# Chapter 2

# Precedents

Vision is fundamentally a dynamic process. For example, if we have a question like: what does it mean, to see?, The most easy answer would be: to know what is where by looking [40]. Thus, vision is the process of discovering from images what is present in the world, and where it is. And this is possible because images of natural scenes, unlike random collections of pixels, are characterized by a high degree of statistical regularity. For instance, pixel values in a given neighborhood tend to be highly correlated owing to the morphological consistency of objects. Thus, a pixel-wise representation of objects obtained from a camera is highly redundant.

But vision has some problematic aspects of description in general. As example, structural descriptions which are considered the ultimate goal of vision has some problems such as:

1. Can we describe using a natural language all the things that can be seen in an image? In applying language to vision, it should exist an universal language in order to denote "things" in images. For example, do all the people agree about a painting? The painting can be seen as an image and people can represent "our" eyes. Thus, can we expect that our eyes agree about what they see? In view of this indeterminacy, one obviously can not expect a one-to-one correspondence between the image and any of its verbal descriptions. Up to this point, we can think that a picture is worth much more than a thousand of words. In this sense, we do not have enough names (nor sentences) for all the things that can be found in an image.

2. The first problem is usually solved creating a list of possible objects that can appear in images that is, to settle for seeing only certain things: those that match our concepts. In other words, we only will search for all these things that we know and expect that would appear in our images. Thus, we presuppose the existence of clearly delimited entities, which need to be detected and labeled. Assuming this kind of restriction, our vision strategy becomes poor since we are looking for "known" objects that are members of some a priori set.

3. When we form the description of an image in terms of a restricted set of objects we should solve some indeterminancies lurking behind the decision of which

object should a given pixel be attributed. Here is where we see the need of the technical issue of image segmentation, which is known in computer vision to be an extremely challenging task. Image segmentation is the task which discards pixels belonging to non-objects (objects or "things" that we do not know a priori) in order to make easy the recognition process. But, what happens when we have two objects together, i.e. one in front of the other? what happens with the pixels that reside in the border zone of both objects?

4. Assuming that pixels of interesting objects can be segmented, it remains an important source of difficulties when we try to group pixels together. The distributed nature of the information is very important since it takes part in relevant grouping decisions. Several times, the ultimate interpretation of an image fragment does not depends on its immediate context but can depend on the entire image. There are several approaches which focus on the identification of a single object using a model-based recognition process without taking into account possible holistic circumstances of the whole scene.

The general problem of visual recognition is too broad and underconstrained to be addressed comprehensively by today's artificial systems. Therefore, the following three subproblems have been identified in the literature that are more easily addressed separately by specialized solutions:

- **Specific Object Recognition.** In a typical scenario, a test image depicts exactly one target object (with or without occlusion and/or clutter). The task is to recognize that object as one of several specific objects [94, 120, 87], i.e. a particular toy or a particular landmark. The appearance or shape of the objects in the database are precisely known.

- **Generic Object Classification.** The scenario is the same as in Specific Object Recognition, but here the task is to label the target object as belonging to one of several known object categories or classes [160]. The individual objects within a class may vary in appearance. A class ("car", "chair") can be defined in various ways, i.e. by object prototypes, by abstract description, or by models containing enough detail to discriminate between classes, but not enough to distinguish between objects within a class.

- **Object Detection.** Test images contain various items and typically show natural (indoor or outdoor) scenes. The task is to localize regions of the image that depicts instances of a target object or class of objects, i.e. faces or people [2]. Object detection is closely related to fundamental problems in image retrieval [111].

The requirements and difficulties differ between these subproblems. Therefore, many existing recognition systems are designed to solve only one of them. Moreover, some practical recognition problems further specialize these categories. For example, face recognition [142] is a distinct subcategory of specific object recognition that must cope with gross non-rigid transformations that are hard to model mathematically. Other practical recognition problems are often addressed in related ways. For example, Weng [154] treat indoor landmark recognition as a simplified object recognition

problem without generalization across size or pose. Riseman [112] represent outdoor landmarks by line models and recognize them by many-to-many model matching in two and three dimensions. Very few promisings attempts have been made to develop systems that are applicable to more than one of these subproblems. Among these, most systems employ features that explicitly allow certain image variations [96].

For visual recognition, two major classes of techniques can be identified:

- **Model-based.** This kind of techniques employ geometric models of the target objects for feature extraction or alignment [112].
- **Appearance-based.** This kind of techniques are related to the appearance of objects in an image. That is, they are related directly to the pixels of an image. They are also called model-free methods since they extract features or extract information from images without explicitly model shape properties of the target objects. Various types of features have been employed and some of them will be explained in the following sections.

Hybrid techniques have also been proposed in which appearance-based methods served as indexing mechanisms to reduce the number of candidate object models [71]. In fact, it appears that this was historically the primary motivation for appearance-based methods. Only later was their full potential discovered, and systems began to emerge that omitted the model-matching phase and relied on appearance-based methods alone.

The human visual system is remarkable. It routinely solves a wide variety of visual tasks with such reliability and deceiving ease that belittles their actual difficulty. These spectacular capabilities appear to rest on at least two foundations. First, the human brain devotes enormous computational resources to vision: about half of our brain is more or less directly involved in processing visual information [68]. Second, essential visual skills are learned in a long process that extends throughout the first years of an individual's life. At the lowest level, the formation of receptive fields of neurons along the early visual pathway is likely influenced by retinal stimulations. Some visual functions do not develop at all without adequate perceptual stimulation within a sensitive period during maturation, i.e. stereo vision [4]. Higher-order visual functions such as pattern discrimination capabilities are also subject to a developmental schedule [49]:

- Neonates can distinguish certain patterns, apparently based on statistical features such as spatial intensity variance or contour density [118].
- Infants begin to note simple coarse-level geometric relationships, but perform poorly in the presence of distracting cues. They do not consistently pay attention to contours and shapes [117].
- At the age of about two years, children begin to discover fine-grained details and higher-order geometric relationships. However, attention is still limited to "salient" features [141].
- Over much of childhood, humans learn to discover distinctive features even if they are overshadowed by more salient distractors.

There is growing evidence that even adults learn new features when faced with a novel recognition task [128]. But despite this evidence of visual feature learning in humans, little is known about the mechanisms of visual learning. At least, recent neurophysiological and psychological studies have shed some light on what the features represent [151]. And it is interesting to see that the bulk of the evidence points to view-specific appearance-based representations in terms of local features. And few definitive statements can be made about the spatial extent of the features used by the visual brain. Nevertheless, it has been shown that even for the recognition of faces (often cited as a prime example of holistic representations) local features play a major role.

It seems clear that local approaches which make use of local features as a first step to object recognition and/or classification are commonly used in front of holistic ones. This is due to the ability of local approaches to inherently solve some of the practical problems of computer vision: occlusions, lighting conditions, changes in viewpoint, etc. However, once a set of local features has been extracted from images it is required some knowledge that groups them into a human concept. For example, figure (2.1) contains two "faces" but if we pay some attention to figures (2.1.a) and (2.1.b) we will see that they do not correspond to real faces. Both images contain local features that belong to the face class but the spatial configuration of these local features is an important issue in order to label images. Thus, if we rotate both images, we obtain figures (2.1.c) and (2.1.d) where we check that figures (2.1.a) and (2.1.b) correspond to unreal faces.

As seen in figure (2.1) if we work with local representations, it is possible to find images where all the local object detectors respond positively but the image does not contain any known object. This problem is known as the binding problem [88]. The binding problem is one of the most persistent worries in the class of architecture where objects in images are based on banks of features detectors, often called receptive fields, each of which is sensitive to some localized spatial configuration of image cues but invariant to one or more spatial transformations of its preferred stimulus. These systems where the goal is to extract useful invariant features suffer the potential for ambiguity in visual representations (the binding problem). A spatial binding problem arises, for example, when each detector in the visual representation reports only the *presence* of some elemental object attribute but not its spatial location (or, more generally, its pose). Under these unfavorable conditions, an object is hallucinated (i.e, detected when not actually present) whenever all of its elemental features are present in the visual field, even when embedded piecemeal in improper configurations within a scattered coalition of distractor objects.

One approach to the binding problem is to build a separate, full-order conjunctive detector for every possible view (e.g., spatial pose, state of occlusion, distortion, degradation) of every possible object, and then for each object provide a huge disjunction that pools over all of the views of that object. Some possible solutions to the binding problem include (1) strategies for image preprocessing designed to segment scenes into regions containing individual objects, thus reducing the clutter load confronting the visual representation, and (2) explicit normalization procedures to reduce pose uncertainty (e.g., centering, scaling, warping), thereby reducing the invariance load that must be sustained by the individual receptive fields. Both strategies reduce

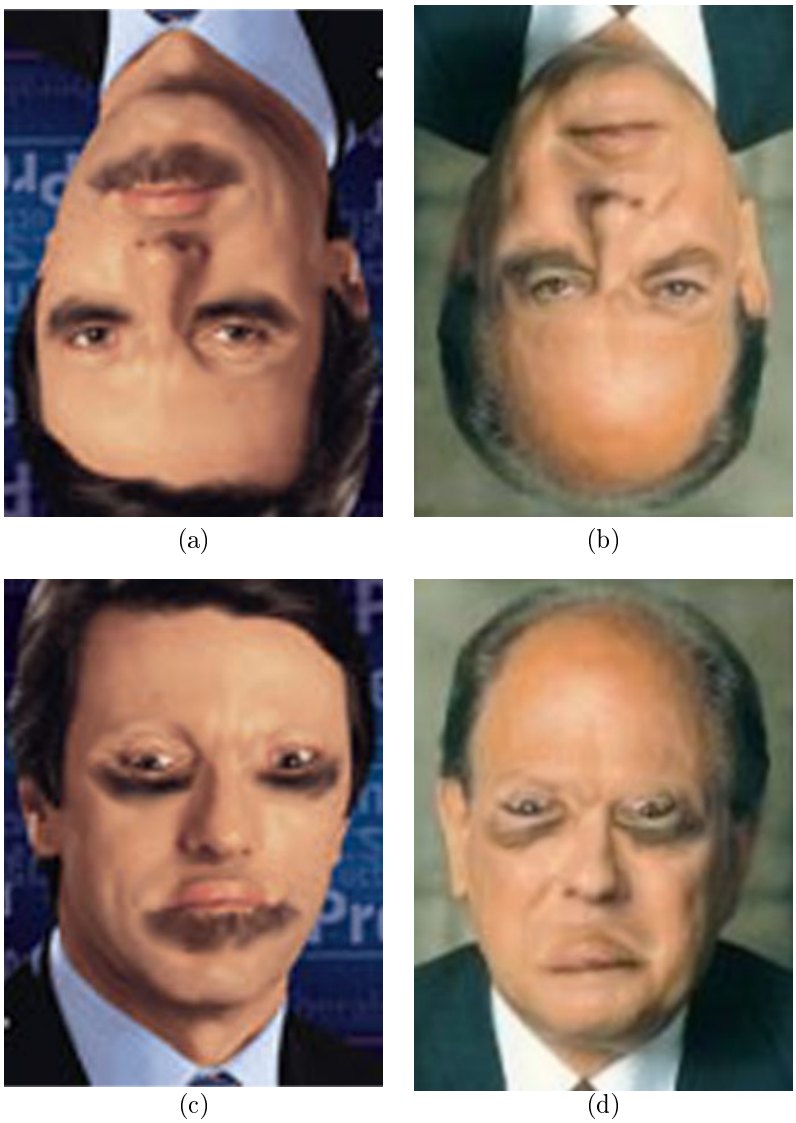


**Figure 2.1:** (a) and (b) are two images that seem to contain "regular" faces. (c) and (d) correspond to the rotated images of (a) and (b) and we see that they are not real face images. Here we see the importance of the spatial layout of local features. This example has been extracted from [108].

the probability that any given object's feature set will be activated by a spurious collection of features drawn from other objects. In any case, we should be aware of this problem in our object representation if we want to reduce possible mismatches.

This chapter introduces a relevant set of local features that can be used to represent local information from images. All methods described here have been used extensively

to describe local attributes of images. We will pay more attention to color features since our research is mainly focused on local color representations. Once a set of feasible local features is exposed we will discuss the topic of topology and location of local features. Finally, when we have described what information should be extracted from specific regions in images we will relate possible methodologies to perform object recognition and/or classification. Thus, we will describe how methods have worked with local features in the last years. We analyze some of the most well-known methods concerned with object recognition using local features and fast matching methods.

## 2.1 Local Features

We start from the assumption that images contain objects and a certain subset of pixels of images belong to an object which we want to identify. The human visual system easily recognizes a wide variety of stimuli, including rigid, articulated, and entirely non-rigid 2D and 3D objects, faces, statistical or fractal objects, surface textures, and views of complex natural scenes, while displaying remarkable insensitivity to changes in viewpoint, partial occlusion and clutter. But why the human visual system can recognize this huge number of objects so well and so fast? One possible answer is that our visual system is organized as a feedforward feature-extracting hierarchy that builds progressively more complex and viewpoint-invariant features useful for identifying objects, where invariance over a group of transformations is achieved by summing over viewpoint-specific elemental representations [106]. According to this view, and in correspondence with the axioms of statistical pattern recognition, visual object recognition in the brain is the process of mapping retinal "pixels" into a feature space that is better suited (than pixels) to the viewpoint invariant classification and identification problems faced by visual animals. Within this feature space, represented by the activity of neurons at the top of the hierarchy, the similarity of one object view to another is given by a simple distance calculation ; recognition of an input is achieved by finding the identity or class of the most similar training view previously stored in memory.

It seems clear that feature dimensions should be chosen such that large changes in object pose produce relatively small variations in feature space, while small changes in object "quality" (shape, texture, color, etc) produce relatively large variations in feature space. Here is where we see that we need a useful feature space, that is, features should be relatively sensitive to object quality but relatively insensitive to an object's pose or configuration.

So that, the choice of the used features is critical and depends on such factors as the considered object classes, the sensor characteristics, the contexts and the task. Generally, there exists a tradeoff between precision and generality of the features. Even though researchers are aware of the difficulties, the choice is most often made by hand and arbitrarily. However, we might expect that our visual features should be able to cope with non-rigid object transformations, or to cope with occlusion and clutter. But one of the most important things is that visual features should be able to maximize object discrimination. Thus, one possible solution to this is the use of multiple visual cues. Also, we want to richly represent objects of many different types,

so that, our visual features should be aware of this fact.

This feature based viewpoint presents several limitations since it is a bottom-up approach. As example of possible problems we have that:

1. Feature-space methods are essentially template-matching methods and so require an intractable number of templates to cope with inputs that have been rotated, scaled, partly occluded, non-rigidly transformed, or presented under varying lighting conditions,

2. Feature-space approaches lack a top-down component essential for resolving featural ambiguities present in real images,

3. Feature-space methods do not scale well to high dimensions (i.e. large numbers of features), either because (i) it is not practical to learn or otherwise assemble a sufficiently large number of sufficiently useful features, (ii) too many dimensions of noise necessarily overwhelm too few dimensions of signal, or (iii) high-dimensional methods are computationally intractable or require too much data.

However, the conjecture that a feature space approach based on feedforward receptive-field-style computations could account for the prodigious recognition capacities of the primate brain as yet lacks direct support in the modeling literature: existing approaches have generally involved one or more assumptions that place them squarely outside the biological "paradigm" for general-purpose visual recognition, such as (i) use of small object corpus (typically no more than a few dozen objects), (ii) limited range of object types (e.g. faces or rigid volumetric objects), (iii) feature computations not amenable to receptive-field-style computations (e.g. use of sophisticated geometric invariants), and/or (iv) strong viewpoint assumptions, or corresponding explicit image pre-normalization operations (typical in OCR and face recognition).

We should take in mind that a feature space describes "WHAT" we are searching for inside images and this is a very important step before we start to create a new visual technique to represent or recognize objects.

Here we describe a set of local feature descriptors that are commonly used in computer vision, some of them are directly related to image features (color, texture, edges or motion) and others are based on more structured information as histograms, filters or contours that are basically based on the local neighborhood of a given region of an image (usually centered around important "points" in the image [25, 55, 124]).

### 2.1.1 Gaussian Derivatives

This section introduces local characteristics based on Gaussian derivatives. Gaussian derivatives are widely used in the literature and well understood [45, 46, 73, 110]. By using Gaussian derivatives we can explicitly select the scale. Additionally, we can "steer" the derivative to arbitrary orientations: it is possible to calculate the $n$th order Gaussian derivative of the orientation $\phi$ based on a linear combination of a finite number of $n$th order derivatives. This section describes Gaussian derivatives in general, develops the equivariance property to scale and finally summarizes the "steerability" to image plane rotation.

Given the Gaussian distribution $G^\sigma(x,y)$ (we are not considering the scale factor for notational convenience) :

$$G^\sigma(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.1}$$

the first order derivative in the $x$-direction is given by:

$$G^\sigma_x(x,y) = -\frac{x}{\sigma^2}G^\sigma(x,y) \tag{2.2}$$

More generally the first order derivative in direction $\vec{v} = (cos\phi \;\; sin\phi)^T$ is given by:

$$G^\sigma_{1,\phi}(x,y) = \frac{\partial}{\partial \vec{v}}G^\sigma(x,y) \tag{2.3}$$

In the same way, the $n$th order Gaussian derivative in direction $\vec{v} = (cos\phi \;\; sin\phi)^T$ is defined by:

$$G^\sigma_{n,\phi}(x,y) = \frac{\partial^n}{\partial \vec{v}^n}G^\sigma(x,y) \tag{2.4}$$

Usually, up to third order Gaussian derivatives are used. If we define the $x$-axis parallel to the vector $\vec{v} = (1 \;\; 0)^T$, which corresponds to $\phi = 0^0$. The $y$-axis is defined by $\phi = 90^0$ and is therefore parallel to $\vec{v} = (0 \;\; 1)^T$. The derivatives in $x-$ and $y-$direction are given by:

$$G^\sigma_x(x,y) = G^\sigma_{1,0^0}(x,y) = -\frac{x}{\sigma^2}G^\sigma(x,y) \tag{2.5}$$

$$G^\sigma_y(x,y) = G^\sigma_{1,90^0}(x,y) = -\frac{y}{\sigma^2}G^\sigma(x,y) \tag{2.6}$$

Based on the first order derivatives one can define the magnitude $Mag(x,y)$ and the direction $Dir(x,y)$ of the first derivative:

$$Mag(x,y) = \sqrt{(G^\sigma_x(x,y))^2 + (G^\sigma_y(x,y))^2} \tag{2.7}$$

$$Dir(x,y) = \arctan\frac{G^\sigma_y(x,y)}{G^\sigma_x(x,y)} \tag{2.8}$$

These two measures that are based on the first order derivatives are usually used in computer vision to represent the neighborhood of a given point in the image, $(x,y)$, that for some reason is interesting for us.

Three second order derivatives are given by:

$$G^\sigma_{xx}(x,y) = (\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2})G^\sigma(x,y) \tag{2.9}$$

$$G^\sigma_{xy}(x,y) = (\frac{xy}{\sigma^4})G^\sigma(x,y) \tag{2.10}$$

$$G^\sigma_{yy}(x,y) = (\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2})G^\sigma(x,y) \tag{2.11}$$

And based on the first and second order derivatives one can define rotation invariant characteristics as:

$$Lap(x,y) = G^{\sigma}_{xx}(x,y) + G^{\sigma}_{yy}(x,y) \quad (2.12)$$

$$G12(x,y) = G^{\sigma}_{xx}(G^{\sigma}_{x})^2 + 2G^{\sigma}_{xy}G^{\sigma}_{x}G^{\sigma}_{y} + G^{\sigma}_{yy}(G^{\sigma}_{y})^2 \quad (2.13)$$

$Lap(x,y)$ is the well known Laplacian operator. The second local characteristic is called $G12$ since it is based on first and second order derivatives. Operator $G12$ was introduced by Koenderink [72] as a rotation (image-plane rotation) invariant filter and has been used by Schmid and Mohr [123, 122] for object recognition.

Finally, we show the four third order Gaussian derivatives that are given by:

$$G^{\sigma}_{xxx}(x,y) = (\frac{3x}{\sigma^4} - \frac{x^3}{\sigma^6})G^{\sigma}(x,y) \quad (2.14)$$

$$G^{\sigma}_{xxy}(x,y) = (\frac{y}{\sigma^4} - \frac{x^2y}{\sigma^6})G^{\sigma}(x,y) \quad (2.15)$$

$$G^{\sigma}_{xyy}(x,y) = (\frac{x}{\sigma^4} - \frac{xy^2}{\sigma^6})G^{\sigma}(x,y) \quad (2.16)$$

$$G^{\sigma}_{yyy}(x,y) = (\frac{3y}{\sigma^4} - \frac{y^3}{\sigma^6})G^{\sigma}(x,y) \quad (2.17)$$

**Equivariance of Gaussian derivatives to Scale**

Gaussian derivatives are a set of local characteristics that can be calculated at an arbitrary scale. This is an interesting property of Gaussian filters and other families of filters as for example Gabor filters (see section 2.1.2). In the following we introduce the equivariance property of Gaussian derivatives to scale.

Given a two-dimensional function $p(x,y)$ and its scaled version $f(x,y) = p(sx,sy)$ we know from analysis:

$$f(x,y) = p(sx,sy) \quad (2.18)$$

$$\frac{\partial}{\partial x}f(x,y) = s\frac{\partial}{\partial x}p(sx,sy) \quad (2.19)$$

$$\vdots$$

$$\frac{\partial^2}{\partial x^2}f(x,y) = s^n\frac{\partial^n}{\partial x^n}p(sx,sy) \quad (2.20)$$

Following the above equations, the $n$th order derivative of the function $f$ can be calculated on the basis of the $n$th order derivative of $p(sx,sy)$. This calculation assumes exact knowledge of the function $p$. In computer vision the exact knowledge of $p$ cannot in general be assumed. By using Gaussian derivatives the $n$th order derivative of $p(sx,sy)$ can be calculated on the basis of the $p(x,y)$. In the following we show this property for the first order derivative. The first order derivative of $f$ is defined as:

$$\frac{\partial}{\partial x}f(x,y) = G^{\sigma}_{x}(x,y) \star f(x,y) \quad (2.21)$$

where $G_x^\sigma(x,y)$ is the Gaussian derivative (see equation 2.5) and $\star$ is the convolution operator. Therefore we obtain (together with equation 2.19):

$$\begin{aligned} \frac{\partial}{\partial x} f(x,y) &= s\frac{\partial}{\partial x} p(sx,sy) && (2.22)\\ &= sG_x^\sigma(x,y) \star p(sx,sy) && (2.23)\\ &= sG_x^\sigma(x,y) \star p(x,y) && (2.24) \end{aligned}$$

The equation shows that we can calculate the first order derivative of $f$ on the basis of the first order derivative of $p(x,y)$, which we call the *adaption of the Gaussian derivative to scale*. In a similar way one obtains an equation for the adaptation of the $n$th order derivative to scale:

$$\frac{\partial^n}{\partial x^n} f(x,y) = s^n G_{x^n}^{\sigma s}(x,y) \star p(x,y) \qquad (2.25)$$

Following this equation, one can calculate the $n$th order derivative of a function $f(x,y)$ directly from the basis of function $p(x,y)$ (when $f$ is a scaled version of $p$: $f(x,y) = p(sx,sy)$). In order to employ this property the scale factor $s$ must be known, which cannot in general be assumed. Usually we calculate the derivative for different values of $s$. Additionally, one has to adapt the support for the calculation of the $n$th order derivative of $p$. This is expressed by the adaptation of the standard deviation $\sigma s$ of the Gaussian filter.

We call the adaption of the Gaussian derivatives to scale changes by the factor $s$ the *equivariance* property of the Gaussian derivatives to scale. More generally we call any local characteristic equivariant to a particular change, whenever there exist a certain parameter which is directly connected with this change.

As expected, the *equivariance to scale* is not only true for characteristics based on Gaussian derivatives. The same property holds, for example, for Gabor filters due to their Gaussian envelope. Gabor filters are introduced in section 2.1.2.

### Steerability of Gaussian Derivatives to Image Plane Rotation

In order to calculate the filter response (for example for a Gaussian filter) at an arbitrary orientation $\phi$ the corresponding version of the filter can be calculated. If the orientation is not known beforehand or if one of the filter responses of many different orientations have to be calculated, it is very time consuming to calculate every time the corresponding version of the filter. Therefore it is desirable to define a finite set of basis filters and an interpolation rule, which allows us to calculate the filter response based only on the response of the basis set. For the first order Gaussian derivative there exists a well known linear interpolation rule [46]:

$$G_{1,\phi}^\sigma = cos\phi G_x^\sigma + sin\phi G_y^\sigma \qquad (2.26)$$

In order to formalize this property, Freeman and Adelson [46] start with a two-dimensional function $F(x,y)$ and define $F^\theta(x,y)$ as the version of $F(x,y)$ rotated by the angle $\theta$. A function $F(x,y)$ *steers* when it can be written as a sum of rotated versions of itself:

$$F^\phi(x,y) = \sum_{j=1}^{J} k_j(\phi) F^{\theta_j}(x,y) \qquad (2.27)$$

This equation is known as the steering constraint. $J$ corresponds to the number of interpolation functions $k_j(\theta)$, the $F^{\theta_j}$ form the finite basis set of oriented functions. Two important questions arise: how many interpolation functions are needed and how to obtain the interpolation functions. In order to formulate two theorems, the function $F(x,y)$ is rewritten in polar coordinates $r = \sqrt{(x^2+y^2)}$ and $\rho = arg(x,y)$:

$$F(r,\rho) = \sum_{n=-N}^{N} a_n(r) e^{in\rho} \tag{2.28}$$

A first theorem [46] states that the steering constraint 2.27 holds for functions expandable in the form of 2.28 if and only if the interpolation functions $k_j(\phi)$ are solutions of:

$$\begin{pmatrix} 1 \\ e^{i\phi} \\ \vdots \\ e^{iN\phi} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ e^{i\theta_1} & e^{i\theta_2} & \dots & e^{i\theta_J} \\ \vdots & \vdots & \ddots & \vdots \\ e^{iN\theta_1} & e^{iN\theta_2} & \dots & e^{iN\theta_J} \end{pmatrix} \begin{pmatrix} k_1(\phi) \\ k_2(\phi) \\ \vdots \\ k_J(\phi) \end{pmatrix} \tag{2.29}$$

By fixing $\theta_j$ the interpolation functions are given as solutions of equation 2.29. A second theorem [46] states that the minimal number of interpolation functions is $T$, whereas $T$ is the number of nonzero coefficients $a_n(r)$ in equation 2.28. Therefore the theoretical answers to the two questions are known: how many interpolation functions are needed and how the interpolation functions are defined.

The two theorems can be applied to Gaussian derivatives. First of all (following the second theorem) the minimal number of interpolation functions for the $n$th order Gaussian derivative is $n+1$. The interpolation functions themselves depend on the chosen $\theta_j$. These angles of the basis filters should be chosen so that the response of the interpolation is robut with respect to noise. An appropiate choice is therefore to distribute the angles equally between $0°$ and $180°$. (Another choice for the $\theta_j$ may be motivated by the desire to obtain separable filters for $x$ and $y$. See for details [46].)

Besides the well known interpolation functions for the first order Gaussian derivatives (see equation 2.26: $\theta_1 = 0°$, $\theta_2 = 90°$,$k_1(\phi) = cos(\phi)$,$k_2(\phi) = sin(\phi)$) the following interpolation functions are obtained for the second order Gaussian derivatives (using $\theta_1 = 0°$, $\theta_2 = 60°$ and $\theta_3 = 120°$):

$$k_j(\phi) = \frac{1 + 2\cos(2(\phi - \theta_j))}{3} \text{ for } j = 1,2,3 \tag{2.30}$$

$$G^{\sigma}_{2,\phi} = k_1(\phi) G^{\sigma}_{2,0°} + k_2(\phi) G^{\sigma}_{2,60°} + k_3(\phi) G^{\sigma}_{2,120°} \tag{2.31}$$

And finally, the following interpolation functions are obtained for the third order Gaussian derivatives (using $\theta_1 = 0°$, $\theta_2 = 45°$, $\theta_3 = 90°$ and $\theta_4 = 135°$):

$$k_j(\phi) = \frac{2\cos(\phi - \theta_j) + 2\cos(3(\phi - \theta_j))}{4} \text{ for } j = 1,2,3,4 \tag{2.32}$$

$$G^{\sigma}_{3,\phi} = k_1(\phi) G^{\sigma}_{3,0°} + k_2(\phi) G^{\sigma}_{3,45°} + k_3(\phi) G^{\sigma}_{3,90°} + k_4(\phi) G^{\sigma}_{4,135°} \tag{2.33}$$

Figure (2.2) shows up to fourth-order Gaussian derivatives that usually are used in computer vision.

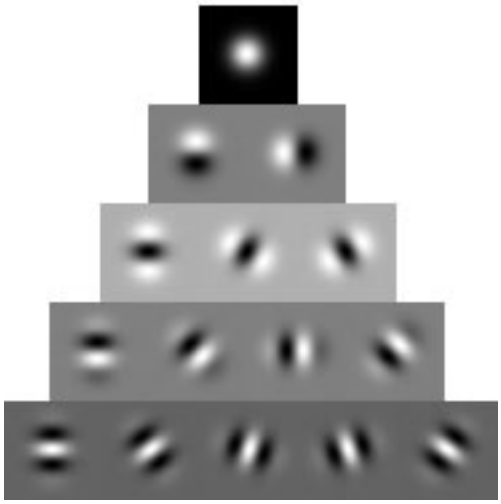


**Figure 2.2:** 14 oriented Gaussian derivative basis filters of up to the fourth-order and the initial Gaussian. From top to down: a Gaussian ; 2 first order Gaussian derivatives; 3 second order Gaussian derivatives ; 4 third order Gaussian derivatives and 5 fourth order Gaussian derivatives. Bright regions denote positive magnitude while darker regions denote negative magnitude.

### 2.1.2 Gabor Filters

Gabor filters are local compact filters tuned to a spatial frequency band. Gabor filters are defined by modulating a Gaussian window with a cosine and an imaginary sine, giving an even and an odd filter pair [48]. The output of a Gabor filter may be represented as a complex number with a real and an imaginary part. This complex number can be expressed in polar coordinates as magnitude and direction.

A Gabor filter pair is compact in both space and frequency domains. Usually, one can use the two-dimensional formulation of the Gabor functions used by Daugman [35]:

$$g(x,y) = e^{-\pi\left(\frac{(x-x_0)^2}{\alpha^2}+\frac{(y-y_0)^2}{\beta^2}\right)} e^{-2\pi i(u_0(x-x_0)+v_0(y-y_0))} \tag{2.34}$$

where $(x_0, y_0)$ are the central coordinates of the filter, $(\alpha, \beta)$ are the standard deviation used to determine the width and the length, and $(u_0, v_0)$ specify the modulation in $x$ and $y$ directions, which has the spatial frequency $w_0 = \sqrt{u_0^2 + v_0^2}$ and direction $\theta_0 = \arctan(v_0/u_0)$. The Fourier domain transfer function $G(u,v)$ is given by:

$$G(u,v) = e^{-\pi((u-u_0)^2\alpha^2+(v-v_0)^2\beta^2)} e^{-2\pi i(x_0(u-u_0)+y_0(v-v_0))} \tag{2.35}$$

The main advantage of the Gabor filters is that one can freely choose the frequency (and therefore the scale) as well as the bandwidth of the filter.

For the design of a 1D Gabor filter, Westelius [156] proposes the tuning of the standard deviation $\alpha$ and the spatial frequency $u_0$. These two parameters determine

the size and bandwidth of the filter. Given a certain spatial frequency $u_0$, Westelius chooses the radius of the frequency support so that the frequency function is sufficiently low at $u = 0$ (DC component). Therefore he defines the ratio between the DC component and the maximum value which should be smaller than a previously selected threshold $P_{DC}$. To generalize this ratio to $2D$ Gabor filters one can set $\alpha = \beta$. The ratio is then given by (for $(x_0, y_0) = (0, 0)$):

$$\frac{G(0,0)}{G(u_0,v_0)} \leq P_{DC} \Rightarrow \alpha \geq \frac{\sqrt{-\ln P_{DC}}}{\sqrt{\pi} w_0} \tag{2.36}$$

Both the spatial support and the frequency support of an ideal Gabor function are infinite. To define a digital filter we must sample the function and limit its spatial support. The limit of the spatial size is chosen such that the amplitude of the filter is lower than a threshold $P_{cut}$ for all positions outside the limit. The spatial radius $R = \sqrt{r_x^2 + r_y^2}$ of the filter is then:

$$\frac{\parallel g(r_x, r_y) \parallel}{\parallel g(0,0) \parallel} \leq P_{cut} \Rightarrow R \geq \alpha \frac{\sqrt{-\ln P_{cut}}}{\sqrt{\pi}} \tag{2.37}$$

One should notice that the chosen radius $R$ affects the $DC$ component of the signal. Therefore the DC component should be checked after truncation of the filter support.

By using Euler's equation ($e^{iwx} = \cos(wx) + i\sin(wx)$) we can rewrite equation 2.34 in polar form:

$$g(x,y) = Re(g(x,y)) + Im(g(x,y)) \tag{2.38}$$

$$Re(g(x,y)) = \cos(-2\pi(u_0(x-x_0) + v_0(y-y_0)))e^{-\pi\left(\frac{(x-x_0)^2}{\alpha^2} + \frac{(y-y_0)^2}{\beta^2}\right)} \tag{2.39}$$

$$Im(g(x,y)) = i\sin(-2\pi(u_0(x-x_0) + v_0(y-y_0)))e^{-\pi\left(\frac{(x-x_0)^2}{\alpha^2} + \frac{(y-y_0)^2}{\beta^2}\right)} \tag{2.40}$$

The real part $Re(g(x,y))$ can be interpreted as a second order derivative and the imaginary part $Im(g(x,y))$ can be interpreted as a first order derivative (both in direction of $\theta_0$). Figure (2.3) shows a typical Gabor filter bank (the real part of the filter).

### 2.1.3 Color Features

#### Color Histograms

In computer vision, one of the most common features used as a color signature is color histograms. Swain and Ballard [137] have initially proposed this scheme where each object is represented by its color histogram (an approximation of its color distribution). Originally, this color histogram technique was used to represent a whole object but it is clear that we can use this technique under a local point of view as seen in the literature [19, 52]. In the first approach, objects are identified in an image by matching a color histogram from a region of the image with a color histogram from a sample of the object (several histogram matching techniques are presented

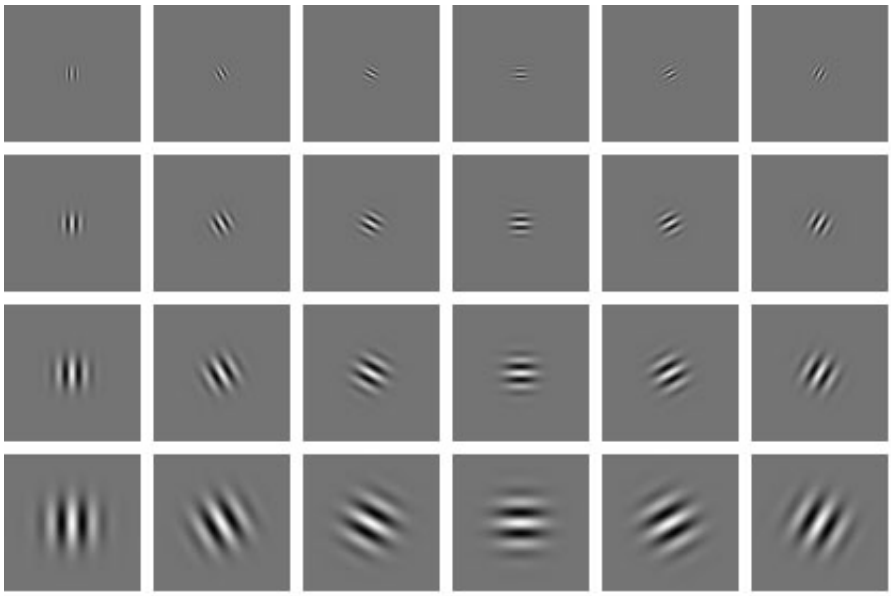


**Figure 2.3:** A typical Gabor filter bank. The real part (see equation 2.39) of the filters is shown. This particular filter bank contains 6 different orientations and 4 different scales.

thereafter). Additionally, the use of color histograms for recognition does not assume segmentation of an object nor an explicit geometric model. An object is described only by its color histogram. Color histograms have been shown to be remarkably robust to changes in the object's orientation, changes of the scale of the object, partial occlusion or changes of the viewing position. Even changes in the shape of an object do not necessarily degrade the performance of the method. However, the major drawback of this method is its sensitivity to the color and intensity of the light source. Furthermore, not all objects can be described by color alone. Thus, color histograms could be inappropiate for many recognition problems. For the Swain and Ballard algorithm, it can be seen that robustness to scale and rotation are provided by the use of color. Robustness to changes in viewing angle and to partial occlusion are due to the use of *histogram matching*. Thus it is natural to exploit the power of histogram matching to perform recognition based on histograms of local shape properties. The most general method to measure such properties is the use of a vector of linear local neighborhood operations, or receptive fields [110, 120].

One possible solution in order to reduce the sensitivity to illumination changes, as proposed by Swain and Ballard is to use a color constancy algorithm prior to histogramming. In this particular case, several authors have improved the original approach in order to reduce the sensitivity to illumination intensity changes. The two most successful approaches have been proposed by Healey and Slater [56] and Funt and Finlayson [47]. Healey and Slater calculate moment invariants of the entire color histogram. The invariants are based on the finite dimensional linear color model,

which allows the modeling of illumination intensity changes by a linear transformation between color histograms. Even though the experimental results [56] are convincing the method is global in the sense that the intensity change is assumed to be constant over the entire image. This makes the approach inappropiate in many situations. Funt and Finlayson [47] use derivatives of the logarithms of the color channels in order to provide illumination invariant features. The underlying assumption is a local constant illumination. The color invariants are based on an approximative color model (coefficient model) which allows us to calculate color invariants locally. Even though the underlying model is a simplification these invariants perform best when compared with other color invariants as for example proposed by Nagao [95]. The main advantage of the proposed invariants is that they can be calculated locally and do not assume a uniform illumination change in the entire image.

It seems clear that before obtaining color histograms from images, we must know, at least, the color space to be used. Knowing which is the best color space for our final purpose, we also must have an idea of the precision (quantization steps). One common color space used in the literature is the hue-saturation-value (HSV) [27]. Other studies using color histograms work directly on the red-green-blue (RGB) color space [19]. In fact and depending on the lighting conditions of the scene, it does not matter which color space we use. For example, if we know a priori that our scenes are controlled and the impact of lighting changes is minimal, we are free to choose the color space. But, as seen in the literature [27], HSV is attractive in theory. It is considered more suitable since it separates the color components (HS) from the luminance component (V) and is less sensitive to illumination changes. Also, distances in the HSV space correspond to perceptual differences in color in a more consistent way than in the RGB space.

A color histogram is a vector where each entry stores the number of pixels of a given color in the image. In order to work with images of different sizes, histograms are normalized and the colors of the image are mapped into a discrete colorspace containing $n$ colors. Typically images are represented in the RGB colorspace, using a few of the most significant bits per color channel to discretize the space. Usually, one can find $8 \times 8 \times 8$ color histograms [19, 51] where each 8 means a partition of each color channel into 8 bins. Thus, using this representation, one can represent an image using $8 \times 8 \times 8 = 512$ values. The number of bins is chosen a priori and reflects the precision required for our task. Once a color histogram is defined in a three dimensional space ($8 \times 8 \times 8$), it is very usual to map this three dimensional representation into a one dimensional one (512 dimensional vector) in order to make it more understandable in terms of the statistical techniques that will be applied after. Figure (2.4) shows the histogram obtained considering a gray image where the horizontal axis represents gray pixels and the vertical axis their frequency in the image. A typical application where one can use color histograms is in the context of skin detection. Figure (2.5) shows a typical RGB distribution of pixels corresponding to skin pixels that usually are modeled using color histograms for fast skin-detection applications.

If we represent a histogram by $H = \{h_i\}$ where $h_i$ is the number of pixels in an image that have a color value in the interval indexed by $i = (R_i, G_i, B_i)$ (we are assuming a RGB color space), one can define a set of histogram matching distances in order to compare two histograms ($H$,$K$). Several measures have been proposed

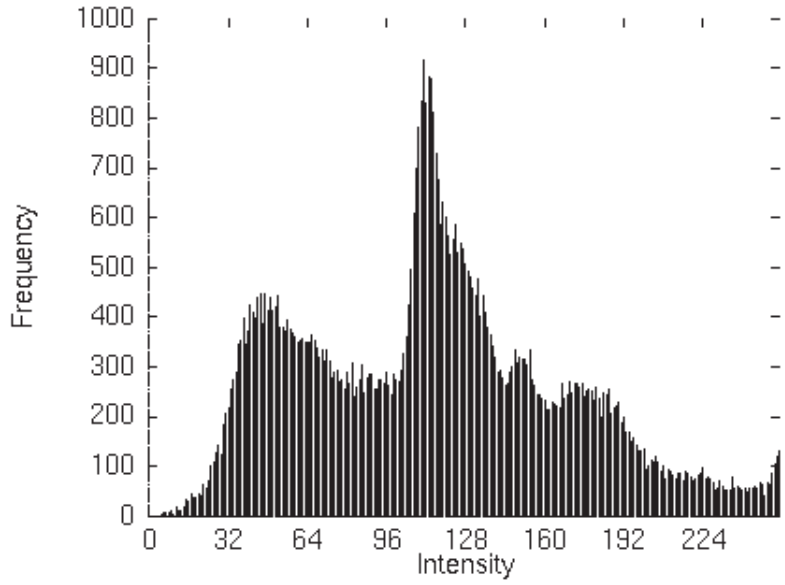



**Figure 2.4:** Intensity histogram of a gray image. The horizontal axis correspond to the gray level and the vertical axis reflects its frequency.

for the dissimilarity between two histograms. One can divide these measures into two categories. The *bin-by-bin* dissimilarity measures that only compare contents of corresponding histogram bins, that is, they compare $h_i$ and $k_i$ for all $i$, but not $h_i$ and $k_j$ for $i \neq j$. And the *cross-bin* measures that also contain terms that compare non-corresponding bins. As we will explain later, cross-bin distances make use of a *ground distance* $d_{ij}$, defined as the distance between the representative features for bin $i$ and bin $j$. Typical bin-by-bin dissimilarity measures are:

- **Minkowski-form distance**

$$d_{L_r}(H,K) = \left( \sum_i |h_i - k_i|^r \right)^{1/r} \tag{2.41}$$

  This dissimilarity measure considers $L_1$ and $L_2$, two typical distance measures. The $L_1$ distance is often used for computing dissimilarity between color images [137] but it also has been shown in other studies like [136] where image retrieval based on $L_1$ results in many false negatives.

- **Histogram Intersection**

$$d_{\cap}(H,K) = 1 - \frac{\sum_i \min(h_i, k_i)}{\sum_i k_i} \tag{2.42}$$

  The histogram intersection was introduced by Swain and Ballard [137] and it is attractive because its ability to handle partial matches when the areas of

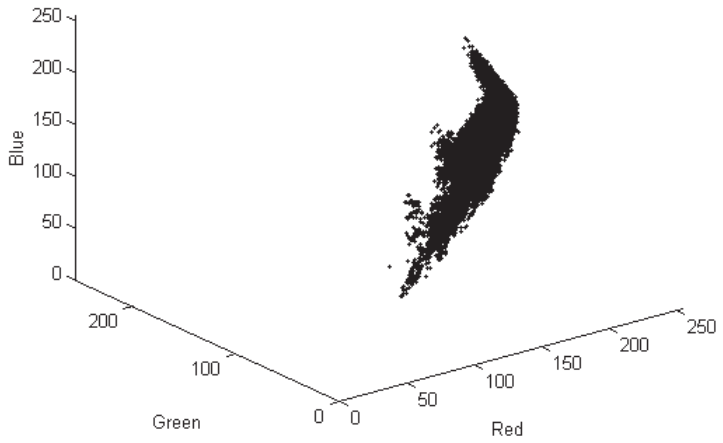



**Figure 2.5:** Distribution of points corresponding to skin pixels in the RGB space. Such distribution of pixels is usually modeled using color histograms for fast skin-detection purposes (real time applications).

the two histograms are different. It is shown in [137] that when the areas of the two histograms are equal, the histogram intersection is equivalent to the (normalized) $L_1$ distance measure.

The intuitive motivation for this measurement is the calculation of the common part (the intersection) of two histograms $H$ and $K$. An advantage of this measurement is that it explicitly neglects background pixels, which may occur in the test histogram $K$ but do not occur in the database histogram $H$. This measurement is computational inexpensive since it is based on simple operations for each histogram cell. Furthermore, the complexity is linear with the number of histogram cells (dimensionality of the feature vector).

In the original work of Swain and Ballard [137] it is claimed the need for a sparse color distribution in the histogram in order to distinguish different objects. For example, a sparse distribution can be achieved by using high dimensional histograms. It is clear that a tradeoff between the ability to discriminate objects and stability with respect to perturbations becomes an important issue. A second inconvenience of the intersection is that all histogram cells are treated equally and should therefore be equally probable. In some particular applications of histograms [120] it is more convenient to use an appropiate scaling of the different histogram cells using some *a priori* knowledge.

- **Kullback-Leibler divergence and Jeffrey divergence**

The Kullback-Leibler (K-L) divergence [74] is defined as

$$d_{KL}(H,K) = \sum_i h_i \log \frac{h_i}{k_i} \tag{2.43}$$

From the information theory point of view, the K-L divergence has the property that it measures how inefficient on average it would be to code one histogram using the other as the code-book [34]. However, the K-L divergence is non-symmetric and is sensitive to histogram binning. The empirically derived Jeffrey divergence is a modification of the K-L divergence that is numerically stable, symmetric and robust with respect to noise and the size of histogram bins [109]. It is defined as:

$$d_J(H,K) = \sum_i \left( h_i \log \frac{h_i}{m_i} + k_i \log \frac{k_i}{m_i} \right) \tag{2.44}$$

where $m_i = \frac{h_i + k_i}{2}$.

- $\chi^2-$ **statistics**
  The formal statistical method for determining if two distributions differ is the $\chi^2-$test. Starting from the null hypothesis that two data sets (histograms) are drawn from the same population (for example measurements of the same object) the goal is to disprove the hypothesis. Disproving the hypothesis proves that the histograms are drawn from different distributions. Failing to disprove the null hypothesis only shows that the two histograms are consistent and could be drawn from the same population. Therefore, the $\chi^2-$test is a consistency test for two histograms.

  $\chi^2-$statistics is a common measure used for the calculation of "dissimilarity" between two histograms [27, 53, 120]. Two different ways of calculation of the $\chi^2-$statistics are considered here . The first - $d_{\chi^2_k}(H,K)$ - assumes exact knowledge of the model histogram K:

$$d_{\chi^2_k}(H,K) = \sum_i \frac{(h_i - k_i)^2}{k_i} \tag{2.45}$$

  A second calculation - $d_{\chi^2_{hk}}(H,K)$ - compares two observed histograms (neither is theoretically derived). This second $\chi^2-$statistics is more appropiate in the context of object recognition, since we typically do not assume exact knowledge of the model histogram $K$. $d_{\chi^2_{hk}}(H,K)$ is defined by:

$$d_{\chi^2_{hk}}(H,K) = \sum_i \frac{(h_i - k_i)^2}{h_i + k_i} \tag{2.46}$$

  As we will see later in this thesis and as can be seen in other studies, this $\chi^2-$statistics provide better recognition results for most cases than other measurements. We should take in mind that neither of the $\chi^2-$statistics is a metric since the triangulation inequality is not satisfied. In order to see this we can consider the degenerate case of one-cell histograms $A = (a)$ and $C = (c)$. For

any one-cell histogram $B = (b)$ with $a < b < c$ the following inequality holds (which is the opposite of the usual triangular inequality of a metric):

$$d_{\chi^2_{ab}}(A,B) + d_{\chi^2_{bc}}(B,C) < d_{\chi^2_{ac}}(A,C) \tag{2.47}$$

These dissimilarity definitions can be appropriate in different areas. For example, the Kullback-Leibler divergence is justified by information theory and the $\chi^2-$statistics by statistics. However, these measures do not necessarily match perceptual similarity well. The major drawback of these measures is that they account only for the correspondence between bins with the same index, and do not use information across bins. Another drawback of bin-by-bin dissimilarity measures is their sensitivity to bin size. A binning that is too coarse will not have sufficient discriminative power, while a binning that is too fine will place similar features in different bins which will never be matched.

Here we enumerate some of the most important *cross-bin* dissimilarity measures. These measures can only be applied when a ground distance that matches perceptual dissimilarity is available for each histogram bin. Thus, incorporating this information into the dissimilarity measure results in perceptually more meaningful dissimilarity measures. These measures are:

- **Quadratic-form distance**
  This distance was suggested for color based retrieval in [98] as:

  $$d_A(H,K) = \sqrt{(\mathbf{h}-\mathbf{k})^T \mathbf{A} (\mathbf{h}-\mathbf{k})} \tag{2.48}$$

  where $\mathbf{h}$ and $\mathbf{k}$ are vectors that list all the entries in $H$ and $K$. That is, a unidimensional vector representing the whole set of features.

  Cross-bin information is incorporated via a similarity matrix $\mathbf{A} = [a_{ij}]$ where $a_{ij}$ denotes similarity between bins $i$ and $j$. Here $i$ and $j$ are scalar indices into the bins.

  According to the recommendation found in [98] one can use $a_{ij} = 1 - d_{ij}/d_{max}$ where $d_{ij}$ is the ground distance between bins $i$ and $j$ of the histogram, and $d_{max} = \max(d_{ij})$. Although in general the quadratic-form is not a true distance, it can be shown that with this choice of $\mathbf{A}$, the quadratic-form is indeed a distance.

  The quadratic-form distance does not enforce a one-to-one correspondence between mass elements in the two histograms. The same mass in a given bin of the first histogram is simultaneously made to correspond to masses contained in different bins of the other histogram. In [136] it was shown that using the quadratic-form distance in image retrieval results in false positives, because it tends to overestimate the mutual similarity of color distributions without a pronounced mode.

  A special case of the quadratic-form distance is when we use $\mathbf{A}$ as the identity matrix. This particular case generates the sum of squared distances (SSD):

  $$d_{SSD}(H,K) = \sqrt{\sum_i (h_i - k_i)^2} \tag{2.49}$$

This sum of squared distances is the same as the $L_2$ distance metric presented in equation (2.41). And another well-known quadratic-form distance that is commonly used in the computer vision community is the *mahalanobis* distance. A sensible choice of matrix $\mathbf{A}$ is the inverse of the covariance matrix of histogram cells, modeling the significance of each cell and the dependencies between different cells. When different cells of the histogram are mutually independent only the diagonal elements of the matrix $\mathbf{A}$ are nonzero and we obtain the Mahalanobis distance:

$$d_{Maha}(H,K) = \sqrt{\sum_i \frac{(h_i - k_i)^2}{\eta_i^2}} \tag{2.50}$$

where $\eta_i^2$ is the variance of the histogram cell $i$.

- **Match distance**

$$d_M(H,K) = \sum_i |\hat{h}_i - \hat{k}_i| \tag{2.51}$$

where $\hat{h}_i = \sum_{j \le i} h_j$ is the cumulative histogram of $\{h_i\}$, and similarly for $\{k_i\}$.

The match distance [130, 155] between two one-dimensional histograms is defined as the $L_1$ distance between their corresponding cumulative histograms. For one-dimensional histograms with equal areas, this distance is a special case of the EMD which we present later with the important difference that the match distance cannot handle partial matches. Also, the match distance does not extend to higher dimensions because the relation $\mathbf{j} \le \mathbf{i}$ is not a total ordering in more than one dimension, and the resulting arbitrariness causes problems.

- **Kolmogorov-Smirnov distance**

$$d_{KS}(H,K) = \max_i(|\hat{h}_i - \hat{k}_i|) \tag{2.52}$$

Again, $\hat{h}_i$ and $\hat{k}_i$ are cumulative histograms.

The Kolmogorov-Smirnov distance is a common statistical measure for unbinned distributions. Similarly to the match distance, it is defined only for one dimension.

- **Parameter-based distances**
These methods first compute a small set of parameters from the histograms, either explicitly or implicitly, and then compare these parameters. For instance, in [136] the distance between distributions is computed as the sum of the weighted distances of the distributions' first three moments. It is unclear how to tune the weights of the different moments. Moreover, the resulting measure is not a metric distance. In [83], textures are compared based on measures of their periodicity, directionality, and randomness, while in [86] texture distances are defined by comparing their means and standard deviations in a weighted-$L_1$ sense.

- **The Earth Mover's Distance (EMD)**
  This thesis uses the earth mover's distance (EMD) in some of the experiments, so that, we will explain this distance measure in detail.

  First of all, we defined a histogram in previous section as deriving from a fixed partitioning of the domain of a distribution. Of course, even if bin sizes are fixed, they can be different in different parts of the underlying feature space. Even so, however, for some images often only a small fraction of the bins contain significant information, while most others are hardly populated. A finely quantized histogram is highly inefficient in this case. On the other hand, for images that contain a large amount of information, a coarsely quantized histogram would be inadequate. Similar problems arise even when adaptive histograms are used. In brief, because histograms are fixed-size structures, they cannot achieve a good balance between expressiveness and efficiency.

  In the context of EMD, a *signature* $\{s_j = (\mathbf{m}_j, w_j)\}$, on the other hand, represents a set of feature clusters. Each cluster is represented by its mean (or mode) $\mathbf{m}_j$, and by the fraction $w_j$ of pixels that belong to that cluster. The integer subscript $j$ ranges from one to a value that varies with the complexity of the particular image. While $j$ is simply an integer, the representative $\mathbf{m}_j$ is a $d$-dimensional vector. The size of the clusters in the feature space should be limited and not exceed the extent of what is perceived at the same, of very similar, feature.

  The main important point here, is that the definition of cluster is open and a histogram $\{h_i\}$ can be viewed as a signature $\{\mathbf{s}_j = (\mathbf{m}_j, w_j)\}$ in which the vectors $\mathbf{i}$ index a set of clusters defined by a fixed *a priori* partitioning of the underlying space. If vector $\mathbf{i}$ maps to cluster $j$, the point $\mathbf{m}_j$ is the central value in bin $\mathbf{i}$ of the histogram, and $w_j$ is equal to $h_i$.

  Intuitively, given two distributions, one can be seen as a mass of earth properly spread in space, the other as a collection of holes in that same space. Then, the EMD measures the least amount of work needed to fill the holes with earth. Here, a unit of work corresponds to transporting one unit of earth by one unit of ground distance. The distance measure between weight locations is known as the *ground distance*.

  The *ground distance* between two single perceptual features can be found by psychophysical experiments. For example, perceptual color spaces were devised in which the Euclidean distance between two single colors approximately matches human perception of the difference between those colors. This becomes more complicated when sets of features, rather than single colors, are being compared. The EMD appeared in order to become a general metric between signatures for image retrieval making use of a ground distance between signatures.

  Computing the EMD is based on a solution to the well-known *transportation problem* [58]. Suppose that several *suppliers*, each with a given amount of goods, are required to supply several *consumers*, each with a given limited capacity. For each supplier-consumer pair, the cost of transporting a single unit of goods is given. The transportation problem is then to find a least-expensive flow of

goods from the suppliers to the consumers that satisfies the consumers' demand. Signature matching can be naturally cast as a transportation problem by defining one signature as the supplier and the other as the consumer, and by setting the cost for a supplier-consumer pair to equal the ground distance between an element in the first signature and an element in the second. Intuitively, the solution is then the minimum amount of "work" required to transform one signature into the other.

This can be formalized as the following linear programming problem: Let $P = (p_1, w_1), \ldots, (p_m, w_{p_m})$ be the first signature with $m$ clusters, where $p_i$ is the cluster representative and $w_{p_i}$ is the weight of the cluster; so that $Q = (q_1, w_{q_1}), \ldots, (q_n, w_{q_n})$ the second signature with $n$ clusters; and $\mathbf{D} = [d_{ij}]$ the ground distance matrix where $d_{ij}$ is the ground distance between clusters $p_i$ and $q_j$. We want to find a flow $\mathbf{F} = [f_{ij}]$, with $f_{ij}$ the flow between $p_i$ and $q_j$, that minimizes the overall cost

$$\text{Work}(P, Q, \mathbf{F}) = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} f_{ij} \tag{2.53}$$

subject to the following constraints:

$$f_{ij} \geq 0 \qquad 1 \leq i \leq m, 1 \leq j \leq n \tag{2.54}$$

$$\sum_{j=1}^{n} f_{ij} \leq w_{p_i} \qquad 1 \leq i \leq m \tag{2.55}$$

$$\sum_{i=1}^{m} f_{ij} \leq w_{q_j} \qquad 1 \leq j \leq n \tag{2.56}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} = \min \left( \sum_{i=1}^{m} w_{p_i}, \sum_{j=1}^{n} w_{q_j} \right) \tag{2.57}$$

Constraint (2.54) allows moving "supplies" from $P$ to $Q$ and not vice versa. Constraint (2.55) limits the amount of supplies that can be sent by the clusters in $P$ to their weights. Constraint (2.56) limits the clusters in $Q$ to receive no more supplies than their weights; and contraint (2.57) forces to move the maximum amount of supplies possible. This amount is usually called *total flow*. Once the transportation problem is solved, and we have found the optimal flow $\mathbf{F}$, the earth mover's distance is defined as the work normalized by the total flow:

$$\text{EMD}(P, Q) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} f_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}} \tag{2.58}$$

Tne normalization factor is the total weight of the smaller signature, because of constraint (2.57). This factor is needed when the two signatures have different total weight, in order to avoid favoring smaller signatures. In general, the ground distance $d_{ij}$ can be any distance and will be chosen according to the problem at hand.

> Thus, the EMD naturally extends the notion of a distance between single elements to that of a distance between sets, or distributions, of elements. The advantages of the EMD over previous definitions of distribution distances should be apparent. First, the EMD applies to signatures, which subsume histograms as explained before. The greater compactness and flexibility of signatures is in itself an advantage, and having a distance measure that can handle these variable-size structures is important. Second, the cost of moving "earth" reflects the notion of nearness properly, without the quantization problems of most current measures. Even for histograms, in fact, items from neighboring bins now contribute similar costs, as appropiate. Third, the EMD allows for partial matches in a very natural way. This is important, for instance, in order to deal with occlusions and clutter in image retrieval applications, and when matching only parts of an image. Fourth, if the ground distance is a metric and the total weights of two signatures are equal, the EMD is a true metric, which allows endowing image spaces with a metric structure. A proof of this is given in [31, 116, 115]
>
> It should be noted that EMD seems a good distance metric to be used when comparing different histogram bins, but we have to take in mind that this distance requires a high computational cost. The implementation that we have used in this thesis is the one that we can find in `http://robotics.stanford.edu/~rubner/`

Additional dissimilarity measures for image retrieval are evaluated and compared in [109]. But up to this point and as seen in the above mentioned measures, not all the measures are metric. Though it seems that the distance functions should be a metric, recent research in computer vision and cognitive science, however, suggest that human perceptual judgment about visual similarity are inherently non-metric, i.e., they may not obey the triangle inequality or may not be symmetric. Some non-metric similarity measures are also suggested for image classification in [66]

Once the color histogram representation scheme was introduced by Swain and Ballard, other approaches have been presented in order to solve some of the problems of the original technique. For example, several schemes for using spatial information about colors to improve upon the histogram method have been proposed recently. One common approach is to divide images into subregions and impose positional constraints on the image comparison (*image partitioning*). Another approach is to augment histograms with local spatial properties (*histogram refinement*).

Smith and Chang [132] partition an image into binary color sets. They first select all colors that are "sufficiently" present in a region. The colors for a region are represented by a binary color set that is computed using histogram backprojection [137]. The binary color sets and their location information constitute the feature. Stricker and Dimai [135] divide the image into five partially overlapping regions and compute the first three moments of the color distributions in each image. They compute moments for each color channel in the HSV colorspace, where pixels close to the border of the image have less weight. The distance between two regions is a weighted sum of the differences in each of the three moments. The distance between two images is the sum of the distance between the center regions, plus (for each of the

four side regions) the minimum distance of that region to the corresponding region in the other image, when rotated by 0, 90, 180 or 270 degrees. Because the regions overlap, their method is insensitive to small rotations and translations. They also explicitly handle a limited set of rotations.

Pass and Zabih [103] use another approach. They partition histogram bins by the spatial coherence of pixels. A pixel is coherent if it is a part of some "sizeable" similar-colored region, and incoherent otherwise. A color coherence vector (CCV) represents this classification for each color in the image. CCVs are fast to compute and appear to perform better than histograms. The notion of CCV is also extended in [103], by using additional feature(s) to further refine the CCV-refined histogram. One such extension uses the center of the image (the centermost 75% of the pixels are defined as the "center") as the additional feature. The enhanced CCV is called CCV with *successive refinement* (CCV/C) and performs better than CCV.

Hsu et al. [61] attemps to capture the spatial arrangement of the different colors in the image. The image is partioned into rectangular regions using maximum entropy, where each regions is predominantly a single color. The similarity between two images is the degree of overlap between regions of the same color. Hsu presents results from a database with 260 images, which show that their approach can give better results than color histograms. While the authors do not report running times, it appears that Hsu's method requires substantial computation, particularly the partioning algorithm. Additionally, Hsu's algorithm is affected by changes in orientation and position. Their method could be extended to be independent of these effects, at the cost of still greater overhead.

We can also find another important approach to improve initial color histograms. Correlograms are graphs (or tables) showing how autocorrelation changes with distance [62]. Traditionally the distance meant the time distance between pairs of observations. But in this particular case, spatial analysts adapted the idea to spatial distance, and in [62] they adapt the idea to spatial distance of color pixels in an image. A color correlogram expresses how the spatial correlation of color changes with distance. A color histogram captures only the color distribution in an image and does not include any spatial information. Thus, the correlogram is one kind of spatial extension of the histogram. In [62] several applications of correlograms are analyzed. They investigate the applicability of correlograms to tasks such as image subregion querying, object localization, cut detection, and image classification. They also propose the *correlogram intersection* method for the image subregion querying problem and show that this approach yields significantly better results than the traditional histogram intersection method. The histogram backprojection approach used for the localization problem in [137] has serious drawbacks. In [62] it is presented a discussion about all the disadvantages and they introduce the idea of *correlogram correction*. They also show that it is possible to locate objects in images more accurately by using local color spatial information in addition to histogram backprojection using correlograms. In the context of video, they try to help video parsing and browsing using correlograms to compare video frames and detect cuts by looking for adjacent frames that are very different. And once again, they show that using correlograms as feature vectors yields superior results compared to using histograms.

### 2.1.4 Contour Features

Under the name of *contour features* we want to include all the set of features that we can extract from the neighborhood of a given point in an object. As examples of these features, one can find *curves*, *junctions* of two curve segments, *convex regions* or *ellipses*. This repertoire of features is obtained when we already know that a specific point in an object is relevant for us and we want to know how we can model its neighborhood. Usually, these kind of features define various geometric arrangements of intensity edges and it is very common to make use of low level features (straight, circular and/or elliptical segments of intensity edges) and group them to obtain higher-level features.

#### Edge Detection

Canny's method [25] is used to identify image pixels that represent intensity edges. These edges are then grouped to form continuous curve segments, each representing a portion of edge that is fit reasonably well by a straight line, circular arc, or elliptical arc. These curve segments are usually chosen because they can describe many edges well, they can be detected in a stable manner and they represent edges at a level of abstraction that makes them useful for forming higher-level features such as junctions and convex regions.

#### Grouping Edges to Form Curve Segments

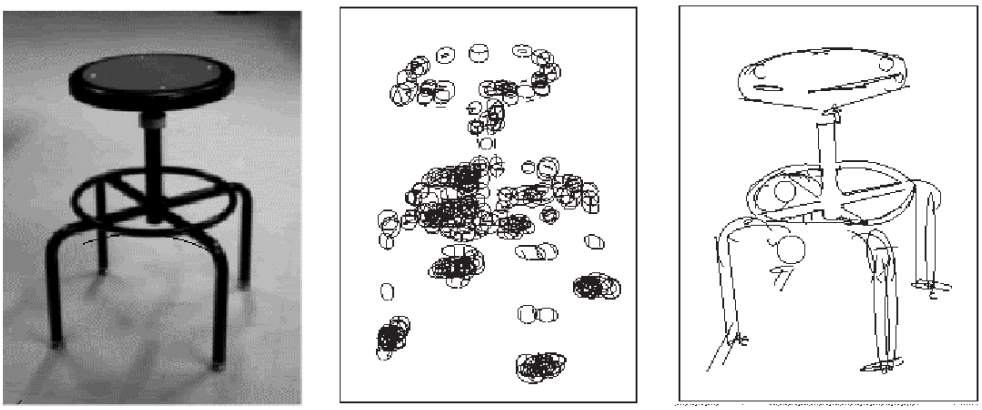
Once a set of edges is obtained, they can be grouped to form curve segments. One of the methods to be used at this specific point is the method of Leonardis [81]. Using this method, grouping occurs in two phases. In the first phase, a large number of curve segments candidates are recovered by subdividing the image into an array of "seed" windows and then "growing" curve segments, starting separately from the edges contained in each window. The group of edges in a seed window are initially fit by a straight line segment using least squares estimation. If an acceptable fit having a sufficiently low error residual is obtained, the group is enlarged by seeking additional edges lying near the endpoints of the line segment. This fit-and-extend process is repeated as long as acceptable fits and additional edges are found. Once the line fit error becomes too large or additional edges cannot be found near the line endpoints, the process switches to circular arcs, which are fit using a least squares formulation. Finally, when circular arcs no longer provide good fits, general conics are fit by solving a generalized eigenvalue problem [138] of the various forms of conic obtained by this method, only single lines, circular arcs, and elliptical arcs are obtained. Among all the acceptable fits obtained during the fit-and-extend process begun from one seed window, the largest line segment, circular arc, and elliptical arc are recorded as candidates.

In the second phase, some of the candidates are selected to become features while the rest are discarded. Initially, all candidates are considered unselected. Selection is then performed by evaluating each unselected candidate, selecting that scoring highest, and repeating this process until no remaining candidate achieves a positive score. One measure used to score candidates is derived from the minimum description

length principle so that it favours candidates that group many edges, fit their edges well, and requires few parameters (i.e., preferring straight lines over circular arcs, and circular arcs over elliptical arcs). The measure also considers pairwise interactions among candidates so that a candidate's score is lowered if some of the edges it groups have already been gropued by another candidate already selected.

This edge grouping method has several qualities that make it a good choice to be selected for this purpose. This method produces curve segments that bridge the small gaps often found in edges detected under near-threshold conditions. It produces a mix of straight, circular, and elliptical segments while choosing the nearly simplest combination adequate in each situation. Weights of the various factors used in scoring candidates can be adjusted to achieve a desired balance between coarse descriptions with few segments, and accurate ones that may break complex curves into many segments. And finally, the method has a high degree of parallelism since with an appropiate hardware it is possible to process each seed window concurrently.

Of course, there are other methods to extract curve segmentations [85, 107, 113]. Once edges and curve segments are found in an image, they can be combined and grouped to obtain higher-level features but this is not the point in this thesis and we do not explain how to obtain this set of features. See [107] for more information about this aspect. We show in figure (2.6) a graphical example extracted from [107] where we show some features (curve segments and parallel curve segments) that can be extracted from a real object.



(a) (b) (c)

**Figure 2.6:** Example of features that can be extracted from an image. (a) Image with an object used to extract its features, (b) Curve segments extracted from the object in (a), (c) L-junctions and parallel curve segments extracted from the object in (a). This graphical example has been extracted from [107].

### 2.1.5 Local Invariants

In this section we want to explain a set of features that can be extracted from images and they can be considered local invariants. The term *local invariant* means that the

image can suffer a transformation such as a rotation, and features keep having the same value under this image transformation. There are features that are invariant to rotations and there are some other ones that can be invariant to changes in the scale. Here we present some of the features used in the literature that are considered local invariants and extensively used in several schemes.

Usually, when we want to obtain invariance under rotations it is common to use Gaussian derivatives (see section 2.1.1) which locally describe an image. Furthermore, Gaussian derivatives can be combined in order to obtain differential invariants [73] which are then inserted into a multi-scale framework in order to deal with scale changes. Therefore the characterisation is invariant to similarity transformations which are additionally quasi-invariant to 3D projections (see [13]).

The image in a neighborhood of a point can be described by the set of its derivatives. Their stable computation is achieved by convolution with Gaussian derivatives (see section 2.1.1). In the computer vision field, this set of derivatives has been named *local jet* by Koenderink [73]. The *local jet* of order $N$ at a point $(x, y)$ for image $I$ and scale $\sigma$ is defined by:

$$J^N[I](x, y, \sigma) = \{L^{\sigma}_{i_1,\ldots i_n}(x, y) | (x, y) \in I; \sigma \in \Re^+; n = 0, \ldots N\} \tag{2.59}$$

where $L^{\sigma}_{i_1,\ldots i_n}(x, y)$ is the convolution of image $I$ with the Gaussian derivatives $G^{\sigma}_{i_1,\ldots i_n}(x, y)$ and $i_k \in \{x, y\}$ (see notation in section 2.1.1). The $\sigma$ of the Gaussian function determines the quantity of smoothing. This $\sigma$ also coincides with a definition of scale-space which will be important if we want to use a multi-scale approach in order to cope with scale changes. In computer vision, $\sigma$ is referred to as the *size* of the Gaussian.

In order to obtain invariance under $2D$ image rotations, differential invariants are computed from the local jet. Differential invariants have been studied theoretically by Koenderink [73] and Romeny et al. [140, 45]. A complete set of invariants can be computed that locally characterises the signal of an image. The set of invariants can be computed up to any order but, usually, up to third order is enough. This set is stacked in a vector, denoted by $V$. Notice that the first component of $V$ represents the average luminance, the second component the square of the gradient magnitude and the fourth the Laplacian,

$$V[0..8] = \begin{bmatrix} L \\ L_i L_i \\ L_i L_{ij} L_j \\ L_{ii} \\ L_{ij} L_{ji} \\ \epsilon_{ij}(L_{jkl} L_i L_k L_l - L_{jkk} L_i L_l L_l) \\ L_{iij} L_j L_k L_k - L_{ijk} K_i L_j L_k \\ -\epsilon_{ij} L_{jkl} L_i L_k L_l \\ L_{ijk} L_i L_j L_k \end{bmatrix} \tag{2.60}$$

with $L_i$ being the elements of the "local jet" and $\epsilon_{ij}$ the 2D antisymmetric Epsilon tensor defined by $\epsilon_{12} = -\epsilon_{21} = 1$ and $\epsilon_{11} = \epsilon_{22} = 0$. We should note that expression (2.60) is given in tensorial notation (the so-called Einstein summation convention).

To be insensitive to scale changes the vector of invariants has to be calculated at several scales. A methodology to obtain such a multi-scale representation of a signal has been proposed in [72].

For a function $f$, a scale change $\alpha$ can be described by a simple change of variables, $f(x) = g(u)$ where $g(u) = g(u(x)) = g(\alpha x)$. For the nth derivatives of $f$, we obtain $f^{(n)}(x) = \alpha^n g^{(n)}(u)$. Theoretical invariants are then easy to derive, for example $\frac{[f^{(n)}(x)]^{\frac{k}{n}}}{f^{(k)}(x)}$ is such an invariant.

However, in the case of a discrete representation of the function, as for an image in our case, derivatives are related by:

$$\int_{-\infty}^{+\infty} I_1(\vec{x}) G_{i_1 \ldots i_n}(\vec{x}, \sigma) d\vec{x} = \alpha^n \int_{-\infty}^{+\infty} I_2(\vec{u}) G_{i_1 \ldots i_2}(\vec{u}, \sigma\alpha) d\vec{u} \tag{2.61}$$

with $G_{i_1,\ldots i_2}$ being the derivatives of the Gaussian.

Equation (2.61) shows that the size of the Gaussian has to be adjusted which implies a change of the calculation support. At it is impossible to compute invariants at all scales, scale quantisation is necessary for a multi-scale approach. Often a half-octave quantisation is used. But, of course, the scale quantisation depends only on the final application and should be selected according to this.

### 2.1.6 Eigenfeatures

In this section we want to introduce the term *eigenfeatures* that is referred to all those feature vectors that are the result of applying the Principal Component Analysis technique. As we will explain later (see section 3.2), the Principal Component Analysis (PCA) technique is used to reduce the dimensionality of a given problem and, at the same time, keeping the maximum amount of the original information in the reduced space.

Instead of working with a feature vector that usually is represented in a high dimensional space one might need real time when performs some object recognition task. In the particular case of working in high dimensional spaces and requiring real time object recognition, it is quite impossible to achieve this goal. Turk and Pentland [142] used Principal Component Analysis (PCA) to describe face patterns in a low dimensional appearance space. Murase and Nayar [93] have shown real time recognition of complex $3D$ objects based on PCA of geometrical shape of objects. As seen in these papers and in other ones in the literature, the PCA approach is very appropiate for real time applications because of the low cost of the recognition algorithms.

The point of this section is not to explain the Principal Component Analysis technique but to show different applications of this technique with some feature vectors that are usually used in computer vision. In section 3.2 we can find the definition of PCA and the mathematical derivations of it. The name *eigenfeatures* comes out from the fact that the PCA is based on the main eigenvectors of the covariance matrix of the original data to represent a low dimensional space. For example, if our feature vectors are color histograms, in the literature, the projected vectors, are referred to *eigenhistograms* [147, 19]. If our feature vectors are faces, the projected vectors are called *eigenfaces* [142].