# Improving Interest Point Object Recognition

by

Shaun Werkhoven
BSc (Hons)

Submitted to the School of Design, Communications & IT,
Faculty of Science & IT,
in partial fulfillment of the requirements of the degree of

Doctor of Philosophy

at the

University of Newcastle

Shaun Werkhoven
September, 2009

## STATEMENT OF ORIGINALITY

Shaun Werkhoven
September, 2009

*The eyes are the window of the mind*

*Young cat, if you keep your eyes open enough, oh, the stuff you would learn! The most wonderful stuff!*

Dr. Seuss

*Living is easy with eyes closed, misunderstanding all you see.*

John Lennon

*A major problem with AI research then was that Ph.D. dissertations were huge programs, and only the writer knew how they worked.*

Marvin Minsky

# Abstract

Vision is a fundamental ability for humans. It is essential to a wide range of activities. The ability to see underpins almost all tasks of our day to day life. It is also an ability exercised by people almost effortlessly. Yet, in spite of this it is an ability that is still poorly understood, and has been possible to reproduce in machines only to a very limited degree.

This work grows out of a belief that substantial progress is currently being made in understanding visual recognition processes. Advances in algorithms and computer power have recently resulted in clear and measurable progress in recognition performance. Many of the key advances in recognizing objects have related to recognition of key points or interest points. Such image primitives now underpin a wide array of tasks in computer vision such as object recognition, structure from motion, navigation. The object of this thesis is to find ways to improve the performance of such interest point methods.

The most popular interest point methods such as SIFT (Scale Invariant Feature Transform) consist of a descriptor, a feature detector and a standard distance metric. This thesis outlines methods whereby all of these elements can be varied to deliver higher performance in some situations. SIFT is a performance standard to which we often refer herein.

Typically, the standard Euclidean distance metric is used as a distance measure with interest points. This metric fails to take account of the specific geometric nature of the information in the descriptor vector. By varying this distance measure in a way that accounts for its geometry we show that performance improvements can be obtained. We investigate whether this can be done in an effective and computationally efficient way.

Use of sparse detectors or feature points is a mainstay of current interest point methods. Yet such an approach is questionable for class recognition since the most discriminative points may not be selected by the detector.

We therefore develop a dense interest point method, whereby interest points are calculated at every point. This requires a low dimensional descriptor to be computationally feasible. Also, we use aggressive approximate nearest neighbour methods. These dense features can be used for both point matching and class recognition, and we provide experimental results for each. These results show that it is competitive with, and in some cases superior to, traditional interest point methods.

Having formed dense descriptors, we then have a multi-dimensional quantity at every point. Each of these can be regarded as a new image and descriptors can be applied to them again. Thus we have higher level descriptors – ‘descriptors upon descriptors’. Experimental results are obtained demonstrating that this provides an improvement to matching performance.

Standard image databases are used for experiments. The application of these methods to several tasks, such as navigation (or structure from motion) and object class recognition is discussed.

## Acknowledgements

I would like to thank my supervisor Jesse Jin who has supported me for a long time now. Without him this work would not have been possible. I would also like to thank some of the other members of the research team he established such as Richard Xu and Ben Yip. I have had many good discussions with them and gained lots of ideas from them.

This work builds on the work of many of the key researchers in object recognition. In particular it builds on the interest point work of David Lowe, Krystian Mikolajczyk, Cordelia Schmid, on the 'biological model' of Thomas Serre, Lior Wolf and Tomaso Poggio, on the class recognition method of Paul Viola and Michael Jones, and on the ‘learning’ work of Stefan Hinterstoisser and Mustafa Ozuysal.

# Table of Contents

# List of Figures

# List of Tables

# Part I Introduction

# Chapter 1 Introduction

## 1.1 Motivation

The urge to understand our environment is one if the most basic instincts we have. The primary information source we use for that is the visual information presented to our eyes. These images represent an extremely rich and complex source of information. We use them to recognize objects and people, know our location, navigate and perform tasks. We also use this visual information as the base on which higher level cognition tasks are built – such as interpreting events or intentions.

In computer vision we seek to understand these visual recognition processes and replicate them on a computer. To do this has been the goal of researchers in computer vision for over a generation now; yet it is still a goal that is far from completion. Within computer vision there are a range of challenges that are generally treated with quite different methods. Some of these topics are 3-dimensional reconstruction, stereo matching and object recognition. Object recognition is thus considered one of the primary problems in computer vision and is the topic of this thesis.

Automatic object recognition is gaining in importance as more and more information becomes available online and in other digital forums. The proliferation of digital cameras and camcorders ensures that this process will only increase.

Object recognition tasks can be treated as falling into 2 separate parts – specific object recognition and object class recognition. Both types of recognition have proven to be very challenging to achieve, due to the viewing variations that may exist between images. Some of these variations include

- scale changes from viewing distance change

- viewing perspective (this is sometimes approximated by an affine or projective transform)
- changes within an object class, i.e. intra-class variation, such as changes in color or texture
- occlusion or blocking from other objects
- clutter from background or other objects
- illumination or lighting variation
- image resolution or blur inconsistency due to the image acquisition technology

Some of these variations are illustrated in Figures 1-1 and 1-2.

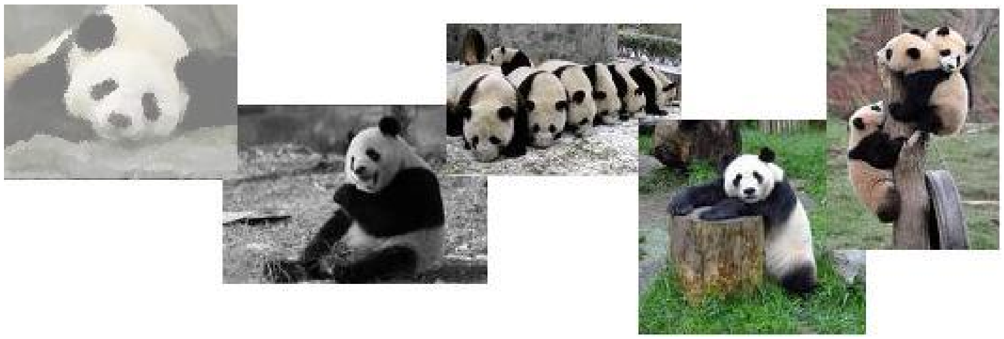


Figure 1-1: Examples of images of the same class.
These images illustrate some of the range of shape and color variation that can occur for images of a class.

The human vision system seems to possess complex methods to deal with such variations easily. However, reproducing such performance artificially has proved much more difficult. Computers vision still lags far behind human vision on most recognition tasks.

Over the last 10 years interest point methods have become very popular in computer vision and have been widely applied to both specific object and class recognition. Interest points are currently the major method for non-class object recognition, structure from motion and navigation. Improvement in basic image primitives has generally brought improvements to a wide array of tasks in computer vision.

For simple point-to-point matching, or object specific recognition, matching rates of up to 40% are now consistently achievable based on a single point-region (point and connected region surrounding it). Thus very high object matching certainty (e.g. > 95%) can be achieved by a small group of points. Moreover this rate is achievable for hundreds or thousands of points in an image. For class recognition, the performance achieved depends greatly on the degree of variation in the class, and the amount of training information available. For relatively simple classes, such as faces, matching rates of ~90% are achievable under good conditions. For more complex classes matching rates of only 30-40% are possible, and this normally requires large numbers of points or the entire image information. The variability within object classes results in a great reduction in performance.

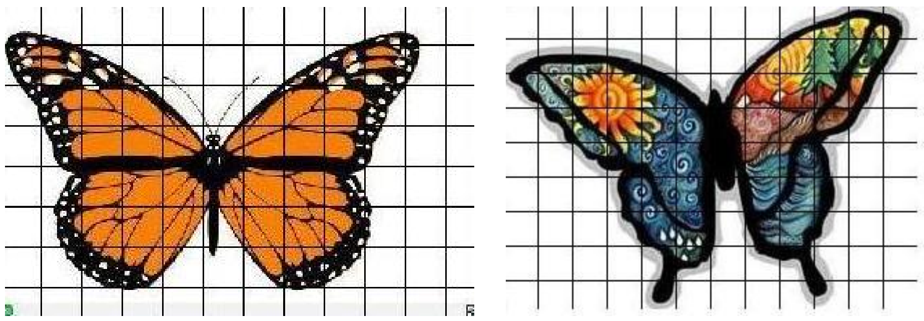


Figure 1-2: Even for quite similar class images, corresponding image regions may be quite different

The main goal of this thesis is to improve the performance of interest point recognition methods. We pursue 3 main strategies – finding new matching metrics, finding better points and finding new point descriptors. The purpose of these is to increase the robustness and flexibility of the final descriptor to give it a higher performance.

Successful recognition need to strike a balance between specificity and flexibility – specificity to distinguish between points, and flexibility to allow for viewing and within class variation. The most popular interest point method for some time now is the SIFT descriptor [62, 63]. For a typical matching task on a standard dataset (e.g. the Oxford set) SIFT can obtain a point-to-point matching rate of ~ 30%. This is the benchmark against which much of this thesis is measured. For low dimensional descriptors, steerable filters [36] can be used as a benchmark since they have achieved excellent performance. Gabor wavelet filters are similar to steerable filters and we use them extensively. For a review of interest point matching on this image set see [73].

**What's New in This Thesis?**

- A 'geometric match' descriptor distance which takes account of the geometric setting of an interest point descriptor in a way that the Euclidean metric does not. Experimental results show that this provides a clear improvement in matching performance for correlation and SIFT descriptors, compared to the Euclidean distance.

- A dense detector that does not rely on Difference of Gaussian (DoG) or Laplace of Gaussian (LoG). It is therefore much more flexible and suitable for object class recognition. Experimental results show that this provides an interest point with good class recognition capability for some classes. It brings some of the power of the Viola-Jones method to a general interest point setting. It also gives a high performance, low dimensional descriptor that performs competitively with the best low dimensional point matching descriptors.

- A multi-level descriptor or 'descriptor upon descriptor'. This differs from traditional descriptors which could be described as 'descriptor upon pixels'. Experimental results that this allows a substantial increase in matching rate compared to traditional single-level descriptors

- Experimental results show that several of these methods outperform previously documented methods for some recognition tasks.

## 1.2 Thesis Outline

After introductory comments the thesis begins by discussing previous object recognition methods and background ideas. This work builds on these previous ideas. Some key ones are that of a detector, a descriptor, scale space and scale invariance, class discrimination, Gabor wavelets and stereo matching methods.

The main body of the thesis then begins with the introduction of the geometric match concept. This is then applied to several different descriptors and tested on standard image sets. We see an improvement in matching performance.

We then introduce the dense detector concept. We choose a suitable flexible detector and discuss how it can be used for point-to-point or object specific matching. We test this on standard image sets and show its performance is superior to that of traditional detector methods while being more flexible. From this we move to the topic of class recognition which is the original purpose for which it is developed. The implementation for class recognition has some differences from that of point matching and we discuss this. Discussion of why and how it was designed is conducted.  We show its performance on standard databases and compare that to other class recognition methods.

Then we discuss the concept of a 'descriptor on a descriptor' and how this improves point-to-point-matching.

We then show example applications involving camera navigation and class determination.

Finally we conclude with a discussion, noting its strengths and limitations, and future improvements that can be made to it. Thesis results and summary are available online at www.visionwurx.com/PhD .

## 1.3 Sketch of 3 Main Concepts

- Concept 1: Geometric match distance: this differs from the standard Euclidean distance and takes more account of the geometric setting of the descriptor. In computer vision feature data is often provided as a vector of values. Instead of simply matching corresponding values in a feature vector, the geometric match tries matching to the spatial neighbors of each feature vector entry. This is similar to established stereo methods. These are generally not the sequential neighbors of the data in the vector. Thus, knowledge of the geometric layout of the data and vector arrangement is necessary for correct matching. See Chapter 3 for further information.

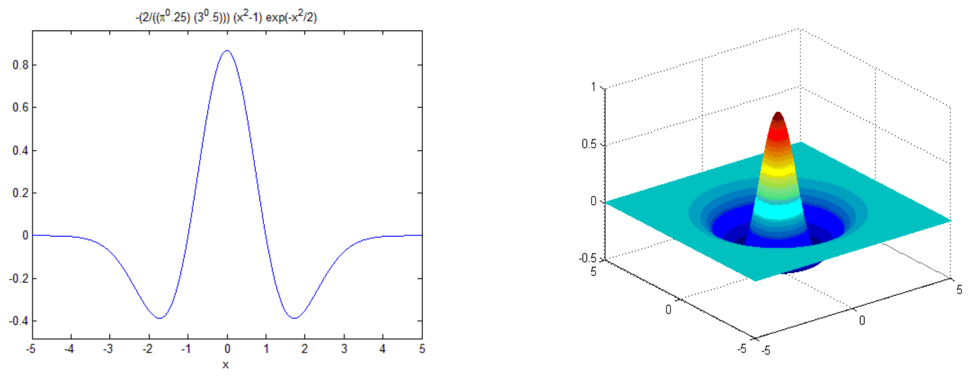
- Concept 2: The dense point detector: This differs from DoG (Difference of Gaussian) and LoG (Laplace of Gaussian) and provides a more flexible point detection framework. DoG and LoG are scale-space based detectors; we believe that only a descriptor-based point detector gives high performance for class recognition. Therefore for each image pixel we calculate the descriptor value, and perform a lookup into the database space of points. The key aspect is descriptors are calculated for all pixels, and that points can be enrolled in the database by any criteria. This increases flexibility and is asymmetric in the processes of point selection and recognition. The final detector is has some features of the traditional detector framework, and the Viola-Jones detector concept. See Chapter 7 for further information.

- Concept 3: Multi-level descriptors: Although many different descriptors have been previously proposed, there is one thing that all these descriptors have in common: they are all applied directly to image pixel data. Here we present the idea of multi-level descriptors, where the lowest level is constructed from image pixels as usual, but each successive level is constructed from the previous level of interest points. This has 2 objectives: to allow more robustness in the descriptor, and to form 'meta-descriptors' that are aggregations of normal descriptors. See Chapter 9 for further information.

# Chapter 2 Background and Related Work

## 2.1 Early Object Recognition Ideas

Formal work in computer vision began in the 1960's and developed slowly initially. Perhaps the earliest formal work, of which this thesis can be said to be descended, is the work of Marr [65] on edge extraction. This used 'Mexican hat' wavelets (see Figure 2-1); thus the usage of wavelets in computer vision is already very old. This work was primarily concerned with edge detection rather than object recognition *per se*, but demonstrated the use of wavelet convolution to achieve vision tasks.



(a) (b)

Figure 2-1: Example of Gabor wavelet
(a) a 1-D Gabor wavelet; (b) a 2-D Gabor wavelet.

In the mid to late 1980's the concept of alignment was proposed [47]. This is based on matching simple point sets e.g. triplets of points, or 3 line directions and a corner point (Figure 2-2). Such methods do not possess high robustness to changing viewing conditions. However, they do demonstrate the use of local features for recognition.

Figure 2-2: Example of line extraction on which typical 'alignment' methods worked.

Other concepts that became popular at that time were indexing by hash tables and pose clustering. This consists of hypothesizing pose changes between a given object and a model object. For each hypothesis a vote for that pose is made, and the pose with the majority is adopted. In some sense, this idea has remained widely used in computer vision, e.g. in such algorithms as the Hough line detector and RANSAC. A practical difficulty is to identify robust image primitives to form the initial pose hypothesis.

In the early 1990's ideas of geometric invariance became prominent in vision [76]. These have their origin in mathematical invariants of geometric systems. For example, under affine transforms the ratio of distances between points is invariant; under projective transforms the cross ratio is preserved; under Lorentz transforms, the Minkowski metric remains invariant. After an initial burst of enthusiasm, such methods have been found to have only restricted usage. This is primarily because such invariants only exist in planar spaces, and in the difficulty in reliably identifying point sets and lines.

It is also possible to perform matching using contours and contour chain approximations, as was illustrated in [35]. Around the same time it was shown that color histograms are also effective in some situations [100]. Such methods generally rely on good quality segmentation and are therefore not robust against occlusion; also they lack high invariance against viewing changes.

Another influential object recognition method is the eigenface or eigenobject method [105] (Figure 2-3). This method relies on the consistent correlation structure between pixels in an image window, and performs Principal Component Analysis (PCA) on that structure. It also depends on good global segmentation and therefore is not robust against occlusion. Also, principal components relate to variation, not discrimination; recognition is more closely related to discrimination.

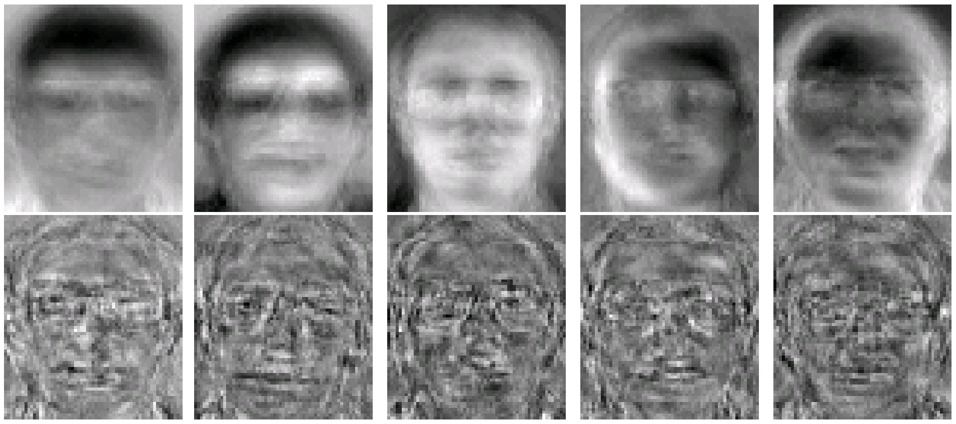


Figure 2-3: The first 10 'Eigenfaces' extracted by using PCA on pixel correlation.
(from Google Images)

## 2.2 Scale Space

Scale space is a concept that dates from the 1980's [112], was developed by Lindeberg [60] and has remained of key importance. The scale space of an image is the representation of the image under convolution kernels of increasing size (see Figure 2-4). It can be shown from mathematical consistency (e.g. the semi-group property) that the only possible kernel is the Gaussian kernel. By decimating the images, the information to pixel ratio is maintained and a pyramid is obtained

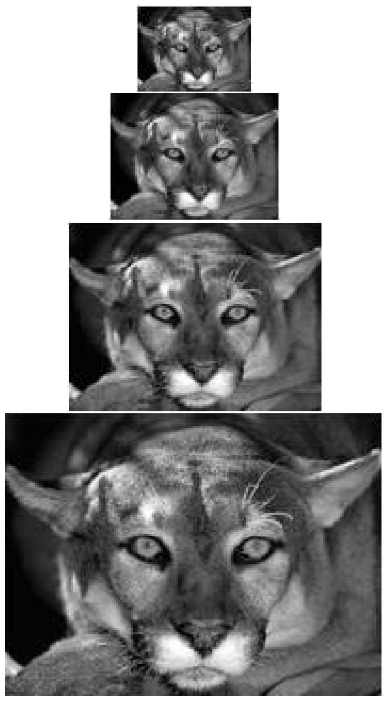


Figure 2-4: Example of scale space.
To maintain a proper information:pixel ratio at different sizes,
Gaussian filtering and decimation must occur.

For practical object recognition systems, the scale space concept determines that we must perform object recognition on all the images of the scale space pyramid.

## 2.3 Interest Points

If we could identify propitious points in the image to apply further recognition processing, this would have several benefits

- Matching on subsets of image points is computationally more efficient than matching all points
- Matching a subset reduces the false positive match number
- A subset may have desirable properties e.g. be more stable or discriminating.

The challenge of this paradigm is to be able to identify points in a reliable and robust manner. Specifically we seek a means to identify the same points under the image transformations listed in Sec. 1.1.

A method to identify keypoints or interest points was outlined by Harris [42] in the context of navigation. This was based on corner points and used a maxima function of the 2nd moment quantity (see Sec. 2.3.3). This point detector was not scale invariant and therefore did not perform well under practical situations which involve scale change. Nonetheless, it demonstrated that reliable point extraction was possible at modest computation (see Figure 2-5).

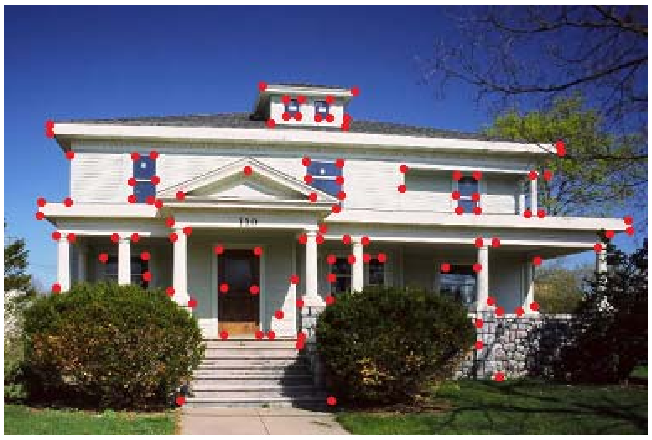


Figure 2-5: Image corner points, such as those returned by the Harris operator.

The next major advance that occurred was the concept of the interest point descriptor [88]. This descriptor was a 'grayvalue' vector of Gaussian convolution derivates around a point (see Sec. 2.3.3). This provided a point descriptor but did not present a means for reliable detection within scale space (under scale variations). The descriptor is given as a vector of values, which was adopted by later approaches-

$$\text{Descriptor} = (u_0, u_2, u_3, \ldots, u_n)$$

### 2.3.1 Scale Invariant Feature Transform (SIFT)

A breakthrough occurred in 1999 with Lowe's [62, 63] conception of the Scale Invariant Image Transform (SIFT). This presented a simple method whereby points could be repeatably located within scale-space (space and scale). It also presented a descriptor for interest point matching which has been proven to be more robust to image transforms than almost any other descriptor.

SIFT Detector: The SIFT scheme uses maxima of the Difference of Gaussians (DoG) as the point detector. Gaussian kernels of different sizes are convolved to obtain the image at different scales. These are then subtracted from one another. Those points that are maxima compared to their 3×3 image neighbors, and also to their neighbors in the adjacent scaled images are selected (see Figure 2.6). A quadratic fit through space and scale is used to find the maxima. Points of low contrast and near edges are rejected.

In a typical 500×500 pixel image, this could extract of the order of 1000 points. Whereas the Harris detector finds corner points, the DoG tends to find 'blob' like points.

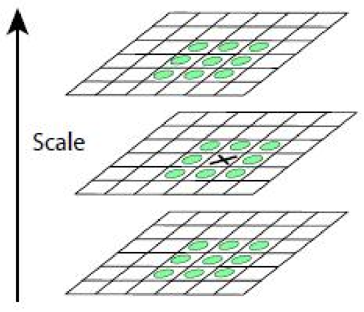


Figure 2-6: Multi-scale DoG scheme.
The image is obtained at different scales, and those pixels selected which are maxima of differences in space and scale (from [63]).

SIFT Descriptor: The SIFT descriptor divides the pixel region around a point into 4×4 sub-regions. At each pixel a gradient direction and value is computed. Typically 8 orientations are used. Within each of the 16 regions, a histogram of these gradient orientations is formed (see Figure 2-7). Thus a histogram vector of 4×4×8 = 128 dimensions is obtained.

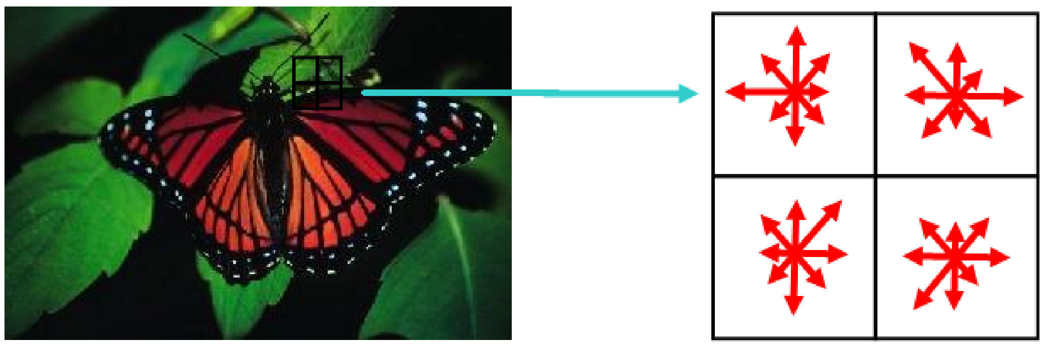


Figure 2-7: SIFT descriptor.
The SIFT descriptor divides the area around a point into a number of spatial regions, and then obtains orientation histograms for each bin. Although this diagram has 2×2 spatial regions, SIFT usually uses 4×4.

Additionally the maximum value of a histogram of gradient directions around the keypoint is used to allocate an orientation for the point.

Operation: In applications the detector is used to extract keypoints from an image, and descriptors are computed for each point. These are then stored in a database. Then for a new test image, the same detector and descriptor application is made i.e. there is a symmetry between point selection in the initial point enrolment stage, and in the image recognition stage. The selected point descriptors are then matched to the database by Euclidean nearest neighbour matching ($L_2$ norm, see Sec. 2.7). This has been found to have excellent robustness and matching performance in [70].

In recent years SIFT has become very widely used in computer vision, for tasks such as object recognition [33, 62], robot navigation [91], structure from motion [3], panorama stitching [17].

## 2.3.2 Affine Invariant Interest Points

The SIFT scheme provides for invariance to x-y translational motion, scale change (movement in z direction), rotation in the x-y plane but there are other possible viewing changes. In particular SIFT does not allow for affine deformation in the x-y plane. Such an affine transform corresponds to the transform of a plane under an affine projection model; it therefore approximates many local viewing changes.

A method to adapt the detector/descriptor framework to allow for affine changes was demonstrated in [69], based on ideas of [61]. This involves repeated interest point detection and image affine adaption. It is not clear that affine covariance is worthwhile since it involves increased computation and delivers modest improvements in matching. For a typical application on standard images (Oxford database [2]) it could increase the matching performance from ~30% to ~37%.

Other affine invariant methods are [9, 41, 56, 74, 85, 86]. More recent affine invariant ideas include [106, 101, 84].

## 2.3.3 Other and Recent Detectors & Descriptors

Following the success of SIFT, a host of other invariant interest point schemes were proposed. These retain the dichotomy of first using a detector to select points, then applying a descriptor to the region around them. These detectors have received widespread use in object recognition. A key characteristic of them is that they select points without regard to class discriminative properties.

**Detectors:** Some interest point detectors are-

a) Difference of Gaussian (DoG) [62]. See Sec. 2.3.1

b) Laplace of Gaussian (LoG) [68]. This identifies points which are maxima of the quantity $\left|s^2\left(L_{xx}(x,s)+L_{yy}(x,s)\right)\right|$ where s is scale.

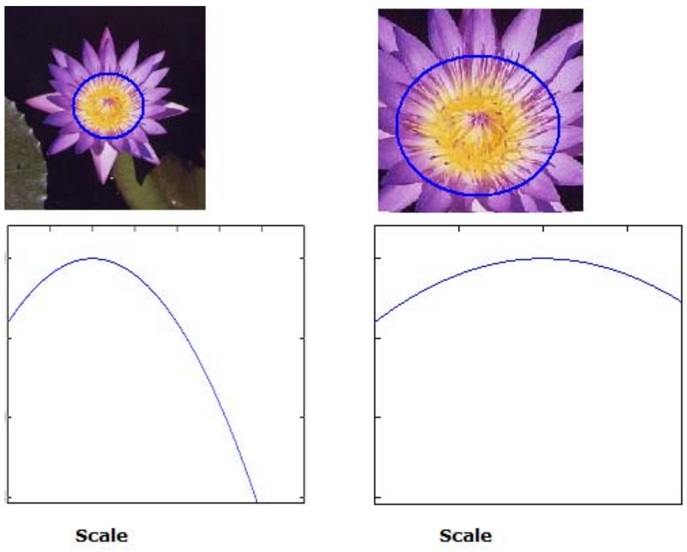


Figure 2-8: The Laplacian of Gaussian (LoG) at a pixel.
Taken at different scales provides the same profile or shape but multiplied by a relative scale factor.

The LoG is a very stable detector and is similar to the DoG. For this reason we will refer to it. Figure 2-8 illustrates the key property of scale invariance – that the maxima are preserved under a scale change.

c) Harris or Hessian matrix detector [42]

$$M = \mu(x, \sigma_I, \sigma_D) = \begin{bmatrix} \mu_{11} & \mu_{12} \\ \mu_{21} & \mu_{22} \end{bmatrix} = \sigma_D^2 g(\sigma_I) * \begin{bmatrix} I_x^2(x, \sigma_D) & I_x I_y(x, \sigma_D) \\ I_x I_y(x, \sigma_D) & I_y^2(x, \sigma_D) \end{bmatrix} \quad (2.1)$$

where $g$ is the Gaussian function.

This is the 2nd moment matrix of a point region. The Harris detector seeks points that maximize the difference between the determinant and the trace$^2$ of this 2nd moment matrix.
A similar detector can be made instead using the Hessian matrix which uses 2nd derivatives

$$H = L(x, \sigma_I, \sigma_D) = \begin{bmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{bmatrix} = \sigma_D^2 g(\sigma_I) * \begin{bmatrix} L_{xx}(x, \sigma_D) & L_{xy}(x, \sigma_D) \\ L_{xy}(x, \sigma_D) & L_{yy}(x, \sigma_D) \end{bmatrix}$$

d) Harris-Affine or Hessian-Affine [69]. The Harris affine detector uses Harris points for initialization but then also uses the M matrix (above) for affine region normalization. This process is applied iteratively until convergence. The Hessian affine method is similar but uses the Hessian matrix instead.

e) Entropy or salient based detector [52, 53]

$$H(x,s) = -\sum_I p(x,s) \log p(x,s) \qquad (2.2)$$

This detector is based on ideas from information theory. It calculates (2.2) at all points and some scales. This is then multiplied by the gradient of *p*, and maximum points are selected. Another entropy based method was described in [57].

f) FAST (Features from Accelerated Segment Test) [83]. This detector considers the distribution of pixel brightness in a circle around a point. If *n* contiguous pixels are sufficiently brighter or darker than the centre point it is considered an interest point. It is a fast detector and is useful for navigation.

g) Maximally Stable Extremal Regions [67]. This detector looks for those point regions which are most stable when the image is thresholded at continuously increasing values.

**Descriptors:** Some point descriptors are-

a) SIFT [62]. See Sec. 2.3.1.

b) Steerable filters [36], Differential Invariants [34], grayvalue [87]. These involve computing the *local jet* which is the set of local Gaussian derivatives up to a certain order. These are then can be made rotation invariant or to vary in a predictable way with rotation change. See Figure 2-9 (a).

$$\mathbf{v}(x,y) = \begin{pmatrix} I(x,y) * G(\sigma) \\ I(x,y) * G_x(\sigma) \\ I(x,y) * G_y(\sigma) \\ I(x,y) * G_{xx}(\sigma) \\ I(x,y) * G_{xy}(\sigma) \\ I(x,y) * G_{yy}(\sigma) \\ \mathrm{M} \end{pmatrix} \tag{2.3}$$

G is the Gaussian function.

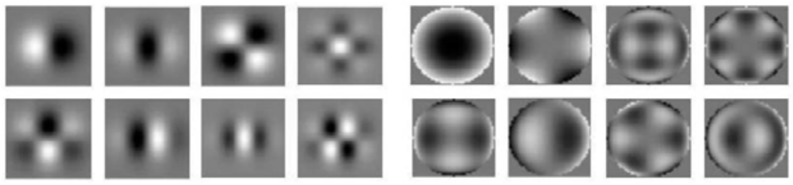

(a) (b)

Figure 2-9: Derivative based filters

(a) Gaussian jet derivatives and (b) complex filters

(from [70]).

c) Histograms [59, 100]. Histograms are a simple statistical descriptor of a set of data. They have been found to be useful for recognition in many cases, particularly involving color or texture.

d) Moment Invariants [107].

$$M_{pq}^{a} = \frac{1}{xy} \sum_{x,y} x^{p} y^{q} \left[ I_d \left( x, y \right) \right]^{a} \tag{2.4}$$

These give a signature of the shape and intensity around the point. They have been used are more effectively with color images.

e) PCA SIFT [54]. This combines the SIFT descriptor with a subsequent application of Principal Component Analysis (PCA). This can reduce the dimensionality of the descriptor from 128 to ~30 dimensions; however it is not clear that it consistently improves matching performance.

f) Shape Context [11]. This creates a histogram of edge points in a log-polar space around the point (Figure 2-10). It has been shown to be effective for OCR (optical character recognition).

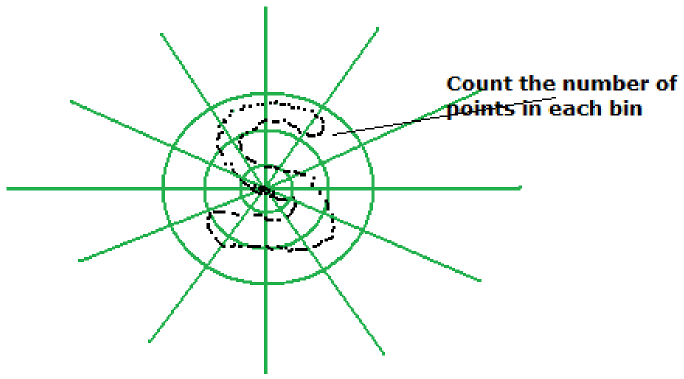


Figure 2-10: Example of shape context construction.

g) GLOH (Gradient Location and Orientation Histogram) [70]. This is similar to SIFT but uses a log-polar space and is of higher dimension.

h) SURF (Speeded Up Robust Features) [10]. This is a fast descriptor that uses Haar wavelets and gives high matching performance. This has become popular and has an OpenCV implementation.

i) HoG (Histogram of Gradient) [24]. This calculates a histogram of gradients but does so using a dense grid of cells and some overlap.

j) DAISY (Dense rapidly computed Gaussian scale variant gradients) [104].

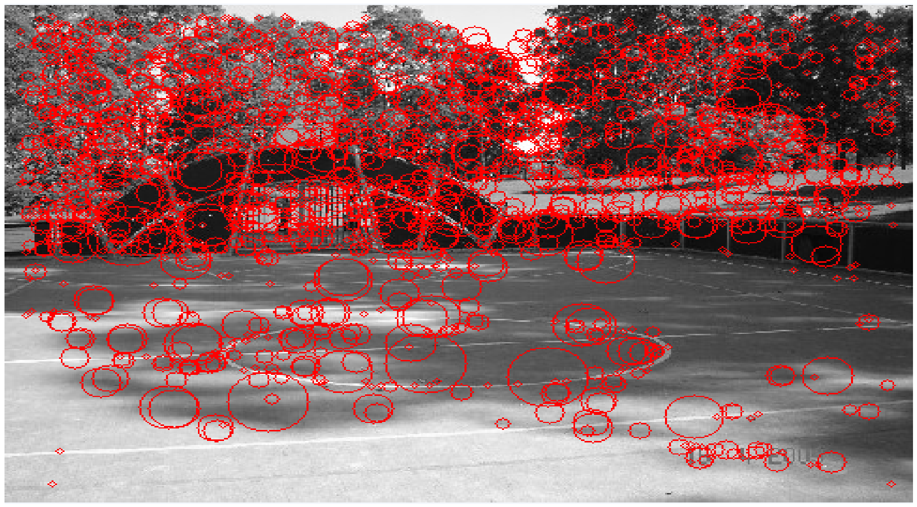
k) SUSAN (Smallest Univalue Segment Assimilating Nucleus) [98]. This forms a circle around a point and for every other point within the circle $c(m) = e^{\frac{(I(m)-I(m_0))^6}{t}}$ (t is the radius) is calculated and summed. A further linear function of this is used to form the final descriptor.

l) Cross Correlation. This is the simplest descriptor, consisting of a Gaussian blur followed by a sub-sampling. This is very efficient since this will normally computed anyway for scale invariance.

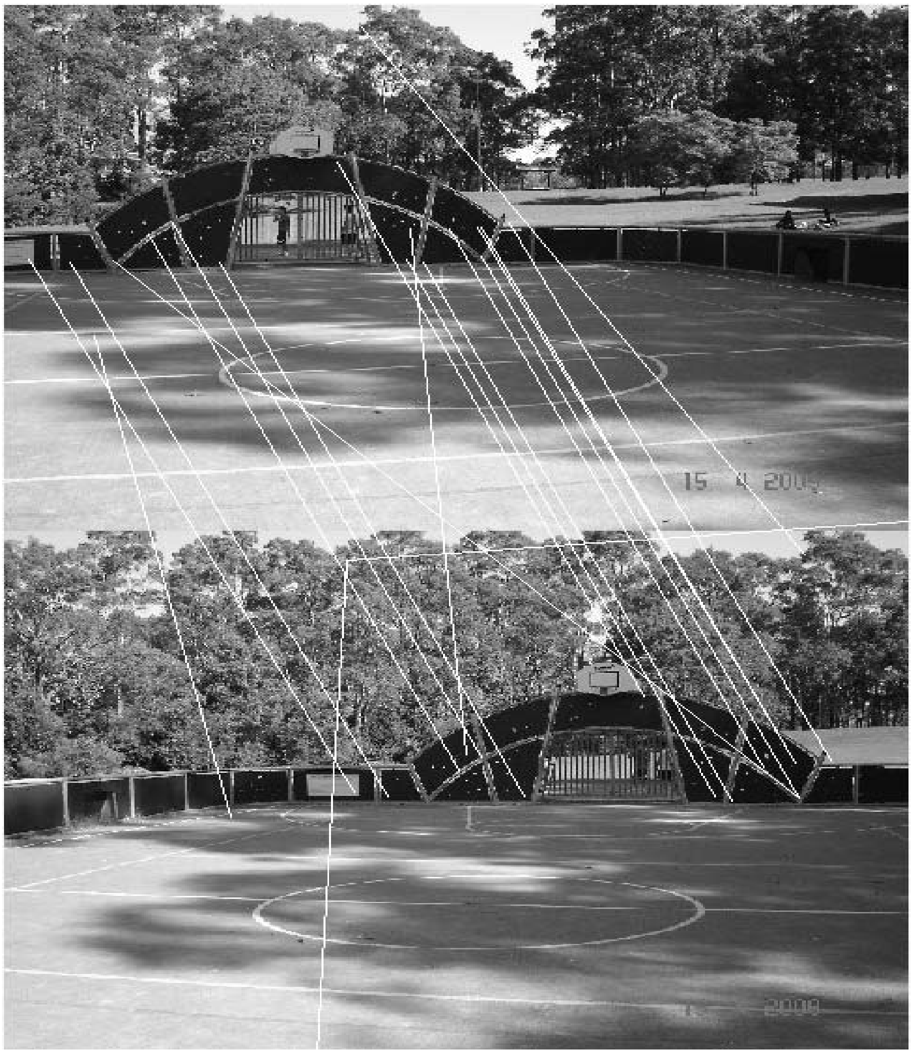
m) Gabor wavelets [38]. See Sec. 2.3.4.

Other notable interest point methods are explored in [13, 14, 55, 82, 84, 24, 25, 31].

In general these methods only offer modest improvement, or only better in some situations, compared to SIFT. In the majority of cases they are inferior. Thus since SIFT was first described only incremental improvement in point-to-point matching capability has taken place. An example of point-to-point matching is given in Figure 2-11.



(a)



(b)

Figure 2-11: Mismatching problem.
(a) An example of interest point extraction or enrolment, from an image and (b) recognition or matching to another image. This example indicates that many enrolled points may not have a match or may be matched incorrectly.

## 2.3.4 Gabor Wavelets

Gabor wavelets are used extensively in this thesis therefore we will describe them separately. They have extensive uses in general signal processing and can also be used as a type of descriptor.

For image processing we are generally concerned with 2-dimensional Gabor wavelets. Such a 2-dimensional Gabor wavelet is a function described by the equation

$$Gabor(x,y) = \exp\left(-\frac{\left(U^2 + \gamma V^2\right)}{2\sigma^2}\right) \times \exp\left(j\frac{2\pi}{\lambda}U\right) \tag{2.5}$$

$$U = x\cos\theta + y\sin\theta$$
$$V = -x\sin\theta + y\cos\theta$$

where $\sigma$ is the Gaussian envelope sigma, $\gamma$ is the skew, $\theta$ is the orientation and $\lambda$ is the wavelength. Normally we only use the real part of this.

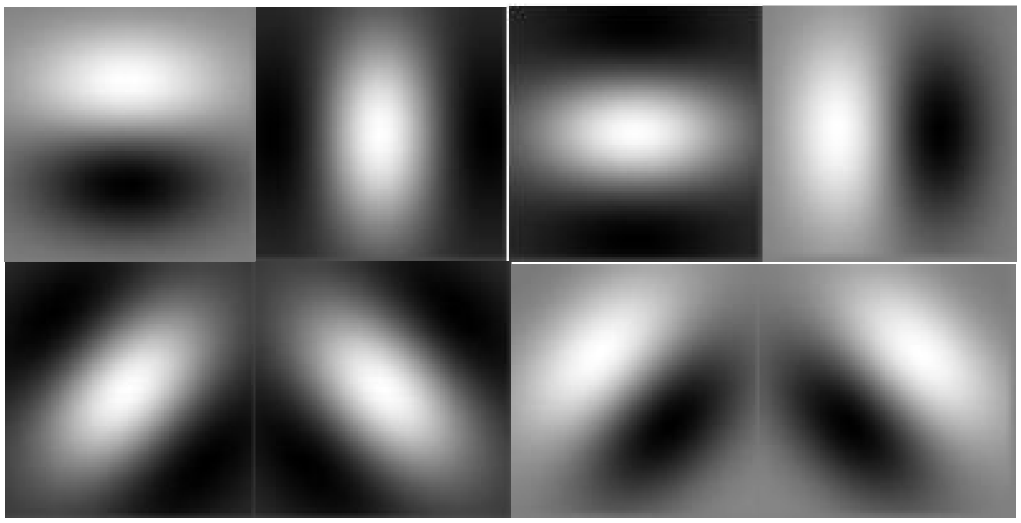


Figure 2-12: A set of 2-D Gabor wavelets.
This example has 4 orientations and 2 phases.

The Gabor wavelet has been adopted in this work for several reasons

- It has been found to be used within the human visual system at a low level
- It achieves the maximum resolution of both spatial and frequency information i.e it satisfies the Heisenberg condition

$\sigma_x \sigma_\omega \geq \frac{1}{4}$ where $\sigma_x$ is the spatial variance and $\sigma_\omega$ is the frequency variation [64].

- The Gabor kernel has been shown to optimize some information theoretic criteria related to the extraction of information from an image [80].
- Gabor wavelets have achieved excellent results in practice [95]. For example, in the Face Verification Competition FVC2004, the top 2 methods used Gabor wavelets.

For a practical descriptor, Gabor wavelets are applied as a filter bank, at different orientations, phase and frequencies. For example, in part of this work we use a bank of 16 Gabor wavelets at 4 orientations, 2 phases and 2 frequencies. If the entire image is convolved with these, a 16 dimensional descriptor is obtained at each point. A set of Gabor wavelets is shown in Figure 2-12.

One useful property of this function is that it is separable. This can be seen in that Eq. (2.5) can be written as f($U$)g($V$). Separability is useful in accelerating the numerical implementation of convolution, since it allows each dimension to be convolved separately. In practical image processing it can speed up the convolution by a factor of ~ 4-5.

### 2.3.5 Interest Point Learning and Class Recognition Techniques

Recently some learning techniques related to interest points have been shown to offer some substantial improvements in performance, at the cost of greater prior processing. These involve fitting a classifier to the points [45, 81]. In this thesis we are interested in methods that do not require extensive pre-training.

Some interest point methods have explicitly used groups of points or meta-descriptors rather than individual points as the matching object [18, 23, 20, 40].

There has also been a lot of work on adapting interest points for class recognition. These include [30, 32, 51, 58, 72, 71, 90, 96, 79, 114, 77, 109, 103, 15].

## 2.4 The 'Biological' Model

A variation to the standard Detector/Descriptor framework is the 'biological' model developed over 10 years at MIT [93]. It is so-called because it explicitly attempts to model visual processing in the primate visual cortex area of the brain. Since it has influenced this work we will describe it in some detail.

This model involves

- Applying a bank of Gabor wavelets of 4 different orientations and a range of different scales
- 2 adjacent scales are treated as a 'band'
- In each band, the maximum of an 8×8 pixel region, and across each of the 2 images in the band, is selected. These are called the C1 points.
- Previously a set of k patches, $P_i$, of varying sizes $n_i \times n_i$ $i = 1...k$ has been selected from training images
- For each of the C1 points, let X be the pixel region around it. We then calculate $Y = \exp\left(-\kappa \|X - P_i\|^2\right)$ for each $P_i$ .These then are the S2 maps.
- For each training patch $P_i$ find the maximum over all positions and scales. We thereby obtain k C2 features, for a final k-dimensional feature vector.

Compared to the descriptors described earlier this model has some advantages for class recognition has been shown to outperform SIFT points for this purpose. This is partially because the maxima operation allows the shape to vary over a greater region, and partially because it does not rely on scale space based detectors (e.g. DoG). As we will see in Chapter 9, this can be considered to be a Level 2 descriptor. The first level is the Gabor descriptor, the second is the maxima operation.

## 2.5 Viola-Jones Object Detection Framework

A fast object detection framework was described by Viola and Jones [108] in 2001. This was proposed in the context of face detection although it can be generalized to a wide range of classes. For single object class detection applications it generally achieves faster, more accurate operation than any previously described methods. In particular, it differs fundamentally from the detector/descriptor framework and can outperform it for class recognition tasks. The detector method presented in Part III attempts to bridge some aspects of the gap between the Viola-Jones method and the detector/descriptor framework.

The Viola-Jones method uses features similar to Haar wavelets as its geometric primitives (see Figure 2-13).

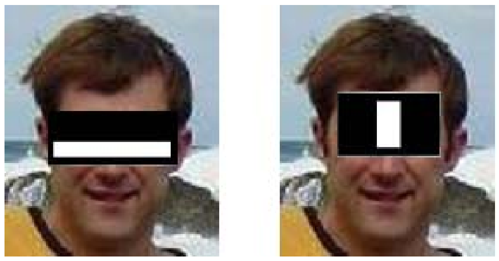


Figure 2-13: An example of face detection using Viola-Jones filters.

Using these primitives of different sizes and positions within a window of 24×24 pixels provides for 45,396 possible features. A learning algorithm, Adaboost is used to select the most discriminative ones for a given object class. Using these discriminative features, a linear threshold is used as the classifier function i.e. a lookup into a 1-dimensional space is performed for discrimination. This process is applied efficiently by using a cascading structure, with the most discriminative features being applied first.

Such a detector applies a descriptor, then ‘looksup’ its value into a space to determine acceptance or rejection. It does so for every point in the image. Thus it is a *dense* point method, unlike DoG or LoG methods (such as SIFT) which use sparse points in the image. We will call such a method a ‘dense point detector’. It is a descriptor-based detector, unlike DoG or LoG for example, which use a quantity extraneous to the descriptor as detector.

## 2.6 Testing and Performance Measurement

To determine how effective an object recognition method is, it is critical to have clear, relevant and objective standards of performance assessment. For interest points, some assessment methodologies were developed very soon after they were conceived. We use similar criteria to those developed in [21], [70] and [73]. These relate to interest point performance between 2 images that have a known relationship e.g. a homography, *H*.

The 2 main indices used are repeatability, which is a measure of detector consistency, and the matching score, which is a measure of descriptor performance. Herein we will make more use of the matching score since we are concerned with descriptor performance. We will also use ROC graphs, and recall-precision graphs. These are also measures of descriptor matching performance.

Suppose interest point detectors have been applied to 2 images with a known homography *H* between the 2 images. Thus a set of interest points exist for each image. Figure 2.12 provides an example of a pair of images with known homography that have interest points extracted. Then-

a) $$\text{repeatability} = \frac{\#\text{corresponding points}}{\min(\#\text{int. points image1}, \#\text{int. points image2})} \tag{2.6}$$

The repeatability is a measure of detector consistency or reliability. It is the proportion of points from the 1st image that are in the corresponding position in the 2nd. We regard points from each image as corresponding if, for positions *a* and *b*, $\|a - Hb\| < 4$.

b) $$\text{matching score} = \frac{\#\,\text{correct matches}}{\min(\#\,\text{int. points image1}, \#\,\text{int. points image2})} \qquad (2.7)$$

The simplest matching measure between the 2 image sets of interest points is by the Euclidean nearest neighbour. A correct match is one for which $\|a - Hb\| < 4$. Note that this measure relates matches to the minimum number of interest points in the images, rather than the number in the original image. Thus in some sense it gives a misleadingly positive indication of matching.

c) $$\text{recall} = \frac{\#\,\text{correct matches}}{\#\,\text{corresponding points}} \qquad (2.8)$$

This is another measure of matching performance. It compares matches to the number of correspondences and therefore gives a descriptor matching indicator more independent of the detector.

d) $$1\text{-precision} = \frac{\#\,\text{false matches}}{\#\,\text{correct matches} + \#\,\text{false matches}} \qquad (2.10)$$

The precision is a measure of how likely a descriptor match is to be correct, compared to all matches.

Recall is generally graphed against 1-precision. In this case, rather than use the nearest neighbour, a match is defined as
$(desc_a\text{-}desc_b) < d$. *d* is allowed to vary to generate the graph.

e) Receiver Operating Characteristic (ROC) is a graph of the true positive rate vs. the false positive rate. The true positive rate is the same as the recall. It is more easily used as a metric for class recognition, rather than point-to-point matching, and this is how we use it here.

f) $$\text{matching rate} = \frac{\#\,\text{correct matches}}{\#\text{int. points image1}} \qquad (2.11)$$

Compared to the matching score, the matching rate gives a more complete measure of the matching system. It encompasses not only the effect of the descriptor matching, but also the effect of the detector. It is therefore more a measure of the whole *system*, rather than the descriptor *per se*.

### 2.6.1 Image Data Sets

The main datasets used in this work are –

- the Oxford affine features dataset [2] at http://www.robots.ox.ac.uk/~vgg/research/affine/index.html
- the Caltech classes dataset [1] at http://www.vision.caltech.edu/Image_Datasets

We make considerable use of these since they facilitate comparison with other published work. Several examples of these images are given in Figures 2-14 and 2-15, more complete examples are given in Appendix I.

The Oxford set is well established to compare performance of point-to-point matching. Some Matlab scripts and homography calculation tools are available for it. It has sequences of image variation by viewpoint change, scale, blur, illumination, and JPEG compression. It draws a distinction between more 'textured' images and more 'structured' images. The Caltech databases consist of collections of images of various object classes; it is widely used to assess class recognition performance. It includes the 'Caltech 101' dataset of 101 object classes, and other individual class collections. We also obtained a small number of images from other sources, such as Google Images. These are ascribed where used.

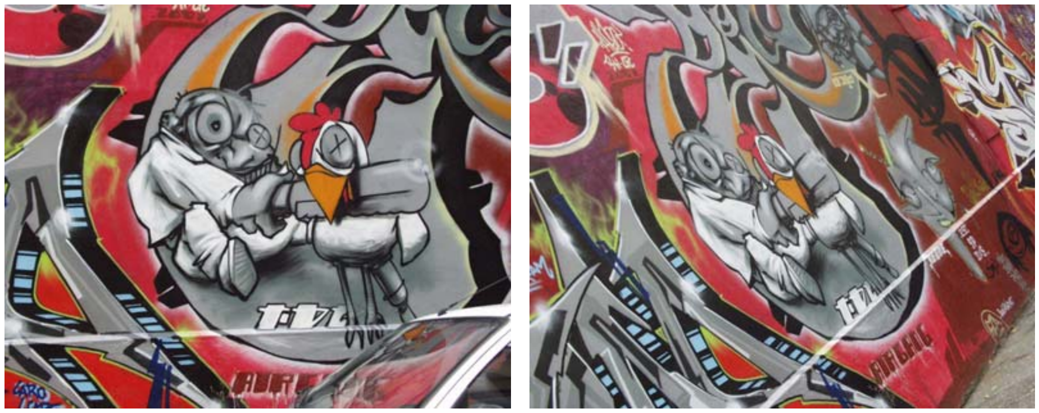


Figure 2-14: Oxford graffiti wall sequence – structured viewpoint change.
These are 2 examples from this sequence of 6 images which show a graffiti wall from differing viewpoint positions. This is considered to be a 'structured' type of image, rather than a 'textured' one. The matching rate is the number of correct interest points detected in the 2nd image relative to the 1st. Correctness is determined by the homography between the images.

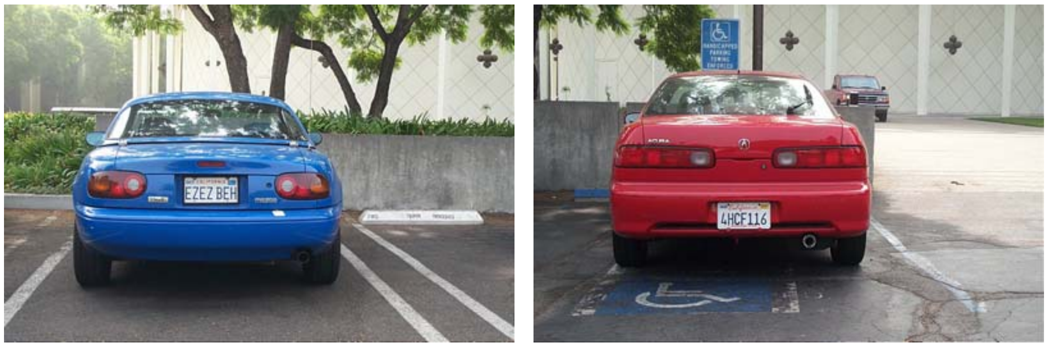


Figure 2-15: Caltech Cars Collection.
These are 2 examples from the 'cars_markus' collection. Although to human eyes these images look very similar, even such small changes can present problems for matching algorithms.

## 2.7 Distance Measures

When features are obtained from the image, we need to compare these to database or enrolled features. Normally the feature information exists as an n-dimensional vector, and we are determining the distance between 2 vectors. That is, we are calculating a distance metric or norm.

There are a variety of standard mathematical metrics for doing this-
In general,

$$L^p \text{ Norm: for } p \geq 1 \mathrm{K} \; \|x\|_p = \left(|x_1|^p + |x_2|^p + \mathrm{K} + |x_n|^p\right)^{\frac{1}{p}} \qquad (2.12)$$

Specifically,

$$\text{L1 or Manhattan metric: } \|x\|_1 = \left(|x_1| + |x_2| + \mathrm{K} + |x_n|\right) \qquad (2.13)$$

$$\text{L2 or Euclidean metric: } \|x\|_2 = \left(|x_1|^2 + |x_2|^2 + \mathrm{K} + |x_n|^2\right)^{\frac{1}{2}} \qquad (2.14)$$

Sometimes variations of these are used [49], [113]. Other metrics, such as the Mahalanobis, Hausdorrf [48], or Chamfer distances [102] are used in computer vision, but not used in this thesis. In Part II we will develop further metrics which are particularly suitable for interest point descriptor vectors.

## 2.8 Machine Learning Techniques

Ideas from statistics or machine learning are now used throughout computer vision.

*Linear Discrimination:*
In this work we make use of linear discrimination algorithms. The object of linear discrimination is to separate 2 classes of data points using linear functions (see Figure 2-16).

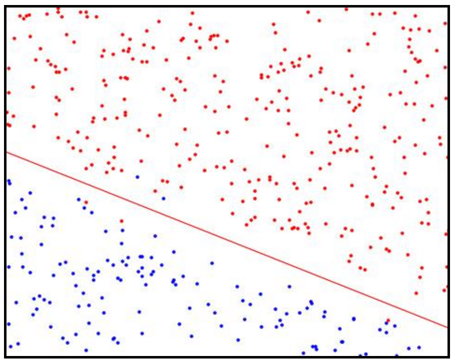


Figure 2-16: Example of 2D linear discrimination.
The 2 datasets in this image are separated by a linear function.
We use the same idea with higher dimensional data.

There are a variety of algorithms for this such as conjugate gradient descent, least squares, etc. Here we have used linear Support Vector Machines (SVM's) [22, 50, 111, 113] for this purpose, using a Matlab statistical library [5]. Linear SVM's will maximize the distance between nearest points of different classes on convex, separable data sets.

*Clustering:*
Another function widely used in visual recognition is clustering. The purpose of this is to separate data points into groups that are close to one another by some metric (see Figure 2-17). In class recognition tasks, it is presumed that such points will be of the same class. There are a variety of popular algorithms for this such as k-means and agglomerative methods.

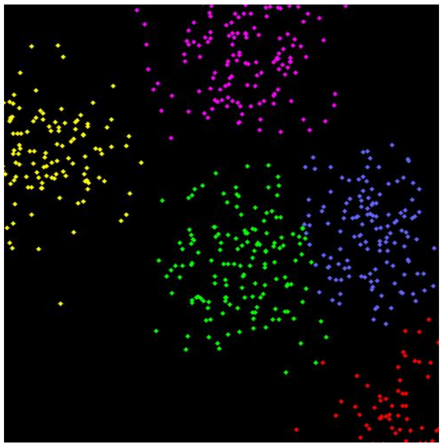


Figure 2-17: Example of clustering.
The 2D data in this image begins unlabelled, but has been clustered into 5 groups that have been colored differently.

## 2.9 Problems with Existing Approaches

Although the methods described in this chapter have constituted a significant step forward for computer vision, they still retain clear limits.

*Lack of a general theory:*
In so far as SIFT has achieved good recognition performance, this has been established empirically. No general theory exists to predict the performance of a detector/descriptor combination. Such a theory could illustrate better descriptors or show that better descriptors do not exist. This thesis does not present such a theory, but its absence motivates this work.

*Matching performance:*
The detector/descriptor framework has clear limits to its matching performance. For example, for the modest matching task involving scale and rotation change in images of Figure 2.12, SIFT achieves matching of < 40%. It may be that methods exist with substantially higher matching ability. Also, descriptors generally increase their performance as their dimensionality increases (since this provides a more detailed descriptor). However, a limit is reached beyond which performance will degrade. For

example, for the GLOH descriptor (which is very similar to SIFT) performance degrades after ~ 200 dimensions [70]. Thus, an ultimate limit to descriptor performance has been reached.

*Detector Limitations:*

The most established detectors such as DoG (Difference of Gaussian) or LoG (Laplace of Gaussian) select points without regard to their descriptor's discriminative properties. For example, the LoG selects points that are at a repeatable position within scale-space based on gradient. However, the descriptor at that point may a very weak response, or one that is very unstable. Also, for class recognition, DoG or LoG points may have little or no discriminative power.

For example, Figure 2.18 shows DoG points overlaid for these images. These points do not correspond to what other methods would identify as the key points of the face for recognition. For example, in Figure 2.16(b), the eyes are almost completely ignored by interest points (other methods have identified the eyes as among the most discriminative).

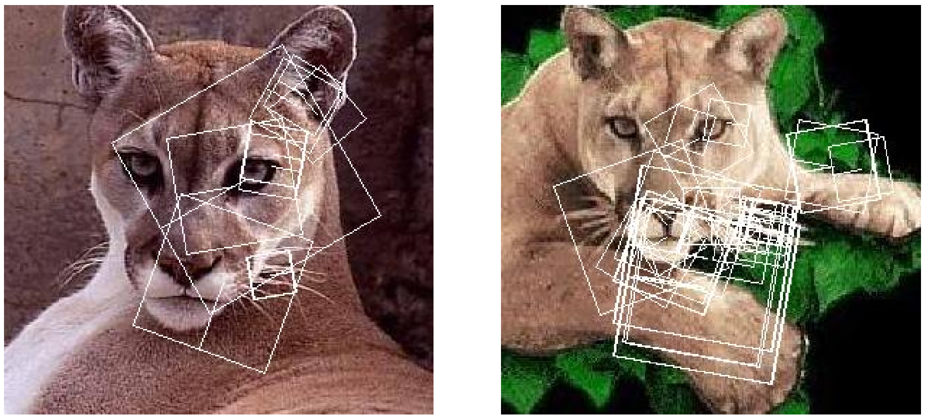


Figure 2-18: Example of DoG interest point extraction.
DoG points are selected for scale space repeatability – they do not necessarily correspond to the most class-discriminative regions of the image.

Points selected by the ‘biological’ model are an improvement since they are based on descriptor properties. However, such points are still selected by criteria only loosely related to class discriminative power (they are selected primarily by length, see Sec. 2.14). Also selecting 1 point from a fixed window size may be too inflexible for some classes. The Viola-Jones method is better in this regard, since it allows any point to be potential match, and selection of points for further processing is done by descriptor properties not by extraneous quantities such as DoG or LoG.

*Viola-Jones Limitations:*

The Viola-Jones method is substantially better for class recognition since the descriptor is the detector. However it has severe limitations in that a specifically selected sequence of descriptors must be used for each object class. Humans are thought to be able to distinguish between 1000’s of object classes. Although the Viola-Jones method is real time for a single class, it is not fast enough to simultaneously detect a large number of classes. In this sense a DoG detector is superior, since it can be applied unchanged to all points or classes. Also, often it requires a very large number of training examples, which may not always be available, and training is often very slow.

*Inflexible Descriptors:*

Although the Viola-Jones method has demonstrated good performance for face recognition, this is related to the fact that faces have a more consistent shape than many other classes. For example, to recognize a dog in images would probably require a method that could deal with greater variation. A more capable method would allow this greater flexibility, and would probably incorporate situation or scene specific information. Issues of scene understanding are however not the subject of this thesis.

*Limits to Point-based Matching*

Point based matching is the subject of this thesis and has been shown to be effective for many tasks. However, simple point regions may not be ideal for all tasks and may be unsuitable for some. Although we do not explore alternative 2D geometries hereafter, in most cases it is straightforward to do so. The basic descriptor operation remains the same with some shape change. For example, line descriptors could be formed along the line by elongating the descriptor to fit the line region-

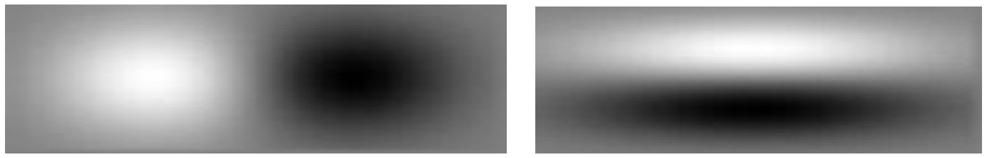


Figure 2-19: Example of descriptors stretched (compare Fig 2-12) to fit a line along the longer axis.

For some objects, recognition by 2D descriptors may be inherently unsuited. For example, recognizing a chair by 2D descriptors is very difficult due to the wide variety of possible appearances. However, 3D descriptors may be more suited. In many cases the descriptors detailed in this thesis could be extended to 3D, although to do so would be much more computationally demanding. For example, 3D Gabor wavelets are easy to conceive but are likely to be very slow to apply. 3D descriptors will likely be a feature of computer vision in the future.

# Part II The Geometric Match

# Chapter 3 Geometric Matching for Correlation

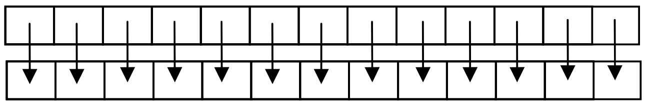
## 3.1 Introduction

In this chapter we introduce the concept of geometric matching, and use the correlation descriptor to illustrate it, since that is the simplest example.

Most popular point matching techniques rely on a 'descriptor' formed in relation to a specific point (see Sec. 2.3). This is generally represented as a vector data structure; the number of vector dimensions depends on the specific type of descriptor. Matching is performed by obtaining the distance between 2 vectors, and for this we need to choose a distance measure (see Figure 3-1). In this chapter we introduce a new distance measure as an alternative to current popular measures. *The primary motivation for this is to increase the flexibility of interest point matching.*

Various distance measures are commonly used in computer vision (as described in Sec. 2.7). Of these, the simplest and most commonly used one is the Euclidean distance, or $L_2$.

The Euclidean distance finds the distance between 2 vectors in a 'flat' space (e.g. unlike a Riemannian metric, or a metric on a set). It is a generic distance measure in that it does not take account of the specific properties of the data vector. It can be used to find the distance between any 2 arbitrary vectors. It treats all dimensions of the vector as being equal and independent dimensions. Since it does not take account of the particular characteristics of the data on hand, it may not be the optimal performing distance measure for a particular application.



Descriptor1 = $(u_0, u_2, u_3, \ldots, u_n)$

Descriptor2 = $(v_0, v_2, v_3, \ldots, v_n)$

Figure 3-1: Descriptors relate to an image patch and are represented by vectors.
Patches are compared by differencing the vectors.

A region descriptor is a data vector that is designed for a particular purpose. Ideally, it very efficiently characterizes data that has a specific geometric meaning. It typically encodes information including spatial, orientation and frequency information. Since it has a specific and consistent geometric meaning it is reasonable that we could find a better distance metric for it than the Euclidean metric.

In matching descriptors obtained from interest points, often the underlying image has been distorted by viewpoint change, occlusion, etc. Also, the point the descriptor is centered on may not be perfectly located due to detector limitations. For these reasons the Euclidean distance may not be the best distance measure available.

## 3.2 Correlation Descriptor

The correlation descriptor is perhaps the simplest descriptor. It is simply the image region around a point smoothed with a Gaussian kernel and sub-sampled. If the Gaussian smoothing and sub-sampling are in proportion, it is simply the image patch at a higher scale (ignoring edge effects). This can be an efficient descriptor to use because higher scale representation of an image must normally be obtained anyway for scale invariance. The correlation descriptor only encodes spatial information, not frequency or orientation information. For this reason it is particularly suitable for geometric matching, and we begin by treating it.

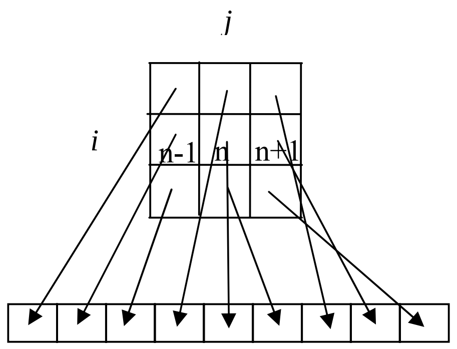
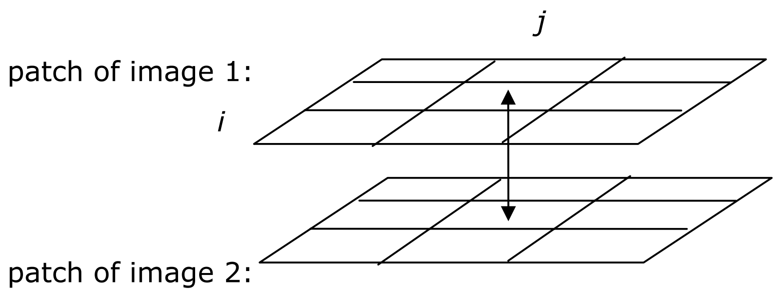


Figure 3-2: Geometric correlation descriptors.
For the correlation descriptor, each element of the descriptor vector has a simple spatial geometric meaning

It can be seen in Figure 3-2 that the individual dimensions of the vector are not completely independent – a geometric relationship or metric exists relating them together. For example, pixel *n* in the diagram is a neighbour of pixel *n*+1 and therefore the value if pixel *n* is likely to be similar to that of pixel *n*+1. More generally, the positions of the pixels in the descriptor are related by a Euclidean metric $\sqrt{i^2 + j^2}$ . The values of the pixels are related in some way by this metric. Treating such data as a vector of independent dimensions ignores this metric and is inappropriate.

## 3.3 Geometric Distance Metrics

The Euclidean distance, as applied to 2 such a correlation descriptors, gives a distance between



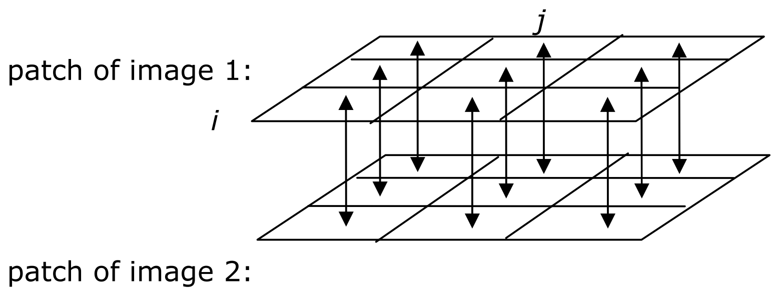


Figure 3-3: Euclidean correlation descriptors.
In Euclidean differencing for the correlation descriptor, each spatial region is only compared to its corresponding region in the other patch

In Figure 3.3 it can be seen that by the Euclidean vector distance the *match* is between exactly corresponding pixels in the descriptor patch. This distance is an approximation to the difference between the 2 descriptors. However, it can be seen that it does not take full account of the geometric setting of the data. What we seek is a distance measure that takes account of this fact - that nearby descriptor pixels are closely related, and that distant descriptor pixels are less related.

One way to do this is to allow the match to vary over an area of the descriptor patch. For example, the match of Figure 3.3 can be generalized somewhat to-

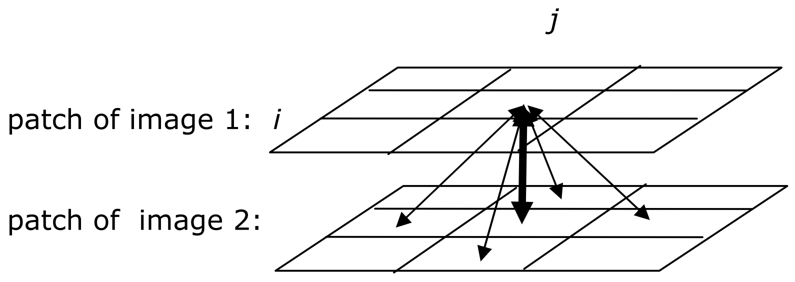


Figure 3-4: Simple geometric matching.
One form of simple geometric matching is to compare each pixel to the 4-connected region around it

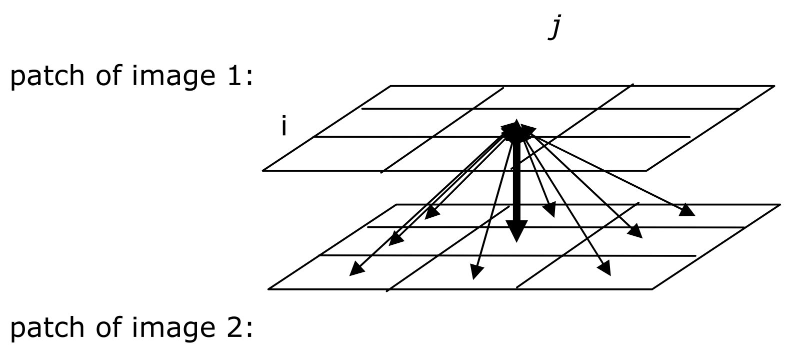


Figure 3-5: Localized geometric matching.
A slightly more complex form of geometric matching is to
compare each pixel to the 8-connected region around it

This match distance allows matching to the immediate neighbors of a given pixel. This can be done in a variety of different ways such as 4-connected or 8-connected regions (see Figures 3.4 & 3.5).

This is an improvement over the Euclidean match but it opens up new questions. Should the match to neighbor points in Figure 3.5 be treated equally to the match to the corresponding centre point? If not, how should we make the distinction between the centre point and the neighbors? Effectively we want to *encourage* the match to the centre point, but still *allow* the match to neighbors. One simple way to do this is with the matching function

$$\begin{aligned} d_{corres} &= x_{i,j} - y_{i,j} \\ d_{neigh} &= \min_{m,n=-1,0,1 m,n\neq 0,0}(x_{i,j} - y_{i+m,j+n}) \\ d &= \sum_{i,j} \min(d_{corres}, k * d_{neigh}) \end{aligned} \tag{3.1}$$

*i*, *j* represent the spatial dimensions of the descriptor, as per Figure 3.3 above.

The parameter *k* is the penalty term for neighbour matching. It expresses our prior belief that matching is most likely to occur to the centre point. We therefore only accept the match to the neighbour if it is markedly closer. Using this penalty is a Bayesian concept, since it is weighting the choice in favor of the centre point (see Figure 3-11 for the smoothed case). The match selects values for *m, n* which are the matching *disparity*.

In the formulas (3.1) we have used the indices *i,j* to denote positions over the descriptor area. This is not appropriate for a practical formula since the data structure is a vector i.e. we need to map *i, j* positions to positions in a vector. So we end up with a match of the form of Figure 3-6.

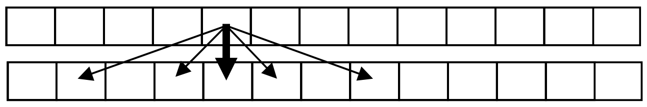


Figure 3-6: The geometric match results in matching to certain specific components in the descriptor vector, depending on the type of matching and the descriptor.

This is slightly more complicated to implement numerically. The optimal value of the penalty parameter *k* must be determined empirically.

*Definition 3.1*: A *geometric match* is a matching between descriptor vectors which allows matching not only between corresponding components of the vector, but also to neighbors in space, orientation, frequency, etc.

## 3.3.1 Geometric Arrangements

If we allow a more general geometric match, rather than a straight Euclidean match, there are a variety of spatial arrangements we could use.

*(a) Simple neighbour points*: as described above in Figs. 3.4 and 3.5, with a penalty factor for immediate neighbors compared to the corresponding central point.

*(b) Higher scale points*: This idea is where, for every pixel, we also obtain a higher scale value. This is slightly different from a standard higher scale image, since we have a value at every pixel of the lower scale (rather than at sub-sampled points). We allow a match to this higher scale pixel, and also a normal match to the corresponding same scale pixel (see Figure 3-7). Again we have a penalty factor *k* but now it weighs towards the same scale pixel.

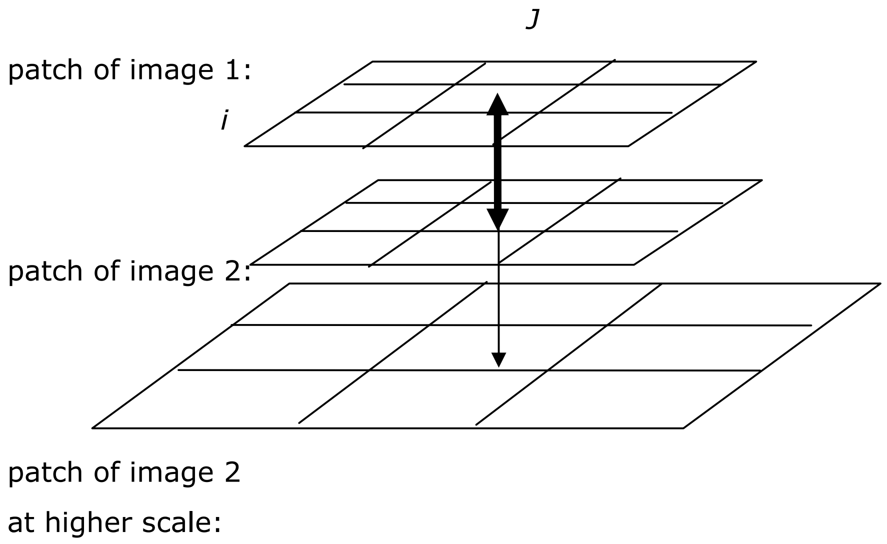


Figure 3-7: Geometric matching can be used to match to other scale levels.

*(c) Overlapping higher scale points*: this is similar to the simple neighboring pixel case but it is done at a higher scale, and we allow some pixel overlapping

Again we have a penalty factor *k* to weight towards the central pixel.

*(d) Multiple radii neighbour points*: in this formulation we have 2 penalty parameters *k*1 and *k*2, and 2 levels of neighbors. These levels correspond to

different distances from the centre point as shown in Figure 3-8. Effectively, we are allowing the point to vary even more widely and therefore can offer improved performance. The potential problem with this approach is that it can be very computationally demanding – each point must be checked against a large number of neighbors.

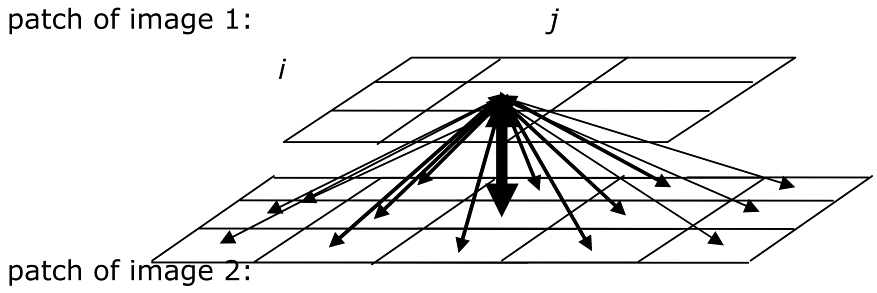


Figure 3-8: Geometric matching can be generalized in many ways. Here we are matching to an extended region around each pixel.

This concept can be generalized to matching to all neighbors within a radius *r*, with a penalty function of *r*, e.g. a quadratic penalty function.

## 3.3.2 Patch or Descriptor Size

A correlation descriptor performs a blurring and a sub-sampling. Thus an image patch is extracted, reduced to a smaller size, and then vectorised. In our testing we have used image patch sizes of 41×41 since this gives good results and is consistent for comparison with [54].

The size of the sub-sampled or decimated patch is a tradeoff. If it is too small (too few dimensions) the original patch has flexibility to be distorted, but the match loses specificity and discrimination power. If the patch is too big, it is very specific which improves specificity and discrimination, but the patch is not flexible to geometric transforms or distortion.

With a simple Euclidean match we have found that maximum performance occurs for a descriptor size of 12×12, decimated from a 41×41 patch. When we use a geometric match with simple neighbour points, we find that the optimum occurs at 18×18 (see Figure 3-9). Thus usage of geometric matching allows us to be more specific with our descriptor, and this improves matching performance.

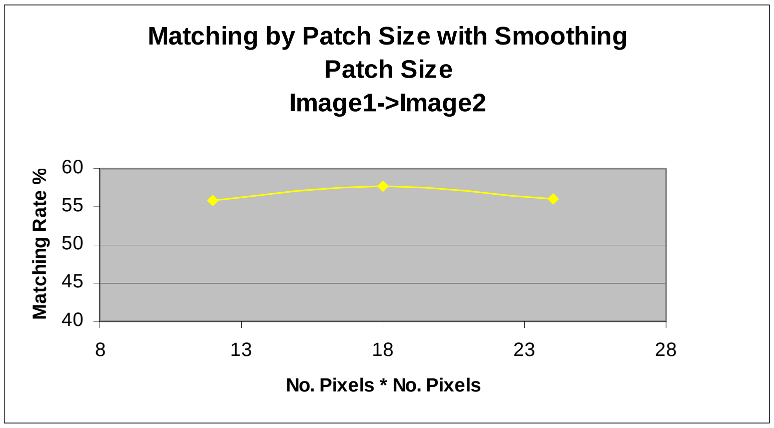


Figure 3-9: The variation of matching rate (with smoothing) with the patch size in pixels

## 3.3.3 Edge Weightings

When we are decimating an image patch we must blur it or apply a filter first. This is to prevent anti-aliasing (by Nyquist's Theorem). For this we apply a discretized, finite approximation to a Gaussian filter. For each point a convolution must be performed centered on that point; however, for edge points a full convolution is impossible, since many of these points do not exist in the patch.

Thus edge points in the decimated image are not determined as reliably as more central points. For this reason we can weight points according to how much of their convolution region exists. In general image decimation this is not such a problem since most points are not near edges. In our application however, the patches are 41×41 so many points in the convolution are near edges. A weighting factor can therefore be applied in the correlation match.

$$
\begin{aligned}
d_{corres} &= w_{i,j}(x_{i,j} - y_{i,j}) \\
d_{neigh} &= \min_{m,n=-1,0,1 m,n\neq 0,0} w_{i,j}(x_{i,j} - y_{i+m,j+n}) \\
d &= \sum_{i,j} \min(d_{corres}, k * d_{neigh})
\end{aligned}
\tag{3.2}
$$

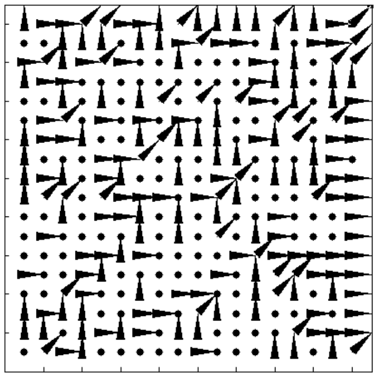
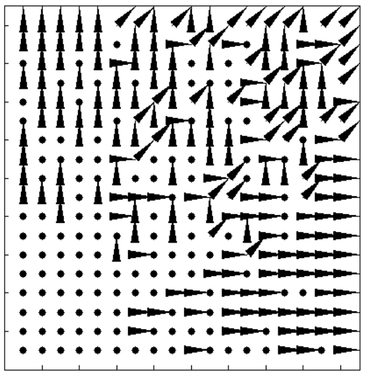
Where $w_{i,j}$ is a position based weighting factor.

## 3.4 Implementation Issues

### 3.4.1 Energy Minimization or 'Stereo' Smoothing

Using geometric matching to find the best match simply using Eq. (3.1) above does not give very good performance. This is because the matching simply selects the closest weighted matching pixel among the neighborhood region of the corresponding pixel in the other patch. This means that multiple pixels can match to a single pixel or some pixels may not match to any others at all. Effectively we have given too many degrees of freedom to the matching process, which results in a somewhat random match. This can be seen in Figure 3.10 (a). Traditional correlation matching only compares corresponding pixels in the patch and this imposes a certain discipline or constraint.

This problem exists because we have allowed each pixel to be matched independently of others. If we can synchronize or maintain some continuity between pixels we can address this problem.

(a) (b)

Figure 3-10: Geometric matching.
(a) Without smoothing geometric matching gives a somewhat random matching. (b) We can use some disparity smoothing methods similar to stereo methods to apply a gentle smooth and improve the match

This is very similar to the problem of stereo matching, where pixels are matched along epipolar lines and continuity should be respected. The difference in this case is that the matching can occur in 2 dimensions – points do not lie on a known line, they in an area around a known point. We can then adopt a stereo matching technique and adapt it to this situation. A common stereo method is to apply a penalty term to a pixel match, based on how different that match is from its neighbors.

Suppose the offsets, or 2-dimensional disparities are *m, n* (see Eq. 3.1) and *N* is the neighborhood region of the pixel (e.g. 4-connected or 8-connected). Let $N_m$ and $N_n$ be the disparities of the neighbors, then we have a penalty function

$$V(m,n) = \phi \bullet T(m, median(N_m)) + \phi \bullet T(n, median(N_n))$$
$$T(a,b) = 1 \qquad a \neq b$$
$$T(a,b) = 0 \qquad a = b$$

This is a variation of the Potts penalty model of stereo matching (see [16]). The parameter *Φ* determines how strongly we impose the disparity penalty. Figure 3.9 (b) shows a moderate application of this.

Then the final descriptor patch matching function becomes

$$
\begin{aligned}
d_{corres} &= x_{i,j} - y_{i,j} + V(0,0) \\
d_{neigh} &= \min\left(_{m,n=-1,0,1 m,n\neq 0,0}(x_{i,j} - y_{i+m,j+n}) + V(m,n)\right) \\
d &= \sum_{i,j} \min(d_{corres}, k * d_{neigh})
\end{aligned}
\qquad (3.3)
$$

The function *d* in Eq. (3.3) can be considered an energy function and matching finds the minimum energy. This approach can be considered as applying stereo matching ideas to the task of interest point matching.

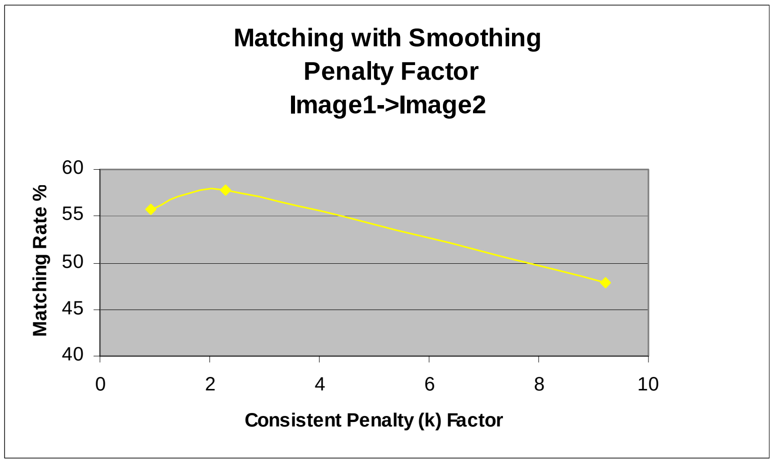


Figure 3-11: Matching with smoothing penalty.
The variation of matching rate (with smoothing) with the penalty factor of Eq. (3.3) for geometric matching

### 3.4.2 Cascaded Implementation

Most of the geometric match schemes in Sec. 3.2.1 above are substantially more computationally demanding than simple Euclidean matching. One way to overcome this is to implement them in a cascaded fashion.

Euclidean nearest neighbour correlation matching is first performed on all points. Then for the best *p* matches we perform a geometric match. Thus it is a 2 stage process. In practice p = 15 has given good performance and the computational demands are comparable to ordinary Euclidean matching.

### 3.4.3 Commutative Operation

The standard Euclidean match is a commutative operation between its 2 arguments. The introduction of geometric flexibility breaks this commutativity. This is not necessarily a problem (e.g. if it provides overall performance improvements) but the lack of symmetry may suggest the method is not perfect.

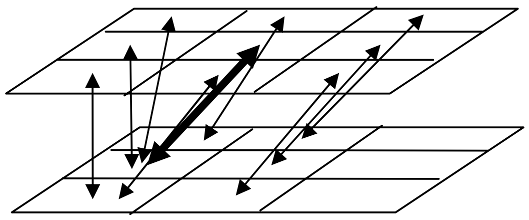


Figure 3-12: Lack of symmetry in matching

In Figure 3.12 we can see that for 1 of the arguments (the top image) every value (pixel) is matched, whereas in the 2nd argument (bottom image) some pixels are not used in the match and others are used several times.

There are several simple ways that symmetry between the 2 vectors in the match could be restored. One way would simply be to take the average of the 2 operations: ½(difference(vec1, vec2) + difference(vec2, vec1)). Another is to take the minimum: min(difference(vec1, vec2), difference(vec2, vec1)). We use this minimum as comparison in Table 3.1.

Table 3.1: Commutative vs Standard Matching Rate (structured image1->image4)

| | Viewpoint | Scale | Blur | Average |
|---|---|---|---|---|
| Standard Matching (%) | 10 | 39 | 47 | 32 |
| Commutative Matching (%) | 11 | 42 | 49 | 35 |

## 3.4.4 Range of Transforms

The geometric match allows us to better deal with geometric transforms that the images may have undergone. For example, it improves performance under affine transforms. However it does not provide invariance against all possible transforms. When used with a smoothing term (Sec. 3.4.1) geometric matching provides good performance against smooth or diffeomorphic geometric transforms. These include affine and projective transforms. Some examples of non-diffeomorphic image transforms are-

- Image transforms that include occlusion
- Photometric transforms (changes in brightness or color)

Geometric matching will be of reduced performance in such cases.

## 3.4.5 Comparison to Elastic Bunch Matching

Elastic Bunch Graph Matching (EBGM) is a recognition method that has found success in face recognition and Optical Character Recognition (OCR). It requires manual location of ‘fiducial’ points in the image and Gabor jets (a set of convolution filters) are applied at these points. The ‘fiducial’ points are allowed to move with some flexibility over an area. A matching metric is constructed based on the similarity of the position of the points and the values of the Gabor jet.

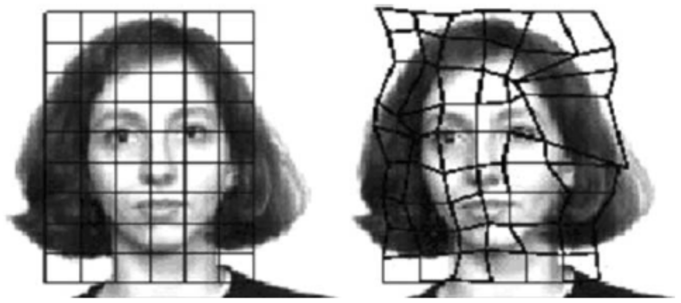


Figure 3-13: Elastic Bunch Graph Matching
[95]

EBGM has a point of similarity to geometric matching in that it allows some degrees of freedom in the matching process. It is different in that the flexibility for EBGM relates to point positions whereas in this scheme interest point position is not flexible; the flexibility pertains to the matching metric. In EBGM point position is determined for a class, whereas we are here using scale-space points (e.g. LoG). This relates to the fundamental difference between the methods – EBGM is used for class recognition whereas geometric matching is a point matching scheme. Thus they are used for different purposes or tasks.

### 3.4.6 Complexity and Timing

The complexity and timings are important for a practical descriptor method. Table 3.1 shows a summary of the computation required in forming and matching the descriptor. The first column relates to the interest point detector, the second shows the main operations on the image to detect these points. The third and fourth relate to the main operations on the descriptor patch and individual pixels respectively. We use 4 different scales to detect the points. It can be seen that the matching time is much greater than the point extraction time. Note that we do not use any acceleration such as approximate matching or GPU acceleration. In Table 3.2 below, G(I) denotes Gaussian smoothing over the whole image, G(P) denotes Gaussian smoothing of the descriptor patch, and Comp. denotes a comparison (subtraction and minimum operation).
As expected, there is a substantial increase in matching time as geometric matching is increased.

Table 3.2: Geometric Correlation Timings

| Descriptor Matching | Image Operations | Descriptor Operations | Pixel Operations | Point Extraction Time (s) | Matching Time (s) | Number of Points |
|---|---|---|---|---|---|---|
| Geometric Matching 4-Connected | #4G(I) | G(P) | 4 Comp. | 2.4 | 92 | 1000 |
| Geometric Matching 8-Connected | #4G(I) | G(P) | 8 Comp. | 2.4 | 173 | 1000 |

## 3.5 Experimental Results for Correlation Matching

In this section we present experimental results on the Oxford image sets. See appendix I for examples of the image sequences. Since are only interested in descriptor performance here, the points used for testing were DoG points in all cases.

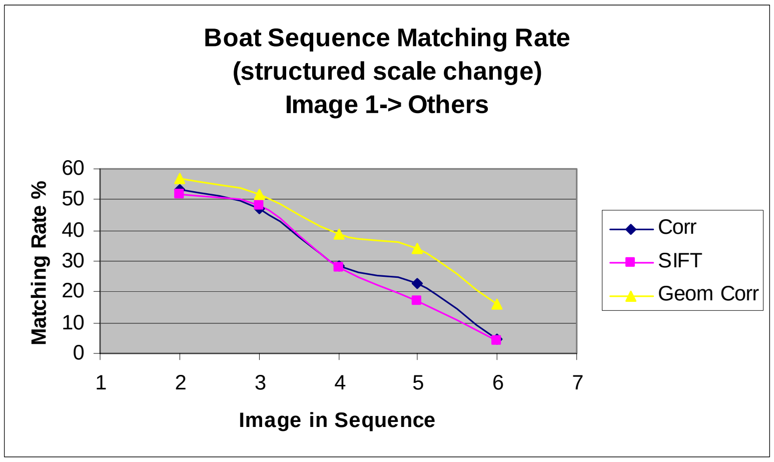
We present data sequences results for

- correlation descriptors with Euclidean matching – 'corr'
- correlation descriptors with geometric matching – 'geom corr'
- SIFT descriptors with Euclidean matching – 'SIFT'

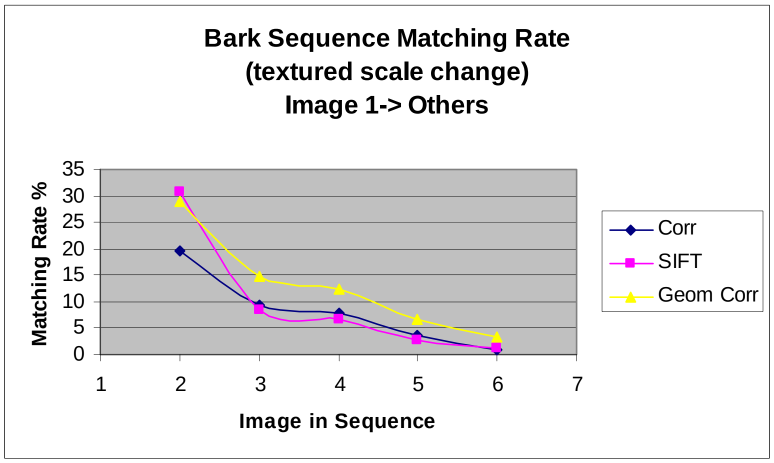
The 'correlation' and the 'correlation with geometric matching' illustrate the improvement that results from using geometric matching. The SIFT data is presented for comparison since SIFT has been the 'gold standard' of point matching for some years. These data sequences assess the performance of descriptors against change in scale, viewpoint, blur, illumination, and JPEG compression. For scale, viewpoint and blur, a distinction is drawn between 'structured' and 'textured' images. For example, the 'Boat' sequence is a series of images of a boat at increasing scale and rotation. It is considered a 'structured' sequence. The 'Bark' sequence also presents scale change, but it is considered a textured sequence. Data presented is the matching score for the sequence images, and also the precision-recall performance for matching image1->image 2 of the series. See Sec. 2.6 for discussion of what these testing parameters mean.

Table 3.3*:* Parameters of the experiments

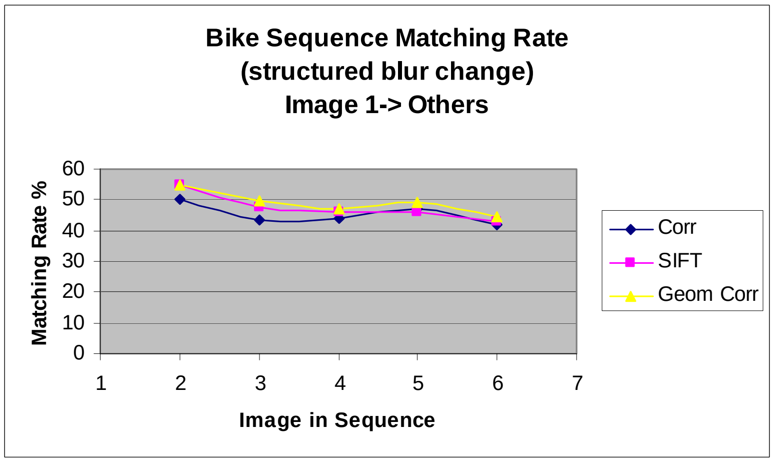
| **Experimental Parameters** | |
|---|---|
| Number of initial DoG points | 1000 |
| Dimensions (1st level) | 12×12 |
| Dimensions (2nd level) | 18×18 |
| Geometric match type | Simple neighbors (8-connected) |
| K Central weighting | 2.0 |
| Φ Energy smoothing factor | 0.3 |
| | |
| Timing for interest point and descriptor extraction | 2.4s |



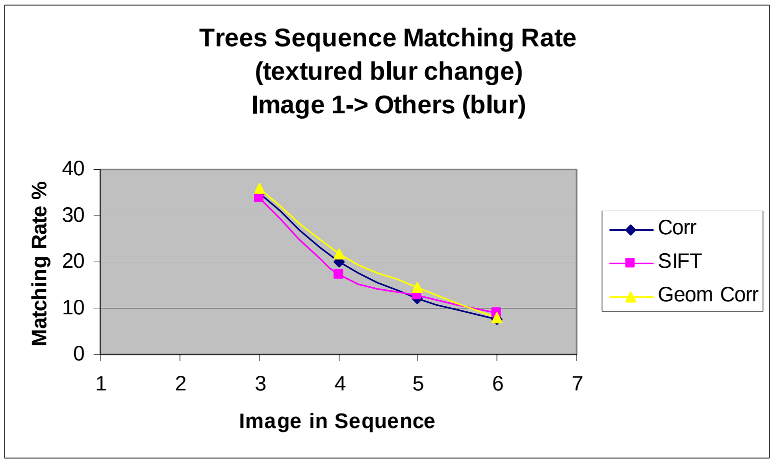
(a) Structured scale change



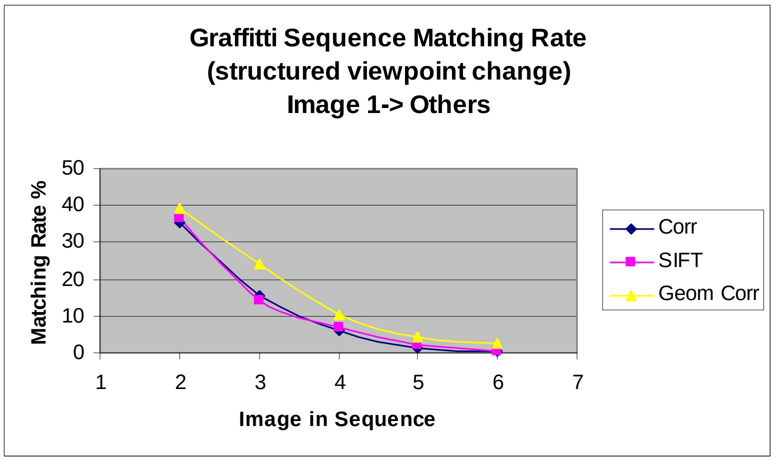
(b) Textured scale change



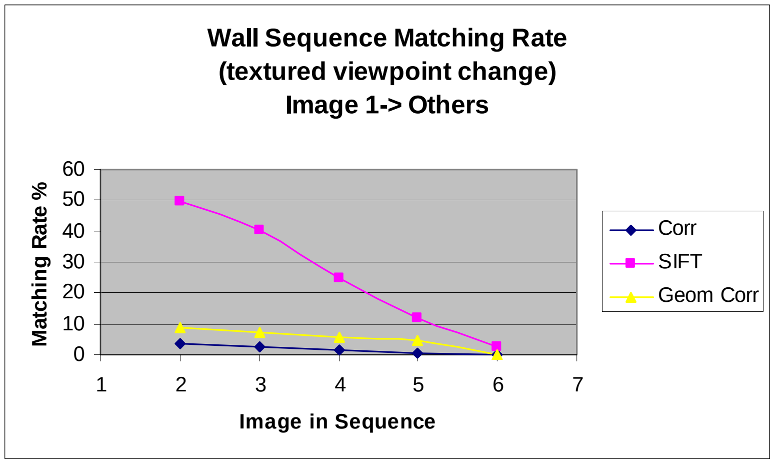
(c) Structured blur change



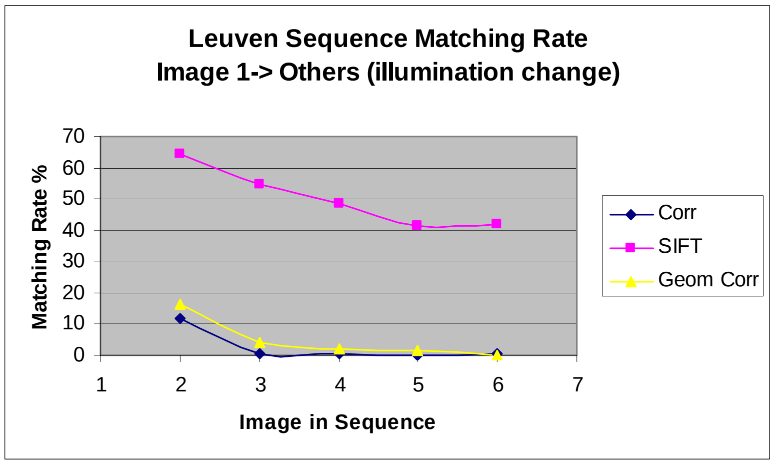
(d) Textured blur change



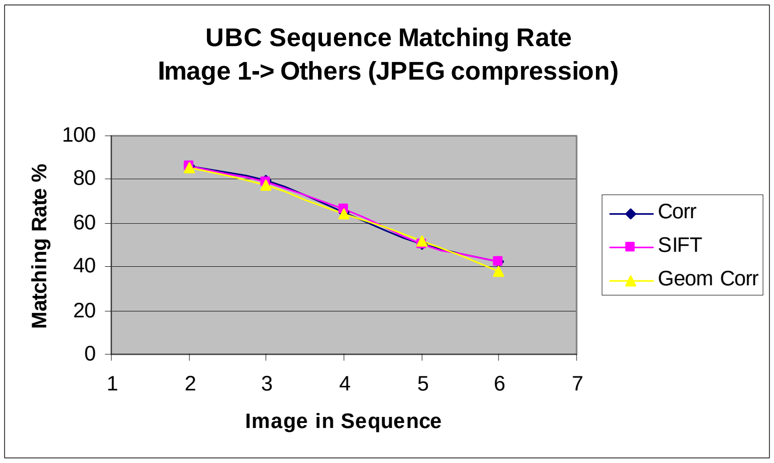
(e) Structured viewpoint change



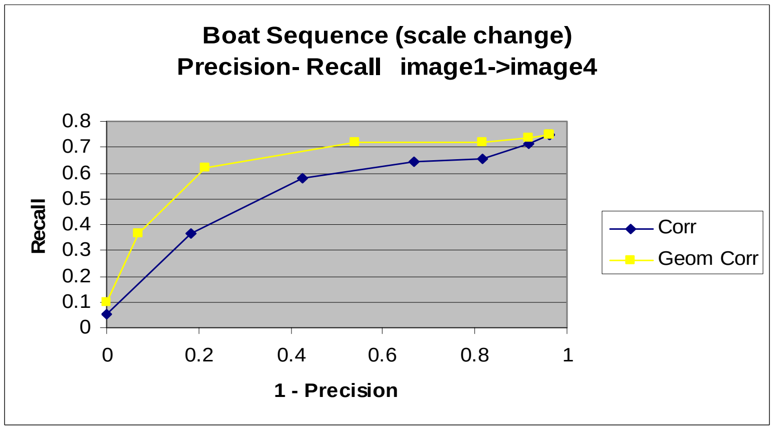
(f) Textured viewpoint change



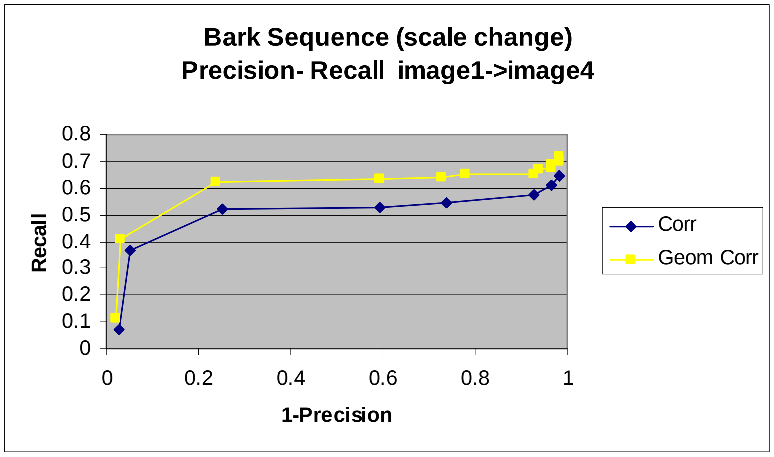
(g) Illumination change



(h) JPEG compression change



(i) Structured scale change precision-recall graph



(j) Textured scale change precision-recall graph

Figure 3-14: Various changes to the matching rate in correlation matching.

Table 3.4: Summary Statistics

| | Textured | Structured | Viewpoint | Scale | Blur | All |
|---|---|---|---|---|---|---|
| | | | | | | |
| Corr | 8.89 | 29.43 | 6.69 | 19.79 | 33.46 | 23.14 |
| SIFT | 17.96 | 29.81 | 18.93 | 19.90 | 34.48 | 32.67 |
| Geom Corr | 12.32 | 34.90 | 10.69 | 26.36 | 36.17 | 26.59 |

### 3.5.1 Discussion of Results

The experimental results above show that use of geometric matching improves performance in almost all cases. Figs 3-14 (a) and (b) show that under scale change geometric matching generally delivers a clear improvement in matching.

For blur change, Figures 3-14 (c) and (d) the geometric match makes less of a difference, although there is a small improvement. This is reasonable since blur is less of a geometric distortion that the geometric match would respond to. Also, the blur change matching rates are already high.

For viewpoint change Figures 3-14 (e) and (f) we again see that the geometric match gives a significant improvement. This is as expected since a viewpoint change creates a geometric deformation of the local patch or area. The geometric match is similar to stereo matching which is usually applied to a viewpoint change; therefore we expect it to be effective in this case.

In these results we do not see a significant difference between structured and textured image performance.

For illumination change the correlation descriptor has much lower matching performance than the SIFT descriptor and the geometric match delivers a small improvement. The low performance here is related to the fact that we have used unnormalized correlation descriptors. For JPEG compression, Figure 3-14 (h), we see no effect from geometric matching. This is again somewhat expected since JPEG distortion is not a smooth geometric distortion.

Figures 3-14 (i) and (j) show that the improvement in performance is not just for nearest neighbour matching, but also for precision-recall profiles.

In some cases the improvement for geometric matching gets larger for successive images in a sequence. This is typical of affine invariant methods, and suggests that geometric matching can be regarded as a form of affine invariance.

Overall the correlation results have recorded a clear improvement and are competitive with SIFT. This has not previously been demonstrated for correlation descriptors.

# Chapter 4 Geometric Match for SIFT

## 4.1 Introduction

In this chapter we extend the concept of geometric matching to SIFT descriptors. Manly of the concepts from Chapter 3 remain valid and we only emphasize those that are different.

The SIFT descriptor is described in Sec. 2.3.1 It is most commonly a descriptor of 4×4×8 dimensions which is composed from 4×4 spatial regions, and 8 orientations. It has become the most widely used of all point descriptors for point recognition.

The main difference between the SIFT descriptor and the convolution descriptor is the inclusion of orientation information. This motivates some changes to our approach but the fundamental concept is unchanged.

In Figure 2.7 it can be seen that the SIFT descriptor has a precise geometric meaning and that therefore its dimensions are not independent. A 2-d Euclidean metric $\sqrt{i^2 + j^2}$ exists between its spatial components, and a similar 1-d metric *u* exists between its orientation components (although globally the topology of orientation is different, this need not concern us).

## 4.2 Geometric Distance Metrics

The Euclidean distance metric on the SIFT descriptor differences as in Figure 4-1.

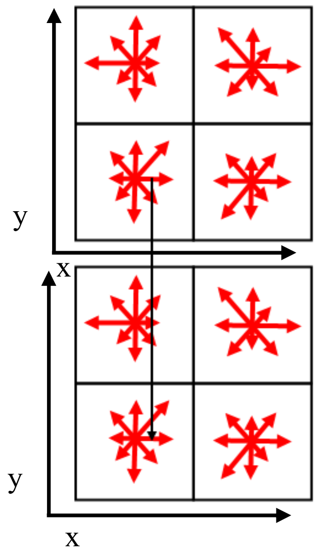


Figure 4-1: Standard Euclidean matching on the SIFT descriptor matches corresponding components of space and orientation

Applying the concept of the geometric match, where we allow matching to neighbors dimensions in space and orientation, gives a distance between descriptors as in Figure 4-2.

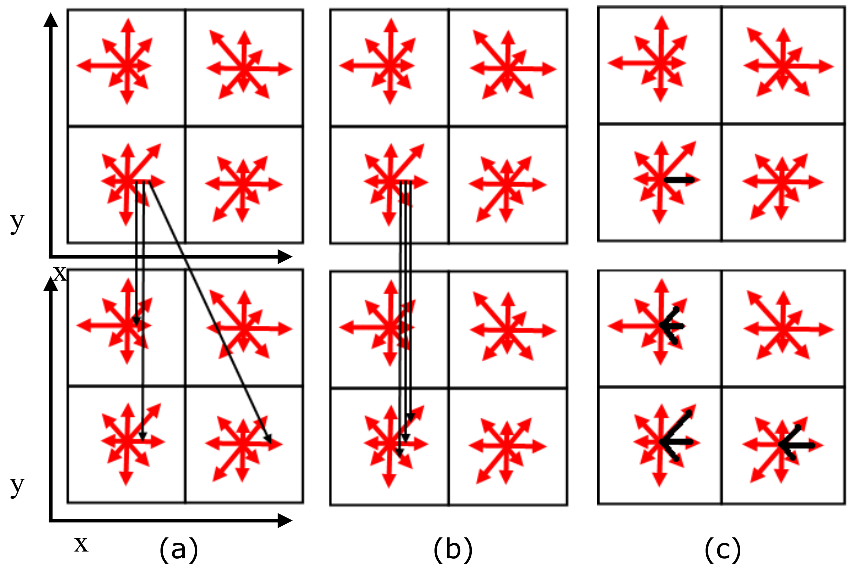


Figure 4-2: Geometric matching can be used to match to the neighboring descriptor components in both space and orientation

This corresponds to a distance measure of

$$
\begin{aligned}
d_{corres} &= x_{i,j,o} - y_{i,j,o} \\
d_{space\_neigh} &= \min{}_{m,n=-1,0,1 m,n\neq 0,0}(x_{i,j,o} - y_{i+m,j+n,o}) \\
d_{orien\_neigh} &= \min{}_{r=-1,1}(x_{i,j,o} - y_{i,j,o+r}) \\
d &= \sum_{i,j} \min(d_{corres}, k_1 * d_{space\_neigh}, k_2 * d_{orien\_neigh})
\end{aligned}
\qquad (4.1)
$$

Here *i, j* correspond to spatial position and *o* is the position in orientation. $k_1$ and $k_2$ are the penalty terms for spatial and orientation movement respectively. This is the simplest form of geometric matching and corresponds to (a) of Sec. 3.2.1. Other geometric match distances similar to (b) – (d) of Sec. 3.2.1 are possible, however we will restrict ourselves to the simplest formulation in this case.

## 4.3 Size, Weightings, Cascading, Smoothing

As in the case of a convolution descriptor, applying a geometric match allows a size increase for the descriptor. For Euclidean matching, the optimum SIFT descriptor size is 4×4×8. Use of geometric matching allows this to increase to 6×6×10 and provides a commensurate increase in performance.

We have not applied edge based weightings (as per Sec. 3.2.3) for the SIFT in this implementation. This is because SIFT has a standard definition and we want to facilitate comparison with this, and we have found the weighting factor does not have a great effect.

Since the geometric match descriptor has more dimensions, it is computationally more demanding. Therefore we implement it in a cascading approach, as per Sec. 3.4. Thus, we use the standard 128-dimensional descriptor matching to identify the best 15 matches, then use 360-dimensional SIFT geometric matching for the final match.

As per Sec. 3.3 we employ energy smoothing to make the matching more consistent. Again this is similar to stereo matching, and we do it for both the spatial and orientation dimensions.

### 4.3.1 Complexity and Timing

Table 4.1 shows a summary of the computation required in forming and matching the descriptor (see Sec 3.46). In Table 4.1 below, G(I) denotes Gaussian smoothing over the whole image, G(subP) denotes Gaussian smoothing on a subset of the descriptor, Grad. denotes taking of gradients, and Comp. denotes a comparison (subtraction and minimum operation). As expected, there is a substantial increase in matching time as geometric matching is increased.

Table 4.1: Geometric SIFT Timings

| **Descriptor Matching** | **Image Operations** | **Descriptor Operations** | **Pixel Operations** | **Point Extraction Time (s)** | **Matching Time (s)** | **Number of Points** |
|---|---|---|---|---|---|---|
| Geometric Matching 4-Spacial | 4G(I) | 8 Grad, 8*16G(SubP) | 4 Comp. | 1.9 | 83 | 1000 |
| Geometric Matching 4-Spatial, 2 - Frequency | 4G(I) | 8 Grad, 8*16G(SubP) | 8 Comp. | 1.9 | 161 | 1000 |

## 4.4 Experimental Results for SIFT Matching

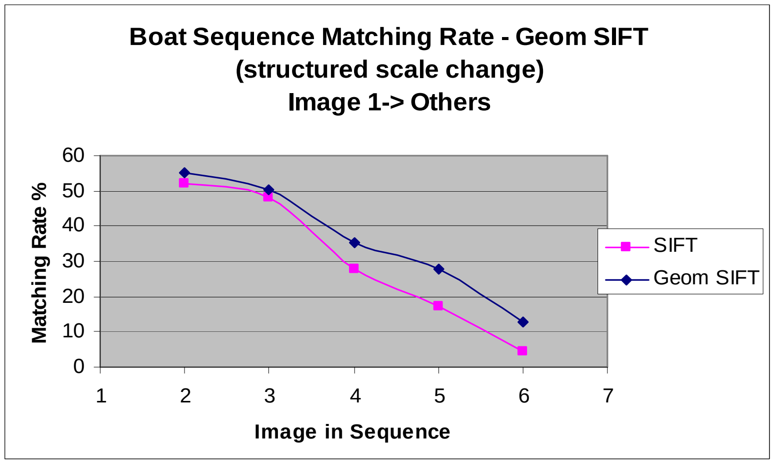
In this section we present experimental results on the same Oxford image sets and using the same DoG points as previously.

We present data sequences results for

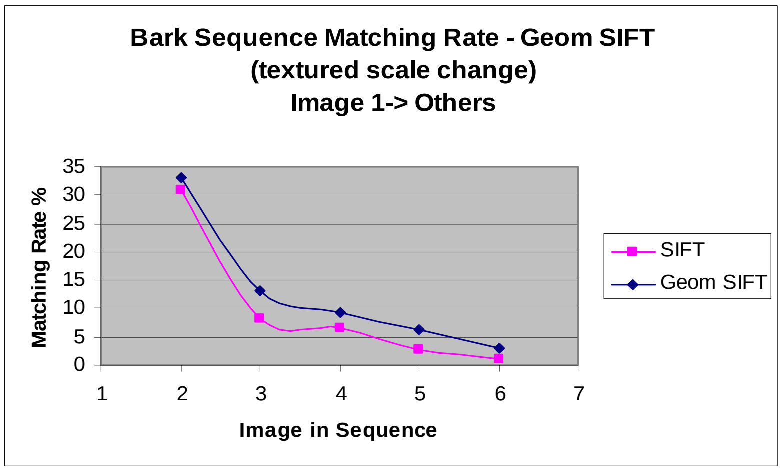
- SIFT descriptors with Euclidean matching – 'SIFT'
- SIFT descriptors with geometric matching – 'Geom SIFT'

Table 4.2: Parameters of the experiments

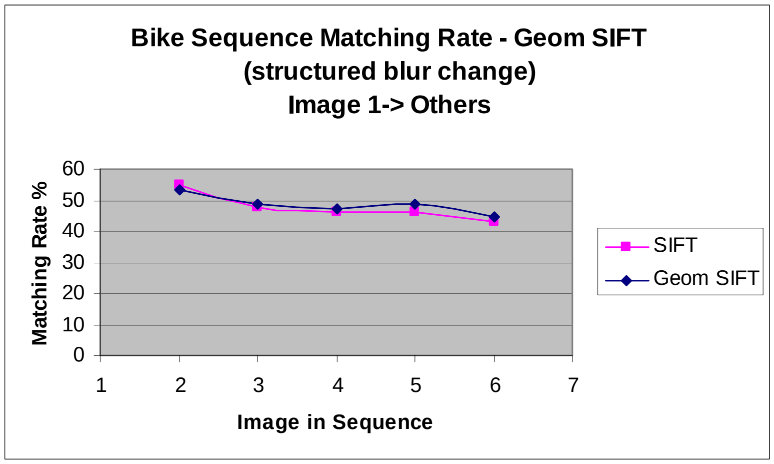
| **Experimental Parameters** | |
|---|---|
| Number of initial DoG points | 1000 |
| Dimensions (1st level) | 4×4×8 |
| Dimensions (2nd level) | 6×6×10 |
| Geometric match type | Simple neighbors (space and orientation) |
| $K_1$ Central weighting (spatial) | 2.5 |
| $K_2$ Central weighting (orientation) | 1.5 |
| Φ Energy smoothing factor | 0.3 |
| | |
| Timing for interest point and descriptor extraction | 1.9s |



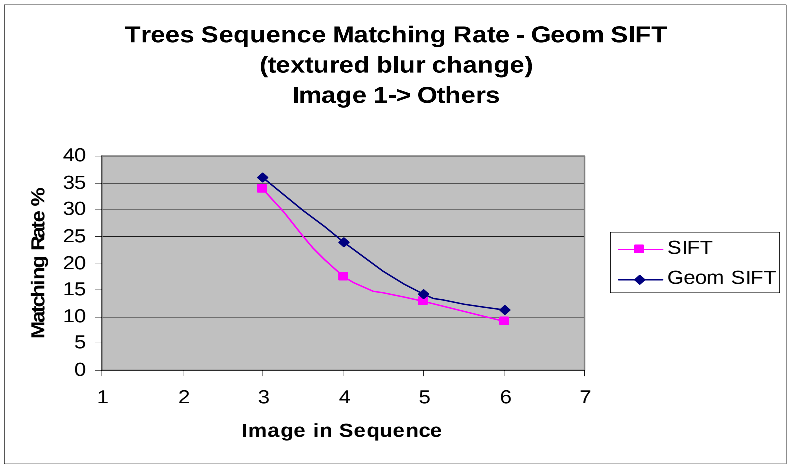
(a) Structured scale change



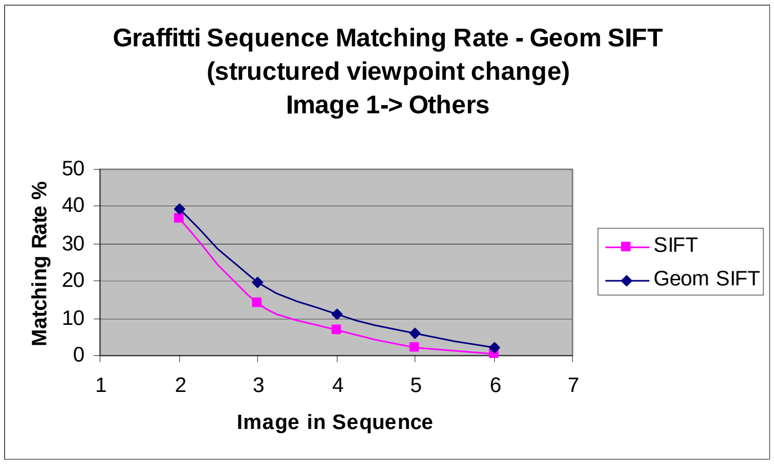
(b) Textured scale change



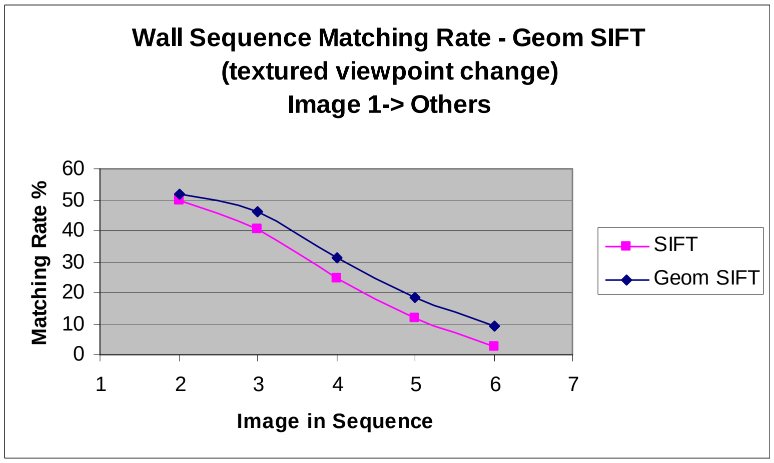
(c) Structured blur change



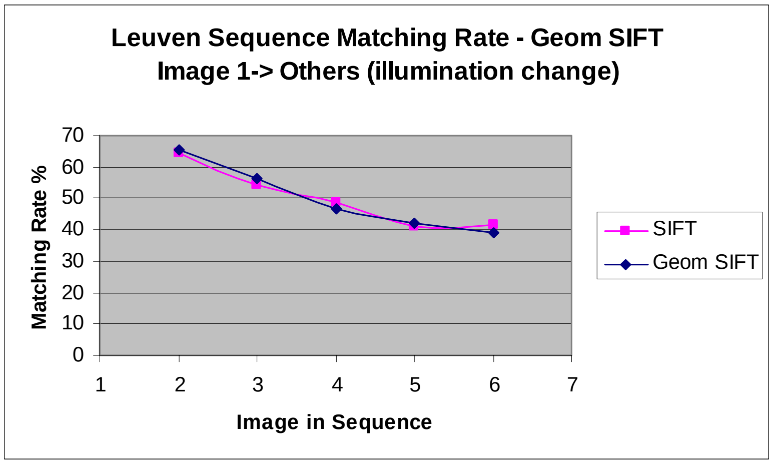
(d) Textured blur change



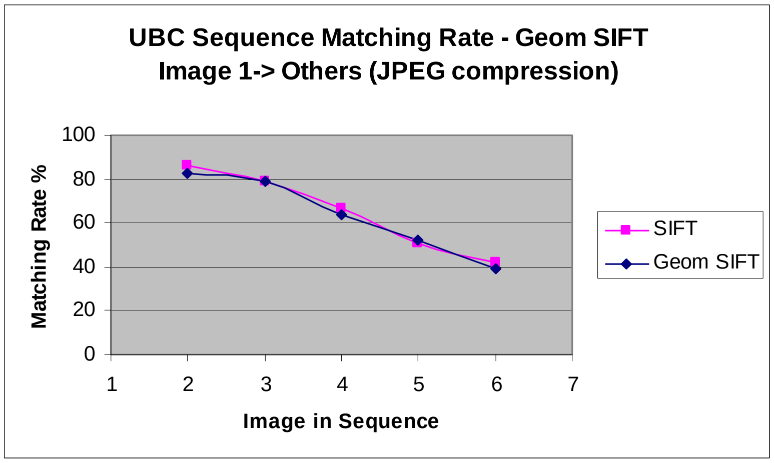
(e) Structured viewpoint change



(f) Textured viewpoint change



(g) Illumination change



(h) JPEG compression change

Figure 4-3: Various changes to matching rate in SIFT matching.

Table 4.3: Summary Statistics

| | Textured | Structured | Viewpoint | Scale | Blur | All |
|---|---|---|---|---|---|---|
| | | | | | | |
| SIFT | 17.96 | 29.81 | 18.93 | 19.90 | 34.48 | 32.67 |
| Geom SIFT | 21.94 | 33.51 | 23.48 | 246.30 | 36.51 | 35.28 |

## 4.4.1 Discussion of Results

The experimental results above show that use of geometric matching for SIFT also improves performance in most cases. Under scale change Figures 4-3 (a) and (b) show that geometric matching generally delivers some improvement in matching. For blur change, Figures 4-3 (c) and (d) the geometric match again makes less of a difference (this may be because blur already has a high matching performance). For viewpoint change Figures 4-3 (e) and (f) we again see that the geometric match gives a significant improvement.

Again, no significant difference between structured and textured images is observed. The improvement in performance occurs for both textured and structured images.

For Figures 4-3 (e) and (f), illumination and JPEG, we see no improvement from the geometric match. This is reasonable since they do not distort the image in spatial manner that the geometric match can respond to.

As previously, the geometric improvement seems to generally be smaller for the first image in a sequence and grows larger for successive ones. This again suggests that geometric matching has similarity to affine invariant methods.

These results demonstrate that the geometric matching concept is not restricted to spatial dimensions, and can also be applied to orientation. However, the performance improvement here depends on increasing the dimensionality of the descriptor to $6\times6\times10$ for the second level matching. Overall, the improvement in performance seems to be similar or slightly lower than for the correlation descriptor.

The results here recorded relate to fairly gentle ‘stereo’ smoothing. Superior performance could be obtained by applying this more aggressively, at a cost of greater compute time.

It is noteworthy that it is able to improve SIFT performance since SIFT has been a standard for interest point matching for some years. SIFT performance has only previously been clearly bettered by methods that include a prior learning stage (e.g. [45]) or by some formal affine invariant methods (e.g. [73]).

# Chapter 5 Geometric Match for Gabor

## 5.1 Introduction

In this chapter we extend the concept of geometric matching to Gabor descriptors. Again, many of the concepts from previous chapters remain valid and we only discuss those that are different.

The Gabor descriptor is described in Sec. 2.3.4. We have used a basic descriptor of 4×2×2 dimensions which is composed from 4 orientations, 2 phases, and 2 frequencies. It is therefore a low dimensional descriptor of 16 dimensions.

Thus the Gabor descriptor includes frequency, phase and orientation parameters. The main difference between the Gabor descriptor and the previous descriptors is that it has a specific localization in frequency space. Also, it lacks the finer gradation of spatial regions that the others possess.

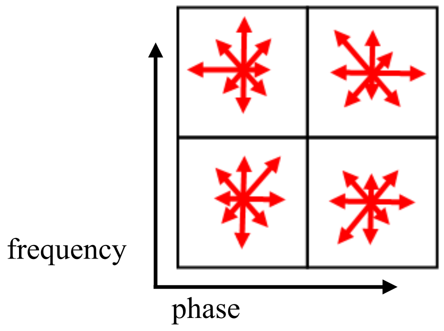


Figure 5-1: Four components in wavelet transform.
We can use the same diagram as for SIFT to represent the Gabor wavelet. However, here the horizontal and vertical dimensions do not simply represent spatial geometry, they represent phase and frequency.

In Figure 5-1 and from consideration of the nature of these parameters, it can be seen that frequency admits a 1-d Euclidean metric, phase admits a 1-d Euclidean metric, and orientation admits a 1-d Euclidean metric (at least locally). Thus geometric matching for the Gabor descriptor occurs in 3 different linear dimensions.

## 5.2 Geometric Distance Metrics

The Euclidean distance metric on the Gabor descriptor can be seen in Figure 5-2 (or in the form of Figure 4-1).

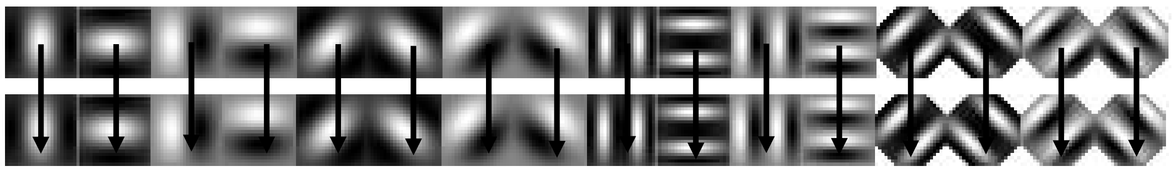


Figure 5-2: The standard Euclidean difference on the Gabor wavelet simply differences corresponding components

Applying the concept of the geometric match, where we allow matching to neighbour dimensions in frequency, phase and orientation, gives a distance between descriptors as in Figures 5-3 and 5-4.

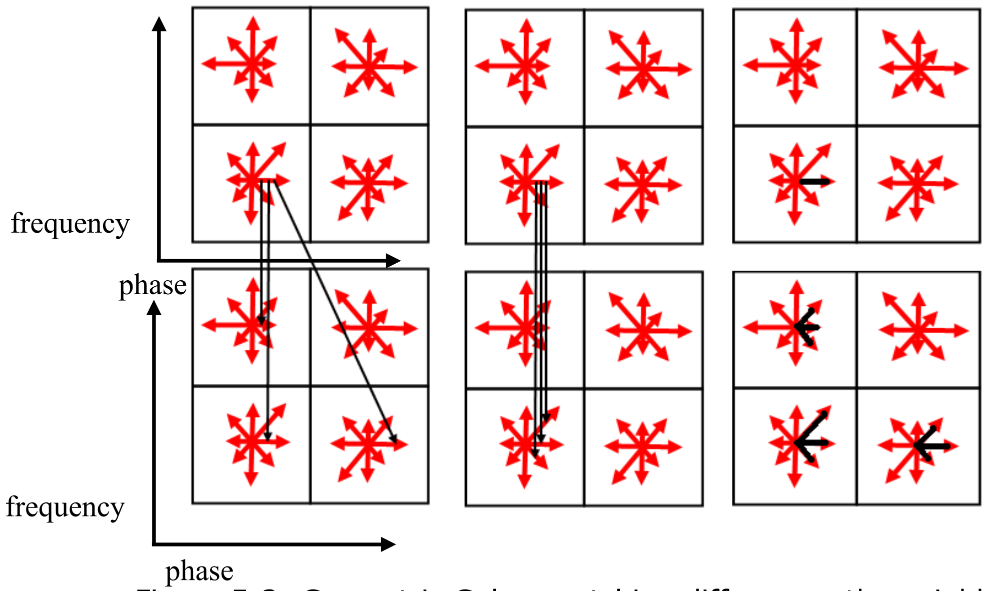


Figure 5-3: Geometric Gabor matching differences the neighbors in space, frequency and phase

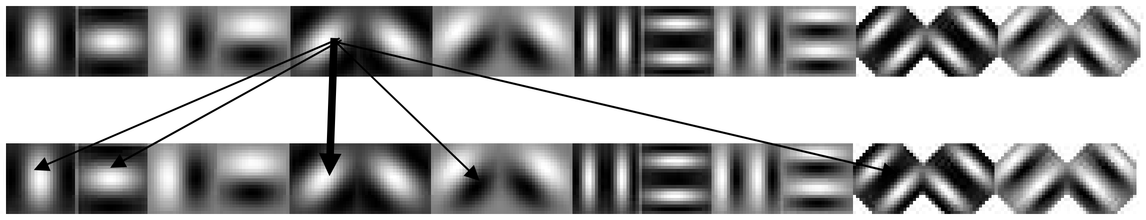


Figure 5-4: The neighbors in space, frequency and phase corresponds to these components of the descriptor vector

Applying these differences, and maintaining a weighting for the Euclidean corresponding point, gives a difference metric of-

$$
\begin{aligned}
d_{corres} &= x_{f,p,o} - y_{f,p,o} \\
d_{freq\_neigh} &= \min_{r=-1or1}(x_{f,p,o} - y_{f+r,p,o}) \\
d_{phase\_neigh} &= \min_{r=-1or1}(x_{f,p,o} - y_{f,p+r,o}) \\
d_{orien\_neigh} &= \min_{r=-1,1}(x_{f,p,o} - y_{f,p,o+r})
\end{aligned} \tag{5.1}
$$

$$
d = \sum_{i,j} \min(d_{corres}, k_1 * d_{freq\_neigh}, k_2 * d_{phase\_neigh}, k_3 * d_{orien\_neigh})
$$

As previously, the parameters $k_1$, $k_2$, $k_3$ must be determined empirically for best performance.

## 5.2.1 Affine Invariance

The parameter k1 allows the wavelet convolution in a given direction to vary in frequency space. This corresponds to a scale change in that direction, which is an affine transform. By allowing unequal scale changes in different directions, together with rotation invariance, we have an approximation to an affine transform. Effectively we are estimating 3 of the 4 non-translation parameters of an affine transform. Geometric matching for correlation and SIFT descriptors also allows some kind of affine invariance, but in a less explicit manner (although it maybe more effective).

## 5.3 Size, Weightings, Cascading, Smoothing

As previously, applying a geometric match allows a size increase for the descriptor. For Euclidean matching, the optimum Gabor descriptor size is 4×2×2 (orientation, phase, frequency). Use of geometric matching allows this to increase to 6×2×2 and provides a commensurate increase in performance.

For simplicity, edge based weightings (as per Sec. 3.2.3) are not used in this implementation. However a cascading operation, as per Sec. 3.4 is used. We first match with the lower 16-dimensional descriptor, then do geometric matching with the 24-dimensional descriptor.

Similarly we can use an energy smoothing factor, as per Sec. 3.3, to smooth the matches and make them more consistent with one another.

### 5.3.1 Complexity and Timing

Table 5.1 shows a summary of the computation required in forming and matching the descriptor (see Sec 3.46). In Table 5.1 below, G(I) denotes Gaussian smoothing over the whole image, Gabor denotes Gabor filter application, Grad. denotes taking of gradients, and Comp. denotes a comparison (subtraction and minimum operation).
As expected, there is a substantial increase in matching time as geometric matching is increased.

Table 5.1: Geometric Gabor Timings

| Descriptor Matching | Image Operations | Descriptor Operations | Pixel Operations | Point Extraction Time (s) | Matching Time (s) | Number of Points |
|---|---|---|---|---|---|---|
| Geometric Matching 2-Orientation | 4G(I) | 16*Gabor | 2 Comp. | 1.9 | 18 | 1000 |
| Geometric Matching 2-Orientation, 2 -Frequency | 4G(I) | 16*Gabor | 4 Comp. | 1.9 | 33 | 1000 |

## 5.4 Experimental Results for Gabor Matching

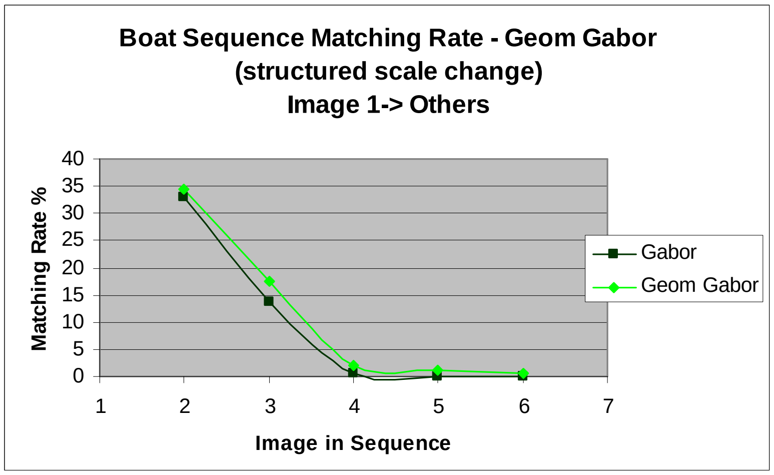
In this section we present experimental results on the same Oxford image sets and using the same DoG points as previously.

We present data sequences results for

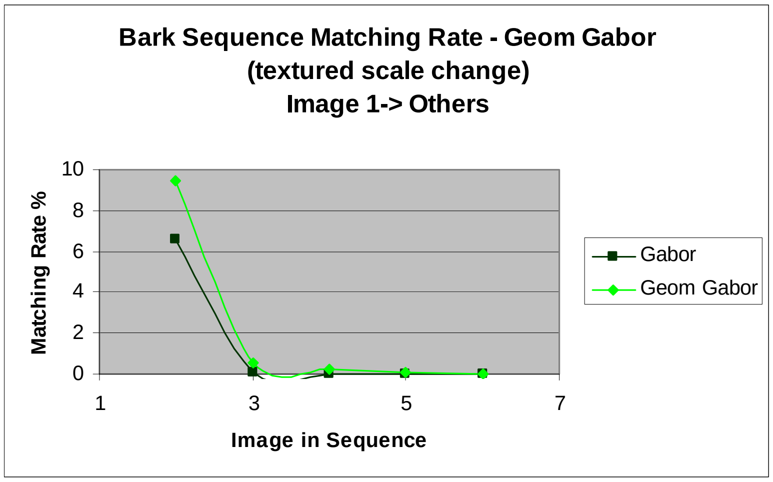
- Gabor descriptors with Euclidean matching – 'Gabor'
- Gabor descriptors with geometric matching – 'Geom Gabor'

Table 5.2: Parameters of the experiments

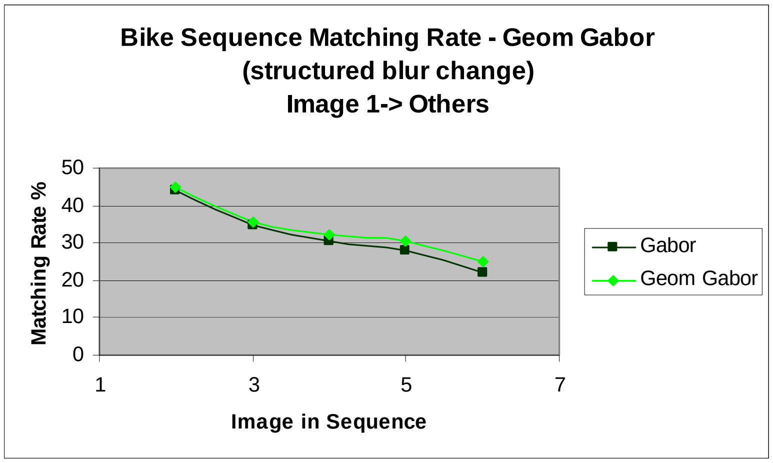
| **Experimental Parameters** | |
|---|---|
| Number of initial DoG points | 1000 |
| Dimensions (1st level) | 16 (4ori×2freq×2phase) |
| Dimensions (2nd level) | 24 (6ori×2freq×2phase) |
| Geometric match type | Simple neighbors (orientation linear) |
| Gabor wavelet λ | 25.0 |
| Gabor wavelet σ | 10.5 |
| γ | 1 |
| K Central weighting (orientation) | 1.8 |
| | |
| Timing for interest point and descriptor extraction | 7.60s |



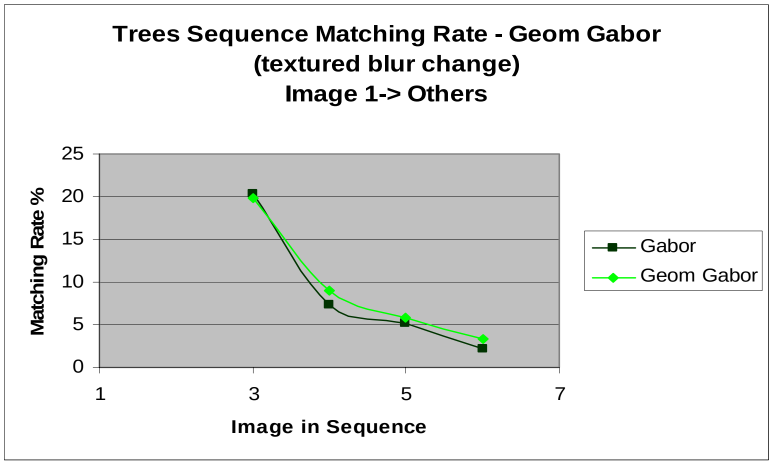
(a) Structured scale change



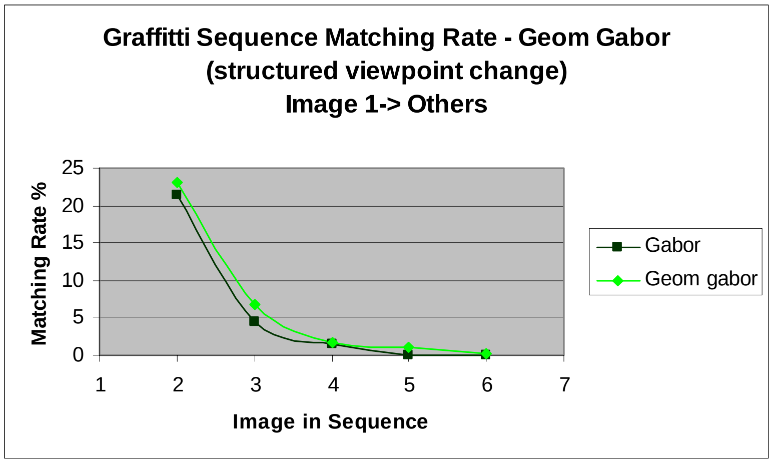
(b) Textured scale change



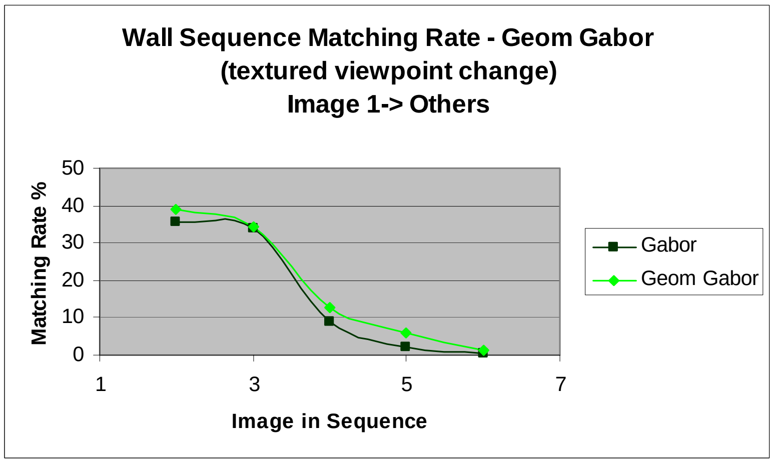
(c) Structured blur change



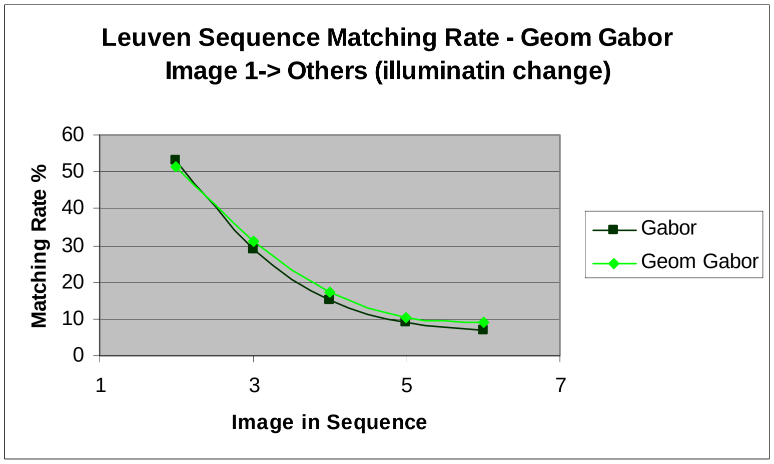
(d) Textured blur change



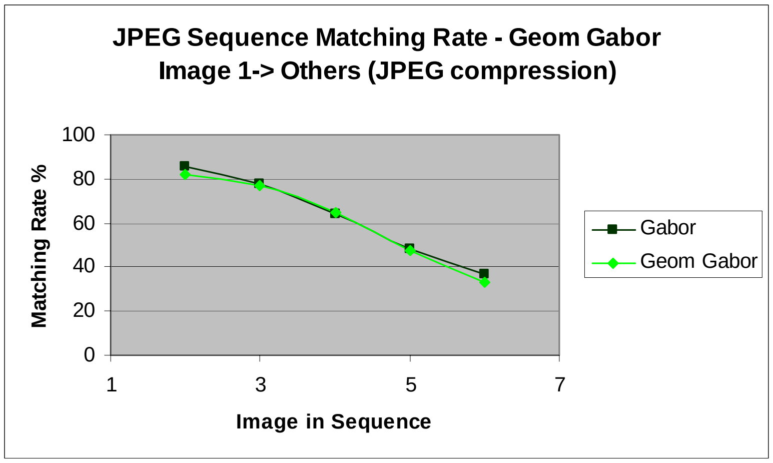
(e) Structured viewpoint change



(f) Textured viewpoint change



(g) Illumination change



(h) JPEG compression change

Figure 5-5: Various changes to matching rate in Gabor matching.

Table 5.3: Summary Statistics

| | Textured | Structured | Viewpoint | Scale | Blur | All |
|---|---|---|---|---|---|---|
| | | | | | | |
| Gabor | 8.76 | 15.60 | 10.82 | 5.38 | 21.62 | 20.05 |
| Geom Gabor | 10.12 | 17.12 | 12.61 | 6.59 | 22.94 | 21.08 |

## 5.4.1 Discussion of Results

From Figures 5-5 (a) – (f) above we can see that geometric matching provides a much smaller improvement for Gabor descriptors. Also, since Gabor descriptors are a low dimensional descriptor they have an overall lower performance than SIFT or correlation. Nonetheless, some small improvement is evident for scale, blur and viewpoint change. Also, the distinction between structured and textured again makes no significant performance difference.

For illumination and JPEG we see no improvement from the geometric match.

These results demonstrate that geometric matching can provide a small improvement for this Gabor descriptor. However, it is much less than that recorded for SIFT and correlation descriptors. This seems to be because the Gabor descriptor is primarily an orientation based descriptor, while geometric matching responds most strongly to spatial information. Better performance may be achievable if we had more frequency values rather than just 2, and also more orientations. The smaller improvement in performance is also related to this being a 16-dimensional descriptor, while correlation and SIFT have > 100 dimensions. Geometric matching seems to respond most effectively to higher dimensional descriptors.

The standard Gabor results here are also higher than other authors have reported for low-dimensional descriptors. The geometrically matched Gabor descriptor therefore outperforms other popular low-dimensional methods such as steerable filters [36, 70].

# Part III Dense Point Detectors

# Chapter 6 Dense Point Detectors

## 6.1 Introduction

In this chapter we give a general introduction to the ideas that will be used in the following 3 chapters.

In Sec. 2.9 some of the limits of current detector based methods were discussed. Ideally, we seek a detector method that combines discriminative power with the ability to deal with many different classes or points simultaneously. *The primary motivation for this is that current detectors such as DoG or LoG points do not necessarily correspond to class discriminative points*. DoG and LoG are non-descriptor, or gradient/scale-space based detectors, and are largely independent of descriptor properties. An example of a descriptor-based detector is the Viola-Jones detection scheme. We would like to combine some properties of the Viola-Jones model, with some of those of the standard detector method.

To perform effectively in the problem of class recognition we believe that descriptor based detectors (such as Viola-Jones templates) are critical. Unlike the Viola-Jones method however, we do not want to use descriptors that are specifically selected for each individual class. Rather we seek a method that uses standard descriptors but that can deal with many classes simultaneously. A key aspect of this is that we must use *dense* detectors i.e. descriptors need to be calculated at every image point, and point selection is based on the descriptor.

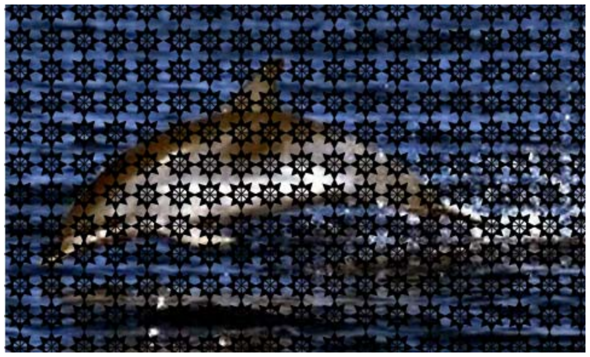


Figure 6-1: Dense interest point descriptors.

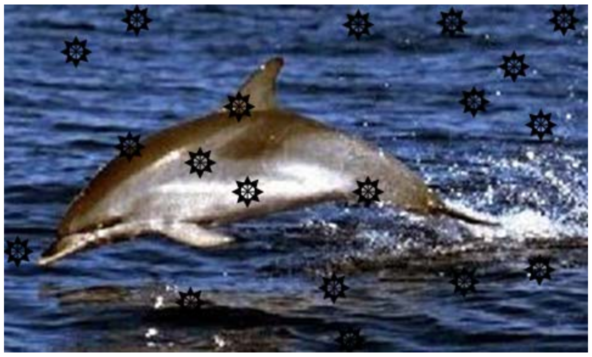


Figure 6-2: Sparse interest point descriptors.

Sparse point detectors have been the most popular in computer vision for the last 10 years [62, 68]. Dense detectors have tended to be avoided partially due to the computational cost of constructing and matching them. They have however been used in [104, 99]. For illustration see Figures 6-2 and 6-2.

*Definition 6.1*: A *dense* detector is one that calculates a descriptor at every image point. A *sparse* detector is one that calculates a descriptor at only a subset of points.

For the reasons outlined in Sec. 2.3.4 we believe that Gabor wavelets are very suitable for this task. The Viola-Jones primitives themselves are similar to a discretization of Gabor wavelets. A principle here is to generalize parts of the Viola-Jones concept to a general interest point setting. If we are using Gabor wavelets, what is the best way to do this? The Viola-Jones templates are carefully selected in a sequence specific for a particular class.

A large number of templates are needed to achieve reliable single class recognition. How then can we generalize this to many classes?

If we examine the Viola-Jones scheme, it applies a filter at each step, and then compares this to a binary threshold. This is a lookup into a 1-dimensional space (see Figure 6-3). Thus a specific sequence of 1-dimensional lookups is required to obtain discriminative power for a single class.

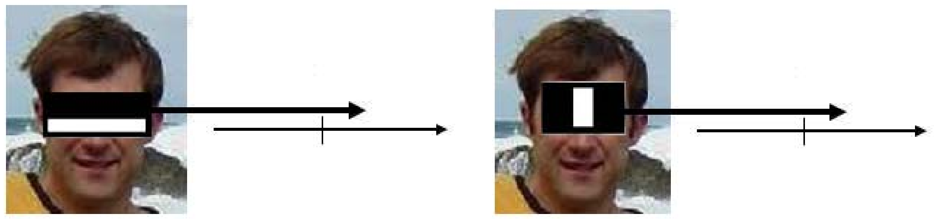


Figure 6-3: Viola-Jones recognition performs a 1-dimensional lookup at each point.

If we are using general 1-dimensional templates, not optimized for a particular class, such a sequence is unlikely to be discriminative. One obvious generalization of this is to change it from a lookup into a 1-dimensional space to a lookup into a higher dimensional space. By using general templates, such as Gabor wavelets, we lose discriminative power for a class, compared to using specifically selected templates. However, by using higher dimensional templates and lookup into a higher dimensional space we gain discriminative power. Thus there is a tradeoff between generality and dimensionality. Using more dimensions for the lookup, compared to Viola-Jones, allows us to use more general templates while maintaining discrimination. Such higher dimensional templates can be formed by using a set of Gabor wavelet filters, each filter providing 1-dimension. In such a way, a set of Gabor wavelets can provide an n-dimensional descriptor at each point. We then perform lookup into an n-dimensional space to determine the object at that point. Effectively we are doing interest point descriptor matching at every point.

Aspects of the Viola-Jones scheme that we have not emulated are the cascading operation or the use of Adaboost, which many consider are its main features. Cascading operation is something we may introduce in future but is not currently a part of our algorithm. We do not use Adaboost since we want generic descriptors suitable for many simultaneous classes. Here the aspect of the Viola-Jones scheme that we are interested is that the descriptor is the detector, and it performs a recognition lookup on every point.

## 6.2 The Point Descriptor

Since we are looking to compute a descriptor for each point in the image we seek a descriptor that is computationally efficient. For the class recognition case it is also critical to be of low dimensionality. A popular descriptor such as SIFT is not suitable there since it is of high dimension. The main reason is that high dimensionality adversely affects computing the nearest neighbour by approximate methods [12]; it also increases the storage requirements (see Figure 6-8). Similarly we dispense with affine invariant or other special descriptors for reasons of computational complexity. Gabor wavelets are able to form a low dimensional descriptor and also provide good matching performance.

Various studies (e.g. [63, 93]) have shown that orientation is a critical property for point descriptors. Even for a low dimensional descriptor we will therefore apply Gabor filters of at least several different orientations. Being a frequency localized function, Gabor wavelets also have a phase (see Eq. 2.5). To be phase independent we will use a Hilbert transform couplet, i.e. pairs of wavelets with a phase difference of $\frac{\pi}{2}$. We have found frequency is not as critical as orientation for descriptor matching, so we will generally only have a small number of frequencies. Our default basic descriptor is a 16-dimensional vector – 4 dimensions for orientation × 2 for phase × 2 for frequency (Figure 6-4). Thus we are applying a Gabor wavelet filter bank to the image.

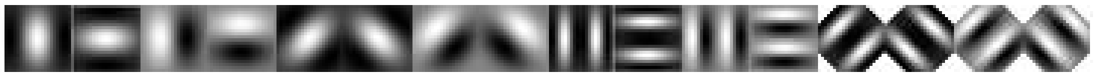


Figure 6-4: The Gabor descriptor we calculate at every pixel.

Although we are using a Gabor wavelet descriptor here, it is possible to use many different descriptors with dense descriptor detection. To some degree, performance depends on how the dense descriptor is implemented not the descriptor itself. However, one key condition for class recognition is that it is a low-dimensional descriptor to allow accurate approximate nearest neighbour matching (see Sec. 6.3.1). For point-to-point matching higher dimensional descriptors may be better.

Other Descriptors

The dense descriptor concept does not intrinsically depend on any particular descriptor. We use Gabor wavelets simply because they have demonstrated good performance on structure matching tasks. We believe color and texture dense descriptors could also be superior in some situations. This is because some classes are intrinsically more easily discriminated by color or texture than by structure. We will later be using an example of a dense histogram application.

## 6.3 Dense Descriptor Matching

Having calculated a descriptor at each point we then perform a lookup into the space of enrolled descriptors. This is effectively standard interest point matching, but requires some changes in this case.

In the Viola-Jones scheme the lookup is a 1-dimensional threshold comparison (see Figure 6-3). This is possible because a 1-dimensional space can be completely divided into two parts. A higher dimensional Euclidean space does not afford this topological possibility. In such a space we will use the Euclidean distance as our decision function. Straight nearest neighbour matching is not appropriate since not every point has a match in the database. Therefore we perform Euclidean nearest neighbour matching with an additional threshold distance as

$\left(|x_1 - y_1|^2 + |x_2 - y_2|^2 + \ldots + |x_n - y_n|^2\right)^{\frac{1}{2}} < threshold$ where $x$ is an image point, and $y$ is an enrolled database point (Figure 6-5).

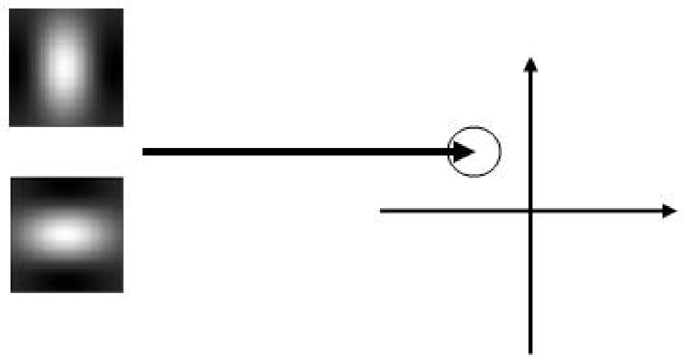


Figure 6-5: Dense matching in 2D.
Performing a 2-D lookup at each point, as compared to Viola-Jones 1-D lookup of Figure 6-4.

## 6.3.1 Approximate nearest neighbour

Since we are not doing database point matching for a small set of points (as per sparse descriptors) but potentially for every image point, we want to accelerate this as much as possible. Thus approximate nearest neighbour matching is critical for dense descriptors [8, 12].

The method we have developed for this involves partitioning the descriptor space into a number of bins and then restricting our search to a certain bin or bins. The way this has been done differs from other well-known algorithms. This allows a substantial speed up in matching performance, and obtains the correct match a high proportion of the time.

We give an example assuming we are using a Gabor wavelet of 8-dimensions e.g. 8 orientations, or 4 orientations & 2 phases. Since the wavelets are set to zero mean, we can partition each dimension into 2 sections or bins, [-∞, 0], [0, ∞] (see Figure 6-6). Then we have partitioned the descriptor space into 256 bins. Empirically, we have found this partition gives good performance.

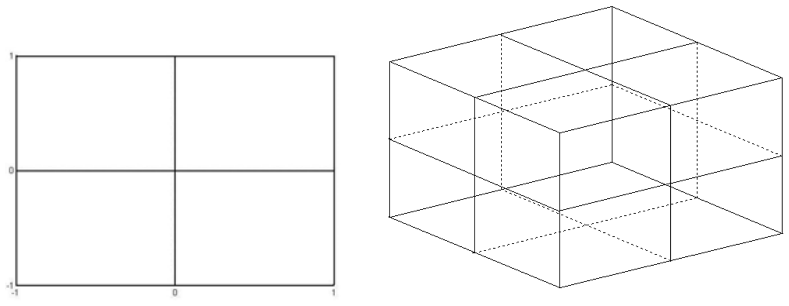


Figure 6-6: Approximation in space.
Dividing up the descriptor space into binary bins, then searching with in those. Here we illustrate the 2D and 3D case. In practice our descriptor space is of higher dimension

This can be implemented in C++ using a vector in a high-dimensional data structure.

**Enrolment**

For each image point, the 8-dimensional descriptor values are used to place this point into one of the 256 bins. In the simplest implementation, that is the end of the enrolment process. In recognition on a new image, we then simply iterate through the points in that bin to find $L_2$ matches below the threshold value.

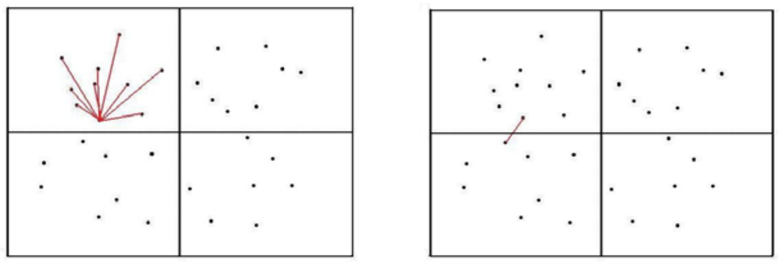


Figure 6-7: Principles of approximation methods.
Approximation methods speed up matching by restricting the search to 1 bin, but introduce errors if the nearest neighbor is outside the bin. In high dimensions this has an increased chance.

This simple algorithm gives poor results in practice because under viewing changes the descriptor values of a matched point will always have changed slightly, and this may have tipped the point over into an adjacent bin (see Figure 6-7). During point enrolment we therefore enroll the point not only in the bin in to which its descriptor corresponds (its *native* bin), *but also to any other bins within a descriptor radius d of the point.* We are therefore enrolling the point in any bins that intersect the hypersphere of radius *d* around the point (Figure 6-8). *d* can be set appropriately for the particular data we are dealing with. The optimal value of *d* is a statistical property of the data and the application on hand. For points that lie deep within their bin (i.e. have large values for all dimensions) it may be that no other bins are within *d* of the point. In such a case the point is still only enrolled in one bin. However, for a point whose descriptor values are close to the origin, it could conceivably be enrolled in all 256 bins.

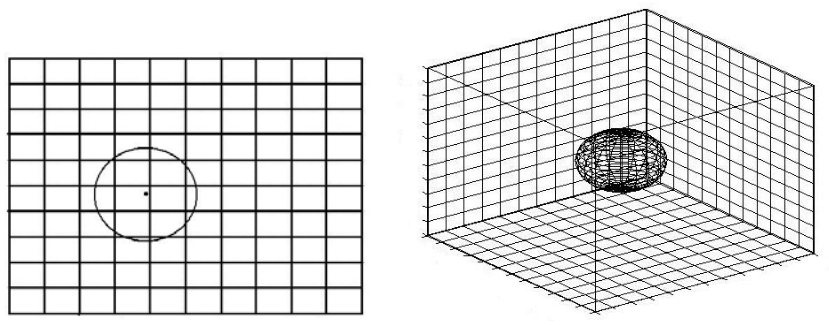


Figure 6-8: High dimensional similarity search.
Searching in nearby cells produces a circle in 2D, a sphere in 3D, and we use a hypersphere in high dimensions. Higher dimensionality dramatically increases the number of bins that must be searched within the hypersphere.

## Recognition

During the recognition of a new point we iterate through the bin to find the minimum $L_2$ match, recording the current minimum. Practically, this means iterating through the vector data structure for that bin. We do this for all bins within a hypersphere of radius *f* of the point, if the distance to that bin is less than the current minimum. In this way, the hypersphere is effectively used twice – when points are enrolled in the bins, and in determining nearby bins during recognition (Figure 6-9). This adds complexity to the algorithm – the enrolment process is somewhat slow - but it speeds up recognition. This is a deliberate choice we have made, since enrolment only occurs once and we can accept some slowness in this process. It is more important that recognition thereafter occurs quickly.

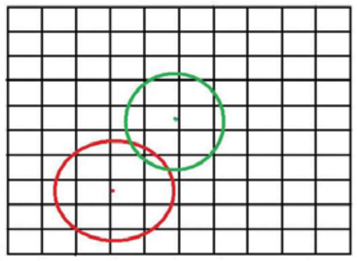


Figure 6-9: Common area in hypersphere.
Enrolling in a hypersphere around a point and then searching in another hypersphere around the other point, allows smaller hyperspheres to be used.

It can be seen from Figure 6-8 that increasing the dimensionality dramatically increases the total number of bins, and also the number that intersect the hypersphere. The total number of bins determines storage requirements; the number that intersects the hypersphere determines computation time of a match. This provides a good demonstration of the 'curse of dimensionality'.

A popular alternative algorithm for approximate nearest neighbour calculation is Best Bin First [12]. Like our algorithm this uses the concept of the hypersphere. However, it only uses it once, during recognition lookup. It does not use it during point enrolment. In that way we believe our algorithm makes more efficient use of the data. The cost of this is that our algorithm is slower during enrolment and has more elaborate storage requirements. Both the BBF and our algorithm are based on approximating the hypersphere with n-cubes (n-dimensional 'cubes') but our algorithm shifts more of the computation to the initial enrolment, rather than each recognition lookup. Figure 6-10 illustrates the main idea. Nearest neighbour matching is not the main object of this thesis, therefore we have not conducted a rigorous study comparing it to BBF.

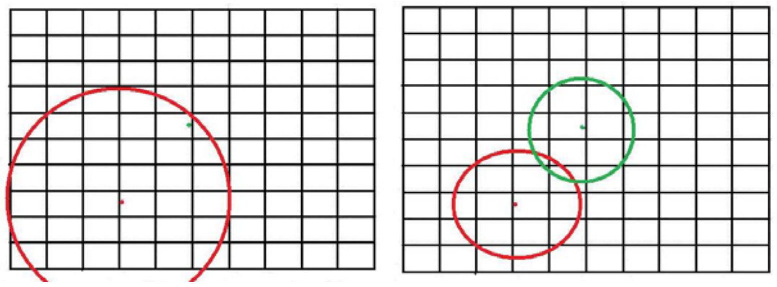


Figure 6-10: High dimensional search.
We can see that the area of the 2 smaller circles together is smaller than the area of the single circle. Thus both enrolling and searching in a hypersphere is more efficient than just searching in a hypersphere. In high dimensions the effect has a much greater difference. This corresponds to a smaller number of bins to search and matches to perform.

## 6.4 Further Aspects

For the most common matching tasks we have found that strong descriptor points are consistently more reliable or useful for matching. Strong descriptors are indicative of clear or strong structure in the image. Weak descriptor points are indicative of an absence of clear structure in the image at that point e.g background, clutter, walls, sky, etc. Weak descriptor points tend to give more random measurements when re-imaged with viewing changes.

For this reason we select only those points in the top *p*% of all points by descriptor length. Descriptor size is taken as the Euclidean length, $L_2$. We can vary *p* according to the given task. For simple point matching *p* can be quite small, e.g. we could choose only those points in the top 5% of length. For class matching, we should allow a greater *p*. This is because we need to find those points that are class discriminative and these points may not necessarily be very strong. i.e. it is possible to have class discriminative points based on image structure of medium strength.

One of the key features of this approach is that points can be enrolled in the database by any criteria. Points could be enrolled in the database for their stability, distinctiveness, class discrimination, etc. Every point in the new image is tested for a Euclidean distance match, without regard to the criteria by which points were entered in the database space. This is different from the SIFT approach, for example, whereby points are enrolled and detection according to the same criteria like maximum of DoG. Inherent in our approach is a fundamental asymmetry between enrolment and recognition (which SIFT, for example, does not make). This allows more flexibility in recognition tasks.

**How is this approach different from traditional interest point usage?**

- Points are first enrolled in a database according to any criteria, such as class discrimination power, rather than DoG or LoG
- A multi-dimensional descriptor is calculated for each image pixel and matching can be done for each point i.e. it is a *dense* method. Generally, however, we only do matching for the strongest *p*% of points.
- Matching is done according to a Euclidean matching distance < threshold, rather than simply nearest neighbour. This is since not every point in the database will have a corresponding image point.
- Matching requires a fast approximate nearest neighbour algorithm

There is some limited similarity between our approach and the 'Biological' model (Sec. 2.4, or [93]) since both use Gabor wavelets. However the 'Biological' model is less flexible in that it always selects the maximum point from an 8×8 pixel window, and that it extracts a fixed number of interest points from the image. Enrolment and recognition of points are also symmetric, so it can only recognize similar points (maxima in 8×8 window).

# Chapter 7 Dense Point-to-Point Matching

## 7.1 Introduction

In this chapter we will use the ideas of the previous chapter to implement a practical recognition system for point-to-point matching. Having developed a plausible recognition framework in the previous chapter, we now want to see if it can give good results in practice. Specifically, we are interested to see if dense methods can give results competitive with detector/descriptors such as DoG and SIFT.

## 7.2 Point Selection for Point-to-Point Matching

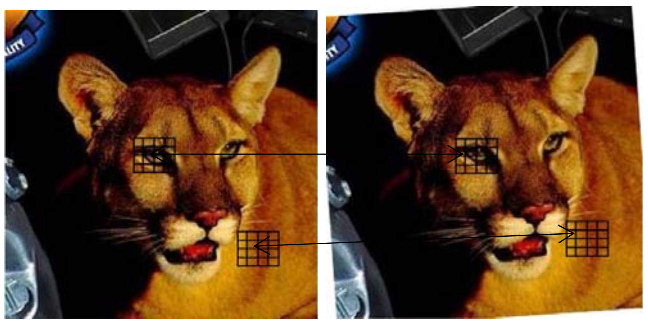
The previous chapter outlines a recognition scheme but some key issues were not addressed. One of the main ones was, by what criteria should points be initially selected? i.e. how do we select points in the first place to enroll in the database?

In enrolling or selecting points for point-to-point matching our main aim is to achieve high point recognition under image viewing transformations. For example, if we select points from an image we want to be able to accurately locate them under a perspective or scale change. Ideally we want points that are robust under all the image variations listed in Sec. 1.1.

If we have calculated descriptors for all image points, what points are the most repeatable? If we were to match descriptors for all image points against one another, what characterizes those points that we can locate again most reliably? Since we are matching by distance to descriptor points in a database, what we seek are descriptors that are *stable*. Are there some properties of the descriptor that indicate stability?

### Descriptor Length

An obvious one to try is descriptor *length* which is determined by the presence of structure in the image. A simple measure of length is the descriptor $L_2$ (the descriptor is an n-dimensional vector). Indeed, when we have tested this we do find that it is a good predictor of descriptor stability. Those points with strong descriptors tend to remain of reasonably consistent value after viewing changes (see Figure 7-1). Moreover, the strongest points exist in sparse regions of the descriptor space and are therefore robust against matching to other points (Figure 7-2). However, length is a very simple parameter; there may be more sophisticated ones that are even more stable.



(a) (b)

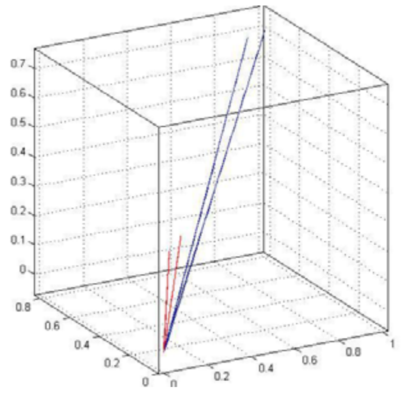
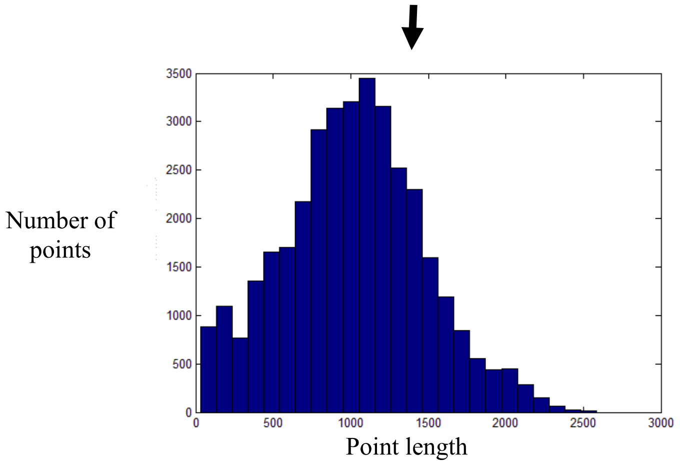


Figure 7-1: Strong and weak features.
Here an image undergoes a small rotational transform (a), (b).
Stronger image features have longer descriptors which change less proportionately compared to weaker features. This makes strong features better for recognition.

(a)



(b)

Figure 7-2: Histogram of descriptor length.
If descriptors calculated at every pixel for a typical image (a) we obtain a histogram as per (b). As the length increases the number of points tapers off. This results in a dramatic reduction in descriptor space density for strong points.

**Descriptor Space Density Measurement**

Another idea is to look for point descriptors that are in sparse or low-density regions of descriptor space. These might be less likely to get mistaken for other points, compared to points in dense regions. In dense regions there are many other points with similar descriptor vectors which, under re-imaging, could be mistaken for the correct point (see Figure 7-3).

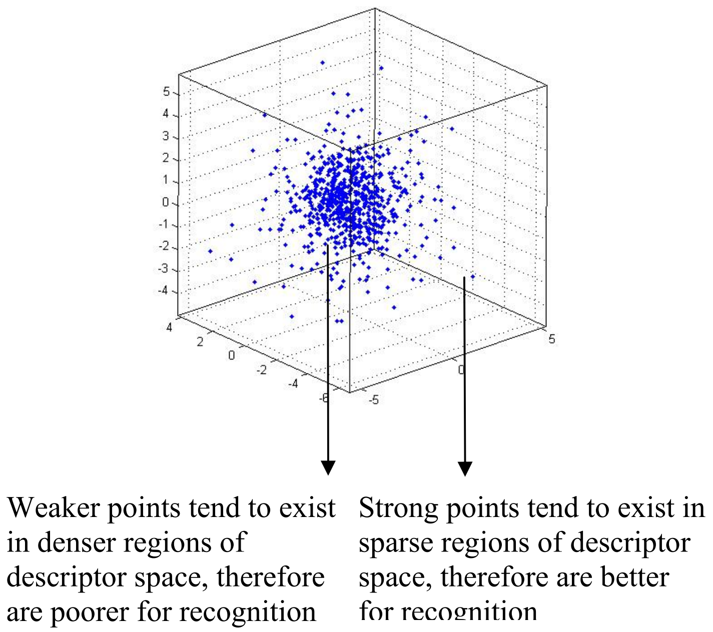


Figure 7-3: Descriptor point strength related to density in descriptor space

Measurement of the density of regions of descriptor space is non-trivial. Since we are already partitioning the descriptor space into a number of bins, e.g. 256, we could just look at the number of points in each bin and take this as a density measure (Figure 7-4). In experiments we have found this to offer poor performance. The descriptor bins are too coarse a scale over which to measure density.

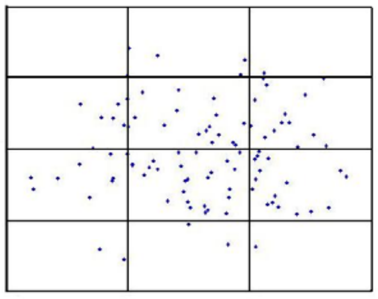


Figure 7-4: Measuring density by using feature bin.
One way to measure the descriptor space density around a point is to measure it for its bin, i.e. count the number of points in its bin.

Another way to determine density is to look at the local region around a point in descriptor space. As shown in Figure 7-5, by looking at the distance to the nearest neighbour in the enrolment image, or the number of such neighbors within a certain radius, we can measure density. Such calculations are more computationally complex to make, but we have found they do indicate matching performance to some degree. i.e. local density does give some matching performance information, but larger scale density measures do not. However, the density effect is not very strong. The length of a descriptor is a much better indicator of its matching performance than the descriptor region density.

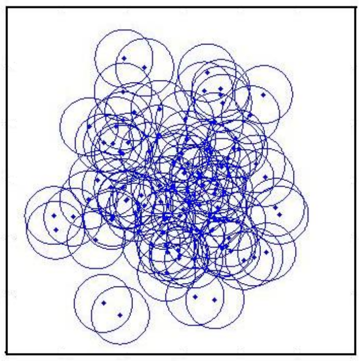


Figure 7-5: Measuring density by using radius.
Another way to measure the descriptor space density around a point is to calculate the number of points in a certain radius. This is more computationally demanding but gives better results.

## Transformed Image Descriptor Stability

Another idea is to apply a small synthetic transform to the image and see how the descriptor changes. For example, we could apply a small affine transform or a Gaussian blur, and examine how much the descriptor changes. A point descriptor that changes only a little is stable and therefore that point is propitious for further usage.

$$\Delta\text{descriptor} = \sqrt{(\sum(d_i(\text{image}(x,y)) - d_i(\text{transformed image}(X,Y))^2)}$$

$$d_i \text{ is the } i^{th} \text{ element of the descriptor vector } d \qquad (7.1)$$

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

*H* is the homography between the 2 images.

For example, in Figures 7.1 (a) and (b) above a small rotation is applied between the images. We then test corresponding points to identify those points whose descriptors that have a minimum change.

In general we have found this quantity gives good results i.e. *small descriptor changes under an applied image transform are indicative of a stable point*, and give good matching performance. In testing we have found this approach to give best results when it is used in conjunction with descriptor length. This has excellent performance, better than descriptor length alone, or any other parameters we tried. We first select those points with a strong descriptor response, then sort these by the change in the descriptor under a small transform. The transform we have generally used is a small rotation and blur, but a general affine transform would give better results. We have used a rotation and blur only for computation speed. We then select those points with minimum descriptor change in the number we need for a given application. One other benefit of this method is that we can choose however many points we need, unlike traditional detector methods.

| **Effectiveness of Dense Point Selection Quantities** |
|---|
| Descriptor space low density measurement |
| Descriptor stability under image transform |
| Descriptor length |
| Descriptor length & stability under image transform |

**Summary of Dense Point Matching Method**

Point enrolment-

- Descriptors are calculated for every point.
- The longest *p*% of points are selected for further processing
- Those points with the minimum descriptor change under an applied image transform Eq. (7.1) are enrolled

Point recognition-

- Descriptors are calculated for every point.
- The longest *p*% of points are subject to approximate nearest neighbour matching

Table 7.1: Comparison of dense and sparse interest point matching

| **Dense Interest Point Method** | **Sparse Interest Point Method** |
|---|---|
| Descriptors at every point | Descriptors at sparse points |
| Points selected by descriptor length, and stability under image transform | Points selected by image scale space properties (e.g. DoG, LoG) |

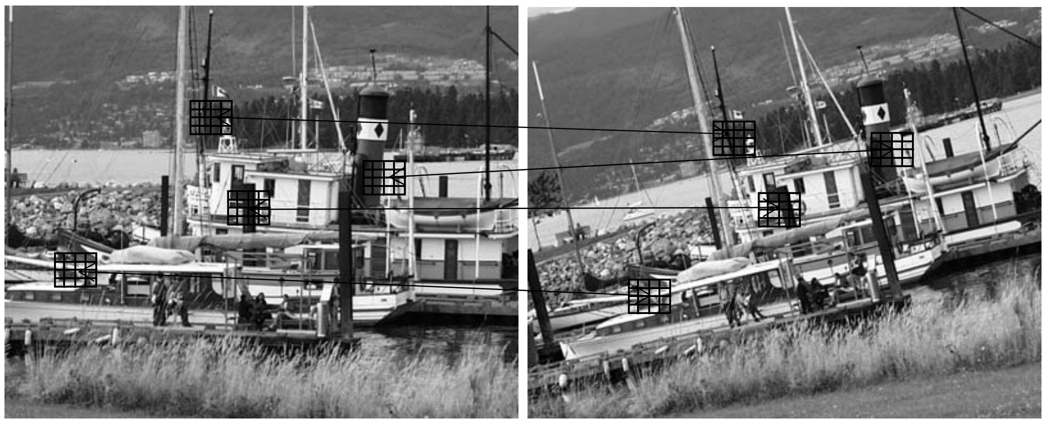
The main advantages of this method, compared to traditional point matching are

- Higher matching or recognition rate
- Achieves good matching rate with low dimensional descriptor
- The number of points in the image can be controlled
- The image density of points can be controlled

Having selected the points we then enroll them in the descriptor space for approximate nearest neighbour lookup, as per 6.3.1.

## 7.3 Point-to-Point Matching System

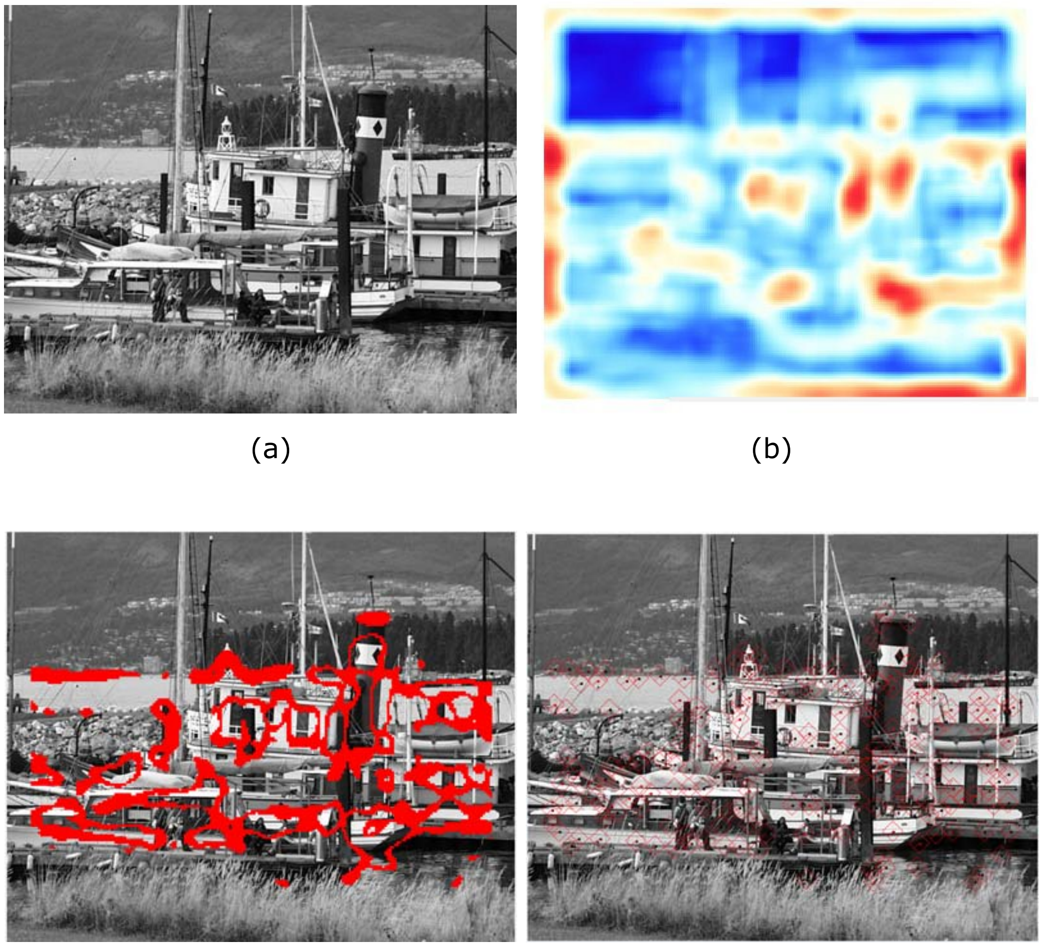
We now discuss the implementation and performance of a point-to-point matching system in more detail. We have seen that the length and stability of a point descriptor under an image transform gives a good indication of its usefulness; we therefore develop this approach.



(a) (b)

Figure 7-6: Descriptor changes.
For (a) and (b) the rotation angle is 13.9° and the scale reduction is 12%. We look at the size of the descriptor change for image points.

In the example of Figure 7-6 we have applied a small rotation and a scale reduction. Ideally we would like an infinitesimal change in both, since we want to determine the descriptor stability at exactly that point. Descriptors are then calculated for every point in the original and transformed image. Since the transform is synthetically applied, we can exactly calculate corresponding points.



(a) (b)

(c) (d)

Figure 7-7: Stable and strong points.

(a) An image and (b) a color map of its descriptor lengths. The strongest responses indicate the strongest structure in the image. (c) We select the strongest points for further processing. Here we have taken the longest ~10%. From the strongest points we apply a small rotation and blur. (d) Those points with minimum descriptor change are considered the most stable. They are our final points selected.

Points are selected by-

1) descriptor length > $p^{th}$ percentile or threshold
2) minimum Δdescriptor by Eq. (7.1)

Points selected are then enrolled in the descriptor space. The number of points can be chosen for a given task.

Figure 7-7 (b) shows the boat image from Figure 7-7 (a) colored with a heat map according to the length of the descriptor. We can see in that strong points tend to be in connected regions that follow image structure. In some cases these are characterized by the presence of edges, in other cases by the coincidence of groups of edges or structure. Those plain or background areas produce few strong points as expected. From this the strongest 10% of points are extracted and these are shown in Figure 7-7 (c). These also are related to the main image structure. From these strongest 10% we then identify the most stable descriptor points by use of the applied synthetic transform Eq. (7.1), and these are shown in Figure 7-7 (d). It can be seen that it tends to identify the perimeter of the regions and structure from (b).

These stable points are our final selection. We can then subject them to experiment testing to determine how effective they are for recognition purposes.

### 7.3.1 Complexity and Timing

Table 7.1 shows a summary of the computation required in forming and matching the descriptor (see Sec 3.46). In Table 7.1 below, G(I) denotes Gaussian smoothing over the whole image, Gabor(I) denotes Gabor filter application, and Comp. denotes a comparison (subtraction operation).

Table 7.2: Dense Point Timings

| **Descriptor Matching** | **Image Operations** | **Pixel Operations** | **Point Extraction Time (s)** | **Matching Time (s)** | **Number of Points** |
|---|---|---|---|---|---|
| Euclidean | 4G(I), 16 * Gabor(I) | 1 Comp. | 6.54 | 12 | 1000 |

## 7.4 Experimental Results for Dense Point Matching

In this section we present experimental results on the same Oxford image sets as previously used. We use sparse low dimensional Gabor results as a benchmark.

We present data sequences results for

- Dense Gabor descriptors – ‘Dense Gabor’
- Traditional or sparse Gabor descriptors. These are of the same 16 dimensions for comparison – ‘Sparse Gabor’

Table 7.3: Parameters of the experiments

| **Experimental Parameters** | |
|---|---|
| Number of initial points selected by length | 2000 |
| Number of final points selected for stability & length | 1000 |
| Dimensions | 16 (4ori×2freq×2phase) |
| Descriptor space bins | 256 |
| Descriptor space enrolment radius | 0.1 |
| Gabor wavelet λ | 28.0 |
| σ | 8.4 |
| γ | 1 |
| Synthetic rotation | 0.1 rad |
| Synthetic blur | 0.1 |
| | |
| Timing for interest point and descriptor extraction | 6.54s |

One fundamental question is how many dimensions the descriptor should be. Figure 7-8 shows a maxima ~ 16 dimensions and we adopt that for further usage.

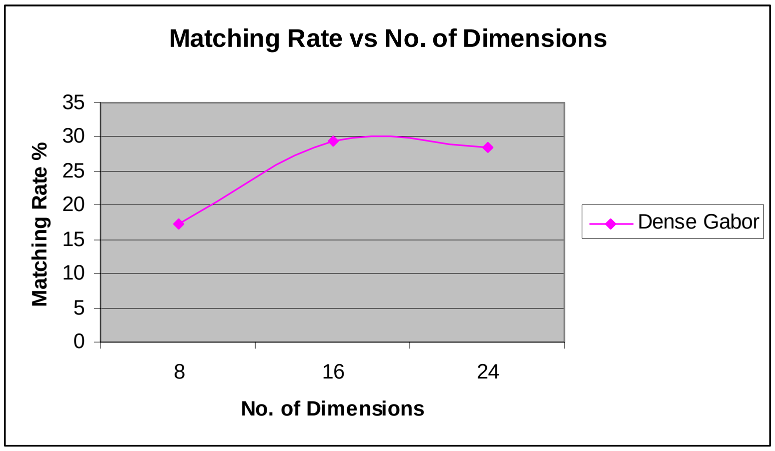


Figure 7-8: Descriptor matching rate vs number of dimensions. For dense Gabor points matching performance reaches a maximum ~ 16 dimensions. The addition of more dimensions (e.g. more frequencies) does not improve performance.

Another important question is whether matching performance depends strongly on the number of points selected from the image. In Figure 7-9 we see that this is not so.

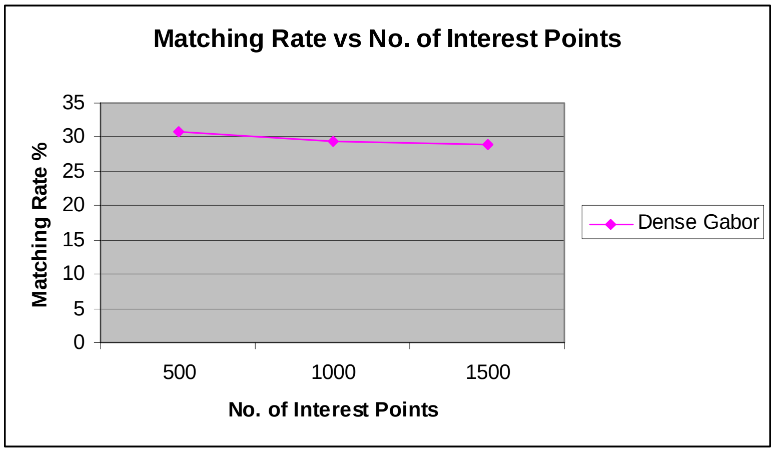


Figure 7-9: Descriptor matching rate vs number of points. Matching performance does not strongly depend on the number of interest points selected.

It is also interesting to see how the matching rate varies as we allow the pixel localization error to increase. This error is the difference in pixel location from the exact homography calculation between the corresponding images. Increasing this means we are allowing more generous definition of a successful match. We can see in Figure 7-10 that this does have a reasonable effect.

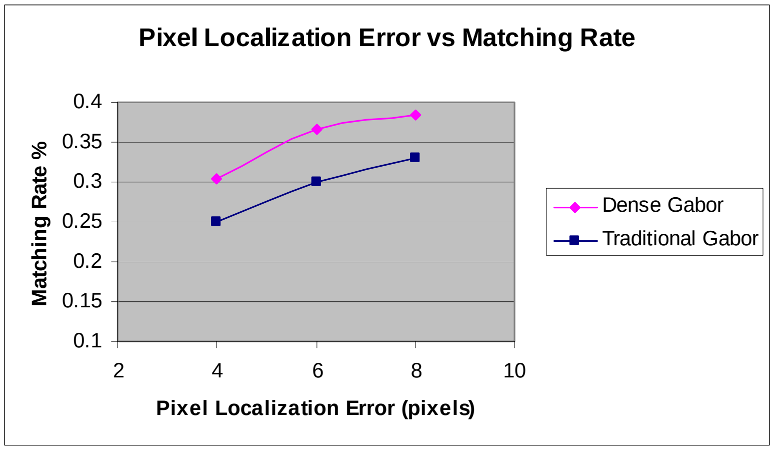
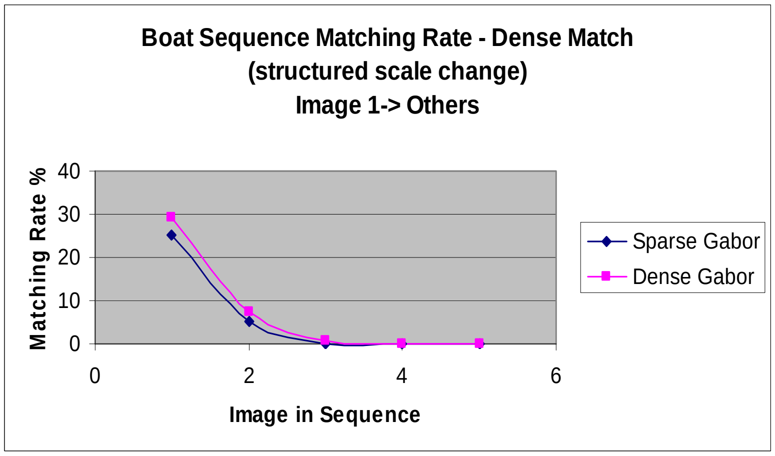
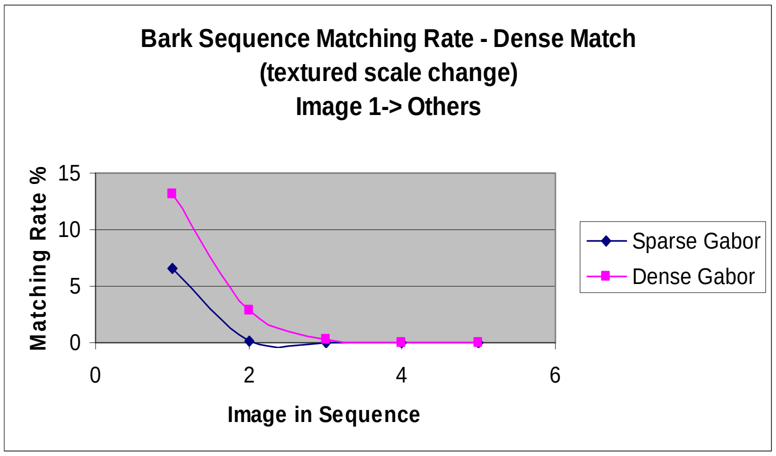


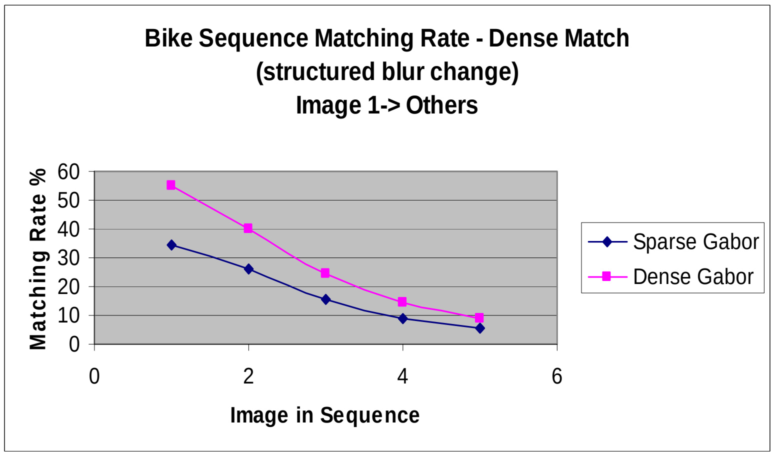
Figure 7-10: Matching rate vs pixel localization error.



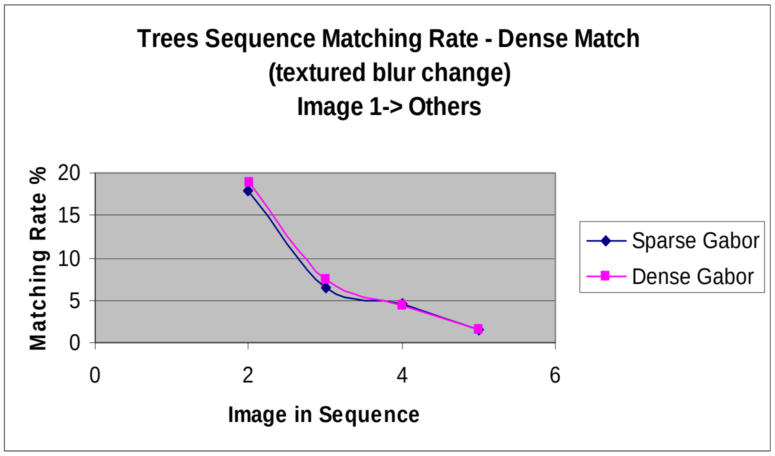
(a) Structured scale change



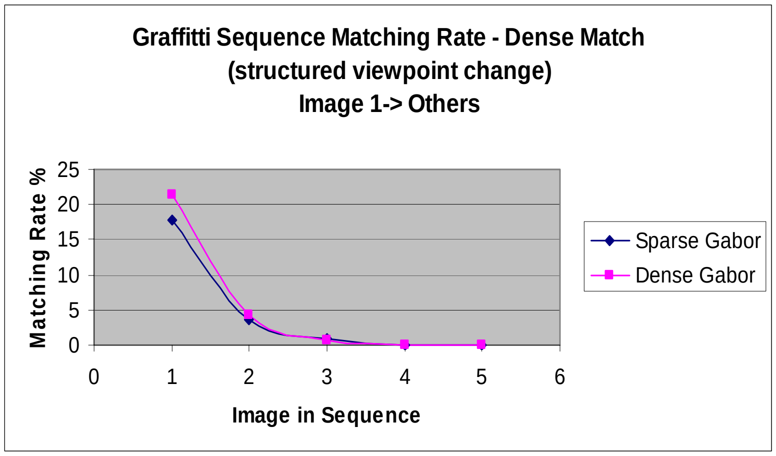
(b) Textured scale change



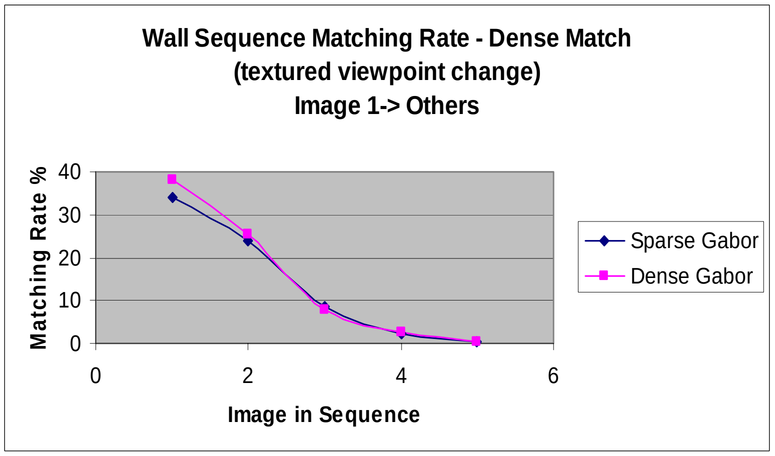
(c) Structured blur change



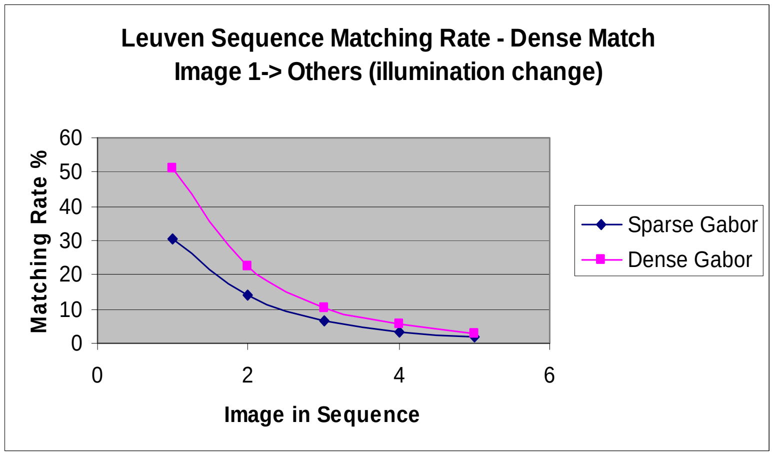
(d) Textured blur change



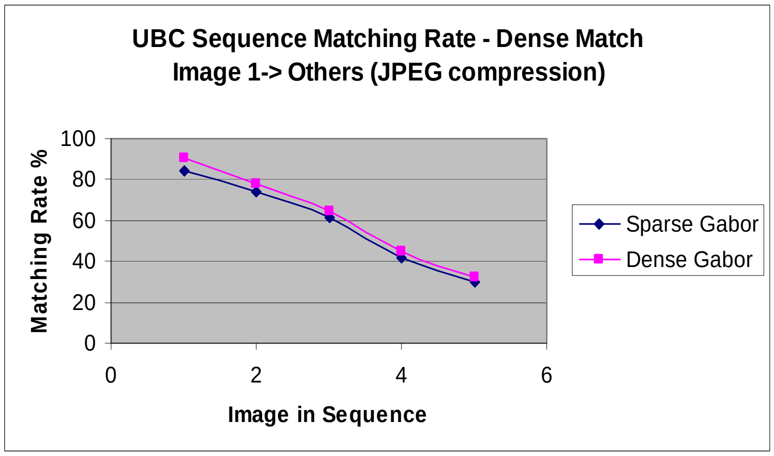
(e) Structured viewpoint change



(f) Textured viewpoint change



(g) Illumination change



(h) JPEG compression change

Figure 7-11: Dense and traditional Gabor matching rates for various image variations

Table 7.4: Summary Statistics

| | Textured | Structured | Viewpoint | Scale | Blur | All |
|---|---|---|---|---|---|---|
| | | | | | | |
| Sparse Gabor | 7.57 | 9.55 | 9.08 | 3.71 | 13.50 | 15.33 |
| Dense Gabor | 8.78 | 13.76 | 10.06 | 5.37 | 19.44 | 18.76 |

## 7.4.1 Discussion of Results

From Figures 7-11 (a) – (h) we can see that dense matching generally delivers higher matching than sparse Gabor matching. For most sequences the improvement is modest. However, for textured scale change (b), structured blur results (c) and the illumination sequence (g), it delivers a large improvement. The improvement seems to be bigger for scale and blur change than viewpoint variation. Again, no significant difference between structured and textured series is observed.

These results are interesting since effective point matching using dense descriptors has not been demonstrated previously – the most successful methods have all been sparse methods. This indicates that dense methods could become a competitive tool for point-to-point matching.

The standard Gabor results here are also higher than other authors have reported for low-dimensional descriptors. We suspect that is because we have spent some time optimizing our Gabor wavelet kernels. The matching results for the dense descriptor outperform all previously documented low dimensional methods. Dense point matching is more suitable for low dimensional descriptors. For example this descriptor markedly outperforms steerable filter performance in [70] which has the highest performance of the low-dimensional methods tested therein.

# Chapter 8 Dense Object Class Discrimination

## 8.1 Point Selection for Object Class Discrimination

In the previous chapter we selected points for point-to-point matching. In this chapter we have a similar framework but we select points for discrimination between classes of objects.

In selecting points for class discrimination we seek those points that effective for determining the presence or absence of a given class. In general, such points correspond to structure that is consistent for the class, but does not occur for non-class members. To select such points we need some way to quantify this discriminative ability.

We begin with a set of images for some class and a set of non-class images. Since we are using dense methods, we then calculate descriptors for every pixel in every image, and populate these into a descriptor space data structure (see Figure 8-1). This will generally be a high-dimensional bin structure suitable for approximate nearest neighbour matching, as described in Sec. 6.3.1.

How then can we determine what points are the most discriminative? We seek a simple metric that can indicate discriminative ability.

**Formal Classifiers such as SVMs**

A formal classifier, such as a linear Support Vector Machine (SVM) is normally used to test points or images. In principle it could also be used to select new points. However it is inefficient when used for point selection on thousands of points such as entire image sets (since it involves numeric optimization of quadratic equations). What we want in enrolment is a simple measure that can easily calculate discriminative power for large numbers of points. What is such a simple measure which gives an indication of class discriminative power and can be used for point selection?

As with the point-to-point matching case we have found that points with small descriptors tend to give less reliable response. Therefore we first of all subsample by those points with a descriptor in the strongest *p*%. However, we take a larger proportion of points in the class matching case compared to point matching. This is because the most discriminative points may not be the strongest i.e. some weaker points may still be discriminative. Thus descriptor length is not as important an indicator in class discrimination as it is in point matching. As previously, the point descriptors are then partitioned into bins to approximate the position in descriptor space. Since we are selecting more points by length, approximate nearest neighbour methods are even more important. Also, in this case we have a class label for each point.

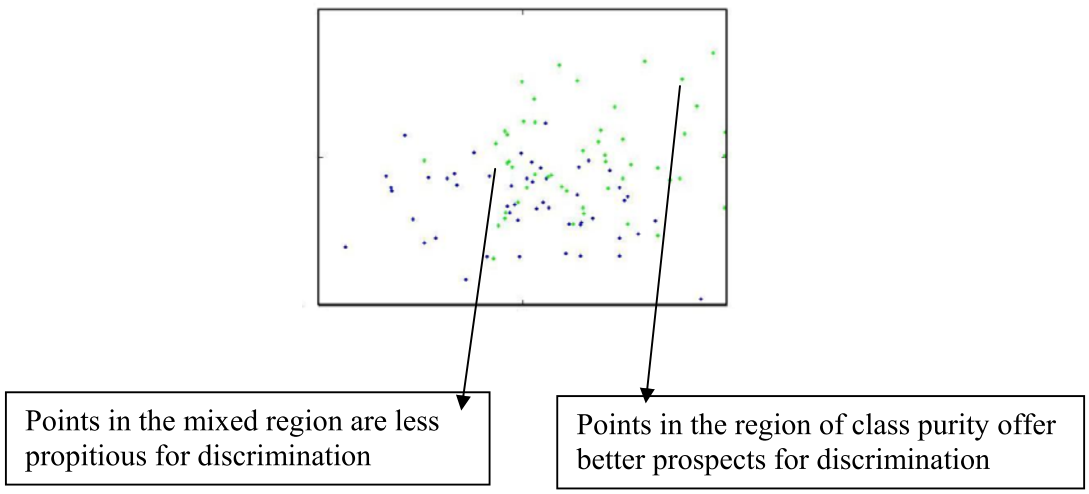


Figure 8-1: Two classes of 2-D data

**Clustering methods**

Those points that are propitious for class recognition tend to exist in class clusters within the descriptor space. Those clusters of points constitute class 'purer' regions of descriptor space. As in the point-to-point case larger scale measures of class purity (such as entire bins) do not give good results. We therefore seek to determine class purity based on local neighbors of a point. Unfortunately, formal clustering methods such as k-means are computationally expensive, may have other drawbacks, and are unsuitable for fast enrolment. If we dispense with such more exact methods, remaining simpler methods include

- The proportion of same class points within a certain distance
- The number of same class points within, say, 10 nearest neighbors
- The distance to neighbors of same and different classes

These are all quite similar measures and we choose to use the last of these.

**Distance to Class and Non-Class Neighbors**

A simple index can be obtained by looking at the distance from a point to neighbors of the same class, and non-class neighbors. This is an approximate measure of the local purity of the descriptor region as per Figure 8-2.

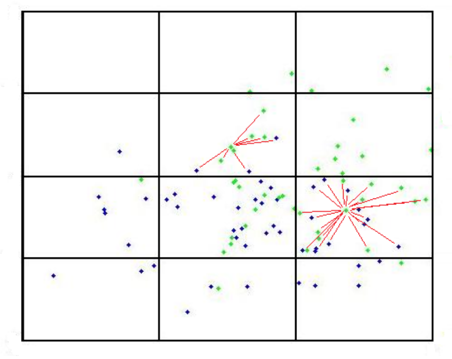


Figure 8-2: The distance to nearest neighbors indicates class purity,
Looking at the distance to nearest within class and out of class neighbors within a spatial bin, gives a simple and quick measure of the local class purity. We do this in high dimensions for interest point class recognition.

For every point (descriptor length > $p^{th}$ percentile or threshold) we calculate the ratio

$$\text{discrimination} = \frac{\text{distance to nearest neighbour same class}}{\text{distance to nearest neighbour different class}} \tag{8.1}$$

After sorting, those points with a low value are selected as being discriminative for this class. Determination of discrimination by characterizing the local neighbors is an alternative to popular methods like SVM's or clustering. Compared to standard SVM or clustering methods, this parameter is simple to calculate for all points. *This parameter is sufficiently simple to calculate that it is suitable for use with dense methods, where descriptors are calculated for all image points*. Nonetheless to perform such calculations for large numbers of points is still intensive and so approximate nearest neighbour techniques are critical to determine the distances.

Having selected the points we then enroll them in the descriptor space database for approximate nearest neighbour lookup.

**Summary of Dense Class Discrimination Method**

Point enrolment-

- Descriptors are calculated for every point.
- The longest *p*% of points are selected for further processing
- Those points with the lowest ratio of neighbor class distance to non-class distance are selected

Point recognition-

- Descriptors are calculated for every point.
- The longest *p*% of points are subject to approximate nearest neighbour matching

| **Dense Interest Point Class Method** | **Sparse Interest Point Class Method** |
|---|---|
| Descriptors at every point | Descriptors at sparse points |
| Points selected by purity of local region in descriptor space, as determined by class and non-class distance ratio | Points initially selected by image scale space properties (e.g. DoG), then subsampled |

The main advantages of this method, compared to traditional point based class recognition [26, 44] are

- Achieves higher recognition rate for some object classes
- Uses a low dimensional descriptor
- The number of points can be selected
- The density of points can be selected
- Greater positional flexibility

## 8.2 Class Discrimination System

*Class discrimination selection criteria*: Points are selected for class discrimination by length and the minimum of

$$\text{discrimination} = \frac{\text{distance to nearest neighbour same class}}{\text{distance to nearest neighbour different class}}$$

In this section we discuss in more detail the implementation and performance of a class recognition system. We have seen that the ratio of the descriptor distance of class members to non-class members gives a simple but effective indication of discrimination ability.

Figure 8-3: Examples of leaves from Caltech leaves image set

In class (e.g. Figure 8-3) and non-class images descriptors are calculated for every point. Every descriptor (descriptor length > $p^{th}$ percentile) is then enrolled into a descriptor space data structure (Figure 8-4). We set the threshold low to increase the chance of capturing discriminative points.

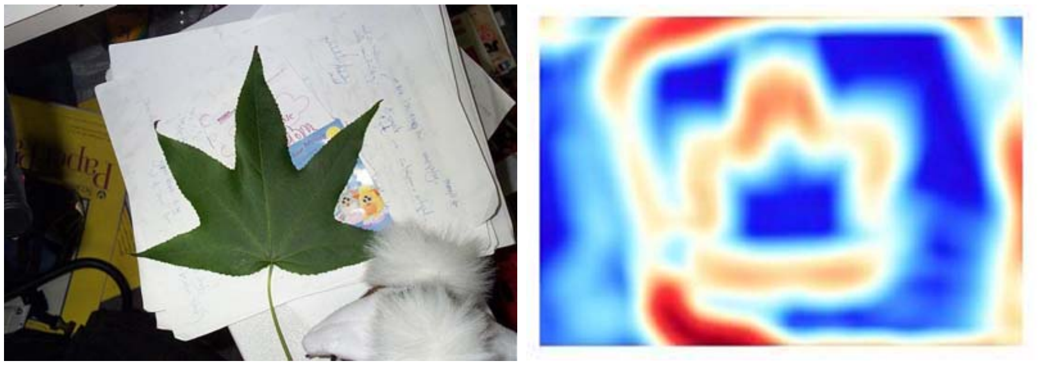
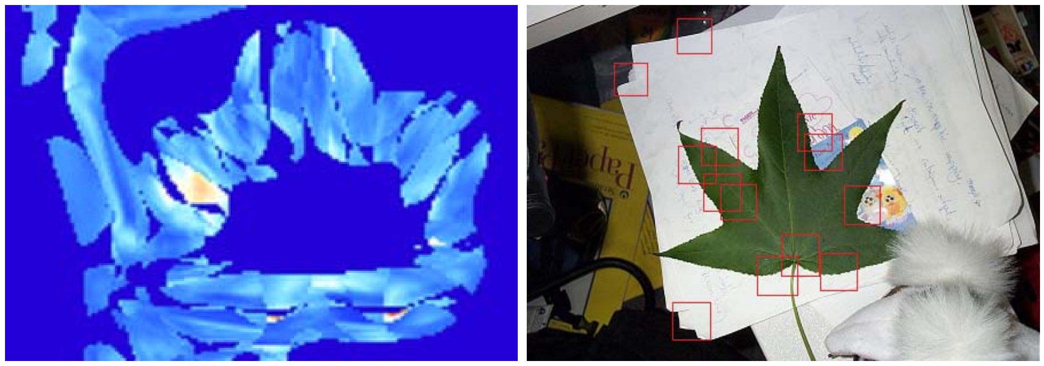

(a) (b)

Figure 8-4: Descriptor is strongest around edge regions.
(a) an example of leaf image, (b) a color map of its descriptor length.



(a) (b)

Figure 8-5: Discriminative class points
(a) The colors of this image are a measure of the discriminative power of the points. (b) Final selection of points based on descriptor length and discriminative power.

Then for every point in the descriptor space, we calculate a profile of its nearest neighbors. Specifically, we calculate the distance to its nearest neighbour of each of the same class images, and the distance to the nearest non-class points. We ensure that these distances come from points of different class and non-class images as this adds robustness; if all of the nearest same class or non-class points come from the same image it makes the selection less robust. Since this is done for every point it represents a slow part of the algorithm.

These nearest neighbour calculations are performed using the approximate nearest neighbour. We only search within the points' descriptor space bin; thus for a descriptor space divided into 256 bins we obtain a speedup of 256-fold. Finally we obtain a discriminative index, the *discrimination ratio*, based on the ratio of distances for every point as per Eq. (8.1). After sorting all points by the *discrimination ratio* we select the number of maximally discriminatory points we want. This is shown in Figure 8-5.

We are left with a set of points that are discriminative for a given class. In the leaves example it selects points in the 'sections' of the leaves, strong edge points and points where 'sections' intersect. For other image sets, such as the cougar set, it selects eyes, ears and mouth points.

Low dimensionality is more critical for class recognition than point-to-point matching for a number of reasons

- Since class discriminatory points may have weaker descriptor responses, we select more points by descriptor length. Since we are processing more points, approximate nearest neighbour methods are more important. Such methods are much more efficient at low dimension

- In point-to-point matching nearest neighbour matching is performed only in one place – in recognition on a new image that is being processed. In class recognition nearest neighbour matching is performed in 2 places - in recognition on a new image, and also in enrolling new points (where it is used in calculating the ratio of nearest neighbour points).

- Low dimensional descriptors have less specificity and therefore may be better able to generalize to classes

**SVM Testing**

One of the main practical tasks for class discrimination is image classification – given an image, a binary classification positive or negative for the presence of a given class. This is also necessary as a form of testing to determine the effectiveness of the method.

Having selected our discriminative points above, we then calculate the distance to the nearest match in the image. For *n* points we therefore obtain an *n*-dimensional vector. Positive class images should provide vectors with small values or differences, closer to the origin; negative class images should provide larger positive values (since we take a positive distance metric). Thus linear separability is reasonable, and we then apply a linear SVM to determine class. A linear SVM is simple to utilize and conforms to the testing framework of [93]. Training for this SVM is provided by the same images we used to select the discriminative points i.e. images from the Caltech 101 database. For SVM operation we have used the Matlab implementation of [5].

## 8.3 Experimental Results for Dense Class Discrimination

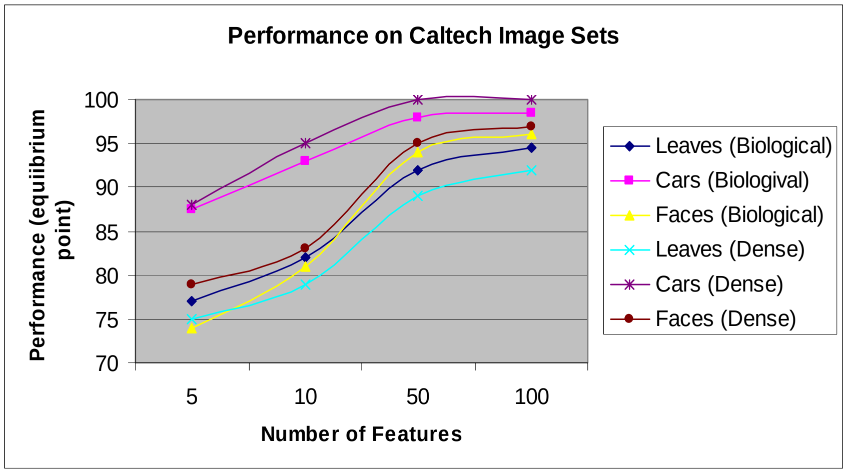
For these experiments we have used various sets from Caltech - the Caltech 101 database and some non-101 sets. Due to limited time we have only obtained experimental results for a small number of classes. Nonetheless they illustrate the class discrimination ability of this method.

We present data sequences results for

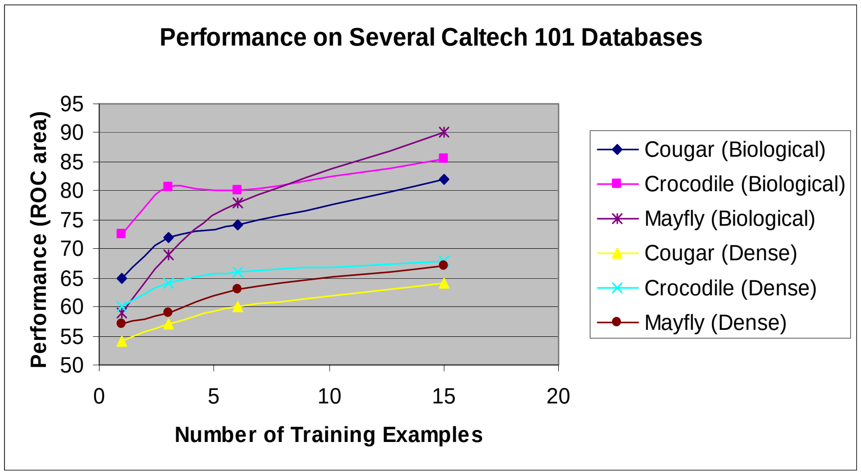
- Dense class selected Gabor descriptors
- The biological descriptor [93] is used for comparison, since it is considered one of the most successful class recognition frameworks.

Table 8.1: Parameters of the experiments

| **Experimental Parameters** | |
|---|---|
| Number of initial points selected by length | 5000 |
| Number of final points selected for stability & length | 100 |
| Dimensions | 16 (4ori×2freq×2phase) |
| Descriptor space bins | 256 |
| Descriptor space enrolment radius | 0.1 |
| Gabor wavelet λ | varied for class e.g. 21.0 |
| σ | varied for class e.g. 14.0 |
| γ | 1 |



(a) Caltech image sets



(b) Examples from Caltech 101 datasets

Figure 8-6: Class recognition performance on Caltech image sets.

## 8.3.1 Discussion of Results

For these experimental results we have obtained the distance to the nearest match for the enrolled points of each class, and then used a SVM to classify an image. We have divided these into 2 separate graphs according to the data sets. The Caltech image sets in the top graph Figure 8.6 (a) tends to encompass less variation and are less challenging. The Caltech 101 datasets in Figure 8.6 (b) are more difficult, with classes that vary considerably in shape, color, etc.

It can be seen that on the Caltech image sets the performance slightly exceeds the biological model for cars and faces, and is slightly inferior for leaves. These images have a high degree of regularity and this dense method seems to respond well to this. For example, this car database only pictures the cars from behind, and recognition is therefore high. In such situations of limited variation, the dense method performs very well.

For the Caltech 101 dataset the performance is below that of the biological method. This seems to be due to the higher variability of this data set. The biological descriptor has more flexibility (probably due to the maxima operation) than the dense descriptor we have used here; this seems to increase its recognition performance on these sets.

Overall the performance of the dense method seems to be slightly below that of the biological model. However, the biological model has been under development for ~10 years, and its more flexible descriptor could also be adopted in future for our dense matching model.

# Chapter 9 Multi-Level Descriptors

## 9.1 Introduction

In point-to-point tasks our overall aim is to maximize the point matching performance. For a given descriptor type, one general way to increase performance is to increase dimensionality. If we form a low dimensional descriptor and steadily increase dimensions (e.g. more orientations, locations, frequencies) performance will generally improve. Can we keep doing this without limit? In fact we cannot - matching performance will maximize or saturate at a limit. Beyond this limit additional dimensions will actually reduce performance. A typical profile is shown in Figure 9-1, was seen in Figure 7.8 and it is also true for well known descriptors such as GLOH [70].

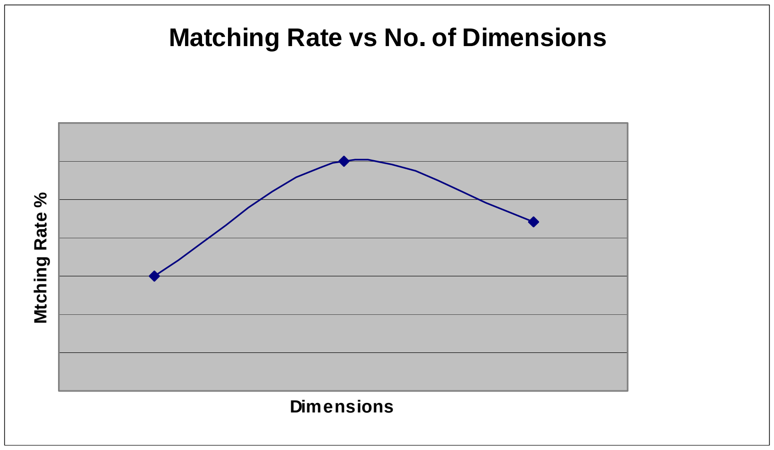


Figure 9-1: Typical profile of descriptor performance with increasing dimensions.

Thus, simple image descriptors seem to reach an inherent limit on their ability to extract information regarding a point region. If we cannot increase the performance of point descriptors by increasing dimensions in a linear manner - can we increase dimensionality and performance in another way?

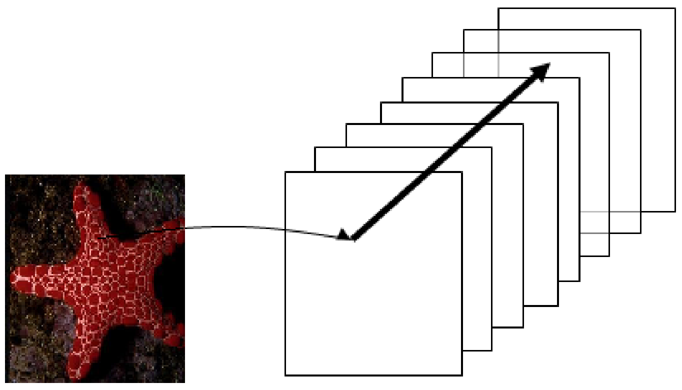
This leads us to consider a fundamental question – why do descriptors work? The application of a descriptor increases the dimensional information available at a point from 1 to the dimensionality of the descriptor. i.e. at any single pixel, the descriptive information goes from 1 scalar pixel value to an *n*-dimensional vector, typically of 16 to 128 dimensions. Such an increase in dimensionality is generally based on the neighbors of a point. We form such descriptors since they dramatically increase matching performance; point matching based simply on pixel value is effectively impossible. This is particularly so when such descriptors draw on nearby pixels and orientation information. Forming descriptors generally achieves 2 main results

- an increase in dimensionality based on nearby points
- increased emphasis to some geometric property such as orientation, or frequency

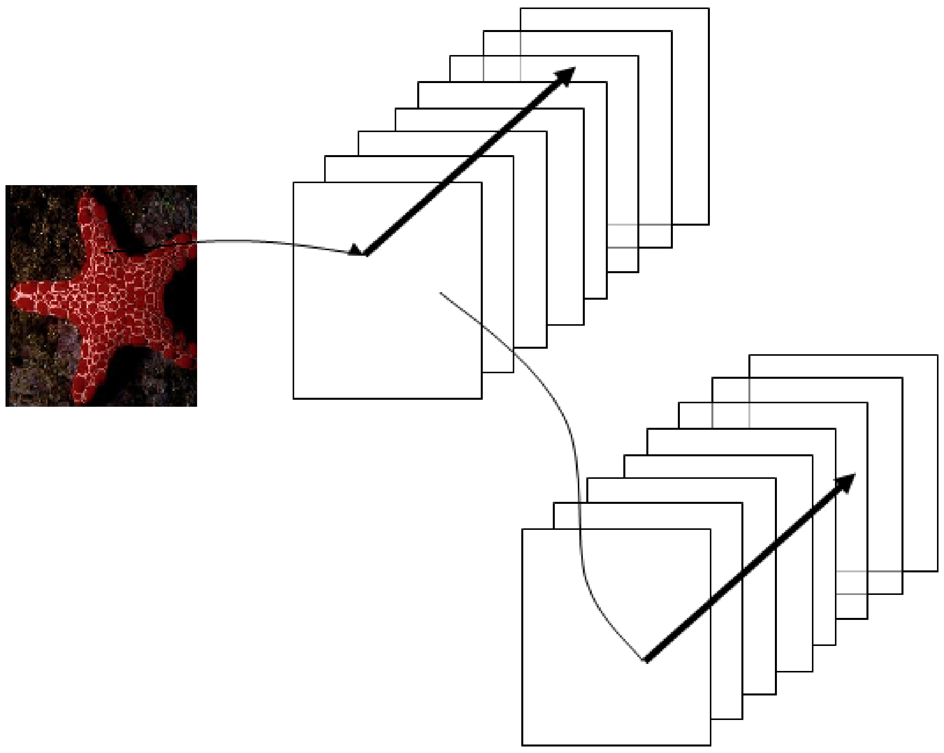
We then ask if such a process of descriptor formation could be applied repeatedly, and whether performance benefits are offered thereby.

## 9.2 Multi-Level Descriptors

We have seen that it is possible to obtain good matching results with a low-dimensional descriptor of 16 dimensions. Moreover, our dense method calculates this descriptor at every image point. Inherent in these dense descriptors is an interesting possibility – that since every descriptor dimension exists at every pixel, we can treat every descriptor dimension as a new image. Thus taking descriptors is a transform from 1 image to 16 as per Figure 9-2 (a).



(a)



(b)

Figure 9-2: Single and multi-level image descriptors

(a) Dense single level descriptors. Since descriptors exist at every point, we can treat every index as another image

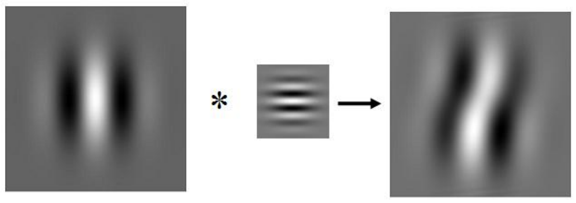
(b) We can then take descriptors again on those images

Then, regarding the descriptors as a set of new images, we can again apply descriptors to this set of descriptor images as in Figure 9-2 (b). Thus we obtain 'descriptors upon descriptors'; traditional interest point methods could be described as 'descriptors upon pixels'. In the SIFT descriptor framework such a construction is unlikely since the basic descriptor is 128 dimensions, making the final point descriptor to be of > 10,000 dimensions. This concept becomes viable because we have achieved good matching performance using few dimensions.

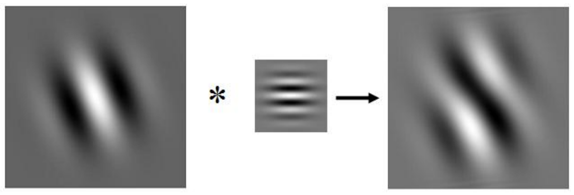
*Definition 9.1*: A *multi-level* descriptor is one in which descriptors are applied to an image in a dense manner, the results are treated as new images, and descriptors are applied again, in repetition.

## 9.2.1 Multi-Level Gabor Wavelets

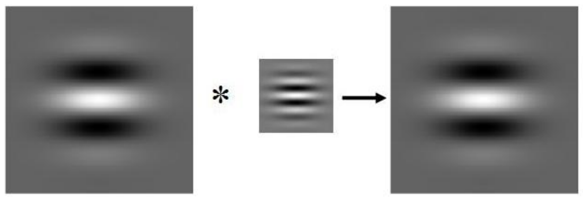
A multi-level Gabor descriptor applies a filter bank of Gabor wavelets to an image, then another Gabor filter bank to each descriptor index image. If each descriptor is of 16 dimensions, the final descriptor is of 256 dimensions. The Gabor wavelet parameters may be different at each level. It takes the properties that make the Gabor wavelet successful and apply it again. So we have 2 levels of dimensionality expansion, and 2 levels of extracting orientation and frequency information.



(a)



(b)



(c)

Figure 9-3: The 2nd level Gabor wavelets
In (a), (b) we see Gabor wavelets convolved on Gabor wavelets of different orientation and the result is a different form. However, when the orientation is the same, (c), the result is another Gabor wavelet.

**Convolving Gabor kernels**

In the 1D case:

$$f = (\cos(x) + i \times \sin(x)) \times \exp\left( -\frac{x^2}{2} \right)$$

$$g = (\cos(k-x) + i \times \sin(k-x)) \times \exp\left( -\frac{(\mathrm{k - x})^2}{2} \right)$$

$$\mathrm{L2\,Gabor}(k) = \mathrm{Real}(\int_{-\infty}^{\infty} f(x) \times f(k-x) dx)$$

$$\mathrm{L2\,Gabor}(k) = \mathrm{Real}(\int_{-\infty}^{\infty} f(x) \times g(x) dx)$$

$$\mathrm{L2\,Gabor}(k) = \int_{-\infty}^{\infty} (\cos(x)\cos(k-x) - \sin(x)\sin(k-x)) \exp\left( -\frac{(\mathrm{k - x})^2 + x^2}{2} \right) dx$$

$$\mathrm{L2\,Gabor}(k) = \cos(k) \int_{-\infty}^{\infty} \exp\left( -\frac{(\mathrm{k - x})^2 + x^2}{2} \right) dx$$

$$\mathrm{L2\,Gabor}(k) = \sqrt{\pi} \cos(k) \times \exp\left( -\frac{k^2}{4} \right)$$

(9.1)

Thus, in 1 dimension, a Gabor kernel convolved upon a Gabor kernel produces another Gabor kernel.

In the 2D case:

For 0° direction wavelets-

$$f(x,y)=(\cos(x)+i\times\sin(x))\times\exp\left(-\frac{(x^2+y^2)}{2}\right)$$

$$g(x,y)=(\cos(k1-x)+i\times\sin(k1-x))\times\exp\left(-\frac{((k1-x)^2+(k2-y)^2)}{2}\right)$$

$$\text{L2 Gabor}(k)=\text{Real}(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(x,y)\times f(k1-x,k2-y)dxdy)$$

$$\text{L2 Gabor}(k)=\text{Real}(\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(x,y)\times g(x,y)dxdy)$$

$$\text{L2 Gabor}(k)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(\cos(x)\cos(k1-x)-\sin(x)\sin(k1-x))$$

$$\times\exp\left(-\frac{((k1-x)^2+(k2-y)^2+x^2+y^2)}{2}\right)dxdy$$

$$\text{L2 Gabor}(k)=\cos(k1)\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\exp\left(-\frac{((k1-x)^2+(k2-y)^2+x^2+y^2)}{2}\right)dxdy$$

$$\text{L2 Gabor}(k)=\pi\cos(k1)\times\exp\left(-\frac{k1^2+k2^2}{4}\right)$$

(9.2)

Here L2 Gabor is the 2nd Level Gabor wavelet.

Thus in the 2D case where the 2 wavelets are in the same direction, the result is another Gabor wavelet. This calculation was for the 0° direction, but is valid for any direction by symmetry (i.e. a change of variable). See Figure 9.3 (c).

Where the wavelets are in different directions we must treat it separately. For 0° and 45° direction wavelets-

$$f(x,y) = (\cos(x) + i \times \sin(x)) \times \exp\left( -\frac{(x^2 + y^2)}{2} \right)$$

$$g(x,y) = (\cos(k1 - (x+y)) + i \times \sin(k1 - (x+y))) \times \exp\left( -\frac{((k1-x)^2 + (k2-y)^2)}{2} \right)$$

$$\text{L2 Gabor}(k) = \text{Real}\left( \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y) \times g(x,y) dxdy \right)$$

$$\text{L2 Gabor}(k) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (\cos(x)\cos(k1 - x - y) - \sin(x)\sin(k1 - x - y))$$
$$\times \exp\left( -\frac{((k1-x)^2 + (k2-y)^2 + x^2 + y^2)}{2} \right) dxdy$$

(9.3)

In this case the final integral cannot be evaluated or simplified in a standard form. Thus in the case that the wavelets are in different directions, the convolution result does not correspond to any standard analytical functions. See Figure 9.3 (a), (b).

We have ignored constant factors throughout.

**Associativity and Separability**

In applying descriptors to descriptors we are performing convolution successively. This allows us to use the associative rule for convolution

$$f * (g * h) = (f * g) * h \qquad (9.4)$$

This means that convolution needs only to be applied once, but the full set of 2nd level kernels must be kept in memory. This is an acceleration technique, but at a cost of increased memory usage. Figure 9.3 shows typical examples of such kernels formed from convolution of Gabor wavelets.

It can be seen from Eq. (9.3) that in general, separability is not preserved for 2nd level Gabor kernels. Thus we cannot accelerate the numerical implementation of the convolution in this case as usual.

It can be seen that a Gabor wavelet convolved upon a Gabor wavelet either produces another Gabor wavelet, or something like another Gabor wavelet, except with a 'twist' applied.

## 9.2.2 Other Multi-Level Descriptors

The concept of 'descriptors on descriptors', or multi-level descriptors is not restricted to Gabor wavelets and in principle can be applied to any other descriptor. Practically, there is a requirement that the descriptors be low dimensional.

### Histogram Descriptors

A potentially useful multi-level descriptor is a 'histogram of histograms'. This is because histograms are very efficient and can be used to represent a wide range of image information.

A multi-level histogram is different from a multi-dimensional histogram. A multi-dimensional histogram can be used to characterize an image by, say, 2 different quantities e.g. gradient and Laplacian [87]. However a multi-level histogram still uses 1 image quantity.

For example, suppose we histogram a region by intensity $I$, using 4 bins. Each bin spans an intensity range which together covers the possible values (0-255). For each pixel, the bin is incremented by 1 within whose range that pixel lies.

For each region we then have a 4-dimensional vector. If we then histogram these 4-dimensional vectors, e.g. by dividing each dimension into 2 parts around the median, we get a final vector of $2^4$ or 16 dimensions. Each of the 4-dimensional histogram vectors constitutes 1 point in the final histogram (see Figure 9-4).

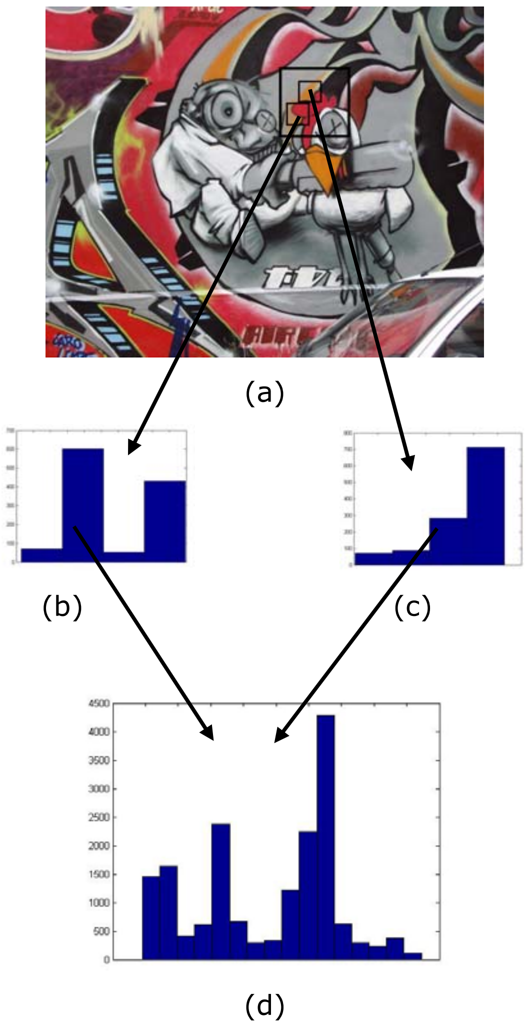


Figure 9-4: The 2nd Level Histograms.
From the image (a), histograms of color or intensity of a small area are produced (b), (c). These histograms are then combined to form another histogram by dividing each bin into 2 around the median.

We make use of multi-level histograms in an application example in Chapter 10. This shows the applicability of these for object class recognition.

## Correlation Descriptors

A Gaussian convolution descriptor characterizes a point by blurring and sub-sampling the region around it. Effectively it performs a correlation match between regions.

Likewise we can perform Gaussian convolution upon Gaussian convolution in a similar way to multi-level Gabor wavelet convolution. i.e. each dimension could be treated as another image and subject to another convolution. Indeed a Gaussian kernel could be considered a special case of a Gabor wavelet (apart from mean normalization).

An interesting possibility is to combine different types of descriptors at different levels. For example, we could have a correlation descriptor as the first descriptor, then a Gabor wavelet as the 2nd level descriptor.

We will not be pursuing such descriptors any further in this thesis.

## Higher Level Descriptors

Taking 'descriptors on descriptors' could be carried on to the 3rd level, or an arbitrary level. However, it is likely that the increased dimensionality will eventually dilute the discriminative power of the descriptor. Also, Gabor wavelet descriptors tend to naturally respond to edges or the absence of edges. Going to higher levels produces descriptors that naturally respond to edges of edges, etc. Such structures are likely to contain gradually less information. Again, distinctiveness will likely gradually reduce with arbitrarily more levels. Nonetheless we intend to investigate this in future.

## 9.2.3 Point Selection Criteria

Point selection is done using the same basic criteria as used in Sec. 7.3. That is, we use points that are maximally stable, using the stability criteria of

1) descriptor length > *p*th percentile or threshold

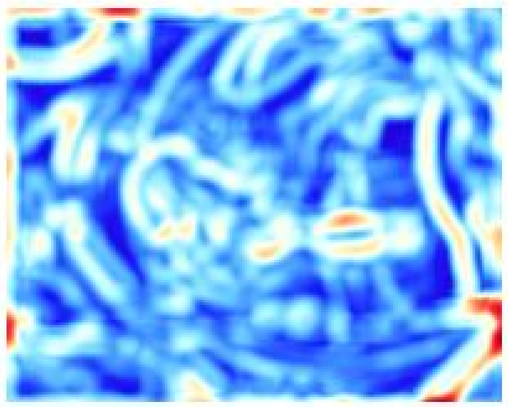
2)
$$\Delta\text{descriptor} = \sqrt{\left(\sum(\mathrm{d}_i(\text{image}(x,y)) - \mathrm{d}_i(\text{transformed image}(X,Y))^2\right)}$$
$$\mathrm{d}_i \text{ is the } i^{\text{th}} \text{ element of the descriptor vector d}$$
$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{9.5}$$

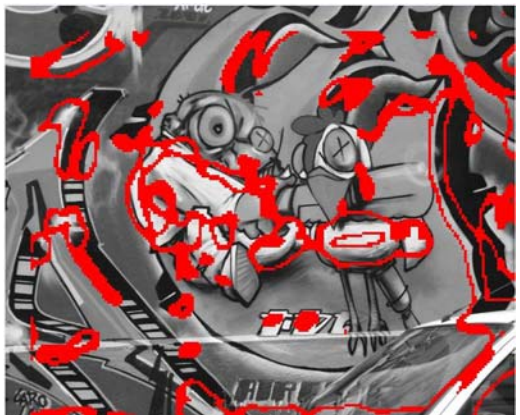
under a small synthetic transform (*H* is the homography between the image and its transform). As previously the synthetic transform we apply is a small rotation and a blurring.

We take the minimum k points by 2) as required. The steps are shown in Figure 9-5.

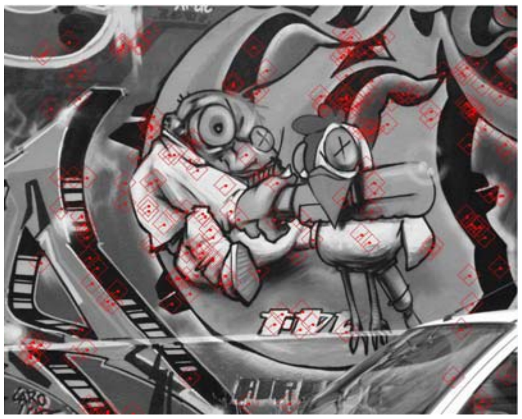
A fundamental difference between this and the descriptor of Sec. 7.3 is that that one was 16 dimensions while this one is of 16×16 = 256 dimensions.

(a)


(b)



(c)



(d)

Figure 9-5: Multi-level descriptor points
(a) the image and (b) the color map of descriptor length. (c) The strongest points are determined. (d) The final points are selected from the strong points based on stability under an applied transform.

## 9.3 Experimental Results for Multi-level Gabor Wavelet Matching

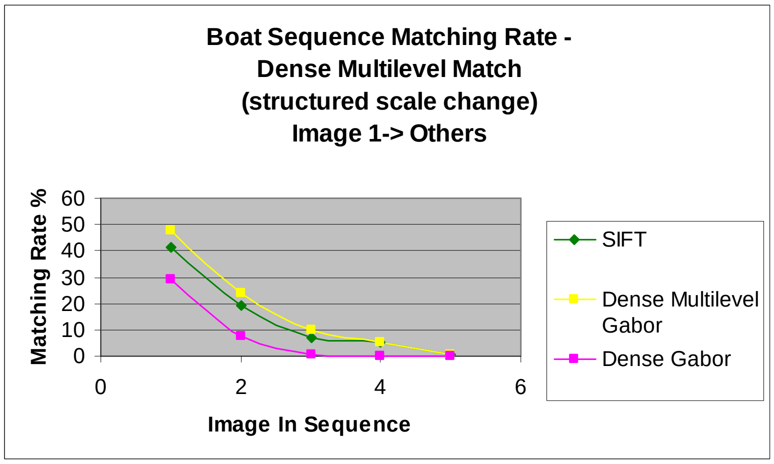
In this section we present experimental results for multilevel Gabor wavelets, using the same Oxford image sets as previously. Again, we use SIFT for reference performance.

We present data sequences results for

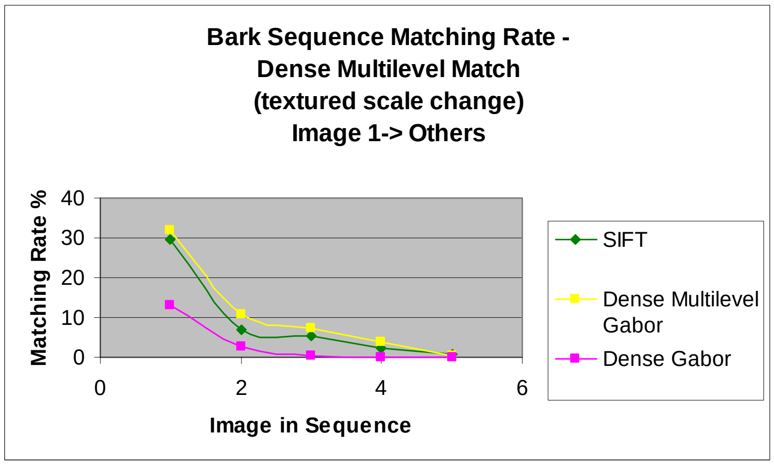
- Multilevel dense Gabor descriptors, 256 dimensions – 'Dense Multilevel Gabor'
- Single level dense Gabor descriptors, 16 dimensions – 'Dense Gabor'
- SIFT descriptors. This is another high-dimension descriptor for comparison – 'SIFT'

Table 9.1: Parameters of the experiments

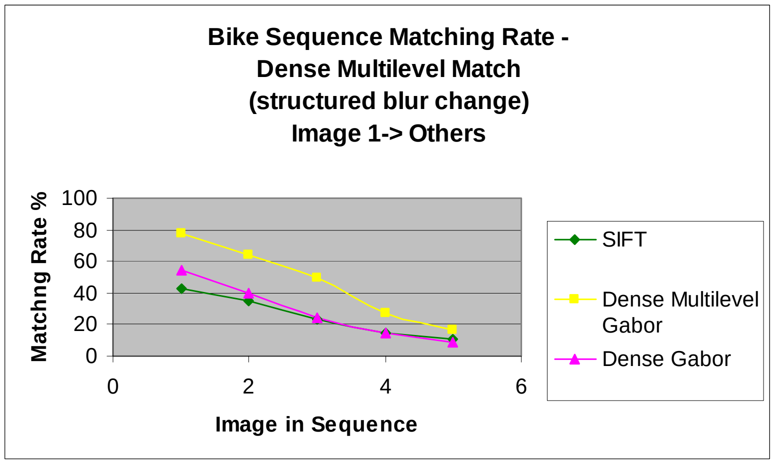
| **Experimental Parameters** | |
|---|---|
| Number of initial points selected by length | 2000 |
| Number of final points selected for stability & length | 1000 |
| Dimensions | 16 (4ori×2freq×2phase) × 16 (4ori×2freq×2phase) |
| Descriptor space bins | 256 |
| Descriptor space enrolment radius | 0.05 |
| 1st level Gabor wavelet λ | 28.0 |
| σ | 7.0 |
| γ | 1 |
| 2nd level Gabor wavelet λ | 20.0 |
| σ | 4.0 |
| γ | 1 |
| Timing for interest point and descriptor extraction | 15.87s |



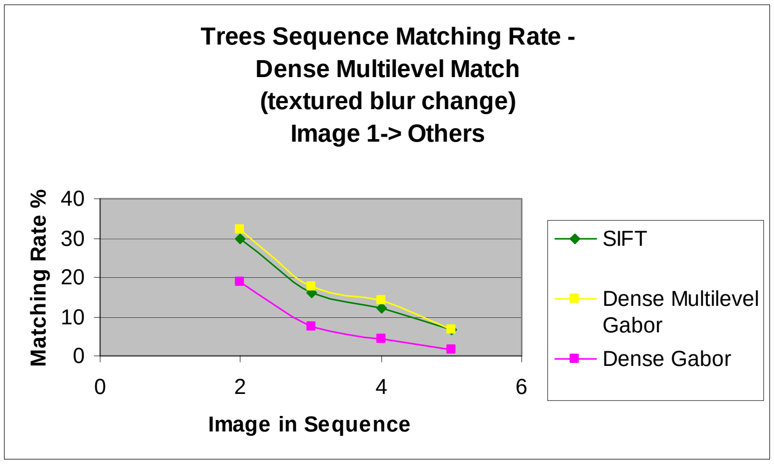
(a) Structured scale change



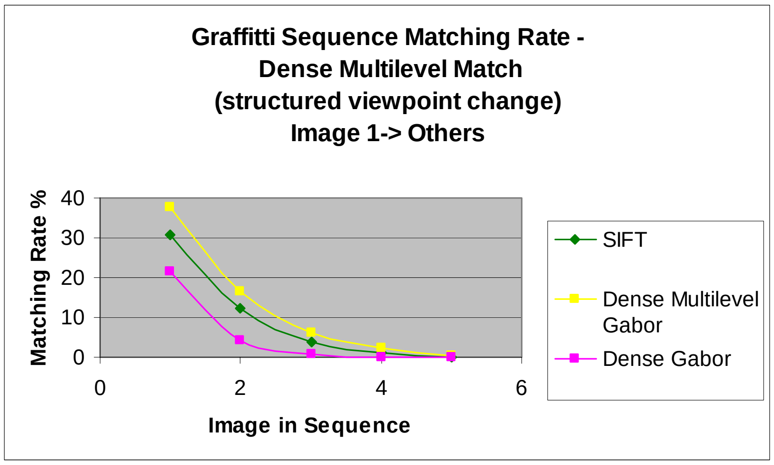
(b) Textured scale change



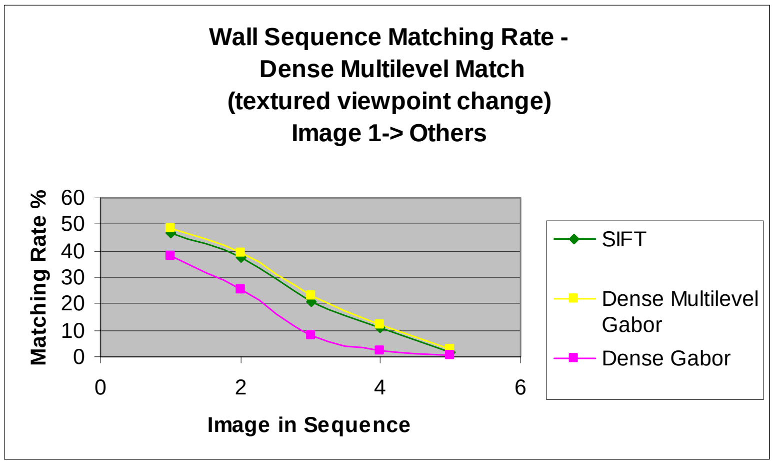
(c) Structured blur change



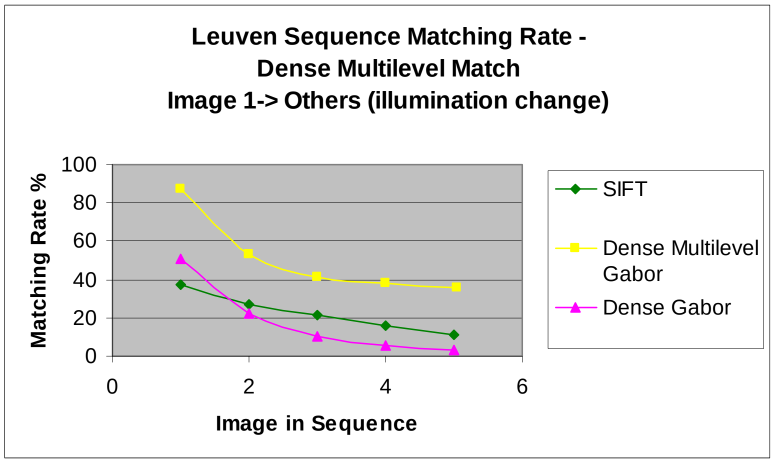
(d) Textured blur change



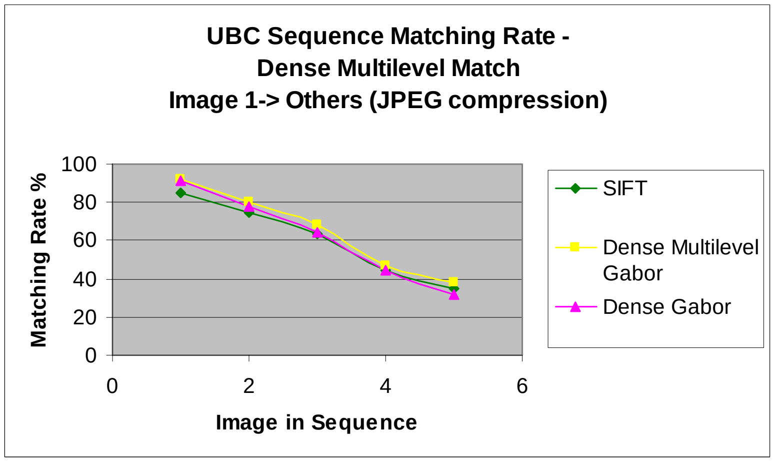
(e) Structured viewpoint change



(f) Textured viewpoint change



(g) Illumination change



(h) JPEG compression change

Figure 9-6: Dense single level, multi-level and SIFT matching rates for various image variations

Table 9.2: Summary Statistics

| | Textured | Structured | Viewpoint | Scale | Blur | All |
|---|---|---|---|---|---|---|
| | | | | | | |
| SIFT | 16.22 | 16.55 | 16.58 | 11.82 | 21.26 | 22.83 |
| Dense Multilevel Gabor | 16.97 | 25.69 | 18.80 | 14.11 | 32.64 | 30.85 |
| Dense Gabor | 8.77 | 13.75 | 10.06 | 5.37 | 19.43 | 18.75 |

## 9.3.1 Discussion of Results

It can be seen Figures 9.6 (a) – (h) that applying a 2nd level of descriptor makes substantial improvements in almost all cases. This is true for scale, viewpoint, blur, and illumination change. The improvement is for both structured and textures sequences. Only JPEG compression (h) didn't show any real improvement (which is not unexpected since it is not a geometrically smooth transform). In many cases the improvement is an almost doubling of matching performance from the 1st level.

In all cases the 2nd level Gabor descriptor also outperforms SIFT, albeit by a modest margin in most cases. SIFT has only rarely been outperformed by other interest point methods [70]. These results suggest that the concept of taking descriptors to a higher level has considerable potential. It is possible that these results apply generally to other descriptors, and also apply to class recognition.

It may be that 3rd or 4th level descriptors will also offer good performance. It is interesting to note that the performance of the 2nd level Gabor descriptor is somewhat similar to SIFT. This may be because it is possible to regard SIFT itself as an approximation to a 2nd level Gabor descriptor – the gradient operator is the first level approximation, the Gaussian smoothing is the 2nd level.

# Part IV  Applications & Conclusions

# Chapter 10 Applications

## 10.1 Introduction

In this section we demonstrate some practical applications of the ideas in this thesis. Our purpose here is just to give some indication of the uses that these ideas could be put to. These applications are far from complete or comprehensive in illustrating the use of these concepts.

Hereafter we demonstrate 2 simple applications, navigation and class recognition. We do not go deeply into either one here – either of these could form a large body of research in itself.

## 10.2 Navigation and Structure from Motion

Navigation is potentially one of the most useful tasks in computer vision. It is likely to be a fundamental function of any future robotic vision systems, and goes hand-in-hand with 3D reconstruction techniques.

More specifically, navigation is also a major application of interest point methods. In [42, 91] interest point matching is used as one of the key steps in autonomous navigation. Here, false point matching results in errors in navigation position and 3D point reconstruction.

Here we define navigation as the determination of the 3-dimensional position of the camera at every given frame. In that sense we are defining navigation as camera external calibration. This calibration determines a sequence of decompositions of projections matrices-

$$P_i = \mathbf{A}[\mathbf{R_i} \quad \mathbf{t_i}]$$
$$P_i = \begin{bmatrix} \alpha_u & \gamma & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R_i} & \mathbf{t_i} \\ \mathbf{0} & 1 \end{bmatrix} \tag{10.1}$$

Where **A** is the internal parameters, of which $\alpha_u$, $\alpha_v$ are the focal length, $\gamma$ is the skew, and $u_0$, $v_0$ are the coordinates of the principal point. $\mathbf{R_i}$ is the rotation matrix and $\mathbf{t_i}$ is the translation of the camera centre in Euclidean space for camera i. See [28, 43] for details of these. An illustration of this is given in Figure 10-1.

A typical camera calibration system has several main aspects

- Interest point extraction and matching
- Robust methods such as RANSAC, to remove false matches
- Calibration algorithm, such as 5-point algorithm [78]
- Numerical method, such as the sparse Levenberg-Marquardt [43]

It is not the purpose of this work to examine any of these aspects in detail except the first one. We will therefore use a publicly available implementation, Bundler [3], for all other steps. Bundler is the engine underlying the Photosynth project from Microsoft (http://photosynth.net/).

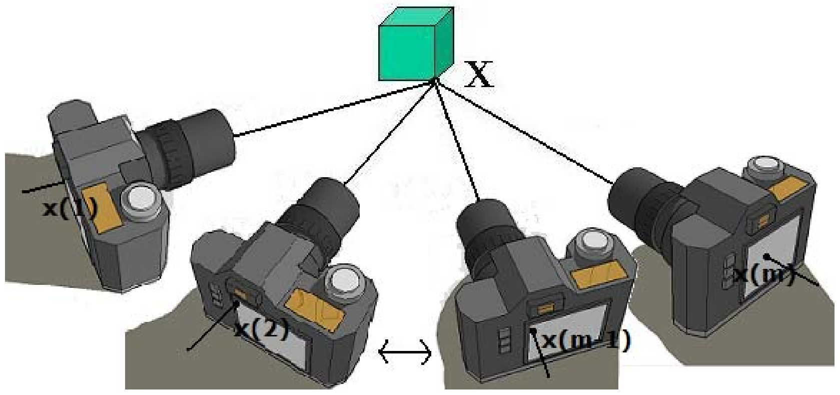
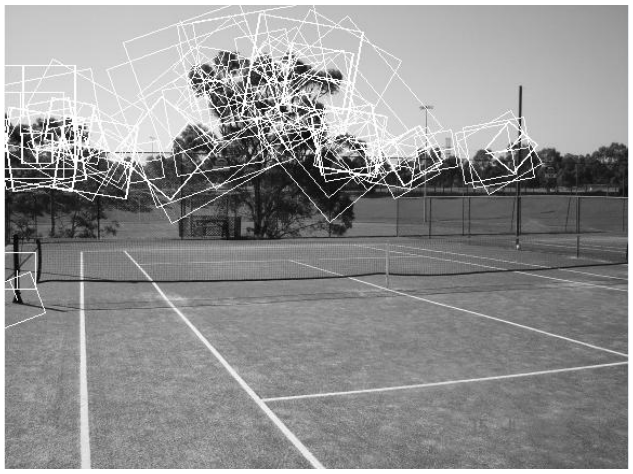


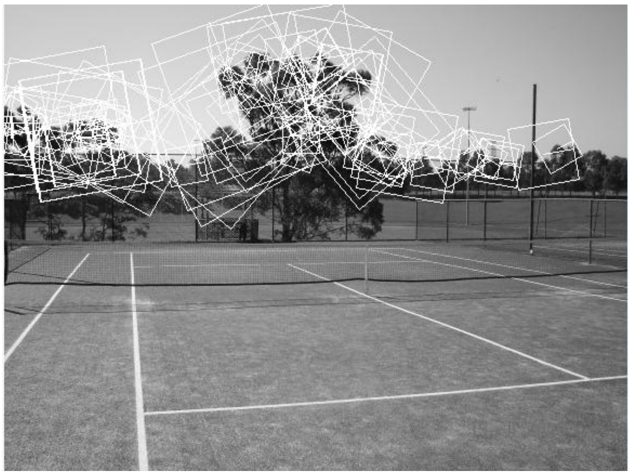
Figure 10-1: Structure from motion.
Determining the 3D structure of a scene and camera position from a sequence of images and corresponding points

## 10.2.1 Setup

A series of photographs (Figure 10-2) were taken from measured positions for which we know the ground truth. This ground truth was obtained by taking the photographs from known 3D positions. To this end we used points on a tennis court. Images were taken from the 10 positions marked in Figure 10-3.

(a)


(b)



(c)

Figure 10-2: Tennis court images used with Bundler.
(a) the original image from a tennis court corner. (b) the same image with interest points extracted. (c) the adjacent corner image with interest points extracted

The Bundler program extracts interest points and matches them.
Part of Bundler was adapted to allow geometric point matching. We used SIFT matching, and SIFT with geometric matching. After this matching step, Bundler applies the 5-point algorithm and does bundle adjustment using the sparse Levenberg-Marquardt method. It finally outputs the 3D interest point positions and camera positions for each photograph.

Although Bundler calculates the 3D positions of the camera it does so with a coordinate frame that may be rotated, scaled and translated compared to the true one that we want. We therefore must apply a transformation to each point

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = a \begin{bmatrix} \cos(\vartheta) & \sin(\vartheta) & 0 \\ -\sin(\vartheta) & \cos(\vartheta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\phi) & 0 & -\sin(\phi) \\ 0 & 1 & 0 \\ \sin(\phi) & 0 & \cos(\phi) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} Tx \\ Ty \\ Tz \end{bmatrix}$$

(10.2)

This transform determines the ambiguous parameters in the Euclidean reconstruction. These 7 parameters are the 3 Euler angles, 3 translation components and a scale factor. To determine the values of the 7 parameters we minimize the distance to the ground truth position again using Levenberg-Marquardt optimization, this time in Matlab. This is similar to least-squares plane fitting through the points, except that the plane fitted also has a normalized canonical coordinate frame.

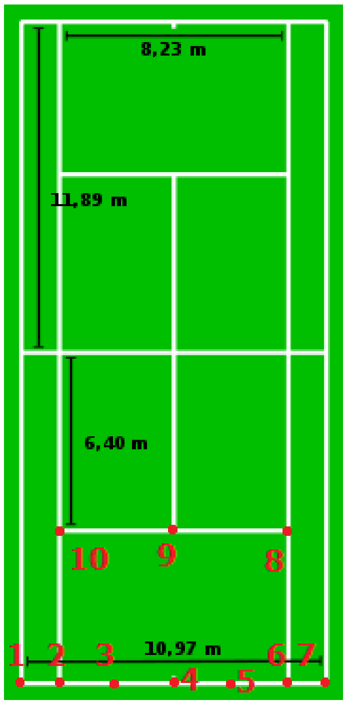
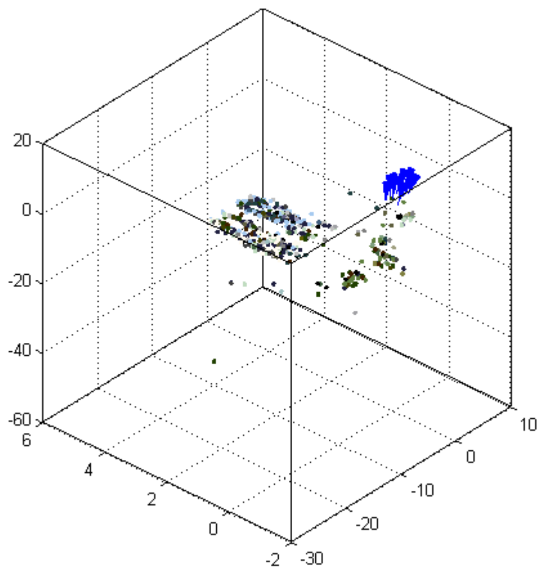
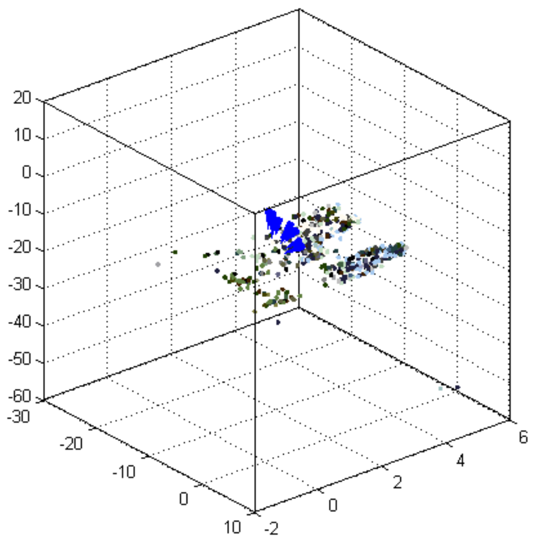


Figure 10-3: Tennis court and positions from which images were taken. Tennis court with dimensions and the points marked from which the images were taken (tennis court image from Wikipedia)

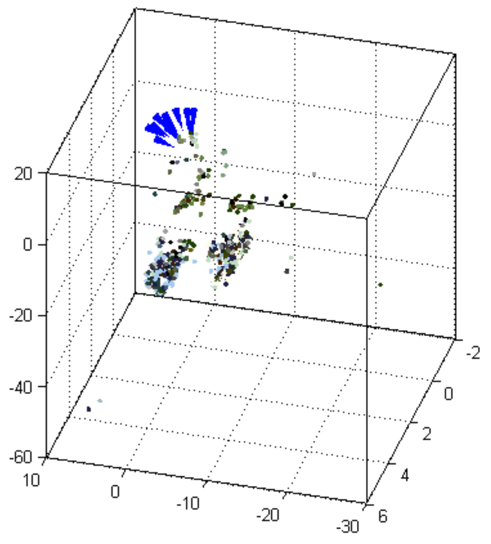
### 10.2.2 Results



(a)



(b)



(c)

Figure 10-4: Bundler reconstruction results with points and cameras from 3 different viewing positions.

Results of the raw Bundler reconstruction from 3 different viewpoints are shown in Figure 10-4. Bundler determines the 3D position of cameras (blue cones) and scene points (in scene color). Most of the points are on background trees. The scale and coordinates of the raw reconstruction are arbitrary; they must be converted to real world scale and coordinates by Eq. (10.2).

Table 10.1: Bundler point position reconstruction

| Point | Point Ground Truth Position (m) | Point SIFT Position | Error | Point Geom SIFT Position (m) | Error |
|---|---|---|---|---|---|
| 1 | (0,0,0) | (5.94, 0.20, -0.95) | 6.02 | (0.46, 0. 01, -0. 25) | 0.52 |
| 2 | (1.37,0,0) | (4.55, 0.19, -0.86) | 4.64 | (1. 37, 0.00, -0.00) | 0.00 |
| 3 | (3.43,0,0) | (2.41, 0.17, -0.73) | 2.53 | (2.94, -0.02, 0.54) | 0.73 |
| 4 | (5.49,0,0) | (-5.05, -0.74, -2.66) | 5.75 | (9.11, -0.20, -1.14) | 3.80 |
| 5 | (7.54,0,0) | (-1.29, 0.23, -0.76) | 1.52 | (3.39, 0.20, -4.43) | 6.06 |
| 6 | (9.6,0,0) | (-3.85, 0.17, -0.68) | 3.92 | (4.35, -0.04, 0.80) | 5.30 |
| 7 | (10.97,0,0) | (-8.02, -0.27, -1.93) | 8.25 | (8.05, -0.12, -1.20) | 3.15 |
| 8 | (9.6,0, -5.49) | (-4.27, 0.15, 4.37) | 6.11 | (9.59, -0.00, -5.49) | 0.00 |
| 9 | (5.49,0, -5.49) | (0.18, 0.18, 4.77) | 4.77 | (5.95, -0.07, 1.24) | 6.74 |
| 10 | (1.37,0, -5.49) | (1.57, -0.27, 3.55) | 3.90 | (5.98, 0.16, -3.20) | 5.15 |
| **Total Error (m)** | | | **47.41** | | **31.45** |

Thus the usage of geometric matching has delivered a more accurate reconstruction of the camera positions. It has reduced the total error in the positional measurements by ~30%. It is interesting to note that although Bundler uses leading algorithms, the error is still quite large.

## 10.3 Object Class Recognition – ASIRRA CAPTCHA

In this section we describe an object class recognition application. Class recognition is another major computer vision area which is still a vigorous area of ongoing research. The particular application here is the ASIRRA CAPTCHA [4] (Completely Automated Public Turing test to tell Computers and Humans Apart) http://research.microsoft.com/en-us/um/redmond/projects/asirra/
CAPTCHAs are uses by websites to distinguish between humans and automatic bots. This CAPTCHA requires the user to distinguish between pictures of dogs and cats. Although easy for humans, this is an extremely challenging task for computers. This work is not intended as a definitive attack on this CAPTCHA; rather is indicative of the type applications that are possible with the descriptors of this thesis. Example images from ASIRRA are shown in Figure 10-5.

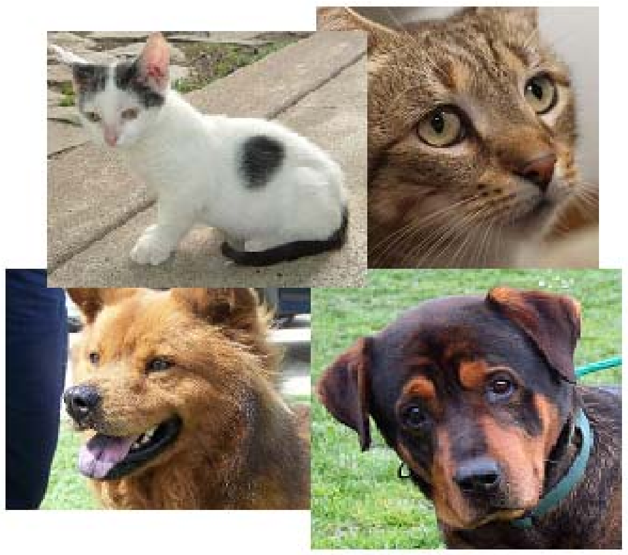


Figure 10-5: ASIRRA dog and cat images

When it was announced, it was claimed that the best machine learning techniques would be unlikely to achieve a success of > 60% per image. The

CAPTCHA requires that the user select all the cats in a 12 image series, giving an expected correct response of 0.2% (at 60% per image). A published success rate, as given in [27] is 56.7%; the best reported to date is 82% in [39]. CAPTCHA methods are also discussed in [75, 110].

Herein, we will use the method of Sec. 9.2.2 to attempt to solve this problem. Histograms are plausible as a primitive for problems of this type since can deal with wide variations in geometry. Following [39], we will use color as the discriminating quantity. Our objective here is not to achieve the maximum headline matching rate, per se, since we are limited by time. Rather, our purpose is to demonstrate that discriminative histograms can achieve credible matching rates, and that this is improved by using multi-level histograms.

A large number of binary features are used in [39], and accepts weak discrimination power from them individually. We adopt a somewhat different approach in that we use smaller numbers of higher dimensional features. It is claimed that histograms give poorer performance than binary features, therefore we modify our histograms somewhat. Rather than histogram the number of pixels in a color range, we will histogram

$$\begin{gathered}(number\_of\_pixels\_in\_color\_range)^{k} \\ 0 < k < 1\end{gathered} \tag{10.3}$$

For values of $k$<1 this parameter increases robustness to image variation (particularly scale).

From Figure 10.5 it can be seen that the images present dogs and cats at a range of sizes or scales.

## 10.3.1 Setup

All images are subject to pre-processing to scale them to 250*250 pixels. The descriptor we are using is a histogram of histograms. This allows greater positional flexibility than a histogram, but does not abandon position information entirely.

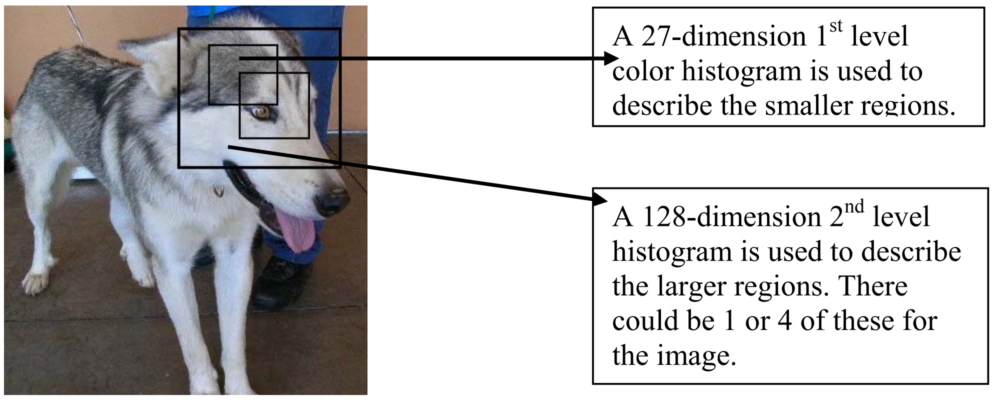

A 27-dimension 1st level color histogram is used to describe the smaller regions.
A 128-dimension 2nd level histogram is used to describe the larger regions. There could be 1 or 4 of these for the image.

Figure 10-6: Multi-level color histogram structure.

**1st level Descriptor**: The first level descriptor operates directly on image pixel color. For each of the 3 channels R, G, B we divide into 3 bins. The histogram is of the combination of the R, G, B values. Thus for 3 bins for each channel, there are a total of 27 histogram bins. Each pixel is allocated to one of these bins. The histogram is made over a smaller area e.g. 50*50 pixels.

**The 2nd level Descriptor**: The 2nd level histogram treats each of the 1st level descriptors as a single point. That is, each of the 27-dimensional vectors from the 1st level is a single entry in the 2nd level histogram. This is achieved by grouping the 27-dimensional vector into 6 groups of 4 components and 1 group of 3 components. Thus 7 groupings are attained and in each group the values are added together. Each of the 7 values is binary divided into 2 ranges around the median. Thus a 2nd level descriptor of 128-dimensions is obtained. Each level 1 histogram adds 1 value to one of the 128 histogram dimensions.

These are illustrated in Figures 10-6 and 10-7.

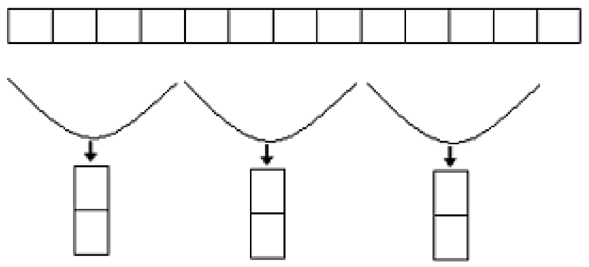


Figure 10-7: Relationship between 1st and 2nd level descriptor formation. Every 4 components of the 1st level descriptor are grouped and added, then allocated a binary value. There are 7 such groups resulting in a final descriptor of $2^7$ = 128 dimensions. We do it in this way to get the right dimensional balance.

A large number of training images are used to provide feature points to train an SVM. For the final discriminant function we use an exponential kernel function

$$K(u,v) = \exp\left(-\gamma |u-v|^2\right) \qquad (10.4)$$

## 10.3.2 Results

Table 10.2: Parameters for ASIRRA testing

| **Experimental Parameters** | |
|---|---|
| Size of 1st level histogram window | 35 pixels |
| Size of 2nd level histogram window | 100 pixels |
| Dimensions of level 1 descriptor | 27 |
| Dimensions of level 2 descriptor | 128 |
| $\gamma$ | 0.01 |
| $k$ | 0.25 |
| No. of testing images in 1 run | 16 |

Table 10.3: ASIRRA classification rate performance by dimensions

| Correct Classification Rate | 1 Level Descriptor | | 2 Level Descriptor | |
|---|---|---|---|---|
| | mean | stdev | mean | stdev |
| 128 dimensions | 53.1 | 2.0 | 58.2 | 2.3 |
| 512 dimensions | 54.6 | 2.1 | 60.9 | 2.3 |
| | | | | |

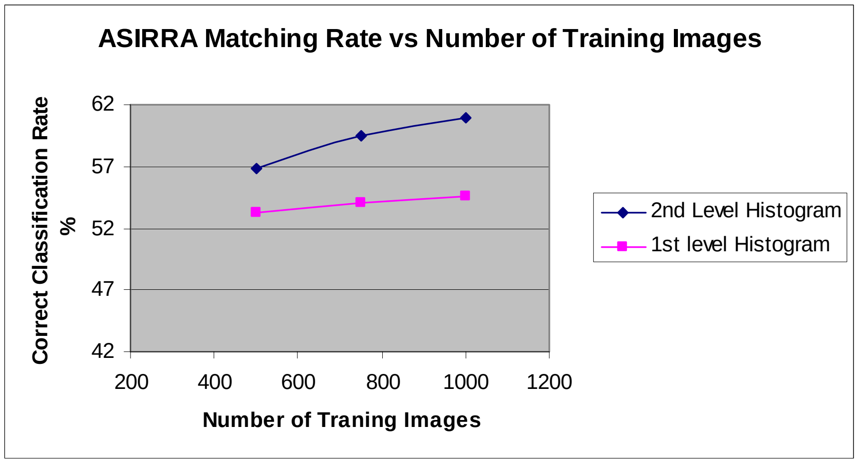


Figure 10-8: ASIRRA classification rates by training images

It can be seen in Figure 10-8 that the matching rates with a 2-level histogram descriptor deliver an improvement compared to a standard 1-level histogram. The correct matching rate exceeds that of [27], but is inferior to [39]. We believe this is due partly to the very large number of features and training images used in [39]; and also to the use of other information, such as using texture descriptors.

We believe the approach of using 2-level histograms has the potential to achieve considerable further improvement in performance, and we look forward to pursuing this in future.

# Chapter 11 Discussion and Conclusions

In this chapter we review the main results of this work and draw some conclusions from them. We also look at some future development of ideas that are motivated by this work.

## 11.1 What We Have Tried to Do

The main objective of this thesis has been to improve interest point operation. After quick advances in the early 2000's, slower progress has since been recorded. Herein we have looked at alternate metrics, point selection and point descriptors to achieve better performance

## 11.2 What has been achieved in this thesis

A range of ideas have been implemented in previous chapters. An experimental regime has been used to evaluate the effectiveness of these ideas. Some of these have been seen to be more significant and offer more improvement than others.

The ideas of geometric matching provided an improvement to matching performance for correlation and SIFT descriptors, with a much smaller improvement for low-dimensional Gabor descriptors. This performance improvement was similar to affine invariant methods.

Dense descriptors were shown to be capable of point matching and class recognition. For dense point matching, performance competitive or better than the best low dimensional descriptors was achieved.

The concept of multi-level descriptors was introduced and shown to provide performance improvements compared to single-level descriptors.

**Main Overall New Results**

- This thesis has presented 3 different, new methods by which SIFT matching performance can be exceeded. Performance exceeding SIFT has only rarely been achieved by other interest point methods.
- The concept of geometric match has been shown to deliver matching improvements compared to the Euclidean distance. For the correlation and SIFT descriptors these are marked. The geometric match brings stereo matching methods into the realm of interest point matching.
- That dense point matching has been shown to deliver results superior to traditional sparse methods, and uses few dimensions. High performance point matching has previously only been demonstrated with sparse point detectors. The results obtained outperform other low-dimensional methods.
- High performance for recognition of some object classes using dense detectors. Overall the class recognition results are somewhat below the 'Biological' model, however on some specific classes superior performance was achieved.
- The concept of the multi-level descriptor is new. It has been shown to deliver clear matching improvements compared to standard single level descriptors. The multi-level Gabor descriptor outperforms the SIFT descriptor.

## 11.3 Limitations and Future Work

All the methods in this thesis have limitations and areas where future development could be made.

Geometric Matching:
The geometric method depends on a stereo-like smoothing method. This is an area of intense research in itself. Herein we have just used a simple Potts model derivative (see [16]). More sophisticated stereo methods such as belief propagation could improve results. The results reported herein for geometric matching represent a tradeoff with processing time. Faster CPU's would increase matching performance by allowing higher dimensional descriptor matching (this is unlike SIFT, for example, where a performance plateau has been reached).

Dense Matching:
In this study we have examined dense matching only for Gabor descriptors. A more comprehensive study would look at other descriptors also.

For point matching we have selected points by stability, using the change under a small rotation and blur. Better performance would be achieved by a more general transform e.g. a full affine transform and blur. This is similar to the idea of Hinterstoisser [45] but we are using it to select points, not just describe them.

For class discrimination we have used a fixed Gabor descriptor. There would probably be benefit in adopting a more flexible Gabor descriptor, such as that of [93]. This involves taking of maxima over a window, which increases flexibility. We believe this is the reason the 'biological' outperformed our descriptor for some difficult classes. Also, our testing for class discrimination involves only a small number of classes. A larger number would provide more certain results.

Multi-Level Descriptor:
The multi-level descriptor in this thesis is slow to compute since it involves 2 levels of 16 convolutions each. For it to be practical it will need to be accelerated. We believe methods can exist to achieve this, and they may be looked at in future.

The multi-level framework is complex and leaves many questions remaining. In this thesis we have only examined descriptors to the 2nd level – more levels may further improve performance. For multiple levels it is not clear what the right dimensionality breakdown between levels is. It is also possible to combine different descriptors at different levels and this could be examined in future.
Also, the multi-level descriptors tend to be very high dimensional. These may benefit from the application of dimensionality reduction, such as by PCA [54].

Other Future Improvements:
The methods of this thesis have been presented largely separately. However, performance improvements could be obtained by combining some or all of these methods. For example, geometric matching could be used together with dense descriptors to improve performance.

Also, following another idea of Hinterstoisser [45], very substantial increases in matching performance can be achieved by combining SIFT and correlation descriptors successively. We intend to develop this in future.

## 11.4 Conclusion

Interest point matching is a subject which is still undergoing evolutionary improvement. After very rapid advances at the beginning of this decade, gradual improvements are now being achieved.

This thesis has presented a variety of methods whereby current interest point matching performance can be improved or expanded. These include both point matching and class discrimination techniques. None of the ideas in this thesis constitute a revolutionary advancement in this topic, and all of them have limitations. Nonetheless they do have usefulness in some situations, and do contribute to steady advancement in the understanding and practice of interest point matching.

# Author’s Publications

## Published or Accepted for Publication

S. Werkhoven, J. S. Jin, ’Projective Invariant and Non-Planar Interest Points’, *Proceedings of 2005 Asia-Pacific Workshop on Visual Information Processing*, Hong Kong, pp.200-205 (2005)

S. Werkhoven and S. Luo. Improving Interest Point Object Recognition and Its Applications in Multimedia. Book chapter in *Computer Vision for Multimedia Applications - Methods and Solution.* IGI Global, accepted on 01/09/2009, (2009)

## Intention to Publish

S. Werkhoven and J. S. Jin. Multi-level interest point descriptors for point matching.

S. Werkhoven and J. S. Jin. Dense interest point matching.

S. Werkhoven and J. S. Jin. Geometric matched interest points.

# Bibliography

[1] Caltech image datasets
http://www.vision.caltech.edu/Image_Datasets,
(retrieved 6/6/2006)

[2] Oxford image set
http://www.robots.ox.ac.uk/~vgg/research/affine/index.html,
(retrieved 10/7/2005)

[3] Bundler http://phototour.cs.washington.edu/bundler/,
(retrieved 16/11/2008)

[4] The Asirra CAPTCHA project
http://research.microsoft.com/en-us/um/redmond/projects/asirra/,
(retrieved 6/2/2009)

[5] Statistical Matlab Package for Matlab
http://cmp.felk.cvut.cz/cmp/software/stprtool/

[6] OpenCV http://sourceforge.net/projects/opencvlibrary/

[7] Matlab http://www.mathworks.com/

[8] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A.Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM*, 45:891–923, 1998.

[9] A. Baumberg. Reliable feature matching across widely separated views. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Hilton Head Island, South Carolina, USA*, pages 774–781, 2000.

[10] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *ECCV*, 2006.

[11] S. Belongie, J. Malik, and J. Puzicha. Matching shapes. The 8th *International Conference on Computer Vision*, Vancouver, Canada, pages 454–461, 2001.

[12] J. Beis and D. Lowe. Shape Indexing using Approximate Nearest-Neighbour Search in High-Dimensional Spaces. In *Conference on Computer Vision and Pattern Recognition*, pages 1000–1006, Puerto Rico, 1997.

[13] A. C. Berg and J. Malik. Geometric blur for template matching. In *CVPR*, pages 607–614, 2001.

[14] A. Berg, T. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *Proc. CVPR*, volume 1, pages 26–33, 2005.

[15] G. Bouchard and B. Triggs. Hierarchical part-based visual object categorization. In *CVPR*, 2005.

[16] Y. Boykov, O. Veksler, and R. Zabih. Fast Approximate Energy Minimization via Graph Cuts. *PAMI*, 23(11), 2001.

[17] M. Brown and D. Lowe. Recognising panoramas. In *Proceedings of the 9th International Conference on Computer Vision, Nice, France*. IEEE Computer Society Press, October 2003.

[18] M. Brown and D. G. Lowe. Invariant features from interest point groups. In *The 13th British Machine Vision Conference, Cardiff University, UK*, pp. 253-262, 2002.

[19] J. Canny. A computational approach to edge detection. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[20] G. Carneiro and A. Jepson. Flexible spatial models for grouping local image features. In *CVPR*, 2004.

[21] G. Carneiro and A.Jepson. The distinctiveness, detectability, and robustness of local image features. In *CVPR*, 2005.

[22] C. Cortes and V. Vapnik. Support vector network. *Machine Learning* 20, 273-297, 1995.

[23] G. Csurka, C. Bray, and C. Dance L. Fan. Visual categorization with bags of keypoints. In *ECCV Workshop on Statistical Learning in Computer Vision*, 2004.

[24] N. Dalal and B.Triggs. Histogram of oriented gradients for human detection. *Computer Vision and Pattern Recognition*, vol 1, pp 886-893, June 2005.

[25] G. Dorko and C. Schmid. Selection of scale-invariant parts for object class recognition. In *Proceedings of the 9th International Conference on Computer Vision, Nice, France*, 2003.

[26] G. Dorko and C. Schmid. Object Class Recognition Using Discriminative Local features. Technical Report RR-5497*, INRIA Rhone-Alps*, 2005.

[27] J. Elson, J. Douceur, J. Saul. Asirra: a CAPTCHA that exploits

interest-aligned manual image categorization. *Proc. of the 14th ACM conference on Computer and communications security*. pp 366-374, 2007.

[28] O.Faugeras and Q.T. Luong. *The Geometry of Multiple Images*. MIT Press, 2001.

[29] P. Felzenszwalb and D. Huttenlocher. Efficient matching of pictorial structures. *CVPR, Hilton Head Island, South Carolina, USA*, pages 66–75, 2000.

[30] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

[31] L. Fei-Fei and P. Perona. A Bayesian Hierarchical Model for Learning Natural Scene Categories. In *CVPR*, 2005.

[32] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, 2003.

[33] V. Ferrari, T. Tuytelaars, and Luc Van Gool. Simultaneous object recognition and segmentation by image exploration. *In ECCV*, 2004

[34] L. Florack, B. Haar Romeny, J. Koenderink and M. Viergever. General intensity transformations and differential invariants. In *JMIV* 4: 171–187, 1994.

[35] H. Freeman. On the encoding of arbitrary geometric configuration. IRE transactions on electronic computers, EC-10(2):260-268, 1961.

[36] 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.

[37] A. Frome, D. Huber, R. Kolluri, T. Bulow, and J. Malik. Recognizing objects in range data using regional point descriptors. In *Proc. 8th Europ. Conf. Comput. Vision*, vol. 3, pp. 224–237, 2004.

[38] D. Gabor. Theory of communication. *Journal I.E.E.*, 3(93):429 – 457, 1946.

[39] P. Golle. Machine learning attacks against the Asirra captcha. *Proc. of the 15th ACM conference on Computer and communications security.* pp 535-542. 2008.+

[40] K. Grauman and T. Darrell. Pyramid match kernels: Discriminative classification with sets of image features. In *Proc. ICCV*, 2006.

[41] T. Goedeme, T. Tuytelaars, and L. Van Gool. Fast wide baseline matching for visual navigation. vol. 1, pp.24-29, *Conference on Computer Vision and Pattern Recognition* (CVPR 04), 2004.

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

[43] R. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision. *Cambridge University Press*, 2nd Edition, 2003.

[44] S. Helmer, and D. Lowe. Object class recognition with many local features. In *Proc. CVPRW'04*, Vol. 12, pp 187, 2004.

[45] S. Hinterstoisser, S. Benhimane, V. Lepetit, P. Fua, and N. Navab. Simultaneous recognition and homography extraction of local patches with a simple linear classifier. *BMVC*, Leeds, September, 2008.

[46] D. Hoiem, R. Sukthankar, H. Schneiderman, and L. Huston. Object-based image retrieval using the statistical structure of images. *Conference on Computer Vision and Pattern Recognition*, 02:490–497, 2004.

[47] D. Huttenlocher, and S. Ullman. Object recognition using alignment. In *Proc. International Conference on Computer Vision*, pp 102-111, 1987.

[48] D. Huttenlocher, R. Lilien, and C. Olson. View-based recognition using an eigenspace approximation to the Hausdorff measure. *PAMI*, vol. +21, no. 9, pp. 951–955, Sept. 1999.

[49] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *ACM Symposium on Theory of Computing*, pp. 604–613, 1998.

[50] T. Joachims. Making large-scale svm learning practical. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. The MIT Press, Cambridge, MA, USA, 1999.

[51] Jurie, F., Schmid, C.: Scale-invariant shape features for recognition of object categories. In: *CVPR*. Vol. 2, 90 – 96, 2004.

[52] T. Kadir, M. Brady, and A. Zisserman. An affine invariant method for selecting salient regions in images. In *Proceedings of the 8th European Conference on Computer Vision, Prague, Czech Republic*,

2004.

[53] T. Kadir and M. Brady. Scale, Saliency and Image Description. *International Journal of Computer Vision*, 45 (2):83-105, 2001.

[54] Y. Ke and R. Sukthankar. PCA-sift: A more distinctive representation for local image descriptors. CVPR, *Washington, DC, USA*, pages 66–75, 2004.

[55] S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *CVPR*, 2006.

[56] S. Lazebnik, C. Schmid, and J. Ponce. Sparse texture representation using affine-invariant neighborhoods. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Madison, Wisconsin, USA*, 2003.

[57] S. Lazebnik, C. Schmid, and J. Ponce. A maximum entropy framework for part-based texture and object recognition. In Proc. ICCV, 2005.

[58] B. Liebe and B. Schiele. Interleaved object categorization and segmentation. In *BMVC*, 2003.

[59] B. Leibe and B. Schiele. Analyzing appearance and contour based methods for object categorization. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Madison, Wisconsin, USA*, June 2003.

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

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

[62] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 1150–1157, 1999.

[63] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 20(2):91–110, 2004.

[64] S. Mallat. A wavelet tour of signal processing. Academic Press, 2nd Edition, 1999.

[65] Marr, D. *Vision*. Freeman. 1982

[66] D. Martin, C. Fowlkes, and J. Malik. Learning to find brightness and texture boundaries in natural images. *NIPS*, 2002.

[67] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of the 13th British Machine Vision Conference, Cardiff, England*, pages 384–393, 2002.

[68] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Proceedings of the 8th International Conference on Computer Vision, Vancouver, Canada*, pages 525–531, 2001.

[69] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *IJCV*, 60(1):63–86, 2004.

[70] K. Mikolajczyk and C. Schmid. A Performance Evaluation of Local Descriptors. *PAMI*, 27(10):1615–1630, 2004.

[71] K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. *The 8th ECCV, Prague, Czech Republic*, volume I, pages 69–81, 2004.

[72] K. Mikolajczyk, B. Liebe and B. Schiele. Local features for object class recognition. In Proc. *ICCV*, 2005.

[73] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1):43–72, 2005.

[74] Mindru, F., Tuytelaars, T., Van Gool, L., Moons, T.: Moment invariants for recognition under changing viewpoint and illumination. *CVIU* **94** 3–27, 2004.

[75] G. Mori and J. Malik, "Recognizing objects in adversarial clutter: Breaking a visual captcha," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recognition,* vol. 1, pp. 134–141, 2003.

[76] J. Mundy and A. Zisserman. Geometric invariance in computer vision. *MIT Press*. 1992.

[77] J. Mutch and D.Lowe. Multiclass Object Recognition with Sparse, Localized Features. In *CVPR*, Vol 1, pp 11-18, 2006.

[78] D. Nister. An efficient solution to the 5-point relative pose problem. *IEEE Conference on Computer Vision and Pattern Recognition*. Vol 2,

pp195-202, 2003.

[79] S. Obdrzˇalek´ and J. Matas. *Toward Category-Level Object Recognition*, chapter 2, pages 85–108. J. Ponce, M. Herbert, C. Schmid, and A. Zisserman (Editors). Springer-Verlag, Berlin Heidelberg, Germany, 2006.

[80] K. Okajima. Two-dimensional Gabor-type receptive field as derived by mutual information maximization. *Neural networks* 11(3):441-447, 1998.

[81] M. Ozuysal, P. Fua, and V. Lepetit. Fast Keypoint Recognition in Ten Lines of Code. *Computer Vision and Pattern Recognition*, CVPR '07, pp. 1-8, June 2007.

[82] F. Porikli. Integral histogram: a fast way to extract histograms in cartesian spaces. In *CVPR*, pages 829–836, 2005.

[83] E. Rosten and T. Drummond. Machine learning for high-speed corner detection, *European Conference on Computer Vision*, May 2006.

[84] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. Object modeling and recognition using local affine invariant image descriptors and multi-view spatial constraints. *International Journal of Computer Vision*, 66(3):231–259, 2006.

[85] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3D Object modeling and recognition using affine-invariant patches and multi-view spatial constraints. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, Madison, Wisconsin, USA*, pages 272-277, June 2003.

[86] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, 2002.

[87] B. Schiele and J. Crowley. Recognition without correspondence using multidimensional receptive field histograms. *IJCV*, 36(1):31–50, 2000.

[88] C. Schmid and R. Mohr. Local Grayvalue Invariants for Image Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–534, May 1997.

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

2000.

[90] Henry Schneiderman and Takeo Kanade. Object detection using the statistics of parts. *IJCV*, 56(3):151–177, 2004.

[91] S. Se, D. Lowe, and J. Little. Global localization using distinctive visual features. In *International Conference on Intelligent Robots and Systems, IROS 2002, Lausanne, Switzerland*, October 2002.

[92] T. Sebastian, P. N. Klein, and B. B. Kimia. Shock-based indexing into large shape databases. In *European Conference on Computer Vision*, vol. 3, pp. 731–746, 2002.

[93] T. Serre, L. Wolf, and T. Poggio. Object Recognition with features inspired by visual cortex. In *CVPR*, San Diego, June 2005.

[94] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter sensitive hashing. In *Proc. 9th Int. Conf. Computer Vision*, vol. 2, pp. 750–757, 2003.

[95] L. Shen, and L. Bai. A review on Gabor wavelets for face recognition. In *Pattern Analysis & Applications*, vol. 9, Issue 2, pp. 273-292, 2006.

[96] J. Sivic and A. Zisserman. Video Google: Efficient visual search of videos. In Toward Category-Level Object Recognition, volume 4170 of LNCS, pages 127–144. Springer, 2006.

[97] J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman. Discovering objects and their location in images. In *Proc. ICCV*, 2005.

[98] S. M. Smith and J. M. Brady. SUSAN - a new approach to low level image processing. *International Journal of Computer Vision,* 23 (1): 45–78. May 1997.

[99] C. Strecha, T. Tuytelaars, and L. Van Gool. Dense Matching of Multiple Wide-Baseline Views. In *ICCV*, 2003.

[100] M. Swain and D. Ballard. Color indexing. *IJCV*, 7(1):11–32, 1991.

[101] D. Tell and S. Carlsson. Combining appearance and topology for wide baseline matching. In *Proceedings of the 7th European Conference on Computer Vision*, Copenhagen, Denmark, pages 814-828, 2002.

[102] A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla. Shape context and chamfer matching in cluttered scenes. In *Proc. Conf.*

*Compute Vision and Pattern Recognition*, vol. I, Madison, USA, pp. 127–133, 2003.

[103] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *CVPR*, pages 762–769, 2004.

[104] E. Tola, V. Lepetit and P. Fua. A fast local descriptor for dense matching. In *Proc of IEEE International Conference on Computer Vision and Pattern Recognition*, June 2008.

[105] M. Turk and A. Pentland. Eigenfaces for recognition, *J. Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–96, 1991.

[106] T. Tuytelaars and L. Van Gool. Wide Baseline Stereo Matching based on Local, Affinely Invariant Regions. In *BMVC*, pages 412–422, 2000.

[107] L. Van Gool, T. Moons, and D. Ungureanu. Affine / photometric invariants for planar intensity patterns. In *Proceedings of the 4th European Conference on Computer Vision, Cambridge, England*, pages 642–651, 1996.

[108] P. Viola, M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR* (1), 511–518, 2001.

[109] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *The 9th ICCV, Nice, France*, volume 1, pages 734–741, 2003.

[110] L. von Ahn, M. Blum, and J. Langford. Telling humans and computers apart (automatically). *CMU Tech Report CMU-CS-02-117*, February 2002.

[111] C. Wallraven, B. Caputo, and A. Graf. Recognition with local features: the kernel recipe. In *Proc. ICCV*, volume 1, pages 257–264, 2003.

[112] A. Witkin. Scale-space filtering, *Proc. 8th Int. Joint Conf. Art. Intell.*, Karlsruhe, Germany,1019–1022, 1983.

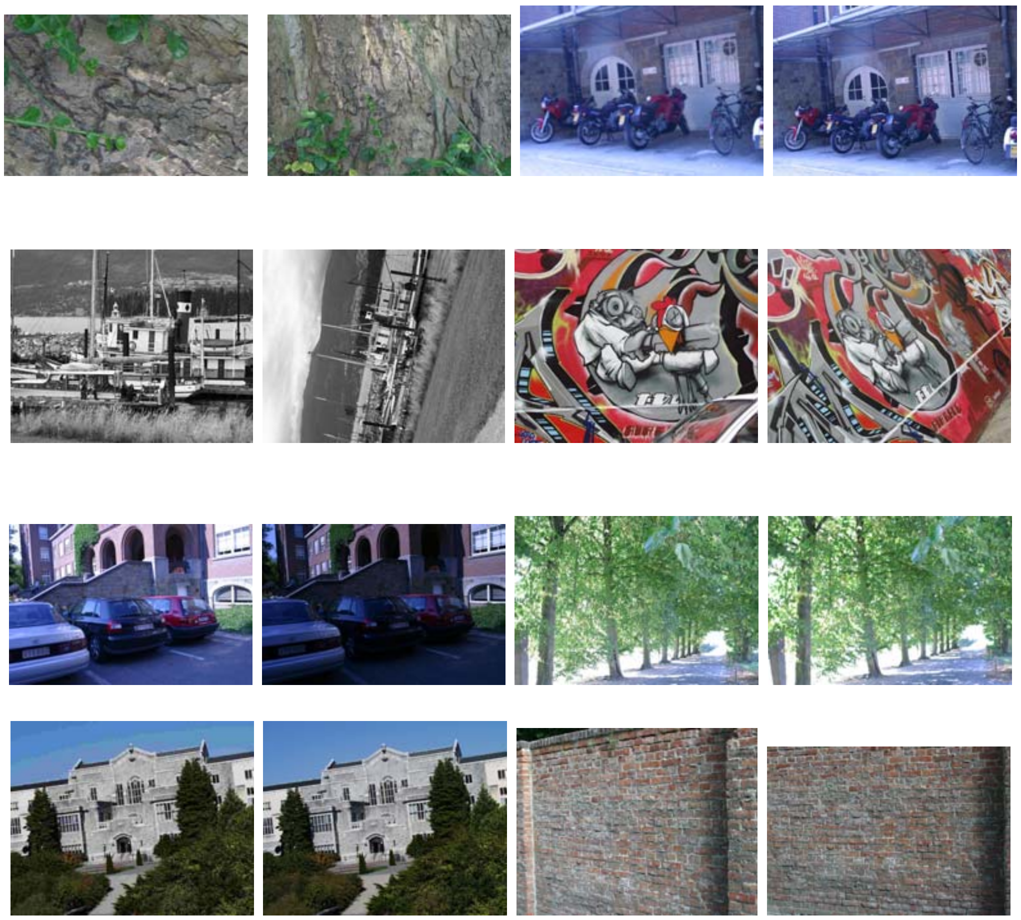
[113] H. Zhang, A. Berg, M. Maire, J. Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *Proc. CVPR*, volume 2, pages 2126–2136, 2006.

[114] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classifcation of texture and object categories: An in-

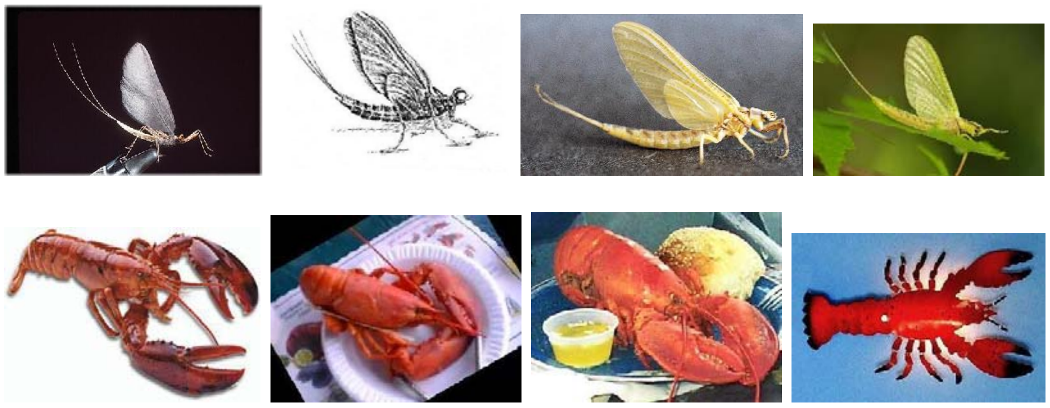
depth study. Technical Report RR-5737, INRIA Rhone-Alpes, 2005.

# Appendix I – Image Data Sets

## Examples of Oxford Affine Set



## Examples of Caltech 101 Set



Examples of Caltech Image Sets