Institut d'Informatique
Université de Fribourg (Suisse)

Content-Based Image Retrieval Using Hand-Drawn Sketches and Local Features: a Study on Visual Dissimilarity

THESE

présentée à la Faculté des Sciences de l'Université de Fribourg (Suisse) pour l'obtention du grade de Doctor scientiarum informaticarum

Folco Banfi

de Cureggia (TI)

Thèse n. 1312
Imprimerie St. Paul, Fribourg
2000

Acceptée par la Faculté des Sciences de l'Université de Fribourg (Suisse) sur la proposition de:

- Prof. Rolf Ingold (Université de Fribourg)
- Prof. Jacques Pasquier (Université de Fribourg)
- Prof. Thierry Pun (Université de Genève)
- Dr. Houda Chabbi (Université de Fribourg).

Fribourg, le 23 août 2000

Le directeur de thèse | Le Doyen

Prof. Rolf Ingold | Prof. Alexander Von Zelewsky

Abstract

This thesis addresses the question of content-based image retrieval (CBIR) in heterogeneous databases. In an analysis of the existing CBIR tools that was done at the beginning of this work, we have shown that there was room for improvement in three key areas: query form, image and query representation, and computation of similarity. This analysis led us to studying the usability of a method for computing dissimilarity between user-produced pictorial queries and database images according to features extracted from automatically segmented homogeneous areas.

The proposed approach differentiates itself from the analyzed ones by giving maximum freedom to the user by using user-produced pictorial queries (sketches) depicting the wanted image(s), extracts visual information from areas of the images automatically recognized as visually homogeneous and allows the comparison of database images with queries containing various levels of detail, thanks to a hierarchical representation of both database images and queries. Sketches can be incomplete (i.e., they do not need to cover all the available canvas), resulting in extra flexibility. Furthermore, the method can be combined with classical CBIR methods, such as keyword indexing.

In order to support our proposal, a prototype CBIR system, SimEstIm, was built. In SimEstIm, the user produces a query image with a paint tool, then submits it to the system, which extracts a query representation. At database population time, database images undergo the same treatment, which consists of two steps: region segmentation and region merging. In order to allow the comparison between database images and sketches containing various levels of detail, several segmentation results are stored for each image. Visual dissimilarity is computed as a combination of dissimilarities between the regions in the query and the regions in the database image's segmentation results, resulting in dissimilarity values for each segmentation result. These results are then used to compute a unique dissimilarity score between query and image. The user can control the behavior of the dissimilarity measure by setting weights associated to each visual feature.

Experiments were performed by several users. The results obtained are extremely encouraging, and show that the proposed method can be successfully implemented in a CBIR system.

Résumé

Cette thèse traite la problématique de la recherche d'images par rapport au contenu (content-based image retrieval, abrégé CBIR, en anglais) dans une base de données hétérogène. Dans une analyse faite au début de ce travail, nous avons montré que des améliorations étaient possibles dans trois secteurs de cette branche: la forme de la requête, la représentation des images et des requêtes et le calcul de la similarité. Cette analyse nous a poussés à étudier l'utilisabilité d'une méthode pour calculer la dissimilarité entre les requêter produites sous forme d'esquisses par l'utilisateur et les images de la base de données, par rapport à des caractéristiques extraites de régions visuellement homogènes qui ont été segmentées automatiquement.

L'approche proposée se différencie de celles qui avaient été analysées en donnant le maximum de liberté à l'utilisateur, qui peut utiliser des esquisses représentant la/les images désirées, en extrayant les caractéristiques visuelles des images de régions qui ont été identifiées comme visuellement homogènes de façon automatique, ainsi qu'en permettant la comparaison des images de la base de données avec des esquisses plus ou moins détailées, grâce a une représentation hierarchique des images et des esquisses. Les esquisses peuvent être incomplètes (càd, peuvent ne pas remplir toute la surface disponible), ce qui resulte en une flexibilité accrue. Accessoirement, la méthode peut être combinée avec les méthodes classique de CBIR, comme l'indexation par mots-clé.

Afin de faire des expériences avec cette approche, un prototype nommé SimEstIm a été construit. Dans SimEstIm, l'utilisateur produit une esquisse avec un programme de dessin, puis la soumet au système, qui calcule sa représentation. Lorsqu'elles sont introduites dans la base de donnés, les images subissent le même traitement, qui consiste en deux étapes: segmentation en régions et fusion de régions. Pour permettre la comapraison entre les images et des esquisses plus ou moins détaillées, chaque image est représentée par plusieurs résultats de ségmentation. Les dissimilarités entre les régions de l'esquisse et les régions contenues dans les résultats de segmentation de l'image sont calculée, ce qui permet de calculer une valeur de dissimilarité pour chaque résultat de segmentation. Ces valeurs sont ensuite utilisées pour obtenir une valeur de dissimilarité unique. L'utilisateur peut contrôler le comportement de la mesure de dissimilarité en modifiant les poids liés aux caractéristiques.

Des tests ont été conduits avec plusieurs utilisateurs. Les résultats obtenus sont très encourageants et montrent que l'approche proposée peut être implémentée avec succès à l'intérieur d'un système de CBIR.

Remerciements / Ringraziamenti

Arrivé à ce point (final?) de mon chemin dans la recherche, je crois qu'il soit temps de remercier quelques-unes des personnes qui m'ont été proches:

- i miei genitori, per aver sempre sostenuto le mie decisioni e non aver mai cercato di influenzare le mie scelte di studio o professionali, nonché per i sacrifici che hanno fatto per farmi studiare. Se sono arrivato fin qui, è sicuramente anche per merito vostro.
- Rolf Ingold, qui m'a proposé un sujet de doctorat des plus intéressants, et qui a su me guider jusqu'ici.
- Thierry Pun, qui a rendu son rapport de thèse en un temps record, malgré un emploi du temps très chargé.
- Houda Chabbi, qui m'a beaucoup aidé le long de ces cinq ans, en relisant mes publications et aussi en me posant des questions toujours très pertinantes.
- les personnes qui ont eu la malchance de partager un bureau avec moi: Abdelwahab Zramdini, Rolf Brugger, Frédéric Bapst (le bureau est devenu plus tranquille après le départ de tes disques), Lyse Robadey (qui a été ma cible de prédilection... mais on ne châtie que ceux qu'on aime) et Oliver Hitz. Si on me demandait de prendre une chose de Fribourg et de la garder pour le reste de ma vie, je demanderais l'ambiance qu'on avait au bureau...
- mes amis (désolé, vous êtes trop nombreux pour tous vous citer ici...), avec lesquels même Fribourg devient un endroit où il fait bon vivre...
- mio fratello (ce la farai anche tu!) e le mie due sorelle, con i quali ho passato (e spero di continuare a farlo) dei momenti veramente "paraguayani", che mi hanno permesso di dimenticare per qualche ora i periodi difficili che ho attraversato durante questo dottorato.

Table of Contents

CHAPTER 1

Introduction

Over the last 15 years, electronic documents have changed from “text-only” documents to “multimedia” documents containing text, images, audio and video. A similar change had occurred to printed documents during the 20th century with the arrival of graphic elements and images. In the case of electronic documents, though, new challenges arise: since documents are in digital form, very large databases that can easily be accessed by a large amount of people can be built. In order to be useful, though, these databases must be well organized, and documents within them must be easily retrievable. Since the Information Retrieval techniques developed for textual documents do not suffice for these new kinds of documents, new methods for indexing and retrieval of images, audio and video according to their content must be developed.

1.1 Content-Based Image Retrieval

In this work, we limit ourselves to the study of content-based image retrieval (CBIR): the act of selecting a subset of an image database corresponding to a description given by the user (the query). A CBIR session can typically be summarized as follows (Figure 1).

- a user produces a query representing the image(s) he/she wants to retrieve from the database and submits it to the CBIR system;
- the CBIR system computes the similarity between the query and the images stored in the database; this is done according to the internal description of query and database image;
- the CBIR system returns a list of images sorted according to their similarity to the query;
- the user modifies the query and/or uses part of the result to form a new query.

Fields that can benefit from CBIR applications are almost countless. Among them [55] we can list art galleries, architectural and engineering design, interior design, geographic information systems, scientific database management, weather forecasting, retail, fabric and fashion design, trademark and copyright database management, law enforcement and criminal investigation, picture archiving and communication systems, press agencies, medical analysis, training and education.

CBIR systems can be classified according to their application, but also according to:

- **Database content.** The images contained in a database can be more or less heterogeneous: databases range from very specific databases, where all images are of the same kind and were taken under the same conditions (think of a database of MRI scans of the brain taken with the same machine), to specific, if not all the parameters are constant throughout the database (like a database containing pictures of stamps against a dark background, but where the pictures were taken

using different cameras and different lighting conditions), to monothematic (like the database used by Das et al. in [32], which contains only images of flowers), to heterogeneous (images collected randomly from the Internet, for example). The different kinds of databases will be discussed in Section 2.2.

- **Query form.** Systems accept queries of very different kinds, depending on the kind of data extracted from the database images and the targeted application. Among the various possibilities we have keywords, natural language queries, feature values (color percentages, for example), example images and user-produced pictorial examples. All these kinds of queries will be discussed in detail in Section 2.3.
- **Image Description.** When an image is added to the database, some kind of image descriptor is built. During the retrieval phase, this descriptor is used to judge the similarity between the query and the database image. Hence, the kinds of queries that can be satisfied by the system are tightly related to the kind of data contained in the descriptor. The data that can be associated to an image is of three kinds (Plate 1): semantic information (a description of the image content: subjects, depicted action, place, etc.), primitive information (colors, textures, shapes, edges, visually homogeneous regions, etc.) and factual information (information that cannot be extracted from the image, like name of the photographer, date of the shoot, belonging of the image to a particular series, conditions under which the image was taken, etc.). These different kinds of data will be discussed in detail in Section 2.4.
- **Interaction between user and CBIR system.** Adding interaction between the user and the CBIR system can help achieving better retrieval results. Interaction ranges from simply allowing the user to submit a new query based on a previous one, to giving the user the possibility to select part of the result image as "relevant" and/or "non relevant" (like in the work of Ciocca and Schettini [25]), to allowing the user to visually arrange a small set of the database images into clusters of similar images and letting the system rearrange the whole database according to these actions, as it is the case in the El Niño system [121].

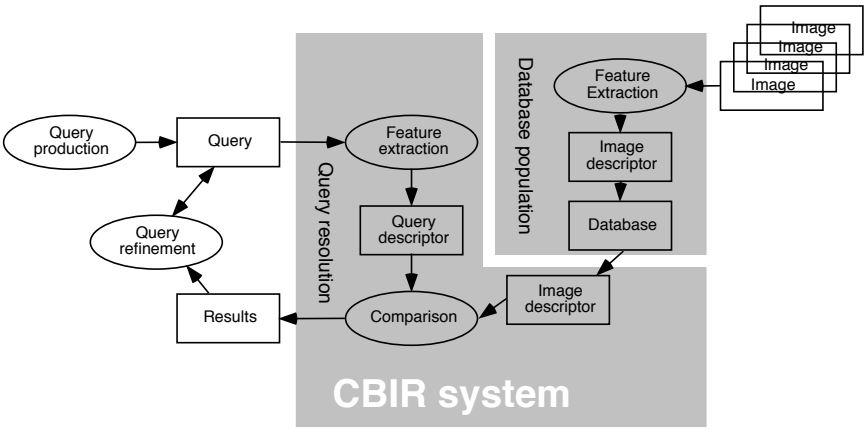

FIGURE 1. The different steps in a CBIR session.

Obviously, the ideal CBIR system would be the one allowing a total freedom query-wise, and in which every image has a descriptor containing a full representation of its semantic, primitive and factual information. Unfortunately, this is not realistic in the general case, because of the following factors:

- extraction of semantic data from general images must be performed by human experts, because image processing techniques for object recognition work only within very strict conditions; the cost of manual processing is extremely high, making its use realistic only for a few selected applications for which precise semantic data is absolutely necessary (like medical applications);
- often, it is impossible to obtain some kinds of data from an image (for example, factual data is in general incomplete, if it is available at all);

- it is impossible to anticipate every query that will be submitted to the system; hence, development of an “universal” image descriptor is utopic;
- complete freedom at query time would make the generation of a query descriptor a very hard task;
- the more complex the image descriptor is, the more complex the techniques that compute the similarity between an image and a query will be; in particular, if combination of different kinds of queries is possible, the production of a single ranking may be problematic.

1.2 The Future of CBIR

It is clear that the rate at which the number of images available to the public in digital form grows will increase in the coming years, because of new image compression techniques, cheaper storage, and faster Internet connections. Hence, the role of CBIR (certainly within the framework of content-based multimedia retrieval) in the future will become even more important.

Furthermore, while systems will certainly be developed for specialized applications (like finding trademarks [95], finding flower patents [32] or finding similar faces [110]), the average user will certainly need CBIR systems able to retrieve images from heterogeneous databases like personal photo collections.

Because of the current lack of efficient and general object recognition techniques, we believe that general CBIR systems will, at least in the short term, focus on extracting only visual information from the images. In particular, techniques allowing indexing and retrieval of the database images according to different uniform areas of the image should prevail over techniques extracting information from the image as a whole. Of course, CBIR systems should make good use of any semantic information available from the images, if such data is already stored with them. An example is extracting semantic information about images contained in html documents by examining the textual information contained in the page, as done in ImageRover [124][80].

In the long term, however, new data formats like MPEG-7 [58] will allow new techniques for indexing and retrieval because the additional data (including semantic information) needed for these operations will be included in the image files themselves. Today it is not clear, though, how this information will be embedded in the files at creation time. Despite this, CBIR systems based on MPEG-7 like the one developed by Paquet and Rioux [107] are already surfacing.

1.3 Goals of This Work

In 1995, when we started working on CBIR, a survey of some of the existing CBIR systems we made [6] showed us that there was room for improvement in several areas. In particular, we decided to focus our attention on three subjects:

- query form;
- image and query description;
- computation of similarity between image and query.

1.3.1 Query Form

In early CBIR systems, queries were either relatively simple (example images like in CANDID [75], feature values as in JACOB [79]) or their resolution relied heavily on data extracted with the help of a human expert (shape sketches in QBIC [43], keyword queries like those supported by Chabot [104]). We felt that some categories of users were being neglected, as no CBIR system suited their needs.

In particular, we considered the following situation: a user wants to retrieve an image he/she knows from a database containing several thousands images. For cost reasons, the information associated to the images was not extracted with the help of human experts (hence, no semantic information is available). In this case, example images and feature values are quite ineffective, as the user may not have an image similar to the sought one readily available, and because it is quite complex for human beings to think of an image in terms of feature values.

We wondered if an approach in which the user draws a sketch of the sought image and the retrieval is based on automatically extracted features was possible. Such an approach allows the user to clearly specify the query and should, in our opinion, achieve better results than the aforementioned methods.

1.3.2 Image and Query Description

The majority of the existing CBIR systems in 1995 relied on global features (image features computed over the whole image, like the color histogram [79]) or on semi-automatically extracted local features (like the color histogram or the contour of a region selected by the user as relevant used in QBIC [43]) or textual descriptions (as in Chabot [104]).

Global features have several shortcomings:

- their discriminative power is limited;
- the contribution of image regions depends on their size (small details are “drowned” in the description);
- searching for a detail contained in a database image is not possible.

On the other hand, in the case of semi-automatically extracted local features, the necessity of human intervention during the database population phase leads to problems due to the cost of the approach (an human operator cannot process more than a handful images a minute) and to the subjectivity of the operator (two different individuals will seldom describe an image with the same words or select the same areas of the image as relevant).

Despite this, extracting features from limited areas of the images is a superior approach. In fact, local features:

- allow the comparison between a subset of the image and the query;
- make querying according to size, position, color, texture, shape, and spatial relations between the extracted areas possible.

Since the size of image databases is constantly growing, human interaction during database population should be as low as possible, so to keep costs at a minimum. We wondered if an approach in which images are represented by local features extracted from automatically selected regions of the image would outperform the CBIR techniques existing at the time. In particular, we focused our research on extraction of features from images segmented according to visual homogeneity. Furthermore, several segmentation results are stored for each database image, so to cope with the unpredictability of the sketches that will be submitted to the system. The feature extraction process is explained in Chapter 6.

The query description is also an important element, as it must be compared to the image descriptors in order to compute a similarity value. We wondered if using the same description for the images and for the queries would simplify the computation of the similarity, all while producing meaningful similarity values. As explained in Chapter 6, sketches undergo the same feature extraction process as database images, but in this case only one segmentation result is stored in the descriptor.

1.3.3 Computation of Similarity

In early CBIR systems, images were often represented by a feature vector, leading to a mapping of the images in a space called “feature space”. Similarity between images was then simply computed with

a distance between the points in feature space. The quality of the results, then, depended on how well distances in feature space approximated the human perception of similarity.

Our interest went towards more complex ways of computing similarity. Since our image descriptor is based on local features extracted from regions obtained by segmentation, we experimented with comparing query and database images at three different levels:

- between regions;
- between segmentation results;
- between query and image.

The goal of such a method of comparison between query and image, which is the subject of Chapter 7, was to allow partial matches, as well as allow a greater flexibility than the methods available at the time. In fact, such an approach is adapted for both precise and general queries. For example, it is possible to produce a very detailed sketch of a particular image, but it is also possible to produce a sketch containing only some shapes and asking the system to retrieve all the images containing similar shapes, independently on their position and color.

1.4 Organization

In Chapter 2, the multiple applications of CBIR will be presented, as well as the several techniques to perform it.

Chapter 4 focuses on a particular CBIR technique: querying by sketch (i.e., retrieving images according to their similarity to a sketch produced by the user and representing the target image or a group of images).

In Chapter 3 some of the currently existing CBIR systems are presented, classified according to the kinds of queries they support.

The result of our research is a measure that estimates the dissimilarity between a sketch drawn by a user and an image. This measure, and the CBIR prototype we built around it, called SimEstIm (SIMilarity ESTimation for IMages) are introduced in Chapter 5.

The dissimilarity measure is computed from a description of the sketch and of the image which contains local information obtained after a segmentation of the sketch and of the image is performed. The extraction of the descriptor is explained in Chapter 6.

The comparison between the image descriptor and the sketch descriptor is the subject of Chapter 7.

In Chapter 8 the results obtained with our dissimilarity measure are presented. The results are also compared with those obtained with other CBIR techniques.

Finally, in Chapter 9 we talk about future options open in CBIR and how our work fits in this framework.

1.5 Color Plates

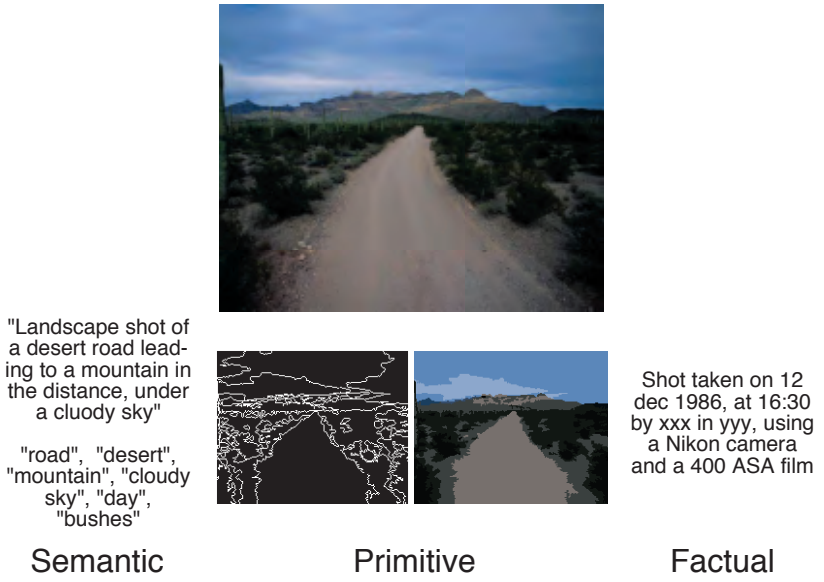

PLATE 1. The three kinds of data that can be associated to an image: semantic (in this case a natural language description and keywords), primitive (for example, contours and a segmentation) and factual (name of the photographer, date and place of the shoot, film and camera used).

CHAPTER 2

Content-Based Image Retrieval

Content-Based image retrieval (henceforth CBIR) is the act of selecting a subset of an image database corresponding to a description given by the user (the query). The goal of this chapter is to give an insight on how this description can be formulated, how relevant images can be selected and how the data allowing this task can be extracted from the database images.

2.1 The Importance of Image Retrieval

Why is finding images in a database that satisfy a particular specification so important? Let's pretend for a moment that you are a stamp collector, that your collection is composed of several thousands stamps, and that you want to be able to make lists of your stamps according to the issuing nation, the date of issue, the nominal value, the current market value, the shape, the pictorial content, the condition (new, used, first day of issue, damaged, ...), the series they belong to, or similarity to a particular stamp. In this case, storing all your stamps in a large box would surely prove to be the wrong solution, since for every search you would probably end up going through the whole collection. It is obvious that you will need to organize your stamps in such a way that the making of these lists will be a manageable task.

Sooner or later, we will all be in this situation: as more and more information is produced as digital images, we will have to find a way to perform efficient searches through the millions of images that will be available to us. Efficient CBIR systems will be needed. After all, "what good is information in the Information Age if you can't find it?" [4]

Being given an image database, what we would like from a CBIR system is the ability to select a subset of the database according to a query submitted to the system. The size of the subset can range from the empty set to the whole database. Since image databases are based on similarity rather than on matching, as pointed out by Santini and Jain in [119], this subset will in general be sorted according to the similarity to the query. In some cases, like in the system developed by Martinez and Guillaume described in [96] , results do not need to be ranked.

In any case, in order to be viable a CBIR system must deliver good performance in the following three fields:

- indexing of the database, which is essential in order to compare the query only to a small subset of the database, the computation of similarity between the query and a database image being in general a time-consuming operation;

- comparison of query and database images, which must correspond as well as possible to the human perception of similarity, so that the retrieved sets will be considered satisfactory by the user;
- retrieval efficiency, since the goal of a retrieval session is to return all the relevant images with a better ranking than the not relevant ones.

CBIR applications are almost countless. Extending the list given by Gudivada and Raghavan in [55] we can list:

- art galleries and museum management;
- architectural and engineering design;
- interior design;
- remote sensing and management of earth resources;
- geographic information systems;
- scientific database management;
- weather forecasting;
- retail;
- fabric and fashion design;
- trademark and copyright database management;
- law enforcement and criminal investigation;
- picture archiving and communication systems;
- press agencies;
- medical analysis;
- training and education;
- industrial quality control.

When trying to classify CBIR systems, big differences can be noticed in:

- the content of the database;
- the form of the query;
- the type of data stored with the images;
- the amount of interaction between user and CBIR system.

In the following sections these four subjects will be discussed in depth. Finally, in Section 2.5 quality measures used for evaluating the results of a CBIR system will be presented.

2.2 Database Contents

Images contained in databases can be of disparate kinds, ranging from a 16x16 2-bit pattern to a 1200 dpi, 32-bit color scan of an A4-size page. Databases containing a more uniform kind of images will in general be easier to handle [4] and will allow more precise searches than heterogeneous databases, since specialized algorithms or domain experts will be available to extract the wanted data. In this section, a classification of image databases according to the variability of the images they contain will be given.

Among the characteristics of an image we have:

- the size of the image (and the aspect ratio);
- the color depth (black and white, grayscale, 8-bit color, 16-bit color, 24-bit color, 32-bit color);

- the conditions under which it was taken (illumination, distance between object and camera, kind of camera, ...), which can be known or not;
- the number of objects/subjects portrayed;
- the knowledge about what kind of object can be in the image (for example: we know that the image is a picture of a dog, a horse or a cat);
- its origin (natural or synthetic);
- the file format;
- whether the objects/subjects are in front of a known background or not;

The variance of these image characteristics within a database allows us to perform a classification of the databases. Henceforth, we will speak of:

- **very specific databases**, when the image characteristics are all known; it's the case of a collection of MRI scans of the brain taken with the same machine under the same conditions (angle, cut, resolution, ...);

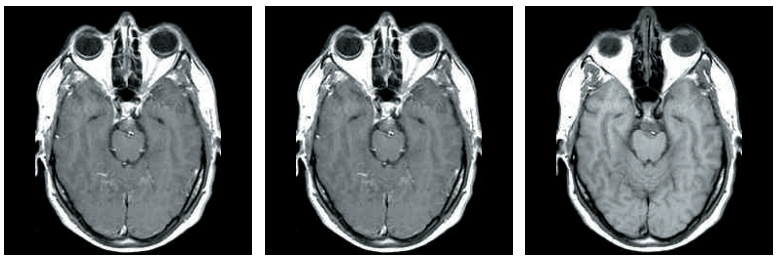

FIGURE 2. MRI scans from a medical database.

- **specific databases**, if some of the image characteristics are not constant throughout the collection, for example in a database containing pictures of stamps, where illumination conditions and distance between the camera and the stamp are unknown, but where other parameters are known (dark background, only one stamp per image, subject placed in the centre of the image, same color depth, ...); another example of a database of this kind is a collection of face shots;

FIGURE 3. Images coming from a database containing face shots.

- **monothematic databases**, when all is known about the images is that every one of them portrays a particular kind of object or subject; a collection of picture of cats is a good example in this case (see Plate 2 for an example);
- **general (heterogeneous) databases**, when nothing is known about the conditions under which the pictures were taken, nor about the portrayed subject; the Internet is a general database (see Plate 3).

Obviously, many other kinds of image databases exist, ranging between the "very specific" and the "general" ones presented above. In this work, though, only these four types will be addressed.

The more specific a database is, the more in depth the analysis of the images it contains can be. Precise data about the images can be extracted, either with highly specialized algorithms or by human experts, since the fact that the images are contained in the database already says many things about their content. With images extracted from general databases this work must be done before the extraction of any precise data.

2.3 The Query

CBIR systems retrieve images from a database according to a specification given by the user. This specification is extracted from the query, whose form depends on the criteria the user wants the images to satisfy, as well as on the kind of answer that the user expects. The form of the query influences the type of data that must be extracted from the database images in order to solve it; this will be the subject of Section 2.4.

2.3.1 What we Expect from a Query

What do we want when we produce a query and submit it to an CBIR system? Are we interested in whole images or objects contained therein? Are we looking for a single image or a collection of images? If it is a collection of images we are looking for, what must they have in common? Do we have an example image available in electronic form or printed form? Are we going to query the database starting from what we remember of an image? All these questions have a deep impact on how the query must be formed [44][4].

Depending on the type of query we intend to submit to the system, different kinds of data will be needed for its resolution: a similarity query based on visual features (i.e., submitting an example image and asking for images visually similar to it) and a conceptual query (composed, for example, uniquely of the word “sadness”) need completely different data in order to be solved. Conceptual queries will require higher level (semantic) data, whose extraction is extremely complex in the general case. For the moment, though, let us pretend that we can extract any kind of data from the database images.

2.3.2 The Multiple Faces of a Query

Querying a database can be done in a multitude of ways, from keywords to natural language, values of features, pictorial queries, example images and so on. In [44], Gupta and Jain suggest that querying in a CBIR system should be performed through a collection of tools including an image processing tool, a feature-space manipulation tool, an object specification tool, a measurement specification tool, a classification tool, a spatial arrangement tool, an annotation tool and a data definition tool. While it is plain that the combination of all these different techniques would allow a large amount of flexibility, the complexity of comparing images to queries formed in such a way would be way higher than that of methods commonly used today. Furthermore, the amount of data to be extracted from the images would be very large, and its organization extremely complex. This is probably the reason why currently most of the existing CBIR systems rely on very few techniques for querying the database, limiting themselves to visual information and keywords, as it is the case in IBM’s QBIC [43], or color information and predefined concepts, like in CHABOT [104], which allows the user to define simple concepts based on color information and keywords. The reader should keep in mind that, even though the techniques will be presented separately in this paragraph, the different methods can (and should) be combined in order to exploit the strengths of each one of them.

KEYWORDS

Querying by keywords is one of the most intuitive ways of querying an image database, since the user only needs to type the words corresponding to (part of) the content of the wanted image(s). If every image could be described using a dictionary of 1000 words, indexing by keywords would even be the perfect method for the task. Unfortunately, this is not the case. Still, keywords prove useful in order to

differentiate visually similar images (in specific or monothematic databases) or to allow a rough indexing of a general database according to the content.

In order to be able to query the database this way, keywords must have been associated to the images during the database population process. This data can be added:

- manually, by having an expert type the keywords that he/she finds relevant for the image; this approach has a high cost, as a human operator can only process a few images every minute;
- by performing object recognition on the image, generally possible only if the database is specific or monothematic, since general object recognition is still an unsolved problem;
- by transforming visual features into concepts, like in Chabot [104], in which a concept is defined as presence of some colors and of some keywords in the image description, and FourEyes [112], which associates names to models based on texture features;
- by analyzing metadata associated with the image (captions, text, technical information about the picture) delivered with the image, as it is done in Piction, which analyzes captions in order to detect faces in the associated images [126].

Even though querying using keywords is semantically powerful, it has some shortcomings:

- information sought is inherently in the form of imagery that a textual language cannot express efficiently, as underlined by Gupta and Jain in [44];
- experience with major archives shows that it is almost impossible to anticipate all likely queries [44];
- if the list of allowed keywords contains a limited number of term, the number of possible queries is also limited;
- if the list of keywords is not limited, synonyms and different languages used must be taken into account;
- if several operators contributed to putting the images in the database, there are chances that the descriptions vary according to the "style" of the operator;
- automatic keyword extraction from visual information can generate false positives and/or negatives, depending on the model used to represent the concept bound to the keyword;
- automatic extraction from textual data can generate false positives if the text is not completely related to the image (for example, a picture of a bicycle rider receiving a prize on the podium after a race may be captioned "despite falling twice, the winner dominated its opponents", and the word "falling" might be taken as relevant).

Keywords can be extremely useful in monothematic databases (for example allowing the separation between different races in a database containing pictures of dogs), when a domain expert can be used to add the data to the images and the dictionary to be used is rather limited (in the dogs database example, the different races of dogs). In the general case, though, problems with synonyms and information stored using different languages arise, although they can be partially solved with automatic translators and thesauri. Automatic extraction of keywords from visual data depends on the model used to represent the various terms, and should be used only in small domains.

Natural Language

Solving natural language queries requires complex techniques, but this way of querying a database is extremely powerful, not to mention the fact that it is the most natural way of querying a database for a user. Unfortunately, lots of semantic data must be stored with every image, as complete annotation (including spatial relationships between objects) of an image containing n objects each with m attributes would require $O(n^2 \cdot m^2)$ entries, as explained by Pentland et al. in [110]. Furthermore, query resolution must take into account the fact that two individuals looking for the same image will almost certainly use quite different queries, leading to a huge number of potential queries for each image in the database. The interested reader will find more information on natural language queries in [117].

Seen from an image processing point of view, the weak chain link is the extraction of the semantic data. The amount of data needed is quite large and, unless the database is extremely specific, automatic extraction of such data is not possible (and even when it is possible, it is not a trivial problem). Hence, a human expert will be needed to perform this task, and the volume of data needed will lead to huge costs, since processing an image will take quite a long time. Furthermore, we will still have the subjectivity and language problems that were present with keywords. Data organization is also a complex problem to be solved in this case.

So far, only researchers at the University of Victoria [137] have tried to implement such a system, which is still in it's early infancy, so it is difficult to say if this way of querying image databases is viable or not. In any case, because of the high cost bound to the manual extraction of the large amount of data needed, such an approach should be used only if the benefits to be obtained from the system justify the huge costs connected with the data extraction.

Feature Values

Queries composed by specifying the values of certain image features (a certain amount of red, green, blue, mixture of sample textures, ...), like in QBIC [43] and JACOB [79] are easily interpreted by the system, but are extremely awkward to compose for the user, who is forced to do "mental gyrations" [44] because of the way she perceives the image isn't organized this way. Furthermore, this kind of query might require some knowledge about the inner workings of the system (to know how textures are mixed, for example). Finally, query results can be very difficult to understand. An example of such a query can be found in Plate 4.

Even though this method seems to have more shortcomings than advantages, it could be useful to select a large subset of the image database in order to speed up the query resolution process.

Example Image

Using an image as the query is a very natural way of looking for visually similar images in a database. Two situations are possible:

- the query image is in the database, like in Filter Image Browsing, developed by Vendrig et al. [141];
- the query image is not in the database, as in the work by Jacobs et al. [69].

In the first case, all the data associated to the database image being used as the query is available (visual information, segmentation, keywords, ...). The ideal situation would have the user select among all this information the part relevant to the query, so to minimize the number of not relevant images retrieved. If semantic data is available, it is even possible to retrieve semantically similar images. The user must browse the database manually to find an image to use as the query; depending on the database and on what the user is looking for, this can take a long time.

In the second case, data must be either extracted from the query image at runtime or supplied by the user, keeping in mind that this information must be compatible with the data stored in the database, so that comparison between the database images and the query image will be possible.

A variation on this technique consists in selecting some areas of the example image and performing the query according to the info contained therein. The user can be asked to draw the boundaries around the interesting areas like Ravela and Manmatha did in [114], or he/she can be asked to select one of the regions extracted by a segmentation algorithm like in Blobworld [17]. This method is particularly useful for finding images containing visually similar regions.

Another possibility is using multiple images as the query, so to stress which visual features are important to the user.

Querying by example image is a technique used by several CBIR systems, some of which will be listed in Chapter 3. It can be used in isolation, or to refine query results obtained with another method.

The latter is what happens in the AltaVista image search engine [1], which uses keywords for the first query of a search session and visual similarity for the following ones.

This technique has some weaknesses:

- changes in position, composition, configuration of the pictured objects will defeat most comparisons [44];
- if the user must browse the database in order to find the images which will compose the query, a long time might be necessary;
- semantic queries won't be successful if the images don't have enough relevant semantic information associated.

User-Produced Pictorial Example

Querying by user-produced pictorial example consists in producing a sketch depicting the sought image(s). This can be done in several ways:

- only sketch the most important edges of the image, or the edges of the sought objects, as possible in QBIC [43] and ETR [34], among others;
- place predefined elements which correspond to objects the image retrieval system is able to recognize in the database images (people, trees, sky, sea, sun, fields, woods, ...), like in ImageSearch [62] and SCORE [4]; an example of an ImageSearch query can be found in Plate 5.
- sketch using a simple paint tool that allows only geometric shapes to be drawn, another way of querying in QBIC [43] (see Plate 6 for an example);
- sketch using a freehand tool, as in PICASSO [30].

Querying by user-produced pictorial example is especially useful when looking for a particular image, as it allows to capture the user's mental image of a specific picture [44] or class of pictures. Depending on the kind of database being searched, a different amount of detail will be needed in order to retrieve a target image: very specific databases (in the medical field, for example) will certainly require more details than homogeneous databases, where painting large patches of colors representing homogeneous areas of the target image(s) will in general be sufficient to retrieve them.

User-produced pictorial examples do not require the extraction of semantic data from the database images, and are, as it will be explained in Chapter 4, a favorite for fully automated systems. Availability of semantic data can, though, make better query results possible. For example, a query composed of a red blob on a larger green blob is not semantically explicit of the fact that the user wanted to retrieve images of red cars in a field. If the keywords "car" and "field" can be added to the query, the number of relevant images retrieved will surely increase, as pictures of red tents in a field will not be retrieved anymore. Furthermore, its visual nature makes it a natural choice for similarity searches, as visualization (the act of looking at an image) and imaging (the act of building a mental image) are extremely similar processes, as explained by Kosslyn in [78].

Combining Different Techniques

Every technique we described above has its strengths and weaknesses: lack of semantic content for visual queries, amount of semantic data needed and problems bound to its extraction for keyword and natural language queries. It would be natural, then, that combining them could generate better results, going a step towards the "query language" defined by Gupta and Jain [44].

Currently, the following combinations have been used:

- visual information and keywords (for example in QBIC [43] and JACOB [104]);
- visual information and semantic information extracted from captions (as in Piction [126]).

2.4 The Image

Before we try to extract information from an image, we must ask ourselves what kind of information it can convey to a human being, and if this kind of data can be useful to process the kinds of query we want to submit to the system. A way to extract and represent this information must then be found, as well as techniques for comparing the database images and the query.

2.4.1 What's in an Image?

By looking at an image, a human being will recognize objects, people, places, maybe understand what was happening when the picture was taken. It is only natural, then, that when a user wants to retrieve an image from a collection he/she thinks of the image in these terms, and would like to query the database according to memories of his/her experience of the sought image. Human perception is not a mere interpretation of a retinal patch, but an active interaction between it and our knowledge about objects [34], which can be influenced by a multitude of factors, including past experiences and cultural background. It can be safely assumed, then, that two individuals will never see exactly the same things when looking at a particular image.

This means that these two individuals will probably query the database differently even if they want to retrieve the same image or the same set of images. Hence, an image retrieval system must be flexible enough to take such situations into account.

Unfortunately, to a computer an image is nothing more than a matrix of n-tuples (the pixels). As of today, computers are not able to analyze an image like a human being would, extracting semantic content from it. Highly specialized algorithms, which require images taken under strict conditions to work properly, are an exception to this rule. All computers can do is try to guess what is in the image by extracting primitive (also called syntactic) information, like color histograms, textures, regions, shapes, edges, spatial positioning of objects, but they aren't able to detect the presence of your uncle laughing with a sunflower in his left hand during a barbecue party in your garden back in 1985.

So, if semantic content is needed manual extraction will be necessary (unless the database is very specific), with all its shortcomings, as it will be explained shortly. Currently, most image retrieval systems (although there are some exceptions appearing) rely on primitive features to try to extract the images' content and to compute the relevance of the database images to the query. This approach can be seen as a first step towards CBIR systems based on semantic features, since "things" are made of "stuff" (i.e., regions of near-constant color or texture), as explained by Forsyth in [44].

2.4.2 Primitive, Semantic and Factual Data

Data associated with an image can be classified into three categories: primitive, semantic and factual. While the first two are related to the content of the image, the latter cannot be deduced by looking at it (date of shoot, place, kind of camera, name of photographer, ...).

Primitive features are low-level visual features, representing roughly the image as it is on the retina and the striate cortex (an example of CBIR system relying on visual features of this kind is Viper [129], in which an image is represented by a subset of 80'000 possible simple features), while semantic features are related to the content of the image as perceived by humans. This paragraph is dedicated to these three kinds of features, while the following one will discuss the extraction of data from the images.

PRIMITIVE FEATURES

Primitive features are those features that relate to the physical appearance of the image. Among them we can list:

- aspect ratio of the image;
- file format;

- color depth: black and white, n-bit grayscale, n-bit color;
- color: average color, color histogram or color correlation for the image or a subset of its pixels;
- texture: physical features of a part (or all) of the image when considered as a single texture;
- edge information: orientation, position and length of edges detected in the image or a subset of it;
- shapes: contour, orientation, elongation, size, bounding rectangle of shapes in the image;
- regions: areas of the image corresponding to homogeneous areas of the image;
- ...

The simpler these features are, the more efficient the algorithms that compute them will be. This means that very simple features (like color, correlations, size, orientation, ...) can be extracted automatically and very quickly. Unfortunately, these features do not convey any semantic meaning (or very little), so an image retrieval system that relies totally on this kind of features will not be able to process queries containing requests for a particular object or about a concept, unless these are defined as given combinations of primitive features (like in Chabot [104]). There are a few exceptions to this, though: if the image database is composed of very specific images (like pictures of pears or apples taken under controlled conditions), some degree of semantic knowledge can be extracted from primitive features through an object model (i.e., we could have an algorithm that classifies the pictured object as “apple” or “pear”). Other ways of extracting semantic data from images are face detection and recognition (see the bibliography compiled by Wahle [143] for more information on this topic) and detection of text in images [145].

Also, primitive features, because of their numerical nature, allow easier similarity computing than semantic features.

Semantic Features

Semantic features are abstract representations of images at different levels of detail, corresponding to human perception of the images [55]. This kind of features includes but is not limited to:

- objects present in the image and their attributes;
- people depicted and their attributes;
- interactions between objects and / or people;
- spatial relations between objects and / or people;
- general descriptions of what was happening at the moment the picture was taken.

Currently, computers are unable (except in extremely specialized applications) to extract semantic data automatically, as this kind of data often needs more information than that available in the image to be extracted [4].

By associating semantic data to database images, it is possible to solve queries about the content of the images, while primitive data only allows queries about the appearance of the images. This additional level of detail is available at a cost, though: semantic data is difficult to represent, and ambiguities in the description of the images’ content can hinder the retrieval process.

For example, if the semantic data is stored as keywords chosen from a closed list, only queries containing these keywords will return satisfactory results: if you are looking for pictures of daffodils, but the only related keyword in the list is “flower”, there would be no way of solving your query, even if a thesaurus was used (the system would probably end up retrieving all the pictures containing flowers). To circumvent this problem, a combination of keywords and primitive features might be useful: combining the word “flower” with the color “yellow” would probably return a subset of the database containing several relevant images. On the other hand, if the list is open, synonyms come into play: two different operators might use different words to describe exactly the same object, and this would have to be taken into account in the query engine, by using a thesaurus like WordNet [100], for example.

Even more complex representations can be extracted by descriptions of the image given in natural language. In order to be usable, these need to be converted into a data structure, like in Piction [126], a task that can be extremely complex. Queries to a system containing such data need to be transformed in order to be comparable to this structure, which must take into account synonyms and ambiguities typical of natural languages. This structure should be independent on the chosen language (automatic translators being still in their infancy), lest the database not be usable by users that do not speak the language used to enter the image descriptions. Furthermore, methods for computing the similarity between two such representations should be developed, a task that may reveal itself to be non-trivial.

Factual Data

Images sometimes come with some attached data, like the date of production, name of the photographer / computer artist / program that generated the image, kind of film the picture was taken with, etc. This information (metadata) isn't semantic, but conveys some information about the image and can be used with primitive and / or semantic data for retrieval purposes [104]. Typically, publishing companies, art galleries and museums need this kind of data.

2.4.3 Extracting Information from the Image

Like features, which can be separated into primitive and semantic, techniques for extracting content from an image can also be classified into categories, depending on the degree of interaction between human and computer needed: we talk of automatic, semi-automatic (or assisted) and manual techniques. As it will be seen, extraction of some kinds of features isn't feasible without human interaction, while in other cases, human interaction isn't necessary (or would even be counter-productive).

Automatic, semi-automatic and manual feature extraction

As we mentioned before, features can be extracted automatically, with human interaction (semi-automatically), or manually.

Automatic extraction can be used only for the most primitive features, like color (computing the average color, the color histogram or color covariances of an area of the image) or size of a region of the image. In these cases, human interaction would even be a nuisance, as computer algorithms are perfectly adapted to the task.

Other primitive features, like edges and regions, might need some human interaction if exact results are needed. In fact, current edge-detection techniques still produce false positives, while region segmentation algorithms for general images aren't 100% reliable. Characterization of texture can also need some interaction, as the operator is asked to select a part of the image containing the texture to be classified. However, these features can be extracted automatically if the database is specific, or if only "rough" results are needed (most evident edges, rough segmentation, ...).

Semantic features, on the other hand, must be extracted by hand, except in some particular cases (simple object recognition in specific databases, for example).

It must be remembered that human interaction is expensive [3]. Furthermore, subjectivity has a major role in the quality of the extracted features, as two individuals will have different perceptions of the same image, because of physical, cultural and psychological reasons. Finally, an appropriate representation for this kind of features must be found. In the following paragraphs, the pros and cons of human interaction and automatic feature extraction will be listed.

Human intervention: advantages and shortcomings

Human interaction when extracting information about images prior to adding them to a database has both advantages and shortcomings. Let us begin by listing the advantages:

- possibility to have a domain expert extract the information, which will be more accurate, allowing more precise queries and thus better results;

- possibility to perform tasks that are poorly (if at all) executed by computer algorithms in the general case;
- possibility to correct information extracted by computer algorithms;
- possibility to extract semantic features, something in general impossible with computer algorithms.

These advantages seem to justify this method. Unfortunately, human interaction also has several shortcomings:

- its cost is very high, as a human being will not be able to process more than a handful of images per minute; the more complex the data to be extracted is, the more time the operator will need to extract it;
- human beings grow tired (especially if the work is very demanding) or have a drop in attention (if the work is tedious), and thus tend to make errors;
- two human beings see the same image differently, giving different interpretations;
- if image descriptions provided by the experts are written in a natural language, ambiguities bound to the nature of natural languages arise.

Automatic feature extraction: advantages and shortcomings

Being given an image, what kind of content information can be extracted automatically? Today, what can be extracted is probably enough to perform simple pictorial queries: color information, texture information, shapes, edges, even some semantic information if the image is part of a multimedia document (like an html page, for example, which contains text that could be related to the image, or a caption).

The obvious advantages of a fully automatic method are:

- extremely low cost;
- processing will normally be faster if no human interaction is needed;
- results will not depend on subjectivity, tiredness or attention of the operator;
- extracted information will be numerical, hence universal.

Shortcomings are just as obvious:

- extracted features are extremely simple; there is no way to extract real semantic information from the images;
- except for very simple tasks, algorithms can make mistakes and will not learn to correct them by themselves (human feedback will be needed);
- some tasks (like segmentation into regions of a color image) are not fully mastered in the general case;

Human vs. the Machine: what to choose?

After listing the relative strengths and weaknesses of automatic and human-assisted feature extraction, it becomes obvious that the choice of which technique to use will depend on several factors, among which:

- **the kind of images contained in the database:** a database containing only texture samples (like Brodasz [15]) will certainly require less human interaction than a database containing pictures of opera singers, where identifying the pictured subjects will require the help of a domain expert; in general, for images containing little semantic information human interaction won't be needed;
- **the amount of images to be processed:** the more images are contained in a database, the more information will be needed to discriminate between them. Primitive features might reach their limit (it has been proved that the discriminative power of color histograms [135] and texels [62], for example, is limited). Semantic data, for which human intervention is in general needed, may

be needed in order to increase the discriminative power of the image descriptions. It must be noticed, though, that for very large databases this might lead to high costs;

- **the cost of the processing:** automatic extraction of data from images has the advantage of being relatively cheap, since the only human intervention needed is at programming time, and that computing time is nowadays extremely cheap. The cost of human interaction, on the other hand, depends on the quantity of data to be extracted from the images and on the number of images to be processed. Such cost may become prohibitive is the database is very large, or if large amounts of new images must be added at a regular basis;
- **the amount of detail needed:** human interaction may be required if very precise information is needed, like in the case of tumor detection in MRI scans.

Currently, one of the major challenges is the Internet, which can be seen as a huge collection of heterogeneous and unannotated images, growing at a stellar rate. In this case, human interaction is impossible, mainly for cost reasons. Other CBIR applications, on the other hand, would certainly make profit of the combination of manually, semi-automatically and automatically extracted data. This work aims at developing methods that can be used to perform retrieval in a general database like the Internet. This is the reason why the proposed method, presented in detail in Chapter 5, relies on automatically extracted features.

2.5 Quality Measures

As it was explained in the previous sections, a multitude of ways for querying an image database exist. Currently, there is a large number of CBIR systems available (commercial or not), 40 of which were compared by Gudivada in [54]. In order to choose one system or one technique over another, users need quality measures allowing them to evaluate their usefulness. Surprisingly, this subject is somewhat neglected within the CBIR community [37].

A CBIR system (just like any other information retrieval system) should retrieve as many relevant images as possible, while rejecting a large quantity of the extraneous images, thus achieving good retrieval effectiveness. In order to compute quality measures, then, we have to define what relevance means. We can list at least two definitions (the first one being the most used one):

- relevance is the correspondence in context between an information requirement statement (a query) and an article (a document), that is, the extent to which the article covers the material that is appropriate to the requirement statement [30];
- the relevant set of items is the subset of the stored items that is appropriate to the user's information need at the time of retrieval [52][53]; thus a document may be relevant if it deals with the appropriate topic classes but it may not be pertinent if the user is already acquainted with its contents, or if other documents retrieved earlier already cover the appropriate topics [118].

On the other hand, user effort time and cost needed for the retrieval should be minimized, obtaining high search efficiency. In this section, only effectiveness measures will be discussed. The interested reader can find more information about efficiency of information retrieval systems in [118].

According to Swets [136], the ideal effectiveness measure should have at least the following properties:

- the measure should be able to reflect retrieval effectiveness alone, separately of the criteria such as cost;
- the measure should be independent of any particular retrieval cutoff, that is, of the number of documents retrieved in a particular search;
- the measure should be expressible as a single number, which can be put on a scale to give absolute and relative values.

Many effectiveness measures exist, the most famous of which are precision and recall. They will be explained in this section, along with two alternate measures: Swets' E-measure and Dimai's RDTA.

The reader should be warned, though, that in order to use these quality measures a relevance judgement for every query may be needed. In particular, this can mean knowing which images in the database are relevant according to a particular query. This cannot always be done, because of the size of the database being searched, but also because relevance judgements can be extremely different depending on the person performing the task.

2.5.1 Precision and Recall

These two measures are by far the most used effectiveness measures used for information retrieval systems. Both of them depend on the amount of documents relevant to the query that were retrieved, thus they depend on a definition of relevance. Here, the objective view of relevance given by Cuadra and Katter in [30] will be used: "relevance is the correspondence in context between an information requirement statement (a query) and an article (a document), that is, the extent to which the article covers the material that is appropriate to the requirement statement".

Recall is defined as the proportion of relevant documents retrieved, while precision is the amount of retrieved documents that is relevant [118]. Both these measures depend on the number of retrieved documents, as well as on the availability of a relevance judgment for every document and the current query. Obtaining such a judgment can be problematic, especially for very large databases. Furthermore, it has been observed that human operators do not perform this task perfectly.

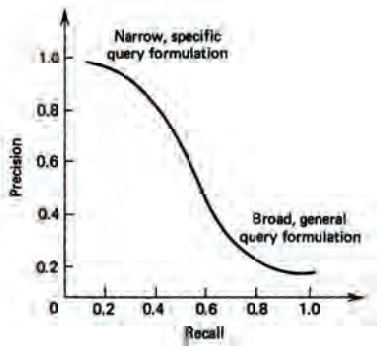

FIGURE 4. A typical precision/recall graph (from [118]).

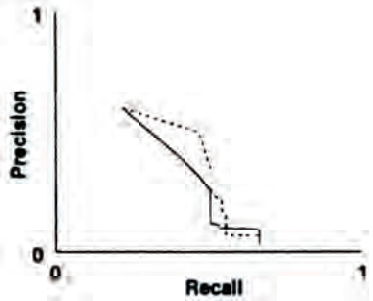

FIGURE 5. It is sometimes hard to decide if a system performs better than another by looking at their precision/recall graphs (from [37]).

By varying the number of retrieved documents from 1 to the size of the database, a precision versus recall graph like the one in Figure 4 can be obtained . By overlaying the graphs generated by two sys-

tems, a comparison between the two can be made, even though in some cases, like the one pictured in Figure 5, the comparison is not straightforward.

Standard precision and recall have several shortcomings among which are their dependence on the number of documents retrieved and the fact that they do not take into account the actual ranking of the relevant documents.

To overcome these weaknesses, normalized versions of these two measures were introduced in [118]. Let REL be the size of the set of relevant images, N the size of the database and $RANK_i$ the position of the i-th relevant image in a particular query. Then, the normalized recall is computed as in Equation 1:

$$RECALL_{norm} = 1 - \frac{\sum_{i=1}^{REL} RANK_i - \sum_{i=1}^{REL} i}{REL - (N - REL)} \quad \textbf{(EQ 1)}$$

This measure says how close to the ideal result (in which the relevant images are retrieved in positions $1, \ldots, REL$) the retrieval result is. In a similar way, normalized precision (Equation 2) shows how close to the ideal one the obtained precision curve is.

$$PRECISION_{norm} = 1 - \frac{\sum_{i=1}^{REL} \log RANK_i - \sum_{i=1}^{REL} \log i}{(\log N!)/((N - REL)! \cdot REL!)} \quad \textbf{(EQ 2)}$$

2.5.2 Other Quality Measures

There are a multitude of effectiveness measures available in the literature (see [81] [76] [77] for more information). Here, two of them will be shortly presented: the E-measure and the Rank-Difference Trend Analysis (RDTA).

E-MEASURE

Swets [136] associates two populations POP_1, POP_2 to the relevant and nonrelevant documents according to a given query. A parameter r (the query-document similarity) is then used and two "probability density functions" $FUNC_1(r)$, $FUNC_2(r)$ indicate the probability that a document of one of the populations has of having score r. By choosing increasing values of r, computing the percentage of relevant and nonrelevant documents retrieved and plotting these percentages one against the other, an operating characteristic (OC) line is obtained.

Swets' E-measure is then proportional to the distance between this line and the line obtained when the two populations have exactly the same distribution. The slope of the OC is also associated to the E-measure.

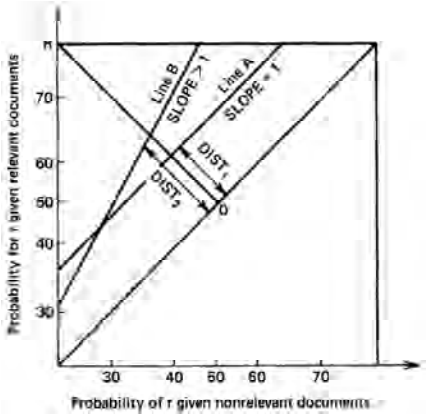

FIGURE 6. Operating characteristics (OC) on normal probability scales. Swets' E-measure is $\sqrt{2} \cdot DIST_i$ (from [118]).

RANK-DIFFERENCE TREND ANALYSIS

Dimai [37] presented a novel method for assessing the effectiveness of two CBIR systems A and B, based on the statistical comparison of the rankings of relevant data.

The evaluation is done in three steps:

- production of the rankings;
- determination of the rankings of the relevant images $\{r_{A_i}\}, \{r_{B_i}\}$;
- comparison of $\{r_{A_i}\}, \{r_{B_i}\}$; an assessment measure $\mu(A, B)$ representing the discrimination of one system with respect to the other is obtained.

$\{r_{A_i}\}, \{r_{B_i}\}$ are compared by linear regression $r_{A_i} = b \cdot r_{B_i}$, and b is determined using a statistical analysis, with the low-ranked relevant data being weighted more than high-ranked relevant data. The comparison measure $\mu = 4/(\pi \cdot \text{atan}(b)) - 1$ is equal to zero if the rank of the relevant images is the same in both systems, is negative if the ranking in A is better than that in B and positive otherwise. During the computation of μ, a goodness of fit measure is also computed, which allows to determine if the computed value μ is significant or not.

2.6 Conclusion

In this chapter, some of the fundamental elements of a CBIR system have been discussed: the image database and its content, the form of the query, the kinds of data associated to the images and their extraction. The description of the various possible organizations of a database were voluntarily omitted.

CBIR applications are almost countless, and almost as disparate: a system built with the intent of classifying textures has little in common with a system allowing natural language queries like "find pictures of people walking hand in hand" or "find pictures of dogs playing". The ideal system, though, should combine all the possibilities and allow the user to choose how to query the database at any given time during the search process.

This work positions itself in the framework of user-produced pictorial queries in a large general database. For this reason an approach based on automatic extraction of data from the images and querying

according to visual cues was chosen, a subject that will be discussed in Chapter 4. Chapter 3, on the other hand, contains an overview of existing CBIR systems.

2.7 Color Plates

PLATE 2. **Images from a database containing only images of cats; the only thing these images have in common is the fact that cats are their main subject.**

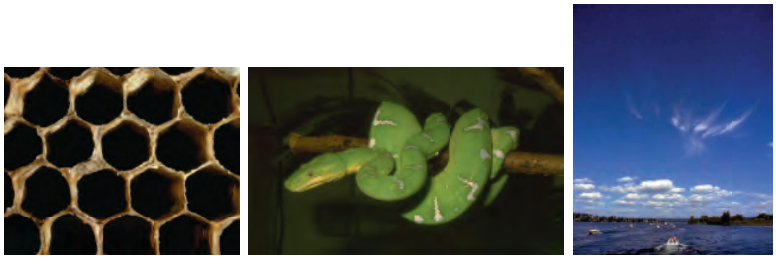

PLATE 3. **Images extracted from a general database; there are no relationships between the images.**

PLATE 4. **In QBIC, the user can query the system by specifying the amount of colors that are to be found in the target images (image taken from http://wwwqbic.almaden.ibm.com/cgi-bin/stamps-demo/drawpicker).**

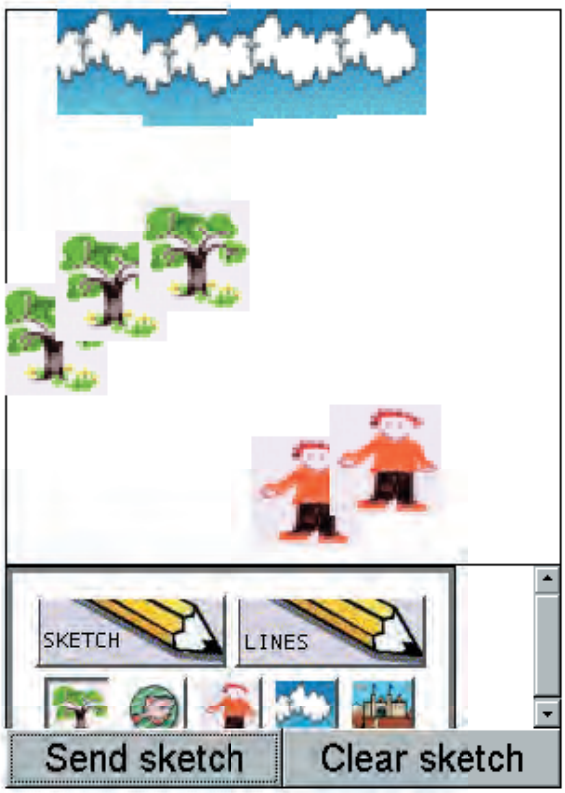

PLATE 5. A sample ImageSearch query (taken from http://ind134a.wi.leidenuniv.nl:2001/test.html).

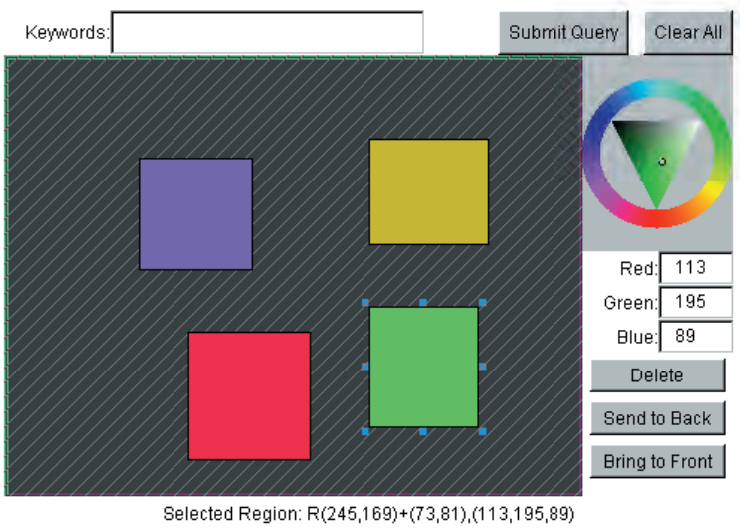

PLATE 6. A painted query in QBIC consists of rectangles of uniform color (taken from http://wwwqbic.almaden.ibm.com/cgi-bin/stamps-demo/drawpicker).

CHAPTER 3

Content-Based Image Retrieval Systems

In this chapter, a selection of Content-Based Image Retrieval (CBIR) systems, both commercial and not, will be presented. For each one of them, the following subjects will be discussed:

- query form;
- database content;
- application;
- image representation;
- query representation;
- query resolution.

While being aware of the current state of research in the CBIR field will certainly prove useful to the reader for the remainder of the reading, it is by no means necessary to fully master the inner workings of all the presented systems in order to be able to understand the content of the following chapters.

CBIR is a fast-moving research area: new systems and techniques appear every month, while "old" systems are continuously updated. This is the reason why this chapter should be taken as a survey of some of the systems documented at the time of writing (october 1999), keeping in mind that, by the time these pages will reach the reader, many things will have changed and new, better systems, may have surfaced.

The systems will be classified according to the form of the queries they accept, according to the taxonomy presented in Section 2.3:

- keyword and natural language queries;
- feature values;
- example image;
- user-produced pictorial example.

Systems accepting queries of several kinds will be presented only in one of the corresponding sections, but will also be mentioned in the other sections they would also belong to.

The reader will notice that, in general, no mention of the systems' performances is made. There are two main reasons behind this choice. First of all, in the journal and conference papers, as well as in the reports this survey is based upon, generally only a few results are presented. Furthermore, in many cases the size of the database or the number of performed tests was quite small. While these results

can give an indication of how well the system performs, they may also not be fully representative. Moreover, in most cases it is not clear how the system would perform in a real-life case (i.e., if the database was composed of several thousands images, for example). For these reasons, it is left to the reader to investigate further about the systems he/she finds interesting. For this purpose, bibliographical references are given for all the surveyed works.

3.1 Keyword and Natural Language Queries

3.1.1 Chabot

AUTHORS

V. Ogle, M. Stonebraker [104].

QUERY

Queries are composed of textual information, date information, numerical information and color information reflecting the target image's data. An example of the Chabot interface can be seen in Plate 7.

DATABASE

The database was composed of 15000 images coming from the State of California Department of Water Resources (DWR), which contains a growing collection of over 500000 images.

APPLICATION

Searches within the DWR database, according to image descriptions stored with the images in a Postgres [113] database, as well as automatically extracted color image content.

IMAGE REPRESENTATION

A part of the DWR images is annotated with data of various kinds: textual (category, subject, organization requesting the shoot, keywords, name of photographer, film format, orientation of the image, name of the indexer, comments, location), date (date of shoot, date of entry in the database) and numerical (CD disk number, CD image number, DWR id, job request number). A 20-color histogram is added to this information, giving the user the possibility to query the database according to color information. All the image descriptors are stored in a Postgres database.

QUERY REPRESENTATION

The user can specify values for any of the textual, date and/or numerical fields. In addition to this, queries about colors and concepts are also possible. Queries are transformed into Postquel queries (Postquel is a query language similar to SQL which was written for Postgres).

In color queries, the user can request that the images contain "some" or that it is "mostly" of a given color. Containing "some yellow" means that at least two of the image's histogram bins classify as yellow. On the other hand, more than 50% of the image pixels must be red for the image to meet the "mostly red" criterion. A boolean function with a color criterion and a histogram as parameters is defined, and will be used by the Postgres query executor to solve queries containing color queries.

The user can also compose higher level queries containing conceptual info, such as "sunset" or "snow". In order to do this, the concepts must be defined previously as a Postquel query. For example, the "purple flowers" concept can be defined as "the image q contains some purple and the word 'flower' appears in its description", which translates into the following Postquel statement:

```
q.description~"flower" and MeetsCriteria("SomePurple",q.histogram)
```

QUERY RESOLUTION

As it was said earlier, the user queries are converted into Postquel queries. These are then solved by the Postgres query executor, and the images satisfying the query are returned. Since Postgres is based on matching and not on similarity, all the retrieved images are considered equally relevant. A query result, where the user asked for images containing some orange and taken in the San Francisco Bay area since 1994 can be seen in Plate 8.

3.1.2 Piction

AUTHOR

R. Srihari [126].

QUERY

Piction support four kinds of query:

- text-based objective text search (exact match);
- text-based content term similarity (inexact match);
- image-based objective term similarity (exact match);
- image similarity (inexact match).

DATABASE

The database used in the experiments was composed of 50 grayscale images. A caption in English was associated to each image.

APPLICATION

Piction was developed as a CBIR system specialized for searches in databases of captioned images.

IMAGE REPRESENTATION

Piction extracts information from images based on the associated caption. In particular, it tries to detect faces and associates them to the people mentioned in the caption. Hence, an image is represented by its caption and three types of constraints: spatial (about the spatial relationships between the detected faces), characteristic (for example, the gender of the pictured subjects) and contextual (the fact that the a particular subject is in the picture).

QUERY REPRESENTATION

A query can be textual, visual (an example image) or both. Features extracted from the query image are the same as those extracted from the database images (i.e., faces are detected and their spatial relationships are computed).

QUERY RESOLUTION

Similarity based on text is computed by estimating how close the words used in the query are to those in the database image's caption. Visual similarity is based on the position, size and orientation of the detected faces in the query image and in the database image. The global similarity is a weighted sum of exact text similarity, inexact text similarity, exact image similarity and inexact image similarity.

3.1.3 Other Systems

Of the systems mentioned elsewhere in this chapter, three in particular use keywords to retrieve images from a database, at least at some stage of a search session: WebSEEK (Section 3.3.17), ImageRover (Section 3.3.18), Flower Patent Retrieval, presented in Section 3.3.19 and QBICAT (Section 3.4.8).

3.2 Query by Feature Values

3.2.1 JACOB

AUTHORS

M. La Cascia, E. Ardizzone [79].

QUERY

Feature values (color histogram, texture features) or example image. The feature values query interface can be seen in Plate 9.

DATABASE

Color images extracted from video footage.

APPLICATION

Content-based video retrieval: the stored images are "relevant images" of video shots. By retrieving an image, the user can view the associated shot.

IMAGE REPRESENTATION

Images contain three features:

- a 512-bin RGB histogram;
- an 8-dimensional vector obtained by computing

$$\max_{m,\,n} f(m, n, r, \theta)$$

$$\sum_{m}\sum_{n} n \cdot [f(m, n, r, \theta)]^2$$

  where $f(m, n, r, \theta)$ is the probability that two pixels whose gray levels m, n are at distance r and orientation θ. The authors used $r = 1$ and $\theta = \{-\pi/4, 0, \pi/4, \pi/2\}$;
- a 4-dimensional edge intensity vector obtained computing the ratio between the pixels whose intensity gradient is above a given threshold and the total number of pixels at 4 orientations $(0, \pi/4, \pi/2, 3\pi/4)$.

QUERY REPRESENTATION

The features extracted from the query are identical to those extracted from the database images.

QUERY RESOLUTION

Computation of distance between the query and the database images is based on vector distance between the individual feature vectors. These distances are then combined according to weights given by the user. Examples of query by feature values and query by example results can be seen in Plate 10 and Plate 11.

3.2.2 Colour Image Retrieval Fitted to "Classical" Querying

AUTHORS

J. Martinez, S. Guillaume [96].

QUERY

Queries are OQL (Object Query Language) queries, which are submitted to an object-oriented database management system (OODBMS). Queries specify the features of the regions contained in the target images.

DATABASE

The database used in the experiments was composed of 147 color images.

APPLICATION

Image search according to features of regions in the database images.

IMAGE REPRESENTATION

An image is represented as a set of regions, which are extracted by a region segmentation algorithm aiming at forming large uniform regions.The features of a region are its bounding box, its centroid, its size and its color (chosen from a list of nine colors covering the whole HSV color space). Spatial relationships between regions are also taken into account. Furthermore, the regions' features are used to generate conceptual information, like "small", "big", "bottom", "top", and so on.

QUERY REPRESENTATION

In order to query the database, the user must compose an OQL query, like in the following example, where the user is looking for a picture of a sunset:

```
select p
from p in Pictures
where exists sun in p.Regions:
      (sun.yellow and exists sky in p.Regions:
            (sky.organge and sky.adjacent(sun)) and
exists ground in p.Regions:
      (ground.dark and ground.bottom and not ground.small)
```

QUERY RESOLUTION

The query is solved by the OODBMS engine, which returns all the images matching the query. Since all retrieved images match the query, no ranking is needed.

3.2.3 Other Systems

Among the other systems presented in this chapter, the one developed by Cha and Chung (Section 3.4.7), QBIC (Section 3.4.6) and WebSEEK (Section 3.3.17) allow querying by specifying feature values.

3.3 Query by Example Image

3.3.1 NETRA

AUTHORS

W. Ma and B. Manjunath [88].

QUERY

The query is composed of a region coming from a pre-segmented database image. The user selects the region and can query the database according to similarity in the color, texture, shape and / or position domains. It is also possible to directly select colors from a color codebook and to draw a rectangle to specify the position of the wanted region. The query interface can be seen in Plate 12.

DATABASE

The database is composed of 2500 pictures, divided into 25 categories of 100 images each. Images are extracted from the Corel database. The search is performed over the whole database. All the images are outdoor shots.

APPLICATION

Search for images having a visually similar region.

IMAGE REPRESENTATION

Images are segmented into homogeneous regions using a robust segmentation algorithm based on edge-detection [89], which exploits color and texture information (Plate 13). Regions are characterized by the following features:

- **color:** a color codebook composed of 256 colors in the RGB color space is used, and the colors actually present in the region are stored;
- **texture:** the texture representation scheme is based on a Gabor decomposition [90]; for each region, a texture feature vector of size 48 is extracted;
- **contour:** each region contains chains representing its contour as curvature, distance from the centroid and complex coordinates, as well as Fourier-based shape descriptors;
- **spatial:** the region centroid and the region's bounding box are recorded.

QUERY RESOLUTION

In case the user submitted a query including more than one domain (texture, color, shape, position), separate queries are solved and an intersection of the results is made. A sample query result can be seen in Plate 14. Similarity between database images and the query is computed as follows:

- **spatial:** spatial similarity is based on the position of the region centroid, the overlapping of the region's bounding boxes or the complete inclusion of the region's bounding box in the rectangle drawn by the user;
- **texture:** texture similarity is computed as the normalized euclidean distance between the texture vectors;
- **shape:** an euclidean metric is used to compute the distance between two shape feature vectors;
- **color:** the authors developed a color difference measure between two regions A and B, which is a sum of the distances from every color in A to the closest color in B and vice-versa. The difference measure is not a distance, as the triangular inequality does not hold.

3.3.2 PICTOSEEK

AUTHOR

Theo Gevers [51].

QUERY

Example image, not necessarily contained in the database. The user can choose which color representation and which image similarity function to use, as it can be seen in Plate 16.

DATABASE

Images retrieved from the Internet and automatically classified in two categories: graphic (computer-generated) and photographic (generated by a camera). The size of the database is of about 100000 images.

APPLICATION

Retrieval of visually similar images, and in particular of images containing the same object under different viewpoints and illumination conditions.

IMAGE REPRESENTATION

Images are represented by histograms based on color information. Since the goal of the system is to retrieve images containing the object pictured in the query image, color features should be invariant to change in the:

- viewpoint;
- object orientation;
- intensity and direction of the illumination.

This is obtained by introducing the m color ratios, based on the RGB values computed at two locations x_1, x_2.

$$m_1 = \frac{R^{x_1} \cdot G^{x_2}}{R^{x_2} \cdot G^{x_1}}, m_2 = \frac{R^{x_1} \cdot B^{x_2}}{R^{x_2} \cdot B^{x_1}}, m_3 = \frac{G^{x_1} \cdot B^{x_2}}{G^{x_2} \cdot B^{x_1}}$$

The color content of the images is represented as an histogram for m, RGB, rgb, Hue, Saturation and Intensity.

QUERY RESOLUTION

Similarity between the query and a database image can be computed either with histogram cross correlation, which works better when no object clutter is present, or with the quadratic similarity function (histogram intersection). Any of the abovementioned color histograms can be used.

The user chooses both the similarity function and the color representation to use.

3.3.3 Leiden 19th Century Image Database

AUTHORS

M. Lew, D. Huijsmans, D. Denteneer [62][83].

QUERY

Example image.

DATABASE

Scans of grayscale portrait images taken during the 19th century. Since at the time several copies of each image were made, the database contains several examples of the same image, with different amount of noise due to the different kinds of abuse the images went through (scratches, bleeding, writing, cutting, etc.).

APPLICATION

Searches for copies of the query image, or of images very similar to it, in a database of 19th century grayscale pictures.

IMAGE REPRESENTATION

Several kinds of representations have been used in this work, namely:

- projection vectors [61]: row and column averages of the intensity values;
- local binary patterns (LBP) [105]: distribution of the 3x3 binary patterns obtained from the image by using the center pixel of the window as a threshold. The LBP representation of an image is an 256-value histogram representing how many times each possible pattern occurs in the image;
- trigrams: distribution of the 3x3 binary patterns obtained by thresholding the gradient image with a noise level threshold. The trigram representation is a 512-value histogram;
- optimal keys: keys are defined as functions that map images onto one or more feature values, so that the difference between two images X, Y can be computed as

$$\sum_m (k(X, m) - k(Y, m))^2 \quad \textbf{(EQ 3)}$$

Optimal keys maximize the probability that the key computed from a corrupted version of an image is closer to that obtained from its original than to that of any other image in the database. The authors supposed that keys are linear, i.e. that adding noise n to image X results in $k(X + n) = k(X) + k(n)$. If noise can be modeled, optimal keys can be computed by maximizing $k^t \Sigma_x k$, with $k^t \Sigma_n k = 1$, where Σ_x, Σ_n are the covariance matrices of X and n. Otherwise, this condition becomes $k^t k = 1$.

QUERY RESOLUTION

The similarity between query image and database image is computed by using only one of the above representations. For projection vectors, LBP and trigrams, the L_1 norm is used, while for optimal keys Equation 3 is used. The database is then ranked using the computed dissimilarities, and the most similar images are returned (Plate 15).

3.3.4 Virage Similarity Engine

AUTHORS

R. Jain, B. Horowitz, C. Fuller, A. Gupta, J. Bach, C. Shu [72].

QUERY

More than a complete CBIR system, the Virage Engine is a library allowing developers to build CBIR systems. Its main features are the ability to perform image analysis (either with predefined methods or with methods provided by the developer) and to compare the feature vectors of two images. It must be said, though, that systems based on the Virage Engine are strongly geared towards query by example and query by sketch.

DATABASE

Being extensible, the Virage Engine can be adapted to any kind of database, as long as the appropriate image analysis and feature comparison functions are provided by the developer.

APPLICATION

General content-based image retrieval engine.

IMAGE REPRESENTATION

Right out of the box, the Virage Engine is able to extract four kinds of features from images:

- global color, representing the distribution of colors within the entire image;
- local color, representing the colors in the image, emphasizing where they are located;
- structure, representing the directions of edges detected in the image;
- texture, representing the periodicity, randomness and roughness of patters found in the image.

More features can be added by the developers. To do this, a feature extraction and a feature comparison function must be provided.

QUERY REPRESENTATION

Beside being able to extract feature descriptions from images, the Virage Engine is also able to compute a dissimilarity score between two images, which is based on the images' feature vectors. The application using the Virage Engine can also feed the Engine a vector containing the importance weights for the single features.

QUERY RESOLUTION

When two feature vectors and a weights vector are passed to the Virage Engine comparison module, the feature vectors are split into vectors representing the values of the single features. Then, comparison is done independently for every feature, giving an array of dissimilarity scores, which are com-

bined linearly according to the values in the weights vector. The result is a dissimilarity score between 0 and 100.

3.3.5 Photobook

AUTHORS

A. Pentland, R. Picard, S. Sclaroff [110].

QUERY

Queries are composed using example images (multiple examples are allowed).

DATABASE

The authors present three different versions of Photobook, called Appearance Photobook, Shape Photobook and Texture Photobook. These were tested using different databases. For Appearance Photobook, a database composed of 7562 color images of faces portraying about 3000 different subjects was used; Shape Photobook was tested using a database composed of 60 images of 12 hand tools and a database containing 74 pictures of tropical fishes shot on a white background; for Texture Photobook a database composed of 1008 images extracted from the Brodatz album [15] was used.

APPLICATIONS

Here again, applications differ depending on which version of Photobook is used. Appearance Photobook, applied to face databases, can be used for customs, security, criminal investigation, dating services, etc. Shape Photobook is useful for browsing catalogues of consumer goods, inventories of mechanical parts, botanical or biological catalogues. Texture Photobook is developed for finding texture patches in the design and decorating industries.

IMAGE REPRESENTATION

In Appearance Photobook, first images are normalized for position, scale, orientation and similar nonlinear effects; then eigenvectors of the normalized image covariance are computed for a set of training images and subregions of these, generating representations for both the whole object and its subfeatures (for faces: the eyes, the mouth, the nose). This way, a class model is built. The textual information added (manually) in the image descriptors includes sex, race, apparent age, facial expression and other salient features.

In Shape Photobook, a symmetric matrix (the stiffness matrix) is computed from a physical model of a shape. This matrix describes how each point of the object is connected to each other point, and plays the same role of the covariance matrix in Appearance Photobook.

In Texture Photobook, textures are represented according to a model based on the Wold decomposition [45], which is a sum of 3 mutually orthogonal components: a harmonic field, a generalized-evanescent field and a purely-indeterministic field. The Wold decomposition produces compact texture descriptions that preserve most of a texture's perceptual attributes [131].

QUERY REPRESENTATION

A query is composed of an example image; in Appearance Photobook multiple examples are allowed. In this case, a mean image is computed. The features extracted from the query are the same as those extracted from the database images.

QUERY RESOLUTION

In Appearance Photobook, the similarity between two images i, j is computed by comparing their within-eigenimage-subspace distance $d_{ij}^2 = \|\Omega_i - \Omega_i\|^2$., where Ω_i describes the input image i in term of the orthogonal eigenfeature basis set. The Appearance Photobook query interface can be seen in Plate 17.

In Shape Photobook, the stiffness matrix of the two objects is used to compute the amount of energy needed to align the two objects (strain energy). Images are ranked according to this energy.

In Texture Photobook (Plate 18), the similarity between the harmonic and evanescent features of two textures is computed using the euclidean distance, while Mahalanobis distance is used for the indeterministic feature.

3.3.6 FourEyes

AUTHOR

R. Picard [112].

QUERY

Image example, along with keywords.

DATABASE

General databases.

APPLICATION

FourEyes is both a tool for the semi-automatic annotation of images and an image-retrieval system: it assists the user in annotating the database images, developing and combining models to represent the concepts introduced by the user (car, street, sky, sea, like in Plate 19), and allows content-based queries based on the known concepts.

IMAGE REPRESENTATION

The images are decomposed into blocks and a label is assigned to them, as long as the block fits into one of the known models. FourEyes performs automatic labeling based on models deduced from interaction with the user, who labels manually a few images and can correct the automatic labeling. This action provokes a change in the models associated to the concepts, so that the error will not be repeated. A single label (or concept) can be associated to a multitude of different models (think of how many different models are needed to represent the concept of “building”). Models are based on texture features, according to the Wold decomposition, like in Photobook [110].

QUERY REPRESENTATION

A query is constructed by selecting areas of the query image as positive or negative examples of the kind of information the user is looking for. Query images contain the same set of features as the database images.

QUERY RESOLUTION

Similarity between images is based on the number of regions in the database image that have the same label as those marked as positive examples in the query image. The location of the labeled regions is not taken into account. The used structure is a histogram of models, and similarity is computed with a histogram distance.

3.3.7 Blobworld

AUTHORS

C. Carson, M. Thomas, S. Belongie, J. Hellerstein, J. Malik [17].

QUERY

Queries are composed of regions, selected from an example image contained in the database (Plate 20). Regions (called blobs by the authors) must be chosen from a segmented version of the image, and a maximum of two regions can be selected, one being the main subject of the query, while the second represents the background. For each region, an overall importance level can be defined (very or some-

what important). The importance of color, texture, position and shape of the selected blob can also be defined (not important, somewhat important, very important).

DATABASE

The used database is composed of 35000 color images extracted from the Corel collection.

APPLICATION

Image retrieval based on coherent image regions roughly corresponding to objects.

IMAGE REPRESENTATION

Images are split into homogeneous regions using a segmentation method based on the Expectation-Maximization algorithm. This segmentation algorithm, described in [18], uses position, color in the L*a*b* color space and texture information (contrast, anisotropy and polarity) for each pixel.

Region features are color (a 218-bin histogram in the L*a*b* space), mean texture contrast and anisotropy. In a previous version of BlobWorld [19], geometric descriptors were also mentioned, and it is not clear whether these are still used or not. These descriptors were region centroid and approximations of area, eccentricity and orientation extracted from the region's scatter matrix.

QUERY REPRESENTATION

Query images are processed like the database images, hence their regions contain exactly the same kinds of features. In a query, the user can select two regions, one of which is considered as the background. The user can select the importance weights of the selected regions, as well as weights for the color, texture, position and shape features.

QUERY RESOLUTION

Candidate images are selected thanks to indexing of the database over the blobs contained in the database images using R*-trees. The authors do not say much about the comparison between regions: all the reader is told is that the comparison between the color of two regions is done using the quadratic distance between the histograms of the regions. In [19], the similarity between two regions was computed as $e^{-(d/2)}$, where d is the Mahalanobis distance between the feature vectors of the two regions, composed of the color, texture and geometric features.

3.3.8 Surfimage

AUTHORS

J. Malki, N. Boujemaa, C. Nastar, A. Winter [93].

QUERY

The user selects an image, along with the kinds of features and the similarity function that should be used to solve the query. Region queries are done by selecting predefined blocks in the image and indicating the spatial constraints between them. Once the system answers, the user can further specify the query by selecting relevant and irrelevant images (Plate 21).

DATABASE

Several databases were used in the experiments, including a database composed of 3670 heterogeneous images.

APPLICATION

Image retrieval by global similarity of (parts of) images in an heterogeneous database of color images.

IMAGE REPRESENTATION

Several kinds of features are computed, among which the color histogram, the histogram of the edges' directions, a wavelet decomposition, eigenimages (as in [35]) and "flexible images" [60][120]. The

images are partitioned using a mutiresolution quadtree approach, and the obtained partitions are used to compose and solve region queries. Features are computed for every extracted subimage.

QUERY REPRESENTATION

A query is either a complete image or a collection of regions at a given resolution level, along with spatial constraints between them. Furthermore, the user selects which features are to be used, the similarity function for each individual feature and the way the similarity results should be combined.

QUERY RESOLUTION

If the query is composed of a complete image, individual similarities are computed for the features the user indicated as relevant, using the similarity function chosen by the user. The combination of these values into a single value is performed as specified by the user.

If the query is composed of subimages, the same process is applied to the single regions. Depending on the spatial constraints defined by the user, the results are combined to obtain a dissimilarity score.

Relevance feedback is based on the estimation of the distribution of the relevant images' features and tries to minimize the chances of retrieving images marked as non-relevant. This process is performed independently for every feature [23][16].

3.3.9 A Relevance Feedback Mechanism for Image Retrieval

AUTHORS

G. Ciocca, R. Schettini [25].

QUERY

Example image. The user can then refine the search results by selecting images among the retrieved set and labelling them as relevant or not relevant (Plate 22).

DATABASE

Experiments were made on a database composed of 5000 color and grayscale images, containing photographs of landscapes, antique textiles, paintings and ceramics.

APPLICATION

Search by global similarity between images in a heterogeneous image database.

IMAGE REPRESENTATION

A large number of features is computed from the image, as well as from five subimages, whose position and size are not explicitly listed by the authors:

- a Color Coherence Vector in the CIELAB color space quantized to 64 colors [108];
- a histogram of the transition between colors (CIELAB color space quantized to 11 colors) [47];
- moments of inertia of the distribution of colors in the unquantized CIELAB color space [134];
- a histogram of contour directions [71];
- the mean and variance of the absolute values of the coefficients of the subimages of the first three levels of the multi-resolution wavelet transform of the luminance image [24];
- the neighborhood Gray-Tone Difference Matrix, i.e. coarseness, contrast, busyness, complexity and strength [2];
- the spatial composition of the color regions identified by the process of quantization to 11 colors [24].

QUERY RESOLUTION

Similarity between images is computed as a weighted sum of the distances between the single features (using the L_1 distance), using a Gaussian normalization to bound the single distances between 0 and 1.

During the relevance feedback phase, the weights of the individual distances are modified by a statistical analysis of the distance feature values of the relevant set and of the non-relevant one. The relevant set is also used to redefine the query, so that it better represents the images of interest to the user.

3.3.10 Circus

AUTHORS

M. Do, S. Ayer, M. Vetterli [38].

QUERY

The system is queried by presenting it with an example image, although the authors claim that it "is also ready to be used in systems with hand-drawn sketch query".

DATABASE

Experiments reported in [38] were conducted on a database of 200 grayscale images, built to demonstrate the invariance of the image features to translation, rotation and scaling. A prototype available on the web (http://lcavwww.epfl.ch/~zpecenov/CIRCUS/Demo.html) uses another database, composed of 6100 color images extracted from the Corel database.

APPLICATION

Retrieval by global visual similarity in a heterogeneous image database.

IMAGE REPRESENTATION

In order to represent the images, their wavelet maxima transform (WMF) [94] is computed (Plate 23). 7 moments of the WMF at 4 different scales are used, giving 28 real coefficients that represent the image. This representation is, because of properties of the WMF, translation, rotation and scale invariant.

QUERY RESOLUTION

Images are compared using the variance weighted Eucliedan distance [43], with the weighting factors being the inverse variances for each vector component, computed over all the database images. Results are then presented to the user like in Plate 24.

3.3.11 Viper

AUTHORS

D. Squire, W. Müller, H. Müller [129].

QUERY

Queries are composed of a set of relevant images and another set composed of non-relevant images. The user can refine a query by selecting images in the query output as relevant or not relevant.

DATABASE

Several databases were used by the authors, including:

- 500 color images provided by swiss television TSR;
- the VisTex database (1610 color images of texture);
- the database used in the Circus [38] system, composed of 6100 images extracted from the Corel collection;

- a database composed of 506 images extracted from the website www.Paris-Roller.com;
- the NASA Xchange database, containing both color and grayscale images.

APPLICATION

Retrieval by global similarity in a heterogeneous image database.

IMAGE REPRESENTATION

The image representation is borrowed from text retrieval: each image is represented by a set of more than 80'000 binary features (terms), divided into color and texture features. These features simulate stimuli on the retina and early visual cortex. Color features are obtained by quantizing the HSV space to 18 hues, 3 saturations, 3 values and 4 gray levels. A histogram is computed for the whole image, as well as for recursively divided blocks (each block contains 4 subblocks). Texture features are computed using Gabor filters at 3 scales and 4 orientations.

QUERY REPRESENTATION

A query is composed of a set of relevant images and a set of non-relevant images. The weight of the individual image features is automatically adjusted depending on the relevance of the query images, the term frequency of the single features in the query and the collection frequency of the features.

QUERY RESOLUTION

First of all, a subset of the database is selected according to the features contained in the query image, using an inverted index file. Then, based on the weights mentioned above, a score is computed for every image in the subset and the sorted subset is returned.

3.3.12 Synapse

AUTHORS

R. Manmatha, S. Ravela, Y. Chitti [95].

QUERY

Two querying methods were developed by the authors: global similarity, which takes an example image as the query, and local similarity, in which the query is formed by selecting regions of an example image.

DATABASE

Two databases were used, both composed of grayscale images. The first is composed of 1561 images retrieved from the Internet or from the Corel collection; the second consists of 2048 grayscale images computed from binary images obtained from the US Patent and Trademark Office; the size of these images was reduced for the experiments.

APPLICATION

Search by local or global similarity between grayscale images, trademark protection.

IMAGE REPRESENTATION

The image representation differs, according to the kind of search to be performed. If the search is a local one (i.e., the query is composed of regions of a selected image), images are represented by a set of vectors containing the first five partial derivatives of the Gaussian at three different scales at uniformly sampled points (typically, every 3 or 5 pixels).

Global search uses a different image representation, based on the histogram of local curvature and phase.

QUERY REPRESENTATION

For local search, the query image is processed like the database images, but only the vectors originated within the selected areas are computed and stored. In global search, the query image and the database images have the same representation.

QUERY RESOLUTION

Retrieval in local search is split in two phases. In the first one, all images whose representation contains vectors similar to those appearing in the query are selected. In the second phase, spatial constraints existing between the query regions are enforced, and the images which satisfy them are returned. All images are considered equally relevant, giving a retrieval result like the one in Figure 7.

FIGURE 7. Local search results in Synapse.

When retrieving images by global search, on the other hand, the image and query feature vectors are compared using normalized cross-covariance, and the database images are ranked according to the obtained score. An example of search by global similarity is to be seen in Figure 8.

FIGURE 8. Query by example results in Synapse.

3.3.13 PISARO

AUTHORS

M. Seaborn, L. Hepplewhite, J. Stonham [125].

QUERY

Queries can be composed of whole images or of parts (tiles) of different database images, arranged on a 3x3 grid. The tiles can be overlaid, and the user can specify which kind of similarity to use for each one of them. The authors call queries of this kind “stackable mosaics”.

DATABASE

The databases used in the experiments are the SoftKey's key photos, formed of 2098 color images, and a collection of 1008 grayscale images composed of samples from the Brodatz database.

APPLICATION

Search by (partial) image similarity in a heterogeneous database or in a database containing pictures of texture.

IMAGE REPRESENTATION

PISARO was developed as a system designed for combining, evaluating and comparing CBIR techniques. Hence, multiple features described in the literature are used. These can be classified between color and texture features. Color features include RGB and HSV histograms, as well as Stricker and Orengo's moments of the HSV color space [133]. The authors also introduce the perceptual category histogram, composed of 11 bins, according to the color classification given by Berlin and Kay [11]. The result of a search performed using this feature can be seen in Plate 25. Ten texture feature extraction methods were implemented, including Gabor filter energy and Liu spectral features [87].

QUERY REPRESENTATION

The query can be composed of a single database image or of parts of images (called tiles), arranged on a 3x3 grid, like in Plate 26. Several tiles can be put on the same point of the grid. Since the images used to form the query are part of the database, their features are already known.

QUERY RESOLUTION

The user selects how similarity must be computed for each tile. Combination of the results is done by simple logical combinations (mathematical interpretations of the logical and, or, nor, nand functions). Tile significance and weighting, as well as position, rotation and scale invariance are also supported. The result is a similarity rating between 0 and 1.

3.3.14 RECI

AUTHORS

M. Bouet, C. Djeraba [13].

QUERY

In this system, querying relies on visual features, extracted from an example image, as well as on textual features. The degree of importance of these features can be chosen by the user.

DATABASE

A database containing pictures of art objects, all shot against a black background, was used. The size of the database is unknown.

APPLICATION

Retrieval by visual similarity and by concept in a database organized into categories.

IMAGE REPRESENTATION

An image is represented as a set of regions, which are characterized by their shape, their texture, their color and the spatial constraints between them. These features are extracted semi-automatically. Similar features are then clustered together under a same symbolic name (for example, color histograms are grouped into categories like "blue", "brown", "yellow" and so on). Images also have textual content associated to them.

The database is organized into categories. The assignment of an image to a particular category is done manually. Furthermore, the system tries to deduce rules from the image features. Rules (as "if the object is gold yellow, then the words 'primitive art' appear in the textual description") have a conditional probability and an implication intensity score attached. Only strong rules (the ones with high

conditional probability and implication intensity) are kept. Using rules, new concepts can be defined. The user is also allowed to define concepts, but he/she is not allowed to define new rules, as those are automatically extracted by the system.

QUERY RESOLUTION

Visual queries are solved by comparing the example image to the images of the database that belong to concepts whose rules are respected by the query image. Comparison between regions is not explained.

If the query also contains textual information, it is used to select the concepts the returned images should belong to.

3.3.15 El Niño

AUTHORS

S. Santini, R. Jain [121].

QUERY

Initially, the user is presented a 2D view of some of the database images, placed so that similar images are close (Plate 27). The query is composed by selecting and moving images on the working surface to indicate to the system how similar the selected images are to be considered. The system reacts to this action by trying to satisfy the user's request, leading to a new distribution of the images in the database space (Plate 28). The process can be repeated iteratively.

DATABASE

Although it is not mentioned explicitly by the authors, it can be assumed that the database used for the experiments is composed of several thousands heterogeneous color images.

APPLICATION

General-purpose image search by example.

IMAGE REPRESENTATION

Database images have three representations, which are used by the three search engines that collaborate within the system: the special features engine, the image decomposition engine and the textual engine.

The special features engine uses features such as color, structure and texture. The image decomposition engine is based on a feature vector, which is general enough to allow the expression of any similarity criterion. To obtain this result, the feature space is derived from a multiresolution decomposition of the image generated by a discrete subgroup of a suitable transformation group. Finally, the text engine works with labels that have been associated to the images, either automatically or manually.

QUERY REPRESENTATION

In El Niño, a query is represented as a graph containing two kinds of nodes: operators and engines. Engine nodes specify queries (comparisons between two images) that must be answered by one of the image engines, while operators describe how the results of the queries must be combined. Engines return a similarity value between 0 and 1, while operators are functions $[0, 1]^n \rightarrow [0, 1]$.

QUERY RESOLUTION

El Niño allows two interrogation modalities:

- retrieving the k images closest to a point in feature space;
- computing the distance between two images.

Computation of similarity depends on the chosen image engine. Results are combined using a similarity algebra developed by the authors.

When configuration feedback (moving images on the feature space) is used, the system asks the search engines to modify their similarity measures so to match the configuration as it was modified by the user. This is done with an optimization process. The newly found similarity measure is then used to recompute the distances in the database, and a new configuration is returned to the user, who can modify it and resubmit it to the system.

3.3.16 Filter Image Browsing

AUTHORS

J. Vendrig, M. Worring, A. Smeulders [141].

QUERY

Querying is organized in Filter Image Browsing (FIB) sessions. At the beginning of the session, the user is presented an overview of the whole database (Plate 29) and he/she selects the most similar image to what he/she is looking for. The system reacts by removing from the current set the images that are most dissimilar to the selected image, and presents a new selection of the current set. The user selects another image, thus zooming in towards the wanted image.

DATABASE

A collection of 10000 images retrieved from the Internet was used in the experiments. All images are GIF files.

APPLICATION

Image retrieval by similarity to an example image in a heterogeneous image database.

IMAGE REPRESENTATION

An image is represented by a set of 9 features, all with value between 0 and 1. In order to compute these features, a 12-bin hue histogram and a 3-bin grayscale histogram are computed. The features are:

- grayness: share of gray pixels in the image;
- number of colors: hue bins containing more than a given amount of pixels;
- average hue;
- color variation;
- background: share of supposed background pixels;
- average saturation;
- average grey value;
- number of regions: number of 8-connected sets of pixels of the same color;
- number of holes: number of regions entirely surrounded by another region.

QUERY REPRESENTATION

Since query images are in the database, their features are the same as the ones of the database images. Images chosen during the previous steps have an impact on the computation of the similarity, although the chosen method is not clear.

QUERY RESOLUTION

At each iteration, the set of relevant images is reduced by a factor ρ . Only the images most similar to the ones selected by the user are kept

Similarity is computed as follows: first all similarities between database images are computed for every feature. Then, a histogram of the results is built. When two images are compared during the retrieval process, the position of the feature dissimilarity in the histogram is used instead of the actual dissimilarity, so that problems with different dissimilarity functions having different distributions are solved. The dissimilarity between two images is the weighted sum of the dissimilarities computed over the nine features. The authors do not tell how similarity between a set of query images and a database image is computed.

3.3.17 WebSEEK

AUTHORS

J. Smith, S. Chang [127].

QUERY

Different kinds of queries are possible. At the beginning of a search session, the user can look for files by:

- selecting a subject from a "Yahoo-like" taxonomy (Figure 9) defined by the authors;
- typing terms related to the sought files;
- specifying the kind of files sought (video, color image, color graphic, grayscale image, black/white image);
- specifying from which national Internet domain the files were retrieved (".com", ".it", ".jp", etc.);
- specifying text to be found in the files' URL.

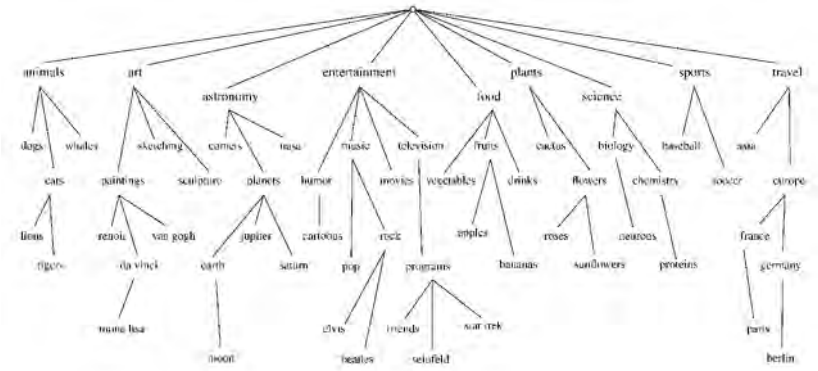

FIGURE 9. Part of the subject taxonomy defined by the authors.

When WebSEEK returns the list of images that match the query, the user can:

- mark some images as relevant, some as non-relevant, and perform a new search (on the whole database or only on the retrieved images), which will be based on visual similarity (Plate 30);
- combine the query results with the results of a new query, using set operations (union, intersection, subtraction);
- produce a color histogram, or modify a color histogram extracted from an image or a set of images (Figure 10), and perform a new query based on color similarity.

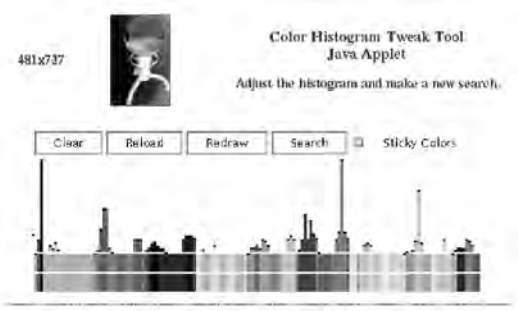

FIGURE 10. The interface with which the users of WebSEEK can create or modify color histograms.

DATABASE

Images and videos retrieved from the World-Wide Web. The database contains 513323 files, of which 1.05% are videos.

APPLICATION

Image search engine for the World-Wide Web.

IMAGE REPRESENTATION

Each file in the database is represented by the following information tables:

IMAGES, containing the fields

- IMID: a unique identifier for the file;
- URL: the URL of the page from which the image was retrieved;
- NAME: the name of the file;
- FORMAT: the file format;
- WIDTH;
- HEIGHT;
- FRAME: the number of video frames;
- DATE;
- TEXT: text extracted form the URL, or from HREF or ALT tags related to the file.

TYPE, containing the fields:

- IMID: a unique identifier for the file;
- TYPE: a value among {Color photo, Color graphic, Video, B/w image, Gray image}.

SUBJECTS, containing the fields:

- IMID: a unique identifier for the file;
- SUBJECT: a subject class from a taxonomy defined by the authors. Membership of an image to a subject class is determined automatically using a key-term dictionary, containing the set of key terms and their mapping to the subject classes. The key-term dictionary is obtained semi-automatically.

TEXT, containing the fields:

- IMID: a unique identifier for the file;

- TERM: terms are extracted from the file's URL, as well as from HREF or ALT tags by chopping the text at non-alpha characters.

FV, containing the fields:

- IMID: a unique identifier for the file;
- COLOR-HISTOGRAM: a 166-bin color histogram in the HSV color space, obtained by reducing the color space to 18 hues, 3 saturation values, 3 luminosity values and adding 4 levels of gray.

QUERY REPRESENTATION

The query is represented using the same information tables used to represent the images.

QUERY RESOLUTION

Queries on the database tables IMAGES, TYPES, SUBJECTS and TEXT are performed using standard relational algebra. For example, finding all videos belonging to the "news" subject and whose description contains the "basketball" term is performed as follows:

```
SELECT IMID
FROM TYPES, SUBJECTS, TEXT
WHERE TYPE="video" AND SUBJECT="news" AND TERM="basketball"
```

Queries defined by selecting a set of relevant images I_R and a set of non-relevant images I_N are based on color histograms (called "visual information" by the authors). By selecting I_R and I_N, the user implicitly produces a color histogram h_q^{k+1}:

$$h_q^{k+1} = \left\| \alpha \cdot h_q^k + \beta \cdot \sum_{i \in I_R} h_i - \gamma \cdot \sum_{j \in I_R} h_j \right\|$$

where h_q^k is the histogram of the previous visual query (if any), and $\| \ \|$ means normalization.

Queries based on color histograms are solved by computing the distance between the query histogram and the histograms of the database images. The used distance is an approximation of the quadratic distance, applied on thresholded query histograms (bins containing less than τ elements are ignored).

3.3.18 ImageRover

AUTHORS

S. Sclaroff, L. Taycher, M. La Cascia, S. Sethi [124][80].

QUERY

At the beginning of a search session, the user types a set of keywords connected to the images he/she is looking for. In the further stages, the user adds/removes images from a relevant images set (Plate 32).

DATABASE

Experiments were conducted on a database containing 100000 images retrieved by robots from the World Wide Web. All images are larger than 64x64 pixels.

APPLICATION

Image search engine for the World Wide Web based on textual and visual statistics.

IMAGE REPRESENTATION

Each image is represented by a vector, which composed of two subvectors: a Latent Semantic Indexing (LSI) vector, and a visual features vector. The visual feature vector is, in turn, composed of

dimensionally reduced color histogram and orientation direction vectors, computed for 6 overlapping regions (whole image, central region, upper left, lower left, upper right, lower right).

The LSI vector of an image is obtained by scanning the HTML document that contains it: a weighted word frequency histogram is computed; words close to the image, as well as words appearing within some HTML tags are given special importance. An unweighted term frequency is also computed over each document in the database. Using this data and the log-entropy scheme, a term term × image matrix $A = (a_{ij})$ is created, where a_{ij} is expressed in terms of the local weight $L(i, j)$ of term i for image j and of the global weight $G(i)$ of term i. Then, a singular value decomposition is performed, $A = U\Sigma V^T$, decomposing the original term-image relationships into a set of linearly independent vectors. The dimension of the problem can be reduced by choosing k most significant dimensions from the factor space which are then used for estimating the original index vectors.

Color histograms are computed in the L*u*v* color space, and contain 64 bins (4 for each axis). Orientation descriptors are obtained by using 4-level steerable pyramids, which generate dominant orientation and orientation strength values for each pixel and each level of the pyramid. Orientation are then stored in 4 (one for each pyramid level) 16-bin orientation histogram. Only pixels in which the orientation strength is above a given threshold contribute to the generation of the histogram.

If it was stored “as is”, the visual feature vector would have dimension 768. In order to reduce its dimension, for each subvector space a principal component analysis is performed, resulting in a dimension reduction of over 85%.

QUERY REPRESENTATION

A query is composed of a set of images, and is represented by the set of the image vectors it contains.

QUERY RESOLUTION

In the first phase, the set of keywords entered by the user is considered as a text document. An LSI index is computed and used to match nearest neighbors in the subspace of all LSI vectors in the database (Plate 31).

ImageRover uses an algorithm which, being given the set of relevant images S, computes on the fly the appropriate normalized $\tilde{L}_m$ Minkosky distances to be used for each subvector in the image descriptors.Once the $\tilde{L}_m$ distances have been determined, a k-nearest neighbor of the image index is performed. Let X, Y be two images, and let x_i, y_i be their i-th subvectors; the distance between X, Y is computed using the following distance metric:

$$\delta(X, Y) = (\omega_1, \ldots, \omega_n) \cdot \begin{pmatrix} \tilde{L}_{m_1}(x_1, y_1) \\ \ldots \\ \tilde{L}_{m_n}(x_n, y_n) \end{pmatrix}$$

where:

- $m_i = \arg\left(\min_m \eta_m^{(i)}\right)$;
- $\eta_m^{(i)} = E[\tilde{L}_m(p_i, q_i)]$, $P, Q \in S$;
- the $\omega_i = 1/(\varepsilon + \eta_m^{(i)})$ are the relevance weights (ε is an approximation factor that can be set by the user).

3.3.19 Flower Patent Retrieval

AUTHORS

M. Das, R. Manmatha, E. Riseman [32].

QUERY

The user can query the system in two ways:

- by selecting color names;
- by choosing an example image (Plate 33).

DATABASE

The database consists of 300 pictures of flowers. Of these, 100 were actual flower patents from the US Patent and Trademarks Office, 100 were taken from CD-ROM collections because of the complexity of their backgrounds, while the rest was scanned from catalogs of flowering plants and photographs, including a few images of colored fruits.

APPLICATION

Image retrieval based on color in a database containing only flower patent images.

IMAGE REPRESENTATION

An image is represented by the names of the colors of the pictured flower, along with the number of pixels of the contained colors. Image colors are extracted after a segmentation process that eliminates the background pixels, leaving only the ones belonging to the flowers (Plate 34). This segmentation process uses knowledge about the color of flowers and about the spatial placement of the flowers in the database images.

The possible color names were obtained by quantizing the HSV color space to 64x10x16 bins, then associating the average colors of the bins to their X Windows and ISCC-NBS names. Since in the ISCC-NBS system , color names are specified by color classes and hue modifiers (“pale blue” being composed of color class name “blue” and hue modifier “pale”), the ISCC-NBS names are used in order to obtain a color hierarchy.

The segmentation process proceeds as follows:

- all pixels belonging to ISCC-NBS color classes “black”, “brown”, “gray” and “green” are removed from the image;
- connected components formed by the surviving pixels are detected, independently on changes in color, and their validity is checked (a connected component is valid if its size is above a given threshold and if its position is towards the center of the image);
- the largest connected component is chosen;
- the colors on the borders of the connected component are analyzed to detect background colors, which are removed from the whole image;
- another scan for valid connected components is performed, and the largest is kept;
- if no valid connected components are found, the eliminated colors are reinstated one by one, and the search for valid connected components is repeated;

If the whole segmentation process does not result in valid connected components (for example if the flower is a different shade of the background color), the segmentation is repeated by using color names instead of whole color classes.

QUERY REPRESENTATION

The query is represented by the names of the chosen colors (if the user is querying the system by selecting a color from the palette), or by the names of the colors contained in the flower pictured in the example image.

QUERY RESOLUTION

If the user types some color names, a search for these color names is performed in the X Windows and ISCC-NBS color names lists. Images containing all the specified colors are returned, with the ones containing more pixels of the desired colors first.

In the query by example case, the aforementioned technique leads to bad retrieval results, because different shades of the same color class are not considered similar. In order to avoid these problems, the retrieval system ranks each image on the basis of the city-block distance of its average color in each color class from the corresponding query color averages. The average color of a color class is obtained by computing a weighted average of the HSV values of all the colors belonging to the color class. The weights are proportional to the relative proportion of the color in the image.

3.3.20 Other Systems

Among the CBIR systems mentioned elsewhere in this chapter, there are some that are able to solve queries composed of example images: PICTION (Section 3.1.2), JACOB (Section 3.2.1), QBIC (Section 3.4.6), Fast Multiresolution Image Querying (Section 3.4.5) and the system developed by Cha and Chung (Section 3.4.7).

3.4 Query by User-Produced Pictorial Example

3.4.1 Leiden ImageSearch

AUTHORS

M. Lew, K. Lempinen, N. Huijsmans [84].

QUERY

Two choices are possible: drawing a line-art sketch and placing icons corresponding to objects (human faces, sand, trees, water and sky, as seen in Plate 5).

DATABASE

Images retrieved from the Internet. The database contains about 100000 images.

APPLICATION

Search according to visual concepts, or to image edges, in a heterogeneous image database.

IMAGE REPRESENTATION

The location of objects (corresponding to the icons the user can place in the query image) is detected with a method described in [63] using color, gradient, Laplacian and texture info for every pixel in the image at multiple scales. Edges are detected using the Sobel operator [14] in conjunction with a Gaussian blurring filter. Trigrams [62] are then used to store the distribution of edges in the database image.

QUERY REPRESENTATION

If the query was an icon-based one, the location and type of the icons is stored. For line-art sketches, on the other hand, trigrams representing the edges are computed and stored.

QUERY RESOLUTION

For line-art sketches, in addition to the comparison of the trigram representation used in [62], weighting by spatial placement of the objects by resorting using Kulback templates [82] is performed.

Computation of similarity for queries containing icons is not explained.

3.4.2 SCORE

AUTHORS

Y. Aslandogan, C. Thier, C. Yu, J. Zou, N. Rishe [4].

QUERY

Queries are composed by placing icons on a canvas. Each icon denotes an object and has a set of attributes whose values (exact or fuzzy) can be defined by the user. Two kinds of relationships between objects are possible: action and spatial. The user can also ask for a given object not to appear in the returned images (negation).

DATABASE

A collection of 250 color images coming from a variety of domains was used.

APPLICATION

Visual search in content-annotated databases.

IMAGE REPRESENTATION

Images are represented as UNISQL/X statements. The metadata describing the images is obtained by using the same interface used for querying, i.e., by placing icons corresponding to objects on a canvas and defining their attributes and the relationships between them.

QUERY REPRESENTATION

Queries are represented exactly like the database images.

QUERY RESOLUTION

Solving a query in SCORE consists in matching the query description against the database images descriptions. The similarity of a query term t_q (entity name, relationship name, attribute value) to a database term t_{db} is given by:

$$sim(t_q, t_{db}) = idf(t_q)/dist(t_q, t_{db})$$

where $idf(t_q)$ is the inverse document frequency of t_q and $dist(t_q, t_{db})$ is 0 if the terms are equal, 1 if they are synonyms, and is a larger integer value if they are more loosely related (like the terms “skyscraper” and “house”, for example). How two terms are related is computed using WordNet, an electronic lexical system [100]. The individual term similarities are combined using a normalization approach, resulting in a similarity value between 0 and 1.

3.4.3 ETM (Elastic Template Matching)

AUTHORS

A. Del Bimbo, P. Pala [34].

QUERY

The query is a line-art sketch containing one or more closed shapes (Figure 11).

DATABASE

The experiments described in [34] were conducted on a database composed of 100 images of paintings. Other examples provided show, though, that the method could be used with any kind of images.

APPLICATION

Retrieval of images containing given shapes in a given spatial configuration. Applicable to any database containing easily recognizable shapes (paintings, medical images, etc.).

IMAGE REPRESENTATION

Images are processed with the Canny edge detector and interest areas in the image are detected (although the authors do not explicitly explain how), and edge images of those are stored, along with their aspect ratio and the spatial relationships between them.

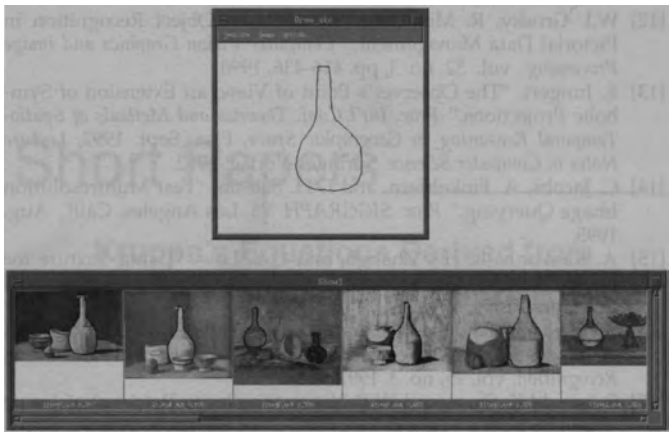

FIGURE 11. A query in ETM and the 6 images containing the most similar shapes.

QUERY REPRESENTATION

The shapes in the sketch are represented as a polygonal and instantiated as a linear combination of B-splines functions. The aspect ratio of their minimum enclosing rectangles, as well as the spatial relationships between the drawn contours, are also stored.

QUERY RESOLUTION

First of all, a selection of the database images takes place: to be matched against a sketch contour, an interest area must have an aspect ratio similar to the one of the sketch contour. Furthermore, if more than one shape is present in the sketch, the spatial relationships must be preserved. This way, a large part of the database images can be removed before computing the similarities between the contours and the interest areas.

The shapes in the sketch are strained and bent to match the edges in the interest areas, and an elastic deformation energy E (sum of the strain energy S and of the bend energy B) is computed. The matching score M depends on how close the deformed contour points are to the edges of the interest area. The goal of the method is to maximize M while minimizing E. This is obtained with a gradient descent technique.

The similarity between the contour and the edges in the interest area also depends on the complexity N of the contour (the number of zeros of its curvature function), as well as on the correlation C between the curvature function of the original template and the curvature function of the deformed one. The five parameters (M, S, B, N, C) are classified by a back-propagation neural network subject to appropriate training, and a similarity value between 0 and 1 is returned. If the sketch contains more than one contour, the total similarity is the sum of the similarities for the individual shapes.

3.4.4 PICASSO

AUTHORS

J.M. Corridoni, A. Del Bimbo, P. Pala [28].

QUERY

Freehand color sketch produced with a paint tool. The user draws region contours that can be filled with color (Plate 35). Inter-region properties, like harmony or contrast, can be defined.

DATABASE

1000 color images of paintings.

APPLICATION

Search for regions with particular colors or perceptual properties induced by color in paintings.

IMAGE REPRESENTATION

Database images are segmented using an improved version of the standard K-means algorithm [70] in the L*u*v* color space. The obtained cluster centers are then back-projected onto the image and regions are formed.

Region properties are color, hue, luminance, saturation, warmth, size and position. The PICASSO system follows Itten's codification of expressive chromatic usage [66]: hue is represented by 12 reference values, while 5 reference values are introduced for luminance, 3 for saturation and 3 for warmth (warm, cold, neutral). A fuzzy representation value is used to describe the actual value of a property, and is also used for region position (9 values, indicating the percentage of the area of the region falling into one of the 9 identical rectangular areas the image is partitioned into) and region size (with 3 reference values: large, medium, small).

Inter-region properties are also stored, as contrast measures are introduced for hue, saturation, luminosity and warmth.

QUERY RESOLUTION

The matching between the query image and a database image is done by defining picture formulae. These are boolean combinations of region formulae, which characterize chromatic and arrangement properties of color patches. Region formulae are extracted from the query regions and are then matched against the region descriptions stored in the database. The matching is done according to satisfaction rules which return a score representing the degree of truth by which the region is represented by the formula (hence the similarity between the region in the database image and the region in the query).

3.4.5 Fast Multiresolution Image Querying

AUTHORS

C. Jacobs, A. Finkelstein, D. Salesin [69].

QUERY

A freehand color sketch or a low-resolution image (Plate 36).

DATABASE

The system was tested on two databases, the largest of which was composed of 20558 images, mostly GIF files retrieved from the Internet.

APPLICATION

Search by sketch or low-resolution image in a general image database.

IMAGE REPRESENTATION

Database images are scaled to 128x128 pixels, and a 2-D Haar wavelet decomposition is performed for each color channel (in the YIQ color space [64]), generating wavelet coefficients $T[i,j]$. Only the

coefficients with the largest magnitude are kept (truncation) and are then quantized to two levels: +1 and -1, giving the truncated and quantized representation $\tilde{T}[i,j]$.

QUERY REPRESENTATION

A query is processed exactly like a database image, giving the decomposition $Q[i,j]$ and the truncated and quantized decomposition $\tilde{Q}[i,j]$.

QUERY RESOLUTION

The metric used to compute the similarity between the query and a database image is

$$\|Q, T\| = w_0 \cdot |Q[0,0] - T[0,0]| + \sum_{i,j,\tilde{Q}[i,j]\neq 0} w_{bin}(i,j) \cdot (\tilde{Q}[i,j] \neq \tilde{T}[i,j])$$

The weights $w_0, w_{bin}(i,j)$ must be determined by training. In the experiments, for sketch queries 60 wavelet coefficients were kept, while for low-resolution images only 40 coefficients were needed.

This approach allows for a query without much detail to match a detailed image rather well. The opposite, however, is not true. A sample query result can be seen in Plate 37.

3.4.6 QBIC

AUTHORS

R. Barber, B. Beitel, W. Equitz, C. Niblack, D. Petkovic, T. Work, P. Yanker [9].

QUERY

In this commercial system developed by IBM, querying can be done by:

- using predefined elements (called icons or thumbnails), which represent color values, textures, shapes, sizes as well as textual content;
- placing colors in a predefined grid;
- drawing a freehand sketch (Plate 6);
- selecting an example image and choosing polygonal relevant areas in it;
- specifying color percentages (Plate 4).

Within the QBIC system, techniques for video retrieval have also been developed, although they won't be discussed here. See [43] for a description of QBIC's capabilities in this domain.

APPLICATION

Image retrieval by local or global visual similarity in a general database.

DATABASE

QBIC being a commercial product, it is built to work with any kind of image database. An example can be found on the website of the St. Petersburg Hermitage Museum (www.hermitagemuseum.org).

IMAGE REPRESENTATION

For every image in the database, a set of regions of interest (masks) is identified, either manually or semi-automatically. For each mask, characteristics describing visual properties for the mask are computed and stored in the data representation. These characteristic are the color histogram, texture coarseness, contrast and directionality, and shape descriptors (area, circularity, eccentricity, major axis orientation and a set of moment invariants). Keywords are also associated to the database images.

QUERY REPRESENTATION

If the query is composed using thumbnails, the feature values associated to the used thumbnails represent the query. The spatial arrangement of the thumbnails is also taken into account. Queries produced by freehand drawing undergo the same process as the image databases, and the features described

above are extracted; the queries generated by placing colors on a predefined grid only have color and spatial features. The queries composed by selecting areas of an image contain only the features of the selected regions.

QUERY RESOLUTION

The distance between the query and an image is a weighted sum of the similarities computed between the masks contained in the query and those in the image. These distances are in turn a weighted sum of the distances between the masks' features (color, textual information, shape and size). The database is sorted according to the computed distance.

3.4.7 Object-Oriented Retrieval Mechanism for Semistructured Image Collections

AUTHORS

G. Cha, C. Chung [21].

QUERY

4 kinds of query are possible:

- simple: attribute values must be explicitly specified, and exact match is required;
- range: ranges for attribute values are specified;
- similarity: a set of example images and/or user drawings is used;
- complex similarity: a combination of the former three techniques.

DATABASE

A collection of 1064 heterogeneous 256-color images was used in the experiments.

APPLICATION

General-purpose search engine.

IMAGE REPRESENTATION

The concept of image is split into a body (the raster image itself) and a header (the features). The header is organized into visual, semantic and keyword features.

Visual features (colors, textures, shapes and positions) are extracted automatically. The color descriptor is formed by 2 moments of a 32-bin histogram for each color channel and for each quarter of the image, resulting in a color feature vector of size 24. Semantic features (title, subject, type, perspective, orientation, date) are extracted manually. They are organized hierarchically and are indexed using a chi-tree. Keywords are also added manually. Visual features and keywords are indexed using a HG-tree.

QUERY REPRESENTATION

Queries are represented as boolean conjunctions of conditions over features, as in the following example

$$Q = (C_{v_1}(colhist) = I_1) \wedge (C_{v_2}(colhist) = I_2) \wedge (C_{s_1}(subject) = \text{car})$$

in which the user queries the system for an image containing a region with color histogram I_1, another region with color histogram I_2 and the word "car" in the subject descriptor.

QUERY RESOLUTION

Queries over semantic data or keywords are solved by exact matching and return sets of equally relevant images. Queries on visual features, on the other hand, return lists of images sorted according to visual similarity to the query. If the query is a "complex similarity" one, semantic queries are solved first in order to restrain the search space. Visual queries with more than one example image are transformed into queries with one image which is mutually similar to all the submitted query images. If

different visual features are used for different example images, a complex nearest neighbor algorithm is used to combine the results into a single ranking.

3.4.8 QBICAT

AUTHORS

J. Favela, V. Meza [41].

QUERY

A query in QBICAT (Query By Image Content and Associated Text) is composed of an user-produced pictorial example (query image) and a list of keywords (query terms). The user also specifies importance weights for the visual and the textual aspects of the query.

DATABASE

The database used in the experiments was composed by extracting images from the digital library of a collection of plants and animals from the Baja California. The digital library was composed of 200 HTML pages, each for a specific species. The HTML pages contained one or more images.

APPLICATION

Querying by image content and text in an heterogeneous database of images retrieved from the web.

IMAGE REPRESENTATION

An image is represented by a list of 10 keywords extracted from the HTML page it is contained in and by a wavelet-based representation of its visual contents.

The visual representation is the one presented by Jacobs et al. in [69] (also see Section 3.4.5).

The keyword-extraction process is the following: first, candidate words are selected. These are words that appear in the image's CAPTION tag, in paragraphs containing links to the image, in text appearing around the image and in the title of the HTML page (in case no other information is available on the page). Stop words are then removed and a weight is computed for each remaining word, using the Tf*IDf (term frequency - inverse document frequency) algorithm. The words with the highest weights are stored in the image representation, along with their weights.

QUERY REPRESENTATION

The query is represented by its query terms and by the wavelet representation of the query image. The used wavelet representation is the same one used for the database images.

QUERY RESOLUTION

The visual similarity $\|Q, W\|$ between a database image W and the query image Q is computed as in the work by Jacobs et al. [69].

The textual similarity $\langle q, w \rangle$ between the query terms q and the database image terms w, on the other hand, are computed as the sum of the weights of the database image terms that are found in the document.

The relevance R of a database image according to a given query is computed as $R = a \cdot \|Q, W\| + b \cdot \langle q, w \rangle$, where a, b are the weights given by the user to specify the importance of the visual, respectively the textual aspect of the query.

3.5 Summary

In Table 1 we summarize the kinds of queries that are supported by the systems we presented in this chapter.

System name	keywords, nat. lang.	feature values	example image	user-prod. ex. image
Chabot [104]	x	x		
Piction [130]	x		x	
JACOB [79]		x	x	
Colour Image... [96]		x		
NETRA [88]			x	
PICTOSEEK [51]			x	
Leiden 19th... [83]			x	
Virage [72]			x	x
Photobook [110]			x	
FourEyes [112]	x		x	
Blobworld [17]			x	
Surfimage [93]			x	
A Relevance... [25]			x	
Circus [38]			x	x
Viper [129]			x	
Synapse [95]			x	
PISARO [125]			x	
RECI [13]	x		x	
El Niño [121]			x	
Filter Image... [141]			x	
WebSEEK [127]	x	x	x	
ImageRover [124]	x		x	
Flower Patent... [32]	x		x	
Imagesearch [84]				x
SCORE [4]				x
ETM [34]				x
PICASSO [28]				x
Fast Multi... [69]			x	x
QBIC [9]	x	x	x	x
O-O Retrieval... [21]		x	x	x
QBICAT [41]	x			x

TABLE 1. A summary of the kinds of queries supported by the CBIR systems presented in this chapter.

3.6 Conclusion

In this chapter, a large number of CBIR systems has been presented, and their main features have been highlighted. As the reader will have noticed, the number of different applications of CBIR retrieval is extremely large, just like the amount of available techniques.

When choosing (or starting to develop) a CBIR system, several factors should be taken into account, among which we can list the following:

- CBIR systems targeted at general, heterogeneous databases cannot discriminate between very similar images (like MRI scans of the same body part of a person taken at different moments). In order to discriminate between these, a specialized CBIR system will be needed;
- querying techniques should be combined: for example, users should be allowed to paint a sketch and enter a textual description of the target images;
- users should be allowed to switch between querying techniques: after querying by sketch, the user could continue his/her search by using one of the retrieved images to query by example;
- relevance feedback is an extremely powerful tool, but only as long as the users do not start to give feedback about the semantic content of the image if this kind of information is not deductible from the images' representation;
- if non-visual (i.e., textual) information about the images is available, it should be used;
- when the size of the database increases, the internal organization of the database becomes even more important than an effective distance measure between query and database images;
- the query methods should be adapted to the users' profile.

In this work, retrieval by user-produced pictorial example (also called querying by sketch) is the chosen technique. In Chapter 4, several ways of querying an image database by sketch will be presented.

3.7 Color Plates

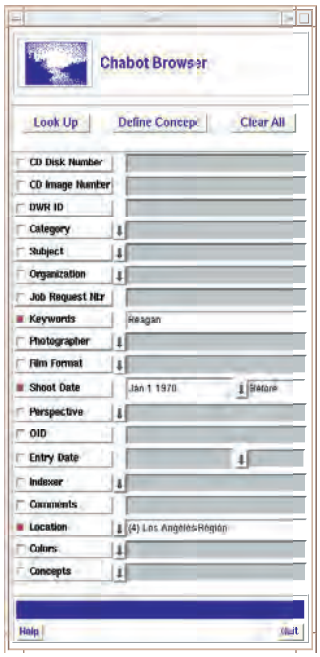

PLATE 7. **The Chabot query interface.**

PLATE 8. **Result obtained with Chabot when asking for images containing some orange taken in the S. Francisco bay area after 1994.**

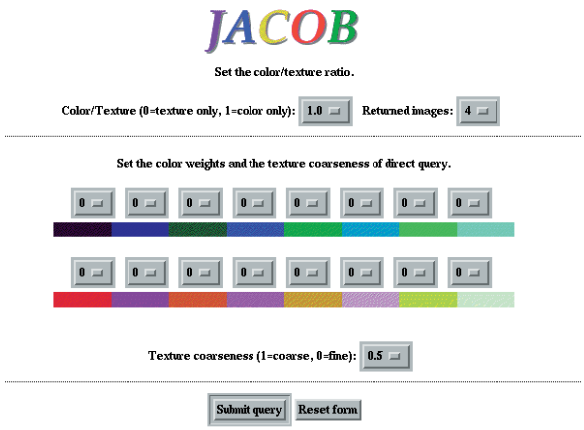

PLATE 9. **JACOB's query by feature values interface.**

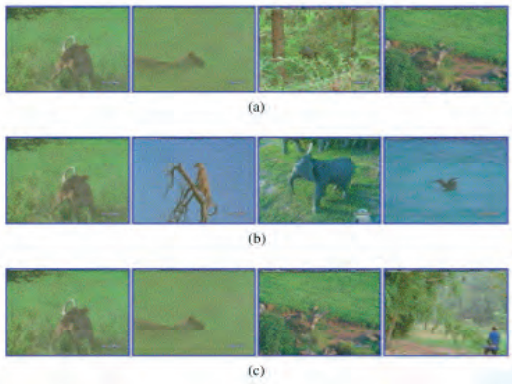

PLATE 10. **Query by example results in JACOB when using only color information (a), only texture information (b), a combination of the two (c). The image on the left is the query image.**

PLATE 11. **Query by feature value results in JACOB. The user asked for: (a) mainly brown images; (b) coarse-textured mainly brown images; (c) fine-textured mainly brown images.**

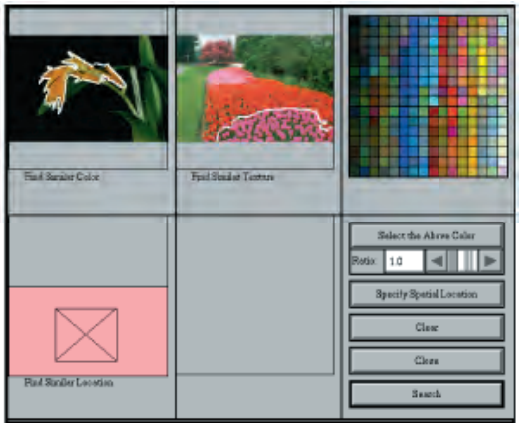

PLATE 12. **The NETRA query interface.**

PLATE 13. Segmentation results in NETRA.

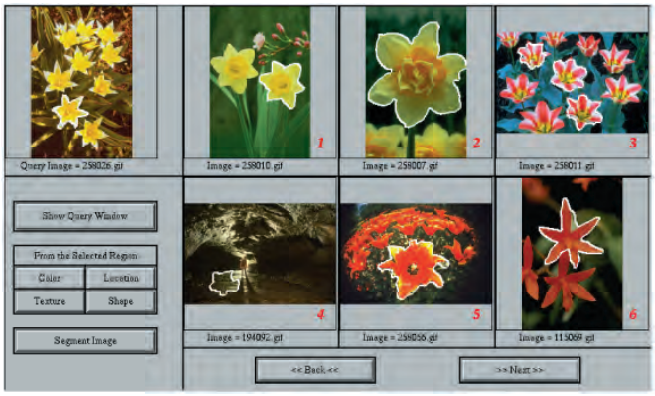

PLATE 14. A NETRA query result based on the shape of the highlighted region in the query image.

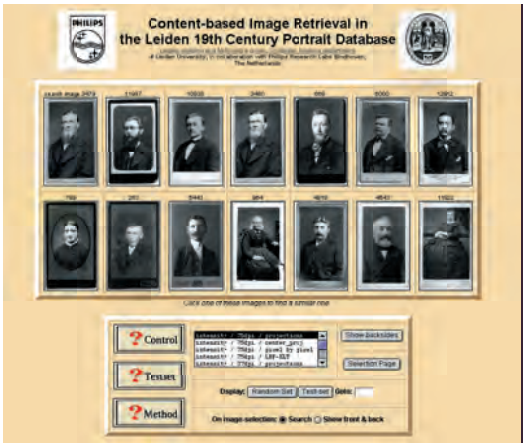

PLATE 15. A query result in the Leiden 19th Century Image Database.

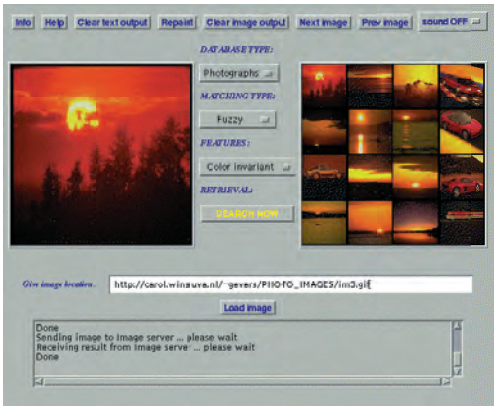

PLATE 16. A query example in PicToSeek.

PLATE 17. **The Appearance Photobook query interface.**

PLATE 18. **A query result in Texture Photobook. The query image is the one in the upper left corner and is followed by the most similar images found in the database, placed in raster scan order.**

PLATE 19. **Automatic labeling in FourEyes: the user labeled some areas in the images on the right as “sky”, and FourEyes automatically labeled all the hashed regions as “sky”.**

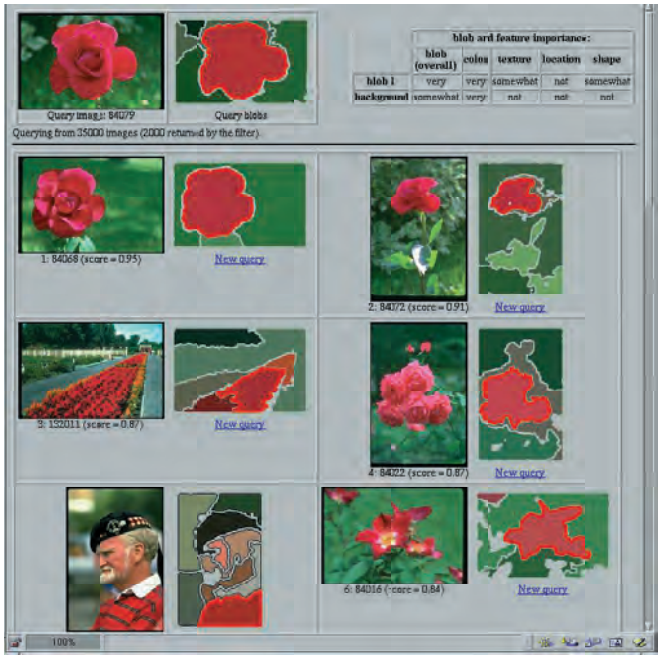

PLATE 20. **Query result in Blobworld. For each retrieved image, its Blobworld representation is shown, with the region corresponding to the sought region highlighted.**

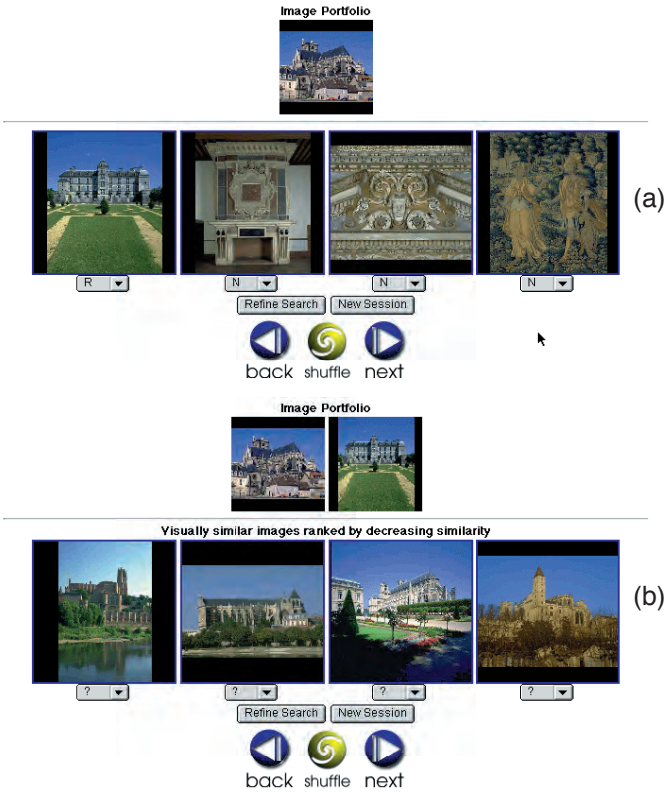

PLATE 21. **Query results in SurfImage. The user can select retrieved images as relevant and non-relevant (a), allowing the system to return images closer to the desired ones (b).**

PLATE 22. **Relevance feedback in the work by Ciocca and Schettini [25]: initial retrieval results (a) can be improved by selecting some of the retrieved images as relevant (b).**

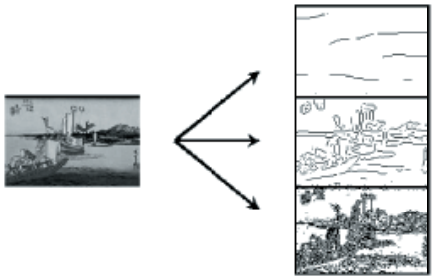

PLATE 23. **Wavelet maxima decomposition in Circus.**

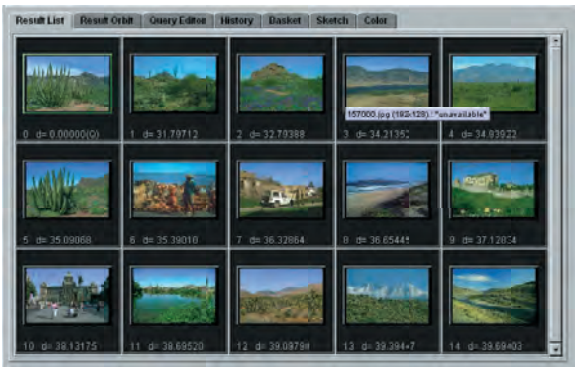

PLATE 24. **A query by example result in Circus.**

PLATE 25. **Retrieval using the perceptual category histogram in PISARO.**

PLATE 26. **A search for landscape scenes (blue at the top and green grass-like texture at the bottom) in PISARO.**

PLATE 27. **The El Niño interface at the beginning of a search process.**

PLATE 28. **The El Niño interface during the search process.**

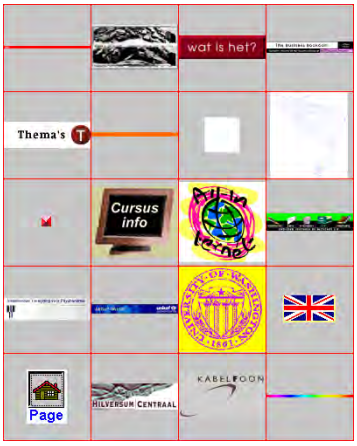

PLATE 29. An example of initial selection of images in Filter Image Browsing.

PLATE 30. In WebSEEK images can be labeled as relevant or non-relevant.

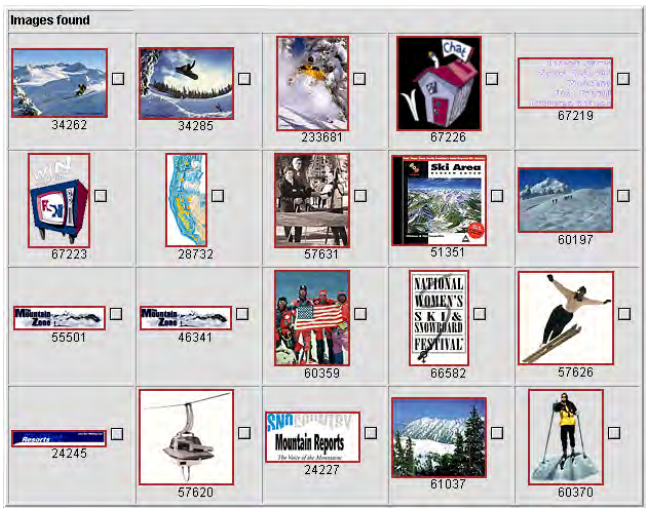

PLATE 31. Query results in ImageRover. In this case, the user queried the system using the words “snow” and “ski”.

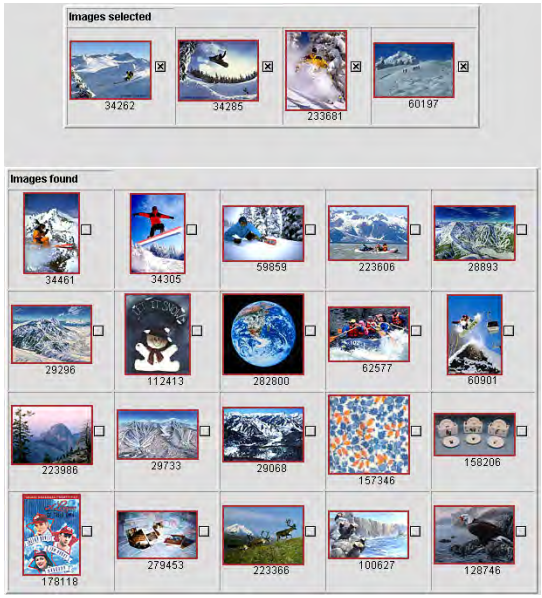

PLATE 32. **In ImageRover query results can be improved by selecting relevant images among the retrieved ones.**

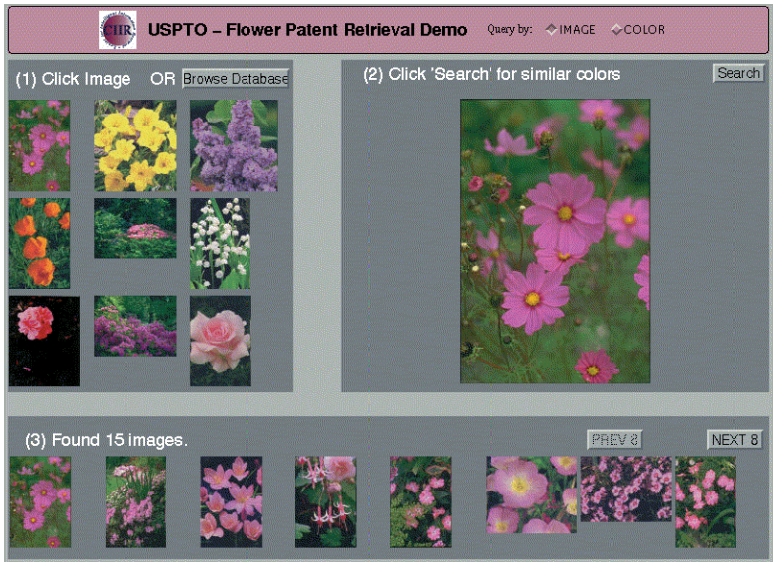

PLATE 33. **Querying by example image in Flower Patent Retrieval.**

PLATE 34. **Background elimination in Flower Patent Retrieval. From left to right: the original image, image after deleting nonflower and background colors, the largest valid connected component found.**

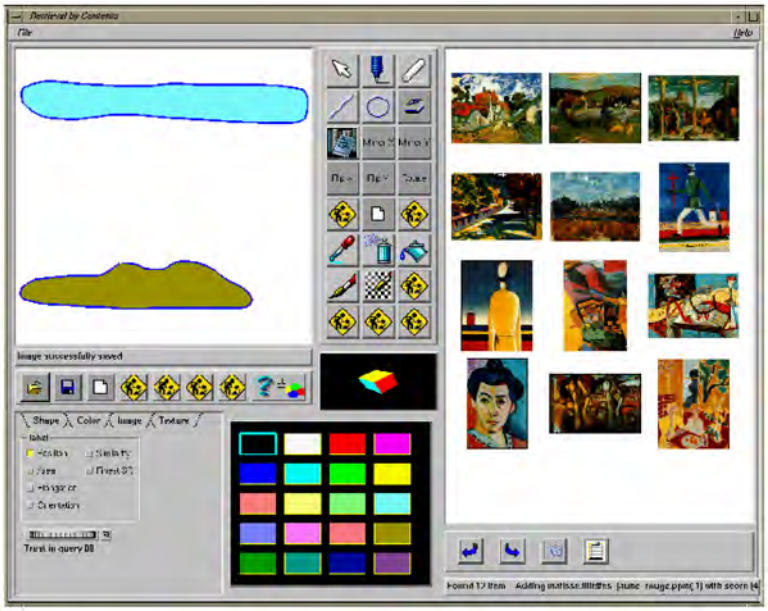

PLATE 35. **A query in PICASSO and the retrieved images.**

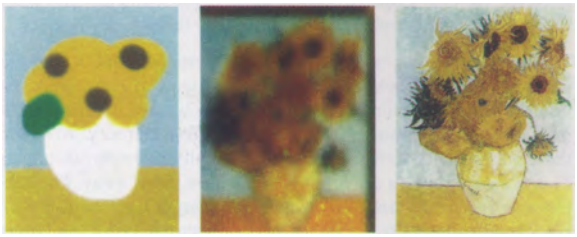

PLATE 36. **In Fast Multiresolution Image Querying, a query can be composed of a sketch (left) or of a low-resolution scan (center) of the target image (right).**

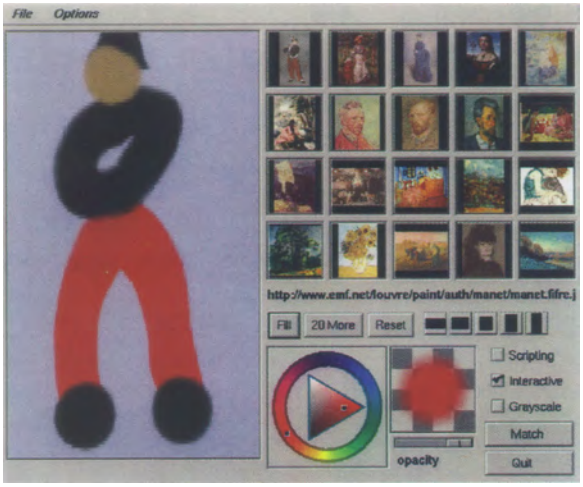

PLATE 37. **A query result in Fast Multiresolution Image Querying.**

CHAPTER 4

Querying by Sketch

When querying by sketch the user builds an image containing elements which will be used to retrieve the sought image(s). This image (the sketch) can contain edge information, color information, texture information, region information, spatial constraints or predefined elements with simple semantic meaning.

Queries produced this way are entirely visual, and are well-adapted to databases in which only primitive features are associated to the images. It also allows to specify clearly the position of the sought objects, as well as spatial constraints between the elements in the sketch. Many current systems use a sketch as query image. Among them we can list QBIC [43], PICASSO [28] and SCORE [4].

In this chapter, the different ways of querying by sketch, as well as extracting information from digital images so that it can be compared to that coming from the sketch will be explored.

4.1 Producing a Sketch

As it was said in Chapter 2, there are many different ways to query a database by sketch, ranging from drawing only the most important edges to trying to paint an image that looks like the sought one.

In the following paragraphs, all the different ways of querying an image database by sketch will be listed, along with their strengths and weaknesses.

4.1.1 Drawing Edges

This techniques consists in drawing the contour of one (or more) of the objects pictured in the image. The retrieval is then performed by trying to find edges in the database images that match the ones in the sketch. This matching can be done in a multitude of ways: using classic pattern matching techniques, comparing the direction of the edges in the image to that of those in the sketch or, like in the ETM system [34], by bending the contour produced by the user to make it match those found in the images and then computing the energy that was necessary to perform the deformation.

Advantages of such a technique:

- extraction of data from the sketch is rather easy;
- production of the sketch is fast and straightforward, even though some kinds of images may be problematic, like the one in Plate 38;

- perfectly adapted if the image contains only one main object, or if objects in the image are clearly separated and there's no occlusion, like in Plate 39;
- allows "semantic" queries by using only primitive features: the shape drawn by the user will have a semantic meaning, while the system will work only on the primitive features of it, completely oblivious of what the shape means to the user. Retrieved images, though, will probably be semantically connected because they will contain similar shapes.

Weaknesses:

- no color information available; such information would be useful to perform queries like "find brown horses";
- edge detection can be problematic if the image does not contain a main subject clearly distinct from the rest of the image, as it is the case for the image in Plate 40, or if the image contains lots of edges (as very textured images, like the one in Plate 38, do);
- occlusion can be a problem: a contour meant to represent a butterfly might (depending on how shape is represented and on how similarity between shapes is computed) not match pictures of butterflies partially hidden by leaves, for example. This way, the semantic limits of this method can be seen;
- completely different objects may have the same contour when seen from a particular angle, leading to retrieval of non-relevant images.

4.1.2 Predefined Elements

Querying using predefined elements consists in placing icons representing entities (objects or concepts) on a canvas. Systems supporting this kind of query are QBIC [43], ImageSearch [84] (Plate 5) and SCORE [4]. Every icon corresponds to an entity that was recognized during database population (for example: sky, sun, grass, face, water, human face, bear, car, etc.), and informations about the presence and the position of these entities is stored with every database image. Database images are then ranked according to the presence of the entities appearing in the sketch and of their spatial relationships.

Advantages:

- no painting is needed (user skills not necessary to obtain a good result);
- no feature extraction from the sketch (thus avoiding the problem of having to compare features extracted from the sketch and features extracted from a digital image).

Weaknesses:

- the attributes of the entities associated to the icons cannot be modified: a "sea" icon will match any kind of sea, which may not correspond to what the user is looking for; this is not true in SCORE, in which the user can adjust the icons' attributes to better specify his/her query;
- limited number of concepts available, thus limited number of possible queries;
- relies quite heavily on object recognition (unless the available objects are extremely simple ones), or on manually extracted features.

4.1.3 Geometric Shapes

Pictorial queries containing only geometric shapes are produced using a simple paint tool (much like QBIC's one [43], see Plate 6), and can be used to specify the color of different regions of the target image, in a rather imprecise way.

Advantages:

- simple to draw;
- precise spatial placement of shapes possible, spatial constraints possible;

- use of color;
- possibility to produce incomplete sketches;

Weaknesses:

- lack of detail (looking for a particular image is not possible this way);
- comparing the shapes with what is extracted from the database's images is hard (not much in common, one can only say "there is a lot of blue in the upper area of the image);
- quality of result depends on quality of the sketch, which will not be high because of the paint tool used.

4.1.4 Freehand Drawing

Freehand drawing is the technique that allows the maximum precision in specifying the query, but is also the more complex technique to master (both for the user and for the image retrieval system). Freehand drawing can be limited (only a few available tools, like "pencil", "line" and "bucket") or not, and colors can be chosen from a palette ranging from a few colors to millions of colors. The drawn sketch can be complete or incomplete (i.e., some areas are left empty). Some examples of incomplete sketches can be found in Plate 41.

Advantages:

- total freedom, hence maximum precision for the object's shapes;
- color;
- incomplete sketches;
- very powerful if the user is looking for a particular image, but also useful to retrieve large categories of images (like "fields under a blue sky").

Weaknesses

- production of sketch is complex (users might require training);
- comparing features extracted from the sketch with features extracted from the digital images can be difficult;
- quality of the sketch is crucial.

4.2 How Much Information do We Need?

When developing a CBIR system allowing querying by sketch, one of the questions that needs to be answered is "how much information is needed in order to find the target images?". In short, we need to know if the set of features extracted from the query should be almost identical to that extracted from the database images, or if it can be a subset of it. In the first case, we will say the sketch is "complete", while in the second case, we will call it "incomplete".

It is clear that if the user is looking for groups of images the constraints will have to be looser, so the second solution should be chosen. When looking for a particular target image, on the other hand, both choices are possible.

If edge sketches and sketches built with predefined elements are by definition incomplete sketches (i.e., not every element present in the image descriptor must be in the sketch in order to consider the sketch and the database image as similar) the same cannot be said of geometric shape and freehand sketches. The choice between complete and incomplete sketches in these cases will depend on:

- the kind of query;

- ease of use of the system.

As it will be seen in this section, incomplete sketches appear to be a more flexible and easy to produce way to visually query an image database.

4.2.1 Complete Sketches

A complete sketch generates a description which corresponds closely to the one of its target images. In particular, if the sketch is composed of edges, this means that the user is asked to draw every "relevant" edge, and not only those of some of the objects in the image. Intuitively, we can say that such a way of querying the database will be useful only when looking for a particular image. The production of a sketch of this kind is bound to a series of problems:

- time: drawing a complete sketch can take a large amount of time;
- precision: especially if the user is sketching from memory, certain areas of the image will be drawn less accurately than others because considered "less relevant".

If some areas of the sketch are not faithful to the target image, chances are that the latter will not achieve a good similarity score. Furthermore, previous experiences [7] showed that the areas considered as relevant are in general smaller than those considered as not relevant, and that the latter often correspond to the background. A sketch with large areas not corresponding to the target image has few chances of resulting in a good set of retrieved images (Plate 42).

To avoid this problem, a saliency detection technique (like the one described by Itti et al. in [67] would be needed, and only parts of the sketch corresponding to salient parts of the database images should be used to assess the similarity between sketch and database images. This way, though, part of the user's work would become useless.

4.2.2 Incomplete sketches

If the user is allowed to produce a sketch which generates a subset of the features used to represent the target images, he/she can obviously concentrate on the parts of the image that are most relevant to him/her (see Plate 43 for an example). This way, all the areas painted by the user can be considered relevant (assuming that, if they were not the user would not have painted them).

The images in the database will be ranked according to the similarity between the painted regions in the sketch and visual information contained in the images. This comparison can be done on a local basis (like in PICASSO [28], in which the comparison is done on a region by region basis) or in a more global way (in their system, Jacobs et al. [69] compute a description for the whole image which is based on a wavelet decomposition). Comparison on a local basis will in general require extraction of region information from the database images (for example with a segmentation of the database images into regions); the comparison between regions will then be based on:

- color: histograms, distribution, average, etc.;
- spatial position;
- shape;
- size;
- texture information;
- etc.

Spatial relationships between the regions can also be part of the similarity computation.

While incomplete sketches can be used to retrieve a particular target image, they are a more powerful retrieval method, as they can be used to cluster images sharing given visual features. By sketching only a few regions, the user can retrieve all the images containing the same visual information (like

"all images with the sun in the top left corner and a textured green patch in the bottom right one"). By carefully choosing the image-to-sketch similarity function, it is possible to look for images containing the depicted areas exactly as pictured, in any part of the image, in any spatial arrangement, with different sizes, and so on. It is easy to see, then, that querying by sketch is a very flexible solution.

Incomplete sketches also have problematic issues associated to the fidelity of color, size and shape of the depicted areas, compared to the corresponding areas of the target image(s), as perception of color in the human is a complex process, involving physical and psychological elements. Notably, the perceived color of an object depends on its albedo, the illumination, the surface of the object and reflected light coming from other objects, just to mention some of the physical elements. When the user paints an object in isolation, he/she will try to match the color she perceives in the target image with the one he/she sees in the sketch. The different environment of the regions, though (white background in the sketch, other objects in the target image) will influence the perception of the color, so that the color used in the sketch will almost surely be different than the color in the target image. Furthermore, the knowledge that a certain object is of a given color (for example, "grass is green") can also lead to incorrect colors in the sketch.

Selection of the areas that are relevant can also be a delicate issue, especially if the computation of similarity is done on a local basis, with region information being extracted from the database images. Partition of the target image by the user may differ significantly compared to the one stored in the CBIR system, whether the latter was obtained automatically, semi-automatically or manually. In particular, regions obtained automatically may differ to those painted by the user because of the knowledge the user has about the content of the image. Hence, the user will tend to partition the image according to the objects depicted, while the system probably will not (except in specialized application, where models for the depicted objects are available).

Despite these shortcomings, querying by incomplete sketch seems to be the most powerful way to query an image database in a visual way, allowing searches for both particular images and categories of images. The method is intuitive, and production of the query image can be extremely fast.

4.3 Extracting Comparable Information from Sketches and Database Images

Sketches and digital images have different characteristics, yet a human being will in general be able to say that a sketch represents a particular image, or at least that an object contained in a sketch corresponds to an area of the image. This means that some features are conserved when a user paints a sketch of a particular image or of a part of it. The goal of a CBIR system is to manage to extract this information from both the sketch and the image, in order to be able to make a comparison of their contents.

This problem does not apply to contour sketches and to sketches built using predefined elements, since in the former the information extracted from the sketch already is of the same kind of that extracted from the database images, while in the latter there is no need to extract new information from the sketch, since the different icons correspond to entities which are represented by feature values. Hence, the only kinds of queries concerned are the ones where painting is needed.

4.3.1 Extracting the Information from Geometric and Freehand Sketches

In the early stages of this work, we observed that, even when given a powerful image production tool like Adobe PhotoShop, the users tend to paint sketches composed of patches of uniform color to represent homogeneous areas depicted in the images. Sketches produced this way bear a strong similarity to digital images having undergone a coarse region segmentation, or being reconstructed starting from the most important elements of a wavelet decomposition, as Jacobs et al. point out in [69]. Hence, information comparable to that found in this kind of sketches can be extracted from the database

images using one of these two techniques. Coarse segmentation, though, seems to have a slight advantage over wavelet decomposition, since region information can be made invariant to translation, rotation and scaling, while information coming from wavelet decompositions cannot.

Color segmentation appears to be an adequate method for extracting region information from images in databases that will be queried using incomplete sketches. It is the method that was chosen in this work. Issues connected to this image processing technique will be discussed in Chapter 6.

4.4 Conclusion

In this chapter, four techniques for querying an image database using a sketch were presented: contours, predefined elements, geometric shapes and freehand drawing. Furthermore, the utility of having the sketches covering the whole available surface was investigated and the problem of the extraction of comparable information from sketches and database images was examined.

As a conclusion, it can be said that querying by incomplete sketch by using freehand drawing appears to be the most intuitive and flexible way of visually querying a database. For this reason, it was the method chosen in this work, which will be described in detail in Chapter 5.

Our CBIR prototype, SimEstIm, which will be presented in the following chapters, represents only a small part (the query-image distance measure) of what needs to be contained in a complete CBIR system. Its main goal is the computation of the dissimilarity between a hand-drawn sketch and a digital image taken from a heterogeneous database.

4.5 Color Plates

PLATE 38. **Some images, as this one, may prove difficult to sketch if the sketch is to be composed of the most evident edges.**

PLATE 39. **Edge detection is easier if the image portrays a single object clearly distinct from the background.**

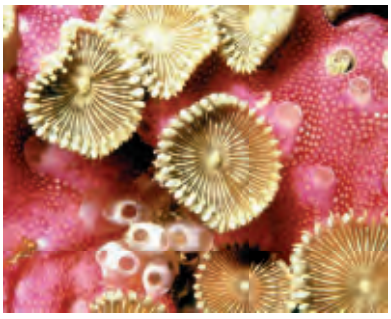

PLATE 40. **Edge detection on an image which does not contain a clearly distinguishable main subject can be problematic.**

PLATE 41. **Some examples of incomplete sketches (white areas are "don't care" areas) (images extracted from [69]).**

PLATE 42. **When painting complete sketches, users will in general draw the regions they considered as less relevant with less detail than the areas that in their opinion are characteristic of the image. Since not relevant regions are in general quite large (the background, for example, is often considered as not relevant by the users), the resulting sketch has large areas which bear only a slight resemblance to the corresponding areas of the target image.**

PLATE 43. **When producing an incomplete sketch, the user can concentrate on the areas of the target image he/she considers relevant.**

CHAPTER 5

Introducing SimEstIm

5.1 Goals and Characteristics of SimEstIm

The main contribution of this work is a measure that can be used to estimate the dissimilarity between a user-drawn incomplete sketch and a digital image belonging to an large (several thousands images) unrestricted database in which no additional data is associated to the images [8]. This estimation is done using features extracted from limited areas (regions) of both the sketch and the image.

In order to prove the measure's usability, a CBIR prototype called SimEstIm (SIMilarity ESTimation for IMages), was built around the measure. SimEstIm allows retrieval of images from a database by visual similarity with an incomplete user-drawn sketch (examples of which can be seen in Plate 45), positioning itself in the "query by user-produced example image" category presented in Chapter 3. Figure 12 summarizes how SimEstIm works. The existing systems most similar to it are QBIC [9], PICASSO [28], and the works by Jacobs et al. [69] and Cha et al. [21].

5.1.1 Retrieving Known Images from Large Databases

The framework in which this work positions itself is that of image retrieval by user-produced pictorial example. In particular, our experiments were centered on a situation in which a user wants to retrieve a particular image (or a set of visually extremely similar images) from a database, while having a vivid memory of at least a part of the image. We believe that for this task, querying by sketch can be a good method, especially if no additional information is associated to the images. As we mentioned in Chapter 2, though, if other kinds of information are available (keywords, textual annotations, etc.), querying by sketch should be combined with them. Querying by sketch has also been chosen by Jacobs et al. [69] (examples of sketches for their system can be seen in Plate 44). Both Jacobs' work and ours rank database images according to their visual similarity to the sketch.

5.1.2 The Query

Different kinds of sketches are possible in a CBIR system using them as queries. As it was explained in Chapter 4, sketches range from line art drawings to full-color freehand sketches. In this work, we decided to use incomplete color freehand sketches built with a simple paint tool containing only the "pencil" (with variable pen sizes) and "bucket" tools, but allowing the user to select colors from the complete 24-bit RGB color space. Some examples of sketches can be seen in Plate 45.

In early experiments in which the users were asked to work with a limited color palette we realized that most of the time they could not find the color they were looking for and were not satisfied with their sketch. These experiences led us to letting the users select the wanted color with a 24-bit RGB

color selector, so that they can produce an image that is as close as possible to the image they have in mind.

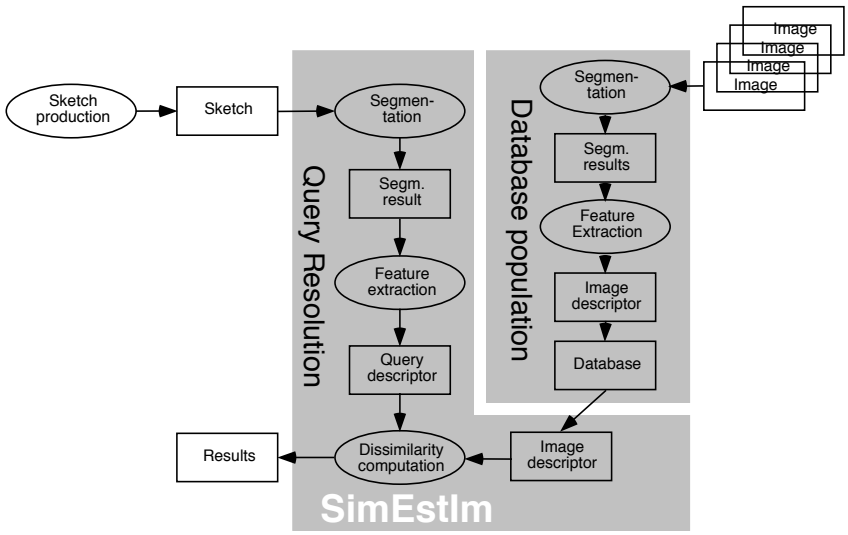

FIGURE 12. CBIR using SimEstIm.

Not limiting the user to geometric shapes (rectangles, lines, ellipses) also goes towards letting the user produce an image similar to his/her memory of the target image. In fact, shapes found in images are in general more easily reproduced by freehand sketching than by composing geometric shapes (which is a long and tiresome process, going against our goal of having the user produce the sketch rather quickly). For this reason, the drawing tool used to produce the sketches allows freehand painting.

All while wanting to allow the user to produce a sketch that is similar in complexity to the sought image, it is clear that limits must be set on how complex the sketch should be. Allowing users to produce sketches using all the features of a program like Adobe Photoshop, for example, would lead to very dissimilar sketches of a same target image because of the different skills of the users. In order to extract information from the sketches, then, a multitude of feature extraction methods (depending on the complexity of the sketch) may be needed. The selection of the correct method, which would have to be performed at run time, after the sketch is submitted, may also lead to a longer waiting time for the user. In order to "level the field", we decided to ask the users to paint their sketches using only the "pencil" (with different pencil sizes) and "bucket" tools. These restrictions limit the complexity of the produced sketches, and result in more uniform sketches (the difference between images produced by skilled and unskilled users is smaller). Furthermore, such limitations mean that the sketches will be produced more quickly. Sketches with a simpler structure are also more easily processed, which results in less waiting time for the user during the feature extraction phase. It must be said, though, that it is possible that some kinds of images are too difficult to sketch using this method, like the one pictured in Plate 46.

5.1.3 Applications

The dissimilarity measure (and the underlying feature extraction and region matching processes) we propose can be used in any application where the user is looking for a specific image and remembers at least some parts of it, and is especially helpful for collections in which the images do not have extra data associated. With some restrictions, it can also be used to retrieve broad categories of images, like "head shots with dark background" or "sea under a grey sky". Examples of fields that could benefit from our work are publishing, advertisement, design and retail. An additional application would be performing searches in personal photo collections stored in digital form.

5.2 Important Concepts Behind SimEstIm

How do SimEstIm and the dissimilarity measure SimEstIm is built around work? In this section we introduce the main concepts that lie behind our prototype and that allow the production of a ranking according to the dissimilarity between the database images and the sketch.

5.2.1 Locally Computed Image Features

One of the most important characteristics of the dissimilarity measure around which SimEstIm is built is that it relies entirely on features computed from limited areas of the images (henceforth local features) and not from the whole image (global features).

Local features have several advantages over global features. The most notable are:

- small details are not drowned by data coming from other parts of the image: for example, if an image contains a small yellow object in the bottom right corner, and some large yellow objects in other parts of the image, a color histogram (a typical global feature) would not be able to tell that there is a small yellow object in the image; all we would be able to deduce is that there is some yellow;
- it is possible to localize objects: extracting features from limited areas of the image, it becomes possible to know in which area of the image a desired feature is located. As an example, if we are looking for a particular contour in an image, and all we have is a histogram of the edge directions, we will be able to say that the sought contour is in the image but we won't know where it is located. If on the other hand, each contour in the image is represented separately, along with information about its position in the image, we will be able to locate it in the image.
- it is possible to compare the wanted features only with a subset of the image, or look for a subset of the target image: when looking for green objects in the lower part of the image, only features coming from accordingly placed regions need to be compared.

5.2.2 Regions

Since the dissimilarity measure we developed is based on local information, it is necessary to define how the areas from which the information is extracted are defined and delimited.

Other CBIR systems split the image in several rectangular regions, regardless of the image's contents, as does FourEyes [112] (Plate 47), or use segmentation algorithms in order to form connected regions from which information is extracted, as it is the case with BlobWorld [17] (Plate 48).

Our work differentiates itself from other approaches by allowing non-connected regions (examples of non-connected regions can be found in Plate 49). These regions are obtained by:

- segmenting the image into connected regions;
- iteratively merging pairs of regions according to a compatibility criterion which allows the merging of regions that are visually similar, even if they do not form a connected region when merged. The merging process is stopped when there are no more pairs of similar regions.

As a result, objects and entities (like flowers in a field or faces in a crowd) that are visually uniform tend to be represented by a single region, despite the fact that no object recognition is performed on the images.

In order to compute the dissimilarity between a sketch and a database image, regions must be compared. Since there is no guarantee that one region in the sketch corresponds exactly to one region in the database image and vice versa, we may be forced to compare groups of regions in the sketch with groups of regions in the database image. Region merging, though, produces comparable results, thus allowing the comparison of regions on a one-on-one basis. This way, the number of comparisons needed to compute the dissimilarity between a sketch and a database image is significantly reduced.

5.2.3 Feature Extraction Through Region Segmentation and Region Merging

As we said in the previous paragraph, regions are obtained as product of a segmentation process. Various approaches are possible here, and some of them are listed in Section 6.1.

Sketches and digital images are very different kinds of images (even if a good sketch will be visually similar to its target image), but the extracted features need to be comparable. In order to achieve this, we decided to reprocess the segmentation result of the database images, which is in general very complex and contains over 100 regions, in order to obtain a segmentation that is as similar as possible to the sketches the users may produce. This is done by iteratively merging pairs of regions (Plate 50), using a criterion based on the shape, position, size and color of the regions, and results in non-connected regions that tend to represent entities in the image (as seen in Plate 49). The region merging process will be explained in detail in Section 6.3.4. As we said before, region merging allows us to compare regions on a one-on-one basis, significantly lowering the number of operations needed to compare a sketch and an image.

Each region in the image description contains information about its position, size, color and shape. These descriptors will be introduced in Section 6.3.2.

5.2.4 Multiple Segmentation Results

Users produce extremely different sketches even when they are looking for the same target image, as shown in Plate 51. In particular, some of the reasons behind these differences are:

- differences in the drawing skills of the users, resulting in more detailed sketches for the more skilled ones;
- previous experiences with SimEstIm, which may influence the users (for example, knowing the fact that very small details are lost in the segmentation process, and are thus useless);
- cultural differences, leading to considering different parts of the target image as significant and drawing those instead of others.

All these reasons lead to sketches that contain different levels of detail. Since we want our dissimilarity measure to be flexible, we decided to store several segmentation results for each database image, hoping that at least one of them will contain regions similar to those painted in the sketch which, on the other hand, will be represented by a single segmentation result. The segmentation results of the database images are organized hierarchically and contain a decreasing number of regions, allowing an accurate computation of the dissimilarity with sketches containing different degrees of detail, as we will explain in Section 6.3.5

5.2.5 Computation of Dissimilarity

In order to rank the images according to their similarity to the sketch, their respective descriptions must be compared. Our dissimilarity measure does this by computing the dissimilarities between the regions contained in the segmentation result of the sketch and those contained in the multiple segmentation results of the database images. Region dissimilarities depend on their color, shape, position and size, as it will be explained in Section 7.2.

The region dissimilarities are then used to compute the dissimilarity between segmentation results. The used algorithm, introduced in Section 7.3, tries to form pairs of similar regions, and assigns penalties to the regions of the sketch that are not similar to any region of the current database image segmentation result.

Finally, the dissimilarity between a sketch and a database image is computed as the lowest dissimilarity between the segmentation result of the sketch and the segmentation results of the database image. The reasons behind this choice will be explained in Section 7.4.

5.3 Color Plates

PLATE 44. Sketches used in Jacobs et al. [69] are remarkably similar to the ones used in SimEstIm, but the internal description of the images is completely different.

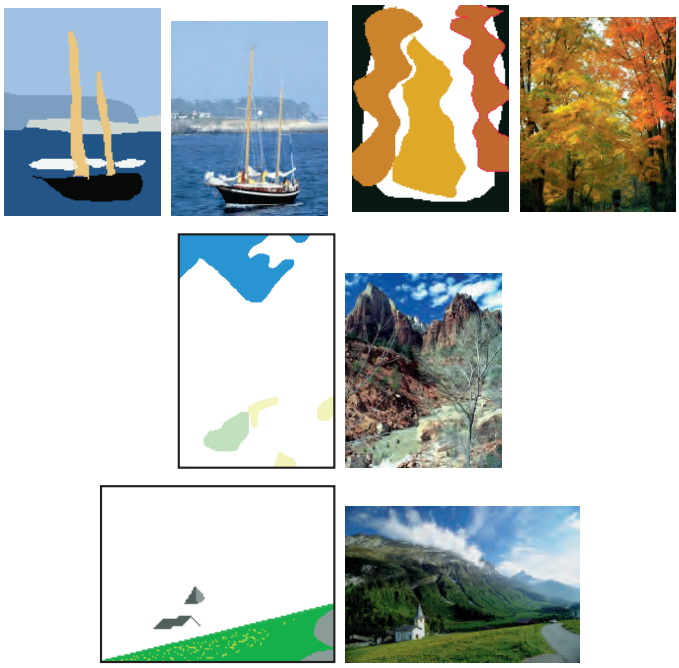

PLATE 45. Some sketches painted by SimEstIm users, along with the target images.

PLATE 46. Some images seem to be more difficult than others to sketch. In this particular case, the users had huge problems in finding a way to sketch the reflections and the soft changing colors of the ceiling.

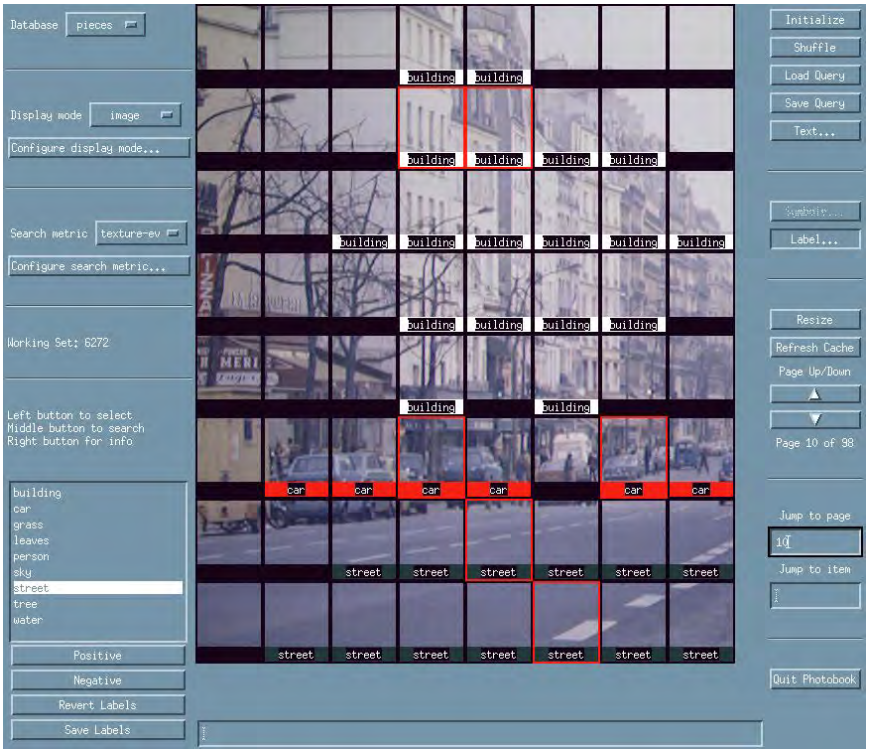

PLATE 47. **FourEyes [112] splits an image regardless of its contents, and tries to associate a label to each region.**

PLATE 48. **In Blobworld [17] the images are segmented in a rough way, and the regions are connected.**

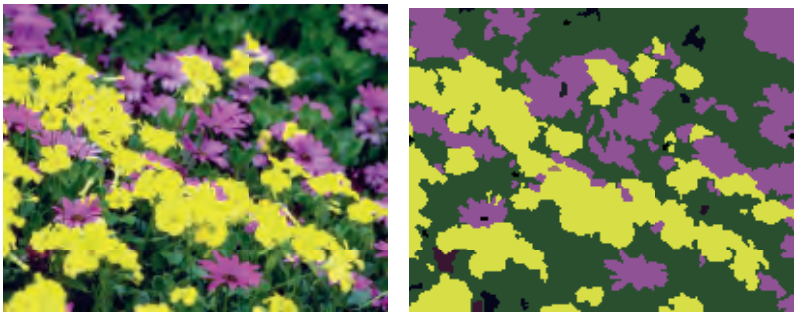

PLATE 49. **In our description, regions can be non-connected: in this example, the yellow flowers form one single region; the same goes for the purple flowers.**

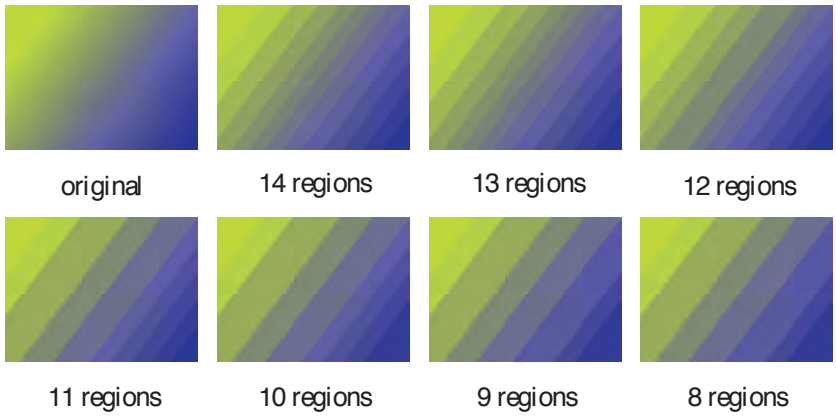

PLATE 50. **After an initial segmentation result is obtained using a standard color image segmentation algorithm, regions are iteratively merged until the minimum similarity between two regions of the image exceeds a given threshold.**

PLATE 51. **Sometimes, sketches of the same image made by different users can be dramatically different (white areas are unspecified). A good CBIR system should be able to consider the target image as similar to both sketches (this is the case with SimEstIm).**

CHAPTER 6

Feature Extraction in SimEstIm

The goal of feature extraction is to compute reliable descriptions of the database images and the sketches. In this chapter, we will explain what kind of image description is extracted. In Section 6.1 we will talk about color segmentation algorithms, listing some of the possible approaches. The segmentation algorithms that were used in this work will be explained in Section 6.2. Then, in Section 6.3 we will introduce the region features that characterize the regions extracted from database images and sketches and we will explain how, by a region merging process, we manage to obtain similar descriptions for the database images and the sketches.

Goals of Feature Extraction

As we said, the goal of feature extraction is to extract descriptions of the database images and for the sketch which are representative of their contents and that allow the computation of a significant dissimilarity value between them. This task is made complex by the different origin of the sketches and of the database images. Notably, in database images:

- objects will not be composed of a single color, because of illumination, reflections between objects, geometry of the objects, noise, camera characteristics, color sensor limitations and noise;
- borders between the objects may not be clear, because of shading or clutter; objects may not be clearly detectable.

On the other hand, in sketches like the ones we are dealing with (Plate 45):

- the image contains patches of color;
- each patch is composed of a single color;
- borders between patches are clear, because patches were painted using the “pencil” tool.

In our case, we assumed that the users would paint patches of color roughly corresponding to visually uniform areas in the image. These patches contain very little information, compared to the regions of the digital images they are meant to represent. For this reason, it was necessary to develop a feature extraction algorithm that would return a description of the database images similar to the one extracted from the sketches. In order to achieve this, we decided to use a region segmentation algorithm followed by a region merging process.

Region Segmentation

It is only natural that, when sketching an image, the user will paint blobs representing the objects pictured. It would be a shame, then, not to use this information to retrieve the image. Of all the image processing techniques available, color region segmentation seems to be the most appropriate for extracting the corresponding areas from the database images.

It must be said, though, that the number of different color image segmentation techniques is very large. The following section gives an insight on the various kinds of algorithms currently available. In Section 6.2 the various techniques we used in our work will be presented

6.1 Color Image Segmentation Techniques

As it was suggested in Chapter 4, one of the ways to extract information from digital images for indexing purposes is to segment them into regions and compute some well-chosen features of the extracted regions. This method is especially useful for comparing digital images between them and for comparing a digital image with a user-produced sketch.

While image segmentation for grayscale images has received a lot of attention by the research community over the last 30 years, color image segmentation had been somewhat neglected until the mid-Eighties. In the recent years, though, a lot of effort has gone into trying to find solutions for this problem, which consists in extracting from the image one or more connected regions satisfying a uniformity criterion which is based on features derived from spectral components, as it is defined by Skarbek and Koschan in [126]. Similar definitions can be found in [106].

In this chapter no particular color image segmentation method will be presented, and only the various options open to someone wanting to segment a color image will be discussed. These options belong to the following domains:

- region definition;
- color space;
- additional features;
- method used to build the regions;
- type of segmentation result.

6.1.1 Defining the region

Prior to developing an image segmentation method, the concept of region must be defined, as it will influence deeply several other choices to be made later. Basically, a region can be defined as [126]:

- a connected component of a pixel set specified by a class membership function defined in a color space (pixel-based definition);
- a (maximal) connected set of pixels for which a uniformity condition is satisfied (area-based definition);
- a connected set of pixels bounded by edge pixels creating a color contour (edge-based definition);
- a set of pixels corresponding to a surface or an object of homogeneous material (physics-based definition).

By changing “color space” to “feature space” in the first definition and removing the word “color” in the third one, we can include regions produced by texture segmentation in the definitions. In fact, texture segmentation is often the result of a clustering process over texture features (structural or statistical) computed over the image pixels.

Choosing one of these definitions will result in a different behavior of the segmentation algorithm. For example, if the edge-based definition is chosen, finding regions must be preceded by edge detection, while in all the other situations the contours of the extracted regions are determined only at the end of the segmentation process. Similarly, choosing a region definition over another may determine how the regions will be formed: while features computed over single pixels suffice to form pixel-based regions (as in histogram-based methods), area-based methods rely on properties computed over sets of pixels (in general homogeneity of the region).

6.1.2 Choosing the Color Space

A multitude of different color spaces have been documented in the literature [64]. The most widespread of them, because of its tight relation with digital cameras and CRT devices, is the RGB model which, as it will be seen, is not the best choice for image segmentation because of its peculiar properties. Other color spaces, such as the HSV one, try to simulate the way human beings think about color, while perceptual color spaces, like the CIELUV one, aim at creating a relation between the perceived difference between two colors and the euclidean distance between the position of the two colors in the color space. In any case, the choice of the color space in which the image processing will be done will depend on the fixed goals. For example, if the algorithm aims at segmenting the image into regions corresponding to the various objects depicted, then a color space with the following properties, pointed out by Gevers and Smeulders in [49], will be needed:

- robustness to change in the viewing direction;
- robustness to change in object geometry;
- robustness to change in the direction of the illumination;
- robustness to change in the intensity of the illumination;
- robustness to change in the spectral power distribution of the illumination;

On the other hand, if the wanted result is the list of the most prominent color changes in the image, and that invariance to shading is not required, these properties will not be needed. Finally, if what is required is a segmentation into perceptually homogeneous color regions, or if the chosen segmentation method requires an isotropic color space (as it is the case for the Comaniciu-Meer algorithm [26]), the best choice will be a perceptually uniform color space.

THE RGB COLOR SPACE

The RGB (Red, Green, Blue) color space [64] is one of the most popular color spaces for digital images, as it is tightly linked to the way CRT displays work. Colors are represented according to the additive color scheme (i.e., white is obtained by mixing the three base colors at full intensity, as it can be seen in Plate 52). This scheme is not natural for human beings, who tend to think in terms of hue and luminosity and are more familiar with the subtractive color mixture scheme (Plate 53).

The advantages of RGB come from its simplicity: the RGB space is a unit cube, hence distances are easily computed. Furthermore, color reduction and the computation of color histograms are extremely easy. On the other hand, distances within the cube have little to do with the differences perceived by human beings, and no kind of invariance is possible using this color space.

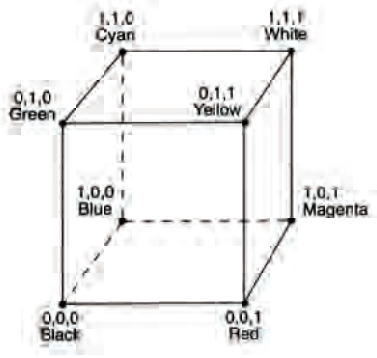

FIGURE 13. The shape of the RGB color space is the unit cube (image taken from [68]).

The normalized version of the RGB coordinates, rgb [64], is invariant to change in viewing direction, surface orientation, illumination direction and illumination intensity [49], but the shape of the rgb domain no longer is the unit cube.

$$r = R/(R+G+B),\ g = G/(R+G+B),\ b = B/(R+G+B)$$

THE HSV COLOR SPACE

The HSV (Hue, Saturation, Value a.k.a. Luminosity) [64] coordinates can be computed directly from the RGB ones and are easier to understand than their CRT-related counterparts. The resulting color

$$H(R,G,B) = \text{atan}(\sqrt{3}\cdot(G-B)/(2R-G-B))$$
$$S(R,G,B) = 1 - min(R,G,B)/(R+G+B)$$
$$V(R,G,B) = (R+G+B)/3$$

space is cone-shaped (Figure 14).

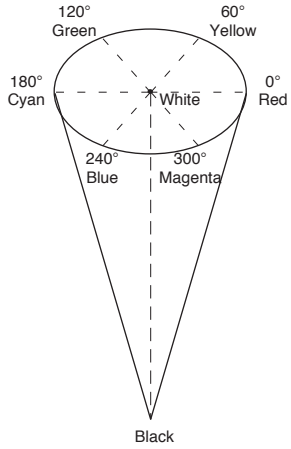

FIGURE 14. The HSV color space.

Hue, saturation and luminosity are, in fact, much closer to the human understanding of color than the additive scheme of the RGB space. This advantage places HSV closer to the border between perceptual and non-perceptual color spaces than RGB. HSV remains, though, a non-perceptual space as "none of the axes is perceptually uniform, meaning that a given change in any coordinate will result in a variable apparent change, according to the location in color space" [64]. Furthermore, the peculiar shape of this color space (a cone) makes it difficult to perform color reductions and compute color histograms, as the colors are not distributed evenly with respect to the coordinates (there are more colors where the luminosity is larger). Computing distances is also awkward, because of the circular nature of the hue component and the perceptual discontinuities that can be noticed around the value axis [7] and for low luminosity values [126].

Finally, as it was shown in [49], only the H and S components have some invariance properties: hue and saturation are invariant to change in viewing direction, object geometry, illumination direction and intensity; hue is also invariant to highlights.

PERCEPTUALLY UNIFORM COLOR SPACES

In the ideal perceptually uniform color space, numerical distances between colors correspond to the difference perceived by a human observer. Such a color space has been sought for years by computer scientists, psychologists and artists [99].

Several attempts at defining a color space with this property have been made. These include the Munsell color system [101][92] and the Optical Society of America Uniform Color Scale [91]. In 1976, the Commission Internationale de l'Eclairage (CIE) introduced the L*u*v* and L*a*b* color spaces [27], also known as CIELUV and CIELAB. L*u*v* was developed for additive color media such as

CRT displays, while L*a*b* is for subtractive media. In both spaces, L* is the psychometric lightness, while the remaining two dimensions are coordinates on a uniform chromaticity diagram.

In order to obtain numerical distances that correspond to perceived differences between colors, it is necessary to keep in mind that these color spaces were built under strictly controlled environmental conditions. Hence, "to maintain the perceptual uniformity of the space, the settings of these parameters in the final application must match the conditions existing when the data were originally recorded" [99]. Among these we have "(a) the size of the color samples, (b) the spacing between color samples, (c) the luminance and chromaticity of the background on which the color samples were compared, and (d) the luminance and chromaticity of ambient light in the test environment" [99]. Obviously, in a CBIR application these conditions are not reproduced: the conditions under which the images were taken, then digitized, are in general not known, and the environment in which the user performs the search isn't controlled either; this lack of controlled conditions sometimes leads to differences between the obtained color distances and the expected results.

The use of such a color space becomes quite complex, as some kind of calibration (like the ones explained in [132] and [29]) must be done, which is possible only if the system generating the images is known. Furthermore, the L*u*v* and L*a*b* spaces are not perfectly uniform: there remains a ratio of about 6:1 in the maximum to minimum distances corresponding to a given perceptual difference [64]. Despite all these restrictions, the L*u*v* space "probably represents the best that can be done with a space having an associated uniform chromaticity diagram" [64] and proves extremely useful each time the human reaction to color must be taken into account.

OTHER COLOR SPACES

A large number of color spaces other than those mentioned above have been used to perform segmentation, like the $c_1c_2c_3$, $l_1l_2l_3$ and $m_1m_2m_3$ ones introduced by Gevers and Smeulders in [49], which were designed in order to have the properties presented at the beginning of this section, and the IJK space presented by Verikas et al. in [142], chosen because the three components are almost uncorrelated and have zero covariance.

6.1.3 Additional Features

Extraction of features from the pixels is not limited to color characteristics. In fact, spatial and texture features that are useful for color image segmentation can also be computed.

In general, the only spatial feature associated to a pixel is its position in the image. Texture features, on the other hand, are in general computed on a window surrounding the pixel. A large number of texture features have been described in the literature: in [144], Wang et al. characterize textures by mean, variance, correlation, entropy, contrast and homogeneity; in the BlobWorld system [18] polarity, anisotropy and contrast are used, while in [122] Santini and Jain use luminosity, scale, verticality and horizontality to estimate the similarity between texture samples. For a review on other ways to represent texture, see [109].

Various algorithms are then used to form regions according to these features, which can be used in combination with color features (like in BlobWorld [18]).

6.1.4 Forming Regions

Several methods have been used in the literature to group pixels into regions. In general, these are divided into:

- region-based techniques;
- edge detection techniques.

In edge detection techniques, regions are built once their contours are known. These are detected using different methods: elementary gradients [103], Laplacian [5], difference of offset gaussians

[146], Markov Random Fields [111][31]. Contours need then to be reprocessed in order to form closed contours, which define the regions.

In this section, we will limit ourselves to describing some region-based techniques: histogram-based methods, clustering methods, region growing and split and merge techniques. The reader should keep in mind that these methods can be used in isolation or combined, taking advantage of their respective strengths.

HISTOGRAM-BASED METHODS

This kind of method consists in computing some features (typically the color) for each pixel in the image, and then building an histogram representing the distribution of the features over the image. Histogram peaks are then selected and regions are formed around them. These methods are especially used for object versus background segmentation, like the ones presented in [12] and [86].

CLUSTERING IN FEATURE SPACE

There are several methods for grouping the pixels into regions according to the features computed for every single pixel in the image. These groups (the clusters) are in general characterized by a representative point (an element of the cluster, the average of all points in the cluster or a predefined point in feature space). Among the clustering methods used in color image segmentation we have Nearest Neighbor (used in [42]), K-means ([28]), fuzzy C-means ([65][85]), ISODATA ([14]), median splitting ([140]), Minimum Volume Ellipsoid (defined in [116] and used in [73]), mean shift [26] and neural networks [144][142].

The interested reader can find descriptions of these and other clustering algorithms in [128].

REGION GROWING

In region growing techniques, the algorithm starts from (automatically or manually selected) seeds, which are pixels or small groups of pixels. The selection of the seeds is in general nontrivial, but heuristics for this task are available (like the ones used by Deng et al. in [36]). Regions are then constructed by iteratively adding neighboring pixels to the regions according to different strategies, that can be local (similarity between a pixel in the region and the pixel to be added) or global (satisfaction of a uniformity criterion for the region). The watershed algorithm (used in [98]) belongs to this category.

A particular variation on region growing (although it was used on grayscale images only) was introduced in by Revol and Jourlin [115]: while in the “classic” approach, pixels are assigned in a definitive way to a region, in this version the pixel assignments are reconsidered on each step to minimize variance in the region.

SPLIT AND MERGE

This technique consists in starting the process with an initial segmentation result (for example, the whole image considered as a single region), and then splitting the regions that do not match a given uniformity condition. The splitting phase is followed by the merging one, where similar neighboring regions are merged [123][137]. Splitting and merging can be repeated iteratively until a global uniformity condition is satisfied.

6.1.5 Segmentation Result

Another important factor when developing or selecting a segmentation method is the kind of result. In general, we can distinguish two situations:

- regions correspond to objects in the image;
- regions correspond to uniform areas in the image.

Regions Corresponding to Objects in the Image

This kind of result could be extremely useful for query by example CBIR systems, as it would allow queries like “find all images that contain this object”. Queries of this kind, though, would require region features invariant to the viewing angle and to the illumination. Furthermore, it must be said that this kind of segmentation performs well only under controlled environment, typically in a “object versus background” situation, in which object contours are easily detected. In QBIC [43], for example, the objects are detected using an assisted segmentation method, while in [50] the user is requested to draw a imprecise contour around the objects. For this reason, some query by example systems, like Blobworld [17] and NETRA [88], use segmentation results corresponding to uniform areas in the image rather than whole objects.

Regions Corresponding to Uniform Areas in the Image

Regions corresponding to uniform areas in the image can be extremely useful if the query is a user-produced pictorial example. In fact, in this kind of query only few details are present, and the image is composed of fairly large blobs of uniform color and/or texture. CBIR systems with queries of this kind are PICASSO [30] and QBIC [43]. Regions of this kind are also used in query by example systems, like Blobworld [17] and NETRA [88].

The kind of region obtained depends deeply on the uniformity criterion chosen in the segmentation: regions that are uniform color-wise and regions that are uniform texture-wise will be significantly different.

Another important factor is the detail of the segmentation. An image can be oversegmented, with every object being split into its component parts that are visually different, or undersegmented, with several objects being merged into one single region (Plate 54). While oversegmentation may seem better because the quantity of information is larger, the number of pairs of regions to be compared during the retrieval phase will be very large, and the number of comparisons will explode if n regions in the query can be compared with m regions in the database image. On the other hand, if the image is undersegmented, there is no way of performing a search for a small detail if it was lost during the segmentation process. Selecting the “right” number of regions, so to avoid oversegmentation and undersegmentation, becomes an important task; unfortunately, it is also a very complex one (the cluster validation problem [39]). To avoid this problem, in this work several segmentation results are stored, and the query is compared to each one of them. This approach is similar to the one taken in the PICASSO system [30].

6.1.6 Conclusion

In this section, several possibilities for color image segmentation were presented. While the list is by no means exhaustive, it should give the reader an idea of the multitude of possible combinations.

The choice of a color image segmentation method for a CBIR system depends essentially on the kind of queries that will be submitted to the system, as information extracted from the image will have to reflect the kind of information that the query will convey.

Query by example systems would probably take best profit from segmentation methods able to segment the individual objects, but these methods tend to work poorly if the database is not restricted in some way (typically by having only one or a few objects in the image, pictured against a uniform background). If regions correspond to objects, then visual “semantic” queries are possible. On the other hand, if the query is a drawing produced by the user, segmentation of the database images into uniform regions would allow the extraction of similar information from queries composed of color patches and from database images. This is the approach that was chosen in our case.

6.2 Segmentation Algorithms Used in SimEstIm

In this work, the goal of segmentation is to partition the image into regions roughly corresponding to uniform areas, producing regions that can be considered similar to those the user will produce when painting a sketch. A multitude of approaches to color segmentation have been presented in the literature, as seen in Section 6.1.

Among the approaches tested in this work there are:

- clustering in color space according to the peaks of the color histogram and then projecting the clusters back to the image;
- clustering in color space according to the colors of the pixels of the image, then projecting the clusters back to the image;
- the Batchelor-Wilkins algorithm [14];
- the ISODATA algorithm [14];
- the Comaniciu-Meer algorithm [26].

Results obtained with these approaches will now be presented.

6.2.1 Segmentation According to Histogram Peaks

This technique is very similar to the work presented by Kanakanhalli et al in [74], and consists in:

- computing a color histogram of the image;
- selecting the histogram peaks (a histogram bin is considered a peak if it contains more elements than any bin surrounding it);
- assigning points in the image to the peak whose color is the closest.

By working this way, regions in the image are formed according to color information only, meaning that two objects approximately of the same color lying in the opposite corners of the image will belong to the same region (i.e., the algorithm segments the image into non-connected regions). If we want to produce connected regions, we have to split these non-connected regions into their connected components, ending up with a large number of small regions, because of noise and quantization effects, as shown in Plate 55. Thus, in order to obtain regions comparable to those the users will draw in the sketches, we would need to reprocess the extracted regions, a process that would be both complex (requiring a complex method to decide which regions should be merged and how) and long (because of the very large number of regions extracted).

On the other hand, taking non-connected regions produced with this technique into account during the computation of dissimilarity between the sketch and the digital image would greatly increase the complexity of the computation. In fact, if a target image contains two separate areas of the same color (which end up in the same region because of their color) and the user produces a sketch containing a color patch corresponding to only one of them, the dissimilarity function has to try to verify if the region coming from the sketch can be considered a subregion of the region extracted from the digital image. Furthermore, there is no guarantee that histogram peaks correspond to large uniform areas in the image.

To summarize, the advantages of this technique are:

- it only requires two passes through the image to segment it;
- the number of clusters does not need to be known in advance;
- there are no parameters to be tuned.

On the other hand, the disadvantages are:

- regions are based only on the color information;
- histogram peaks do not necessarily correspond to prominent uniform areas in the image;
- building an histogram in which bins have a clear neighboring relationship is difficult with some color spaces (like HSV, in which colors are not distributed evenly).

6.2.2 Segmentation After Parsing of the Image Colors

This technique consists in parsing the image and creating a new cluster center every time the color of the current pixel is far enough from the color of all the already created clusters. The image is parsed a second time to assign the pixels to the cluster center whose color is the closest.

Like the previous one, this technique generates regions which are based on the color information only. Furthermore, the visiting order of the pixels deeply influences the final result, as in the example shown in Plate 56.

The advantages of this segmentation technique are:

- it requires only two passes through the image;
- only one parameter has to be set (the minimum distance between cluster centers);
- the number of clusters is not known a priori, although it must be said that it is strongly influenced by the aforementioned threshold.

On the other hand, disadvantages are:

- the visiting order of the pixels influences the final result;
- it is extremely sensitive to noise;
- regions are based on color information only.

6.2.3 Batchelor-Wilkins Algorithm

The Batchelor-Wilkins algorithm is a general-purpose clustering algorithm, and it was not developed with image segmentation as its main goal. A few experiments were made to see if it was a viable solution for SimEstIm.

The algorithm iteratively creates clusters by scanning the points in the dataset and works as follows:

1. select one point of the dataset as the first cluster center C_1;
2. select the point farthest from C_1 and select it as the second cluster center, C_2.
3. compute the distance to the already selected cluster centers $C_1, \ldots, C_n$ for every remaining point $p(x, y)$ in the dataset and save the minimum $m(x, y)$ for each point;
4. take the maximum $M = \max_{x, y}(m(x, y))$ of the computed distances; if it is below a threshold τ, select the corresponding data point $p(x, y)$ as new cluster center C_{n+1};
5. return to point 3 until $M < \tau$;
6. assign the points in the dataset to the closest cluster center.

In order to use this algorithm and avoid the problems connected with forming clusters according to the color information only, a distance measure between points in the image which takes into account both the position of the pixel and its color must be developed. This task turns out to be more complex that it could be expected. In fact, using the euclidean distance after merging the three color coordinates and the two spatial coordinates to form a 5-dimensional space does not yield good results, even after normalization of the components, because this 5-dimensional space is not isotropic.

For this reason, we decided to separate the color contribution from the spatial one. Let x, y be two points in the image, containing each three color coordinates and two spatial coordinates. In the experiments, the following distance was used:

$$d(x, y) = \alpha \cdot d_c(x, y) + (1 - \alpha) \cdot d_s(x, y)$$

where $d_c(x, y)$ is the color distance between the points, while $d_s(x, y)$ is the distance between their locations in the image. Finding the right α, though, can be cumbersome, since regions tend to be dominated by one of the two distances, like in the example shown in Plate 57.

Furthermore, the result is deeply influenced by the choice of the first cluster center. In order to minimize problems connected with this choice, it could be decided to set the first cluster center as a point of the most represented color, which would require one scan through the image to generate a color histogram. The choice of the position of this point would still be problematic, though. In the experiments, the first cluster center was set to pure black at position (0,0) and was removed from the list of cluster centers as soon as two other cluster centers had been found.

Advantages of the Batchelor-Wilkins algorithm are:

- the number of clusters does not need to be known in advance;
- only the threshold τ must be set.

Weaknesses of the technique:

- one pass through the image is required for every cluster;
- the result depends on the choice of the first cluster center;
- finding a good distance is cumbersome.

6.2.4 ISODATA

Like the Batchelor-Wilkins algorithm, ISODATA is a general-purpose clustering algorithm. It was tested in this context because it tends to form homogeneous clusters, splitting clusters whose standard deviation is larger than a given threshold. Hence, it looked like a good candidate for producing the homogeneous regions that were needed in the image descriptions.

The algorithm produces a partition of the dataset by iteratively splitting and merging regions. Parameters to be set are:

- the number of desired clusters M;
- the minimum number of points in a cluster η;
- the maximum standard deviation allowed σ_s;
- the minimum distance required between clusters δ;
- the maximum number of pairs of clusters that can be merged during a single iteration L;
- the number of allowed iterations I.

The partition is produced as follows:

1. choose some initial cluster centers;
2. assign points to their nearest cluster center;
3. recompute the cluster centers (average of the points assigned to the cluster);
4. discard clusters containing too few elements;
5. compute standard deviation for each cluster; split clusters whose standard deviation is larger than σ_s; if the partition already contains more than half of the desired clusters, the cluster is split only if it contains at least $2 \cdot (\eta + 1)$ points;

6. compute the distance between the cluster centers; clusters whose centers are closer than δ should merge;
7. repeat until desired number of iterations I is reached.

In order to work the best, ISODATA requires an isotropic feature space (i.e., the distances between points measured with the euclidean distance correspond to the perception of the dissimilarity between the points). Unfortunately, this is not the case if the feature space is built by merging the three color components and the two spatial components associated with every pixel in the image. For this reason, segmentation results obtained with the ISODATA algorithm may appear surprising.

Advantages of ISODATA:

- homogeneity of clusters is guaranteed because of the way clusters are built.

Disadvantages of ISODATA

- the number of clusters must be known in advance (although there are modifications of the algorithm that allow to bypass this restriction);
- several parameters to be set, some of which are not intuitive;
- requires an isotropic feature space to return good results;
- one pass through the image is needed for every iteration;

6.2.5 Comaniciu-Meer Algorithm

The Comaniciu-Meer algorithm [26] is a color image segmentation technique based on the mean-shift algorithm [22][46]. It avoids the problems associated with merging spatial and color information by working in both spaces alternatively. Henceforth, we will talk of “image domain” for the spatial information and of “feature domain” for the color information, which is represented in the L*u*v* color space, because the mean-shift algorithm needs an isotropic space in order to be effective.

The algorithm was designed to perform three kinds of segmentations:

- undersegmentation, for which regions are extracted using a large tolerance margin in the homogeneity; the boundaries of the regions are the most prominent edges in the image;
- oversegmentation, which results in a large number of regions, recommended when the segmentation result must be used for object recognition purposes;
- quantization, in which the image is quantized using all the important colors in the image.

According to the kind of segmentation chosen, three parameters are automatically set:

- the radius of the search window used in the mean-shift algorithm (which is proportional to the square root of the trace of the covariance matrix);
- the minimum number of pixels required to form a region in the L*u*v* space (N_{min});
- the minimum number of pixels required to form a region in the image domain (N_{con}).

The algorithm proceeds as follows:

1. M points are randomly chosen in the image. Their colors are chosen as candidates to be the center of a search window, and the colors of their 3x3 neighbors are also mapped into the feature space. The search window containing the highest density is chosen.
2. The mean-shift algorithm is used to make the window converge to a mode of the color distribution.

3. The center of the search window is chosen as cluster center in the feature space. All the points within the search window are deleted from both the image space and the feature space, as well as their 8-connected neighbors in image space (this eliminating aliasing effects).
4. Steps 1 to 3 are repeated until the search window contains less than N_{min} points.
5. All cluster centers that do not generate at least a connected component of size N_{min} in the image domain are deleted.
6. Points associated to the remaining cluster centers are assigned to their color, regardless of their position in the image. Then, the radius of the search window is multiplied by $\sqrt[3]{2}$, and the points newly included in the window are associated to the center's color if one of their 8-neighbors already is.
7. Connected components smaller than N_{con} are deleted. Their pixels are associated to the color the most represented in their 3x3 neighborhood. If there's a tie between colors, the most similar is chosen.

The regions extracted by undersegmentation are those that are very similar to what we could expect from a user painting a sketch with a simple paint tool, as it can be seen in Plate 58.

Advantages of this algorithm:

- the number of regions is unknown a priori;
- only one parameter needs to be set (kind of segmentation);
- combines spatial and color information;
- borders of regions correspond to most prominent edges in the image;
- regions are built according to an homogeneity criterion;
- fast processing.

Disadvantages:

- unable to detect small changes in color (as in the example shown in Plate 59);
- the selection of the search windows is a partially random process, which may mean that processing an image twice would give two different results. In the implementation of the algorithm that we used, though, the random number generator is reset every time an image is submitted for segmentation; thus, the segmentation result of an image is always the same.

6.2.6 Summary

As we mentioned at the beginning of this section, our goal was to find a segmentation method that produces regions which are roughly similar to those the user will produce when painting a sketch. Furthermore, such method must be rather fast, since it is also used to process the sketches at querying time.

Of the tested methods, we decided to use the Comaniciu-Meer algorithm, because:

- it was developed with image segmentation as a goal (as opposed to standard clustering techniques like ISODATA and the Batchelor-Wilkins algorithms) and generates regions corresponding to a homogeneity criterion;
- the produced regions depend on color and spatial information (unlike those produced bythe first two methods presented in this section, which only depend on color);
- it does not require to know how many regions must be extracted;
- processing time is low (compared to ISODATA, for example).

6.3 Image Description

The Comaniciu-Meer segmentation algorithm, which we chose to perform an initial segmentation of the images, produces a partition of the original image into a set of regions, which must be represented by a set of features. In our case, we were guided in our decisions by the information provided in the user's sketches, as it would be useless to represent regions extracted from database images using features that are not present in the users' sketches. For this reason, we represent regions according to size, position, color and shape information.

The reader should also remember that, as we mentioned in Section 5.2, regions are in general non-connected.

6.3.1 Notation

Henceforth, we will speak of a sketch S and of a database image I. The sketch S contains a single segmentation result $S = \{S_1\}$, while the database image I contains n_I segmentation results $I = \{I_1, \ldots, I_{n_I}\}$. In turn, a segmentation result contains a set of regions. For example, $I_i = \{I_i^1, \ldots, I_i^{l_i}\}$, $S_1 = \{S_1^1, \ldots, S_1^{l_S}\}$.

6.3.2 Region Features

POSITION

Three main techniques can be used: contours, bounding boxes and centroids.

Contours allow to know exactly where the region ends, but comparison is cumbersome: in order to compare the position (and not the shape) of two regions using their contours, the contour lengths will need to be normalized, then the position of the contour points will have to be compared. The results of these comparisons will then need to be combined in a meaningful way, returning a value that describes their spatial relationship, which is a non-trivial task.

FIGURE 15. It is not possible to deduce the spatial relationship between two regions by just looking at the position of their center of gravity.

The bounding box B_a of a region $a = \{(x_1^a, y_1^a), \ldots, (x_n^a, y_n^a)\}$ is a rectangular region delimited by the points $(\underline{x}^a, \underline{y}^a), (\underline{X}^a, \underline{Y}^a)$, such that

$$\underline{x}^a = \min_i x_i^a$$

$$\underline{y}^a = \min_i y_i^a$$

$$\underline{X}^a = \max_i x_i^a$$

$$\underline{Y}^a = \max_i y_i^a$$

Bounding boxes (Plate 60) describe the position of the region less precisely than contours (especially if the main axe of the region is diagonal), but are useful for limiting the space within which the region is located. They show some weaknesses, though, if the regions are not compact.

The spatial center of gravity $(\bar{x}, \bar{y}) = \left(\frac{1}{n} \cdot \sum_i x_i^a, \frac{1}{n} \cdot \sum_i y_i^a\right)$ of region a (also called centroid), allows a more precise definition of the position of the region, but does not say anything about the distribution of the region points: regions with very different shapes, like the rectangles in Figure 15, can have the same center of gravity.

Despite this, we decided to represent position with the region's center of gravity. In fact, this description is simple, and guarantees that corresponding regions will be considered similar. It will, though, also produce false positives. These should be eliminated by the other features participating in the computation of region dissimilarity, though.

Points in the image are assigned a normalized position within the [0,1]x[0,1] square. In general, images are not perfectly square, meaning that we have two possibilities to fit an image in the [0,1]x[0,1] square: scaling the image so that it fills the square, or placing it somewhere within the square.

Scaling the image has the following shortcomings: shapes are distorted, and all information about the height/width ratio of the image is lost. Since shapes are a factor in the way we compute the dissimilarity between regions in the image, we decided not to scale the image. Thus, we have to find a way to place the image within the [0,1]x[0,1] square, which is done by centering the image in the square. To do this, we assume that the largest dimension will span the whole [0,1] interval, and that the other dimension will span the $[0 + \delta, 1 - \delta]$ interval, where δ is the difference between the two dimensions, divided by 2. For example, if width w is larger than height h, the image points will cover the $[0, 1] \times [0 + (w - h)/2, 1 - (w - h)/2]$ rectangle, thus respecting the width/height ratio of the image. An example can be found in Plate 61.

COLOR

The first choice to be made is the color space. Color spaces were briefly introduced in Section 6.1. The color space must be chosen according to what must be done with color information. In our case, what we want is to compare colors between the sketch and the image, so that the computed dissimilarity corresponds approximately to the difference perceived by the user. For this reason, we decided to use a perceptual color space like L*u*v*.

Another factor to be taken into account is how the color of a region should be represented. While color of regions in the sketch will in general be uniform, this is not the case for almost all of the images in the database. Essentially, we have two possibilities: representing the regions' color with their average color, or using color histograms.

Using the average color has the obvious advantage that comparison consists in computing the difference between two colors, which is computationally fast. On the other hand, if a region is composed of pixels of different colors, computing the average will generate a color which is less saturated than what the user sees (Plate 62). This effect can be limited by not taking into account the pixels on/near the border of the region, where most color transitions occur.

When using color histograms, it is possible to preserve the color distribution of the region. On the other hand, comparing color histogram is computationally more expensive, and results can suffer from a bad quantization of the color space. In our work, both average colors and color histograms are used.

The color histogram consists of 125 bins, whose bin centers (also called representative colors, Plate 63) were selected as follows:

- 125 equally-spaced RGB colors were selected and transformed into L*u*v values;
- additional 32768 equally spaced RGB colors were then selected and converted to the L*u*v color space;
- the k-means algorithm was then used with the first 125 points as initial bin centers, and the additional 32768 points as data points; the algorithm was stopped when the bins were stable (total movement of bin centers not above a given threshold).

The color histogram of a region is formed by assigning each pixel of the region to the bin whose representative color is the closest in the L*u*v* space. An example of color histogram can be seen in Plate 64.

Size

In our work, the size of a region is the percentage of image pixels it contains. In sketches, the ratio is computed according to the number of pixels the sketch could contain if its canvas was completely used. This way, the size feature of sketch regions and database image regions are directly comparable.

Shape

There are several approaches presented in the literature to describe the shape of a two-dimensional object. These go from approximating the contour with a function (splines, polygons, ...), to computing a histogram of the contour directions, and using spatial moments.

While choosing a description for shape, we had to keep in mind that, as we mentioned in Section 5.2, our regions are not connected. This means, for example, that it is not possible to approximate their contour with a single spline or polygon. If we were to represent the shape as the splines of the different connected parts composing the region, computing the dissimilarity between two regions would mean comparing all the combinations of splines, which would defeat the purpose of having non-connected regions. The same applies to any other method that approximates the region's contour.

Using the histogram of the region's contour directions, on the other hand, may seem a better choice. We observed, though, that regions painted by users have contours that are extremely "flat" compared to those extracted from database images (they lack of detail). Furthermore, if a good visual correspondence can be found between painted regions and regions extracted from database images, it is not because of the contour, but because of "mass" and general appearance of the regions.

For this reason we decided to use moments to represent the regions' shapes (Plate 65). A regions is represented by an ellipse of unit surface, whose main axes have the same direction of the eigenvectors of the region's covariance matrix and whose axes respect the ratio between the corresponding eigenvalues. Such a description gives an idea of the main direction and of the elongation of the region, but fails to convey any other characteristic (particular shape of the contour, for example) (Plate 66). Furthermore, very scattered regions generate a shape description that does not really correspond to the visual experience of the region (Plate 67).

Other features

Among the features that were not implemented in the final prototype we have texture descriptors, spatial relationships between regions, a unified shape, position and size feature and a compactness feature. These will now be briefly discussed.

Texture features were not implemented because producing sketches with texture makes the sketching process significantly more complex, slowing it down drastically. Furthermore, experience showed us that the users would not use textures even if they were told they could. Furthermore, it is not always easy to reproduce natural textures with artificial patterns, and there is no guarantee that their descriptions will actually be similar. In order to avoid problems connected with synthetic textures, we would have to provide the user with a palette of natural textures, allowing him/her to combine them to obtain the wanted texture. Composition of such a palette is without a doubt a very hard task, especially if its size must be contained within manageable proportions. Furthermore, composition of textures is not a natural task for a human being.

Spatial relationships between regions are quite useful to filter out not relevant images just by looking at the regions' position. Such relationships, though, become difficult to express with non-convex regions, and become even more complex when regions are non-connected (as in our case). The particular nature of non-connected regions requires a whole new set of spatial relationships, some of which are extremely difficult to define. For this reason, spatial relationships between regions were left out of the image description.

The spatial scatter matrix was also taken into account. Let $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ be the spatial position of the points of a region a, and let $(\bar{x}, \bar{y})$ be the center of gravity of a. The geometric scatter matrix $U = (u_{ij})$, where

$$u_{11} = \sum_{i=1}^{n} (x_i - \bar{x}) \cdot (x_i - \bar{x})$$

$$u_{22} = \sum_{i=1}^{n} (y_i - \bar{y}) \cdot (y_i - \bar{y}) \qquad \text{(EQ 4)}$$

$$u_{12} = u_{12} = \sum_{i=1}^{n} (x_i - \bar{x}) \cdot (y_i - \bar{y})$$

can be used to summarize information about the shape, the size and the position of the region. Hence, U could be used as a region feature. This region feature is already used during the region merging phase of the feature extraction process (which will be explained in detail in Section 6.3.4), where regions are compared using Mahalanobis' generalized distance. It would appear natural, then, to use this distance also to compute the distances between a region of the sketch and a region of a database image. It must be noted, though, that Mahalanobis' generalized distance does not measure how similar two distributions are, but how close to a normal distribution the distribution obtained by merging the two data sets is. For this reason, regions whose shapes are completely different may be very close according to this distance. An example of this is shown in Plate 68, where the "sky" region of a sketch is considered to be very similar to the "clouds" region of the target image, because by merging the two regions (if they were part of the same image) a compact region would be obtained. Since the regions we compare are issued from separate images, this kind of distance should not be used.

Knowing how compact a region is can also help in differentiating regions that are similar according to other features. For this reason, we made some experiments with a region compactness feature that was computed as follows:

- for each point (x, y) of a region a, the number of 4-neighbors belonging to the same region of the current point $N(x, y, a)$ is computed, as well as the number of 4-neighbors of (x, y) that do actually exist $N(x, y)$ (this value can be lower than 4 if (x, y) is located on the border of the image);
- the compactness of the region a, $C(a)$, is:

$$C(a) = \frac{\sum_{(x,y) \in a} N(x, y, a)}{\sum_{(x,y) \in a} N(x, y)} \qquad \text{(EQ 5)}$$

The value obtained with Equation 5 summarizes how compact the region is. In particular, a value of 0 will be obtained if no point of region a has another point of region a as a neighbor. On the other hand, the only region for which compactness will reach 1 is a region containing every single point in the image.

We decided not to put this feature in the final set of region features for the following reasons:

- color patches painted by users tend to be significantly more compact (both visually and according to our compactness measure) than the regions they are supposed to represent; thus, a comparison between regions extracted from the sketch and regions extracted from database images is not as useful as it could be thought;
- the compactness value of a small region is lower than that of a larger region of the same shape, because of the different growing rate of perimeter and surface; as an example, in Figure 16 the compactness values of squares of growing side are shown. As it can be seen, smaller squares are considered as "less compact" than large ones;
- comparing compactness values is not always meaningful, partially because of the above remark.

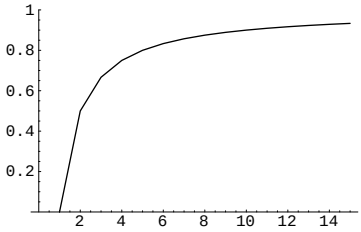

FIGURE 16. The compactness values for squares of growing size.

6.3.3 Obtaining Comparable Descriptions

Segmentation results for sketches and database images are in general extremely different (except if images in the database were produced in a similar way as the sketches). In general, the number of regions in the database images' segmentation results will be 10-20 times higher than the number of regions extracted from the corresponding sketch. Since comparison between regions must be performed on a one-to-one basis, lest having the number of operations to be performed grow exponentially, it is necessary to reduce the number of regions in the database images' description to a more manageable amount.

In our work, regions are merged iteratively by pairs, according to the technique presented in the following section.

6.3.4 Region Merging Techniques

The initial segmentation result of a digital image contains in general a large number of regions (between 80 and 400, see Plate 70). These regions are in general smaller than the ones extracted from sketches (Plate 71). In order to be able to compare sketch regions and digital image regions on a one-on-one basis (as it is done by our dissimilarity measure), the description of digital images must contain regions comparable to those that are found in the sketches. In order to obtain this, regions must be merged until a partition of the image that is similar to what can be expected from a sketch is obtained. The fact that two users have different drawing abilities, as well as different interests, when sketching the same image should also be taken into account.

The technique we use consists in iteratively merging the two most similar regions, until there are no regions similar enough to be merged. Regions that are not merged are not modified. A similar technique is used in PICASSO [30], in which new segmentation results are formed by simultaneously merging all similar regions.

Let $I_1 = \{I_1^1, \ldots, I_1^{n_I}\}$ be the initial segmentation result, and $d_I(I_1^i, I_1^j)$ a distance between regions $I_1^i, I_1^j \in I_1$. The new segmentation result, I_2, is produced by merging I_1^i, I_1^j such that $d_I(I_1^i, I_1^j) = \min_{k,l}(d_I(I_1^k, I_1^l)), k < l$.

The result is $I_2 = \{I_2^1, \ldots, I_2^{n_I - 1}\}$, where

$$I_2^k = I_2^k, k < i \wedge i < k < j$$
$$I_2^k = I_2^{k+1}, k > j$$
$$I_2^i = I_2^i \cup I_2^j$$

The choice of the region distance function d_I is essential for obtaining a good solution, and will be the subject of the following paragraph.

Choosing a Region Distance

In order to merge regions in the image, a distance between regions must be defined. This region should allow the following:

- the distance between visually similar and spatially close regions should be low;
- the distance between regions of similar color should be low;
- the spatial distance between the regions must be taken into account;
- merging should, as far as possible, result in compact regions.

Speaking of the last requirement, it can be argued whether regions should be connected (like in Blobworld [17], for example) or if merging should be allowed to form non-connected regions. Non-connected regions can be a good choice for representing some kinds of entities (like a cluster of flowers in a field, such as the one that can be seen in Plate 67) that would be impractical to be considered as separate objects. In fact, comparing all the flowers in the database image to all the dots painted by the user in the sketch would lead to a huge number of possibilities. For this reason, it was decided to allow the creation of non-connected regions, all by trying to keep regions as compact as possible during the region merging phase.

Using non-connected regions can lead to lack of precision in the description of the regions for the position and shape descriptors. In fact, the centroid of a non-connected region can lie outside the region (although this is also true for connected but non-convex regions, as the example in Plate 69 clearly shows), furthermore, the shape descriptor described previously no longer does represent accurately the distribution of region points in the image, especially if the region, like the one in Plate 67, is not compact.

In the following paragraphs, some solutions that were tested will be discussed.

Merging According to Average Color and Centroid Position

This technique is based on the following definition of the region distance d_I:

$d_I(a, b) = \alpha \cdot d_C(a, b) + (1 - \alpha) \cdot d_S(a, b)$

where d_C is the euclidean distance between the regions' average colors in the L*u*v* space (normalized between 0 and 1), d_S is the euclidean distance between the regions' centroids and α is a value between 0 and 1.

While this method may appear appropriate, it presents a number of disadvantages, the most important of which is the fact that the distance between the regions' centroids does not accurately represent the perceived distance between the regions, as it can be seen in Figure 17. As a result, the merging order may not always correspond to the one that would be obtained by asking a human operator to iteratively merge the regions which are the closest.

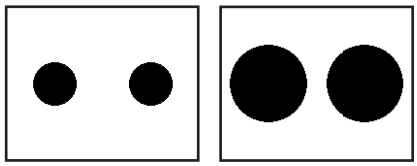

FIGURE 17. The distance between the regions' centroids is the same, but the regions in the image on the right appear closer than the ones in the image on the left.

Merging Using Mahalanobis' Generalized Distance

Another possible choice for selecting the pair of regions to be merged is statistical analysis. Let a, b be two regions, whose points are defined in a five-dimensional space composed by the three color coordinates and the two geometric coordinates:

$$a = \{(a_{11}, \ldots, a_{15}), \ldots, (a_{n_a 1}, \ldots, a_{n_a 5})\}$$
$$b = \{(b_{11}, \ldots, b_{15}), \ldots, (b_{n_a 1}, \ldots, b_{n_a 5})\}$$

We can define $d_I(a, b)$ as Mahalanobis' generalized distance between the two regions, and select the two regions in the image for which d_I is minimum, since Mahalanobis' generalized distance reaches its minimum if a perfect normal distribution can be obtained by merging the two.

Mahalanobis' generalized distance $D(a, b)$ is computed as follows:

$$D(a, b) = \sqrt{\left(\frac{1}{R(a, b)} - 1\right) \cdot \frac{(n_a + n_b) \cdot (n_a + n_b - 2)}{n_a \cdot n_b}} \quad \textbf{(EQ 6)}$$

where

$$R(a, b) = \frac{\left|U^a + U^b\right|}{\left|U^{ab}\right|} \quad \textbf{(EQ 7)}$$

and U^a, U^b, U^{ab} are the scatter matrices of a, b, and of the region obtained by merging a and b, respectively.

We could expect to obtain very good results from this distance, but unfortunately this is not the case, as Mahalanobis' generalized distance works only if the space in which the points are defined is isotropic (i.e., if the euclidean distance between points in the space corresponds to the perceived distance between the points). This is not true in this case, as color and spatial information are combined.

Merging According to Color Distance and Mahalanobis' Generalized Distance in the Spatial Domain

In order to avoid the aforementioned problems with Mahalanobis' generalized distance, we decided to separate color dissimilarity from shape-position-size dissimilarity, computing two separate values and combining them in order to obtain one single region dissimilarity score.

The color distance d_C between two regions a and b is the euclidean distance between their average colors in the L*u*v* color space. Computing the distance between the color histograms of the two regions would also have been possible, but experiences showed that the merging order remains essentially the same, at the expense of a large increase in computing time.

Difference in size, position and shape between the two regions, on the other hand, is computed using Mahalanobis' generalized distance d_M on the geometric scatter matrices of a and b. In this case, since the geometric space is isotropic, using this kind of distance does not provoke unexpected effects: small values correspond to regions that, if merged, would form a compact region (one of our goals in the region merging phase).

Combining the values computed with d_M and d_C is not straightforward, since d_C is bound within 0 and 1, while d_M is not bound upwards.

One solution to this problem would be of replacing $d_M(a, b)$ with $P_{d_M}(d_M(a, b))$, where $P_{d_M}(x)$ is the probability that $d_M(a, b) \leq x$. Since $P_{d_M}(x)$ is bound between 0 and 1, we could then easily combine it with d_C. In order to compute $P_{d_M}(x)$, we computed the distribution of the values of d_M over the regions of some of the images of our database:

- 50 images were randomly selected from a database composed of approximately 4000 images;
- the selected images were segmented using the Comaniciu-Meer algorithm described in Section 6.2;
- For each image, Mahalanobis' generalized distance between the segmented regions was computed, giving $N = 679131$ values: $d_M^{(1)}, \ldots, d_M^{(N)}$.

$P_{d_M}(x)$ can then be computed as follows:

$$P_{d_M}(x) = \frac{z}{N} \Leftrightarrow P_{d_M}(x) \leq d_M^{(z)} \wedge P_{d_M}(x) > d_M^{(z+1)} \quad \textbf{(EQ 8)}$$

The computation of $P_{d_M}(x)$ with this definition would certainly take a long time ($\log(N)$ in the worst case). An approximation can be made by carefully selecting a lower number of values (for example $N' = 1000$) and defining $P'_{d_M}(x)$ with a formula similar to Equation 8, such that $\left|P'_{d_M}(x) - P_{d_M}(x)\right| < \varepsilon$.

Another alternative is to approximate $P_{d_M}(x)$ with a function. We found that it is possible to approximate $P_{d_M}(x)$ with

$$\tilde{P}_{d_M}(x) = \tanh\left(\frac{x}{t}\right) \quad \textbf{(EQ 9)}$$

and that the best approximation is obtained when $t = 58.61$. Graphs of $P_{d_M}(x)$ and $\tilde{P}_{d_M}(x)$ can be seen in Figure 18.

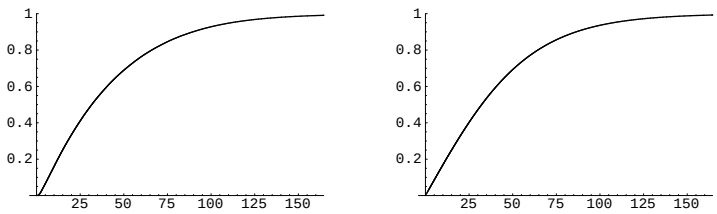

FIGURE 18. Graphs of $P_{d_M}(x)$ (left) and $\tilde{P}_{d_M}(x)$ with $t = 58.61$ (right).

$\tilde{P}_{d_M}(x)$ being faster to compute than $P_{d_M}(x)$, we decided to use it in d_I:

$$d_I(a, b) = \alpha \cdot \tilde{P}_{d_M}(d_M(a, b)) + (1 - \alpha) \cdot d_C(a, b) \quad \textbf{(EQ 10)}$$

By setting the α weight the importance of the similarity in the color domain and in the spatial domain can be modified. Experimentally, it was found that values around 0.1 give satisfying results. Higher values of α tend to provoke the merging of regions of different color but compatible shape.

Since d_M and d_C are distances, and *tanh* is a strictly growing function, d_I is also a distance. Let a and b be any two regions in an image I. To demonstrate that d_I is a distance we have to prove that:

$$d_I(a, a) = 0$$

$$d_I(a, b) = d_I(b, a)$$

$$d_I(a, b) \leq d_I(a, c) + d_I(c, b), \text{ for any region c in I}$$

The first two properties being obvious, we will only demonstrate the triangular inequality. Let us pretend that there exist three regions a, b, c such that

$$d_I(a, b) > d_I(a, c) + d_I(c, b)$$

by expanding and moving some terms, we obtain that

$$\alpha \cdot (d_C(a, b) - d_C(a, c) - d_C(c, b)) > (1 - \alpha) \cdot (\tanh(d_M(c, b)/t) - \tanh(d_M(a, b)/t) + \tanh(d_M(a, c)/t))$$

Since d_C is a distance, we know that $d_I(a, b) - (d_I(a, c) + d_I(c, b)) \leq 0$, which means that we can remove the terms on the left hand side, leaving us with

$$0 > (1 - \alpha) \cdot (\tanh(d_M(c, b)/t) - \tanh(d_M(a, b)/t) + \tanh(d_M(a, c)/t))$$

Knowing that tanh is a strictly growing function, the inequality reduces itself to

$$d_M(a, b) > d_M(a, c) + d_M(c, b)$$

which is not possible, since d_M is a distance.

This solution has the advantage of taking into account color and spatial information separately, all while being more precise than the first method presented in this section. This is the reason why we chose it.

STOPPING THE MERGING PROCESS

Ideally, the merging process should be stopped before unrelated regions merge. In general, this will mean that the process must end before regions belonging to different entities in the image merge.

We do this by setting a threshold ξ for the minimum distance between two regions. The merging phase is stopped when $\min_{i,j} d_I(I_1^i, I_1^j) > \xi$. ξ was empirically set to 0.1 for digital images (Plate 72) and to 0.02 for sketches. Other systems, like PICASSO [30], continue the merging process until there is only one region left. This may lead to false matches because excessive merging can result in regions that are similar despite the fact that the regions that formed them are not.

6.3.5 Multiple Segmentation Results

As we mentioned in Section 5.2, users will produce extremely different sketches even when they are looking for the same target image, as shown in Plate 51. Since we want our dissimilarity measure to be able to return low dissimilarity scores for sketches that, despite being similar to the target image, are quite different, we store several segmentation results for each database image, hoping that at least one of them will contain regions similar to those painted in the sketch.

In the current implementation of the feature extraction process, for each database image the last 10 segmentation results obtained before the threshold ξ is met are kept. Sketches, on the other hand, are represented by a single segmentation result.

When a sketch is compared to a database image, each segmentation result of the database image is compared to the one of the sketch. Despite this fact, storing multiple segmentation results does not mean a huge increase in the number of comparisons between regions needed in order to compute the dissimilarity between sketch and image, since when going from segmentation result I_i to the following one I_{i+1} two regions disappear and generate a new one. Hence, most of the region dissimilarities computed for in I_i can be reused.

Another way of choosing which segmentation results to store (like, for example, keeping every fifth segmentation result) would have a large impact on the number of region dissimilarity measures to be computed. It could, however, also be beneficial for the computation of the dissimilarity between sketch and database image, since it would probably allow for a greater flexibility.

6.4 Conclusion

In this chapter, the image description that allows us to compute the dissimilarity between a sketch S and a database image I was introduced. The image description is based on segmentation results obtained by first segmenting the image with the Comaniciu-Meer algorithm, then iteratively merging pairs of regions until the minimum distance between two regions of the image exceeds a threshold. In order to allow comparison between database images and sketches containing various levels of detail, multiple segmentation results are stored.

Segmentation results contain regions, which are characterized by 4 features: color histogram, size, position of the center of gravity and shape.

In the following chapter, we will explain how this image description is used in order to compute the dissimilarity between a sketch S and a database image I.

6.5 Color Plates

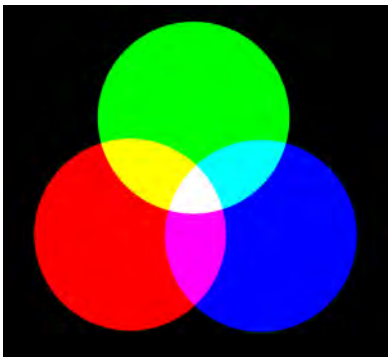

PLATE 52. **In the additive color scheme, white is obtained by mixing the three basic colors (red, green and blue) at full intensity.**

PLATE 53. **In the subtractive color scheme, darker colors are obtained by mixing colors**

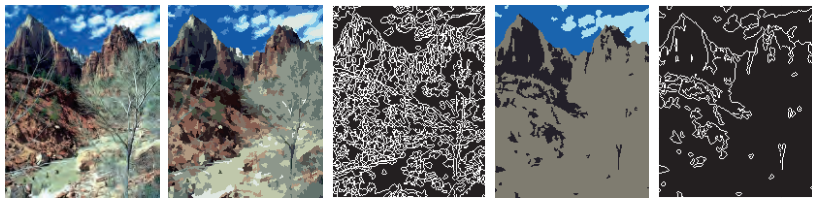

PLATE 54. **From left to right, the original image, an oversegmented version with its contour image and an undersegmented version with its contour image.**

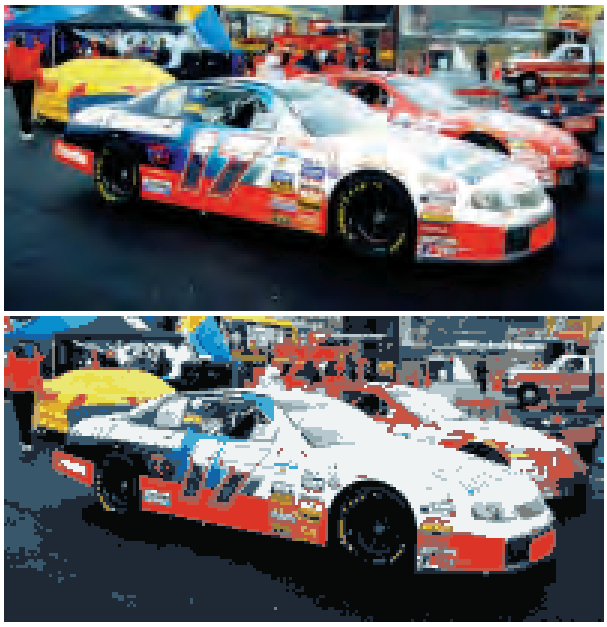

PLATE 55. When segmenting the above image, 19 histogram peaks were found, giving the segmentation result shown. Because of noise and quantization effects, if we were to form connected regions according to the assignment of points to the histogram bins we would obtain an extremely large number of regions.

PLATE 56. An image segmented to exactly 9 clusters with two different visiting orders, along with the difference between the two results (lighter pixels mean bigger differences).

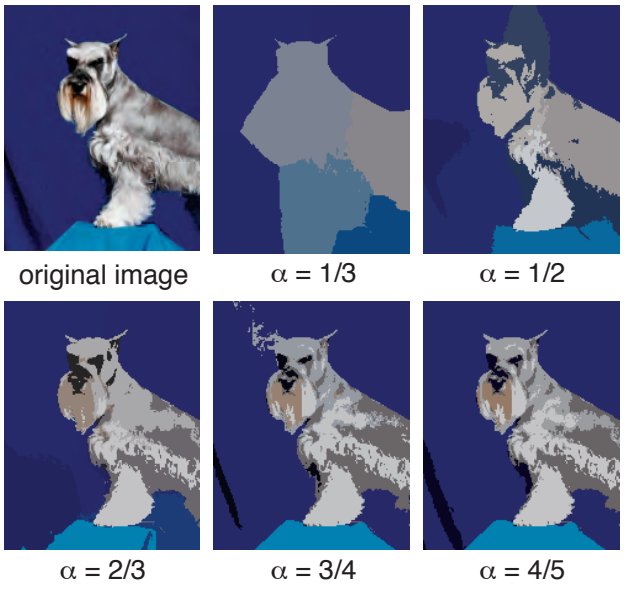

PLATE 57. Results obtained with the Batchelor-Wilkins algorithm by computing color distances with the Euclidean distance within the HSV color space and setting threshold

$\tau = 0.4$. **As it can be seen, low values of** α **result in regions that do not respect the changes in color, while high values of** α **generate very scattered regions.**

PLATE 58. The Comaniciu-Meer algorithm produces regions that look strikingly similar to the ones we expect to see in the sketches.

PLATE 59. The Comaniciu-Meer algorithm shows some weaknesses when dealing with small changes in color. In this case, the white part of the motorbike was merged with the background.

PLATE 60. Despite the fact that the majority of the pixels belonging to the blue region are located in the upper right area of the image, the region's bounding box covers the whole image.

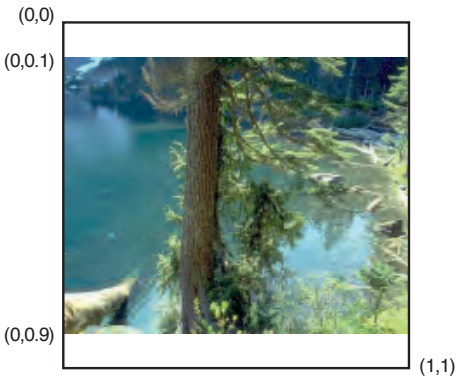

PLATE 61. Images that are not square are centered in a square, whose largest dimension spans the whole [0,1] interval, while the other dimension spans a smaller interval centered around 0.5. In this case, the image (whose dimensions are 200x160) spans the [0,1] interval horizontally and the [0.1,0.9] one vertically.

PLATE 62. Although the average color of a region can in general give a very good idea of the overall color of the region, all information about other colors present in the region will be lost.

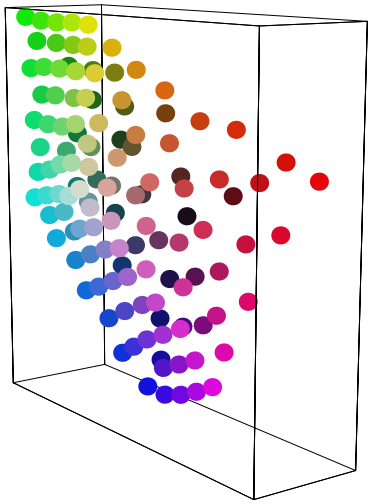

PLATE 63. The 125 bin centers in the L*u*v* color space

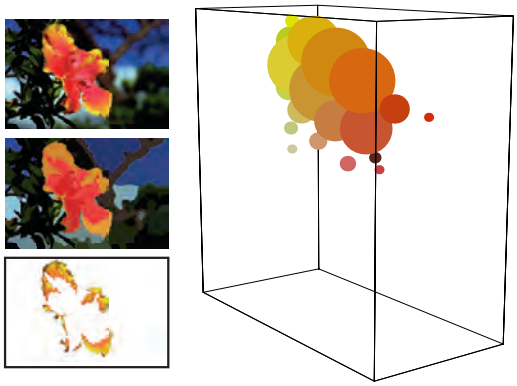

PLATE 64. On the left, from top to bottom: the original image, a segmentation result and a single region of the segmentation result. On the right: a representation of the L*u*v* histogram of this region.

PLATE 65. In our work, the shape of a region is represented by an ellipse.

PLATE 66. The representation of the shape as an ellipse fails to represent the particularity of the region's contour (as it is the case for the background in this image), all while being able to convey information about the general shape of the region (as in the case of the region representing the dog).

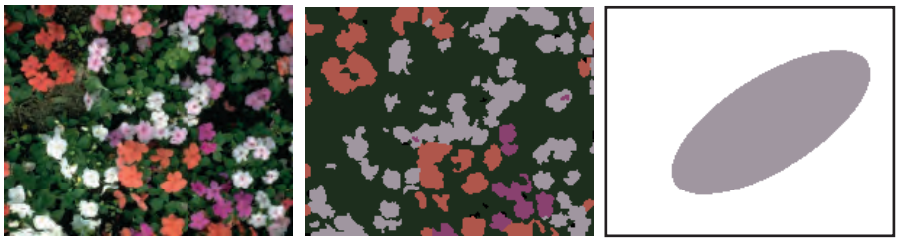

PLATE 67. The ellipse associated to very scattered regions does not convey any information about how scattered the region is.

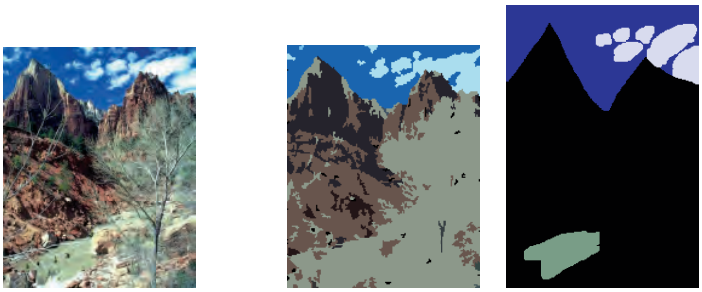

PLATE 68. Mahalanobis distance is not adapted to computing the dissimilarity between shapes: in this example, the distance between the blue region in the sketch and the light blue region in the segmentation result of the database image is very low, because merging the two regions would result in a compact region.

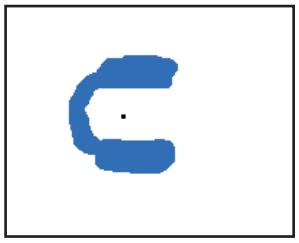

PLATE 69. The center of gravity of a region can lie outside the region even if, like in the above image, the region is connected.

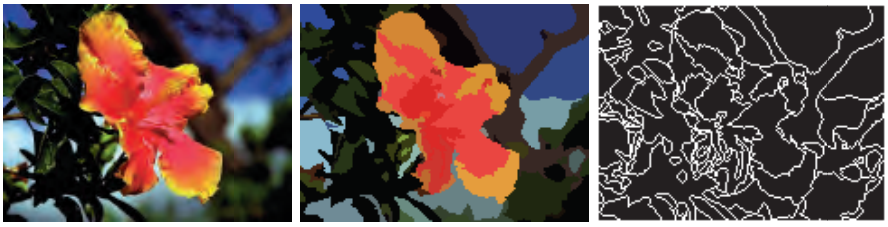

PLATE 70. **The initial segmentation result of the database images consists in general of a large number of regions, as in this case (98 regions).**

PLATE 71. **The initial segmentation of database images returns in general a larger number of regions than the segmentation of the sketches meant to represent them.**

PLATE 72. **By stopping the merging process when the minimum distance between two regions exceeds a threshold ξ, we can avoid creating regions composed of unrelated objects.**

CHAPTER 7

Comparing Sketches and Images

7.1 Goal of the Comparison

The obvious goal of a dissimilarity measure between a sketch S and a digital image I is to produce a value $d(S, I)$ that significantly represents the dissimilarity between S and I as perceived by the user.

Reaching such a goal is utopic, though, since a correct judgement of similarity requires a complete codification of the context in which the user sees the pictures and that "context is a set of social and cultural conventions that depend critically on our role as the participants in a network of interactions", as pointed out by Santini and Jain in [121]. As a consequence of this observation, the computation of dissimilarity between a sketch and a database image is, in our work, limited to the visual aspect, putting aside any semantic information (Plate 73) .

Another important topic is how dissimilarity is to be computed. As images are represented by numerical features, and humans tend to associate "amounts of similarity" to pairs of images, we could start thinking that similarity may be approximated by a distance defined on a multidimensional space in which images are points. However, it is not clear whether this would be a good way of simulating human judgement of similarity, as some works (among which we can list Tversky's [139], which inspired Santini and Jain's works on CBIR [122]) tend to show that the human estimation of similarity does not behave like a distance, but is more closely related to set operations in which set elements are very simple image features.

In short, we are aiming at defining a dissimilarity measure $d(S, I)$ that, being given the feature descriptions of the sketch S and of the image I presented in Chapter 6, returns a numerical value effectively representing their visual dissimilarity, as it would be perceived by the user. The definition of such a measure must take into account the following factors:

- Sketches produced by different users differ under many aspects, as we mentioned in Section 6.3.5; for this reason, several segmentation results for each database image are stored in the corresponding descriptor (in the hope that at least one of them will be similar to the submitted sketch) and $d(S, I)$ corresponds to the minimum dissimilarity between the segmentation result of S , S_1 , and those of I , $\{I_1, \ldots, I_I\}$.
- The sketch contains color patches, which may not all be equally important to the user. In order to propagate this to the dissimilarity measure $d_L(S_1, I_i)$ between segmentation results S_1, I_i , weights should be attached to the regions in the sketch, either manually or automatically.

- The dissimilarity $d_R(S_1^k, I_i^j)$ between regions should tell us how visually similar a sketch region S_1^k and an image region I_i^j are. Hence, such a measure should be based on the elements that visually characterize a region: its color and its geometric appearance (i.e., characterized by position, size and shape).

In the following sections we will talk in detail about how dissimilarity between sketches and database images can be computed. In Section 7.2 some approaches that combine dissimilarities between segmentation results and / or regions into a single dissimilarity result are presented. In Section 7.3 the comparison between segmentation results is discussed, while Section 7.4 is dedicated to the study of dissimilarity between regions.

7.2 Producing a Dissimilarity Score

As we said in the previous section, we want to compute a dissimilarity score $d(S, I)$ between a sketch S and a database image I representing the visual dissimilarity between the two. The data we have available to perform this task is the image description that was presented in Chapter 6: the sketch S contains a single segmentation result $S = \{S_1\}$, while the database image I contains n_I segmentation results $I = \{I_1, \ldots, I_{n_I}\}$. In turn, a segmentation result contains a set of regions. For example, $I_i = \{I_i^1, \ldots, I_i^{l_i}\}$. In this section, we will present some solutions to solve the problem of the production of such a dissimilarity score.

For the remainder of this section, we will suppose that we can compute a meaningful dissimilarity measure $d_R(S_1^k, I_i^j)$ between any region S_1^k of the sketch and any region I_i^j of the database image, as well as a dissimilarity measure $d_L(S_1, I_i)$ between segmentation results S_1, I_i. Ways to define such dissimilarity measures are given in Section 7.3 and Section 7.4.

While computing $d_R(S_1^k, I_i^j)$ for each pair of regions (S_1^k, I_i^j) may appear as a huge amount of computations, in our case it is not: as we explained in Section 6.3.4, segmentation results are obtained by merging pairs of regions, and regions that do not merge remain unchanged from one segmentation result to the following one. This way, a large number of the region dissimilarities computed between S_1 and I_i will also be used between S_1 and I_{i+1}.

7.2.1 Region-Centric Solution

In a region-centric solution, the regions of the sketch are associated to the regions of the database image that are most similar to them, regardless of the segmentation result they belong to. In short, a region S_1^k will be associated to a region $I_{s(k)}^{r(k)}$ that satisfies Equation 11.

$$d_R(S_1^k, I_{s(k)}^{r(k)}) = \min_{i,\, j} d_R(S_1^k, I_i^j) \qquad \textbf{(EQ 11)}$$

The obtained dissimilarity values (one for each region in S_1) need to be combined to generate the dissimilarity between S and I. Among the huge number of ways to perform this task, one of the most used is the weighted sum, in which weights are associated to the regions of S, depending on their importance, as in Equation 12.

$$d(S, I) = \sum_{k=1}^{l_S} d_R(S_1^k, I_{s(k)}^{r(k)}) \cdot \omega_k \quad \textbf{(EQ 12)}$$

Among the existing CBIR systems, PICASSO [30] computes the dissimilarity between images this way. For a more detailed discussion on assigning weights to regions, see Section 7.3.1.

7.2.2 Segmentation Result-Centric Solution

Due to the unpredictability of the submitted sketches, the database image descriptor must be flexible enough to be compared to sketches of various level of detail, as we said in Section 6.3.5. For this reason we decided to store multiple consecutive segmentation results, with different detail levels, in the descriptors, hoping that at least one of the stored segmentation results will be visually similar to the submitted sketch. With such an image descriptor available, it may be useful to compare the single segmentation result of the sketch (S_1) to each segmentation result I_i of the database image, and then compute the dissimilarity between S and I as a function of the dissimilarities $d_L(S_1, I_i)$ between their segmentation results.

In particular, since the goal of storing multiple segmentation results is to have at least one of them visually similar to the sketch, it would make sense to select the minimum dissimilarity between segmentation results as the dissimilarity between sketch and database image, as in Equation 13.

$$d(S, I) = \min_i d_L(S_1, I_i) \quad \textbf{(EQ 13)}$$

This is the method chosen in this work.

7.3 Comparison of Segmentation Results

In this section, we will talk about issues related to the computation of a dissimilarity score $d_L(S_1, I_i)$ between the segmentation result S_1 of a sketch S and the segmentation result I_i of a database image I. As we explained previously, both segmentation results contain regions. If we suppose that we are able to define a way to compare regions, which we will do in Section 7.4, we can generate a set of dissimilarity scores that must be combined in order to compute $d_L(S_1, I_i)$.

When speaking about comparisons, it should be remembered that in our image descriptor no information about spatial relationships between regions are stored in the regions (nor in the segmentation results). While such information might prove useful to compute similarity between segmentation results more accurately, the computational load bound to its use would probably be too large, because of the complex nature of spatial relationships between non-connected regions. For this reason, we decided not to store this information.

Furthermore, one question arises when we talk about comparing regions: should the comparison be done between one region of S_1 and one region of I_i, between one region of S_1 and n regions of I_i, between n regions of S_1 and one region of I_i or between n regions of S_1 and m regions of I_i? We chose to compare regions on a one-on-one basis, in order to reduce as much as possible the complexity of the approach. In fact, the number of dissimilarity values to be computed in order to obtain $d_L(S_1, I_i)$ grows extremely quickly if we allow other kinds of comparisons. In a way, though, our method allows one region of S_1 to be compared to more than one region of the database image, since multiple segmentation results are stored and a region I_i^j may be the result of the merging of several regions (Plate 74).

Another factor is important when comparing a sketch to a database image: in the eyes of the user, not all the regions in the sketch will have the same importance, as some of them may be more characteristic of the sought image. Hence, a way to specify the importance of a region should be given to the user.

In the following paragraphs, let us suppose that we can compute $d_R(S_1^k, I_i^j)$, which effectively represents visual dissimilarity between regions S_1^k and I_i^j.

7.3.1 Associating Weights to Sketch Regions

When a user produces a sketch, not all the color patches are equally important, as some of them may be considered as more characteristic of the sought image. Hence, a way to specify the importance of a sketch region should be given.

This can be done by associating a weight to every region S_1^k of the sketch. The weights can be set in different ways:

- manually, by asking the user to assign saliency values to the regions in the sketch;
- automatically, by computing a saliency value and assigning it to the regions.

The first solution would probably be the best one, since the user knows which areas of the sketch are the most important. Such an approach, though, would make the sketching process significantly longer, since the user would have to click on the regions and then set the weights. Furthermore, there is the possibility that the user will select semantically meaningful regions instead of visually significant ones. Since no semantic information is contained in the image description, this would almost surely lead to poor retrieval results.

Automatic selection of region saliency, on the other hand, is a complex process. Although several works for saliency estimation in photographic images exist (among which we can list the work by Itti et al. [67], in which a saliency map is associated to an image, indicating the regions of the image that are more likely to attract the eyes of the observer) exist, it is not clear whether these techniques may be adapted to the hand-drawn incomplete sketches we are dealing with.

In our work, the ω_k are proportional to the size of the sketch regions. Hence, larger regions are considered as more important that small ones. This will not always respect the user's point of view, since sometimes the most important object in an image is small, as in Plate 75.

7.3.2 Associating Sketch Regions to the Most Similar Image Region

Similarily to the method described in Section 7.2.1, it is possible to compute $d_L(S_1, I_i)$ by associating to each region S_1^k of the sketch a region $I_i^{r(k)}$ of the database image such that Equation 14 is satisfied.

$$d_R(S_1^k, I_i^{r(k)}) = \min_j d_R(S_1^k, I_i^j) \quad \textbf{(EQ 14)}$$

The advantage of computing $d_L(S_1, I_i)$ with this technique is the very low complexity associated, as it only requires the computation of the dissimilarities between the regions of S_1 and those of I_i.

On the other hand, it is possible that several regions of S_1 are associated to the same region of I_i, resulting in a low dissimilarity score for an image which is not the target image. In order to avoid such problems, the region dissimilarity d_R should be made very selective.

Once every region of S_1 is associated to a region of I_i, $d_L(S_1, I_i)$ can be computed, for example with a weighted sum similar to the one in Equation 12:

$$d_L(S_1, I_i) = \sum_{k=1}^{l_S} d_R(S_1^k, I_i^{r(k)}) \cdot \omega_k \quad \text{(EQ 15)}$$

7.3.3 Associating Image Regions to at Most One Sketch Region

If what we are aiming at is a very close match between the sketch and a database image, we must ensure that the regions of I_i are associated to at most one region of S_1. In this case, we have to solve the following problems:

- S_1 may contain more regions than I_i; some regions of S_1 would not be associated to any region in this case;
- the solution space (the set of all possible combinations of regions of S_1 and I_i) grows exponentially with the number of regions contained in S_1 and I_i. If we assume that S_1 contains l_S regions and that I_i contains l_i regions, the number of possible combinations $\varphi(i, s)$ is obtained as in Equation 16. Some values of $\varphi(i, s)$ can be seen in Table 2.

$$\varphi(i, s) = \frac{l_i!}{\max(1, l_i - l_S)!} \cdot l_S \quad \text{(EQ 16)}$$

	$l_i = 1$	$l_i = 2$	$l_i = 3$	$l_i = 4$	$l_i = 5$	$l_i = 6$	$l_i = 7$
$l_S = 1$	1	2	3	4	5	6	7
$l_S = 2$	2	2	6	12	20	30	42
$l_S = 3$	3	6	6	24	60	120	210
$l_S = 4$	4	8	24	24	120	360	840
$l_S = 5$	5	10	30	120	120	720	2520

TABLE 2. The number of possible combinations between regions of S_1 and regions of I_i grows exponentially with the number of regions they contain.

For the sake of simplicity, let us suppose that the region dissimilarities are combined using a weighted sum, as in Equation 17 (where j is a function defined from $[1, l_S]$ to $[1, l_i]$, meaning that each region of I_i can be associated only to one region of S_1)

$$d_L(S_1, I_i) = \min_j \left(\sum_{k=1}^{l_S} d_R(S_1^k, I_i^{j(k)}) \cdot \omega_k \right) \quad \text{(EQ 17)}$$

Obviously, Equation 17 applies only if $l_S \leq l_i$. If this is not the case, some regions of S_1 will not be associated to a region of I_i. These regions, though, must participate in the computation of $d_L(S_1, I_i)$, otherwise database images containing few regions would have an advantage. A solution to this problem can be found by associating a penalty value to these regions, thus modifying Equation 17 as follows:

$$d_L(S_1, I_i) = \min_{\rho}\left(\sum_{k=1}^{q} d_R(S_1^{\sigma(k)}, I_i^{\rho(\sigma(k))}) \cdot \omega_k + \sum_{k=q+1}^{l_S} P(S_1^{\sigma(k)}) \cdot \omega_k\right) \quad \textbf{(EQ 18)}$$

where $q = \min(l_S, l_i)$, σ is a permutation of $\{1, \ldots, l_S\}$ and ρ is a function of $\{1, \ldots, l_i\}$ towards $\{1, \ldots, q\}$.

CHOOSING PENALTY VALUES

Let σ, ρ and q be the parameters of one of the solutions explored in Equation 18 for segmentation results S_1 and I_i, and suppose that $q < l_S$, hence there is at least one region of S_1 that did not find a compatible region in I_i and must be penalized, since the user's goal is to find an image containing all the sketched regions.

Examples of penalties $P(S_1^{\sigma(k)})$ are:

- $P(S_1^{\sigma(k)}) = p$, where p is a constant;
- $P(S_1^{\sigma(k)}) = f(s_{\sigma(k)})$, where $s_{\sigma(k)}$ is the size of $S_1^{\sigma(k)}$ and f is a function defined on [0,1] such that $f'(x) \geq 0$ for any $x \in [0, 1]$.

LIMITING THE SOLUTION SPACE

Another problem connected to the use of Equation 18 is that, in the worst case, all the possibilities generated by all possible values of ρ are taken into account, independently on the value returned by $d_R(S_1^{\sigma(k)}, I_i^{\rho(\sigma(k))})$. This is obviously a huge waste of computing time, as most of the regions will not have low dissimilarity and that the combination minimizing Equation 18 will certainly not contain pairs of regions with high dissimilarity.

In order to limit the solution space, we can select only the possibilities in which all pairs of regions $(S_1^{\sigma(k)}, I_i^{\rho(\sigma(k))})$ are such that $d_R(S_1^{\sigma(k)}, I_i^{\rho(\sigma(k))}) < \tau$, τ being a fixed threshold. This way, we are able to greatly reduce the solution space. Let us suppose, for example, that region dissimilarities are bound between 0 and 1, and that they are uniformly distributed within this interval. Thus, if we set $\tau = 0.2$, only 20% of the region pairs $(S_1^{\sigma(k)}, I_i^{\rho(\sigma(k))})$ will be such that $d_R(S_1^{\sigma(k)}, I_i^{\rho(\sigma(k))}) < \tau$, and we can consider that every region of S_1 is similar to at most $\lceil l_i \cdot \tau \rceil$ regions of I_i, leading to the reduction in the number of possibilities shown in Table 3.

solutions	$l_i = 5$	$l_i = 6$	$l_i = 7$	$l_i = 8$	$l_i = 9$	$l_i = 10$	$l_i = 11$
$\tau = 1$	120	720	2520	6720	15120	30240	55440
$\tau = 0.2$	1	32	32	32	32	32	729

TABLE 3. The number of possible combinations is greatly reduced by setting a threshold (S_1 contains 5 regions in this example).

When using this method, it is possible that a region S_1^k is not compatible with any region in I_i. It is also possible that the only regions compatible with S_1^k are already "taken" by other regions of S_1. Hence, such a region cannot participate to the first term of Equation 18, and its contribution must be moved to the second term. Equation 18 must be modified as follows:

$$d_L(S_1, I_i) = \min_{\rho} \sum_{k=1}^{q} d_R(S_1^{\sigma(k)}, I_i^{\rho(\sigma(k))}) \cdot \omega_k + \sum_{k=q+1}^{l_S} P(S_1^{\sigma(k)}) \cdot \omega_k \quad \textbf{(EQ 19)}$$

in which q no longer is the minimum value between l_S and l_i, but is a value freely floating between 0 and $\min(l_S, l_i)$.

Replacing Penalties with Non-Ideal Associations

The methods presented thus far have the shortcoming of not taking into account the regions of I_i, which would be a good idea, since $S_1^{\sigma(k)}$ is being given a penalty value because it is not similar to any region of I_i (meaning that there was no region $I_i^{\rho(\sigma(k))}$ such that $d_R(S_1^{\sigma(k)}, I_i^{\rho(\sigma(k))}) < \tau$), which does not mean that $S_1^{\sigma(k)}$ didn't have any visual similarity to any region in I_i. In our opinion, a distinction must be made between regions of S_1 that are penalized because they share no similarity at all with the regions of I_i and regions of S_1 that are not similar enough to regions of I_i to be considered compatible, all while sharing some kind of visual similarity with one region of I_i, like regions with the approximately the same shape but different colors.

For this reason, the penalty $P(S_1^{\sigma(k)})$ in Equation 19 may be replaced with:

$$M(S_1^{\sigma(k)}, I_i) = \left(\min_j d_R(S_1^{\sigma(k)}, I_i^j)\right) \cdot \varpi_P \quad \textbf{(EQ 20)}$$

which selects the most similar region of I_i to compute the penalty, regardless of the fact that I_i^j may already be associated to another region of S_1. By setting ϖ_P, we can decide whether these regions should be more or less penalized. Notice, though, that ϖ_P must be larger than 1 to obtain the desired effect. If we decide to use Equation 20, Equation 19 must be modified as follows:

$$d_L(S_1, I_i) = \min_{\rho} \sum_{k=1}^{q} d_R(S_1^{\sigma(k)}, I_i^{\rho(\sigma(k))}) \cdot \omega_k + \sum_{k=q+1}^{l_S} M(S_1^{\sigma(k)}, I_i) \cdot \omega_k \quad \textbf{(EQ 21)}$$

This solution is quite similar to the one we gave in Equation 15. The main difference between the two is that Equation 21 returns lower dissimilarity scores for images which have regions very similar to the ones in the sketch.

Another possibility would be to make M depend also on which regions of I_i are associated to a region of S_1 and which ones are not. In this case, we must also take into account situations in which all regions of I_i are already associated to regions of S_1. This is done by using penalties.

$$M(S_1^{\sigma(k)}, I_i, \rho) = \min\left(\min_j d_R(S_1^{\sigma(k)}, I_i^{\gamma(j)}), P(S_1^{\sigma(k)})\right) \cdot \omega_P \quad \textbf{(EQ 22)}$$

where $\gamma : [1, l_i - q] \rightarrow [1, l_i]$, $\gamma(j) \neq \rho(k)$, for each $k \in [1, q]$.

Using this solution, Equation 21 becomes:

$$d_L(S_1, I_i) = \min_{\rho} \sum_{k=1}^{q} d_R(S_1^{\sigma(k)}, I_i^{\rho(\sigma(k))}) \cdot \omega_k + \sum_{k=q+1}^{l_S} M(S_1^{\sigma(k)}, I_i, \rho) \cdot \omega_k \quad \textbf{(EQ 23)}$$

7.4 Comparing Regions

In this section, we will explain how the region features introduced in Section 6.3.2 are used in order to compute the region dissimilarity $d_R(S_1^k, I_i^j)$ between a region of the sketch and a region of a database image. Regions are the building bricks of the image description used in our work, and the computation of $d_R(S_1^k, I_i^j)$ is fundamental in order to compute dissimilarities between segmentation results and the dissimilarity between sketch and image.

Our aim is, being given the descriptions of a sketch region S_1^k and of a database image region I_i^j, to produce a dissimilarity value $d_R(S_1^k, I_i^j)$ which effectively reflects how visually similar the two regions are. This dissimilarity measure should be computed knowing that the two regions are obtained in very different ways: S_1^k is painted by a user with a simple paint tool, while I_i^j is obtained from a digital image by segmentation and merging, as explained in Chapter 6.

Regions are characterized by two factors: their color and their geometric appearance. It is only logical, then, that these two elements are the ones that are used in order to compute d_R.

7.4.1 Comparing Colors

As we mentioned in Section 6.3, in out region descriptor the color of a region is represented in two ways:

- as the average L*u*v* color of the points contained in the region;
- as a 125-bin L*u*v* color histogram representing the color distribution of the points in the region.

There are different possibilities to compare the colors of two regions, which will be introduced in the following paragraphs. Let $H_k = (H_k^1, \ldots, H_k^{125})$ and $h_j = (h_j^1, \ldots, h_j^{125})$ be the L*u*v* color histograms respectively of regions S_1^k and I_i^j. Histograms are normalized so that

$$\sum_{i=1}^{125} H_k^i = \sum_{i=1}^{125} h_j^i = 1 \quad \textbf{(EQ 24)}$$

USING AVERAGE COLOR

As we mentioned in Section 6.1, distances between colors in the L*u*v* color space are strongly related to the human perception of color difference. It would then appear natural that, when the colors of two regions S_1^k and I_i^j must be compared, the comparison is done by computing the distance between their average colors. In our case, though, this technique has some shortcomings:

- in the database images, the average colors of the regions are “polluted” by the points around the contour of the region, where color transition occur; this leads to colors that are less saturated than what the users expect, which in general are the colors they use in the sketches. An example can be seen in Plate 62; a solution to this problem would be not to consider the pixels from around the borders of the region in the computation of the average color;

- a region is in general the result of the merging of several regions (this is especially true for regions extracted from database images); keeping only the average color leads to a loss of information and generates regions which may have an average color which wasn't found in any of the original regions; this may lead to false matches because of these "artificial" colors.

Using the average color, though, has the significant advantage of being easy and computationally fast, and may be used in a filtering stage before using another method for computing color dissimilarity, like the histogram distance we will present in the following paragraph.

USING HISTOGRAMS

As we explained in Section 6.3, region color is also represented as a 125-bin L*u*v* color histogram. Color histograms are a kind of feature that has been used since the very beginning of CBIR, and are extremely useful in our case:

- they provide more information than a single color value;
- they are less sensitive to region merging than average colors, as contributions from all the merged regions cooperate instead of combining themselves, resulting in a color that may not be truly representative of the region;
- distances between corresponding sketch regions and database image regions are smaller, despite the fact that sketch regions are in general monomodal; in fact, using a histogram allows to "filter out" the less represented colors;
- they offer meaningful distance measures.

There are several techniques to compare histograms. Among them, the most used are:

- comparing the number of elements bin to bin;
- using a quadratic distance;

Comparing the number of elements bin to bin means computing a sum of differences:

$$d_C(H_k, h_j) = \frac{1}{2} \cdot \sum_{i=1}^{125} \left| H_k^i - h_j^i \right| \qquad \textbf{(EQ 25)}$$

While being extremely fast to compute, Equation 25 is also very sensitive to shifts in color, since the similarity between the bins' representative colors is not taken into account. An example can be seen in Plate 76.

Using a quadratic distance, on the other hand, is computationally more expensive and requires the computation of a matrix $M = (m_{ij})$, where m_{ij} is a value between 0 and 1 representing the similarity between the representative colors C_i, C_j of the ith and of the jth histogram bins. These values were obtained as follows: $m_{ij} = 1 - d_{Luv}(C_i, C_j)/d_{max}$, where d_{Luv} is the euclidean distance between two colors in the L*u*v* space, and d_{max} is the maximum distance between two colors in the L*u*v* space.

The distance between histograms H_k, h_j is computed as follows:

$$d_C(H_k, h_j) = (H_k - h_j)^t \cdot M \cdot (H_k - h_j) \qquad \textbf{(EQ 26)}$$

This distance keeps track of the number of elements in each bin, as well as of the dissimilarity between the bins' representative colors, and is less sensitive to color shifts than the distance presented in Equation 25, as the example in Plate 76 clearly shows.

The high complexity of this method, though, may require a filtering step, so that it is not executed for every pair of regions. For example, the distance between the average colors of the regions can be computed; if the result is lower than a threshold, then the quadratic distance between the histograms is computed, and the result is stored. This way, a significant reduction of computation time can be obtained.

7.4.2 Comparing Geometric Appearance

There is a very large number of techniques that can be used to compare the geometric appearance of two sets of points (or, in our case, regions). In this section we will study some of them, keeping in mind that they must be used to compare sketch regions with database image regions.

PATTERN MATCHING

Probably the simplest pattern recognition technique, pattern matching consists in comparing the two regions pixel by pixel and counting the number of points that are different. Let S_1^k, I_i^j be two regions and define $S_1^k(x, y)$, $I_i^j(x, y)$ such that they return 1 if point (x, y) is contained in the corresponding region and 0 otherwise. Assuming that the two regions are extracted from images of width w and height h, we can compute their regions dissimilarity:

$$d_G(S_1^k, I_i^j) = \sum_{x=1}^{w} \sum_{y=1}^{h} \left| S_1^k(x, y) - I_i^j(x, y) \right| \quad \textbf{(EQ 27)}$$

If we were to use this method to compare regions in our case, we would be faced to some problems:

- In order to be effective, the techniques requires that the sketch and the image have the same size. This is in general not true, hence some normalization would be necessary. Normalizing the sizes, though, would almost certainly modify the shape of the regions.
- The technique is very sensitive to rotation and translation, especially if the regions contain holes.
- Computing the geometric dissimilarity using Equation 27 is rather slow, as the whole image (or at least the union of the regions' bounding boxes) must be scanned.
- Furthermore, a very large amount of information is needed in order to compute this shape descriptor, since we need a list of contour points.

On the other hand, this technique does not produce false positives: low dissimilarity scores are obtained only if the regions are visually similar.

PROJECTION PROFILES

Another classic technique is using horizontal and vertical projection profiles i.e., histograms counting the number of pixels belonging to a region for each line, respectively column of the image.

Let S_1^k, I_i^j be two regions extracted from images of width w and height h. Their projection profiles are then defined as:

$$\begin{aligned} HS_1^k(y) &= \sum_{x=1}^{w} S_1^k(x, y), \; VS_1^k(x) = \sum_{y=1}^{h} S_1^k(x, y) \\ HI_1^k(y) &= \sum_{x=1}^{w} I_1^k(x, y), \; VI_1^k(x) = \sum_{y=1}^{h} I_1^k(x, y) \end{aligned} \quad \textbf{(EQ 28)}$$

Once the projection profiles are known, the difference between the two images can be computed using a histogram distance between them.

The main advantages of this method are that results are rather intuitive, and that computing is fast. On the other hand, though, the following shortcomings should be noticed:

- a large quantity of information is lost when the profiles are built;
- it requires normalization if images of different sizes are to be compared;
- it can introduce false matches, as in Figure 19.

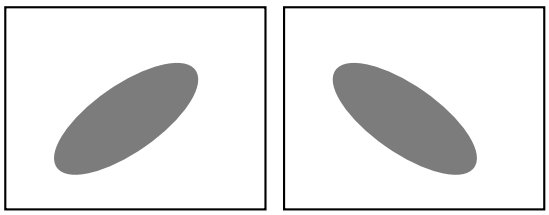

FIGURE 19. The projection profiles of the two blobs above are strictly identical, even though the blobs themselves are visually different.

Moments

Regions can also be described using central moments, which are defined in Equation 29, where $f(x, y)$ is equal to 1 if the point (x, y) belongs to the region, and 0 otherwise.

$$\mu_{pq} = \sum_{x=1}^{w} \sum_{y=1}^{h} (x - \bar{x})^p \cdot (y - \bar{y})^q \cdot f(x, y) \quad \textbf{(EQ 29)}$$

Moments of several orders can be computed, and regions can be compared by putting the obtained values in a vector and computing a distance between the vectors.

Central moments can also be normalized, as in Equation 30, so that they are independent on the size of the region. From these, invariant moments can be extracted, resulting in features that are invariant to translation, rotation and scale change [59].

$$\eta_{pq} = \mu_{pq} / \mu_{00}^{\gamma}, \text{ where } \gamma = \frac{p+q}{2} + 1, \; p + q = 2, 3, 4, \ldots \quad \textbf{(EQ 30)}$$

Although central moments give a good way of comparing region shapes, we still have to find a way to compare the regions' positions (since central moments are independent on the position). Furthermore, high-order moments are very sensitive to noise and quantization effects.

Fourier Descriptors

The Discrete Fourier Transform (DFT) of the region's contour can be used for obtaining a set of regional descriptors [57]. Let us assume that the regions we want to compare are lying in the complex plane, and let us sample a given number of points from their contour. By applying the DFT on these sequence of complex numbers (the contour points) we can obtain a regional descriptor. Normalization is then required so that the contours have the same starting point.

The difference between two regions can then be obtained by computing the distance between two vectors containing the Fourier coefficients.

Using Fourier descriptors has several advantages:

- there is no loss of information, since the DFT is reversible; even if we decide to store only the Fourier coefficients with the largest magnitude, the distortions in the reconstructed shapes are minimal (Figure 20);
- computation of the descriptor is rather fast;

- comparison between the descriptors is straightforward.

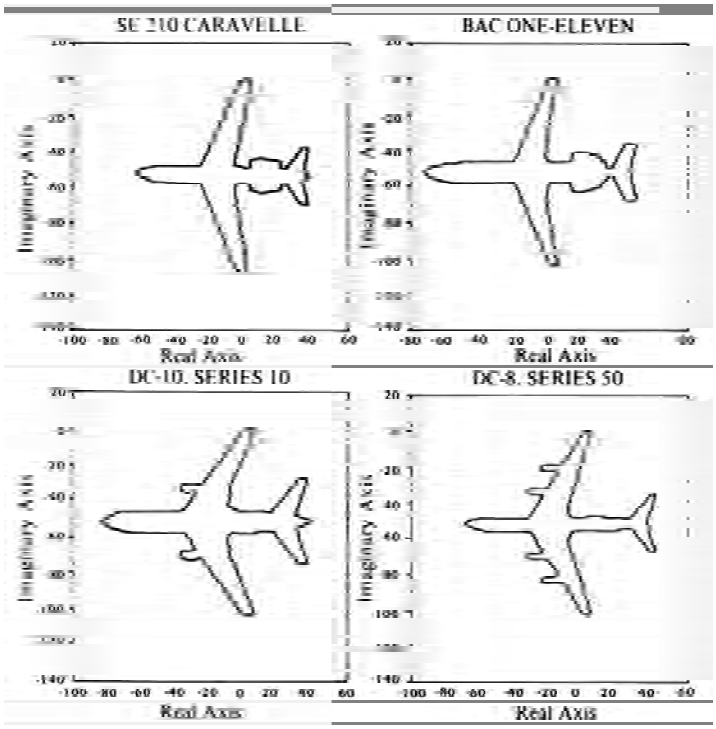

FIGURE 20. When using Fourier descriptors, it is possible to use only a fraction of the computed coefficients without significant degradation of the represented shape; in this case, taken from Hall [57], only 32 coefficients out of 512 were kept.

In our particular case, though, Fourier descriptors may not be the best choice, since:

- the method requires contours of the same length to be extracted from the regions. Since some regions are significantly smaller than others, this would result in small regions being sampled at a higher rate than large regions, leading to more precise distances for small regions and less precise distances for large regions;
- the regions that are found in our descriptors are not necessarily connected. This means that a single region would require several Fourier descriptors (one for each connected component it contains); comparing regions of this kind would require the comparison between all the Fourier descriptors.

Using the Covariance Matrix

The covariance matrix of a region can also be used to extract a description of its geometric appearance. In fact, by computing the eigenvectors and the eigenvalues of the covariance matrix we can obtain information about the general orientation and the elongation of the region. This information, along with the position of the region's center of gravity and the region's size, can be used to picture the region as an ellipse representing the distribution of its points in the image (Plate 77).

Once all regions are represented this way, the dissimilarity between two regions can be computed as the ratio between the overlapping surface of the corresponding ellipses and the surface of the largest of the two compared ellipses (Figure 21).

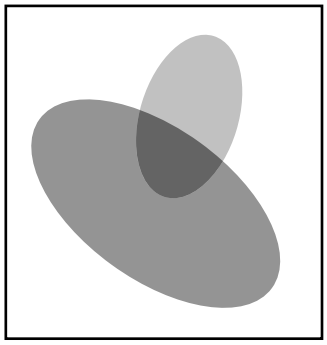

FIGURE 21. The similarity between two regions can be computed as the overlapping surface of the ellipses extracted from their covariance matrices.

Although an exact solution is too complex to compute, an approximation of the common surface may be computed with a Monte Carlo approach. In order to obtain a stable solution, though, several hundreds points should be computed, making the whole process extremely slow. Furthermore, in some cases, like the one pictured in Figure 22, this solution does not reproduce the human perception of similarity. For this reason, we decided to study an alternative way of computing geometric dissimilarity in which we split size, position and shape dissimilarity. How these can be computed will be explained in the coming sections.

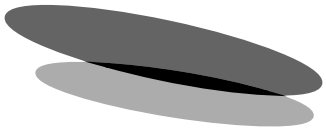

FIGURE 22. In some cases, similar ellipses can have a small overlapping surface.

7.4.3 Comparing sizes

Although the size of a region is represented by a single real value, between $1/(w \cdot h)$ and 1, where w, h are the width and height of the image in pixels, there are different ways of comparing the sizes of two regions; in our work, we experimented with some of them. In the following paragraphs, let s^k be the size of region S_1^k and s_j be the size of region I_i^j.

DIFFERENCE BETWEEN THE SIZES

The most straightforward way of comparing the size of two regions is to subtract one from the other:

$$d_S(S_1^k, I_i^j) = \left| s^k - s_j \right| \qquad \textbf{(EQ 31)}$$

All while being extremely simple to compute, Equation 31 has a disadvantage: the difference is an absolute one, and does not depend on the size of the compared regions. A simple example, like the one in Plate 78, shows that humans do not think in terms of absolute differences. Hence, this way of computing the size dissimilarity fails in reproducing the human perception of size similarity.

RATIO BETWEEN DIFFERENCE AND SUM OF THE SIZES

In order to make the dissimilarity depend on the sizes of the compared regions, and not only on the difference between their sizes, we tested the following formula:

$$d_S(S_1^k, I_i^j) = \frac{\left|s^k - s_j\right|}{s^k + s_j} \quad \text{(EQ 32)}$$

While this version of the size dissimilarity takes into account the size of the regions being compared, it still has one major drawback: dissimilarities between very small regions are huge, as it can be seen in Figure 23.

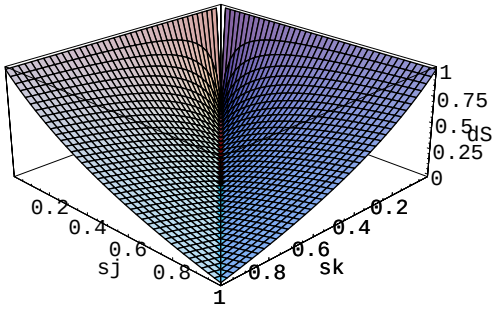

FIGURE 23. Using the ratio between the difference and the sum of the regions' sizes results is very large values for small differences between small regions.

ADJUSTED RATIO

Since Equation 32 showed a general behavior not unlike the one we were looking for (apart from the problems with very small regions), we modified it and obtained the following formula:

$$d_S(S_1^k, I_i^j) = \frac{\left|s^k - s_j\right|}{\delta + (1-\delta)\cdot(s^k + s_j)} \quad \text{(EQ 33)}$$

As it can be seen in Figure 24, where $\delta = 1/20$, Equation 33 maintains the general behavior of Equation 32, but it eliminates the problems with very small regions. The obtained dissimilarity values match more closely the human perception of size difference.

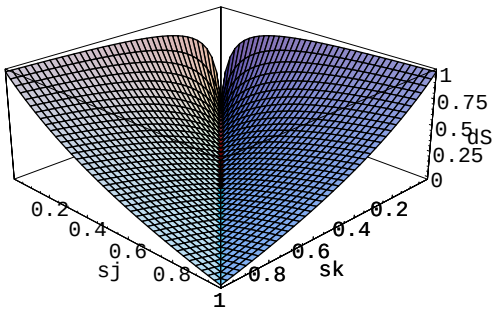

FIGURE 24. A modified version of Equation 32 allows to compute a size dissimilarity which is closer to the human judgement of size similarity.

7.4.4 Comparing positions

The only positional information stored in the region descriptor is the center of gravity of the region. Hence, difference in positioning between a sketch region S_1^k and a database image region I_i^j is mea-

sured by computing the euclidean distance between the regions' centers of gravity $(\bar{x}(S_1^k), \bar{y}(S_1^k)), (\bar{x}(I_i^j), \bar{y}(I_i^j))$ as in Equation 34.

$$d_P(S_1^k, I_i^j) = \left\| (\bar{x}(S_1^k), \bar{y}(S_1^k)) - (\bar{x}(I_i^j), \bar{y}(I_i^j)) \right\| \quad \textbf{(EQ 34)}$$

It should be noticed that in this framework this measure is a good way of estimating the distance between two regions, while in the region merging phase of the feature extraction process (presented in Section 6.3) it was not. In this case, we have to handle regions coming from separate images, hence they may (and should, if one is meant to represent the other) overlap, leading to very similar positions of their center of gravity.

7.4.5 Comparing shapes

As explained in Section 6.3.2, The shape of a region is represented as an ellipse with unit surface. We suppose that similar regions will generate similar ellipses, and in this paragraph we will explain how these shape descriptors are compared. As we mentioned before, an early version of Blobworld [19] also represented regions as ellipses. Other systems (QBIC [43], for example) compute moments and compare the obtained values using a distance between the obtained vectors.

OUR APPROACH

Similar regions have similar shape descriptions. Hence, we can expect that, when overlapped, ellipses generated from similar regions will have a large percentage of their surface in common, as shown in Plate 79. We could then think that an appropriate shape dissimilarity measure could be extracted from this value. An empirical analysis of the problem showed us, though, that this is true only if the ratio between the ellipses' axes is close to 1. The common surface of two very elongated ellipses is low even if the angle between them is small (Figure 25).

FIGURE 25. Very elongated ellipses have small overlapping surfaces even when the angle between them is small, despite the fact that they are visually quite similar. In this case, the common surface is 63%.

What we need, then, is a way to estimate the similarity of two ellipses of unit area whose origins are located in the same point, as in Figure 26. This measure can be based on the common surface of the ellipses, but must take into account the fact that elongated ellipses have a small overlapping surface even if they are visually similar.

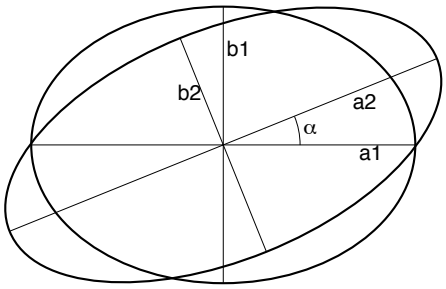

FIGURE 26. Two ellipses $E_1(a_1, b_1), E_2(a_2, b_2)$ centered at the same point and with the same surface.

Let a_1, a_2 be the length of the major axes of two ellipses of unit surface, like the ones pictured in Figure 26, with $a_1 \geq a_2$, and let α be the angle between the ellipses' major axes. Since the ellipses have unit surface, their minor axes measure $b_1 = 1/(a_1 \cdot \pi)$ and $b_2 = 1/(a_2 \cdot \pi)$ respectively.

The technique we used approximates the surface $G(S^k, I_i^j)$ of the smallest of the two rectangles pictured in Figure 27. This is done using Equation 35.

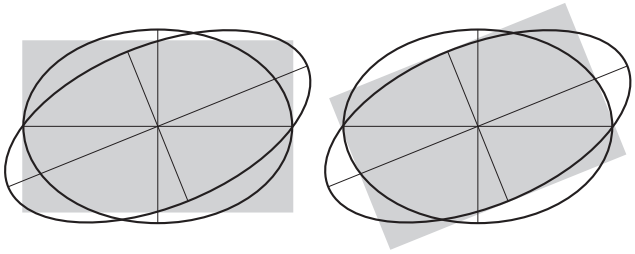

FIGURE 27. The more similar the two ellipses are, the bigger the two grey rectangles will be.

$$
\begin{aligned}
G(S^k, I_i^j) = min(& \\
& min(a_1, \sqrt{(a_2 \cdot \cos(\alpha))^2 + (b_2 \cdot \sin(\alpha))^2}) + \\
& min(b_1, \sqrt{(b_2 \cdot \cos(\alpha))^2 + (a_2 \cdot \sin(\alpha))^2}), \\
& min(a_2, \sqrt{(a_1 \cdot \cos(\alpha))^2 + (b_1 \cdot \sin(\alpha))^2}) + \\
& min(b_2, \sqrt{(b_1 \cdot \cos(\alpha))^2 + (a_1 \cdot \sin(\alpha))^2}))
\end{aligned}
\qquad \textbf{(EQ 35)}
$$

Since the computed surface does not range between 0 and 1, a normalization is needed in order to use $G(S^k, I_i^j)$ to compute shape dissimilarity $d_H(S_1^k, I_i^j)$, as in Equation 36:

$$
d_H(S_1^k, I_i^j) = 1 - \frac{G(S^k, I_i^j)}{\pi} \qquad \textbf{(EQ 36)}
$$

The value obtained with Equation 36 satisfies our needs: in fact, it reaches its maximum (i.e., 1) when $\alpha = 0$, it reaches its minimum when $\alpha = \frac{\Pi}{2}$ and grows in a monotonous way between these two points. Furthermore, for a given α, the dissimilarity between two ellipses depends on the length of their major axes: the more elongated the ellipses are, the bigger their dissimilarity will be (Figure 28).

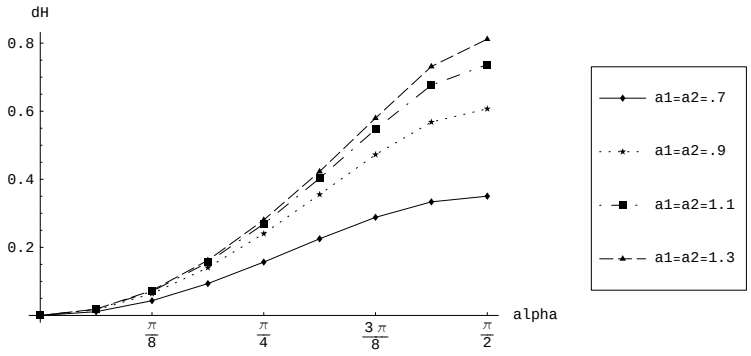

FIGURE 28. Values of d_H obtained with various values of a_1, a_2.

This corresponds to what a human observer would perceive (Figure 29).

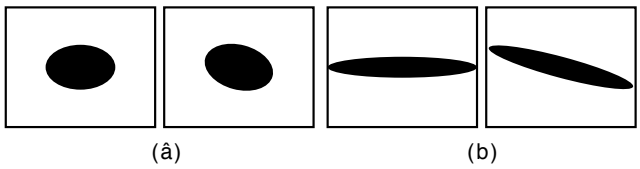

FIGURE 29. When rotated by the same angle, the ellipses in (a) look more similar than those in (b).

7.4.6 Combining the Dissimilarities

Since we compare the regions' features individually, the comparison of S_1^k and I_i^j generates four values: the color dissimilarity $d_C(S_1^k, I_i^j)$, the size dissimilarity $d_S(S_1^k, I_i^j)$, the position dissimilarity $d_P(S_1^k, I_i^j)$ and the shape dissimilarity $d_H(S_1^k, I_i^j)$, all bound between 0 and 1.These individual dissimilarities must be combined to generate $d_R(S_1^k, I_i^j)$. There are two possibilities to do so:

- combine them independently;
- group them according to the domain they belong to (color or geometric information) and then combine the newly obtained color and geometric dissimilarities.

In both cases, though, it is not clear how the various dissimilarities should be combined in order to accurately reproduce the human perception of dissimilarity between two regions, assuming that this could be done with the four values we compute from the regions.

For this reason, among the multiple existing possibilities, we chose to compute $d_R(S_1^k, I_i^j)$ as a weighted sum (Equation 37) where $\omega_S + \omega_C + \omega_P + \omega_H = 1$. Using a weighted sum has the advantage that, by changing the weights, the user can explicitly put more importance on one feature rather than another, and that the results are rather easy to understand. Furthermore, with this solution, low dissimilarity scores are obtained only if all the individual dissimilarities are low.

$$d_R(S_1^k, I_i^j) = \omega_S \cdot d_S(S_1^k, I_i^j) + \omega_C \cdot d_C(S_1^k, I_i^j) + \omega_P \cdot d_P(S_1^k, I_i^j) + \omega_H \cdot d_H(S_1^k, I_i^j) \quad \textbf{(EQ 37)}$$

In our experiments, reported in Chapter 8, several sets of weights were tested.

7.5 Conclusion

In this chapter, we presented several techniques that can be used to compare the description of a sketch with that of a database image. We introduced methods for comparing the regions contained in the descriptors, as well as techniques that use the region dissimilarities to compute a dissimilarity value between segmentation results by associating regions whose dissimilarity is low. We also presented techniques for combining the dissimilarity results obtained when comparing the single segmentation result of a sketch S with those of a database image I, thus obtaining a single dissimilarity score that can be used to rank the database images according to their dissimilarity to the sketch.

In the following chapter we will present some experiments we made, which show that the techniques described in this chapter and in Chapter 6 can be useful in CBIR.

7.6 Color Plates

PLATE 73. **Despite being semantically similar, the above images are not visually similar.**

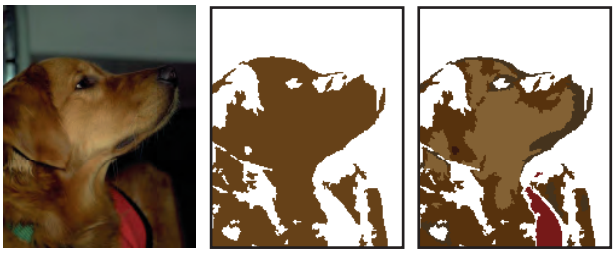

PLATE 74. **The region in image (b) is the result of the merging of the regions in image (c); (a) is the original image.**

PLATE 75. **In this image, the user may consider the red number on the car as the most significant area of the image. If the user were to sketch the image as a red patch in the middle, and two grey areas for the tarmac, our system would consider the grey areas as more significant because they are larger.**

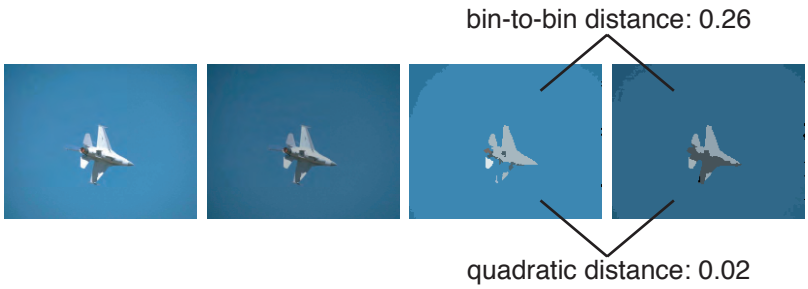

PLATE 76. **From left to right, the original image, an image obtained by darkening it, and their respective segmentation results. A simple color shift was enough to obtain a rather large value for the bin-to-bin color distance, while the quadratic distance is less sensitive to this kind of transformation.**

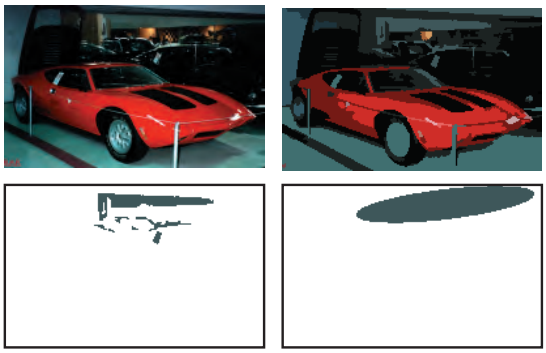

PLATE 77. An original image (top left), a segmentation result (top right), one of the extracted regions (bottom left) and its representation as an ellipse (bottom right).

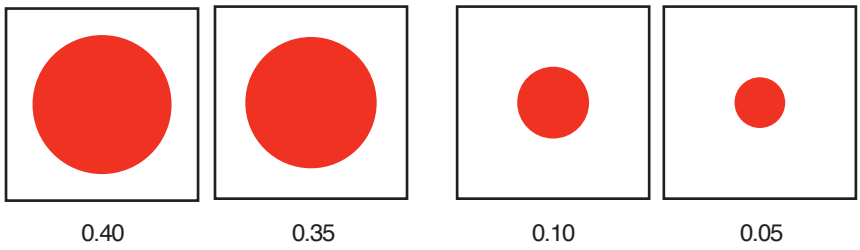

PLATE 78. Size differences between regions are more notable if the regions are small than if they are rather large. For this reason, using the difference between the sizes to represent how visually different the sizes of the regions are is not a good solution.

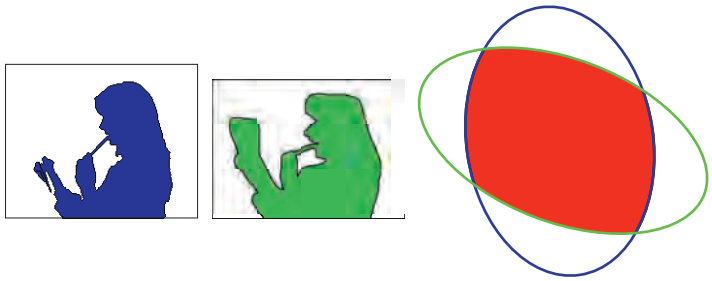

PLATE 79. The comparison between the shapes of two regions may be done by computing the overlapping area of the ellipses representing their shape (the red area in the figure).

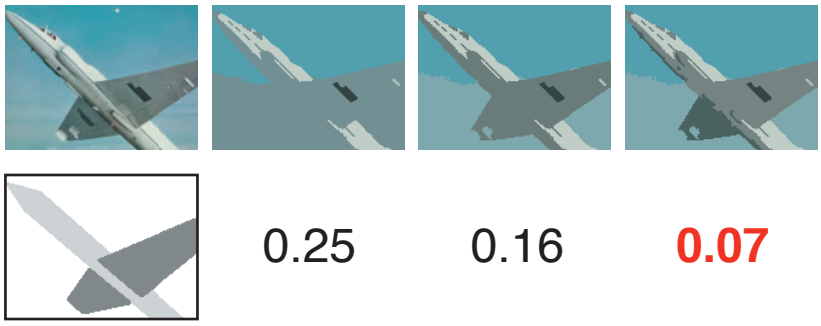

PLATE 80. Each segmentation result of the database image is compared to the sketch, resulting in a set of dissimilarity values, of which the smallest is chosen as the dissimilarity between the sketch and the database image.

CHAPTER 8 *Results*

In this chapter we will present some results obtained with our technique. When possible, we will compare them with results obtained with other techniques. With our experiments we would like to show that combining local features extracted from regions obtained by segmentation with incomplete sketches gives an effective way to perform CBIR.

As we wrote in the previous chapters, our approach is based on sketches produced by the user. In Section 8.1 we will explain how these sketches were produced and how we decided which ones to use in the experiments.

In Section 8.2 we discuss the utility of computing local features, as opposed to computing global features, when the sketches cover all the available canvas (complete sketches). The results obtained with SimEstIm will be compared with two other methods:

- computation of dissimilarity according to the images' histograms;
- computation of dissimilarity according to a technique developed at the University of Fribourg by Lyse Robadey, which is briefly presented in Section 8.2.2.

Finally, in Section 8.3 we will move on to incomplete sketches. Unfortunately, we were unable to repeat our experiences with another CBIR system (an attempt in this direction with the QBICAT system [41] failed for technical reasons, namely because the program was written in a version of Java which is not compatible with the current one), hence the evaluation of results will be limited to the efficiency of retrieval.

8.1 Experimental Setup

In this section we will talk about the database, as well as about the sketch production and to the sketch selection processes. In Section 8.1.1 the database we used for our experiments will be introduced. In Section 8.1.2 we will explain how the sketches were produced, while in Section 8.1.3 we will explain why and how we decided to remove some sketches from the set used for the experiments.

8.1.1 The Database

Our image database consists of 3985 color images originating from two sources:

- various Internet sites;

- the bonus photo collection contained in the “Incredible 65000 Image Pak” published by Click-ART.

Among the portrayed subjects there are animals, people, buildings, landscapes, cars, airplanes, sea scenes, indoor pictures, food, monuments, flowers, everyday objects and paintings. Hence, according to the classification we defined in Section 2.2, the database is an heterogeneous one.

All the images are converted from the JPEG format to the TIFF format, and are scaled so that their largest dimension is 200 pixels.

An image representation is then extracted from every database image, as explained in Chapter 6. For each database image, 10 segmentation results are stored.

8.1.2 Sketch Production

Our experiments are meant to simulate a situation in which a user wants to search for a particular image in a database, supposing that he/she has a vivid memory of the image. Hence, in order to produce a sketch, users had to be given a target image. Target images were randomly selected from the database, and were printed on A6-sized paper with a 300 dpi HP LaserJet 1600 CM printer. These images were then given to the users.

Users were also given some guidelines on how to paint the sketches. In particular, they were told to:

- use color patches to produce the sketch;
- paint only the areas of the target image they found relevant or easy to paint, and leave the rest unpainted;
- try to produce a sketch which was visually similar to the target image when seen from a distance;
- not paint very small detail.

In order to produce the sketch, users used Adobe Photoshop with the following limitations:

- use only the “pencil” (with any brush size) and “bucket” tools;
- do not use anti-aliasing.

It should be noticed that the users were free to:

- choose the size of the sketch, although they were advised to keep it rather small;
- choose the colors to be used in the sketch from all the possible colors available in Photoshop.

During the production of the sketch, the user was allowed to keep the printed image in front of him/her. A sketch production session, once the user had looked at the picture for a couple of minutes, took from one to three minutes, depending on the level of detail the user wanted to achieve and the complexity of the image. We also noticed that users tended to become increasingly fast with the number of sketches they produced.

8.1.3 Sketch Selection

The people that were so kind to produce sketches for our experiments (colleagues, friends, relatives, students) did not always produce sketches that were visually similar to the target images.

This is essentially due to two reasons. First of all, these people were not really looking for a particular image, since they were given an image to look for; hence, their interest was somewhat low compared to that of a person who is searching for an image because he/she actually needs it. This lack of interest surely resulted in some unsatisfactory sketches. The second reason for “bad” sketches is that sometimes users did not understand our guidelines. A clear example of this can be seen in Plate 81: in this

case, the user painted the outline of the koala instead of producing a sketch containing color patches. Since our system is not able to process a sketch of this kind, retrieval results would be extremely poor.

For these reasons, prior to running the tests, we had to make a selection of the obtained sketches. This selection was based on two main criteria:

- how well the sketch follows the guidelines;
- fidelity of the color patches to the regions in the target image (color, position, size).

The sketches were judged by three people, one being the author and the other two being PhD students but not computer science specialists. A sketch was rejected if at least two of the three people judging it found it not appropriate.

9 users submitted 69 sketches, of which 43 were considered appropriate and were used for our tests. Some of the rejected sketches can be seen in Plate 82.

8.2 Complete Sketches

In this section, we will discuss the results obtained with SimEstIm when the users produced a sketch which covered the whole available canvas.

The results will be compared with those obtained with the following two techniques:

- comparison of the images' color histograms;
- comparison using a global features technique developed by Lyse Robadey at the University of Fribourg.

These techniques will be shortly introduced in Section 8.2.1 and Section 8.2.2. In Section 8.2.3 we will explain how our technique was used in order to generate the listed results, which will be presented in Section 8.2.4. Section 8.2.5 will contain the discussion of the results obtained with the color histogram comparison technique, while in Section 8.2.6 we will compare the behavior of the global features technique to that of the local features one.

8.2.1 Comparison of Sketch and Database Image According to the Color Histograms

The color histogram we used is the 125-bin L*u*v* histogram presented in Section 6.3.2.

The dissimilarity between a sketch and a database image was computed using the quadratic distance between their color histograms, as explained in Section 7.2.3. The database was then sorted according to the dissimilarities computed.

8.2.2 Comparison of Sketch and Database Image According to Complex Global Features

This kind of comparison was made using a technique developed by Lyse Robadey, which extracts global features from the sketch and from the image, generating a feature vector.

Let $r(x, y)$, $g(x, y)$, $b(x, y)$ be the RGB color coordinates of the pixel at position (x, y), and let w, h be the width and height of the image. For simplicity, we will omit some definitions when they can be easily deduced from other definitions. The extracted features are:

- the average color $(\bar{r}, \bar{g}, \bar{b})$ in the RGB color space:

$$\bar{r} = \frac{1}{w \cdot h} \cdot \sum_{x=1}^{w} \sum_{y=1}^{h} r(x, y) \quad \text{(EQ 38)}$$

$\bar{g}, \bar{b}$ are similarly defined;

- $\overline{rx}, \overline{ry}, \overline{gx}, \overline{gy}, \overline{bx}, \overline{by}$, defined as

$$\overline{rx} = \frac{1}{w \cdot h} \cdot \sum_{x=1}^{w} \sum_{y=1}^{h} r(x, y) \cdot x \quad \text{(EQ 39)}$$

the other definitions are similar;

- the correlations $(C_{rx}, C_{ry}, C_{gx}, C_{gy}, C_{bx}, C_{by})$ between the RGB values and the spatial position of the pixels, defined as:

$$C_{rx} = \frac{\sigma_{rx}}{\sqrt{\sigma_r^2 \cdot \sigma_x^2}}, \quad \text{(EQ 40)}$$

where

$$\begin{aligned} \sigma_r &= \sqrt{\frac{1}{w \cdot h} \cdot \sum_{x=1}^{w} \sum_{y=1}^{h} r(x, y)^2 - \bar{r}^2} \\ \sigma_x &= \sqrt{\frac{1}{w \cdot h} \cdot \sum_{x=1}^{w} \sum_{y=1}^{h} x - \bar{x}^2} \\ \sigma_{rx} &= \sqrt{\frac{1}{w \cdot h} \cdot \sum_{x=1}^{w} \sum_{y=1}^{h} (r(x, y) \cdot x)^2 - \overline{rx}^2} \end{aligned} \quad \text{(EQ 41)}$$

the other definitions are similar

- D_r, D_g, D_b, defined as:

$$\overline{D_r} = \frac{1}{w \cdot h} \cdot \sum_{x=1}^{w} \sum_{y=1}^{h} D_r(x, y) \quad \text{(EQ 42)}$$

where

$$D_r(x, y) = \begin{cases} |g(x, y) - b(x, y)| & \text{if} \quad r(x, y) > g(x, y) \wedge r(x, y) > b(x, y) \\ 1 & \text{otherwise} \end{cases}; \quad \text{(EQ 43)}$$

the other definitions are similar

- $\overline{D_r x}, \overline{D_r y}, \overline{D_g x}, \overline{D_g y}, \overline{D_b x}, \overline{D_b y}$, defined as:

$$\overline{D_r x} = \frac{1}{w \cdot h} \cdot \sum_{x=1}^{w} \sum_{y=1}^{h} D_r(x, y) \cdot x \quad \text{(EQ 44)}$$

(the other definitions are similar)

- the correlations $C_{D_r x}, C_{D_r y}, C_{D_g x}, C_{D_g y}, C_{D_b x}, C_{D_b y}$ between $D_r(x, y), D_g(x, y), D_b(x, y)$ and (x, y):

$$C_{D_r x} = \frac{\sigma_{D_r x}}{\sqrt{\sigma_{D_r}^2 \cdot \sigma_x^2}} \quad \text{(EQ 45)}$$

defined like in Equation 40 and Equation 41.

- $\overline{U}, \overline{V}, \overline{W}$, where

$$\begin{aligned} \overline{U} &= \frac{1}{w \cdot h} \cdot \sum_{x=1}^{w} \sum_{y=1}^{h} U(x, y) \\ \overline{V} &= \frac{1}{w \cdot h} \cdot \sum_{x=1}^{w} \sum_{y=1}^{h} V(x, y) \\ \overline{W} &= \frac{1}{w \cdot h} \cdot \sum_{x=1}^{w} \sum_{y=1}^{h} W(x, y) \end{aligned} \quad \text{(EQ 46)}$$

$$\begin{aligned} U(x, y) &= 2 \cdot r(x, y) - g(x, y) - b(x, y) \\ V(x, y) &= 2 \cdot g(x, y) - r(x, y) - b(x, y) \; ; \\ U(x, y) &= 2 \cdot b(x, y) - g(x, y) - r(x, y) \end{aligned} \quad \text{(EQ 47)}$$

- $\sigma_U, \sigma_V, \sigma_W$, where

$$\sigma_U = \sqrt{\frac{1}{w \cdot h} \cdot \sum_{x=1}^{w} \sum_{y=1}^{h} U(x, y)^2 - \overline{U}^2} \quad \text{(EQ 48)}$$

definitions of σ_V, σ_W are similar;

The result of the feature extraction process is a 36-component vector. All components are normalized so that they range between 0 and 100. The dissimilarity between a sketch and an image is represented by the city block distance between their corresponding feature vectors.

8.2.3 Local Features

When using local features, region dissimilarity d_R was computed with Equation 26 (color dissimilarity), Equation 33 (size), Equation 34 (position), Equation 35 (shape) and Equation 37. Dissimilarity between segmentation results d_L , on the other hand, was computed using Equation 23 and the final dissimilarity between query and database image d was obtained with Equation 13.

Since the users produced the query images using Photoshop, no information apart from the image itself was available for the query. In particular, we did not know what kind of information (color, size, position, shape) was the most important for the users when they produced their queries. For this reason, we compared the given query images to the images in the database using several combinations of the weights used in the region dissimilarity measure described in Section 7.2.

The weights used in the computation of region dissimilarity are shown in Table 4. In all the tests, the threshold τ (Section 7.3.3) was empirically set to 0.1, after observing that in general regions whose dissimilarity exceeds this value are not visually similar. The penalty weight ϖ_P introduced in Equation 20 was set (also empirically) to 2.

The reported result is the best rank obtained by the target image.

Weights set	ω_S	ω_P	ω_C	ω_H
W_1	0.20	0.30	0.30	0.20
W_2	0.20	0.30	0.40	0.10
W_3	0.25	0.25	0.25	0.25
W_4	0.25	0.25	0.40	0.10
W_5	0.30	0.30	0.20	0.20
W_6	0.30	0.30	0.30	0.10
W_7	0.35	0.35	0.20	0.10

TABLE 4. The region dissimilarity weight sets used in our experiments.

8.2.4 Results

SKETCHES

For this experiment, 17 sketches, produced by 8 different users, were used. 2 sketches represented the same image, while the other 15 were sketches of different target images. The sketches, along with their target images, can be seen from Plate 83 to Plate 98.

RESULTS

In Table 5 the rank obtained by the target image when using the three compared techniques is listed. By looking at it, we can see that:

Image name	Histograms	Global features	Local features
amcamx3r (Plate 83)	469th	1st	1st
anmin001 (Plate 84)	330th	1st	1st
anmpt068 (Plate 85)	35th	2nd	1st
bldcm006 (Plate 86)	1st	7th	1st
bldhm124 (Plate 87)	518th	3rd	1st
crypt1 (Plate 88)	405th	85th	1st
fp0233 (Plate 89)	1st	2nd	1st
fodvf021 (Plate 90)	1st	1st	2nd
frmag065 (Plate 91)	19th	6th	1st
frmma003 (Plate 92)	164th	8th	732nd
info-hwy (1) (Plate 93)	593rd	33rd	3rd
info-hwy (2) (Plate 93)	154th	23rd	6th
intcl078 (Plate 94)	14th	1st	1st
pplm1019 (Plate 95)	290th	3rd	2nd
ssgp1563 (Plate 96)	1st	2nd	1st
ssgp5174 (Plate 97)	1582nd	82nd	3rd
wld-16 (Plate 98)	4th	6th	1st

TABLE 5. Comparison of results obtained with histograms, global features and local features.

- in general, both global features and local features perform significantly better than the color histogram technique;
- there are only few differences between the results obtained with the global features technique and our technique.

Both these facts become even clearer if we look at the percentage of target images whose rank is below a given position, shown in Figure 30. In the following section we will discuss the reasons behind this behavior.

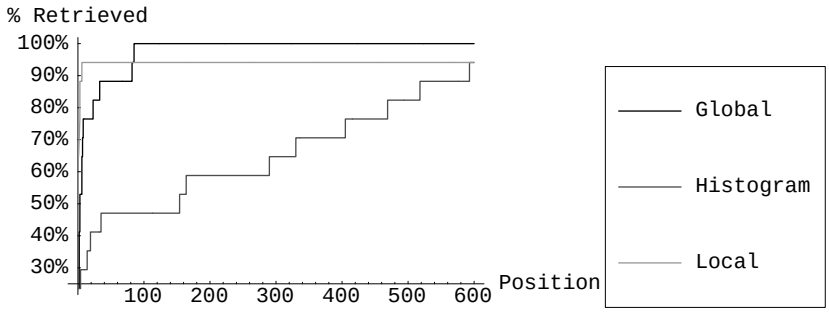

FIGURE 30. Results obtained with global features and with local features are extremely similar. Both techniques perform significantly better than color histograms.

8.2.5 Discussion of Histogram Technique Results

The reasons behind the poor results obtained when using color histograms to compute the dissimilarity between sketches and database images is due to the fact that color histograms only represent the distribution of colors in the image, and do not offer any information about where in the image the colors are located. Thus, there can exist a very large number of images having a very similar color histogram, including images with completely different contents. An example of this can be seen in Plate 99.

It is not surprising, then, to see that the color histogram produces good results only in a very limited number of cases, while most of the time it performs significantly worse than the other two methods we tested.

8.2.6 Comparison of Local and Global Features Results

By looking at Table 5 and Figure 30, we realize that searching using local features and searching using global features seem to perform in a very similar way.

In order to compare the results, before running the tests we supposed results would be shown to the user in pages containing 30 images, leading to the following classification of results:

- the methods perform **identically** if the image is retrieved at the same position by both methods;
- one method performs **slightly better** than the other if the position of the target image using one method is better than the position obtained using the other method but the target image appears on the same result page with both methods (for example, image fodvf021 is retrieved in 1st position using global features and in 5th position using local features, but in both cases the target image would appear in the first page of results);
- one method performs **significantly better** than the other if the position of the target image using one method is better than the position obtained using the other method and the target image appears on different result pages (for example, image crypt1 is retrieved in 85th position using global features and would appear in the third page of results, while with local features it is retrieved in 1st position and would be in the first page of results).

If we compare results this way, we find that:

- the two methods perform identically 3 times;
- global features perform slightly better once;
- local features perform slightly better 9 times;
- global features perform significantly better once;

- local features perform significantly better 3 times.

GLOBAL FEATURES PERFORMING SIGNIFICANTLY BETTER

The query that gives a significantly better result with global features is frmma003 (Plate 92). In this case, the bad retrieval result is obtained because the stored segmentation results (some of which can be seen in Plate 100) are not correct. Since the regions extracted from the sketch do not look like those extracted from the target image, it is natural that the dissimilarity score is rather high, thus leading to the low position of the target image in the ranking.

The reason behind the poor segmentation result is due to the following reasons. First of all, the original image is rather blurred, and contains greyish colors. In this case, the colors of the extracted regions will be quite similar (especially because greyish colors in the L*u*v* color space are closer than they appear to the human observer). This will have a significant impact during the region merging phase (Section 6.3.4). Furthermore, while the initial segmentation result (Plate 101) may appear to be rather similar to the one extracted from the sketch, it contains a multitude of small compact regions whose color significantly differs from the color of the regions surrounding them. Let us remind that in the region merging phase we merge pairs of regions whose shape and color are the most compatible. In this case, we find ourselves with:

- small regions whose shape is compatible with that of their neighbors, but whose color is significantly different;
- large regions whose shape is not totally compatible with that of their neighbors, but whose color is quite similar (in the L*u*v* space).

In this situation, a merging between neighboring large regions will be considered a better choice than a merging between a small region and a large region surrounding it. Hence, large regions will tend to merge, even if the resulting regions do not represent homogeneous areas in the original image, giving the segmentation results shown in Plate 100.

Image frmma003 is not the only image for which the segmentation is not correct. Another example of this (rather rare) behavior of the region merging algorithm can be seen in Plate 102.

LOCAL FEATURES PERFORMING SIGNIFICANTLY BETTER

There are three queries for which local features perform significantly better than global features: crypt1 (Plate 88), info_hwy (1) (Plate 93) and ssgp5174 (Plate 97). In each case, it can be seen (Plate 103, Plate 105 and Plate 105) that one of the segmentation results stored in the target images' descriptor is extremely similar to the query image, thus resulting in a low dissimilarity score.

Query image ssgp5174 (Plate 97) is particularly interesting, as the colors are not too accurate. Despite this, our method still manages to retrieve the image in 3rd place, because size, position and shape of the painted regions are very similar to those of the segmentation result, something that cannot be done by the global features techniques, which does not have access to information of this kind.

8.2.7 Conclusion

After our experiments with complete sketches, we can draw some interesting conclusions:

- The results obtained with local and global features for complete sketches are very similar.
- Errors in the feature extraction phase (segmentation, region merging) will almost certainly lead to poor retrieval results.
- The local feature technique appears to be more robust than the global features one.

There is another fact that should be taken into account: users were not explicitly told to produce complete sketches, hence they did only when they were comfortable with it. This means that the images for which complete sketches were produced may be images whose structure is simpler. Producing a complete sketch for some kinds of images, in fact, might be too complex. An example of such images

can be found in Plate 106. Forcing users to produce complete sketches for these kinds of images would probably result in inaccurate query images and poor retrieval results.

In the following section we will discuss the results obtained with incomplete sketches, which make the sketch production process lighter for the user, who is no longer forced to produce an image representing the whole target image.

8.3 Incomplete Sketches

This section is dedicated to the discussion of the results obtained with our technique and incomplete sketches. The results will be presented in Section 8.3.1, and they will be analyzed in Section 8.3.2.

8.3.1 Results

SKETCHES

For this experiment, 26 incomplete sketches, produced by 8 different users, were added to the 17 complete sketches used in Section 8.2.4, giving a total of 43 query images for 31 target images. Table 6 contains a summary of the number and kind of sketches available for the 31 target images. The incomplete sketches (in which white areas are "not painted" areas) can be seen in Plate 107 to Plate 126, along with their target images.

Image name	incomplete	complete
18c_1997_	1	0
amcamx3r	2	1
anmin001	0	1
anmpt068	2	1
bldcm006	0	1
bldcm114	1	0
bldcm229	1	0
bldhm124	0	1
crypt1	0	1
fodvf021	0	1
fp0233	0	1
frmag065	1	1
frmma003	0	1
info_hwy	0	2
intcl078	0	1
intcl113	1	0
koala	1	0
natot167	3	0
natot410	2	0
natot447	2	0
natot519	1	0
natot685	1	0
natsn087	1	0
pplm1019	1	1
pplw1047	1	0
ssgp0910	1	0
ssgp1563	0	1
ssgp5174	1	1

TABLE 6. Number of incomplete and complete sketches available for each target image.

Image name	incomplete	complete
swan	1	0
wld-16	0	1
yumiko43	1	0

TABLE 6. Number of incomplete and complete sketches available for each target image.

The experimental setup used was the same as in our experiment with complete sketches, described in Section 8.2.3: each incomplete sketch was compared to every image in the database using the seven sets of weights in Table 4, and the best ranking reached by the target image was chosen.

RESULTS

The best result obtained for each incomplete sketch is reported in Table 7.

Query image	Position
18c_1997_ (Plate 107)	39th
amcamx3r (1) (Plate 108)	15th
amcamx3r (2) (Plate 108)	4th
anmpt068 (1) (Plate 109)	1st
anmpt068 (2) (Plate 109)	1st
bldcm114 (Plate 110)	12th
bldcm229 (Plate 111)	1st
frmag065 (Plate 112)	1st
intcl113 (Plate 113)	90th
koala (Plate 114)	1st
natot167 (1) (Plate 115)	1st
natot167 (2) (Plate 115)	1st
natot167 (3) (Plate 115)	9th
natot410 (1) (Plate 116)	1st
natot410 (2) (Plate 116)	10th
natot447 (1) (Plate 117)	1st
natot447 (2) (Plate 117)	4th
natot519 (Plate 118)	1st
natot685 (Plate 119)	1st
natsn087 (Plate 120)	1st
pplm1019 (Plate 121)	1st
pplw1047 (Plate 122)	3rd
ssgp0910 (Plate 123)	25th
ssgp5174 (Plate 124)	265th
swan (Plate 125)	83rd
yumiko43 (Plate 126)	41st

TABLE 7. Results obtained with incomplete sketches.

8.3.2 Discussion of Results

The results obtained with incomplete sketches are extremely encouraging. In fact, as it can be seen from Figure 31, 80% of the target images were retrieved in the top 30 positions, meaning that they would have appeared in the first page of results shown to the user. The number of target images appearing in the top 10 returned images is also extremely high: 65%.

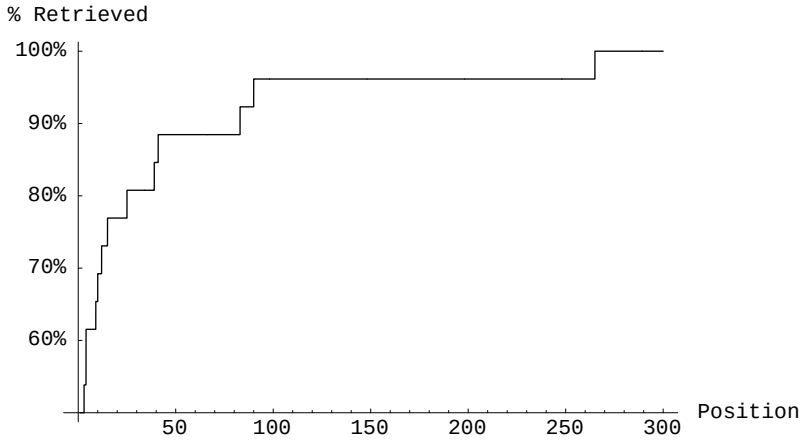

FIGURE 31. Results obtained with incomplete sketches.

Furthermore, if we add the results obtained with incomplete sketches to those obtained with complete sketches (Figure 32), things look even better: over 85% of the target images appear in the top 30 retrieved images, while 75% are in the top 10.

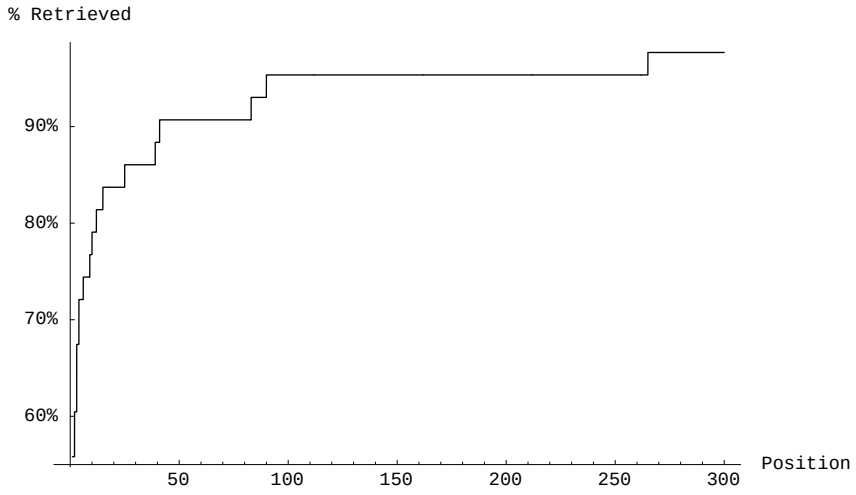

FIGURE 32. Results obtained over the whole collection of sketches (complete and incomplete).

Although the number of sketches used in the experiments is rather small, these results make us believe that our technique can be used to perform CBIR in an effective way. Good results were obtained also with simple sketches like amcam3x (1) (Plate 108), anmpt068 (Plate 109), bldcm006 (Plate 110), koala (Plate 114), natot167 (Plate 115) and natot685 (Plate 119).

8.3.3 Reasons Behind Poor Retrieval Results

Even though the results reported in the previous sections are rather good, there are still queries that do not result in the target image being retrieved among the first images. In this section, we will try to list the various reasons why a query may be unsuccessful.

Errors in Initial Segmentation

Although it is quite reliable, even when used with different kinds of images, the segmentation algorithm we use (Section 6.2.5) does occasionally undersegment images, especially if the original image contains gradients or regions with very similar colors (Plate 127).

If the user produces a sketch containing only a part of the undersegmented region, retrieval results will almost surely be poor. Actually, this was not the case with any of the query images used in our experiments.

Errors During the Region Merging Phase

As we mentioned in Section 8.2.6, errors may appear during the region merging phase, leading to regions issued from unrelated areas of the image to merge. The resulting behavior is similar to what we experience with undersegmented images.

This problem would be eliminated by storing every segmentation results produced by merging regions, although this would certainly increase the time needed to compare the query with the database images. More realistically, other techniques for the region merging phase should be studied.

User Painting Only Part of a Region

Another source of poor retrieval results is given by sketches in which only a part of a region contained in the target image is painted. An example is sketch intcl113 (Plate 128), in which the user only painted the horizontal dark lines, ignoring the oblique ones. The segmentation algorithm, though, produced a single region containing all the dark lines in the upper part of the image.

In situations like the one pictured in Plate 128, a region of the sketch is compared to a region of the target image of which it represents a subset. Currently, our region dissimilarity measure is not able to handle this case and returns a rather large dissimilarity score (since size, position and shape of the two regions are significantly different), which ultimately leads to a low ranking of the target image.

The solution to this problem could be twofold:

- users could be given some information on how the system works internally; this would help them avoiding this kind of problem;
- a different region dissimilarity technique that is able to recognize that a sketch region is a subset of a target image region could be developed, although it is not clear how this could be done without risking to introduce a large amount of false matches.

8.4 Conclusion

In this chapter, we presented the results obtained with our method with complete and incomplete sketches.

When used with complete sketches, our approach produces retrieval results very similar to those that can be obtained with a method based on global features. It must be said, though, that complete sketches are hard to produce for some kinds of images. For these images, incomplete sketches are in general much easier to produce. Techniques based on global features, though, cannot be effectively used with incomplete sketches, as no information about from which part of the image the features were extracted is available.

Incomplete sketches, on the other hand, give the user the freedom of producing query images containing only the relevant information about the target image (or the areas that are the easiest to paint). The results we obtained with incomplete sketches are very similar to those obtained with complete sketches: in 80% of the tests, the target image was retrieved in the top 30 images.

8.5 Color Plates

PLATE 81. **Some users did not understand our guidelines. In this case, the user painted the contour of the koala, while we asked for sketches composed of color patches.**

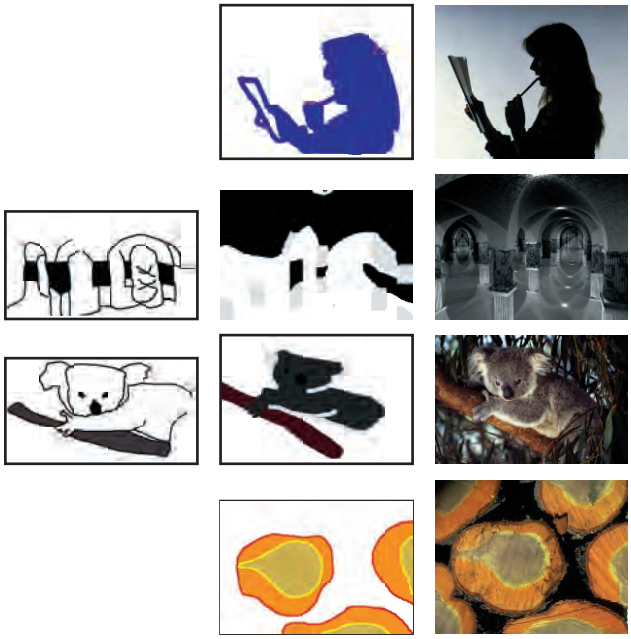

PLATE 82. **Some rejected sketches, along with their target images.**

PLATE 83. **Sketch and target image anmcamx3r.**

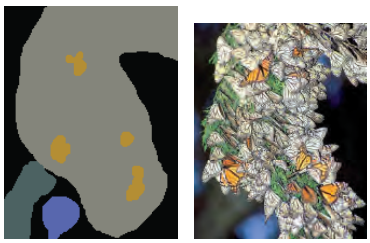

PLATE 84. **Sketch and target image anmin001.**

PLATE 85. Sketch and target image anmpt068.

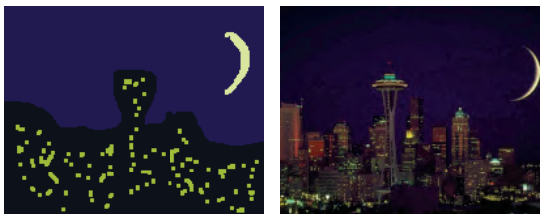

PLATE 86. Sketch and target image bldcm006.

PLATE 87. Sketch and target image bldhm124.

PLATE 88. Sketch and target image crypt1.

PLATE 89. Sketch and target image fp0233.

PLATE 90. Sketch and target image fodvf021.

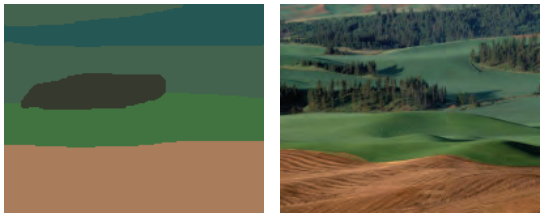

PLATE 91. Sketch and target image frmag065.

PLATE 92. Sketch and target image frmma003.

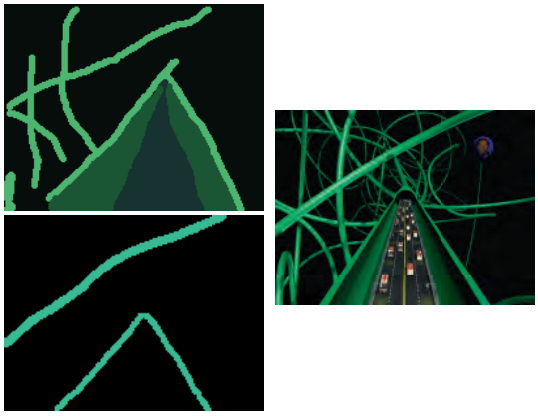

PLATE 93. Sketches and target image info_hwy.

PLATE 94. Sketch and target image intcl078.

PLATE 95. Sketch and target image pplm1019.

PLATE 96. Sketch and target image ssgp1563.

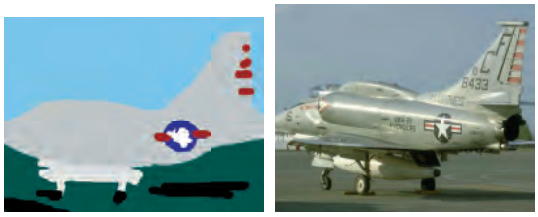

PLATE 97. Sketch and target image ssgp5174.

PLATE 98. Sketch and target image wld-16.

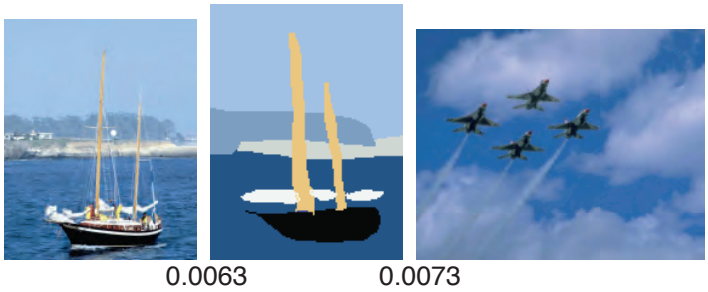

PLATE 99. **Two images can have color histograms very similar to the one of the query image even if their content is totally different.**

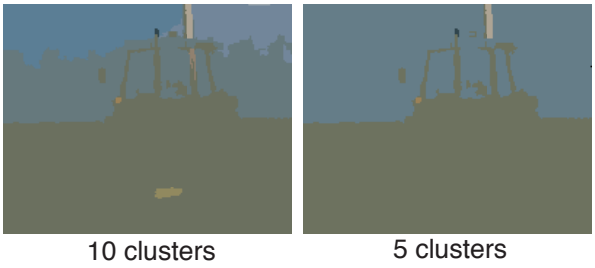

PLATE 100. **Segmentation results for image frmma003.**

PLATE 101. **The segmentation result of the sketch and the initial segmentation result of the target image.**

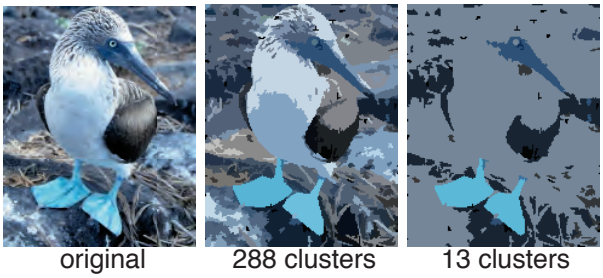

PLATE 102. **Another example of incorrect merging order. While the initial segmentation result (containing 288 clusters) is correct, the body of the bird ends up being merged with the background even in the most detailed segmentation result (containing 13 clusters) stored in the image descriptor.**

PLATE 103. The segmentation result of the crypt1 query image, and the less dissimilar segmentation result in the target image (the original images are in Plate 88).

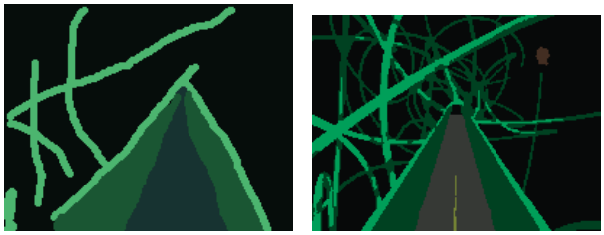

PLATE 104. The segmentation result of the info_hwy (1) query image, and the less dissimilar segmentation result of the target image (the original images are in Plate 93).

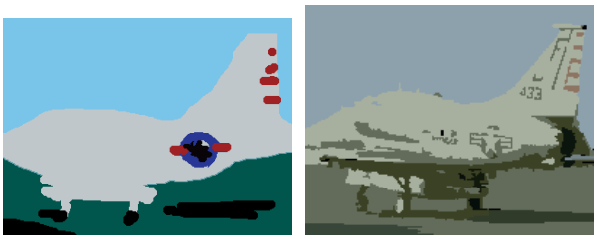

PLATE 105. The segmentation result of the ssgp5174 query image, and the less dissimilar segmentation result in the target image (the original images are in Plate 97).

PLATE 106. A selection of images for which the production of a complete sketch may be problematic.

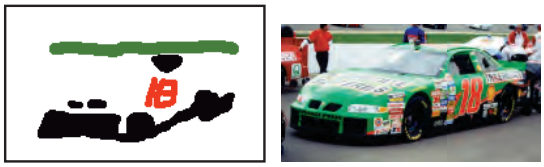

PLATE 107. Incomplete sketch for target image 18c_1997_.

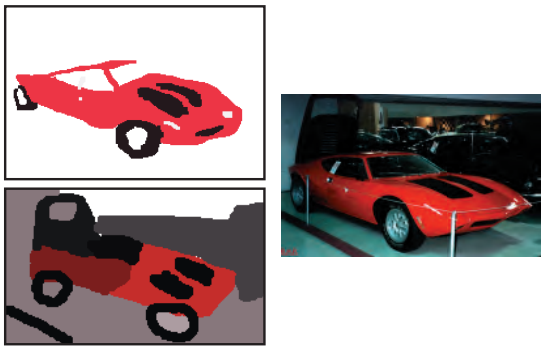

PLATE 108. Incomplete sketches for target image amcamx3r.

PLATE 109. Incomplete sketches for target image anmpt068.

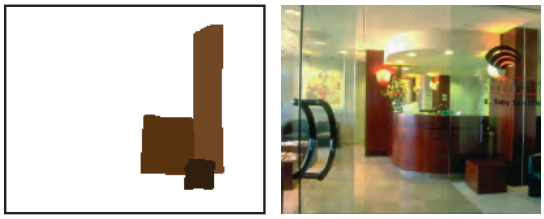

PLATE 110. Incomplete sketch for target image bldcm006.

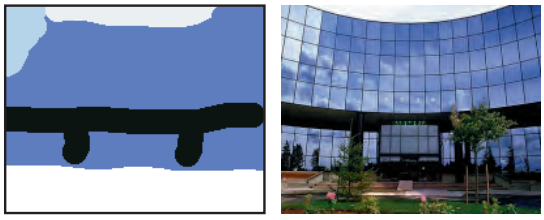

PLATE 111. Incomplete sketch for target image bldcm229.

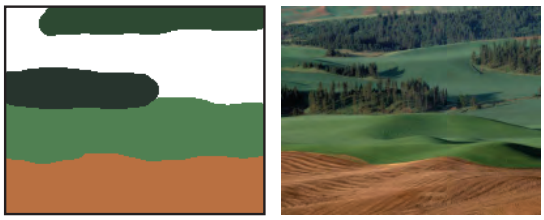

PLATE 112. **Incomplete sketch for target image frmag065.**

PLATE 113. **Incomplete sketch for target image intcl113.**

PLATE 114. **Incomplete sketch for target image koala.**

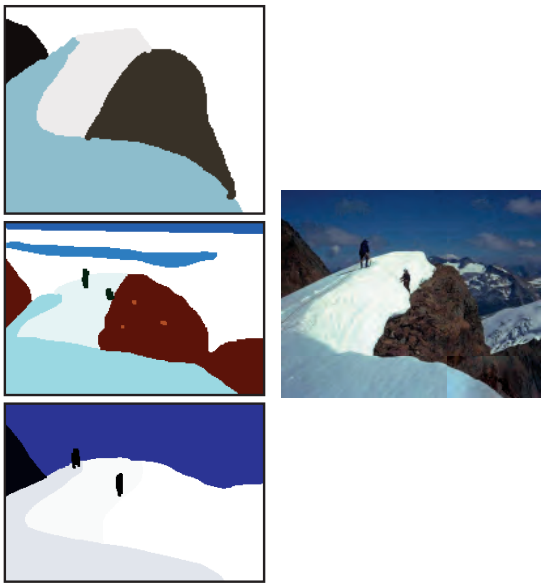

PLATE 115. **Incomplete sketches for target image natot167.**

PLATE 116. Incomplete sketches for target image natot410.

PLATE 117. Incomplete sketches for target image natot447.

PLATE 118. Incomplete sketch for target image natot519.

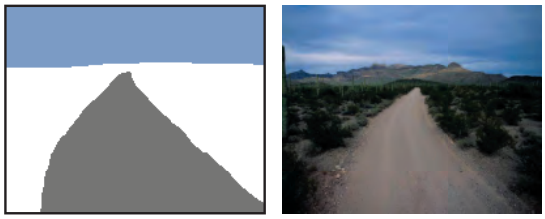

PLATE 119. Incomplete sketch for target image natot685.

PLATE 120. Incomplete sketch for target image natsn087.

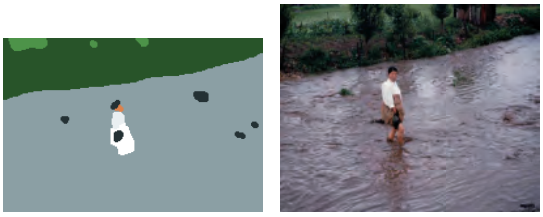

PLATE 121. **Incomplete sketch for target image pplm1019.**

PLATE 122. **Incomplete sketch for target image pplw1047.**

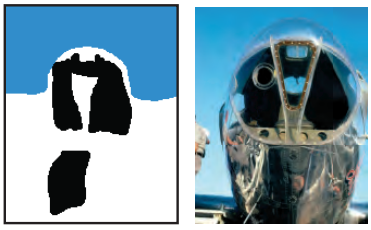

PLATE 123. **Incomplete sketch for target image ssgp0910.**

PLATE 124. **Incomplete sketch for target image ssgp5174.**

PLATE 125. **Incomplete sketch for target image swan.**

PLATE 126. Incomplete sketch for target image yumiko43.

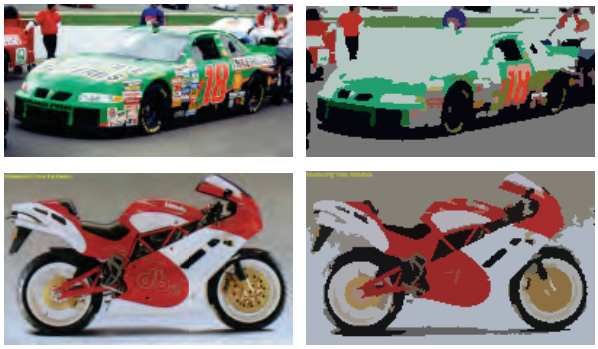

PLATE 127. Two examples of undersegmented initial segmentation results.

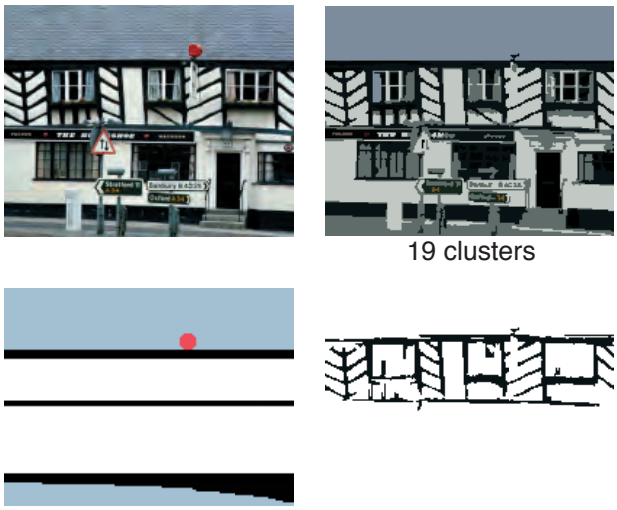

PLATE 128. In this case, the color patches painted by the user correspond to subsets of the region extracted from the target image.

CHAPTER 9 *Conclusion*

Our work led us to explore a multitude of subjects related to CBIR. In this chapter we draw some conclusions, and we present how our work can be useful for future works.

Our goal was not that of building a complete and operational CBIR system. Instead, we wanted to study new techniques for CBIR. As we mentioned in Section 1.3, we found that there was room for improvement in three key aspects of CBIR: query form, representation of database images and queries and comparison between queries and database images.

In particular, we felt that there were some kinds of users that were being neglected, as none of the systems existing at the time could satisfy their needs. Our attention was especially focused on users wanting to retrieve a known image from an heterogeneous database in which image information had not been extracted manually. Users that may find themselves in such a situation can come from various fields: publishing, advertisement, design and retail are some of the examples.

Our analysis of the problem (Section 2.3 and Chapter 4) led us towards querying by user-produced pictorial examples (in form of query images or sketches), as this seems to be the most effective way for a user to search visually for a given image. More precisely, we focused our attention on incomplete queries (i.e., pictorial examples that do not cover all the available canvas).

Furthermore, we also noticed (Section 2.4) that most of the CBIR systems were extracting information from database images as a whole (global features). By extracting features from limited areas of the images (local features) we managed to improve the results obtained with global features, as shown in Section 8.2

While investigating local features, we studied different ways of extracting them. After analyzing the strengths and weaknesses of having human operators perform the task (Section 2.4.3), we decided to explore the possibilities given by automatic feature extraction. In particular, we studied the selection of the areas from which the features are extracted via a segmentation method. Our experiences showed that segmentation can produce relevant regions for feature extraction, as reported in Chapter 6.

Once regions are extracted from images, they must be represented by a feature set. Our choice was to represent regions as a combination of size, position, shape and color (Section 6.3).

Also, we studied the consequences that sketches with different levels of detail would have on the retrieval results, resulting in an image representation composed of several segmentation results. On

the other hand, the different nature of the user-produced queries led us towards a descriptor containing a single segmentation result.

CBIR ultimately rests on the comparison of the query with (a subset of) the database images. We took advantage of our representation of query and database image to develop a complex and customizable method for comparing a query and a database image (Chapter 7). Our solution is composed of a dissimilarity measure between regions, which is then used to compute the dissimilarity between segmentation results (by associating pairs of regions that have a low dissimilarity score), which in turn is used to compute the dissimilarity between the query and the database image. According to this dissimilarity, it is then possible to rank all the database images. The user can control (to a certain extent) how similarity is computed.

In Section 9.1 we summarize the scientific contributions of our work. Section 9.2, on the other hand, we talk about the lessons that are to be drawn from our work. Criticism and future perspectives are the content of Section 9.3, while in Section 9.4 we draw a short conclusion.

9.1 Contribution

This work shows that it is possible to develop efficient CBIR methods in which:

- the query consists of an incomplete user-produced pictorial example of a target image;
- the retrieval is based on automatically extracted features;
- features are extracted from regions which are automatically formed according to their visual homogeneity;
- the same representation is used for both queries and database images;
- the database images representation is flexible enough to be compared to sketches of different levels of detail;
- the comparison of query and database images allows partial matches.

Our principal contributions are split among three areas: feature extraction, image and query representation and comparison.

Feature Extraction

We present a technique for the automatic selection of visually homogeneous areas for later extraction of image features, which is based on region segmentation followed by region merging.

Image and Query Representation

We introduce a common representation for database images and sketches, which contains a set of segmentation results, each of which is composed of a set of regions. In order to ensure the maximum flexibility, database images can be represented by a variable number of segmentation results. Using regions as the basic element of the representation allows the use of incomplete sketches.

Comparison

A dissimilarity measure between database images and query images is presented. This measure is based on a region dissimilarity, which is then used to compute the dissimilarity between segmentation results by associating pairs of regions that have low dissimilarity. The dissimilarities between segmentation results are then used to compute the dissimilarity between query and database image. The way the dissimilarity is computed can be influenced by the user by setting a set of weights, thus making it customizable to different drawing techniques and needs. Results show that the measure handles equally well complete and incomplete sketches.

9.2 What We Learned

This work led us to analyze different aspects of CBIR. In this section we will list some of our observations on the subject.

- The first important observation is that effective CBIR by user-drawn pictorial example in a heterogeneous database using only automatically extracted local features is possible.
- We also found that our method is effective even if the queries are incomplete. Incomplete queries free the user from the burden of having to produce an image representing the whole target image. This way, it is possible to retrieve an image by painting only a part of its content.
- Associating several consecutive segmentation results to the database images does indeed allow efficient retrieval with queries containing various levels of detail: good retrieval results are obtained with sketches ranging from extremely raw to rather detailed.
- Associating importance weights to sketch regions mirrors the fact that not all areas of the query are equally important to the user. Assigning weights proportionally to the regions' sizes turns out to be a rather accurate estimation of region importance.

9.3 Appreciation and Future Work

In this section, we talk about testing that should be carried out and possible improvements to our method. We also try to list new functionalities that would be necessary in order to transform it into a fully operational CBIR system.

FURTHER TESTING

- Our experiments were conducted using a database of about 4000 images. Obviously, in a real-life situation the amount of images contained in the database would be of hundreds of thousands of images. It would be interesting, then, to verify how our technique would perform in such a case.
- It would be extremely interesting to verify if letting the user directly specify the importance weights of the regions in the query image would lead to better retrieval results. Similarly, the user may also be asked to give relative importance weights for size, position, color and shape information for each region, and these would then influence the way the dissimilarity between the query image and the database images is computed. This kind of testing, though, is not practical in the current state of our work, as no graphical user interface (GUI) is available to the user.
- Another interesting kind of testing consists in, instead of trying to retrieve a target image, trying to retrieve all images containing a particular shape, or a region of a given color and size, and so on. This can be done by setting the weights for size, position, color or shape information in the region dissimilarity measure to zero or low values, so that the corresponding features are ignored.
- Comparing the results obtained with our method with those obtained with other methods relying on incomplete sketches and automatically extracted features would be advisable. Unfortunately, our attempt to use the QBICAT system (Section 3.4.8) for this purpose was unsuccessful for technical reasons (notably the incompatibility between the version of Java it was written in and the current version).
- Our evaluation of results was uniquely based on the position of the target image, because no information about the relevance of the images in the database compared to the target images was available. It would be interesting to generate such information and verify how relevant the images retrieved at the top of the ranking are. In order to do this, though, it should be kept in mind that relevance must be evaluated only visually, and not semantically, since our dissimilarity measure is only sensitive to visual similarity.
- Testing reported in this work was performed on queries produced by colleagues, friends and relatives of the author. Although these people surely gave their best to follow the guidelines and produce "good" queries, it would be extremely interesting to verify how the method would perform if

it was used in a real-life situation (like in an advertisement firm, for example) by people with a real need to retrieve a particular image.

- The sketches used in the experiments were produced using a mouse as a drawing device, which is not a very natural way of drawing. Tests should be performed using other drawing devices, such as a graphic tablet.

Improvements

- Probably the area of this work where improvement would be most effective is the region merging algorithm. New approaches should be tested. Furthermore, we believe that adaptive thresholds should be used to decide when to stop the merging process, instead of the fixed thresholds that are currently used.
- Another weak point of this work is the region shape descriptor. As we mentioned in Section 6.3.2, representing region shapes as ellipses and comparing them by computing the overlapping surface is not a very good approach. Alternative techniques may prove difficult to find, though, because they will need to be applicable to non-connex regions.
- Currently, region dissimilarity is computed as a weighted sum of size, position, color and shape dissimilarity. While this approach appears to be effective, it is not proven that it corresponds to the way human beings combine different types of dissimilarities. More investigation should be made to verify if other meaningful ways of combining dissimilarities can be found.

The Road to a Complete CBIR System

Although building a complete CBIR system was never our goal, we can list some of the things that would need to be done before our method can be used in one.

- The method should be re-implemented in a more effective way. In fact, performance not being a goal in this work, the current data structure is extremely heavy. This leads to both high computing time and memory occupancy. Notably, a search in our database of 3985 images takes between 1 and 10 minutes (depending on the number of regions in the sketch) on a Sun Ultra workstation, and the program requires 540 MB of memory to run. Profiling showed us, though, that about half of this time is spent to navigate through the data structure (which was implemented using hashtables); a lighter data structure would surely lead to lower memory requirements and faster execution. Other significant gains (both time-wise and memory-wise) would be obtained by using simple data types instead of floating-point ones to describe the regions' features.
- A method for indexing the images in the database according to the extracted regions and their features should be studied. In a real-life scenario, a CBIR system simply cannot compare the query to all the images in the database. Hence, a method for selecting plausible candidates for the retrieval should be developed.
- A relevance feedback mechanism should be developed, allowing the user to obtain better retrieval result by marking retrieved images as "relevant" or "non relevant" (as done by Ciocca and Schettini in [25]) or by grouping together images to be considered similar (as in El Niño [121]). The user actions should have an impact on how the dissimilarity between query and database images is computed.
- As we showed in Chapter 8, our technique performs very well for visual searches. It is clear, though, that a complete CBIR system should not be limited to this kind of search. Hence, other CBIR techniques should be combined with ours to broaden the possibilities. Notably, CBIR techniques based on semantic data (keywords, natural language descriptions, etc.), contours (histogram of contour directions, Fourier descriptors, etc.) and texture should be combined with ours, resulting in a more complete CBIR system.
- A GUI would need to be built. Ideally, the user should be able to produce the query, adjust any weights, see the retrieval results and refine or modify the query from within the same GUI.
- The way results are presented to the user should be studied in detail. In particular, the user should be able to understand why the retrieval results are as they are. One way of letting the user understand is by showing the internal representation of the images, as done in Blobworld [17].

9.4 Finally...

Developing a complete CBIR requires a huge amount of work, as well as knowledge in several very different domains (image processing, information retrieval, interface building, vision, natural language processing and many others). With our work, we hope to give people writing the CBIR systems of tomorrow some more information about what to do and what not to do when building such a system.

CHAPTER 10

References

1. AltaVista Image Search Engine, http://www.altavista.com/cgi-bin/query?mmdo=1&stype=simage
2. M. Amadasun, R. King, "Textural Features Corresponding to Textural Properties", *IEEE Transactions on System, Man and Cybernetics, Vol. 19*, 1989, pp. 1264-1274.
3. P. Aigrain, H. Zhang, D. Petkovic, "Content-Based Representation and Retrieval of Visual Media: A State-of-the-Art Review", *Multimedia Tools and Applications, No. 3*, 1996, pp. 179-202.
4. Y. Alp Aslandogan, C. Thier, C. Yu, Y. Zou, N. Rishe, "Using Semantic Contents and WordNet" in Image Retrieval", *Proceedings of SIGIR 97, Philadelphia, USA*, 1997, pp. 286-295.
5. D. Baker, S. Hwang, J. Aggarwal, "Detection and Segmentation of Man-Made Objects in Outdoor Scenes: Concrete Bridges", *Journal of the Optical Society of America A*, Vol. 6, No. 6, 1989, pp. 938-951.
6. F. Banfi, "Image Databases: State of the Art and Future Directions", *Internal Report 96-10, IIUF, University of Fribourg*, 1996.
7. F. Banfi, R. Ingold, "Computing Dissimilarity Between Hand-Drawn Sketches and Digitized Images", *Proceedings of VISUAL 99, Amsterdam, The Netherlands*, 1999, pp. 625-631.
8. F. Banfi, R. Ingold, "A Dissimilarity Measure for Query by Example Retrieval", *Proceedings of the 1999 Eurographics Multimedia Workshop, Milan, Italy*, 1999, pp. 43-52.
9. R. Barber, B. Beitel, W. Equitz, C. Niblack, D. Petkovic, T. Work, P. Yanker, "Image Query System and Method", *United States Patent, No. 5,579,471*, 1996.
10. G. Becker, "Content-Based Query of Image Databases", *in G. Becker (Ed.), Information in Images*, Ch. 6, Thomson Technology Labs White Paper, 1996.
11. B. Berlin, P. Kay, "Basic Color Terms: their Universality and Evolution", University of California Press, 1969.
12. L. Bonsiepen, W. Coy, "Stable Segmentation Using Color Information", *Proceedings of CAIP'91, Dresden, Germany*, 1991, pp. 77-84.
13. M. Bouet, C. Djeraba, "Powerful Image Organization in Visual Retrieval Systems", *Proceedings of the 6th ACM Multimedia Conference, Bristol, UK*, 1998.
14. S. Bow, "Pattern Recognition - Applications to Large Data-Set Problems", Marcel Dekker Inc., 1984.
15. P. Brodatz, "Textures: a Photographic Album for Artists and Designers", Dover, 1966.
16. W. Campbell, W. Mackeown, B. Thomas, T. Troscianko, "Interpreting Image Databases by Region Classification", *Pattern Recognition, Vol. 30, No. 4*, 1997, pp. 555-563.

17. C. Carson, M. Thomas, S. Belongie, J. Hellerstein, J. Malik, "Blobworld: A System for Region-Based Image Indexing and Retrieval", *Proceedings of VISUAL 99, Amsterdam, The Netherlands*, 1999, pp. 509-516.

18. C. Carson, S. Belongie, H. Greenspan, J. Malik, "Color- and Texture-Based Image Segmentation Using EM and Its Application to Image Querying and Classification", *submitted to IEEE Transactions on Pattern Recognition and Machine Intelligence*, 1999.

19. C. Carson, S. Belongie, H. Greenspan, J. Malik, "Region-Based Image Querying", *Workshop on Content-Based Access of Image and Video Libraries, San Juan , Puerto Rico*, 1997.

20. R. Cattel (Ed.), "The Object Database Standard: ODMG-93", Morgan Kaufmann, San Francisco, 1994.

21. G. Cha, C. Chung, "Object-Oriented Retrieval Mechanism for Semistructured Image Collections", *Proceedings of the 6th ACM Multimedia Conference, Bristol, UK*, 1998.

22. Y. Cheng, "Mean Shift, Mode Seeking, and Clustering", IEEE Transactions on Pattern Analyisis and Machine Intelligence, Vol. 17, pp. 790-799, 1995.

23. L. Cinque, F. Lecca, S. Levialdi, S. Tanimoto, "Retrieval of Images Using Rich Image Descriptions", *Proceedings of the ICPR*, 1998.

24. G. Ciocca, I. Gagliardi, R. Schettini, "Retrieving Color Images by Content", *Proceedings of the Image and Video Content-Based Retrieval Workshop*, 1998.

25. G. Ciocca, R. Schettini, "Using a Relevance Feedback Mechanism to Improve Content-Based Image Retrieval", *Proceedings of the third Conference on Visual Information Systems (VISUAL 99), Amsterdam, The Netherlands*, 1999, pp. 107-114.

26. D. Comaniciu, P. Meer, "Robust Analysis of Feature Spaces: Color Image Segmentation", *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR'97, Puerto Rico*, 1997, pp. 750-755.

27. Commission Internationale de l'Eclairage, "Recommendations on Uniform Color Space, Color-Difference Equations, and Psychometric Color Terms", *Supplement No. 2 to Publication CIE*, No. 15, 1978.

28. J. Corridoni, A. Del Bimbo, P. Pala, "Retrieval of Paintings Using Effects Induced by Color Features", *1998 International Workshop on Content-Based Access of Image and Video Databases*, Bombay, India, 1998, pp. 2-11.

29. W. Cowan, "An Inexpensive Scheme for Calibration of a Colour Monitor in Terms of CIE Standard Coordinates", *Computer Graphics*, Vol. 17, No. 3, 1983, pp. 315-321.

30. C. Cuadra, R. Katter, "Experimental Studies of Relevance Judgments, *Report TM-3520, Final Report*, Vol. 1, System Development Corporation, Santa Monica, USA, June 1967.

31. M. Daily, "Colour Image Segmentation Using Markow Random Fields", *Proceedings of CVPR'89, San Diego, USA*, 1989, pp. 304-312.

32. M. Das, R. Manmatha, E. Riseman, "Indexing Flower Patent Images Using Domain Knowledge", IEEE Intelligent Systems, Vol. 14, No. 5, pp. 24-33, 1999.

33. A. Del Bimbo, M. Mugnaini, P. Pala, F. Turco, L. Verzucoli, "Image Retrieval By Color Regions", *Proceedings of the International Conference on Image Analysis and Processing, ICIAP 97 , Firenze, Italy,* 1997, pp.180-187.

34. A. Del Bimbo, P. Pala, "Visual Image Retrieval by Elastic Matching of User Sketches", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 2, February 1997, pp. 121-132.

35. A. Del Bimbo, E. Vicario, "Using Weighted Spatial Relationships in Retrieval by Visual Contents", *IEEE Workshop on Image and Video Libraries, Santa Barbara, USA*, 1998.

36. Y. Deng, B. Manjunath, H. Shin, "Color Image Segmentation", *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR '99*, Fort Collins, USA, 1999.

37. A. Dimai, "Assessment of Effectiveness of Content Based Image Retrieval Systems", *Proceedings of VISUAL 99, Amsterdam, The Netherlands*, 1999, pp. 525-532.

38. M. Do, S. Ayer, M. Vetterli, "Invariant Image Retrieval Using Wavelet Maxima Moment", *Proceedings of the third International Conference on Visual Information Systems (VISUAL 99), Amsterdam, The Netherlands*, 1999, pp. 451-458.

39. R. Duda, P. Hart, "Classification and Scene Analysis", Wiley, 1973.

40. H. Durrett (Ed.), "Color and the Computer", Academic Press, 1987.

41. J. Favela, V. Meza, "Image-Retrieval Agent: Integrating Image Content and Text", IEEE Intelligent Systems, Vol. 14, No. 5, pp. 24-33, 1999.

42. F. Ferri, E. Vidal, "Colour Image Segmentation and Labeling Through Multiedit-Condensing", *Pattern Recognition Letters*, Vol. 13, No. 8, 1992, pp. 561-568.

43. M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, P. Yanker, "Query by Image and Video Content: The QBIC System", *IEEE Computer*, September 1995, pp. 23-32.

44. D. Forsyth, J. Malik, R. Wilensky, "Searching for Digital Pictures", *Scientific American*, June 1997, pp. 72-77.

45. J. Francos, "Orthogonal Decompositions of 2-D Random Fields and Their Applications for 2-D Spectral Estimation", *in N. Bose, C. Rao (Eds.), Signal Processing and its Applications*, North-Holland, 1993, pp. 287-327.

46. K. Fukunaga, "Introduction to Statistical Pattern Recognition", Second Edition, Academic Press, Boston, USA, 1990.

47. I. Gagliardi, R. Schettini, "A Method for the Automatic Indexing of Color Images for Effective Image Retrieval", *The New Review of Hypermedia and Multimedia, Vol. 3*, 1997, pp. 201-224.

48. J. Gauch, C. Hsia, "A Comparison of Three Color Image Segmentation Algorithms in Four Color Spaces", *Proceedings of SPIE Vol. 1818, Visual Communications and Image Processing '92*, pp. 1168-1181.

49. T. Gevers, A. Smeulders, "Color-Based Object Recognition", *Pattern Recognition*, Vol. 32, 1999, pp. 453-464.

50. T. Gevers, A. Smeulders, "Interactive Query Formulation for Object Search", *Third International Conference on Visual Information Systems, Amsterdam, The Netherlands*, 1999, pp. 593-600.

51. T. Gevers, "PicToSeek: a Content-Based Image Search System for the World Wide Web", *Proceedings of VISUAL '97, San Diego, USA*, 1997.

52. W. Goffman, "On Relevance as a Measure", *Information Storage and Retrieval, Vol. 2, No. 3*, December 1964, pp. 201-203.

53. W. Goffman, V. Newill, "Methodology for Test and Evaluation of Information Retrieval Systems", *Comparative Systems Laboratory, Report CSL: TR-2, Western Reserve University, Cleveland, USA*, July 1964.

54. V. Gudivada, "Distributed Semantic Multimedia Information Retrieval", *Tutorial Report, ICAST 97 in conjunction with ICMIS 97, Chicago, USA*, 1997.

55. V. Gudivada, V. Raghavan, "Content-Based Image Retrieval Systems", *IEEE Computer*, September 1995, pp. 18-22.

56. A. Gupta, R. Jain, "Visual Information Retrieval", *Communications of the ACM*, Vol. 40, No. 5, May 1997, pp. 71-79.

57. E. Hall, "Computer Image Processing and Recognition", Academic Press, San Diego, USA, 1979.

58. http://drogo.cselt.stet.it

59. M. Hu, "Visual Pattern Recognition by Moment Invariants", *IRE Trans. Inf. Theory*, Vol. 8, pp. 179-187.

60. J. Huang, R. Kumar, M. Mitra, W. Zhuand, R. Zabih, "Image Indexing Using Color Correlograms", *CVPR, Puerto Rico, 1997*, pp. 762-768.

61. D. Huijsmans, M. Lew, "Efficient Content-Based Image Retrieval in Digital Picture Collections Using Projections", Proceedings of the 13th ICPR, Vienna, Austria, 1996, pp. 104-108.

62. D. Huijsmans, M. Lew, D. Denteneer, "Quality Measures for Interactive Image Retrieval with a Performance Evaluation of Two 3x3 Texel-Based Methods", *Proceedings of the International Conference on Image Analysis and Processing, ICIAP 97 , Firenze, Italy,*, 1997, pp.23-29.

63. D. Huijsmans, S. Poles, M. Lew, "2D Pixel Trigrams for Content-Based Image Retrieval", *Proceedings of the International Workshop on Image Databases and Multi-Media Search, Amsterdam, The Netherlands*, 1996, pp. 139-145.

64. R. Hunt, Measuring Color (Second Edition), Ellis Horwood, 1992.

65. T. Huntsberger, C. Jacobs, R. Cannon, "Iterative Fuzzy Image Segmentation", *Pattern Recognition*, Vol. 18, No. 2, 1985, pp. 131-138.

66. J. Itten, "Kunst der Farbe", Otto Maier Verlag, 1961.

67. L. Itti, C. Koch, E. Niebur, "A Model on Saliency-Based Visual Attention for Rapid Scene Analysis", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 11, pp. 1254-1259, 1998.

68. R. Jackson, L. MacDonald, K. Freeman, "Computer Generated Color", Wiley, 1994.

69. C. Jacobs, A. Finkelstein, D. Salesin, "Fast Multiresolution Image Querying", Proceedings of SIGGRAPH 95, Los Angeles, USA, 1995, pp. 277-286.

70. A. Jain, "Algorithms for Clustering Data", Prentice Hall, 1991.

71. A. Jain, A. Vailaya, "Image Retrieval Using Color and Shape", *Pattern Recognition, Vol. 29*, 1996, pp. 1233-1244.

72. R. Jain, B. Horowitz, C. Fuller, A. Gupta, J. Bach, C. Shu, "Similarity Engine for Content-Based Retrieval of Images", *U.S. Patent Nb. 5,893,095*, 1999.

73. J. Jolion, P. Meer, S. Bataouche, "Robust Clustering with Applications in Computer Vision", *IEEE Transactions on Pattern Recognition and Machine Intelligence*, Vol. 13, 1991, 791-802.

74. M. Kankanhalli, B. Mehtre, J. Wu, "Cluster-Based Color Matching for Image Retrieval", Pattern Recognition, Vol. 29, No. 4, pp. 701-708, 1996.

75. P. Kelly, M. Cannon, "Query by Image Example: the CANDID Approach", *Los Alamos National Laboratory White Paper*, 1995.

76. A. Kent, O. Taulbee, J. Belzer, G. Goldstein (eds.), "Electronic Handling of Information: Testing and Evaluation", Thompson Book Company, 1967.

77. D. King, E. Bryant, "The Evaluation of Information Services and Products", Information Resources Press, 1971.

78. S. Kosslyn, "Image and Brain", M.I.T. Press, 1996.

79. M. La Cascia, E. Ardizzone, "JACOB: Just a Content-Based Query System for Video Databases", *Proceedings of ICASSP 96, Atlanta, USA*, 1996.

80. M. La Cascia, S. Sethi, S. Sclaroff, "Combining Textual and Visual Cues for Content-Based Image Retrieval on the World Wide Web", *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries, Santa Barbara, USA*, 1998.

81. F. Lancaster, "Information Retrieval Systems - Characteristics, Testing and Evaluation", Wiley, 1979.

82. M. Lew, T. Huang, "Optimal Supports for Image Matching", Proceedings of the IEEE Digital Signal Processing Workshop, Loen, Norway, 1996, pp. 251-254.

83. M. Lew, D. Huijsmans, D. Denteneer, "Optimal Keys for Image Database Analysis", retrieved from http://www.wi.leidenuniv.nl/home/lim/cbir.okeys.ps.

84. M. Lew, K. Lempinen, N. Huijsmans, "Webcrawling Using Sketches", *Proceedings of VISUAL '97, San Diego, USA*, 1997, pp. 77-84.

85. Y. Lim, S. Lee, "On the Color Image Segmentation Algorithm Based on the Thresholding and the Fuzzy C-means Techniques", *Pattern Recognition*, Vol. 23, No. 9, 1990, pp. 935-952.

86. X. Lin, S. Chen, "Color Image Segmentation Using Modified HSI System for Road Following", *Proceedings of IEEE Conference on Robotics and Automation, Sacramento, USA*, 1991, pp. 1998-2003.

87. S. Liu, M. Jernigan, “Texture Analysis and Discrimination in Additive Noise”, *CVGIP, Vol. 49*, 1990, pp. 52-67.

88. W. Ma, B. Manjunath, “NETRA: a Toolbox for Navigating Large Image Databases”, *in Proc. IEEE International Conference on Image Processing, Santa Barbara, USA*, 1997.

89. W. Ma, B. Manjunath, “Edge Flow: a Framework of Boundary Detection and Image Segmentation”, *in Proceedings of CVPR ‘97, San Juan, Puerto Rico*, 1997, pp. 744-749.

90. W. Ma, B. Manjunath, “Texture features for Browsing and Retrieval of Image Data”, *IEEE Transactions on Pattern Recognition and Machine Intelligence*, Vol. 18, No. 8, 1996, pp. 837-842.

91. D. MacAdam, “Uniform Color Scale”, *Journal of the Optical Society of America*, Vol. ??, No. 64, 1974, pp. 1691-??.

92. Macbeth, “The Munsell Book of Color”, Macbeth, 1979.

93. J. Malki, N. Boujemaa, C. Nastar, A. Winter, “Region Queries without Segmentation for Image Retrieval by Content”, *Proceedings of the third International Conference on Visual Information Systems (VISUAL 99), Amsterdam, The Netherlands*, 1999, pp. 115-122.

94. S. Mallat, S. Zhong, “Characterization of Signals from Multiscale Edges”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14. 1992, pp. 710-732.

95. R. Manmatha, S. Ravela, Y. Chitti, “On Computing Global Similarity in Images”, *Proceedings of SPIE Conference on Human and Electronic Imaging III, San Jose, USA*, 1998.

96. J. Martinez, S. Guillaume, “Colour Image Retrieval Fitted to ‘Classical’ Querying”, *Proceedings of the 9th International Conference on Image Analysis and Processing (ICIAP'97), Florence, Italy*, 1997, pp. 14-21.

97. J. Martinez, S. Marchand, “Towards Intelligent Retrieval in Image Databases”, *Proceedings of the 5th International Worshop on Multi-Media Data Base Systems (MMDBMS'98), Dayton, USA*, 1998, pp. 38-45.

98. F. Meyer, “Colour Image Segmentation”, *Proceedings of IEEE International Conference on Image Processing and its Applications, Maastricht, The Netherlands*, 1992, pp. 303-306.

99. G. Meyer, D. Greenberg, “Perceptual Color Spaces for Computer Graphics”, *in H. Durrett (Ed.), Color and the Computer*, Wiley, 1987, pp.83-100.

100. G. Miller, R. Beckwith, C. Fellbaum, D. Gross, K. Miller, “Introduction to WordNet: an On-line Lexical Database”, *International Journal of Lexicography, Vol. 4, No. 3*, 1990, pp. 235-244.

101. A. Munsell, “A Color Notation“, Munsell Color Company, 1946.

102. G. Murch, “Color Displays and Color Science”, *in H. Durrett (Ed.), Color and the Computer*, Wiley, 1987, pp. 1-26.

103. R. Nevatia, “A Color Edge Detector and its Use in Scene Segmentation”, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 7, No. 11, 1977, pp. 820-826.

104. V. Ogle, M. Stonebraker, “Chabot: Retrieval from a Relational Database of Images”, *IEEE Computer*, September 1995, pp. 40-48.

105. T. Ojala, M. Pietikainen, D. Harwood, “A Comparative Study of Texture Measures with Classification Based on Feature Distributions”, *Pattern Recognition*, Vol. 29, No. 1, 1996, pp. 51-60.

106. N. Pal, S. Pal, “A Review on Image Segmentation Techniques“, *Pattern Recognition*, Vol. 26, No. 9, pp. 1277-1294, 1993.

107. E. Paquet, M. Rioux, “Crawling, Indexing and Retrieval of Three-Dimensional Data on the Web in the Framework of MPEG-7”, *Proceedings of VISUAL 99, Amsterdam, The Netherlands*, 1999, pp. 179-186.

108. G. Pass, R. Zabih, J. Miller, “Comparing Images Using Color Coherence Vectors”, *Proceedings of the Fourth ACM Multimedia Conference*, 1996.

109. J. Payne, L. Hepplewhite, T. Stonham, “Evaluating Content-Based Image Retrieval Techniques Using Perceptually Based Metrics”, *TR-696, Neural Network and Pattern Recognition Laboratory, Brunel University, United Kingdom*, 1999.

110. A. Pentland, R. Picard, S. Sclaroff, “Photobook: Content-Based Manipulation of Image Databases”, *International Journal of Computer Vision*, Vol. 18, No. 3, 1996, pp. 233-254.

111.F. Perez, C. Koch, “Toward Color Image Segmentation in Analog VLSI: Algorithm and Hardware”, *International Journal of Computer Vision*, Vol. 12, No. 1, 1994, pp. 17-42.

112.R. Picard, “A Society of Models for Video and Image Libraries”, *M.I.T: Media Laboratory Perceptual Computing Section Technical Report No. 360*, 1996.

113. The Postgres Group, “The Postgres Reference Manual”, Computer Science Division, University of California at Berkeley, 1993.

114.S. Ravela, R. Manmatha, “Image Retrieval by Appearance”, *Proceedings of SIGIR 97, Philadelphia, USA*, 1997, pp. 278-285.

115.C. Revol, M. Jourlin, “A New Minimum Variance Region Growing Algorithm for Image Segmentation”, *Pattern Recognition Letters*, Vol. 18, No. 3, 1997, pp. 249-258.

116.P. Rousseeuw, A. Leroy, “Robust Regression and Outlier Detection”, Wiley, 1987.

117.G. Salton, “Automatic Text Processing”, Addison-Wesley, 1988.

118.G. Salton, M. McGill, “Introduction to Modern Image Retrieval”, McGraw-Hill, 1983.

119.S. Santini, R. Jain, “Image Databases are not Databases with Images”, *Proceedings of the International Conference on Image Analysis and Processing, ICIAP 97 , Firenze, Italy,*, 1997, pp. 38-45.

120.S. Santini, R. Jain, “The Graphical Specification of Similarity Queries”, *Journal of Visual Languages and Computing*, 1997.

121.S. Santini, R. Jain, “Integrated Browsing and Querying for Image Databases”, *submitted to IEEE Multimedia*, 1999.

122.S. Santini, R. Jain, “Similarity Matching“, *IEEE Transactions on Pattern Recognition and Machine Intelligence*, Vol. 21, No. 9, pp. 871-883, 1999.

123.R. Schettini, “A Segmentation Algorithm for Colour Images”, *Pattern Recognition Letters*, Vol. 14, No. 6, 1993, pp. 499-506.

124.S. Sclaroff, L. Taycher, M. La Cascia, “ImageRover: A Content-Based Image Browser for the World Wide Web“, *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries, Puerto Rico*, 1997.

125.M. Seaborn, L. Hepplewhite, J. Stonham, “PISARO: Perceptual Colour and Texture Queries Using Stackable Mosaics”, *TR-694, Brunel University*, 1998.

126.W. Skarbek, A. Koschan, “Color Image Segmentation - A Survey”, *Technischer Bericht 94-32, Technische Universität Berlin, Fachbereich 13 Informatik*, 1994.

127.J. Smith, S. Chang, “Searching for Images and Videos on the World-Wide Web”, Technical Report #459-96-25, Department of Electrical Engineering and Center for Image Technology for New Media, Columbia University, New York, USA, 1996.

128.H. Späth, “Cluster Analysis Algorithms”, Ellis Horwood, 1980.

129.D. Squire, W. Müller, H. Müller, “Relevance Feedback and Term Weighting Schemes for Content-Based Image Retrieval”, *in Proceedings of the third International Conference on Visual Information Systems (VISUAL 99), Amsterdam, The Netherlands*, 1999, pp.549-556.

130.R. Srihari, “Automatic Indexing and Content-Based Retrieval of Captioned Images”, *IEEE Computer*, September 1995, pp. 49-56.

131.R. Sriran, J. Francos, W. Pearlman, “Texture Coding Using a Wold Decomposition Model”, *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Jerusalem, Israel*, 1994, pp. 35-39.

132.N. Strachan, P. Nesvadba, A. Allen, “Calibration of Video Camera Digitising System in the CIE L*u*v* Colour Space”, *Pattern Recognition Letters*, Vol. 11, No. 5, 1990, pp. 771-777.

133.M. Stricker, M. Orengo, “Similarity of Color Images”, *Storage and Retrieval for Still Image and Video Databases, Vol. 2420*, 1995, pp. 381-392.

134.M. Stricker, M. Orengo, “Similarity of Color Images”, *Proceedings of the SPIE Storage and Retrieval for Image and Video Databases III Conference*, 1995.

135.M. Stricker, M. Orengo, “Similarity of Color Images”, *Proceedings of SPIE 95, San Jose, USA*, 1995.

136.J. Swets, “Effectiveness of Information Retrieval Methods”, *American Documentation*, Vol. 20, No. 1, January 1969, pp. 72-89.

137.A. Tam, C. Leung, “Structured High-Level Indexing of Visual Data Content”, *Proceedings of VISUAL 99, Amsterdam, The Netherlands*, 1999, pp. 409-417.

138.D. Tseng, C. Chang, “Color Segmentation Using Perceptual Attributes”, *Proceedings of the 11th International Conference on Pattern Recognition, The Hague, The Netherlands*, Vol. III, 1992, pp. 228-231.

139.A. Tversky, “Features of Similarity“, Psychological Review, Vol. 84, No. 4, pp. 327-352, 1977.

140.S. Umbaugh, R. Moss, W. Stoecker, G. Hance, “Automatic Colour Segmentation Algorithms with Application to Skin Tumor Feature Identification”, *IEEE Engineering in Medicine and Biology*, Vol. 12, No. 3, 1993, pp.75-82.

141.J. Vendrig, M. Worring, A. Smeulders, “Filter Image Browsing - Interactive Image Retrieval by Using Database Overviews”, *submitted to Multimedia Tools and Applications*, 1999.

142.A. Verikas, K. Malmqvist, L. Bergman, “Colour Image Segmentation by Modular Neural Network”, *Pattern Recognition Letters*, Vol. 18, No. 2, 1997, pp. 173-185.

143.P. Wahle, “Face Recognition Bibliography”, http://manip.crhc.uiuc.edu/~wahle/research/bibliography.html

144.Z. Wang, A. Guerriero, M. De Sario, “Comparison of Several Approaches for the Segmentation of Texture Images”, *Pattern Recognition Letters*, Vol. 17, No. 5, 1996, pp. 509-521.

145.V. Wu, R. Manmatha, E. Riseman, “Finding Text in Images”, Proceedings of the 2nd ACM International Conference on Digital Libraries, Philadelphia, USA, 1997, pp. 3-12.

146.R. Young, “Simulation of Human Retinal Function with the Gaussian Derivative Model”, *Proceedings of the International Conference on Computer Vision and Pattern Recognition, Miami Beach, USA*, 1986, pp. 564-569.

147.S. Zhu, A. Yuille, “Region Competition: Unifying Snakes, Region Growing, and Bayes/MDL for Multiband Image Segmentation”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 9, 1996, pp. 884-900.