1 Gaussian Filtered Representations of Images

S. Ravela and R. Manmatha[†]

Department of Computer Science
University of Massachusetts, Amherst, MA 01003
Email: {ravela,manmatha}@cs.umass.edu

[†]This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623, in part by the United States Patent and Trademark Office and Defense Advanced Research Projects Agency/ITO under ARPA order number D468, issued by ESC/AXS contract number F19628-95-C-0235, in part by the National Science Foundation under grant number IRI-9619117 and in part by NSF Multimedia CDA-9502639. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsors.

1.1 INTRODUCTION

Gaussian filtered representations of images have been used to address several important visual tasks. In early work Marr and Hildreth used them to attenuate noise and detect edges [29]. Specific operators such as the Laplacian of Gaussians and Difference of Gaussians have been used for multi-resolution analysis of images [3, 6]. More recently, several researchers [19, 21, 22, 42, 9] have shown that Gaussian derivatives may be used to robustly represent the local image structure at multiple scales. In this chapter, the Gaussian derivative filter and its spatial and frequency domain properties are examined and it is used to create a robust description of the local intensity surface. Motivated by its multi-scale properties Gaussian filtered representations are constructed to address two specific problems.

The first problem considered is of matching images that are affine deformed versions of each other. Solutions to this problem form an important component for several applications such as video mosaicing, registration, object recognition, structure from motion and shape from texture. In particular, consider an example where successive views of a scene are observed in a video. These views will be deformed versions of each other and under certain circumstances may be approximated using an affine transformation. Gaussians and its derivatives are used to recover affine deformations. Consider two patches related by an affine transform. If the patches are filtered using Gaussian filters, the outputs are equal provided the Gaussian is affine-transformed in the same manner as the function. Thus, one can construct a solution that minimizes the error with respect to the affine parameters, where the error is defined between corresponding Gaussian derivative filter outputs. The affine matching problem is discussed in detail in Section 1.3.

The second focus of this paper is in addressing the problem of image retrieval. Advances in computational power and the rapid increase in performance to cost ratio of most computational devices has led to the acquisition and storage of pictorial information on digital media. One of the important challenges that this trend presents is the development of algorithms for managing digitally stored pictorial information. While machine stored text can be searched using one of several text search engines, there are as yet no good tools available to search and manage image collections.

The reason that *image retrieval* has proven to be hard is because users expect the system to find relevant images based on some personal or cultural semantics. Representing semantics is hard and requires solutions to problems such as automatic feature detection, segmentation and recognition. These problems are as yet unsolved. However, in certain cases, many image attributes like color, texture, shape and "appearance" are often directly correlated with the semantics of the problem. For example, logos or product packages (e.g., a box of Tide) have the same color wherever they are found. The coat of a leopard has a unique texture while Abraham Lincoln's appearance is uniquely defined. These image attributes can often be used to index and retrieve images.

One such approach is to exploit the structure of the image intensity surface. In recent work Ravela and Manmatha, [34] have shown that robust representations of the intensity surface may be used to retrieve objects that appear visually similar. Arguably an object's visual appearance in an image is closely related to several factors including, among others, its three dimensional shape, albedo, surface texture and the imaged viewpoint. It is non-trivial to separate the different factors constituting an object's appearance. For

example, the face of a person has a unique appearance that cannot just be characterized by the geometric shape of the 'component parts'.

We argue that Gaussian filtered representation of images can be used for retrieval by appearance. A paradigm for retrieval that is used widely, and is adopted here, is that images in the database are processed and described by a set of feature vectors. These vectors are indexed ahead of time. During run-time, a query is provided in the form of an example image and its features are compared with those stored. Images are then retrieved in the order indicated by the comparison operator. In this work, feature vectors are constructed using responses to Gaussian derivative filters at multiple scales. Using this approach it is shown that whole images or parts thereof can be retrieved. This flexibility is important because a user interacting with an image retrieval system might be interested in the image as a whole, such as a trademark, or in only a part of the image, such as a face within a scene. In the former case the representation must capture the appearance of the whole image and similarity is global. In the latter case, the representation must allow for local similarity.

The remainder of this chapter is organized as follows. Section 1.2 provides a review of the Gaussian filter, some key properties and derives the features that will be used subsequently to address the affine image matching and retrieval tasks. In Section 1.3 matching of images under an affine deformation is considered and finally, in Section 1.4 image retrieval by appearance is discussed.

1.2 THE GAUSSIAN FILTERED REPRESENTATION OF IMAGES

This section begins by examining the spatial and frequency characteristics of the Gaussian filter Then, the role of the Gaussian filter in providing a robust representation of the intensity surface is discussed. The section ends with a discussion of how to implement discrete versions of Gaussians and their derivatives.

1.2.1 The Gaussian Filter: Preliminaries

The Gaussian and its derivatives: The isotropic normalized Gaussian in two dimensions is a C^∞ smooth function, defined as

$$G(\mathbf{p}, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-\mathbf{p}^T\mathbf{p}}{2\sigma^2}} \tag{1.1}$$

where $\mathbf{p} = \langle x, y \rangle \in R^2$, and $\sigma \in R$ is referred to as the scale. The derivatives of the normalized Gaussian are defined to an arbitrary order. The n^{th} derivative of the Gaussian (in two dimensions) is written in tensor form as

$$G_{i_1 \ldots i_n}(\cdot, \sigma) = \frac{\delta^n G(\cdot, \sigma)}{\delta i_1 \ldots \delta i_n} \tag{1.2}$$

where the free variables $i_1 \ldots i_n$ cycle through all the degrees of freedom (x, y). Thus the first derivative is written as G_{i_1}, which in the Cartesian frame is G_x and G_y.

Filtering: In the spatial domain, the discrete two dimensional image $Z(\mathbf{p})$ filtered with the Gaussian[1] is expressed as the discrete convolution, $I(\mathbf{p},\sigma) = (Z \star G)(\mathbf{p},\sigma)$. In the case of Gaussian derivatives since differentiation commutes with convolution, the following expression may be written:

$$I_{i_1 \ldots i_n}(\mathbf{x},\sigma) = (Z \star G)_{i_1 \ldots i_n}(\mathbf{x},\sigma) = (Z_{i_1 \ldots i_n} \star G)(\mathbf{x},\sigma) = (Z \star G_{i_1 \ldots i_n})(\mathbf{x},\sigma) \tag{1.3}$$

Frequency domain interpretation: The Fourier transform of the two dimensional Gaussian defined in Equation 1.1 is written as

$$\mathcal{G}(\mathbf{u},\sigma) = e^{\frac{-\sigma^2 \mathbf{u}^\mathbf{T}\mathbf{u}}{2}} \tag{1.4}$$

where $\mathbf{u} = \langle u_x, u_y \rangle \in R^2$ is the two dimensional frequency variable. Similarly, the Fourier transform of the n^{th} derivative of the Gaussian defined in Equation 1.2 is defined as

$$\mathcal{G}^{(i_1 \ldots i_n)}(\mathbf{u},\sigma) = j^n (u_{i_1} \ldots u_{i_n}) \mathcal{G}(\mathbf{u},\sigma) \tag{1.5}$$

Here, $i_1 \ldots i_n$ are free variables that are substituted with any of the Cartesian degrees of freedom and $j = \sqrt{-1}$. For example, the Fourier transform of the second mixed derivative G_{xy} is $\mathcal{G}^{(i_1 i_2)}(\mathbf{u},\sigma) = -u_x u_y \mathcal{G}(\mathbf{u},\sigma)$ where the substitution $i_1 = x$ and $i_2 = y$ is made. In a Cartesian coordinate system the Fourier transform of the n^{th} Gaussian derivative may be written as

$$\mathcal{G}^{(i_1 \ldots i_n)}(\mathbf{u},\sigma) = \mathcal{G}^{(x^p y^q)}(\mathbf{u},\sigma) = j^{p+q}\left(u_x^p u_y^q\right)\mathcal{G}(\mathbf{u},\sigma) \tag{1.6}$$

for some integers $p, q \geq 0, p+q = n$, and p free variables are instantiated to x and q to y.

From the above two definitions and using composition one may immediately observe the following properties:

Cartesian Separability: The Fourier transform of the n^{th} Gaussian derivative may be expressed as the composition of 1D filters.

$$j^{p+q}\left(u_x^p u_y^q\right)\mathcal{G}(\mathbf{u},\sigma) = j^p u_x^p e^{\frac{-\sigma^2 u_x^2}{2}} \cdot j^q u_y^q e^{\frac{-\sigma^2 u_y^2}{2}} = \mathcal{H}_x^{(p)}(u_x,\sigma) \cdot \mathcal{H}_y^{(q)}(u_y,\sigma) \tag{1.7}$$

Since composition implies convolution in the spatial domain, one can implement the n^{th} Gaussian derivative using separable 1D convolutions.

Cascade Property: The cascade property of Gaussian derivatives may be observed from the composition of two filters in the frequency domain.

$$\mathcal{G}^{(x^p y^q)}(\mathbf{u},\sigma_1) \cdot \mathcal{G}^{(x^m y^n)}(\mathbf{u},\sigma_2) = \mathcal{G}^{\left(x^{p+m} y^{q+n}\right)}\left(\mathbf{u}, \sqrt{\sigma_1^2 + \sigma_2^2}\right) \tag{1.8}$$

[1]The Gaussian has infinite extent. Since discrete images are typically of finite extent, the truncation of the Gaussian extent is considered in Section 1.2.4

Thus, filtering a signal successively with several Gaussian filters of scales $\sigma_1 \ldots \sigma_n$ is equivalent to filtering the signal with a single Gaussian filter of scale $\sigma = \sqrt{\sigma_1^2 + \ldots + \sigma_n^2}$.

Center Frequency and Bandwidth: The Gaussian filter is a low pass filter and the derivatives are band-pass filters. In what follows, the center frequency and bandwidth of the n^{th} derivative of the (spatial) Gaussian are derived. For the sake of simplicity 1D Gaussians are considered.

The Fourier transform of the n^{th} derivative of a *1D Gaussian* G_{x^n} is written as

$$\mathcal{G}^{(x^n)}(u,\sigma) = j^n u^n e^{-\frac{\sigma^2 u^2}{2}}$$

Differentiating with respect to u and computing the extremum one obtains the center frequency u_0:

$$\begin{aligned} \frac{d\mathcal{G}^{(x^n)}(u,\sigma)}{du} &= (n - u^2\sigma^2) j^n u^{n-1} e^{-\frac{\sigma^2 u^2}{2}} = 0 \\ u_0 &= \stackrel{+}{-} \frac{\sqrt{n}}{\sigma} \end{aligned} \tag{1.9}$$

It should be noted that the 1D function $\mathcal{G}^{(x^n)}$ is unimodal in either half plane and has a Gaussian envelope. It is odd and imaginary for odd order spatial derivatives and even and real for even order spatial derivatives. This equation states that the center frequency is coupled both to the bandwidth and to the order of the derivative. As the order of the derivative increases, so does its center frequency and therefore, higher derivatives enhance higher level of spatial detail.

There are several ways to to define the bandwidth of the filter. Here we adopt the equivalent rectangular bandwidth formulation [13]. This formulation equates the area of the power of the filter with an equivalent ideal filter of width W and height equal to the peak instantaneous power of the filter. Therefore, one may write

$$W \times \left| (ju_0)^n e^{-\frac{\sigma^2 u_0^2}{2}} \right|^2 = \int_0^\infty \left| (ju)^n e^{-\frac{\sigma^2 u^2}{2}} \right|^2 du \tag{1.10}$$

After some manipulation the solution to this relation can be shown to be the following:

$$W = W(n,\sigma) = \frac{e^n \pi}{4\sigma n^n} \Pi_{i=1}^{i=n} \left(i - \frac{1}{2} \right), n \geq 1 \tag{1.11}$$

$$W(0,\sigma) = \frac{\sqrt{\pi}}{\sigma} \tag{1.12}$$

The above expressions show that the center frequency and bandwidth of the Gaussian and its derivatives are related to the order and scale of the derivative in the spatial domain. The Gaussian is a low pass filter filter while its derivatives are band pass filters.

Noise attenuation: Since the Gaussian derivatives are band pass filters they may be used to attenuate noise. In particular, consider a one dimensional function $Z(x) = P(x) + \epsilon sin(\omega_1 x)$. The Fourier Transform of $z(x)$ is

$$\mathcal{Z}(u) = \mathcal{P}(u) + N(u)$$

$$N(u) = j\pi[\delta(u + w_1) - \delta(u - w_1)]$$

Consider the application of the n^{th} derivative of a 1D Gaussian to $Z(x)$ using the RHS of Equation 1.3, i.e. $I_n(x,\sigma) = Z(x) \star G_n(x,\sigma)$. The Fourier transform of $I_n(x,\sigma)$ is

$$\begin{aligned} \mathcal{I}^{(n)}(u,\sigma) &= \mathcal{G}^{(n)}(u,\sigma)\mathcal{P}(u) + \mathcal{G}^{(n)}(u,\sigma)N(u) \\ &= \ldots + j^n u^n e^{-\frac{\sigma^2 u^2}{2}} N(u) \end{aligned} \tag{1.13}$$

where $\mathcal{G}^{(n)}$ is the Fourier transform of the n^{th} Gaussian derivative. The second term on the RHS of Equation 1.13 can be made arbitrarily small by choosing an appropriate σ, eliminating the noise in the function Z.

1.2.2 Representation of the Intensity Surface

The local spatial structure of the intensity surface can be approximated using the local spatial derivatives of the surface. This can be seen from the Taylor series expansion of the intensity surface. The Taylor expansion around the neighborhood of a point in the image will fully describe the local intensity surface up to the order to which the series is constructed. Consider an image I at point $\mathbf{p}$, $\mathbf{p} = \langle x, y \rangle$. The value at a location $\mathbf{p} + \mathbf{\Delta p}$ can be estimated using the Taylor series expansion (written in tensor form):

$$I(\mathbf{p} + \mathbf{\Delta p}) \approx \sum_{n=0}^{N} \frac{1}{n!} \left(\mathbf{\Delta p}_{i_1} \ldots \mathbf{\Delta p}_{i_n} \frac{\delta^n I(\mathbf{p})}{\delta i_1 \ldots \delta i_n} \right)$$

Each term i_j $(j = 1 \ldots n)$ is substituted for all the degrees of freedom, which in the case of a 2D image is two. Up to order two, the expansion becomes

$$\begin{aligned} I(\mathbf{p} + \mathbf{\Delta p}) &= \sum_{n=0}^{2} \frac{1}{n!} \left(\mathbf{\Delta p}_{i_1} \ldots \mathbf{\Delta p}_{i_n} \frac{\delta^n I(\mathbf{p})}{\delta i_1 \ldots \delta i_n} \right) \\ &= I(\mathbf{p}) + \mathbf{\Delta p}_{i_1} \frac{\delta I(\mathbf{p})}{\delta i_1} + \frac{1}{2!} \left(\mathbf{\Delta p}_{i_1} \mathbf{\Delta p}_{i_2} \frac{\delta^2 I(\mathbf{p})}{\delta i_1 \delta i_2} \right) \\ &= I(\mathbf{p}) + \mathbf{\Delta p}_x \frac{\delta I(\mathbf{p})}{\delta x} + \frac{1}{2!} \left(\mathbf{\Delta p}_x^2 \frac{\delta^2 I(\mathbf{p})}{\delta x^2} + \mathbf{\Delta p}_x \mathbf{\Delta p}_y \frac{\delta^2 I(\mathbf{p})}{\delta x \delta y} \right) \\ &\quad + \mathbf{\Delta p}_y \frac{\delta I(\mathbf{p})}{\delta y} + \frac{1}{2!} \left(\mathbf{\Delta p}_y \mathbf{\Delta p}_x \frac{\delta^2 I(\mathbf{p})}{\delta y \delta x} + \mathbf{\Delta p}_y^2 \frac{\delta^2 I(\mathbf{p})}{\delta y^2} \right) \end{aligned}$$

Rearranging the terms we get the more familiar Cartesian form of the Taylor series (again up to order two) :

$$I(\mathbf{p}+\mathbf{\Delta p}) = I(\mathbf{p}) + \mathbf{\Delta p}_x \frac{\delta I(\mathbf{p})}{\delta x} + \mathbf{\Delta p}_y \frac{\delta I(\mathbf{p})}{\delta y} + \frac{1}{2!}\left(\mathbf{\Delta p}_x^2 \frac{\delta^2 I(\mathbf{p})}{\delta x^2} + 2\mathbf{\Delta p}_x \mathbf{\Delta p}_y \frac{\delta^2 I(\mathbf{p})}{\delta x \delta y} + \mathbf{\Delta p}_y^2 \frac{\delta^2 I(\mathbf{p})}{\delta y^2}\right)$$

The above equation states that in order to estimate the intensity in the neighborhood of a point $\mathbf{p}$ the derivatives at $\mathbf{p}$ must be known. Therefore, it can be argued that spatial derivatives may be used to approximate the local intensity surface.

In the case of digital images, which are two dimensional discrete functions of finite range, derivatives may be approximated using finite difference operators. However, while finite differences can be computed, their outputs must be meaningful or *well conditioned* in the presence of noise. In Section 1.2.1 it is shown that adding a high frequency, low amplitude noise may make the derivatives unstable. In discrete images this will result in noisy measurements.

A solution to the problem lies in the fact that the derivatives of a possibly discontinuous function become well-conditioned if it is first convolved with the derivative of a smooth (C^∞) test function [9]. The Gaussian is a smooth test function and therefore the derivatives of the smoothed image $I(\mathbf{x},\sigma) = (Z \star G)(\mathbf{x},\sigma), \mathbf{x} \in \Re^2$, are well conditioned for some value of σ. Another way of observing this is from the noise attenuation property presented in Section 1.2.1.

The operational scheme for computing local structure at a given scale of observation σ is as follows. Each image is filtered with Gaussian derivatives(at a certain scale) to the order to which the local structure is desired to be approximated. Therefore, each pixel is associated with a set of derivatives that completely define the Taylor expansion to the desired order. Koenderink [21] has advocated the use of this representation and calls it the local N-jet. The local N-jet of $I(\mathbf{x})$ at scale σ and order N is defined as the set:

$$J^N[I](\mathbf{x},\sigma) = \{I_{i_1 \ldots i_n,\sigma} | n = 0 \ldots N\} \tag{1.14}$$

Observe that $\lim_{N\to\infty} J^N[I](\mathbf{x},\sigma)$, bundles all the derivatives required to fully reconstruct the surface I_σ in a locality around $\mathbf{x}$ at a particular scale. This is the primary observation that is used to characterize local structure. That is, up to any order the derivatives locally approximate the regularized intensity surface. As a practical example consider the local 2-jet of an image $I(\mathbf{p}), \mathbf{p} = \langle x, y\rangle \in \Re^2$, at scale σ.

$$J^2[I](\mathbf{p},\sigma) = \{I, I_x, I_y, I_{xx}, I_{xy}, I_{yy}\}(\mathbf{p},\sigma)\,^2$$

Image I is filtered with the first two Gaussian derivatives (and the Gaussian itself) in both x and y directions. Point $\mathbf{p}$ is, therefore, associated with a *derivative feature vector* of responses at scale σ.

[2] $I_{yx} = I_{xy}$ and is therefore dropped

1.2.2.1 Multi-Scale Representation and Scale-space The derivative feature vector is computed at a single scale, and therefore, constitutes observations of the intensity surface at a fixed bandwidth. Equivalently, in the spatial domain, the intensity surface is observed at a fixed window size. In effect, the derivative feature vector constitutes observations, not of the original image but, of a smoothed version of it. Therefore, computing derivatives at a single scale is not likely to be a robust representation of local structure. Fundamentally, this is because the local structure of the image depends on the scale at which it is observed. An image will appear different depending on the scale at which it is observed. For example, at a small scale the texture of an ape's coat will be visible. At a large enough scale, the ape's coat will appear homogeneous.

A better characterization of local structure is obtained by computing derivatives at several scales of observation. In the frequency domain this amounts to sampling the frequency spectrum of the original image using several bandwidths (scale) around multiple center frequencies (derivatives). In the spatial domain, it may be viewed as computing local derivatives at several neighborhoods around a point.

The Gaussian forms a very attractive choice for a multi-scale operator for several reasons. First, it is naturally defined with respect to a continuous scale parameter. Second, under certain conditions [19], it uniquely generates the linear scale-space of an image.

The term scale-space was introduced by [46] to describe the evolution of image structure over multiple scales. Starting from an original image images successively smoothed images are generated along a scale dimension. In this regard several researchers [19, 22, 9, 42] have shown that the Gaussian uniquely generates the linear scale-space of the image when it is required that structures present at a coarser scale must already be present at a finer scale. That is, no new structures must be introduced by the operator used to generate scale-space. Typically, these structures are the zero crossings or local maxima of the image intensity. This is a very significant result because it provides a formal mechanism to represent multi-scale information using a well defined operator.

More formally, the scale-space of an image $Z(\mathbf{p}), \mathbf{p} = \langle x, y \rangle \in \Re^2$ may be written as the one parameter family of derived images obtained using the Gaussian operator G.

$$I(\mathbf{p}, \sigma) = Z(\mathbf{p}) \star G(\mathbf{p}, \sigma)$$

The linear scale-space representation models an important physical observation. As an object moves away from a camera (in depth) its image appears less structured and finer contrasts get blurred. The change in intensity in a locality around a pixel that occurs with changing distance is accurately represented in the scale-space trajectory of this pixel. A detailed analysis deriving the Gaussian as the unique linear scale-space operator is beyond the scope of this document. For an in depth study the reader is pointed to Florack's dissertation [9].

From a practical perspective, the Gaussian allows local structure to be computed at several scales of obser-

vation that are related in a precise manner. Local structure as represented by the spatial derivatives may be computed directly across scales without explicitly computing the scale-space. This may be seen from Equation 1.3. In fact, Lindeberg [22] shows that the scale-space is well defined for the Gaussian derivatives as well. Therefore the Gaussian filtered representation is useful in at least two ways. First, it allows for the stable and efficient computation of local-structure. Second, it is the only solution to generate the linear scale-space.

An argument is therefore made for a *multi-scale feature vector* which describes the intensity surface locally at several scales. A multi-scale vector represents the intensity surface better than a single-scale vector and, is therefore, a robust representation. From an implementation stand point a *multi-scale feature vector* at a point p in an image I is simply the vector:

$$J^N_{(\sigma_1 \ldots \sigma_k)}[I] = \left\{ J^N[I](\mathbf{p}, \sigma_1), J^N[I](\mathbf{p}, \sigma_2) \ldots J^N[I](\mathbf{p}, \sigma_k) \right\} \tag{1.15}$$

for some order N and a set of scales $\sigma_1 \ldots \sigma_k$.

A natural question that arises in building representations for various applications is the parameterization required for the multi-scale feature vector. That is, the number of scales to be used, their spacing and the number of orders to be considered. In the applications described in this paper the scales are placed half an octave apart ($\sqrt{2}$) and typically three to five scales are used. In all cases only the first two orders are used and higher orders are ignored.

1.2.3 Behavior under Coordinate Deformations

There are several additional properties that make the Gaussian a suitable operator for analysis of images. In this section we examine the behavior of the Gaussian and its derivatives with respect to coordinate deformations of the image. In particular behavior with respect to size changes and 2D rotations of the coordinate frame are considered.

1.2.3.1 Scaling Theorems Gaussian derivatives may be used to compare image patches that are scaled versions of each other in a straightforward manner. Consider two images I_0 and I_1 that are scaled versions of each other (but otherwise identical). Without loss of generality assume that the scaling is centered at the origin. That is $I_0(\mathbf{p}) = I_1(s\mathbf{p})$ Then the following relations hold [25, 28]

$$\begin{array}{ll} I_0(\mathbf{p}) \star G(\cdot, \sigma) & = I_1(s\mathbf{p}) \star G(\cdot, s\sigma) \\ I_0(\mathbf{p}) \star G^{(k)}(\cdot, \sigma) & = I_1(s\mathbf{p}) \star G^{(k)}(\cdot, s\sigma) \\ \text{where, } G^{(k)}(\cdot, t) & = t^k G_{i_1 \ldots i_k}(\cdot, t) \end{array} \tag{1.16}$$

We call these the *Scale Shifting Theorems* or simply *scaling theorems.* These equations state that if the image I_s is a scaled version of I_0 by a factor s then in order to compare any two corresponding points in these images the filters must also be stretched (i.e. scaled) by the same factor. For example, if a point p_0 is being

compared with a point p_1 in images I_0 and I_1 where I_1 is twice the size of I_0, then for the responses to be equal, the filter used to compute the response at p_1 must be at twice the scale of that applied at p_0. This property has been exploited for matching affine deformed images (see Section 1.3), object recognition [32] and image retrieval [35].

1.2.3.2 Steerability The Gaussian derivatives may be combined under rotations to synthesize filters in an arbitrary orientation. This has been called the steering property [10]. This property is interesting for two reasons. First, images may be filtered using Gaussian derivatives tuned to any arbitrary orientation without actually rotating the filters. The tuned filters may be expressed as a combination of filters in a normal coordinate frame. Therefore, responses to any steered direction may be computed as a simple rotation of the responses. Thus, separable implementations are feasible even for rotated filters. Second, it may be used as a basis for generating feature vectors that are invariant to 2D rotations discussed in the next section. The results for the first two orders are now derived.

Consider a 2D rotated version of the Cartesian coordinate frame $\mathbf{p} =< x, y >$ written as $\mathbf{q} =< x_2, y_2 >$ such that $\mathbf{q} = R^T\mathbf{p}$, where $\mathbf{p}$ and $\mathbf{q}$ are the respective coordinates and $\mathbf{R}$ is the rotation matrix. Assume for simplicity that all coordinates are right handed.

Gaussian: It is straightforward to show that $G(\mathbf{q},\sigma) = G\left(\mathbf{R^T p},\sigma\right) = G(\mathbf{p},\sigma)$ That is, the Gaussian is isotropic.

First derivatives: Consider the first derivatives of the 2D Gaussian. Then the following relationship holds

$$\begin{aligned}
\begin{bmatrix} G_{x_2}(\mathbf{q},\sigma) \\ G_{y_2}(\mathbf{q},\sigma) \end{bmatrix} &= -\frac{1}{\sigma^2}\mathbf{q}G(\mathbf{q},\sigma) \\
&= -\frac{1}{\sigma^2}\left(\mathbf{R^T p}\right)G\left(\mathbf{R^T p},\sigma\right) \\
&= \mathbf{R^T}\left(-\frac{1}{\sigma^2}(\mathbf{p})\,G(\mathbf{p},\sigma)\right) \\
&= \mathbf{R^T}\begin{bmatrix} G_x(\mathbf{p},\sigma) \\ G_y(\mathbf{p},\sigma) \end{bmatrix}
\end{aligned} \tag{1.17}$$

Second derivatives: Similarly the second derivative may also be steered.

$$\begin{aligned}
\begin{bmatrix} G_{x_2y_2}(\mathbf{q},\sigma) & G_{x_2y_2}(\mathbf{q},\sigma) \\ G_{y_2x_2}(\mathbf{q},\sigma) & G_{y_2y_2}(\mathbf{q},\sigma) \end{bmatrix} &= \frac{1}{\sigma^2}\left(\frac{1}{\sigma^2}\begin{bmatrix} x_2^2 & x_2y_2 \\ x_2y_2 & y_2^2 \end{bmatrix} - \mathbf{I_2}\right)G(\mathbf{q},\sigma) \\
&= \frac{1}{\sigma^2}\left(\frac{1}{\sigma^2}\mathbf{q}\mathbf{q^T} - \mathbf{I_2}\right)G(\mathbf{q},\sigma) \\
&= \frac{1}{\sigma^2}\left(\frac{1}{\sigma^2}\mathbf{R^T}\mathbf{p}\mathbf{p^T}\mathbf{R} - \mathbf{I_2}\right)G(\mathbf{Rp},\sigma) \\
&= \mathbf{R^T}\frac{1}{\sigma^2}\left(\frac{1}{\sigma^2}\mathbf{p}\mathbf{p^T} - \mathbf{I_2}\right)G(\mathbf{p},\sigma)\,\mathbf{R} \\
&= \mathbf{R^T}\begin{bmatrix} G_{xx}(\mathbf{p},\sigma) & G_{xy}(\mathbf{p},\sigma) \\ G_{yx}(\mathbf{p},\sigma) & G_{yy}(\mathbf{p},\sigma) \end{bmatrix}\mathbf{R}
\end{aligned} \tag{1.18}$$

where $\mathbf{I_2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Several authors have exploited the steerability property of Gaussian derivatives. Rao [32] steered multi-scale derivative vectors so as to represent the orientation with the best local responses and Ravela et. al. [33] exploit steerability to track image patches. In the next section this property is used in conjunction with differential invariants to construct multi-scale invariant vectors that are derivatives expressed in the local coordinate frame and are invariant to 2D rotations.

1.2.3.3 Rotational Invariants Gaussian derivatives may be steered to any given orientation. Therefore, the image derivatives can be stably computed along any orientation using Gaussian derivatives tuned to that orientation. This property allows the creation of 'features' that are invariant to 2D rotations of the image plane.

It is well known [22] that if some property of the local intensity surface is used to define a local coordinate frame that is a rotation of the image coordinate frame, then, derivatives computed in the local frame will be invariant to rotations of the intensity surface. For example, assume that a new coordinate frame is defined by the local gradient direction in the image I. In the above mentioned framework let the axis y_2 represent a direction parallel to the local gradient and let x_2 be orthogonal to it in a right hand coordinate sense. Then one may define an orthonormal matrix $\mathbf{R}$ such that

$$\mathbf{R} = \frac{1}{\sqrt{I_x^2+I_y^2}} \begin{bmatrix} I_y & I_x \\ -I_x & I_y \end{bmatrix} \tag{1.19}$$

Thus, an image filtered with the first Gaussian derivative steered to the $< x_2, y_2 >$ coordinate frame can be equivalently expressed in the image coordinate frame $< x, y >$ in the following manner:

$$\begin{bmatrix} I_{x_2} \\ I_{y_2} \end{bmatrix} = Z \star \begin{bmatrix} G_{x_2} \\ G_{y_2} \end{bmatrix} = Z \star \mathbf{R^T} \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \mathbf{R^T} \begin{bmatrix} Z \star G_x \\ Z \star G_y \end{bmatrix} = \begin{bmatrix} 0 \\ \sqrt{I_x^2 + I_y^2} \end{bmatrix} \tag{1.20}$$

The interpretation of this result is rather simple. The gradient magnitude is the directional derivative parallel to the direction of the gradient. It is also invariant to rotations. The second derivative may similarly be expressed. There are several ways in which one may construct the rotation matrix R. Further, given a multi-scale derivative feature vector to any order an infinite number of rotational invariants may be constructed.

However, Florack [9] has shown that given the derivatives of an image I up to a certain order, only a finite number of *irreducible differential invariants* exist and may be computed in a systematic manner. The

irreducible set of invariants up to order two of an image I are:

$$\begin{array}{lll} d_0 & = I & \text{Intensity} \\ d_1 & = I_x^2 + I_y^2 & \text{Magnitude} \\ d_2 & = I_{xx} + I_{yy} & \text{Laplacian} \\ d_3 & = I_{xx}I_xI_x + 2I_{xy}I_xI_y + I_{yy}I_yI_y & \\ d_4 & = I_{xx}^2 + 2I_{xy}^2 + I_{yy}^2 & \end{array}$$

The reason these are termed irreducible is that other invariants (up to that order) may be expressed as combinations as these. Thus, the multi-scale derivative vectors may be transformed so that they are invariant to 2D rotations in the image plane. In the retrieval by appearance application we use the vector (minus the intensity term) $\mathbf{\Delta}_\sigma = \langle d_1, \ldots d_4 \rangle_\sigma$, computed at three different scales. This representation to a higher order has also been used by Schmid and Mohr [38] for object recognition.

1.2.4 Implementation

Filtering may be carried out either in the spatial domain using convolution or in the frequency domain using composition. In the latter case the Fourier transform and the inverse Fourier transform will need to be computed before and after the composition operation.

The choice of the domain for filtering is dependent on the size of the kernel. For an image of size N and filter of size w, the complexity of spatial domain filtering using separable filters is $O(wN)$ while that using frequency domain filtering is $O\left(Nlog(N) + \frac{N^2}{w^2}\right)$. Thus, when the image and kernel sizes are small spatial domain filtering may be preferred while for large images frequency domain filtering may be advantageous. In addition if several operations need to be performed such as filtering with multiple derivatives then frequency domain filtering may be preferred.

In either domain, the issue of *discretization* has to be faced. The derivation in the previous section relied on continuous functions whereas practical implementations require discrete versions of these filters. In addition, in the spatial domain, *truncation* effects need to be considered as well. That is, the effect of truncating the Gaussian to a finite extent. In this section the effects of discretizing and truncating the filter in the spatial domain will be discussed.

1.2.4.1 Discretizing Gaussians and Gaussian Derivatives The derivations of the algorithms in the previous sections have assumed that the Gaussians and Gaussian derivatives were continuous functions. To apply them, they first need to be discretized. The discretization needs to be performed carefully since errors arise due to the discretization process [15, 22]. A number of different procedures have been suggested in the literature. These include:

1. Block Averaging: Block averaging the continuous kernel over each pixel i.e. the filter value is integrated over each pixel and then sampled [15]. Let the discrete filter be defined over the values -w to w (that is, its width is 2w + 1). Then the value of the discrete Gaussian kernel at a point i is given by:

$$g[i] = \int_{i-1/2}^{i+1/2} G(x,\sigma)dx = erf(\frac{i+1/2}{\sigma}) - erf(\frac{i-1/2}{\sigma}) \tag{1.21}$$

where $erf(x) = \int_0^x \exp(-t^2/2)dt$ is the erf function. The first derivatives may be similarly computed. For example, the first derivative of the Gaussian in the x direction is given by:

$$g_x[i] = \int_{i-1/2}^{i+1/2} G_x(x,\sigma)dx = G(i+1/2,\sigma) - G(1-1/2,\sigma) \tag{1.22}$$

2. Discrete Derivative: The Gaussian is first sampled. The image is convolved first with the sampled Gaussian and the output convolved with a discrete version of the derivative. This approach is widely used. The discrete Gaussian kernel at a point i is given by:

$$g[i] = G(i,\sigma) \tag{1.23}$$

A discrete version of the first derivative is given by the kernel $D = [-1, 0, 1]$. Then, a discrete version of the first Gaussian derivative in the x direction is given by:

$$g_x = D * g[i] \tag{1.24}$$

Since convolution is associative, the image may be first filtered with the discrete Gaussian and the result may then be filtered with the first derivative kernel D. Second derivatives may be computed using the second derivative kernel $D_2 = [-1, 2, -1]$.

3. Sampled Gaussian Derivatives: The continuous version of the Gaussian or Gaussian derivative is directly sampled at each pixel and the sample values used for the discrete version of the filter. The sampled Gaussian derivative is given by:

$$g_{x^n}[i] = G_{x^n}(i,\sigma) \tag{1.25}$$

4. Discrete Scale-Space: The values of the discrete Gaussian are computed using a discrete scale space. The image is filtered with the discrete Gaussian and then filtered with a discrete version of the derivative. See [22] for how to compute the discrete Gaussian.

The question arises as to which technique is appropriate. It may be argued that block averaging takes account of the imaging process and may, therefore, be assumed to be the best discretization [15, 22]. For example, when a scene is imaged by a charge coupled device (CCD) camera, the output of each pixel is proportional

to the total light falling over the entire area of each pixel [3] (that is, the integral of the brightness over that pixel).

The results obtained using sampled Gaussian derivatives approximate those due to block averaging provided the scales (σ) used are large. As the scale is reduced, the errors due to using a sampled Gaussian derivative increase. Typically, below $\sigma = 0.5$ sampled Gaussian derivatives should not be used. In practice, most scales used are larger and hence sampling Gaussian derivatives is usually a good method of computing discrete Gaussian derivatives.

Assume that a large number of Gaussian derivatives need to be computed. Then for each order of a Gaussian derivative, the Gaussian needs to be sampled and the image filtered with the appropriate discrete kernel. This can be expensive. Consider, for example, the 1D case and assume that the image needs to be filtered with derivatives up to order 2. Let the kernel width for the discrete versions of the Gaussian, first derivative and second derivative be 2w + 1. Then to filter the image with Gaussian derivatives up to order 2 requires time proportional to 3 * (2w + 1) = 6w + 3. This time may be reduced by using discrete derivatives to compute Gaussian derivatives. First, the image is filtered with a sampled Gaussian. The output of this image is filtered with the kernels D and $D2$ and this is equivalent to filtering the image with the first and second derivatives of the Gaussian. The time taken to filter, however, is only 2w + 7. Since w may often be large, Gaussian derivative filtering accomplished using discrete derivatives is cheaper to compute than using sampled Gaussian derivatives. The tradeoff is that the results obtained using discrete derivatives are not as accurate. That is, sampled Gaussian derivatives are a better approximation to block averages than discrete Gaussian derivatives (see [27])

1.2.4.2 Truncation of Gaussian Derivatives Gaussians and Gaussian derivatives are infinite in extent. However, most of their energies reside in a small region around the origin. Thus, for all practical purposes they may be truncated. Truncation also reduces the time taken to filter the images since the resulting kernel sizes are smaller. There has been some discussion about where Gaussians and Laplacian of Gaussians should be truncated [14, 15], but a general discussion of how Gaussian derivatives should be truncated seems to be absent in the literature (but see [27]).

Figure 1.1 shows the truncation errors for different values of the truncation radius. The truncation error is computed as follows:

$$TruncationError = \frac{\int_{-\infty}^{\infty} |G_{x^i}(x,\sigma)|dx - \int_{-k\sigma}^{k\sigma} |G_{x^i}(x,\sigma)|dx}{\int_{-\infty}^{\infty} |G_{x^i}(x,\sigma)|dx} \tag{1.26}$$

where $G_{x^i}(x,\sigma)$ is the ith order Gaussian derivatives (with G denoting the Gaussian) and k is the truncation radius. In the above equation, the difference between the areas of the absolute values of the truncated

[3] The area is actually better approximated as a weighted integral - the weight being a Gaussian [22].

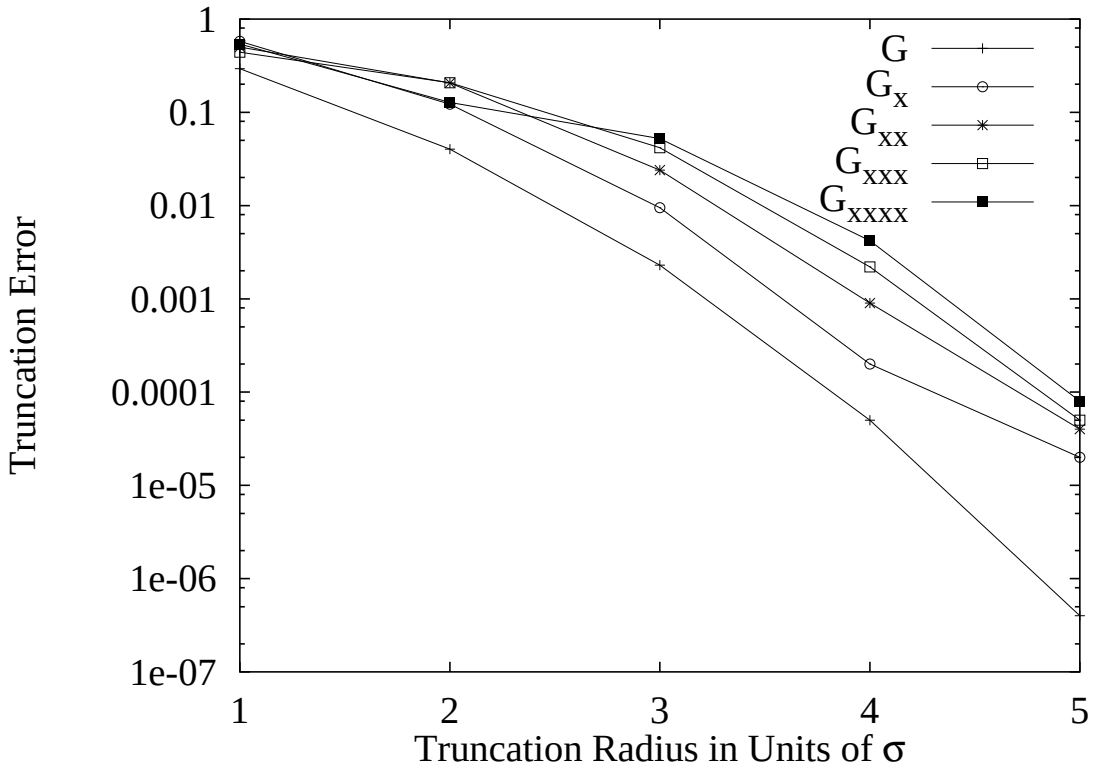

Fig. 1.1 Truncation errors as a function of the truncation radius for Gaussians and Gaussian derivative filters. G denotes the Gaussian and G_x, G_{xx}, G_{xxx} and G_{xxxx} denoting the first, second, third and fourth Gaussian derivatives respectively. The truncation errors are taken as a proportion of the total area of the absolute value of filter.

function and the untruncated function is first computed. Then this difference is divided by the area of the absolute values of the untruncated filter to give the truncation error as a proportion of the area of the untruncated filter. Note that the truncation error is independent of σ. The truncation errors in Figure 1.1 were computed by summing over discrete versions of the filter using large $\sigma's$ instead of computing the integrals analytically (the difference should be insignificant).

As Figure 1.1 shows, for a given truncation radius, the error increases with the order of the derivative. Many researchers assume that it suffices to truncate Gaussians such that the truncation radius is $\pm 2\sigma$ i.e. the filter width = 4σ. For a truncation radius of 2σ, 96% of the energy of the Gaussian is contained within the filter width (i.e. the truncation error is 0.04). But for the same truncation radius, Gaussian derivatives have a much larger truncation error. For example, the first derivative of the Gaussian has an error of about 12 % if it is truncated to within $\pm 2\sigma$ while higher derivatives have a much larger error. Figure 1.1 shows that the truncation error is less than 0.01 for the first four Gaussian derivatives if the truncation radius is greater than or equal to $\pm 4\sigma$. It may also be shown that the qualitative errors produced are large if the truncation radius is less than $\pm 4\sigma$ for derivatives up to order 4 (see [27]).

1.2.5 Suggested Reading

Multi-resolution representations are related to multi-scale representations. A classical multi-resolution representation, the Laplacian pyramid [3], may be generated as follows.

$$\begin{aligned} I^{(n)} &= F\left[I^{(n-1)}\right] \\ I^{(0)} &= Z_{xx} + Z_{yy} \end{aligned}$$

where Z is the original image and $I^{(n)}$ is a representation at a coarser resolution. The operator F consists of two operations, the first one is a smoothing step and the second is a sub-sampling step. The smoothing step is required to reduce aliasing effects due to sub-sampling and may be implemented using a Gaussian. The sub-sampling factor of 2 is typically used and a coarser image is a quarter the size of its immediate predecessor. Multi-resolution representations may be used to compress images, detect features in a coarse to fine manner, and match features between images efficiently.

Multi-resolution representations are related to, but somewhat different from multi-scale representations. Multi-scale representations do not change the resolution of the original image but rather vary the size of the operator. It is trivial to build a multi-resolution representation from a multi-scale representation but the reverse is not true.

Although this section presents enough detail to motivate the use of Gaussian filtered representations, there are several aspects that are not covered. In particular, the study of scale-space is abbreviated and the user is referred to [46, 19, 22] for further review. Similarly an indepth study of rotational invariants is available in [9]. For some additional properties of the Gaussian filter such as optimality with respect to the uncertainty principle the reader is refered to [12]. In addition, there is a physiological motivation for using Gaussian filtered representations. For example, Young [47] shows that visual receptive fields in the primate eye are better modeled by Gaussian derivatives.

Several researchers have used multi-scale derivatives as a representation. In particular [32] uses multi-scale vectors and the steerability property to recognize objects. In earlier work [35] derivatives were combined with the scaling theorems to retrieve visually similar objects at different sizes. The next section is also a good example of using multi-scale derivatives. Multi-scale derivatives are used to recover the affine transform between deformed images.

1.3 MATCHING AFFINE DEFORMED IMAGES

In this section we will discuss how Gaussian and Gaussian derivative filters may be used to match images under affine transforms. The ability to match two images or parts of images is required for many visual tasks and is, therefore, quite useful. For example, recovering the structure of a scene requires matching two or more image patches arising from a scene viewed from different viewpoints. Other applications which require matching image patches include the registration of video sequences [36] and image mosaicing [16].

Successive images of a scene when taken from different viewpoints are deformed with respect to each other. To first order, the transformation between images caused by viewpoint change may be modeled using an affine transform. The affine transform interprets the image motion in terms of an image translation and a deformation. In 2D, the affine transformation may be described by the six parameters ($\mathbf{t}$, $\mathbf{A}$) where

$$\mathbf{r}' = \mathbf{t} + \mathbf{A}\mathbf{r} \tag{1.27}$$

$\mathbf{r}$' and $\mathbf{r}$ are the image coordinates related by an affine transform, $\mathbf{t}$ is a 2 by 1 vector representing the translation and $\mathbf{A}$ the 2 by 2 affine deformation matrix. The affine transform is useful because the the image projections of a small planar patch from different viewpoints are well approximated by it.

In general, affine transforms between image patches have been recovered in a number of different ways (for more details see [28]):

1. Matching image intensities by searching through the space of all affine parameters. This approach adopts a brute force search strategy which is slow [17].
2. By linearizing the image intensities with respect to the affine parameters. This may be done at each pixel to give one equation per pixel. By assuming that the same affine transformation is valid over some region, an over-constrained system of equations is obtained. Linearization limits these algorithms to cases when the affine transforms are small. [1, 4, 40]
3. By filtering the image with Gaussians and linearizing the filter outputs [45]. The results are poor for general affine transforms because only the filter outputs from a single pixel are used.
4. Line based methods which match the closed boundaries of corresponding regions [5, 37]. However, they are limited to homogeneous regions with closed boundaries.
5. Patches deformed under similarity transforms may also be matched using the Mellin-Fourier Transform ([39]). Although possible, recovery of the affine transform has not been demonstrated. The main drawback to these techniques is that they are inherently global and they are not applicable to general affine transforms.

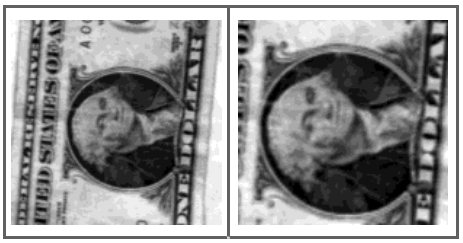

Fig. 1.2 Dollar Bill scaled 1.4 times

The difficulty with measuring affine transforms is indicated in Figure (1.2) where the image on the right is scaled 1.4 times the image on the left. Even if the centroids of the two images are matched accurately,

measuring the affine transform is difficult since the sizes of every portion of the two images differ. This problem arises because traditional matching uses fixed correlation windows or filters. The correct way to approach this problem is to deform the correlation window or filter according to the image deformation.

Here, we present a computational scheme where Gaussian and derivative of Gaussian filters are used and the filters deformed according to the affine transformation. First, it is shown that if an image is filtered with a Gaussian (or Gaussian derivative), then the affine transformed version of the image needs to be filtered with a deformed Gaussian (or Gaussian derivative) if the two filter outputs are to be equal; the deformation is equal to the affine transform. Thus, the problem of recovering the affine transform may be recast into the problem of finding the deformation parameters of the Gaussian (or Gaussian derivative). For example let Z_1 and Z_2 be two images which differ by a scale change s. Then, the output of Z_1 filtered with a Gaussian of σ will be equal to the output of Z_2 filtered with a Gaussian of $s\sigma$.

The resulting equations are solved by linearizing with respect to the affine parameters. Unlike the technique used in [45], the filter outputs from a number of points in a region are pooled together. This substantially improves the accuracy of the technique. For example, using Werkhoven and Koenderink's algorithm [45] on the images in Figure (1.2) returns a scale factor of 1.16 while the algorithm here matches correctly and, therefore, returns a scale factor of 1.41.

1.3.1 Deformation of Filters

The initial discussion will assume zero image translation; translation can be recovered as suggested in section 1.3.4. It is also assumed that shading and illumination effects may be ignored.

Consider two Riemann-integrable functions Z_1 and Z_2 related by an affine transform i.e.

$$Z_2(\mathbf{Ar}) = Z_1(\mathbf{r}) \tag{1.28}$$

Consider first the case where Z_1 is a scaled version of Z_2 i.e.

$$Z_2(s\mathbf{r}) = Z_1(\mathbf{r}) \tag{1.29}$$

Then,

$$\int Z_1(\mathbf{r})G(\mathbf{r},\sigma)d\mathbf{r} = \int Z_2(s\mathbf{r})G(\mathbf{r},\sigma)d\mathbf{r} \tag{1.30}$$

$$= \int Z_2(s\mathbf{r})G(s\mathbf{r},s\sigma)d(s\mathbf{r}) \tag{1.31}$$

That is, the output of Z_1 filtered with a Gaussian is equal to the output of Z_2 filtered with a scaled Gaussian. Note that this equation is also true for similarity transforms i.e. $\mathbf{A} = s\mathbf{R}$.

Define a generalized Gaussian by:

$$G(\mathbf{r}, \mathbf{M}) = \frac{1}{(2\pi)^{n/2} det(\mathbf{M})^{1/2}} exp(-\frac{\mathbf{r}^T \mathbf{M}^{-1} \mathbf{r}}{2}) \tag{1.32}$$

where $\mathbf{M}$ is a symmetric positive semi-definite matrix.

Then, if Z_1 and Z_2 are related by an affine deformation, the output of Z_1 filtered with a Gaussian is equal to the output of Z_2 filtered with a Gaussian deformed by the affine transform (see [26, 27] for a derivation).

$$\int Z_1(\mathbf{r}) G(\mathbf{r}, \sigma^2 \mathbf{I}) d\mathbf{r} = \int Z_2(\mathbf{Ar}) G(\mathbf{Ar}, \mathbf{R\Sigma R}^T) d(\mathbf{Ar}) \tag{1.33}$$

where the integrals are taken from $-\infty$ to ∞. $\mathbf{R}$ is a rotation matrix and $\mathbf{\Sigma}$ a diagonal matrix with entries $(s_1\sigma)^2, (s_2\sigma)^2 ... (s_n\sigma)^2$ $(s_i \geq 0)$ and $\mathbf{R\Sigma R}^T = \sigma^2 \mathbf{AA}^T$ (this follows from the fact that $\mathbf{AA}^T$ is a symmetric, positive semi-definite matrix).

Intuitively, equation (1.33) expresses the notion that the Gaussian weighted average brightnesses must be equal, provided the Gaussian is affine-transformed in the same manner as the function. The problem of recovering the affine parameters has been reduced to finding the deformation of a known function, the Gaussian, rather than the unknown brightness functions.

The level contours of the generalized Gaussian are ellipses rather than circles. The tilt of the ellipse is given by the rotation matrix while its eccentricity is given by the matrix $\mathbf{\Sigma}$, which is actually a function of the scales along each dimension. The equation clearly shows that to recover affine transforms by filtering, one must deform the filter appropriately; a point ignored in previous work [1, 4, 17]. The equation is local because the Gaussians rapidly decay.

The integral may be interpreted as the result of convolving the function with a Gaussian at the origin. It may also be interpreted as the result of a filtering operation with a Gaussian. To emphasize these similarities, it may be written as

$$Z_1 * G(\mathbf{r}, \sigma^2 \mathbf{I}) = Z_2 * G(\mathbf{r}_1, \mathbf{R\Sigma R}^T) \tag{1.34}$$

where $\mathbf{r}_1 = \mathbf{Ar}$.

Similar equations may be written using derivative of Gaussian filters (for details see [26, 27]).

1.3.2 Solution for the Case of Similarity Transforms

To solve equation(1.33) requires finding a Gaussian of the appropriate scale $s\sigma$ given σ. A brute force search through the space of scale changes is not desirable. Instead a more elegant solution is to linearize the Gaussians with respect to σ. This gives an equation linear in the unknown α

$$Z_1 * G(., (\sigma)^2) \quad = \quad Z_2 * G(., (s\sigma)^2)$$

$$\begin{aligned} &\approx Z_2 * G(.,\sigma^2) + \alpha\sigma Z_2 * \frac{\partial G(.,\sigma^2)}{\partial\sigma} \\ &= Z_2 * G(.,\sigma^2) + \alpha\sigma^2\nabla^2 Z_2 * G(.,\sigma^2) \end{aligned} \tag{1.35}$$

where $s = 1 + \alpha$. The last equality follows from the diffusion equation $\frac{\partial G}{\partial\sigma} = \sigma\nabla^2 G$.

Equation (1.35) is not very stable if solved at a single scale. By using Gaussians of several different scales σ_i the following linear least squares problem is obtained:

$$\mathbf{\Sigma}_i \| Z_1 * G(.,\sigma_i^2) - Z_2 * G(.,\sigma_i^2) + \alpha\sigma_i^2 Z_2 * \nabla^2 G(.,\sigma_i^2)\|^2 \tag{1.36}$$

and solved using Singular Value Decomposition (SVD).

It is not necessary to use every possible scale for σ_i. It turns out that the information at closely spaced scales is highly correlated and it usually suffices to use σ_i spaced apart by half an octave (a factor of about 1.4). For example, a possible set of scales would be (1.25,1.7677,2.5,3.5355,5.0).

1.3.3 Choosing a Different Operating Point:

For large scale changes (say scale change ≥ 1.2) the recovered scale tends to be poor. This is because the Taylor series approximation is good only for small values of α. The advantage of linearizing the Gaussian equations with respect to σ is that the linearization point can be shifted i.e. the right-hand side of (1.33) may be linearized with respect to a σ different from the one on the left-hand side be linearized around σ_j to give the following equation

$$Z_1 * G(.,\sigma_i^2) \approx Z_2 * G(.,\sigma_j^2) + \alpha'\sigma_j^2 Z_2 * \nabla^2 G(.,\sigma_j^2) \tag{1.37}$$

where $s = \sigma_j/\sigma_i(1+\alpha')$. The strategy therefore is to pick different values of σ_j and solve (1.37) (or actually an over-constrained version of it). Each of these σ_j will result in a value of α'. The correct value of α' is that which is most consistent with the equations. By choosing the σ_j appropriately, it can be ensured that no new convolutions are required.

In principle, arbitrary scale changes can be recovered using this technique. In practice, only a range of scales need to be recovered and, therefore, a small set of operating points will suffice.

1.3.4 Finding Image Translation:

Image translation, i.e. optic flow can be recovered in the following manner. Let Z_1 and Z_2 be similarity transformed versions of each other (i.e. they differ by a scale change, a rotation and a translation). Assume that an estimate of the translation $\mathbf{t}_0$ is available. Linearizing with respect to $\mathbf{r}$ and σ gives

$$Z_1(\mathbf{r}+\mathbf{t}_0) * G(\mathbf{r},\sigma^2) - \delta\mathbf{t}^T Z_1(\mathbf{r}+\mathbf{t}_0) * G(\mathbf{r},\sigma^2) \approx Z_2 * G(.,\sigma^2) + \alpha\sigma^2 Z_2 * \nabla^2 G(.,\sigma^2) \tag{1.38}$$

which is again linear in both the scale and the residual translation $\delta\mathbf{t}$. As before an over-constrained version of this equation using multiple scales is obtained and solved for the unknown parameters. Large scales are handled as before.

$\mathbf{t}_0$ is obtained either by a local search or from a coarser level in a pyramid scheme, while $\delta\mathbf{t}$ is estimated from the equation.

1.3.5 Solving for the General Affine

There are two factors which need to be taken into account in the general case. First note that in the similarity case all the filtering was done at one point (the origin). The results may be further improved by filtering at many points rather than just one point. However, the rotation invariance will then be lost. In the general affine case, because of the larger number of parameters that have to be recovered, the filtering must be done at many points. The deformation must also be accounted for and this can be done by linearizing the generalized Gaussian with respect to the affine parameters.

Filtering at a point $\mathbf{l}_i$ modifies the generalized Gaussian equation 1.33 as follows: Given a point with coordinates $\mathbf{l}_i$,

$$\int Z_1(\mathbf{r})G(\mathbf{r}-\mathbf{l_i},\sigma^2\mathbf{I})d\mathbf{r} = \int Z_2(\mathbf{Ar})G(\mathbf{A}(\mathbf{r}-\mathbf{l_i})),\mathbf{R\Sigma R}^T)d(\mathbf{Ar}) \tag{1.39}$$

Thus if the image is filtered at point $\mathbf{l}_i$ in the first image patch, it must be filtered at point $\mathbf{Al}_i$ in the second image patch.

$$\begin{aligned} Z_1 * G(\mathbf{r}-\mathbf{l}_i,\sigma) &\approx Z_2 * G(\mathbf{r}_1-\mathbf{l}_i,\sigma) - (\mathbf{Bl}_i)^T Z_2 * G'(\mathbf{r}_1-\mathbf{l}_i,\sigma) \\ &+ \sigma^2[b_{11}Z_2 * G_{xx}(\mathbf{r}_1-\mathbf{l}_i,\sigma) + b_{22}Z_2 * G_{yy}(\mathbf{r}_1-\mathbf{l}_i,\sigma) + (b_{12}+b_{21})Z_2 * G_{xy}(\mathbf{r}_1-\mathbf{l}_i,\sigma)] \end{aligned} \tag{1.40}$$

where the b_{ij} are elements of $\mathbf{B} = \mathbf{A} - \mathbf{I}$ and $\mathbf{I}$ is the identity matrix. Note that this is linear in the affine parameters b_{ij}. A number of methods incorporate the idea of filtering at many points [1, 4, 40]. However, none of these compensate for the deformation terms (in essence the difference between the traditional linearization methods and the technique presented here are the additional second derivative terms).

Translation may be incorporated by noticing that the effect of translation is similar to that of $\mathbf{l}_i$. Thus, with translation included the above equation may be rewritten as:

$$\begin{aligned} Z_1 * G(\mathbf{r}-\mathbf{l}_i,\sigma) &\approx Z_2 * G(\mathbf{r}_1-\mathbf{l}_i,\sigma) - (\mathbf{Bl}_i)^T Z_2 * G'(\mathbf{r}_1-\mathbf{l}_i,\sigma) \\ &+ \sigma^2[b_{11}Z_2 * G_{xx}(\mathbf{r}_1-\mathbf{l}_i,\sigma) + b_{22}Z_2 * G_{yy}(\mathbf{r}_1-\mathbf{l}_i,\sigma) + (b_{12}+b_{21})Z_2 * G_{xy}(\mathbf{r}_1-\mathbf{l}_i,\sigma)] \\ &- \mathbf{t}^T Z_2 * G'(\mathbf{r}_1-\mathbf{l}_i,\sigma) \end{aligned} \tag{1.41}$$

The equation may be turned into an over-constrained linear system by choosing a number of scales σ_i and a number of points $\mathbf{l}_i$. 2 or 3 scales are chosen as before. The points $\mathbf{l}_i$ are picked as follows: either every

point in the region is used or a regularly spaced subset of the points are used. The resulting over-constrained system may be solved using least mean squares and minimizing with respect to the affine parameters. One way of writing the solution to this least mean squares system is:

$$\mathbf{b} = M^{-1}\mathbf{z} \tag{1.42}$$

where $\mathbf{b} = [a_{11} - 1, a_{12}, a_{21}, a_{22} - 1, t_x, t_y]$ are the required affine parameters, and the i^{th} row of the n by 6 matrix M is given by

$$[\sigma^2 Z_2 * G_{xx} - x_i Z_2 * G_x, \sigma^2 Z_2 * G_{xy} - y_i Z_2 * G_x, \sigma^2 Z_2 * G_{xy} - x_i Z_2 * G_y, \sigma^2 Z_2 * G_{yy} - y_i Z_2 * G_y, -Z_2 * G_x, -Z_2 * G_y] \tag{1.43}$$

where the Gaussian derivatives are taken at point $\mathbf{r}_1 - \mathbf{l}_i$. The i^{th} element of the vector $\mathbf{z}$ is equal to $Z_1 * G(\mathbf{r} - \mathbf{l}_1, \sigma) - Z_2 * G(\mathbf{r}_1 - \mathbf{l}_1, \sigma)$

The solution is done iteratively. At each step, the affine transformation is solved for. The image is then warped according to the affine and the residual affine solved for. Convergence is very rapid. A good solution is obtained using two scales $1.25, 1.77$ spaced half an octave apart and with a window of size 13 by 13 (that is points from a region of size 13 by 13 are selected) [27]. The technique allows fairly large affine transforms to be recovered (scaling of as much as 40%).

The technique has some limitations. For large translations, a good initial estimate of the translation is required. This may be obtained in a number of ways. A coarse to fine technique may be used to estimate the translation. Alternatively, the method used to find similar points in the next section may be used to provide an estimate of the translation.

1.4 IMAGE RETRIEVAL

In this section Gaussian filtered representations of images are applied to the task of retrieval by image appearance. The paradigm used for retrieval is that a collection of images are represented using feature vectors constructed from Gaussian derivatives. During run-time, a user presents an example image or parts thereof as a query to the system. The query's feature vectors are compared with those in the database, and the images in the database are ranked and displayed to the user.

There are several objectives that govern the design of a retrieval system. Primary among those is speed and the ability to find visually similar objects within a reasonable space of deformations. There is a third objective that can provide considerable flexibility to a user. This is the ability of a system to query parts of images if required. This is because a user might be interested in a part of an image such as a face in a crowd or a whole image such as a trademark. The interesting aspect of the algorithms presented here is that both these types of retrieval can be achieved without a system automatically trying to compute salient features or regions, which can be extremely challenging. All the system does in either case is compare signals (feature

vectors). When the user has a notion of what is important in an image it is exploited to find other vectors similar to it.

However, the desired flexibility imposes different constraints on the retrieval algorithms. Finding parts of images requires measurement of local similarity and the representation must be local. Therefore, individual feature vectors might be used. On the other hand finding whole images implies global similarity and distributions of features can be used.

In previous work [35] derivative feature vectors in conjunction with the scaling theorems were used for retrieval by appearance. Database images were filtered with the Gaussian derivatives up to the second order, at several scales. A query image (or parts of it) was also filtered but at a single scale. Then using the scaling theorems in Section 1.2.3 the query feature vectors were correlated across scales with the each database image's feature vectors. The results indicated that visually similar objects can be retrieved within about 30^o of rotations. Similar results were shown by [32] in object recognition experiments.

However, correlation is slow. Further correlation does not allow the feature vectors to be indexed. Thus, one cannot expect to develop a system of reasonable speed even for moderately sized databases using this method. Here, we present two progressively faster and indexable methods to retrieve images. The first method may be used to find parts of images and the second for finding whole images. Finding parts of images and requires similarities of local image features to be computed. This implies an explicit representation of local features. On the other hand whole image matching implies global similarity. Global similarity may be inferred using distributions of local features rather than the features themselves. In the next two sections the local and global similarity retrieval algorithms are elaborated.

1.4.1 Local Similarity Retrieval

Local similarity retrieval is carried out as follows. Database images are uniformly sampled. At each sampled location, the multi-scale invariant feature vector Δ_σ defined in Section 1.2.3 is computed at three different scales. Then, vectors computed for all the images in the database are indexed using a binary tree structure. During run-time, the user picks an example image and "designs" a query. Since an image is described spatially (uniform sampling), parts of images or (imaged objects) can be selected. For example, consider Figure 1.3(a). Here the user wants to retrieve white wheeled cars and therefore, appropriately selects the white wheel. The feature vectors that lie within this region are submitted to the system. Database images with feature vectors that match the set of query vectors both in feature space (L2 norm of the vector) and coordinate space (matched image locations spatially consistent with query points) are returned as retrievals.

The approach for local similarity retrieval is divided into two parts. During the off-line phase, images are sampled and multi-scale derivatives are computed. These are transformed into rotational invariants and indexed. During the on-line phase the user designs a query by selecting regions within an image. Feature vectors within the query are matched with those in the database both in feature space and in coordinate

(a) Car Query

(b) Ranked Retrieval

Fig. 1.3 A query and its retrieval

space. The off-line and on-line phases are discussed next.

1.4.1.1 Off-line Operations: Invariant Vectors and Indexing

Computing Features: A multi-scale invariant vector is computed at sampled locations within the image. The vector, $\mathbf{\Delta}_\sigma = \langle d_1, \ldots d_4 \rangle_\sigma$ (see Section 1.2.3) is computed at three different scales. The element d_0 is not used since it is sensitive to gray-level shifts. The resulting multi-scale invariant vector has at most twelve elements. Computationally, each image in the database is filtered with the first five partial derivatives of the Gaussian (i.e. to order 2) at three different scales at uniformly sampled locations. Then the multi-scale invariant vector $\mathbf{D} = \langle \mathbf{\Delta}_{\sigma_1}, \mathbf{\Delta}_{\sigma_2}, \mathbf{\Delta}_{\sigma_3} \rangle$ is computed at those locations.

Indexing: A location across the entire database may be identified by the *generalized coordinates*, defined as, $c = (i, x, y)$ where i is the image number and (x, y) a coordinate within this image. The computation described above generates an association between generalized coordinates and invariant vectors. This association may be viewed as a table $M : (i, x, y, D)$ with $3 + k$ columns (k is the number of fields in an invariant vector) and the number of rows R, equal to the total number of locations (across all images) where invariant vectors are computed. To retrieve images, a 'find by value' functionality is needed, with which, a query invariant vector is found within M and the corresponding generalized coordinate is returned. Inverted files (or tables) based on each field of the invariant vector are first generated for M. To index the database by fields of the invariant vector, the table M is split into k smaller tables $M_1^{'} \ldots M_k^{'}$, one for each of the k fields of the invariant vector. Each of the smaller tables $M_p^{'}, p = 1 \cdots k$ contains the four columns $(D(p), i, x, y)$. At this stage any given row across all the smaller tables contains the same generalized coordinate entries as in M. Then, each $M_p^{'}$ is sorted and a binary tree is used to represent the sorted keys. As a result, the entire database is indexed. A given invariant value can, therefore, be located in $\log(R)$ time (R = number of rows).

1.4.1.2 On-line Operation Run-time computation begins with the user selecting regions in an example image. At sampled locations within these regions, invariant vectors are computed and submitted as a query. The success of a retrieval in part depends on well designed queries. More importantly, letting the user design queries eliminates the need for automatically detecting the salient portions of an object, and the retrieval may be customized so as to remove unwanted portions of the image. Based on the feedback provided by the results of a query, the user can quickly adapt and modify the query to improve performance.

The search for matching images is performed in two stages. In the first stage each query invariant is supplied to the 'find-by-value' algorithm and a list of matching generalized coordinates is obtained. In the second stage a spatial check is performed on a per image basis, so as to verify that the matched locations in an image are in spatial coherence with the corresponding query points. In this section the 'find-by-value' and spatial checking components are discussed.

Finding by invariant value: The multi-scale invariant vectors at sampled locations within regions of a query image may be treated as a list. The n^{th} element in this list contains the information $Q_n = (D_n, x_n, y_n)$, that is, the invariant vector and the corresponding coordinates. In order to find by invariant value, for any query entry Q_n, the database must contain vectors that are within a threshold $t = (t_1 \ldots t_k) > 0$. The coordinates of these matching vectors are then returned. This may be represented as follows. Let p be any invariant vector stored in the database. Then p matches the query invariant entry D_n only if $D_n - t < p < D_n + t$. To implement the comparison operation two searches can be performed on each field. The first is a search for the lower bound, that is the smallest entry larger than $D_n(j) - t(j)$ and then a search for the upper-bound i.e. the largest entry smaller than $D_n(j) + t(j)$. The block of entries between these two bounds are those that match the field j. In the inverted file the generalized coordinates are stored along with the individual field values and the block of matching generalized coordinates are copied from disk. Then an

intersection of all the returned block of generalized coordinates is performed. The generalized coordinates common to all the k fields are the ones that match query entry Q_n. The find by value routine is executed for each Q_n and as a result each query entry is associated with a list of generalized coordinates that it matches to.

Spatial-fitting: The association between a query entry Q_n and the list of f generalized coordinates that match it by value may be written as

$$\begin{aligned} A_n &= \langle x_n, y_n, c_{n_1}, c_{n_2} \ldots c_{n_f} \rangle \\ &= \langle x_n, y_n, (i_{n_1}, x_{n_1}, y_{n_2}) \ldots (i_{n_f}, x_{n_f}, y_{n_f}) \rangle \end{aligned}$$

Here x_n, y_n are the coordinates of the query entry Q_n and $c_{n_1} \ldots c_{n_f}$ are the f matching generalized coordinates. The notation c_{n_f} implies that the generalized coordinate c matches n and is the f^{th} entry in the list. Once these associations are available, a spatial fit on a per image basis can be performed. Any image u that contains two points (locations) which match some query entry m and n respectively are coherent with the query entries m and n only if the distance between these two points is the same as the distance between the query entries that they match. Using this as a basis, a binary fitness measure may be defined as

$$\mathcal{F}_{m,n}(u) = \begin{cases} 1 & \text{if } \exists j \exists k \mid \left| \delta_{m,n} - \delta_{c_{m_j}, c_{n_k}} \right| \leq T, i_{m_j} = i_{n_k} = u, m \neq n \\ 0 & \text{otherwise} \end{cases}$$

where $\delta_{m,n}$ is the Euclidean distance between the query points m and n, and $\delta_{c_{m_j}, c_{n_k}}$ is the Euclidean distance between the generalized coordinates c_{m_j} and c_{n_k} That is, if the distance between two matched points in an image is close to the distance between the query points that they are associated with, then these points are spatially coherent (with the query). Using this fitness measure a match score for each image can be determined. This match score is simply the maximum number of points that together are spatially coherent (with the query). Define the match score by: $score(u) \equiv \overset{max}{m} \sum_{n=1}^{f} \mathcal{F}(u)_{m,n}$. The computation of $score(u)$ is at worst quadratic in the total number of query points. The array of scores for all images is sorted and the images are displayed in the order of their score. T used in $\mathcal{F}$ is a threshold and is typically 25% of $\delta_{m,n}$. Note that this measure not only will admit points that are rotated but will also tolerate other deformations as permitted by the threshold. It is placed to reflect the rationale that similar images will have similar responses but not necessarily under a rigid deformation of the query points.

1.4.1.3 Experiments The database used for the local similarity retrieval has digitized images of cars, steam locomotives, diesel locomotives, apes, faces, people embedded in different background(s) and a small number of other miscellaneous objects such as houses. 1561 images were obtained from the Internet and the Corel photo-cd collection to construct this database. These photographs were taken with several different cameras of unknown parameters, and under varying uncontrolled lighting and viewing geometry. Prior to

(a) Ape Query (b) Ranked Retrieval

Fig. 1.4 An ape query and its retrieval

describing the experiments, it is important to clarify what a correct retrieval means. A retrieval system is expected to answer questions such as 'find all cars similar in view and shape to this car' or 'find all faces similar in appearance to this one'. In the examples presented here the following method of evaluation is applied. First, the objective of the query is stated and then retrieval instances are gauged against the stated objective. In general, objectives of the form 'extract images similar in appearance to the query' will be posed to the retrieval algorithm.

A measure of the performance of the retrieval engine may be obtained by examining the recall/precision table for several queries. Briefly, recall is the proportion of the relevant material actually retrieved and precision is the proportion of retrieved material that is relevant [44]. Consider as an example the query described in Figure 1.3(a). Here the user wishes to retrieve 'white wheeled cars' similar to the one outlined and submits the query. The top 25 results ranked in text book fashion are shown in Figure 1.3(b). Note that although there are several valid matches as far as the algorithm is concerned (for example image 12 is a train with a wheel), they are not considered valid retrievals as stated by the user and are not used in measuring the recall/precision. This is inherently a conservative estimate of the performance of the system. The average precision (over recall intervals of 10^4) is 48.6%.

One of the important parameters in constructing indices is the sample rate. Recall that indices are generated

[4]The value $n(= 10)$ is simply the retrievals up to recall n.

Given(User Input)	Find	Precision (5)	Precision (3)
Face	All Faces	74.7%	61.5%
Face	Same Person's Face,	61.7%	75.5%
Monkey's coat. See Figure 1.4(a)	Dark Textured Apes. See Figure 1.4(b)	57.5%	57%
Both wheels	White Wheeled Cars	57.0%	63.7%
Coca Logo	All Coca Cola Logos	49.3%	74.9%
wheel see Figure 1.3(a)	White Wheeled Cars, see Figure 1.3(b)	48.6%(see text)	54.4%
Patas Monkey Face	All Visible Patas Monkey Faces	44.5%	47.1%

Table 1.1 Queries submitted to the system and expected retrieval

Recall	0	10	20	30	40	50	60	70	80	90	100
Precision(5) %	100	95.8	90.3	80.1	67.3	48.9	39.9	34.2	31.1	18.2	12.4
Precision(3) %	100	100	90.4	80.9	75.7	55.9	49.4	47.6	40.6	20.7	17.1

average(5)	56.2%
average(3)	61.7%

Table 1.2 Precision at standard recall points for seven Queries

by computing multi-scale invariant feature vectors at uniformly sampled locations within the image. The performance of the system is evaluated under sample rates of three pixels and five pixels respectively. The case where every pixel is used could not be implemented due to prohibitive disk requirements and lack of resources to do so. Six other queries that were also submitted are depicted in table 1.1. The recall/precision table over all seven queries is in Table 1.4.1.3. The third column of the Table shows the average precision for each query with a database sampling of 5 pixels while the fourth column displays the average precision for a sampling of 3 pixels. The average precision and precision at recall intervals of 10, over all the queries for both samplings is shown in Table 1.4.1.3. This compares well with text retrieval where some of the best systems have an average precision of 50%(based on personal communication with Bruce Croft). The average precision over the same seven queries with both three and five pixel sampling is 56.2% for the five pixel case and 61.7% in the three pixel case. However, while the increase in sampling improves the precision it results in an increased storage requirement.

Unsatisfactory retrieval occurs for several reasons. First it is possible that the query is poorly designed. In this case the user can design a new query and re-submit. A second source of error is in matching itself. It is possible that locally the intensity surface may have a very close value. Many of these 'false matches' are eliminated in the spatial checking phase. Errors may also occur in the spatial checking phase because it admits much more than a rotational transformation of points with respect to the query configuration. Overall the performance to date has been very satisfactory and we believe that by experimentally evaluating

each phase the system can be further improved. The time it takes to retrieve images is dependent linearly on the number of query points. On a Pentium Pro-200 Mhz Linux machine, typical queries execute in between one and six minutes. However, the primary limitations of the local matching technique are that it is relatively slow, and requires considerable disk space. Further, as presented the system cannot search for images in their entirety. That is, it does not address global similarity.

1.4.2 Global Similarity Retrieval

The same Gaussian derivative model may be used to efficiently retrieve by global similarity of appearance. Since the task is to find similarity of whole images significant improvements in space as well as speed may be achieved by representing images using distributions of feature vectors as opposed to the vectors themselves. One of the simplest ways of representing a non-parametric distribution is a histogram. Thus, a histogram of features may be used.

There are several features that may be exploited. Here the task is to robustly characterize the 3 dimensional intensity surface. A 3-dimensional surface is uniquely determined if the local curvatures everywhere are known. Thus, it is appropriate that one of the features be local curvature. The principal curvatures of the intensity surface are differential invariants. Further they are invariant to monotonic intensity variations and their ratios are in principle insensitive to scale variations of the entire image. However, spatial orientation information is lost when constructing histograms of curvature (or ratios thereof) alone. Therefore we augment the local curvature with local phase, and the representation uses histograms of local curvature and phase.

Three steps are involved in order to compute global similarity. First, local derivatives are computed at several scales. Second, derivative responses are combined to generate local features, namely, the principal curvatures and phase and, their histograms are generated. Third, the 1D curvature and phase histograms generated at several scales are matched. These steps are described next.

A. Computing local derivatives: Derivatives are computed stably using the formulation shown in Equation 1.3. The first and second order derivatives are computed at several scales by filtering the database images with Gaussian derivatives.

B. Feature Histograms: The normal and tangential curvatures of a 3-D surface (X,Y,Intensity) are defined by [9]:

$$N(\mathbf{p},\sigma) = \left[\frac{I_x^2 I_{yy} + I_y^2 I_{xx} - 2 I_x I_y I_{xy}}{\left(I_x^2 + I_y^2\right)^{\frac{3}{2}}}\right](\mathbf{p},\sigma)$$

$$T(\mathbf{p},\sigma) = \left[\frac{\left(I_x^2 - I_y^2\right) I_{xy} + \left(I_{xx} - I_{yy}\right) I_x I_y}{\left(I_x^2 + I_y^2\right)^{\frac{3}{2}}}\right](\mathbf{p},\sigma)$$

where $I_x(\mathbf{p},\sigma)$ and $I_y(\mathbf{p},\sigma)$ are the local derivatives of image I around point $\mathbf{p}$ using Gaussian derivative at scale σ. Similarly $I_{xx}(\cdot,\cdot)$, $I_{xy}(\cdot,\cdot)$, and $I_{yy}(\cdot,\cdot)$ are the corresponding second derivatives. The normal curvature N and tangential curvature T are then combined [20] to generate a shape index as follows:

$$C(\mathbf{p},\sigma) = atan\left[\frac{N+T}{N-T}\right](\mathbf{p},\sigma)$$

The index value C is $\frac{\pi}{2}$ when $N = T$ and is undefined when either N and T are both zero, and is therefore not computed. This is interesting because very flat portions of an image (or ones with constant ramp) are eliminated. For example in Figure 1.6(middle-row), the background in most of these face images does not contribute to the curvature histogram. The curvature index or shape index is rescaled and shifted to the range $[0,1]$ as is done in [7]. A histogram is then computed of the valid index values over an entire image.

The second feature used is phase. The phase is simply defined as $P(\mathbf{p},\sigma) = atan2(I_y(\mathbf{p},\sigma), I_x(\mathbf{p},\sigma))$. Note that P is defined only at those locations where C is defined and ignored elsewhere. As with the curvature index P is rescaled and shifted to lie between the interval $[0,1]$.

Although the curvature and phase histograms are in principle insensitive to variations in scale, in early experiments we found that computing histograms at multiple scales dramatically improved the results. An explanation for this is that at different scales different local structures are observed and, therefore, multi-scale histograms are a more robust representation. Consequently, a feature vector is defined for an image I as the vector $V_i = \langle H_c(\sigma_1)\ldots H_c(\sigma_n), H_p(\sigma_1)\ldots H_p(\sigma_n)\rangle$ where H_p and H_c are the curvature and phase histograms respectively. We found that using 5 scales gives good results and the scales used were from 1 to 4 in steps of half an octave.

C. Matching feature histograms: Two feature vectors are compared using normalized cross-covariance defined as

$$d_{ij} = \frac{V_i^{(m)} \cdot V_j^{(m)}}{\left|\left|V_i^{(m)}\right|\right| \left|\left|V_j^{(m)}\right|\right|}$$

where $V_i^{(m)} = V_i - mean(V_i)$.

Retrieval is carried out as follows. A query image is selected and the query histogram vector V_j is correlated with the database histogram vectors V_i using the above formula. Then the images are ranked by their correlation score and displayed to the user. In this implementation, and for evaluation purposes, the ranks are computed in advance, since every query image is also a database image.

1.4.2.1 Experiments The curvature-phase method is tested using two databases. The first is a trademark database of 2048 images obtained from the US Patent and Trademark Office (PTO). The images obtained from the PTO are large, binary and are converted to gray-level and reduced for the experiments.

The second database is the collection of 1561 assorted gray-level images used for the local similarity case.

In the following experiments an image is selected and submitted as a query. The objective of this query is stated and the relevant images are decided in advance. Then the retrieval instances are gauged against the stated objective. In general, objectives of the form 'extract images similar in appearance to the query' will be posed to the retrieval algorithm. The measure of the performance of the retrieval engine is obtained by examining the recall/precision table for several queries.

Table 1.3 Precision at standard recall points for six queries

Recall	0	10	20	30	40	50	60	70	80	90	100
Precision(trademark) %	100	93.2	93.2	85.2	76.3	74.5	59.5	45.5	27.2	9.0	9.0
Precision(assorted) %	100	92.6	90.0	88.3	87.0	86.8	83.8	65.9	21.3	12.0	1.4

average(trademark)	61.1%
average(assorted)	66.3%

Queries were submitted to each of the collections (trademark and assorted image collection) separately for the purpose of computing recall/precision. The judgment of relevance is qualitative. For each query in both databases the relevant images were decided in advance. These were restricted to 48. The top 48 ranks were then examined to check the proportion of retrieved images that were relevant. All images not retrieved within 48 were assigned a rank equal to the size of the database. That is, they are not considered retrieved. These ranks were used to interpolate and extrapolate precision at all recall points. In the case of assorted images relevance is easier to determine and more similar for different users. However, in the trademark case it may be quite difficult to determine relevance and, therefore, the recall-precision may be subject to some error. The recall/precision results are summarized in Table 1.3 and both databases are individually discussed below.

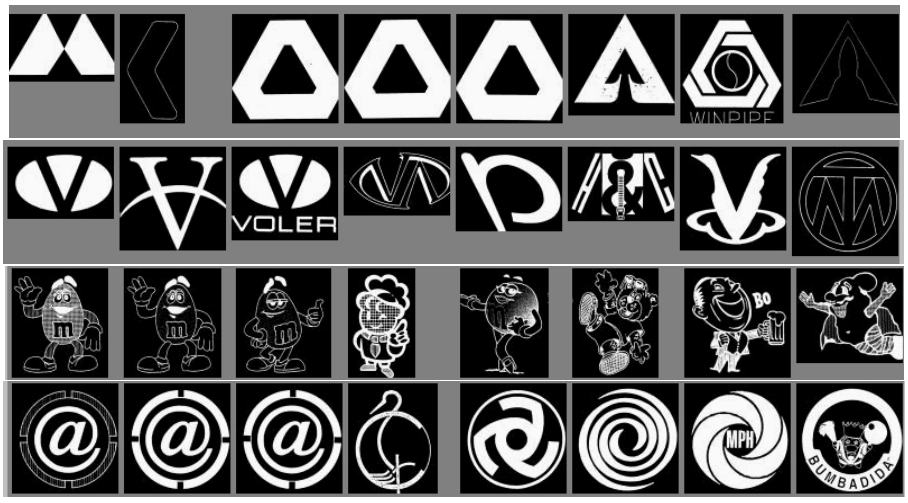

Fig. 1.5 Trademark retrieval using curvature and phase

Figure 1.5 shows the performance of the algorithm on the trademark images. Each strip depicts the top 8 retrievals, given the leftmost image as the query. Most of the shapes have roughly the same structure as the query. Six queries were submitted for the purpose of computing recall-precision depicted in Table 1.3.

Fig. 1.6 Image retrieval using curvature and phase

Experiments were also carried out with assorted gray level images. Six queries submitted for recall-precision are shown in Figure 1.6. The left most image in each row is the query and is also the first retrieved. The rest from-left to right are seven retrievals depicted in rank order. Note that flat portions of the background are never considered because the principal curvatures are very close to zero and, therefore, do not contribute to the final score. Thus, for example, the flat background in Figure 1.6(second row) is not used. Notice that visually similar images are retrieved even when there is some change in the background (row 1). This is because the dominant object contributes most to the histograms. In using a single scale poorer results are achieved and background affects the results more significantly.

The results of these examples are discussed below, with the average precision over all recall points depicted in parentheses:

1. Find similar cars(65%). Pictures of cars viewed from similar orientations appear in the top ranks because of the contribution of the phase histogram. This result also shows that some background variation can be tolerated. The eighth retrieval although a car is a mismatch and is not considered a valid retrieval for the purpose of computing recall/precision.

2. Find same face(87.4%) and find similar faces: In the face query the objective is to find the same face. In experiments with a University of Bern face database of 300 faces with 10 relevant faces each, the average precision over all recall points for all 300 queries was 78%. It should be noted that the system

presented here works well for faces with the same representation and parameters used for all the other databases. There is no specific "tuning" or learning involved to retrieve faces. The query "find similar faces" resulted in a 100% precision at 48 ranks because there are far more faces than 48. Therefore it was not used in the final precision computation.

3. Find dark textured apes (64.2%). The ape query results in several other light textured apes and country scenes with similar texture. Although these are not mis-matches they are not consistent with the intent of the query which is to find dark textured apes.

4. Find other Patas monkeys. (47.1%) Here there are 16 Patas monkeys in all and 9 within a small view variation. However, here the whole image is being matched so the number of relevant Patas monkeys is 16. The precision is low because the method cannot distinguish between light and dark textures, leading to irrelevant images. Note, that it finds other apes, dark textured ones, but those are deemed irrelevant with respect to the query.

5. Given a wall with a Coca Cola logo find other Coca Cola images (63.8%). This query clearly depicts the limitation of global matching. Although all three database images that had a certain texture of the wall (also had Coca Cola logos) were retrieved (100% precision), two other very dissimilar images with Coca Cola logos were not.

6. Scenes with Bill Clinton (72.8%). The retrieval in this case results in several mismatches. However, three of the four are retrieved in succession at the top and the scenes appear visually similar.

While the queries presented here are not "optimal" with respect to the design constraints of global similarity retrieval, they are however, realistic queries that can be posed to the system. Mismatches can and do occur. The first is the case where the global appearance is very different. The Coca Cola retrieval is a good example of this. Second, mismatches may occur at the algorithmic level. Histograms coarsely represent spatial information and therefore will admit images with non-trivial deformations. The recall/precision presented here compares well with text retrieval. The time per retrieval is of the order of milli-seconds. In on going work we are experimenting with a database of 63000 images and the amount of time taken to retrieve is still less than a second. The space required is also a small fraction of the database. These are the primary advantages of global similarity retrieval. That is, to provide a low storage, high speed retrieval with good recall/precision.

1.4.3 Suggested Reading

Image Retrieval has attracted the attention of several researchers in recent years and several retrieval systems have been proposed. The earliest general image retrieval systems were designed by [8, 31]. In [8] the shape queries require prior manual segmentation of the database which is undesirable and not practical for most applications.

Several authors have tried to characterize the appearance of an object via a description of the intensity surface. In the context of object recognition [30] represent the appearance of an object using a parametric eigen space description. This space is constructed by treating the image as a fixed length vector, and then computing the principal components across the entire database. The images, therefore, have to be size and intensity normalized, segmented and trained. Similarly, using principal component representations described in [18] face recognition is performed in [43]. In [41] the traditional eigen representation is augmented by using most discriminant features and is applied to image retrieval. The authors apply eigen representation to retrieval of several classes of objects. The issue, however, is that these classes are manually determined and training must be performed on each. The approach presented in this chapter is different from all the above because eigen decompositions are not used at all to characterize appearance. Further the method presented uses no learning, does not depend on constant sized images and deals with embedded backgrounds(local similarity) and heterogeneous collections of images using local representations of appearance.

It should be noted that the method presented by [38] is similar to the local similarity algorithm. However, there are some differences. Schmid and Mohr's algorithm does not allow the user to select query regions and relies on corners as features. The algorithm used to spatially compare the query vectors with those of a database image (spatial consistency) is also different. The motivation behind the algorithm presented here is that algorithms such as feature detection or segmentation, which are used to determine salient parts of an image cannot be determined *a priori* in a general retrieval system, because it is not possible to define in advance the needs of a user. Thus instead of establishing an *a priori* bias towards any feature, images are uniformly sampled. The selection of salient portions of an image is obtained in the form of the user defined query. In addition we find that using low two orders rather than the three orders gives better results in terms of locating similar features.

With regard to the global retrieval algorithm, Schiele and Crowley [2] used a technique for recognizing objects using grey-level images based on histograms. Their technique used the outputs of Gaussian derivatives as local features. Several feature combinations were evaluated. In each case, a multi-dimensional histogram of these local features is then computed. Two images are considered to be of the same object if they had similar histograms. The difference between this approach and the one presented by Schiele and Crowley is that here we use 1D histograms (as opposed to multi-dimensional) and further use the principal curvatures as the primary feature (which they do not use).

Texture based image retrieval is also related to the appearance based work presented in this chapter. Using Wold modeling, in [23] the authors try to classify the entire Brodatz texture and in [11] attempt to classify scenes, such as city and country. Of particular interest is work by [24] who use Gabor filters to retrieve texture similar images, without user interaction to determine region saliency.

Acknowledgement

The authors wish to thank Bruce Croft, and the Center for Intelligent Information Retrieval for continued support of this work. This work is made possible due to efforts by Adam Jenkins, David Hirvonen and Yasmina Chitti.

References

1. J.R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Proc. 2nd European Conference on Computer Vision*, pages 237–252, 1992.

2. bernt Schiele and James L. Crowley. Object recognition using multidimensional receptive field histograms. In Bernard Buxton and Roberto Cipolla, editors, *Computer Vision - ECCV '96*, volume 1 of *Lecture Notes in Computer Science*, Cambridge, U.K., April 1996. 4th European Conf. Computer Vision, Springer.

3. P. Burt and T. Adelson. The laplacian pyramid as a compact image code. *IEEE Trans. on Communications*, 1983.

4. M. Campani and A. Verri. Motion analysis from optical flow. *Computer Vision Graphics and Image Processing:Image Understanding*, 56(12):90–107, 1992.

5. R. Cipolla and A. Blake. Surface orientation and time to contact from image divergence and deformation. In *Proc. 2nd European Conference on Computer Vision*, pages 187–202, 1992.

6. James L. Crowley and Alice C. Parker. A representation for shape basedon peaks and ridges in the difference of low-pass transform. *IEEE Trans. on Pattern Anal. and Machine Intelligence*, 6:156–169, 1984.

7. Chitra Dorai and Anil Jain. Cosmos - a representation scheme for free form surfaces. In *Proc. 5th Intl. Conf. on Computer Vision*, pages 1024–1029, 1995.

8. Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Denis Lee, Dragutin Petkovix, Devid Steele, and Peter Yanker. Query by image and video content: The qbic system. *IEEE Computer Magazine*, 28(9):23–30, September 1995.

9. Ludvicus Maria Jozef Florack. *The Syntactic Structure of Scalar Images.* PhD thesis, University of Utrecht, 1993.

10. W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Trans. Patt. Anal. and Mach. Intelligence*, 13(9):891–906, 1991.

11. M. M. Gorkani and R. W. Picard. Texture orientation for sorting photos 'at a glance'. In *Proc. 12th Int. Conf. on Pattern Recognition*, pages A459–A464, October 1994.

12. Gosta Granlund and Hans Knutsson. *Signal Processing for Computer Vision.* Kluwer Academic Publishers, 1995.

13. S. Haykin. *Communication Systems.* Wiley, 1978.

14. E. C. Hildreth. *The Measurement of Visual Motion.* The MIT Press, Cambridge, MA, 1984.

15. R. Hummel and D. Lowe. Computational considerations in convolution and feature extraction in images. In J. C. Simon, editor, *From Pixels to Features*, pages 91–102. Elsevier Science Publications B. V. (North-Holland), 1989.

16. M. Irani, P. Anandan, and S. Hsu. Mosaic based representations of video sequences and their applications. In *Proc. 5th Intl. Conf. on Computer Vision*, pages 605–611, June 1995.

17. D. G. Jones and J. Malik. A computational framework for determining stereo correspondence from a set of linear spatial filters. In *Proc. 2nd European Conference on Computer Vision*, pages 395–410, 1992.

18. M Kirby and L Sirovich. Application of the karhunen-loeve procedure for the characterization of human faces. *IEEE Trans. Patt. Anal. and Mach. Intel.*, 12(1):103–108, January 1990.

19. J. J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–396, 1984.

20. J. J. Koenderink and A. J. Van Doorn. Surface shape and curvature scales. *Image and Vision Computing*, 10(8), 1992.

21. J. J Koenderink and A. J. van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics*, 55:367–375, 1987.

22. Tony Lindeberg. *Scale-Space Theroy in Computer Vision.* Kluwer Academic Publishers, 1994.

23. Fang Liu and Rosalind W Picard. Periodicity, directionality, and randomness: Wold features for image modeling and retrieval. *IEEE Trans. PAMI*, 18(7):722–733, July 1996.

24. W. Y. Ma and B. S. Manjunath. Texture-based pattern retrieval from image databases. *Multimedia Tools and Applications*, 2(1):35–51, January 1996.

25. R. Manmatha. Measuring the affine transform – I: Scale and rotation. In *Proc. Computer Vision and Pattern Recognition Conference*, pages 754–755, 1993.

26. R. Manmatha. A framework for recovering affine transforms using points, lines or image brightnesses. In *Proc. Computer Vision and Pattern Recognition Conference*, pages 141–146, 1994.

27. R. Manmatha. *Matching Affine-Distorted Images.* PhD thesis, University of Massachusetts at Amherst, 1997.

28. R. Manmatha and J. Oliensis. Measuring affine transform - I, scale and rotation. In *Proc. DARPA Image Understanding Workshop*, pages 449–458, Washington D.C., 1993.

29. D. Marr and E. Hildreth. Theory of edge detection. *Proc. Royal Society of London(B)*, B207, 1980.

30. S. K. Nayar, H. Murase, and S. A. Nene. Parametric appearance representation. In *Early Visual Learning.* Oxford University Press, February 1996.

31. A. Pentland, R. W. Picard, and S. Sclaroff. Photobook: Tools for content-based manipulation of databases. In *Proc. Storafe and Retrieval of Image and Video Databases II*, volume 2, pages 34–47. SPIE, 1994.

32. Rajesh Rao and Dana Ballard. Object indexing using an iconic sparse distributed memory. In *Proc. International Conference on Computer Vision*, pages 24–31. IEEE, 1995.

33. S. Ravela, B. Draper, J. lim, and R. Weiss. Adaptive tracking and model registration across distinct aspects. In *International Conference on Intelligent Robots and Systems*, volume 1, pages 174–180. IEEE, August 1995.

34. S. Ravela and R. Manmatha. Retrieving images by appearance. In *IEEE Intl. Conf. Comput. Vision.* IEEE, Jan. 1998.

35. S. Ravela, R. Manmatha, and E. M. Riseman. Image retrieval using scale-space matching. In *Computer Vision - ECCV '96*, pages 273–282, Cambridge, U.K., April 1996. 4th European Conf. Computer Vision, Springer.

36. H. S. Sawhney, S. Ayer, and M. Gorkani. Model-based 2d and 3d dominant motion estimation for mosaicing and video representation. In *Proc. 5th Intl. Conf. on Computer Vision*, pages 583–590, June 1995.

37. H. S. Sawhney and A. R. Hanson. Identification and 3D description of 'shallow' environmental structure in a sequence of images. In *Proc. Computer Vision and Pattern Recognition Conference*, pages 179–186, 1991.

38. Cordelia Schmid and Roger Mohr. Combining greyvalue invariants with local constraints for object recognition. In *Proc. Computer Vision and Pattern Recognition*. IEEE, June 1996.

39. J. Segman, J. Rubinstein, and Y.Y. Zeevi. The canonical coordinates method for pattern deformation: Theoretical and computational considerations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(12):1171–1183, 1992.

40. J. Shi and C. Tomasi. Good features to track. In *Proc. Computer Vision and Pattern Recognition Conference*, pages 593–600, 1994.

41. D. L. Swets and J. Weng. Using discriminant eigen features for retrieval. *IEEE Trans. Patt. Anal. and Mach. Intel.*, 18:831–836, August 1996.

42. Bart M. ter Har Romeny. *Geometry Driven Diffusion in Computer Vision*. Kluwer Academic Publishers, 1994.

43. M. Turk and A. Pentland. Eigen faces for recognition. *Jrnl. Cognitive Neuroscience*, 3:71–86, 1991.

44. C. J. van Rijsbergen. *Information Retrieval*. Butterworths, 1979.

45. P. Werkhoven and J. J. Koenderink. Extraction of motion parallax structure in the visual system I. *Biological Cybernetics*, 1990.

46. A. P. Witkin. Scale-space filtering. In *Proc. Intl. Joint Conf. Art. Intell.*, pages 1019–1023, 1983.

47. Richard A. Young. The Gaussian derivative model for spatial vision: I. Retinal mechanisms. *Spatial Vision*, 2(4):273–293, 1987.

Index