag{4.1}$$

Qualitatively, if $\mathcal{C}$ is less than zero then we are doing well on average.

Let us consider the distance $\text{Dist}(\vec{x}_i, \vec{x}_j)$. Now we assume that we can learn some arbitrary mapping function $\phi(\vec{x}_i)$ which projects each feature vector from $\mathcal{R}^P \rightarrow \mathcal{R}^Q$. The we can re-

write our distance function using the kernel, Dist$(\phi(\vec{x}_i), \phi(\vec{x}_j))$. The L_1 and L_2 distances between mapped feature vectors are written as follows:

$$L_1 : Dist(\phi(\vec{x}_i), \phi(\vec{x}_j)) = \sum_{r=1}^{Q} \left|\phi(\vec{x}_i) - \phi(\vec{x}_j)\right| \tag{4.2}$$

$$L_2 : Dist(\phi(\vec{x}_i), \phi(\vec{x}_j)) = \sqrt{\sum_{r=1}^{Q} \left(\phi(\vec{x}_i)\phi(\vec{x}_j)\right)^2}. \tag{4.3}$$

The above mapping ϕ can be an arbitrary function. However, if we consider ϕ to be a linear function (i.e., a matrix Φ) and we let $M = \Phi\Phi^t$, then we can re-write the squared L_2 distance as:

$$L_2 : Dist(\phi(\vec{x}_i), \phi(\vec{x}_j)) = (\vec{x}_i - \vec{x}_j)^t M (\vec{x}_i - \vec{x}_j). \tag{4.4}$$

Now consider again equation 4.1. We would like to minimize this function. We proceed by using the conjugate gradient method, which requires the derivatives of the cost function w.r.t. Φ for the L_1 and L_2 distances.

For the L_2 distance, we consider the simpler task of computing derivatives of the squared cost function w.r.t. $M = \Phi\Phi^t$:

$$\frac{\partial}{\partial M}(Dist^2_{L_2}(\phi(\vec{x}_i), \phi(\vec{x}_j))) \tag{4.5}$$

$$= \frac{\partial}{\partial M}\left((\vec{x}_i - \vec{x}_j)^t M (\vec{x}_i - \vec{x}_j)\right) \tag{4.6}$$

$$= (\vec{x}_i - \vec{x}_j)^t M (\vec{x}_i - \vec{x}_j) \tag{4.7}$$

(note that this is a matrix of size ??)

For the L_1 distance, the component (p_0, q_0) of the derivative is

$$\frac{\partial}{\partial \Phi_{p_0 q_0}} \sum_p \left| \sum_q \Phi_{pq} a_q \right| = \frac{\partial}{\partial \Phi_{p_0 q_0}} \left| \sum_q \Phi_{p_0 q} a_q \right| \tag{4.8}$$

$$= a_{q_0} \cdot sign(\sum_q \Phi_{p_0 q} a_q) = a_{q_0} \cdot sign((Ma)_{p_0}) \tag{4.9}$$

where we used the simplifying notation $a = x_i - x_j$. This can be written as the product of two

vectors:

$$\frac{\partial}{\partial M}\big(Dist_{L_1}(Mx_i - Mx_j)\big) = \begin{pmatrix} sign(y_1) \\ . \\ . \\ . \\ sign(y_Q) \end{pmatrix} (x_i - x_j)^t \tag{4.10}$$

where $y = M(x_i - x_j) = Ma$. Note that there is a measure zero set when the derivative is undefined, namely when $\Phi(vecx_i) = \Phi(\vec{x}_j)$. If all feature vectors $\vec{x}$ are unique this can only be satisfied when Φ is not full rank. We never encountered this situation in practice.

Finally consider again equation 4.1. Depending on the task at hand we may want to change how much we penalize the difference, $\mathcal{U} = \text{Dist}(\vec{x}_i, \vec{x}_j) - \text{Dist}(\vec{x}_i, \vec{x}_k)$, when it is violated. For instance if we are interested in the *Closest Rank Image* we may not want to penalize heavily inequalities which result in large positive values of U, while, on the contrary, if we are interested in the *Average Rank Distance* we would want to penalize large positive values of U. We can include a non-linearity into our cost function with this desired behavior:

$$\mathcal{C}(x) = e^{\frac{x}{\alpha}} \tag{4.11}$$

Increasing the value of α reduces the influence of large positive values of U while decreasing α increases the influence of large U values. We ran experiments using various values of α, shown in figure 4.7.

The cost function is optimized using the conjugate gradient algorithm and usually converges after 100 iterations. We restart the algorithm multiple times ($3\times$) to avoid local minima.

4.6.2 Creating triplet distances

Now consider again equation 4.1 and the process of optimizing the function. Which images do we assign as being close and which images are farther from one another? Consider a celebrity c and all images i of this celebrity. We enforce that images of the same celebrity, j are always closer to

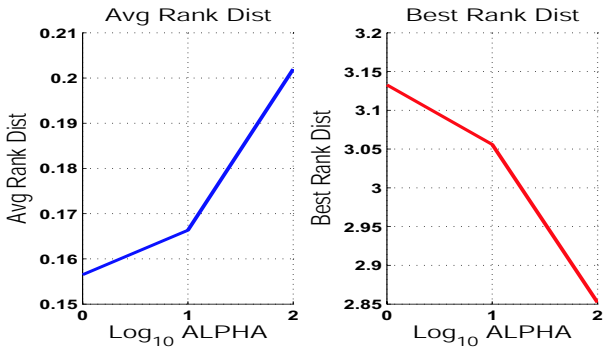


Figure 4.7: The effect of varying the steepness of the slope for our exponential cost function $\mathcal{C}$ on two different performance metrics. (Left) Effects on Average Rank Distance Metric. Note that decrease in performance as we increase α. (Right) Effects on Best Rank Distance. Note that performance *increases* as we increase α. Increasing α results in the optimization giving equal weight to incorrect relative distances which are very far from one another and very close to one another. This intuition is consistent with the results shown in the plot. Results shown from using an L1 metric on 7×7 extracted intensity patches.

one another than to images of other celebrities which are indexed by k. By enforcing this criteria we generate a large list of 'triplets' of the form $[ijk]$ which specify that image $\vec{x}_i$ and $\vec{x}_j$ should be closer to one another than image $\vec{x}_i$ and $\vec{x}_k$. This list of triplets is then used to optimize our cost function.

4.7 Results

We evaluate the performance of our complete system on the two data-sets described in Section 4.5. We use the three performance criteria described in Section 4.3.1. We refer the readers to figure 4.3 to get a sense for the difficulty of the data-set being used. We did not perform any pre-processing on these images.

4.8 Experiments using learned distance metric

We conducted numerous experiments using our learned distance metrics. We compared this performance to both the raw feature representations as well as FisherFaces. In all experiments we initially projected our features down to 100 PCA dimensions, and our mapped feature space always contained 50 dimensions, i.e., Φ was a matrix of dimensionality 50×100. We use $\alpha = 1$ for

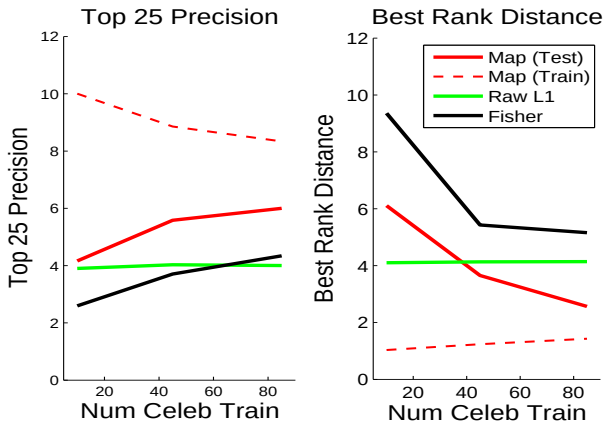


Figure 4.8: Effects of the number of individuals (celebrities) used for training on performance. (Left) Top 25 Precision metric. Results averaged over all celebrities in the test set. An L1 distance metric is optimized using 7×7 intensity patches. The x-axis is the number of individuals used for training ($\times 10$ this number of images are used for training). The y-axis is the precision of the top 25 returned celebrities. Note that the maximum achievable value is 9. Green line is the raw L1 performance of these features *before* mapping. Black line is the performance of FisherFaces [BHK97] when trained using the same number of celebrities. Dotted red line is the performance on the train set of individuals. Solid red line is the performance on the test set of individuals. Note that we are over-fitting: we expect the solid and dotted red lines to converge when we are not over-fitting. Our distance metric is out-performing both the raw L1 distance as well as FisherFaces when we train with 85 individuals. (Right) Same as left plot but using the Best Rank Performance metric. Lower is better. Again we see over-fitting, but our distance metric still far outperforms the baseline methods despite over-fitting. In this case best performance would be 1, which occurs when identical celebrities would always be neighboring one another.

these experiments.

In figure 4.8 we varied the number of celebrities we trained with in order to understand the asymptotic properties of learning with our distance metric. These plots have two main take-home messages: (1) Our learned distance metric performs well when compared to using either the raw features or FisherFaces (both techniques are widely used in the literature). (2) We are over-fitting, as indicated by the distance between the training and test error, indicating that if we trained with more individuals (i.e., collected more celebrities) we might be able to increase performance even further. The over-fitting is the result of the large number of parameters in the projection matrix Φ (5000) and the rather limited set of individuals we train with (we train with up to 85 individuals and a total of about 150 images).

In figure 4.9 we compare the performance of various feature sets (4 of them) using both L1/L2

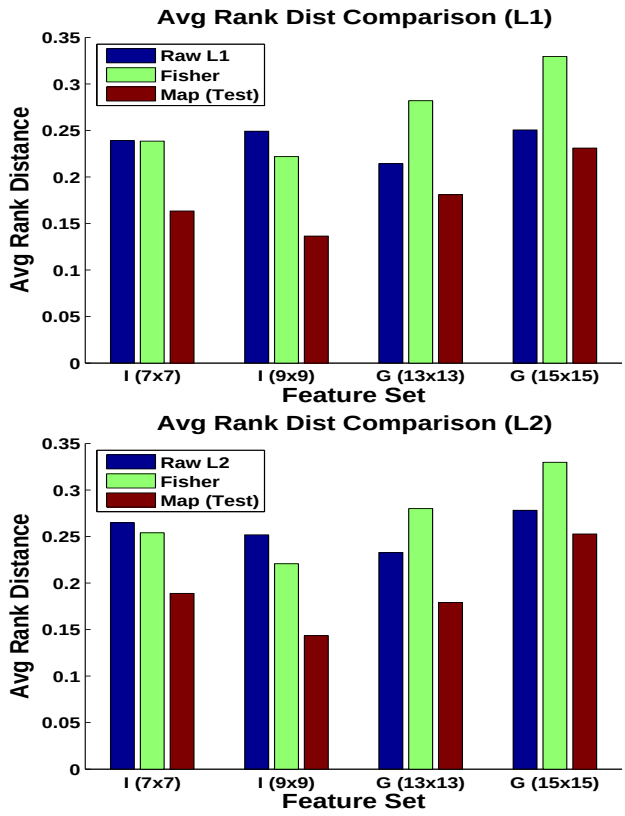


Figure 4.9: Comparison of the Average Rank performance metric. X-axis is 1 of 4 different feature representations: 7×7 Intensity patches, 9×9 Intensity patches, 13×13 Gradient patches, and 15×15 Gradient patches. These were chosen as they performed the best in Figure 4.6. First column is the raw L1 distance performance before learning the mapping. Second column is the performance using FisherFaces. Last columns are the performance using our mapping. We outperform the other metrics in every feature set tried here. Notably gradients perform worse here relative to intensities, see text for a discussion. (Bottom) Same plot as above but using the L2 distance to optimize the mapping and to compute the raw distance. The same trends seem to hold. $\alpha = 1$ for these experiments. Results averaged over 3 iterations.

distances. We note that using Gradient features yielded the high performance in figure 4.6, i.e., prior to mapping, while in our experiments using the mapping yielded the best performance. This is most likely due to the large size of the gradient feature vectors used compared to the size of the feature vectors used with only intensity (the intensity patches extracted were smaller than the gradient patches extracted). Indeed if we analyze the variance of the PCA coefficients obtained when projecting to 100 dimensions, we find that the gradient vectors encompass about 90% of the variance, while the intensity vectors encompass about 65% of the variance. The plots in figure 4.9 indicate that this loss of information has detrimental effects on performance. Note that due to over-

Avg Rank			Best Rank			Top 25		
Rand	Best	Feat	Rand	Best	Feat	Rand	Best	Feat
.5	**.13** (.25)	L1 I 9		**2.7** (4.2)	L2 I 9	1.92	**5.8** (4.8)	L1 I 7

Table 4.2: The best mapping functions. Rand: performance if we chose random images. Best: the performance of our best mapping algorithm. In parentheses: the performance without mapping, i.e. on the raw feature vectors. Feat: the feature set used. L1/L2: the distance metric used. I: intensity features. 7/9 the size of the patches used (e.g., 7: 7×7 patches).

fitting, increasing the projected space of PCA dimensions above 100 results in worse performance as well; i.e., if the number of parameters which must be optimized in our map Φ is too large, we will suffer from over-fitting.

In table 4.2 we show which mappings perform best on all three performance metrics. We also note the large performance gains achieved over not using the mapping, i.e., using only the extracted feature vector $\vec{x}$ in the variance performance metrics.

4.8.1 Filtering through results of Google-Images

Our algorithm was used as a filtering stage used to improve the results of the Google-Images search engine. Currently this search engine returns lots of irrelevant results, as it is only based on the text located on the web page near the image. For example, if a page about Tylenol mentions that Tom Cruise uses this pain killer, a query about Tom Cruise will likely return pictures with bottles of Tylenol pills. Our algorithm was used to filter through the results when one queries for a person.

The performance of the Google-Images search engine is displayed in figures 4.10-4.12 (bottom panels, blue curves). The horizontal axis represents the number of images recalled. The vertical axis represents the fraction of 'correct' recalled images, i.e. that contain the query person. Ground truth on the dataset was established manually prior to the experiment.

Starting from set of images returned by Google-Images, we first process them with the Viola-Jones face detector, and keep only the images on which it fired, and that passed our confidence test. The performance of this subset of images is displayed on the red curve. Naturally, in many images the detector does not fire, so that the new curve ends before the original curve on the horizontal axis.

The purple curve shows the performance obtained when only the images on which the Viola-Jones detector fired once (and passed our confidence test) were considered. The motivation is that there could be confusion between the different detections in images containing several individuals, which would require user intervention.

Face features are then extracted for each image, and encoded in a long vector. For example, if we use 7×7 patches, each face is represented by a vector of $7 \times 7 \times 19 = 931$ dimensions. The dimension of this vector is reduced to 50 using PCA. In this 50-dimensional face-space, the next step performs a simple greedy clustering: the two images with smallest distance form the seed of the cluster, then the image closest to this seed is added... this greedy clustering ends when the cluster reaches a given size. Although very simple, this clustering scheme seems extremely effective in order to filter through images of faces. Figures 4.10-4.12 (bottom panels) display the performance obtained both when this greedy clustering is used with the Euclidean norm (cyan curve), and when it is used with our optimized distance (green curve). Overall, the improvement gained from our distance metric is of the order of 10%. The global improvement compared to the raw results of the Google-Images search engine is dramatic. Figures 4.10-4.12 (top and middle panels) show a few examples of the top returned images both with the raw Google-Images engine, and after applying our algorithm, when searching for images of celebrities.

4.9 Discussion

We demonstrate excellent recognition results on a challenging data-set of facial images. There are numerous avenues for further exploration which include other cues used for facial recognition. For instance the hair is a very useful feature, as shown remarkably well in [Sin02] where one confuses Al Gore for Bill Clinton by mapping the hair from one to the other. In addition, the work could be extended to encompass larger variations in pose by learning a more powerful distance metric. Our current system was only trained on frontal poses, presumably we could also train our system on facial images which exhibit more pose variability.

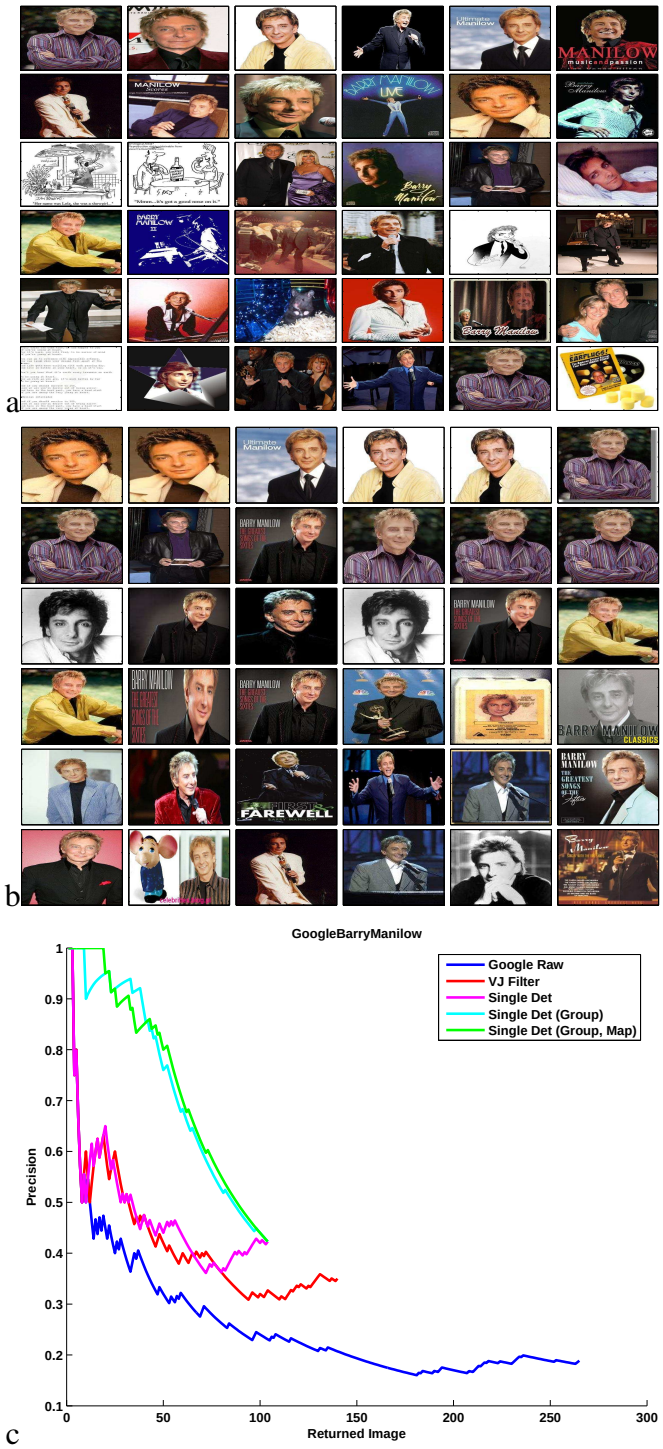


Figure 4.10: (top) Top images returned by the Google-Images engine (middle) Top images after applying our algorithm (bottom) performance curves

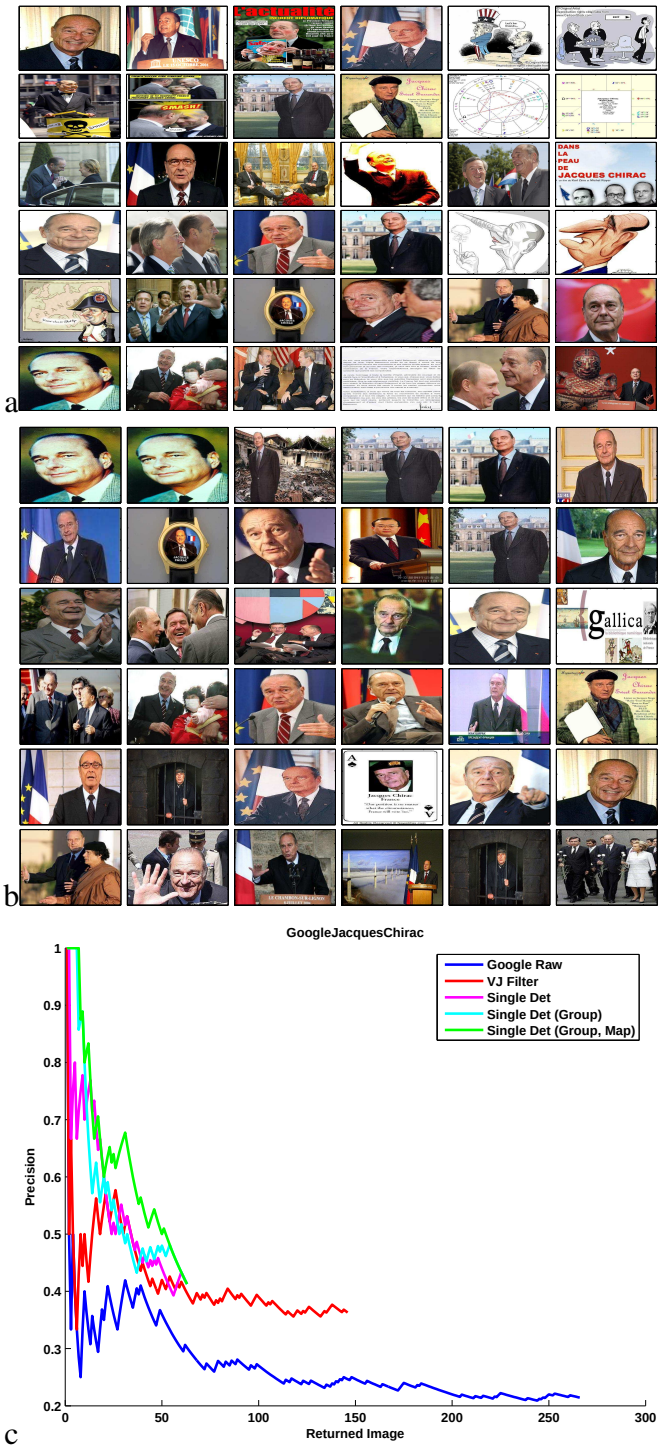


Figure 4.11: (top) Top images returned by the Google-Images engine (middle) Top images after applying our algorithm (bottom) performance curves

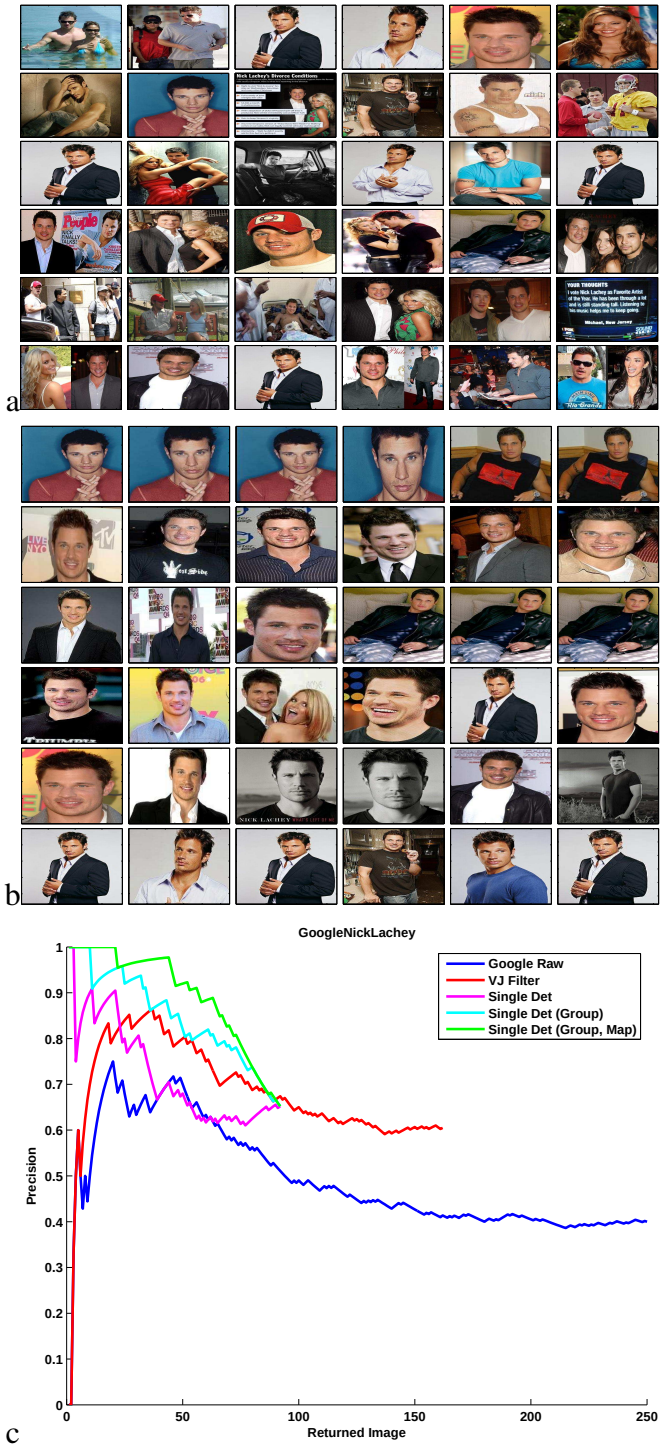


Figure 4.12: (top) Top images returned by the Google-Images engine (middle) Top images after applying our algorithm (bottom) performance curves

Chapter 5

Features-based mercer kernels for object recognition

5.1 Abstract

We present a new class of kernels that offer both good speed and classification performance in object recognition applications. We build upon the work from [GD05], and extend their feature-based kernel to a continuous range of scales. This leads to an improved discrimination power, better recognition performance, and enables the use of our kernel in features spaces with higher dimensionality. Our kernel may also be viewed as an efficient approximation of the Earth-Mover Distance introduced by [RTG00], with a computation time significantly lower than theirs. We show that it verifies the Mercer condition and is therefore suitable for use in SVM-based classifiers. We apply it to the multi-class object recognition problem, and report state-of-the-art results on the 'Caltech101' database.

5.2 Introduction

In the context of object recognition or image classification, many image representations recently proposed in the computational vision research community are based on unordered sets of feature elements or image parts. Representing images as a collection of local parts (e.g. salient points) is particularly useful because the entire image is often not informative for the target learning task. Numerous groups have proposed features- or parts-based techniques for object recognition [Low99, Low04], or to learn object classes [WWP00, ea05a, ea05b]. Part-based approaches have also been developed for image retrieval applications, such as in the work by Ye et al. [ea04b], Mikolajczyk et al. [MS01] or that of Carson et al. [ea02a].

In recent years, Support-Vector-Machine (SVM) has established itself as the algorithm of choice for classification. Unlike nearest- neighbor-like algorithms, it produces a compact representation of the target classes in a form of a small subset of the training vectors (support vectors), together with a quantification of uncertainty on the classification task in the form of bounds. However, most proposed techniques using SVM for image classification have been limited to using global image attributes represented by fixed dimensional vectors [OPV99, KEK01, TC01].

Kernel-based methods directly address this limitation by enabling the use of any custom-

designed similarity metric between data points. In the context of classification, kernel methods with SVM (Kernel-SVM) have demonstrated effectiveness at learning complex decision boundaries, and been showed to generalize well on unseen data [JTN04]. In order to be a valid kernel functional, it must however satisfy the Mercer conditions enforcing the kernel matrix to be symmetric positive definite.

This paper addresses the challenge of designing a set of valid kernel metrics on unorganized sets of feature points.

A number of previous approaches have been proposed to address this problem. A comprehensive list of previous work is presented in the work of Grauman et al. in [GD05]. Most previous work may be categorized in two groups. Some techniques rely on individual point-matches in order to compute the similarity between two unorganized point clouds [WC03, Lyu05, SJPF04]. Most approaches lead to kernel matrices that are non-positive definite (such as in [WC03] and [SJPF04]) and are often very computationally expensive. The other category of methods aims at calculating similarity between the two point clouds as a whole [LA03, SH05, KJ03]. For example [LA03] calculates similarity as a function of the principal angle between the two point clouds, while in [KJ03], the authors propose to model the two point clouds as mixtures of Gaussians, and suggest to calculate similarity based on the overlap of the probability density functions. These techniques lead to Mercer kernels, however they have been observed to lack to preserve the discriminative information in the local features, something that is critical for accurate classification. Grauman et al. recently proposed a new kernel on sets of points, Pyramid Match, that presents two key advantages [GD05]. First, it is very computationally efficient since is relies on a simple histogram overlap calculation at multiple scales without explicitly establishing correspondence between points. Second, it produces a positive-definite matrix making it a valid Mercer kernel.

In this work, we propose a new class of kernels, Max-Kernels, that can be been as a generalization of the Pyramid Match Kernel proposed by Grauman et al. [GD05]. We address in this paper several shortcomings of the Pyramid Match kernel that we believe are critical when designing a good similarity measure. First, we propose a way to overcome the significant distortion induced on the data by the step of histogram quantization. Since our new kernel does not explicitly collapse the raw point clouds into multi-dimensional histograms, no such distortion is created. Second,

by means of extending the mathematical formalism of Pyramid Match, our proposed kernel is also guaranteed to satisfy the Mercer conditions and thereby lead to positive-definite kernel matrices. Third, our proposed kernel also allows for incorporating domain specific knowledge into the similarity measure, making it a tunable kernel object that can be easily ported to other domains problems. Finally, we show that this new kernel performs better than Pyramid Match at the task of image classification.

The paper is organized as follows. We start in Section 5.3 by giving a brief description of the Pyramid Match kernel proposed by Grauman et al. [GD05]. Section 5.4 introduces our novel kernel as a continuous generalization of Pyramid Match. Section 5.5 shows how domain knowledge can be incorporated into the similarity function in forms of weights. Section 5.6 reports experimental results on image classification followed by conclusions in Section 5.7

5.3 The Pyramid Match kernel

In this section we provide a brief description of the pyramid match kernel. Further information can be found in Grauman et al.[GD05]. This kernel provides a measure of the similarity between two images, based on the similarity of sets of features within these images.

5.3.1 Description

An image A is represented by the set of features detected in it, $\mathcal{F}_A = \{f_i^A\}_i$. Let $N_A = |\mathcal{F}_A|$, and i indexes features in an image. Features location is obtained using an interest point operator, each feature is then encoded into a vector of user-defined dimension D.

A multi-resolution set of nested histograms (the pyramid) is built in feature space. The size of the bins or *scale* s_n, grows with the level of the pyramid by a factor 2^D. Scales are indexed by n. Each bin from level $n+1$ contains 2^D bins from level n. The finest resolution $n = 0$ is taken such that each feature falls into a separate bin. Conversely, at the coarsest resolution $n = L$ all features fall into a same huge bin.

The pyramid-match kernel compares two images A and B by mapping their features sets $\mathcal{F}_A$

and $\mathcal{F}_B$ in the pyramid. The histogram at scale n is denoted by H_n, and the function $I\Big(H_n(\mathcal{F}_A), H_n(\mathcal{F}_B)\Big)$ measures the 'overlap' between bins associated to A and B at scale n: if histogram H_n has r bins $\{H_n^j\}_{1\leq j\leq r}$,

$$I\Big(H_n(\mathcal{F}_A), H_n(\mathcal{F}_B)\Big) = \sum_{j=1}^{r} min\Big(|H_n^j(\mathcal{F}_A)|, |H_n^j(\mathcal{F}_B)|\Big) \tag{5.1}$$

Histograms are explored from the finest level of resolution to the coarsest. The kernel computes a weighted sum of bin overlaps at each level. The kernel is defined as

$$K(\mathcal{F}_A, \mathcal{F}_B) = \sum_{n=0}^{L} \frac{1}{2^n}\big[I(H_n(\mathcal{F}_A), H_n(\mathcal{F}_B)) - I(H_{n-1}(\mathcal{F}_A), H_{n-1}(\mathcal{F}_B))\big] \tag{5.2}$$

where the weight factor $\frac{1}{2^n}$ accounts for the fact that a small distance between two features f_i^A and f_i^B is more likely to characterize a correct match between I_A and I_B than a large large distance. The kernel is then normalized by the self-similarity of both images.

In order to prove that the kernel is positive-definite, Eq.(5.2) is rewritten as

$$K = \frac{min(N_A, N_B)}{2^L} + \sum_{n=0}^{L-1} \frac{1}{2^{n+1}} \cdot I(H_n(\mathcal{F}_A), H_n(\mathcal{F}_B)) \tag{5.3}$$

Mercer kernels stay positive-definite under addition and scaling by a positive constant. Besides, [OBV05] proves that the histogram intersection $I(H_n(\mathcal{F}_A), H_n(\mathcal{F}_A))$ is a Mercer kernel: the cardinality of an input set x is written as a long vector constituted by $|x|$ ones followed by $Z - |x|$ zeros, where Z is the cardinality of the largest set. The inner product between two such expansions is equivalent to the cardinality of the smaller set. Then, Mercer's theorem ensures that the resulting kernel is positive-definite.

5.3.2 Shortcomings of the Pyramid Match

The first issue is associated with the bin boundaries of the histograms: If we consider a correspondence [1] between two features f_i^A and f_i^B, let $d_i = f_i^A - f_i^B$, with $d_i = (d_i^1 ... d_i^D)$. The *optimal*

[1] note: we will indistinctly use 'correspond' and 'match' for features and for images. For images, it means that subsets of both images represent the same object or scene. For features, it means that both features are one and the

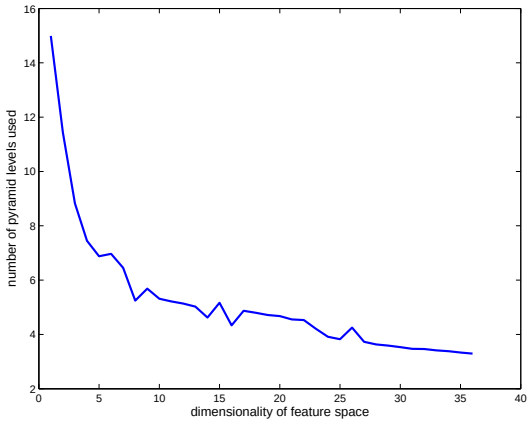


Figure 5.1: Average number of pyramid levels used during the computation of the similarity between two images

scale at which this correspondence can be detected is n_{d_i} such that $2^{n_{d_i}-1} < |d_i|_\infty \leq 2^{n_{d_i}}$, where $|d_i|_\infty = \underset{j}{max}(|d_i^j|)$ is the L_∞ norm.

However, the correspondence will be actually counted at this scale only if no bin boundary falls between f_i^A and f_i^B. The probability of this favorable situation is $(1 - \frac{|d_1|}{2^{n_d}})...(1 - \frac{|d_D|}{2^{n_d}})$, which statistically tends to 0 when the features dimension D becomes large. In fact, since $s_n = 2 * s_{n-1}$, this probability is always lower than $1/2$.

To illustrate how the discrimination power of the pyramid degrades when the dimension of the feature space increases, we counted the number of levels of the pyramid actually used, when computing similarity scores between real images. Features dimension was varied by changing the number of components in the PCA-SIFT representation [KS04]. Figure 5.1 displays the results. When $dimensionality = 35$, the computation of image similarities uses in average less than 4 levels, which is equivalent to using features that can take only 16 values along each dimension. This number was lower than 3 levels in average, if using the 128-dimensional SIFT descriptor.

Another issue is associated to the $n \rightarrow \frac{1}{2^n}$ weights. If we add to the database a feature whose minimum distance to other features is $\frac{1}{2^C}$ with $C >> 1$, we can cause the lowest scale of the pyramid to be arbitrarily small. Since all other points of the database are unchanged, in eq.(5.2) all terms will be washed away and tend to zero when $C \rightarrow +\infty$, except for the self-similarity terms (image matched to itself). The result is a gram matrix with ones on the diagonal, but neg-

same point, possibly viewed in different conditions.

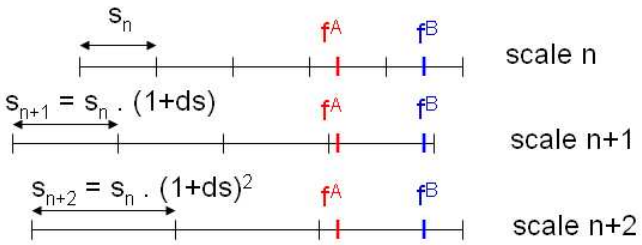


Figure 5.2: 1D diagram of the histograms pyramid.

ligible off-diagonal terms. Grauman tries to solve this 'large diagonal problem' with the matrix scaling technique developed in [ea02c]. However, a weight function carefully engineered should not present this explosion of the weights when the distances tend to zero. Section 5.5 presents one possible scheme for the design of this weight function.

5.4 From discrete to continuous: a new kernel

Our new kernel aims at solving the problems associated with the pyramid match kernel, namely the bin boundary effects due to the discretization at each scale, and the lack of a principled formulation of the weights.

. Starting from a pyramid as defined in [GD05], our first modification concerns the scale increments between consecutive levels: instead of using a constant multiplicative factor of 2, we choose a smaller factor $(1+ds)$, such that $s_n = s_0 \cdot (1+ds)^n$ for $-\infty < n < \infty$, with e.g. $s_0 = 1$. When $ds \rightarrow 0$, the levels of the pyramid become closer and closer.

Due to this finer discretization of scales, when $ds \rightarrow 0$, all matches will eventually be detected at different scales. This is to compare with the situation in [GD05], where many 'soft-matches' can be detected at the same scale. When $ds \rightarrow 0$, the computation of the pyramid match kernel first finds the pair of features with lowest distance from each other. This distance is weighted and added to the kernel value, then both features are discarded and not available any more for other matches. This process is repeated until there are no more features available in one of the images.

Simultaneously, in order to reduce border effects each level of the pyramid is shifted by a random amount with a uniform distribution in $[0, s_n]$. Intuitively, this means that matches will statistically be detected at their optimal scale or close to it (the optimal scale approximates best the

L_∞ distance between both features). The counterpart of the random shifts is that once a correspondence has been detected at one scale, we are not guaranteed anymore that it will be detected at all larger scales, as was the case in [GD05] since all bins boundaries are aligned. Eq.(5.2)& (5.3) are still valid, which proves that the resulting kernel is still positive-definite.

Besides, the weights $\frac{1}{2^i}$ are an arbitrary function of the scale, i.e. here the function $x \rightarrow \frac{1}{x}$. This function can be replaced by any user-specified positive function w. The only constraint is that w has to be monotonically decreasing, in order to have a positively weighted sum of Mercer kernels in eq.(5.3). Section 5.5 provides a principled probabilistic formulation that can be used to design w.

Let's consider to simplify notations, that the feature space has dimension $D = 1$ and that each image has a single feature: $\mathcal{F}^A = f^A, \mathcal{F}^B) = f^B$ (see figure 5.2). Eq.(5.3) is rewritten as (we extend the summation to $n \rightarrow -\infty$ and $n \rightarrow +\infty$)

$$K_{ds}(f^A, f^B) = \sum_{n=-\infty}^{+\infty} \big[w(s_n) - w(s_{n+1})\big] \cdot I(H_{ds,n}(f^A), H_{ds,n}(f^B)) \tag{5.4}$$

In the limit when $ds \rightarrow 0$, we can prove using infinitesimal calculation, that

$$\lim_{ds \rightarrow 0} K_{ds}(f^A, f^B) = -\int_d^\infty \frac{s-d}{s} w'(s) ds \tag{5.5}$$

with probability 1, where $d = |f^B - f^A|$. The term 'probability 1' concerns the random uniform draw used to shift the histogram grids at different scales.

In order to save space we will not detail the proof here. Intuitively, $\frac{s-d}{s}$ represents at a given scale s, the probability of not having any boundary between f^A and f^B. The integral's lower bound d is due to the fact that no bin smaller than d can contain both f^A and f^B. The term $-w'(s)ds$ corresponds to the term $\big[w(s_n) - w(s_{n+1})\big]$ in (5.4)

We denote $K_0(f^A, f^B) = K_0(d) = -\int_d^\infty \frac{s-d}{s} w'(s) ds$, K_0 is automatically positive-definite, as the limit of a series of positive-definite kernels.

The weight $w(d)$ associated to the match $f^A \leftrightarrow f^B$ has been replaced in the continuous case

by $K_0(d)$. Note that the mapping $w \rightarrow K_0$ can be inverted using the relations

$$\begin{aligned} \frac{d[K_0(f_i^A, f_i^B)]}{dd} &= \int_d^\infty \frac{w'(s)}{s} ds \\ \text{from which we obtain } w'(d) &= -d \cdot \frac{d^2 K_0(d)}{dd^2} \end{aligned} \tag{5.6}$$

In higher dimension D, the kernel K_0 becomes

$$K_0(f^A, f^B) = -\int_d^\infty \int_d^\infty ... \int_d^\infty \big[\frac{s_1 - d^1}{s_1} \cdot \frac{s_2 - d^2}{s_2} ... \frac{s_D - d^D}{s_D} \cdot w'(d) \cdot ds_1 \cdot ds_2 ... ds_D\big] \tag{5.7}$$

where $d = |f^A - f^B|_\infty$ and $d^1 = |(f^A - f^B)_1| ... d^D = |(f^A - f^B)_D|$ (absolute distances along each component).

For images with multiple features, the discrete kernels $K_{ds}(\mathcal{F}^A, \mathcal{F}^B)$ converge similarly to $K_{max}(\mathcal{F}^A, \mathcal{F}^B) = \sum_{i\_sorted} K_0(f_i^A, f_i^B)$. This kernel is positive-definite as well, as a sum of positive-definite kernels. It is important to note that here the matches are sorted by increasing L_∞ norm, to reflect the pyramid match counting process: the best match is counted first then discarded, then the second best match....

This leads to the following algorithm to compute our new kernel :

Algorithm:

1. Compute all L_∞ distances between a feature in A and a feature in B; sort these distance in increasing order; initialize kernel to 0.

2. Select the smallest distance d_1; add $w(d_1)$ to the kernel, where w is a user-defined weight function, and discard both features involved.

3. Repeat step 2 until no feature is left in one of the images.

The distance used here is the L_∞ norm, i.e., the maximum value along components of a vector, hence we call this kernel the *max-kernel*. Also, we used the abusive notation w instead of K_0.

5.5 Probabilistic interpretation of the weights

In the following section we present here a probabilistic model that can be used to compute the weights used in the max-kernel.

5.5.1 Notations and assumptions

The notations $I_A \leftrightarrow I_B$ and $I_A \not\leftrightarrow I_B$ denote ground truth regarding the image match between images I_A and I_B. $I_A \leftrightarrow I_B$ means that there is a subset of I_A and a subset of I_B that depict the same scene or object, while $I_A \not\leftrightarrow I_B$ that I_A and I_B don't correspond to each other.

Ground truth at the feature level is written $f_i^A \leftrightarrow f_i^B$ and $f_i^A \not\leftrightarrow f_i^B$: $f_i^A \leftrightarrow f_i^B$ denotes that the match between f_i^A and f_i^B is correct, $f_i^A \not\leftrightarrow f_i^B$ denotes that it is incorrect.

Let $\hat{M}_{AB} = min(N_A, N_B)$ the number of matches estimated by the features matching algorithm underlying our kernel, and M_{AB} the number of true matches between I_A and I_B. If $I_A \not\leftrightarrow I_B$, obviously $M_{AB} = 0$. If $I_A \leftrightarrow I_B$, $0 \leq M_{AB} \leq min(N_A, N_B)$, and the extremes $M_{AB} = 0$ and $M_{AB} = min(N_A, N_B)$ are both unlikely.

The distance in feature space between two features f_i^A and f_i^B being matched, is denoted by d_i.

We estimate empirically the distributions $P(d_i|f_i^A \leftrightarrow f_i^B)$ and $P(d_i|f_i^A \not\leftrightarrow f_i^B)$ from collections of ground truth correct and incorrect matches. Ground truth correct and incorrect matches can be collected automatically with the method developed in [MP05, MP07b], or via users who click on corresponding features. $P(I_A \leftrightarrow I_B)$ and $P(I_A \not\leftrightarrow I_B)$ are also assumed to be known, they are the prior probability of two images matching or not matching.

5.5.2 Decomposition

We are interested in the following ratio, to use as a weight function for the hypothesized features correspondences:

$$R = \frac{P(I_A \leftrightarrow I_B|\{d_i\})}{P(I_A \not\leftrightarrow I_B|\{d_i\})} \tag{5.8}$$

The use of this quantity is motivated by the fact that the only information extracted from the pair of images is the distance between features being matched. It is comparable to the 'reward' used in [YC99].

Bayes rule leads to

$$R = \frac{P(\{d_i\}|I_A \leftrightarrow I_B)}{P(\{d_i\}|I_A \not\leftrightarrow I_B)} \frac{P(I_A \leftrightarrow I_B)}{P(I_A \not\leftrightarrow I_B)} \tag{5.9}$$

We decompose the numerator and denominator respectively into (unnecessary dependencies are omitted)

$$P(\{d_i\}|I_A \leftrightarrow I_B) = \prod_i \left[P(d_i|\hat{M}_{AB}, I_A \leftrightarrow I_B) \cdot P(\hat{M}_{AB}|N_A, N_B) \cdot P(N_A) \cdot P(N_B)\right] \tag{5.10}$$

A similar equation (with $I_A \not\leftrightarrow I_B$) holds for the denominator. Eq.(5.10)uses a strong a priori assumption: namely, that all features correspondences are independent of each other. In a more accurate computation, when conditioning on one match, one should condition on the set of previous matches, as this is how both the pyramid match and our algorithm proceed. The quantities $P(d_i|\hat{M}_{AB}, I_A \leftrightarrow I_B)$ and $P(d_i|\hat{M}_{AB}, I_A \not\leftrightarrow I_B)$ are the probabilities of the observed distances.

In the denominator, we are considering 2 images that don't match, so that all features correspondences estimated by the algorithm are incorrect.

In the numerator, some distances come from a correct match (counted in M_{AB}), while some come from an incorrect match (incorrect matches account in particular for the difference $\hat{M}_{AB} - M_{AB}$). For each feature correspondence, we use a model that is a linear combination of these two contributions.

$$P(d_i|\hat{M}_{AB}, I_A \leftrightarrow I_B) = P(d_i|f_i^A \leftrightarrow f_i^B) \cdot \alpha + P(d_i|f_i^A \not\leftrightarrow f_i^B) \cdot (1 - \alpha) \tag{5.11}$$

where $\alpha = P(f_i^A \leftrightarrow f_i^B|\hat{M}_{AB}, I_A \leftrightarrow I_B)$

We marginalize over the number of true features correspondences M_{AB} and obtain

$$\alpha = \sum_{M_{AB}=0}^{min(N_A,N_B)} \frac{M_{AB}}{\hat{M}_{AB}} \cdot P(M_{AB}|\hat{M}_{AB}, I_A \leftrightarrow I_B) \tag{5.12}$$

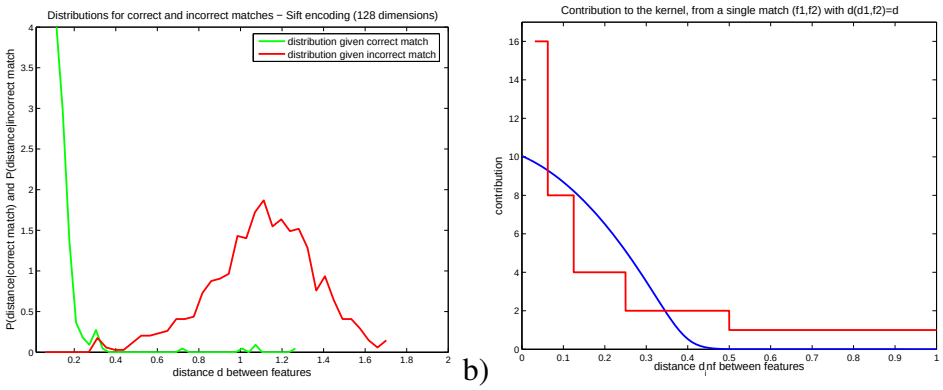


Figure 5.3: a) empirical densities $P(d_i|f_i^A \leftrightarrow f_i^B)$ and $P(d_i|f_i^A \not\leftrightarrow f_i^B)$ b) corresponding weight function according to our probabilistic model (blue - the densities in a) have been smoothed by low-pass filtering). This function performs a soft threshold of the $L - \inf$ distance between a pair of features. The red curve displays the weights used by the pyramid match. They take only discrete values as the pyramid match takes into account the size of the bin and not the actual inter-features distance. Besides, the weight function tends to ∞ when $d \to 0$

(we used $P(f_i^A \leftrightarrow f_i^B | M_{AB}, \hat{M}_{AB}, I_A \leftrightarrow I_B) \approx \frac{M_{AB}}{\hat{M}_{AB}}$)

Let $f(M_{AB}, \hat{M}_{AB}) = P(M_{AB}|\hat{M}_{AB}, I_A \leftrightarrow I_B)$, we can assume that $f(kx, ky) = \frac{1}{k} f(x, y)$, which reflects the idea that M_{AB} is expected to vary linearly with $\hat{M}_{AB}$.

We approximate the summation in eq.(5.12) by an integral and obtain (with the change of variables $M_{AB} = k.\hat{M}_{AB}$)

$$\begin{aligned} \alpha &= \int_{M_{AB}=0}^{\hat{M}_{AB}} \frac{M_{AB}}{\hat{M}_{AB}} f(M_{AB}, \hat{M}_{AB}) dM_{AB} \\ &= \int_{k=0}^{1} k.f(k.\hat{M}_{AB}, \hat{M}_{AB}).\hat{M}_{AB}.dk = \int_{k=0}^{1} k.f(k, 1).dk \end{aligned} \tag{5.13}$$

(As a sanity check, notice that $\int_{k=0}^{1} f(k, 1).dk = 1$ since f is a probability density, therefore we are guaranteed to have $0 < \alpha < 1$)

In summary, with the terms canceling out between numerator and denominator, the ratio R from eq.(5.8) becomes

$$R = \prod_i \left[(1 - \alpha) + \alpha \cdot \frac{P(d_i|f_i^A \leftrightarrow f_i^B)}{P(d_i|f_i^A \not\leftrightarrow f_i^B)} \right] \tag{5.14}$$

We can switch to an additive formula, more suited to our kernel approach:

$$\begin{aligned} log(R) &= \sum_i g(d_i) \\ \text{where } g(d_i) &= log\big[(1-\alpha) + \alpha \cdot \frac{P(d_i|f_i^A \leftrightarrow f_i^B)}{P(d_i|f_i^A \not\leftrightarrow f_i^B)}\big] \end{aligned} \tag{5.15}$$

and we can use $g(d)$ as a new weight of each match, for our max-kernel.

In order to compute the parameter α, we model $f(.,1)$ by a gaussian with mean 0.2 and variance 0.8. We obtain $\alpha \approx 0.45$. We noticed that this value varies little with the mean or standard deviation chosen (value usually between 0.4 and 0.5).

Figure 5.3-a displays experimental values of $P(d_i|f_i^A \leftrightarrow f_i^B)$ and $P(d_i|f_i^A \not\leftrightarrow f_i^B)$, collected with the method from [MP05, MP07b]. The features are encoded with the 128-dimensional SIFT descriptor [Low99, Low04]. The horizontal axis is the euclidean distance between both features in a candidate correspondence. Gaussian approximations to the densities are also displayed. The curves corroborate our intuition: correct correspondences are peaked close to zero distance. The distribution for incorrect matches is wider, and peaked at a higher value. Figure 5.3 shows the corresponding weight function obtained with our probabilistic model. This function performs a soft threshold that also matches our intuition: correspondences with a low distance between both features receive a high similarity score, while correspondences with a high distance get a score close to zero. Besides, this weight function has a finite limit when $d \to 0$, which will lead to a kernel matrix that is well-behaved (no large diagonals).

5.6 Experiments and results

5.6.1 A good approximation of the optimal distance

As mentioned in [GD05], the inverse of the pyramid-match similarity is an approximation of the optimal distance between 2 sets of features. 'Optimal' means here the distance that minimizes the sum of euclidean distances between corresponding features. When the weights are chosen as $w(d) = \frac{1}{d}$ (similar to the pyramid match weights of $\frac{1}{2^n}$), the inverse of our kernel is also an approximation - hopefully more accurate - of the optimal distance.

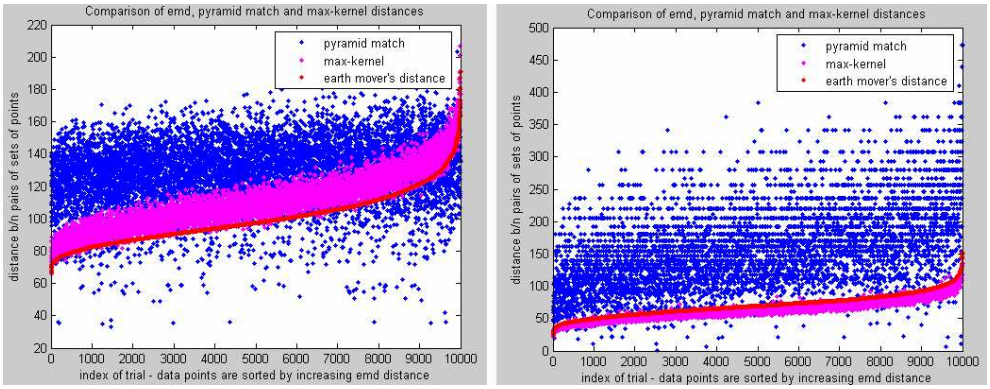


Figure 5.4: With a weight function set to $w(d) = \frac{1}{d}$, the inverse of our kernel (violet) is an approximation of the Earth Mover's Distance [RTG00] (red). The distance corresponding to the pyramid match kernel for the same samples is reported in blue. Left: equal cardinality for both 'images'. Right: cardinalities are chosen randomly between 5 and 100 points

An experiment was run on synthetic $2D$ data to compare both our kernel and the pyramid match to the optimal distance. The optimal distance was computed by a linear programming solution to the Transportation Problem [RTG00].

We followed the same setup as in [GD05], with two datasets. One dataset had 'images' with a fixed number of 100 features, while in the other the number of features was set to a random value between 5 and 100. Figure 5.4 displays the scores obtained for 10,000 trials, for the pyramid-match kernel, our kernel and the earth mover's distance (optimal match). Trials are sorted from 1 to 10,000 (left to right) according to the value of the optimal distance.

Our kernel provides a distance value that is significantly closer to the earth mover's distance than the pyramid match. This is due partly to the fact that our kernel uses the actual value of the euclidean distance between corresponding features, instead of the crude approximation of the form $\frac{1}{2^n}$ used by the pyramid match. Besides, accuracy is improved by using actual features correspondences, instead of only soft-matches.

5.6.2 Computation time

The trade-off of the improved performance over the pyramid match, is an increased computation time. [GD05] reports a theoretical complexity that is linear in the number of features in the images. In contrast, the theoretical complexity of our kernel is $O(N^2.log(N))$, since the set of all pairwise

Figure 5.5: Exemplars from the first 20 categories of the Caltech101 dataset. The 'easiest categories' are Faces, Motorbikes, Airplanes.

distances is sorted. The complexity of the simplex algorithm used in the earth mover's distance is exponential in the worst case.

In practice, the limiting step in our algorithm was not the sorting step in real-world images with 500-1000 detections, but the computation of the pairwise distances, which is $O(N^2)$.

Actual computation times were of the order of 1/100s per image pair for the pyramid match (including building the pyramid and the computation of the histograms overlap), 0.1s per image pair for our algorithm, and 3s per image pair with the earth mover's distance. All implementations were coded in C and executed on a Pentium 4 2.4GHz.

Note that if one wants to stay within a linear complexity, one can use the intermediary stage described in section 5.4, i.e. a pyramid match kernel with fine but still discrete scale increments $(1 + ds)$. Even with fine discretization, as long as we keep using discrete scale increments, the complexity stays in O(N) (although, possibly with a large constant factor).

We conjecture that the computation time should scale sub-linearly with the size of the database. Indeed, the principle underlying the SVM architecture, is that a small subset of support vectors should be able to characterize the class of interest. This number should not increase significantly when the database becomes larger.

In practice, for the small training sets used for each category in sec.5.6.3, about 80% of the training samples end up as support vectors. This is also partly due to the fact that the images within each category are highly correlated.

5.6.3 Performance on a classifi cation task

Our kernel was applied in an object recognition system using SVM classifiers. At the training stage, for a given class all values of the kernel are computed on pairs of images from the training

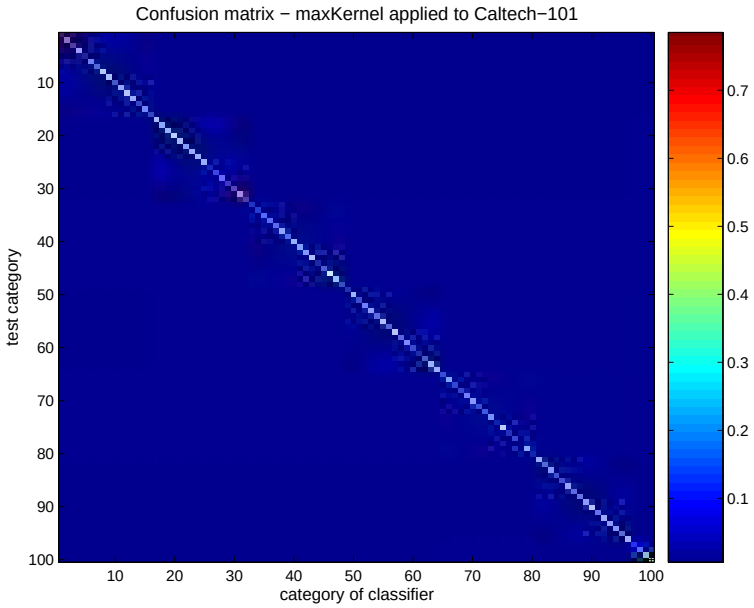


Figure 5.6: Confusion matrix for a multi-class classification task on the Caltech101 database. The vertical axis corresponds to the category from which the test examples are drawn. The horizontal axis corresponds to the winning classifier, i.e. the one that had the highest performance on the considered test image. One of the two 'Faces' categories was discarded, as well as the 'Background_google" category. These results were obtained with 30 training images per category

set, these values form the gram matrix supplied to the classifier. At the testing stage, we need only to compute kernel values between test images and those images that were selected as support vectors.

Features were extracted from training as well as test images using the popular difference-of-gaussians/128D-SIFT combination of detectors and descriptors [Low99, Low04]. These features were showed to be quite robust to perturbations like viewpoint or light change [MP05, MP07b, MS05]. Typical number of detections ranged from 200 to 500 features per image. For the classifier, we used the libsvm package available from http://www.csie.ntu.edu.tw/ cjlin/libsvm/.

We applied our system to the Caltech-101 dataset available from http://www.vision.caltech.edu/. This database consists of 101 object categories that contain significant clutter, occlusion and intra-class variations.

The setup used was similar to the one reported in [GD05],[ZMLS05]: 30 training images, testing performed on all the remaining images in each category. We used a one-versus-all test scheme. When averaged over the testing images, the performance of our kernel was 50%, versus respec-

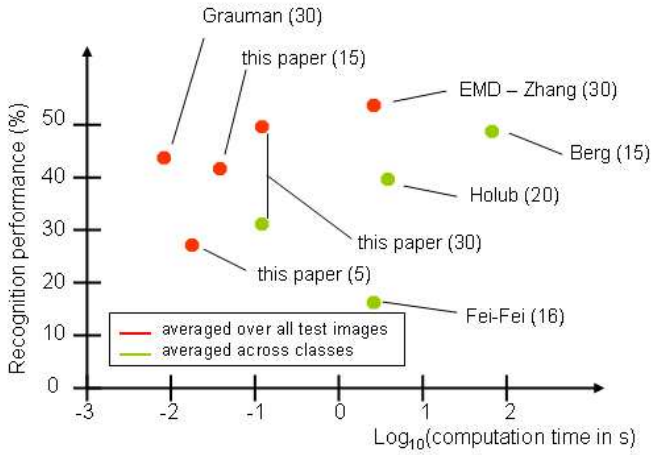


Figure 5.7: Summary of performances of various methods on the Caltech101 dataset. The performances obtained with the 'average over all test images' and 'average along the diagonal of the confusion matrix' are not directly comparable. Methods using each scheme are displayed respectively in green and red. The number of training samples is mentioned in parentheses.

tively 43% for the pyramid match [GD05], and 53% for the earth mover's distance [ZMLS05]. If we replace the probabilistic weights from sec.5.5, by the 'default' function $w(d) = \frac{1}{d}$, the performance drops a few percent to 47%. We believe that this difference is due in large part to the fact that $w(d) = \frac{1}{d}$ leads to a gram matrix with a large diagonal.

It is important to note that these results were obtained with a different averaging scheme from the performances of 16%, 40% and 48% respectively reported by [FFFP04], [ea05a] and [BBM05]. In these studies, results are averaged first inside the same class, then the average result over all classes is considered (this amounts to taking the average diagonal value of the confusion matrix in figure 5.6). The performance with this second normalization scheme is significantly lower, because the 'easiest' classes (Faces, Motorbikes, Airplanes), are precisely those with the highest numbers of exemplars. With the 'average of diagonal values' normalization scheme, our performance drops to 31%.

Another interesting point concerns the number of samples used for each class during the training stage. We used 30 examples to train each category, as do [GD05] and [ZMLS05], while [ea05a] and [BBM05] use only respectively 20 and 15 training examples per category. We repeated our classification experiments with 15 and 5 examples, the performance dropped to 42% and 28% respectively.

Figure 5.7 displays the results of various methods on the Caltech101 dataset. To date, the method from [BBM05] obtains the highest performance, but this is also the slowest method. Besides, most of the results reported in [BBM05] are obtained with manually segmented training images, although they mention that performance on unsegmented data is not much lower. Two different colors are used in order to distinguish between both normalization schemes used to compute average recognition performances.

5.7 Conclusion

The kernel presented in this paper offers a generalization of the multi-scale approach from [GD05] to a continuous range of scales. The use of continuous scales solves the boundary problems associated with histograms. In contrast with [GD05] which offers only fixed parameters for the design of the histograms in the pyramid, our kernel can be tuned via a weight function that takes into account the distance between pairs of features put into correspondence. We introduced a domain knowledge based, probabilistic approach to design this weight function. A side benefit is to reduce the emphasis on self-similarity that was intrinsic to the pyramid match in [GD05]. We show that our max-kernel retains the positive-definiteness of the pyramid match kernel, making it a valid Mercer kernel that can be used to generate the gram matrix in SVM-based classifiers.

An important point is that our max-kernel constitutes a useful trade-off between speed and recognition performance. It provides a better approximation to the optimal matching distance than the pyramid match as well as an improved recognition rate on real-world images. At the same time, it runs significantly faster than the Earth Mover's Distance [RTG00] or Berg's approach [BBM05], making it more suited for object recognition tasks with large datasets.

Appendix - Mercer condition

We wish to prove the transition from eq.5.4 to eq.5.5. In order to simplify notations and calculations, let's assume that s_n is additive of the form $s_n = n.ds$, instead of the multiplicative form

$s_n = (1+ds)^n$ described in sec.5.4. We rewrite eq.5.4 using Taylor's formula:

$$K_{ds}(f^A, f^B) = \sum_{n=0}^{+\infty} -w'(\xi_n) \cdot (s_{n+1} - s_n) \cdot I(H_{ds,n}(f^A), H_{ds,n}(f^B)) \tag{16}$$

where $\xi_n \in [s_n, s_{n+1}]$. In the expression above, $I(H_{ds,n}(f^A), H_{ds,n}(f^B))$ is a random variable that takes value 1 with probability $\frac{s_n - d}{s_n}$ (when no boundary falls between f^A and f^B) and 0 with probability $\frac{d}{s_n}$. Note that if $s_n < d$, then $I(H_{ds,n}(f^A), H_{ds,n}(f^B))$ is always 0.

Eq. 16 is a Monte Carlo approximation. When $ds \to 0$ we have, with probability 1 on the draw of the grid shifts,

$$K_{ds}(f^A, f^B) \underset{ds \to 0}{\to} \int_d^{\infty} -w'(s) \frac{s-d}{s} ds \tag{17}$$

Proof: let $g(x) = -w'(x)$, $G(x,y) = g(x)$,

$$h(x) = \begin{cases} 0 & \text{if } x \leq d \\ \frac{x-d}{x} & \text{if } x \geq d \end{cases} \text{ (see fig.8)} \tag{18}$$

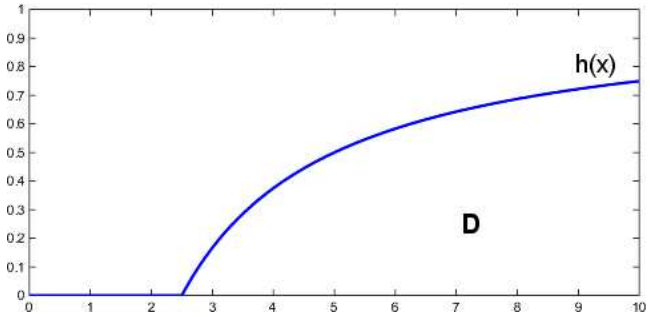


Figure 8: function $h(x)$ and domain D.

Let D the domain of 2D space bounded by $y = 0$ and $y = h(x)$, and D_1 the larger domain of 2D space bounded by $y = 0$ and $y = 1$. Let

$$
\begin{aligned}
J &= \int_0^\infty g(x)h(x)dx = \iint_D G(x,y)dxdy \\
&= \iint_{D_1} G(x,y)k(x,y)dxdy \\
&= \frac{1}{V}\iint_{D_1} G(x,y)k(x,y)Vdxdy
\end{aligned} \tag{19}
$$

where

$$
k(x,y) = \begin{cases} 1 & \text{if } (x,y) \in D \\ 0 & \text{if } (x,y) \notin D \end{cases} \tag{20}
$$

and V is the volume between $y = 0$ and $y = 1$ (since this volume is infinite, we can truncate the integration domain to $x \in [0, T]$ with a large value of T).

Using Monte-Carlo summation to approximate the integral in eq.19, we obtain

$$
J \approx \frac{V}{n_{max}} \sum_{n=0}^{n_{max}} G(x_n, y_n)k(x_n, y_n) \tag{21}
$$

where (x_n, y_n) is a 2D random variable with uniform density in D_1. n_{max} characterizes how dense the sampling of the domain is. Looking back at eq.16, $\frac{V}{n_{max}} = (s_{n+1} - s_n)$, $x_n = \xi_n$, and $k(x_n, y_n)$ corresponds to $I(H_{ds,n}(f^A), H_{ds,n}(f^B))$ when y_n is uniform in $[0, 1]$.

When the sampling of D_1 becomes denser, i.e. when $ds \to 0$, the value of the Monte-Carlo approximation tends to the value of the integral in eq.17.

Bibliography

[AGF04] Y. Amit, D. Geman, and X. Fan. A coarse-to-fine strategy for multiclass shape detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26((12)):1606–1621, 2004.

[Bal81] D.H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13((2)):111–122, 1981.

[BBM05] AC. Berg, TL. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[Bea78] P.R. Beaudet. Rotationally invariant image operators. *International Joint Conference on Pattern Recognition*, pages 579–583, 1978.

[BG98] C. Ballester and M. Gonzalez. Affine invariant texture segmentation and shape from texture by variational methods. *Journal of Mathematical Imaging and Vision*, 9:141–171, 1998.

[BHK97] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[BL97] J.S. Beis and D.G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1000–1006, 1997.

[BL03] M. Brown and D.G. Lowe. ecognising panoramas. *International Conference on Computer Vision*, 2003.

[BMP02] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24((24)):509–522, 2002.

[BMW98] M.C. Burl and P. Perona M. Weber. A probabilistic approach to object recognition using local photometry and global geometry. *European Conference on Computer Vision*, pages 628–641, 1998.

[Bou99] J.Y. Bouguet. Visual methods for three-dimensional modeling. *PhD thesis, Caltech*, 1999.

[Che95] Y. Cheng. Mean shift mode seeking and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17((8)):790–799, 1995.

[CJ04] G. Carneiro and A.D. Jepson. Flexible spatial models for grouping local image features. *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[CM05] O. Chum and J. Matas. Matching with prosac - progressive sample consensus. *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[CP84] J.L. Crowley and A.C. Parker. A representation for shape based on peaks and ridges in the difference of low-pass transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:156–168, 1984.

[DS04] G. Dorko and C. Schmid. Object class recognition using discriminative local features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004.

[dWS06] J. Van de Weijer and C. Schmid. Coloring local feature extraction. *European Conference on Computer Vision*, 2006.

[ea93] M. Lades et al. Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, 42((3)):300–310, 1993.

[ea02a] C. Carson et al. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[ea02b] J. Matas et al. Robust wide baseline stereo from maximally stable extremal regions. *British Machine Vision Conference*, 2002.

[ea02c] J. Weston et al. Dealing with large diagonals in kernel matrices. *Principles of Data Mining and Knowledge Discovery. Spring Lecture Notes in Computer Science 243*, 2002.

[ea04a] T.L. Berg et al. Names and faces in the news. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 848–854, 2004.

[ea04b] Y. Ye et al. An efficient parts-based near-duplicate and sub-image retrieval system. *ACM international conference on Multimedia*, 2004.

[ea05a] A. Holub et al. Combining generative models and fisher kernels for object recognition. *International Conference on Computer Vision*, 2005.

[ea05b] J. Sivic et al. Discovering object categories in image collections. *International Conference on Computer Vision*, 2005.

[ea05c] K. Mikolajczyk et al. A comparison of affine region detectors. *International Journal of Computer Vision*, 2005.

[ea06a] M. Everingham et al. The pascal visual object classes challenge 2006 (voc2006) results. *Technical Report*, 2006.

[ea06b] P. Sinha et al. Face recognition by humans: nineteen results all computer vision researchers should know about. *Proceedings of the IEEE*, 94((2)), 2006.

[ESZ06] M. Everingham, J. Sivic, and A. Zisserman. Hello! my name is... buffy – automatic naming of characters in tv video. *British Machine Vision Conference*, 2006.

[EZ06] M. Everingham and A. Zisserman. Regression and classification approaches to eye localization in face images. *FG*, pages 441–448, 2006.

[FA91] W. Freeman and E. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13((9)):891–906, 1991.

[FB81] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Communications ACM*, 24:381–395, 1981.

[FB04] F. Fraundorfer and H. Bischof. Evaluation of local detectors on non-planar scenes. *OAGM/AAPR workshop, Austrian Association for Pattern Recognition*, 2004.

[FBF77] J. Friedman, J. Bentley, and R. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Software*, 3:209–226, 1977.

[FFFP04] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 categories. *Workshop, IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[FG01] F. Fleuret and D. Geman. Coarse-to-fine face detection. *International Journal on Computer Vision*, 41:85–107, 2001.

[FH00] P. Felzenschwalb and D. Huttenlocher. Efficient matching of pictorial structures. *IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[For86] W. Forstner. A feature based correspondence algorithm for image matching. *Intl. Arch. Photogrammetry and Remote Sensing*, 24:160–166, 1986.

[FPZ03] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[FPZ05] R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[FTG04] V. Ferrari, T. Tuytelaars, and L. Van Gool. Simultaneous object recognition and segmentation by image exploration. *European Conference on Computer Vision*, 2004.

[FTVG05] V. Ferrari, T. Tuytelaars, and L. Van-Gool. Wide-baseline multiple-view correspondences. *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[GD05] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. *International Conference on Computer Vision*, 2005.

[HS88] C. Harris and M. Stephens. A combined corner and edge detector. *Alvey Vision Conference*, pages 147–151, 1988.

[HZ00] R. Hartley and A. Zisserman. Multiple view geometry in computer vision. *Cambridge editor*, 2000.

[Int] Intel. Opencv computer vision library. *http://www.intel.com/technology/computing/opencv/*.

[JTN04] J.Shawe-Taylor and N.Cristianini. Kernel methods for pattern analysis. *Cambridge*, 2004.

[KEK01] K.Goh, E.Y.Chang, and K.T.Cheng. Svm binary classifier ensembles for multiclass image classification. *Conference on Information and Knowledge Management*, 2001.

[KJ03] R. Kondor and T. Jebara. A kernel between sets of vectors. *International Conference on Machine Learning*, 2003.

[KS04] Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[KSP07] A. Kushal, C. Schmid, and J. Ponce. Flexible object models for category-level 3d object recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[KZB04] T. Kadir, A. Zisserman, and M. Brady. An affine invariant salient region detector. *European Conference on Computer Vision*, pages 228–241, 2004.

[LA03] L.Wolf and A.Shashua. Learning over sets using kernel principal angles. *Journal of Machine Learning Research*, 2003.

[LG97] T. Lindeberg and J. Garding. Shape-adapted smoothing in estimation of 3-d shape cues from affine deformations of local 2-d brightness structure. *Image and Vision Computing*, 15((6)):415–434, 1997.

[Lin94] T. Lindeberg. Scale-space theory: a basic tool for analising structures at different scales. *Journal of Applied Statistics*, 21((2)):225–270, 1994.

[Lin98] T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30((2)):79–116, 1998.

[LK81] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *7th International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[LLS04] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. *ECCV04 Workshop on Statistical Learning in Computer Vision*, 2004.

[Low85] D.G. Lowe. Perceptual organization and visual recognition. *Kluwer Academic*, 1985.

[Low99] D.G. Lowe. Object recognition from local scale-invariant features. *International Conference on Computer Vision*, 1999.

[Low01] D.G. Lowe. Local feature view clustering for 3d object recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[Low04] D.G. Lowe. 'distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60((2)):91–110, 2004.

[LSP04] S. Lazebnik, C. Schmid, and J. Ponce. Semi-local affine parts for object recognition. *British Machine Vision Conference*, 2004.

[LSP06] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[LSS05] B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 878–885, 2005.

[Lyu05] S. Lyu. Mercer kernels for object recognition with local features. *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[MLS05] K. Mikolajczyk, B. Leibe, and B. Schiele. Local features for object class recognition. *International Conference on Computer Vision*, 2005.

[MLS06] K. Mikolajczyk, B. Leibe, and B. Schiele. Multiple object class detection with a generative model. *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[MMP04] P. Moreels, M. Maire, and P. Perona. Recognition by probabilistic hypothesis construction. *European Conference on Computer Vision*, 2004.

[MP04] P. Moreels and P. Perona. Common-frame model for object recognition. *Neural Information Processing Systems Conference*, 2004.

[MP05] P. Moreels and P. Perona. Evaluation of features detectors and features descriptors based on 3d objects. *International Conference of Computer Vision*, 2005.

[MP07a] P. Moreels and P. Perona. Probabilistic coarse-to-fine object recognition. *submitted, International Journal on Computer Vision*, 2007.

[MP07b] P. Moreels and P.Perona. Evaluation of features detectors and features descriptors based on 3d objects. *International Journal of Computer Vision*, 73((3)):263–284, 2007.

[MS01] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. *International Conference on Computer Vision*, 2001.

[MS02] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. *European Conference on Computer Vision*, 2002.

[MS05] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005.

[OBV05] F. Odone, A. Barla, and A. Verri. Building kernels from binary strings for image matching. *IEEE Transactions on Image Processing*, 2005.

[OFPA04] A. Opelt, M. Fussenegger, A. Pinz, and P. Auer. Weak hypotheses and boosting for generic object detection and recognition. *European Conference on Computer Vision*, 2004.

[OPV99] O.Chapelle, P.Haffner, and V.Vapnik. Svms for histogram-based image classification. *Transactions on Neural Networks*, 1999.

[PZ98] P. Pritchett and A. Zisserman. Wide baseline stereo matching. *International Conference on Computer Vision*, 1998.

[RBK98] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20((1)):23–38, 1998.

[Rot04] F. Rothganger. 3d object modeling and recognition in photographs and video. *PhD thesis, University of Illinois, Urbana Champaign*, 2004.

[RTG00] Y. Rubner, C. Tomasi, and L.J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 2000.

[Sch99] C. Schmid. A structured probabilistic model for recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[SH05] A. Shashua and T. Hazan. Algebraic set kernels with applications to inference over local image representations. *Neural Information Processing Systems Conference*, 2005.

[Sin02] P. Sinha. Identifying perceptually significant features for recognizing faces. *Proc. SPIE*, 4662:12–21, 2002.

[SJPF04] S.Boughhorbel, J-P.Tarel, and F.Fleuret. Non-mercer kernels for svm object recognition. *British Machine Vision Conference*, 2004.

[SK00] H. Schneiderman and T. Kanade. Neural network-based face detection. *IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[SLL02] S. Se, D.G. Lowe, and J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *International Journal of Robotics Research*, 21((8)):735–738, 2002.

[SM97] C. Schmid and R. Mohr. Local greyvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19((5)):530–535, 1997.

[SMB00] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37((2)):151–172, 2000.

[SSZ04] J. Sivic, F. Schaffalitzky, and A. Zisserman. Object level grouping for video shots. *European Conference on Computer Vision*, 2004.

[ST94] J. Shi and C. Tomasi. Good features to track. *IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[SW95] A. Shashua and M. Werman. On the trilinear tensor of three perspective views and its underlying geometry. *International Conference on Computer Vision*, 1995.

[SZ01] F. Schaffalitzky and A. Zisserman. Viewpoint invariant texture matching and wide baseline stereo. *International Conference on Computer Vision*, pages 636–643, 2001.

[TC01] S. Tong and E. Chang. Support vector machine active learning for image retrieval. *ACM international conference on Multimedia*, 2001.

[TG00] T. Tuytelaars and L. Van Gool. Wide baseline stereo matching based on local affinely invariant regions. *British Machine Vision Conference*, 2000.

[TG04] T. Tuytelaars and L. Van Gool. Matching widely separated views based on affine invariant regions. *International Journal on Computer Vision*, 59((1)):61–85, 2004.

[TP91] M. Turk and A. Pentland. Face recognition using eigenfaces. *IEEE Conference on Computer Vision and Pattern Recognition*, 1991.

[VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[VS91] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1991.

[VYV98] L.J. Van Vliet, I.T. Young, and P.W. Verbeek. Recursive gaussian derivative filters. *International Conference on Pattern Recognition*, pages 509–514, 1998.

[WC03] C. Wallraven and B. Caputo. Recognition with local features: the kernel recipe. *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[WFKvdM97] L. Wiskott, J-M Fellous, N. Kruger, and C. von der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19((7)):775–779, 1997.

[WWP00] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. *European Conference on Computer Vision*, 2000.

[WZFF06] G. Wang, Y. Zhang, and L. Fei-Fei. Using dependent regions for object categorization in a generative framework. *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[YC99] A.L. Yuille and J.M. Coughlan. High-level and generic models for visual search: When does high level knowledge help? *IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[ZMLS05] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classifcation of texture and object categories: an in-depth study. *Tech Report, INRIA*, 2005.