UNIVERSITÀ DEGLI STUDI DI CATANIA

DOTTORATO DI RICERCA IN INFORMATICA – XXIII ciclo

Tesi di Dottorato

Feature Extraction and Randomized Learning for Image Analysis and Classification

Tony MECCIO

Tutor	**Coordinatore**
Chia.mo Prof. Sebastiano Battiato	Chia.mo Prof. Domenico Cantone

ANNO ACCADEMICO 2009-2010

To my family, my girlfriend, my friends, and my co-workers.

Contents

1. Introduction

Computer Vision, the discipline of empowering machines with the ability to extract semantic information from visual data, boomed as a field of study in the last few decades, helped by the impressive evolution of modern computers, which have become powerful enough to easily process large amounts of data. This field of study is closely intertwined with other disciplines: biological vision, whose physiology is often taken into account when designing artificial vision systems; machine learning, since the employed methodologies often involve analyzing large data sets to detect and learn relevant patterns; artificial intelligence, since understanding the surrounding environment is essential for fully autonomous agents (e.g., robots) to make decisions about the actions to perform and receive feedback about the actions performed.

Research in Computer Vision has grown in interest over the years and has developed a number of novel ideas and methodologies, up to the point where it became of interest for corporate research, as proven by several collaborations between academic and corporate research groups dealing with related areas of research all over the world. However, academic and corporate research are not trivial to join together, since the latter must deal with a certain range of problems which do not affect, or only affect marginally, the former:

- Developed applications must generate enough interest for the market, so that the sustained expenses for the research may (with a certain degree of probability) be recouped with the profit generated by the products implementing the applications.

- Development must follow such a schedule not to miss the "market window": the products which implement the developed applications must reach the market in time so that interest by potential customers is still high.

- When possibile, applications must be designed in such a way that they may be implemented on existing (or next to be produced) hardware, so that they may hit the market without requiring (nor waiting) the design and production of new hardware.

- Moreover, if the applications are designed to be implemented on embedded systems for small devices (e.g., mobile phones), they must be able to run in power-, memory- and energy-constrained environments.

It is then of interest to study how Computer Vision techniques and methodologies may be properly applied in corporate research, taking into account all the related issues. The focus of this thesis is on studies which have been undertaken about designing Computer Vision applications for image analysis and classification in embedded systems. This research was funded by STMicroelectronics, AST Imaging, Catania Lab, where it was performed, and it is part of the activities of the Joint Lab between STMicroelectronics and the Image Processing Lab of the University of Catania.

More specifically, Chapter 2 explains a paradigm of image analysis based on the selection and representation of peculiar interest points. Chapter 3 discusses the problem of red eye removal in digital photography, surveying the most recent techniques to address the problem, and detailing an approach based on interest points analysis. Chapter 4 deals with the problem of image hashing robust to small changes in image content, and its application in image indexing and forensic analysis. Chapter 5 treats a methodology of learning based on random extraction of decision functions, and shows how it can be used for image categorization. Last, Chapter 6 explores the issues related to the creation of an image database for training and testing of image classification algorithms.

Publications produced

- D. Ravì, T. Meccio, G. Messina and M. Guarnera. Jbig for printer pipelines: A compression test. In *2009 Computational Color Imaging Workshop (CCIW09)*, 2009.

- S. Battiato, M. Guarnera, T. Meccio and G. Messina. Red eye detection through bag-of-keypoints classification. In Berlin Heidelberg 2009, editor, *LNCS 5646*, pages 180 - 187, Salerno, Italy, Semptember 2009. 15th International Conference on Image Analysis and Processing, Springer-Verlag.

- T. Meccio and G. Messina. *Image Processing for Embedded Devices - From CFA data to image/video coding*, volume 1, chapter 8. Red Eyes Removal. S. Battiato and A. Bruna and G. Messina and G. Puglisi, Bentham edition. 2010.

- T. Meccio and G. Messina. *Image Processing for Embedded Devices - From CFA data to image/video coding*, volume 1, chapter 9. Video Stabilization. S. Battiato and A. Bruna and G. Messina and G. Puglisi, Bentham edition. 2010.

- S. Battiato, G.M. Farinella, G.C. Guarnera, T. Meccio, G. Puglisi, D. Ravì, R. Rizzo, *Bags of Phrases with Codebooks Alignment for Near Duplicate Image Detection*, in Proceedings of ACM Workshop on Multimedia in Forensics, Security and Intelligence (MiFor 2010), in conjunction with the 2010 ACM Multimedia (ACM-MM).

Part I

Image Analysis using Interest Points

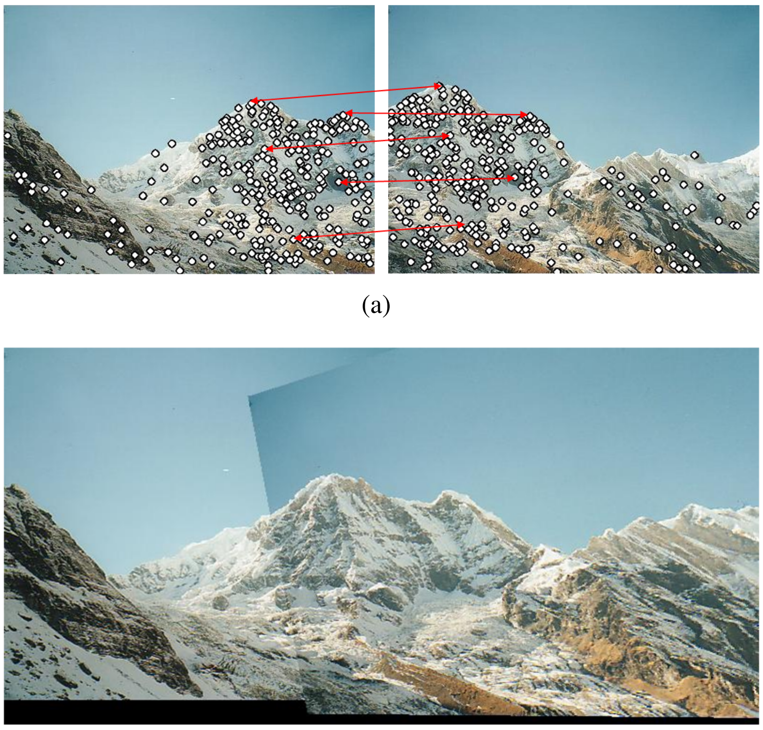
2. Interest Points

2.1 Overview

Image analysis, the extraction of meaningful information from images, may be performed in a variety of ways. Global statistics (e.g., about texture or spatial envelope) some general information about the scene [77, 82]. Frequency analysis (e.g., Fourier Transform) may give some understanding about the objects represented in the image [92, 91]. Detecting lines and estimating their 3D position is useful to perform 3D reconstruction of the scene from a single image or set of images [88, 95]. According to each particular application for which image analysis is performed, there is a lot of information which can be extracted from an image, and a lot of different tools which can be used to extract it. One of such tools is the selection and characterization of distinctive points in the image, which are called interest points, feature points or keypoints.

Interest points are characteristic points of an image which, for their peculiar properties, especially geometric and photometric invariances, may be used for a number of applications, ranging from image alignment and motion tracking to object recognition and image indexing. The main idea behind interest points is the ability to correctly locate and match corresponding points in different images depicting the same object(s). For example, given an image pair it is possible to locate feature points in both, then look for a proper matching between them. It is then possible to create a "panorama" using the feature point pairs to properly align the images (Fig. 2.1). It is also possible, given a picture of an object, to find that object in another image, even in a different position, under a different illumination or from a (slightly) different 3D viewpoint [61,62]

(Fig. 2.2).


(a)

(b)

Figure 2.1 : Example of feature point matching for panorama composition.

Keypoint-based analysis consists of two steps: keypoint detection [93] and keypoint description [71]. The goal of the first step is to select points in the image which are:

- distinctive of the scene, that is, its surrounding content carries meaningful information;
- well localized, that is, its position is well defined and its surrounding content varies greatly with a small displacement of the point;
- repeatable, that is, it is selected in other images of the same object/scene, even under a certain degree of transformations.

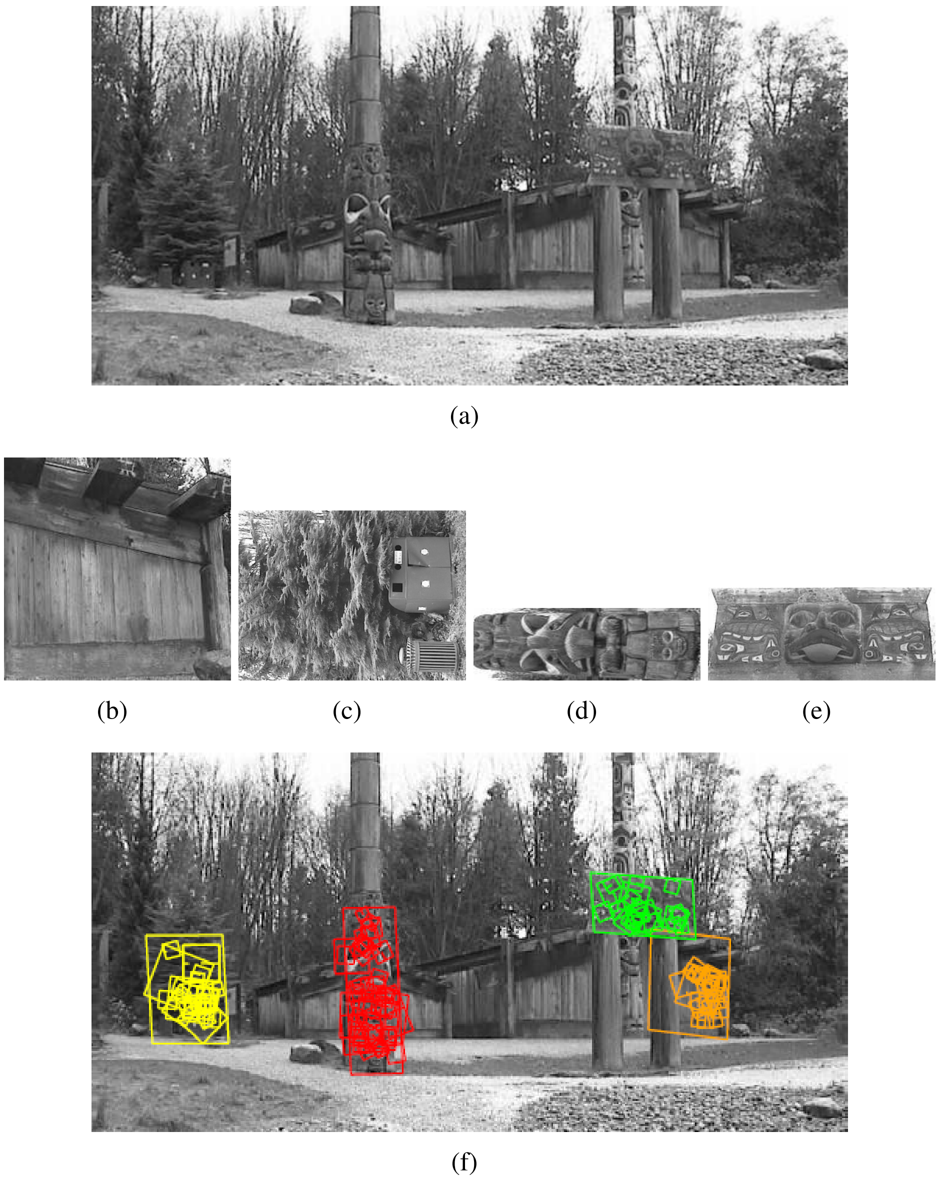


Figure 2.2 : Example of object detections using interest points. *(a)*: Original scene; *(b)* through *(e)*: objects to detect; *(f)*: detection results.

The goal of the second step is to provide a representation of the visual content surrounding each keypoint which is:

- discriminative, so that descriptors of different keypoints are very different;
- stable, so that descriptors of corresponding keypoints in different images are very similar (therefore easily matched).

For repeatability and stability, it is extremely important for keypoints to be invariant to a wide range of transformations which may occur between different images of the same object(s) or scene. Among the most common are translation, scaling, rotation, illumination changes and viewpoint changes. While translation is trivially addressed, since keypoint detection is homogeneous across the image and all subsequent analysis is performed relative to the estimated position of the keypoint, the other transformations are to be taken into account, either in the detection or in the description step.

2.2 Keypoint Detection

2.2.1 Multi-Scale Representations

The detection of keypoints, that is, the fact that certain points of the image are selected to be points of interest, depends upon their "surrounding" content. This poses the problem of what is meant by "surrounding": more specifically, how close a point can be to be considered part of the "surrounding content", or equivalently, how large the "neighborhood" of each point is. Defining a priori a certain size for the neighborhood is not an acceptable choice, except in very peculiar cases, because it causes only image structures at or near that particular size to be properly detected, while smaller or larger structures are typically missed (smaller ones give low responses, larger ones cannot be properly detected as a whole). Moreover, corresponding keypoints in different images may be present at different scales, due to a global transformation of the object/scene. It is then necessary to adopt strategies which consider suitable image structures at different scales. This goal is usually achieved by considering resampled versions of the original image at different scales and/or using variable-sized operators to detect the keypoints.

One multi-scale representation of images is the Gaussian Pyramid [28]. It consists in a set of down-sampled versions of the original image (Fig. 2.3). In this representation, the first level is the original image, and each subsequent level is obtained by smoothing and sub-sampling the previous one. Given an image:

$$I : \mathbb{R}^2 \rightarrow \mathbb{R}, \tag{2.1}$$

its Gaussian Pyramid is defined as:

$$P : \mathbb{R}^2 \times \mathbb{N} \rightarrow \mathbb{R} \tag{2.2}$$

$$P(x,y,0) = I(x,y) \tag{2.3}$$

$$P(x,y,n+1) = S\left(G(x,y,\sigma) * P(x,y,n)\right), \tag{2.4}$$

where S is a sub-sampling operator, $G(x,y,\sigma)$ is the 2-dimensional Gaussian kernel with standard deviation σ, and the convolution is computed along the x and y axes. Generally, at each level the image dimensions are halved, and $\sigma = 2$.

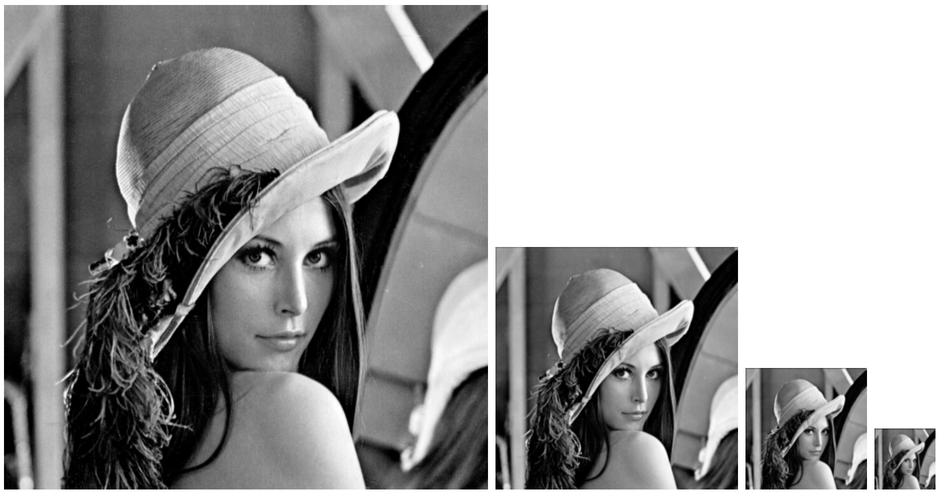


Figure 2.3 : Four levels of a Gaussian pyramid.

Another way to produce a multi-scale representation of an image is the scale-space [97, 57, 60]. It is based upon a continuous scale parameter. It is obtained by convolution of the image with Gaussian kernels of increasing standard deviation σ:

$$L : \mathbb{R}^2 \times \mathbb{R}_+ \rightarrow \mathbb{R} \tag{2.5}$$

$$L(x,y,0) = I(x,y) \tag{2.6}$$

$$L(x,y,t) = G\left(x,y,\sqrt{t}\right) * I(x,y), \tag{2.7}$$

where t is the scale parameter. The idea underlying the scale-space representation is the creation of a family of "low-pass" images, whose fine-scale information is progressively suppressed (Fig. 2.4).

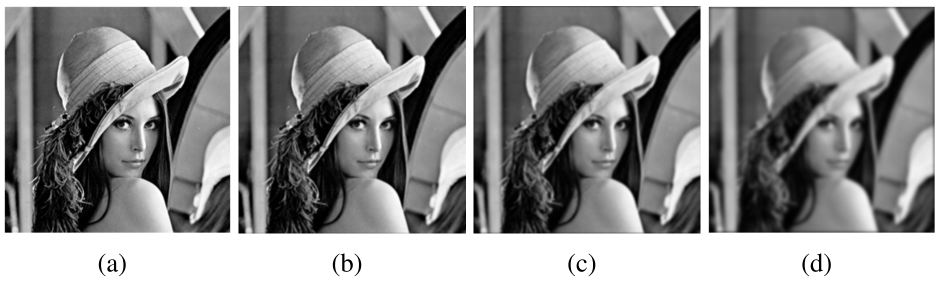


Figure 2.4 : Four scales of a scale-space.

The 2-dimensional Gaussian function is separable in the product of the two 1-dimensional components:

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-x^2-y^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}} \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-y^2}{2\sigma^2}} = g(x,\sigma)\, g(y,\sigma), \tag{2.8}$$

where $g(\cdot,\sigma)$ is the 1-dimensional Gaussian kernel with standard deviation σ. This allows to evaluate a 2-dimensional convolution by cascading two 1-dimensional convolutions, which is computationally much less expensive to compute:

$$\begin{aligned} &G(x,y,\sigma) * I(x,y) = \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} I(\xi,\upsilon)\, G(x-\xi, y-\upsilon, \sigma)\, d\xi d\upsilon = \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} I(\xi,\upsilon)\, g(x-\xi,\sigma)\, g(y-\upsilon,\sigma)\, d\xi d\upsilon = \\ &= \int_{-\infty}^{+\infty} \left(\int_{-\infty}^{+\infty} I(\xi,\upsilon)\, g(x-\xi,\sigma)\, g(y-\upsilon,\sigma)\, d\xi \right) d\upsilon = \\ &= \int_{-\infty}^{+\infty} \left(\int_{-\infty}^{+\infty} I(\xi,\upsilon)\, g(x-\xi,\sigma)\, d\xi \right) g(y-\upsilon,\sigma)\, d\upsilon = \\ &= I(x,y) * g(x,\sigma) * g(y,\sigma), \end{aligned} \tag{2.9}$$

where the two 1-dimensional convolutions are computed along the respective axes.

When the scale-space is precalculated at certain scales, to speed up its construction and any subsequent calculation the image may be sub-sampled at regular scale intervals, generating sets of images called octaves (Fig. 2.5). Generally, each octave has half the height and width of the previous one, and is filtered with the same Gaussian kernels, but the corresponding scales are higher: halving both image dimensions is equivalent to doubling the standard deviation of the kernels, which means increasing the scale parameter by a factor 4, which is consistent with the size of images decreasing to one quarter.

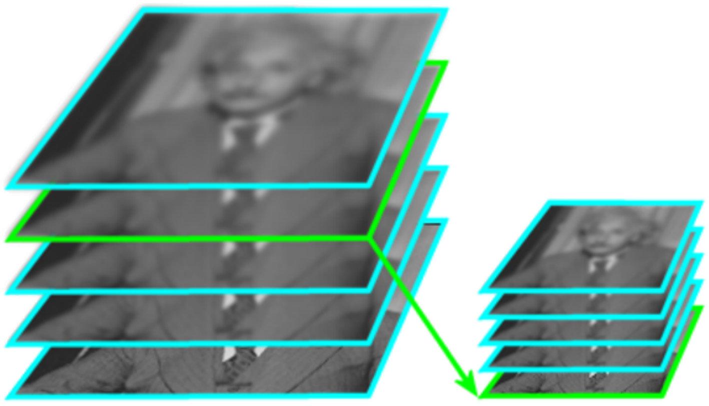


Figure 2.5 : Subdivision of the scale-space into octaves.

If the detector is based on derivatives (and it usually is) it is possible to use an alternative method based on the derivatives of the Gaussian. To that purpose, it can be shown that blurring and then differentiating an image is equivalent to differentiating the Gaussian kernel:

$$\begin{aligned} L_x(x,y,t) &= \\ &= \frac{\partial}{\partial x}\left(G\left(x,y,\sqrt{t}\right) * I\left(x,y\right)\right) = \frac{\partial}{\partial x}\left(G\left(x,y,\sqrt{t}\right)\right) * I\left(x,y\right) = G_x\left(x,y,\sqrt{t}\right) * I\left(x,y\right). \end{aligned} \tag{2.10}$$

The same applies to the derivatives with respect to the y axis, and for any subsequent order of derivatives, both pure and mixed. It is then possibile to build a family of Gaussian differentiation kernels (Fig. 2.6). The scale parameter t is also called differentiation scale.

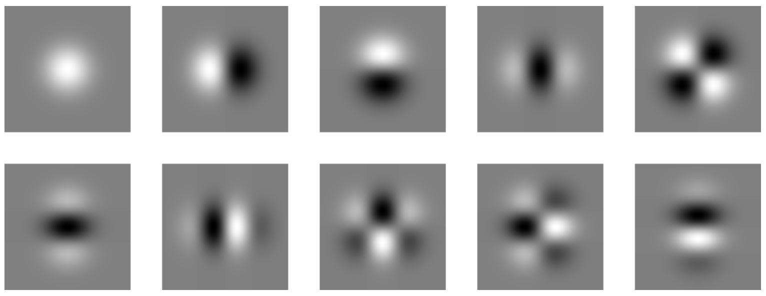


Figure 2.6 : Family of Gaussian differentiation kernels. Top row: G, G_x, G_y, G_{xx}, G_{xy}. Bottom row: G_{yy}, G_{xxx}, G_{xxy}, G_{xyy}, G_{yyy}.

Convolution with Gaussian differentiation kernels, in the same way as seen above for Gaussian kernels, is separable (at any order). Thus, it is possible to perform two 1-dimensional convolutions instead of a 2-dimensional one. The differentiation order of each 1-dimensional Gaussian is the same as the 2-dimensional kernel along the corresponding axis.

The differentiated Gaussian scale-space may be precalculated, possibly subdividing it into octaves as seen above. Its memory requirement is of course higher than the simple Gaussian scale-space, but a greater amount of pre-made calculations is provided for the algorithms to use. As an alternative, it is possible to compute the Gaussian derivatives “on the fly” each time they are needed. Thus, the trade-off between speed and memory requirements of scale-space-based analysis may be customized for each system.

Another convenient property of Gaussian differentiation kernels is provided by the linearity of the convolution operator, which allows to define more complex operations by directly precalculating the kernels, as is the case of the Laplacian-of-Gaussian operator (see Sec. 2.2.3).

Regardless of the method used to obtain a multi-scale representation, it is important, for the reasons explained above, that the search for keypoint is performed in a wide range of scales. The scale found for each keypoint must be also taken into account when computing descriptors, either by considering image data at the appropriate precalculated scale or by computing pixel intensities and/or derivatives applying the appropriate Gaussian smoothing.

2.2.2 Corner-Based Detectors

Corners are characteristic points, which for their nature are good choices for feature points. The Harris corner detector [44] considers as an estimate for the ”cornerness” of a point the weighted SSD (Sum of Squared Differences) function between its local neighborhood and a shifted neighborhood:

$$SSD(x,y) = \sum_{(u,v)\in W} w(u,v)\left(I(u+x,v+y)-I(u,v)\right)^2, \tag{2.11}$$

where W represents the weighting window centered over the point of interest and $w(u,v)$ is the weight associated to each pixel. Usually a Gaussian function is used, to avoid abrupt transitions between pixels inside and outside the window. To avoid isotropy problems which arise from directly calculating the SSD from the values of nearby pixels, the Harris detector considers the first-order Taylor expansion centered on (u,v):

$$I(u+x,v+y) \approx I(u,v)+I_x(u,v)x+I_y(u,v)y. \tag{2.12}$$

Applying this approximation, the weighted SSD becomes:

$$SSD(x,y) \approx \sum_{(u,v)\in W} w(u,v)\left(I_x(u,v)x + I_y(u,v)y\right)^2, \tag{2.13}$$

the squared sum may be written in matrix form:

$$\begin{aligned} SSD(x,y) &\approx \\ &\approx \sum_{(u,v)\in W} w(u,v) \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} I_x^2(u,v) & I_x(u,v)I_y(u,v) \\ I_x(u,v)I_y(u,v) & I_y^2(u,v) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \\ &= \begin{bmatrix} x & y \end{bmatrix} A \begin{bmatrix} x \\ y \end{bmatrix}. \end{aligned} \tag{2.14}$$

The matrix A thus defined:

$$A = \sum_{(u,v)\in W} w(u,v) \begin{bmatrix} I_x^2(u,v) & I_x(u,v)I_y(u,v) \\ I_x(u,v)I_y(u,v) & I_y^2(u,v) \end{bmatrix} \tag{2.15}$$

is called autocorrelation, second-moment or Harris matrix, and describes how the values of the pixels vary in the neighborhood of the point being considered. To maintain isotropy, the eigenvalues of A are considered. These eigenvalues are proportional to the principal curvatures of the image around the point, and describe (in a rotation invariant way) its cornerness. Three cases may be considered:

- if both eigenvalues have small values, the point belongs to a flat region;
- if one of the eigenvalues is big and the other is small, the point belongs to an edge;
- if both eigenvalues are big, the point belongs to a corner.

To avoid directly computing the eigenvalues, since an estimate of their magnitude is needed rather than their exact values, a corner response function is used, defined as:

$$R = \det(A) - k\operatorname{tr}(A)^2, \tag{2.16}$$

where k is a tuning parameter usually set in the range $[0.04, 0.06]$. Since the determinant of A is the product of its eigenvalues, while the trace is their sum, in order for the R function to have an high value both the eigenvalues must be big, otherwise the negative term is larger than the positive one. The Harris corner detector selects as interest points the local maxima of R which surpass a certain threshold. Varying the threshold, it is possible to extract a variable number of points.

The Harris corner detector aforementioned is not scale-invariant, since it uses punctual differentiation and uses a fixed-size weighting window. To overcome this drawback, the multi-scale Harris corner detector can be used. It basically consists of an adaptation where the derivatives are computed in the scale-space (using one of the methods seen earlier) and the weighting window is sized accordingly:

$$A = t \sum_{(u,v)\in W_I} w_I(u,v) \begin{bmatrix} L_x^2(u,v,t) & L_x(u,v,t)L_y(u,v,t) \\ L_x(u,v,t)L_y(u,v,t) & L_y^2(u,v,t) \end{bmatrix}; \tag{2.17}$$

the square of standard deviation of the Gaussian weighting window is called integration scale, and is usually proportional to the differentiation scale (e.g. by a factor 2), to avoid introducing an unnecessary degree of freedom in detection of the corners. The t which multiplies the entire matrix acts as a normalization factor, and is used to obtain coherent ranges of values at different scales.

The multi-scale Harris corner detector, if applied independently at different scales, detects the same corners multiple times, at a different position for each scale. It is then necessary to select the characteristic scale of each corner. The simplest approach is to look for the local maxima of the cornerness function with respect to x, y and t. However, this does not yield satisfactory results, since in lots of cases the cornerness does not reach a well defined maximum along the scale axis, then the corresponding corners are not detected. This problem is solved by the hybrid Harris-Laplace detector (see Sec. 2.2.3).

2.2.3 Blob-Based Detectors

Blobs, in the context of interest point detection, are a geometrical concept in some extent complementary to corners. Blobs are approximatively circular regions whose intensity is distinctively higher or lower than those of their surrounding. Since their position is well defined, they can be used as interest points.

An useful operator to detect blobs is the Laplacian operator:

$$\nabla^2 I(x,y) = I_{xx}(x,y) + I_{yy}(x,y)\,. \tag{2.18}$$

It has big (positive or negative) values in points whose value is very different from its neighbors.

Using the Gaussian differentiation kernels to calculate the second-order derivatives (or, equivalently, computing derivatives directly in the precalculated scale-space), and normalizing the response across scales by multiplying by t, the Laplacian-of-Gaussian (LoG) operator is obtained:

$$\begin{aligned} t\nabla^2 L(x,y,t) &= t\left(L_{xx}(x,y,t) + L_{yy}(x,y,t)\right) = \\ &= t\left(\frac{\partial^2}{\partial x^2} G\left(x,y,\sqrt{t}\right) * I(x,y) + \frac{\partial^2}{\partial y^2} G\left(x,y,\sqrt{t}\right) * I(x,y)\right) = \\ &= t\nabla^2 G\left(x,y,\sqrt{t}\right) * I(x,y). \end{aligned} \tag{2.19}$$

The LoG operator has high responses (in absolute value) in presence of blobs whose size is near t. This allows to find not only the position, but also the characteristic scale of each blob. The selected keypoints are, then, the local extrema (maxima and minima) of the LoG function with respect to x, y and t. By imposing a threshold on the absolute value of the function, low contrasted (then unstable) extrema may be discarded. Moreover, since this operator has high responses in presence of edges, which is usually not desirable since they are not well localized, a further filtering may be applied, based on the Hessian matrix of the LoG function [62]. Given the Hessian matrix:

$$H(x,y,t) = \begin{bmatrix} L_{xx}(x,y,t) & L_{xy}(x,y,t) \\ L_{xy}(x,y,t) & L_{yy}(x,y,t) \end{bmatrix} \tag{2.20}$$

its eigenvalues are proportional to the principal curvatures of the LoG function around the point, similarly to what seen in Sec. 2.2.2 for the second-moment matrix. It is then desirable that the ratio between the highest and the smallest eigenvalue does not exceed a certain threshold. Assuming that the determinant of H is positive (otherwise the point is a saddle rather than a blob), be α the eigenvalue with the highest magnitude, β the other one, and $r = \alpha/\beta$, the following relationships hold:

$$\frac{\mathrm{tr}(H)^2}{\det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}. \tag{2.21}$$

Since the quantity $(r+1)^2/r$ increases monotonically with r, to impose a maximum threshold on r it is sufficient to impose a corresponding threshold on it to select the final keypoints.

The LoG operator can also be used in conjunction with the Harris corner detector. As seen in Sec. 2.2.2, searching for the local maxima of the cornerness function with respect to x, y and t does not yield satisfactory results. Instead, the LoG operator is much more accurate when selecting the characteristic scale of corners. This hybrid combination of detectors is called Harris-Laplace detector, and selects feature points which are simultaneously local maxima of the cornerness function with respect to x and y and local extrema of the LoG over t.

An operator which is often used to approximate the LoG is the Difference-of-Gaussians (DoG). It consists in subtracting two subsequent scales of the scale-space representation to obtain a band-pass representation which only contains details at a certain scale. This is very efficient when the scale-space is preconstructed (possibly with octaves) but the derived scale-space is not, because it is very fast to compute and does not require subsequent differentiation to extract keypoints.

The fact that the DoG approximates the LoG can be shown using the heat diffusion equation (parametrized, for convenience, in terms of σ rather than t) [62]:

$$\frac{\partial G(x,y,\sigma)}{\partial \sigma} = \sigma \nabla^2 G(x,y,\sigma); \tag{2.22}$$

then, assuming that two subsequent scales are σ and $k\sigma$, with constant k, and using a finite difference to approximate the derivative:

$$\sigma \nabla^2 G(x,y,\sigma) = \frac{\partial G(x,y,\sigma)}{\partial \sigma} \approx \frac{G(x,y,k\sigma) - G(x,y,\sigma)}{k\sigma - \sigma} \tag{2.23}$$

from which follows:

$$G(x,y,k\sigma) - G(x,y,\sigma) \approx (k-1)\sigma^2\nabla^2 G(x,y,\sigma)\,. \tag{2.24}$$

This shows that, when subsequent scales are spaced by a constant factor k, the DoG operator produces an approximation of the LoG, except for a constant coefficient $k-1$. It does not affect extrema location, which can be performed simply by comparison of each point not only in the adjacent positions, but also in the adjacent scales. A further refinement of the pixel position may be obtained by applying a quadratic interpolation to the DoG function at the extremum and then looking for the extremum of the interpolated function [62].

Another detector is based of the determinant of the Hessian matrix of the image:

$$\det HI(x,y) = I_{xx}(x,y)\,I_{yy}(x,y) - I_{xy}(x,y)^2\,. \tag{2.25}$$

This determinant has high positive values in one point when the intensities of the neighborhood pixels vary with the same sign along both principal curvatures. Using the Gaussian differentiation and applying the normalization factor t^2:

$$t^2 \det HL(x,y,t) = t^2\left(L_{xx}(x,y,t)\,L_{yy}(x,y,t) - L_{xy}(x,y,t)^2\right). \tag{2.26}$$

This function has local maxima in presence of blobs whose size is near t. Imposing a minimum threshold, it is possible to discard low contrasted keypoints. Ill-localized blobs (situated along edges) are automatically filtered out, since in this cases one of the eigenvalues has a small value, therefore the determinant is not high.

To select the characteristic scale of the keypoints, local maxima over x, y and t may be looked for. However, as is the case with the Harris detector, best results are obtained using the Laplacian operator to select the scale: this hybrid detector is called Hessian-Laplace.

Another keypoint detector, called Maximally Stable Extremal Regions (MSER), is based on the thresholding operation [67]. Differently from others, it does not process derivatives in the scale-space, but acts directly on the pixels of the original image. It extracts segmented regions rather than blobs, however, since the regions are well-contrasted with the surrounding content, they share some properties with blobs, and a circular keypoint can be fitted on each region to make it behave like other detectors.

The thresholding operation transforms an image in a binary map (often represented as a black/white image) which includes only the pixels whose intensity is greater than a certain value th. When this value is lowered from the maximum (white) to the minimum (black) value, the thresholded image includes more and more pixels, which form connected regions which increase in area and merge together. Such regions are called extremal regions, since their intensity is greater than their surrounding. If a region is well-defined, the number of pixels it includes varies slowly with th. Therefore, the extremal regions are considered maximally stable, and selected as interest point, if the variation of their area with respect to th, normalized with respect to the area, reaches a local minimum. Be Q_{th} an extremal region with respect to th, if the function:

$$q(th) = \frac{|Q_{th-\Delta} \backslash Q_{th+\Delta}|}{|Q_{th}|} \tag{2.27}$$

has a local minimum in th^{*}, then $Q_{th^{*}}$ is an MSER. All regions found as MSERs, for all possible values of th, are selected. Subsequently, the image is converted to negative and the process is repeated, to select also extremal regions which are darker than the surrounding content.

A number of other keypoint detectors exist in the recent literature [93], each with its own strengths and weaknesses. Some of them were also adapted to extract elliptical keypoints, thus also gaining invariance to affine transformations, which are typical under viewpoint changes of planar objects [72]. It is worth noting that getting elliptical keypoints is also possible with the MSER detector, since it is sufficient to fit an ellipse instead of a circle on each detected region.

2.2.4 Orientation Assignment

As seen in Sec. 2.1, keypoints must be invariant to a wide array of transformations, among which is rotation. Since, generally, the descriptors used with keypoints are not rotation invariant, that is, the information they gather from pixels are not only dependent on the distance between the pixels and the keypoint center, but also on their angle, each keypoint must be assigned an orientation in such a way that it is covariant, with a good degree of reliability, with the rotation that the object may undergo between different images. In this way, by rotating the descriptor accordingly (or, equivalently, by rotating the pixels underlying the keypoint before feeding it to the description algorithm), the keypoint description can be made rotation invariant.

Orientation assignment is usually performed based on the orientation of the local image gradients around the keypoint, since it is a feature which rotates coherently with the object (except for parts not belonging to the object itself, like background objects and occlusions) and is robust under illumination changes. The simplest approach is to compute the average of the orientations of image gradients around the keypoint, weighted by their magnitude and possibly by a Gaussian function to give more importance to nearest pixels.

A more robust approach [62], which discards outliers and accounts for the possibility of more than one dominant orientation, which in turn increases stability, consists in creating a histogram of orientations, which covers all 360 degrees (usually with 36 bins). Each gradient added to the histogram is weighted according to its magnitude and a Gaussian weighting window. The highest peak and other peaks with at least 80% value of the highest are selected as dominant orientations. If more than one orientation is chosen, the keypoint is replicated, with one instance for each orientation. A parabolic fitting is performed on each peak, considering the peak itself and the two adjacent bins, to provide more accuracy in determining the orientation.

2.3 Keypoint Description

The most simple keypoint descriptor is the set of values of the local image patch, centered, sized and rotated according to the keypoint position, scale and orientation. It is then resized to a fixed (usually small) size, to have a constant-dimensional descriptor. The local patch is sensitive to illumination changes, thus to measure similarity between patches their correlation can be used:

$$r(I_1, I_2) = \frac{1}{wh} \frac{\sum_{x=0}^{w-1} \sum_{y=0}^{h-1} \left(I_1(x,y) - \overline{I_1}\right)\left(I_2(x,y) - \overline{I_2}\right)}{\sigma_{I_1} \sigma_{I_2}}, \tag{2.28}$$

where w and h are, respectively, the width and height of the patches, $\overline{I_1}$ and $\overline{I_2}$ are the averages of the values of the patches, and σ_{I_1} and σ_{I_2} are the standard deviation of the patches. By subtracting the average value and dividing by the standard deviation, invariance to linear changes in brightness and contrast is achieved.

The most widespread keypoint descriptor in literature is the SIFT descriptor [61, 62]. Although the name “SIFT” (Scale Invariant Feature Trasnform) was originally given to an entire keypoint-based object recognition methodology, the employed descriptor was widely used afterwards, and took the name on its own. The descriptor consists in a set of histograms arranged in a 4x4 square matrix around the keypoint (Fig. 2.7(a)). Each histogram overlays 4x4 pixels (at the keypoint scale) and accumulates the orientation of the gradients beneath it, weighted by their magnitude and by a Gaussian window centered on the keypoint. To avoid abrupt transition effects, a tri-linear interpolation is used to distribute each value on the nearest two bins (along the orientation axis) of the nearest four histograms (along the x and y axes). The resulting 128-dimensional vector is then normalized to unit Euclidean lenght, to be invariant to linear contrast changes (the gradients are, by themselves, invariant to constant brightness changes). To achieve some robustness to non-linear illumination changes (which usually have a noticeable impact on the magnitude but not on the orientation of gradients), if there are any values greater than 0.2, they are thresholded to 0.2 and the vector is normalized again.

A variant of the SIFT descriptor is the Gradient Location-Orientation Histogram (GLOH) descriptor. It works in the same way as the former, accumulating magnitude of local gradients in histograms, but it has a "dartboard"-like shape, with one histogram at the center and two concentric annuli extending outwards, each divided in 8 histograms (Fig. 2.7(b)). Each histogram has 16 bins which, multiplied by the 17 histograms total, amounts to a 272-dimensional vector.

Another commonly used descriptor, which was invented with a priority on efficiency, is the Speeded Up Robust Features (SURF) descriptor [22, 21]. Apart from other simplifications which the authors made, like the use of integral images and box-shaped approximation to Gaussians to detect keypoints, the SURF descriptor considers a 4x4 square matrix of histograms which cover 20x20 pixels. Instead of considering the

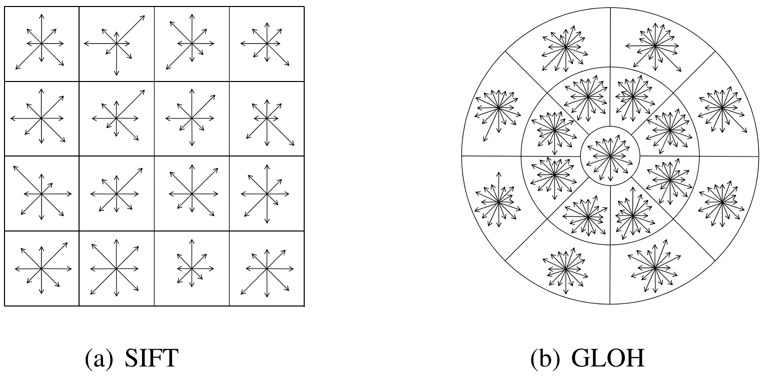

(a) SIFT (b) GLOH

Figure 2.7 : Schematic representation of the two descriptors.

orientation of gradients (like SIFT and GLOH), the SURF descriptor is based upon the Haar wavelets [64], which in practice consist in computing simple pixel differences, in one direction, at the keypoint scale. This is done independently for the horizontal and vertical direction, which increases robustness to noise. For each direction, the absolute values of the Haar wavelets are also accumulated. Thus, each histogram accumulates four values: $\sum dx$, $\sum |dx|$, $\sum dy$, $\sum |dy|$. The samples are weighted by a Gaussian window centered on the keypoint center, in the same way as the SIFT descriptor. The 64-dimensional vector is then normalized, to obtain invariance to contrast changes (pixel differences are natively invariant to brightness changes).

Other descriptors exist in recent literature which may be useful for keypoint-based image analysis [71]. Color information may be included in the keypoint description, tripling the dimensionality of the descriptors instead of considering grayscale pixel intensities, or by searching for ways to exploit color information more robustly [15]. To obtain more compact representations, Principal Component Analysis [53] can be applied to the descriptor, using a reference dataset to pre-compute the basis of the reduced-dimensionality space into which descriptors are to be projected [55]. Software is available on the Internet which allows to experiment with the keypoint detectors and descriptors discussed in this chapter and many others [3, 5, 11, 7, 12].

3. Red Eye Removal

3.1 Introduction

Red eye artifacts are a well-known problem in digital photography. They are caused by direct reflection of the light of the flash from the blood vessels of the retina through the pupil to the camera objective. When the flash is fired, light reflected from the retina forms a cone, whose angle α depends on the opening of the pupil. Be β the angle between the flash-gun and the camera lens (centered on the retina), the red eye is formed if the red light cone hits the lens, that is, if α is greater than β (Fig. 3.1). Small compact devices and point-and-click usage, typical of non-professional photography, greatly increase the likelihood for red eyes to appear in acquired images.

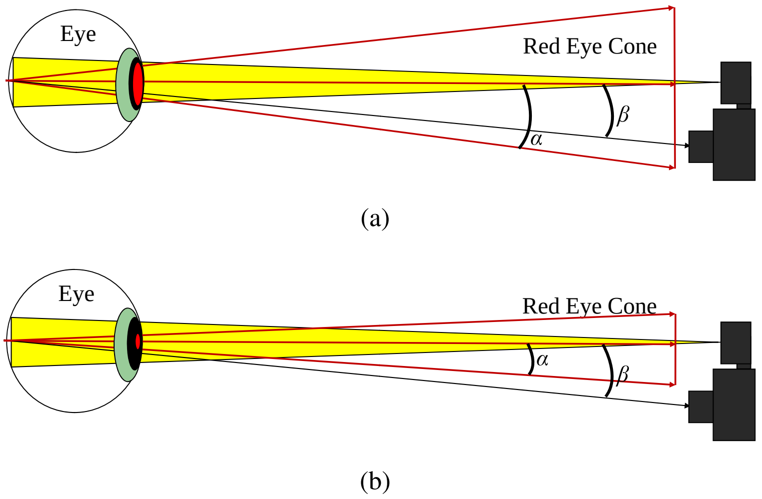


Figure 3.1 : The red eye phenomenon depends upon various factors, including the distance between the flashgun and the objective and the opening of the pupil.

High-end cameras often have a separate flashgun with a long bracket, which spaces out the flash from the lens, reducing the probability for red eyes to appear. Another possible measure is to make additional flashes before taking the photograph (pre-flash). This method, first proposed by Kodak [73], gives eyes time to react and shrink and is quite effective, but has the disadvantage of greatly increasing power consumption, and to be somewhat uncomfortable to people.

Red eye prevention methods reduce the red eye probability but cannot remove it entirely. Most of the times the acquired pictures must be corrected during post-processing, which is a very challenging task: red eyes vary in shape and color, and even in position and size relative to the whole eye. Sometimes light is reflected on a part of the retina not covered with blood, creating a yellow or white reflection (golden eyes). Some examples of the variability of the phenomenon are shown in Fig. 3.2. Designing a system which can effectively cope with such variability is very difficult [38, 39, 68].

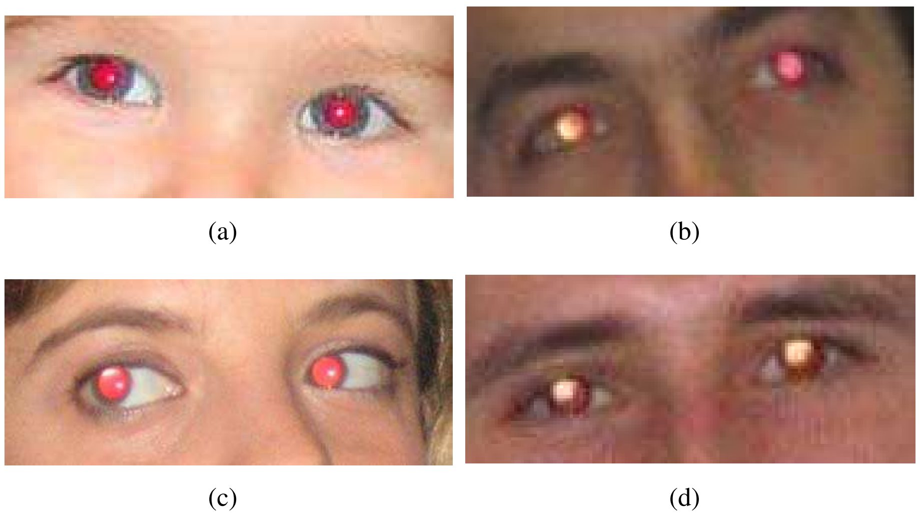


Figure 3.2 : Examples of the variability of the red eye phenomenon. Golden eyes are also visible.

Red eye removal requires first reliable detection, then proper correction. Detection methods are divided into semi-automatic methods, which ask the user to manually localize and point the red eyes, and automatic methods, which detect the red eyes themselves. In the first case the eyes are manually selected using a visual interface (e.g., Adobe Photoshop [2], Corel Paint Shop Pro [4], ACDSee [1], etc.). Eyes are easy to localize for men, but requiring manual intervention for every picture taken is undesirable for non-professional usage; moreover, on mobile devices, it may be difficult to have such an interface.

Automatic methods attempt to find red eyes on their own. They are easier to use and more appealing, however automatic detection of red eyes is a very challenging task, due to the variability of the phenomenon and the difficulty in distinguishing eyes from other details.

Red eye correction, on the other hand, may be more or less invasive. Some cases may be addressed with a soft correction, but sometimes a stronger intervention is needed. The aim is to provide a corrected image which looks as natural as possible, thus a less invasive correction is preferred when possible.

3.2 Red Eye Detection

The main difficulty in red eye detection of red eyes, as seen in Sec. 3.1, is their great variability. In the easier cases, the pupil differs from a regular one only by its color. However, it is not uncommon for the red reflection to spread over the iris and show an unnatural luminance distribution. Usually a small white glint is also present, representing the direct reflection of the flash on the eye and giving the eye much more naturalness.

Typical red eye detection approaches involve extraction of red zones combined with skin extraction, shape template matching, and/or face detection. Some approaches also make use of classifiers to further refine their results.

3.2.1 Color Based

Color based approaches are the simplest ones. They are based on detecting red zones which are possible red eyes. In most cases they also detect the human skin, then consider some criteria about the relative position of red eyes and skin (usually, the eyes must be surrounded by nearby skin). Some color based approaches also detect the sclera (the white part of the eye), distinguishing it from the skin. Additional constraints may be imposed about the geometry of the red zones. This kind of approaches is quite simple, but does not take into account more complex features like, e.g., the various parts of the eye or the face.

One of the biggest problems of color-based techniques is characterizing the “red eye” color. Usually, interesting portions of the color space (corresponding to red, skin color, etc.) are delimited by hard thresholds, but they may also delimited by soft margins, giving a fuzzy probability for the color to belong to the region. Finding proper boundaries for the regions is a challenging task: the color of red eyes is heavily influenced by the type of flash used, the sensor and the processing pipeline. While this is not a big issue, since the thresholds may be fine-tuned to adapt to the acquisition system, there are external factors which may influence the color of the eyes: age, distance, eye opening, angle, etc. Even one subject in one picture may have two different red eyes, or a red eye and a regular one (Fig. 3.3).

The red color region may be defined in different color spaces. In the RGB space, a possible definition is [100]:

$$\begin{cases} R > 50 \\ R/(R+G+B) > 0.40 \\ G/(R+G+B) < 0.31 \\ B/(R+G+B) < 0.36 \end{cases} \tag{3.1}$$

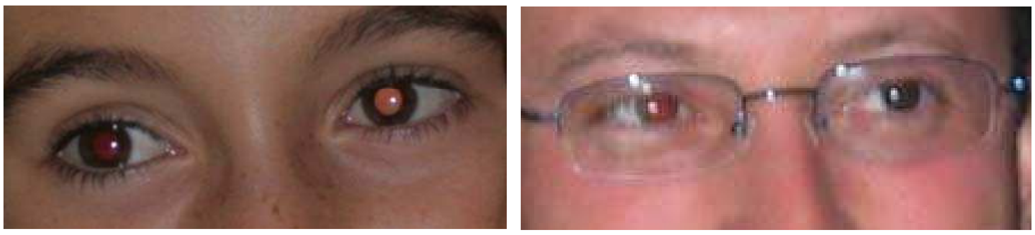


Figure 3.3 : Picture (a) shows two very different red eyes; picture (b) shows one red eye along with a regular one.

Often, instead of hard thresholds, a *Redness* function is provided, which is used to define soft margins for the red color region. Some possible redness functions are [47, 40, 89, 37]:

$$Redness = (R - \min\{G, B\}) \tag{3.2}$$

$$Redness = \frac{R^2}{(G^2 + B^2 + 14)} \tag{3.3}$$

$$Redness = \frac{\max\{0, (R - \max\{G, B\})\}^2}{R} \tag{3.4}$$

$$Redness = \max\left\{0, \frac{2R - (G + B)}{R}\right\}^2 \tag{3.5}$$

A possible alternative is to compare a redness function with a luminance function, discarding pixels whose luminance is predominant over the redness [96]:

$$Redness = R - (G + B)/2 \tag{3.6}$$

$$Luminance = 0.25R + 0.6G + 0.15B \tag{3.7}$$

$$RedLum = \max\{0, 2 \cdot Redness - Luminance\} \tag{3.8}$$

Search for red regions may be performed in other color spaces, such as YCC [81] or HSL [23]. See Sec. 3.2.4 for a complete example of red eye search and patch extraction performed in the HSL space.

Given a particular choice for the red color region, it is possible to convert each image into a "redness map". According to whether the red color region is hard-delimited or soft-delimited, the redness map is a black-and-white or full-grayscale image (in the latter case, the redness function is adjusted to the possible maxima and minima of the redness function, or to the maximums and minimums over each particular image). Figs. 3.4 and 3.5 show redness maps computed using the above formulas.

Other useful information may be gained by searching for the sclera [94] and selecting the zones which were effectively lighted by the flash [35]. Thresholding and morphological operators may be used to combine the information, as shown is Sec. 3.2.4.

3.2.2 Shape Based

Shape based look for eyes based on their shape. Generally, they use shape templates at different positions and resolutions, in order to search the image for shapes which may correspond to eye features. Using circular or square templates it is possible to recognize, e.g., the difference in intensity between the inner pupil and the outer skin and sclera. Slightly more complex templates may be useful in locating the other parts of the eye, which helps to assess the presence or the absence of a red eye [63].

Edge detection filters may also be used to extract information about shape. It is possible to use them in conjunction with color tables to make advantage of both spatial and chromatic information [84].

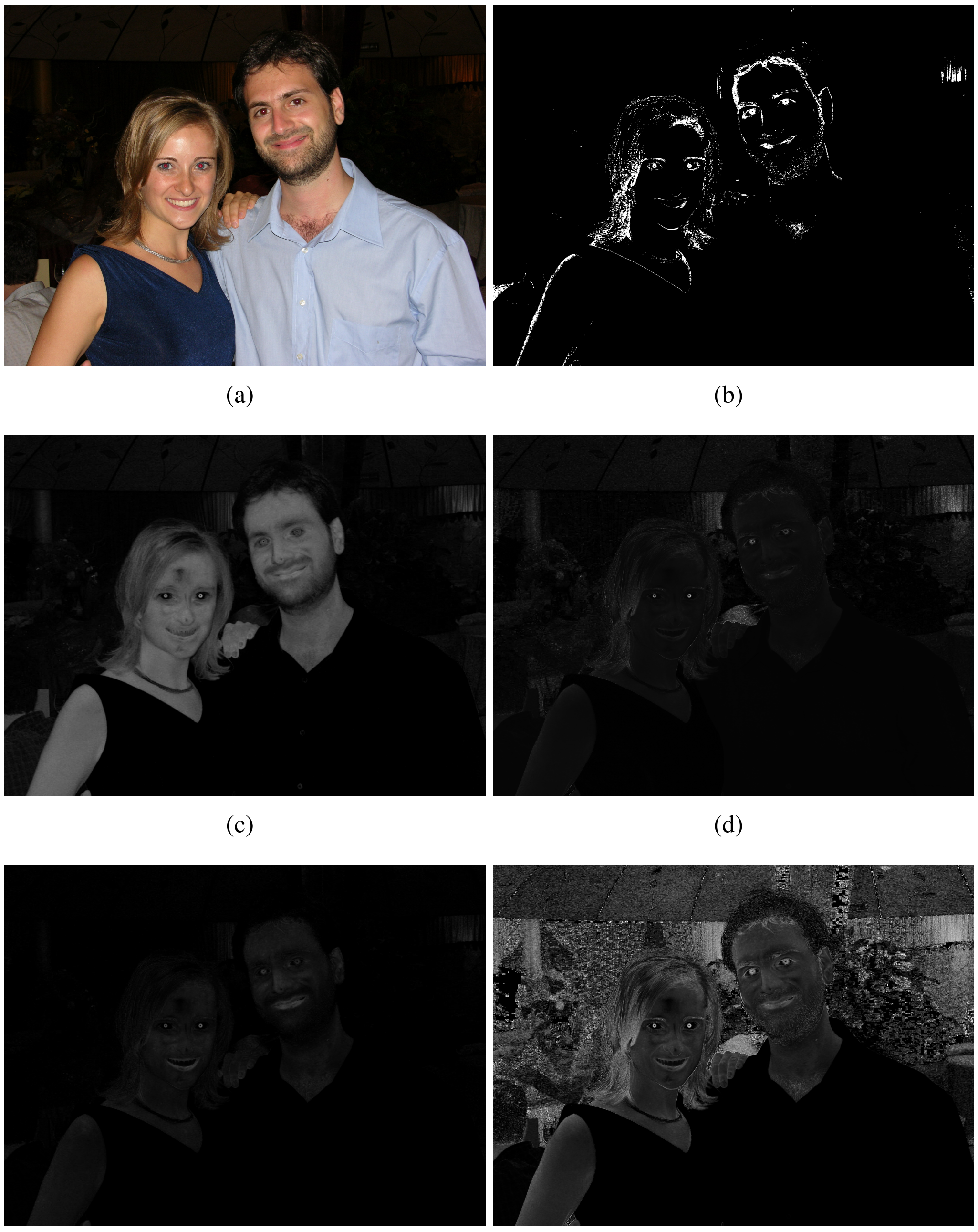
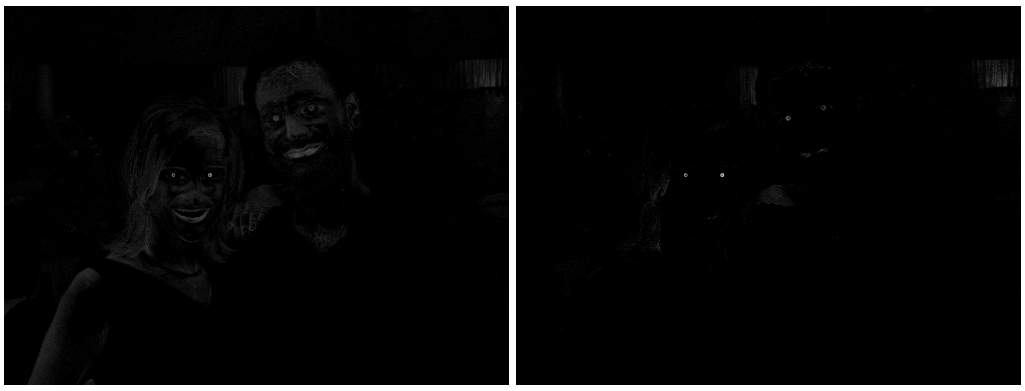


Figure 3.4 : Examples of redness maps. (a) Original image; (b-f) redness maps obtained from (3.1), (3.2), (3.3), (3.4), (3.5), respectively.



(a) (b)

Figure 3.5 : (a) Redness map obtained from (3.6); (b) redness vs. luminance map computed according to (3.8).

3.2.3 Pairing Verification

One constraint which can be useful to filter out false detections is eye pairing verification [85]. It is based on the assumption that every eye is paired with the other one on the same subject's face, with appropriate geometric constraints; otherwise, it is discarded, since it is most probably a false detection.

This approach is effective, since it is very unlikely for two false positives to satisfy the pairing criteria, but it presents a major drawback: if a face is partially occluded, or only one eye is red, or even if both eyes are red but one is not detected, the detection fails, because the paring criteria are not verified (Fig. 3.6).

3.2.4 Example

Explained here is an example of red eye detection procedure. First, the image is converted to the HSL color spaces, with $H \in [0°, 360°]$, $S \in [0,1]$, $L \in [0,1]$.

The redness map includes the pixels where $-60° \leq H \leq 20°$ and $S \geq 0.6$. A morphological closing operation is used to "clean" the redness map and fill possible small

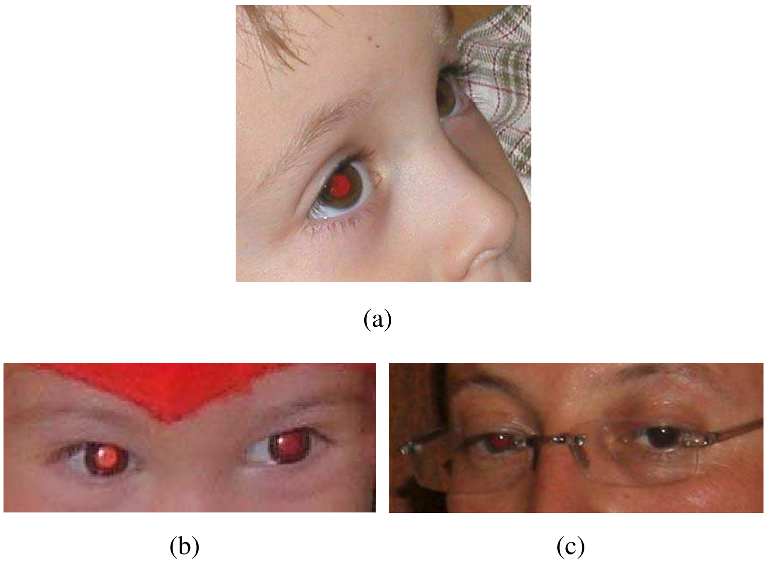


Figure 3.6 : In picture (a), only one of the eyes is visible; the red eyes in picture (b) are very different, and in most cases only one of them will be properly detected; in picture (c) only one of the eyes is affected by the red eye phenomenon. In all these cases, the pairing verification will fail.

holes. Afterward, for each connected component in the redness map, some geometrical constraints are enforced:

- the size S_i of the connected region i is within a certain range.
- the binary roundness constraint R_i, of the connected region i is verified:

$$R_i = \begin{cases} true & \rho_i \in [Min_\rho, Max_\rho];\ \eta_i \leq Max_\eta;\ \xi_i \gg 0 \\ false & otherwise \end{cases} \tag{3.9}$$

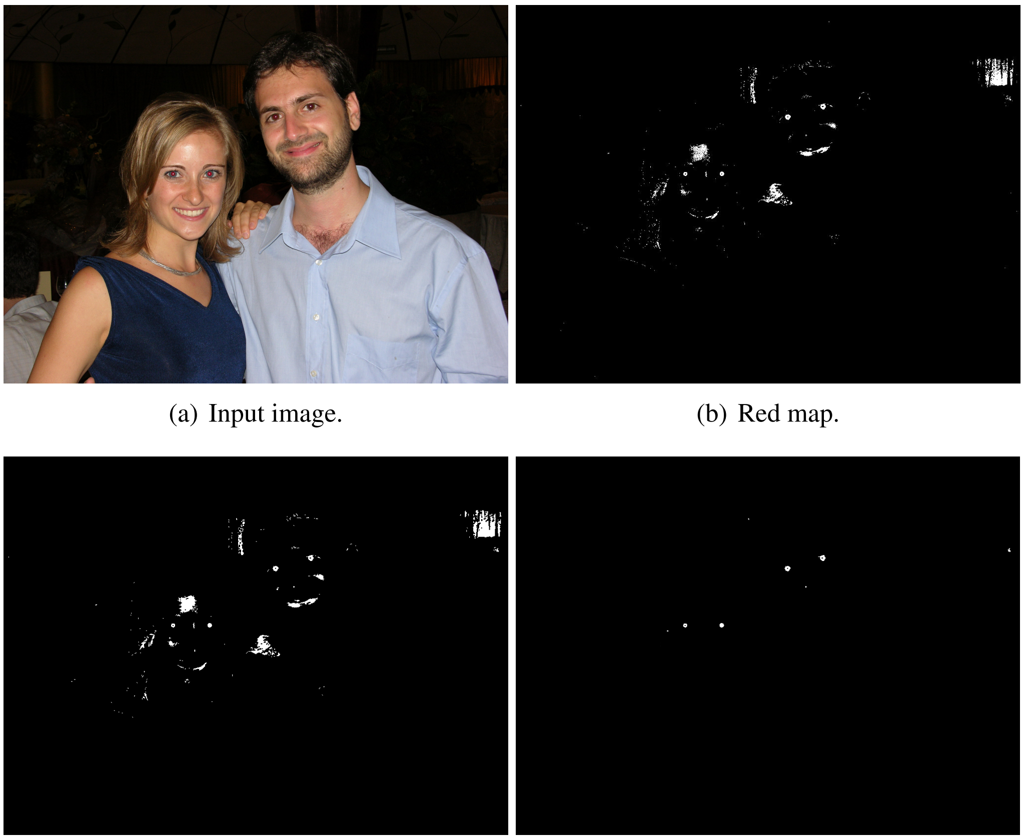
where

$\rho_i = \frac{4\pi \times A_i}{P_i^2}$ is the ratio between the estimated area A_i and the perimeter P_i of the connected region; the more this value is near 1 the more the shape will be similar to a circle: a maximum and minimum threshold on ρ is then imposed.

$\eta_i = \max\left(\frac{\Delta_{x_i}}{\Delta_{y_i}}, \frac{\Delta_{y_i}}{\Delta_{x_i}}\right)$ is the distortion of the connected region along the axes, which must be below a certain value.

$\xi_i = \frac{A_i}{\Delta_{x_i}\Delta_{y_i}}$ is the filling factor, the more this parameter is near 1 the more the area is filled.

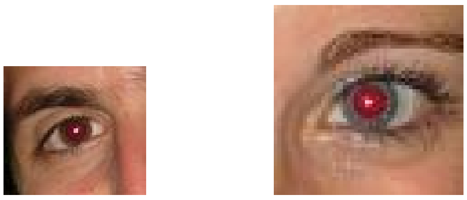
In Fig. 3.7 all these steps are illustrated. The regions which satisfy all the constraints are used to extract the red eye patches candidates from the original input image (Fig. 3.8). The extracted patches can then be resized to a fixed size for further examination.



(a) Input image. (b) Red map.

(c) Closing mask. (d) Remaining candidates.

Figure 3.7 : Red patches extraction from a CCD input image.


(a) True red eyes.

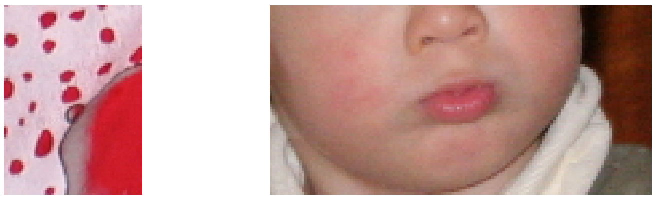

(b) False positives.

Figure 3.8 : Examples of possible candidates after red patches detection.

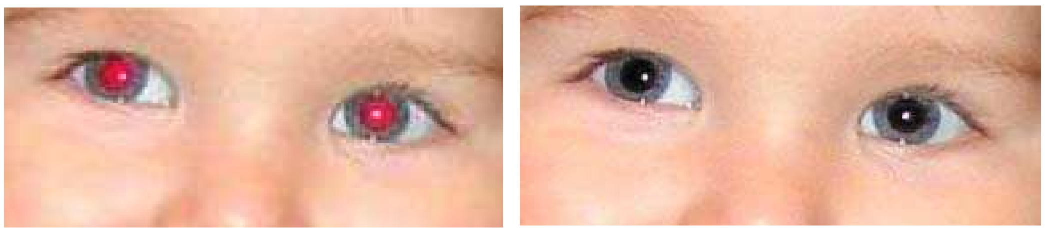
3.3 Eye Correction

The goal of red eye correction is to remove the red eyes keeping the image as natural as possible. According to the extent to which the artifact has corrupted the image, the correction algorithm may need to adjust the hue, brightness, luminance distribution, and/or even the shape and size of the pupil. Since naturalness of the image is the goal, it is best to use a minimally invasive technique to correct each case. This also means that a way to distinguish the gravity of each artifact (either in the detection phase or at the very beginning of the correction phase) is preferred, in order to adapt the correction method on a case-by-case basis [65].

3.3.1 Desaturation

In the simplest cases the artifact only consists in the wrong color of the pupil. In these cases, the optimal solution is desaturating, that is, suppressing (totally or partially) the chrominance, while leaving intact or slightly lowering the luminance (Fig. 3.9).

A simple way of desaturating red pupils is to replace each pixel with a gray shade



(a) (b)

Figure 3.9 : In the simplest cases, pupil desaturation produces good results.

at 80% of original pixel luminance [81]. An adaptive desaturation may be performed in the CIELAB color space by stretching the lightness values of the pupil so that its darkest point becomes black [43]:

$$\begin{aligned} L^*_{corrected} &= \frac{\max L^*}{(\max L^* - \min L^*)}(L^* - \min L^*) \\ a^*_{corrected} &= 0 \\ b^*_{corrected} &= 0 \end{aligned} \tag{3.10}$$

When desaturating, the transition between the corrected and uncorrected area may be noticeable and unpleasant, and some pixels outside the pupil may be incorrectly desaturated. To overcome these problems, a Gaussian mask is generally used to modulate the strength of the correction.

3.3.2 Inpainting

In the hardest cases, a more invasive correction is needed. Often, the distribution of reflected light is influenced by the direction of the flash with respect to the face. Sometimes eyes present the “washed out” effect, where the reflected light spreads off the pupil onto the iris. In these cases a simple desaturation may give unnatural results (see Fig. 3.10).

It is then necessary to use a more complex method to reconstruct a realistic image of the eye. Inpainting may vary from an adaptive recoloring of red pixels to a

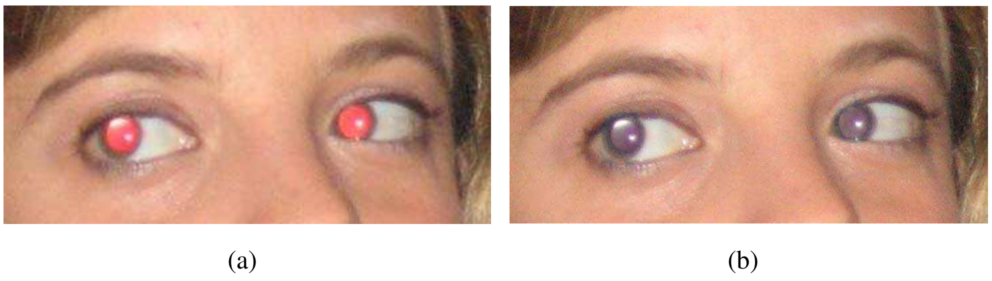
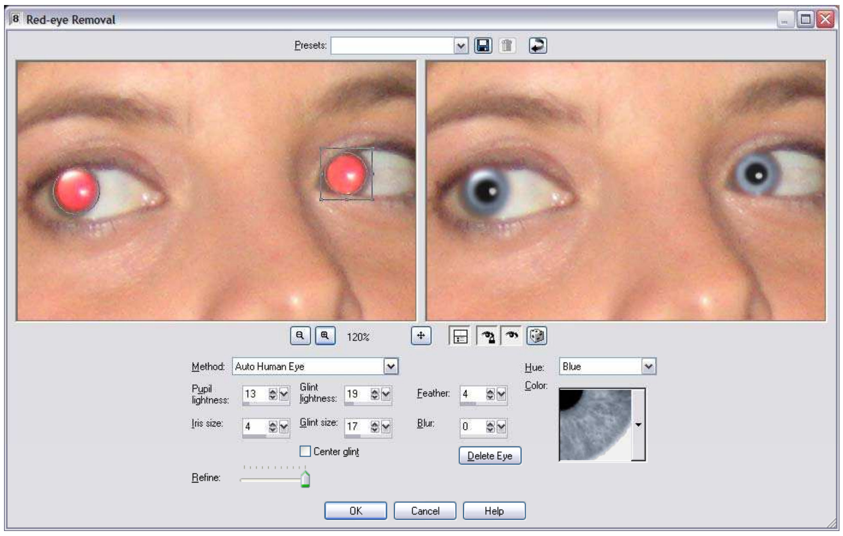


Figure 3.10 : When reflected light spreads over the iris, simple desaturation gives unnatural results.

complete redrawing of iris and pupil [99]. The results, however, can be be unrealistic (Fig. 3.11[1]).



Method: Auto Human Eye
Hue: Blue
Color:
Pupil lightness: 13
Glint lightness: 19
Feather: 4
Iris size: 4
Glint size: 17
Blur: 0
☐ Center glint
Refine:

Figure 3.11 : Correction of washed-out red eyes with an inpainting technique.

3.3.3 Flash/no-Flash

Another way of addressing the red eye problem is the “flash/no-Flash” technique [69], which aims to combine the advantages of taking a non-flashed picture and a flashed

[1]Corel Paint Shop Pro Red-eye Removal tool.

one. The main idea is to take a high-quality flashed picture and a low-quality non-flashed one, which is used to detect the red eyes and recover the natural colors of the affected zones (Fig. 3.12).

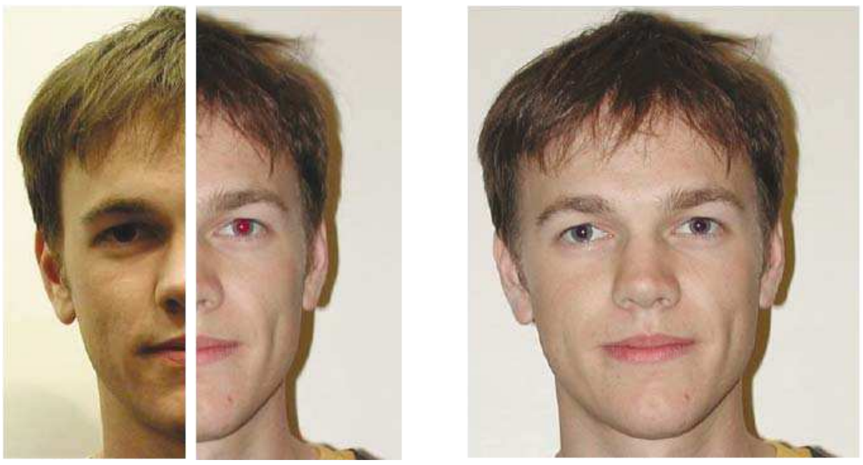


Figure 3.12 : Flash/no-Flash technique. (a) Dark non-flashed picture used to recover the correct color of the eyes; (b) high-quality picture affected by red eye artifacts; (c) corrected picture.

The first picture is shot without flash with high sensitivity, large lens aperture and with a relatively short exposure time. This yields a poor image, which is however suitable to help recover the unaltered colors of the eyes. The second picture is taken with flash immediately after and is the “real” picture to correct. Search for red eye artifacts is performed in the CIELAB space. The a^* channel is used as a measure of redness. For the pixels above a certain threshold, the difference between the a^* channel in the flashed and non-flashed images is checked against another threshold. Morphological operators are used to cluster them into blobs, discarding isolated pixels or very narrow regions. Red eyes are then corrected by considering the color in the non-flashed picture, applying a compensation for the different color cast of the two images.

The approach is simple and effective, but it presents lots of weaknesses. First of all, the memory and processing requirements double. Moreover, the images may suffer from registration problems, or they may simply be misaligned due to movements of the hand or of the subjects, which makes this method especially unsuitable for snapshot pictures. The illumination may be uneven, which may further confuse the color comparison between the two images.

3.4 Classification

With research on red eye removal algorithms continuously advancing, more and more detection techniques adapt a two-step method, consisting in a first candidate extraction phase and a subsequent classification phase [100,63,84,50,20]. The candidate extraction phase detects the possible red eyes in the picture, and is much more permissive than a stand-alone detection technique. This way, it detects a lot of false positives, but misses very few (if any) red eyes. The classification phase, then, is used to validate or reject each candidate, according to various features computed over the candidate patch.

3.5 Red Eye Detection through Bag-of-Keypoints Classification

In this section a methodology of red eye detection based on interest points is discussed [20]. It involves extraction of local image features, quantization of the feature space into a codebook through clustering, and extraction of codeword distribution histograms. A classifier is used to decide to which class each histogram, thus each image, belongs. Approaches of this kind have been shown to be able to recognize different kinds of images in a variety of applications [17, 25].

The main idea is to employ a classification technique to discriminate images representing red eyes from ones representing false candidates. The input dataset is analyzed considering a set of well-known keypoint detectors/descriptors, as discussed in Sec. 2. Support Vector Machine [26, 32] is used as final classifier. Such an approach is shape-based, thus robust to red image features which often cause false positives in color-based red eye detection algorithms, and is capable of detecting more complex features than most template-based methods. This, combined with a color-based red eye candidate extractor, and/or with a correction technique which leaves non-red zones untouched, may contribute to a full system robust to both color-based and shape-based false positives.

3.5.1 Algorithm Overview

The overall algorithm pipeline is depicted in Fig. 3.13. First, keypoints are extracted from images and descriptors are computed. In this application, the objects to describe are the various parts of the eye. Thus, it is fundamental to extract features distinctive of such parts, in order to well discriminate them from parts belonging to false candidates. To compute a fixed-length vector from each image, local features extracted (which are variable in number) are counted into a histogram. The bins are distributed across the feature space in a way such that more populated portions of the space have more bins, in order to obtain more meaningful histograms. The expression "Bag of Keypoints", or equivalently "Bag of Visual Words", is commonly used to name this type of representation, by way of analogy with a bag (the histogram) containing all the interest points (counted in the different bins).

Histograms are given as input to a classifier to separate the characteristics of the histograms of the two classes (eye and non-eye) and to discriminate them with high accuracy. A training set of labeled images is used to train the classifier and to find the optimal parameters for it.

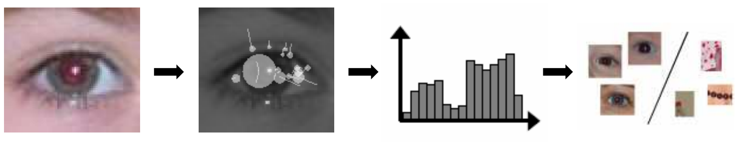


Figure 3.13 : The proposed algorithm pipeline. Left to right: Input image, feature extraction, feature quantization, classification.

3.5.2 Feature Extraction

The keypoint detectors used for the experiments are Harris-Laplace, Hessian-Laplace, Harris-Hessian-Laplace (combination of both, which is useful since they gather somewhat complementary information), Harris-Affine and Hessian-Affine. The descriptors used are SIFT and GLOH. SIFT was also tested with the DoG detector, as in the original SIFT image recognition pipeline [62].

3.5.3 Feature Space Quantization

Clustering is used to select a meaningful subdivision of the feature space into histogram bins. Descriptors from the training set are clustered with the k-means algorithm [45], with k=50. The set of centroids found is used as a “codebook” of representative features, and one bin is assigned to each, thus obtaining a finer quantization in more populated regions of the feature space. Each descriptor contributes to the bin relative to the closest centroid.

In some cases, no keypoints are detected in a given candidate image. Since it almost always happens for false candidates, e.g., for blurry background objects, images without keypoints are considered non-eyes and are discarded from further consideration.

Prior to classification, histograms are normalized in order to make them less dependent to the number of keypoints detected. Then, since the classifier used (see below) considers euclidean distance between vectors, i.e. the 2-norm of the difference, a trasformation is performed to make this distance more meaningful: histograms are transformed by taking the square root of each bin. This converts 1-norm normalized vectors into 2-norm normalized vectors. The dot product between two of these vectors, which is equivalent to the cosine of the angle between them, is the Bhattacharyya coefficient between the original (1-norm normalized) vectors [31]. This coefficient is a common measure of similiarity between frequency distributions, and it can be shown that the euclidean distance calculated this way is proportional to the metric commonly associated to the coefficient:

$$BC(\mathbf{v},\mathbf{w}) = \sum_{i=1}^{n} \sqrt{v_i} \cdot \sqrt{w_i}\,. \tag{3.11}$$

$$\begin{aligned} d(\overline{\mathbf{v}},\overline{\mathbf{w}}) &= \sqrt{\sum_{i=1}^{n} (\overline{v_i} - \overline{w_i})^2} = \sqrt{\sum_{i=1}^{n} \overline{v_i}^2 + \sum_{i=1}^{n} \overline{w_i}^2 - 2 \cdot \sum_{i=1}^{n} \overline{v_i} \cdot \overline{w_i}} = \\ &= \sqrt{\sum_{i=1}^{n} v_i + \sum_{i=1}^{n} w_i - 2 \cdot \sum_{i=1}^{n} \sqrt{v_i} \cdot \sqrt{w_i}} = \\ &= \sqrt{2 - 2 \cdot BC(\mathbf{v},\mathbf{w})} = \sqrt{2} \cdot \sqrt{1 - BC(\mathbf{v},\mathbf{w})}\,. \end{aligned} \tag{3.12}$$

In the above formulas, BC is the Bhattacharyya coefficient, d is the Euclidean distance, $\mathbf{v},\mathbf{w}$ are 1-norm normalized vectors and $\overline{\mathbf{v}},\overline{\mathbf{w}}$ are the corresponding 2-norm normalized vectors.

3.5.4 SVM Classification

Histograms are classified using a Support Vector Machine classifier [26,32]. This classifier works by finding an optimal separation hyperplane between two different classes of labeled training vectors and specifying a small number of weighted vectors which lie on the boundaries of the classes (these vectors are called “support vectors”). Since a linear separation usually is not meaningful, vectors are usually projected into a higher-dimensional space and then linearly classified in that space. However, computing the projected vectors explicitly can be expensive or even impossible. Instead, the problem can be expressed in a form where the projected vectors only appear in dot products between two of them. Thus, a common practice is to employ a function which, given two vectors in the original space, computes the dot product of their projections in the higher-dimensional space (usually in a much simpler way). This function is named “kernel”, and this practice is named “kernel trick”. It is proven that any continuous, symmetric, semidefinite positive function from $\mathbb{R}^n$ to $\mathbb{R}$ can be used as a classification kernel.

The kernel used in the experiments is the Radial Basis Function (RBF) kernel, which is a multidimensional non-normalized gaussian:

$$K(\mathbf{v},\mathbf{w}) = e^{-\gamma\cdot\|\mathbf{v}-\mathbf{w}\|_2^2}\,,\ \gamma > 0\,. \tag{3.13}$$

As shown above, the aperture of the gaussian function is controlled by a parameter γ. This is one of the parameters which must be adjusted in order to obtain the most accurate classification for each given training set. The other parameters to find, called C_1 and C_2, are penalty factors for outliers in the two classes of the training set. It is important to carefully adjust them in order to find an optimal tradeoff between accuracy of the classification and tolerance to outliers, which is important to prevent overfitting. The two parameters are adjusted independently to achieve more generality.

Optimal parameters are searched with a multi-level grid search using 8-fold cross-validation for training and testing: first, a grid of parameter triples, spaced evenly in a logarithmic scale, is tried, then a finer grid covering the parameters who gave the best results is tried, and so on, up to the fifth level of grid refinement.

3.5.5 Experimental Results

The proposed red eye detection system has been trained on a data set of 4079 image patches, including 956 red eyes, and tested on a data set of 5797 image patches, including 234 red eyes. The sets has been collected from photographs taken with various sources, including DSLR images, compact cameras and Internet photos, and the image candidates have been extracted using an automatic red cluster detector [19]. This means that the "eye" class is mostly trained with red eye patches: however, since the method is shape-based, regular eyes are recognized as well, as red eye artifacts have little impact on the luminance. Training on red eyes allows to learn the slight differences in shape which occur (e.g. biggest and brightest pupils, different luminance distribution), while classifying regular eyes along with red eyes is not a problem, since regular eyes have no red pupils to correct, then a color-based correction algorithm can discard them easily, and a "blind" correction algorithm can desaturate them with almost no harm.

Table 3.1 shows results achieved with the different detector+descriptor combinations. Hit rate is the percentage of eyes correctly classified out of the total number of eyes; False positives is the percentage of false candidates incorrectly classified out of the total number of false candidates; Overall performance is (Hit rate + (100% - False positives))/2; Accuracy is the percentage of patches correctly classified out of the total number of patches.

Table 3.1 : Classification results for each detector/descriptor combination tested.

Detector + Descriptor	**Hit rate**	**False positives**	**Overall perf.**	**Accuracy**
DoG + SIFT	83.3%	8.9%	87.20%	90.78%
HarLap + SIFT	78.2%	12.2%	83.00%	87.47%
HesLap + SIFT	71.8%	7.1%	82.35%	92.12%
HarHesLap + SIFT	82.9%	3.5%	89.70%	95.94%
HarAff + SIFT	70.1%	8.1%	81.00%	91.08%
HesAff + SIFT	82.9%	7.0%	87.95%	92.59%
HarLap + GLOH	75.2%	12.1%	81.55%	87.38%
HesLap + GLOH	80.3%	14.4%	82.95%	85.41%
HarHesLap + GLOH	76.9%	7.3%	84.80%	92.07%
HarAff + GLOH	73.5%	11.9%	80.80%	87.52%
HesAff + GLOH	69.2%	7.3%	84.80%	92.07%

It can be seen from the results that, while there are quite a significant amount of eyes that are missed, a small percentage of false candidates are misclassified as eyes (using the best performing detector+descriptor combinations). This is important, since correction of false positives is one of the biggest problems in red eye removal, thus it is very important to keep false positives as low as possible. It is also evident how the blob-based detectors yielded better results than the corner-based detectors. This is not surprising, as many parts of the eye are more suitable to be extracted by a blob-like detector (expecially the iris and the pupil). Using both types of detector together, however, helps keep the false positives lower. SIFT performed better than GLOH. Examples of correct and wrong classifications are shown in Fig. 3.14.

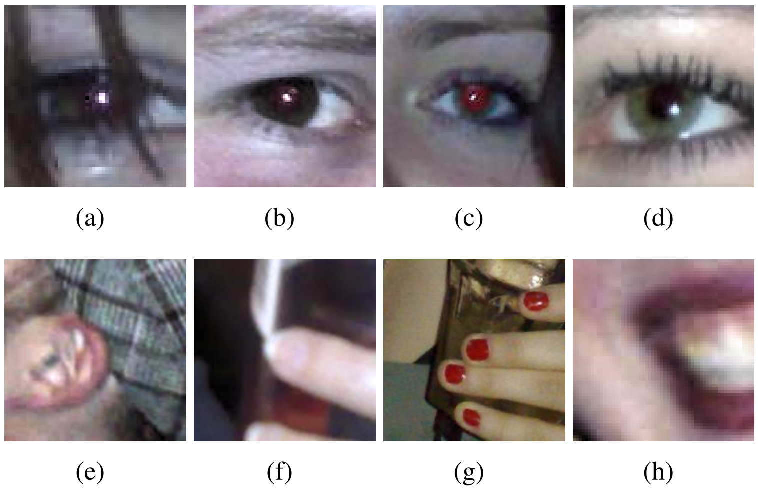


Figure 3.14 : Classification examples obtained with Harris-Affine + SIFT. Top row: *(a)*, *(b)*, *(c)* are eye patches correctly classified; *(d)* has been incorrectly classified as non-eye. Bottom row: *(e)*, *(f)*, *(g)* have been correctly classified as non-eyes; *(h)* is a false positive.

4. Robust Image Hashing for Near-Duplicate Image Detection

4.1 Introduction

The increasing use of low cost imaging devices and the availability of large databases of digital photos and movies makes the retrieval of digital media a frequent activity for a number of applications. In particular, image retrieval from large databases, such as popular social networks, collections of surveillance images and videos, or digital investigation archives, is a fundamental issue for forensic tasks. Recent estimates about popular websites [36] report about 4 billion photographs on Flickr [8], with about 4000 uploads per minute. In YouTube [14] approximately 20 hours of video are uploaded per minute, for a total of about 120 million videos. Facebook [6] proceeds at a rate of about 22,000 uploads per minute for an estimated 15 billion images. Moreover, among the other archives used in image forensic, a huge amount of data comes from video surveillance systems.

During digital investigation (e.g., for copyright violation, child abuse, etc.), hashing techniques are commonly used to index large quantities of images to detect copies from different archives. However, classic hashing methods (e.g., MD5 [83]) are unsuitable to find altered copies, even in case of slight modifications (near duplicates). Classic hashing techniques mainly fail because just a small change in the image (even a single bit) will, with overwhelming probability, result in a completely different hash code. For example, two copies of the same image depicting a scene of crime are perceptually identical under small viewpoint changes, partial occlusion, and/or low photometric distortions, but their hash code is completely different when a classic hashing approach is used to check their similarity. This fact could implicate the missing of important evidences for related investigations. In order to cope with this problem, robust hashing techniques must be developed: perceptually identical images should have the same hash value with high probability, while perceptually different images should have independent hash values.

Recently, some commercial approaches of robust hashing have been proposed for photos (PhotoDNA [10]) and videos (Videntifier [13]). These techniques make use of the recent developments in the field of Near Duplicate Image (NDI) retrieval. The definition of near duplicate depends on the degree of variability (photometric and geometric) that is considered acceptable for each particular application. Some approaches [56] consider as NDI images obtained by slightly modifying the original ones through common transformations such as changing contrast or saturation, scaling, cropping, etc. Other techniques [48] consider as NDI images of the same scene but with different viewpoint and illumination.

In the last few years, different image hashing techniques have been proposed in literature to cope with image retrieval and near-duplicate images detection problems. Most of these techniques are based on the Bags of Visual Words paradigm [90] to build a holistic representation of the images. Ke et al. [56] detected near-duplicate images by employing local descriptors to represent and match images under several transformations. They used a hash-based indexing technique to efficiently search the descriptor database, then applied an optimized storage layout to further improve efficiency. Chum et al. [30] proposed two novel image similarity measures for image indexing using local feature descriptors and enhanced min-Hash techniques. The authors of [29] introduced a method to combine visual words with geometric information to improve hashing-based image retrieval and object detection, obtaining a novel algorithm called

Geometric min-Hash which shows significant advantages against geometrical deformations and occlusions.

Recently, the Bag of Visual Words paradigm has been augmented by using multiple descriptors ("Bags of Visual Phrases") to exploit the coherence between different feature spaces in which local image regions are described. Specifically, to reduce the amount of false matches in the Bag of Visual Words model, the authors of [48] introduced the coherent phrase model. In this model, a local image region is described by a visual phrase of multiple descriptors instead of a visual word of a single descriptor. In the Bags of Visual Phrases approach, both feature coherence (local regions are described by descriptors of different types) and spatial coherence (multiple descriptors are obtained from local areas at different sizes) are exploited.

4.2 Bag of Visual Phrases with Codebook Alignment

A further improvement on the Bags of Visual Phrases approach will now be discussed, which exploits the coherence between feature spaces not only in the image representation, but also in the codebooks generation [18]. This is obtained by aligning the codebooks of different descriptors to produce a more significant quantization of the involved descriptors spaces, which leads to a more distinctive representation. In particular, instead of separately obtaining each codebook as in [48], to further enforce feature correspondence we generate the final codebook taking into account the correspondence of the clusters of the involved descriptors spaces.

Taking into account an image I, M local regions are extracted by making use of some detectors [70] or dense sampling [58]. Each region i is hence described by H

descriptors ϕ_{ih} [71]:

$$\phi_i = \{\phi_{i1}, \phi_{i2}, ..., \phi_{iH}\} \tag{4.1}$$

Considering a vocabulary V_h for each descriptor type, each ϕ_{ih} can be associated with a visual word in V_h and hence ϕ_i can be associated with a H-tuple of visual words, called “visual phrase”:

$$v = \{v_h | h \in [1, 2, ..., H]\} \tag{4.2}$$

where v_h is a visual word belonging to the codebook V_h. Each image is then described by the frequency distribution of visual phrases. The coherent phrase model tries to incorporate the coherence across multiple descriptors properly chosen in order to describe different aspects of the appearance of a local region within an image.

The proposed approach augments the coherent phrase model improving the vocabularies generation step. In [48], H codebooks (one per descriptor type) are created separately by using a classical clustering approach on each descriptor space. Then the images are described with a normalized multidimensional histogram in which each bin is related to a visual phrase. The rationale underlying the proposed apporach is that, although different descriptors encode different properties of a local region, they represent the same image area, hence the clustering, and the visual words, are in “some way” related. So, the coherence of different kind of descriptors should be exploited also in the vocabulary generation step. The main schema of the proposed approach is shown in Fig. 4.1 where, for sake of simplicity, only two descriptors (SIFT [61], SPIN [52]) are shown.

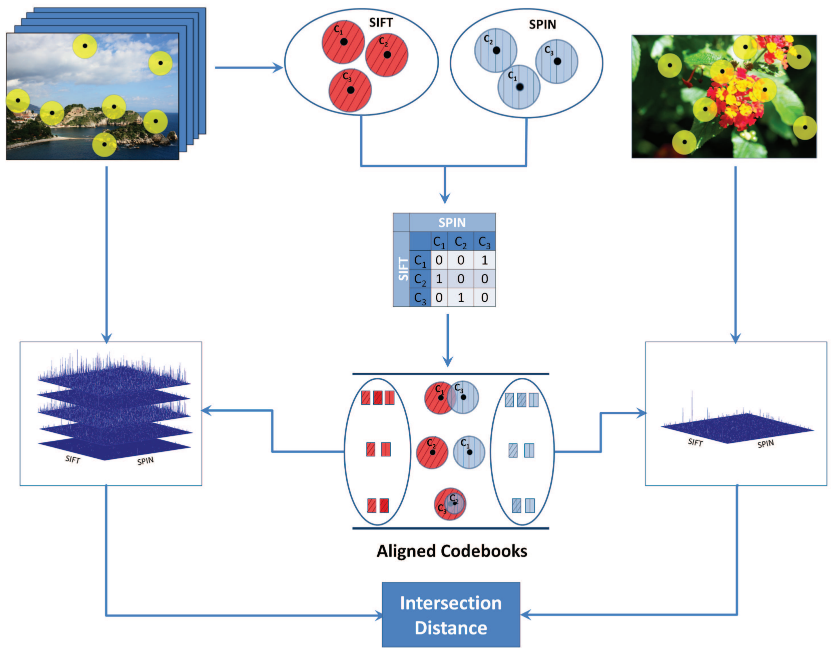


Figure 4.1 : Bags of Visual Phrases with Codebooks Alignment.

First, the H different descriptors spaces are clustered separately and K visual words (cluster centroids) are obtained for each vocabulary V_h (one visual vocabulary per descriptor). The orders of cluster labels in all of the clustering are then rearranged according to the first one. A $K \times K$ similarity matrix between pairs of partitions in two clustering algorithms is obtained counting the number of elements (local image areas) they share. The Hungarian algorithm [80] is then used to find the best assignment for the cluster correspondence problem from the previous obtained similarity matrix. The correspondences are then used to create H novel vocabularies where visual words are generated considering the centroids relative to both common and uncommon elements between aligned clusters. Hence three new visual words (cluster means) per descriptor space are generated from two aligned clusters considering the operations of intersection and difference. Notice that, although Hungarian algorithm aligns all the clusters, some of them can have no common elements (Fig. 4.1). In that case the only two obtained visual words are equal to the original one.

4.3 Experimental Results

In this section the effectiveness of the proposed approach is demonstrated through a number of experiments on a dataset of NDI captured by hand held digital cameras. The proposed approach is compared with respect to the coherent phrase model proposed in [48].

Our system has been trained using a set of 116 images, acquired with different cameras, in different conditions (e.g., illumination, distance from the subjects, etc.), with high scene variability (indoor, outdoor, natural, artificial, etc.), as shown in Fig. 4.2.

For testing purposes we used two different NDI test sets: a synthetic one and a real one. The synthetic NDI set consists of 5684 images generated from the training set by using transformations typically available on image manipulation software (Fig. 4.3). Specifically, accordingly with [56], the following transformations have been used: Colorizing, contrast changing, cropping, despeckling, downsampling, format changing, framing, rotating, scaling, saturation changing, intensity changing, shearing. Considering the different parameter settings, for each training image 49 near duplicate copies have been obtained. The transformations have been performed with ImageMagick [9].

The real NDI test set consists of 182 real images which depict the same subjects as the training set images (Fig. 4.4). These copies vary, with respect to the training set, in viewpoint, occlusion, scale, illumination conditions, etc.

Training images have been used for the codebooks generation. First, local interest points have been detected in these images using the Hessian-Laplace detector [71]. Afterward, two different types of descriptors have been extracted on each interest point: SIFT [62] and SPIN [52]. Since these descriptors are extracted considering different

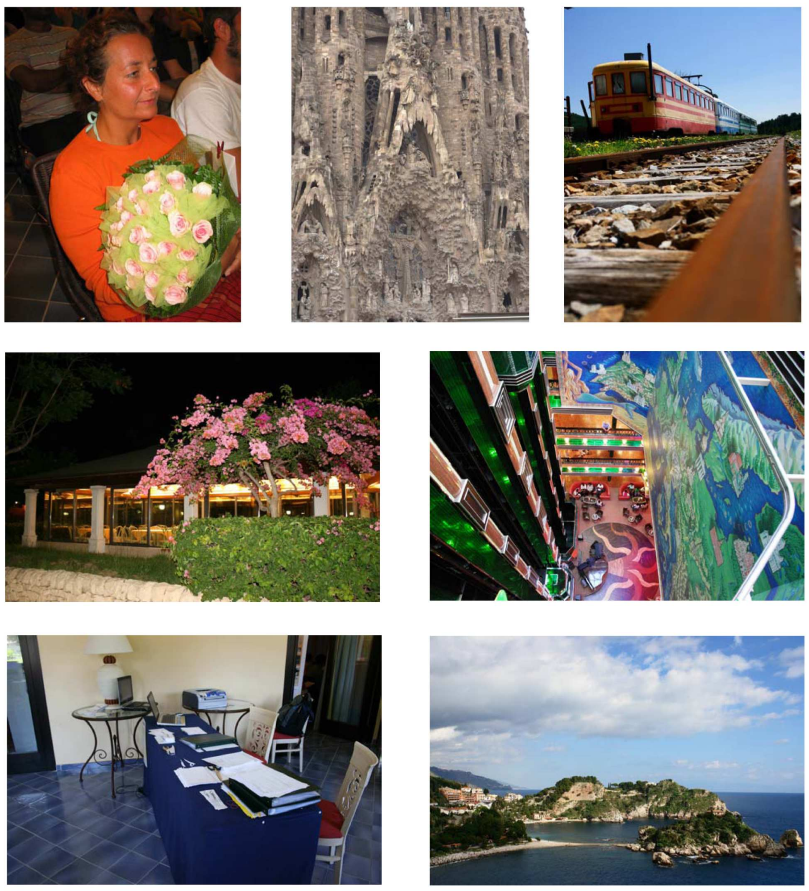


Figure 4.2 : Example images from the training set.

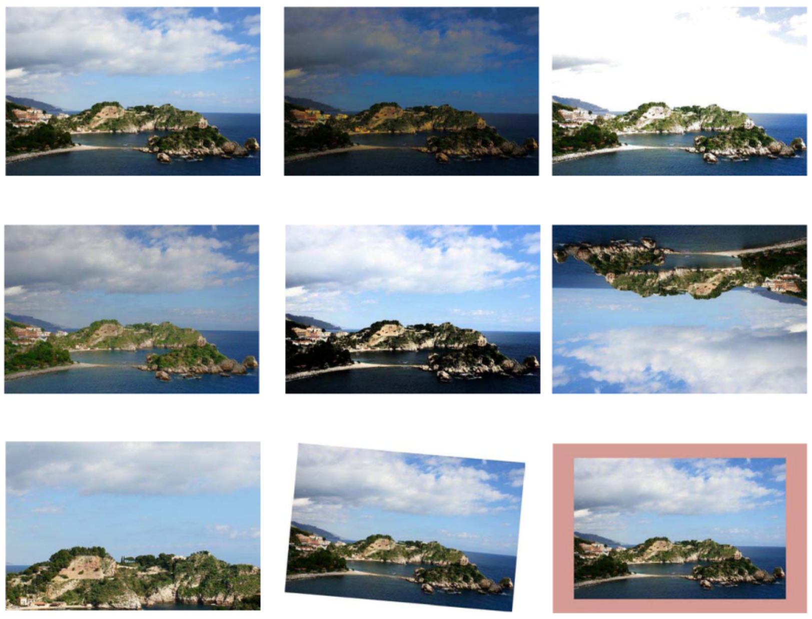


Figure 4.3 : Examples of synthetic near duplicate test images.

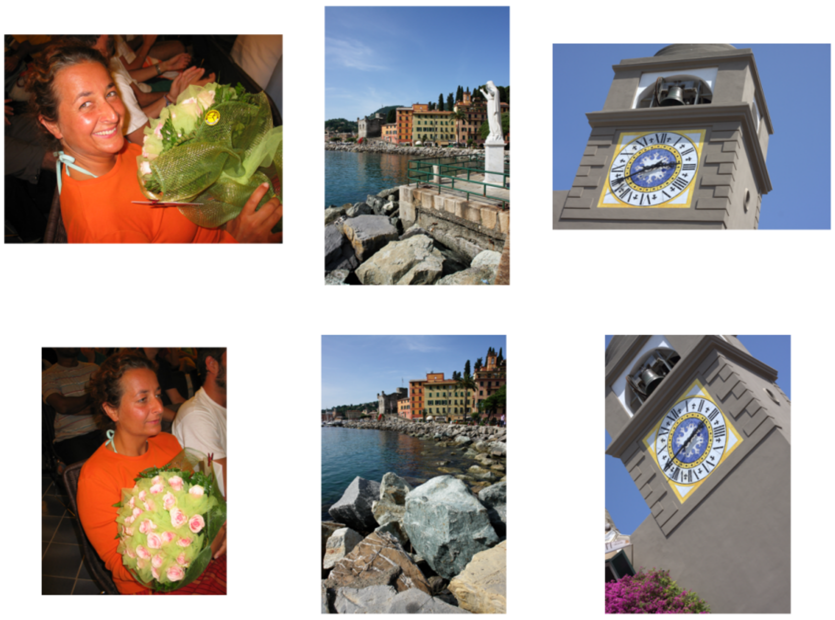


Figure 4.4 : Examples of real near duplicate test images.

image properties (gradient orientation (SIFT) and intensity distribution at different distance from the center (SPIN)), they are somewhat complementary, hence can be fruitfully combined. K-means algorithm (K=500 in our tests) has been then used to produce the two independent codebooks corresponding to the two involved types of descriptors. The two obtained partitions have been aligned to generate the new codebooks. Finally, training images have been represented by visual phrases (a 2D histogram) obtained considering the new codebooks.

Test images are used to perform queries on the training image dataset. Each test image is represented by a visual phrase histogram obtained considering the aligned codebooks. This representation is then used to find the corresponding image in the training dataset, by means of a similarity function between bag of phrases histograms. To cope with partial matching, we use the intersection distance τ defined as follows [42]:

$$\tau(H_I, H_J) = \sum_{v=1}^{V} min(H_v(I), H_v(J)) \tag{4.3}$$

where H_I, H_J are two visual phrase histograms and $H_v(.)$ is the vth bin of the histogram.

Each query is associated to a list of training images. The retrieval performance is evaluated with the probability of the successful retrieval $P(n)$ in a number of test queries:

$$P(n) = \frac{Q_n}{Q} \tag{4.4}$$

where Q_n is the number of successful queries according to top-n criterion, i.e. the correct NDI is among the first n retrieved images, and Q is the total number of queries.

The proposed approach has been compared with the original Bags of Phrases approach [48]. Results obtained are reported in Fig. 4.5 and Fig. 4.6. In both tests, the proposed approach outperforms the original Bags of Visual Phrases. The precision/recall values at top-n=1 are shown in Table 4.1 and Table 4.2. Note that, since there is only one correct match for each query, the precision and recall for top-n=1 are equal. Finally, Fig. 4.7 shows an example of the first retrieved image on a specific query.

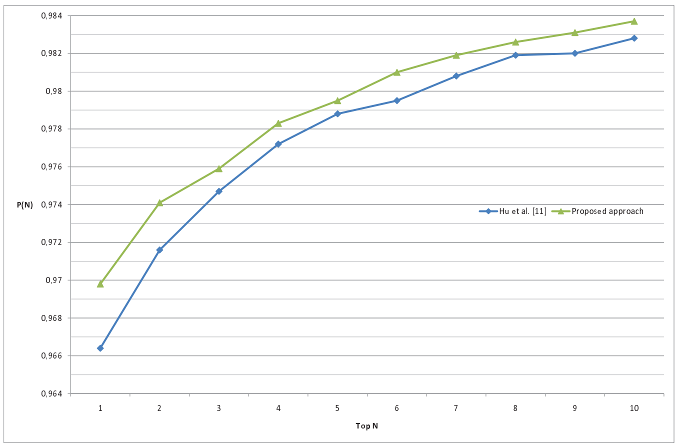


Figure 4.5 : Top-n NDI retrieval performance comparison on the synthetic NDI dataset.

Table 4.1 : Precision/Recall values on the synthetic NDI dataset. The best result is shown in bold.

	Precision/Recall
Hu et al. [48]	0,9664
Proposed Approach	**0,9698**

Table 4.2 : Precision/Recall values on the real NDI dataset. The best result is shown in bold.

	Precision/Recall
Hu et al. [48]	0,7033
Proposed Approach	**0,7308**

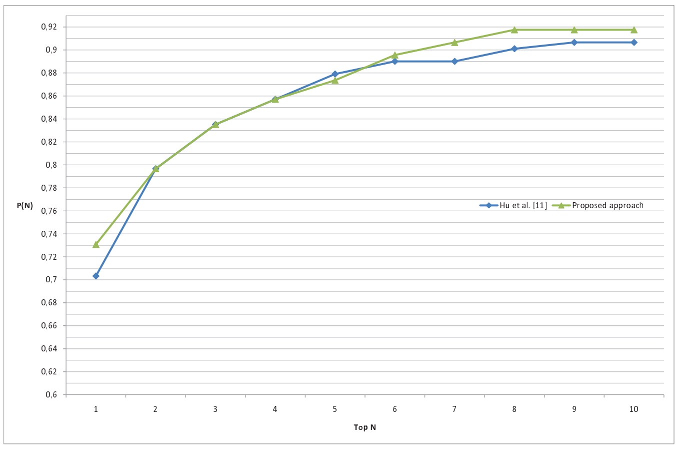


Figure 4.6 : Top-*n* NDI retrieval performance comparison on the real NDI dataset.

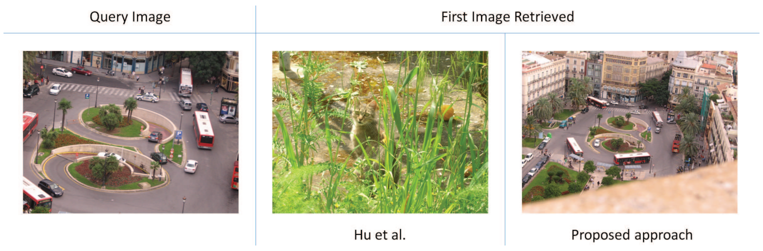


Figure 4.7 : A visual example of the first retrieved image on a specific query.

Part II

Image Classification with Randomized Learning

5. Image Classification with Randomized Learning

5.1 Introduction

Image classification, the problem of assigning a discrete label to an image from a set of possible labels [34], is a complex problem which usually employs sophisticated Machine Learning methodologies, being treated as a supervised learning problem [46]. Such problems are addressed by systems which need to be trained with a set of previously labeled image examples (called training set). Image categorization engines must be able to generalize well on the intra-class variation (differences between elements on the same class), like different lighting condition, viewpoint, differences in the subjects themselves, etc.

Most of the systems for image categorization use sets of binary classifiers, one for each image category C_i. In each of the two-class categorization problems, given an image I, the problem is to understand if it contains a particular visual class C_i or not, taking into account the representation of I into some feature space $f = F(I)$. From a statistical point of view, the task of categorization becomes a comparison between two probability scores:

- $P(C_i|f)$: the probability of having C_i given the feature vector f;
- $P(Other|f)$: the probability of not having C_i given the feature vector f;

The ratio between the two probabilities may be expressed, by using the Bayes theorem [24], as follows:

$$\frac{P(C_i|f)}{P(Other|f)} = \frac{\frac{P(f|C_i)P(C_i)}{P(f)}}{\frac{P(f|Other)P(Other)}{P(f)}} = \frac{P(f|C_i)P(C_i)}{P(f|Other)P(Other)} \tag{5.1}$$

There are two different classes of approaches to the above equation [33,76,51,98]:

- Discriminative methods directly estimate $P(C_i|f)/P(Other|f)$, which is called posterior probability ratio, by finding boundaries of the classes in the feature space. The ratio acts as a function which directly discriminates the target class for each image.

- Generative methods learn a model of $P(f|C_i)P(C_i)$, the joint probability distribution, and make predictions by using Bayes' rule to calculate the posterior probability $P(C_i|f)$. Training data are used to learn a model for the likelihood $P(f|C_i)$ and the prior distribution $P(C_i)$. The likelihood can be seen as describing how to generate random instances of an image conditioned on the target class.

5.2 Randomized Learning

Given a problem to learn and a set of possible functions which can be used to model the problem, an exhaustive optimal search on the entire space of possible function may not always be feasible: in some cases it may be simply not computable, in others it may be very expensive [49]. Moreover, choosing the absolute best function for the training data, even when possible, may "overfit" the training data, with some loss of generality with respect to the problem itself.

Randomized learning is a decisional process based on the random extraction of decisional functions [16]. Given a labeled training set of input data and a set of possible functions defined in the domain of the input type, it performs a sub-optimal search in the function space by randomly extracting a given number of possible functions and selecting among them the most discriminative one, i.e., the one which best divides the training data. To measure the "goodness" of the subdivision, the Shannon entropy is used to measure the set impurity [86]:

$$E\left(X\right) = -\sum_{i=1}^{n} p\left(x_i\right) \log p\left(x_i\right), \tag{5.2}$$

where $p\left(x_i\right)$ is the number of elements of the set X which belongs to the class i divided by the total number of elements in X, and n is the total number of classes. Given this measure, the information gain which is achieved by subdividing the training data in two subsets (by the extracted decisional function) is measured as the drop in entropy which results from the subdivision:

$$\Delta I = -\Delta E = E\left(X\right) - \frac{|X_l|}{|X|} E\left(X_l\right) - \frac{|X_r|}{|X|} E\left(X_r\right), \tag{5.3}$$

where X_l and X_r are the two resulting subsets. Thus, it is possible to choose, among the randomly extracted function, the one which maximizes the information gain. An example is shown in Fig. 5.1.

This type of decisional process is quite fast and straightforward, but of course may not perfectly divide the training data, even when it is separable. However, as seen above, overfitting the training data may cause loss of generality, so allowing for a small misclassification margin is actually a good characteristic.

A single function is almost always not enough to classificate the data: there may be more than two classes, and in general it is preferable to combine more simpler functions than to consider a pool of overly complex single functions. So, a strategy to

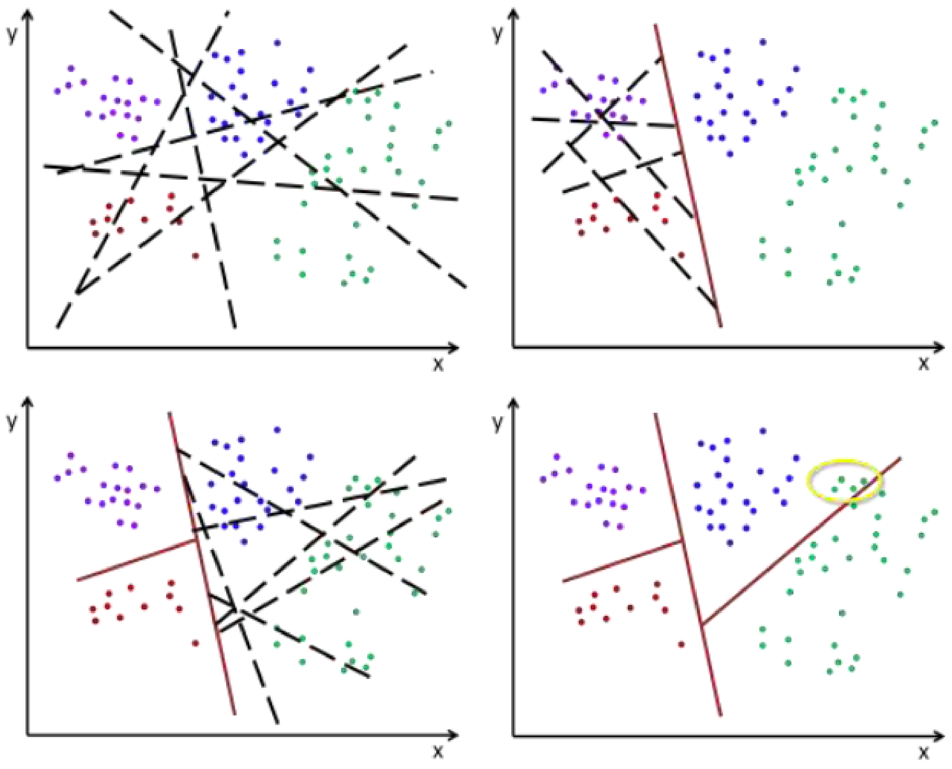


Figure 5.1 : Toy example of randomized learning. The decision functions are straight lines in the plane which divide it in two half-planes. Several lines are tried, and the best one is chosen. Each resulting half-plane is subject to a subsequent subdivision. A small misclassification is visible in the last subdivision.

combine more random decisional functions into a greater decisional structure is needed to obtain a powerful random-based classification tool.

5.2.1 Random Decision Forests

Decision trees are well-known classification tools which consist in decisional structures which arrange tests in a tree-like hierarchy [54, 27]. They are quite simple to run, but are prone to overfitting training data; moreover, the search for optimal training may be expensive [49]. These characteristic make them well suited for the adaption of randomized learning.

Random decision trees consist in decision trees whose functions at split nodes are determined (at training time) using a randomized learning methodology [41]. For each subtree, learning is iterated recursively considering the corresponding training subset. Tree growth is stopped at a certain depth, and/or when the information gain falls below a certain threshold. Leaf nodes are populated with histograms representing the class frequency distribution for the training data which reach each node. To account for possible imbalances, each class is weighted with the inverse of the class probability distribution in the training set. The histogram at each leaf node is used, at test time, to estimate the probability for the unlabeled input to belong to each class. Since the information gain is higher for functions which better separate the data, histograms typically tend to have a high peak for one class. The class with maximum probability is then chosen as the classification output for the tree (Fig. 5.2).

Random decision trees may suffer from a stability problem: if the decision functions at the highest nodes succeed to capture some characteristics but fail at representing others, classification in lower nodes may not manage to recover. To overcome this problem, a common methodology is the creation of a set of random decision trees,

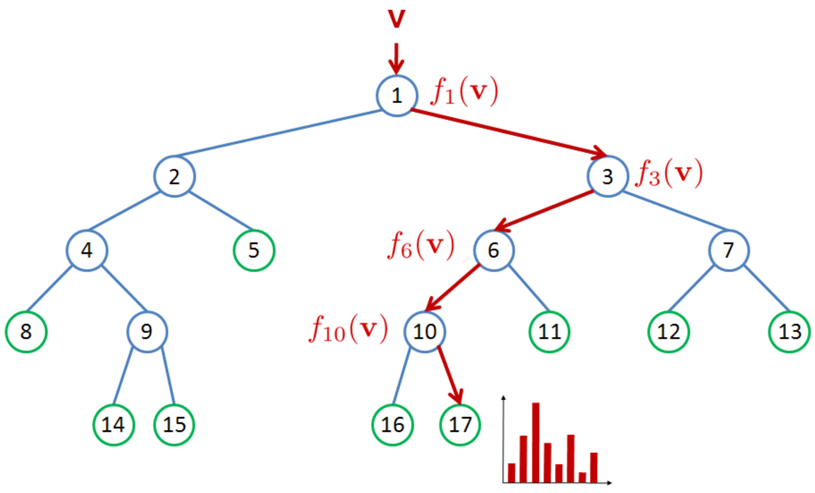


Figure 5.2 : Classification with a random decision tree. Split functions and leaf histograms are produced at training time.

called random decision forest. By separately considering more trees and averaging their classification output, stability problems can be easily contained. To reduce the memory and time requirements of training, and to achieve a further amount of generality, for each tree a training subset is randomly sampled from the entire training set (with possible overlap with other subsets).

Random decision forests have been fruitfully employed for many applications, among which are shape recognition [16], codebooks generation [75, 74] and keypoint recognition [59].

5.2.2 Random Ferns

Random ferns are decisional structures, invented as an alternative to random decision forests for keypoint recognition [79, 78], which make use of a considerable number of randomly extracted decisional functions but does not arrange them in a hierarchical structure.

Considering simultaneously a (possibly large) number of decisional functions poses a remarkable problem in computing the probability histogram for each function combination. Writing this problem as a conditional probability and applying the Bayes' rule:

$$P\left(C_i|f_1,f_2,\ldots,f_N\right) = \frac{P\left(f_1,f_2,\ldots,f_N|C_i\right)P\left(C_i\right)}{P\left(f_1,f_2,\ldots,f_N\right)}. \tag{5.4}$$

Assuming a uniform prior and factoring out the denominator (which is independent from the class):

$$P\left(C_i|f_1,f_2,\ldots,f_N\right) \propto P\left(f_1,f_2,\ldots,f_N|C_i\right). \tag{5.5}$$

As stated above, estimating the joint probability for all the functions is usually not feasible, since it would require computing 2^N probabilities to account for every function combination. Assuming complete independence between the functions makes the problem much simpler:

$$P\left(f_1,f_2,\ldots,f_N|C_i\right) \approx \prod_{j=1}^{N} P\left(f_j|C_i\right). \tag{5.6}$$

However, this ignores any correlation between functions, which is usually an unacceptable approximation. A compromise between the two positions is to consider some groups of functions, estimate the joint probability for each group, and assume independence between them:

$$P\left(f_1,f_2,\ldots,f_N|C_i\right) \approx \prod_{k=1}^{M} P\left(F_k|C_i\right). \tag{5.7}$$

In analogy to the random decision trees, these function groups are called “random ferns”. This approach is considered as “semi-naive Bayesian”, since it is a compromise between a proper Bayesian and a naive Bayesian approach. In training, probabilities for all function combinations for all ferns are estimated and stored based on the training data, making it very easy to compute, at test time, the above probabilities from the stored values (“ten lines of code”).

5.3 Image Classification

An image classification paradigm which employs randomized learning is based on the classification of image subwindows [66]. It consists in randomly extracting a number of subwindows from the image and independently classifying them. The subwindows are extracted by randomly picking their width, height and position among all the possible parameters for the image. After classification of individual subwindows, a voting procedure determines the classification of the entire image. At training time, subwindows are extracted from the training images and labeled with the classes of the images before feeding them to the training phase of the classifier.

The aforementioned methodology is independent of the classifier and the type of decision functions used. In the experiments performed, the subwindows were rescaled to 16×16 pixels, converted to the HSL color space, then transformed using the Haar-Wavelet transform. The decision functions are simple comparisons between pixels of the transformed subwindows [74]. Different possibilities for the classifier were experimented, as detailed in Sec. 5.3.1, 5.3.2, 5.3.3. Details about the dataset used for the experiments are provided in Chapter 6.

5.3.1 Semantic Texton Forests

As thoroughly explained in [87], simple decision functions on pixel pairs from local neighborhood can give some basic information about image structures which can be used for further classification. Random forests based on such functions are called semantic texton forests (STF for short). Results achieved from subwindow classification through STF are detailed in Table 5.1.

Table 5.1 : Classification results obtained with STF.

	Indoor	Outdoor
Indoor	67.39%	32.61%
Outdoor	21.82%	78.18%

	Natural	Artificial
Natural	31.30%	68.70%
Artificial	12.69%	87.31%

5.3.2 Random Ferns

Along with STF, experiments using ferns for image classification were performed. Results are detailed in Table 5.2.

Table 5.2 : Classification results obtained with ferns.

	Indoor	Outdoor
Indoor	52.17%	47.83%
Outdoor	12.70%	87.30%

	Natural	Artificial
Natural	86.26%	13.74%
Artificial	52.24%	47.76%

5.3.3 STFerns

Along with STF and ferns, a novel classification procedure was experimented, which aims to combine the strenghts of the two above classifiers in an unique way. It consists in a multi-branched decision tree, whose decision function in each split node is an entire fern. This novel classifier has been named “STFern”, by the combination of “STF” and “Fern”. By doing a full fern-based classification at each tree level, a greater level of refinement can be achieved. Split nodes have as many subtrees as the number of classes in the classification problem, and the subtree to visit is chosen based on the classification made by the fern. This has a good potential for multi-modal problems, because different aspects of the problem can be modeled at the different levels of the tree. Altough more trees can be combined in one forest, fern classification has proven stable enough not to require the added complexity. Results achieved are detailed in Table 5.3.

Table 5.3 : Classification results obtained with ferns.

	Indoor	Outdoor
Indoor	47.83%	52.17%
Outdoor	7.82%	92.18%

	Natural	Artificial
Natural	88.55%	11.45%
Artificial	45.90%	54.10%

6. Dataset Construction

Building a dataset, while in appearance is only a minor task, may prove a challenging problem. It must be representative of all the characteristics, more or less typical, that elements of a certain class may have (intra-class variance), and must also express well the differences which hold between elements of different classes (inter-class variance). Similar considerations apply not only for the training set, but also for the test set, which is used as a benchmark for the obtained classifiers.

First of all, it is necessary to define the classes which it is necessary to distinguish. There are plenty of types of classifications which may be useful for image understanding, among which are recognizing the environment, estimating the distance of subjects, or detecting the presence of people. Understanding the scene is important to tune the imaging device and/or to perform subsequent post-processing enhancements to images, or even to index them.

In designing the dataset used for the experiments, a number of possible useful classifications were considered:

- Indoor vs. Outdoor
- Day vs. Night
- Portrait vs. Close-up (non-person) vs. Landscape
- Natural vs. Artificial

Instead of considering separate classifiers and/or datasets for the above classifications, a comprehensive database was crafted, with the idea of considering all possible combinations of the above categories (in a certain sense, the Cartesian product of the datasets). After pruning out nonsensical combinations, the following categories were chosen (each letter represents one of the above categories):

- IDC
- IDP
- ODCA
- ODCN
- ODLA
- ODLN
- ONCA
- ONLA

Given a comprehensive classifier able to distinguish all the above cases, it is then easy to perform more granular classifications by simply joining the interested classes. While having more classes leaves more possibility to misclassification, this procedure allows to extract a great deal of information all in one step, and could prove useful for imaging applications. Two of the possible classifications (Indoor vs. Outdoor and Natural vs. Artificial) were chosen as test examples for Chapter 5.

The pioneering dataset used in the experiments consists in 394 training images and 399 test images, roughly evenly distributed among classes. They were taken from different sources; most of them were taken from Flickr [8].

Acknowledgements

So, that's it. My course (or “my adventure”, depending upon the point of view) has come to an end. Like reaching the top of a mountain (a foothill?) one once wouldn't believe he would reach.

Yes, I know, acknowledgements are supposed to be written here. And they will.

First, I would like to thank my advisor, Prof. Sebastiano Battiato, who encouraged me to embark on this adventure, put much trust on me, and assisted me in a lot of troubled moments.

A big appreciation goes to my Ph. D. colleagues at STMicroelectronics, Rosetta Rizzo and Giuseppe Messina, for the mutual support we gave each other, the (abundant) funny moments spent together, the coffee breaks, and the liters of coffee I presumably owe them. I have a lots of other reasons to say “Thank you” to them, but, as someone once said, this page is too small to contain them all.

I'd like to take this opportunity also to say “Thank you” to my external advisor Mirko Guarnera and the entire AST Imaging, Catania Lab at STMicroelectronics, whose friendly environment and professional work were very inspiring to me, and where I spent three very instructive years.

A special mention is also deserved by the IPLab Group, whose professors, researchers, and Ph. D. colleagues were always friendly and helpful. In particular, other than Prof. Battiato, I'd like to thank Giovanni Maria Farinella and Giovanni Puglisi, which supported me in a lot of tasks, and Prof. Giovanni Gallo, who acted almost as a "spiritual advisor" when I needed it, and was always ready to spare a cheerful word when possible.

Thanks to my family, for bringing me up to here and always supporting and trusting me.

Uncountable thanks to my precious girlfriend Luisa, whose patience and support (other than caring and sweetness) have helped me through LOTS (I mean it!) of hardships.

To my friends, for the many (I mean it, too!) funny days and nights spent together during these three years.

To Mr. Antani, for being so nice and helpful also during my Ph. D. years, like he was before.

To Amilcare Laudani, whose sport performances have always been of great inspiration to me.

To the Italian community of Hattrick and my federation "La Compagnia del Cazzeggio Perpetuo", for the emotions we had together.

I know that in years to come, I will reread these pages and think of a lots of people I have missed to mention. I know. I'm sorry. I'm trying my very best to remember you all right now.

To anyone, for being here reading this.

Again, like three years ago, to anyone who trusted me.

Thank you.

Bibliography

[1] **ACDSee**. www.acdsee.com.

[2] **Adobe Photoshop**. www.adobe.com/products/photoshop.

[3] **Affine Covariant Features**. www.robots.ox.ac.uk/˜vgg/research/affine.

[4] **Corel Paint Shop Pro**. www.jasc.com.

[5] **Demo Software: SIFT Keypoint Detector**. www.cs.ubc.ca/˜lowe/keypoints.

[6] **Facebook**. www.facebook.com.

[7] **FAST Corner Detection**. mi.eng.cam.ac.uk/˜er258/work/fast.html.

[8] **Flickr**. www.flickr.com.

[9] **ImageMagick**. www.imagemagick.org.

[10] **PhotoDNA**. www.microsoftphotodna.com.

[11] **Scale Saliency**. www.robots.ox.ac.uk/˜timork/salscale.html.

[12] **SURF: Speeded Up Robust Features**. www.vision.ee.ethz.ch/ surf.

[13] **Videntifier Forensics**. www.eff2.net.

[14] **Youtube**. www.youtube.com.

[15] ALAA E. ABDEL-HAKIM AND ALY A. FARAG. **CSIFT: A SIFT Descriptor with Color Invariant Characteristics**. In *CVPR (2)*, pages 1978–1983, 2006.

[16] YALI AMIT AND DONALD GEMAN. **Shape Quantization And Recognition With Randomized Trees**. *Neural Computation*, **9**(7):1545–1588, 1997.

[17] SEBASTIANO BATTIATO, GIOVANNI MARIA FARINELLA, GIOVANNI GALLO, AND DANIELE RAVÌ. **Scene categorization using bag of Textons on spatial hierarchy**. In *ICIP*, pages 2536–2539, 2008.

[18] SEBASTIANO BATTIATO, GIOVANNI MARIA FARINELLA, GIUSEPPE CLAUDIO GUARNERA, TONY MECCIO, GIOVANNI PUGLISI, DANIELE RAVÌ, AND ROSETTA RIZZO. **Bags of phrases with codebooks alignment for near duplicate image detection**. In *ACM workshop on Multimedia in forensics, security and intelligence*, pages 65–70, 2010.

[19] SEBASTIANO BATTIATO, GIOVANNI MARIA FARINELLA, MIRKO GUARNERA, GIUSEPPE MESSINA, AND D. RAVI. **Boosting Gray Codes for Red Eyes Removal**. In *ICPR*, pages 4214–4217, 2010.

[20] SEBASTIANO BATTIATO, MIRKO GUARNERA, TONY MECCIO, AND GIUSEPPE MESSINA. **Red Eye Detection through Bag-of-Keypoints Classification**. In *ICIAP*, pages 528–537, 2009.

[21] HERBERT BAY, ANDREAS ESS, TINNE TUYTELAARS, AND LUC J. VAN GOOL. **Speeded-Up Robust Features (SURF)**. *Computer Vision and Image Understanding*, **110**(3):346–359, 2008.

[22] HERBERT BAY, TINNE TUYTELAARS, AND LUC J. VAN GOOL. **SURF: Speeded Up Robust Features**. In *ECCV (1)*, pages 404–417, 2006.

[23] PAUL J. BENATI, ROBERT T. GRAY, AND PATRICK A. COSGROVE. **Automated detection and correction of eye color defects due to flash illumination**. *U.S. Patent*, (US5748764), 1998.

[24] CHRISTOPHER M. BISHOP. *Pattern Recognition and Machine Learning*. Springer, 2006.

[25] ANNA BOSCH, ANDREW ZISSERMAN, AND XAVIER MUÑOZ. **Scene Classification Using a Hybrid Generative/Discriminative Approach**. *IEEE Trans. Pattern Anal. Mach. Intell.*, **30**(4):712–727, 2008.

[26] BERNHARD E. BOSER, ISABELLE GUYON, AND VLADIMIR VAPNIK. **A Training Algorithm for Optimal Margin Classifiers**. In *COLT*, pages 144–152, 1992.

[27] LEO BREIMAN, JEROME H. FRIEDMAN, RICHARD A. OLSHEN, AND CHARLES J. STONE. *Classification and Regression Trees*. Wadsworth, 1984.

[28] PETER J. BURT, TSAI-HONG HONG, AND AZRIEL ROSENFELD. **Segmentation and Estimation of Image Region Properties through Cooperative Hierarchial Computation**. *Systems, Man and Cybernetics, IEEE Transactions on*, **11**(12):802–809, 1981.

[29] ONDREJ CHUM, MICHAL PERDOCH, AND JIRI MATAS. **Geometric min-Hashing: Finding a (thick) needle in a haystack**. In *CVPR*, pages 17–24, 2009.

[30] ONDREJ CHUM, JAMES PHILBIN, AND ANDREW ZISSERMAN. **Near Duplicate Image Detection: min-Hash and tf-idf Weighting**. In *BMVC*, pages 493–502, 2008.

[31] DORIN COMANICIU, VISVANATHAN RAMESH, AND PETER MEER. **Kernel-Based Object Tracking**. *IEEE Trans. Pattern Anal. Mach. Intell.*, **25**(5):564–575, 2003.

[32] CORINNA CORTES AND VLADIMIR VAPNIK. **Support-Vector Networks**. *Machine Learning*, **20**(3):273–297, 1995.

[33] BRAD EFRON. **The efficiency of logistic regression compared to normal discriminant analysis**. *Journal of the American Statistical Association*, **70**(352):892–898, 1975.

[34] GIOVANNI FARINELLA AND DANIELE RAVÌ. **Image Categorization**. In *Image Processing for Embedded Devices - From CFA data to image/video coding*, pages 237–269. Bentham, 2010.

[35] AHMET MUFIT FERMAN. **Automatic detection of red-eye artifacts in digital color photos**. In *ICIP*, pages 617–620, 2008.

[36] JOHN F. GANTZ. **The Diverse and Exploding Digital Universe - An Updated Forecast of Worldwide Information Growth Through 2011**. Technical report, International Data Corporation, 2008.

[37] FRANCESCA GASPARINI AND RAIMONDO SCHETTINI. **Automatic Redeye Removal for Smart Enhancement of Photos of Unknown Origin**. In *VISUAL*, pages 226–233, 2005.

[38] FRANCESCA GASPARINI AND RAIMONDO SCHETTINI. **Automatic Red-Eye Removal for digital photography**. In *Single-Sensor Imaging: Methods and Applications For Digital Cameras*, pages 429–457. CRC Press, 2008.

[39] FRANCESCA GASPARINI AND RAIMONDO SCHETTINI. **A Review of Redeye Detection and Removal in Digital Images Through Patents**. *Recent Patents on Electrical Engineering*, **2**(1):45–53, 2009.

[40] MATTHEW GAUBATZ AND ROBERT ULICHNEY. **Automatic red-eye detection and correction**. In *ICIP (1)*, pages 804–807, 2002.

[41] PIERRE GEURTS, DAMIEN ERNST, AND LOUIS WEHENKEL. **Extremely randomized trees**. *Machine Learning*, **63**(1):3–42, 2006.

[42] KRISTEN GRAUMAN AND TREVOR DARRELL. **The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features**. In *ICCV*, pages 1458–1465, 2005.

[43] JON YNGVE HARDEBERG. **Red Eye Removal using Digital Color Image Processing**. In *PICS*, pages 283–287, 2001.

[44] CHRIS HARRIS AND MIKE STEPHENS. **A Combined Corner and Edge Detection**. In *Alvey Vision Conference*, pages 147–151, 1988.

[45] JOHN. A. HARTIGAN AND M. ANTHONY WONG. **A K-Means Clustering Algorithm**. *Applied Statistics*, **28**(1):100–108, 1979.

[46] BERND HEISELE, ALESSANDRO VERRI, AND TOMASO POGGIO. **Learning and vision machines**. *Proceedings of the IEEE*, **90**(7):1164–1177, 2002.

[47] ANDREAS HELD. **Model-Based Correction of Red-Eye Defects**. In *Color Imaging Conference*, pages 223–228, 2002.

[48] YIQUN HU, XIANGANG CHENG, LIANG-TIEN CHIA, XING XIE, DEEPU RAJAN, AND AH-HWEE TAN. **Coherent phrase model for efficient image near-duplicate retrieval**. *IEEE Trans. Multi.*, **11**(8):1434–1445, 2009.

[49] LAURENT HYAFIL AND RONALD L. RIVEST. **Constructing Optimal Binary Decision Trees is NP-Complete**. *Inf. Process. Lett.*, **5**(1):15–17, 1976.

[50] SERGEY IOFFE. **Red eye detection with machine learning**. In *ICIP (2)*, pages 871–874, 2003.

[51] TONY JEBARA. *Machine Learning : Discriminative and Generative*. Springer, 2003.

[52] ANDREW EDIE JOHNSON AND MARTIAL HEBERT. **Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes**. *IEEE Trans. Pattern Anal. Mach. Intell.*, **21**(5):433–449, 1999.

[53] IAN T. JOLLIFFE. *Principal Component Analysis*. Springer, second edition, 2002.

[54] GORDON V. KASS. **An exploratory technique for investigating large quantities of categorical data**. *Applied Statistics*, **29**:119–127, 1980.

[55] YAN KE AND RAHUL SUKTHANKAR. **PCA-SIFT: A More Distinctive Representation for Local Image Descriptors**. In *CVPR (2)*, pages 506–513, 2004.

[56] YAN KE, RAHUL SUKTHANKAR, AND LARRY HUSTON. **Efficient near-duplicate detection and sub-image retrieval**. In *ACM Multimedia*, pages 869–876, 2004.

[57] JAN J. KOENDERINK. **The structure of images**. *Biological Cybernetics*, **50**(5):363–370, 1984.

[58] SVETLANA LAZEBNIK, CORDELIA SCHMID, AND JEAN PONCE. **Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories**. In *CVPR (2)*, pages 2169–2178, 2006.

[59] VINCENT LEPETIT AND PASCAL FUA. **Keypoint Recognition Using Randomized Trees**. *IEEE Trans. Pattern Anal. Mach. Intell.*, **28**(9):1465–1479, 2006.

[60] T. LINDEBERG. **Scale-space theory: A basic tool for analysing structures at different scales**. *Applied Statistics*, **21**(2):224–270, 1994.

[61] DAVID G. LOWE. **Object Recognition from Local Scale-Invariant Features**. In *ICCV*, pages 1150–1157, 1999.

[62] DAVID G. LOWE. **Distinctive Image Features from Scale-Invariant Keypoints**. *International Journal of Computer Vision*, **60**(2):91–110, 2004.

[63] HUITAO LUO, JONATHAN YEN, AND DAN TRETTER. **An Efficient Automatic Redeye Detection and Correction Algorithm**. In *ICPR (2)*, pages 883–886, 2004.

[64] STÉPHANE MALLAT. **A Theory for Multiresolution Signal Decomposition: The Wavelet Representation**. *IEEE Trans. Pattern Anal. Mach. Intell.*, **11**(7):674–693, 1989.

[65] LUCA MARCHESOTTI, MARCO BRESSAN, AND GABRIELA CSURKA. **Safe Red-Eye Correction Plug-in Using Adaptive Methods**. In *ICIAPW*, pages 192–165, 2007.

[66] RAPHAËL MARÉE, PIERRE GEURTS, JUSTUS H. PIATER, AND LOUIS WEHENKEL. **Random Subwindows for Robust Image Classification**. In *CVPR (1)*, pages 34–40, 2005.

[67] JIRI MATAS, ONDREJ CHUM, MARTIN URBAN, AND TOMÁS PAJDLA. **Robust Wide Baseline Stereo from Maximally Stable Extremal Regions**. In *BMVC*, 2002.

[68] TONY MECCIO AND GIUSEPPE MESSINA. **Red Eyes Removal**. In *Image Processing for Embedded Devices - From CFA data to image/video coding*, pages 191–216. Bentham, 2010.

[69] XIAO-PING MIAO AND T. SIM. **Automatic red-eye detection and removal**. In *ICME*, pages 1195–1198, 2004.

[70] KRYSTIAN MIKOLAJCZYK AND CORDELIA SCHMID. **Scale & Affine Invariant Interest Point Detectors**. *International Journal of Computer Vision*, **60**(1):63–86, 2004.

[71] KRYSTIAN MIKOLAJCZYK AND CORDELIA SCHMID. **A Performance Evaluation of Local Descriptors**. *IEEE Trans. Pattern Anal. Mach. Intell.*, **27**(10):1615–1630, 2005.

[72] KRYSTIAN MIKOLAJCZYK, TINNE TUYTELAARS, CORDELIA SCHMID, ANDREW ZISSERMAN, JIRI MATAS, FREDERIK SCHAFFALITZKY, TIMOR KADIR, AND LUC J. VAN GOOL. **A Comparison of Affine Region Detectors**. *International Journal of Computer Vision*, **65**(1-2):43–72, 2005.

[73] JOSE M. MIR. **Apparatus and Method for Minimizing Red-Eye in Flash Photography**. *U.S. Patent*, (US4285588), 1981.

[74] FRANK MOOSMANN, ERIC NOWAK, AND FRÉDÉRIC JURIE. **Randomized Clustering Forests for Image Classification**. *IEEE Trans. Pattern Anal. Mach. Intell.*, **30**(9):1632–1646, 2008.

[75] FRANK MOOSMANN, BILL TRIGGS, AND FRÉDÉRIC JURIE. **Fast Discriminative Visual Codebooks using Randomized Clustering Forests**. In *NIPS*, pages 985–992, 2006.

[76] ANDREW Y. NG AND MICHAEL I. JORDAN. **On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes**. In *NIPS*, pages 841–848, 2001.

[77] AUDE OLIVA AND ANTONIO TORRALBA. **Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope**. *International Journal of Computer Vision*, **42**(3):145–175, 2001.

[78] MUSTAFA ÖZUYSAL, MICHAEL CALONDER, VINCENT LEPETIT, AND PASCAL FUA. **Fast Keypoint Recognition Using Random Ferns**. *IEEE Trans. Pattern Anal. Mach. Intell.*, **32**(3):448–461, 2010.

[79] MUSTAFA ÖZUYSAL, PASCAL FUA, AND VINCENT LEPETIT. **Fast Keypoint Recognition in Ten Lines of Code**. In *CVPR*, 2007.

[80] CHRISTOS H. PAPADIMITRIOU AND KENNETH STEIGLITZ. *Combinatorial optimization: algorithms and complexity*. Prentice-Hall, Inc., 1982.

[81] ANDREW J. PATTI, KONSTANTINOS KONSTANTINIDES, DANIEL TRETTER, AND QIAN LIN. **Automatic Digital Redeye Reduction**. In *ICIP (3)*, pages 55–59, 1998.

[82] LAURA WALKER W. RENNINGER AND JITENDRA MALIK. **When is scene identification just texture recognition?** *Vision research*, **44**(19):2301–2311, 2004.

[83] RONALD L. RIVEST. **The MD5 Message-Digest Algorithm (RFC 1321)**, 1992.

[84] ILIA V. SAFONOV. **Automatic Red-Eye Detection**. In *GraphiCon*, 2007.

[85] JAY S. SCHILDKRAUT AND ROBERT T. GRAY. **A fully automatic redeye detection and correction algorithm**. In *ICIP (1)*, pages 801–803, 2002.

[86] CLAUDE E. SHANNON. **A mathematical theory of communication**. *Bell system technical journal*, **27**, 1948.

[87] JAMIE SHOTTON, MATTHEW JOHNSON, AND ROBERTO CIPOLLA. **Semantic texton forests for image categorization and segmentation**. In *CVPR*, 2008.

[88] SUDIPTA N. SINHA, DREW STEEDLY, RICHARD SZELISKI, MANEESH AGRAWALA, AND MARC POLLEFEYS. **Interactive 3D architectural modeling from unordered photo collections**. *ACM Trans. Graph.*, **27**(5):159, 2008.

[89] BOGDAN SMOLKA, K. CZUBIN, JON YNGVE HARDEBERG, KOSTAS N. PLATANIOTIS, MAREK SZCZEPANSKI, AND KONRAD W. WOJCIECHOWSKI. **Towards automatic redeye effect removal**. *Pattern Recognition Letters*, **24**(11):1767–1785, 2003.

[90] RICHARD SZELISKI. *Computer Vision: Algorithms and Applications*. Springer, 2010.

[91] ANTONIO TORRALBA. **Contextual Priming for Object Detection**. *International Journal of Computer Vision*, **53**(2):169–191, 2003.

[92] ANTONIO TORRALBA AND PAWAN SINHA. **Statistical Context Priming for Object Detection**. In *ICCV*, pages 763–770, 2001.

[93] TINNE TUYTELAARS AND KRYSTIAN MIKOLAJCZYK. **Local Invariant Feature Detectors: A Survey**. *Foundations and Trends in Computer Graphics and Vision*, **3**(3):177–280, 2007.

[94] FLAVIEN VOLKEN, JOHANN TERRIER, AND PATRICK VANDEWALLE. **Automatic Red-Eye Removal based on Sclera and Skin Tone Detection**. In *CGIV*, pages 359–364, 2006.

[95] YINGZE WANG, YU CHEN, JIANZHUANG LIU, AND XIAOOU TANG. **3D reconstruction of curved objects from single 2D line drawings**. In *CVPR*, pages 1834–1841, 2009.

[96] JUTTA WILLAMOWSKI AND GABRIELA CSURKA. **Probabilistic Automatic Red Eye Detection and Correction**. In *ICPR (3)*, pages 762–765, 2006.

[97] ANDREW P. WITKIN. **Scale-Space Filtering**. In *IJCAI*, pages 1019–1022, 1983.

[98] JING-HAO XUE AND D. MIKE. TITTERINGTON. **Comment on "On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes"**. *Neural Processing Letters*, **28**(3):169–187, 2008.

[99] SEUNGHWAN YOO AND RAE-HONG PARK. **Red-eye detection and correction using inpainting in digital photographs**. *IEEE Transactions on Consumer Electronics*, **55**(3):1006–1014, 2009.

[100] LEI ZHANG, YANFENG SUN, MINGJING LI, AND HONGJIANG ZHANG. **Automated red-eye detection and correction in digital photographs**. In *ICIP*, pages 2363–2366, 2004.