Feature Detectors and Feature Descriptors:
Where We Are Now

J. Bernal, F. Vilariño, J. Sánchez

Computer Vision Center and Computer Science Department UAB
Campus UAB, Edifici O, 08193, Bellaterra, Barcelona, Spain
E-mail:jbernal@ccc.uab.es

September 6, 2010
List of Figures

1.1 Stages in Pattern Recognition Processes .................................. 4
2.1 Classification of Feature Detection Methods ................................ 6
2.2 Simulation of the use of several sizes of grid ............................. 8
2.3 Comparison of the Output of Several Edge Detectors ...................... 11
2.4 Comparison of Application of Several Harris Detectors ................... 13
2.5 Fourier series of a square wave and the sum of the first four terms ... 14
2.6 Phase Congruency Detector ................................................. 15
2.7 Example of the use of SUSAN .............................................. 16
2.8 Example of the output of FAST detector ................................... 17
2.9 Example of gray-level blobs ................................................. 17
2.10 Application of Difference of Gaussians .................................... 19
2.11 Use of Laplacian ............................................................ 20
2.12 Laplacian of Gaussian ...................................................... 21
2.13 Application of several versions of Hessian Detectors ..................... 22
2.14 Example of application of MSER ......................................... 23
2.15 Regions Detected by PCBR ................................................. 24
2.16 Colour Saliency Maps ...................................................... 25
2.17 Intensity Extrema-based Region Detector .................................. 26
3.1 Taxonomy of Feature Descriptors ............................................ 31
4.1 Convex Hull example ....................................................... 33
4.2 Distance to the Centroid .................................................... 34
4.3 Example of curves and their evolution .................................... 35
4.4 Curvature Scale Space image .............................................. 36
4.5 Wavelet transform coefficients ............................................ 37
4.6 Comparison of reconstruction between Wavelet and Fourier-based
  Shape Descriptors .......................................................... 37
4.7 Minimum Boundary Circle-based approximation of an image ........... 38
4.8 Example of Chain Codes ................................................... 38
4.9 Polygon Decomposition example ......................................... 39
4.10 Shape Context .............................................................. 41
4.11 Shape Context and Chamfer comparative example ....................... 42
4.12 Blurred Shape Model example ........................................... 43
4.13 Shape Matrix ............................................................... 44
4.14 Grid-based Descriptor ..................................................... 45
4.15 Angular Radial Partitioning example .................................... 47
4.16 Angular Radial Transform basis functions ................................ 48
<table>
<thead>
<tr>
<th>Section Number</th>
<th>Section Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.17</td>
<td>Delaunay triangulation</td>
<td>49</td>
</tr>
<tr>
<td>4.18</td>
<td>Skeleton of an image</td>
<td>50</td>
</tr>
<tr>
<td>5.1</td>
<td>RGB colour triangle</td>
<td>55</td>
</tr>
<tr>
<td>5.2</td>
<td>HSL and HSV Colour Spaces</td>
<td>56</td>
</tr>
<tr>
<td>5.3</td>
<td>CIELab Colour Spaces</td>
<td>57</td>
</tr>
<tr>
<td>5.4</td>
<td>Simulation of Dominant Colour</td>
<td>58</td>
</tr>
<tr>
<td>5.5</td>
<td>Hue Colour Circle</td>
<td>59</td>
</tr>
<tr>
<td>5.6</td>
<td>HMMD Colour Space</td>
<td>60</td>
</tr>
<tr>
<td>5.7</td>
<td>Opponent Colour Space</td>
<td>60</td>
</tr>
<tr>
<td>5.8</td>
<td>Histogram normalization</td>
<td>62</td>
</tr>
<tr>
<td>6.1</td>
<td>Difference of Gaussians in SIFT</td>
<td>65</td>
</tr>
<tr>
<td>6.2</td>
<td>Keypoints detection in SIFT</td>
<td>65</td>
</tr>
<tr>
<td>6.3</td>
<td>Gradient orientation histogram</td>
<td>66</td>
</tr>
<tr>
<td>6.4</td>
<td>Use of PCA-SIFT</td>
<td>67</td>
</tr>
<tr>
<td>6.5</td>
<td>Box filters used in SURF</td>
<td>68</td>
</tr>
<tr>
<td>6.6</td>
<td>SURF descriptor</td>
<td>69</td>
</tr>
<tr>
<td>6.7</td>
<td>Set of Gabor filters</td>
<td>70</td>
</tr>
<tr>
<td>6.8</td>
<td>Application of Gabor filters</td>
<td>71</td>
</tr>
<tr>
<td>6.9</td>
<td>Example of regularity</td>
<td>71</td>
</tr>
<tr>
<td>6.10</td>
<td>Homogeneous Texture Descriptor</td>
<td>72</td>
</tr>
<tr>
<td>6.11</td>
<td>Edge Histogram Descriptor</td>
<td>73</td>
</tr>
<tr>
<td>6.12</td>
<td>Example of Radon Transform</td>
<td>75</td>
</tr>
<tr>
<td>6.13</td>
<td>Example of Hough Transform</td>
<td>76</td>
</tr>
<tr>
<td>6.14</td>
<td>Example of use of Steerable Filters</td>
<td>77</td>
</tr>
<tr>
<td>6.15</td>
<td>Dissociate Dipoles</td>
<td>78</td>
</tr>
<tr>
<td>6.16</td>
<td>Haar-like structural elements</td>
<td>79</td>
</tr>
<tr>
<td>6.17</td>
<td>Face Detection Algorithm</td>
<td>80</td>
</tr>
<tr>
<td>6.18</td>
<td>RIFT descriptors</td>
<td>80</td>
</tr>
<tr>
<td>6.19</td>
<td>Local Binary Pattern features</td>
<td>81</td>
</tr>
<tr>
<td>6.20</td>
<td>Co-Occurrence Matrix</td>
<td>82</td>
</tr>
<tr>
<td>7.1</td>
<td>Three-step Block Matching search method</td>
<td>87</td>
</tr>
<tr>
<td>7.2</td>
<td>Optical Flow example</td>
<td>91</td>
</tr>
<tr>
<td>7.3</td>
<td>Angular Circular Motion</td>
<td>92</td>
</tr>
</tbody>
</table>
Abstract

Feature Detection and Feature Description are clearly nowadays topics. Many Computer Vision applications rely on the use of several of these techniques in order to extract the most significant aspects of an image so they can help in some tasks such as image retrieval, image registration, object recognition, object categorization and texture classification, among others. In this paper we define what Feature Detection and Description are and then we present an extensive collection of several methods in order to show the different techniques that are being used right now. The aim of this report is to provide a glimpse of what is being used currently in these fields and to serve as a starting point for future endeavours.
7 Motion Descriptors

7.1 Introduction

7.2 Review of Existing Methods

7.3 Discussion

8 Conclusions
Chapter 1

Introduction

If we think about some of nowadays hottest topics in Computer Vision, such as image retrieval or object recognition, it is almost impossible not to think about using features. Features can be thought as significant properties of an object than can be used as part of input to processes that lead to distinguish the object from other objects or to recognize it. Thus, it is important to know which parts of the object are relevant and characteristic of it, and then describe them in a way that we can enhance the differentiable properties. Imagine that we want to detect some specific person in an image. The first step could be to separate possible candidates from things that are not persons. Hence, it could be necessary to detect face-shape forms in the image. Once we have detected these shapes (and considering only that part of the image) it is important to extract as much information as we can (colour and position of the eyes - Colour Descriptors, presence or not of beard - Texture Descriptors and even the general aspect of the whole face - Shape Descriptors). Although this example may seem an easy one, it shows that in fact Feature Detection and Feature Description are easy-to-recognize tasks. Before starting to develop the theory behind those processes, let us introduce one field where these techniques are commonly used: Pattern Recognition Processes.

Based on [1], Pattern Recognition processes can be thought as a three-steps processes (Transducer, Preprocessor and Classifier, see Figure 1.1), where each one can be considered as an independent component. But there are some doubts about whether the recognition occurs before the eyes are directed to the object because we would not look at it if we were not interested on it. So our eyes are doing what we could call as searching for 'regions of interest', which could be the action that the transducer would have to perform. Some authors [1] think that both transduction and preprocessing stages can be integrated into a whole process, being the result more meaningful. While in this report we will not cover classification aspects, it is also a field that has seen a great development in last decades, although now there is a trend that states that a classifier is good if its input is good, focusing more in Feature Detection and Description.

There is no clear theory that allows us to choose which features are relevant for a particular problem, only a correct definition of a problem and a correct observation of the material we have would lead us to consider one group of techniques and discard others. In this technical report we will cover many techniques for both Feature Detection and Description, taking into account that,
although some of them are more used than others, each technique presented could be useful when facing a new problem and yet the great majority of them are being used today (and there are still research lines centred on them).

The structure of this report is as it follows. First we will cover Feature Detection, starting from defining what it is and then reviewing the different techniques that appear in the literature. As Feature Description is an even larger field, we will divide its study according to four general classes: *Shape*, *Colour*, *Texture* and *Motion*. For this reason, in chapter 3 we will introduce the concepts associated to Feature Description and we will present a taxonomy. Finally, from chapters 4 to 7 we will focus on each type of Feature Descriptor before closing with the conclusions.
Chapter 2

Feature Detection

2.1 Introduction

Feature Detection is a compulsory step to do in order to obtain Feature Descriptors. This family of processes locates points and regions, and they are generally capable of reproducing similar performance that the human would provide in locating elemental features in images. But...what is a feature?

There is no universal or exact definition of what constitutes a feature, and the exact definition often depends on the problem or the type of application where we want to use them. Given that, features can be defined as singular visual traits, associated to the visual primitives that constitute an object, such as edges, corners or lines among others. Since features are used as the starting point and main primitives for subsequent algorithms, the overall algorithm will be as good as the Feature Detector is, and hence, is the relevance of this step. Consequently, the desirable property of a Feature Detector is repeatability: whether or not the same feature will be detected in two or more different images of the same scene.

Feature Detection is often a low-level image processing operation. It is usually performed as the first operation on an image, and it examines every pixel to assess the presence of a feature at that pixel. There are several Feature Detection methods and we have divided them into four main groups: Edge Detectors, Corner Detectors, Blob Detectors and Region Detectors.

In order to fix basic concepts, we define edges as points in a digital image at which the image brightness changes sharply or more formally, has discontinuities. A corner can be defined as the intersection of two edges or as a point for which there are two dominant and different edge directions in a local neighborhood. Blobs are regions in the image that are either brighter or darker than the surrounding. Finally we can define a Region of Interest (ROI) as an area of the image with uniform distribution that is either limited by contours or a significant change in some of its characteristics such as colour or texture. In Figure 2.1 we can see our schema of the Feature Detection Methods. In the next section we will present the general characteristics of some of the most well-known methods.
Figure 2.1: Classification of Feature Detection Methods
2.2 Review of Existing Methods

2.2.1 Basic Preprocessing Techniques
Before entering into detail about the several types of Feature Detectors we will explain two techniques that, although they are not considered Feature Detectors by themselves, they are being used as the first step in some Pattern Recognition processes.

Random Patches
This is the most basic preprocessing technique and for this reason, it can be considered as a complementary option, but not as the main one.

Algorithm: This detector samples a random pixel position of the image and extracts a random patch around it. It can be seen that if we take a large number of random initial pixel positions, we can gather enough information to describe the image, but the performance of the algorithm is based on pure chance.

Grid-based Patches
Grid-based Patches can be seen as an evolution of the previous Random Patches procedure.

Algorithm: This method calculates features in a grid evenly separated along the horizontal and vertical dimensions of the image. At each point of the grid an homogeneous sized patch, which can be thought as a cell from a grid, is extracted from the picture.

Using this method we can detect features with different degrees of depth. Using a bigger size of the cell we will get less patches, but bigger ones, which can represent general information of the image (colour distribution or general shapes, only to mention a few) and by using smaller cells we can gather more detailed information from the image. For this reason, one possible implementation could consist of using two different Grid-based Detectors with different grid size, in order to have further different degrees of description, as it can be seen in Figure 2.2

2.2.2 Edge Detectors
Gradient
The simplest way to detect edges in an image consists of identifying those points with a great change in the intensity gradient of the image.

Algorithm: Taking this into account, if we take the derivative of the intensity values along the image and detect the points where its value is maxima we can make a first approximation of the edges in the image. The gradient of a scalar function \( f(x_1, x_2, x_3, ..., x_n) \) is denoted \( \nabla f \) and it is defined as the vector whose components are the partial derivatives of \( f \), that is:

\[
\nabla f = \left( \frac{\partial f}{\partial x_1}, ..., \frac{\partial f}{\partial x_n} \right).
\]

It is clear that it is the simplest and weakest approximation of an edge detector. It is very common to see a gradient operation followed by a thresholding process in order to get a more informative representation of the image. Gradient by itself is also very sensitive to noise.
Gaussian-kernels based Edge Detectors

One important group of Edge Detectors consists of applying different Gaussian kernels to an image in order to approximate the gradient of the image intensity function. Belonging to this group we have Prewitt, Roberts and, perhaps the most well-known of all of them, the Sobel Edge Operator.

a. Prewitt Edge Detector

Prewitt Edge Detector [2] is an algorithm which calculates the maximum response of a set of convolution kernels to find the local edge orientation for each pixel. There is not a unique kernel that can be used for this operation. The whole set of 8 kernels is produced by taking one of them and then rotate its coefficients circularly. Each of the resulting kernels is sensitive to an edge orientation ranging from 0° to 315° in steps of 45°, where 0° will correspond to a vertical edge.

Algorithm: The maximum response for each pixel will be the value of the corresponding pixel in the output magnitude image. The values for the output orientation image lie between 1 and 8, depending on which of the 8 kernels was responsible of the maximum response. The operator uses two $3 \times 3$ kernels which are convolved with the original image to calculate approximations of the derivatives for both horizontal and vertical changes. We will define $I$ as the source image and $G_x$ and $G_y$ are two images which at each point contain the horizontal and vertical derivative approximations, the computations are as follows:
The Prewitt Edge Detector is an appropriate way to estimate both the magnitude and orientation of an edge and it avoids the excess of computing directly the gradient by using the Gaussian-kernel Based approximation. Source code for the Prewitt Edge Detector is already implemented in Matlab [3].

b. Roberts Edge Detector

The only difference between Prewitt’s and Roberts’ Edge Detector [4] relies on the Gaussian-kernel mask that is used. In the case of Roberts’ approach the mask is the following:

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ 0 & -1 & 0 \\ +1 & 0 & -1 \end{bmatrix} \ast f \quad \text{and} \quad G_y = \begin{bmatrix} 0 & +1 & +1 \\ -1 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \ast f.$$

The Roberts Edge detector is fast since the filter is small but it is also subject to interference by noise. If edges are not very sharp the filter will tend not to detect the edge. It is also already implemented in Matlab being a possible parameter value for the 'edge' function [3].

c. Sobel Edge Detector

The Sobel operator [5] is one of the most well-known Edge Detection algorithms. It is again a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. Because of this, at each point in the image, the result of this operator is either the corresponding gradient vector or the norm of this vector.

**Algorithm:** To describe it in a simple way, the operator calculates the gradient of the image intensity at each point, giving as result the direction of the largest possible increase from light to dark and the rate of change in that direction. The result therefore measures the abruptness or smoothness in the change of image intensity at that point, which can be thought as 'edginess' of that part of the image and how it is oriented. In order to make this visible, the result of the Sobel operator at an image point which is in a region of constant intensity is a zero vector and at a point on an edge is a vector which points perpendicularly to the edge, from darker to brighter values.

As happened with Prewitt and Roberts Edge Detectors the difference relies on the mask used. In the case of Sobel Edge Detectors, the mask is the following:

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \ast f \quad \text{and} \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \ast f.$$

The x-coordinate value is considered as increasing towards the right-hand side and the y-coordinate value as increasing towards the bottom. At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude $G = \sqrt{G_x^2 + G_y^2}$. We can also calculate the gradient’s direction $\Theta = \arctan \left( \frac{G_y}{G_x} \right)$ where for example $\Theta$ is 0 for a vertical edge that is darker on the left side.
Sobel operator, although several decades old, is still considered as reference in edge detection and some of the sub-methods used are part of some other algorithms (i.e. gradient calculation). It is also implemented in Matlab as an available parameter of the 'edge' function [3].

**Canny Edge Detector**

Canny’s aim was to discover the optimal *Edge Detection* algorithm [6]. In this situation, an 'optimal' *Edge Detector* means: that it marks as many real edges as possible, that the marked ones are as close as possible to the real edges, and noise effects are avoided. To do so, Canny’s algorithm uses the calculus of variations, which is a technique that finds the function which optimizes a given functional. The optimal function in Canny’s detector is described by the sum of four exponential terms, but it can be approximated by the first derivative of a Gaussian.

*Algorithm*: To reduce noise, it uses a filter based on the first derivative of a Gaussian, giving as result a slightly blurred version of the original image, not affected by a possible single noisy pixel. Since an edge may point out in a variety of directions, the Canny algorithm uses four filters to detect horizontal, vertical and diagonal edges in the blurred image. The *Edge Detection* operator returns a value for the first derivative in the horizontal and vertical directions. From this, the edge gradient and direction can be determined. The edge direction angle is rounded to one of four angles representing vertical, horizontal and the two diagonals (0, 45, 90 and 135 degrees for example). Later, from a stage referred to as non-maximum suppression, a set of edge points, in the form of a binary image, is obtained. These are sometimes referred to as “thin edges”.

Then, after thresholding is complete (using also an hysteresis process) we will have a binary image where each pixel is marked as either an edge pixel or a non-edge pixel. The reliability of this method can be seen by observing that it is still been used in actual research.

In Figure 2.3 we can see the result of applying the three presented edge detectors to the same image.

### 2.2.3 Corner Detectors

A corner can be defined as the intersection of two edges. A corner can also be defined as a point for which there are two dominant and different edge directions in a local neighborhood of the point. Also, an interest point is a point in an image which has a well-defined position and can be robustly detected.

In practice, most so-called *Corner Detection* methods detect interest points (points where interesting features may be present) in general rather than corners in particular. As a consequence, if only corners are to be detected it is necessary to do a local analysis of detected interest points to determine which of these are real corners. In this subsection we will cover some corner detection algorithms.

**Moravec corner detection algorithm**

This is considered by many authors as one of the first *Corner Detection* algorithms [7]. It uses a definition of corner as a point with low self-similarity and tests each pixel in the image to find out if it can be a corner or not.
Algorithm: It considers how similar a patch centred on the pixel is to near (and overlapping) patches. If the pixel is in a region of uniform intensity, then the nearby patches will look like similarly. If the pixel is on an edge, then nearby patches in a direction perpendicular to the edge will look quite different, but nearby patches in a direction parallel to the edge will result only in a small change.

As pointed out by the author, the main problem with this operator is that it is not isotropic and hence, if an edge is present and it is not in the direction of the neighbors, therefore it will not be detected as a corner.

Harris Matrix-Based Corner Detectors

a. Harris Corner Detector

Harris and Stephens [8] improved Moravec’s Corner Detector by directly considering the differential of the corner score with respect to direction.

Algorithm: If we use a 2-dimensional image (that we will call $I$) and let $I_x$ and $I_y$ be the derivatives of $I$ respect to $x$ and $y$, we can calculate the Harris matrix as:

$$
\begin{pmatrix}
\langle I_x^2 \rangle & \langle I_{xy} \rangle \\
\langle I_{xy} \rangle & \langle I_y^2 \rangle
\end{pmatrix}
$$
The angle brackets denote averaging around an area. If we use a circular window then the response will be isotropic solving one of the problems of Moravec’s approach. By analysing the magnitude of the eigenvalues ($\lambda_1$ and $\lambda_2$) of the Harris matrix, we can find out if a concrete pixel has or not features of interest.

- If $\lambda_1$ and $\lambda_2$ are low and near to zero, then this pixel ($x,y$) has no features of interest.
- If $\lambda_1$ is near to zero and $\lambda_2$ has some large positive value, then an edge is found.
- If $\lambda_1$ and $\lambda_2$ have large positive values, then a corner is found.

It is one of the most used corner detectors and it is the basis of the two methods that we are presenting now. Matlab Code can be found at [9].

b. **Harris-Laplace Detector**

*Algorithm:* In this case the second moment matrix (also called structural tensor) is calculated. It means that we have to calculate the derivatives of the image ($I_x$ and $I_y$) in image domain and also the sum of every non-linear combination of those derivatives in local environments. The Harris-Laplace detector [10] uses a scale-adapted Harris function to localize points in space and then selects the points for which the Laplacian of Gaussian attains a maximum over scale.

The algorithm consists of two steps: a multi-scale point detection and an iterative selection of the scale and the location. The Harris-Laplace approach provides a compact and representative set of points which are characteristic in the image and also in the scale dimension. A result of the application of a Harris-Laplace Detector can be seen in Figure 2.4, where we can see the difference between the output of both versions of Harris Corner Detector. In the image we can see the corners detected represented as ellipses and we can observe that the radius is bigger for the Harris-Laplace approach which can be thought as some degree of impreciseness on the exact location of the corner. Source code can be downloaded from [11].

c. **Harris-Affine Detector**

Harris-Affine detector relies on interest points detected at multiple scales using the Harris corner measure on the second-moment matrix.

*Algorithm:* It is really a modification of the Harris-Laplace detector, following these steps: [12] First it is necessary to identify the initial points which we will use to localize the region of interest by using the invariant scale detector Harris-Laplace. For each initial point the region is normalized to make it affine invariant by using an affine shape adaptation. Iteratively, the affine region is estimated, selecting both the correct integration and differentiation scales and by locating spatially the interest point. Then, the affine region is updated by using the previously calculated scales and spatial locations. This process is repeated iteratively until the chosen stop criteria is reached.

The Harris-Affine detector can identify similar regions between images that are related through affine transformations and have different illuminations. These
affine-invariant detectors should be capable of identifying similar regions in images taken from different viewpoints that are related by a simple geometric transformation: scaling, rotation and shearing. A result of its implementation (that can be found in [13]) can be seen in Figure 2.4.

Figure 2.4: Application of Harris Detectors (left: Harris-Laplace, right: Harris-Affine[14].

d. Shi and Tomasi

Also known as Kanade-Tomasi Corner Detector, it is based in the previously mentioned Harris corner detector but with some small differences. The authors of this method indicate that in case of image patches under affine transformations, it is better to use $\min(\lambda_1, \lambda_2)$ as corner strength measure than using the multi-scale Harris corner measure. Code can be found at [15].

Phase Congruency Detector

Rather than assuming that a feature is a point of maximal intensity gradient, the Local Energy Model [16] postulates that features are perceived at points in an image where the Fourier components are maximally in phase as shown in Figure 2.5.

Notice how the Fourier components are all in phase at the point of the step in the square wave. Congruency of phase at any angle produces a clearly perceived feature. The measurement of phase congruency at a point in a signal can be seen geometrically in Figure 2.6. The local, complex-valued, Fourier components have an amplitude $A_n(x)$ and a phase angle $\phi_n(x)$ at each location $x$ in the signal. Figure 2.6 plots these local Fourier components as complex vectors adding head to tail. The magnitude of the vector from the origin to the end point is the Local Energy, $|E(x)|$. The measure of phase congruency is:

$$PC_1(x) = \frac{|E(x)|}{\sum_n A_n(x)}$$

Under this definition, phase congruency is the ratio of $|E(x)|$ to the overall path length taken by the local Fourier components in reaching the end point. If all the Fourier components are in phase all the complex vectors would be aligned and the ratio would be 1. If there is no coherence the ratio falls to a minimum of 0. Phase congruency provides a measure that is independent of the overall
magnitude of the signal making it invariant to variations in image illumination and/or contrast. Fixed threshold values of feature significance can then be used over wide classes of images.

This method performs better than the simple Harris Corner Detector, as it can be seen in [16], and it is also good in edge detection, because the phase congruency edge map includes the corner map (which is a strict subset of the phase congruency edge map) which greatly simplifies the integration of data computed from edge and corner information. This facilitates the process of building a model of the scene from point and edge data matched over two or more views. In contrast, if the edge and corner information is computed via separate means, say the Canny and Harris operators respectively, the edge data is unlikely to include the corner data. Source code can be found at [17].

Smallest Unvalue Segment Assimilating Nucleus

Algorithm: For Feature Detection, Smallest Unvalue Segment Assimilating Nucleus (SUSAN) [18] places a circular mask over the pixel to be tested (the nucleus). Every pixel is compared to the nucleus using the comparison function that is proportional to the difference of intensity between a point in the region covered by the mask and the nucleus. This function has the appearance of a smoothed top-hat or rectangular function.

SUSAN returns the number of pixels in the mask that are within a previously decided radius. It also uses a threshold (called geometric threshold) that makes SUSAN operator give positive score only if the area is small enough. The final SUSAN region can be locally found using non-maximal suppression, and this is the complete SUSAN operator. The value of the radius determines how similar
Figure 2.6: Polar diagram showing the Fourier components at a location in the signal plotted head to tail. The weighted mean phase angle is given by $\overline{\phi}(x)$. The noise circle represents the level of $E(x)$ one can expect just from the noise in the signal[16].

the points within the mask and the nucleus are, in order to consider them to be part of the univalue segment. The value of the threshold determines the minimum size of the univalue segment. If this value is large enough, then this becomes an edge detector.

For corner detection, two further steps are used. First, we have to find the centroid of the SUSAN because a proper corner will have the centroid far from the nucleus. The second and final step includes into the SUSAN those points on the line from the nucleus through the centroid out to the edge of the mask. In its one-dimensional version, it can be used to detect edges. An example of the results it provides can be seen in Figure 2.7. Source code in C can be found in [19].

**Features from Accelerated Segment Test**

This detector, also known as FAST, [20] considers the pixels under a Bresenham circle of radius $r$ around the interest point. The original detector classifies a point $p$ as a corner if there exists a set of $n$ contiguous pixels in the circle which are all brighter than the intensity of the candidate pixel ($I_p$) plus a threshold $t$, or all darker than $I_p - t$.

*Algorithm:* Initially $n$ is chosen to be twelve because it leads to a higher
speed process that examines pixels 1 and 9. If both of these are within $t$ then $p$ can not be a corner. If $p$ can still be a corner, pixels 5 and 13 are examined. If $p$ is a corner then at least three of these must all be brighter than $I_{p} + t$ or darker than $I_{p} - t$. If neither of these is the case, then $p$ cannot be a corner.

This detector exhibits high performance, but there are several weaknesses. If $n < 12$ it does not reject many candidates, and the efficiency of the detector will depend on how we distribute corner appearances, but still it is a fast and reliable corner detection method. It is often used combined with ID3 trees to optimize the order in which pixels have to be tested. Source code can be found in [21] whereas an example of what kind of features it detects can be seen in Figure 2.8.

**High-speed Corner Detector**

It is not really a completely new Corner Detection method but an evolution of FAST. It uses Machine Learning methods [20] to accelerate Corner Detection using, for the detection step, a process very similar to FAST. In this case it uses a circle of radius 3 and compares nearby pixels to decide whether both are corner candidates or not. In order to concrete the decision it uses non-maximum suppression.
2.2.4 Blob Detectors

In the area of Computer Vision, Blob Detection is related to visual modules that are aimed at detecting regions in the image that are either brighter or darker than the surrounding. In this subsection we will cover some Blob Detectors from the well-known Hessian Detector to more recent ones like the MSER detector.

Gray-level Blobs

One natural approach to detect blobs is to associate a bright (dark) blob with each local maximum (minimum) in the intensity landscape [23]. As a general result of the process, we can think of having something like what is shown in Figure 2.9. But the problem is that local extrema are very sensitive to noise. To address this problem it was studied how to detect local maxima with extent at multiple scales in scale-space.

Algorithm: To adapt the method [23] to scale-space, a region with spatial extent defined from a watershed analogy was associated with each local maximum, as well as a local contrast defined from a so-called delimiting saddle point. A

Figure 2.8: FAST Feature detection in an image patch. The highlighted squares are the pixels used in the feature detection. The pixel at C is the centre of a detected corner: the dashed line passes through 12 contiguous pixels which are brighter than C by more than the threshold[22].

Figure 2.9: Example of Blob detection[24].
local extrema with extent defined in this way was then referred to as a grey-level blob. Still there were more approaches around this idea.

By proceeding with the watershed analogy beyond the delimiting saddle point, a grey-level blob tree can be defined to capture the nested topological structure of level sets in the intensity landscape, in a way that is invariant to affine deformations in the image domain and monotone intensity transformations. By studying how these structures evolve with increasing scales, the notion of scale-space blobs was introduced.

The interesting aspect of this family of methods is that the regions of interest and scale descriptors obtained from them can be used in a later stage for guiding other early visual processing tasks.

As an example, such regions of interest and scale descriptors were used for directing the focus-of-attention of an active vision system [23]. In this way, it can be used maybe not as a main detector, but as a first step in a Feature Detection chain.

Gaussian-based Detectors

Gaussian functions are used as smoothing kernels for generating multi-scale representations. In this subsection we will explore several methods that are somewhat related to Gaussian kernels, starting from the simple Difference of Gaussians to the more advanced Laplacian of Gaussians. Before entering into details on the Laplacian of Gaussian (LoG) blob detectors we will explain briefly some basic concepts that can help to fully understand the LoG method although they are not blob detectors by themselves.

Difference of Gaussians

Algorithm: This is not a Blob Detector by itself but an algorithm that consists of the subtraction of one blurred version of an original grey level image from another less blurred version of the original. The kernel’s calculation can be seen in the formula below:

$$f(x,y,\sigma) = \frac{1}{2\pi\sigma^4}exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \frac{1}{\pi\sigma^4}exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

The blurred images are obtained by convolving the original image with Gaussian kernels using different standard deviations. It is important to mention that blurring an image by using a Gaussian kernel only suppresses high-frequency spatial information and that by subtracting one image from the other one helps to preserve spatial information that lies in the range of frequencies preserved, acting like a band-pass filter.

The Difference of Gaussians [25] can be used to increase the visibility of edges and other details present in a digital image, as it can be seen in Figure 2.10. This algorithm removes high frequency details that often include random noise, so this approach can be considered as one of the most suitable for processing images with a high degree of noise. A major drawback to the application of the algorithm is an inherent reduction in overall image contrast produced by the operation. Differences of Gaussians have also been used for blob detection as part of the SIFT algorithm.
Laplacian operator

The simple use of gradient is effective to detect edges where the grey level of the pixel changes abruptly in little space. But when the grey level changes slowly from dark to bright levels, gradient operator will detect wide edges. In this cases it is better to use the Laplacian operator [26] in order to detect the real edges in an image.

Algorithm: In the case mentioned of wide edges, the second derivative of this edge will have a zero crossing in the middle of the edge. So the localization of this edge can be fixed by calculating the zero crossings of the second order derivative of the image. Mathematically the Laplace operator is defined as the divergence (\(\nabla\cdot\)) of the gradient (\(\nabla\)). So the Laplacian of \(I\) is defined by

\[
\triangle I = \nabla^2 I = \nabla \cdot \nabla I.
\]

Laplacian is not commonly used as a Edge Detector, as it can be seen in Figure 2.11 (that is why it is not included in the proper section) but is part of some other blob detection methods, like LoG or Hessian-Laplace.

Laplacian of Gaussians

One of the first and also most common Blob Detectors is based on the Laplacian of the Gaussian (LoG) [26]. It combines the use of two operators (Laplacian and Gaussian) to improve Blob Detection.

Algorithm: Given an input image \(I(x,y)\), this image is convolved by a Gaussian kernel \(g\) at a certain scale \(t\)

\[
g(x,y,t) = \frac{1}{2\pi t} e^{-\frac{x^2+y^2}{2t}}
\]

to give a scale-space representation \(L(x,y,t) = g(x,y,t) \ast I(x,y)\) and then the Laplacian operator is applied \(\nabla^2 L = L_{xx} + L_{yy}\). This operator usually gives strong positive responses for dark blobs of extent \(\sqrt{t}\) and strong negative responses for bright blobs of similar size. We may face a problem when applying
this operator in single scale because the response of the operator depends on
the size ratio between the size of the blob structure and the size of the Gaussian
kernel, that is why a multi-scale approach can be necessary.

To do so one straightforward method could be applying the scale-normalized
Laplacian operator and then detect scale-space maxima/minima, which are
points that are simultaneously local maxima/minima of the normalized Lapla-
cian operator with respect to both space and scale. So, given a discrete two-
dimensional input image $I(x, y)$ a three-dimensional discrete scale-space volume
is computed and a point is considered as a bright (dark) blob point candidate if
the value at this point is greater (smaller) than the value in all its 26 neighbors.
This method is quite strong in Blob Detection and it is currently being used.
Source code can be downloaded from [28] and one example of its use can be
seen in Figure 2.12, where the result image has been obtained by applying a
Laplacian of Gaussian filter to the original image and then applying a threshold
to create a better representation of the output image.

Hessian-based Detectors

The Hessian matrix (we will refer to it as the Hessian) is the square matrix
of second-order partial derivatives of a function; that is, it describes the local
curvature of a function of many variables.

Determinant of the Hessian

It is again not a Blob Detector by itself but it is used as part of some of them.
By considering the scale-normalized determinant of the Hessian, also referred to
as the Monge-Ampere operator, $detHL(x, y; t)$, where HL denotes the Hessian
matrix of L (which is the Laplacian of Gaussians), and then detecting scale-
space maxima of this operator, we can obtains a straightforward differential
Blob Detection with automatic scale selection which also responds to saddles.
The Hessian matrix of a real-valued function $f(x_1, x_2, ..., x_n)$ can be calculated
as follows:
Figure 2.12: (up) Original Image (down) Laplacian of Gaussian[28].

\[
\begin{vmatrix}
\frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\
\frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\
\cdots & \cdots & \cdots & \cdots \\
\frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n}
\end{vmatrix}
\]

\textbf{a. Hessian-Laplace}

This method chooses interest points where the trace and determinant of the Hessian are maxima. An hybrid operator between Laplacian and Hessian determinant [29] is used to localize points in space at the local maxima of the Hessian determinant and in scale at the local maxima of the Laplacian of Gaussian. Implementation of this algorithm can be found in [30] and an example of its properties can be seen in 2.13 where the blobs are represented as ellipses around the center of the blob.

\textbf{b. Hessian-Affine}

The Hessian-Affine [12] uses a multiple scale iterative algorithm to spatially localize and select scale and affine invariant points. However, at each individual
scale, the Hessian-Affine detector chooses interest points based on the Hessian matrix at that point.

Algorithm: As with the Harris-Affine algorithm, these interest points based on the Hessian matrix are also spatially localized using an iterative search based on the Laplacian of Gaussians and are called Hessian-Laplace interest points and, by using these points, the Hessian-Affine detector uses an iterative shape adaptation algorithm to compute the local affine transformation for each interest point [31]. Source code for both Hessian Laplace and Affine can be taken from [30] and an example of its output can be seen in Figure 2.13.

![Figure 2.13: (left) Hessian-Laplace (right) Hessian-Affine][14].

In image 2.13 we can see the difference in performance between Hessian-Laplace and Hessian-Affine approaches. The area where the detector finds a corner is clearly higher in the Hessian-Laplace image, possibly denoting a fewer degree of precision in corner localization.

Maximally Stable Extremal Regions Detector

The concept of Maximally Stable Extremal Regions (MSERs) was proposed by Matas et al [32]. MSERs denote a set of distinguished regions that are detected in a grey scale image. All of these regions are defined by an extremal property of the intensity function in the region and on its outer boundary. The set of MSERs is closed under continuous geometric transformations and it is invariant to affine intensity changes. Furthermore, MSERs can be detected at different scales.

Algorithm: Extremal regions have two desirable properties. The set is closed under continuous (and thus perspective) transformation of image coordinates and, secondly, it is closed under monotonic transformation of image intensities. The concept behind MSERs can be explained informally as follows: We can consider all possible thresholdings of a grey level image I. We can refer to the pixels below a threshold as 'black' and to those above or equal as 'white'. If we were shown a movie of thresholded images $I_{\text{base},t}$, with frame t corresponding to threshold t, we would see first a white image. Subsequently black spots corresponding to local intensity minima will appear and grow. At some point regions corresponding to two local minima will merge. Finally, the last image will be black. The set of all connected components of all frames of the movie is the set of all maximal regions; minimal regions could be obtained by inverting
the intensity of I and running the same process. An example of its output can be seen in Figure 2.14 where we can see the effect of changing the threshold value has in the output image up to the point of losing regions when the threshold is high.

Figure 2.14: Application of MSER with different values of the threshold[33].

This technique has the advantage of making it possible to find correspondences between image elements from two images with different viewpoints. This method of extracting a comprehensive number of corresponding image elements contributes to the wide-baseline matching, and it has led to better stereo matching and object recognition algorithms. An implementation of MSER can be downloaded from [33].

**Principal Curvature-Based Region Detector**

In many object recognition tasks, within-class changes in pose, lighting, colour, and texture can cause considerable variation in local intensities. Consequently, local intensity no longer provides a stable detection cue. One alternative could be to capture semi-local structural cues such as edges and curvilinear shapes, that tend to be more robust to variations in intensity, colour and pose. Principal Curvature-Based Region Detector was developed with these things in mind.

*Algorithm:* The steps that the algorithm consists of are the following: First it detects curvilinear structures (also called ridges) which generate a single response for both lines an edges, creating a clearer sketch of the image (better than the one produced by the gradient). Then it uses Hessian matrix to get the
principal curvature [34]. Following, it searches for characteristics and robustness in scale-space and tries to define regions using a slightly enhanced version of watershed algorithm, cleaning the principal curvatures images by using a morphological closing and a thresholding. Finally, using a procedure similar to MSER, it selects stable regions across local scale changes, measuring the region overlap error along each triplet of consecutive scales. In Figure 2.15 an example of its use can be seen.

Figure 2.15: Regions Detected by PCBR[34].

What makes PCBR interesting is that it takes into account local variations in intensity caused by variation in pose or texture within classes, where this parameter does not give enough information to get an stable detection. So it may be suitable to be used when we have an object that can change its appearance in a short space, in order to improve its detection. In order to try this detector, source code can be found at [35].

2.2.5 Region Detectors

This group of detectors identify ROIs in an image by selecting key-points and then creating patches around them. In this subsection we will cover two of the most important Region Detectors.

Salient Region Detector

Saliency maps can also be used for salient object segmentation by using the segmentation information of the original image [36]. The accuracy of saliency maps plays a crucial role in such segmentation tasks. While the answer to 'what is salient in an image' may be loaded with subjectivity, it is nevertheless agreed upon that certain common low-level processing governs the task independent visual attention mechanism in humans. The process eventually results in a majority agreeing to a region in an image as being salient.

Algorithm: The calculation of this detector [37] is based on the probability density function of the intensity value of the image calculated in a neighborhood of an elliptic region. A summarized version of the process can be this: we calculate, for each pixel, the entropy of the probability density function using three different parameters (scale, orientation and major axis radius). The extrema of the entropy are considered as candidates to be 'salient regions' and they are consequentially sorted by the value of the derivative of the probability density function in respect to scale. This method can be also extended to compute colour cluster saliency in order to extract the attention regions on colour images as it can be seen in [37]. In Figure 2.16 an example of colour saliency maps can be seen.
This method, although it can detect potentially good salient regions, needs that the images to be marked manually to select the good matches, which is not practical in real life (and more if you want some kind of real-time processing). Code can be found in [39].

**Intensity Extrema-based Region Detector**

This method [31] is based on values of intensity in extrema points in different scales. It detects affine covariant regions that start from intensity extrema (detected at multiple scales), and explores the image around them in a radial way, delineating regions of arbitrary shape, which are then replaced by ellipses.

**Algorithm:** More precisely, given a local extrema in intensity, the intensity function along rays emanating from the extrema is studied and evaluated along each ray. The function is the following:

\[
 f_I(t) = \frac{I(t) - I_O}{\max(\frac{\|\text{abs}(t(t) - I_O)\|}{d}, d)}
\]

being \(t\) an arbitrary parameter along the ray, \(I(t)\) the intensity at position \(t\), \(I_O\) the intensity value at the extremum and \(d\) a regularization parameter that prevents from a possible division by zero. The point for which this function reaches an extrema is invariant under affine geometric and linear photometric transformations (given the ray).

Typically, a maximum is reached at positions where the intensity suddenly increases or decreases. Nevertheless, it is necessary to select the points where
the intensity function $f$ reaches an extrema to make a robust selection. Next, all points corresponding to maxima of $f$ along rays originating from the same local extrema are linked to enclose an affine covariant region. This often irregularly-shaped region is replaced by an ellipse having the same shape moments up to the second order. This ellipse-fitting is again an affine covariant construction and it tries to cover the whole image, as it can be seen in Figure 2.17. Source code can be found in [40].

Figure 2.17: Image with Intensity Extrema-based Region Detector applied[31].

2.3 Discussion

As it can be seen, there is a great number of different Feature Detectors, classified according to what they detect. Most of them could have been classified in some other way (for instance, Harris can be tweaked to also detect edges and while Difference of Gaussians is used as part of a blob detector it can also detect edges).

On Edge Detection we have not presented as many options as in the other groups but it does not mean that they are less used. Both Canny [6] and Sobel [5] detectors are commonly used (Gradient is clearly the simplest option of all) and while Canny seems to be more concerned about noise effects, Sobel (and some of its sub-methods) can also bee a good option in certain cases. As we have shown in Figure 2.3 the output that the three of them produces is very different and Sobel seems to give more intuitive results. Another possibility
could be to use Harris as an Edge Detector taking into account that a corner is just a junction of two edges.

In the field of Corner Detection we clearly have two trends. The use of Harris Corner Detection methods has had and still has a great support in the research community, being the affine version [13] the most evolved and efficient. Shi and Tomasi is also put into this group because of being a modification of the Harris Corner Detector which improves its functioning by considering the minimum of the eigenvalues as an indicative of corner strength. The Harris Corner Detector is widely used but, the response from the Harris operator, and other corner operators, varies considerably with image contrast. This makes difficult the setting of thresholds that can be appropriate for long image sequences. In this case the use of Phase Congruency Detector can be very useful [16]. On the other hand, we have two methods (they are really three but one is an enhancement of another) that rely on the use of masks, (SUSAN [18], which uses a circular one and FAST, [20], which uses a Bresenham circle) around a pixel to test if it is a corner or not. Both trends give good results and while Harris right now is predominant, some approaches like FAST are more recent (2008) and their results are promising [20].

On Blob Detection there are clearly two options widely used: Laplacian of Gaussians [26] (maybe because its integration in SIFT) and Hessian-related methods. Hessian-Affine [12] is the most evolved version until now of the Hessian methods and it has been proven to overcome the disadvantages of the Laplacian approach (such as its lack of noise robustness or incorrect edge detection). MSER [32] is useful when we have an object which can change its appearance (by means of change in pose or illumination) and we want to match it between two images and PCBR [32] takes further this approach, considering local variations in short space. Finally Gray-level Blobs can be considered the simplest but cheaper option. So, summarizing, if we want to detect blobs in a simple isolated image, the best option could be to use a Hessian method but if we are facing a task which involves some kind of motion or deformation in objects, both MSER and PCBR should be taken into account.

Finally, talking about Region Detectors, we have covered Salient Region Detector [36] and Intensity-based Region Detector [31] that are less used than the other types because their more general nature that cannot be useful when solving more specific problems. Nevertheless they are interesting and, for example, saliency is a topic that is fair interesting because of its bonds with human visual attention.

It is important to mention that we have not included in the Feature Detection methods neither SIFT or SURF detectors. The reason behind this decision is that both methods are usually seen along with their respective Feature Descriptors and, because of that, they will be explained in the following chapters. A summary of the most important detectors that have been covered can be seen in Tables 2.1 and 2.2:
<table>
<thead>
<tr>
<th>Feature Detector</th>
<th>Basis</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian-kernel Based</td>
<td>Approximation of the gradient of the image intensity function</td>
<td>Fast edge detection</td>
<td>Scale and rotation variance</td>
</tr>
<tr>
<td>Canny</td>
<td>Calculus of variations which finds the function that optimizes a given functional</td>
<td>Fast and simple, noise avoidance</td>
<td>Scale and rotation variance</td>
</tr>
<tr>
<td>Corner detectors</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Harris</td>
<td>Differential of the corner score with respect to direction</td>
<td>Fast, corner and edge detection</td>
<td>Simple, scale and rotation variance</td>
</tr>
<tr>
<td>Harris-Laplace</td>
<td>Use of structural tensor matrix</td>
<td>Scale invariance</td>
<td>Rotation variance, problems with affine regions</td>
</tr>
<tr>
<td>Harris-Affine</td>
<td>Affine invariant normalization of the results from Harris-Laplace</td>
<td>Multi-scale approach, affine regions considered</td>
<td>Some problems with blurred images, rotation and luminance dependant</td>
</tr>
<tr>
<td>SUSAN</td>
<td>Placing of a mask over the pixel to be tested, comparing every pixel to the nucleus</td>
<td>Good corner detection</td>
<td>Computation cost, scale and rotation variance</td>
</tr>
<tr>
<td>FAST</td>
<td>For every pixel under a Bresenham circle classifies it as corner candidate or not according to the brightness of its neighbors</td>
<td>High performance</td>
<td>Illumination, scale and rotation variance</td>
</tr>
</tbody>
</table>

Table 2.1: Summary of Feature Detectors (I)
<table>
<thead>
<tr>
<th>Feature Detector</th>
<th>Basis</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Blob detectors</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DoG</td>
<td>Subtracting one blurred version to the original image</td>
<td>Used in SIFT, preserves spatial information</td>
<td>Supression of high-frequency information, reduction in overall contrast</td>
</tr>
<tr>
<td>LoG</td>
<td>Apply a Gaussian kernel to convolve the image and then applying Laplacian operator</td>
<td>Strong and good results, relatively fast</td>
<td>Problems when applying the operator in single-scale</td>
</tr>
<tr>
<td>Hessian-Laplace</td>
<td>Choose interest points that maximize determinant of Hessian</td>
<td>Hybrid operator, selects point in space with Hessian and in scale with LoG</td>
<td>Problem with affine regions</td>
</tr>
<tr>
<td>Hessian-Affine</td>
<td>Interest points detected at multiple scales using Harris corner measure on the second moment matrix</td>
<td>Adapted to affine regions</td>
<td>Computation cost</td>
</tr>
<tr>
<td>MSER</td>
<td>Detects regions defined by an extremal property of the intensity function in the region and on its outer boundary</td>
<td>Good shape fitting, eases the possibility of finding correspondences from images (of the same scene) with different viewpoints</td>
<td>Scale and rotation variance</td>
</tr>
<tr>
<td>PCBR</td>
<td>Detection of structural cues such as edges and curvilinear shapes that are robust to variations</td>
<td>Use of semi-local properties, adapted to slight changes in pose, lighting or colour</td>
<td>Too much object classification oriented that sometimes cannot be useful</td>
</tr>
<tr>
<td><strong>Region detectors</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Salient Region Detector</td>
<td>Based on the probability density function of the intensity value of the image</td>
<td>Detect good salient regions, extension to colour images</td>
<td>Scale and rotation variance, needs images to be marked individually</td>
</tr>
<tr>
<td>Intensity Extrema-based Region Detector</td>
<td>Detect points based on values of intensity in extrema points in different scales</td>
<td>Multi-scale approach</td>
<td>Illuminance variant (as it is based on intensity values)</td>
</tr>
</tbody>
</table>

Table 2.2: Summary of Feature Detectors (II)
Chapter 3

Feature Description

3.1 Introduction. Definitions

In the previous section we have explained what a Feature Detector is and presented a series of methods. The results of all this methods are a series of points, called key-points or points of interest. Once we know which points are relevant and hold the most distinctive information, it is necessary to describe this information. This step is as necessary as the previous one and, as a result, we will get a Feature Vector which can be used in later stages in many ways: it can be compared to a series of Feature Vectors extracted from objects in a database to perform object recognition or use the information to decide, for example, if an object is a face or not depending on whether it has some compulsory features or not.

As it happens with Feature Detectors, there are many Feature Description algorithms and, of course, mostly all of them are being used currently. Again, the problem that we want to solve will tell us which Feature Descriptors we want to test. Maybe we only need to find out the shape of an object to get to a decision but in more complicated tasks (i.e. classification of images) some other characteristics, like texture or colour can be discriminative as well.

We have divided the existing methods in four general classes: Shape Descriptors (subdivided in Contour-based and Region-based, depending on which part of the object we use to get to a description), Colour Descriptors, Texture Descriptors and Motion Descriptors. In each of the following chapters we will present the most relevant methods in each class but in this chapter we show the general taxonomy of Feature Descriptors as it can be seen in 3.1.
Chapter 4

Shape Descriptors

4.1 Introduction

One of the important characteristic of an object, that can make it different from other objects in an image, is the shape. Nowadays the goal to find a similar object belonging to some database is part of many different processes (it is a basic step in object recognition). Shape is, indeed, an important visual cue and it is one of the most used to describe image content. But it also has its complications because we can not forget that when we project 3-D objects into a 2-D image we are losing one whole dimension of information, so the 2-D representation of an object gives only a partial representation of it. Even shape is a feature that is highly affected by noise, defects, occlusion or deformation, which can make harder the process of object recognition. In this section we will consider that the object whose shape we want to describe has been segmented (or, at least, somehow separated) from the image so we have a binary image patch which contains the object.

Shape Description techniques are usually divided into two groups [41, 42]: Contour-based Descriptors and Region-based Descriptors. The difference consists on whether we extract features only from the contour of the shape or we use the whole shape region. Even more, each group is also subdivided into structural approaches (representing the shape by segments/sections) or global approaches (represent the shape as a whole).

In the next sections we will cover many methods in each category, explaining their principles and possible areas of use.

4.2 Contour-based Shape Descriptors

Contour-based techniques, as mentioned before, extract a Feature Vector from the information gathered from the contour of the shape. Here we will present some methods from two approaches, namely global and structural, without entering in many details, which are available in the bibliography.
4.2.1 Global

Simple Descriptors

Sometimes even the most simple descriptors can give enough information about the shape of an object. Within this group we can include area, eccentricity, axis orientation and the radius of the principal axis, only to mention a few. Although they do not give enough information to describe two similar objects (and point out the differences between them), they can be used to make a first decision. They can be used as a first and easy approach or combined with some other descriptors, but they are rarely used as the only descriptors.

Descriptor: In this case the descriptor will depend on the method we used (area, perimeter, etc.)

Convex Hull

One region is considered as convex [43] only if by taking any two points of it the segment that binds them is inside the region. Convex Hull is defined as the minimal convex region. Before dividing the contour in segments, it is smoothed to avoid some non-desired effects such as hysterical responses to noise. At last the whole shape will be represented as a chain of concavities.

For planar objects, i.e., lying in the plane, the Convex Hull may be easily visualized by picturing out an elastic band stretched open to encompass the given object; when released, it will assume the shape of the required convex hull, as it can be seen in Figure 4.1. It may seem natural to generalize this

![Approximation of a Convex Hull](image)

picture to higher dimensions by imagining the objects enveloped in a sort of idealised unpressurised elastic membrane or balloon under tension. However, the equilibrium (minimum-energy) surface in this case may not be the Convex Hull - parts of the resulting surface may have negative curvature, like a saddle surface. In the case of points in 3-dimensional space, if a rigid wire is first placed between each pair of points, then the balloon will spring back under tension to take the form of the Convex Hull of the points. Using these kind of descriptors but in 3-D can also be interesting, as it can be seen in [45]. Matlab source code for a Convex Hull calculation can be found at [46].
Descriptor: In this case the descriptor will be the convexity [47], which can be defined as the ratio of the perimeters of the Convex Hull of the contour and the original contour.

Shape Signature

This method represents the shape of an object by means of an uni-dimensional function which is extracted from the points belonging to the contour of the shape. There are several possible Shape Signatures such as the centroidal profile, shape radii, complex coordinates, distance to the centroid (as it is shown in Figure 4.2), tangent or accumulative angle, curvature, arc length, etc.

![Figure 4.2: An apple and its centroidal distance Shape Signature][41].

Shape Signatures [41] are usually normalized for translation and scale invariance. In order to compensate for orientation changes, shift matching is needed to find the best matching between two shapes. Most of the signature matching is normalized to shift matching in 1-D space, however, some signature matching requires shift matching in 2-D space, such as the matching of centroidal profiles. In either case, the matching cost could potentially be too high for on-line retrieval.

In addition to the high matching cost, Shape Signatures are sensitive to noise, and slight changes in the boundary can cause large errors in matching. Therefore, it is undesirable to directly describe shape using a Shape Signature. Further processing is necessary to increase its robustness and reduce the matching load. For example, a shape signature can be simplified by quantizing the signature into a signature histogram, which is rotationally invariant.

Descriptor: As it happens with Simple Descriptors, in this case the output of these group of methods will depend on the method used (centroidal profile, distance to centroid, etc.).

Scale-space

By using Scale-space representation noise sensitivity and contour-variations problems are somehow solved. To create Scale-space representation we have to follow the position of the contour pixels (filtered by low-pass variable width Gaussian filters). As we increase the filter’s width non-significant parts or incidences are eliminated from the contour and the shape is smoothed.

One implementation is the Curvature Scale Space (CSS). CSS image representation along with a small number of global parameters are used for this purpose. The CSS image consists of several arch-shaped contours representing the inflection points of the shape as it is smoothed. The maxima of these contours are used to represent a shape. Code for CSS can be found in [48]. More complete information about Curvature Scale Space Images can be found in [49].
In Figure 4.3 we can observe, on the right, a planar curve depicting the shoreline of Africa and then we show several evolved versions of that curve obtained by changing the value of $\sigma$, the standard deviation used in the Gaussian kernel that is utilized to create the different evolved versions of the curve. By increasing the values of $\sigma$ [49] we can remove noise as well as small features from the original contour, leaving only the major shape components. At a sufficiently large value of $\sigma$, all shape structures disappear and the evolved contour becomes convex. Then, in Figure 4.4, we can observe the Curvature Scale Space image. Note that each of the arch-shaped contours in that image corresponds to a feature on the original contour with the size of the arch being proportional to the size of the corresponding feature. At the top of each arch we can observe a small gap, due to the discrete sampling of the CSS image.

**Descriptor:** The descriptor (in the case of CSS) will consist of the coordinates of the inflection points of the shape so the CSS image can be constructed.

**Wavelets Transform**

By using Wavelet Transform [50] we can obtain a hierarchical descriptor of planar curves which is able to decompose a curve in several scales. Higher scale components convey more general information and those with lower scale have more local and detailed information. The descriptor also has some interesting properties like multi-resolution representation, is invariant and it keeps the unicity, stability and spatial location properties. Another option would be to use a deformable wavelet descriptor [50] by interpreting the transform coefficients as...
random variables.

Wavelet Transform descriptors have been applied in applications such as character recognition and contour extraction, based on model comparison among others. One of the drawbacks of the method is that it has a relatively high computation cost. Despite this, they are commonly used in shape contour extraction, both in 2-D and 3-D spaces. In Figure 4.5 we can see the computation of Spherical wavelet coefficients of some 3-D shapes. Matlab information on how to use wavelets can be found at [51].

**Descriptor:** The descriptor will consist of the coefficients of the Wavelet Transform.

---

**Fourier-based**

This is not a *Shape Descriptor* by itself [53, 54], but it acts as one of them because of the way it is implemented. First we need to have a characteristic function of the object that is known as Shape Signature already mentioned above. After this step is when we apply the Discrete Fourier Transform (DFT) to represent this signature in another domain.

Direct representation can capture subtle details of the shape but it is sensitive to noise and slight variations which are somewhat solved in Fourier domain. One of the most used Shape Signatures (in combination with Fourier transform) is shape radii [54] which consists on calculating the distance between uniformly sampled contour points and their centroid or mass center. An implementation of shape radii can be seen in [55] These methods are translation but not rotation or scaling invariant. Sometimes the performance of Fourier-based methods is compared to the one by Wavelets, as it can be seen in Figure 4.6.

**Descriptor:** As in the case of Wavelets Transform-based Descriptor, the Feature Vector will consist of the coefficients of the Fourier Transform applied to the Shape Signature of the object.
Minimum Boundary Circle-based

In the case of Minimum Boundary Circle-based methods, features are extracted from the minimum circle that surrounds the object (that is, the circle that touches its further borders). This process can be better seen in Figure 4.7. The Feature Descriptors that it provides are the following: center coordinates, radius, minimum circle crossing points, angle sequence of these crossing points, vertex angle sequence and angle sequence starting point.

To do so, first a calculation of the minimum circle is needed, then the cited features are computed and the Shape Signature is generated using a combination of these features. At last an unique representation of the object is obtained from the Discrete Fourier Series Expansion of this signature [56].

This method is rotation, translation and scale invariant which makes it a good candidate. In order to calculate the difference between two objects (in order to do a comparison o matching with a model) Euclidean distance is used.

Descriptor: The Feature Vectors will consist of the values of the parameters
that have been mentioned before (center coordinates, radius, etc.).

4.2.2 Structural

Chain Code

This algorithm works by codifying lines into a determinate code. The nodes that surround a certain point (or central node) are enumerated counter-clockwise in ascending order from inside to outside. A chain will consist of an ordered link sequence. The inverse and the length of the chain can also be calculated. An example of possible Chain codes and their implementation into a shape representation can be seen in Figure 4.8.

![Chain Code Examples](image)

Figure 4.8: (up-left) One possible chain code (up-right) Another possible chain code (down) Contour described by using a 4 words chain code[57].

One of the possible disadvantages related to the use of Chain Codes [57] is that it depends on the starting point. One advantage related to the use of
Chain Codes is its translation invariance. We can obtain scale invariance by changing the size of the sampling grid. Rotation invariance can be obtained by using an enhanced version called Difference Chain Code, where each element is calculated by the subtraction of one element and its precedent. A complete Chain Code shape description algorithm can be downloaded from [58].

Descriptor: In this case the Feature Descriptor will be composed of the codification of the shape structural elements, following the Chain Code that was used.

Polygon Decomposition

In Polygon Decomposition [59], the contour of the shape is divided in approximated segments or polygons, using as primitives the vertex of these polygons. For each primitive the feature extracted consists of a chain of four elements (inner angle, distance to the next vertex and \(x\) and \(y\) coordinates). An example of a Polygon Decomposition of a shape can be seen in Figure 4.9.

![Figure 4.9: Partition of a Shape into trapezoids and triangles][59].

It is not translation, scale or rotation invariant. In order to calculate the similarity between two shapes described by this method, Edit Distance is used. Edit Distance between two strings of characters is the number of operations required to transform one of them into the other. More information about Polygon Decomposition can be found at [59].

Descriptor: The descriptor will consist of the four elements that have been mentioned before.
Shape Invariants

Shape invariants do not depend on the viewpoint and represent properties that
do not vary even if the object is under a series of transformations. Invariants
theory [60] is based on a collection of transformations that can be linked and
inverted. Examples of invariants are: cross ratio, ratio length, distance, angle,
area and triangle. Here we will explain what Similarity and Projective Invariants
are [60].

Similarity Invariant

For image locations \((x_1, y_1), (x_2, y_2), (x_3, y_3)\), \(g_E()\) can be defined as a
function which does not change even if the points undergo any 2-D translation,
rotation and scaling transformation, applying the similarity invariant:

\[ g_E((x_1, y_1), (x_2, y_2), (x_3, y_3)) = \theta \]

where \(\theta\) is the angle at image coordinate \((x_1, y_1)\) between line \((x_1, y_1)(x_2, y_2)\)
and \((x_1, y_1)(x_3, y_3)\).

Projective Invariant

For the projective case, geometric properties of the shape of an object should
be invariant under a change in the point of view. From the classical projective
geometry we know that the so called cross-ratio is independent of the projection
viewpoint. We can define the projective invariant \(g_P()\) as

\[ g_P((x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)) = \frac{\sin(\theta_1+\theta_2)\sin(\theta_2+\theta_3)}{\sin(\theta_2)\sin(\theta_1+\theta_2+\theta_3)} \]

where \(\theta_1, \theta_2, \theta_3\) are the angles at image coordinate \((x_1, y_1)\) between \((x_1, y_1)(x_2, y_2)\)
and \((x_1, y_1)(x_3, y_3)\), \((x_1, y_1)(x_3, y_3)\) and \((x_1, y_1)(x_4, y_4)\), \((x_1, y_1)(x_4, y_4)\) and
\((x_1, y_1)(x_5, y_5)\) respectively.

Shape representation using invariants has several problems: First, invariants
are usually derived from pure geometric transformation of shape. In reality,
shape rarely changes according strict geometric transformations, espe-
cially shapes from non-rigid objects. Second, the invariants are very sensitive to
boundary noise and errors. Third and last, designing new invariants is difficult.

 Descriptor: The descriptor will depend on the method used as happened
with Shape Signatures, i.e, the value of the \(g_E\) and \(g_P\) functions.

Shape Context

This method extracts a global feature, called Shape Context [61] for each point
in the contour. The Shape Context is intended to be a way of describing shapes
oriented to measure shape similarity and recover of point correspondences. To
get this, and by using polar coordinates, vectors from the chosen point to every
point of the frontier/contour are calculated. The length and orientation of these
vectors are quantified so a histogram can be created and then used to represent
this point. The histogram of each point is flattened and concatenated to be
part of the context of the certain shape. An example of how to extract Shape
Context from an object can be seen in 4.10.

If a Feature Descriptor wants to be useful, it needs to have certain in-
viances. In particular it needs to be invariant to translation, scale, small
perturbations, and depending on application, rotation. Translational invariance
comes naturally to shape context. Scale invariance is obtained by normalizing
all radial distances by the mean distance between all the point pairs in
the shape. Shape Contexts are empirically demonstrated to be robust to deformations, noise, and outliers [61]. Finally we can achieve complete rotation invariance in shape contexts. One way is by measuring angles at each point relative to the direction of the tangent at that point (since the points are chosen on edges). Source code can be found at [62].

Descriptor: The descriptor will be the concatenation of all the Shape Contexts for each of the points in the contour.

Chamfer

In this kind of techniques it is compulsory to find a model within an image. First we have to find weak and noisy edges and then remove them from the image. Next, transformation distance of the remaining pixels has to be calculated. The value of this pixel according to this transformation is proportional to the distance of this pixel to the nearest edge pixel.

This model [63] is then shifted around the already transformed image and in each shift position the sum of distances up to the model is calculated. The shift position with less accumulative sum is taken as the best correspondence to the model. This technique needs a good edge detector to do its first step, so it would work better along with some edge detectors like Sobel, Canny or Harris. Source code can be downloaded from [64] and additional information (in this case on how to use this descriptor in cluttered scenes) can be found at [65].

Sometimes both Shape Context and Chamfer Descriptor are used together to provide a more reliable description. This happens in pedestrian detection [63] and a comparison of how the two of them perform is shown in Figure 4.11.

Descriptor: The descriptor will be the distance between the model and the edge image at each image location.

Blurred Shape Model

In this method [66], the shape of the object is aligned based on its principal components to make the whole process rotation and reflection invariant. BSM codifies the probability of appearance of the pixels that can identify by a glance the shape of an object.
The algorithm works in this way: given a set of object shape points, they are treated as features to compute the BSM descriptor. The image region is divided in a grid of $n \times n$ equal-sized subregions (where the grid size identifies the blurring level allowed for the shapes). Each cell receives votes from the shape points in it and also from the shape points in the neighboring sub-regions. Thus, each shape point contributes to a density measure of its cell and its neighboring ones. This contribution is weighted according to the distance between the point and the center of coordinates of the region. A graphical example of its implementation is shown in Figure 4.12. In the image of the left [66] we can see the distance from a shape point to the nearest sub-regions centers. In order to give the same importance to each shape point, all the distances to the neighbors centers are normalized. The output descriptor, shown on the right, is a vector histogram of length $n \times n$ where each position corresponds to the spatial distribution of shape points in the context of the sub-region and their neighbours.

The output Feature Vector will consist of a vector histogram $v$ of length $n \times n$, where each position corresponds to the spatial distribution of shape points in the context of the sub-region and their neighbours. The resulting vector histogram, obtained by processing all shape points, is normalized in the range $[0..1]$ to obtain the probability density function (pdf) of $n \times n$ bins. In this way, the output descriptor represents a distribution of probabilities of the object shape considering spatial distortions, where the distortion level is determined by the grid size.

**Descriptor:** The Feature Vector is the vector histogram where each position is related to the spatial distribution of the shape points in its sub-region context.

### 4.3 Region-based

#### 4.3.1 Global

**Zernike Moments**

There are several Moment-based feature descriptors. Some authors have suggested the use of continuous orthogonal moments to overcome the problems associated with the geometric and invariant moments [67]. In order to solve
this problem, the use of Zernike moments has been generalised. Zernike moments \cite{68} are rotation invariant and robust to noise, so if we use a descriptor based in these moments, it will inherit this characteristics.

In \((\rho, \theta)\) polar coordinates, the Zernike radial polynomials \(R_{nm}(\rho)\) are defined \cite{69} as

\[
R_{nm}(\rho) = \sum_{s=0}^{\frac{n-|m|}{2}} (-1)^s \frac{(n-s)!}{\pi^{(n+|m|)/2-s}[(n-|m|)/2-s]!} \rho^{n-2s}
\]

where \(n\) is a non-negative integer, and \(m\) is a non-zero integer subject to the following constraints: \(n - |m|\) is even and \(|m| \leq n\). The \((n, m)\) order of the Zernike basis function, \(V_{nm}(\rho, \theta)\) defined over the unit disk is

\[
V_{nm}(\rho, \theta) = R_{nm}(\rho) \exp(jm\theta), \rho \leq 1.
\]

The Zernike moment of an image is defined as

\[
Z_{nm} = \frac{n+1}{\pi} \int \int_{\text{unit disk}} V_{nm}^*(\rho, \theta) f(\rho, \theta)
\]

where \(V_{nm}^*\) is a complex conjugate of \(V_{nm}\).

The extraction of the Zernike moment descriptor from an image is done this way: First, the input image is binarized. Since the Zernike moments are defined over a unit disk, the radius \(R\) of a circle is determined to enclose the shape completely, from the centroid of the binarized shape in the image to the outer most pixel of the shape. The shape is then re-sampled to normalize to the size of \(2R \times 2R\) pixels. This normalization step allows the scale invariance for the descriptor. Thirty-six Zernike moments of order from zero to ten for both \(n\) and \(m\) are then extracted from the normalized image, and the magnitudes are used as the descriptor.

Zernike moments have the following properties: They are rotation invariant, robust to noise, have no redundancy in the information they provide (because the bases are orthogonal), effective in a way that they describe better an image than any other type of moments, and they offer multi-level representation. Although they have interesting characteristics, their calculation has several problems: the image coordinate space must be transformed to the domain where the orthogonal polynomial is defined (unit circle for the Zernike polynomial),
the continuous integrals used to compute them have to be approximated by discrete summations (which leads to numerical errors and affects the rotation invariance) and they are costly to compute as their order is increased. Matlab code can be found at [70]. Zernike Moments are also used as a part of more complex descriptors, as it can be seen in [71].

**Descriptor:** The Feature Vector will consist of the coefficients of the Zernike moments.

**Shape Matrix**

It is similar to the Grid-based preprocessing procedure that was explained in the Feature Detection chapter (and that will in the Grid-based Descriptor that will be presented later) with the difference of using a series of concentric circles and radial lines instead of rectangular grids.

In this method [72, 73] a shape is transformed into a matrix by polar quantization of the shape. This method can be better understood by looking at Figure 4.13. If we consider the centre of gravity of the shape O and we keep the maximum radius of the shape OA with length L, in order to obtain an m × n size matrix representation of the shape, we divide OA into n − 1 equal distances and draw circles centred in O and radii L/(n-1), 2L/(n-1),...(n-1)L/(n-1). If the circles intersect in the maximum radius of the shape, each circle is divided into m equal arcs, each arc being 360/m degrees.

![Figure 4.13: (a) A shape and (b) its shape matrix][73]

To construct the Shape Matrix the next algorithm is used:

1. Create an m × n matrix that will be called M
2. From i = 0 to n − 1 and from j = 0 to m − 1, if the point with polar coordinates (iL/(n-1), j(360/m)) lies inside the shape, then M(i, j) = 1 otherwise let M(i, j) = 0.

When implementing the Shape Matrix information is gathered in an object-centred coordinate system that is normalized with respect to the maximum radius of the shape. The obtained Shape Matrix (and its related descriptor)
is translation, rotation and scale invariant. One example of its implementation can be seen at 4.13.

**Descriptor:** In this case the descriptor will consist of the values of the matrix.

**Grid-based Descriptor**

This method [74] could be seen as a different version of the Shape Matrix approach. In this case, a grid of cells is superimposed over a shape and this grid is scanned from side to side, giving a bitmap as a result. The cells that cover the shape of the object are set to 1 and the rest, to 0. The difference between the two shapes can be calculated as the number of cells in the grids which are covered by one shape and not the other and hence the sum of 1’s in the result of the exclusive-or of the two binary numbers.

However, it must be noted that the binary number obtained for the same shape with a different orientation in space or with a different scale will be quite different. The criteria for invariance of indexes is not met and hence it is required to normalized the shape boundaries prior to indexing. The normalization process involves three steps. Firstly, the shape boundaries are normalized for rotation, and in the second step they are normalized for scale and finally they are normalized for translation. The indexes generated after normalization of the shape boundaries are used for similarity measure. An example of what a Grid-based Descriptor will produce on a grey-scale image can be seen in Figure 4.14. It can be seen that the Grid preserves the location and shape context of the activity. The finer granularity - those with a smaller cellsize - provide more detail, at the cost of being a larger representation and more computationally expensive.

![The Sustained temporal change.](image1)

![The grid with cellsize $\lambda = 4$.](image2)

![The grid with cellsize $\lambda = 8$.](image3)

![The grid with cellsize $\lambda = 16$.](image4)

Figure 4.14: Grid-based Descriptor [74] computed using several cellsizes.
Descriptor: The descriptor will consist of the bitmap that represents the shape of the object.

Angular Radial Partitioning

Angular Radial Partitioning [75, 76] transforms the image data into a new structure that supports measurement of the similarity between images in an effective, easy and efficient manner with emphasis on capturing scale and rotation invariant properties.

The edge map of an image carries the solid structure of the image, independent of the colour attributes. Its applicability is well known in Computer Vision, pattern recognition and image retrieval [77]. At first the images in the database are converted to grey intensity by eliminating the hue and saturation, while retaining the luminance. We can obtain the edge image by applying an edge extraction operator (i.e. Canny edge operator) on this grey-scale image. In order to achieve the scale invariance property, the resulting edge image is then normalized to $W \times W$ pixels. This normalized edge image is called $I$ and it is used for feature extraction. In the following, we consider pixels to be either equal to ‘1’ for edge pixels or ‘0’ for non-edge pixels.

The algorithm uses the surrounding circle of $I$ for partitioning it into $M \times N$ sectors, where $M$ is the number of radial partitions and $N$ is the number of angular partitions. The angle between adjacent angular partitions is $\theta = 2\pi/N$ and the radius of successive concentric circles is $\rho = R/M$, where $R$ is the radius of the surrounding circle of the image. The number of edge points in each sector of $I$ is chosen to represent the sector feature. An example of how to use a combination of an edge map and Angular Radial Partitioning can be seen in Figure 4.15.

Descriptor: The descriptor will be, for each sector, the number of edge points in it.

Angular Radial Transform

Angular Radial Transform [78] (ART) is a moment-based image description method adopted in MPEG-7 as a region-based shape descriptor which gives a compact and efficient way to express pixel distribution within a 2-D object region. ART can describe both connected and disconnected region shapes. Mathematically it is a complex orthogonal unitary transform defined on a unit disk that consists of the complete orthogonal sinusoidal basis functions in polar coordinates.

This descriptor [79] involves the calculation of several coefficients. The ART descriptor is defined as a set of normalized magnitudes of the set of ART coefficients. Rotational invariance is obtained by using the magnitude of the coefficients. It is really similar to the already described moment-based descriptors such as Zernike moments because it takes into account the region within the shape contour. It is translation invariant and it works well with changes in scale and rotation, as it has been mentioned before.

It is also used as a region-based Shape Descriptor by using a series of basis functions (that can be seen in Figure 4.16). It has some enhanced versions such as Colour ART which widens its use to colour images and studies also its implications with colour luminance and chrominance.
Figure 4.15: ARP example a Image Example b 90° rotated version c,d Edge images superimposed with angular radial partition[76].

**Descriptor:** The Feature Vector will be composed of the coefficients of the Angular Radial Transform.

**Delaunay**

In this method first we have to find the corners of the object. Then a Delaunay triangulation [80] is calculated. The Delaunay triangulation, for a set $P$ of points in the plane, is a triangulation $DT(P)$ such that no point in $P$ is inside the circumcircle of any triangle in $DT(P)$, as it can be seen in 4.17). Its calculation is followed by a histogram of the interest points, created by discretizing the angles generated by the triangulation in a series of bins and then counting the number of times that an angle appears in the triangulation.

There are several criteria to choose the angles to be considered (the two biggest angles, the two smallest...). To calculate the difference between two objects described by Delaunay triangulation, Euclidean distance is often used.

One modification to the technique implies the identification of the straight lines in the contour of a shape and eliminate the rest of the points that could have generated it, erasing the possible defects in the generated line. Delaunay triangulation-based methods are used in nowadays research such as manuscript treatment, as it can be seen in [81]. Matlab implementation of Delaunay triangulation can be found in [82].

**Descriptor:** The Feature Vector will consist of the Delaunay triangulation of the corners of the object.
4.3.2 Structural

Skeleton Calculation

The Skeleton of a region [83, 84] can be used to represent and describe a shape. The Skeleton is defined as the connected contour of medial lines along the different parts of an object. One of the ways to calculate the Skeleton of an image is by using the Medial Axis Transform (MAT), where the Medial Axis is the geometric place where the centres of the maximum discs that fit into the shape coincide. The Skeleton can be decomposed in segments and represented as a graph. The calculation of the Medial Axis is as follows. The Medial Axis of an object is the set of all points having more than one closest point on the object’s boundary.

In 2-D, the Medial Axis of a plane curve S is the locus of the centres of circles that are tangent to curve S in two or more points, where all such circles are contained in S. The Medial Axis of a simple polygon is a tree whose leaves are the vertices of the polygon, and whose edges are either straight segments or arcs of parabolas [85].

The Medial Axis generalizes to k-dimensional hyper-surfaces by replacing 2-D circles with k-dimension hyper-spheres. 2-D medial axis is useful for character and object recognition, while 3-D medial axis [86] has applications in surface reconstruction for physical models, and for dimensional reduction of complex models. The Medial Axis together with the associated radius function of the maximally inscribed discs is called the medial axis transform. The medial axis transform is a complete shape descriptor, meaning that it can be used to reconstruct the shape of the original domain. An example of Medial Axis Transform applied in 3-D shapes can be seen in Figure 4.18. Source code can be found at [87]. Another way of calculating the Skeleton consists of using morphological operations by removing pixels on the boundaries of objects but not allowing the objects to break apart. The pixels remaining will make up the image Skeleton [88].

Descriptor: If we are using Medial Axis Transform to calculate the skeleton of an image, its coefficients will be the feature descriptor.
4.4 Discussion

In the case of Shape Descriptors we have covered a great variety of methods. In order to decide what to choose first we have to determine whether we need only the contour of the shape to recognize the object or if we also need information of the region within. This will make the first decision between Contour-based Descriptors and Region-based Descriptors.

Once we have reached to the first decision, we have yet another one to be made. In each of the two groups mentioned before we can use either the whole thing (contour or region) or divide it in parts. So the methods in each group will also be divided in Global or Structural. In this final discussion section we will give comparative information so that the decision of which method to use can be easier.

Talking about Contour-based Global Descriptors we have gone from the simpler descriptors, such as Simple Descriptors or Shape Signatures [41] to more complex approaches such as Wavelets [50] or Scale-space. In this case the decision has to be taken not only based on the performance of the methods (where Wavelets or MBC-based [56] can be seen as the best options) but taking also into account the context of the problem we want to solve. It could happen than using a simple descriptor, such as the area or main axis radius, we can get enough information to arrive to a first decision, and then using more complex descriptors to differentiate between more similar objects. If we want good results no matter how much time it takes we will have to go with Wavelets, MBC-based or Fourier-based [53] that appear in the literature as good options.
to solve certain problems.

If we decide to divide the contour information, we will enter into the field of Contour-based Structural Descriptors. There are simpler approaches such as Chain Code or Polygon Decomposition [59]. In this case an Scale-Space approach could be not as useful as with Global approaches, and the use of some other descriptors such as Shape Invariants [60] should not be discarded. More complex and specific approaches like Blurred Shape Model [66] or Chamfer [63] have also been presented along with one interesting approach that is Shape Context [61], which seems to work well without having a great complexity.

But if we want to consider the whole region of the shape, we have a good variety of Region-based Global Descriptors. Moment-based descriptors [68] are used in a large amount of methods as they are rotate invariant and robust to noise (and they can be also used in real-time). There are another group of approaches that consist of placing a mask-like structure over the shape. Shape Matrix [72] and Grid-based belong to this group, giving good results, but they can be sensitive to noise. Another method that works well is Angular Radial Partitioning [76] that offers a good comparison between a shape and a model along with being rotation and scale invariant.

Finally if we divide the shape within the contour in several elements, we have some Region-based Structural Descriptors to choose from. The most used in this group is Medial Axis Transform [83] that creates a skeleton of the image which can be use to recognize a shape. The problem associated with this method is its sensitivity to noise (which can be even worse in small shapes) and its computational cost. A summary of all the descriptors in this section is shown in Tables 4.1, 4.2, 4.3 and 4.4.
<table>
<thead>
<tr>
<th>Feature Descriptor</th>
<th>Basis</th>
<th>Rotation Invariance</th>
<th>Translation Invariance</th>
<th>Scale Invariance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape Signature</td>
<td>Representing the shape of an object by means of an 1D function extracted from the contour points.</td>
<td>Not direct</td>
<td>After normalization</td>
<td>After normalization</td>
</tr>
<tr>
<td>Convex Hull</td>
<td>Calculation of the minimal convex region.</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Scale-space</td>
<td>Following the position of the contour while applying several filters in order to eliminate non-significant parts or incidences.</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Wavelets Transform</td>
<td>Hierarchical descriptor of planar curves with several degrees of information.</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Fourier-based</td>
<td>DFT of the Shape Signature to avoid sensitivity to noise and slight variations.</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>MBC-based</td>
<td>Extracting features from the minimum circle that surrounds the object.</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 4.1: Summary of Shape Descriptors (I)
<table>
<thead>
<tr>
<th>Feature Descriptor</th>
<th>Basis</th>
<th>Rotation Invariance</th>
<th>Translation Invariance</th>
<th>Scale Invariance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chain Code</td>
<td>Nodes that surround a certain point are enumerated counter-clockwise in ascending order from inside to outside.</td>
<td>Possible</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Polygons Decomposition</td>
<td>Division of the contour in approximated segments or polygons.</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Shape Invariants</td>
<td>Avoid dependence on the viewpoint and represent properties that do not vary even under transformations.</td>
<td>Possible</td>
<td>Possible</td>
<td>Possible</td>
</tr>
<tr>
<td>Shape Context</td>
<td>Calculates vectors from a chosen point to every point in the contour in order to quantize their length and orientation into a histogram that will represent the point.</td>
<td>Possible</td>
<td>Possible</td>
<td>Possible</td>
</tr>
<tr>
<td>Chamfer</td>
<td>Tries to find a model within image after removing weak and noisy edges from the image.</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Blurred Shape Model</td>
<td>Align the shape of an object based on its principal components. Reflection invariant.</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 4.2: Summary of Shape Descriptors (II)
<table>
<thead>
<tr>
<th>Feature Descriptor</th>
<th>Basis</th>
<th>Rotation Invariance</th>
<th>Translation Invariance</th>
<th>Scale Invariance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zernike moments</td>
<td>Use of continuous orthogonal moments to overcome problems with geometric and invariant moments. Robust to noise</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Shape Matrix</td>
<td>Use of a circular grid over the image and then find out which points are within the shape.</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ARP</td>
<td>Calculation of the edge map of an image in order to capture the solid structure of the image.</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Angular Radial Transform</td>
<td>Expresses pixel distribution within a 2-D object region, describing connected and disconnected region shapes.</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Grid-based</td>
<td>Similar to Shape Matrix but using an square grid.</td>
<td>Possible</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Delaunay</td>
<td>After calculating the corners of an object Delaunay triangulation is applied.</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 4.3: Summary of Shape Descriptors (III)

<table>
<thead>
<tr>
<th>Feature Descriptor</th>
<th>Basis</th>
<th>Rotation Invariance</th>
<th>Translation Invariance</th>
<th>Scale Invariance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skeleton</td>
<td>Obtaining the skeleton of an image, defined as the connected contour of medial lines along the different parts of an object.</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 4.4: Summary of Shape Descriptors (IV)
Chapter 5

Colour Descriptors

5.1 Introduction

Another way to describe a patch provided by the several detectors that we have presented by using the Colour. Colour is the visual perceptual property corresponding in humans to the categories called red, yellow, blue and others. Colour derives from the spectrum of light (distribution of light energy versus wavelength) interacting in the eye with the spectral sensitivities of the light receptors [89]. Colour categories and physical specifications of colour are also associated with objects, materials, light sources, etc., based on their physical properties such as light absorption, reflection, or emission spectra.

In this chapter we will present several Colour Description methods, from the most basic ones such as Dominant Colour or Mean Grey Value to more complex approaches such as CCCI or CBOR. First we will describe several Colour Spaces that, while they are not descriptors by themselves, are used in all Colour Descriptors.

It is also important to define what a colour histogram is. Colour histograms in either 2-D or 3-D spaces are frequently used in digital cameras for estimating the scene illumination, as part of the camera’s automatic white balance algorithm. In remote sensing, colour histograms are typical features used for classifying different ground regions from aerial or satellite photographs. In the case of multi-spectral images, the histograms may be four-dimensional, or more.

In Computer Vision, colour histograms can be used in object recognition and image retrieval systems/databases, by taking a look at the colour distribution in a patch or region of the image.

5.2 Colour Spaces

RGB colour space

The RGB colour model [90] is an additive colour model in which red, green, and blue lights are added together in various ways to reproduce a broad array of colours. The name of the model comes from the initials of the three additive primary colours, red, green, and blue.
The main purpose of the RGB colour model is for the sensing, representation, and display of images in electronic systems, such as televisions and computers, though it has also been used in conventional photography. Before the electronic age, the RGB colour model already had a solid theory behind it, based in human perception of colours.

RGB is a device-dependent Colour Space: different devices detect or reproduce a given RGB value differently, since the colour elements (such as phosphors or dyes) and their response to the individual R, G, and B levels vary from manufacturer to manufacturer, or even in the same device over time. Thus an RGB value does not define the same colour across devices without some kind of colour management. In Figure 5.1 we show the colour triangle of RGB. Only colours within this triangle can be reproduced by mixing the primary colours. Colours outside the triangle therefore are shown in grey.

![Figure 5.1: RGB colour triangle](image)

**HSV, HSL and HSI Space**

HSL and HSV [91] are the two most common cylindrical-coordinate representation of points in an RGB colour model, which rearrange the geometry of RGB in an attempt to be more perceptually relevant than the cartesian representation.

The purpose of these models is to aid selection, comparison, and modification of colours by organizing them into a cylindrical geometry which roughly corresponds to human perception.

HSL stands for hue, saturation and lightness while HSV changes the last parameter to value. A third model, common in Computer Vision applications, is HSI which stands for hue, saturation and intensity. The colours are often
represented into cylinders and, in each of them, the angle around the central vertical axis corresponds, as it can be seen in figure 5.2.

![HSL and HSV Colour Spaces](image)

Figure 5.2: HSL and HSV Colour Spaces[91].

In each cylinder, the angle around the central vertical axis corresponds to hue, the distance from the axis corresponds to saturation and the distance along the axis corresponds to lightness, value or brightness, depending on the representation we have chosen. Note that while hue represents the same in both HSL and HSV colour spaces, their definition of saturation differ. Because HSL and HSV are simple transformations of device-dependent RGB models, the physical colours they define will depend on the colours of the red, green, and blue primaries of the device or the particular RGB space (and on the gamma correction used to represent the amounts of those primaries). So each unique RGB device has unique HSL and HSV spaces to accompany it, and numerical HSL or HSV values can describe a different colour for each basis RGB space.

**Lab Colour Space**

Lab Colour Space [92] is a Colour-opponent Space with dimension $L$ for lightness and $a$ and $b$ for the colour-opponent dimensions, based on non-linearly compressed CIE XYZ Colour Space coordinates. Unlike the RGB model, Lab Colour is designed to approximate the way humans see and understand colours. It aspires to perceptual uniformity, and its $L$ component closely matches human perception of lightness. It can thus be used to make accurate colour balance corrections by modifying output curves in the $a$ and $b$ components, as it can be seen in Figure 5.3, or to adjust the lightness contrast using the $L$ component. The colour opponent process is a colour theory that states that the human visual system interprets information about colour by processing signals from cones and rods in an antagonistic manner. The three types of cones have some overlap in the wavelengths of light to which they respond, so it is more efficient for the visual system to record differences between the responses of cones, rather than each type of cone’s individual response. In RGB space, which models the output of physical devices rather than human visual perception, these trans-
formations can only be done with the help of appropriate blend modes in the editing application.

![Figure 5.3: Examples of several CIELab colour spaces, showing only colours that fit within the sRGB gamut, using several values of L.](image)

Because Lab Space is much larger than the gamut of computer displays, printers, or even human vision, a bitmap image represented as Lab requires more data per pixel to obtain the same precision as an RGB bitmap.

### 5.3 Review of Existing Methods

#### Dominant Colour

This is the easiest way to describe a patch. This method consists of finding out the main colour in the area of the patch. This descriptor could be used to differentiate between large areas in the image but it is very noise sensitive (imagine if we take a small patch and a reflection appears on it, it is possible that the dominant colour could be taken as a strong white while it was really another). A simulated example of what a Dominant Colour algorithm can do is shown in Figure 5.4. An application that is based (although in its first stages) in Dominant Colour algorithm can be found at [93].
Mean Grey Value

This method is another different take to the Dominant Colour Descriptor. In this case we will describe a region or patch by its mean grey value. Yet it is not recommended to be used as a primary descriptor but it can help to discard reflections or zones with different grey value. It is applied in nowadays research, as it can be seen in [94] where it is used as a help technique in classifying colon polyps.

Hue

The hue [95] is one of the principal properties of a colour and can be associated to the classical concept of ink. Hue is more specifically described by the dominant wavelength in models such as the CIE system. Hue is also a term which describes a dimension of colour we readily experience when we look at colour. It will be the first of three dimensions we use to describe colour. Hue Colour Circle is shown in Figure 5.5.

Hue itself can be used to describe (in colour terms) a region, but as it will be seen afterwards, it has been used as part of more complex descriptors and treated as part of colour invariant algorithms [96].

Descriptor: The descriptor will consist of the hue representation of the image.

MPEG-7 Colour Descriptor

a. Scalable Colour Descriptor

It is a generic descriptor [97], part of MPEG-7, that consists of a Colour Space, colour quantification and histogram descriptors. It lets specify colour histograms with a variable number of bins and non-uniform quantification of several colour spaces.

The MPEG-7 implementation defines a 256-bins histogram providing 16 different values for H, and 4 different values for S and V. Normally, H is defined in the range [0, 360], while S and V are defined in [0, 1]. This procedure aims to improve the representation power of HSV colour space. Firstly a non uniform quantization of ‘hue’ is introduced, dividing the entire spectrum into 7 classes:
red, orange, yellow, green, cyan, blue and purple. Then another quantization is proposed for S-V plane, in order to distinguish the black area ($V \leq 0.2$), and the grey area ($S \leq 0.2$), providing a fixed general subdivision of the chromatic area of the plane into 4 sub-parts.

This descriptor goes beyond what HSV offers, improving its performance by being more specific. It has some problem with achromatic images although a method to overcome it is shown in [97].

**Descriptor:** The descriptor will consist of the histogram that was defined before.

### b. Colour Structure Descriptor

This descriptor [98] expresses local colour structure in an image by using an $8 \times 8$ structural element. It counts the number of times that a colour is contained inside the structural element while this element scans the image. A histogram of colour structures can be defined in the following way: the value of each bin represents the number of structural elements in the image that contains one or more pixels with a certain colour.

It uses HMMD colour space [99] which has as 'hue' value the same that HSV has and it also counts with two magnitudes called respectively maximum and minimum that contains the maximum and minimum values in each R, G and B channels. The last differential component keeps the difference between
the maximum and minimum in each channel. As it can be seen only 3 out of the 4 components is necessary to describe this HMMD space. A graphical representation of the HMMD colour space is shown in Figure 5.6.

![Figure 5.6: HMMD Colour Space](image)

As the other colour descriptors that have been shown before, it gives a more complex colour representation than the basic approaches. It can complement the just described Scalable Colour Descriptor to confirm some information.

**Descriptor:** In this case the descriptor will consist of the HMMD representation of the image.

### Opponent Colour SIFT

This method is an extension of SIFT in Opponent Colour Space [101]. In this representation we have three different channels (red vs green, blue vs yellow and black vs white, as it can be seen in Figure 5.7). When our eyes react to a colour, the response in the other channels is the opposite.

![Figure 5.7: Opponent Colour Space](image)

While RGB is an additive colour model, Opponent Space [102] gives a more perceptual response. Given that this representation is more similar to how
As it has been said, using this approximation we can get results closer as human understanding of colours. Matlab code referred to the Opponent Colour SIFT can be found at [103].

Descriptor: In the case of Opponent Colour SIFT the descriptor will be the image representation in this new space (combined with the output of the normal SIFT descriptor).

Normalized RGB Histogram

This method increases the global contrast of some images [104], especially when usable image data are represented by near contrast values. Using this adjustment, intensity values can be scattered along the histogram and it let us gain a major local contrast without having incidence in global contrast, as it can be seen in Figure 5.8. Histogram normalization does it by extending the most frequent intensity values. This method can be useful when we have images with background or foreground both clear or dark.

Descriptor: The descriptor will consist of the values of each bin of the histogram.

Colour Constant Colour Indexing

This technique [105], also known as CCCI, identifies an object by comparing its colours to the colours of each object stored in a database (taking also into account the total area covered by each colour).

To do so, the different areas associated to each colour are calculated and the histograms of both images are compared (the comparison is between the original image and the ones from the database). A colour histogram is 3-D and represents the number of image pixels that have a certain RGB value. The colour histograms of the database elements are calculated in advance and stored in a new database (after segmenting objects from the background) in order to make faster the intersection with the object to identify.

There is another version, called BR-CCCI [106] that improves CCCI sensitivity to blur. Image degradation due to blur can have multiple causes. Relative motion of the camera with respect to the object and varying acquisition parameters such as aperture and shutter time result in blurring effects.

Descriptor: The Feature Vector, in this case, will consist of the histogram of the number of image pixels that have a certain RGB value.

Colour Based Object Recognition

Colour Based Object Recognition (CBOR) [107] shares the basics with the previous CCCI method because at the end they compare objects using colour histograms.

The difference between them relies on that the group of algorithms belonging to CBOR-family also try to solve another problems such as selecting an appropriate colour model and correcting reflectance effects by using white or coloured illuminance.
Figure 5.8: (up-left) the Einstein original (up-right) enhanced RMSHE \((r = 2)\), (down-left) MWCVMHE (Minimum Within-Class Variance MHE method) \((k = 6)\), and (down-right) MMLSEMHE (Minimum Middle Level Squared Error MHE) \((k = 7)\) images[104].

As with CCCI, there is a blur-avoidance method that in this case is called BR-CBOR [106].

Descriptor: The Feature Vector, similar to the previous case, will consist of the histogram of the number of image pixels that have a certain RGB value (taking into consideration the differences between the two methods).

### 5.4 Discussion

As with Shape Descriptors, Colour Descriptors can be ordered by their complexity (not only in terms of computational cost but because of the things they take into account). In this group of descriptors we have very simple descriptors (such as Dominant Colour [93] or Mean Grey Value [94] that only take account of the grey value of the patch that we are considering. Both methods are high noise sensitive and reflections can hurt their performance.
If we consider different Colour Spaces, such as HSV [108], HSL or HSI [91] we can use more information from colour (which is more similar to how human beings perceive colours). Hue, HSV, HSL or HSI space methods belong to this group and give a more ‘realistic’ information.

Another Colour Space that has gained a lot of attention in recent years is Opponent Colour Space [101] which gives a more perceptual response, closer to human perception of colours. Its combination with SIFT is used in several research projects such as object recognition.

We cannot forget simpler Colour Spaces such as RGB, being Normalized RG Histogram [104] a method that uses it by equalizing the histogram. Some other spaces, such as HMMD (used in Colour Structure Descriptor) are variations of simpler ones (in this case, a mixture of HSV, HSL or HSI, and RGB) and are used in standard approaches like MPEG-7.

Finally there is a group of techniques, more object recognition-oriented that prepares the information they gather to ease these processes. We have CCCI [105] and CBOR [107] in this group having both of them blur concerned approaches. A summary of Colour Descriptors can be seen in Table 5.4.

<table>
<thead>
<tr>
<th>Feature Descriptor</th>
<th>Basis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dominant Colour</td>
<td>Predominant colour in a patch or portion of an image. Noise dependant.</td>
</tr>
<tr>
<td>Mean Grey Value</td>
<td>Description of a region or patch by its mean grey value. It can be useful to discard reflections.</td>
</tr>
<tr>
<td>Hue</td>
<td>Gives a representation of the colour of the image in a more near-to-human visual point of view.</td>
</tr>
<tr>
<td>Scalable Colour Descriptor</td>
<td>Improves representation power of HSV colour space by using an 256-bins histogram that provides 16 possible values for H and 4 different for S and V.</td>
</tr>
<tr>
<td>Colour Structure Descriptor</td>
<td>Expression of local colour structure in an image by using structural elements and counting the number of times a colour is contained within them.</td>
</tr>
<tr>
<td>Opponent Colour SIFT</td>
<td>Uses Opponent colour space by taking into account that human visual system relies on differences between the responses of cones rather than on the individual response of each of them.</td>
</tr>
<tr>
<td>Normalized RGB Histogram</td>
<td>Increasing of some images global contrast by adjusting intensity values along the histogram.</td>
</tr>
<tr>
<td>CCCI</td>
<td>Identification of an object by comparing its colours to the colours of the objects stored in a database. Blur avoidance is also possible.</td>
</tr>
<tr>
<td>CBOR</td>
<td>Similar to CCCI but it is concerned about some other problems such as selecting an appropriate colour model and correcting reflectance effects.</td>
</tr>
</tbody>
</table>

Table 5.1: Summary of Colour Descriptors
Chapter 6

Texture Descriptors

6.1 Introduction

Another very interesting group of descriptors are those related to Texture. Texture itself is an ambiguous word and may have one of the following meanings:

In common speech, texture is used as a synonym for surface structure and it has been described by five different properties in the psychology of perception: coarseness, contrast, directionality, line-likeness and roughness [109]. In 3-D computer graphics, a texture is a digital image applied to the surface of a three-dimensional model by texture mapping to give the model a more realistic appearance. In our field, image processing, every digital image composed of repeated elements is called a texture.

Belonging to this family of Feature Descriptors there are a few popular ones, such as SIFT, SURF or Haar-like. While any of them can be used to describe the texture of a patch, in the next pages we will cover the main aspects of each of them so the reader can have enough arguments to decide.

6.2 Review of Existing Methods

SIFT

This method, Scale-invariant Feature Transform, is nowadays one of the most used because of its special characteristics. Although it consists of both a Feature Detector and a Feature Descriptor it is very strange to see them separately, that is, they both come together. So in this subsection we will explain together both the SIFT Detector and SIFT Descriptor.

SIFT’s approach [110] transforms an image into a large collection of local Feature Vectors, each of which is invariant to image translation, scaling and rotation (and partially invariant to illumination changes and affine or 3-D projection).

The detection part of the algorithm works by identifying scale-invariant features using a filtering approach through different stages. In the first stage key locations in scale space are selected by looking for locations that are maxima or minima of a difference of Gaussian (DoG) function. Furthermore, it locates
key-points at very different regions (at different scales) so it can make these relevant points particularly stable for characterizing the image.

The input image is first convolved with a Gaussian function using a certain value of $\sigma = \sqrt{2}$. Then this is repeated a second time using a further incremental smoothing up to $\sigma = 2$. The difference of Gaussian is obtained by subtracting image smoothed with $\sigma = 2$ from the one with $\sigma = \sqrt{2}$, resulting in a ratio of $\sqrt{2}$ between the two Gaussians. The next steps will consist of re-sampling the smoothed image and repeat the same procedure using bilinear interpolation with a pixel spacing of 1.5 in each direction. This process can be better seen in Figure 6.1.

![Figure 6.1: Difference of Gaussians in SIFT[110].](image)

The depth (defined here as the number of resampling stages) is used to conform the scale-space. Maxima and minima of this scale-space function are determined by comparing each pixel in the pyramid with its neighbors. First a pixel is compared to its 8 neighbors at the same level of the pyramid. If a maxima or minima is found in this level, the closest pixel location is calculated at the next lower level of the pyramid and if it remains higher (or lower) the test is repeated for the level above, considering a total of 26 neighbors, as it shows in Figure 6.2.

![Figure 6.2: Keypoints detection in SIFT[110].](image)
The goodness of this detection algorithm is that it detects key-points that are translation, scale and rotation invariant so the features that we will extract from the points detected will probably characterize better the portion of image because the information that they will give us will be from what the object is (intrinsic characteristics) and not only from how it appears in the image. MATLAB implementation of SIFT detector can be found in [111]. Nowadays it is used in a wide range of applications, from object identification to action recognition [112].

Features returned by SIFT [110] are local and appearance-based according to the key-points given by the detector. SIFT is scale and rotation invariant and is also robust to illumination changes, noise and minimal changes in the viewpoint. Related to the SIFT descriptor we have $HOG$ [113] which is the acronym of Histogram of Oriented Gradients (HOG). This method consists of describing a patch of the image based on pondered gradient histograms, as it can be seen in Figure 6.3. The advantage of this technique is that it describes in a fair good way local aspect properties, it is affine illumination invariant and its computation is fast and non-size dependant because of the use of integral histograms. The main drawback it has is related to its affine illumination invariance, because it leads to loose extrema information and an amplification of the noise related to the gradient. It also does not let separate interesting corners/edges from the background. Source code of HOG can be found at [114]. It is used in a wide range of applications such as helping in pedestrian detection [115].

Another approach to SIFT descriptor is $PCA-SIFT$, which uses a more efficient mechanism to ease matching processes (although it uses the same basic procedures than SIFT). The steps taken are the following: First a $41 \times 41$ patch centred in each key-point is extracted using a certain scale, taking up to $N$ patches of the image. Then we rotate each patch according to its dominant orientation following a canonic direction. We calculate, for each image patch, gradient maps in both horizontal and vertical direction, combining both results to create a $39 \times 39 \times 2$ Feature Vector. Each Feature Vector is normalized to unit length in order to minimize the effects of changes in illumination. Finally we use PCA to reduce the dimensionality of the vector passing from size 304 to 20. The only problem with this method (that is faster than usual SIFT) is that we may lose information when we reduce the dimensionality, but the results, as

Figure 6.3: Figure showing gradient orientation histogram descriptors[110].
it can be seen in [116] can be better than the ones by using SIFT, as it can be seen in Figure 6.4, where it shows an example of feature matching between two different images from the same scene.

Figure 6.4: (left) SIFT feature matching between two images (right) PCA-SIFT feature matching[116].

Descriptor: The Feature Vector will be the pondered gradient histogram coefficients of each ‘interest’ patch that was detected.

GLOH

This method, acronym of Gradient Location and Orientation Histograms [117] is similar to SIFT (it is considered as an extension of SIFT) but using circular regions with 17 spatial bins and within each of them 16 orientation bins, which gives us a total of 278 dimensions (this number can be reduced by using PCA). Source code can be found at [118]. It is used in several areas of research such as 3-D Human Pose Estimation from Static Images [119].

Descriptor: The Feature Vector, in this case, will consist of the same type of data as SIFT but with less coefficients.

SURF

SURF descriptor [120] is based of SIFT although it reduces its computation time. As SIFT, it has both a Feature Detector and a Feature Descriptor that usually come together.

SURF detector works in a similar way as SIFT but it has some differences, being the most important the use of integral images to reduce the computation time [120]. Integral image is an algorithm for quick and efficient generation of the sum of values in a rectangular subset of a grid, where the value at any point \((x, y)\) in the summed area table is just the sum of all the pixels above and to the left of \((x, y)\).
The detection method in SURF is also known as the Fast-Hessian Detector [120]. It is based on the calculation of the determinant of the Hessian that was described in the Feature Detection chapter. As with SIFT, Gaussian are optimal for scale-space analysis but in practice, Gaussian needs to be discretized and cropped, and aliasing effects can appear, so it seems that the use of Gaussian filters could not be the best option. The authors [120] suggest the use of box filters that approximate second order Gaussian derivatives and can be evaluated very fast using integral images.

The $9 \times 9$ box filters (an example of them can be seen in Figure 6.5) are used at different scales in order to have different levels of depth. Scale spaces are implemented, as in SIFT, as image pyramids. The images are also repeatedly smoothed with a Gaussian and subsequently sub-sampled in order to achieve a higher level in the pyramid.

Thus, by using box filters and integral images we do not have to apply the same filter to the output of a filtered layer but we can apply the same filters directly on the original image, even in parallel (which reduces computation time). The Scale-space here is analysed by up-scaling the filter size rather than iteratively reducing the image size. The output of the first layer of filters is considered as the initial scale layer. The following layers are obtained by filtering the image with gradually bigger masks ($9 \times 9$, $15 \times 15$, $21 \times 21$, etc.).

In order to localize interest points in the image and over scales, a non-maximum suppression in a $3 \times 3$ neighborhood in three scales is applied (considering up to 26 neighbors as happens in SIFT) and the maxima of the determinant of the Hessian is interpolated in scale and space. Here the interpolation is important because of the difference in scale between the first layers of every octave is relatively large.

This detector outperforms Hessian-Laplace, Harris-Laplace and Difference of Gaussians (which is the basis of SIFT detectors) in terms of points detected versus computation time, detecting slightly less interest points but in a fifth of time, which can make it a solution if we are going for a real-time application.

SURF Descriptor first fixes a reproducible orientation based on the information it gathers from a circular region around the interest point. Afterwards it builds a square region aligned according to the selected orientation and SURF descriptor is then extracted like this:

To calculate the orientation Haar wavelets response in $x$ and $y$ direction is calculated along with its calculation over a circular region of radius $6 \times s$ around the interest point (being $s$ the scale on which the interest point was detected). The predominant orientation is estimated from the sum of every response inside a sliding orientation window which covers a $\pi/3$ angle.
To extract the descriptor, first a square region around the interest point is built and oriented according to the previously calculated orientation. Each region is subdivided in small $4 \times 4$ regions so it keeps important spatial information in. For each sub-region a set of a few simple features at $5 \times 5$ regularly spaced sample points are computed. For reasons of simplicity, we call $d_x$ the Haar wavelet response in horizontal direction and $d_y$ the Haar wavelet response in vertical direction. 'Horizontal' and 'vertical' here is defined in relation to the selected interest point orientation. To increase the robustness towards geometric deformations and localization errors, the responses $d_x$ and $d_y$ are first weighted with a Gaussian centered at the interest point.

Then, the wavelet responses $d_x$ and $d_y$ are summed up over each sub-region and form a first set of entries to the Feature Vector. In order to bring in information about the polarity of the intensity changes, we also extract the sum of the absolute values of the responses, $|d_x|$ and $|d_y|$. Hence, each sub-region has a four-dimensional descriptor vector $v$ for its underlying intensity structure $v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$. In Figure 6.6 we can see the output of the descriptor for four different sub-regions.

![Figure 6.6: The descriptor entries of a sub-region represent the nature of the underlying intensity pattern.](image)

Left: In case of a homogeneous region, all values are relatively low. Middle: In presence of frequencies in $x$ direction, the value of $\sum |d_x|$ is high, but all others remain low. If the intensity is gradually increasing in $x$ direction, both values $\sum d_x$ and $\sum |d_x|$ are high[120].

This results in a descriptor vector for all $4 \times 4$ sub-regions of length 64. The wavelet responses are invariant to a bias in illumination (offset). Invariance to contrast (a scale factor) is achieved by turning the descriptor into a unit vector. Source code can be downloaded from [121].

**Descriptor:** The Feature Vector, as it has just been mentioned, consists of a description vector for every sub-region.

### Gabor Filters

The first advantage that we gain when introducing Gaussians is that Gabor functions [122] are located in both spatial and frequency domain which does not happen with sinusoidal functions that are perfectly located in the spatial (or in temporal) domain but completely unallocated in the frequency domain. So this set of functions is more suitable to be used in operations with signals that require operations in both domains.

---

69
Fourier transform of a Gabor filter is a Gaussian centered in the harmonic function frequency (being these transforms temporal or spatial Gaussian’s Fourier transform). This result can be achieved by using the convolution property of Fourier transform, which turns products into convolutions. By doing so, transform to delta function of Gabor corresponds to the convolution of the harmonic and Gaussian functions transforms.

Gabor filters are directly related to Gabor wavelets taking into account that they are somehow bandpass functions that can be designed as a bank of directional filters (whose appearance can be seen in Figure 6.7 with different dilations and rotations). Gabor Filters source code can be found at [123] and an example of its application (in this case, applied in Chinese OCR) can be seen in Figure 6.8.

Figure 6.7: The filter set in the spatial-frequency domain (256 × 256). There are a total of 28 Gabor filters. Only the half-peak support of the filters are shown. The origin is at (row, col) = (128, 128)[122]

Descriptor: The Feature Descriptor in this case will be the image’s response to the set of Gabor filters, usually the modulus of the response. Usually Gabor Filters (and some other set of filters) are jointly used to provide a complete image description, as it can be read in [124].

MPEG-7 Texture Descriptors

a. Texture Browsing Descriptor

This is a compact Texture Descriptor [125] that needs only a maximum of 12 bits to characterize the following: the texture’s regularity (2 bits), directionality (3 x 2 bits) and depth (2 bits x 2). Because of a texture can have more than one dominant direction and scale associated, two different values for the last two parameters are permitted. 

The regularity of a texture [126] can go from 0 to 3, representing a 0 value an irregular or random texture and a 3 value a periodic pattern with well-defined directionality and depth values. A graphical example of the concept of regularity is shown in Figure 6.9. Directionality is quantified in 6 values, from 0 to 150° with steps of 30°. We can also specify two directions, using 3 bits to represent them. A value of 0 signals textures with no dominant directionality and the
Figure 6.8: Demonstration of a Gabor filter applied to Chinese OCR. Four orientations are shown on the right $0^\circ$, $45^\circ$, $90^\circ$ and $135^\circ$. The original character picture and the superposition of all four orientations are shown on the left[122].

rest of directions make the value go from 1 to 6. Depth value is also associated with this and it is related to image’s scale or resolution. It is quantified with 4 values, being 0 a sharp texture and 3 a coarse one. Information on how to implement it can be found at [127]. Its use in image retrieval can be consulted at [128].

<table>
<thead>
<tr>
<th>00 (irregular)</th>
<th>01</th>
<th>10</th>
<th>11 (periodic)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Figure 6.9: Example of regularity[126].

**Descriptor:** The Feature Descriptor will be composed of the 12 bits that have been mentioned before.

**b. HTD**

Acronym of Homogeneous Texture Descriptor [125, 129, 126]. Homogeneous texture is an important visual primitive for searching and browsing through
large collections of similar looking patterns. If an image can be partitioned into a set of homogeneous texture regions, then the texture features associated with the regions can index the image data. An examples of homogeneous textured patterns are viewed is a car parking with cars parked at regular intervals, or also agricultural areas and vegetation patches in aerial and satellite imagery.

This descriptor uses 62 8-bit numbers per image or image region in order to allow for accurate search and retrieval. The image is filtered with a bank of orientation and scale sensitive Gabor filters, and the means and the standard deviations of the filtered outputs in the spatial frequency domain (5 scales × 6 orientations per scale) are used as the descriptor components. The frequency space is partitioned into 30 channels with equal angular divisions at 30° intervals and five-octave division in the radial direction as it can be seen in Figure 6.10:

![Figure 6.10: Homogeneous Texture Descriptor][126].

The use of this method in the literature proves that it is a robust, effective and easy to compute descriptor and it is also part of the MPEG-7 [130] standard. The computation complexity of this descriptor can be dramatically reduced if values are calculated in the frequency instead of the spatial domain along with the use of the Radon transform, which will be explained later.

Descriptor: The descriptor will consist of the 62 numbers per image or image region that contain information of the response to the bank of filters.

**Edge Histogram Descriptor**

This descriptor [131] captures spatial distribution of edges that is very representative when we want to search for image per image correspondences, even when the underlying texture is not homogeneous.

The calculation of this descriptor is somehow straightforward. First we divide an image in 4 × 4 sub-images and histograms of local edges are calculated for each of them. Edges are grouped in five categories: vertical, horizontal, 45° diagonal, 135° diagonal and isotropic. Each local histogram has also 5 bins corresponding to each of the mentioned categories so we will have up to 80 bins,
which are often quantified in a non-uniform way using 3 bits per bin, giving us a final descriptor of 240 bits. The five types of edges and the five types of edge bins for each sub-image are shown in Figure 6.11.

Figure 6.11: (top) Groups of possible edges (bottom) Edge Histogram Descriptor of an image[131].

In order to compute edge histograms, each of the 16 sub-images are divided in blocks and a simple edge detector is then applied. Thus, the histogram for each sub-image represents the relative frequency of occurrence of the 5 types of edges in the corresponding sub-image. Source code of this descriptor can be found at [132]. To see how it can help in bigger application (such as X-Ray Image Classification and Retrieval) [133] can be a good reading.

Descriptor: The Feature Vector will be composed of the values associated
to the edge histogram.

**Radon representation-based Feature Descriptor**

Bi-dimensional Radon Transform is an integral transformation of a function along a group of lines. As an example, if a line is represented by the equation $x \cos(\theta) + y \sin(\theta) = s$, where $s$ is the minimum distance between the line and the origin and $\theta$ is the angle that the line makes with the $x$-axis, the Radon transform is equal to:

\[
R[f](\theta, s) = \int_{-\infty}^{+\infty} \delta(x \cos(\theta) + y \sin(\theta) - s) \, dx \, dy
\]

In a $n$-dimensional space, the Radon transform [134] is the integral of a function on hyperplanes. The integral of a function along a group of lines in the $n$-dimensional space is also known as X-ray transform. As it has been stated, Radon and Fourier transform are strongly related. Bi-dimensional Fourier transform of $x = (x, y)$ is:

\[
\hat{f}(w) = \frac{1}{2\pi} \int f(x) e^{-iwx} \, dx \, dy
\]

We will use the next notation:

\[
R_{\theta}[f](s) = R[f](s, \theta)
\]

because we will do the Fourier transform of variable $s$. Projection-slice theorem is formulated as:

\[
R_{\theta}[f](\sigma) = \sqrt{2\pi} \hat{f}(\sigma n(\theta))
\]

where $n(\theta) = (\cos \theta, \sin \theta)$.

By using this property we have an explicit way to invert the Radon transform (and to study whether it is possible or not to invert it). But this method is way too complex and the operations involved can consume a lot of time. Radon transform of an image are shown in Figure 6.12.

After presenting the basics of the Radon transform, we will describe a Radon representation-based feature descriptor (RRFD). RRFD converts the original pixel represented images into Radon-pixel images by using the Radon transform. The new Radon-pixel representation is more informative in geometry and has a much lower dimension. Subsequently, RRFD efficiently achieves affine invariance by projecting an image (or an image patch) from the space of Radon-pixel pairs onto an invariant feature space by using a ratiogram, i.e., the histogram of ratios between the areas of triangle pairs. The illumination invariance is also achieved by defining an illumination invariant distance metric on the invariant feature space. Comparing to the existing Radon transform-based texture features, which only achieve rotation and/or scaling invariance, RRFD achieves affine invariance.

**Descriptor:** In the case of Radon transform, the Feature Descriptor will consist of the associated coefficients of this transform.
Figure 6.12: (left) Original Image (right) Radon Transform

**Hough Transform**

Hough Transform [135] is an algorithm used in image pattern recognition which eases the process of finding some special shapes in an image, such as lines, circles, etc. Its simplest version finds lines. Its way of operating is mainly statistical and for each point it tries to find out if it belongs to a line. Then an operation within a range of pixels is applied in order to find all the possible lines that this point could belong to. This is done to every point in the image and in the end lines with more possible points are determined as the lines in the image.

The Hough transform algorithm uses an array, called accumulator, to detect the existence of a line $y = mx + b$. The dimension of the accumulator is equal to the number of unknown parameters of the Hough transform problem. For example, the previous linear Hough transform problem has two unknown parameters: $m$ and $b$. The two dimensions of the accumulator array would correspond to quantized values for $m$ and $b$. For each pixel and its neighborhood, the Hough transform algorithm determines if there is enough evidence of an edge at that pixel. If so, it will calculate the parameters of that line, and then look for the accumulator’s bin that the parameters fall into, and increase the value of that bin. By finding the bins with the highest values, typically by looking for local maxima in the accumulator space, the most likely lines can be extracted, and their (approximate) geometric definitions read off. The simplest way of find-
ing these peaks is by applying some form of threshold, but different techniques may yield better results in different circumstances - determining which lines are found as well as how many. Since the lines returned do not contain any length information, it is often next necessary to find which parts of the image match up with which lines. Moreover, due to imperfection errors in the edge detection step, there will usually be errors in the accumulator space, which may make it non-trivial to find the appropriate peaks, and thus the appropriate lines.

Figure 6.13: (top) Original Image (bottom) Hough transform

Hough Transform is related with Radon transform up to the point that Hough’s representation of an image in a \((\rho, \theta)\) plane can also be referred to the Hough space for the set of straight lines in two dimensions. In some way they both are the same transform. Hough Transform methods are used in nowadays applications such as bar-code reading or another applications that imply lines identification. Source code can be found at [136]. Hough Transform is used in line detection and also helps in bigger tasks as whole Object Detection [135]. In the field of Texture Descriptors, it aids on the computation of the Texture Browsing Descriptor [125] by extracting the two dominant directions of the texture primitives, that are considered as parallelograms.

Descriptor: The results of this transform are stored in a matrix. Cell value represents the number of curves through any point. Higher cell values are rendered brighter. The two distinctly bright spots are the Hough parameters of the two lines. From these spots’ positions, angle and distance from image center of the two lines in the input image can be determined.
Steerable Filters

One way to find out the response of a filter in several orientations is to apply several versions of the same filter that differ is a little angular rotation. A more efficient method consists of applying a few filters which correspond to a little number of angles and then interpolating the response to the other angle values. So, the number of initial filters is needed to know and how to interpolate correctly the several responses. An example of Steerable Filters can be found at Figure 6.14.

Figure 6.14: Example of steerable filters: (a) $G_{0^\circ}^1$, first derivative with respect to $x$ (horizontal) of a Gaussian; (b) $G_{90^\circ}^1$, which is $G_{1^\circ}^0$, rotated by 90°. From a linear combination of these two filters, one can create $G_\theta^1$, which is an arbitrary rotation of the first derivative of a Gaussian; (c) $G_{60^\circ}^1$, formed by $\frac{1}{2}G_{1^\circ}^0 + \frac{\sqrt{3}}{2}G_{1^\circ}^9$.

The same linear combinations used to synthesize $G_\theta^1$ from the basis filters will also synthesize the response of an image to $G_\theta^1$ from the responses of the image to the basis filters; (d) image of circular disk; (e) $G_{0^\circ}^1$ (at a smaller scale than pictured above) convolved with the disk (d); (f) $G_{90^\circ}^1$ convolved with (d); (g) $G_{60^\circ}^1$ convolved with (d), obtained from $\frac{1}{2}$ (image (e)) + $\frac{\sqrt{3}}{2}$ (image (f))[137].

By using the correct set of filters and an appropriate interpolation mechanism the response of a filter with arbitrary orientation can be calculated without having to really apply the filter. The term Steerable Filters [137] is then used to describe a class of filters where an arbitrary-orientation filter is synthesized as a linear combination of basic filters. Source code of steerable filters can be found at [138].

Descriptor: The descriptor will consist, then, of the response of the image to the different steerable filters.

Dissociated Dipoles

Dissociated dipoles [139] is a method that is relatively based on Haar-like features, being a more general feature set, as it can be seen in Figure 6.15.

This Feature Descriptor compare the mean illuminance values of two regions, the so called excitatory and the inhibitory dipoles. From a computational point of view, the evaluation of a dissociated dipole has the same cost as the evaluation of a Haar-like feature. They also share the robustness in front of
noise and illuminance changes. In fact, Haar-like features can be represented by means of the dissociated dipoles.

The dipoles set is composed by all the possible sizes and positions of each one of the two regions. Any exhaustive search (i.e. Adaboost approach) over this feature set is unfeasible.

Another version is the called 'Weighted Dissociated Dipoles' [139] which are a more general definition of the dissociated dipoles, where each region has an associated weight. With this simple modification, we can represent all the edge, line and center-surrounding types of the Haar-like feature set. As in the case of the normal dissociated dipoles, the huge dimension difficults the use of this type of features by classical approaches.

Descriptor: The Feature Vector set will be composed of the response of each possible dissociated dipole that have been defined.

Haar-like Features

Haar-like Features [140] are commonly used in object recognition. Their name comes because of their relationship with Haar wavelets. This feature set (after abandoning the use of RGB methods which are more expensive) considers rectangular regions of the image and sums up the pixels in this region, as it can be seen in Figure 6.16. This sum is then used to categorize images. For example if we have an image database with human faces and buildings and we want to distinguish between them. It is possible that if the eye and the hair region of the faces is considered, the sum of the pixels in this region would be quite high for the human faces and arbitrarily high or low for the buildings. The value for the latter would depend on the structure of the building, its environment while the values for the former will be more less roughly the same.

So, as it can be seen, a recognition process can be much more efficient if it is based on the detection of features that encode some information about the class to be detected. This is the case of Haar-like features that encode the existence of oriented contrasts between regions in the image. A set of these features can be used to encode the contrasts exhibited by a human face and their spacial relationships as part of texture description.

This method is scale and illumination invariant and it can be made rotation
Figure 6.16: A set of Haar-like structural elements[140].

invariant. It has been used mainly in face/person recognition but it can also be used to detect certain structures by applying the correct filter and in some other applications such as surface inspection and edge detection of a solar cell as it can be read in [141]. Source code can be found at [142].

In face detection, the Haar features demonstrated their great ability to discover local patterns - intensity difference between two subregions. A scheme of face detection is shown in Figure 6.17.

But it is difficult to find discriminative local patterns on more difficult cases such as animals heads which have more complex and subtle fine scale textures. On the contrary, the use of oriented gradients features (like HOG) mainly consider the marginal statistics of gradients in a single region. It effectively captures fine scale texture orientation distribution by pixel level edge detection operator. However, it fails to capture local spatial patterns like the Haar feature. The relative gradient strength between neighboring regions is not captured either.

RIFT

The Rotation Invariant Feature Transform algorithm (RIFT) [144] consists of the following steps:

- Extraction of a set of elliptic regions of an image with texture by Blob Detectors, which can detect complementary types of structures.

- Elliptic regions are normalized up to an unit circle to reduce the affine ambiguity to a rotational one.

- The circular patch is normalized in concentric circles with the same width, using 4 rings.
• Final calculation of HOG with 8 orientations for each ring, giving as a result a $4 \times 8$ descriptor.

In order to achieve rotation invariance the orientation is measured in each point from a high intensity value to a lower one (i.e., passing from a white region to a dark one). RIFT is sensitive to transformations that imply turnings of the normalized patch of the image. An example of RIFT descriptor calculation can be seen in Figure 6.18.

**Figure 6.17:** Schema of Face Detection Algorithm[143].

**Figure 6.18:** Construction of RIFT descriptors. Three sample points in the normalized patch (left) map to three different locations in the descriptor (right)[144].

*Descriptor:* As it has been explained the RIFT descriptor is very similar to SIFT’s one, giving as output the values associated to HOG.
Local Binary Patterns

This kind of features [145] are used in classification processes. Combined with HOG classifier give great results in human classification.

The histogram of the binary patterns computed over a region is used for texture description. The operator describes each pixel by the relative grey levels of its neighboring pixels -see Figure 6.19 for an illustration with 8 neighbors. If the grey level of the neighboring pixel is higher or equal, the value is set to one, otherwise to zero. The descriptor describes the result over the neighborhood as a binary number (binary pattern):

\[ LBP_{R,N}(x,y) = \sum_{i=0}^{N-1} s(n_i - n_c) 2^i, \]

where \( n_c \) corresponds to the grey level of the center pixel of a local neighborhood and \( n_i \) to the grey levels of \( N \) equally spaced pixels on a circle of radius \( R \).

Since correlation between pixels decreases with distance, a lot of the texture information can be obtained from local neighbourhoods. Thus, the radius \( R \) is usually kept small. In practice the equation means that the signs of the differences in a neighborhood are interpreted as a \( N \)-bit binary number, resulting in \( 2^N \) distinct values for the binary pattern.

![Figure 6.19: LBP features for a neighborhood of 8 pixels[146].](image)

From above, it is easy to figure out that the LBP has several properties that favour its use in interest region description. The features have proven to be robust against illumination changes, they are very fast to compute, and do not require many parameters to be set [146]. Matlab source code can be downloaded from [147].

Descriptor: The descriptor, as it has been mentioned before, will describe the result of the operator for each pixel.

Co-occurrence Matrices

Co-occurrence (spatial dependence) Matrices are used in texture description. We will explain the basis on how to calculate a Co-occurrence Mmatrix of a \( 5 \times 5 \) image [148]. First it is necessary to calculate the number of different pixel
values and then rank these values from the smallest to the largest. The next step is to scan the image in the direction noted (in this case from East to West) to determine the frequency with which each of the pixel values follows another.

With respect to the digital image presented, six different pixel values are observed, from 0 to 5. So, the co-occurrence matrix is a $6 \times 6$ matrix. In Figure 6.20 an example of the calculation of the co-occurrence matrix is shown.

\[
\begin{array}{cccccc}
1 & 1 & 2 & 2 & 5 & [A] = \\
3 & 2 & 3 & 1 & 1 & 1: 142000 \\
0 & 1 & 1 & 0 & 1 & 2: 012111 \\
3 & 2 & 4 & 0 & 1 & 3: 012000 \\
2 & 1 & 1 & 2 & 2 & 4: 100000 \\
& & & & & 5: 000000 \\
\end{array}
\]

Figure 6.20: (left) Original $5 \times 5$ digital image (right) Co-Occurrence Matrix\[148]\.

Once this matrix is determined, many statistical parameters can be computed (in [148] seven different parameter calculations are explained). Among them, the maximum value for any entry, first and second-order element difference moments (both normal and inverse element) or entropy. Once these statistical parameters are computed, a similar sized window is used centered over pixels where we want to test a concrete texture pattern. Source code can be found at \[149]. It is used in classification processes (as a way to identify texture patterns) as it can be seen in \[150\] where Co-occurrence Matrices are used to extract features as contrast, homogeneity and energy.

**Descriptor:** In the case of Co-occurrence Matrices, the descriptor will consist of several parameters that have been calculated from the Co-occurrence Matrix that was calculated by this method.

### 6.3 Discussion

Texture, as it has been shown throughout this chapter, can be a key feature to use in recognition processes (or even if we only want to differentiate between two similar objects). There are many **Texture Descriptors**, some of them have been presented in this report. We can group them according to the main technique that they use and through this discussion section we will show the different groups and the main differences between the methods related to them.

The first group is the one related to the use of Scale-space approaches, that is, trying to describe a feature that has been detected (in a previous step) taking into account different scales when deciding if one point is really a key-point or not. Among the methods belonging to this group we can find SIFT [110], GLOH [117] and SURF [120]. SIFT is nowadays one of the most used algorithms for both description and detection and is applied in many different procedures which involve some kind of feature-based object recognition. While it is indeed a good technique there are good alternatives such as SURF which uses a similar schema. While its calculation of the orientation may seem more
complicated it can be faster than SIFT. GLOH can be considered as an slightly enhanced version of SIFT.

Some other approaches, such as HTD [129] or Texture Browsing Descriptor [125] go in a different route by giving as output smaller descriptors by applying different procedures. Texture Browsing Descriptor highlights some aspects that the other presented descriptors does not such as texture's regularity or depth. HTD is a competitive alternative and it is computationally efficient (with help of some techniques such as the Radon transform [134]).

Talking about this transform (and its close-relative Hough Transform [135]) there are some descriptors that can be used for specific tasks (that is the case of Hough Transform and its property of line identification and description or the edge-description oriented Edge Histogram Descriptor).

Another family of methods is the one where RIFT [144] belongs to. This family of methods use the values of the intensity in the patches to create the descriptor having differences in the latter steps of the process.

Finally there are a group of methods that are not usually considered as descriptors by themselves but can aid in description processes, such as Gabor Filters [122] and Steerable Filters [137]. They can be used as a preliminary step in description by enhancing some properties of the object to be described. GIH [151] uses a special type of distance (Geodesic Distance) to calculate the descriptor.

In short, the variety in Texture Descriptors can be seen as a matter of granularity. The first (let us call them 'SIFT-related' descriptors are used to describe a patch more generally where some of the other approaches are more specific (such as the transform-based methods). It is not easy to compare the different Texture Description methods because some of them are application-oriented and it is difficult to compare them if they are not applied on the same context. So the decision of which descriptor to use should be based on what we want to emphasize, what we think is more descriptive of the object that we want to recognize. In Tables 6.1 and 6.2 a summary of the most important Texture Descriptors can be found.
<table>
<thead>
<tr>
<th>Feature Descriptor</th>
<th>Basis</th>
<th>Rotation Invariance</th>
<th>Translation Invariance</th>
<th>Scale Invariance</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIFT</td>
<td>Describing a patch of an image by pondered oriented gradient histograms.</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>SURF</td>
<td>Similar to SIFT but uses integral images.</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Gabor Filters</td>
<td>Bank of filters with different dilations and rotations.</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Texture Browsing Descriptor</td>
<td>Characterization of the texture's regularity, directionality and depth.</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>HTD</td>
<td>Find the homogeneous texture pattern along a patch to help image retrieval.</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Edge Histogram Descriptor</td>
<td>Captures edges spatial distribution.</td>
<td>No</td>
<td>Yes</td>
<td>Possible</td>
</tr>
<tr>
<td>Radon-based Descriptor</td>
<td>Converts the original pixel represented images to Radon-pixel images.</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Hough Transform Descriptor</td>
<td>Useful to fit some special shapes in an image such as lines, circles under a statistical approach.</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 6.1: Summary of Texture Descriptors (I)
<table>
<thead>
<tr>
<th>Feature Descriptor</th>
<th>Basis</th>
<th>Rotation Invariance</th>
<th>Translation Invariance</th>
<th>Scale Invariance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disassociated Dipoles</td>
<td>Comparison of the mean illuminance value of two regions.</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Steerable Filters</td>
<td>Finds out the response of a filter in several orientations without having to calculate them all.</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>GIH</td>
<td>Keeps information of the joint distribution of the geodesic distance and the intensity of the sampled points.</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RIFT</td>
<td>Calculates Histogram of Oriented Gradients for each ring.</td>
<td>Possible</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Local Binary Patterns</td>
<td>Description of each pixel by the relative grey levels of its neighbors.</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Co-occurrence Matrices</td>
<td>Frequency of each pixel value’s following to another possible value appearance.</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 6.2: Summary of Texture Descriptors (II)
Chapter 7

Motion Descriptors

7.1 Introduction

Another group of descriptors are those which describe motion in video sequences. Motion carries a lot of information about spatio-temporal relationship between image objects [152], that can be used in several applications such as traffic monitoring or security surveillance (by identifying objects that enter or leave the scene or simply objects that just moved). Motion can be related to the movement of an object in a scene (for example, by checking its position in two consecutive frames) or to the movement of the device that captures the image (such as zooming or panning of the camera). Thus, Motion Descriptors can be used to track objects in video sequences or to describe the general movement of the objects in a scene.

Related to this area there are a few motion descriptors that will be explained in this chapter, going from the well-known Optical Flow to Angular-based descriptors.

7.2 Review of Existing Methods

Motion Estimation Methods

Optical flow [153, 154] is the pattern of apparent motion of objects, surfaces and edges in a visual scene caused by the relative motion between an observer (an eye or a camera) and the scene. There are several Optical Flow applications such as motion detection, object segmentation, time-to-collision and focus of expansion calculations, motion compensated encoding, and stereo disparity measurement which use the motion of the surfaces and edges of a given object.

Before explaining what Optical Flow consists of, we will make a general introduction to two methods that can be used as part of a Motion Description chain that uses Optical Flow methods.

Block Matching

Block Matching [152] is the simplest algorithm for the estimation of local motion. It uses a spatially constant and temporally linear motion model over a
rectangular region of support. They are useful given its simplicity and regularity (it applies the same operations for each block of the image). In video compression, motion vectors are used to eliminate temporal video redundancy by means of motion-compensated prediction.

The Block Matching module estimates motion between two images or two video frames using "blocks" of pixels. The Block Matching module matches the block of pixels in frame $k$ to a block of pixels in frame $k+1$ by comparing the block of pixels over a search region. Suppose that the input to the Block Matching module is the frame $k$. The Block Matching module performs the following steps:

1. First the frame is divided using the values that are decided for both the block size and overlap parameters.

2. For each subdivision or block in frame $k+1$, the Block Matching algorithm establishes a search region based on the value of the maximum displacement parameter.

3. The block searches for the new block location using either the Exhaustive or Three-step Search method that is explained in Figure 7.1.

---

Figure 7.1: Three-step Block Matching search method[155].
As mentioned above, Block Matching is not used as a Motion Descriptor by itself but is part of several related methods. An example of its use, in this case to calculate Predictive Motion Vector Fields, can be consulted at [156].

**Phase Correlation**

Phase Correlation [152] is a computationally efficient method that is able to estimate two and three dimensional translations. The usual Phase Correlation schemes are able to estimate relatively large rotation, scaling and translation, based on the shift property of the Fourier transform.

If we denote $\hat{I}(\vec{\omega}) = \mathfrak{I}I(\vec{t})$ the Fourier transform of the image $I$, where in the 2-D case $\vec{\omega} = (\omega_x, \omega_y)$ and $\vec{t} = (t_x, t_y)$, then

$$\mathfrak{I}I(\vec{t} + \vec{\Delta}) = \hat{I}(\vec{\omega} e^{i(\vec{\omega} \cdot \vec{\Delta})})$$

where $\vec{\Delta} = (\Delta_x, \Delta_y)$. If the input images $I_1(\vec{t})$ and $I_2(\vec{t})$ are related by $I_1(\vec{t} + \vec{\Delta}) = I_2(\vec{t})$ then $\hat{I}_1(\vec{\omega}) e^{i(\vec{\omega} \cdot \vec{\Delta})} = \hat{I}_2(\vec{\omega})$ and

$$\hat{C}(\vec{\omega}) = \frac{\hat{I}_2(\vec{\omega})}{\hat{I}_1(\vec{\omega})} = e^{i(\vec{\omega} \cdot \vec{\Delta})}$$

The translation $\vec{\Delta}$ that is given by $\hat{C}(\vec{\omega})$ can be recovered in either the spatial or frequency domains. The final step will be to calculate the inverse FFT of $\hat{C}(\vec{\omega})$, whose maximum value will let us recover the value of $\vec{\Delta}$.

This scheme was proven robustly estimate large translations where the corresponding overlap between the registered images is as small as 30%.

**Optical Flow**

In order to estimate motion we can use a sequence of ordered images and from them we can obtain instantaneous image velocities or discrete displacements.

Without entering in too many details, the Optical Flow methods calculate the motion between two frames (taken at times $t$ and $t + \Delta$ at every voxel position. These methods are referred to as differential because they are based on approximations on local Taylor series of the original images by using partial derivatives with respect to spatial and temporal coordinates.

Optical Flow not only studies how to determine the optical flow field by itself but it is also used to find the way the scene is structured and its 3-D nature. It also helps estimate the movement of both the objects and the relative movement of the observer to the scene. Thus, its strength is based on that it infers both the structure of objects and the environment along with the motion of the observer. It has been used in many areas such as object detection and tracking, movement detection, video compression and robot navigation. An example of Optical Flow in a image can be seen in Figure 7.2.

Matlab source code can be found at [158]. More information and an application of Optical Flow (in this case, recognition of basketball matches events) can be found at [159].

**Descriptor:** The descriptor will consist on the motion vectors that will describe the relative movement between the two consecutive images.

88
Angular Circular Motion

This method [160] uses as its basis the fact that the shape of an object is not fixed and it can change with time. Given that the effects associated to the camera (such as zoom or panning) can be compensated, we can use the deformations present in objects as source of movement information.

Using this, the variation of the area associated to an object can indicate the amount of deformation that it has suffered. Thus, the authors of the Angular Circular Motion method propose to divide the mask of a shape of a video object in $M$ angular and $N$ circular segments (as it can be seen in Figure 7.3) and use the variation of the pixels that belong to each segment in order to describe local movement.

The proposed Feature Descriptor [79] is then scale invariant since the number of angular and circular segments is the same for all video objects, and the size of each segment is scaled. Rotation invariance can be approximated by using an appropriate query matching similar to the one used for matching in the contour based shape descriptor employed in MPEG-7 [130].

Descriptor: In Angular Circular Motion, the descriptor will consist of, for each segment, the variation of the intensity value of the pixels that belong to the concrete segment.

ART

Angular Radial Transform (ART) was already explained in the Shape Description chapter and it was proven to be an efficient way to retrieve shape information, because its descriptors are easy to extract and match. In this case the variance of ART coefficients, computed using each Video Object Plane (VOP) of a video object, are used as local motion descriptors [160]. VOP are the binary shape masks, as they were described in the Angular Circular Motion method. As the ART descriptors describe the region of a shape, they are capable to represent holes and unconnected regions. So the proposal mentioned in [160] would capture a large variety of shape region deformations caused by the local motion.

Descriptor: The ART descriptor is defined as a set of normalized magnitudes of the set of ART coefficients, $F_{nm}$ of order $n$ and $m$, that are defined by:

$$F_{nm} = \int_{0}^{2\pi} \int_{0}^{1} V_{nm}(\rho, \theta) f(\rho, \theta) \rho d\rho d\theta$$

(7.1)

where $f(\rho, \theta)$ is an image function in polar coordinates and $v_{nm}(\rho, \theta)$ is the ART basis function that is separable along the angular and radial directions [78].

7.3 Discussion

In the case of Motion Descriptors we do not have the same variety of methods than in the other cases yet the ones presented are quite interesting. One of the most used when finding the motion process in a video is Optical Flow [153] that takes into account the variation of an object between consecutive frames. This method can describe quite well the motion, except in cases where the movement is quite abrupt (i.e., a sudden disappearance of an object).
The other techniques presented are quite different. Angular Circular Motion [160] uses information from the deformation that an object may suffer to inherit the motion (considering that we can compensate typical camera effects such as zooming) while Angular Radial Transform [78] is moment-based and can describe movement by finding the connected region between different frames (it can be seen similar to use a shape descriptor between consecutive frames). A summary of the descriptors presented can be consulted at Table 7.1.

<table>
<thead>
<tr>
<th>Feature Descriptor</th>
<th>Basis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optical Flow</td>
<td>Pattern of apparent motion of objects, surfaces and edges in a visual scene based on the relative motion between an observer and the scene.</td>
</tr>
<tr>
<td>Angular Circular Motion</td>
<td>Based of the change of appearance of the shape of an object in time and the use of the deformations present in objects as source of movement information.</td>
</tr>
</tbody>
</table>

Table 7.1: Summary of Motion Descriptors
Figure 7.2: (top) This image shows a scene from a simulated environment, where the agent has just moved one step forward from its initial starting position (bottom) Displacement of some selected points (a dense matrix of evenly spaced pixels) in the image plane - superimposed on the upper image - that is caused by the motion of the agent[157].
Figure 7.3: Binary shape maps of a video object divided into 4 angular and 2 circular segments[160].
Chapter 8

Conclusions

Introduction

The objective of this technical report was never to explain in depth all the different feature detectors and feature descriptors that exist on the literature. That objective would never be able to be obtained because every day a new method is developed (even if it is only a slight evolution of another one) and it would be impossible to cover them all. So the main goal of this report is to provide a first information of techniques that are being used as of today and provide information in order to ease a first decision on which method to use.

In Computer Vision there are a lot of applications where Features have a key role. Whether it is face recognition or object classification, extracting features and then provide useful information, which is what descriptors do is always present. When we talk about features we mean the parts of the image than can give us enough information to develop a task. For example, if we want to detect boats in the sea, first it could be useful to separate what is ‘sea’ from what is not. In this case, by detecting big patches in the image and using a colour description we can make a first decision. But once we have decided which parts of the image interest us, we can use a different detector to find areas that can contain certain shapes (contours, edges) that can help us to find the boats.

While this example is not complete, what we want to say is that in certain processes there is not only a way to do things. There is no a perfect detector that can offer instantaneously the position of the key part of the image that will lead to the perfect solution. Each detector and descriptor has its good points and its flaws and there is a big number of them that can help only in certain applications.

This is, then, the aim of this report. To present where we are now in terms of actual feature detector and feature descriptors and show possible tools to create the method that can solve our problem.

Feature Detection

Processes that involve the use of features usually are divided in two: detection and description. First we detect where a feature may be and then we describe that zone in order to provide enough information about this feature. There are many feature detection methods and they are usually divided according to
what they look for. They can be searching for Edges (like Sobel or Canny-Edge) or Corners (Harris or SUSAN), which are specific parts of an image or detecting structure-like areas (Blob Detectors or Region Detectors). Again the concrete aim of the project will tell us which of them we can use, but there is a certain trend in actual search going to the use of scale-space approaches (such as SIFT or SURF) to detect interest patches.

Among the other type of Detectors, Harris (and its multiple versions) can give good corner and edge detection while Hessian or MSER can act as good Blob Detectors. One thing that can be useful when deciding is the number of key-points that each method detect or how much time they consume (if our aim is to develop a real-time application).

Feature Description

Imagine that we want to describe some object in an image. What properties will we use? While this question may seem a very open one it can be reduced to at least three group of answers: the colour, the shape and the texture. These three properties (using all of them or a sub-group) can give enough information to achieve our goal. So the great variety of Feature Descriptors can be also sub-divided in Shape, Colour and Texture Descriptors. There is another group of Descriptors, Motion Descriptors that are useful where our application not only involves the treatment of individual images but the analysis of a sequence of frames extracted from a video.

The shape of an object can really provide enough information for certain tasks like object recognition and are the group of descriptors that has more methods associated. But first we have to make a decision: if the important part of the shape, the one that will provide us the key information, is contained in its contour or in the whole shape. So the methods are sub-divided in Contour and Region-based. One last subdivision is made according to if we want to use all the information from the contour or region or divide it into elemental parts (that is the difference between Global and Structural approaches). We will not repeat here the conclusions that we have obtained after the study, but there are indeed quite interesting methods such as MBC-based, Shape Context, Shape Matrix or Medial Axis Transform.

Once we have a patch where we know there is an important characteristic inside, we can use the colour information of this patch as an input to our solution. While we use the general colour distribution or we use histogram representation, the methods that have been presented are being used in current applications and some of them are easy to use. Some of the methods try to approximate how human being understand and perceive colours, adding not only information about the 'colour' that we see (what is really known as 'hue') but some other factors such 'saturation' and 'value' ('lightness' or 'illuminance', depending on the colour model of choice) and they can be useful in Computer Vision tasks where we want computers to act somehow like humans will do so the more similar to us we make them, the more similar to us they will react.

Texture is a challenging topic because it can be really hard to use it in certain tasks and sometimes it is even difficult to define. There are many Descriptors that are used in this field (and the more popular are also commonly attached to this group like SIFT, SURF or Transform-based). As we said before, the option to choose will be application-dependant, and we have a wide range of them that
go from general descriptions to more specific ones.

Talking about Motion Descriptors there are less methods introduced but yet they can be useful in video-processing related applications whether we want to estimate where an object is in consecutive frames to predict patterns of movement or to find the same object in other frames in object recognition (when this object changes its appearance).

Final Conclusions

We hope that this technical report is taken as what it is, a guideline of existing methods for both Feature Detection and Feature Description than can be consulted when it is necessary to apply these kind of methods. The descriptions of all of them have been done having in mind this aim but we offer in the bibliography section references for basic articles (which define the methods in full) and articles that show some recent applications that use some of the methods, so we suggest to extend the information provided in this report when facing a more concrete task.

Acknowledgements

The authors would like to thank the contribution of Dr. David Gerónimo, Dr. Joost van der Weijer and Dr. Antonio López to the discussion. This work was supported in part by a research grant from Universitat Autònoma de Barcelona 471-01-3/08, by the Spanish Government through the founded project ”COLOQ-QA” (TIN2009-10435) and by research programme Consolider-Ingenio 2010: MIPRCV (CSD2007-00018).
Bibliography


101


102


106


