UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

IMAGE DESCRIPTIONS FOR SKETCH BASED IMAGE RETRIEVAL

THESIS SUBMITTED FOR THE DEGREE OF PHD IN COMPUTER SCIENCE

JOSÉ MANUEL SAAVEDRA RONDO

This work has been partially funded by CONICYT.

SANTIAGO DE CHILE
2013, JANUARY

Resumen

Debido al uso masivo de Internet y a la proliferación de dispositivos capaces de generar información multimedia, la búsqueda y recuperación de imágenes basada en contenido se han convertido en áreas de investigación activas en ciencias de la computación. Sin embargo, la aplicación de búsqueda por contenido requiere una imagen de ejemplo como consulta, lo cual muchas veces, puede ser un problema serio, que imposibilite la usabilidad de la aplicación. En efecto, los usuarios comúnmente hacen uso de un buscador de imágenes porque no cuentan con la imagen deseada. En este sentido, un modo alternativo de expresar lo que el usuario intenta buscar es mediante un dibujo a mano compuesto, simplemente, de trazos, *sketch*, lo que conduce a la búsqueda por imágenes basada en *sketches*. Hacer este tipo de consultas es soportado, además, por el hecho de haberse incrementado la accesibilidad a dispositivos táctiles, facilitando realizar consultas de este tipo.

En este trabajo, se proponen dos métodos aplicados a la recuperación de imágenes basada en *sketches*. El primero es un método global que calcula un histograma de orientaciones usando gradientes cuadrados. Esta propuesta exhibe un comportamiento sobresaliente con respecto a otros métodos globales. En la actualidad, no existen métodos que aprovechen la principal característica de los *sketches*, la información estructural. Los *sketches* carecen de color y textura y representan principalmente la estructura de los objetos que se quiere buscar. En este sentido, se propone un segundo método basado en la representación estructural de las imágenes mediante un conjunto de formas primitivas que se denominan *keyshapes*.

Los resultados de nuestra propuesta han sido comparados con resultados de métodos actuales, mostrando un incremento significativo en la efectividad de la recuperación. Además, puesto que nuestra propuesta basada en *keyshapes* explota una característica novedosa, es posible combinarla con otras técnicas para incrementar la efectividad de los resultados. Así, en este trabajo se ha evaluado la combinación del método propuesto con el método propuesto por Eitz et al., basado en *Bag of Words*, logrando un aumento de la efectividad de casi 22%.

Finalmente, con el objetivo de mostrar el potencial de nuestra propuesta, se muestran dos aplicaciones. La primera está orientada al contexto de recuperación de modelos 3D usando un dibujo a mano como consulta. En esta caso, nuestros resultados muestran competitividad con el estado del arte. La segunda aplicación explota la idea de buscar objetos basada en la estructura para mejorar el proceso de segmentación. En particular, mostramos una aplicación de segmentación de manos en ambientes semi-controlados.

Abstract

Due to the massive use of Internet together with the proliferation of media devices, content based image retrieval has become an active discipline in computer science. A common content based image retrieval approach requires that the user gives a regular image (e.g. a photo) as a query. However, having a regular image as query may be a serious problem. Indeed, people commonly use an image retrieval system because they do not count on the desired image. An easy alternative way to express what the user is looking for is by giving a line-based hand-drawing, a sketch, leading to the sketch-based image retrieval (SBIR). This kind of query is also supported by the fact that emerging touch screen based technology is becoming more popular, allowing the user to make an sketch directly on the screen.

In this work, we propose methods aiming to deal with the sketch based image retrieval problem. The first method is a global method that computes a histogram of local orientations based on the gradient squared technique. This proposal method exhibits outperforming results in comparison with current global methods. To the best of our knowledge, any of the current sketch based image retrieval approaches have not exploited yet the structural composition of an image (or sketch) represented by a set of strokes. Therefore, the second proposed method is a local approach that exploits the structural information given a sketch representation. This method is based on representing an image by a set of simple shapes called *keyshapes*. Each keyshape belongs to one of five keyshape classes. The set of keyshapes computed over an image defines the image structure representing the image in a higher semantic level than interest points do. To this end, we propose a method for detecting keyshapes that processes stroke pieces to classify it as one of the keyshape classes.

We experimentally compare our results with current methods, showing an increase in the retrieval effectiveness. We also demonstrate that our approach may successfully be combined with other leading methods. The reason for that stems from the fact that our method is based on a novel feature, different from those used by the other current methods. We show that a combination of our method with one of the state of the art significantly outperforms current SBIR methods, achieving an increase in effectiveness of almost 22%.

Finally, in this work we show two applications of our keyshape-based approach. In the first case, we adapt our method to be applied in the context of sketch-based 3D model retrieval, achieving competitive results with respect to current methods. In the second case, we extend our method for the hand segmentation problem in semi-controlled environments. In particular our keyshape-based approach permits locating a hand in an image in order to improve the subsequent segmentation stage.

To my wife, Viole, and my little daughter, Adrianita, for giving me the motivation to finish this thesis.

Acknowledgements

I would like to acknowledge all people who collaborate with me providing interesting ideas to address the sketch based retrieval in the image domain as well as in the 3D model domain. In particular, I thank people who helped drawing sketches or providing data set for experimental purposes.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Over the last years, the massive use of Internet has fueled the study of information retrieval systems which has become a very active research area among computer sciences. The classical approach for information retrieval is that based on textual information where a user writes a textual query composed of target concepts and the retrieval system returns a ranking with the most relevant documents stored in a database (or on the web). From the early days of Internet to present, there has been a great effort in the research community to develop efficient algorithms capable of searching and indexing textual documents [90, 4]. The result of this effort is reflected in the fact that the usage of search engines nowadays is an ubiquitous activity with results characterized by efficiency and effectiveness.

Because of the proliferation of media devices, the web content is no longer based only on textual documents, but also on multimedia content like images, videos, music, etc. Ongoing research on information retrieval is focused on multimedia information retrieval and in particular, image retrieval systems has attracted a great number of researchers coming from different communities like computer vision, multimedia retrieval, data mining, among others, leading to a vast number of related publications. The reason why image retrieval is the focus of much research is basically due to the fact that images are ubiquitous and easy to capture by users.

An image retrieval system returns a set of images which are ranked in a certain order under a similarity function in response to a query given by a user. A simple way to formulate a query is by textual concepts. Although this approach is the most common way in which users make queries, it requires images to be represented by a set of textual information called *metadata*. In this way, the image retrieval based on concepts is reduced to a textual information retrieval. Unfortunately, this approach still has many serious limitations. For instance, not all images in the web are correctly tagged. In addition, the metadata describing an image does not reflect the whole content of the image reducing the expressiveness power provided by the images themselves. To describe completely the image content using concepts, we would have to possibly write long texts, which becomes impractical. To address these problems, current research is focused on searching by the image content itself leading to what is known as the content based image retrieval approach.

Content based image retrieval (CBIR) requires another image as query as opposed to the concept based query. The retrieval system has to extract some relevant information from the input image and from those stored in the database (test images). The extracted information is commonly represented as vectors of numerical features that will be compared by using a similarity function. Traditionally, a CBIR system receives a regular image showing color and texture as the input. In this way, in order to start the search, the user must have an example image resembling what the user is looking for.

Although content based image retrieval seems to overcome the aforementioned drawbacks related to the concept based image retrieval, the fact of having a regular image as query may be a serious problem. Indeed, people commonly use an image retrieval system because they do not count on the desired image, thereby, having such an image query may not be possible, limiting the image retrieval system usability.

An alternative for querying is by simply drawing what the user has in mind. That is, making a sketch of what the user expects as an answer could overcome the absence of a regular example image. This kind of query is also supported by the fact that emerging touch screen based technology is becoming more popular, allowing the user to make a sketch directly on the screen. Of course, a sketch may be enriched by adding color, however we claim that making a sketch only by strokes is the easiest and natural way for querying. Therefore, in this work, our focus is on image retrieval using stroke-based sketches as queries.

1.1 Related Work

Although a vast number of authors have been involved in content based image retrieval using an example image as query, some relevant works on sketch based image retrieval have just appeared in the last three years. The work of Eitz et al. [31] is one of the most relevant methods in this context. They propose two techniques based on the well known *SIFT* and *Shape Context* approaches for extracting relevant information from sketch representations. The extracted information, in the form of feature vectors, is clustered to form a codebook that is then used under the Bag of Features (BoF) approach. This work is also relevant because the authors propose the first systematically built benchmark for the SBIR problem. One outstanding property of this benchmark is that it takes into account the user opinion about the similarity between sketches and test images. This is highly important because the ultimate goal of a retrieval systems is to satisfy the user requirements.

Another approach is that proposed by Hu et al. [44, 45]. This work is also based on the Bag of Feature approach but addressed the extraction information step in a very different way with respect to the Eitz's approach. The novel idea in this work is the transforming of sketch representations into gradient field (GF) images. The test images are converted into sketch representations by edge maps using the Canny operator. The GF images are then used to compute HOG descriptors in three different scales with respect to each edge pixel. After that, a BoF approach is applied to form a frequency histogram. This method requires solving a sparse linear equation system where the number of variables is the order of the size of the underlying image.

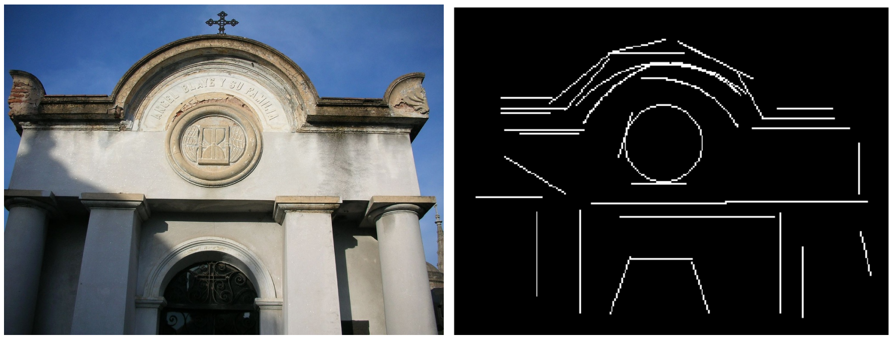


Figure 1.1: A keyshape representation example.

Another work was recently presented by Yan Cao et al. [17]. This work presents a method based on the Chamfer Distance [12]. The method does not present any kind of invariance, although the authors present a technique to deal with large database based on the inverted index structure.

Moreover, to the best of our knowledge, the SBIR methods proposed so far have not exploited yet one of the main features of a sketch representation, its *structure*. In fact, a sketch is a simple drawing that lacks color or texture but maintains the structural composition of what the user is looking for. For instance, if we are interested in retrieving pictures containing the sun over a beach, we will probably draw a circle representing the sun and some horizontal lines representing the beach. These shapes (circle and lines) represent structural components. In addition, the term structure also implies a relationship between the underlying structural components. In this way, it is not the same to draw a circle over the lines as to draw a circle below the lines, with respect to our aforementioned example.

1.2 Contributions of this Thesis

Therefore, the contribution of this thesis is to propose novel approaches to dealing with the sketch based image retrieval problem. Our first proposal is a global approach that is based on representing sketches by a *histogram of edge local orientations* (HELO). This technique is inspired by the estimation of orientations in the context of fingerprint processing using the square gradient method. This proposal also shows outperforming results when it is compared with other SBIR global methods.

The second proposal is a local technique that takes into account structural information from sketch representations. This approach is novel because it is the first method that takes into account the structural composition of a sketch representation. We represent the structural components of a sketch by decomposing it into a set of primitive geometric shapes named *keyshapes*. An example of a keyshape representation obtained from an input image is showed in Figure 1.1 . Further, our proposal forms feature vectors that take into account the spatial relationship between the keyshapes. Our results show an increase in the retrieval effectiveness with respect to current methods.

We also demonstrate that our keyshape based approach may successfully be combined with another leading method. The reason for that stems from the fact that our method is based on a novel feature, different from those used by the other current methods. Specifically, we show that a combination of our method with the Eitz's proposal significantly outperforms current SBIR methods, achieving an increase in terms of effectiveness of almost 22%.

Additionally to the image retrieval domain, we also present two novel approaches for the sketch-based 3D model retrieval problem which are extended from our keyshape based method. The first approach named STELA, derived from *StrucTurE based Local Approach*, is a local technique that leverages the structural information provided by sketch strokes. Furthermore, in this approach, a variation of the HELO descriptor is used as a filtering step to reduce the database size. The second proposal named HKO-KASD as acronym of *Histogram of Keyshape Orientations - Keyshape Angular Spatial Descriptor* also exploits the structural information from sketch representation, but different from STELA, it computes a local descriptor composed of the local distribution of four types of strokes. In addition, for the filtering step, this proposal applies a global technique based on the distribution of keyshape orientations.

The results of our proposal show an increase in precision for many classes of 3D models with respect to current strategies applied for sketch-based 3D model retrieval. Particularly, our method achieves significant improvement over 3D models with a well defined structure as explained later in this chapter.

We also show in this work, how our keyshape based method, that takes into account the structural information of objects, can be used in a very different domain as the case of the *hand segmentation* problem. In particular, we use the STELA technique for finding a hand in an image. Using the hand location we extract a training region that is used to get an accurate segmentation of hand. Interesting applications may be derived from this proposal, for instance a non-intrusive system may be developed for biometric recognition using palmprints. In this case, the first stage would be the segmentation of a hand whose result would be the input for the subsequent analysis.

Our contribution on this thesis is supported by a set of publications related to our proposed methods. First, the HELO [80] descriptor is published in the *Proceedings of the 32nd Annual Symposium of the German Association for Pattern Recognition* (DAGM 2010). Second, the STELA proposal [82] is published in the *Proceedings of the ACM International Conference in Multimedia Retrieval (ICMR 2011).* Third, the HKO-KASD [83] is published in the *Proceeding of the Eurographics Workshop on 3D Object Retrieval* (2012) together with a SHREC competition paper [57]. Fourth, the work where we use STELA to improve the accuracy of the hand segmentation task was accepted to be presented in the International Conference on Computer Vision Theory and Applications (VISAPP 2013). Finally, we recently have submitted the paper entitled "Sketch based Image Retrieval using Keyshapes" to Multimedia Tools and Applications [81].

The organization of this thesis is established as follows:

- Chapter 2 describes basic concepts that are necessary to understand the subsequent chapters.

- Chapter 3 defines the sketch based image retrieval problem and the related work known so far.
- Chapter 4 describes our global proposal, HELO.
- Chapter 5 describes the *keyshape* based proposal.
- Chapters 6 and 7 present extensions of our proposal for 3D model retrieval based on sketches and hand segmentation, respectively.
- Chapter 8 discusses the conclusions of this work.

1.3 Evaluation Methodology

On the one hand, to evaluate the performance of our global descriptor we have developed a benchmark containing a set of test images and a set of sketch images (queries). For the test image database, we have randomly selected 1326 images. We selected 1285 color images from the Caltech101 database [34]. Additionally, we added 46 images of castles and palaces to our database.

For the query database, we have chosen 53 images from the test database. For each chosen image, a line-based sketch was hand-drawn resembling accurately the underlying chosen image. Then, the evaluation of the methods was performed by querying each query sketch for the most similar images and finding the target image rank. The lower the value of the rank, the better the method.

On the other hand, to evaluate our local approach based in *keyshapes*, we use a public benchmark proposed by Eitz et al. [31]. We pick this benchmark because this was used to compare state-of-the-art local approaches.

The Eitz's benchmark consists of 31 query sketches, each one associated with a set of 40 test images. In addition, each test image has been ranked by people using a 7-point Likert scale with 1 representing the best rank and 7 representing the worst rank. This provides the baseline ranking which is called the *user ranking*. The evaluation of a method is carried out by comparing the ranking of the method against the user ranking. To this end, Eitz et al. propose to use the *Kendall's correlation* that determines how similar two rankings are.

1.4 Applications of this Thesis

A potential application of a SBIR system could promote the cognitive development of children who have not yet acquired writing skills. In this way, a SBIR will allow children to draw simple shapes and the SBIR system will return a set of images resembling what the child drew. This may allow children to associate abstract shapes with real objects, improving their ability to understand the real environment. This application is also supported by the emerging touch-screen based technology that allows any kind of users to make a sketch query drawing directly on the screen.

People with hand motility disability is another user group that may receive significant benefits from a SBIR solution. The impact of a SBIR application becomes more relevant due to the fact that the majority of computer applications commonly are not accessible for this kind of users. In this way, a SBIR system would provide great benefits to people with hand-motility issues who can not interact with a computer, writing queries or taking pictures to use them as queries. Moreover, current research on gaze interaction [13, 91] have allowed disabled people to interact with computer making any kind of task. However, gaze interaction may result in an overload work for eyes, specially when the user is exposed to this kind of interaction for long time. Making a sketch may result faster than typing a query, letter by letter, with an eye tracker system. Therefore, an image retrieval based on sketches offers an amazing alternative for people with hand-motility issues, reducing in this way the time for making a query.

Chapter 2

Basic Concepts

2.1 Introduction

In this chapter we present a review of basic concepts which are required to completely understand the proposal methods discussed in this thesis. In particular, we focus on describing fundamental operations in image processing. The image processing operations are frequently required in the pre-processing stage in any application of computer vision. In particular, we describe linear filtering, edge detectors and morphological operators.

Furthermore, considering that this thesis falls in the context of image retrieval, we dedicate some sections in this chapter for describing classical techniques applied for the content based image retrieval problem. In this regard, we review techniques based on color, shape and texture properties.

2.2 Image Processing

Image processing is a key discipline in computer vision, in fact, it is the first stage in most computer vision applications. The idea is to pre-process an image to highlight some kind of information in order to facilitate its subsequent analysis. For instance, getting edge information of an image is very important in the context of sketch processing. Other useful operations applied in computer vision include exposure correction, color balancing, noise reduction, increasing sharpness, among others [94]. Image processing is a fundamental piece in the computer vision domain in such a way that the results of many computer vision applications depend on the well design of the image processing algorithms. Moreover, image processing in the context of image retrieval should be understood as a step for enhancing the image information, not for describing the content of the image in its entirety [25].

In the image processing domain we can classify two types of operators. The first one covers those operators that compute each output pixel value depending solely on the corresponding input pixel value (plus, potentially, some globally collected information or parameters). This

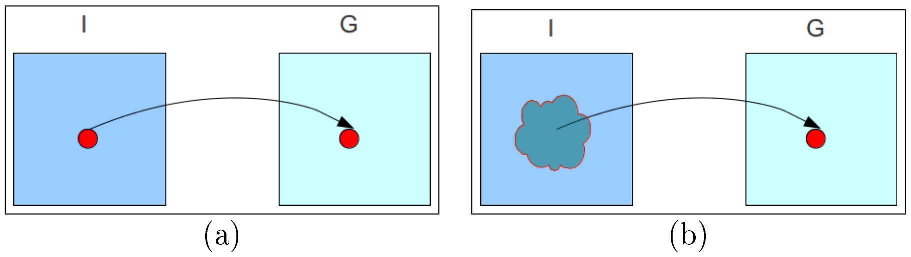


Figure 2.1: (a) Point operator scheme. (b) Neighborhood operator scheme.

class of operators is known as *point operators*. Examples of such operators include brightness and contrast adjustment as well as color correction and thresholding. A complete description of these operators can be found in many text books such as the book of Gonzalez et al. [40] and the book of Russ et al. [49].

The second operator type covers operators which take into account a neighborhood of the input pixel to compute its corresponding output value. These neighborhood operators are known as *filters* and allow us to remove noise, sharpen details, accentuate edges, among many other tasks. In Figure 2.1 we show a scheme representing the behavior of a point operator and a neighborhood operator.

In the subsequent sections we will describe some of the most common neighborhood operators or *filters*.

2.2.1 Linear Filtering

The most commonly used type of neighborhood is a linear filter, in which an output pixel value is determined as a weighted sum of input pixel values. The weights are represented by means of a mask known as *kernel*. A simple equation to calculate an output value with respect to a pixel (x, y) (the new pixel value) is shown below, where it uses a $(2r+1) \times (2r+1)$-size kernel.

$$g(x, y) = \sum_{k,l} f(x-k, y-l)h(k,l) \tag{2.1}$$

The entries in the weight kernel are often called the filter coefficients. The kernel is also known as a *convolution mask*. The above equation can be written in a more compact way as:

$$g = f \otimes h \tag{2.2}$$

The symbol $\otimes$ is the convolution operator and it is, probably, the most common operator in image processing. Depending on the designing of the mask, one could get diverse results under a given input image. An example of how the convolution works is shown in Figure 2.2.

Generating an $m \times n$ linear spatial filter requires that we specify $m \times n$ mask coefficients. These coefficients are selected based upon what the filter is supposed to do. Next, we will describe some commonly used filters.

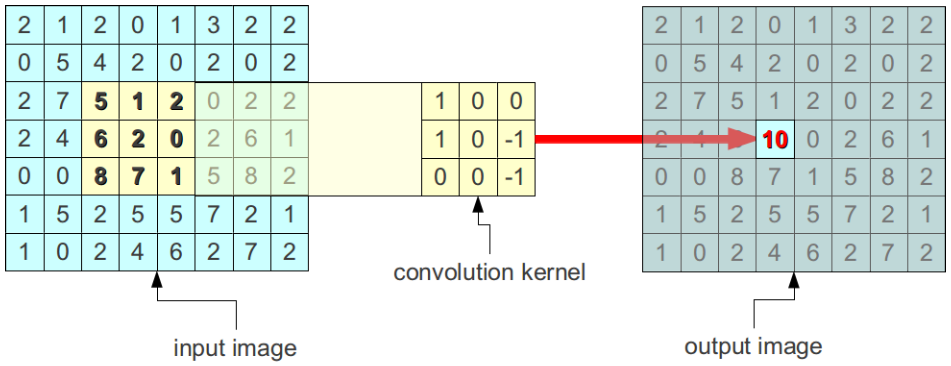


Figure 2.2: An example of a convolution process.

Gaussian Filter

In this case, the coefficients of the filter are calculated using the following continuous function of two variables:

$$h(x, y) = \frac{1}{2 * \pi * \sigma} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.3}$$

where σ is the standard deviation and x, y are integers which take values depending on the h's support. Remember that a 2D Gaussian function has a *bell* shape, and that the standard deviation controls the tightness of the bell. For example, a 3×3 Gaussian filter mask with $\sigma = 0.5$ is shown below:

$$\begin{bmatrix} 0.0113 & 0.0838 & 0.0113 \\ 0.0838 & 0.6193 & 0.0838 \\ 0.0113 & 0.0838 & 0.0113 \end{bmatrix}$$

A set of examples applying a Gaussian filter over an input image is shown in Figure 2.3, where we use a 5×5 neighborhood with $\sigma = 0.5$ and $\sigma = 1$. The greater the value of σ, the stronger the blur effect.

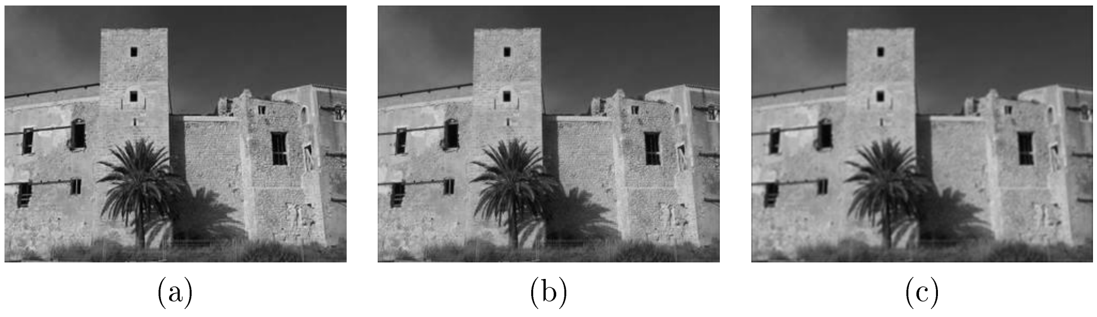


Figure 2.3: (a) Original image. (b) Image convolved using a 3×3 kernel with $\sigma = 0.5$. (c) Image convolved using a 3×3 kernel with $\sigma = 1$.

Edge Detector

Edge pixels are pixels at which the intensity of an image function changes abruptly, and edges (or edge segments) are sets of connected edge pixels. The edge detector methods also

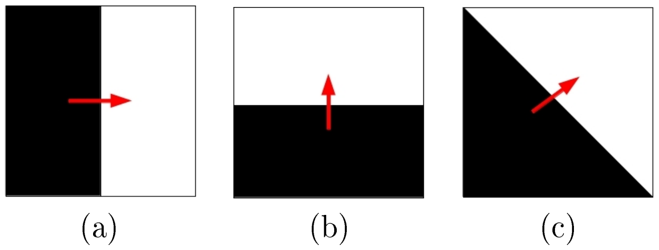


Figure 2.4: (a) Gradient in horizontal direction. (b) Gradient in vertical direction. (c) Gradient in diagonal direction.

fall on the group of neighborhood operators. They are also known as *high-pass filters* since edges are characterized by showing high frequency in the frequency domain.

Edge operators work finding local changes on the image. Local changes in intensity can be detected using derivatives. In this way, the first and second order derivatives are particularly well suited for this purpose.

The mathematical tool for finding edge strength and direction at location (x, y) of an image f, is the gradient, denoted by ∇f, and defined as follows:

$$\nabla f = grad(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}. \tag{2.4}$$

An important geometrical property of the gradient vector is that it always points out in the direction of the greatest rate of change of f at location (x, y) as shown in Figure 2.4.

The magnitude of vector ∇f, denoted as $M(x, y)$ is defined by:

$$M(x, y) = mag(\nabla f) = \sqrt{g_x^2 + g_y^2}. \tag{2.5}$$

$M(x, y)$ is the value of the rate of change in the direction of the gradient vector at (x, y). We must note that g_x, g_y, and $M(x, y)$ are images that are the same size as the original image f. In addition, the direction of the gradient vector is given by the angle

$$\alpha(x, y) = tan^{-1} \begin{bmatrix} \frac{g_y}{g_x} \end{bmatrix}. \tag{2.6}$$

Getting the gradient of an image requires computing the partial derivatives $\partial f/\partial x$ and $\partial f/\partial y$ at every pixel on the image. A discrete approximation of the partial derivatives at a pixel (x, y) is:

$$g_x = \frac{\partial f(x, y)}{\partial x} = f(x + 1, y) - f(x, y), \tag{2.7}$$

$$g_y = \frac{\partial f(x, y)}{\partial x} = f(x, y + 1) - f(x, y). \tag{2.8}$$

It is possible to write the above equations as the convolution masks g_x^M and g_y^M, which are computed as follows:

$$g_x^M = \begin{bmatrix} -1 & 1 \end{bmatrix}, g_y^M = \begin{bmatrix} -1 \\ 1 \end{bmatrix}. \tag{2.9}$$

When a 3×3 neighborhood is considered, the simplest approximation is by using the Prewitt masks (Prewitt operator):

$$g_x^P = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, g_y^P = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}. \tag{2.10}$$

A slight variation on the Prewitt operator uses a weight of 2 in the center. This variation is known as the Sobel operators, and it is the most common operator to highlight edges in images. The reason why this operator is preferable relies in its characteristic to reduce noise (smoothing) in an image. The corresponding masks for the Sobel operator are shown below:

$$g_x^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, g_y^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}. \tag{2.11}$$

The edge masks discussed previously allow us to compute the gradient images G_x and G_y by convolution. For instance, if we choose to use the Sobel approximation to detect edges from the image f, the corresponding gradient images G_x and G_y are computed as shown below:

$$G_x = f \otimes g_x^S \tag{2.12}$$
$$G_y = f \otimes g_y^S \tag{2.13}$$

Using these two images it is possible to estimate edge strength and direction using Equations 2.5 and 2.6. Additionally, in order to avoid time consuming issues, the magnitude may be computed without using multiplications in the following way:

$$M(x, y) \approx |G_x| + |G_y| \tag{2.14}$$

Examples of using edge detector operators are shown in Figure 2.5

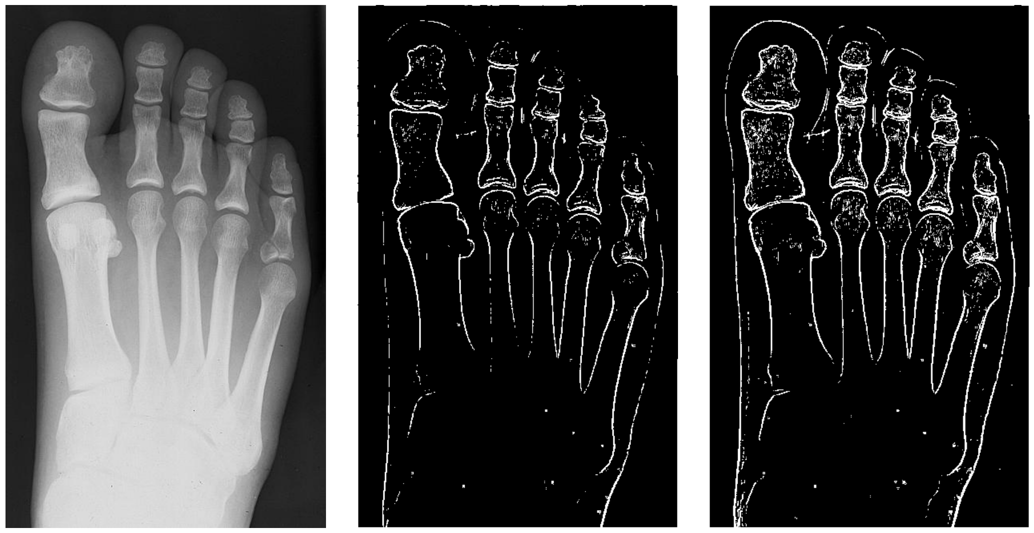
2.2.2 Canny Edge Detector

The Canny operator [16] allows us to improve the quality of the edge detection procedure based on minimizing spurious response, increasing the accuracy of the detected edge points, and maximizing the single edge point response.

First, the Canny operator applied a smoothing operator over an image f. To this end, a Gaussian filter g is applied on f by convolution. This operation is followed by computing the gradient magnitude $M(x, y)$ and the direction $\alpha(x, y)$ at each point (x, y). To approximate the gradients image G_x and G_y, the Sobel operator is used.

The next step, known as the *nonmaxima suppression*, is to thin those ridges that commonly appear wide around a local maxima. For nonmaxima suppression the gradient direction at each pixel is quantized in four directions: horizontal, vertical, dir_45 (45°), and dir_135 (135°).

Let $\mathrm{d}_1, \mathrm{d}_2, \mathrm{d}_3$, and d_4 be the four directions mentioned. The nonmaxima suppression proceeds as follows:



(a) (b) (c)

Figure 2.5: Examples of detecting edges on the image shown in (a). (b) Using Prewitt Operator. (c) Using Sobel Operator.

```
1: Let $g_N$ be the nonmaxima suppression image.
2: **for** each point $(x, y)$ **do**
3:     Find the direction $\mathrm{d}_k$ that is closest to $\alpha(x, y)$.
4:     **if** $M(x, y)$ is less than at least one of its two closest neighbors along $\mathrm{d}_k$ **then**
5:         $g_N(x, y) = 0$ (suppression);
6:         $g_N(x, y) = M(x, y)$
7:     **end if**
8: **end for**
```

The final step is called *hysteresis thresholding.* To this end, the Canny operator uses two thresholds Th_{low} and Th_{high}. Therefore, a pixel (x, y) with $M(x, y) < Th_{low}$ is discarded as edge pixel; if $M(x, y) >= Th_{high}$, the pixel (x, y) is marked as a *strong* edge pixel; otherwise (i.e. $Th_{low} \geq M(x, y) < Th_{high}$) the pixel is marked as a *weak* edge pixel. After that, a *weak* edge pixel becomes a *strong* edge pixel only if it is connected to a *strong* edge pixel. Finally, only *strong* edge pixels are considered as edge pixels of the underlying image. Figure 2.6 shows examples of the result produced by the Canny operator when it is applied over the image appearing in Figure 2.5(a). The result shown in Figure 2.6(a) is obtained by the Canny operator using a Gaussian filter with $\sigma = 1$, $Th_{low} = 0.05$ and $Th_{high} = 0.125$ and the result of Figure2.6(b) is computed with $\sigma = 1.5$ and the values of Th_{low} and Th_{high} are the same as before.

2.2.3 Morphological Operators

The most common binary operations[1] are called *morphological operations*, since they change the shape of the underlying binary objects. To carry out this task, a binary *structuring*

[1]A binary operation is applied over a binary image.

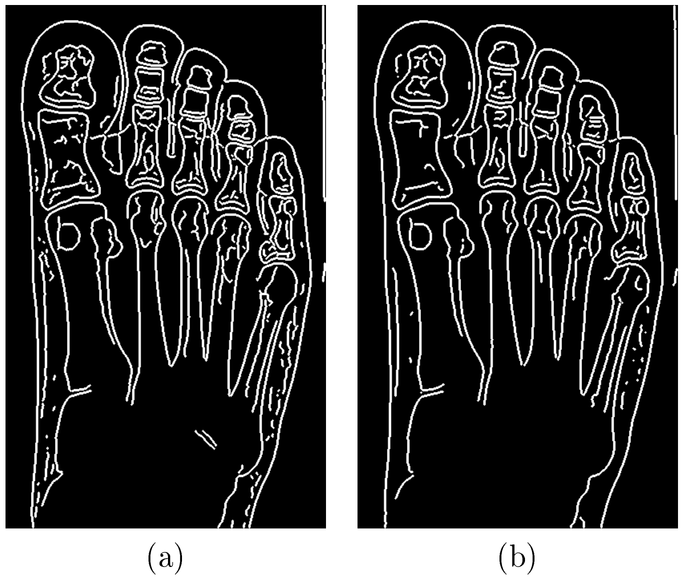


Figure 2.6: Example of the performance of the Canny operator. The result on the left is computed with $\sigma = 1$ and the result on the right is obtained with $\sigma = 1.5$. Both results are obtained using $Th_{low} = 0.05$ and $Th_{high} = 0.125$.

element is required. The process works as the convolution process, calculating a binary output value which depends on a specified threshold. The structuring element can be any shape, from a simple 3×3 mask to more complicated shapes like disk-shape structures. Some examples of *structuring elements* are shown in Figure 2.7.

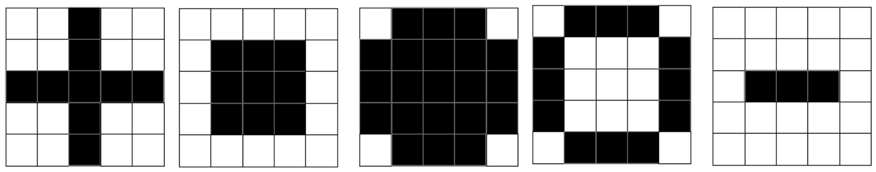


Figure 2.7: Examples of structuring elements.

To formalize how these operations work we follow the explanation given by Szeliski [94] because it is short and very clear. In this way, we define the binary operator $\oslash$ that operates on an image f using a structuring element s, similar to that of the convolution operator. We define the result of this pseudo convolution process as follows:

$$c = f \oslash s \tag{2.15}$$

where, c is the count of the number of 1's that fall inside the structuring element s as it is scanned over f. In addition, we define S as the size of the structuring element (number of pixels) and $\theta(\cdot, \cdot)$ to be a thresholding function defined as:

$$\theta(x,t) = \begin{cases} 1 & \text{if } x \geq t \\ 0 & otherwise \end{cases} \tag{2.16}$$

The standard operations used in binary morphology include:

1. **Dilation**: $\text{di}late(f, s) = \theta(c, 1)$
2. **Erosion**: $erode(f, s) = \theta(c, S)$
3. **Majority**: $maj(f, s) = \theta(c, S/2)$
4. **Opening**: $open(f, s) = \text{di}late(erode(f, s), s)$
5. **Closing**: $close(f, s) = erode(\text{di}late(f, s), s)$

Examples of the result yielded by the dilation and erosion operations using a 3×3 square mask are shown in Figure2.8.

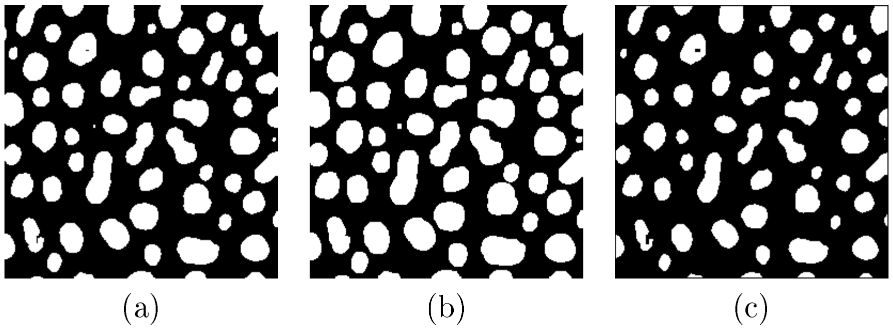


Figure 2.8: (a) A binary image of cells. (b) Resulting image after a dilation operation. (c) Resulting image after a erosion operation. Both results are obtained using a 3×3 square mask.

2.2.4 Thinning

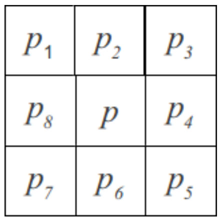
Thinning is a fundamental preprocessing step in many image processing and pattern recognition applications. When the fundamental primitives in an image are strokes or curves of varying thickness it is usually desirable to reduce them to thinned representations located along the approximate middle of the original stroke or curve.

There are many algorithms for obtaining a thinned representation of a binary image. The survey of Lam et al. [55] is a valuable source to read about it. Here, we will describe the two-subiteration based thinning algorithm proposed by Guo and Hall [41] because of its simplicity, efficiency, and its proven good performance.

The algorithm proposed by Guo and Hall processes a binary image evaluating each foreground pixel (with value 1) in two different stages. The first one takes place during odd iterations and the second one during even iterations. The algorithm uses operators with a 3×3 support where the pixel p lies on the center and its eight neighbors are named $p_1, p_2, \ldots, p_8$ as defined in Figure 2.9.

Additionally, two functions are defined over the point p that is being processed. These functions are:

- $C(p)$: The number of distinct 8-connected components of ones in the neighborhood



p_1	p_2	p_3
p_8	p	p_4
p_7	p_6	p_5

Figure 2.9: Support for the thinning operator.

around p. $C(p) = 1$ means that p is a boundary pixel.

$$C(p) = \neg p_2 \wedge (p_3 \vee p_4) + \neg p_4 \wedge (p_5 \vee p_6) \tag{2.17}$$

$$+\neg p_6 \wedge (p_7 \vee p_8) + \neg p_8 \wedge (p_1 \vee p_2) \tag{2.18}$$

- $N(p)$: This function provides an endpoint check. $N(p)$ is the number of pairs which contain one or two ones around p without overlapping.

$$N(p) = min(N_1(p), N_2(p)) \tag{2.19}$$

where,

$$N_1(p) = (p_1 \vee p_2) + (p_3 \vee p_4) + (p_5 \vee p_6) + (p_7 \vee p_8) \tag{2.20}$$

$$N_2(p) = (p_2 \vee p_3) + (p_4 \vee p_5) + (p_6 \vee p_7) + (p_8 \vee p_1) \tag{2.21}$$

To decide if a pixel p is deleted, three conditions must exist. We call these conditions the *Guo-Hall conditions*. These are:

1. $C(p) = 1$
2. $2 \leq N(p) \leq 3$ and
3. Apply one of the following conditions depending whether the underlying iteration is odd or even.
   (a) $(p_2 \vee p_3 \vee \neg p_5) \wedge p_4 = 0$ in odd iterations
   (b) $(p_6 \vee p_7 \vee \neg p_1) \wedge p_8 = 0$ in even iterations

The algorithm works in a parallel way, so all modifications occur after an iteration is completed. An example of using the described thinning operator is depicted in Figure2.10.

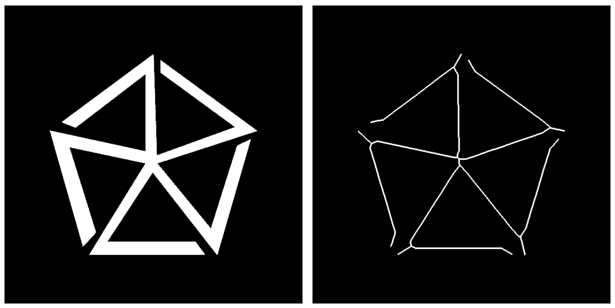


Figure 2.10: Examples of the thinning operation.

2.3 Image Retrieval

Nowadays, research in image retrieval (IR) is an active discipline in computer science. Image retrieval (IR) is focused on the study of techniques for finding images in large collections which are spread over a wide variety of media such as DVDs or the web. Moreover, image retrieval may be extendend to work on video retrieval since a video is in fact a sequence of images [38, 46].

Image retrieval involves topics of multiple disciplines such as computer vision, artificial intelligence, statistics, databases, high performance computing, and human-computer interaction. In this section, we will describe some fundamental on image retrieval including traditional techniques which are based on color, texture or shape.

2.3.1 Classification

There are several ways in which we could classify an image retrieval system. However, Datta et al. presented a very comprehensible description of image retrieval classification [25] and which is the base for the classification described in this section. In this vein, an image retrieval system may be classified from two perspectives. The first one is set from the user perspective, taking into account the modality of querying and classifying the type of the input. The second one is set from the system perspective, classifying the modality of processing.

2.3.2 Query modality

- **Keyword:** In this case, the user writes the query using concepts about what he or she is looking for. Therefore, this kind of query is in fact a textual query. This is currently the most common way under which the search engines are implemented.
- **Image:** Here, the user wishes to find images resembling a query image. To search in this way, the user requires a regular image representing what he or she is actually looking for.
- **Graphics:** This kind of query is created by hand or by a computer program. Therefore, a query falling in this category may be regarded as an artificial picture. In addition, this kind of query may be based on a colorful graphic or a simple stroke-based drawing.

2.3.3 Process modality

Next, we describe a characterization of how image retrieval system algorithms are implemented.

- **Text-Based:** In this situation, a search process is carried out by comparing textual information registered on the images. The problem is that the majority of images come without such information. In this case, an annotation process is required for tagging images automatically. However, it is quite rare for complete text annotation to be reliable and available because it would entail describing every color, texture, shape, and objects appearing in the visual media. Commonly, text-based processing may involve some techniques for natural processing to understand the query as a whole.

- **Content-Based:** The content-based search is the focus of current research on content based image retrieval. In this case, a content-based approach requires extracting visual information from images. The extracted information is then used to evaluate similarity between images. Commonly, this approach also involves image processing and computer vision techniques. An appropriate feature representation and a similarity measure to rank images, given a query, is essential here.
- **Composite:** Composite processing is a combination of techniques falling in the two aforementioned categories.

2.3.4 Architecture of an Image Retrieval System

An image retrieval system basically consists of three components: preprocessing, feature extraction, and similarity measurement. The results of the search are usually shown as a list which is sorted according to a score given by a selected similarity measure.

- **Preprocessing:** This component uses image processing techniques in order to get a simpler representation of the input image that could be analyzed in an easier way by the subsequent steps. Some techniques applied here are noise elimination, edge detection, image enhancement, thresholding, morphological operations among others.
- **Feature Extraction:** This component is very critical in order to select appropriate features for the image retrieval task. These features are typically low level features related to color, texture and/or shape. The set of features must possess a good or an optimal discriminatory power in the particular application.
- **Similarity Measurement:** The similarity measure is used to compute a distance value between the user query and a candidate match from the database. A good similarity measure is characterized by showing a low distance value when the underlying compared images are very similar and vice-versa, it must show a high distance value when the underlying images are very different.

2.3.5 Applications

Image retrieval has very important applications that are beyond the traditional application based on searching the web. Some of these are described below:

- **Architecture and Interior Design:** Describing buildings by text may become a difficult task owing mainly to the absence of a standard terminology that describes correctly the underlying architectural aspects. Therefore, content-based image retrieval appears as an valuable alternative that allow interested users to find images containing particular structures. Moreover, in interior design, a content based image retrieval system may assist users in finding paintings that math with a room with certain features [51, 86].

- **Biochemical:** "Each day more molecules are found and cataloged by biochemical researchers"[65]. Therefore, computational methods that permit searching molecules with particular shape properties would be helpful in the design of drugs [84]. For example, to study the effects of a certain drug researches may want to look for molecules sharing similar shapes. This is supported by the principle of Johnson and Maggiora, which states "similar compounds have similar properties".
- **Digital Catalog Shopping:** Commonly, people spend a lot of time looking for something that covers him or her requirements when they go shopping. In this case an image retrieval system may assist shopper in finding the correct object (sofa, television, car, etc) by a relevant feedback strategy starting with an image that represents a vague idea of what the user is actually looking for [42].
- **Education:** Education is a potential environment where an image retrieval system could become a very important learning tool. A specific application is in a history course, where students are interested in exploring history events including medieval and modern events. For instance, if a class session is focused on the World War II, an image retrieval system would allow student to retrieve relevant pictures or videos about this specific event [76].
- **Film and Video Archives:** An extension of image retrieval can be applied to the case when what we want to process is a sequence of images like videos or films. A particular instance where a video retrieval [46, 38] would impact enormously is in the case of television stations. Until recently, the only option available for finding a particular video shot was to manually fast-forward through the video tapes until the video segment was found. A video retrieval would permit to find a certain video shot faster searching in a large collections of videos using attributes as color, shape or texture. "The ability to find video shots quickly is particularly important to news stations because they often only have minutes to put together the late-breaking news story" [65].
- **Medicine:** Typically, medical diagnosis is carried out by using visual information of abnormal conditions [5] which are either directly viewed by the physician or scanned into an image using techniques as X-rays, magnetic resonance, or computed tomography. Using a large collection of examples of both normal and abnormal states of organs or tissues, an image retrieval system could support the medical staff decision in order to provide a more accurate diagnosis.

2.3.6 Image Retrieval Techniques

In the early years of the image retrieval systems (early 90's), the methods for image retrieval were based on textual information. To this end, images are required to be annotated with appropriate tags. However, text-based annotation has important drawbacks especially when dealing with huge databases. Automatic annotation does not provide a high level of semantic as the manual annotation does. However, manual annotation is actually labor intensive. Moreover, people may describe the same image in different ways. Furthermore, since images are rich in content, text may, in many applications, not be descriptive enough. To overcome the problem arising from the text-based image retrieval, the current direction in this context is on the content based image retrieval (CBIR) which describes images by their own visual information like color, texture or shape. Therefore, in the next paragraphs, we will describe some classical methods to address the content-based image retrieval problem.

Color-based Image Retrieval Methods

Color is the visual attribute that a user can perceive directly from images. It is probably the easiest image attribute users can use to judge similarity between two images. Indeed, color is one of the most widely used visual features in image and video retrieval systems [63, 21, 103].

In order to describe an image using color, first we need to specify a color system. Color systems have been developed for different purposes. Some of the most used color systems are: RGB, HSV, YCbCr, $L^\star a^\star b^\star$, $L^\star u^\star v^\star$, HMMD, among others.

RGB is the most common color space in computer based processing. This system represents a color using the channels red (R), green (G), and blue (B). In this way, this space is defined as the unit cube in the Cartesian coordinate system. A RGB color system is not perceptually uniform and it is dependent on the image conditions. This situation may cause problems when the images are recorded under different lighting conditions for instance.

HSV [88] color space is another popular color system. Here, the colors are represented by three properties: hue (H), saturation (S) and value or intensity (V). The *hue* attribute corresponds with the dominant wavelength of the spectral energy distribution, specifying a kind of color like: red, yellow, green, cyan, blue or purple. The *saturation* value points out how pure a color is. In other words, saturation is the richness of a hue denoting how pure a hue is. Finally, the *intensity* value corresponds to the level of darkness or brightness of the color. This value is related to the gray-scale value. HSV is an intuitive color, and it can be obtained from the a RGB representation by a non-linear transformation.

YCbCr is a way of encoding RGB information used principally for video transmission. In the YCbCr color system, Y represents the luminance component and Cb, Cr represent the chromatic components related to blue and red differences, respectively.

$L^\star a^\star b^\star$ is a visually uniform color system proposed by CIE (Commission Internationale d'Eclairage). The color feature $L^\star$ corresponds to the perceived luminance, the color feature $a^\star$ correlates with the red-green content, and the color feature $b^\star$ correlates with the yellow-blue content.

HMMD is a color space used by the standard MPEG 7 [63], and consists of the color features Hue, Max=max(R,G,B), Min=min(R,G,B), and Diff =Max-Min. In addition, a fifth feature Sum=0.5*(Max+Min) is defined. Although HMMD has five features, a set of three components $\{H, Max, Min\}$ or $\{H, Diff, Sum\}$ is sufficient to form the HMMD color space and specify a color point.

Following we describe the color based algorithms for image retrieval proposed by MPEG-7

[21, 87, 63].

1. **Dominant Color Descriptor (DCC)**: This descriptor provides a compact description of representative colors in an image. The DCC descriptor works determining a set of clusters, in which each cluster centroid c_{i} is regarded as a dominant color. RGB is used as the default color space. For clustering the Generalized Algorithm of Lloyd [39] is used. In addition, two statistical properties are computed for each cluster c_{i}.
   - The percentage p_{i} of pixels belonging to c_{i}.
   - The variation v_{i} of cluster c_{i}.

   Additionally, a spatial coherency s of all dominant colors is computed. The spatial coherency is based on the average connectivity of the pixels within their respective clusters.

   In this way, the dominant color descriptor of an image I is represented as follows:

   $$DCD(I) = \{\{c_{\text{i}}, p_{\text{i}}, v\text{i}\}, s\}, (\text{i} = 1, 2, \ldots, N) \tag{2.22}$$

   where N is the number of dominant colors.
2. **Scalable Color Descriptor (SCD)**: This descriptor is obtained by using the Haar transform across values of a color histogram represented in the HSV color system. This method computes a 256-bin histogram on the HSV space, which is quantized by using 16 bins for the hue (H) component, 4 bins for the the saturation (S) component, and 4 bins for the intensity (V) component. This histogram is passed through a series of 1-D Haar Transform to reduce the number of bins of the histogram. In addition the resulting Haar coefficients may be quantized with a certain number of bits allowing a higher level of interoperability. Finally, the matching between SCD descriptors is carried out by the L_1 metric applied over the Haar representations.
3. **Color Layout Descriptor (CLD)**: This is a descriptor based on spatial distribution of pixels on an image represented in the L*a*b* color space. The CLD descriptor consists of four stages. The first stage partitions the image in 8×8 blocks to deal with resolution or scale variation. In this case, the size of a block depends on the size of the image. In the second stage a representative color for each block is determined. The average of colors on a block is recommended. This results in a reduced image of 8×8 pixels. For each channel of the reduced image a DCT transform [72] is computed, so we have DCT_L, DCT_a, and DCT_b. The final descriptor is composed of three arrays, each listing the coefficients of each DCT representation scanning the corresponding coefficients in a zig-zag way.
4. **Color Structure Descriptor (CSD)**: The CSD represents an image by both the color distribution of the image and the spatial structure of the color. This descriptor is a way to face the problem of having two images with the same color distribution but with completely different color spatial distribution. An example of this problem is shown in Figure 2.11 in which the images are represented by black and white pixels.

The CSD is a one-dimensional array defined as follows:

$$CSD = h_s(m), m \in \{1, \ldots, M\} \tag{2.23}$$

where M, the number of bins, may be any value in $\{256, 128, 64, 32\}$ and s is an associated structuring element that passes through all pixels on the image.

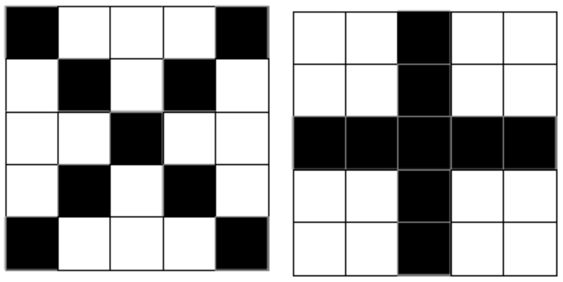


Figure 2.11: Two images having the same color distribution but different color spatial distribution.

The structuring element allows the descriptor to represent the color spatial distribution. The CSD works by increasing a histogram bin $h_s(m)$ by 1 if the color m occurs in the set of pixels within the structuring element s each time s visits a pixel on the image. MPEG-7 defines the structuring element as an 8×8 square.

Texture-based Image Retrieval Methods

Texture is another property of images that characterizes the surface of objects and it also may be used for image browsing. Although many authors have proposed diverse methods for texture based image retrieval [62, 78], there is not an appropriate definition for *texture*. However, authors coincide in two points: (1) within a texture there is a significant variation in intensity levels between nearby pixels and (2) texture is an homogeneous property at some spatial scale, larger than the image resolution. Therefore, a single scene may contain different textures at varying scales [65].

At the core of the MPEG-7 [63], there are three texture descriptors for image retrieval which are discussed below:

1. **Texture Browsing Descriptor (TBD)**: This descriptor is a compact representation of the image texture. This is composed of three perceptual texture attributes: directionality, regularity, and coarseness. The regularity of a texture describes how well defined the texture pattern is. So, this attribute varies from 0 to 3, where 0 indicates an irregular or random pattern and 3 indicates a well defined pattern. The directionality of a texture is quantized to be in $\{0°, 30°, 60°, 90°, 120°, 150°\}$. Patterns do not necessarily have only one dominant direction, the TBD considers two dominant directions (e.g a brick wall has two orthogonal directions). Coarseness is related to image scale or resolution. It may represent pattern varying from the finest to the coarsest pattern resolution. This attribute is represented using four integer values from 0 to 3. The value 0 indicates a fine grain texture and 3 indicates a coarse texture.

   The computation of the attributes values of the TBD is carried out by a bank of Gabor filters. A detailed description of this algorithm can be found in [63].
2. **Homogeneous Texture Descriptor(HTD)**: This descriptor is computed over the frequency domain of an image. We can represent an image in its frequency domain using the Fourier Transform [40, 72]. The frequency space is divided into 30 channels

with equal division in the angular direction and octave division in the radial direction. The angular direction is partitioned in 6 equal-size regions in steps of 30° and the radial direction is divided in 5 regions.

The individual feature channels are modeled using 2D Gabor functions as explained in [78]. For each filtered channel $C_{\mathrm{i}}, \mathrm{i} = 1 \dots 30$ the texture energy and the energy deviation is computed. Both values are logarithmically scaled to obtain two values $\mathrm{e_i}$ and $\mathrm{d_i}$. These values as well as the mean of the pixel intensities f_m and the standard deviation of the entire image pixels $f_{s\mathrm{d}}$ form the texture descriptor. Finally, the homogeneous texture descriptor has the following form:

$$HTD = [f_m, f_s\mathrm{d}, \mathrm{e}_1, \dots, \mathrm{e}_{30}, \mathrm{d}_1 \dots, \mathrm{d}_{30}] \tag{2.24}$$

3. **Edge Histogram Descriptor (EHD)**: The EHD is an edge based descriptor. This approach captures the local spatial distribution of edges. The term local arises from the fact that the EHD divides the image in 4×4 blocks and computes a local spatial distribution (histogram) of edges for each block. The spatial edge distribution is represented by a 5-bin histogram, where each bin represents a type of edge. Edges are typified using five categories: vertical, horizontal, 45° diagonal, 135° diagonal, and isotropic. The final descriptor is obtained concatenating the local histograms. Since the entire image is partitioned into 16 blocks, the final descriptor is composed of 80 bins.

Shape-based Image Retrieval Methods

Object shape provides a powerful clue to the object identity and functionality and can even be used for object recognition tasks [65]. The survey of Schomaker et al. [85] about cognition aspects of image retrieval showed that users are more interested in retrieval by shape than by color and texture as pointed out by Veltkamp et al. [97].

Image retrieval by shape is still considered one of the most challenging tasks for content based search. Even, representative search systems as Query by Image Content (QBIC) [35] of IBM performs poorly when searching on shape.

MPEG -7 standard defines three descriptors [10] with different properties: *the contour-based shape*, *the region-based shape* and *the 3D shape spectrum* descriptors. Since we are interested in 2D shape descriptors, the following will describe the two former descriptors.

1. **Region Based Shape Descriptor (RBSD)** : The importance of the region based shape descriptors is that they may perform well for complex shapes that consist of several disjoint regions, such as trademarks or logos, emblems, clipart, and characters. Further, these kinds of descriptors may solve certain problems that contour based descriptors undergo. Typical problems with contour based techniques arises due to distortions such as crack-like opening or objects touching neighboring objects that may drastically change the contour.

   The RBSD is based on the Angular Radial Transform (ART), that is an orthonormal unitary transform defined on a unit disk that consists of orthonormal sinusoidal basis functions in polar coordinates. The ART coefficients are defined as follows:

$$\psi_{mn} = \int_0^{2\pi} \int_0^1 V^*_{mn}(\rho,\theta) f(\rho,\theta) \rho \mathrm{d}\rho \mathrm{d}\theta \tag{2.25}$$

where $f(\rho,\theta)$ corresponds to the image in polar coordinates and V^*_{mn} is the ART basis function that is separable along the angular and radial redirections and it is defined by:

$$V_{mn}(\rho,\theta) = A_n(\theta) R_m(\rho), \tag{2.26}$$

where $A_n(\theta)$ is the angular basis function and $R_m(\rho)$ is the radial basis function. In order to cope with rotation invariance, the angular basis function is computed using an exponential function:

$$A_n(\theta) = \frac{1}{2\pi} \exp(\mathrm{i} \cdot n \cdot \theta) \tag{2.27}$$

and the radial basis function is defined by a cosine function as follows:

$$R_m(\rho) = \begin{cases} 1 & m = 0 \\ 2cos(\pi m \rho) & m \neq 0 \end{cases} \tag{2.28}$$

Finally, the MPEG-7 region based descriptor is defined as the normalized magnitudes of the ART coefficients. For scale normalization, ART coefficients are divided by the magnitude of the ART coefficient of order $m = 0, n = 0$. Furthermore, the standard MPEG-7 recommends using 35 coefficients for the region-based descriptor, where each one is quantized using 4 bits.

2. **Contour Based Shape Descriptor** :

   The contour based descriptor expresses shape properties of the object outline. The MPEG-7 contour based shape descriptor is based on the Curvature Scale Space (CSS) representation of a contour [70, 69]. The CSS method requires having a digital contour C extracted from an image. This contour is parameterized as:

$$C(\mu) = (x(\mu), y(\mu)) \tag{2.29}$$

   where (x, y) are the coordinates of the pixels belonging to the contour. Having a contour C, the CSS decomposes it into convex and concave sections by determining inflection points (zero-crossing points). The curvature of a contour is computed as:

$$k(\mu) = x'(\mu) y''(\mu) - x''(\mu) y'(\mu) \tag{2.30}$$

   This curvature computation is done in a multiresolution way, where a contour is analyzed at various scales, each obtained by a Gaussian smoothing process.

   CSS keeps information of the zero-crossing of curvature function associated with the corresponding scale yielding a CSS image. The CSS image is obtained by plotting all the zero crossing points on a 2D plane, where vertical axis corresponds to the amount of filtering and horizontal axis corresponds to the position on the contour. In this way, zero-crossing points appearing at various scales will show a high amount of filtering. Next, the location of the prominent peaks (x_{css}, y_{css}) on the CSS image are extracted. These peaks are ordered according to the decreasing values of y_{css}. This ordered set of

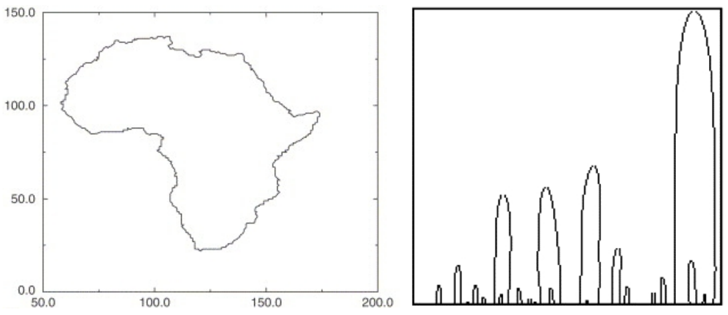


Figure 2.12: A contour of Africa with its corresponding CSS image. This example is courtesy of Mokhtarian et al. [71]

peaks together with the eccentricity and circularity value form the final CSS descriptor. An example of a contour with it corresponding CSS image is depicted in Figure 2.12.

So far we have reviewed shape descriptors proposed in MPEG-7. However, one of the approaches for describing hapes that have received much attention during the last ten years is the proposal of Belongie et al. [9] describing the Shape Context method. We describe this approach following.

Shape Context: This method proposed by Belongie et al. [9] is a contour based approach for determining a similarity value between two shapes. The central idea relies on finding correspondences between these shapes.

As a contour-based approach this method requires a contour representation as input. However, unlike ART descriptor, this does not require the contour to be continuous, and edge map representation of the image computed by Canny operator is enough.

Shape Context treats an image as a set of points. Moreover, a shape is represented by a subset of random points sampled from the contours on the image. In this way, a shape may be represented by a set $P = p_1, p_2, \ldots p_n$ (typically $n = 100$ is used).

For each $p_{\mathrm{i}} \in P$ this method computes a histogram h_{i} of the relative coordinates of the remaining n-1 points under the following expression:

$$h_{\mathrm{i}}(k) = |\{q \neq p_{\mathrm{i}} : (q - p_{\mathrm{i}}) \in bin(k)\}| \tag{2.31}$$

To favor nearby pixels, the bins are distributed using a log-polar space around the underlying point p_{i}. An example of this technique is shown in Figure 2.13.

For setting a correspondence between sets of points, a cost function must be defined. In this way, the shape context method uses the χ^2 test statistics to determine the cost of matching $C(p_{\mathrm{i}}, q_j)$ between to descriptors p_{i} and q_j. Using this function, the method looks for minimizing the total matching cost expressed as:

$$H(\pi) = \Sigma_{\mathrm{i}=1}^{n} C(p_{\mathrm{i}}, q_{\pi(\mathrm{i})}) \tag{2.32}$$

For sake of simplicity we will suppose that two shapes are represented with the same number of points. Let $P = \{p_1, \ldots p_n\}$ and $Q = \{q_1, \ldots, q_n\}$ be such representations. In the previous equation π is a permutation of the sequence $1, \ldots n$ turning the matching to be a one to one representation.

In order to solve the matching problem, shape context uses the Hungarian Method used for Bipartite Graph problems. In addition, the alignment process is carried out

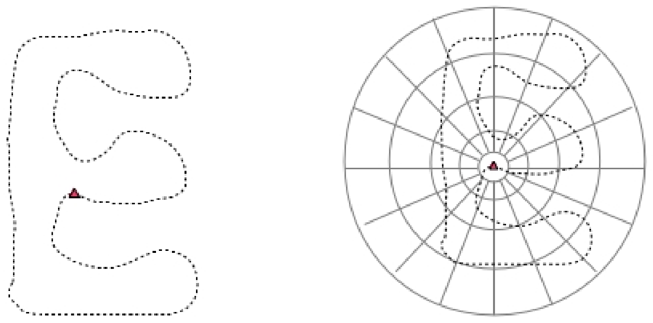


Figure 2.13: On the left, a shape is presented with its sample edge point. On the right, the diagram of the log-polar bins with respect to the sample point (marked with a red triangle) used by the *shape context* descriptor is depicted.

by the Thin Plane Spline model [11]. Finally, the dissimilarity between two shapes is represented by the sum of the matching cost.

For further description on image shape descriptors we recommended going to surveys presented by Loncaric [58] and Veltkamp [97].

2.4 Local Descriptors

Local descriptors have been spread vastly in the computer vision community with a variety of applications like image retrieval, object recognition, object categorization, image stitching and 3D modeling [94]. Unlike global descriptors, which represent an image using information from the whole image, local descriptors represent an image using interest local regions. This allows methods to handle a variety of image invariance problems like geometric variations (scale, translation, rotation, affinity), occlusions and intra-category variations. In the Figure 2.14, two images representing the same scene are shown. In this case, a global approach commonly fails. The book that appears in the image of Figure 2.14(a) is missing on the image of Figure 2.14(b), so a global representation of the two images may be very different. However, local descriptors perform very well in the mentioned case due to there still being a lot of common regions between the images.

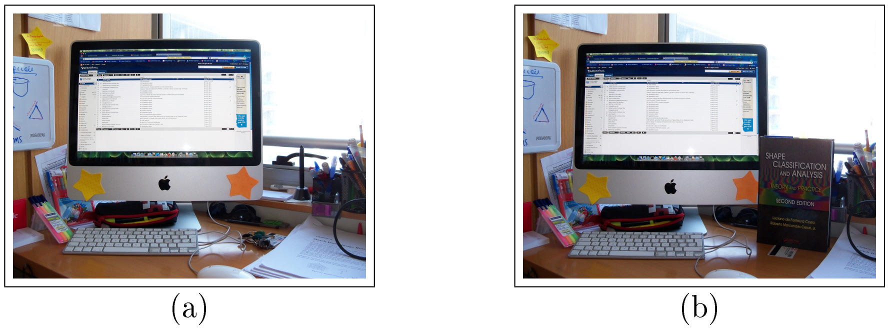


Figure 2.14: Example of two different images representing the same scene.

A general approach for working with local descriptors consists of four steps: (1) find a set of distinctive interest points, (2) define a local region around each interest point, (3) compute a local descriptor from each local region, and (4) match local descriptors.

Finding interest points also known as *keypoints* has been studied broadly and many algorithms have been proposed to this end [67, 96]. A *keypoint* detector method should hold two properties (1) repeatability and (2) informativeness.

- Repeatability: Given two images of the same object or scene, taken under different viewing conditions, a high percentage of the features detected on the scene part in one of the images should be found in the other.
- Informativeness: This feature, also known as the distinctiveness property, sets that the intensity patterns underlying the detected features should show a lot of variation such that features can be distinguished and matched.

Keypoint detectors may be classified as corner-based detectors or blob-based detectors. In the first case, the Harris detector [43] is the most representative one. It is based on evaluating the distribution of derivatives on a image region by means of the second moment matrix. In the second case, the Hessian detector, based on the assessment of second order derivatives, and DoG (Difference of Gaussians) [59] are the most representatives. The Harris and Hessian detectors have been extended by Mikolalzyk et al. [67] to deal with scale variations using the Laplace operator for determining a characteristic scale as a peak of Laplacian response. In addition, an affine invariant detector was proposed by the same authors [66].

Another keypoint detection strategy is to simply sample points on the image. The Shape Context approach [9] is based on this strategy.

After detecting keypoints on an image the next problem is to appropriately describe the region around a keypoint. This task is carried out by local descriptors. Many local descriptors have been proposed, however the current methods are based on the SIFT method [59]. We could consider that the publication of the SIFT method represents a milestone in the local descriptor field.

SIFT (Scale Invariant Feature Transform) is computed from gradients distributed on a 16×16 rectangular region around a keypoint. The rectangular region is divided into 4×4 non-overlaped blocks. For each block, an 8-bin histogram representing the gradient orientation distribution is formed with the corresponding gradient magnitudes falling in the block. In order to reduce the influence of gradients far from the center, the magnitudes are weighted by a Gaussian kernel. The SIFT descriptor is the juxtaposition of the gradient histograms for all the blocks. The final 128-bin histogram, produced by the 16 local 8-bin histograms, is normalized to unit length.

In the same vein of the SIFT descriptor, Ken an Sukthankar [53] proposed the PCA-SIFT, where x and y derivatives are computed over a 39×39 patch. The derivative values make up a 3042-dimensional vector. This vector is reduced to only 36 principal components using PCA.

Another interesting local descriptor is SURF (Speed Up Robust Features) [7], a variant of SIFT that uses integral images and Haar-like features to get a descriptor in a fast way.

Similar as SIFT, SURF divides a rectangular region around a keypoint into 4×4 subregions. For each subregion, the method computes four values using Haar-like features. The final descriptor is a 64-dimensional vector.

The GLOH (Gradient Location-Orientation Histogram) method [68] is another variant of SIFT. This method uses a log-polar binning instead of the 4×4 blocks used by SIFT. This type of partition produces 17 local regions, the center with radius=7 and the other 16 regions formed by partitioning the image into two radial section (radius=11 and 15) and eight angular sections. Among each local region a 16-bin histogram is formed in a similar way as SIFT. The final 272 (17×16)-dimensional descriptor is reduced to 128 principal components.

The final step is to match keypoints coming from two different images using their corresponding local descriptors. To this end, each descriptor from one of the images (generally, the test image) has to be mapped to a descriptor on the other image (generally, the query image). The Nearest Neighbor (NN) strategy [108] is commonly used for mapping descriptors.

In the NN approach, a cost function or distance function is required to map two descriptors. In addition, a threshold is used to determine the goodness of the mapping. There are a variety of alternatives from which we could choose a distance function. The Minkowski distance is typically the option when descriptors are represented as vectors. The Minkowski distance $D_M(\mathbf{x}, \mathbf{y})$ between d-dimensional descriptors $\mathbf{x}$ and $\mathbf{y}$ is defined as follows:

$$D_M(\mathbf{x}, \mathbf{y}) = \left(\sum_{\mathrm{i}=}^{\mathrm{d}} |x_{\mathrm{i}} - y_{\mathrm{i}}|^{\alpha} \right)^{1/\alpha}. \tag{2.33}$$

Typical values of α are 1 or 2. The former is known as the Manhattan distance, while the latter is the Euclidean distance. Further, when $\alpha \to \mathrm{i}nf$ we obtain the Chebyshev distance:

$$lim_{\alpha \to \mathrm{inf}} \left(\sum_{\mathrm{i}=}^{\mathrm{d}} |x_{\mathrm{i}} - y_{\mathrm{i}}|^{\alpha} \right)^{1/\alpha} = max_{\mathrm{i}=1}^{\mathrm{d}} |x_{\mathrm{i}} - y_{\mathrm{i}}|. \tag{2.34}$$

If the histogram is represented by a probability density function (pdf), typical distance functions include the K-L divergence and the $\chi^2 - test$. The K-L divergence comes from the Theory of Information field [22], and it is defined as:

$$D_{KL}(\mathbf{x}, (\mathbf{y}) = \sum_{\mathrm{i}=1}^{\mathrm{d}} x_{\mathrm{i}} log(\frac{x_{\mathrm{i}}}{y_{\mathrm{i}}}). \tag{2.35}$$

The $\chi^2 - test$ is an approximation of the statistical χ^2 function. It is defined as:

$$D_{\chi^2}(\mathbf{x}, \mathbf{y}) = \sum_{\mathrm{i}=1}^{\mathrm{d}} \frac{(x_{\mathrm{i}} - y_{\mathrm{i}})^2}{x_{\mathrm{i}} + y_{\mathrm{i}}}. \tag{2.36}$$

Another approach to set a mapping between local descriptors comes from the Operation Research field. In this case, we try to minimize the total cost of the mapping. Formally, let A be a referent image represented by m local descriptors and let B be a query image represented by n local descriptors, the objective function is defined as:

$$min \sum_{\mathrm{i}=1}^{m} \sum_{j=1}^{n} c_{\mathrm{i}j} x_{\mathrm{i}j} \tag{2.37}$$

where $c_{\mathrm{i}j}$ is the cost of mapping a descriptor i from A with a descriptor j from B, and $x_{\mathrm{i}j}$ is a characteristic function defined as below:

$$x_{\mathrm{i}j} = \begin{cases} 1 & \text{if descriptor i of } A \text{ is mapped with descriptor } j \text{ of } B \\ 0 & \text{otherwise} \end{cases} \tag{2.38}$$

This leads to the assignment problem, which has been studied in depth in the Operation Research field [14, 74]. The assignment problem aims to achieve an optimal matching under a certain cost function. One of the most popular methods to solve this problem is the Hungarian Method [54] which is suitable in cases where the number of descriptors is low (in order of tens).

2.5 Summary

In this chapter we have discussed relevant topics required to understand the subsequent chapters. The discussed topics come from a variety of computer science areas like image processing, computer vision and multimedia information retrieval. For further reading reference regarding image processing, please see the book “Digital Image Processing” of Gonzales and Woods [40] or “The Image Processing Handbook” of Russ [49]. In the case of computer vision, there are also very good reference books like “Computer Vision, Algorithms and Applications” of Szeliski [94] and “Computer Vision, A Modern Approach” of Forsyth and Ponce [36]. Finally, further topics in multimedia information retrieval can be found in "Principles of Visual Information Retrieval" of Lew [65].

Chapter 3

Sketch Based Image Retrieval

3.1 Introduction

Due to the progress in digital imaging technology, image retrieval has become a very relevant discipline in computer science. In a content-based image retrieval system an image is required as input. This image, known as the example image, should express what the user is looking for. However, people commonly use an image retrieval system because they do not count on the desired image, thereby, having such an example image may not be possible, limiting the image retrieval system usability. An easy way to express the user query is using a line-based hand-drawing, a *sketch*, leading to the *sketch-based image retrieval* (SBIR). In fact, a sketch is the natural way to make a query in applications like CAD or 3D model retrieval [37, 32].

Definition 1 A *sketch* is a free hand-drawing consisting of a set of strokes. A sketch lacks texture and color. It is drawn without filling or shadows. The main characteristic of a sketch is that it represents the structure of the objects. Figure 3.1 depicts three examples of sketches.

Definition 2 The *sketch based image retrieval* (SBIR) is part of the image retrieval field. In a SBIR system, the input is a simple sketch representing one or more objects. According

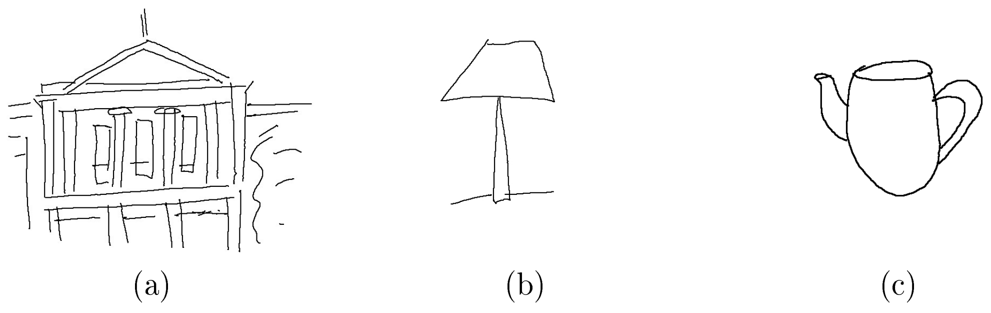


Figure 3.1: Examples of sketches. In this figure, (a) shows a sketch of a government building, (b) shows a sketch of a lamp and (c) depicts a sketch of a teapot.

with the image retrieval classification discussed in Section 2.3.1, a SBIR system falls into the *graphic* category with respect to its input. Due to the underlying differences between a sketch nd a regular image, classical methods applied in the content based image retrieval field cannot be applied directly to the sketch based image retrieval problem.

Although a vast number of researchers on multimedia image retrieval are mainly focused on content-based image retrieval systems using a regular image as query, in the last few years the interest in the sketch based image retrieval problem has been increased. This interest may be owed to the emerging touch screen technology that allow users to draw a query directly on a screen, turning the process of making a query easy and accessible.

The current SBIR methods vary from global approaches to local ones. In the case of global techniques, classical methods are based on building a frequency histogram of orientations [93] or a histogram representing the distribution of edge pixels [18]. In the case of local techniques, approaches based on the Bag of Features strategy are predominant [31, 44, 45, 32].

Another kind of approach is that based on elastic contours [28], where a sketch is represented by a parametric curve that is then strained or bent in order to fit the border of an object in an image. Another approach for the SBIR problem is based on converting the input sketch into a regular image with color an texture [20, 30]. This conversion process is known as image montage. After applying the montage process, the SBIR problem is reduced to the classical CBIR problem in which an example image required as input is the result of the montage process. The montage based image retrieval leads to an expensive process, owing mainly to the additional process to convert the image into a regular image.

In the following, we will describe the most relevant approaches that have been proposed in the last few years to tackle the SBIR problem.

3.2 Query by Visual Example

The Query by Visual Example (QVE) method, proposed by Kato et al.[52], was one of the first approaches dealing with the sketch based image retrieval problem. This method is mainly focused on retrieving art gallery pictures having a rough sketch as a query.

The algorithm consists of two steps:

1. Adaptive image abstraction.
2. Flexible image matching.

3.2.1 Adaptive Image Abstraction

An input sketch and a test image are compared by means of their corresponding edge map representations whose size is regularized to 64×64 pixels. On the one hand, since sketches are already edge map representations, only a thinning operation is applied to them in order to obtain a *linear sketch*. On the other hand, test images are RGB representations, so a specific edge detection algorithm is required to produce an *abstract image*. The edge detection algorithm followed by the QVE approach is based on a differential filter. The final result of this stage is an abstract image that will be the input for the next stage. The algorithm of this stage is described below:

1. Given a pixel $p_{\mathrm{i},j}$, calculate the gradients for RGB intensity values in four directions $\partial^1_{\mathrm{i}j}, \partial^2_{\mathrm{i}j}, \partial^3_{\mathrm{i}j}, \partial^4_{\mathrm{i}j}$.

$$\begin{aligned}
\partial \mathrm{i}j^1 &= \frac{1}{|I_{\mathrm{i}j}|}\frac{1}{3}\left[(p_{\mathrm{i}-1j-1} + p_{\mathrm{i}j-1} + p_{\mathrm{i}+1j-1}) - (p_{\mathrm{i}-1j+1} + p_{\mathrm{i}j+1} + p_{\mathrm{i}+1j+1})\right] \\
\partial \mathrm{i}j^2 &= \frac{1}{|I_{\mathrm{i}j}|}\frac{1}{3}\left[(p_{\mathrm{i}-1j-1} + p_{\mathrm{i}-1j} + p_{\mathrm{i}-1j+1}) - (p_{\mathrm{i}+1j-1} + p_{\mathrm{i}+1j} + p_{\mathrm{i}+1j+1})\right] \\
\partial \mathrm{i}j^3 &= \frac{1}{|I_{\mathrm{i}j}|}\frac{1}{3}\left[(p_{\mathrm{i}-1j-1} + p_{\mathrm{i}-1j} + p_{\mathrm{i}j-1}) - (p_{\mathrm{i}+1j} + p_{\mathrm{i}+1j+1} + p_{\mathrm{i}j+1})\right] \\
\partial \mathrm{i}j^4 &= \frac{1}{|I_{\mathrm{i}j}|}\frac{1}{3}\left[(p_{\mathrm{i}j-1} + p_{\mathrm{i}+1j-1} + p_{\mathrm{i}+1j}) - (p_{\mathrm{i}-1j} + p_{\mathrm{i}-1j+1} + p_{\mathrm{i}j+1})\right]
\end{aligned}$$

   where $|I_{\mathrm{i}j}|$ is the intensity power defined as:

$$|I_{\mathrm{i}j}| = \sqrt{\frac{1}{9}\sum_{u=\mathrm{i}-1}^{\mathrm{i}+1}\sum_{v=j-1}^{j+1} p^2_{uv}} \tag{3.1}$$

2. Get the local maximum gradient:

$$|\partial_{\mathrm{i}j}| = max(\partial^1_{\mathrm{i}j}, \partial^2_{\mathrm{i}j}, \partial^3_{\mathrm{i}j}, \partial^4_{\mathrm{i}j}) \tag{3.2}$$

3. Calculate the mean μ and deviation σ for all gradients previously calculated.
4. Select as global candidates those pixels (i, j) holding $|\partial_{\mathrm{i}j}| \geq \mu + \sigma$.
5. Calculate the local mean $\mu_{\mathrm{i}j}$ and local deviation $\sigma_{\mathrm{i}j}$ for gradient values of global candidates (i, j). The local region of a pixel (i, j) is specified by a 7×7 size local window centered on (i, j).
6. Select the local edge candidates as those pixels (i, j) keeping $|\partial_{\mathrm{i}j}| \geq \mu_{\mathrm{i}j} + \sigma_{\mathrm{i}j}$. The local edge candidates are represented by a binary image.
7. Apply a thinning algorithm to get the final edge image named the *abstract image*.

3.2.2 Flexible Image Matching

The flexible matching is based on evaluating a local correlation between a linear sketch and an abstract image which afterward is aggregated to compute a global correlation value. Given an abstract image P and a linear query Q, the algorithm for the QVE matching proceeds as follows:

1. Divide P and Q into 8×8 local blocks. In this way, $P = \{P^{\mathrm{i}j}\}$ and $Q = \{Q^{\mathrm{i}j}\}$, $\mathrm{i}, j = 0 \cdots 7$.

2. Calculate the local correlation $C^{ab}_{\delta,\varepsilon}$ between P^{ab} and Q^{ab} where shifting (δ,ε) with respect to Q^{ab}.

$$C^{ab}_{\delta,\varepsilon} = \sum_{r=na}^{n(a+1)-1} \sum_{s=na}^{n(a+1)-1} (\alpha p_{rs} \cdot q_{r+\delta s+\varepsilon} + \beta \overline{p_{rs}} \cdot \overline{q_{r+\delta s+\varepsilon}} + \gamma p_{rs} \oplus q_{r+\delta s+\varepsilon}) \tag{3.3}$$

where $n = 8$, $p_{ij} \in P$, $q_{ij} \in Q$, and (α, β, γ) are control parameters used for weighting one-match, zero-match or non-match, respectively. The control parameter values described in QVE [52] are $\alpha = 10$, $\beta = 1$, and $\gamma = -3$.

3. Maximize the local correlation

$$C^{ab} = max_{-\frac{n}{2} \leq \delta,\varepsilon \leq \frac{n}{2}}, (C^{ab}_{\delta\varepsilon}). \tag{3.4}$$

4. Calculate the global correlation C between P and Q

$$C = \sum_{a=0}^{7} \sum_{b=0}^{7} C^{ab}. \tag{3.5}$$

Finally, the global correlation C is used as a similarity score for the retrieving task.

The computational complexity of this method is $(\delta + \varepsilon)O(N)$, where N is the number of pixels in the image and δ y ε are the offset parameters set for the local correlation. The offset parameters are directly related with the property of tackling position variations. Therefore, to handle a wide range of position variations the complexity of the method increase too. In addition, it is not clear how QVE could deal with rotation invariance. Furthermore, the QVE approach requires the sketch be drawn very similar to the object contour which is not always possible because, frequently, people draw sketches in an uncontrolled way.

3.3 Elastic Matching of User Sketches

Another approach for SBIR was presented by Del Bimbo and Pala [28] based on an *Elastic Matching Model*. In that work, they assume that a user draws a sketch as a contour representing the shape of the object aimed to retrieve. Furthermore, the sketch must be drawn with similar aspect ratio as the target object .

In this approach a sketch is regarded as a deformable template modeled by a linear combination of *splines*. The term “deformable” or “elastic” is set because the template is capable of being warped to fit the border of an object on a test image. The match between the final deformable template and an object of an image, as well as the elastic deformation energy spent during the warping process are used to evaluate the similarity between the sketch and the underlying image.

The deformable template parameterized with respect to its normalized arc length is given by:

$$\overrightarrow{\phi}(s) = \overrightarrow{\tau}(s) + \overrightarrow{\theta}(s) \tag{3.6}$$

where $\overrightarrow{\tau} : \mathbf{R} \mapsto \mathbf{R}^2$ is the sketched template and $\overrightarrow{\theta} : \mathbf{R} \mapsto \mathbf{R}^2$ is the deformation undergone by the template.

The warping process is constrained to satisfy two criteria: (1) the template must undergo elastic deformation avoiding discontinuities and (2) the template must be pushed to the edges of the image.

For the first criterion, the deformation undergone by the template is represented by the energy spent with respect to how much the template is strained and how much the template is bent to fit an object. This energy $\mathcal{E}$ is formulated by following:

$$\begin{aligned} \mathcal{E} &= \mathcal{S} + \mathcal{B} \qquad (3.7) \\ &= \alpha \int_0^1 \left[\left(\frac{\mathrm{d}\theta_x}{\mathrm{d}s} \right)^2 + \left(\frac{\mathrm{d}\theta_y}{\mathrm{d}s} \right)^2 \right] \mathrm{d}s + \beta \int_0^1 \left[\left(\frac{\mathrm{d}^2\theta_x}{\mathrm{d}^2 s} \right)^2 + \left(\frac{\mathrm{d}^2\theta_y}{\mathrm{d}^2 s} \right)^2 \right] \mathrm{d}s, \qquad (3.8) \end{aligned}$$

where the term $\mathcal{S}$ represents the straining energy and $\mathcal{B}$ is the bending energy. In the formula, α and β are control parameters that control how much a template may be strained or bent, respectively.

For the second criterion, the edge map representation I_E of an image I is used. To measure how well the deformable template fits the edges, the following formula is applied:

$$\mathcal{M} = \int_0^1 I_E(\overrightarrow{\phi}(s))\mathrm{d}s, \qquad (3.9)$$

where a value $\mathcal{M} = 1$ means that the template lies completely on edges of the image, while $\mathcal{M} = 0$ means that the template lies entirely in areas where the gradient is null (no edges found).

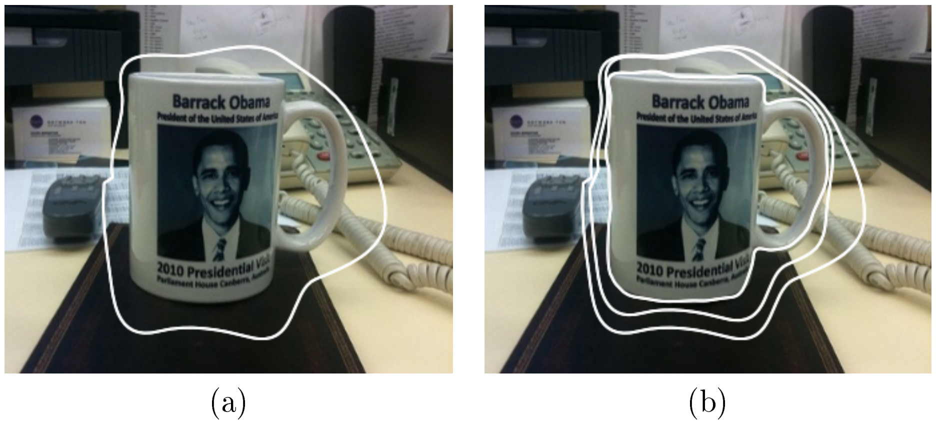
Since the idea is that the template gets closer to the edges spending the least amount of energy as possible, the goal is to maximize $\mathcal{M}$ minimizing $\mathcal{E}$. This is formulated as an optimization problem trying to minimize the functional $\mathcal{F}$ defined as follows:

$$\mathcal{F} = \mathcal{S} + \mathcal{B} - \mathcal{M} \qquad (3.10)$$

To solve the optimization problem Del Bimbo and Pala proposed to use the well known *gradient descend method* [28]. An example of the dynamic of this approach is shown in Figure 3.2.

To determine how well a sketch resembles an object of an image, the energy spent during the warping process represented by $\mathcal{S}$ and $\mathcal{B}$, as well as the $\mathcal{M}$ value are taken into account. Moreover, since the energy spent by the deformable template is related to the complexity of the shape, the number $\mathcal{N}$ of the zeros of the curvature function associated with the sketch is also considered. Finally, a correlation value $\mathcal{C}$ between the original template and the final one obtained by the warping process is considered for the similarity computation.

To get the similarity score between a sketch and a test image, a learning process is carried out. Specifically, a multilayer neural network is trained using the five parameters $(\mathcal{S}, \mathcal{B}, \mathcal{M}, \mathcal{N}, (\mathcal{C})$ previously discussed. The output of the neuronal network is the final similarity score.



Barrack Obama
President of the United States of America
2010 Presidential Visit
Parliament House Canberra, Australia

Figure 3.2: An example of the performance of the *elastic matching* algorithm. (a) shows a test image over which an sketch has been drawn. (b) shows the deformation undergone by the initial curve to fit the mug on the test image.

3.3.1 Spatial Relationship

Assuming that the objects on images are represented by *minimum enclosing rectangles* (MER), Del Bimbo and Pala proposed to consider the spatial relationship between objects on a test image and on a sketch to filter out some test images before applying the elastic matching approach.

Let I be an image containing N objects $o_1, \cdots o_N$, where each object is represented by its MER with the beginning boundaries defined by (b_x, b_y) and the ending boundaries defined by $(\mathrm{e}_x, \mathrm{e}_y)$. The relationship between o_i and o_j is represented by $R_{\mathrm{i}j} = [C_{\mathrm{i}j}, O_{\mathrm{i}j}]$, where $C_{\mathrm{i}j}$ pointing out one of five relationship classes. These classes are *disjoint, meet, contain, inside,* and *partly_overlap.* The term $O_{\mathrm{i}j}$ is a 4-tuple $O_{\mathrm{i}j} = [O^1_{\mathrm{i}j}, O^2_{\mathrm{i}j}, O^3_{\mathrm{i}j}, O^4_{\mathrm{i}j}]$ indicating an orientation vector between o_i and o_j. This is defined as:

$$O^1_{\mathrm{i}j} = \begin{cases} 1 & c_x(o_\mathrm{i}) \leq b_x(o_j) \\ 0 & b_x(o_j) < c_x(o_\mathrm{i}) \end{cases}$$

$$O^2_{\mathrm{i}j} = \begin{cases} 1 & c_x(o_\mathrm{i}) \leq \mathrm{e}_x(o_j) \\ 0 & \mathrm{e}_x(o_j) < c_x(o_\mathrm{i}) \end{cases}$$

$$O^3_{\mathrm{i}j} = \begin{cases} 1 & c_y(o_\mathrm{i}) \leq b_y(o_j) \\ 0 & b_y(o_j) < c_y(o_\mathrm{i}) \end{cases}$$

$$O^4_{\mathrm{i}j} = \begin{cases} 1 & c_y(o_\mathrm{i}) \leq \mathrm{e}_y(o_j) \\ 0 & \mathrm{e}_y(o_j) < c_y(o_\mathrm{i}) \end{cases}$$

where $(c_x(o_\mathrm{i}), c_y(o_\mathrm{i}))$ are the projections of the centroid of o_i on the two axes.

To speed up the search, a *signature file* that represents the spatial relationship between objects is proposed by Del Bimbo and Pala. The signature file is composed of five fields, each one representing one the five classes of relationship defined previously. Each field is composed of n_b bits. A bit of a field λ is turned on depending on a hashing function $H = H(\mathrm{i}, j, O_{\mathrm{i}j}) \mapsto \{0, \cdots n_b - 1\}$ only if the relationship class of objects o_i and o_j corresponds to the λ class. In this way images whose signature files do not match the sketch signature file are filtered out.

In multi-objects images the final similarity score R is obtained summing up similarity scores computed for each object $R = \sum_{\mathrm{i}=1}^{N} S_{\mathrm{i}}$, where N is the number of objects. Retrieved images are sorted depending on the values of R.

The main drawback of this approach is that it requires a continuous curve as a sketch in order for the application of the warping process be possible. This fact is far from the real environments . For instance, the sketches depicted in Figure 3.1 can hardly be represented by a simple curve.

3.4 Edge Histogram Orientation

The *Edge Histogram Descriptor* (EHD) was proposed in the visual part of the MPEG-7 [63] and was improved by Sun Won et al. [93] in the context of image matching. The idea is to use global and semi-global edge histograms generated from the local histograms computed according to the MPEG-7.

This approach can be divided in two steps: (1) compute local edge histograms and (2) aggregate the local histograms to get a global and semi-global histograms. It is worth noting that both the input sketch and the test image used for comparison follow the same process.

3.4.1 Local histograms

The local edge histograms are computed as following:

1. An image (a query sketch or a test image) is divided into 4×4 grid, where each cell of the grid is called a sub-image. An edge histogram is then computed for each sub-image. The edge histogram represents the local distribution of five types of edges in the underlying sub-image. The five types of edges are (1) vertical edge, (2) horizontal edge, (3) 45-degree edge, (4) 135-degree edge, and (5) non-directional edge. A picture of these edges are shown in Figure 3.3.

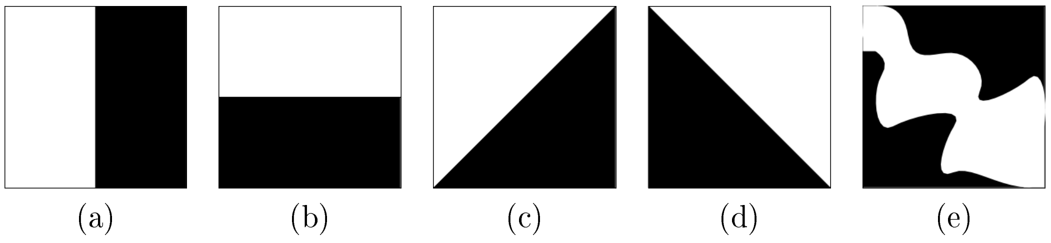


Figure 3.3: The five types of edges in the EHD approach: (a) vertical edge, (b) horizontal edge, (c) 45-degree edge, (d) 135-degree edge and (e) non-directional edge.

2. To determine the occurrence of a certain edge type, each sub-image is further divided into $H \times W$ image blocks, where the H and W depend on the size of the image block,

which in turn, depends on the global image size. Next, we show how to compute the size of the image block (*block_size*) for an image of $m \times n$ pixels.

$$block\_size = \left\lfloor \sqrt{\frac{m \times n}{D}} \right\rfloor \times 2, \tag{3.11}$$

where D is the desired number of blocks, which is fixed to deal with the various image resolutions.

3. Each image block is divided into 2×2 sub-blocks which are numbered from 0 to 3 in a zig-zag way starting from the left-top corner. Additionally, the intensity value of a sub-block b, $I(b)$, is obtained by averaging the intensity values of all pixels falling in b.
4. Compute the response of five directional filters in an image-block to detect a predominant edge direction. These filters are defined as follow:

$$\begin{aligned} f_v &= [1, -1, 1, -1] && (3.12)\\ f_h &= [1, 1, -1, -1] && (3.13)\\ f_{45} &= [\sqrt{2}, 0, 0, -\sqrt{2}] && (3.14)\\ f_{135} &= [0, \sqrt{2}, -\sqrt{2}, 0] && (3.15)\\ f_{nd} &= [2, -2, -2, 2] && (3.16) \end{aligned}$$

The graphical representation of the filters are shown in Figure 3.4.

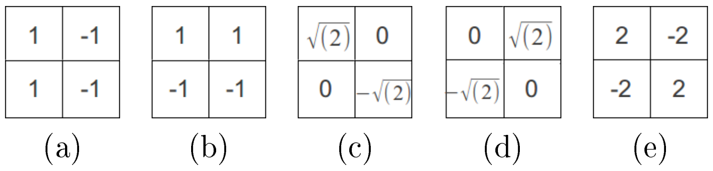


Figure 3.4: Five filters used by the EHD approach: (a) vertical edge filter, (b) horizontal edge filter, (c) 45-degree edge filter, (d) 135-degree edge filter and (e) non-directional edge filter.

5. Compute the magnitude of the response M_τ^B for a filter f_τ over an image block B, where $\tau \in \{v, h, 45, 135, nd\}$

$$M_\tau^B = \sum_{k=0}^{3} |I(b_k^B) \times f_\tau(k)| \tag{3.17}$$

where b_k^B corresponds to the sub-block k in the image block B.

6. The edge type of a block B is that with the maximum filter response.

$$type(B) = arg\ max_{\tau \in \{v,h,45,135,nd\}}(M_\tau^B) \tag{3.18}$$

7. Finally, a 5-bin edge histogram is computed for each sub-image representing the frequency of occurrence of each edge type in the underlying sub-image. In this way we obtain sixteen 5-bin local edge histograms.

3.4.2 Global and semi-global histograms

Sun Won el al. [93] proposed to extend the original EHD descriptor considering not only local histograms, but also a global and semi global histograms. The global edge histogram represents the edge distribution for the whole image. For the semi-global case, the edge histograms are computed in three types of clusters.

The first type of cluster is formed with sub-images falling in the same row yielding four row edge histograms as shown in Figure 3.5(a). Likewise, four column edge histograms are computed by grouping the sub-images by columns (Figure 3.5(b)). This corresponds to the second cluster type. The third type of cluster is formed with nearby sub-images as shown in Figure 3.5(c), yielding five histograms. In this way, the final edge histogram is composed of sixteen local histograms (80 bins), 1 global histogram (5 bins) and thirteen semi-global histograms (65 bins). Thereby, this approach yields a histogram consisting of 150 bins.

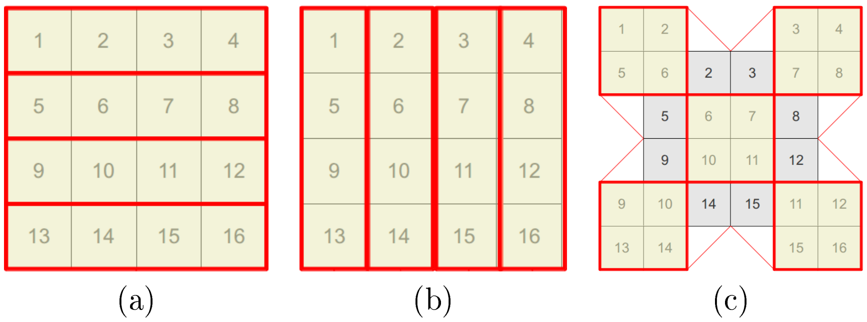


Figure 3.5: Clusters used to compute semi global edge histograms by the EHD approach. (a) clusters formed with sub-images of the same row. (b) clusters formed with sub-images of the same column and (c) clusters formed with nearby sub-images.

In terms of computational complexity, the EHD method is linear $O(N)$ with respect to the number of pixels N in the image. Although, this method is simple to be computed, it does not deal with rotation variations. Furthermore, this method fails when it has to face partial matching.

3.5 Angular Partitioning of Abstract Images

The Angular Partitioning of Abstract Images (APAI) proposed by Chalechale et al. [18] method is another approach for the SBIR problem. This method computes an abstract representation from both the query image and the test image based on the corresponding edge images. Then, the angular distribution of pixels on the abstract image is employed for making up a feature vector. This method addresses the rotation problem by applying the Fourier Transform to the APAI descriptor. The steps followed by this approach are (1) get an abstract image from both the sketch and a test image, (2) apply a feature extraction based on an angular partitioning and (3) apply the Fourier Transform to get rotation invariance. Next, we describe each of these steps in detail.

3.5.1 Abstract Image

The first step is to compute an edge map representation from the query and from a test image. Since the former is almost an edge map, only a thinning operation will be required. However, for the latter, the Canny operator is applied using $\sigma = 1$ and a Gaussian mask of size=9. To compute the high and low threshold required for the Canny hysteresis process, this method proposes the following strategy:

1. Compute an edge filter H based on the smoothed derivative g of the 1D version of the Gaussian filter applied for the Canny operator.
$$H = G \otimes g, \tag{3.19}$$
where G is a 1D Gaussian filter, and $\otimes$ is the convolution operator.
2. Calculate two directional components $X(u,v), Y(u,v)$ for each pixel (u,v) of the image I composed of $M \times N$ pixels:
$$X(u,v) = \sum_{j=1}^{M}(u,j)H(v-j) \tag{3.20}$$
$$Y(u,v) = \sum_{\mathrm{i}=1}^{N}(\mathrm{i},v)H(u-\mathrm{i}) \tag{3.21}$$
3. Compute an edge magnitude $\Gamma(u,v)$ of a pixel (u,v).
$$\Gamma(u,v) = \sqrt{X(u,v)^2 + Y(u,v^2)} \tag{3.22}$$
4. Make up a cumulative histogram h_c of Γ values and pick the smallest index κ of h_c such as $h_c(\kappa) \geq \alpha \times M \times N$ $(\alpha = 0.7)$.
5. The high threshold is set to $\beta \times \kappa$ and the low threshold is set to $0.4 \times \beta \times \kappa$.

After computing the edge map representation, the method obtains the bounding box of the edge image resizing it to a $J \times J$ pixels in order to deal with scale problems. The resulting image is called the abstract image.

3.5.2 Feature Extraction

For the feature vector extraction, an abstract image Ω is divided in angular partitions (slices) with respect to the corresponding surrounding circle with radius R (an example of this partition is depicted in Figure 3.6). In this way, the angle between adjacent slices is $\phi = 2\pi/K$, where K is the number of angular partitions and it will correspond to the size of the feature descriptor as well.

The number of edge points falling in each slice is chosen to represent the slice feature. Thus, the feature vector $f(\mathrm{i}), \mathrm{i} = 1, \cdots, K$ is defined as:
$$f(\mathrm{i}) = \sum_{\phi=\frac{\mathrm{i}2\pi}{K}}^{\frac{(\mathrm{i}+1)2\pi}{K}} \sum_{\rho=0}^{R} \Omega(\rho, \phi), \tag{3.23}$$

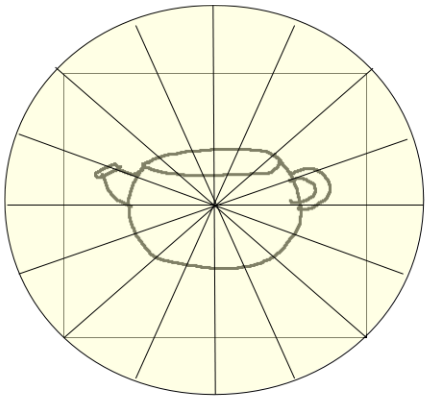


Figure 3.6: Angular partitioning of the APAI method.

which becomes a scale and translation invariant descriptor.

3.5.3 Rotation Invariance

In order to tackle the rotation invariance problem, the magnitude of the Fourier Transform applied to f is used because of its shift-invariant property. The 1D Fourier Transform $F(\cdot)$ of $f(\cdot)$ is obtained as follows:

$$F(u) = \frac{1}{K}\sum_{k=0}^{K-1} f(K)\mathrm{e}^{\frac{-\mathrm{i}2\pi uk}{K}}, \tag{3.24}$$

whose magnitude $||F(u)||$ is defined as follows:

$$||F(u)|| = \sqrt{\frac{1}{K}\left(\sum_{k=0}^{K-1} f(k)cos\left(\frac{2\pi uk}{K}\right)\right)^2 + \left(\sum_{k=0}^{K-1} f(k)sin\left(\frac{2\pi uk}{K}\right)\right)^2} \tag{3.25}$$

Finally, the feature vector is given by $\{||F(u)||\}, u = 0 \cdots K-1$. The similarity between feature vectors is computed by the Manhattan distance.

In the case of the APAI approach, it entails an overload of computing the parameters for the Canny method yielding a computational complexity of $O(N_r \times N_c)(N_r + N_c)$, where N_r and N_c are the number of rows and number of columns of the underlying image, respectively. This method is one of the first methods to deal with rotation issues.

3.6 Structure Tensor Descriptor

The *Structure Tensor Descriptor* (STD) was proposed by Eitz et al. [29] as a novel alternative to deal with large scale image retrieval based on sketched feature lines. This proposal is constructed in such a way that both the query sketch and the test image undergo the same processing to get the descriptor or feature vector.

The STD approach is based on getting a local descriptor that captures the main directions in local regions of both the test image and the query. Local descriptors corresponding to equivalent local regions are compared, and an aggregation function that computes a dissimilarity score based on the local comparison is defined.

Specifically, the local regions are obtained by dividing the image into a number of regular cells, typically between 24×16 and 32×24 cells per image. Let $C_{\mathrm{i}j}$ be a cell, $(u, v) \in C_{\mathrm{i}j}$ if the pixel (u, v) falls within the cell $C_{\mathrm{i}j}$.

Having divided the image into cells, the gradients g_{uv} are computed for each pixel (u, v) discarding some gradients with small magnitude. In particular, the method discards those gradients g where $g^T g < \varepsilon^2$ ($\varepsilon = \sqrt{(}2)/20)$.

The structure tensor provides information about the main orientation of the gradients in a cell. The idea is to find a single vector $\mathbf{x}$ in a cell that is as parallel as possible to the image gradients in that cell. Since $x^T g_{uv}$ attains a maximum when $x || g_{uv}$ then finding such a vector $\mathbf{x}$ could be formalized as the following maximization problem:

$$\mathbf{x} = arg\ max \left(\sum_{(u,v)\in C_{\mathrm{i}j}} (\mathbf{x}^T g_{uv})^2 \right), \tag{3.26}$$

where

$$\sum_{(u,v)\in C_{\mathrm{i}j}} (\mathbf{x}^T g_{uv})^2 = \mathbf{x}^T \left(\sum_{(u,v)\in C_{\mathrm{i}j}} g_{uv} g^T_{u_v} \right) \mathbf{x}. \tag{3.27}$$

Defining $G_{\mathrm{i}j} = \sum_{(u,v)\in C_{\mathrm{i}j}} g_{uv} g^T_{uv}$, we get:

$$\mathbf{x} = argmax \left(\mathbf{x}^T G_{\mathrm{i}j} \mathbf{x} \right). \tag{3.28}$$

Since the optimization problem relies just on $G_{\mathrm{i}j}$ the descriptor $T_{\mathrm{i}j}$ proposed in this approach for a cell $C_{\mathrm{i}j}$ is $G_{\mathrm{i}j}$ normalized by the corresponding Frobenius norm as below:

$$T_{\mathrm{i}j} = \frac{G_{\mathrm{i}j}}{||G_{\mathrm{i}j}||_F} \tag{3.29}$$

For comparing a sketch and a test image, the corresponding structure tensor based descriptors are compared. To this end, a distance $\mathrm{d}_{\mathrm{i}j}$ between two tensor descriptors $T^S_{\mathrm{i}j}$ and $T^I_{\mathrm{i}j}$ corresponding to cell (i, j) of a sketch and a test image, respectively, is defined as:

$$\mathrm{d}_{\mathrm{i}j} = ||T^S_{\mathrm{i}j} - T^I_{\mathrm{i}j}||_F \tag{3.30}$$

The final distance or dissimilarity score d is computed as summing up the distances for each cell as:

$$\mathrm{d} = \sum_{\mathrm{i}=1}^{M} \sum_{j=1}^{j=N} \mathrm{d}_{\mathrm{i}j}, \tag{3.31}$$

where M is the number of cells per row and N is the number of cells per column.

The tensor-based descriptor requires to pass through all pixels of an image to form all the *tensor* matrices $G_{\mathrm{i}j}$, which leads to a computational complexity of $O(N)$, where N is the number of pixels. Similar to the EHD method, the tensor based descriptor does not deal with rotations variations.

3.7 Bag Of Features Approach

Widely used in the context of *text retrieval*, the Bag of Words (BoW) approach has also been applied in the computer vision field successfully, specifically for the *category recognition* problem.

The Bag of Words approach, also known as Bag of Features in the computer vision community, simply computes the distribution or histogram of "visual words" found in the query image and compares this distribution with those found in the test images [94]. Csurka et al. [23] were the first to use the term *bag of keypoints* to describe such approaches and among the first to demonstrate the utility of frequency-based techniques for the category recognition task.

Different from the text retrieval environment where *words* can easily be separated and extracted, in the context of images, we do not have such a behaviour, we have only pixels which do not contain semantic information[94].

A common approach to represent *visual words* is to construct a visual vocabulary known as a *codebook* by clustering local descriptors, given a set of training images. Each cluster center represents a *codeword*, and the final histogram for an image I is computed counting the number of local descriptors of I falling in each cluster (codeword). Typically, the size of the codebook is to the order of 1000.

In the categorization problem, the distribution of visual words are then classified using typical classifiers like Naïve Bayes or SVM. [23]. However, due to SVM has showed an outperforming behaviour over Naïve Bayes, SVM is the classifier commonly used in the categorization problem.

In this vein, Eitz et al. [31] have recently presented modified versions of the well known local descriptors like Shape Context [9] or SIFT [59] to be applied in the context of SBIR, using the BoW approach. The variations they propose are the *Spark Feature* and the *DoIGOH Feature*. In addition, they use 500 descriptors per image in random locations and have

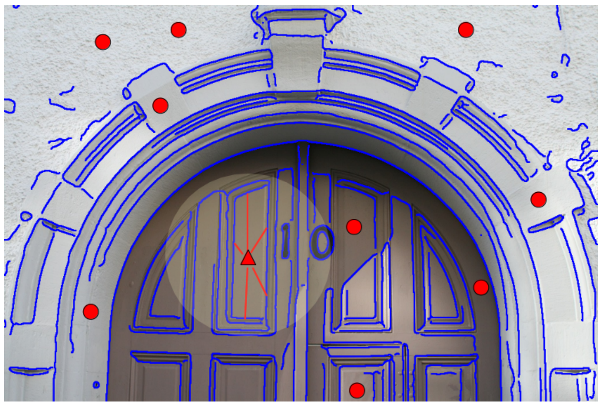


Figure 3.7: Spark Features

reported outperforming results with a codebook of 1000 visual words. Finally, as a search strategy they rely on standard inverted index.

3.7.1 Spark Feature

In the Spark descriptor, the random points are generated to lie in the empty areas between image edges. For each sampled point, the method traces rays of random directions until the first edge point is achieved. The descriptor then represents the distribution of features of the rays that generated a hit. The features of rays that the authors propose are: (1) the distance of the ray and (2) the angle of the ray or the orientation of the hit point. According to the authors, using a 2D histogram storing the distance and the angles information of rays, as in the *shape context* approach, results in a better performance. An example of this descriptor is shown in Figure 3.7.

3.7.2 DoIGOH Feature

This is a variant of the well known SIFT descriptor. The name of this approach stems from *Dominant Image Gradient Orientation Histogram* (DoIGOH). In this case, the locations on the image over which the local descriptors are computed are determined randomly. The variation is that only orientations that correspond to important feature lines are stored. The method uses Canny lines as an indicator for important features and stores only orientations that correspond to data lying under a slightly blurred version of the Canny feature lines. An example of this descriptor is depicted in Figure 3.8

One of the advantages of using the BoW approach is related to the necessary time for searching. Different from the classical local approach, where many descriptors need to be matched with many descriptors, in the BoW approach an image is represented by only one histogram. This means that comparing two images is reduced to compare the two corresponding histograms, allowing to decrease the time for searching in the context of CBIR, for instance. However, this approach losses the locality property of the descriptors, turning

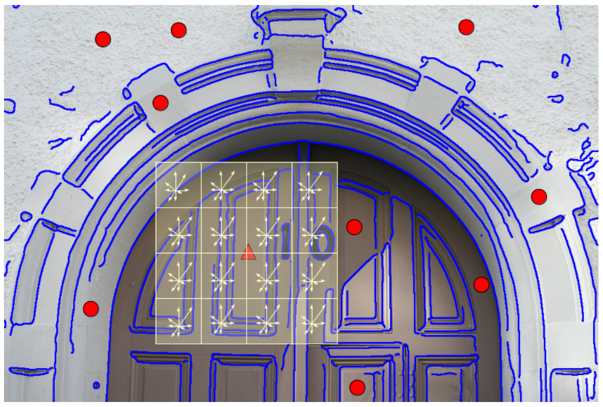


Figure 3.8: DoIGOH Features

impossible detecting the position of an object of interest. In addition, the BoW ignores the spatial relationship among features of the image, which represents a relevant feature, specially in cases with objects are represented in a minimalist way like in the case of sketches that are represented only by strokes.

In terms of computational complexity, the spark feature approach and the DoIGOH approach compute the descriptors of an image in $O(N + P \times W)$, where N is the number of pixels of the image, P is the number of selected points on the image, and W is the number of pixels corresponding to the local region in which a descriptor is computed.

3.7.3 Gradient Field - Histogram of Orientated Gradients

In the *Gradient Field-Histogram of Oriented Gradients* (GF-HOG) approach, that was presented by Hu et al. [44, 45], each test image is represented by and edge map using the Canny operator. The resulting edge map is considered as a sketch representation for the underlying image. The edge maps of the all test images are then compared with the input sketch. Considering that sketch representations are commonly sparse with respect to edge pixels, this method propose an interesting approach consisting in using a dense gradient field over which descriptors are computed. The gradient field is computed from a sketch representation interpolating the corresponding set of edge pixels. This gradient field is also called *gradient image*.

Having a sketch representation $M : \Omega \mapsto \{0, 1\}$, where $\Omega \in \mathbf{R}^2$, a gradient image $G : \Omega \mapsto [-\pi, \cdots \pi]$ is computed over M. G is computed such that $G(x, y) = tan^{-1}(G_x(x, y)/G_y(x, y))$, where G_x is the gradient image computed in the abscissa direction and G_y in the ordinate direction. The gradient field $\mathcal{F}$ of M is computed solving the following minimization problem:

$$\mathcal{F} = argmin \int\int_{\Omega} (\bigtriangledown \mathcal{F} - G)^2 \text{ s.t. } \mathcal{F}|_E = G|_E, \tag{3.32}$$

where $E = (x, y) \in \Omega | M(x, y) = 1$. The process of computing $\mathcal{F}$ by using the Equation 3.32 is known as a *guidance interpolation* where G is a guidance field. The solution of Equation 3.32 is achieved by solving a Poisson's Equation with Dirichlet boundary condition. Perez et

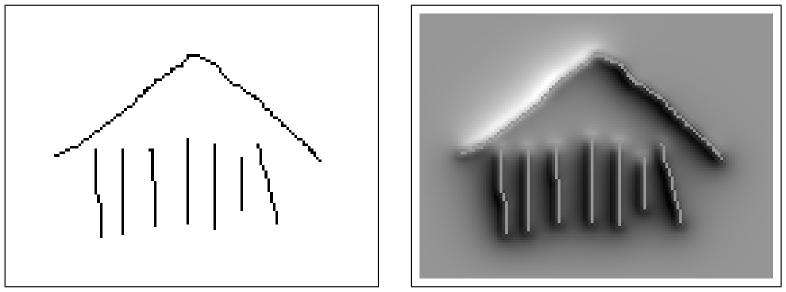

(a) (b)

Figure 3.9: A sketch representation shown in (a) and the correspondent gradient field shown in (b).

al. [77] proposed a discrete solution for the Poisson's equation and it is the base of the GF-HOG approach. The discretization process consists in solving a set of sparse linear equation defined as follow:

$$\triangle \mathcal{F}(x,y) = \sum_{(p,q)\in N_{(x,y)} G(x,y)} , \forall (x,y) \notin E. \tag{3.33}$$

where $N_{(x,y)}$ represents a neighborhood of (x,y). In addition, in order to satisfy the constrain specified in Equation 3.32, $\mathcal{F}$ must be equal to $G(x,y)$ for the points $(x,y) \in E$. Moreover, Hu et al. [44] propose to approximate $\triangle\mathcal{F}$ using a 5×5 window. In Figure 3.9, we show a sketch representation together with its corresponding gradient field image.

Having computed the gradient field image, the next step is the characterization process. To this end, Hu et al. use the Histogram of Gradient Orientations approach (HOG) [24]. Specifically, a HOG descriptor is computed over each pixel (x,y) on $\mathcal{F}$ that corresponds to an edge pixel in M ($M(x,y) = 1$). To compute a HOG descriptor with respect to a pixel (x,y), the method requires to define a support local region around (x,y). Having defined the local region $R_{(x,y)}$ around (x,y) the HOG descriptor in the GF-HOG method works as follows:

1. Divide the local region R into a 3×3 grid. Each cell of R is called a window.
2. Divide each window W into $n \times n$ grid. Each cell of W is called a block.
3. Compute a histogram of gradient orientation for each block. The orientation of the gradients are quantized into a 9-bin histogram.
4. Compute an aggregate histogram for each window concatenating all the block histograms. We obtain a $n \times n \times 9$-bin histogram.
5. Compute an aggregate histogram for R concatenating all the window histograms. The final result is a $9^2 \times n^2$-bin histogram.

In order to deal with scale variations, Hu et al. propose to use $n = \{5, 10, 15\}$, where each value of n represents a different scale for computing the HOG descriptors. For the retrieval task, the GF-HOG approach is based on the Bag of Feature approach. In this way, the set of local descriptors computed for each test image are clustered by using the k-means algorithm to form a codebook. The input sketch as well as the test images are characterized by mean of a frequency histogram which is computed using the computed codebook. To measure a similarity value between a sketch and a test image the corresponding frequency histograms are compared using a histogram intersection functions. Let H^S be the frequency histogram constructed from an input sketch $S$and H^I be the frequency histogram constructed from a test image I the similarity value $\mathrm{d}(H^S, H^I)$ between S and I is computed by using the Equation 3.34.

$$\mathrm{d}(H^S, H^I) = \sum_{\mathrm{i}=1}^{k}\sum_{j=1}^{k}(\omega_{\mathrm{i}j} min(H^S(\mathrm{i}), H^I(j))) \tag{3.34}$$

where k is the size of the underlying codebook and $w_{\mathrm{i}j}$ is a weight computed with respect to a normalized version $\hat{H}$ of H defined as follows:

$$w_{\mathrm{i}j} = 1 - |\hat{H}^S(\mathrm{i}) - \hat{H}^I(j)|. \tag{3.35}$$

The computational cost of the GF-HOG approach depend strongly on the cost of solving the sparse linear equation system to obtain the gradient image. Considering that a classical sparse linear equation system solver is cubic in terms of its computational complexity, computing the gradient image is the order of $O(N^3)$, where N is the size of the image. Considering besides that the HOG descriptor may be computed in $O(N)$ the computational cost of the GF-HOG approach is $O(N^3 + N)$.

3.8 Edgel Index

This approach, presented by Yang Cao et al. [17], proposes to describe a sketch representation, generated from the input sketch or from a test image, using a triplet (x, y, θ) where (x, y) corresponds to the position of edge pixels and θ is the edge orientation at (x, y). For matching an input sketch and a test image, the approach is based on the Oriented Chamfer Matching (OCM) [12, 92] applied over the triplet representations.

One of the relevant structures used by the Edgel Index approach is the **hit map**. Let T_S be a triplet representation of a sketch S, the **hit map** of T_S is computed as follows:

$$Hit_S(p) = \begin{cases} 1 & \exists q \in T_S, ||(x,y)_p - (x,y)_q||_2 \leq r \wedge \theta_p = \theta_q \\ 0 & otherwise \end{cases}, \tag{3.36}$$

where r is a tolerance radius and $||\dot{|}||$ is the Euclidean distance.

Let T_I be a triplet representation of the test image I, the Edge Index method computes a similarity value between S an I as follows:

$$sim(I, S) = \left[\left(\frac{1}{|T_I|}\sum_{p \in T_I} Hit_S(p)\right) \cdot \left(\frac{1}{|T_S|}\sum_{p \in T_S} Hit_I(p)\right)\right]^{\frac{1}{2}}. \tag{3.37}$$

The proposed method compares two images with a computational cost of $O(N)$, where N is the number of pixels of the biggest image. In addition, for indexing purposes, an inverted index approach is applied with respect to the triplets that characterize the test images.

Although this approach allows a user to obtain the retrieval result fast, it lacks minimal invariance properties. This method does not permit, for instance, to handle position or scale variations. This may be a critical drawback because commonly users are interested in a concept further than in the right position of the objects. For instance, a user looking for pictures containing the sun will draw simply a circle as the input sketch. If the circle is drawn on the left side of the canvas, the Edgel Index approach will never return pictures containing the sun on the right side.

3.9 Summary

In this chapter we have discussed the most relevant methods that address the sketch based image retrieval problem. All of these methods share a common property. They all transform test images into edge map representations in order to compare them with an input sketch. To this end, the discussed methods tend to use the Canny operator. Although there are other state-of-the-art approaches for computing edge maps such as the *Berkeley boundary detector* [64], many authors still prefer Canny due to its processing time.

The SBIR methods permit representing a sketch in a global or local way. On the one hand, global approaches are commonly based on computing a histogram of orientations or on computing a distribution of edge pixels with respect to a certain image partitioning. Methods falling in this group are *QVE*, *APAI* and *ST*. Another global method is the *EHD* that forms a global edge histogram concatenating local ones. These global methods fail when local variations are affecting the images.

On the other hand, local methods extract relevant information from interest points. The interest points may simply be the edge pixels on an image such as the *GF-HOG* approach does, or non edge-pixels randomly selected like in the case of the *Spark* or the *DOIHoG* approach. Furthermore, it has been observed a predominant tendency toward using histograms of orientations to extract local information in the form of feature vectors. In addition, for matching purposes the local feature vectors are commonly clustered using a *Bag of Features* approach which ultimately represents an image by a distribution of local descriptors with respect to a set of clusters.

Chapter 4

Histogram of Edge Local Orientations

4.1 Introduction

Among the Content Based Image Retrieval field, an image may be characterized globally or locally. The former one known as a global approach is carried out by means of global descriptors which have the main advantage of being easy to compute and fast to compare with other descriptors, in some cases, producing effective results. This chapter describes a global approach for the sketch based image retrieval problem based on computing local orientations of sketches.

Orientation information is an important attribute of images and it has been used by many successful techniques in the field of computer vision such as SIFT [59] or HOG [24]. In the case of SBIR, the orientation of strokes is one of the most noticeable features since a sketch lacks color and texture having only strokes as objects. Therefore, a descriptor has to rely on the information provided only by strokes.

Orientation estimation is also an important issue for other tasks. A particular case where orientation estimation turns in a critical stage is in the context of fingerprint processing [61]. In terms of the fingerprint processing field, the task of estimating the orientation of fingerprints is also known as the computation of the *directional field* of the image. The *directional field* (DF) describes the coarse structure, or basic shape, of a fingerprint [8] as shown in Figure 4.1. The directional field is defined as the local orientation of the ridge-valley structure. An accurate directional field estimation is important not only for fingerprint reconstruction but also for a global classification of a fingerprint. The latter is very useful for reducing the number of comparisons during the fingerprint matching stage.

Although many methods for estimating directional field on a fingerprint have been proposed [8], the most successful approach is a gradient based method known as the *square gradient method*. The idea behind this method is to average gradients in a neighborhood. To avoid opposite gradients cancel each other, the angles of the gradients are doubled, and to give more weight to strong orientations, the length of the gradients are squared.

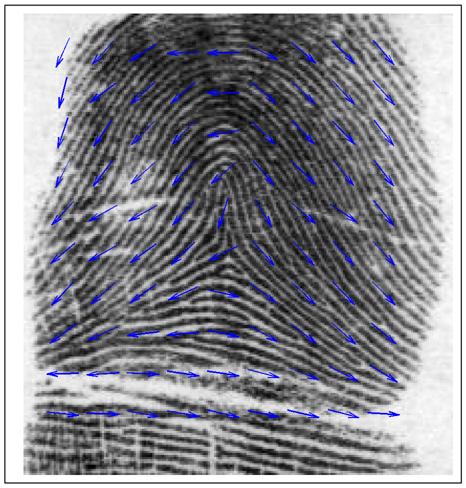


Figure 4.1: Directional Field of a fingerprint.

Inspired by the squared gradient method, we propose a global descriptor based on estimating edge local orientations to form a histogram. Our descriptor is named HELO as initialism of *Histogram of Edge Local Orientations*. Our proposed approach is invariant to scale and translation transformations. In addition, the orientation histogram turns out invariant to lighting conditions, as well. Although orientation invariance is rarely required for an SBIR method, we tackle the orientation problem applying two different normalization processes; one using principal component analysis (PCA) and the other using polar coordinates (PC). Finally, we use a combined distance as a similarity measure.

Moreover, since noise adversely affects the edge orientation estimation [26], its presence in an image may cause descriptors to perform poorly in the image retrieval context. Thus, our proposal attempts to reduce this undesired effect, computing each edge orientation in a local way. Additionally, using a local estimation, the sketches do not need to be drawn with continuous strokes.

4.2 Square Gradient Method

This method estimates the gradient orientation in a local way using a defined neighborhood. Let I be a gray scale image. First of all, the gradient vector for all pixels in I are estimated using a well known approach such as Sobel or Prewitt [40]. Let $[G_x, G_y]^T$ be the corresponding gradient vector for a pixel (x, y) in an image I. For doubling the angle and squaring the length, the gradient vector is represented in its corresponding polar coordinates. This polar representation is given by:

$$\begin{bmatrix} G_\rho \\ G_\phi \end{bmatrix} = \begin{bmatrix} \sqrt{G_x^2 + G_y^2} \\ tan^{-1}\frac{G_y}{G_x} \end{bmatrix} \tag{4.1}$$

The gradient vector is also transformed back to its Cartesian representation from the polar coordinates by the following equation:

$$\begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} G_\rho cos(G_\phi) \\ G_\rho sin(G_\phi) \end{bmatrix} \tag{4.2}$$

After doubling the angle and squaring the length of the gradient, we obtain the corresponding squared gradient $[G_{sx}, G_{sy}]$ as follows:

$$\begin{bmatrix} G_{sx} \\ G_{sy} \end{bmatrix} = \begin{bmatrix} G_\rho^2 cos(2G_\phi) \\ G_\rho^2 sin(2G_\phi) \end{bmatrix} \tag{4.3}$$

Applying trigonometric properties for double angles, we can transform the above equation leaving only terms in the Cartesian representation as follows:

$$\begin{bmatrix} G_{sx} \\ G_{sy} \end{bmatrix} = \begin{bmatrix} G_\rho^2 (cos^2(G_\phi) - sin^2(G_\phi)) \\ G_\rho^2 (2sin(G_\phi)cos(G_\phi)) \end{bmatrix} = \begin{bmatrix} G_x^2 - G_y^2 \\ 2G_xG_y \end{bmatrix} \tag{4.4}$$

Finally, the average squared gradient $[\overline{G_{sx}}, \overline{G_{sy}}]$ is calculated in some neighborhood using a window W.

4.3 Histogram of Edge Local Orientation

Our method works in two stages. The first one performs preprocessing tasks to get an abstract representation from both the sketch and the test image. The abstract representation tries to make the feature extraction stage easier. The second stage makes up the orientation based histogram extracting local orientation of edges. A detailed description of this method is discussed below.

4.3.1 Preprocessing

On the one hand, the test images are preprocessed in an off-line way. In this case, the method uses the Canny algorithm [16] to get an edge map from each test image. For the Canny algorithm, we use a 9×9-size Gaussian mask with $\sigma = 1.5$. The method then applies a cropping operation to the result using horizontal and vertical projections to cover the interest object with a minimum bounding rectangle defined between the upper left corner $(x_\mathrm{i}, y_\mathrm{i})$ and the lower right corner (x_f, y_f).

Let E be a $m \times n$ binary edge map image with a white background, with the vertical V_E and horizontal H_E projection of E defined as:

$$V_E(\mathrm{i}) = \sum_{j=1}^{n} E(\mathrm{i}, j), \mathrm{i} = 1, \cdots, m \tag{4.5}$$

$$H_E(j) = \sum_{\mathrm{i}=1}^{m} E(\mathrm{i}, j), j = 1, \cdots, n. \tag{4.6}$$

In addition, the limits of the bounding rectangle is computed as follow:

- x_{i}: first index q in H_E complying with $H_E(x_{\mathrm{i}}) \geq 0.2 \times max(H_E)$, where indexes are evaluated from $q = 1$ to $q = n$.
- x_f: first index q in H_E complying with $H_E(x_{\mathrm{i}}) \geq 0.2 \times max(H_E)$, where indexes are evaluated from $q = n$ down to $q = 1$.
- y_{i}: first index q in V_E complying with $V_E(x_{\mathrm{i}}) \geq 0.2 \times max(V_E)$, where indexes are evaluated from $q = 1$ to $q = m$.
- y_f: first index q in V_E complying with $V_E(x_{\mathrm{i}}) \geq 0.2 \times max(V_E)$, where indexes are evaluated from $q = m$ to $q = 1$.

On the other hand, the sketch (the query) is preprocessed in an on-line manner. The processing is similar as before, but since the sketch is already an edge map representation, this method applies a thresholding operation instead of the Canny operation.

4.3.2 Histogram Computation

Here, we compute a K-bin histogram (usually K=72) where each bin corresponds to a quantization of a gradient orientation regardless the direction of the gradient. The computation of the gradient orientation is carried out within a local region which is called "a block" and which size is defined to be regular around the whole image. We define the size of block as $B_h \times B_w$. The complete description of this computation is presented in the next lines.

1. Compute the gradient vector for each pixel on the image. Let $[G_x(r, s), G_y(r, s)]$ be the gradient vector computed for the pixel (r, s).
2. Divide the image into $W_h \times W_w$ regular blocks as shown in Figure 4.2. We regard each block as a local area where we will estimate the local orientation. In this approach the block size $(B_h \times B_w)$ is dependent on the image size (I_h, I_w). The reason for defining (B_h, B_w) with respect to (I_h, I_w) is to get the method to be robust to scale changes. In this way, B_h and B_w are defined as:

$$B_h = \left\lfloor \frac{I_h}{W_h} \right\rfloor, \qquad B_w = \left\lfloor \frac{I_w}{W_w} \right\rfloor \tag{4.7}$$

3. Compute local orientations according to the squared gradient method as follows:
   - Let $b_{\mathrm{i}j}$ be a block and $\alpha_{\mathrm{i}j}$ its corresponding orientation ($\mathrm{i} = 1, \cdots W_h, j = 1, \cdots, W_w$).
   - Let $L_x^{\mathrm{i}j}$ and $L_y^{\mathrm{i}j}$ be the components of the local gradient with respect to x and y directions, respectively, computed on a block $b_{\mathrm{i}j}$ as follows:

$$L_y^{\mathrm{i}j} = \sum_{(r,s)\in b_{\mathrm{i}j}} 2G_x(r, s)G_y(r, s) \tag{4.8}$$

$$L_x^{\mathrm{i}j} = \sum_{(r,s)\in b_{\mathrm{i}j}} (G_x(r, s)^2 - G_y(r, s)^2) \tag{4.9}$$

   here, $L_\beta^{\mathrm{i}j}$ is the gradient on $b_{\mathrm{i}j}$ in the β direction.

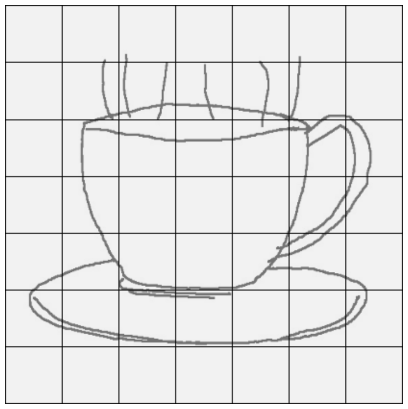


Figure 4.2: Block division for the HELO descriptor.

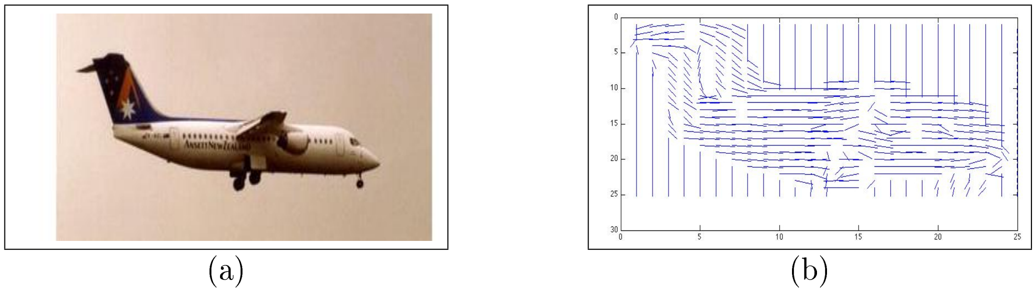


Figure 4.3: An image (a) with its corresponding orientation field (b). Here, $W_h = W_w = 25$.

- Calculate the block orientation α_{ij} correspondent to the block b_{ij} as follows:

$$\alpha_{ij} = \frac{1}{2}tan^{-1}\left(\frac{L_y^{ij}}{L_x^{ij}}\right) + \frac{\pi}{2}. \tag{4.10}$$

  In Equation 4.10, the angle is divided by a factor of two because the gradients were doubled with respect to their angles. Additionally, since the gradient is orthogonal to the underlying edge, we get the real edge orientation adding $\pi/2$. Finally, supposing that $-\pi \leq tan^{-1}(\cdot) \leq \pi$, α_{ij} ranges between 0 and π.

4. Create a K-bin histogram h to represent the distribution of the block orientations in the image. Each block orientation α_{ij} will contribute by one unit to the bin $index(\alpha_{ij})$ of h, where:

$$index(x) = \left\lfloor x \times \frac{K}{\pi} \right\rfloor \tag{4.11}$$

5. To avoid the effect of noise in the construction of the histogram, we discard blocks containing few edge points. To this end, we use a threshold th_{edge} to filter out those blocks. We set th_{edge} to be equal to the half of the maximum value between B_h and B_w. The resulting histogram is named the *Histogram of Edge Local Orientation* whose acronym is HELO. Figure 4.3 shows an orientation field of a test image, computed by this method.

Our proposed descriptor HELO is invariant to translation because the orientation is independent of edge positions. In addition, since the block size depends on the image size, HELO is also invariant to scale changes. Furthermore, as the computation of orientations is invariant with respect to lighting conditions, our method becomes a descriptor invariant to illumination, as well.

Finally, we measure similarity between two HELO descriptors using the L_1 distance, also known as the Manhattan distance function (See Equation 2.33).

Similar to the previous orientation-based method discussed in the previous chapter, the HELO descriptor is computed in linear time $O(N)$ with respect to the number of pixels of the image.

4.3.3 Tackling the rotation problem

To deal with the rotation invariance, we need to normalize, in terms of the object orientation, both the sketch and the test image before computing the HELO descriptor. To this end, we use two different normalization processes and then we compute two HELO descriptors, one for each normalization process. After that, we measure similarity by combining linearly the partial distances stemming from the comparison between descriptors produced by two different normalization processes. For normalization, we use principal component analysis (PCA) and polar coordinates (PC). We present a detailed description of this approach below:

- **Preprocessing**: This stage is similar to the previous one (Section 4.3), except that in this case the cropping operation is performed after the orientation normalization process.
- **Orientation normalization**:
  - **Principal Component Analysis based Approach (PCA)**: We compute a 2-d eigenvector v representing the axis with higher variance of the edge pixels (those with value 1). To this end, we represent each edge pixel as a coordinate vector [x,y].

    Let **X** be a matrix with N rows and 2 columns, where N is the number of edge pixels. Thus, each row of **X** corresponds to an edge pixel in the form of a 2-component vector. The normalization process using PCA requires linear time with respect to the number of pixels of the input image, and the steps are summarized below:

    1. Center the points of **X** at the origin [0,0]. This is attained subtracting the mean of **X**.
    2. Compute the covariance matrix mc of **X**.
    3. Compute the eigenvalues and eigenvectors of mc.
    4. The eigenvector $v = [v_x, v_y]$ with the highest eigenvalue is the vector representing the main orientation of the underlying image object.
    5. Compute $\alpha = tan^{-1}(v_y/v_x)$.
    6. Normalize the input image rotating it $-\alpha$ degrees around their center of mass.
    7. Crop the resulting image as in Section 4.3. Some examples of the result of this normalization are shown in Figure 4.4.

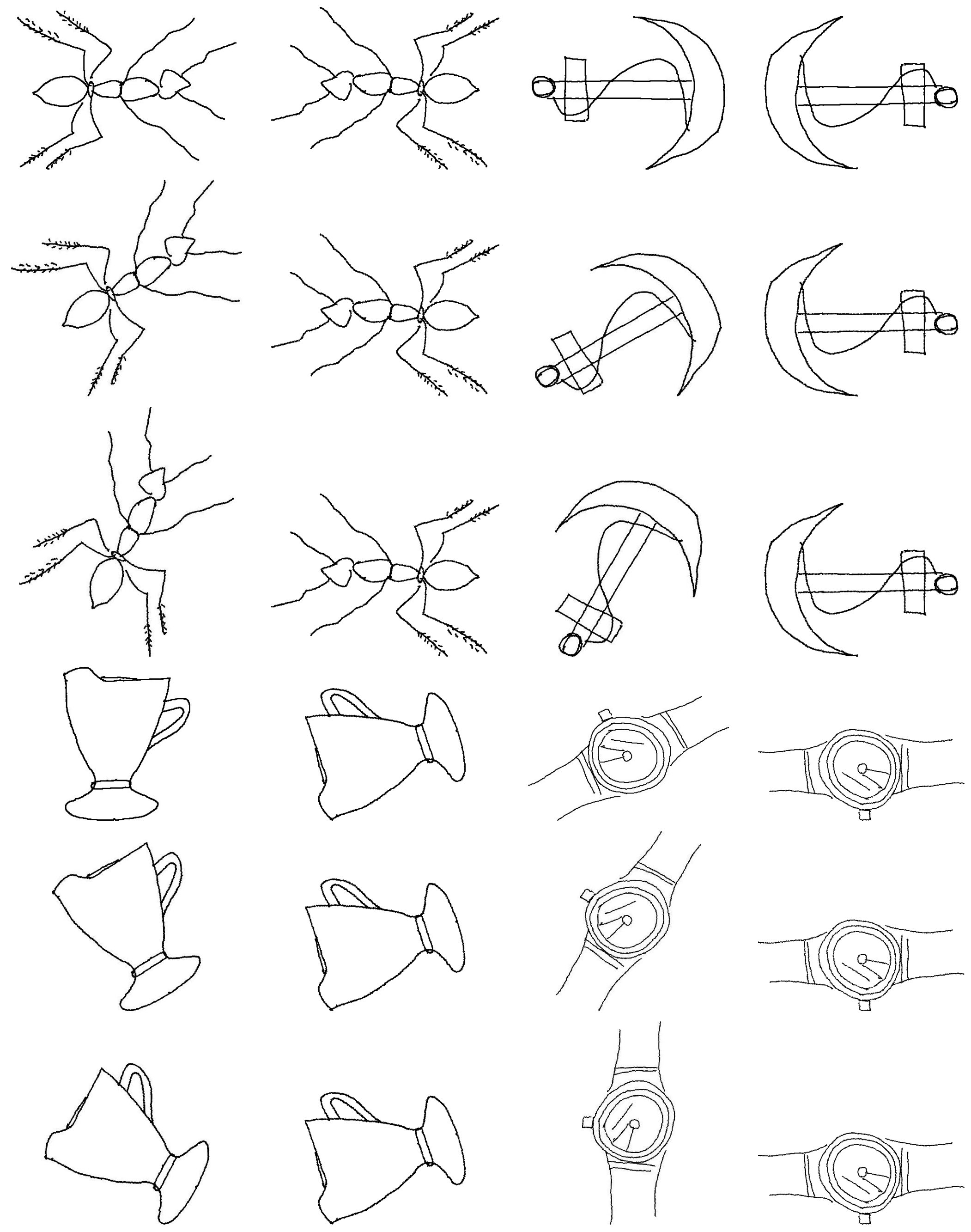


Figure 4.4: PCA normalization results. (a) and (c) show sketches affected by a rotation variation. (b) and (d) show the sketch after applying the PCA normalization process.

– **Polar Coordinate based Approach (PC)**: We transform both the test image and the sketch abstract representation into polar coordinates. In this case, two rotated images containing the same object become similar images only affected by an horizontal shifting. A simple algorithm to convert a binary image to its corresponding polar representation is described in Algorithm 1. The algorithm is linear with respect to the number of pixels of the input image. A couple of examples of the result of this PC-based normalization process are shown in Figure 4.5.

Algorithm 1 toPolarImage

```
Require: $im$: a binary image
    $h \leftarrow height(im)$
 2: $w \leftarrow width(im)$
    $R \leftarrow \sqrt{\frac{h}{2}^2 + \frac{w}{2}^2}$
 4: $N\_ANGLES \leftarrow 360$
    $newIm \leftarrow zeros(R, N\_ANGLES)$
 6: $E = \{(x,y) | im(x,y) = 1\}$
    $[c_x, c_y] \leftarrow$ center of mass of $E$
 8: for $(x,y) \in E$ do
      $x' \leftarrow x - c_x$
10:   $y' \leftarrow c_y - y$
      $r \leftarrow \sqrt{y'^2 + x'^2}$
12:   $\alpha \leftarrow atan2(y', x')$
      if $\alpha < 0$ then
14:     $\alpha = \alpha + 2\pi$
      end if
16:   $angleIdx \leftarrow (N\_ANGLES)(\alpha/2\pi)$
      $newIm(r, angleIdx) \leftarrow 1$
18: end for
    return $newIm$
```

- **Histogram Computation**: This stage is exactly similar to the previous one (Section 4.3). However, we compute similarity between a test image I and a sketch S combining PCA-based HELO an PC-based HELO. Let I_{PCA} and S_{PCA} be the PCA-based HELO descriptors computed over I and S, respectively. Let I_{PC} and S_{PC} be the correspondent PC-based HELO descriptors. The similarity measure $sm(I,S)$ is computed as follows:

$$sm(I,S) = w_{PCA} L_1(I_{PCA}, S_{PCA}) + w_{PC} L_1(I_{PC}, S_{PC}) \tag{4.12}$$

where, $w_{PCA} + w_{PC} = 1$, $w_{PCA} \geq 0, w_{PC} \geq 0$ and L_1 is the Manhattan distance.

Our proposal may be configured for working with or without rotation invariance. This is an advantage, because the rotation invariance requirement commonly depends on the application. For example, in the context of handwriting recognition rotation invariance may result in confusing the digit 6 with digit 9.

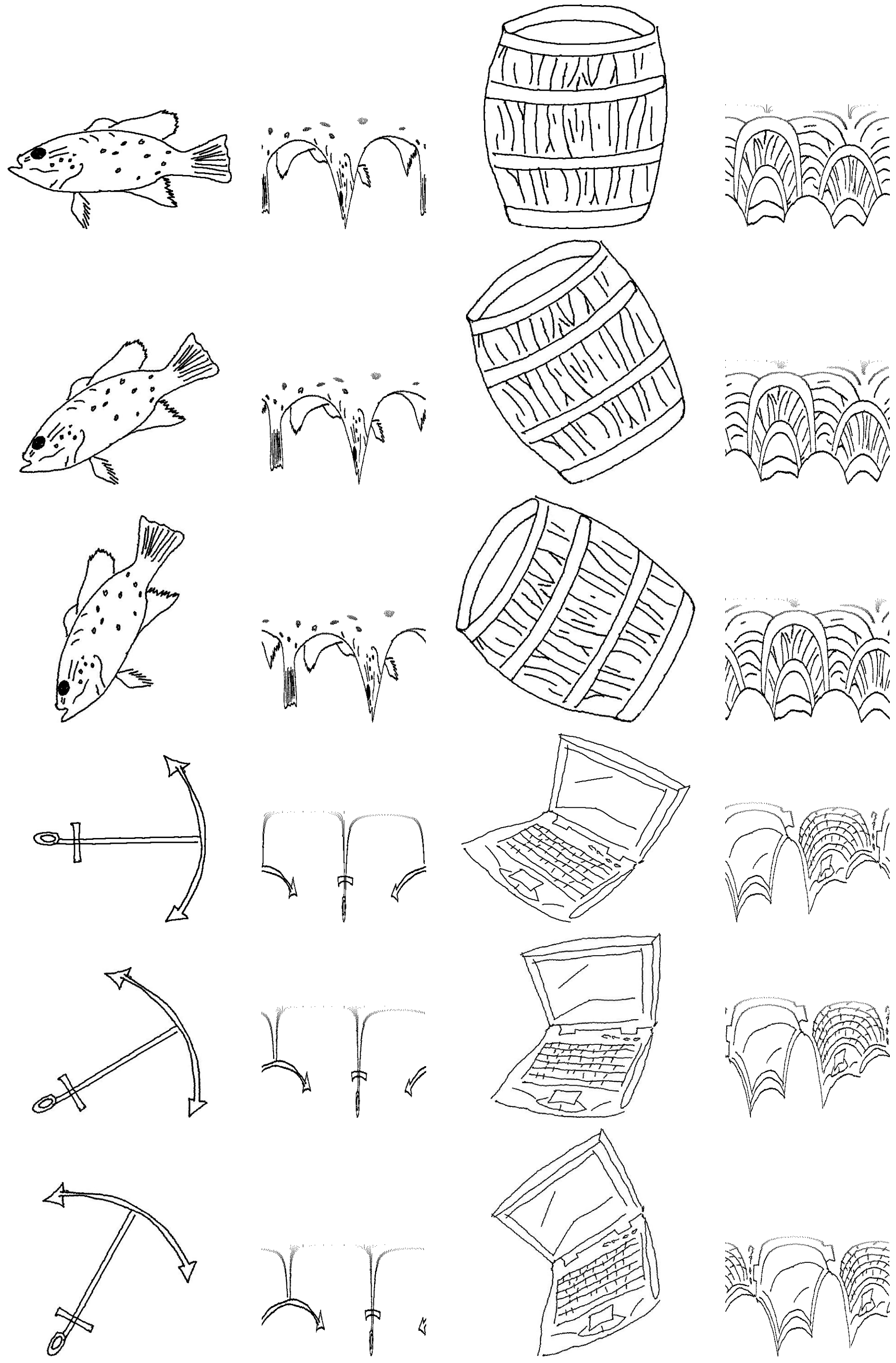

(a) (b) (c) (d)

Figure 4.5: PC normalization results. (a) and (c) show sketches affected by a rotation variation. (b) and (d) show the sketch after applying the PC normalization process.

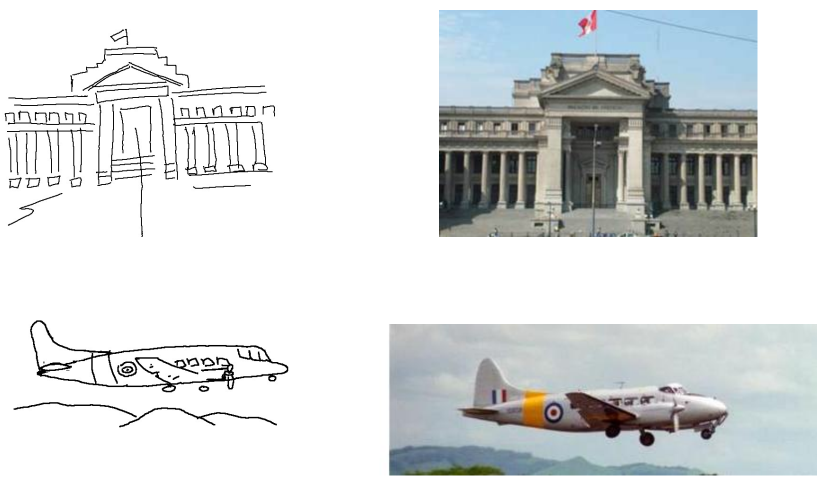

(a) (b)

Figure 4.6: A sketch on column (a) and its corresponding target image on column (b).

4.4 Experimental Evaluation

To compare the performance of our global descriptor we have developed a benchmark containing a set of test images and a set of sketch images (queries). For the test image database, we have randomly selected 1326 images. We selected 1285 color images from the Caltech101 database [34]. Additionally, we added 46 images of castles and palaces to our database.

Many of the test images consist of multiple objects or are cluttered images. For the query database, we have chosen 53 images from the test database. For each chosen image, a line-based sketch was hand-drawn. Thus, we have 53 sketches [1]. In order to produce sketches, a target image was presented to participants and they were asked to draw a line-based sketch resembling the target image. Two examples of a sketch together with its corresponding target image is shown in Figure 4.6. Moreover, some examples of the sketches are shown in Figure 5.14, and a variety of images from the test database are shown in Figure 4.8.

We compare our method with five other methods according to the state of the art. Four of these methods are: APAI [18], ST [29], HED [48], and EHD [93], which were implemented following the specification that appears in the corresponding papers. Additionally, we compare our method with the *histogram of distance distribution* (HDD) which represents the distribution of Euclidean distances between a random set of edge pixels.

The evaluation of the methods was performed by querying each sketch for the most similar images and finding the target image rank. We called this rank *query rank*. For measuring our results, we use two metrics. First, we use *Mean of Query Rank* (MQR), for which the average of all *query ranks* is computed. Second, we use the *recall ratio* R_n, which shows the ratio to retrieve the target image in the best n-candidates. R_n is defined as follows:

$$R_n = \frac{\text{target images among first } n \text{ responses}}{\text{total number of queries}} \times 100 \tag{4.13}$$

[1] http://prisma.dcc.uchile.cl/archivos_publicos/Sketch_DB.zip

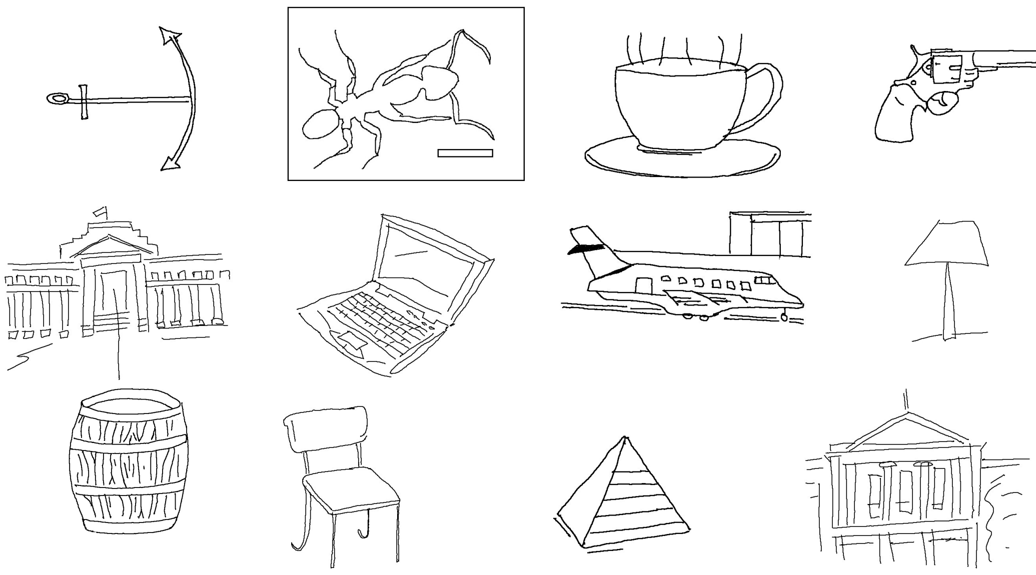


Figure 4.7: Examples of sketches.

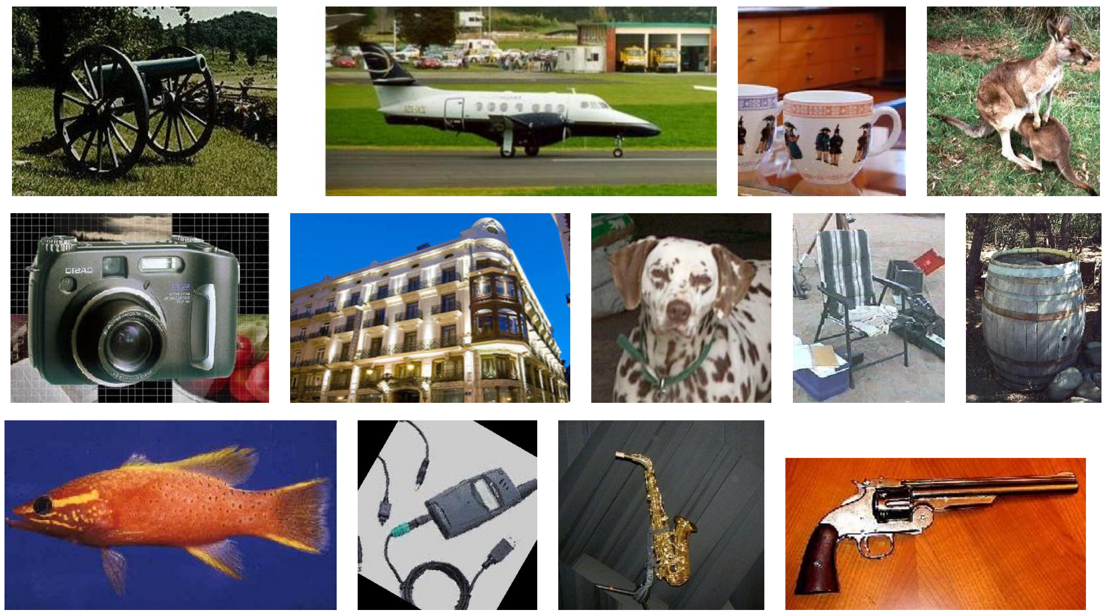


Figure 4.8: Examples of test images.

To evaluate translation, scale, and rotation robustness, we divide the experiments in two parts. First, we evaluate our method with sketches having different scales and positions from the corresponding target images. Second, we evaluate our method with sketches that have been rotated by three different angles (30°, 60°, and 90°). Therefore, in the last case, we have $53 \times 4 = 212$ sketches . The results of these experiments are discussed in the following subsections.

Our method needs three parameters to be specified, the histogram length (K), the number of horizontal and vertical blocks (W), and a threshold (th_{edge}). We fix $K = 72$, $W = 25$, and we set th_{edge} as 0.5 times the maximum image dimension. These parameter values were chosen experimentally.

In Figure 4.9 we present a MQR graphic showing the performance of HELO for different values of K (the histogram size). We note that for values of K below 50 the method degrades. However for values greater than 50 the method has a stable performance achieving a MQR value around 25.

In the same way, in Figure 4.10, we present a MQR graphic showing the performance of HELO for different values of W (the grid size). We note that a small value of W degrades drastically the performance of the method. Also, a large value of W may degrade the performance of our method. A large value of W means a small region within the orientation is estimated. Therefore, decreasing the local region causes the method has to estimate the orientation relying on a reduced number of neighbors which leads a less reliable orientation.

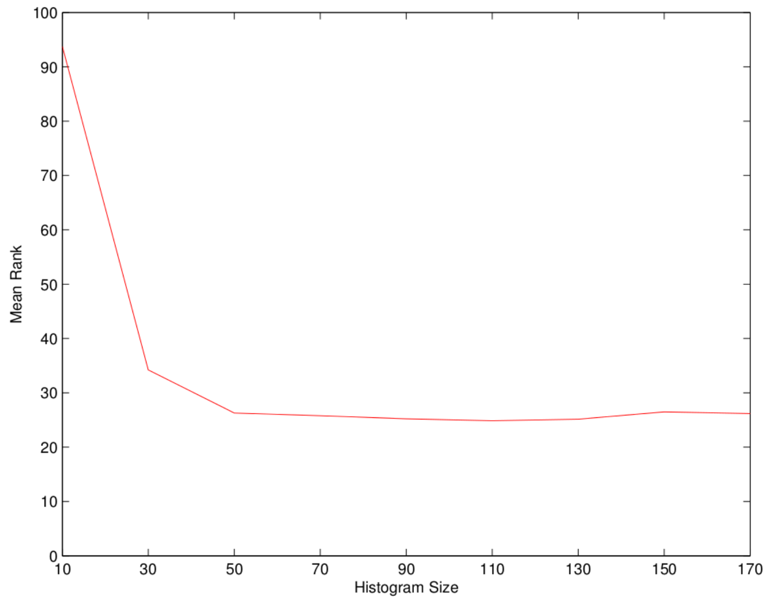


Figure 4.9: Mean Query Rank for different values of K (histogram size) .

4.4.1 Translation and scale invariance comparison

Figure 4.11 shows the MQR for each evaluated method. We observe that our method is more robust than the other methods when scale and position changes exist. We achieve 24.60

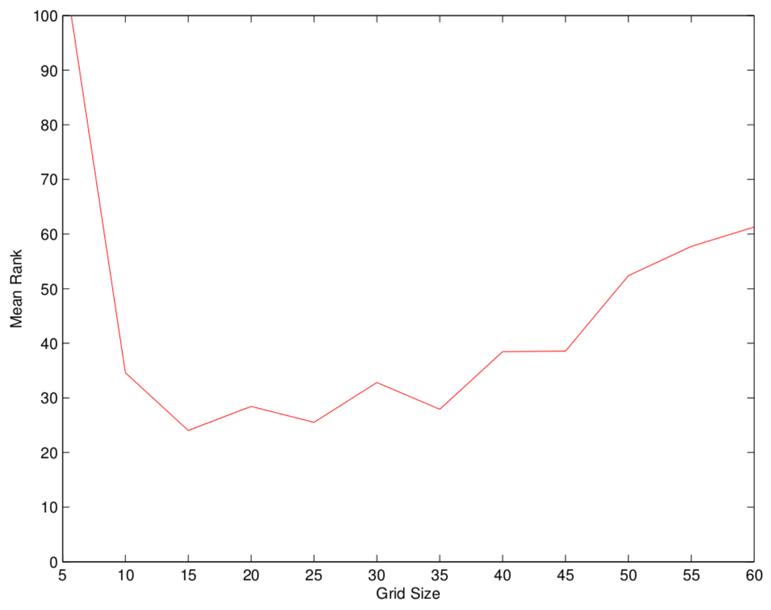


Figure 4.10: Mean Query Rank for different values of W (grid size).

as MQR value. This indicates that HELO needs to retrieve less than 25 images from the database to recover the target image. In comparison with the other methods, the EHD is the closest to ours with a significant difference. EHD achieves 208.26 as MQR, specifically, EHD would require retrieving almost 208 images to find the target one. Thus, our method improves effectiveness on recall by 8.4 times with respect to that of the state of the art.

Figure 4.12 shows the recall ratio graphic. This graphic shows that HELO outperforms the state of the art methods for any value of n. In addition, an example of image retrieval using HELO is shown in Figure 4.13.

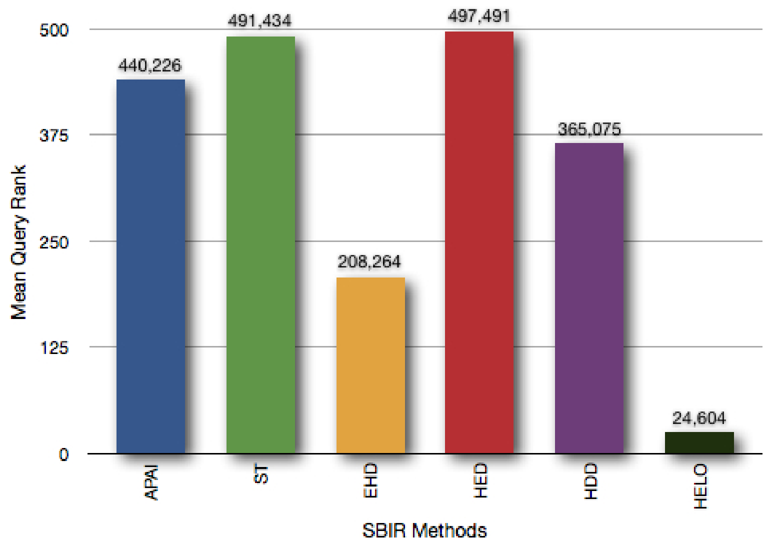


Figure 4.11: Mean Query Rank of the evaluated methods.

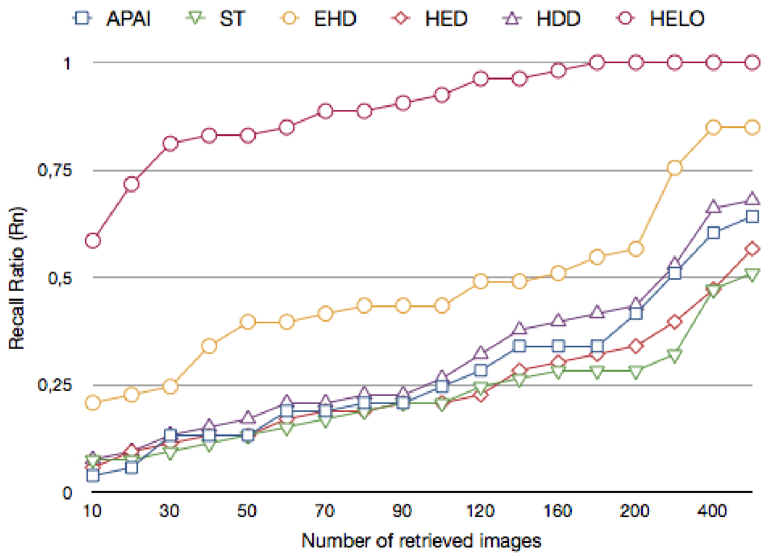


Figure 4.12: Recall ratio graphic for the evaluated methods.

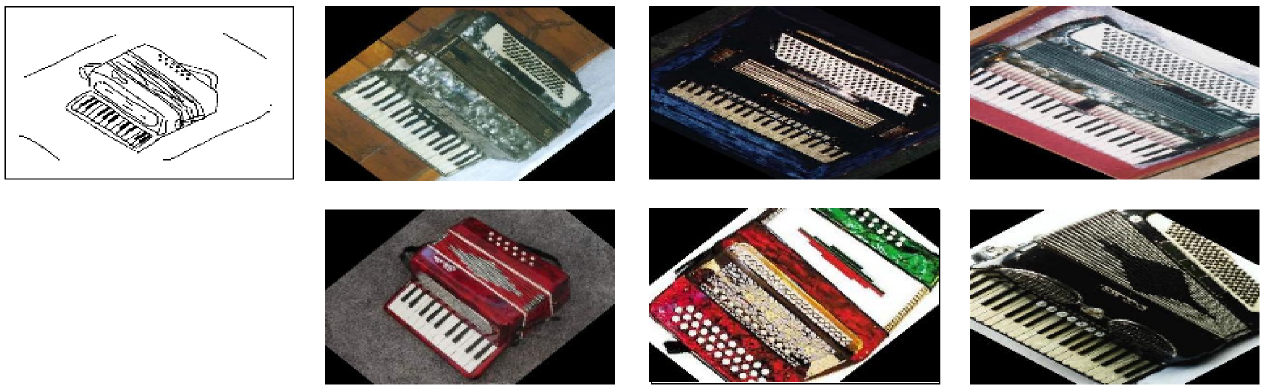
4.4.2 Rotation invariance comparison

In this section, we show the HELO performance under rotation changes on the sketches. First of all, we evaluate the HELO descriptor using separately PCA and PC as orientation normalization methods. Using PCA, we obtain a MQR value of 197.04. The principal axis is computed over the edge point distribution. However, a sketch is a simple rough hand-made drawing without details as those appearing in the target image. Due to this, the input sketch and the target image may have a very different principal axis affecting the retrieval effectiveness. Using PC, sketches affected by different angular shifts have similar representations in polar coordinates. This is the reason why PC gives a better MQR value (156.09) than that given by PCA. However, PC changes drastically the edge point distribution decreasing the discriminative power.

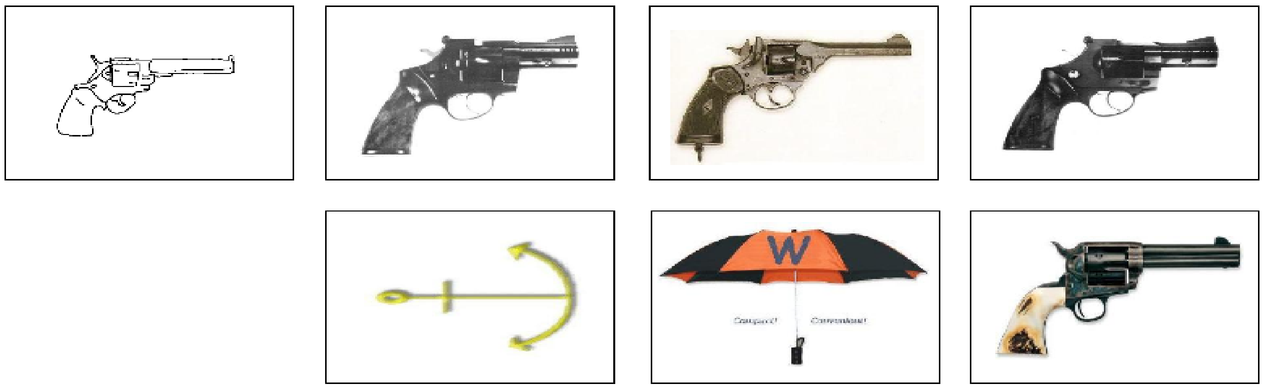
Therefore, to take advantage of each orientation normalization method we propose a linear combination of PC-based HELO and PCA-based HELO, that allows us to improve the retrieval effectiveness. Using our approach we get a MQR value to 101.09. We described this method in Section 4.3.3. We will refer to the combined-based HELO descriptor as HELO_R. To implement the HELO_R descriptor, we use $w_{PCA} = 0.3$ and $w_{PC} = 0.7$.

Figure 4.14 shows the MQR for the evaluated methods under a rotation distortion. Clearly, under this kind of change, our proposal improves the effectiveness on recall by 2.6 times with respect to that of the state of the art.

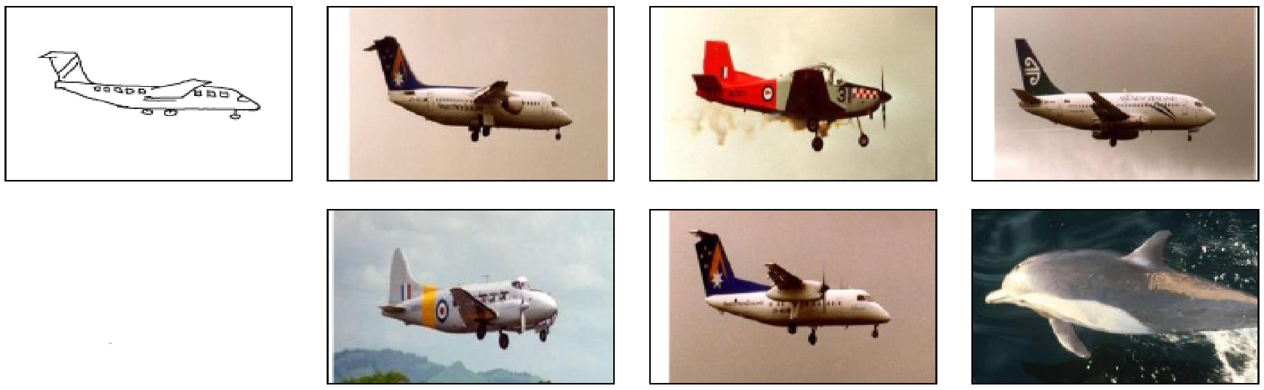
To visualize how many images are needed to retrieve the target image, Figure 4.15 shows the recall ratio graphics comparing the six evaluated methods.



(a)



(b)



(c)

Figure 4.13: (a), (b) and (c) are examples of the first six retrieved images using the HELO approach. The first image is the query.

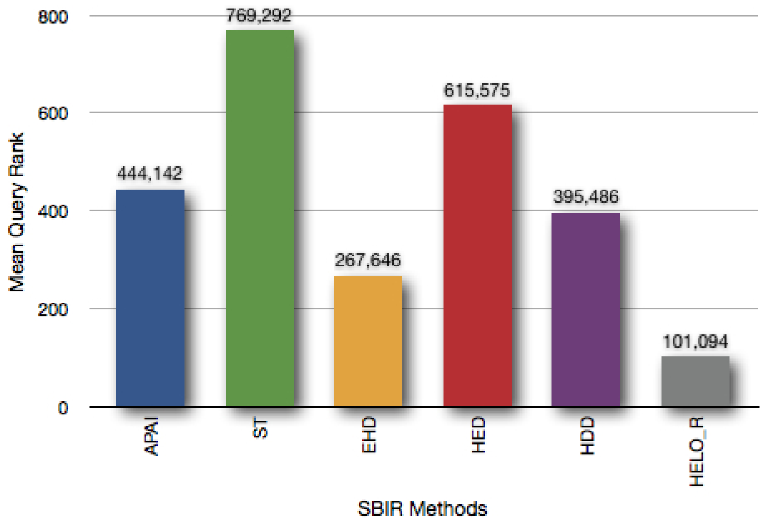


Figure 4.14: Mean Query Rank of the evaluated methods with respect to rotation invariance.

4.5 Summary

In this chapter we have described a global approach (HELO) to address the sketch based image retrieval problem. Our proposal is based on computing an orientation histogram using the gradient square method. In this way, the image is divided by a cell grid, and a edge orientation is estimated within each cell. In addition, the HELO method discards cells where there is not a significant amount of edge pixels, in order to be robust to noise.

Although our approach shows a very competitive performance with respect to the results achieved by current global approaches, it still has some drawbacks owing to its global characteristic. For instance, our proposal has poor performance in the cases of partial matching and occlusion. These are important properties since users commonly draw only the objects that they are interested in, hoping to get, as response, all images containing objects similar to those drawing in the query. A simple example is when a user is looking for images containing the sun, the user will possibly draw only a circle.

Unfortunately, a global approach does not overcome the aforementioned problems, we require a local approach. Therefore, in the next chapter we discuss a novel local approach for the sketch based image retrieval based on extracting structural information from sketches and test images, providing a higher semantic level representation.

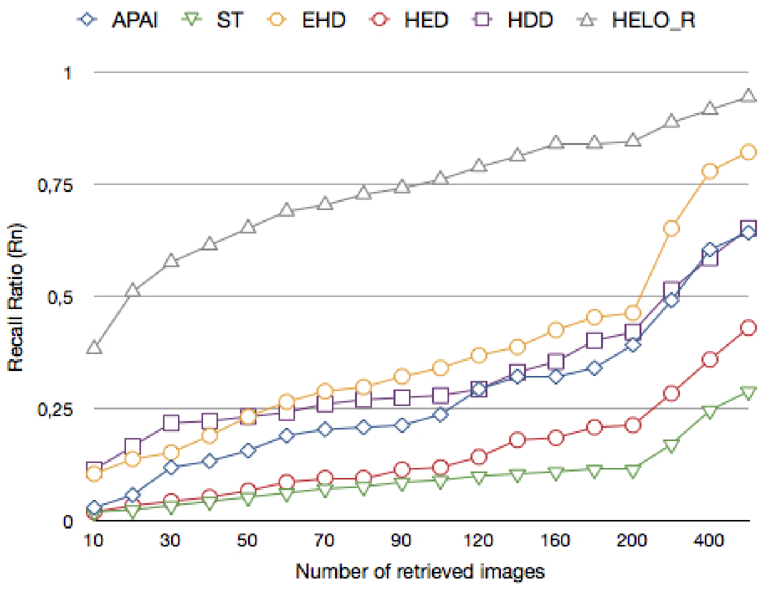


Figure 4.15: Recall ratio graphic for the evaluated methods with respect to rotation invariance.

Chapter 5

Keyshape Based Approach

5.1 Introduction

Sketches are characterized by representing the structural components of an object instead of representing color or texture information. For instance, when a person is asked to make a simple drawing of a teapot, he or she will probably draw only three components: the body, the spout and the handle like one of the pictures depicted in Figure 5.1. In addition, the absence of color and texture information in sketches may cause the retrieval process becomes a difficult task. This fact also means that techniques thought to work on regular images do not work appropriately when the input is a simple sketch. Therefore, in this section we present a novel method for retrieving images using a sketch as query. Our proposal is characterized principally by exploiting the *structural* information provided by sketches, which allow us to work in a higher semantic level representation.

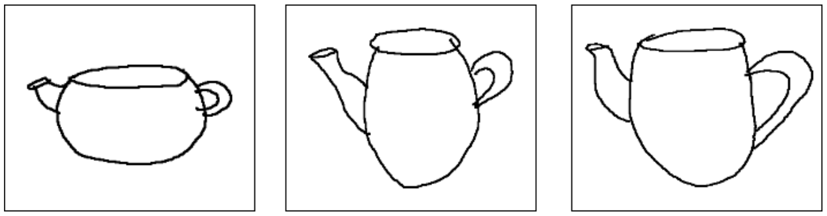


Figure 5.1: Examples of hand drawings of teapots.

Definition 3 The object structure is the distribution of the parts that compose such an object.

From the Definition 3, we extract two relevant terms: (1) the component of an object, and (2) the distribution of the components.

1. *Component*: The components of an object are difficult to define because these exist in diverse scales. However, we will define a component from the geometric perspective. Therefore, a component is a simple **geometric shape** like, arcs, ellipses, circles, triangles, rectangles, squares, or simply straight lines. In this way the teapot from the

Figure 5.1 may be decomposed into a set of primitive shapes like an arc representing the handle, two arcs representing the spout, and an ellipse representing the body.

2. *Distribution*: The distribution of the components describes the **spatial relationship** between such components. In this way, using the teapot example again, the distribution should point out that the handle and the spout are located on the sides of the body; one on the left and the other on the right side.

To our knowledge, the methods proposed for sketch based image retrieval have not yet exploited the structural property of sketches. Instead, many of the current methods are based on edge point distribution [18] or edge point orientations [31, 45] which do not appropriately represent the structural components of the objects on the image. Furthermore, the interest point (*keypoint*) approach [96], proposed in the computer vision field, does not represent components with the semantic level as we are defining here. Moreover, the local region around *keypoints* could not be discriminating enough since sketches are simple line-based drawings.

Therefore, our proposal is based on describing objects appearing in an image using structural components. Representing sketches by means of their structural components brings up the following advantages:

- A structural representation allows methods to represent objects in a higher semantic level which is reflected in the increment of the retrieval effectiveness.
- A structural representation allows methods to handle a small number of components with respect to handle interest points or simply edge points. This leads to a more efficient matching step.

In order to deal with the structural property of objects we detect simple geometric shapes that we call *keyshapes* and which will represent the *components* of the objects in an image. In addition, we propose two local descriptors computed over each detected keyshape. These descriptors are characterized by representing the spatial distribution of the components. In short, our proposal exploits the structural property of an object describing the distribution of the parts composing it.

In addition, owing to our method is based on a different feature (the structure) from those used by current methods, a combination of our method with a method that has showed good results would lead to a significant improvement in the retrieval effectiveness. In this document, therefore, we experimentally demonstrate such a property.

Definition 4 A keyshape is a simple geometric shape that composes a more complex object. Examples of a keyshape may be a circle, an ellipse, a square, a line, among others.

5.2 Proposal

Our proposal consists of three stages (1) **keyshapes detection**, that allows us to detect simple shapes from an input image, (2) **local descriptor computation**, that allows us to locally represent the spatial relationship between a reference shape with respect to the others,

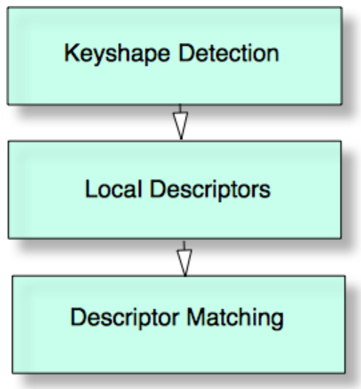


Figure 5.2: Proposal Stages.

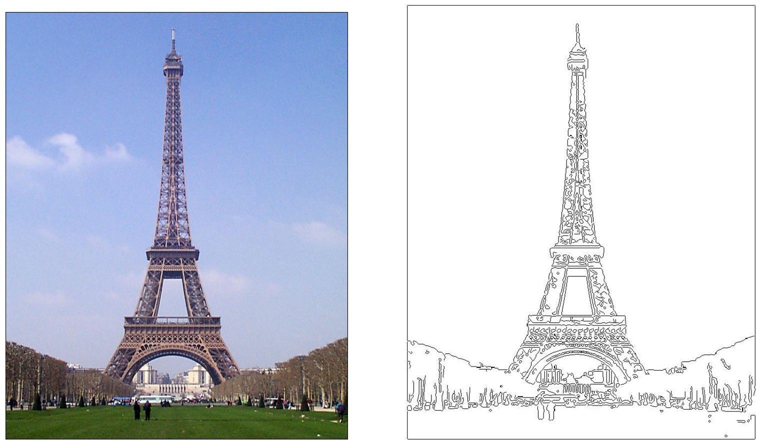


Figure 5.3: Simple Canny edges of a test image.

(3) **matching**, that computes a cost value after setting some relation between two sets of local descriptors. In the following sections we will describe each one of these stages in detail.

5.2.1 Keyshapes Detection

Sketch-like Representation

We will detect keyshapes from the input sketch (the query) and the images from the database (test images). In order to compare a sketch with a test image we require to transform a test image into a sketch-like representation. In particular, we need to transform the test images into sketches since they are regular color images. A simple way to carry out this transformation is by using an edge detection procedure. To this end, we use the Canny operator [16]. We prefer Canny than other methods like the *Berkeley boundary detector* [64] because of two folds: (1) Canny allows us to get edge pixels accurately, and (2) computing Canny is much faster that the Berkeley's approach. In Figure 5.3 we depict an image and its edge map representation computed using the well known Canny approach with $\sigma = 0.3$.

A simple edge detection method produces a chaotic edge image. Many of the edge pixels

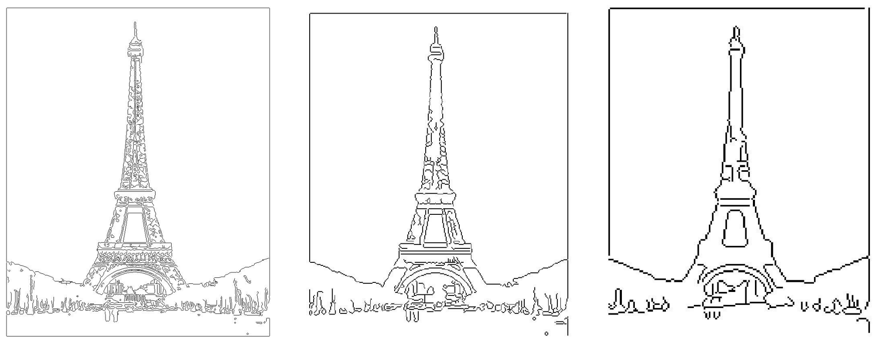


Figure 5.4: Edge image produced by a multiscale Canny.

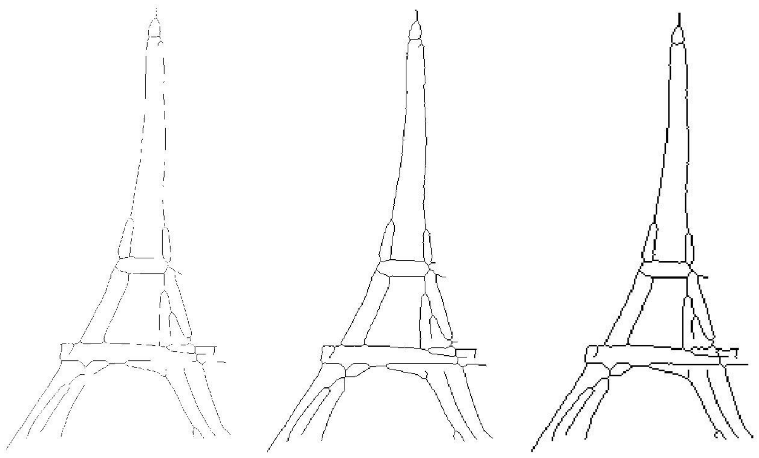


Figure 5.5: Edge images produced by a multiscale approach over a sketch.

do not provide relevant structural information as we note in Figure 5.3. Furthermore, many of them may be a result of noise which may cause degradation in the keyshape detection and consequently in the retrieval effectiveness. To solve this problem, we apply the Canny operator in a multiscale manner. Each scale is computed by the Canny operator applied over a downsampled image. For each scale, an image is downsampled using a factor of 0.5 with respect to the previous scale. In our method, we apply the downsampling process iteratively until the size of the resulting image is less than 200 pixels in one of its dimensions. In our experiments, the downsampling stops after approximately three iterations. We show an example of this process in Figure 5.4.

In the same vein, we apply a multiscale approach for the query. However, due to a query already being a sketch we apply a thinning operation [41] instead of the Canny operator. The result of this approach is shown in Figure 5.5. We can note that a sketch do not change through the multiscale approach, the resulting sketch is just a resized image.

After the multiscale stage we keep only the edge map of the third scale, where the noise has been reduced considerably. In addition, small details have been deleted leaving only high-scale edges representing the object from a coarser level.

Having a sketch-like image from both the test image and the query, the next step is to obtain an abstract representation from them that allows us to detect simple shapes in a

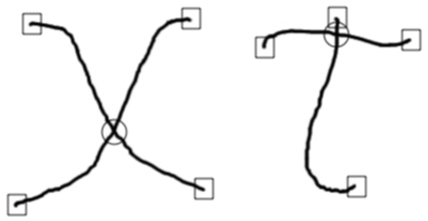


Figure 5.6: Branching and terminal points.

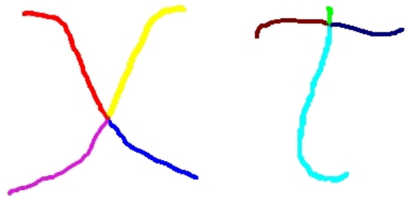


Figure 5.7: Strokes approximated by endge links.

easy way. A good approach for getting such an abstract representation is to decompose the sketch-like image into strokes.

Definition 5 A stroke is a set of pixels produced when a user is making a line-based hand-drawing between a "pen down" and a "pen up" event.

Considering that the underlying images are produced in an offline environment, we do not have available information about the real strokes. To solveF this problem, we propose to approximate real strokes by edge links.

Edge Links

An edge link is a sequence of edge pixels starting and ending in a branching or terminal point. In Figure 5.6, branching points are shown enclosed by circles and terminal points are shown enclosed by squares. We call the branching or terminal points simply *breakpoints*.

A simple algorithm to compute edge links is by tracing neighbor edge pixels starting from a breakpoint until another breakpoint is reached. Having a binary image as input, the output of the algorithm is a set of edge links.

The algorithm for edge links computation described in Algorithm 2 returns a set of edge links which approximate real strokes. For this reason, henceforth we call these edge links simply "strokes". In Figure 5.7, the image of Figure 5.6 is depicted with its corresponding strokes which are showed with different colors.

Algorithm 2 Edgelink

```
Require: bim: a binary image
 1: $E \leftarrow$ edgepixels(bim)
 2: $B \leftarrow$ detect_breakpoints(bim)
 3: $L \leftarrow \emptyset$
 4: for $\forall$e $\in E$ do
 5:     $visited$(e) $\leftarrow false$
 6: end for
 7: for $\forall b \in B$ do
 8:     $\tau \leftarrow \emptyset$
 9:     q $\leftarrow$ create a queue
10:     $push(q, b)$
11:     $end \leftarrow false$
12:     while $!empty(q)\&!end$ do
13:         $x \leftarrow pop(q)$
14:         $visited(x) \leftarrow true$
15:         $\tau \leftarrow \tau \bigcup x$
16:         if $isBreakpoint(x)$ then
17:             end $\leftarrow true$
18:         else
19:             for $\forall v$ that is a 4-connected neighbor of $x$ do
20:                 if $!visited(v)$ then
21:                     $push(q, v)$
22:                 end if
23:             end for
24:         end if
25:     end while
26:     $L \leftarrow L \bigcup \tau$
27:     $visited(b) \leftarrow false$
28:     $visited(x) \leftarrow false$
29: end for
30: return L
```

Keyshapes

Detecting simple shapes like circles or ellipses on an image might lead to a time consuming process, so we propose an efficient strategy for detecting six types of keyshapes. These keyshape classes are: (1) vertical line, (2) horizontal lines, (3) diagonal line (slope=1), (4) diagonal line (slope =-1), (5) arc, and (6) ellipse (see Figure 6.10). The latter also includes circular shapes. It is important to note that as the number of keyshapes increases, the complexity for scaling the method to large databases also increases. For this reason we decided to keep just six keyshape types. In spite of the small number of keyshape classes, we still get important structural information from sketch representations. We will now describe the complete process for detecting keyshapes.

Considering that a stroke may include one or more keyshapes, the first step is to divide

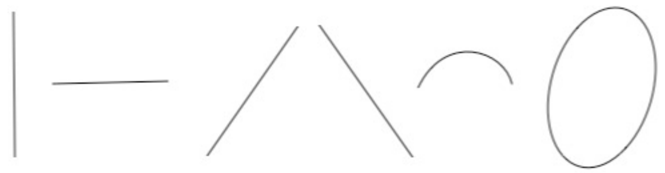


Figure 5.8: The six clases of keyshapes.

each stroke S, computed previously, into a set of one or more stroke pieces SP_S. To this end, a stroke S is divided with respect to its inflection points.

Definition 6 An inflection point is a point where the local edge pixels around it are distributed significantly in two directions.

To determine whether a stroke point q is actually an inflection point or not, we compute the two eigenvalues (λ_1, λ_2) of the covariance matrix of the points falling in a local region around q. Then, we evaluate the ratio $r = max(\lambda_1, \lambda_2)/min(\lambda_1, \lambda_2)$ and proceed to mark q as an inflection point if r is less than a threshold.

Trying to evaluate all points of a stroke is, of course, a time consuming task. Thus, instead of applying the inflection point test over all stroke points, we test only a small set of points. This set is named *set of maximum deviation points* (SoM). We obtain the SoM by approximating a stroke S, which is defined by a set of points $S = \{p_{\mathrm{i}}, \cdots p_j\}$, by straight lines. To this end, a line L is set between p_{i} and p_j. Then we look for the point $p_k \in S$ with the maximum distance δ_M to L. We call p_k a *maximum deviation point.* If $\delta_M > \mu$ then we process recursively the resulting substrokes $S_1 = \{p_{\mathrm{i}}, \cdots p_k\}$ and $S_2 = \{p_{k+1}, \cdots, p_j\}$. In our experiments, we set μ equal to 6 . Finally, all the maximum deviation points produced by this method form the SoM. A description of this algorithm is described in Algorithm 3.

Algorithm 3 getSoM

```
Require: $S = \{p_{\mathrm{i}}, \cdots, p_j\}$
 1: if i $> j$ then
 2:    return $\emptyset$
 3: end if
 4: $L \leftarrow$ a line defined between $p_{\mathrm{i}}$ and $p_j$.
 5: $[k, \delta_M] \leftarrow getMaxDistPoint(L, S)$
 6: $SoM \leftarrow \emptyset$
 7: if $\delta_M > \mu$ then
 8:    $SoM_1 \leftarrow getSoM(\{p_{\mathrm{i}}, \cdots, p_k\})$
 9:    $SoM_2 \leftarrow getSoM(\{p_{k+1}, \cdots, p_j\})$
10:    $SoM \leftarrow SoM_1 \bigcup \{(p_k, \delta_M)\} \bigcup SoM_2$
11: end if
12: return SoM
```

We should note that the Algorithm 3 returns the set of maximum deviation points (SoM) together with their corresponding deviation value. In Figure 5.9 we show a synthetic example of the stroke decomposition in a set of stroke pieces.

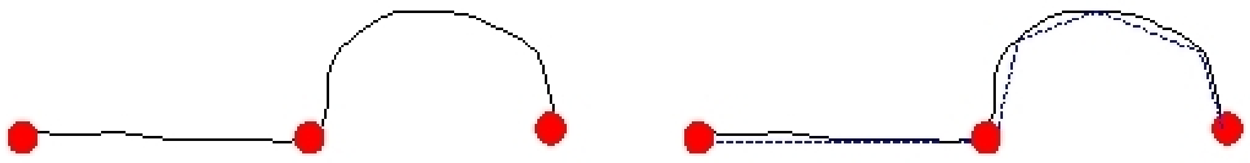


Figure 5.9: On the left, a synthetic example showing inflection points on a stroke. On the right, straight lines (dashed lines) approximating stroke pieces.

After the stroke decomposition stage, we have a set of stroke pieces $\{sp_1, \cdots, sp_K\}$ for a given stroke S. The next task it to classify each of these stroke pieces as one of the six predefined keyshape types. In addition, we could also get the number of straight lines approximating a stroke piece from the SoM. Let N_i be such a number for a stroke piece sp_i. This will be a valuable information for the subsequent steps.

To determine the occurrence of a keyshape, we start applying a test which will determine whether sp_i is an ellipse or not. We apply this test only if $N_i \geq 2$. To this end, we use the function *testEllipse*, that will be described afterward, that returns a fitness value indicating how well sp_i is approximated by an ellipse and the underlying approximated ellipse parameters. If $N_i < 2$ or the ellipse fitness value is less than a threshold, the stroke piece sp_i may be composed by arcs or straight lines. In this way, if $N_i = 1$, a line is detected. Otherwise, we apply a test for detecting arcs using the function *testArcs*, described later, that returns an error of arc approximation and the underlying arc paramaters. If the error of arc approximation is greater than a treshold we divide sp_i in the point with maximum deviation using the SoM. The resulting two substrokes are tested separately to detect arcs and lines recursively. The algorithms for detecting keyshapes are described in Algorithms 4 and 5.

```
**Algorithm 4** detectKeyshapes
Require: $SP$: a stroke piece
Require: $SoM_{SP}$: the set of maximum deviation points with respect to $SP$, $SoM_{SP} = SoM \bigcap SP$.
  $KS \leftarrow \emptyset$
  $N_i \leftarrow$ number of straight lines approximating $SP$.
  if $N_i > 2$ then
    $[t, param] \leftarrow testEllipse(SP)$
    if $t > TH_E$ then
      $KS = \{(\text{“ellipse”}, param)\}$
    else
      $[lines, arcs] \leftarrow detectArcsLines(SP, SoM_{SP})$
      $KS \leftarrow lines \bigcup arcs$
    end if
  end if
  return KS
```

Algorithm 5 detectArcsLines

```
Require: $SP = \{p_{\mathrm{i}}, \cdots, p_j\}$: a stroke piece
Require: $SoM_{SP}$: the set of maximum deviation points with respect to $SP$, $SoM_{SP} = SoM \bigcap SP$.
  $lines \leftarrow \emptyset$
  $arcs \leftarrow \emptyset$
  $N_{\mathrm{i}} \leftarrow$ number of straight lines approximating $SP$.
  if $N_{\mathrm{i}} = 1$ then
    $lines = \{(\text{“line”}, [p_{\mathrm{i}}, p_j])\}$
  else
    $[\mathrm{e}, param] \leftarrow testArc(SP)$
    if $\mathrm{e} < TH_A$ then
      $arcs = \{(\text{“arc”}, param)\}$
    else
      $k \leftarrow maximumDeviationPoint(SoM_{SP})$
      $SP_1 \leftarrow \{p_{\mathrm{i}}, \cdots p_k\}$
      $SP_2 \leftarrow \{p_{k+1}, \cdots p_j\}$
      $[lines_1, arcs_1] \leftarrow \mathrm{det}ectArcsLines(SP_1, SP_1 \bigcap SoM_{SP})$
      $[lines_2, arcs_2] \leftarrow \mathrm{det}ectArcsLines(SP_2, SP_2 \bigcap SoM_{SP})$
      $lines \leftarrow line_1 \bigcup line_2$
      $arcs \leftarrow arcs_1 \bigcup arcs_2$
    end if
  end if
  return $[lines, arcs]$
```

Detecting Ellipses

The *testEllipse* function takes a stroke piece SP and tries to approximate SP by an ellipse. To approximate an ellipse we use the ellipse general equation as in the work of Yao et al. [104].

$$Ax^2 + 2Bxy + Cy^2 + 2Dx + 2Ey + 1 = 0 \tag{5.1}$$

where the five parameters $(x_c, y_c, r_{max}, r_{min}, \theta)$ corresponding to the center of the ellipse (x_c, y_c), the maximum and minimum radii (r_{max}, r_{min}), and the angle respect to the major radius are obtained as follow:

$$\begin{aligned}
x_c &= \frac{BE - CD}{W} && (5.2)\\
y_c &= \frac{DB - AE}{W} && (5.3)\\
r_{max} &= \sqrt{\frac{-2 \cdot \det(M)}{W(A + C - R)}} && (5.4)\\
r_{max} &= \sqrt{\frac{-2 \cdot \det(M)}{W(A + C + R)}} && (5.5)\\
\theta &= \frac{1}{2} tan^{-1}\left(\frac{2B}{A - C}\right), && (5.6)
\end{aligned}$$

where

$$\begin{aligned}
W &= AC - B^2 && (5.7)\\
R &= \sqrt{(A - C)^2 + 4B^2} && (5.8)\\
M &= \begin{pmatrix} A & B & D \\ B & C & E \\ D & E & 1 \end{pmatrix}. && (5.9)
\end{aligned}$$

To solve the Equation 5.1 with respect to a stroke piece $SP = \{p_1, \cdots, p_n\}$, we take the following five points $\{p_1, p_{\frac{n}{4}}, p_{\frac{n}{2}}, p_{\frac{3n}{4}}, p_n\}$. We then validate the estimated ellipse with respect to the pixels on the image. To this end, we use a fitness function defined by Yao et al. [104].

The fitness function takes each point p on the approximating ellipse E and looks for the closer edge pixel on the image. Let d_p be the closer distance from p to any edge pixel on SP and we compute the fitness value as:

$$fitness_E = \frac{\sum_{\forall p \in E} \frac{1}{exp(\gamma \mathrm{d}_p)}}{|E|}, \tag{5.10}$$

where the distance function is the Manhattan distance and γ is a regularization factor that we set to 0.2. The fitness value varies from 0 to 1, where 1 indicates a perfect approximation while 0 indicates a *null* approximation.

Detecting Arcs

Since an arc is a segment of a circle, the *testArc* function tries to approximate a circle with five points selected from the underlying stroke piece.

These five points are chosen in the same way as in the case of the *testEllipse* function. The resulting parameters are the circle center (x_c, y_c), and the circle radius r. To evaluate how well a stroke piece SP is approximated by an arc, we compute an approximation error in the following way:

$$error_A = \sum_{p \in SP} |r - dist(p, (x_c, y_c))| \tag{5.11}$$

where $dist$ is the Euclidean distance.

After detecting ellipses, arcs, or straight lines, we represent each detected keyshape with a set of parameters. In this way, each keyshape is especified as follows:

- Lines: $[x_1, y_1, x_2, y_2, L, \iota]$, where (x_1, y_1) is the initial point and (x_2, y_2) is the final point of the detected line. L is the line length. In addition, we use ι to indicate the type of the line (horizontal, vertical, diagonal (slop 1), diagonal (slope -1)).
- Arcs: $[x_c, y_c, r]$, where (x_c, y_c) is the center of the estimated circle, and r_c is the corresponding estimated radius.
- Ellipse: $[x_c, y_c, r_{max}, r_{min}, \phi]$, these are the five parameters of an ellipse representing the center (x_c, y_c), the maximum and minimum radii (r_{max}, r_{min}), and the orientation ϕ.

Clustering Arcs

It is possible that a circle shape be detected as a set of arcs sharing similar parameters. For this reason, we apply an additional step in the keyshape detection stage aiming to solve this problem. Our algorithm works clustering the detected arcs using their three parameters $[x_c, y_c, r]$. Arcs falling in the same cluster are tested to determine if a circle is fitted to the set of arcs. The tested function we use here is similar as that used for testing ellipses.

Finally, we produce a new stroke representation by means of an image called **keyshape image** drawing on it each detected keyshape. The keyshape image might be regarded as a normalized stroke representation. Two examples of a **keyshape representation** produced by the keyshape detection process is shown in Figure 5.10.

5.2.2 Local Descriptors

In this section we focus on describing a local region around each keypoint, aiming to represent the distribution of *keyshapes*. We know from Definition 3 that the distribution of the object components is a very important aspect for the structural characterization of the underlying objects. In this stage, we propose two local descriptors for the sketch based image retrieval problem which are characterized by taking into account spatial information of the keyshapes.

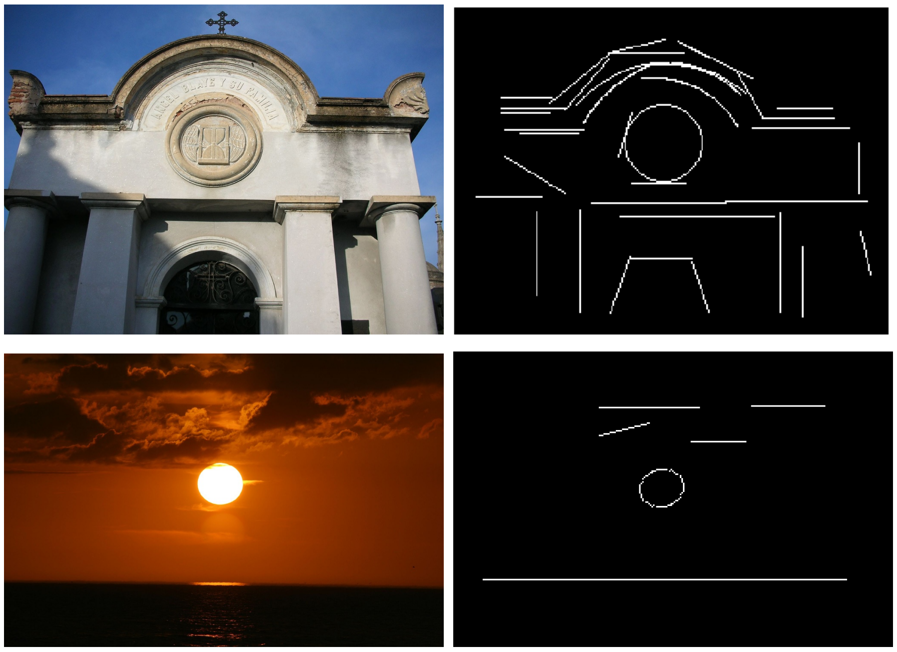


Figure 5.10: Test images on the left and their keyshape images on the right.

The first descriptor is called *Keyshape Angular Spatial Descriptor* (KASD). It divides a local region in an angular way and takes some information from each slice to make a histogram, where the histogram size is the same as the number of slices. The angular partitioning has been used in the computer vision field for the object recognition task [68] and for describing line-base hand-drawings in a global manner [18].

The second descriptor is named *Histogram of Keyshape Orientations* (HKO). It is a SIFT-like descriptor which computes a local histogram of keyshape orientations on a local region around a referent keyshape.

Local Region

We define a local squared region with respect to a keyshape (the referent keyshape) in order to compute a local descriptor that characterizes the spatial distribution of the keyshapes around the referent keyshape. The center of the local region coincides with the center position of the underlying keyshape. Thus, let k be a keyshape. If k is a line, the local region is centered on the center of that line. If k is an arc, the local region is centered in the center of the circle containing the arc. In the case of k is an ellipse, the region is centered on the ellipse center.

In order to face with scale variations, we define the region size depending on the keyshape size. Thus, let k be a keyshape and l be the length of the underlying square-like region side, we proceed to compute l as follows.

- If k is a line then $l = length(k) \cdot \eta$.
- If k is an arc then $l = radius(k) \cdot \eta$.
- If k is an ellipse then $l = major\_radius(k) \cdot \eta$.

The symbol η is a constant that we define to be equal to 3. Figure 5.11 shows a keyshape with the scope of its corresponding local region.

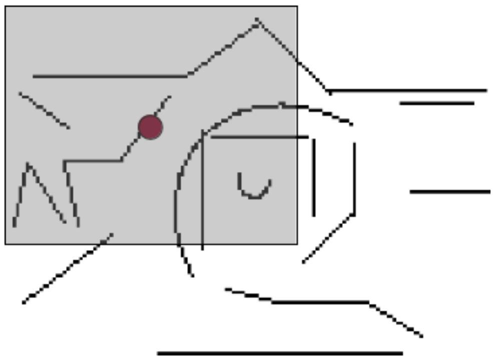


Figure 5.11: A local region of around of a diagonal-type keyshape (that marked with a red filled circle).

Keyshape Angular Spatial Distribution (KASD)

Let k be a reference keyshape. We divide the local region in angular partitions around k as shown in Figure 5.12(a). The number of angular regions or slices (N_{SLICES}) is fixed (we suggest using four or eight slices). For each slice, we compute a local histogram that represents the local distribution of the keyshape points with respect to the six keyshape types. Consequently, we obtain a 6-bin histogram for each slice. Then, we concatenate all the local histograms built for each slice to form a $N_{SLICES} \times 6$-size descriptor. Finally, we transform the descriptor into its unit representation.

The spatial distribution of keyshapes is represented by the local histograms spread through the slices in which the local region is partitioned.

Histogram of Keyshape Orientations (HKO)

Let k be a reference keyshape, we divide the local region around k into a 2×2 grid, where each cell of the grid is called a subregion, as shown in Figure 5.12(b). For each subregion we compute a histogram of orientations. The orientation angle varies between 0 and π, and it is quantized using 8 bins. The keyshape orientation is computed approximating local gradients using Sobel masks [40]. The local gradients are computed over the keyshape image where many of the outliers have been removed. Therefore, computing orientations over the keyshape image is more robust than computing them over a simple edge map representation. The final descriptor is the concatenation of the four histograms of orientations. We obtain a 32-size descriptor as a result of having four 8-bin histogram for each one of the four subregions. Similar to the above, we take the unit representation of the descriptor.

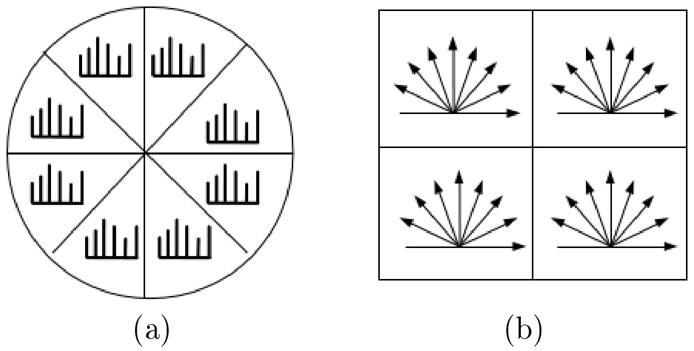


Figure 5.12: On the left, an angular partitioning descriptor. On the right, a histogram of orientations descriptor.

Combined Descriptor (CD)

To take advantage of both descriptors discussed previously, we propose to use a combined descriptor. This is a descriptor composed of two parts. The first one corresponds to the *Keyshape Angular Partitioning Descriptor* (KASD) and the second to the *Histogram of Keyshape Orientations* (HKO) descriptor. A schematic example of the combined descriptor is depicted in Figure 5.13. In this case, we reduce the number of slices for the KASD representation to 4. Thus, the combined descriptor is a 56-size vector, where the first 24 bins correspond to the KASD descriptor and the last 32 bins correspond to the HKO descriptor. As with both descriptors discussed previously, we use the unitary representation of the combined descriptor.

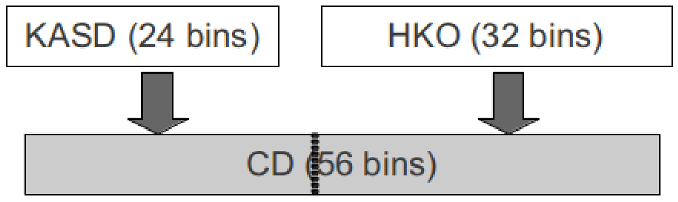


Figure 5.13: Scheme of the combined descriptor.

5.2.3 Matching

As seen in previous sections, both a test image and an input sketch are represented by a set of local descriptors whose size depends on the complexity of the underlying image (for instance, see the images presented in Figure 5.10). The next step is to find a way to map descriptors from an input sketch to descriptors computed from a test image in order to get a similarity or dissimilarity score. This score will allow us to rank the test images in an increasing order with respect to the similarity score or in an decreasing order with respect to the dissimilarity score.

Let S be an input sketch and $LD(S)$ be the set of local descriptors of S. We define $LD(S)$

as:

$$LD(S) = \bigcup_{t \in \{v,h,\mathrm{d}_1,\mathrm{d}_{-1},a,\mathrm{e}\}} LD_t(S) \tag{5.12}$$

where $LD_v(S)$ is the set of local descriptors of vertical lines in S. In the same way, $LD_h(S)$ is set for horizontal lines, $LD_{\mathrm{d}_1}(S)$ is set for diagonal lines having slope 1, $LD_{\mathrm{d}_{-1}}(S)$ is set for diagonal lines having slope -1, $LD_a(S)$ is set for arcs, and $LD_{\mathrm{e}}(S)$ is set for ellipses.

Further, let I be a test image that will be compared with S. Similarly as the case of S we define $LD(I)$ as:

$$LD(I) = \bigcup_{t \in v,h,\mathrm{d}_1,\mathrm{d}_{-1},a,\mathrm{e}} LD_t(I) \tag{5.13}$$

The matching process is performed between local descriptors corresponding to the same keyshape type. Since the number of local descriptors is much lower than in the case of having keypoints like those used by the SIFT or the *shape context* approach [59, 9], we could solve an instance of the *bipartite graph problem* using the well known *Hungarian method* [54] between $LD_t(S)$ and $LD_t(I)$, $t = h, v, \mathrm{d}_1, \mathrm{d}_{-1}, a, \mathrm{e}$. The final match results from the union of the partial matches.

Specifically, the Hungarian method solves an instance of the *assignment problem* where, given two sets A,B, the goal is to assign objects of B to objects of A. This produces a one-to-one relationship between A and B. Moreover, an assignment between two objects produces a cost known as the assignment cost. Thus, the objective is to set an appropriate assignment between objects of A and objects of B minimizing the total assignment cost.

Coming back to our case, A is the set of local descriptors of S ($LD(S)$), and B is the set of local descriptors of I ($LD(I)$). The objects are unitary vectors which represent local descriptors. Finally, the cost function we use is the Manhattan distance.

Let $M(S, I)$ be the resulting match between S and I from the previous process, defined as follows:

$$\begin{aligned} M(S,I) \quad = \quad & \{(\mathrm{i}_s, j_I, c) | (\mathrm{i}_s, j_I, c) \text{ is a match} \\ & \text{whose cost is } c, \wedge \mathrm{i}_s \in LD(S) \wedge \\ & j_I \in LD(I)\}, \end{aligned} \tag{5.14}$$

we define the following properties on M:

- $|M(S, I))|$: The cardinality of M, i.e the number if matches between S and I.
- $A(M(S, I))$: The average match cost defined as:

$$A(S,I) = \sum_{\forall(\mathrm{i},j,c) \in M} \frac{c}{|M(S,I)|}. \tag{5.15}$$

  In the case that no matches occur, $A(S, I) = max(LD(S), LD(I))$.
- $U(M(S, I))$: The number of unmatched descriptors. It is defined as follows:

$$U(S,I) = max(LD(S), LD(I)) - |M(S,I)| \tag{5.16}$$

Match Filtering

The matching is followed by a filtering step that aims to discard matches discordant with a pose estimation. To this end, we apply a vote based approach that takes into account spatial information from the corresponding matched keyshapes in order to estimate a geometric transformation. This transformation considers the scale and location parameters of the computed keyshapes. In this way, each match vote for a certain transformation, the predominant transformation will correspond to the pose estimation. Only the matches that voted for such a pose are held, the remaining matches are discarded. In Algorithm 6 we show how to filter a set of matches.

```
Algorithm 6 Match Filtering
Require: $M$: the set of matches.
Require: $K_S$: The set of keyshapes of the input sketch.
Require: $K_I$: The set of keyshapes of the test image.
  for $\forall m = (\mathrm{i}_S, j_I, c) \in M$ do
    $k_S \leftarrow$ the keyshape of $\mathrm{i}_S$
    $k_I \leftarrow$ the keyshape of $j_I$
    $(x_S, y_S) \leftarrow center\_of(k_S)$
    $(x_I, y_I) \leftarrow center\_of(k_I)$
    $s_S \leftarrow scale(k_S)$
    $s_I \leftarrow scale(k_I)$
    $\mathrm{d}_x = x_I - x_S \frac{s_I}{s_S}$
    $\mathrm{d}_y = y_I - y_S \frac{s_I}{s_S}$
    $q_x = quantize(\mathrm{d}_x)$
    $q_y = quantize(\mathrm{d}_y)$
    $matches(q_x, q_y) \leftarrow matches(q_x, q_y) \bigcup m$

  end for
  $the\_matches \leftarrow$ the set of matches in $matches$ with the major number of elements.
  return $the\_matches$
```

In Algorithm 6, $scale(k)$ gives the scale of a keyshape k. The scale for lines is simply the length of the underlying line, the scale for arcs is the associated radius, and the scale for ellipses is the corresponding major radius. In addition, for the quantization step we divide both the x-dimension and the y-di mension by a factor 3.

Dissimilarity Score

After the match filtering step we need to get a score value that represents the similarity or dissimilarity between both images. To this end, we define three dissimilarity functions based on the average match cost, the number of matches, and the number of unmatched descriptors.

1. $D_1(S, I) = A(S, I) + U(S, I)$.
2. $D_2(S, I) = 1/|M(S, I)|$.

3. $D_3(S, I) = A(S, I)/|M(S, I)|$.

The dissimilarity function can be regarded as a cost function. Thus, D_1 and D_3 are based on the average match cost $A(\cdot, \cdot)$. Furthermore, the number of unmatched descriptors $U(\cdot, \cdot)$ also can express a cost function. Thus, as the value of $U(\cdot, \cdot)$ increases, the resemblance level between the comparing images decreases. In contrast, as the number of matches $|M(\cdot, \cdot)|$ increases, the resemblance level between both images also increases. Therefore, we also use the number of matches in D_2 and D_3, but inversely.

To exploit the benefits of all dissimilarity functions defined above, we compute different rankings combining local descriptor with dissimilarity functions. Thus, we define $r^S_{(D,F)}$ as the resulting ranking for an input sketch S, where D indicates one of the local descriptors (KASD, HKO, or CD) and F is one of the dissimilarity functions (D_1, D_2, or D_3) that allows us to sort the test images.

Let $rank(r, \Gamma)$ be a function that provides the position where a test image Γ appears in the ranking r. For computing the final rank, we need to determine a new score (the final score) for each test image. This score is computed as follows:

$$final\_score(\Gamma) = \sum_{\forall r} rank(r, \Gamma). \tag{5.17}$$

The final ranking then is formed in an increasing order with respect to the new scores. Images with low scores must appear first in the final ranking. Therefore, if an image appears in the top of all rankings, it also will appear in the top of the combined ranking.

In order to get the rankings, we propose to use the followings:

- $r_{(CD,D_2)}$: Ranking produced by the combined descriptor and the dissimilarity function D_2.
- $r_{(KASD_4,D_3)}$: Ranking produced by the KASD descriptor using 4 partitions and the dissimilarity function D_3.
- $r_{(KASD_8,D_1)}$: Ranking produced by the KASD descriptor using 8 partitions and the dissimilarity function D_1.

5.3 Computational Complexity

In this section, we discuss the computation complexity of our approach. For the sake of clarity, we divide the discussion of the computational complexity of our algorithms in three groups. First, we discuss the complexity of the detection keyshape stage. Second, we discuss the complexity of computing the proposed descriptors. Finally, we discuss the complexity of comparing a test image with an input sketch.

5.3.1 Keyshape Detection

This stage involve many algorithms including pre-processing tasks, *edgelink* detection, inflection point detection and the keyshape classification.

- Pre-processing: This task consists in analyzing each pixel of an image to obtain an edge map representation. The complexity of this task is $O(N)$ where N is the number of pixels of the underlaying image.
- EdgeLink: In this stage, we trace the edge map pixels to determine potential strokes. Therefore, the complexity of this algorithm is $O(E)$, where E is the number of edge pixels.
- Set of Maximum Deviation Points: To get the *Set of Maximum deviation* (SoM), the algorithm receives a set of strokes and, by approximating straight lines, determines points of maximum deviation with respect to those lines. Tho this end, the algorithm has to evaluate a number of points proportional to the size of E. Therefore, the complexity of *getSoM* is $O(E)$.
- Inflection Points: To determine inflection points the algorithm evaluates each maximum deviation point together with a local region of 25×25 pixels. As the size of SoM is less than the size of E, the complexity of this algorithm is also bounded by $O(E)$.
- Detection of Keyshapes: In this stage, each stroke piece is evaluated to determine what keyshape the underlying stroke piece corresponds to. However the algorithm for classifying arcs and lines may have a cost proportional to the number of straight lines that approximate a stroke piece. Therefore, this stage has a complexity of $O(NL)$, where NL is the number of approximating straight lines, which is less that the size of E.

Therefore, the complexity of detecting keyshapes is $O(N + E)$. However, since $N > E$, the complexity results to be linear with respect to the image size.

5.3.2 Keyshape Descriptors

We have presented two descriptors. The first one, KASD, processes an image in time $O(K^2)$, where K is the number of keyshapes. The second approach, HKO, runs in time $O(K \times W)$, where W is the size of the local regions around a descriptor is computed.

5.3.3 Keyshape Matching

We use the Hungarian method to find a set of matches between two sets of keyshapes. This algorithm runs in cubic time with respect to the size of the input [102]. As our input is a set of keyshapes, the time of the algorithms runs in $O(K^3)$, where K is the number of keyshapes . In practical issues, the average number of keyshapes computed in our data set is approximately 30, which requires low computational cost in terms of matching. Experimentally, the reported time for comparing two sets of keyshapes is approximately 8 ms.

In conclusion, the total cost for comparing two images is $O(N + K \times W + K^3)$, where N is the number of pixels in the image, K is the number of keyshapes and W is the size of the local region used for computing local descriptors.

5.4 Experimental Evaluation

Due to sketch-based image retrieval being a young research area, there is not a standard benchmark for comparison purposes. However, Eitz et al. [31] recently have conducted an interesting systematic work to propose a benchmark in this area. We choose this benchmark because this takes into account the user's opinion. This fact turns very important as the main goal of a retrieval system is, precisely, to calculate a ranking very close to what the user is expecting to get.

This benchmark consists of 31 query sketches, each one associated with a set of 40 test images. An example of two query sketches and seven of their associated test images in the data set are shown in Figure 5.14.

Furthermore, each test image has been ranked by people using a 7-point Likert scale [31] with 1 representing the best rank to 7 representing the worst rank. This provides the baseline ranking which is called the *user ranking.*

We proceed to compare the ranking of the proposal method against the user ranking. To this end, Eitz el al. propose to use the *Kendall's correlation* τ that determines how similar two rankings are. The correlation coefficient τ may vary from 1 to -1, with -1 indicating that one ranking is the reverse of the other, 0 indicating independence between the rankings, and 1 indicating that the two rankings have the same order. Therefore, as we achieve a correlation value near 1, our proposal will be better, indicating that the resulting ranking is very similar to that expected for the users.

Considering that both the user ranking and the ranking of a proposal method may include tied scores, a variation of the *Kendall's correlation* is used. This variation is denoted by τ_b and it is defined as:

$$\tau_b = \frac{n_c - n_\mathrm{d}}{\sqrt{(n_0 - n_1)(n_0 - n_2)}}, \tag{5.18}$$

where

- n_c = Number of concordant pairs.
- n_d= Number of discordant pairs.
- n_0= $n(n-1)/2$.
- n_1= $\sum_{\mathrm{i}=1}^{t} t_\mathrm{i}(t_\mathrm{i}-1)/2$.
- n_2= $\sum_{\mathrm{i}=1}^{u} [u_\mathrm{i}(u_\mathrm{i}-1)/2$.
- t_i=Number of tied values in the ith group of ties for the user ranking.
- u_i=Number of tied values in the ith group of ties for the proposed ranking.
- t = Number of groups of ties for the user ranking.

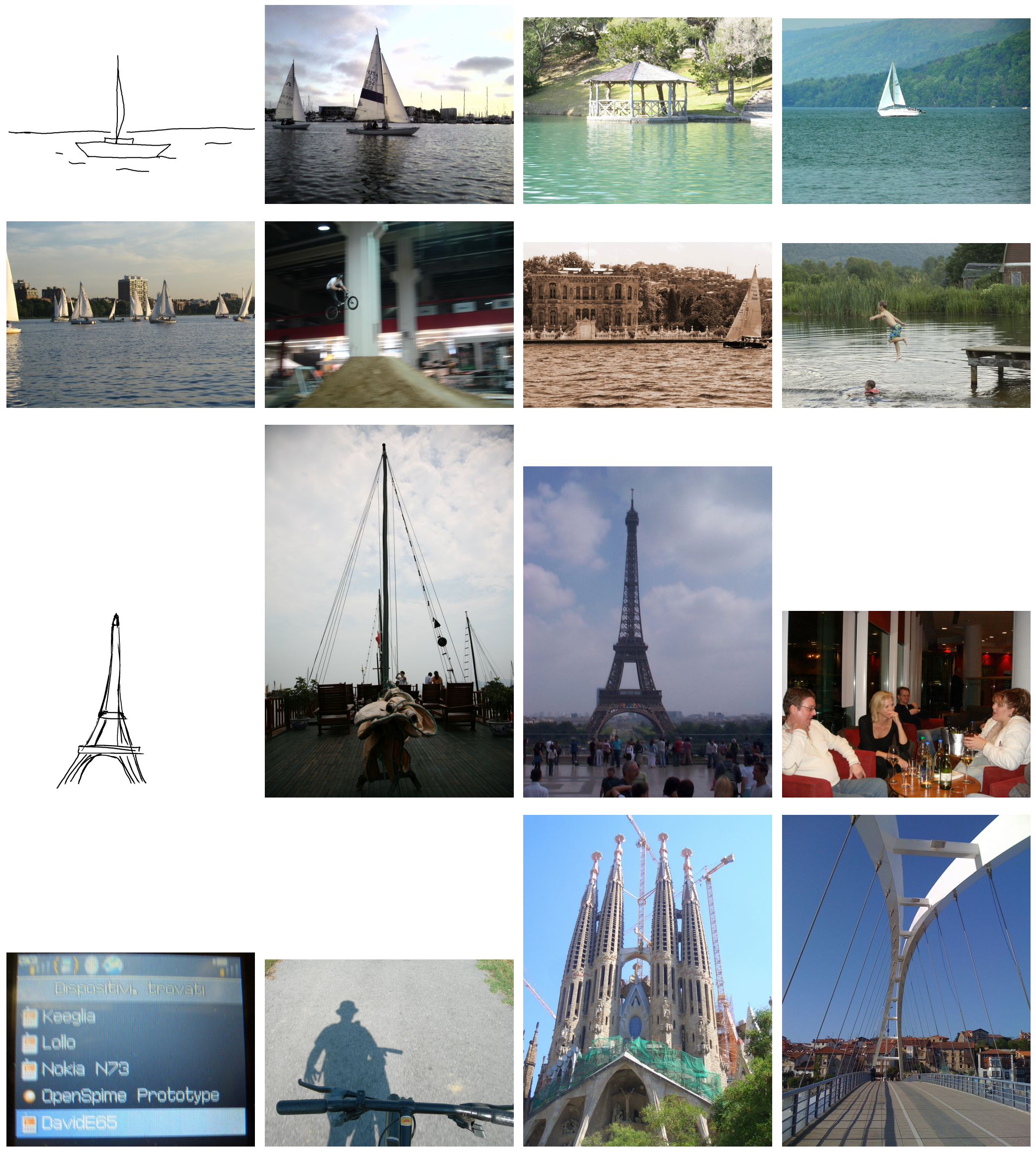


Figure 5.14: Two query sketches.

- u=Number of groups of ties for the proposed ranking.

In the experiments conducted by Eitz et al. [31], the best correlation value they achieved was 0.277. This result was obtained using a Bag of Features methodology with a variant of the histogram of gradient descriptor. The approach uses a 1000-size codebook, thas means that the codebook is formed with 1000 codewords. In addition, a correlation value under 0.18 is reported for the Shape Context method [9].

Our results show a slight increment in the effectiveness of the retrieving process without requiring a tedious learning stage. We achieve a correlation value of 0.289. A graphic showing the difference correlation value for different SBIR methods is depicted in Figure 5.15. Considering that our proposal is based on the structural feature of a sketch representation, which has not been taked into account for current methods, our method is potentially adequate for being combined with a leading method to increase the retrieval effectiveness.

To show the goodness of combining our method with a current leading method, we show, in this document, the results of combining our proposal with the BoF approach proposed by Eitz et al. [31]. For combination, we follow the same approach we use to combine our two proposed descriptors. In Figure 5.16 we show that the combined method (Keyshape+BoF) allows us to achieve a correlation value of 0.337 which means an increase of almost 22% with respect to the correlation value achieved by the BoF approach. To validate these results, we run a statistical test (T-Test) comparing the results produced by the BoF approach and the results achieved by our combined proposal. The statistical test gave a p_value of 0.1% which indicates that our combined method improves significantly the BoF approach.

Furthermore, we present in Figure 5.17 and 5.18 the correlation value achieved for each sketch using the BoF approach, our keyshape-based proposal and the combined method. We note that although our approach gains a significant improvement for some queries (for instance, see query 25 and 22), the overall improvement is not significant enough. However, when we analyze the correlation achieved by the combined method, we see that it achieves an improvement in 24 queries with respect to the BoF results. Additionally, the overall effectiveness is improved in almost 22%.

Finally, it is important to note that it is possible that a variation of the image may produce a different set of keyshapes. Specifically, the detection of arcs and lines may be affected. However, as our method is characterized by many keyshapes where straight lines are the most frequent type of keyshapes, we will still have a good number of keyshapes for the matching purpose.

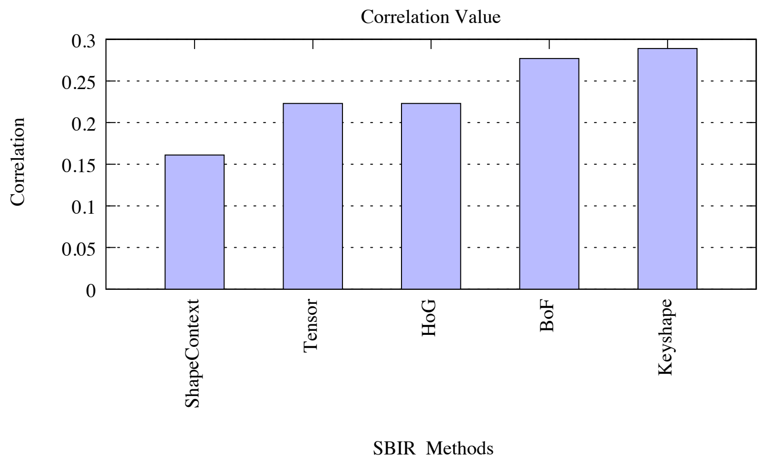


Figure 5.15: Kendall's correlations for SBIR methods. Our proposal (the last bar on the right) outperforms that of the state of the art.

5.5 Summary

In this chapter we have presented a novel local method for sketch based image retrieval. Our method is based on detecting simple shapes called *keyshapes*. This allows us to get structural representation of images leading to an improvement of retrieval effectiveness. Our proposal improve the results of the state of the art methods achieving a correlation value of 0.289.

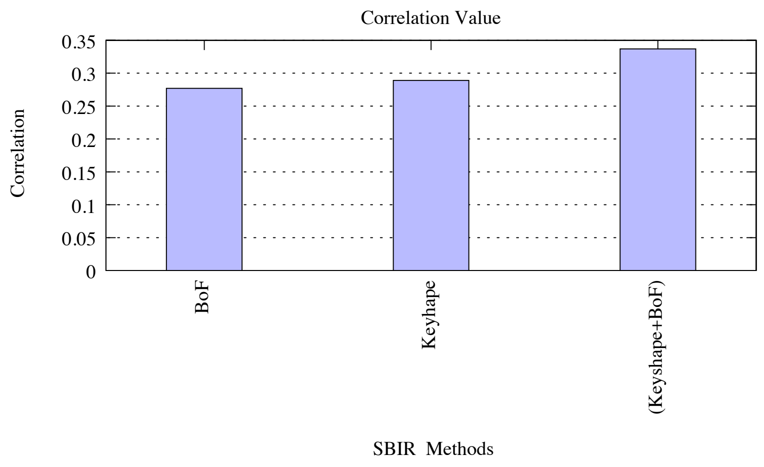


Figure 5.16: Kendall's correlations for the approaches: BoF, Keyshape-based, and the Keyshape+BoF.

Furthermore, we analyze a combined proposal exploiting the results of the BoF approach and the results of our keyshape based approach showing that our method is complementary to the BoF approach. This fact is reflected by the results achieved by the combination of both methods. The results shown that using a combined method allows us to increase the retrieval effectiveness in almost 22%. Additionally, we shown that this result is significantly better than that of the state-of-the-art methods.

In addition, we have presented an efficient method for detecting simple shapes on an image. This strategy could be applied for other applications requiring a reduction of the image complexity.

Of course, sketch based image retrieval is still a challenging task. In this vein, our current work is focused on defining a more efficient metric for comparing keyshapes.

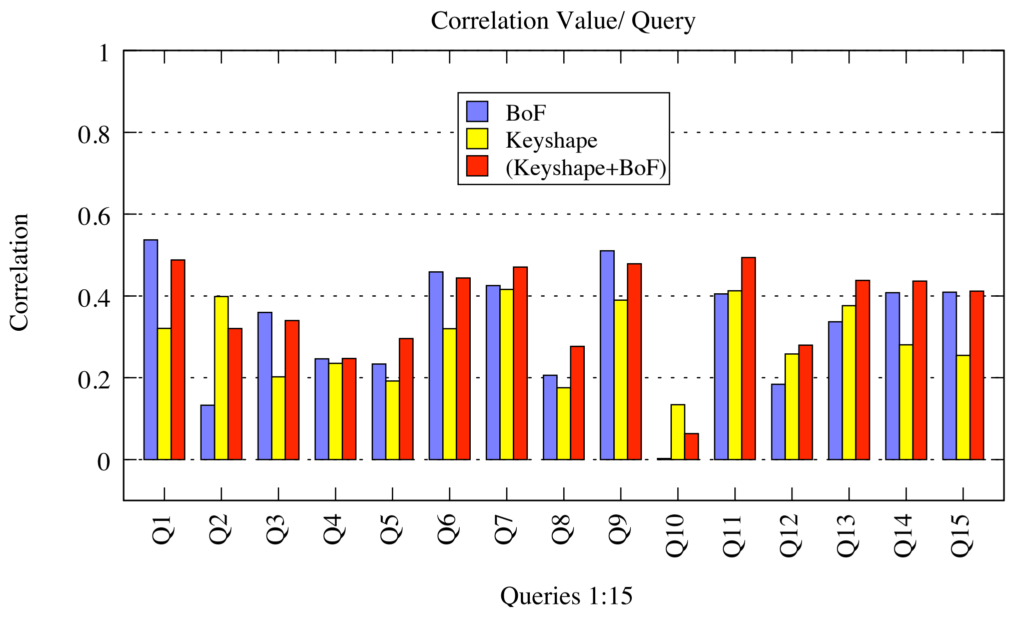


Figure 5.17: Correlation values for the first 15 queries.

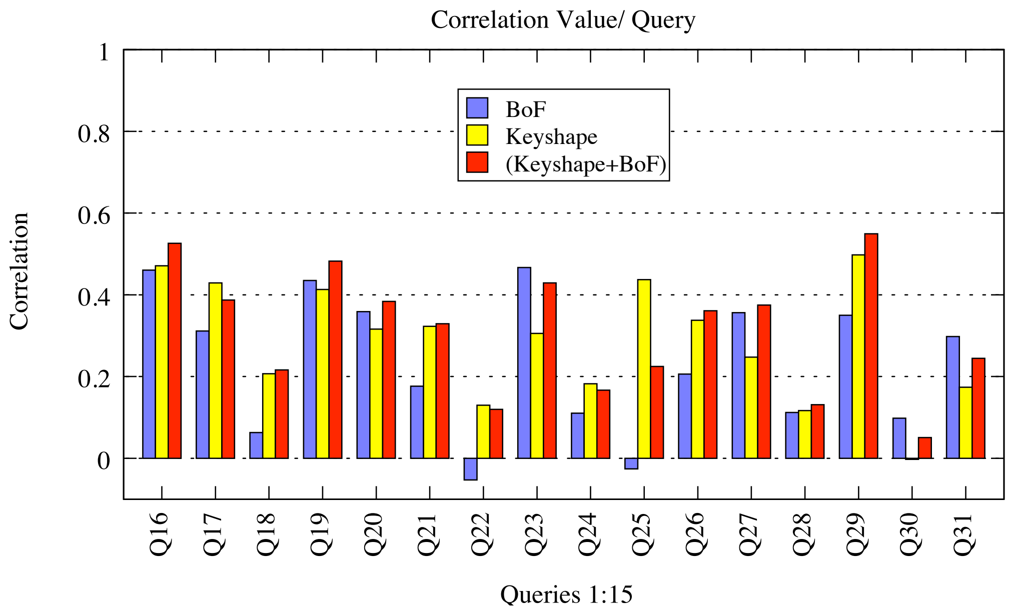


Figure 5.18: Correlation values for the last 16 queries.

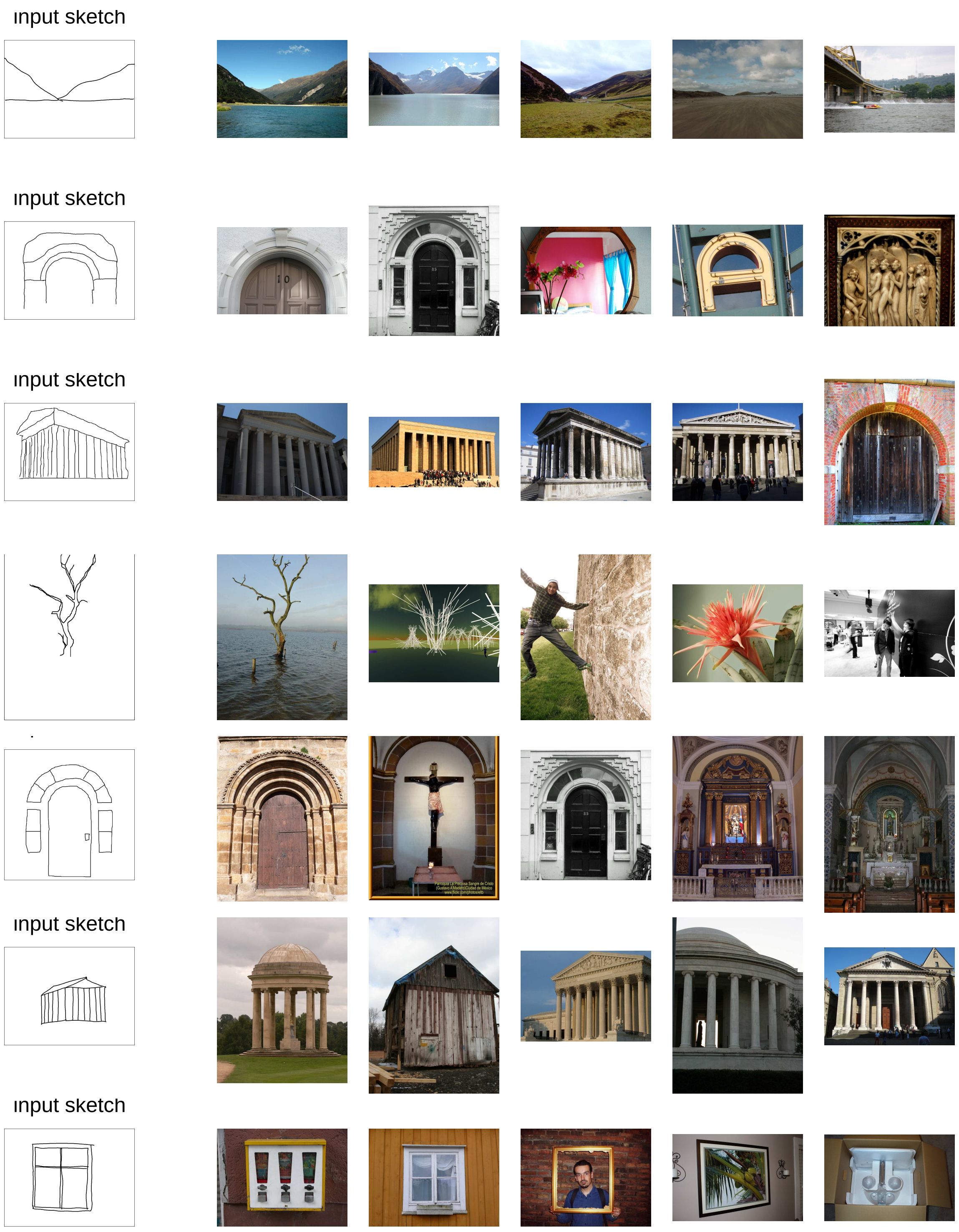


Figure 5.19: Example of SBIR using our proposal.

Chapter 6

Sketch Based 3D Model Retrieval

6.1 Introduction

In the last years there has been a wide interest and progress on computer aided retrieval of multimedia data. The advances in this area have allowed users to look for a multimedia object in large repositories in a more efficient way. As advances in multimedia retrieval increase, new interesting challenging applications are coming up. One of the current interesting applications is the 3D model retrieval with impact extending from design to medical issues [15].

The simplest way for retrieving 3D models is by means of textual metadata describing the 3D objects. This requires 3D models to have reliable metadata. However, 3D models not always come with reliable human tags. Although many authors have addressed the multimedia data annotation problem, this is still considered as an open problem [100, 101]. Due to this fact, the ongoing research on multimedia retrieval relies on a content-based approach [25].

In the context of content-based 3D model retrieval, several approaches for computing similarity between two 3D models have been proposed [95, 15]. Among these methods are *shape histogram* [1], *shape distribution* [75], *moments* [33], *light field* [19] or *spherical harmonics* [37]. Following any of these approaches, users require a 3D model as an example for querying.

However, a 3D model example as query is not always available, frequently requiring some kind of technical expertise to produce it. Even though some tools for making the 3D modeling task easier for any kind of users (e.g. Google Sketchup) are coming up, they are still difficult to operate and produce detailed models on the fly. This fact clearly limits the 3D model retrieval usability.

An easy alternative for querying is simply drawing a 2D stroke-based sketch lacking of color and texture. A simple example of a user-drawn sketch is shown in Figure 6.1. Although such a kind of sketch is composed of few strokes, it still keeps enough structural information representing what the user is looking for. This kind of querying leads to *the sketch-based 3D model retrieval.*

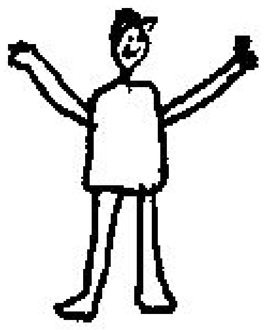


Figure 6.1: An example of a user-drawn sketch.

In this work we are interested in rough sketches that novice users can draw easily. One important issue here is how to compare a 3D model with a 2D sketch. To this end, some strategies project and render the model from different viewpoints getting 2D representations [105]. They then process these 2D representations as in the context of sketch-based image retrieval using, for instance, HOG [24] or HELO descriptors (see Chapter 4).

An interesting technique for getting a 2D representation from a 3D model is using *suggestive contour images* [27]. This technique applied in the context of non-photorealistic rendering resembles hand drawings of 3D dimensional objects very closely. Yoon et al. [105] showed that this technique performs better than the classical *contour* or *ridge and valley* techniques for retrieval tasks. Therefore, our proposal use *suggestive contours* to get 2D sketch representations from 3D models.

Since a sketch image provides structural information of the underlying objects, our proposed descriptors are based on extracting structural components named *keyshapes* similar as in the case of image retrieval (see Chapter 5).

Our contribution in this chapter is to present two novel approaches for the sketch-based 3D model retrieval problem which are extended from our keyshape based method discussed in Chapter 5. The first approach named STELA, derived from *StrucTurE based Local Approach*, is a local technique that leverages the structural information provided by sketch strokes. Furthermore, in this approach, a variation of the HELO descriptor is used as a filtering step to reduce the database size. The second proposal named HKO-KASD as acronym of *Histogram of Keyshape Orientations - Keyshape Angular Spatial Descriptor* also exploits the structural information from sketch representation, but different from STELA, it computes a local descriptor composed of the local distribution of four types of strokes. In addition, for the filtering step, this proposal applies a global technique based on the distribution of keyshape orientations.

The results of our proposal show an increase in accuracy for many classes of 3D models with respect to current strategies applied for sketch-based 3D model retrieval.

6.2 Related Work

Sketch-based retrieval is a young and challenging area not only in the case of 3D model retrieval but also in image retrieval systems. In the case of image retrieval the proposals of Eitz et al. [31] and our proposal presented in Chapter 5 are the most representatives. While for 3D model retrieval using sketches as queries we highlight the approaches of Yoon et at. [105] that is based on representing a 3D model by means of a set of suggestive contour images.

The general idea for comparing a sketch (a 2D object) against 3D models is representing each 3D model by a set of 2D projection images. To this end, contour or ridge and valley representations may be used. However, Yoon et al. [105] showed that a *suggestive contour*-based representation outperforms the performance of the contour-based or ridge and valley-based representations. In particular, Yoon et al. propose the HOG-DFT approach, which uses *suggestive contour* images [27] as projections of 3D models from fourteen different viewpoints. For the matching problem they form histograms of orientations using a *diffusion tensor field* based approach. Since the histogram of orientations is a global technique, it could get confused easily. That is, two sketches representing different objects may have similar global descriptors. For instance, in Figure 6.2, we show a sketch and a suggestive contour of a 3D model with their corresponding global descriptor (in this case we are using the HELO descriptor). Both descriptors look very similar even though they are representing different objects. Moreover, a global descriptor does not take into account structural information which is the main information represented by sketches.

Following the proposal of Yoon et al., we also use suggestive contour images to get a sketch representation from a 3D model for both of our keyshape based proposal (STELA and HKO-KASD). In the following sections we describe both proposals in detail.

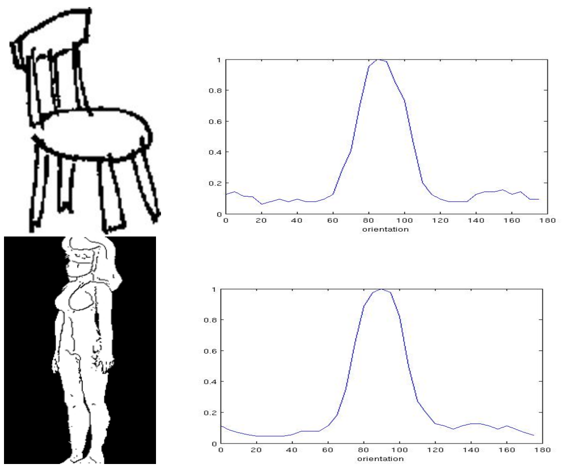


Figure 6.2: The behavior of an oriented-based global approach for comparing an sketch with a 3D model suggestive contour.

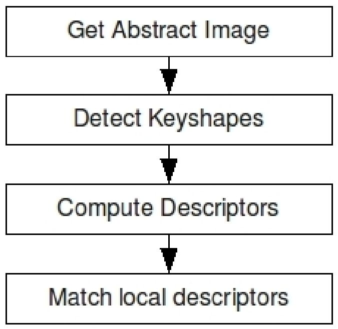


Figure 6.3: The pipeline of the proposed local approach.

6.3 The STELA Approach

In this section we describe STELA in detail for the sketch based 3D model retrieval. It is necessary to be aware that this method requires 2D sketch-like representations. The query sketch is already a sketch representation, but a 3D model is not. To obtain a 2D sketch representation from a 3D model we use the suggestive contours technique proposed by DeCarlo et al. [27]. The main property of STELA is that it is the first method that takes advantage of the structural information as well as of the locality information over the sketches representations.

For getting structural information, we follow the basic ideas of our proposal for the sketch based image retrieval, in which a sketch image is represented by a set of *keyshapes*, but instead of detecting arcs and ellipses, STELA works detecting only straight lines. In this way, our proposal is composed of the following steps: (1) *get an abstract image*, representing in a simpler way a sketch or suggestive contour image. (2) *detect keyshapes*, that should return a set of simple primitive shapes from an abstract image, (3) *compute local descriptors*, that computes a fingerprint for each *keyshape*, and (4) *matching*, allowing us to set a correspondence mapping between local descriptors of a query sketch and local descriptors of a suggestive contour image representing a 3D model from a certain viewpoint. These four steps are shown in Figure 6.3. The next subsections describe each step in detail.

6.3.1 Abstract Image

The sketches and the suggestive contour images obtained from the 3D models are 2D images. These will be represented by a set of *keyshapes*. Before detecting *keyshapes*, we pre-process the images using thinning operator [41] for sketches and Canny operator [16] for suggestive contour images. In addition, we apply a cropping operation keeping only the bounding box surrounding the object on the image.

Let I be a sketch or suggestive contour image after the preprocessing step. We represent I by a set of strokes $I = \{S_1, S_2, \ldots, S_{N_s}\}$. We approach real strokes by means of *edgelinks* using the same algorithm discussed in the Chapter 5. In Figure 6.4 we show an example of a sequence of *edgelinks*. Henceforth an *edgelink* will be called simply **stroke**.

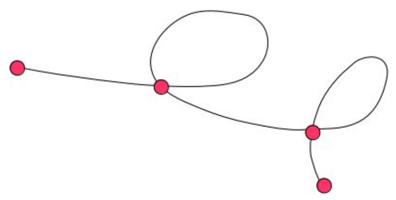


Figure 6.4: A synthetic example of a sequence of *edgelinks*.

After representing an image I as a set of strokes, we proceed to detect *keyshapes*. In this step, we may define many kinds of *keyshapes* like circles, arcs, ellipses, lines, etc. However, due to the complexity of a sketch image, drawn in a rough way together with the scarce information provided by 3D model projections, circles, arcs and other shapes could not be detected appropriately causing a decrease in efficiency. In this way, we only consider straight lines as *keyshapes*. Although lines are simple shapes, these still keep enough structural information which will be exploited by our descriptors. Event though we will use only straight lines as *keyshape* we decide to keep the term *keyshape* because it could be extended to other shapes in other contexts.

6.3.2 Detecting Keyshapes

For getting *keyshapes* (or *keylines*), we take each stroke S to be approximated by a set of straight lines. First, we define a line l between the start and end point of S. Second, we calculate the maximum deviation τ from a point on S to l. If $\tau > TH$, S is divided into two lines l_1 and l_2, taking the maximum deviation point on S as division point (see Figure 6.5) . We proceed processing l_1 and l_2 in the same way leading to a recursive procedure. In addition, the detected lines are split if they are longer than a threshold. We define the minimum line length as 20 per cent of the image size.

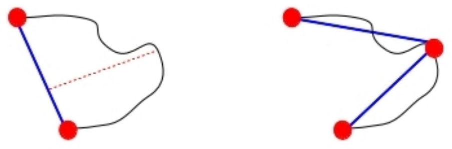


Figure 6.5: Approximating a stroke by a set of straight lines.

Finally, a stroke S of I is decomposed into a set of lines. Therefore, image I can be represented by the set of lines, each one corresponding to a stroke. Now we can define $I = l_1, l_2, \ldots l_n$, where n is the total number of detected lines or *keyshapes*.

The resulting set of lines contains lines of different size. It is worth pointing out that curve strokes will yield a set of very small lines. So, to get a set of lines representing appropriately straight strokes we need a threshold T_{short} to reject small lines. In addition, considering the possible discontinuities of strokes on the sketches, a process for merging nearby lines with similar slope is required. We use a threshold T_{near} to evaluate nearness between lines. Furthermore, lines with length above a threshold T_{large} should be split into smaller ones. We propose to use the following threshold values: $T_{short} = D * 0.05$, $T_{near} = 5$, $T_{large} = D * 0.5$,

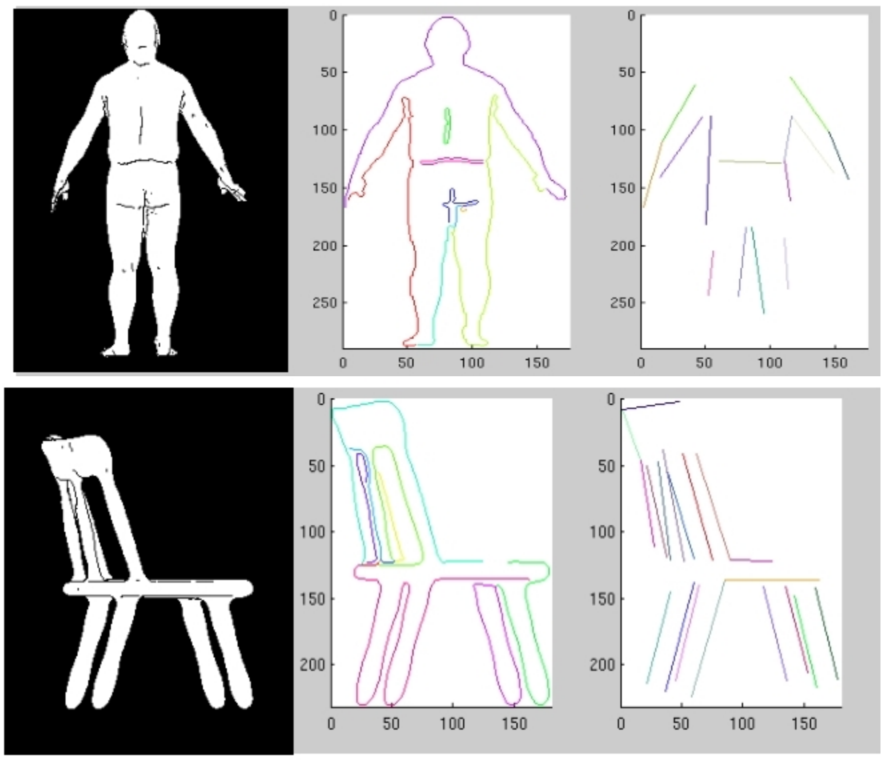


Figure 6.6: First column shows the suggestive contour of two 3D models, second column shows the corresponding abstract images, and third column shows the detected *keyshapes*.

where D is the length of the abstract image diagonal. The final set of lines represent the set of *keyshapes*.

Finally, we regard the center of each line as the representative point of each *keyshape*. Figure 6.6 shows two 3D models, represented by one of their suggestive contours, together with their corresponding abstract representations (second image) and the detected *keyshapes* (third image). In our proposal, each *keyshape* L is represented as a 5-tuple $[(x_1, y_1), (x_2, y_2), (x_c, y_c), s, \phi]$, where (x_1, y_1) is the start point, (x_2, y_2) is the end point, (x_c, y_c) is the representative point, s is the line length, and ϕ is the slope. Although the two first components representing L are enough to compute the remaining three components, we decided to keep the 5-tuple notation just for making our algorithm easy to be understood.

6.3.3 The Local Descriptor

Different descriptors could be used in this step. For instance, an extension of Shape Context [9] to work over keyshapes instead of working over a point sampling is an alternative. However, this choice could yield a sparse descriptor considering that the number of keyshapes is much smaller than the number of sampled points.

Therefore, in our approach we use an *oriented angular 8-partitioning descriptor*. Figure 6.7 depicts a graphical representation of this descriptor.

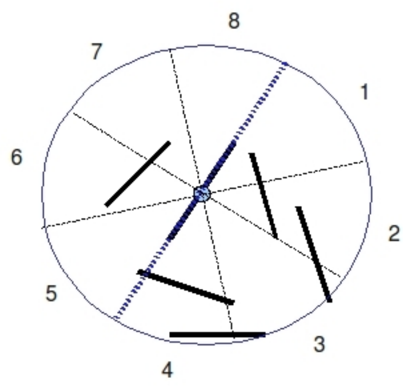


Figure 6.7: A synthetic representation of the partitioning to make up the proposed local descriptor.

Having a keyshape L as reference, this descriptor works as follows:

- Create a vector h, containing 8 cells. Initially, $h(\mathrm{i}) = 0$, $\mathrm{i} = 1 \ldots 8$.
- Let L be the reference keyshape represented as :

$$L = [(x_1, y_1), (x_2, y_2), (x_c, y_c), s, \phi']. \tag{6.1}$$

- Let $f_r : \mathbf{R}^2 \rightarrow \mathbf{R}^2$ be a rotation function around the point (x_1, y_1) with rotation angle $\beta = -\phi$. This function is defined as below:

$$\begin{aligned} f_r(x, y) &= (x_r, y_r), \text{where,} \\ x_r &= [(x - x_1)cos(\beta) - (y_1 - y)sin(\beta)] + x_1 \\ y_r &= y_1 - [(x - x_1)sin(\beta) - (y_1 - y)cos(\beta)] \end{aligned} \tag{6.2}$$

- Let $(\hat{x}_c, \hat{y}_c) = f_r(x_c, y_c)$ be the normalized version of (x_c, y_c).
- For each keyshape $Q \neq L$ represented by $[(x'_1, y'_1)$, (x'_2, y'_2), $(x'_c, y'_c), s', \phi']$.
  - Get $(\hat{x}'_c, \hat{y}'_c) = f_r(x'_c, y'_c)$.
  - $D_x = \hat{x}'_c - \hat{x}_c$ and $D_y = \hat{y}'_c - \hat{y}_c$
  - If $D_x > 0 \wedge D_y > 0 \wedge |D_x| > |D_y|$, $bin = 1$.
  - If $D_x > 0 \wedge D_y > 0 \wedge |D_x| <= |D_y|$, $bin = 2$.
  - If $D_x \leq 0 \wedge D_y > 0 \wedge |D_x| <= |D_y|$, $bin = 3$.
  - If $D_x \leq 0 \wedge D_y > 0 \wedge |D_x| > |D_y|$, $bin = 4$.
  - If $D_x \leq 0 \wedge D_y \leq 0 \wedge |D_x| > |D_y|$, $bin = 5$.
  - If $D_x \leq 0 \wedge D_y \leq 0 \wedge |D_x| <= |D_y|$, $bin = 6$.
  - If $D_x > 0 \wedge D_y \leq 0 \wedge |D_x| <= |D_y|$, $bin = 7$.
  - If $D_x > 0 \wedge D_y \leq 0 \wedge |D_x| > |D_y|$, $bin = 8$.
  - $h(bin) = h(bin) + s'/MAX\_LEN$, where MAX_LEN is the length of the abstract image diagonal. This is a normalization parameter depending on the image size.
- Finally, $h(bin) = \dfrac{h(bin)}{\sum_{\mathrm{i}=1}^{8} h(\mathrm{i})}$, $bin = 1 \ldots 8$.

6.3.4 Matching

In our approach, we treat an object, a query sketch or suggestive contour image, as a set of descriptors. This set captures the object shape.

Let $P = \{p_1, p_2, \ldots, p_m\}$ be the set of descriptors representing a query sketch, and $Q = \{q_1, q_2, \cdots q_n\}$ be the set of descriptors representing a suggestive contour obtained from a 3D model from a certain viewpoint. Here, $p_{\mathrm{i}}, q_j \in \mathbf{R}^8$. Without loss of generality, we will suppose $n < m$. So, we need to find an assignment from Q to P. This is, for each q_j we need to look for the p_{i} that allows us to minimize an overall cost. We define the function $\pi : \{1, \ldots, n\} \rightarrow \{1, \ldots, m\}$ that maps the j-th descriptor from Q with the i-th descriptor from P.

Furthermore, we define the cost T of the assignment using a certain mapping function π as follows:

$$T(\pi) = \sum_{\mathrm{i}=1}^{n} C(q_{\mathrm{i}}, p_{\pi(\mathrm{i})}) \tag{6.3}$$

where, $C(q,p)$ is the cost of matching a descriptor $q \in Q$ with $p \in P$. This cost function could be thought as the distance between p and q. In this way, the less similar the descriptors are, the more expensive the match become. As our proposal descriptor is, in fact, a probability distribution, we use the χ^2 test statistics:

$$C(q,p) = \frac{1}{2} \sum_{\mathrm{i}=1}^{8} \frac{[q(\mathrm{i}) - p(\mathrm{i})]^2}{q(\mathrm{i}) + p(\mathrm{i})} \tag{6.4}$$

Therefore, the problem of minimizing the overall cost is defined as:

$$\pi^\star = argmin\ T(\pi) \tag{6.5}$$

This problem may be regarded as an instance of the *Bipartite Graph Matching*. Different from the case of classical local methods for the image context, our number of descriptor per image is much lower. So, in our case we will resolve the assignment problem applying the Hungarian Method [54].

After the assignment stage, we need to look for a representative pose transformation between the matched descriptors. This will allow us to achieve a more consistent matching. To this end, we will use the stored information of the corresponding *keyshap*e (see Eq. 6.1). We are only interested in finding the scale and position transformation. The position is represented by the center of the keyshape (x_c, y_c) as the scale is represented by the keyshape length s.

For estimating the pose transformation, we use the Hough Transform [6], where each candidate match must vote just for three parameters (scale, translation in x-axis and in y-axis). We keep the set of parameter with the highest vote. This set of transformation parameters characterizes the estimated pose. Only the matches which agree with the estimated pose are retained for the next process, the others are discarded.

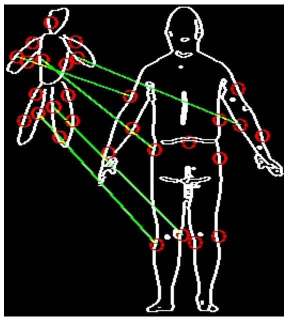


Figure 6.8: Matching between a sketch (top image) and a suggestive contour (bottom image).

Finally, the similarity between a sketch and a suggestive contour is computed as the average cost of the matched descriptors. The cost of the unmatched suggestive contour descriptors are set to 1. Figure 6.8 shows an example of this matching step.

6.3.5 Invariance issues

Our local approach is robust under positions, scale, and rotation changes. The translation invariance is directly derived as our descriptor extract local information. We achieve scale invariance normalizing the length of the keyshape by the MAX_{LEN}, a parameter depending on the image size. Finally, we get rotation invariance making the keyshape be coincident with the x-axis of the partitioning system as shown in Figure 6.7.

6.3.6 Filtering Step by HELO

Taking into account that local approaches are commonly expensive in time, we add a filtering step, where a reduced number of suggestive contour images are selected. In addition, this filtering step allows us to get an improvement in precision, reducing the number of false positives that could arise from the fact that we have 14 suggestive contour images for each 3D model. That means, having many viewpoints for each 3D model could make the method get confused during the retrieval process.

Each chosen suggestive contour represents a different 3D model. Our filter considers the global shape of the objects, therefore only 3D model with global shape similar to that of the query are kept. To this end, we use HELO [80] as the global descriptor with a slight variation.

In this approach, instead of applying HELO just over the whole image, we apply it over three kind of zones, leading to three types of HELO:

1. HELO : This is applied over the whole image, using a 36-size descriptor.
2. HELO_V: This is applied over four equal-sized vertical regions, juxtaposing each descriptor to make up the global one. In this case, since the HELO_V is applied over

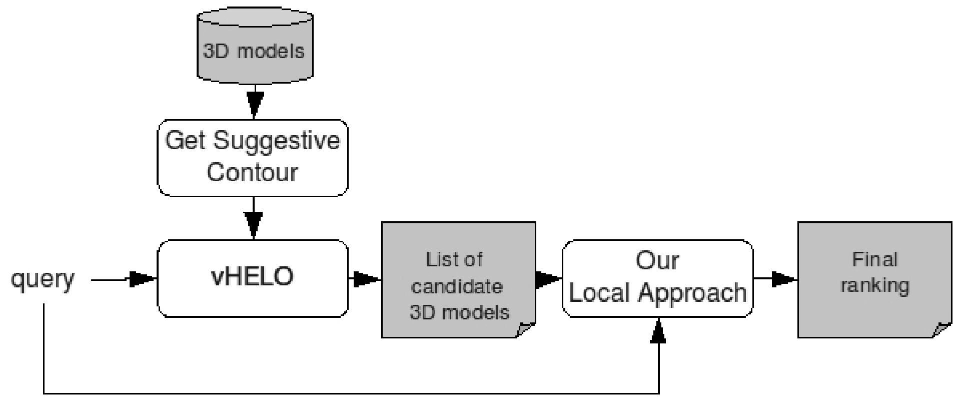


Figure 6.9: Combining a global approach with a local one for 3D model retrieval.

smaller regions, we use 18-size descriptors for each region, generating a 72-size global descriptor.

3. HELO_H: This is applied over four equal-sized horizontal regions, juxtaposing each descriptor to make up the global one. The size of this descriptor is the same as that of the HELO_V.

Each one of the three mentioned descriptors are evaluated separately using the χ^2 distance, producing 3 different distances, $\mathrm{d}_{HELO}, \mathrm{d}_{HELO_V}, \mathrm{d}_{HELO_H}$. The final distance D is computed as follows:

$$D = w_1 * \mathrm{d}_{HELO} + w_2 * \mathrm{d}_{HELO_V} + w_3 * \mathrm{d}_{HELO_H} \tag{6.6}$$

where w_i are appropriate weights such that $\sum_{\mathrm{i}=1}^{3} w_\mathrm{i} = 1$. Empirically, we set $w_1 = 0.2$, $w_2 = 0.4$, $w_3 = 0.4$. We call this global approach vHELO.

6.3.7 Handling viewpoint changes

In our approach, we project each 3D model from 14 different viewpoints [105] getting the corresponding 14 suggestive contours. The global dissimilarity between a query sketch and a 3D model is computed as the minimum distance between the input and the 14 corresponding suggestive contours.

After selecting the candidates using the global filter, we keep only one suggestive contour for each selected 3D model. The local approach explained in the previous section will give the final rank. A graphical representation of how our proposal works is illustrated in Figure 6.9.

6.4 The HKO-KASD Approach

Following the STELA method, in this approach we also get a 2D sketch representation from 3D models using fourteen suggestive contour images, each one corresponding to a specific viewpoint as specified by Yoon et al. [105]. Our approach comprises two stages. The first stage is a filtering step which determines the most appropriated projection for each 3D model having a query sketch as input. The filtering step is carried out by means of a global descriptor. Classical global descriptors are based on gradient orientations as HOG or HELO. We claim that gradient-based methods are sensitive to noise or outliers. Instead, we propose the *Histogram of Keyshape Orientations* (HKO), where orientations are estimated with respect to *keyshapes* that compose a sketch or a suggestive contour image.

After the filtering step we keep only one suggestive contour image for each 3D model of the database. The next stage is a refining step, which will determine the final ranking of the database objects with respect to the input sketch. As in the case of STELA, we are interesting in exploiting the structural information provided by sketches or contour images. In this way, we propose to use the *keyshapes* as structural componentes. In addition, to take into account locality information, we compute a local descriptor for each *keyshape*. We propose a new local descriptor representing the *keyshape* spatial distribution around a referent *keyshape*. The next sections are dedicated to describe in detail each of our mentioned stages.

6.4.1 Keyshapes

This approach follows the same flow of STELA but the local descriptor together with the matching step is different. In particular, an abstract representation from a sketch query or from a suggestive contour image is required. To this end, the same strategy applied for STELA is used. In addition, in this approach we decompose an sketch representation into a set of keyshapes based on detecting straight lines similar as STELA.

Once *keyshapes* have been detected, we classify them as horizontal line (H), vertical line (V), diagonal line with slope 1 (D_1), or diagonal line with slope -1 (D_2). In Figure 6.10 we displayed an example of a suggestive contour image with its corresponding *keyshape* representation.

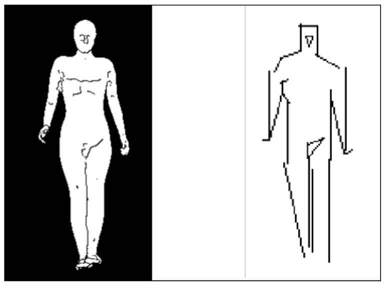


Figure 6.10: A suggestive contour image with its corresponding *keyshape* representation.

6.4.2 Histogram of Keyshape Orientation

This global approach named HKO is used for filtering purposes determining the best suggestive contour image for each 3D model in the database. This could be understood as choosing the appropriate projection image for a 3D model.

Unlike gradient-based global methods [24, 80], we take into account the information given by *keyshapes*. In this way, we compute a histogram of *keyshape* orientations (HKO) made up with the orientations of the lines detected previously. Since we are not interested in line directions, the orientation of line L, $\theta(L)$ varies from 0 to π. We quantize $\theta(L_{\mathrm{i}})$ $\mathrm{i} = 1 \dots n$ into 8 bins. In this way, each HKO bin b represents the number of lines with orientation quantized as b, $b = 1, \dots, 8$. The final descriptor is the corresponding unitary version of the HKO descriptor.

To determine the appropriate suggestive contour image for each 3D model, we compare the input sketch with each of the 14 contour images of each 3D model using the HKO descriptor. We choose the suggestive contour image with the smallest distance to the sketch's HKO. We use L_1 metric (Manhattan distance) as distance function. Finally, we keep only one suggestive contour image for each 3D model, instead of the 14 contour images at the beginning.

To rank the 3D models of the database, we proceed using a local approach based on *keyshapes* as well.

6.4.3 Keyshape Angular Spatial Distribution

Sketches are characterized by keeping structural information instead of color and texture. Structural information represents the components of an object. For instance, a hand-drawing of a chair is composed of vertical lines representing legs and horizontal line representing its body. Moreover, structural information also represents a spatial relationship between components. For instance, the body of a table is drawing over its corresponding legs. We are interested in our local descriptor exploits such as information.

Taking into account structural information of sketch images we propose a new local descriptor. The proposed local descriptor is computed around of each *keyshape*. So, we will have n local descriptors, where n is the number of *keyshapes*.

Getting Local Descriptor

We named our descriptor as *Keyshape Angular Spatial Distribution* (KASD) because it takes into account the spatial distribution of the *keyshapes* around a referent *keyshape* where an angular-partitioned local circular region is defined.

Let L_R be a referent *keyshape*, we define a circular local region around L_R. The radius of the local region is three times the referent *keyshape* length. In addition, the local region is divided in angular partitions (slices). Each partition corresponds to a spatial region around L_R. An example of a local region and its partitions is depicted in Figure 6.11 (a).

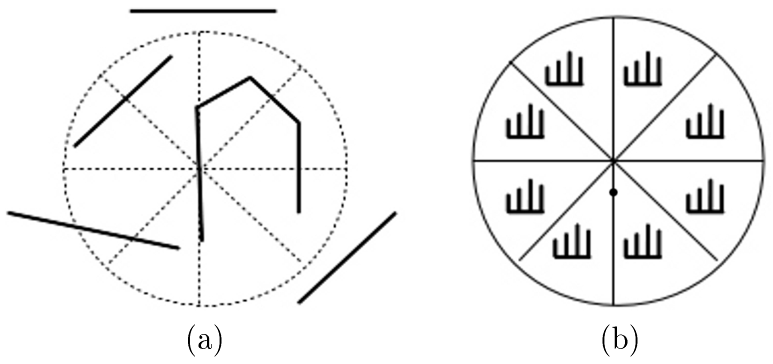


Figure 6.11: (a) Local region around a referent *keyshape*. (b) Local descriptor and its 4-bin histograms for each slice.

Having partitioned the local region of L_R, we proceed to compute a 4-bin histogram for each partition (see Figure 6.11) (b). This histogram represents the distribution of *keyshape* types around L_R computed for each foreground pixel. Because we have four classes (H, V, D_1, D_2), we compute a 4-bin histogram. The local descriptor is the unitary version of the juxtaposition of the eight histograms. The detailed algorithm is presented in Algorithm 7.

Algorithm 7 Local descriptor algorithm

```
1: for all partition i = 1 : 8 do
2:     $h_{p_i}(1 \dots 4) = 0$
3:     for all $pixel(x, y) = 1 \in$ partition i do
4:         $class = getClass(x, y)$
5:         $h_{p_i}(class) = h_{p_i}(class) + 1$
6:     end for
7: end for
8: $h = h_{p_1} \circ h_{p_2} \circ \cdots \circ h_{p_8}$
9: $h = toUnit(h)$
10: return $h$
```

In Algorithm 7, the function $getClass(x, y)$ returns a value in $\{1, 2, 3, 4\}$ depending on the line type which the pixel (x, y) falls on. In this way, if pixel (x,y) falls on a horizontal line (class H), getClass(x,y)=1. For classes V, D_1 and D_2, $getClass$ returns 2,3, or 4, respectively.

Furthermore, symbol $\circ$ means juxtaposition, and function $toUnit(v)$ returns the unitary vector of v.

Matching

Let I be an image (sketch or suggestive contour). We define $LD(I)$ as:

$$LD(I) = \bigcup_{t \in \{H, V, D_1, D_2\}} LD_t(I) \qquad (6.7)$$

where $LD_H(I)$ is a set of local descriptor for horizontal lines in I, $LD_V(I)$ for vertical lines, $LD_{D_1}(I)$ for diagonal lines having slope 1, and $LD_{D_2}(I)$ for diagonal lines having slope -1.

We compare an input sketch against a suggestive contour for each 3D model. The matching process is carried out between local descriptors corresponding to the same *keyshape* type.

Since the number of local descriptors is much lower than in the case of having *keypoints* [59, 9], for the matching problem, we could solve an instance of the bipartite graph problem using the well known Hungarian Method [54] between $LD_t(S)$ and $LD_t(C)$, $t = H, V, D_1, D_2$, where S is an input sketch and C is a suggestive contour of a 3D model.

The cost of matching two local descriptors is measured by the L_1 metric (Manhattan distance). The final match is the union of the partial matches, and the cost of the match is defined as the average cost of all matches.

A question arising in this point is what the cost of comparing an input sketch S with a suggestive contour image C must be. In other words, we need to determine the dissimilarity value between S and C. There are many ways to represent such a dissimilarity value. Some could be the average matching cost, a function respect to the number of matches, among others. We define the dissimilarity value as:

$$DV(S,C) = AC(S,C)/n\_matches(S,C) \tag{6.8}$$

where, AC indicates the average matching cost and $n\_matches$ is the number of matches. The final raking is presented starting with the 3D model having the smallest dissimilarity value between its corresponding suggestive contour and the input sketch ending with that having the highest dissimilarity value.

6.5 Experimental Results

6.5.1 Dataset Description

For our experiments, we used the same benchmark as in [105]. This benchmark has been developed using several 3D mesh models from the Princeton Shape Benchmark, from where 260 models belonging to 13 different classes were selected. These classes are: *ant*, *bear*, *bird*, *chair*, *cup*, *fish*, *glasses*, *hand*, *human*, *octopus*, *plane*, *table*, *tool*.

Using the 3D models, $260 \times 14 = 3640$ suggestive contours from different viewpoints are rendered, which we use for our experiments as training examples. Additionally, the benchmark provides 250 user hand-drawn sketches, which are used as input for the retrieval task evaluation. It is worth mentioning that the sketches are, in fact, rough sketches drawn by users in a free way. No constraint are imposed to the users for drawing such sketches.

Figure 6.12 shows examples of the 3D model used as training data and Figure 6.13 shows several sketches corresponding to four different classes.

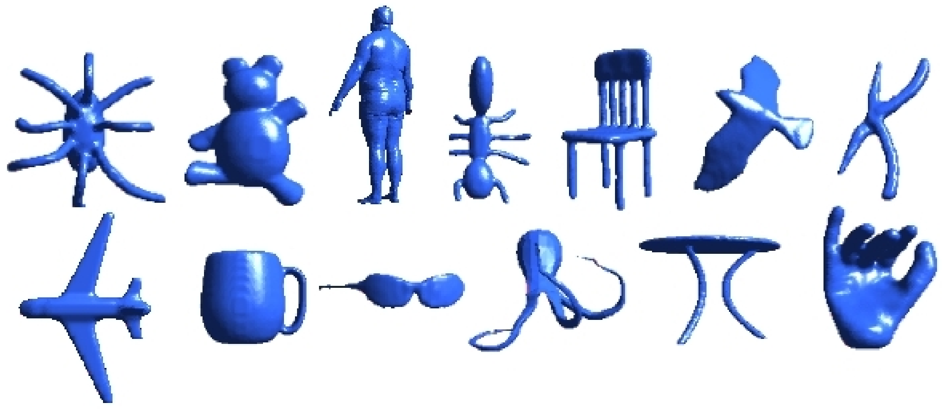


Figure 6.12: Examples of 3D models used as training data.

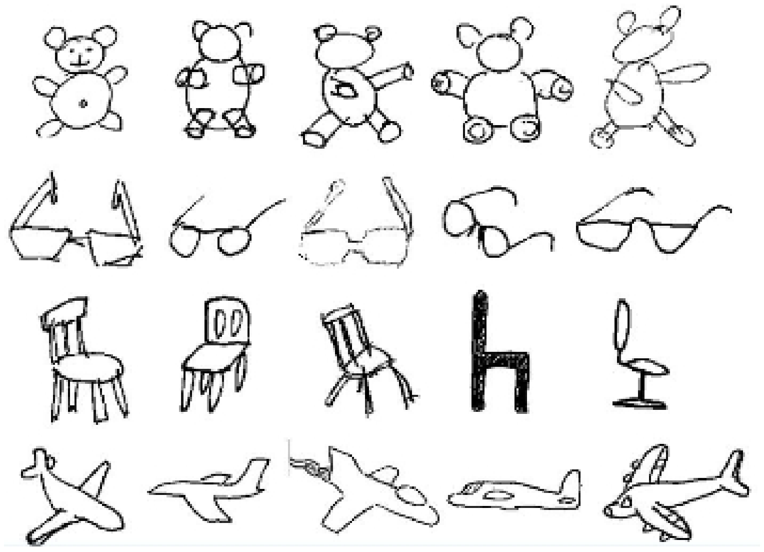


Figure 6.13: Examples of sketches used as queries.

6.5.2 Result Analysis

In this section, we show the results of the retrieved 3D models from various test sketches. Two examples of the retrieval task using the HKO-KASD approach are shown in Figure 6.14. In these examples we depict only the first five retrieved models. We can note that our proposal retrieves only one false positive among the first five retrieved objects, the remaining four objects correspond to objects belonging to the same class of the query.

It is important to note two aspects in our method. First, our method retrieves relevant objects even though they have different viewpoints. Second, our method can retrieve relevant objects even though such as objects, belonging to the same class, undergo shape variations. For instance, looking at the first and last retrieved chair 3D models in Figure 6.14 we can notice that these chairs belong to different chair models. Despite of that fact, our method is capable of retrieving them. The key point here is that our method takes into account the structural components of the underlaying objects. In this way, both chairs are composing of the same components: legs, seat, and backrest.

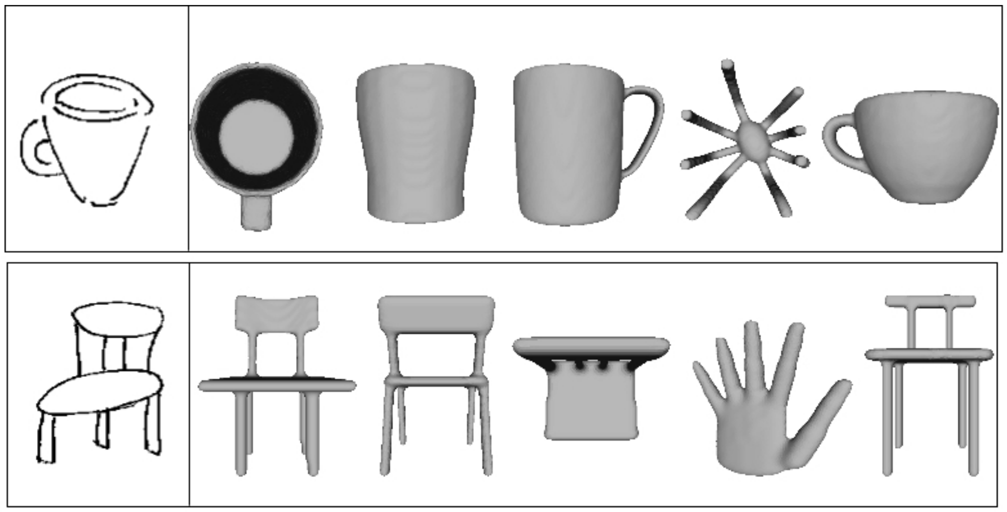


Figure 6.14: Examples of the 3D model retrieval using the proposed local approach. The first columns show a sketch query, the other five images correspond to the first five retrieved models.

To measure the performance of our approaches we use the *first-tier precision* (FTP) metric. In short, the FTP metric evaluates the precision of a retrieval method after retrieving a number of object equal to the number or relevant models in the data set. In our case, each of the 13 classes contains 20 different models, so any input sketch, belonging to one of classes must retrieve 20 relevant models. The FTP metric, then, is computed as the number of the retrieved models that are relevant to the input query divided by the number of the total retrieved models (in our case this value is equal to 20).

We compare our results with recent proposals in the context of sketch-based 3D model retrieval. In particular we present results against HOG-DFT [105] and HELO [80] achieving an increasing in effectiveness for many classes.

In Table 6.1, we show the results achieved by our methods. The result are presented independently for each class in terms of the first tier precision metric. We can note that our two proposals STELA and HKO-KASD have a comparative performing with respect to the state of the art (HOG-DFT). However, STELA and HKO-KASD outperform the HOG-DFT by six classes. For instance, the HKO-KASD achieves outperforming results for the classes: *chair*, *cup*, *glasses*, *hand*, *table* and *tool*, with a maximum improvement for the class *tool*.

In terms of retrieval time, our approaches and the HOG-DFT method require similar times. In particular, our two approaches require approximately 4.2 seconds for retrieving 3D models, while the HOG-DFT requires 3.2 seconds.

Table 6.1: First-tier precision for each class.

Class	**HELO**	**HOG-DFT**	**STELA**	**HKO-KASD**
Ant	0.147	**0.253**	0.126	0.153
Bear	0.210	0.290	**0.338**	0.135
Bird	0.107	**0.145**	0.110	0.121
Chair	0.088	0.068	0.121	**0.167**
Cup	0.138	0.140	0.142	**0.244**
Fish	0.162	**0.192**	0.152	0.124
Glasses	0.029	0.029	0.079	**0.109**
Hand	0.333	0.134	**0.319**	0.252
Human	0.255	**0.452**	0.321	0.180
Octopus	0.108	**0.240**	0.150	0.195
Plane	0.021	**0.120**	0.117	0.088
Table	0.135	0.071	**0.120**	0.110
Tool	0.079	0.005	0.045	**0.157**
AVG	0.139	0.165	0.165	0.157
DES	0.084	0.117	0.093	0.049

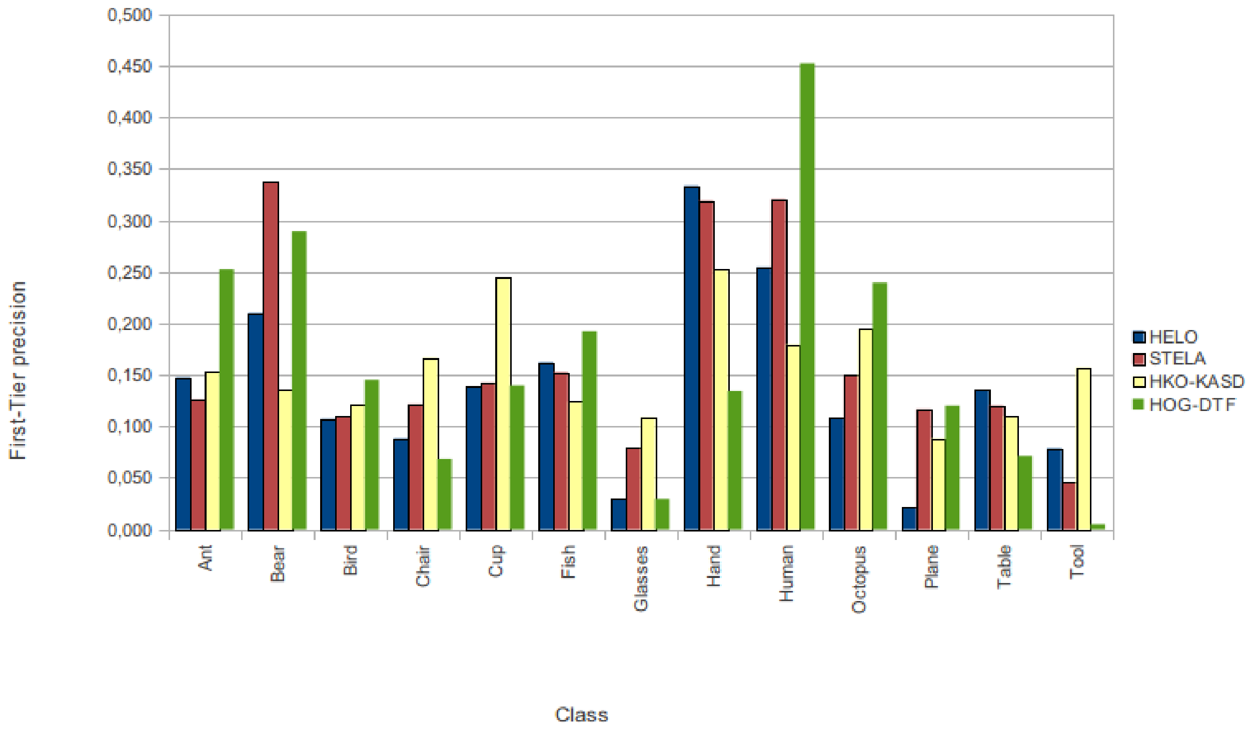


Figure 6.15: The first-tier precision for each class. We compare HKO-KASD with HOG-DFT, HELO and STELA methods.

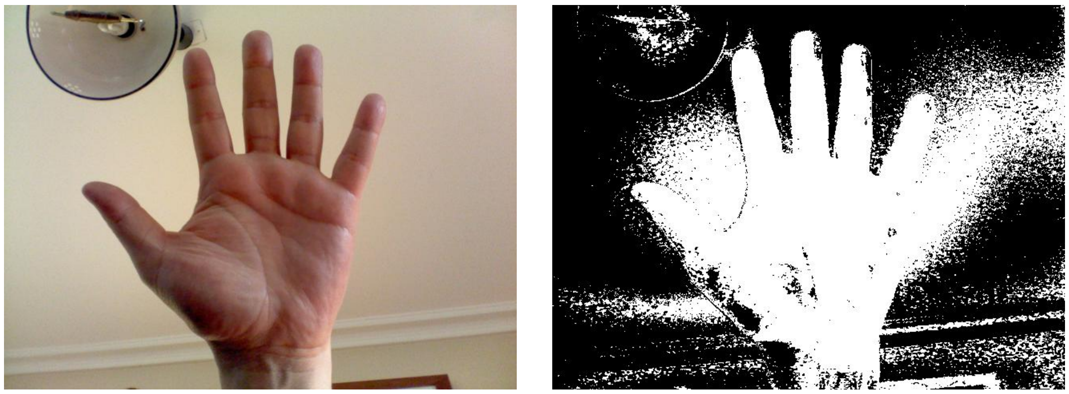


Figure 6.16: A low accurate segmentation result using a skin color based approach [50]. The white pixel on the right binary are the pixels detected as hand points.

6.6 Case of Study: Hand Segmentation using STELA

Hand segmentation has become a very important task for many applications such as those related to vision-based virtual reality [107], gesture recognition [99], and biometric recognition [47, 106]. Furthermore, the hand segmentation task turns more critical in cases where a depth analysis of the hand is required. This kind of analysis could imply getting some measures from the hand image. For instance, in the case of hand biometrics getting reliable hand measurements is essential. Commonly, hand-based biometrics require a special device to capture the hand properties limiting its usability. This problem could be solved if we could get hand measurements directly from a digital cam. Of course, this would require an accurate segmentation stage. Yoruk et al. [106] proposed a method to carry out this process. However, in their work, they consider that the image background is dark and homogeneous leading to a trivial segmentation stage and to low usability as well.

Commonly, hand segmentation is carried out by skin color based techniques [50, 56]. These techniques build a generic model (e.g. a statistical model) from a large collection of training skin images. The model will decide whether a pixel in the image is actually a hand point or not. An important study on this kind of model proposed by Jones et al. [50] is based on a *Gaussian mixture model.* A critical drawback of a skin color based technique is its low performance under illumination variations. Moreover, this kind of technique performs poorly in presence of skin color-like regions. Figure 6.16 shows a segmentation result using the statistical model proposed by Jones et al. [50] for skin segmentation.

In addition, the hand segmentation problem has also been studied in the context of 3D tracking of hand articulations. In this context techniques based on mixing chromatic information with depth information (using Kinect for instance) have showed outperforming results. In this vein, the work of Oikonomidis et al. [73] shows how hand segmentation is achieved blending a skin color model with depth information. The problem with this kind of approach is that a special device (the Kinect) is required. So, our discussion in this document is focus on techniques that simply take as input, a color image.

Instead of using a generic color model we could use a model dependent on the user skin color leading to an adaptive approach. The adaptive term arises due to the fact that the color model will be adapted to the user skin color. This approach allows us not only to tackle the diversity of skin color but also to handle diverse illumination conditions.

In the adaptive segmentation approach, a hand portion called *the training region* needs to be previously marked on an input image. The pixels inside the marked region are used to build a color model that will be used to segment the rest of the hand by means of color similarity. Following this idea, Yuan et al. [107] proposed an algorithm that makes up color clusters using a training region and then labels the clusters as hand or background depending on the size of each cluster. Finally, the image points are classified depending on what cluster they belong to. The problem with this algorithm is that it requires one to mark the training region manually, an undesired task in automatic environments.

To get a training region automatically we could locate the hand on the input image. Using this location we could mark an appropriate hand region, commonly using the location coordinates as the center of such a region.

For the localization problem, the Viola and Jones detector [98] may be used. The problem with this approach is that it requires large training image collection and is time consuming for the training. Because a hand is a simple object with a well defined shape, an expensive training stage is unnecessary. Moreover, the localization problem may be carried out using a local structure-based approach exploiting not only locality information but also the structure of the hand shape.

Our contribution in this work is to present a very accurate hand segmentation technique composed of two main steps: (1) estimate the hand location on an image, and (2) separate the hand region from the background. For the localization stage, we use a local structure-based approach exploiting both structural and locality information of a hand. Structural information is related to the components forming a hand and locality information is related to the spatial relationship between these components. To this end, we use our keyshape based approach named STELA in the context of the stetch based 3D model retrieval [82]. For the segmentation stage we extend the idea of Yuan et al. [107] proposing strategies to compute the underlying parameters. In this case, we make up color clusters from a training region obtained directly from the localization stage (a manual localization is no longer required). For color representation we use only the chromatic channels of the L*a*b* color space as suggested by Yuan et al. [107]. The segmentation stage ends with a post-procesing phase to reduce imperfections caused by noise. An example of our results is shown in Figure 6.17 where the segmentation is specified by a blue contour.

Moreover, our proposal of using a training region may be exploited for applying interactive energy-based segmentation methods like the Grab-Cut approach [79]. This kind of algorithm, that has shown outperforming behavior for the segmentation problem, requires a initialization region. In this way, the training regions resulting from the STELA approach may be used as the initialization region. We show later results achieved using STELA together with the GrabCut approach.

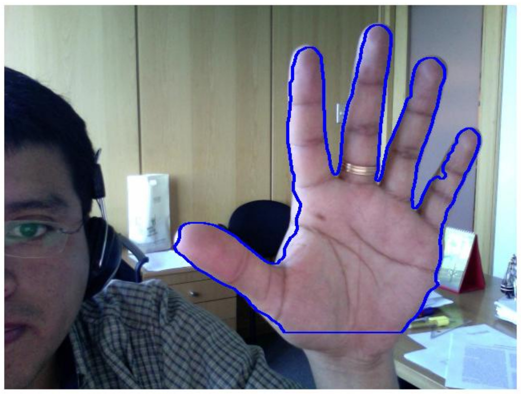


Figure 6.17: An example of hand segmentation using our approach. The blue contour defines the segmented region.

6.6.1 Hand Segmentation

We divide our approach in two stages, the first one corresponds to the hand localization, where STELA is applied to estimate the location of the occurrence of a hand. After this, we extract a training region from the center of the located region and proceed to segment the hand using the color information given by the training region. Finally we carry out a post-processing stage to make our result more robust to environment conditions like illumination or skin color-like regions near to the hand region. A framework of our proposal is presented in Figure 6.18.

Hand Localization

Unlike the machine learning approach for detecting objects, our approach only requires a simple hand prototype. In our implementation we use a 80×80 hand prototype image.

We use STELA to estimate the hand location in an image. Since we are interested in detecting a hand shape we could make STELA faster reducing the image size. We resize the input image to a 100×120 image. To appropriately determine the location of a hand, we apply the sliding window strategy. Each region inside a 80×80 window is compared with the hand prototype by STELA. The center of the window keeps the dissimilarity value between the windowed region and the prototype.

Let $\mathrm{d}_{\mathrm{i}j}$ be the STELA dissimilarity value for each pixel (i, j) in the resized input image. We define D as a set of points where the dissimilarity value is close to the minimum dissimilarity value (mdv). That is:

$$D = \{(\mathrm{i}, j) : \mathrm{d}_{\mathrm{i}j} - mdv \leq 0.1, \mathrm{i} = 1 \ldots 100, j = 1 \ldots 120\},$$

Finally, the occurrence of a hand is located in the centroid (i_c, j_c) of D only if the $mdv \leq TH_h$. This means that if we have $mdv > TH_h$ the method will report a *hand not found*

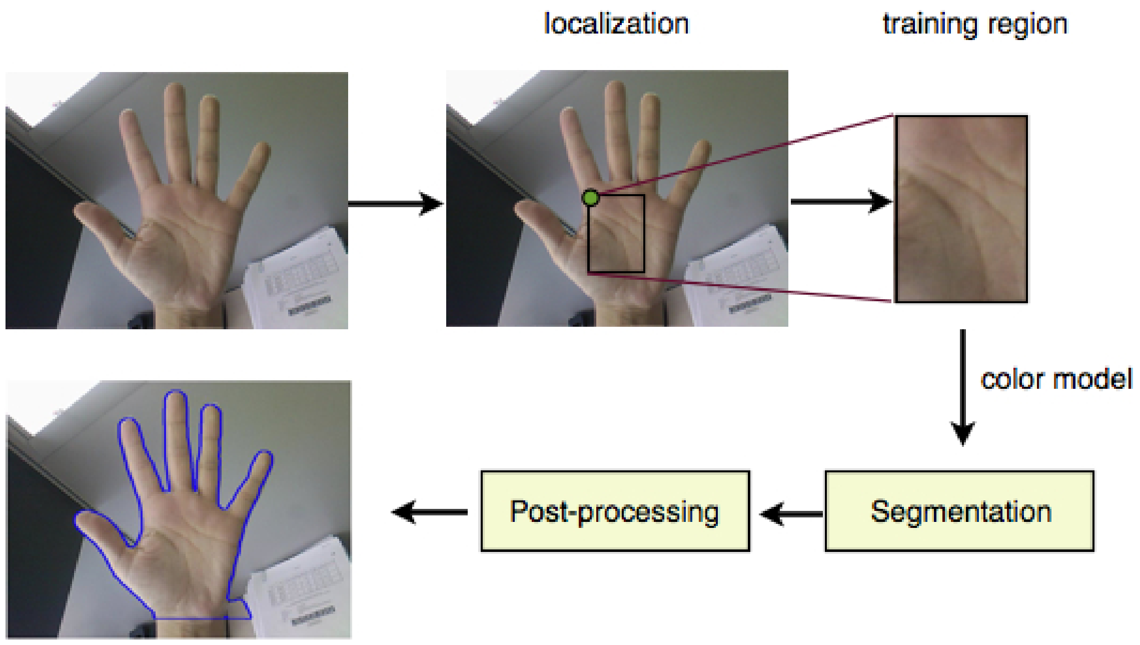


Figure 6.18: Proposal framework.

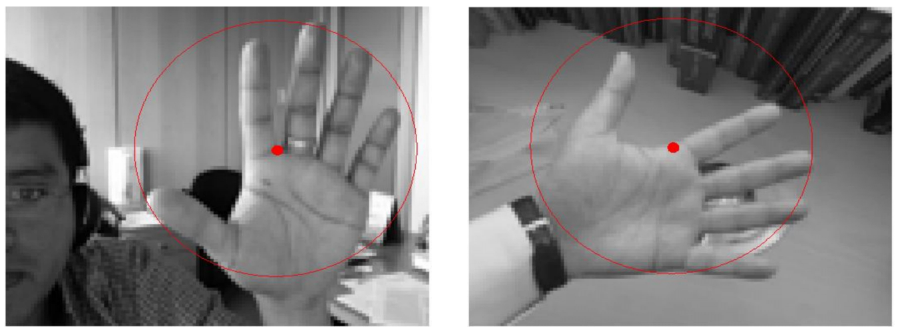


Figure 6.19: Two examples of hand localization.

message. In our experiments we set $TH_h = 0.65$. If a hand is located, the centroid (i_c, j_c) needs to be rescaled to have the real location. Two examples of hand localization are shown in Figure 6.19 where we notice that the proposed method works even when the hand undergoes rotation variations.

Hand Segmentation

After locating the hand, we extract a 150x100 region where the upper left corner corresponds to the hand location point (see Figure 6.18). A great number of pixels inside this region must correspond to the hand. We will refer to this region as the *training region*.

We proceed to make up color clusters. In this way, each pixel of the training region must fall within one of the built clusters. We represent the image by the L*a*b* color space. We only use the chromatic channels $a*$ and $b*$ similiar as the proposal of Yuan et al. [107]. Though we have conducted experiments using different color spaces, we got better results using L*a*b*.

Each color cluster q needs to keep two values, the first one corresponds to the number of pixels falling inside it (N_q), and the second one is a representative point of the cluster (this is not necessarily a real pixel), expressed in terms of its corresponding a* b* color information. We represent this point as $[r_q^a, r_q^b]$. The representative pixel corresponds to the average of the a* b* components of all pixels falling inside q.

We make up the clusters following this given algorithm:

- $SoC = \emptyset$ (set of clusters)
- $N = 0$ (number of clusters)
- For each pixel p in the training region
  - Let [a,b] the corresponding a*b* color information of p.
  - $m = min_{1\leq q \leq N}(L_2([a,b],[r_q^a, r_q^b]))$
  - $q^* = argmin_{1\leq q \leq N}(L_2([a,b],[r_q^a, r_q^b]))$
  - If $m < TH$, update $[r_q^a, r_q^b]$ and increase N_q by 1.
  - Otherwise, add a new cluster w to SoC using p, increasing N by 1. In this case, $[r_w^a, r_w^b] = [a,b]$ and $N_w = 1$.
- return SoC

In the previous algorithm $TH = 0.01$ and L_2 corresponds to the Euclidean distance.

After building the set of clusters SoC, we segment a hand starting from the pixel corresponding to the hand location point expanding the process to the rest of the pixels using the *breadth first search* strategy through the pixels detected as hand point. This strategy avoids detecting objects that are far from the hand region, minimizing the false positive points. The algorithm determines what cluster a pixel p must belong to. An appropriate cluster $q\star$ for the pixel p must satisfy two criteria:

1. The number of pixels in $q\star$ must be greater than TH_N. Clusters with few pixels may correspond to the background.
2. Considering only clusters satisfying the first criterion, the cluster $q\star$ corresponds to that with minimum distance (md) between p and each cluster representative point. Here, the distance function is the *Euclidean distance.*

If $m\text{d} > TH_D$ the pixel is marked as background, otherwise it is marked as a hand point.

In our implementation TH_N is the median of the cluster sizes. Formally:

$$TH\_N = median(N_1, \ldots, N_N) \tag{6.9}$$

In the case of TH_D we conduct a different strategy. The main idea is that TH_D has to be computed depending on the distances between hand points from the training region. Therefore TH_D is a distance that allows us to discard the 10% of the training region points with higher distance value whit respect to the cluster they belong to.

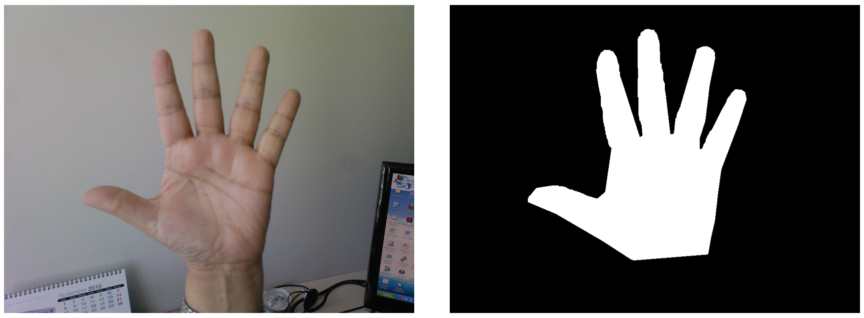


Figure 6.20: An image containing a hand (left) and its corresponding mask (right).

Post-processing

After the segmentation stage we have a binary representation where detected hand points are set to 1 and background points are set to 0. To have an accurate segmentation we apply two post-processing operations. First, the method discards a point detected as hand point if the number of hand points in a local region around the point is less than 50% of the local region size. In this case, we use a 21x21 local region. Second, the method applies morphological operations to fill holes in the hand region [89].

6.6.2 Experimental Results

In this section we show results of the automatic hand segmentation using our proposal. To test our approach, we used a collection of 25 640×480 images containing a hand captured with different kinds of illumination. In terms of pixels, our collection is composed of 5,761,985 hand pixels (positive set) and 18,084,503 non-hand pixels (negative set). We compare our result against a renowned skin color model, specifically we use as baseline as proposed by Jones et al. [50]. We also show the performance of the GrabCut algorithm [79] for the hand segmentation problem. In this case, the GrabCut algorithm is initialized with a foreground region obtained using the STELA approach.

A good tool to assess the performance of our segmentation results is the ROC curve, which allow us to evaluate a method with respect to the relation between false positive points and correct detection points. In this way, we have a mask indicating the hand region for each test image. An image with its corresponding mask is shown in Figure 6.20.

ROC curve analysis also takes into account the area under the curve (AUC) as a quality measure. The higher the AUC value is, the better the quality of our method is. Furthermore ROC curves show us the cost we have paid in terms of false positives when a high correct detection is desired. The ROC curve comparing our method with the skin color model is presented in Figure 6.21.

Our method achieves an AUC value of 0.97 as the skin color model only achieves 0.83. In addition to the overall good performance of our method, it is worth pointing out that our proposal achieves a correct detection rate over 90% at the expense of having only 5% for

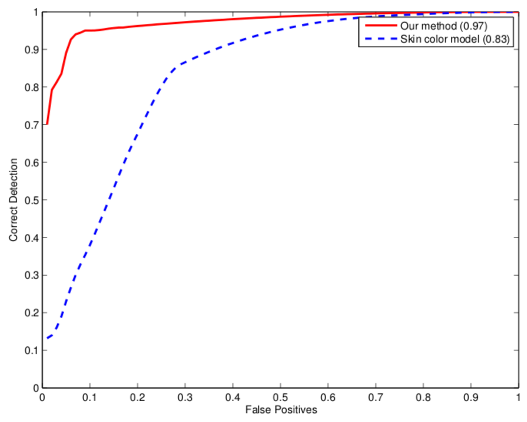


Figure 6.21: ROC curves comparing our method with the skin color model proposed by Jones [50]. The AUC value is indicated in the legend.

CD	FP (Our method)	FP (Skin color model)
0.95	0.146	0.641
0.90	0.052	0.490
0.85	0.046	0.334
0.80	0.009	0.234

Table 6.2: Correct detection rate (CD) vs. false positive rate for our method and the skin color model proposed by Jones [50].

false positives. The skin color model results in 49% of false positives to achieve a comparable result. In Table 6.2 the relationship between false positive and correct detection is shown for both evaluated methods.

Figure 6.22 shows how well our method segments a hand in an image in comparison with the baseline method. Additionally, six examples of hand segmentation using our method are depicted in Figure 6.23.

Finally, we have to say that our method correctly locates a hand for different images. In our experiments we get 100% of correct localization. Therefore, any application requiring a hand localization step could take advantage of our proposal, hand tracking and hand biometrics are two potential applications.

Comparison with the GrabCut approach

In this section we show the performance of the GrabCut approach for the hand segmentation problem and we compare it with our proposed approach. –

To apply the GrabCut approach, we define the training region or foreground region using our localization-based proposal as discussed in the previous sections. Using a training region of foreground pixels provides a better performance than using a bounding rectangle. For this reason, we only show the results of using a foreground training region in the GrabCut approach.

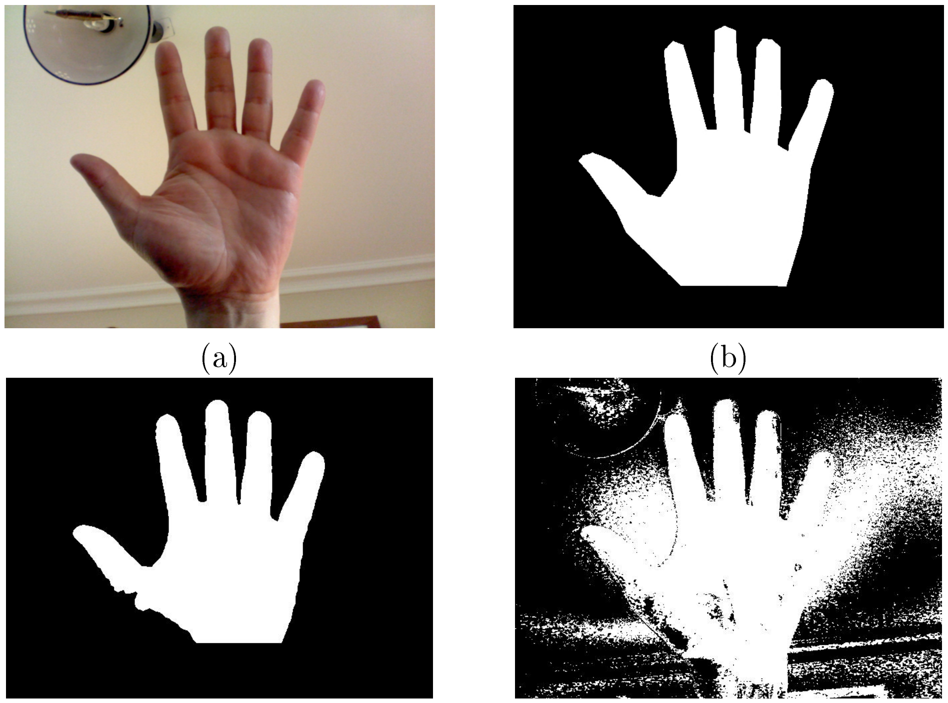


Figure 6.22: Hand segmentation comparison. (a) Input image, (b) target segmentation, (c) output using our method, and (d) output using the skin color model.

In Figure 6.24 we show a ROC curve comparing the proposed method and the GrabCut approach using STELA. We note that the GrabCut approach outperforms the skin-based method, but its performance is still below ours. However, to achieve the better results the GrabCut algorithm requires less time than our approach. In particular we achieve the better result in approximately 34 seconds, while the GrabCut algorithm achieved its better results in 22 seconds using an Intel processor of 2.20 GHz.

6.7 Summary

We have presented two novel approach for retrieving 3D model using rough sketches as queries. In this context, traditional methods do not work appropriately since sketches lack color and texture. However, sketches provide structural information that defines how an object is composed of.

Our proposals STELA and HKO-KASD are a kind of filter-refine methods. We use a global descriptor as the filtering stage and a local descriptor for the refining stage. Both stages rely on structural components called *keyshapes*. These proposals takes into account the structural information that sketches provide.

Our results shows that STELA and HKO-KASD achieves outperforming results for many classes with respect to the HOG-DFT descriptor. For instance, the HKO-KASD descriptor

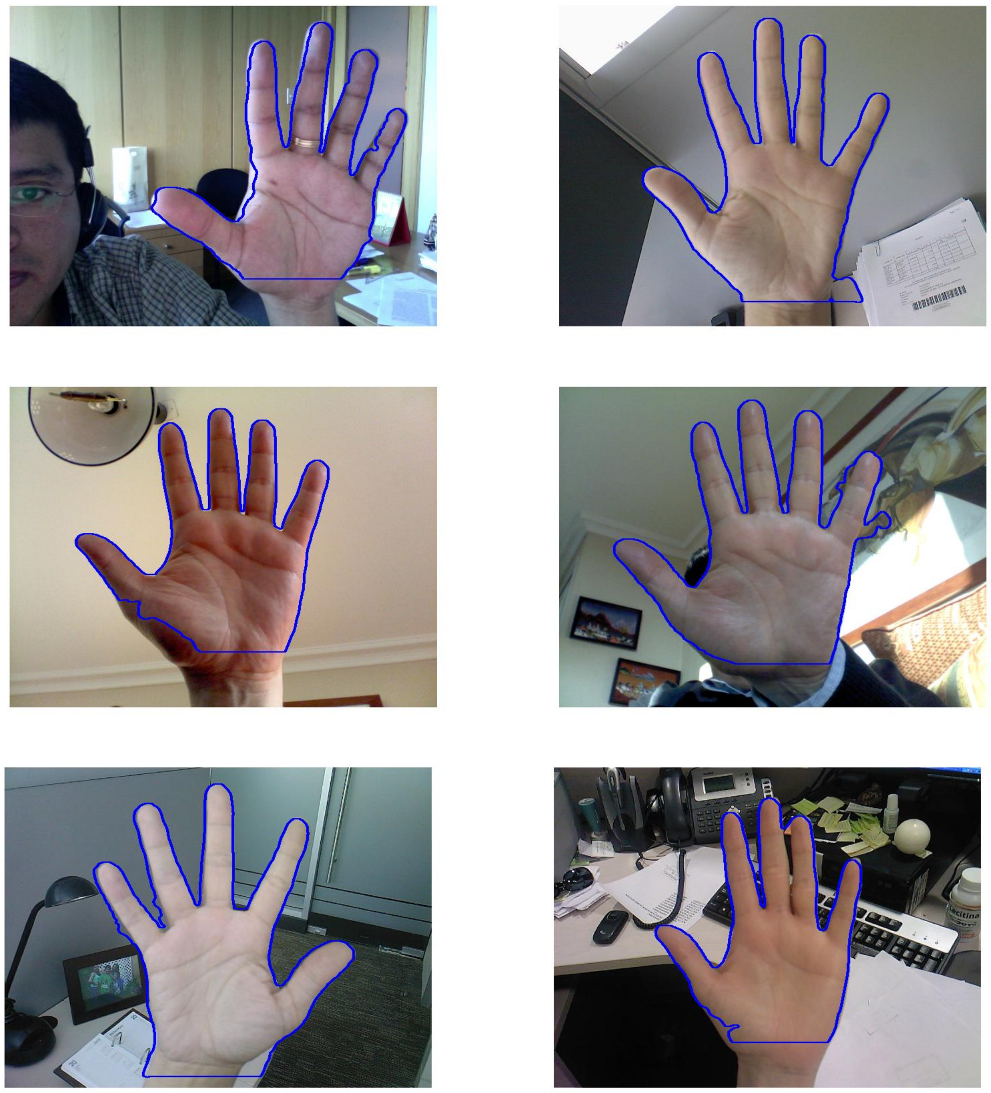


Figure 6.23: Examples of hand segmentation using our proposed approach.

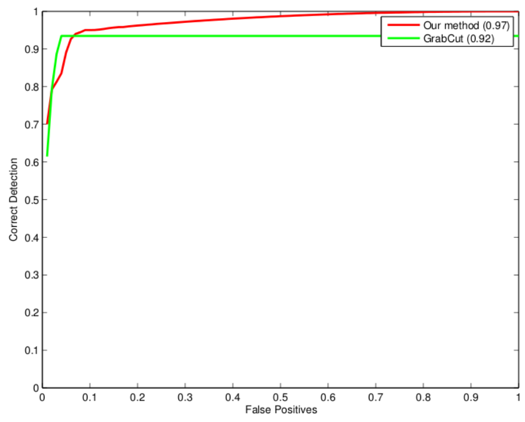


Figure 6.24: ROC curves comparing our method with the GrabCut approach. The AUC value is indicated in the legend.

shown an improvement by a factor of 31 with for the class *tool* with respect to the HOG-DFT descriptor.s

In addition, we have presented a novel approach for the hand segmentation task based on a hand localization using STELA. Our method estimates the hand location to capture a training hand region. This allows us an adaptive color model based on the user skin color. This allows us to handle illumination changes and diverse skin colors. We compare our method with a skin color based model achieving notable improvement in the accurate segmentation. One advantage of our method is that it segments the hand in an accurate way without requiring a lengthy time consumption for the training stage.

We have also showed that a state-of-the art method for object segmentation like the GrabCut algorithm may exploit the hand localization strategy using STELA to outperform the effectiveness. Moreover, using the STELA approach allows the GrabCut algorithm to perform in an automatic way, without requiring any kind of user participation.

Chapter 7

Conclusions

Two novel approaches to dealing with the sketch based image retrieval problem have been addressed in this thesis. The first approach called HELO is a global approach that is based on computing a histogram of orientation using, to this end, the squared gradient technique. Although this global approach exhibits outperforming results with respect to current global methods, it still undergoes critical problems. For instance, our global proposal has poor performance in the cases of partial matching and occlusion. These are important properties since users commonly draw only the objects that they are interested in, hoping to get, as response, all images containing objects similar to those drawing in the query. A simple example is when a user is looking for images containing the sun, the user will possibly draw only a circle.

The second proposed approach represents the core of this thesis. This is a local approach that addresses the sketch based image retrieval problem by exploiting the structural property of a sketch representation. The structural property becomes the main property of a sketch representation since sketches lack color and texture information. In addition, state-of-the-art methods have ignored this property, facing the sketch based image retrieval problem using low level features like edge pixels distribution or edge pixels orientations.

Our structural based approach can be divided in two coarse stages. The first one is focused on detecting the structural components of a sketch representation. To this end, our method detects six types of primitive shapes: ellipses, arcs, horizontal lines, vertical lines, diagonals lines with slope 1 and diagonal lines with slope -1. The occurrence of one of this primitive shapes is called a *keyshape*. We keep only six keyshape types due to the fact that detecting keyshapes on a sketch image may lead in a time consuming process. Therefore, having a large number of keyshape types may makes the structural representation of a sketch, an impractical task.

The second stage consists in extracting local information from the keyshape distribution. To this end, we have proposed three local descriptors characterized by taking into account the spatial relationship between detected keyshapes. The spatial relationship is a critical feature of our methods, since it is not the same to draw a circle over an horizontal than to draw it under the line, if what the user wants to express is “the sun on the beach”.

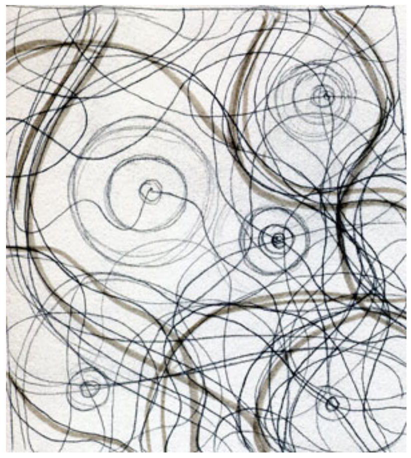


Figure 7.1: A very complex image

Although our keyshape based method outperforms the state-of-the-art techniques, a valuable property of this proposal is that it can be combined with other leading methods increasing significantly the retrieval effectiveness. We show experimentally that our method improves the retrieval results achieved by the Eitz's BoF approach in almost 22%, for instance.

Although detecting primitives shapes of a sketch provides a high level semantic representation of images, this stage may fail when it has to face very complex images like that shown in Figure 7.1. Detecting keyshapes on such an image may be degraded to detect only small segments of lines yielding a decrement in the retrieval effectiveness. Therefore, one of our ongoing tasks is focused on applying techniques like *machine learning*, or *graphical models* to improve the keyshape detection stage. In addition, we are interested in extending our proposal for large repositories. Thereby, we are studying techniques for indexing keyshapes and to turn our method in a parallel one.

In addition, we have shown that our keyshape based proposal may be extended to be used in other domains. In particular, we have show how our keyshape proposal may be adapted to address the 3D model retrieval and how it may be useful for the segmentation of hands in a semi-controlled environment.

3D model retrieval using sketches is a challenging problem owing mainly to the fact that a sketch is a 2D representation in contrast to the 3D models. In this thesis, we also have proposed two approaches to dealing with the 3D model retrieval using sketches as queries. Our proposals leverage the structural information given by sketch strokes. Both proposal fall into the class of filter-refine methods, where the filter step is aimed to pick up the projection image of a 3D model that is closer to the input sketch.

Our first proposal for the 3D model retrieval is STELA which is a first attempt of describing a sketch representation of 3D models by means of a set of keyshapes. For the filtering step, STELA uses a variation of the HELO descriptor that takes into account orientations from local regions. The second approach is HKO-KASD that follows the same pipeline as STELA, but proposes a new descriptor based on computing local distribution of four types of lines. The filter step in this case is carried out by a histogram of keyshape orientations. Our keyshape orientation descriptor provides a more robust results because it does not rely on a gradient approximation that is sensitive to noise. Although, our results show an increase in retrieval effectiveness for some classes of 3D models, they also show that the sketch based 3D model retrieval is still a very challenging problem.

Finally, we have presented a application of the STELA method for the case of hand segmentation outperforming classical techniques based on skin segmentation. For the hand segmentation problem, we use STELA to localize a hand in an image. After that, we used a training region derived from the localization point to learn the distribution of hand colors. Using this information the pixels on the image are classified as a hand point or a non-hand point. Even though our proposal presents outperforming results, its effectiveness may decrease with highly cluttered backgrounds.

7.1 Future Work

Although we have proposed a novel approach to dealing with the sketch based image retrieval, exploiting structural information from a sketch representation, this is still a challenging problem. This situation is reflected in the effectiveness achieved with respect to the Eitz's benchmark. Our proposal is the best method known up to now, achieving a correlation value of 0.34. However this value is still far from the optimal value 1, even though we have improved the retrieval effectiveness in almost 22% with respect to current methods. Therefore our future work is based on improving the retrieval effectiveness of the sketch based image retrieval. In addition, we are planning to extend our work to be applied in large repositories. Next, we summarize four directions of future work derived from this thesis:

1. **Keyshape Detection**: The *keyshape detection* is a critical stage of our proposal. We have presented an efficient approach for detecting six types of keyshapes. Unfortunately, noisy and cluttered images may lead to a poor results. In this regard, a future work must be focused on improving the precision in detecting keyshapes. *Machine learning* and *probabilistic graphical models* may be useful for predicting the occurrences of certain keyshapes on noisy and cluttered images.
2. **Sketch-like Representation**: One of the problems in the sketch based image retrieval domain is that test images are not sketches. A robust technique to turn a test image in a sketch representation is required. We have proposed to use the Canny operator computed in a multiscale manner because Canny operator is fast an permit detecting image edges accurately. However, other approaches based on the image contour detection may be useful for this stage. Some outstanding contour detector methods are *Ultrametric Contour map* [2, 3] and *Global Probability of Boundary* [60]. These techniques have shown outperforming results on the Berkeley benchmark [64]. A study on using this boundary detection techniques in the context of sketch based image retrieval in large databased would be a valuable topic of research.
3. **Multi Sketch based 3D Model Retrieval**: We have presented two techniques to dealing with the sketch based 3D model retrieval problem. Our results have shown that this problem is very difficult, even more when we want to retrieve 3D objects using only a 2D representation like a sketch. Therefore, an interesting future work in this context

is to research techniques that permit combining different input sketches representing different viewpoints of a target 3D model in order to improve the retrieval effectiveness.

4. **Combining Descriptors**: We have show in this work that a combination of descriptor may allow us to increase the retrieval effectiveness. This becomes very important when the descriptors used in the combination extract different information from a source (image or 3D model). For instance, we have shown that a traditional method using a histogram of orientation together with our proposal based on structural information increase the retrieval effectiveness in almost 22% in the context of image retrieval. However, a deep study about the properties that may be kept by descriptors in order to be combined is required.
5. **Indexing**: One goal of the content based image retrieval community is to propose methods that exhibit good performance in large repositories. This goal is also shared by the sketch based image retrieval problem. To this end, data structures that permit speeding up the retrieval process is imperative. Therefore, another direction of future work derived from this thesis must be related to the study of methods for indexing keyshapes in order to make the comparison of an input sketch with test images to work in real time.
6. **Parallelism**: Nowadays, parallel algorithms are preferable that sequential ones, so another direction of future work is based on turning the proposal method in a parallel one. This will allow the proposal method to be practical for searching the web, for instance.
7. **Other Domains**: The ideas of structural representation as the proposed in this thesis may be applied for other domains different from the multimedia retrieval. A potential area where the structural information becomes really important is the handwriting recognition. For instance, in the case of handwritten digits recognition, digits of the same classes share similar structural composition, due to this reason we claim that an extension of our work in that area would produce high effectiveness.
8. **Novel Applications**: Although there still are a few people working on the sketch based image retrieval area, in the last two years it has been observed an increased interest in this area. Furthermore, due to the advances in the human-computer interaction technology such as touch-screen displays and gaze based human-computer interaction, potential applications based on image retrieval by sketches are appearing. For instance, our proposal may be useful to improve the cognitive skills of children and to allow users with hand-motility issues to draw a simple sketch query using a gaze interaction system.

Glossary

APAI	Angular partitioning of abstract images.
BoF	Bag of features.
BoW	Bag of words.
CBIR	Content based image retrieval.
CLD	Color layout descriptor.
CSD	Color structure descriptor.
DCD	Dominant color descriptor.
DoIGOH	Dominant image gradient orientation histogram.
EHD	Edge histogram descriptor.
GF-HOG	Gradient field- histogram of oriented gradients.
HELO	Histogram of edge local orientations.
HKO-KASD	Histogram of keyshape orientations - Keyshape angular spatial distribution.
HOG	Histogram of oriented gradients.
HOG-DTF	Histogram of oriented gradients - Diffusion tensor field.
PC	Polar coordinates.
PCA	Principal component analysis.
QBIC	Query by image contents.
QVE	Query by visual example.
SBIR	Sketch based image retrieval.
SIFT	Scale invariant feature transform.

ST	Structure tensor.
STELA	A structure local approach for multimedia retrieval using sketches as query.
TBD	Texture browsing descriptor.

Bibliography

[1] Mihael Ankerst, Gabi Kastenmüller, Hans-Peter Kriegel, and Thomas Seidl. 3D shape histograms for similarity search and classification in spatial databases. In *Proc. of the 6th International Symposium on Advances in Spatial Databases*, pages 207–226, 1999.

[2] Pablo Arbelaez. Boundary extraction in natural images using ultrametric contour maps. In *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*, CVPRW '06, pages 182–, 2006.

[3] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, May 2011.

[4] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval: Concepts and Technology behind Search*. Addison-Wesley Professional, USA, second edition, 2011.

[5] Alfonso Baldi, Raffaele Murace, Emanuele Dragonetti, Mario Manganaro, Oscar Guerra, Stefano Bizzi, and Luca Galli. Definition of an automated content-based image retrieval (cbir) system for the comparison of dermoscopic images of pigmented skin lesions. *Biomedical Engineering Online*, 8(18), 2009.

[6] D. H. Ballard. Readings in computer vision: issues, problems, principles, and paradigms. chapter Generalizing the hough transform to detect arbitrary shapes, pages 714–725. 1987.

[7] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, June 2008.

[8] Asker M. Bazen and Sabih H. Gerez. Systematic methods for the computation of the directional fields and singular points of fingerprints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):905–919, July 2002.

[9] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, April 2002.

[10] Miroslaw Bober. Mpeg-7 visual shape descriptors. *IEEE Transactions on Circuits and*

Systems, 2001.

[11] F.L. Bookstein. Principal warps: thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):567 –585, 1989.

[12] Gunilla Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):849–865, November 1988.

[13] Andreas Bulling and Hans Gellersen. Toward mobile eye-based human-computer interaction. *IEEE Pervasive Computing*, 9:8–12, 2010.

[14] Rainer Burkard, Mauro Dell'Amico, and Silvano Martello. *Assignment Problems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009.

[15] Benjamin Bustos, Daniel Keim, Dietmar Saupe, and Tobias Schreck. Content-based 3D object retrieval. *IEEE Computer Graphics and Applications*, 27(4):22–27, 2007.

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

[17] Yang Cao, Changhu Wang, Liqing Zhang, and Lei Zhang. Edgel index for large-scale sketch-based image search. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–768. IEEE Computer Society, 2011.

[18] Abdolah Chalechale, Golshah Naghdy, and Alfred Mertins. Sketch-based image matching using angular partitioning. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 35(1):28–41, 2005.

[19] Ding-Yun Chen, Xiao-Pei Tian, Yu te Shen, and Ming Ouhyoung. On visual similarity based 3D model retrieval. In *Proc. in Eurographics, Computer Graphics Forum*, pages 223–232, 2003.

[20] Tao Chen, Ming-Ming Cheng, Ping Tan, Ariel Shamir, and Shi-Min Hu. Sketch2photo: internet image montage. *ACM Transactions on Graphics*, 28(5):124:1–124:10, December 2009.

[21] Leszek Cieplinski. Mpeg-7 color descriptors and their applications. In *Proceedings of the 9th International Conference on Computer Analysis of Images and Patterns*, pages 11–20. Springer-Verlag, 2001.

[22] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* John Wiley & Sons, New Jersey, USA, 2006.

[23] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[24] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, pages 886–893. IEEE Computer Society, 2005.

[25] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2):1–60, April 2008.

[26] E. R. Davies. The effect of noise on edge orientation computations. *Pattern Recognition Letters*, 6(5):315–322, 1987.

[27] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. Suggestive contours for conveying shape. *ACM Transaction og Graphics*, 22:848–855, July 2003.

[28] Alberto Del Bimbo and Pietro Pala. Visual image retrieval by elastic matching of user sketches. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):121–132, 1997.

[29] Mathias Eitz, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. A descriptor for large scale image retrieval based on sketched feature lines. In *Proc. of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*, pages 29–36, 2009.

[30] Mathias Eitz, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. Photosketch: a sketch based image query and compositing system. In *SIGGRAPH 2009: Talks*, SIGGRAPH '09, pages 60:1–60:1, 2009.

[31] Mathias Eitz, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. Sketch-based image retrieval: Benchmark and bag-of-features descriptors. *IEEE Transactions on Visualization and Computer Graphics*, 17(11):1624–1636, 2011.

[32] Mathias Eitz, Ronald Richter, Tamy Boubekeur, Kristian Hildebrand, and Marc Alexa. Sketch-based shape retrieval. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 31(4):31:1–31:10, 2012.

[33] Michael Elad, Ayellet Tal, and Sigal Ar. Content based retrieval of vrml objects: an iterative and interactive approach. In *Proc. of the sixth Eurographics workshop on Multimedia 2001*, pages 107–118, 2002.

[34] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. page 178, 2004.

[35] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, David Steele, and Peter Yanker. Query by image and video content: The qbic system. *Computer*, 28:23–32, 1995.

[36] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach.* Prentice Hall, 2012.

[37] Thomas Funkhouser, Patrick Min, Michael Kazhdan, Joyce Chen, Alex Halderman, David Dobkin, and David Jacobs. A search engine for 3d models. *ACM Transactions on Graphics*, 22(1):83–105, 2003.

[38] P. Geetha and Vasumathi Narayanan. A survey of content-based video retrieval. *Journal of Computer Science*, 4(6):474 –486, 2008.

[39] Allen Gersho and Robert M. Gray. *Vector Quantization and Signal Compression.* Kluwer Academic Publisher, Massachusetts, USA, 1993.

[40] Rafael Gonzalez and Richard Woods. *Digital Image Processing.* Pearson Prentice Hall, New Jersey, USA, third edition, 2008.

[41] Zicheng Guo and Richard W. Hall. Parallel thinning with two-subiteration algorithms. *Communications of the ACM*, 32(3):359–373, March 1989.

[42] Yan Ha and BoYoun Kim. Shopping mall system with image retrieval based on uml. In *First ACIS International Symposium on Software and Network Engineering (SSNE), 2011*, pages 103 –106, 2011.

[43] C. Harris and M. Stephens. A Combined Corner and Edge Detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[44] Rui Hu, M. Barnard, and J. Collomosse. Gradient field descriptor for sketch based retrieval and localization. In *17th IEEE International Conference on Image Processing (ICIP), 2010*, pages 1025 –1028, 2010.

[45] Rui Hu, Tinghuai Wang, and J. Collomosse. A bag-of-regions approach to sketch-based image retrieval. In *18th IEEE International Conference on Image Processing (ICIP)*, pages 3661 –3664, sept. 2011.

[46] Weiming Hu, Nianhua Xie, Li Li, Xianglin Zeng, and S. Maybank. A survey on visual content-based video indexing and retrieval. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 41(6):797 –819, 2011.

[47] De-Shuang Huang, Wei Jia, and David Zhang. Palmprint verification based on principal lines. *Pattern Recognition*, 41, April 2008.

[48] Anil Jain and Aditya Vailaya. Image retrieval using color and shape. *Pattern Recognition*, 29:1233–1244, 1996.

[49] Russ John C. *The Image Processing Handbook.* CRC Press, Florida, USA, fourth edition, 2006.

[50] Michael J. Jones and James M. Rehg. Statistical color models with application to skin detection. *Int. Journal Comput. Vision*, 46, January 2002.

[51] Sabrina Kacher, Jean-Claude Bignon, Gilles Halin, and Gérald Duffing. The content-based image retrieval as an assistance tool to the architectural design domain. In Harry Timmermans & Bauke de Vries, editor, *Architecture and Urban Planning*, 2002.

[52] Toshikazu Kato, Takio Kurita, Nobuyuki Otsu, and Kyoji Hirata. A sketch retrieval method for full color image database-query by visual example. In *Proc. of the 11th IAPR International Conf. on Computer Vision and Applications, Conf. A: Pattern Recognition*, pages 530–533, 1992.

[53] Yan Ke and Rahul Sukthankar. Pca-sift: a more distinctive representation for local image descriptors. In *Proceedings of the 2004 IEEE computer society conference on Computer vision and pattern recognition*, pages 506–513. IEEE Computer Society, 2004.

[54] Harold W. Kuhn. The hungarian method for the assignment problem. Technical report, 2010.

[55] Louisa Lam, Seong-Whan Lee, and Ching Y. Suen. Thinning methodologies-a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):869–885, September 1992.

[56] Y.P. Lew, A.R Ramli, S.Y.Koay, R. Ali, and V. Prakash. A hand segmentation scheme using clustering technique in homogeneous background. In *Proc. of 2nd Student Conference on Research and Development*, 2002.

[57] B. Li, T. Schreck, A. Godil, M. Alexa, T. Boubekeur, B. Bustos, J. Chen, M. Eitz, T. Furuya, K. Hildebrand, S. Huang, H. Johan, A. Kuijper, R. Ohbuchi, R. Richter, J. M. Saavedra, M. Scherer, T. Yanagimachi, G. J. Yoon, and S. M. Yoon. SHREC'12 Track: Sketch-Based 3D Shape Retrieval. In *Eurographics Workshop on 3D Object Retrieval*, pages 109–118, 2012.

[58] Sven Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31:983–1001, 1998.

[59] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[60] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008.*, pages 1 –8, june 2008.

[61] Davide Maltoni, Dario Maio, Anil K. Jain, and Salil Prabhakar. *Handbook of Fingerprint Recogniton*. Springer-Verlag, London, UK, second edition, 2009.

[62] B. S. Manjunath, P. Wu, S. Newsam, and H. D. Shin. A texture descriptor for browsing and similarity retrieval. *Signal Processing: Image Communication*, 16(1,2):33–43, 2000.

[63] B.S. Manjunath, J.-R. Ohm, V.V. Vasudevan, and A. Yamada. Color and texture descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):703 –715, 2001.

[64] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *8th International Conference on Computer Vision*, volume 2, pages 416–423, July 2001.

[65] Lew Michael S. *Principles of Visual Information Retrieval.* Springer-Verlag, London, UK, 2001.

[66] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proceedings of the 7th European Conference on Computer Vision-Part I*, pages 128–142. Springer-Verlag, 2002.

[67] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, October 2004.

[68] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[69] F. Mokhtarian and A.K. Mackworth. A theory of multiscale, curvature-based shape representation for planar curves. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(8):789 –805, 1992.

[70] Farzin Mokhtarian and Miroslav Bober. *Curvature Scale Space Representation: Theory, Applications, and MPEG-7 Standardization.* Kluwer Academic Publishers, Norwell, MA, USA, 2003.

[71] Farzin Mokhtarian, Yoke Khim Ung, and Zhitao Wang. Technical section: Automatic fitting of digitised contours at multiple scales through the curvature scale space technique. *Computer Graphics*, 29(6):961–971, 2005.

[72] Mark Nixon and Alberto Aguado. *Feature Extraction & Image Processing.* Academic Press, California, USA, 2008.

[73] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *Proceedings of the British Machine Vision Conference*, pages 101.1–101.11. BMVA Press, 2011.

[74] Temel Oncan. A survey of the generalized assignment problem and its applications. *INFOR: Information Systems and Operational Research*, (3):123–141, 2007.

[75] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Matching 3D models with shape distributions. In *Proc. of the International Conference on Shape Modeling & Applications*, pages 154–, 2001.

[76] Geng Peng, Wang Tongming, and Wu Weina. Application of the image retrieval technique on the education resources image database. In *Computational Intelligence and Design, 2009. ISCID '09. Second International Symposium on*, volume 1, pages 152 –154, 2009.

[77] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Transactions on Graphics*, 22(3):313–318, 2003.

[78] Yong Man Ro, Munchurl Kim, Ho Kyung Kang, and B. S. Manjunath. Mpeg-7 homogeneous texture descriptor. *ETRI Journal*, 23(2):41–51, 2001.

[79] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM TRANSACTIONS ON GRAPHICS*, 23:309–314, 2004.

[80] Jose Saavedra and Benjamin Bustos. An improved histogram of edge local orientations for sketch-based image retrieval. In *Pattern Recognition*, volume 6376 of *Lecture Notes in Computer Science*, pages 432–441. 2010.

[81] Jose M. Saavedra and Benjamin Bustos. Sketch-based image retrieval using keyshapes. *Submitted to Multimedia Tools and Applications*, pages –, 2012.

[82] Jose M. Saavedra, Benjamin Bustos, Maximilian Scherer, and Tobias Schreck. Stela: sketch-based 3d model retrieval using a structure-based local approach. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, ICMR '11, pages 26:1–26:8. ACM, 2011.

[83] Jose M. Saavedra, Benjamin Bustos, Tobias Schreck, Sang Min Yoon, and Maximiliam Scherer. Sketch-based 3D Model Retrieval using Keyshapes for Global and Local Representation. In *Eurographics Workshop on 3D Object Retrieval*, pages 47–50, 2012.

[84] Gisbert Schneider and Uli Fechner. Computer-based de novo design of drug-like molecules. *Nature Reviews Drug Discovery*, 4:649–663, 2005.

[85] Lambert Schomaker, Edward de Leau, and Louis Vuurpijl. Using pen-based outlines for object-based annotation and image-based queries. In *Proceedings of the Third International Conference on Visual Information and Information Systems*, pages 585–592. Springer-Verlag, 1999.

[86] Gayane Shalunts, Yll Haxhimusa, and Robert Sablatnig. Architectural style classification of building facade windows. In *Proceedings of the 7th international conference on Advances in visual computing - Volume Part II*, pages 280–289. Springer-Verlag, 2011.

[87] Thomas Sikora. The mpeg-7 visual standard for content description-an overview. *Circuits and Systems for Video Technology, IEEE Transactions on*, 11(6):696 –702, 2001.

[88] Alvy Ray Smith. Color gamut transform pairs. *SIGGRAPH Computer Graphics*, 12(3), 1978.

[89] Pierre Soille. *Morphological Image Analysis: Principles and Applications*. Springer-Verlag Telos, 1999.

[90] Amanda Spink and Jansen Bernard. *Web Searching: Public Searching of the Web*. Kluwer Academic Publishers, USA, 2004.

[91] Sophie Stellmach, Sebastian Stober, Andreas Nürnberger, and Raimund Dachselt. Designing gaze-supported multimodal interactions for the exploration of large image collections. In *Proceedings of the 1st Conference on Novel Gaze-Controlled Applications*, pages 1:1–1:8, 2011.

[92] Bjorn Stenger, Arasanathan Thayananthan, Philip H. S. Torr, and Roberto Cipolla. Model-based hand tracking using a hierarchical bayesian filter. *IEEE Transactions on Pattern Analyis and Machine Intelligence*, 28(9):1372–1384, September 2006.

[93] Chee Sun Won, Dong Kwon Park, and Soo-Jun Park. Efficient use of MPEG-7 edge histogram descriptor. *Electronic and Telecomunications Research Institute Journal*, 24:23–30, 2002.

[94] Richard Szeliski. *Computer Vision.* Springer-Verlag, London, UK, 2011.

[95] Johan W. Tangelder and Remco C. Veltkamp. A survey of content based 3d shape retrieval methods. *Multimedia Tools Applications*, 39:441–471, September 2008.

[96] Tinne Tuytelaars and Krystian Mikolajczyk. Local invariant feature detectors: a survey. *Foundations and Trends Computer Graphics and Vision*, 3(3):177–280, July 2008.

[97] Remco C. Veltkamp and Michiel Hagedoorn. State of the art in shape matching. In Michael S. Lew, editor, *Principles of visual information retrieval*, pages 87–119. Springer-Verlag, London, UK, UK, 2001.

[98] Paul Viola and Michael Jones. Robust real-time object detection. *Int. Journal of Computer Vision*, 2002.

[99] Juan Wachs, Mathias Kölsch, Helman Stern, and Yael Edan. Vision-based hand gesture interfaces: Chall. and innov. *Communications of the ACM*, February 2011.

[100] Chong Wang, D. Blei, and Fei-Fei Li. Simultaneous image classification and annotation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1903 –1910, 2009.

[101] Xin-Jing Wang, Lei Zhang, Ming Liu, Yi Li, and Wei-Ying Ma. Arista - image search to annotation on billions of web photos. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2987 –2994, 2010.

[102] JinKue Wong. A new implementation of an algorithm for the optimal assignment problem: An improved version of munkres' algorithm. *BIT Numerical Mathematics*, 19:418–424, 1979.

[103] Zijun Yang and Jay C. C. Kuo. Survey on image content analysis, indexing, and retrieval techniques and status report of mepg-7. *Tamkang Journal of Science and Engineering*, 2(3):101–118, 1999.

[104] Jie Yao, Nawwaf Kharma, and Peter Grogono. A multi-population genetic algorithm for robust and fast ellipse detection. *Pattern Analysis & Applications*, 8:149–162, 2005.

[105] Sang Min Yoon, Maximilian Scherer, Tobias Schreck, and Arjan Kuijper. Sketch-based 3D model retrieval using diffusion tensor fields of suggestive contours. In *Proc. of the international conference on Multimedia*, pages 193–200, 2010.

[106] Erdem Yörük, Helin Dutağaci, and Bülent Sankur. Hand biometrics. *Image and Vision Computing*, 24:483–497, 2006.

[107] Miaolong Yuan, Farzam Farbiz, Corey Mason Manders, and Ka Yin Tang. Robust hand tracking using a simple color classification technique. In *Proc. of The 7th ACM SIGGRAPH Int. Conf. on Virtual-Reality Continuum and Its Applications in Industry*, 2008.

[108] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity Search: The Metric Space Approach.* Springer Science+Business Media Inc, USA, 2010.