Interactive Image Search using Similarity-Based Visualization

Giang Phuong Nguyen

This book is typeset by the author using $\LaTeX2_{\varepsilon}$.

Printing: Febodruk BV, Enschede, the Netherlands.

ISBN-10: 90-5776-160-2
ISBN-13: 978-90-5776-160-7

Interactive Image Search using Similarity-Based Visualization

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor

aan de Universiteit van Amsterdam,

op gezag van de Rector Magnificus prof. mr. P.F. van der Heijden

ten overstaan van een door het College voor Promoties ingestelde commissie,

in het openbaar te verdedigen in de Aula der Universiteit

op dinsdag 19 december 2006 te 11:00 uur

door

Giang Phuong Nguyen

geboren te Hanoi, Vietnam

The work described in this thesis was supported by the ImIk project part of the program IOP MMI (Man-Machine Interaction).

Advanced School for Computing and Imaging

The work described in this thesis has been carried out within the graduate school ASCI, at the Intelligent Systems Lab Amsterdam of the University of Amsterdam. ASCI dissertation series number 130.

Intelligent Systems Lab Amsterdam
University of Amsterdam
The Netherlands

Contents

Chapter 1

Introduction

1.1 Background

The development of technology brings a range of techniques for digital image capture, processing, storage and transmission. With this development, we have seen a rapid increase in the size of digital image collections. The huge amount of images requires efficient techniques for browsing and searching.

One of the main problems while working with large and varied image collections is finding a desired image. In a small collection of a hundred or a thousand images, it is feasible to identify the desired image. For ten thousand images browsing is not practical, let alone for a million. An annotated collection, where the user searches with a set of keywords, can solve the problem but this requires intensive manual annotation of all the images.

The term content-based image retrieval (CBIR) first appeared in 1992 in a paper by Kato [52] describing his experiments on browsing in an image collection by colors and shapes. From that moment up to now, many papers have been published making the term CBIR very popular. It appeared to become an interesting and challenging field of research. A number of overview papers can be found in [104, 93, 2, 116, 72, 61]. In Smeulders et al. [104], various aspects in CBIR are analyzed with features and similarity as the main focus of the reference. In [116], a list of 39 CBIR systems is presented. The authors summarize and discuss techniques used in different systems. The very large number of papers cited in these references show the vitality of research in CBIR.

Also in [104], the goals in CBIR are broadly classified into three main categories: association search, target search, and category search. Search by association is a search class where the user starts the search with no specific aim other than to find interesting things. Target search aims at finding a specific image. Finally, category search looks for images that belong to a specific class. In any of these search tasks the main issue is to compare images. Therefore, CBIR is also known as retrieval of images by similarity.

It is stated in [104] that *"similarity is an interpretation of the image based on* *the difference with another image"*. Up to now, the difference is mainly obtained by comparing features extracted from the images themselves. In literature, features are classified into low-level features such as colors, textures and shapes, and higher level features such as spatial relation, or hierarchically ordered features. There is a number of existing methods developing and improving the performance of retrieval using features. Review papers are presented in [85, 66, 115]. In the meantime, a large number of meaningful types of similarity have been defined. Some of them are associated with specific features. For instance, the color histogram [110] is developed for comparing images based on their color features. With texture features, it is possible to measure the similarity based on the degree of coarseness, directionality and regularity or the response of Gabor filters [85, 63]. For shape based retrieval, global shape matching, skeletonization, or edge based comparison can be applied [66]. In summary, features and similarity are the two building blocks in any CBIR system.

Good features with good similarity functions are usually developed for specific search tasks in a narrow domain. In a broad domain, which *"has an unlimited and unpredictable variability in its appearance even for the same semantic meaning"* [104], techniques in CBIR currently yield unacceptable results. This is mainly because of the difference between the user's understanding of similarity, and the capability of the system in interpreting relations between images. This problem is explained in [30]. The authors point out that within the three search categorizations, there are three levels of user search queries. The first level concentrates on retrieving images by low-level features. For example, for retrieving images in an iconic collection shape features outperform other features. Level 2 is retrieval by logical features which requires some degree of logical inference to identify the search object. A beach scene can be retrieved by using color, e.g., specifying a large region of blue at the top of the image and yellow at the bottom. Level 3 is called retrieval by abstract attributes, which are the highest level of search concepts such as finding images of a certain activity. While most of the user searches are based on the last two levels, current systems operate mainly at level 1 [30]. Therefore, despite all research devoted to CBIR, performance of existing systems is still unsatisfactory for the user [30, 104, 2]. In [104], the problem is made explicit as the *semantic gap*: *"The semantic gap is the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for the user in a given situation."*

The aim of the research in CBIR is to reduce the gap for the user. To do so, different approaches have been proposed. Applying statistical pattern recognition techniques is one of those [49]. Pattern recognition has a close relation with CBIR in the sense that techniques in both fields are meant for separating relevant images from irrelevant ones. The authors discuss various techniques on feature extraction, feature selection, and different ways of creating classifiers in a feature space. The classifiers are learnt on the feature space to best separating relevant and irrelevant images from one another.

Only the user knows exactly what he is looking for, hence in order to reduce the gap, a lot of papers have been considered the human into the loop of retrieving images. *The need for user-in-the-loop in CBIR systems stems from the fact that images reside in a continuous representation space, in which semantic concepts are* *best described in discriminative subspaces "car" are of certain* shape *while "sunset" is more describable by* color. *More importantly, different users at different times may have different interpretations or intended usages for the same image, which makes off-line learning unfeasible in general* [128]. Having the user involved in the search, the system is guided to a better interpretation of similarity using the user's knowledge. This is known as *relevance feedback*. A CBIR system with the appearance of the user in the search is called an *interactive CBIR system*. Since the first interactive system has been proposed, techniques in CBIR not only concentrate on features and similarity, but also developing efficient strategies for user interaction and system adjustment.

A number of examples of such interactive systems can be found in [94, 26, 127, 124, 128, 99, 59]. For example, in [94], the authors propose a relevance feedback based interactive retrieval approach that takes into account the gap between high-level concepts and low-level features, and the subjectivity of human perception of visual content. Interaction between human and computer is to refine high-level queries to representations based on low-level features. The system automatically processes existing query such that the adjusted query is a better approximation to the user's information need. Another way of using relevance feedback is proposed in [59]. The user provides feedback as being relevant or not, using decision trees is to learn a common thread among relevant samples. An overview of existing techniques in CBIR using relevance feedback can be found in [127, 128, 26].

For communicating between human and computer, an intermediate interface is an essential element in interactive CBIR systems. Through this interface, the user provides feedback, and the system shows its current learning progress. The interface is commonly a visualization of images. Different ways of displaying images can be found in [16, 91, 71, 74]. For instance, in [16], the authors present a PathFinder network that visualizes images in a database by visual similarities. The PathFinder is a tree-based structure of representing images. In [91, 71], interfaces are presented where images are displayed using a dimensional reduction of high dimensional feature space to a 2-dimensional space for user interaction with images. A 3D model of displaying images called 3D MARS is presented in [74]. In these references, a common argument is that the interface plays an important role as it affects the relevance feedback given by the user as well as helping the user in understanding the image collections and the system's performance.

We will focus on interactive CBIR systems where we consider the two influent factors, namely visualization and interaction, to reduce the *semantic gap*.

1.2 Problem definition

As mentioned in the previous section, a large number of overview papers on relevance feedback show the importance of interactive search in CBIR. In [104], they present a general framework for interactive systems. In this reference, a *query space* is defined, which consists of four components $\{\mathcal{I}, \mathcal{F}, \mathcal{S}, \mathcal{Z}\}$. $\mathcal{I}$ is an image collection, $\mathcal{F}$ is a set of features derived from $\mathcal{I}$, $\mathcal{S}$ is a dissimilarity function and $\mathcal{Z}$ contains labels of the images in $\mathcal{I}$. From there, an interactive CBIR system performs five main steps: query initialization, query specification, visualization, feedback, and output. In each step, all of the components might be affected. Figure 1.1 shows the proposed framework with the five steps described in [104].

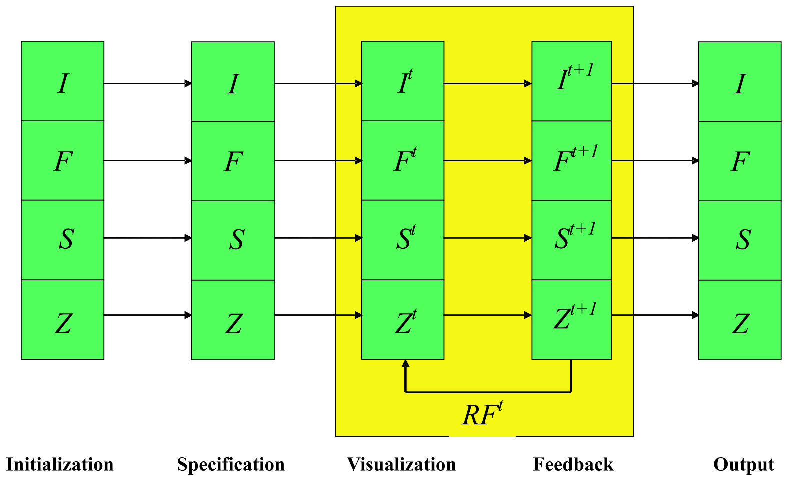


Figure 1.1: A general framework for interactive content based image retrieval systems.

The first step involves the data preparation including normalization of feature values and similarities. In the query specification step, the user poses a search query to define an initial set of images. These two steps are important for any general CBIR system. The interactive stage is formed by two steps: visualization and feedback, which are iterated. The output step presents the final search result to the user. The interactive stage is the main focus of this thesis.

Existing interactive systems normally emphasize one aspect of CBIR. Sometimes this is the sketching capability in the user interface, sometimes it is the new indexing data structure, etc. [116]. Overview papers [94, 128, 59] focus on relevance feedback learning techniques only. On the other hand, systems with advanced visualization techniques only focus on the interface [74, 91, 16]. However, an interactive system is not only affected by the learning technique but also by the quality of the relevance feedback. The quality of feedback strongly depends on the interface with the user. A well designed interface will support the user better in searching. Therefore he will be able to give more useful information to the system. In other words, an interactive system should consider the interaction and visualization interface as one.

Visualization of images is presenting images to the user on a display component such as the computer screen. In visualizing images, a number of issues should be considered carefully. For instance, not all images in the collection can be shown at the same time. A set of images has to be selected, so the question is *which images should be displayed to the user?* Another issue is that the presentation of images on the display should take into account that visualization is not just displaying images but also providing information helpful for user interaction. Hence, another question is *how should images be displayed?*

At first, the visualization step received much less attention compared to the feedback step. At this early stage of interactive CBIR, systems usually employ a simple 2-dimensional grid of images. The ordering of displayed images depends on the purpose of the system. In some systems, images are displayed in random order, while the others are ordered top-down, left-to-right on the degree of relevance to a given query. Examples can be found in [116, 104, 122]. Recent advanced systems have introduced different techniques in visualizing images. For example, systems using hierarchical tree structures to display images [18, 56] where the systems support traversing both on the current level of the hierarchy and between levels of the hierarchy with operations such as panning and zooming. In [22, 11, 91, 87], images are organized such that similar images are grouped or located near each other in the display. This visualization is called similarity-based visualization. Together with these visualization techniques, these systems develop interaction tools to assist the user in giving feedback.

There are different ways of the user giving feedback [128, 94]. In the beginning, the majority of relevance feedback methods asked the user to tune the system parameters during the retrieval process [59, 128]. This requires a lot of effort from the user as well as user expertise in CBIR. To ease the user's tasks, another feedback mechanism is to ask the user to provide feedback on images indicating whether they are relevant or irrelevant to the current search [32], or the user marks the degree of relevance [73]. Some other ways of giving feedback are drawing a sketch, defining the region of interest in BlobWorld [14] or FOCUS (Fast Object Color-based Query System) [24], and moving images to define their similarity in [97]. Details of existing systems with different solutions for user interaction can be found in [116]. Regarding the user interaction, a question is *how to let the user interact with images displayed?*.

In summary, visualization of images should take all these issues into account leading to a general question:

Q1: *What is a good way of visualizing images for the purpose of obtaining useful feedback?*

The second step in the interactive stage, namely the feedback step, involves the learning capability of the system based on relevance feedback. An interactive CBIR should be designed such that it effectively obtains the most information from the user, i.e., *how to obtain user's relevance feedback?* and *how to update the query space with given feedback?*. In the interactive system in figure 1.1, after the user provides feedback, the system will learn the query space. One or more components of the query space are then iteratively updated by the relevance feedback **RF**. We have:

$$\{\mathcal{I}^t, \mathcal{F}^t, \mathcal{S}^t, \mathcal{Z}^t\} \xrightarrow{\mathbf{RF}^t} \{\mathcal{I}^{t+1}, \mathcal{F}^{t+1}, \mathcal{S}^{t+1}, \mathcal{Z}^{t+1}\}$$

where $\mathbf{RF}^t$ is the relevance feedback at the t^{th} iteration.

Techniques for updating the components in query space have been discussed in [104]. For changing the $\mathcal{I}$ components, a common technique is image collection filtering [117]. With the feature component $\mathcal{F}$, typical approaches are adjustment of the weights for feature weighting or feature selection [107, 54, 37], or changing the feature definition itself [76]. The $\mathcal{S}$ component is usually represented as a parameterized function, where parameters are optimized based on relevance feedback. Finally, the last component is changing the probability for the set of labels $\mathcal{Z}$ as {*relevant, irrelevant*} [67]. Some other review papers on learning relevance feedback recently have mentioned the use of support vector machines (SVM) as a basis of the learning strategy [23, 59, 128]. This statistical pattern recognition learning technique is currently the most effective in CBIR systems because of its effective performance [113, 62, 20, 125].

For updating the features, one of the issues discussed in [104] is the use of salient features such as salient points [101] or salient regions [80] which define the most important part of a search image. In related references, the saliency of features has not been considered from a context and user perspective. That leads to another question related to a definition of saliency of features:

Q2: *How to define the saliency of features (points, lines, or regions) such that it is context and user interpretation dependent?*

Although the four components are described independently, they all update the similarity between images. In other words, adjusting one or more components in the query space directly leads to the update of similarity $\mathcal{S}$. This makes similarity user and context dependent. As mentioned in the previous section, current research mainly stays on the primitive level, retrieving images by low-level features, while the user search focuses on the semantic and abstract levels. The question is:

Q3: *How to iteratively learn similarity on a higher level than the primitive level?*

Finally, after a system has been built, the task is to evaluate the performance of that system. Evaluation is usually either objective or subjective. Subjective evaluation has the advantage that it involves real users hence it is close to its actual use. However, at the same time it is very difficult to set up an experiment as it involves a number of parameters on the user groups, for instance age, gender, and expertise. Therefore, it is difficult to repeat an experiment. In an objective evaluation, these difficulties can be solved by simulating the search scenario with simulated user's and system's actions. Our final research question is:

Q4: *How to objectively evaluate performance of an integrated interactive CBIR system?*

1.3 Organization of the thesis

The following chapters of the thesis are organized as follows. In chapter 2 to answer question **Q1**, we analyze in detail the problems occurring when visualizing a large visual collection. From there, we establish a visualization framework satisfying three proposed requirements: overview, visibility, and structure preservation. As stated in the previous section the essence is treating visualization and interaction as a whole. We focus on the integration of visualization and feedback learning in chapter 3 using the visualization framework proposed in chapter 2. In such an integrated system, we present the optimization of the system's performance with simulated search scenarios (question **Q4**). In chapter 4, we introduce a new approach to learn dissimilarity for interactive search in content based image retrieval (question **Q3**). Similarity definition is learned in dissimilarity space instead of feature spaces in existing approaches. Finally, to answer question **Q2**, an overview of salient details extraction in CBIR such as points, lines, and regions is presented in chapter 5.

Chapter 2

Interactive access to large image collections using similarity based visualization

2.1 Introduction

Through the development of multimedia technologies and the availability of cheap digital cameras, the size of image collections is growing tremendously. Collections range from consumer pictures, to professional archives such as photo stocks of press agencies, museum archives, and to scientific pictures in medicine, astronomy, or biology. Hence, large image collections are common everywhere.

The use of image collections is domain dependent. For consumer pictures, a task is finding all pictures taken of a family member in a certain event. In medicine, a doctor wants to search images similar to a given one, to diagnose a disease. In general, when working with image collections the main task is searching relevant pictures in the collection.

When the collection contains a couple of hundred images, one can find all relevant images by visually inspecting the whole collection. If the size of the collection increases to thousands or even millions of images, one needs efficient methods for searching and browsing those collections. To that end, we should note the *"semantic gap"* between the system's automatic indexing capability and the user's conceptual interpretation of the data [104]. Interaction and visualization are needed to bring the system perspective and user perspective together.

The appropriate visualization and interaction method are task dependent. For understanding the structure of the collection, visualization should allow interaction with clusters, local structures and outliers [17]. For browsing, some form of rapid serial visual presentation can be appropriate [25], or network relations between images

This chapter is accepted for publication in the Journal of Visual Language and Computing [78] can be visualized for easy navigation [91]. Searching requires the possibility to see overviews as well as being able to interactively zoom-in on the information [1]. Finally, annotation requires that images that should receive the same annotation are close to each other so they can be annotated with one interaction step [78]. The examples indicate the need for generic tools which can be combined to support each of these tasks.

In content based image retrieval literature, few systems use visualization as a tool for exploring the collection. In most query by example based systems, a randomly selected set of images from the collection is displayed in a 2D grid [104, 116, 122]. In this way, the user does not get an overview of the collection. As a consequence, much time is wasted in considering non-relevant images and relevant images are easily missed. Visualization offers the opportunity to guide the user in her exploration.

To guide a user, the structure of the collection is of prime importance. Thus, focus must not be on the images alone, but especially on the relations between the images. These relations are captured by a *similarity* function. Similarity is a very generic notion and can be based on *features* computed from the image content, free text descriptions, or semantic annotations. Recently, advanced systems have been developed for browsing images based on similarity [11, 17, 22, 57, 74, 91, 97, 118]. In these systems the similarity, and thus structure, based visualization of the collection is the guide for the user.

None of the above methods explicitly addresses the problems occurring when visualizing large visual collections. The most important problem is the limited display size, not allowing to show the whole collection as images on the screen. Some systems have made an effort to relief this limitation by showing the whole collection as a point set [17]. Once the user selects a point, the corresponding image is displayed [22]. To see the visual structure of the collection, many images need to be shown simultaneously. A problem here is visibility. If small thumbnails are used, the images cannot be understood by the user. Larger thumbnails lead to substantial overlap of the images on the screen. Most of the existing systems do not take this problem into account. Exceptions are [71] and [87], but their treatment of the problem is rather ad-hoc. A final issue to consider is the difference between the high dimensional feature space, typically of dimension 50 or more, and the 2-dimensional display. A mapping between the two is needed. For instance, in [71] PCA (Principal Component Analysis) is employed, whereas [91] uses MDS (Multi-Dimensional Scaling). Inevitably in the projection, information on the structure of the collection is lost. In summary, there are a number of problems when visualizing large image collections. In this paper, we make them explicit.

The advanced visualization tools mentioned above make the development of systems more complex. The question arises whether it is worth the effort. This requires extensive evaluation. Evaluating the usability of a system can be done subjectively or objectively [47]. To evaluate subjectively, real users judge the performance of the system. In this direction, only Rodden [87, 86], performed evaluations of similarity based visualization. However, the use of subjective evaluation is quite expensive and cannot be repeated easily. When the task is well defined some aspects of usability evaluation can be automated [47]. Such objective evaluation has several advantages such as reducing the cost of evaluation, reducing the need for evaluation expertise and increasing the coverage of evaluated features [47]. It effectively allows to decompose the evaluation into evaluating the methodology and the design of the interface and its usability. We view objective evaluation as an important step in the development of complex systems.

Various techniques that aim at providing this type of evaluation can be found in [47] such as testing, inspection, inquiry, analytical modelling, and simulation. Simulation is most suited for our case as it reports the performance of the system and user's interactive actions. This method uses models of the user and interface design to simulate a user interacting with the system. We employ this method making the scenario of use and criteria for success of our new approach explicit. From there, we develop a method to simulate the user actions.

This chapter is organized as follows. In section 2.2, we analyze the general requirements for visualizing large image collections. A set of requirements is proposed. Solutions for each requirement are then given in section 2.3. From there, cost functions are established for optimal visualization in section 2.4. A visualization system to illustrate the proposed theory with a collection of 10000 Corel images and experiments based on a user simulation system are presented in section 2.5.

2.2 Problem analysis

In this section, we analyze in detail the problems occurring when visualizing a large visual collection. From there requirements for a generic system are defined.

First of all, we give some notations and conventions used throughout the chapter. An *image collection* is a set of N images, where we assume $N \gg 1000$. Each image I in the collection is represented by a feature vector f_I for example a color or texture histogram. The similarity of two images I, J is denoted by $\mathcal{S}_{IJ}$. Often we will rather use a distance, or dissimilarity measure D_{IJ}, which is zero whenever the images have exactly the same feature vector, and larger than zero otherwise. Thus, each image in the collection corresponds to a point in a high-dimensional feature space. The image collection, the corresponding feature vectors, and the distances between them define the *information space*. Figure 2.1 shows a simple example of a 3D information space.

Apart from the data preparation step, the general scheme of a visualization system contains three steps. First, in the projection step, the information space is projected to the *visualization space* yielding a 2-dimensional view. The image collection now corresponds to a set of positions $\{\vec{y}_i\}_{i=1}^{N}$ in the 2D space. In the selection step, the system selects a subset of images to display. Finally, the interaction step involves the visualization of the selected set and user feedback. The overall scheme is illustrated in figure 2.2.

An obvious issue in visualizing a large collection is the *limited display size* of the visualization space. This dictates that only a restricted number of images can be shown simultaneously since the goal is to show the content of the images. Randomly selecting images is certainly not a good approach as it does not capture the distribution of images in the collection [82]. Therefore, the first requirement is:

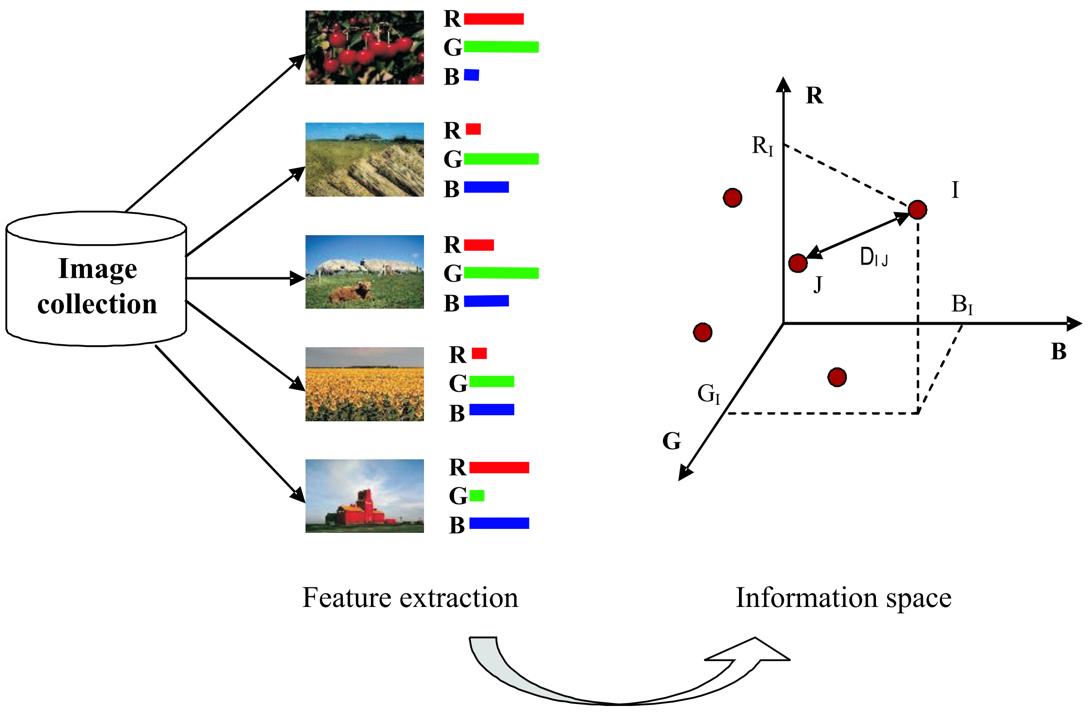


Figure 2.1: An example of a 3-dimensional information space based on the amount of red, green and blue in an image. Note that in practice the dimensionality is much higher.

Overview requirement *The visualization should give a faithful overview of the distribution of images in the collection.*

Secondly, the information space often exhibits considerable structure in the form of clusters, low-dimensional manifolds, and outliers. Many examples are found in literature, e.g., in [83, 84]. Figure 2.3 shows examples of different structures found in a set of images from a video sequence, similar to those in [84]. Another example of structure is present in a set of images of the same object in different conditions such as different light source, viewpoint, and/or orientation (see figure 2.4). This structure should be preserved in the visualization. Thus we have the

Structure preservation requirement *The relations between images should be preserved in the projection of the information space to the visualization space.*

Finally, it must be stressed that the image itself provides important information for the user, but only when it is large enough to be understood. Now, when a set of images is displayed, they tend to overlap each other partially or fully [71, 87]. The overlap between images influences the quality of a visualization tool greatly. Due to overlap important images can be missed. Therefore, the overlap among them should be reduced as much as possible, leading to the following requirement.

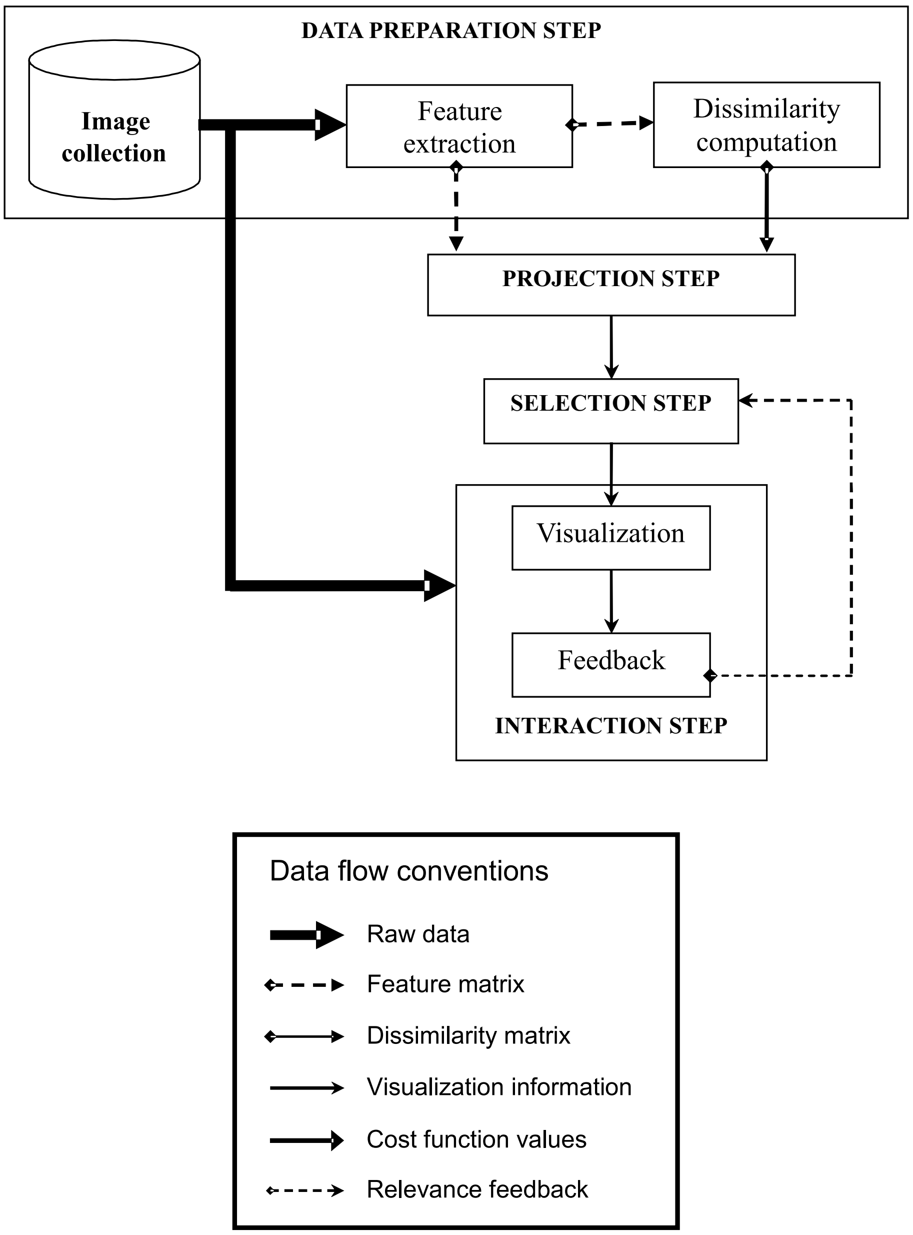
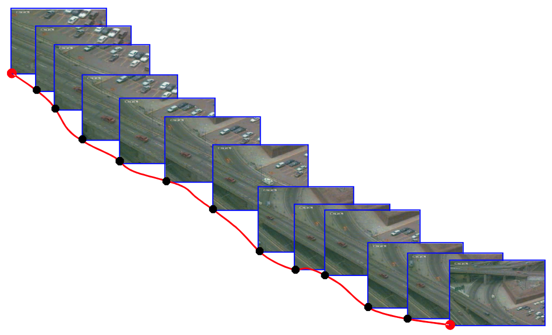


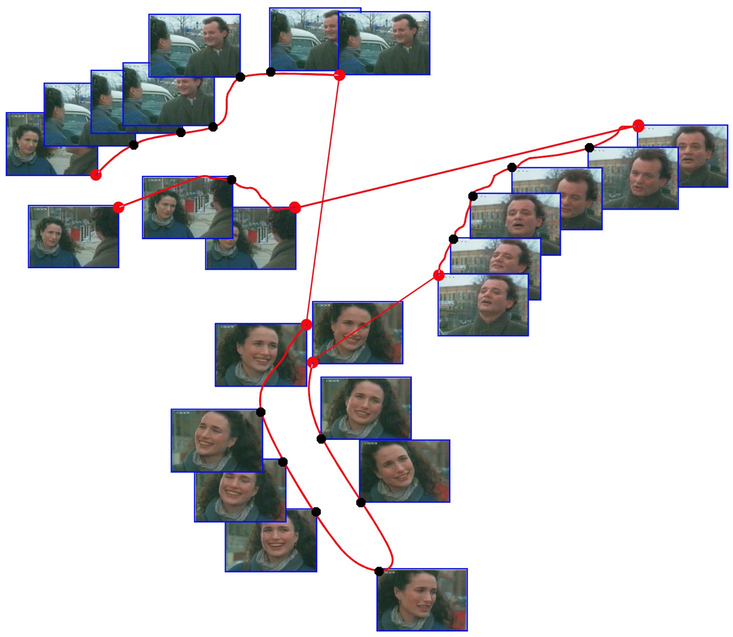
Figure 2.2: General scheme of an image collection visualization system.

Visibility requirement *All displayed images should be visible to the extent that the user can understand the content of each image.*

So we have three general requirements: *overview*, *structure preservation*, and *visibility*. Those requirements are not independent. To increase the visibility, images should be spread out. This has a negative effect on the preservation of structure. Furthermore, more representatives yield better overviews, but the visibility decreases



(a)



(b)

Figure 2.3: (a) Linear structure of a sequence showing a car riding in a street. (b) Nonlinear structure of a video sequence capturing the conversation between two people.

because overlap becomes more likely. The relations among the three requirements are illustrated in figure 2.5. To generate appropriate visualizations, we need means to balance the different requirements. In the next sections, we find appropriate cost functions for each of the requirements, which are then combined and jointly optimized.

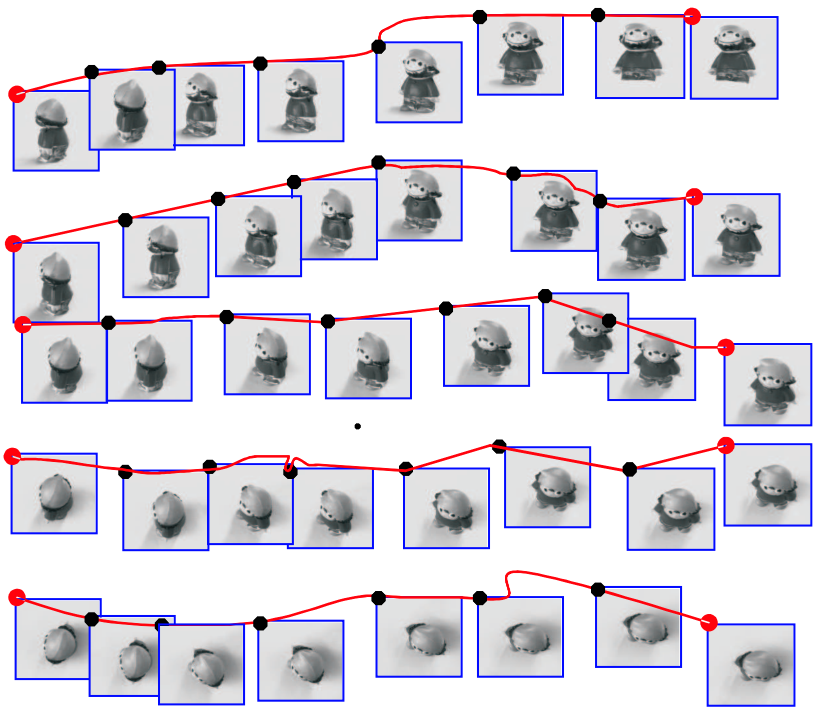


Figure 2.4: A set of images of a toy taken from different orientations [83]. When the viewpoint changes in one direction, the structure is linear.

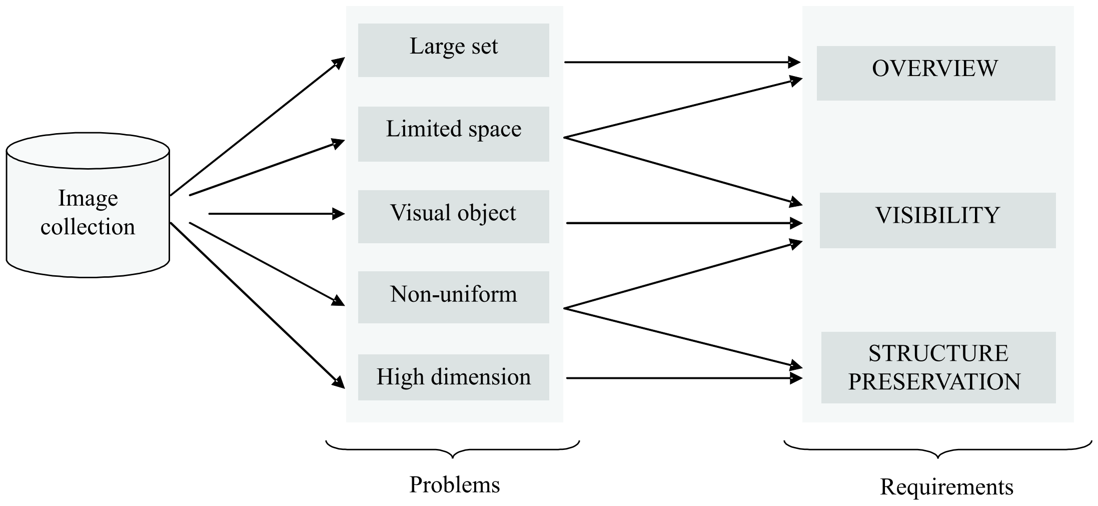


Figure 2.5: Problems and requirements for visualization of large image collections.

2.3 Projection and selection methods

In this section, we consider different methods for projecting the information space to visualization space, as well as methods for selecting images for the overview.

2.3.1 From information space to visualization

Well-known techniques used in existing systems are principal component analysis (PCA) [71] and multi-dimensional scaling (MDS) [87, 91]. These methods are assumed that the structure of the information space is linear. If the information space contains a non-linear structure they will not satisfy the structure preservation requirement. This is illustrated in figure 2.6.

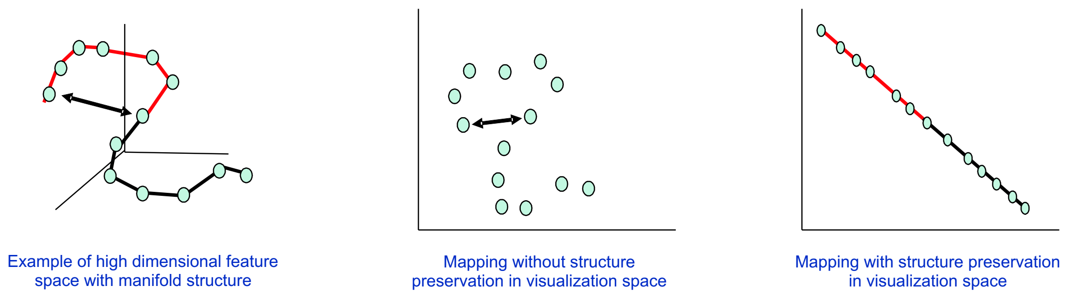

Example of high dimensional feature space with manifold structure
Mapping without structure preservation in visualization space
Mapping with structure preservation in visualization space

Figure 2.6: Illustration of non-linear mapping.

This issue is considered in new techniques, known as non-linear embedding algorithms, namely isometric mapping (ISOMAP) [111], local linear embedding (LLE) [89], and more recently stochastic neighbor embedding (SNE) [43]. The proposed mapping algorithms are able to preserve the real structure of the data and perform better than PCA and MDS. We consider non-linear techniques only.

ISOMAP was introduced in 2000 [111]. Instead of directly computing the distance between points, the authors use graph-based distance computation aiming to measure the distance along local structures. Their algorithm contains three main steps. First, the algorithm builds the neighborhood graph using t-nearest neighbors (the t closest points) or ϵ-nearest neighbors (all points with distance to the point less than ϵ). The second step uses Dijkstra's algorithm to find shortest paths between every pair of points in the graph. The distance for each pair is then assigned the length of this shortest path. After the distances are recomputed, MDS is applied to the new distance matrix.

A different approach is SNE [43], a probabilistic projection method. This method first computes the probabilities that two points take each other as neighbors, assuming a Gaussian distribution, in both the high and the 2-dimensional space. The method then tries to match the two probability distributions. Hence, it provides preservation of local geometric structure and also keeps points which are distant in the high dimensional information space distant in the visualization space. The working principle of the SNE can be briefly described as follows. Let $\mathcal{P} = P_{IJ}$ denote the probability that an image I would pick J as its neighbor in the high-dimensional space. Under the Gaussian distribution assumption, P_{IJ} is given by:

$$P_{IJ} = \frac{\exp(-D_{IJ}^2)}{\sum\limits_{L \neq I} \exp(-D_{IL}^2)}. \tag{2.1}$$

with $P_{II} = 0$. Note that this measure in general is asymmetric: $P_{IJ} \neq P_{JI}$.

In the 2-dimensional space, SNE initializes $\{\vec{y}_i\}_{i=1}^{N}$ at random positions. The induced probability $\mathcal{Q} = \{Q_{IJ}\}$ is then calculated for every pair of images:

$$Q_{IJ} = \frac{\exp\left(-||\vec{y}_i - \vec{y}_j||^2\right)}{\sum\limits_{k \neq i} \exp(-||\vec{y}_i - \vec{y}_k||^2}, \quad Q_{II} = 0. \tag{2.2}$$

To measure the distance between these two distributions $\mathcal{P}$ and $\mathcal{Q}$, the Kullback-Leibler distance is used. This asymmetric distance is commonly used in measuring a natural distance from a “true” probability distribution, $\mathcal{P}$, to a “target” probability distribution, $\mathcal{Q}$.

$$D_{\mathcal{P}\mathcal{Q}} = \sum_I \sum_J P_{IJ} \log \frac{P_{IJ}}{Q_{IJ}}.$$

The algorithm then finds the optimal positions $\{\vec{y}_i\}$ by minimizing $D_{\mathcal{P}\mathcal{Q}}$.

SNE uses direct distance computation among points, hence it can benefit from replacing this distance by the graph based distance from ISOMAP. We therefore propose ISOSNE, a combination of ISOMAP and SNE.

Since SNE uses gradient descent methods it requires substantial computation time, especially when the size of the data reaches several thousand images. LLE [89] can be viewed as an approximation to SNE which is much faster to compute. This method first constructs the t-nearest neighborhood graph. Then, for each point $\vec{y}_i$ it computes the weights w_{ij} that optimally reconstruct $\vec{y}_i$ from its neighbors by minimizing the cost function $\sum_i ||\vec{y}_i - \sum_j w_{ij}\vec{y}_j||^2$.

In the reference, the LLE distance is computed directly, but we can also recompute distance like in ISOSNE. We denote this combination by ISOLLE.

2.3.2 Selection methods

To select a representative set from the collection to be used in the overview, a common method is dividing the collection into a number of groups. One image from each group is then selected as representative. This requires finding clusters based on the distance matrix.

A comprehensive overview of different clustering techniques is presented in [50]. Comparisons among different methods are given in [8, 28, 40, 126]. They conclude that the k-means algorithm is one of the most successful methods because of its simplicity in implementation and its linear time complexity. We therefore employ k-means to select images for the overview in our system.

The k-means algorithm is applied after projection of the information space to visualization space. We initialize the k center points at random positions. The reassignment of points to the nearest center is repeated until the clustering satisfies certain requirements, or when the maximum number of iterations is reached. The image corresponding to the point nearest to the cluster center $\{m\}$ is selected as the representative of that cluster.

2.4 Balancing the requirements

As mentioned in the problem analysis, the three requirements are dependent on one another. In order to balance them, we first develop a cost function for each requirement. For the first two requirements, existing functions are employed, while for the last requirement, we introduce our own cost function as it has not yet been investigated in literature. From there, balancing functions are introduced and applied to realize our final visualization scheme.

2.4.1 Cost functions

2.4.1.1. Structure preservation cost function

To preserve the nonlinear structure of a collection, the projection algorithm should at least map the neighbors of an image in the information space in such a way that they are neighbors in the visualization space also, which is less strict than preserving distance. In addition, users are also using comparisons of distance rather than absolute distance. The cost function used in SNE is therefore a good option for evaluating different projections. It is given by the Kullback-Leibler distance between the two distributions P and Q defined in section 2.3.1:

$$\mathcal{C}_{\mathrm{S}} = \sum_{I} \sum_{J} P_{IJ} \log \frac{P_{IJ}}{Q_{IJ}}. \tag{2.3}$$

Clearly, the lower this cost, the better the projection has preserved the relations between neighbors.

2.4.1.2. Overview cost function

When the images are assigned to their corresponding cluster, we need to find a cost function to evaluate the overview provided by the representative images of each cluster. Clearly this depends on the number of clusters k asked for, and how well the representatives cover the whole data set. A commonly used measure for the quality of the clustering is the modified Hubert statistic [28] ranging from 0 to 1, where the higher the value the better the clustering. So our overview cost function is:

$$\mathcal{C}_{\mathrm{O}} = 1 - \frac{r - M_p M_c}{\sigma_p \sigma_c} \tag{2.4}$$

where

$$
\begin{aligned}
r &= (1/M)\sum\sum D_{IJ}d(m_I,m_J),\\
M_p &= (1/M)\sum\sum D_{IJ},\\
M_c &= (1/M)\sum\sum d(m_I,m_J),\\
\sigma_p^2 &= (1/M)\sum\sum D_{IJ}^2 - M_p^2,\\
\sigma_c^2 &= (1/M)\sum\sum d^2(m_I,m_J) - M_c^2,\\
M &= k(k-1)/2,
\end{aligned}
$$

where m_I is the center of the cluster containing image I and $d(m_I, m_J)$ is the distance between two cluster centers.

2.4.1.3. Visibility cost function

So far the solution for the requirements can be based on existing measures, mainly because they apply equally well to point-sets.

The major issue in visibility is the overlap of the images displayed. This depends on the number and size of images displayed. A few small images will not overlap, but when 1000 large images are displayed there is always overlap. It also depends on the structure of the data. If images are clustered in information space, the structure preservation requirement will dictate that a lot of overlap is present in visualization space.

To define a cost for the overlap among images displayed, we consider the overlap of two images, and combine them over all pairs. Finding the overlap between the rectangles defining two images is not difficult, however, many different cases have to be distinguished. To develop an analytical function, we make the simplifying assumption that all images have width w and height h, with $w = h$. We then represent an image as a circle, with radius $R = \frac{w}{2}$ (see figure 2.7a). This is a reasonable approximation as the area of the circle covers $\frac{\pi}{4} \simeq 80\%$ of the image area. So, if the two images overlap outside the area of the circle and inside the image area (see figure 2.7b), the viewable area of the image is larger than 75% of the image area, which is sufficient for visibility.

The overlap between two circles i and j is given by:

$$
\mathcal{O}_{ij} = \begin{cases} R^2\left(2\arccos\left(\frac{d_{ij}}{2R}\right) - \sin\left(2\arccos\left(\frac{d_{ij}}{2R}\right)\right)\right) & \text{, if } d_{ij} < 2R,\\ 0 & \text{, otherwise.}\end{cases} \tag{2.5}
$$

where d_{ij} is the Euclidean distance between the center points of the images I and J.

Hence, if the number of displayed images is n, the total visibility cost is defined by:

$$
\mathcal{C}_{\mathrm{V}} = \frac{1}{n(n-1)}\sum_{i}^{n}\sum_{j\neq i}^{n}\frac{\mathcal{O}_{ij}}{\Pi R^2}. \tag{2.6}
$$

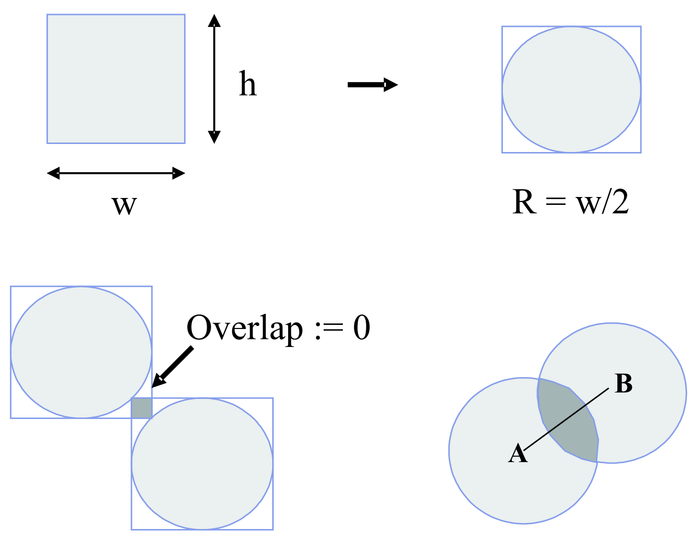


Figure 2.7: Image with its corresponding enclosed circle and the overlap area between two images

Because of the limited size of the visualization space, there is a maximum number of images which can be displayed while satisfying the visibility requirement. Let us assume an image is called viewable if its visible area occupies $t\%$ of the image and the visualization space has size $\mathcal{W}$ and $\mathcal{H}$. Then, the maximum number of displayed images with $t\%$ visible is $\frac{\mathcal{H}\times\mathcal{W}}{h\times w\times t}$. This yields a strong constraint on the design of the visualization method.

2.4.2 Balancing functions

There are two main relations among the three requirements. First, the relation between overview and visibility, which is affected by the number of representative images. The more images, the better the overview, but visibility is reduced. To balance these two requirements we take a linear combination of their cost functions Eq.(2.4) and Eq.(2.6):

$$\mathcal{C}_1 = \lambda_1\mathcal{C}_{\mathrm{O}} + (1-\lambda_1)\mathcal{C}_{\mathrm{V}} \tag{2.7}$$

where $0 \leq \lambda_1 \leq 1$. Now, for given λ_1 we find the n where $\mathcal{C}_1(n)$ reaches its maximum value:

$$n_{\mathrm{opt}} = \underset{n\in[2..n_{\mathrm{max}}]}{\operatorname{argmax}}\ \mathcal{C}_1(n). \tag{2.8}$$

Since this step is done offline, we use a brute-force approach computing $\mathcal{C}_{\mathrm{O}}$ and $\mathcal{C}_{\mathrm{V}}$ for n from 2 to n_{max}.

The second relation is between the visibility and the structure preservation requirement. The more visibility, in other words less overlap, the less structure is preserved. We again use a linear combination of Eq.(2.3) and Eq.(2.6). The problem boils down to finding the best optimal positions of images in visualization space where the joint cost of overlap and disobeying structure preservation is minimal:

$$\vec{y}_{\text{opt}} = \min_{\vec{y}} \mathcal{C}_2(\vec{y}).$$

with

$$\mathcal{C}_2(y) = \lambda_2 \mathcal{C}_{\text{S}}(y) + (1 - \lambda_2)\mathcal{C}_{\text{V}}(y) \tag{2.9}$$

where $0 \leq \lambda_2 \leq 1$.

To find the optimum, gradient descent is applied. This requires to compute how the cost function changes when images are moved away from their positions, i.e., the derivative of $\mathcal{C}_2$ with respect to the $\vec{y_i}$:

$$\frac{\partial \mathcal{C}_2}{\partial \vec{y_i}} = \lambda_2 \frac{\partial \mathcal{C}_S}{\partial \vec{y_i}} + (1 - \lambda_2)\frac{\partial \mathcal{C}_V}{\partial \vec{y_i}}.$$

The differentiation of $\mathcal{C}_{\text{V}}$ is given in [43]:

$$\frac{\partial \mathcal{C}_S}{\partial \vec{y_i}} = 2 \sum_J (\vec{y_i} - \vec{y_j})(P_{IJ} - Q_{IJ} + P_{JI} - Q_{JI}).$$

Given Eq.(2.6), we derive:

$$\frac{\partial \mathcal{C}_{\text{V}}}{\partial \vec{y_i}} = \frac{1}{k(k-1)\Pi R^2} \sum_J (\vec{y_i} - \vec{y_j}) \frac{-\sqrt{4R^2 - d_{ij}^2}}{d_{ij}}.$$

2.4.3 Final visualization scheme

Up to this point, we have analyzed the visualization requirements and how to optimize them using balancing functions. We propose a new visualization scheme conform the general scheme in figure 2.2 (see figure 2.8).

The data preparation step, the projection step and a part of the selection step can be prepared beforehand. Therefore, we call these steps the offline process. In this stage, features are first extracted for all images in the collection and a dissimilarity matrix is computed. Next, a projection from the information space to the visualization space is applied. After that, k-means is applied with n ranging from 2 to n_{max} (we select $n_{\text{max}} = 300$). We then calculate $\mathcal{C}_{\text{O}}$ and $\mathcal{C}_{\text{V}}$. From there, the balancing function in Eq.(2.7) with given λ_1 returns the optimal number of images to be displayed in each iteration. The optimal clustering is kept for subsequent steps.

The other part of the selection step, involving the selection of the representative set, and the interaction step are part of the interactive process. In the first iteration, representatives are the cluster centers. The visualization step computes the arrangement of representative images on the screen according to the second and the third requirement. This means that the balancing function $\mathcal{C}_2$ (Eq.(2.9)) is optimized to find positions for all displayed images. These positions assure that relations between them are preserved as much as possible and the content of images are sufficiently

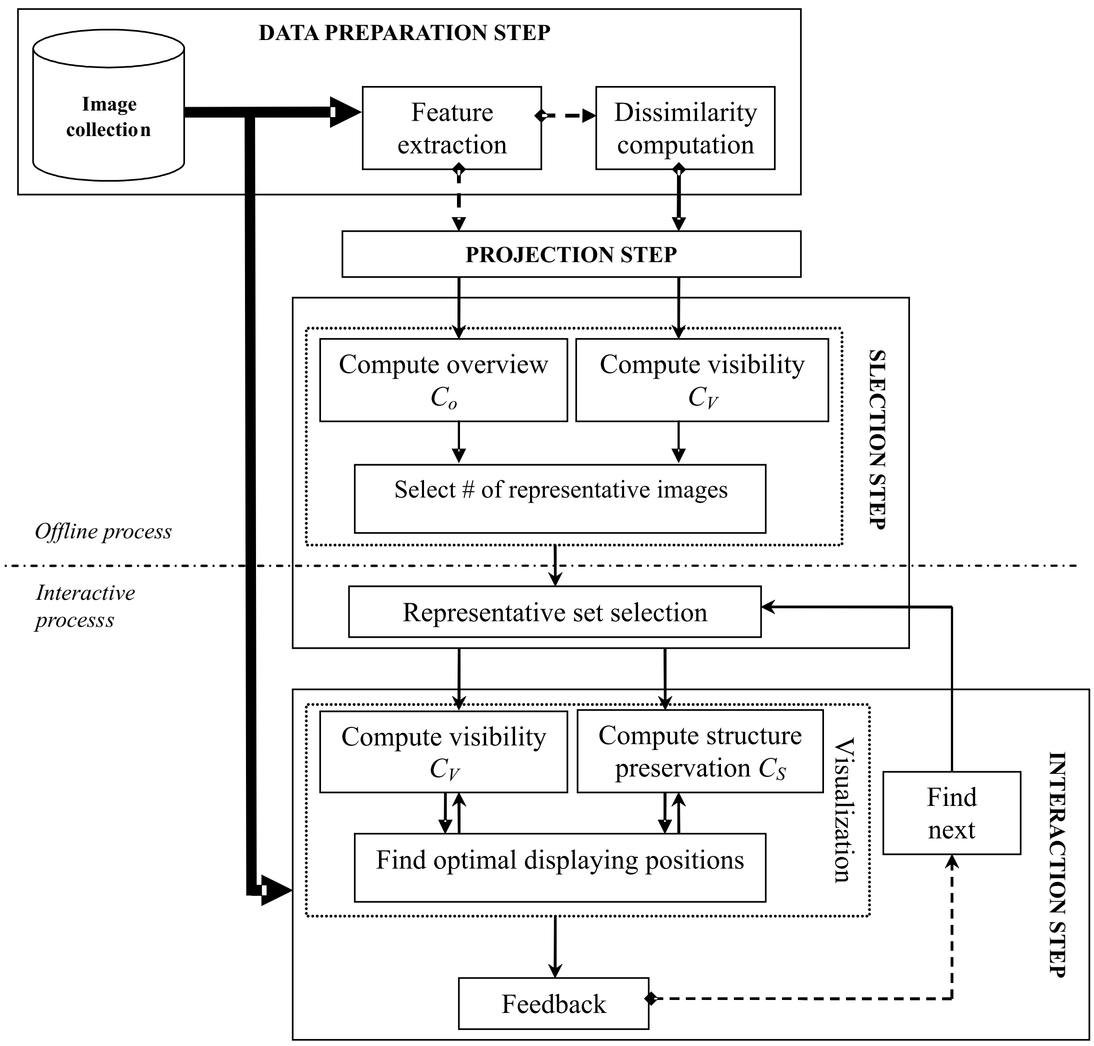


Figure 2.8: Scheme of an image collection visualization system, where we combine the general scheme of figure 2.2 with balancing functions.

visible. After the *find next* step, the system selects the set of images to display in the next iteration. The selection contains images which have not been displayed before and are closest to the corresponding center points.

2.5 Experiments

In this section, we present experiments to validate the different components of the visualization system. The preparation of the offline stage contains the data selection, feature selection, and dissimilarity computation. Then, we compare different mapping algorithms for the projection step. A system to demonstrate the scheme is presented in section 4.3. Finally, we apply our system to a search task, comparing our approach with a more traditional visualization.

2.5.1 Data collection

We select a collection of $N = 10000$ Corel images. There are 100 predefined categories, where each category contains 100 images. The existing categorization will be used as ground-truth for later evaluation. The images depict different scenes and objects. Computing the dissimilarities between images strongly depends on the features and dissimilarity function chosen. Because of the large variety in images in this particular collection, no features and/or dissimilarity functions exist which correctly classify images. As in practice this is also the case and we are focussing on the interaction process, we employ simple global color histograms. In particular, HS (Hue and Saturation) and L*a*b (L* defines lightness, a* denotes red/green value, and b* the yellow/blue value). We compute the histogram using 32 bins for each color channel, this means that for HS we get a histogram of 64 dimensions, and 96 in the case of L*a*b. Histogram intersection and Euclidean distance are used as similarity functions for HS and L*a*b histogram, respectively.

2.5.2 Comparison of projection methods

As mentioned above, we concentrate on the nonlinear dimension reduction methods ISOMAP, LLE, SNE, ISOSNE, and ISOLLE. MDS is used as a baseline.

	MDS	ISOMAP	SNE	LLE	ISOSNE	ISOLLE
$\mathcal{C}_\mathrm{S}$ with HS	0.008653	0.006785	0.000247	0.000252	0.000076	0.000225
$\mathcal{C}_\mathrm{S}$ with Lab	0.043584	0.004489	0.000089	0.000202	0.000063	0.000190
$\mathcal{T}$(hour)	~1.5	~1.5	~10	~2.0	~10	~2.0

Table 2.1: Results for MDS, ISOMAP, SNE, LLE, ISOSNE, and ISOLLE in preserving original structure when mapping data from high dimensional feature space to 2d visualization space. The first row gives results of mapping from HS feature space. The second row is for L*a*b feature space. The last row shows the computation time of each method.

SNE and ISOSNE are expected to perform best as we have chosen Eq.(2.3) as the evaluation criterion. However, these two use gradient descent in finding the optimal solutions hence they have a long processing time. LLE and ISOLLE are fast as they use approximations to find the embedding. Hence, in the comparison, we also take into account time complexity. All the experiments are run on the same PC PenIV, 2Ghz. The results are in table 2.1. In both experiments, MDS yields the worst performance. SNE and ISOSNE outperform the others, but require 10 times longer processing time than LLE which still has good performance. So when computations are done offline, SNE based methods are preferred. In practical situations LLE can often be employed.

2.5.3 A system demonstration

In our system the projection is computed once in the offline stages so we choose ISOSNE. Figure 2.9(a) and 2.9(b) show the mapping results of ISOSNE on the given collection using the HS and L*a*b feature space, respectively.

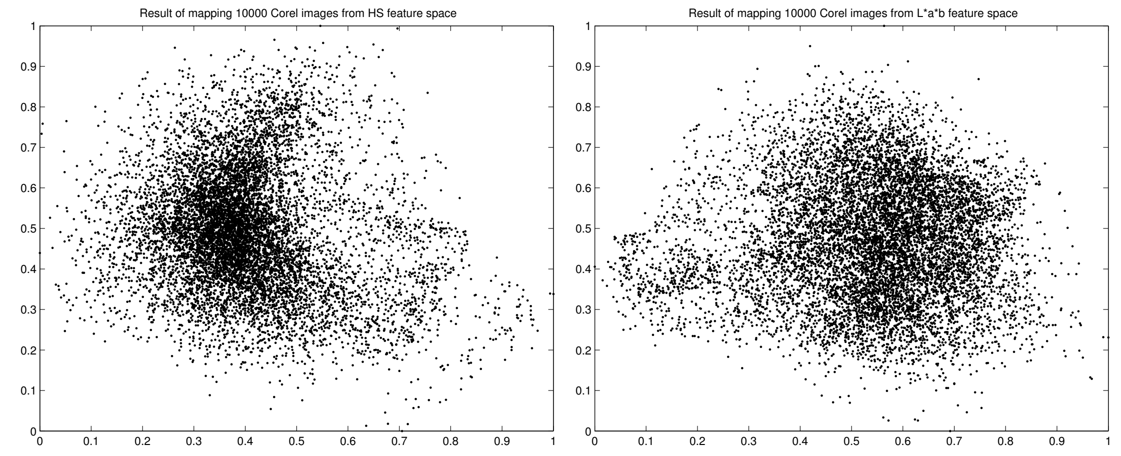


Figure 2.9: Result of projecting 10000 Corel images from 64 dimensional HS feature space and 96 dimensional L*a*b feature space to the 2-dimensional visualization space.

For further demonstrating the system, we use the HS features. First, to find out the optimal number of clusters the collection should be divided into, we apply Eq.(2.7) with λ equal to 0.5 and $n \in [2..300]$. From Eq.(2.8), we find $n_{\text{opt}} = 55$, so the collection is divided into 55 clusters.

Note that, the above optimal number of images are with no overlap reduction. This means that in the visualization space, with $n_{\text{opt}} = 55$, all displayed images satisfy the visibility requirement. As in practice, one may prefer to have more images on the screen, the value of n_{opt} is used as a threshold. If there is a higher number of displayed images, we will get a better overview, but the visibility is reduced. Then we need to consider the overlap problem. Applying Eq.(2.9), one can increase the number of displayed images.

In the subsequent online process, in the first iteration of the visualization, images closest to the center points are selected for display. Each image represents one cluster. The second balancing function $\mathcal{C}_2$ with $\lambda_2 = 0.5$ is used to find the optimal positions for the displayed set. This process is repeated to display the subsequent sets of images.

In selecting the optimal positions, not only the number of displayed images, but also the value of λ_2 can affect the result. With λ_2 equal to 1.0, only structure is preserved. When λ_2 goes to 0, images are spread out loosing much of the structure. An example is shown in figure 2.10. We have to notice that there is another factor affecting the selection of these parameters, which is size of the display space. We assume the visualization space is equal to the size of a standard computer screen.

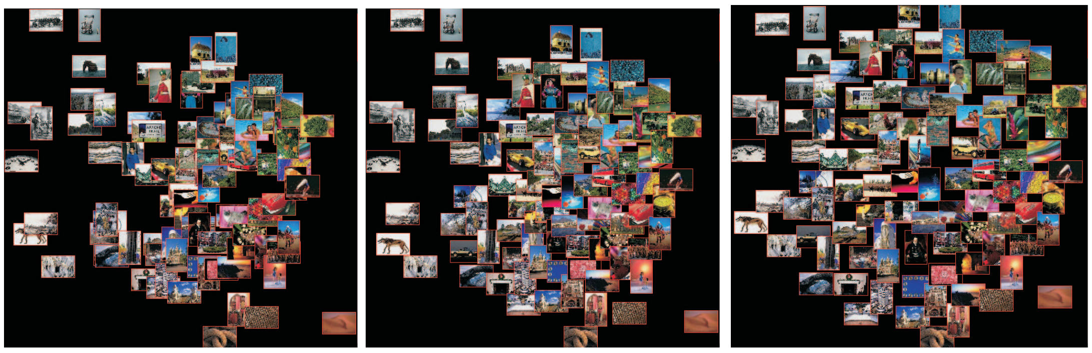


Figure 2.10: Example of displaying 100 images with $\lambda_2 = 1$, 0.5, and 0, respectively.

We, therefore, experiment with different numbers of images, as well as different values of λ_2 to see the effect of those two parameters on the visibility and the structure preservation requirements. In order to do that, the k-means clustering is applied with $n = 50, 100, 150, 200$. After the clustering, each time a set of n representative images is displayed. The balancing function in Eq.(2.9) is applied with λ_2 ranging from 0 to 1. We then calculate for the currently displayed set the percentage of images visible for at least t%, with $t = \{25, 50, 75, 100\}$. The cost $\mathcal{C}_S$ is also computed for each case.

Figure 2.11 shows the results for different n and t. The figures clearly illustrate the relation between number of images, λ_2, structure preservation and the visibility. With a small number of images, the system easily finds a solution for Eq.2.9. For example, in case of 50 images, without any constraint on visibility ($\lambda_2 = 1$), the percentage of images 75% visible is very close to 100%. With $\lambda_2 = 0.5$ all images are visible while structure is well preserved. In contrast, with 200 images, even when $\lambda_2 = 0$ meaning no structure preservation, the total percentage of 50% visible images is not increased much. This is to be expected from the discussion in 2.4.1. In fact, too few images will increase the browsing and exploration time through the image collection. From the above, selecting 100 images with $\lambda_2 = 0.9$ is a good option.

2.5.4 Similarity based vs. 2D sequential visualization

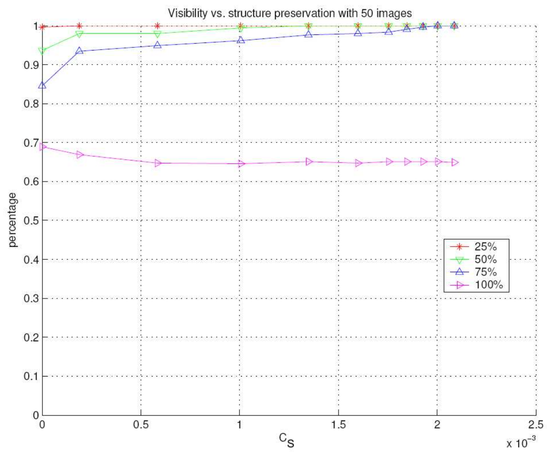
In this section, we compare traditional 2D sequential visualization with our 2D similarity based visualization. We do so by setting up an explicit scenario of use and then we simulate the user actions.

2.5.4.1. Scenario setup and evaluation criteria

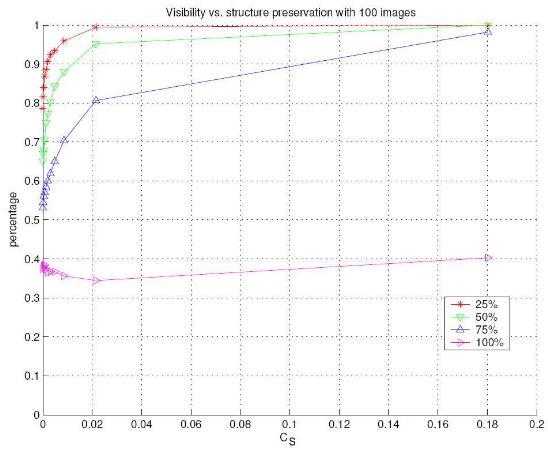
The scenario we use is full *database annotation.*

Database annotation *is assigning all images in the database to their corresponding category.*

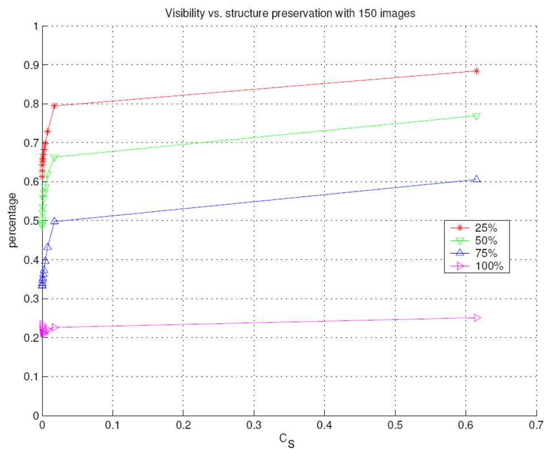
Manual database annotation is very time consuming and labor expensive, especially when the size of the data is getting larger. Now let us see how this scenario is



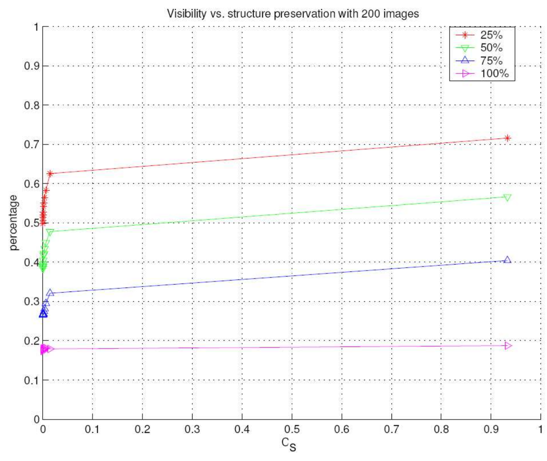
(a)



(b)



(c)



(d)

Figure 2.11: Examples of visibility versus structure preservation with different number of images, and different values of λ_2.

performed in our system. In visualization space, a set of n images, where each image represents one cluster, is displayed to the user. He/she selects an image and then goes inside the corresponding cluster. Images in that cluster are then annotated. One user action *is an interaction of the user to annotate one image a a group of images.* We finally obtain the total number of *actions* to annotate the whole database, the so called *annotation effort.*

Annotation effort *is the total number of user actions needed to perform the task.*

In sequential visualization, displayed images are arranged on a grid with no relations between them taken into account. So a user action is needed for each separate image on the screen. This means that annotation effort equals the size of the collection.

In 2D similarity based visualization, similar images are located close to each other. In the ideal case, all images in a cluster belong to the same category. In practice the cluster contains images from different categories, therefore the user draws a rectangle around each set of images belonging to the same category to annotate them. Figure 2.12 shows an illustration of the process.

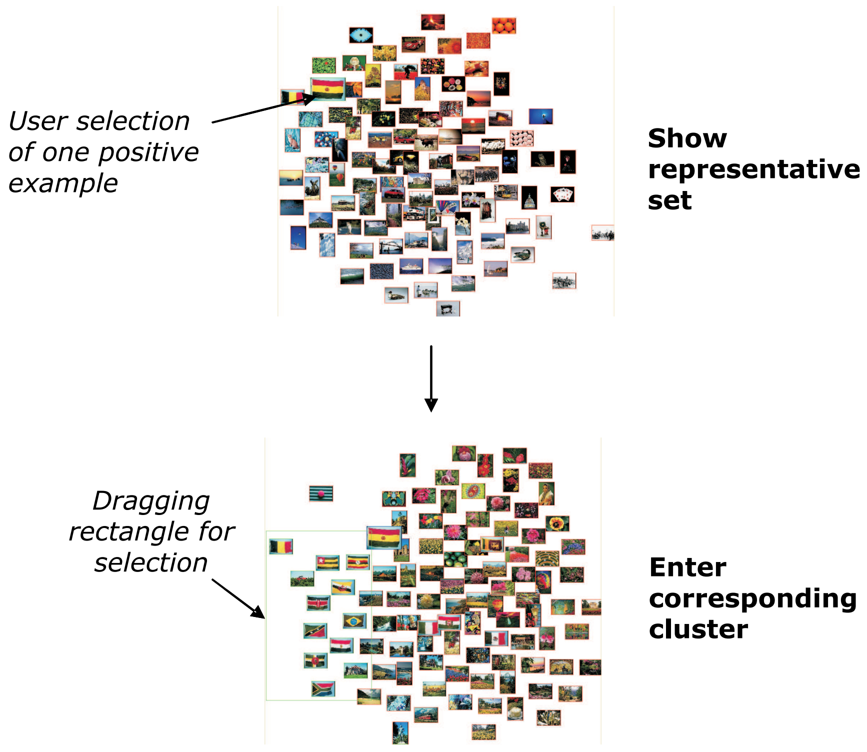

User selection of one positive example
Show representative set
Dragging rectangle for selection
Enter corresponding cluster

Figure 2.12: An example to illustrate user actions during annotation process.

As we propose a user simulation scheme to implement a quantitative evaluation, we need to mimic the above defined user action. We do so by implementing an algorithm finding the number of rectangles needed to assign all images displayed to the appropriate category. Optimal search for the minimum number of rectangles can be employed. However, in reality, the user often does not draw an optimal number of rectangles to cover all images of interest. Hence, a simpler greedy search appropriately fits the user action. The pseudo code is as follows.

```
Rectangle-search(an image set M)
   For each element m in M
       If (m has not been annotated as positive)
           X = sort(neighbors(m)) on distance to m;
           R = draw_rectangle(m);
           For each element x in X, where category(x)=category(m)
```

```
store(R);
R = draw_rectangle({R, m}); //increase size of the rectangle.
If R contains y, where category(y)!=category(m)
    R = store(R);
    break;
else
    annotate(x);
```

2.5.4.2. Comparison

Of course the success of similarity based visualization for annotating groups depends on how well the categories are separated from each other. When elements of a category appear in more clusters, i.e., yielding high entropy, more actions will be needed. The entropy of category i^{th} is computed by the allocation of all elements in this category over the clustering [40]:

$$\mathrm{E}_i = -\sum_j \frac{\alpha_{ij}}{\alpha_i} \log \frac{\alpha_{ij}}{\alpha_i}$$

where α_i is the number of images in category i, α_{ij} is the number of images in category i which appear in cluster j.

From the results of the previous section, we select $n = 100$, with $\lambda_2 = 0.9$, the average percentage of $t = 75\%$ visible images will be more than 80% and reaches 100% of displayed images visible for 50% and 25%. We use the above clustering result with 100 clusters. As a result of clustering, the sizes of clusters vary, therefore in the display of the cluster's content, if the size of a cluster is larger than 100, the system will show subsets of the cluster containing at most 100 images. Figure 2.13 shows the annotation effort decomposed into the different categories.

We observe that if the categories are reasonably separated, i.e., entropy is not too large, the number of actions needed is reduced significantly. For example, in the given collection, some categories containing black and white images can be well distinguished from other colorful categories. Using color histogram as a feature, they will be placed close to each other, less actions will be needed in this case. The annotation effort for those categories reduces from 100 to only 6 actions. For other categories, the categorization is semantic and cannot be easily distinguished based on the color histogram only. The entropies of these categories are high making the mixture of images among different categories on the display. Therefore, more actions will be required, but in general always less than the number of actions in the baseline. On average, our system reduces the annotation effort by 20% up to 94%. We can conclude that more complex implementation pays off, we can significantly reduce the annotation effort.

2.6 Conclusion

Visualization is an essential tool for exploring visual collections. To build a good visualization system, a set of related requirements should be taken into account.

In this chapter, we established three general requirements for similarity based visualization systems: overview, visibility, and structure preservation. These require-

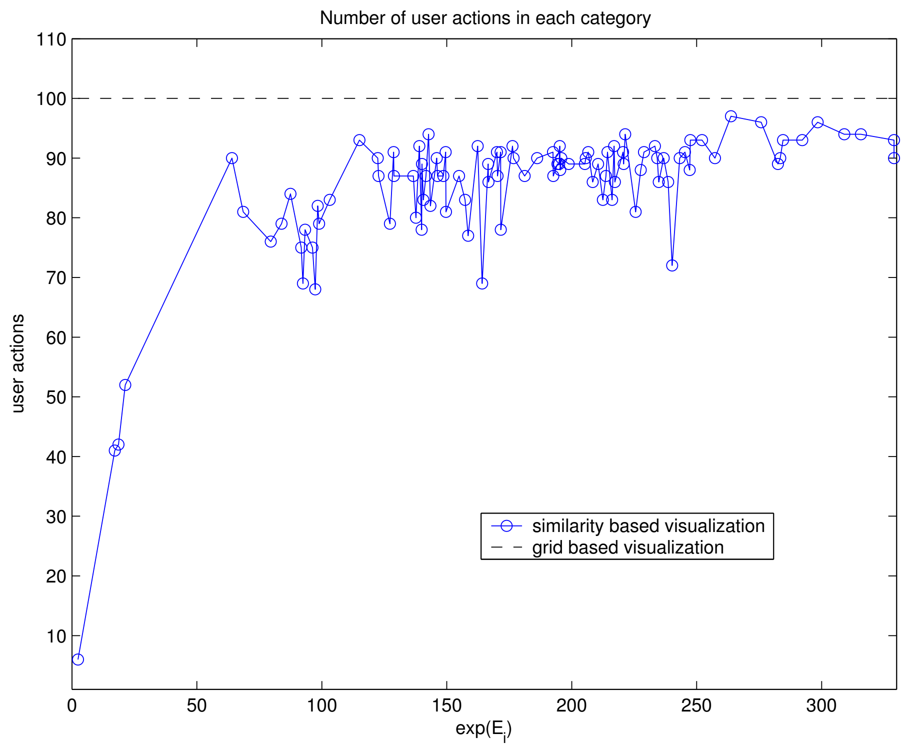


Figure 2.13: Counting the number of user actions for annotation by simulation using grid-based and similarity based visualization. The different categories are ordered by their entropy. For better viewing the result, we use an exponential function on the x-axis. The y-axis shows the annotation efforts. The baseline is the dashed line representing the actions in grid based visualization. As mentioned in previous sections, because of the sequential display, the number of actions equals the size of the categories. In this case, there are 100 images in each category. So, annotation effort is 100 user actions. The solid line shows the result for the proposed visualization interface.

ments provide the user an optimal way to interactively access a large image collection. The overview gives the user the overall look of the whole collection, and guides the user to the right search direction. The structure preservation ensures that original relations between images in the collection are kept in the visualization. Visibility is essential for interaction between the user and images displayed. As these requirements are not independent, compromises among them are needed. We proposed novel balancing cost functions and algorithms used to define the relative importance of these requirements to the overall visualization goal.

Using a rather large data set of 10000 images, we conducted experiments to evaluate the proposed framework. In a first experiment, we compared the performance of different projection methods, namely LLE, ISOMAP, SNE, ISOSNE, and ISOLLE. ISOSNE is the best option when computation time is not the limiting factor. However, for interactive performance, LLE or ISOLLE should be selected. To evaluate the

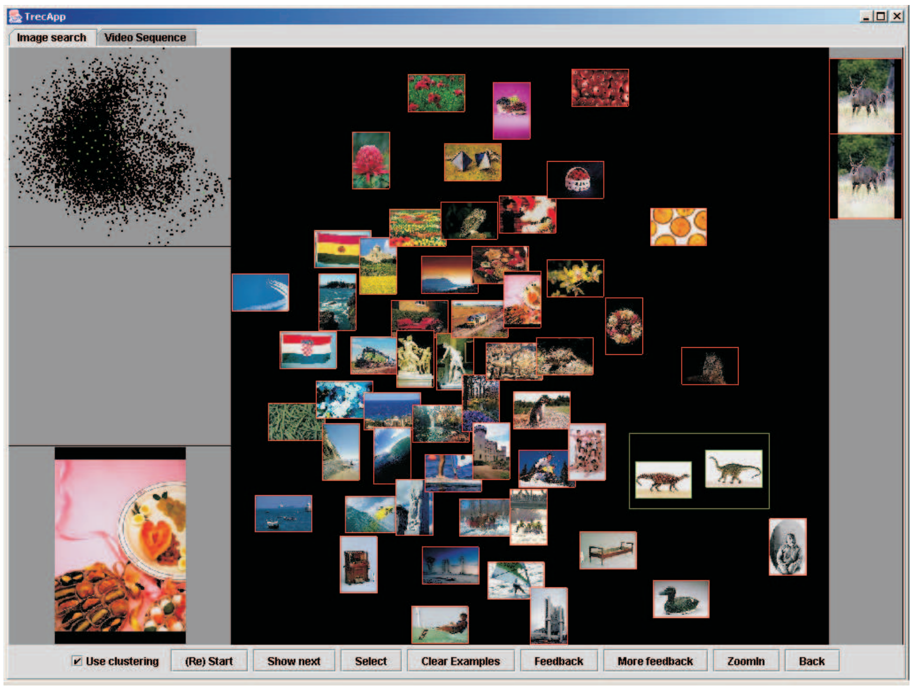


Figure 2.14: A screendump of the visualization system. The upper-left corner shows the whole collection as a point set with red points representing the currently displayed set. The bottom-left corner shows an enlarged version of the currently selected thumbnail. The main screen shows the representative set as images. The green rectangle is an example of a user selection of a group of images by dragging the mouse.

interactive system as a whole, instead of doing user based evaluation which is quite expensive and not easy to repeat, an objective user model is built. In particular we defined a *database annotation* scenario. The proposed visualization scheme reduces the total annotation effort significantly ranging from small reduction to 16 times lower effort depending on the separation of the different categories.

Not only in the given scenario, but for different tasks such as target search, category search, category annotation, or example based search, one could apply the three requirements and balancing functions to build an optimal system by simply changing the scenario.

Acknowledgment

The work is within the ImIk project (Interactive disclosure of Multimedia Information and Knowledge) sponsored by IOP MMI (Man-Machine Interaction). The authors would like to thank J.J.Verbeek for his valuable discussions and suggestions.

Chapter 3

Optimization of interactive visual similarity-based search

3.1 Introduction

Research on interactive search mechanisms is currently split into two separate fields. In content based retrieval, concentration is on machine learning methods for effective use of relevance feedback [94, 104]. The information visualization community focuses on effective methods for conveying information to the user [53, 103, 6]. What lacks is research considering the information visualization and interactive content based retrieval as truly integrated parts of one search system.

Features form the basis for content-based retrieval (CBR). In literature a large number of features have been introduced to improve search performance. Some features are only proposed for specific tasks or specific domains, for example medical images, or satellite images. These methods have limited application. General features such as global color histograms quite often fail because of the *semantic gap* between user expectation and system ability. Employing accompanying text, the semantic gap can be reduced either by using generic features such as keywords in the text [75] or multi-modal concepts [106]. The required indices are either labor expensive, time consuming or hard to derive. For instance, often text is not available, hence processing a textual query requires annotating the whole collection, which is impractical for large collections.

When no suitable concept is present, many image retrieval systems use query by example, where examples are taken from inside or outside the collection. In reality, there are many search tasks where the user does not have any example to start with. Then, the user needs to explore the collection to find relevant examples. In both query by example and exploration a good similarity function is essential.

Finding a suitable similarity function is strongly dependent on the chosen feature.

This chapter is accepted with subject to minor revision in the ACM Transactions on Multimedia Computing, Communications, and Applications

Systems in literature usually select the similarity function, commonly associated with the chosen features. For example, histogram intersection [110] is typically selected for comparing two color histograms. Features and similarity are computed offline.

The interactive stage contains two main steps namely visualization and relevance feedback, which are iterated [104]. The visualization step displays a selected set of images to the user. Based on that the user judges how relevant those images are with respect to what she is looking for. The system learns from the user's feedback and repeats the above step.

Using learning is well-known in interactive CBR. Comprehensive overviews of techniques are presented in [94, 104, 128]. Recently the use of support vector machines in learning has gained interest. It has proved to give the highest boost to the performance [17, 38, 39, 62, 113, 19].

For effective interaction, a good learning method is not sufficient, an interface for communication between the user and the system is needed. Such an interface should not be judged on its aesthetic value alone, but more importantly on its effectiveness in supporting the user's search process. It should therefore combine CBR and advanced visualization. We call such an integrated system a visual search system.

To realize visual search systems, recent systems apply similarity based visualization techniques giving a more informative and effective interface [71, 78, 42, 91, 97]. chapter 2, we present comparisons of existing techniques in visualizing image collections and conclude that for an optimal similarity based visualization system, three requirements have to be obeyed: *overview, structure preservation*, and *visibility.*

(i) *Overview requirement*: ensure that the set displayed represents the whole collection since not all images from the collection can be shown on the screen at once.

(ii) *Structure preservation requirement*: preserve the relations between images in the original feature space on the screen.

(iii) *Visibility requirement*: keep the content of displayed images visible to make interaction feasible.

The similarity based visualization we proposed in chapter 2 satisfies these requirements. It forms the basis for our interactive system, which in this chapter is extended with an active learning component *.

The ultimate goal of a search system is to have real users working with it. Thus, when optimizing retrieval, one should not only evaluate the individual steps, but the integral process. However, an experiment with real users introduces a large number of variables influencing the result such as age, sex, level of expertise, and the type of questioning. Therefore, obtaining repeatable results from experiments such as the one in [87] is difficult and time consuming. When the scenarios and criteria for success have been made explicit, the user and its actions can be simulated. It effectively allows for decomposition of the evaluation of the visualization methodology and the design

*This system is part of the MediaMill system which received the Best Techinical Demo Award at ACM Multimedia [106] of the interface and system performance [47]. Therefore, we develop an evaluation method integrating both user interaction as well as system aspects in a simulated scenario where many of the free parameters can be optimized before the user study.

In this chapter, we aim for a similarity based search system combining visualization and interaction. The system builds upon the visualization and active learning components described above. The key issue here is that we integrate these elements into a unique framework. Within the system, there are many free parameters like the similarity function and the method for processing relevance feedback. To optimize the performance of the system, we consider an interactive category search scenario in various large image collections. The search tasks range from finding images sharing simple properties such as images of the same object, to complex properties, for instance, images of a person entering a vehicle.

The chapter is organized as follows. To optimize the search task, each step is analyzed to find the best solution in section 3.2. In section 3, we present experiments of the proposed system for three different classes of data: ALOI (Amsterdam Library of Object Images), the Corel dataset, and the TRECVID dataset.

3.2 Methods

In this section, we first briefly describe the system in chapter 2, and from there its proposed active learning extension. Finally, we describe the built-in optimization method.

3.2.1 Similarity based visualization

The system in chapter 2 is based on the requirements (i), (ii), (iii) from the introduction. Cost functions are introduced for each of them.

Let us consider an image collection $\mathcal{I} = \{I_1, I_2, \cdots, I_N\}$ described using features. For similarity based visualization, a projection has to be made from the high dimensional feature space to the *visualization space.* From the results in chapter 2, the ISOSNE turns out the best one among others in obeying the *structure preservation requirement.* In this projection, nearest neighbors of each image in the feature space are computed to create a nearest neighbor graph. The graph-based distance is then used to redefine the distances between non-neighboring images. Finally, the non-linear mapping SNE obtains the positions of images in visualization space. To evaluate this requirement, a structure preservation cost function $\mathcal{C}_S$ is defined. Assume $\mathcal{P} = \{P_{ij}\}$ and $\mathcal{Q} = \{Q_{ij}\}$ are the discrete probability density functions on relations between N images in the high dimensional space and the visualization space, respectively, then we have:

$$\mathcal{C}_S = \sum_{i}^{N} \sum_{j}^{N} P_{ij} \log \frac{P_{ij}}{Q_{ij}}. \tag{3.1}$$

Thus, when this function is minimized, the relations between images are optimally preserved in the visualization space. As a consequence, similar images tend to be grouped together. They are likely to overlap on the screen. It influences the interaction process, because the user can miss the relevant ones as they are hidden. Therefore, the need for the *visibility requirement*. The cost function of this requirement $\mathcal{C}_V$ is based on the inscribed circles of the two images. For simplicity, assuming all images are square and have equal size:

$$\mathcal{C}_V = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} \left(1 - \frac{\mathcal{O}_{ij}}{\pi R^2}\right). \tag{3.2}$$

where $\mathcal{O}_{ij}$ is the overlap area of the inscribed circles of image I_i and I_j. R is the radius of these inscribed circles. In reality, images can appear with different sizes, in this case, the value R will be varied for different images.

To satisfy the *overview requirement*, $\mathcal{I}$ is divided into a number of clusters on the visualization space. A set containing selected images from each cluster, called the representative set, is the basis for presentation to the user. To measure how well this set represents the whole collection, the modified Hubert statistic [28] is used as the overview cost function $\mathcal{C}_O$. Assume we have n clusters. Let $d(I_i, I_j)$ denote the distance between image I_i and I_j and $d_c(I_i, I_j)$ the distance between the two cluster centers containing the two images, we have:

$$\begin{aligned}
r &= (1/M) \sum \sum d(I_i, I_j) d_c(I_i, I_j), \\
M_p &= (1/M) \sum \sum d(I_i, I_j), \\
M_c &= (1/M) \sum \sum d_c(I_i, I_j), \\
\sigma_p^2 &= (1/M) \sum \sum d^2(I_i, I_j) - M_p^2, \\
\sigma_c^2 &= (1/M) \sum \sum d_c^2(I_i, I_j) - M_c^2, \\
M &= k(k-1)/2,
\end{aligned}$$

The overview cost function $\mathcal{C}_O$ is defined as:

$$\mathcal{C}_O = \frac{r - M_p M_c}{\sigma_p \sigma_c} \tag{3.3}$$

The above requirements are conflicting. For example, to satisfy the overview requirement, the number of representative images should be large. Because of the fixed size of the visualization space, the more images the higher chance of overlapping images, hence, the visibility requirement will be violated. On the other hand, while preserving visibility, images are spread out, original relations between them are changed i.e structure is not preserved. Therefore, two balancing functions between the above cost functions, namely $\mathcal{C}_1$ and $\mathcal{C}_2$, are needed. The first one is the relation between overview and structure preservation. The second one is between structure preservation and visibility. The two parameters influencing the cost functions are the size n of the representative set and the positions $\mathbf{y}$ of the images in visualization space. From results in chapter 2, we have:

$$\mathcal{C}_1(n) = \lambda_1 \mathcal{C}_O(n) + (1 - \lambda_1)\mathcal{C}_V(n) \tag{3.4}$$

$$\mathcal{C}_2(\mathbf{y}) = \lambda_2 \mathcal{C}_S(\mathbf{y}) + (1 - \lambda_2)(1 - \mathcal{C}_V(\mathbf{y})) \tag{3.5}$$

where $0 \leq \lambda_1, \lambda_2 \leq 1$, n is the size of the representative set and $\mathbf{y}$ denotes the set of positions of images in the visualization space.

In brief, after the collection is projected to the visualization space, images are clustered into n groups. The k-means algorithm is selected to do the clustering because of its simplicity and good performance. Each time a set of n images representing clusters are displayed. The value of n is the result of optimizing the balancing function $\mathcal{C}_1$ using exhaustive search:

$$n_{\text{opt}} = \underset{n \in [2..n_{\max}]}{\text{argmax}} \; \mathcal{C}_1(n). \tag{3.6}$$

Because of the limitation of the display, there is a restricted number of images which can be shown to the user at one time. The maximum number of displayed images depends on the size of the image, the size of the display, and the area of the images visible to the user. Let us assume all images have equal sizes denoted w and h for width and height, respectively. Let the the size of the display be W and H. Finally, assume that for interaction, displayed images should be visible for at least v%. We have:

$$n_{\max} = \frac{H \times W}{h \times w \times \frac{v}{100}}$$

In our system, v is set to 75% which we consider feasible for a user to view the image content. With $W = 900$, $H = 1024$, $w = h = 64$, from the above equation, $n_{\max}$ is set to 300.

Positions $\mathbf{y}$ are found as

$$\mathbf{y}_{\text{opt}} = \min_{\mathbf{y}} \mathcal{C}_2(\mathbf{y}). \tag{3.7}$$

by gradient descent.

In figure 3.1, examples of displaying different sets of images are shown. It can be seen that similar images are indeed placed close to each other in the visualization space and almost every image is visible.

3.2.2 Features and similarity functions

In [78], one feature and one similarity function is used. As these two parameters also affect the overall performance of the system, we extend the consideration with different features and similarity functions.

3.2.2.1. Features

As mentioned in section 3.1, features are generally task-dependent. Using shape features to detect circles with constraints in their spatial relations to find cars may give satisfying results, but will completely fail when applied in searching for flowerbeds.

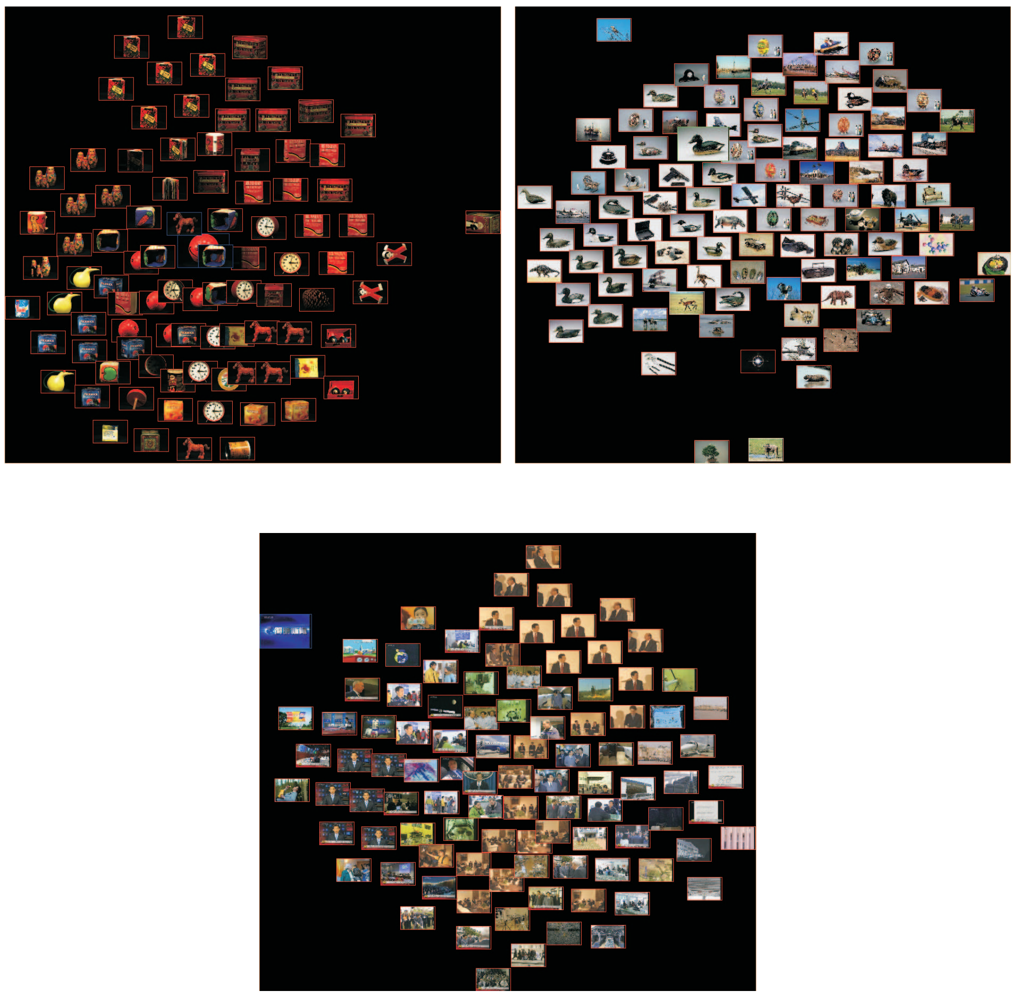


Figure 3.1: Examples of visualizing images with similar ones are close together.

Here we aim at a generic method, therefore, we select from literature two generic features, one for color and one for texture.

We denote a feature set as $\mathcal{F} = \{F^1, F^2, \cdots, F^l\}$, where l is a number of features. For each image $I_i \in \mathcal{I}$, $\vec{F_i} = (F_i^1, F_i^2, \cdots, F_i^l)$ is the corresponding feature vector. The global color histogram is the most commonly used feature because of its simple computation, and its invariance to rotation and small changes in the images. The L*a*b color space, compared to color spaces such as RGB and HSI, has the advantage that it is designed to be perceptually uniform. A distance in color space leads to equal human color difference perception. In our experiment, the L*a*b histogram is computed using 32 bins for each color channel, so in total, each image is represented by a 96 dimensional feature vector, i.e. $l = 96$.

The texture feature is the Wiccest (Weibull Invariant Color Contrast ESTimator). This is a new class of color invariant features introduced by Geusebroek in [36]. Wiccest features combine color invariance with natural image statistics. Color invariance aims to remove accidental lighting conditions such that the result is constant under varying illumination color, shadow effects and shading, while natural image statistics efficiently represent image data. We implement the feature extraction following [106]. An image is first divided into 3×3 regions. In each region the Wiccest features with 2 Wiccest parameters (β, γ) in 6 color channels are computed. This means that each region is represented by a vector of 12 values. Therefore, an image is represented by a vector of $l = 108$ dimensions.

3.2.2.2. Similarity or distance functions

Similarity is interpreted as a distance function. The more similar two images, the smaller the distance between them. In literature, a number of distance functions have been proposed [5, 90]. The simplest ones are in the family of Mikowski based functions:

$$d(I_i, I_j) = \left(\sum_{t=1}^{l} (F_i^t - F_j^t)^m \right)^{\frac{1}{m}} \tag{3.8}$$

with $m = 1$ we have the city-block distance L_1, $m = 2$ is the Euclidean distance L_2, and the special case is the maximum value distance (Chebyshev) L_∞.

Some distance functions measure the difference of two probability distributions. Examples are Kullback Leibler distance and its symmetric version Jeffrey divergence.

$$\text{Kullback divergence: } d(I_i, I_j) = \sum_{t=1}^{l} (F_i^t - F_j^t) \left(\log \left(\frac{F_i^t}{F_j^t} \right) \right)$$

$$\text{Jeffrey divergence: } d(I_i, I_j) = \sum_{t=1}^{l} F_i^t \log \left(\frac{F_i^t}{\frac{F_i^t + F_j^t}{2}} \right) + F_j^t \log \left(\frac{F_i^t}{\frac{F_i^t + F_j^t}{2}} \right)$$

Some other distance functions are also used often, for instance the Bhattacharyya, and Matusita.

$$\text{Bhattacharyya distance: } d(I_i, I_j) = -\log \left(\sum_{t=1}^{l} \sqrt{F_i^t F_j^t} \right)$$

$$\text{Matusita distance } d(I_i, I_j) = \sqrt{\sum_{t=1}^{l} \left(\sqrt{F_i^t} - \sqrt{F_j^t} \right)^2}$$

Though it is not an explicit rule that one should use a specific distance function for a specific feature, it is a tacit consent. For example, the common way of comparing images represented by L*a*b color histograms is Euclidean distance:

$$d_{\mathrm{L^*a^*b}}(I_i, I_j) = \sqrt{\sum_{t=1}^{l} \left(F_i^t - F_j^t)\right)^2} \tag{3.9}$$

The Wiccest features are Weibull-based. A distance measure for two Weibull distributions with parameters (β_1, γ_1) and (β_2, γ_2), is defined in [36] as

$$d_{\text{Wiccest}} = 1 - \frac{\min(\beta_1, \beta_2)}{\max(\beta_1, \beta_2)} \frac{\min(\gamma_1, \gamma_2)}{\max(\gamma_1, \gamma_2)} \tag{3.10}$$

As the Wiccest features are computed for all the regions in the partitioning of the image, we have:

$$d_{\text{Wiccest}}(I_i, I_j) = 1 - \frac{1}{N_r} \sum_{t=1}^{N_r} \frac{\min(\beta_i^t, \beta_j^t)}{\max(\beta_i^t, \beta_j^t)} \frac{\min(\gamma_i^t, \gamma_j^t)}{\max(\gamma_i^t, \gamma_j^t)} \tag{3.11}$$

where N_r is the number of regions in each image. In our case 3×3 regions are used hence $N_r = 9$.

3.2.3 Relevance feedback and active learning algorithm

We now extend the visualization method described with an active learning component. A common way of giving feedback is to have the user label a set of images as positive (relevant images) and/or negative (non-relevant images). For an overview see [94, 128]. To provide a set of images, the system actively selects images which are most informative. Labelled images are used as the training set [38, 79, 124]. After learning, a new set of images is then selected, and the process is iterated. This process is known as active learning.

In literature, the active learning methods mostly use SVM as a feedback learning base [17, 62, 39]. SVM was first introduced in data mining research as a classification method. Given a training set $\mathcal{I_T}$ of $r \ll N$ images, each image $I_i \in \mathcal{I_T}$ is represented as a point $\mathbf{x}_i$ with label $l_i \in \{-1, 1\}$. The aim is to produce a model that predicts the class for unlabelled images by creating a hyperplane that separates the collection into a positive and a negative class based on the user feedback. This requires to optimize the following function:

$$\begin{array}{ll} \min_{\omega,b,\xi} & \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{r} \xi_t \\ \text{subject to} & l_i(\omega^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \end{array} \tag{3.12}$$

with C a penalty parameter for the error term. Training image $\mathbf{x}_i$ is mapped to a higher dimensional space by the function ϕ . The function $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ is called the kernel function. The SVM originally uses linear classifiers. Several non-linear kernel functions have been introduced later. In [15], the authors point out that the radial basis function is a good choice, which we will use in the chapter, namely

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2},$$

$\gamma > 0$ is a kernel parameter.

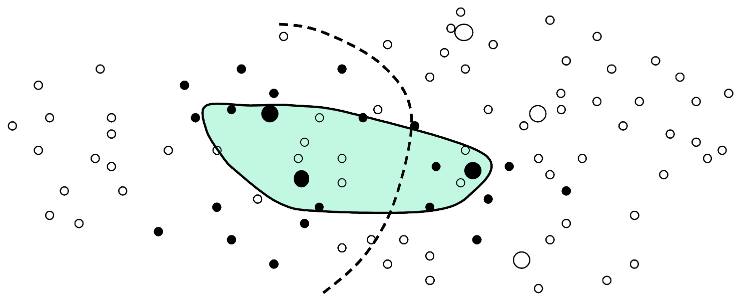


Figure 3.2: An example of a decision boundary. The small circles are unlabelled data, the bigger ones are the labelled ones used as training data. The empty circles are irrelevant images, and the filled circles are relevant ones. In case of one-class SVM, only the labelled images are used as positive examples, SVM finds the optimal border to cover those training examples (solid boundary). The dashed line is the result in case of two-class SVM.

The performance of SVM depends on the selection of its parameters. However, there is no optimal set of parameters that works well in all cases. The choice of default parameters are a good option for building a general search system using SVM [15]. Therefore, we also use the default parameters as suggested in [15].

With different ways of giving relevance feedback, $\mathcal{I}_\mathcal{T}$ can contain either both positive and negative examples or only one of those. Hence, two main approaches for learning are available. The first approach is based on both positive and negative examples, which is called two-class SVM. At each iteration, both sets are used to find the best classifier separating positive examples from negative ones [124]. The second one is based on only positive examples known as one-class SVM. The main argument for using this approach over the other one is that in a large collection, the number of relevant images is normally much smaller than the total size of the collection. Moreover, the distribution of non-relevant ones is unpredictable and therefore it is difficult to find the borders for those. Hence, in this approach when interacting with the images displayed, the user labels positive examples only. By assuming similar images are clustered in feature space, the algorithm learns the border of the area covering as much as possible of those examples, i.e. the set of images one is searching for [17, 62].

After training is finished, there are two common ways of selecting a new set of images to display for another round of feedback. The first approach is selecting images in the positive class with maximum distance to the border, which have highest chance of being relevant to the search task. Alternatively, in the second approach, images closest to the border are returned, and this approach is said to provide the most informative set to the user [17, 62].

Those images returned are then also labelled as positive or negative by the user. The system repeats the process to obtain new feedback information. The iteration is stopped when the performance satisfies given constraints such as the number of iterations, time limitation, or simply that the user does not want to give any more feedback.

Finally, when the interaction process is finished, images inside the decision boundary are ranked by their distances to the border. Images having maximum distance are supposedly most relevant to the search task.

3.2.4 Scheme

Up to this point, we have analyzed the two main techniques used in the system namely similarity based visualization (SimVis) and active learning ($\text{SimVis}_{\text{Active}}$). We will now define the proposed scheme combining these techniques. The user starts the search without any examples at hand, and the general scenario contains the query initialization step and the interactive step.

The SimVis system requires the preparation of the projections, and representative set. We can add these steps in the interactive stage, but usually this is computationally impractical. Therefore, they are computed beforehand. First, features of images in the collection are selected and extracted. Distances between images are then obtained with a selected similarity function. After that, ISOSNE is applied to project images to the visualization space. Next, we employ k-means to cluster images into n clusters. A set of images selected from different clusters form the representative set of the collection. Information of each image belonging to a certain group, and its position in the visualization space are stored as offline data.

Figure 3.3 shows the interactive stage. Each time a set of n images is displayed. In the first screen, the representative set is shown to the user. He then uses the system to explore the collection and find relevant images. Particularly, if the currently displayed set contains any positive examples, the user selects that image and asks for a set of nearest neighbors expecting to find more images that are similar. The number of nearest neighbors displayed is set to k. Those neighbors are based on the selected similarity function.

The system satisfying the visibility requirement, assures feasible interaction with all images displayed. Moreover, employing the advantage of similarity based visualization, instead of clicking on an individual image for labelling, the system allows the user to select several images by dragging a rectangle around images in the same category. As a consequence, our system can reduce the number of actions needed from the user. In case there is no positive image in the current set, the user asks the system to display another representative set. This set contains images, which have not been displayed before and are closest to the previous representative set.

In the feedback step, training examples are selected by the user. When a certain number of examples are provided, the SVM trains the support vectors. We use the well-known SVM library developed by [15], which gives one-class as well as two-class SVM implementations. As indicated, the parameters of the SVM implementations are set to their default values.

3.2.5 Scenario optimization

First of all, we remind the search scenario that we will work on:

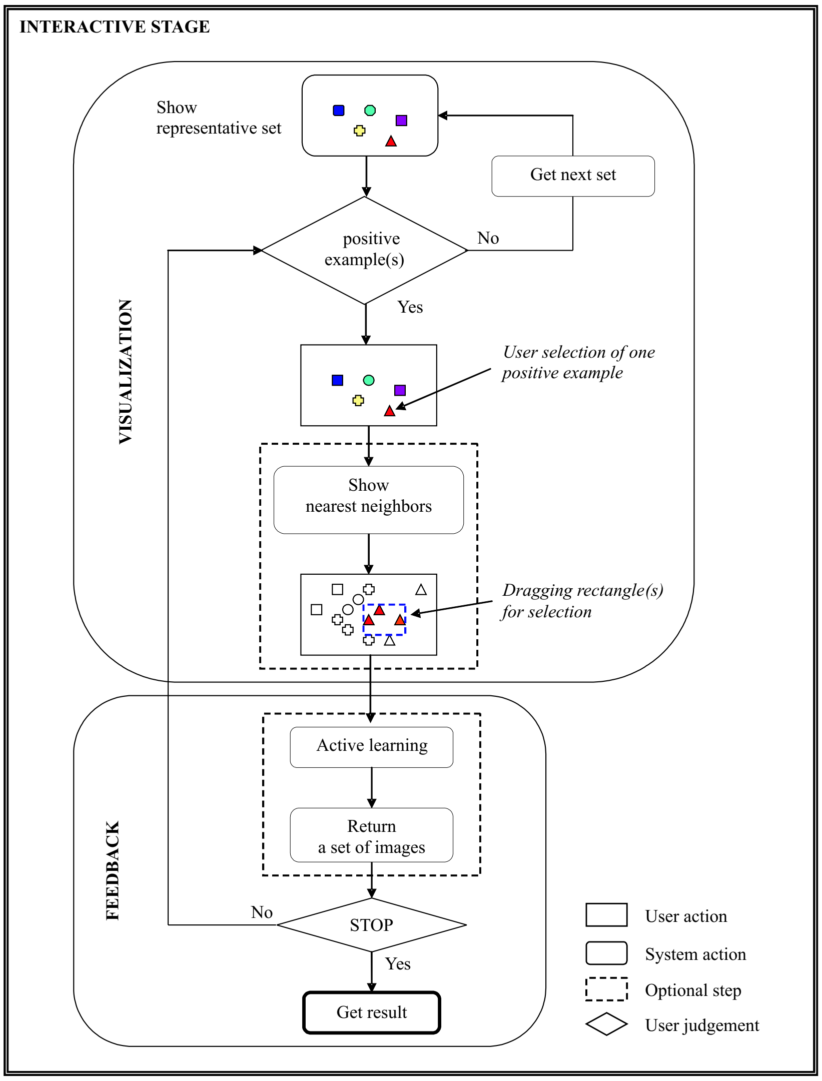


Figure 3.3: The interactive stage of the SimVis$_{\text{Active}}$ system.

Category search: *finding as many images as possible belonging to a certain category.*

In the general scheme above, there are several degrees of freedom in each step either from the user or the system point of view. This leads to the difficulty of finding the factors that are user specific and those that are objectively improving the system. In the introduction section, we have pointed out that the use of simulated user and system actions helps in optimizing the overall performance. Hence, in this section, all possible system and user actions in the proposed scheme are simulated.

The main concentration is the interactive stage, where the *system actions* are:

- Selection of images displayed following the balance between the overview and visibility requirement.
- Adjustment of those images on the display such that the balancing cost function

between the visibility and the structure preservation requirements are optimized.

- Remembering examples selected or removed by the user.
- Reaction to user requests such as: get feedback, show neighbors, and show results.

Secondly, we simulate *user actions* including user judgment. There are four typical types of actions, which count as one action:

- Selecting one image
- Clicking to display another set of representative images to find more examples.
- Clicking to go to the cluster corresponding to the current example.
- Dragging a rectangle around a set of positive images. In the worst case, a rectangle contains only one image.

The first three actions are straightforward in implementation. For the last one, we proposed an algorithm for finding the minimum number of rectangles needed to cover all relevant images in the currently displayed set [78]. Briefly, for each relevant image, the system finds a rectangle containing the maximum number of neighboring relevant images such that none of the irrelevant ones is inside. The pseudo-code for simulating the user action of dragging a rectangle is described as

```
Rectangle-search(an image set M)
   For each element m in M
       If (m has not been annotated as positive)
           X = sort(neighbors(m)) on distance to m;
           R = draw_rectangle(m);
           For each element x in X, where category(x)=category(m)
               store(R);
               R = draw_rectangle({R, m}); //increase size of the rectangle.
               If R contains z, where category(z)!=category(m)
                   R = store(R);
                   break;
               else
                   annotate(x);
```

Figure 3.4 shows the simulated scenario with simulated user actions and simulated system actions associated with the scheme. The *counting user actions* will calculate the number of actions the simulated user needs during the interactive process. The *optimization* part involves the adjustments of parameters in the simulated scenario.

3.3 Experiments

3.3.1 Data preparation

For studying the proposed methods, we use three different image collections with widely varying characteristics. First, the ALOI (Amsterdam Library Object Images) dataset which contains high quality images of 1000 objects [35]. Each image contains

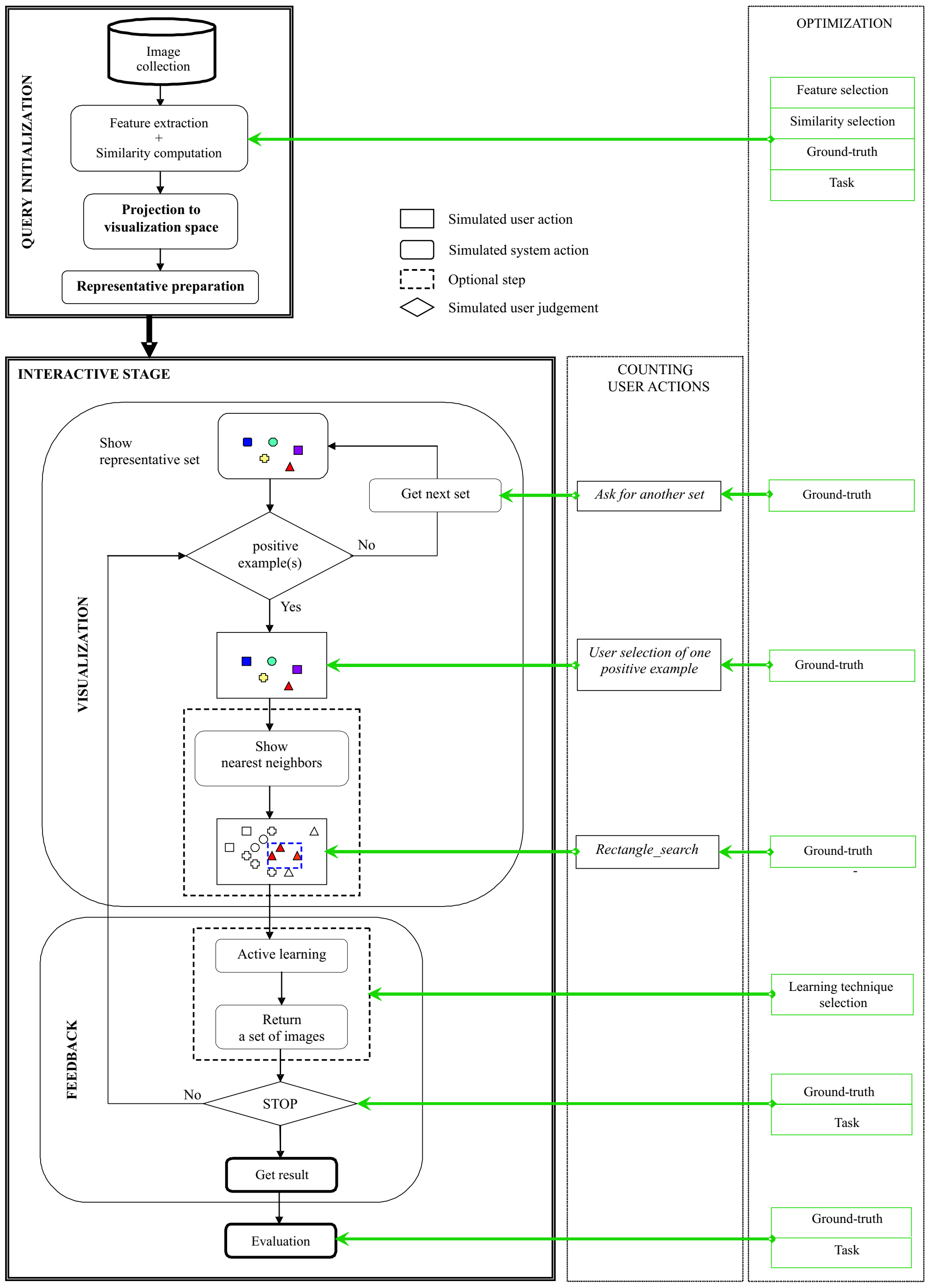


Figure 3.4: Scheme with simulated user and system actions.

one object, and this object is captured in different orientations and lighting conditions. In total, the dataset has 110,250 images. For each object, we select 12 different recording conditions, which includes 4 different in-plane viewing angles (0^0, 30^0, 90^0, 270^0), 6 different illumination angles, and 2 illumination colors. The 12 images of the

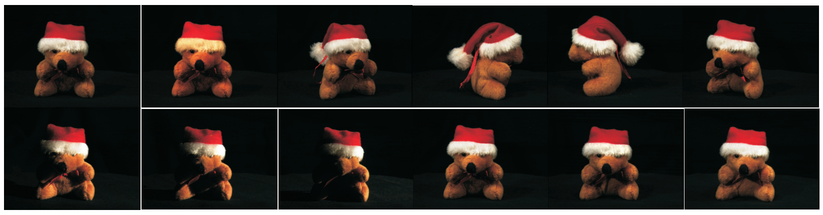


Figure 3.5: Left-right, top-down order: two images with changing illumination colors, images under 4 different viewing angles, images with 6 different illumination angles.

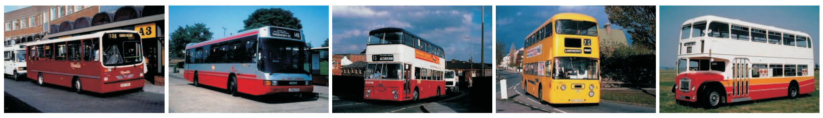


Figure 3.6: An example of images in the category *Bus* from the Corel dataset.

object form one category. Thus, the dataset has 12,000 images in total, with 1000 categories. As images of the same class capture the same object, we expect a lot of structures in the information space spanned by these images. Figure 3.5 shows the 12 different images of one object.

The second one is the well-known Corel dataset, which is used often in existing search systems. Images in this dataset are also of high quality. Different from the first collection, they are classified into semantic categories, where images in one category might vary widely in visual appearance. We select a collection of 33326 Corel images with different scenes such as winter and surfing, and objects such as flowers, buses and birds. As ground truth, we use the pre-defined categorization of Corel. Thus, we have 460 categories where the size of each category is different. An example from this collection is shown in figure 3.6.

The final collection contains images taken from the TRECVID 2005 news video archive [106]. As we deal with images, each video is first segmented into shots; each shot is then represented by a keyframe. Thus, TRECVID05 contains 45765 images. TRECVID defines a set of 24 topics such as *images of Tony Blair*, or *images of people shaking hands*. For these 24 search topics ground-truth information is available and these topics are thus used as categorization. The selection of this dataset into the evaluation is challenging in many aspects. As images are extracted from videos, which are encoded, the quality of this dataset is low compared to the other two. Moreover, they have a semantic categorization, and within these categories elements vary largely in content (see figure 3.7).

3.3.2 Selection of similarity/distance functions

With the L*a*b and Wiccest selected as features, this section concentrates on selecting suitable similarity or distance functions. The primary role of the distance function

Figure 3.7: An example of images in the category *People shaking hands* from the TRECVID dataset.

is to add structure to the data. In the ideal structure, the distance yields semantic clusters, where neighboring images belong to the same category. Therefore, the aim of this section is to choose the distance function able to return the maximum number of relevant nearest neighbors. This is important as in the projection step, nearest neighbors are first computed after which the mapping tries to preserve those neighbors in the projected space. So the k-nearest neighbors of an image ideally belong to the same category.

Now let $\mathcal{I}_C$ be the set of images in category C. For evaluation of the distance functions in 3.2.2, we compute for each image $I_i \in \mathcal{I}_C$ the set $\mathcal{I}_i^k$ of k nearest neighbors. Results are evaluated, using the ground truth categorization. Recall and precision values for image I_i are:

$$\text{recall}(I_i) = \frac{||\mathcal{I}_i^k \bigcap \mathcal{I}_C||}{||\mathcal{I}_i^k||} \tag{3.13}$$

$$\text{precision}(I_i) = \frac{||\mathcal{I}_i^k \bigcap \mathcal{I}_C||}{||\mathcal{I}_C||} \tag{3.14}$$

From there, we compute mean recall and mean precision as:

$$\text{mean recall} = \frac{\sum_{I_i \in \mathcal{I}_C} recall(I_i)}{||\mathcal{I}_C||}, \tag{3.15}$$

$$\text{mean precision} = \frac{\sum_{I_i \in \mathcal{I}_C} precision(I_i)}{||\mathcal{I}_C||} \tag{3.16}$$

where $||.||$ denotes the size of a set.

Figure 3.8 shows results for each collection. The results are surprising. The default distance L_2 used for L*a*b feature is not the best one. Its performance is even worse than L_1. For all of the three collections, the Matusita function is the best out of all the distance functions considered for L*a*b. For the Wiccest feature, the standard distance function, as expected, performs significantly better than the rest.

From the experiments, we decide to use the Matusita function in case of L*a*b, and the original Wiccest distance for comparison of images described with Wiccest feature.

3.3.3 System setup

In the query initialization or the offline stage, because of the need for a balance between the different requirements, parameters need to be set such as the number of

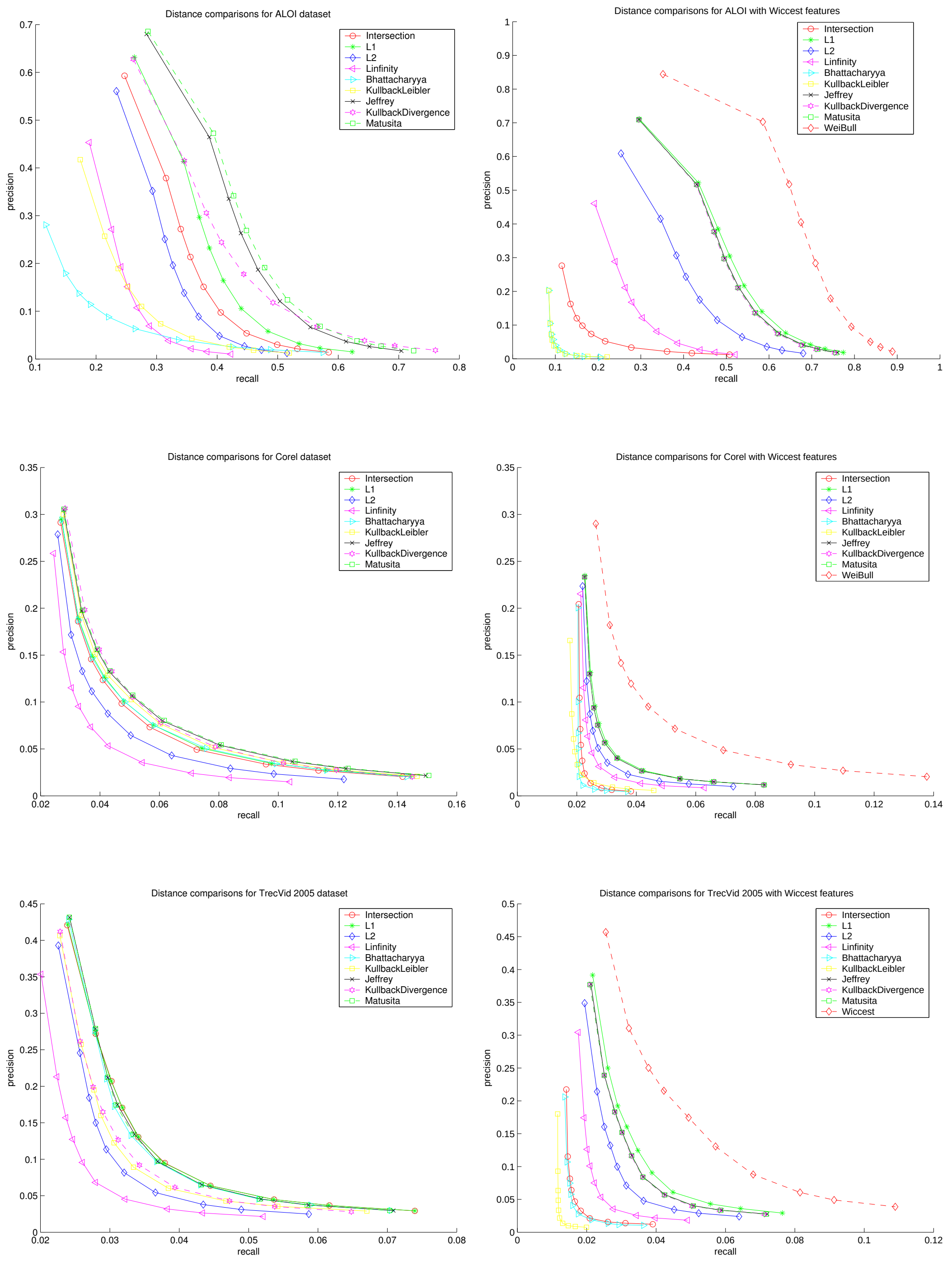


Figure 3.8: Comparison of distance functions: The first column gives results for L*a*b; in top-down order the ALOI, Corel, and TRECVID collection. In a similar way, the second column gives results for the Wiccest features.

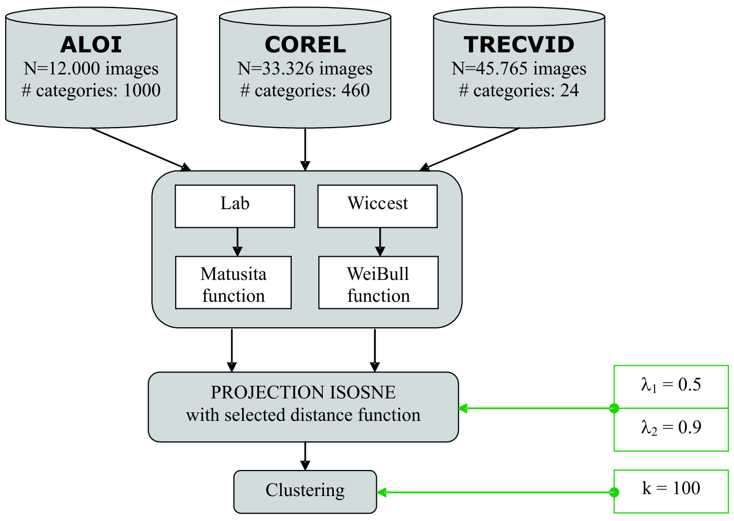


Figure 3.9: Query initialization in experiments.

clusters and the values for λ_1 and λ_2. We use the result from [78], λ_1 and λ_2 (in the equations Eq.3.4 and Eq.3.5) set to 0.5 and 0.9, respectively. In this reference, we experimented with different values for n in Eq.3.4, $n = 100$ turned out being optimal.

K-means algorithm with $n = 100$ is applied to the whole collection. With large images sets, the original k-means algorithm has two major disadvantages: high memory requirements and large computation time it requires the calculation of the $N \times N$ distance matrix, where N is number of images. As this process is done only one time in the offline stage, timing is not an important issue. To overcome the memory problem, we use the competitive learning technique for the k-means algorithm [95].

The data preparation for the experiments is shown in figure 3.9. Each of the selected features is computed for each image collection and stored. In the projection from high dimensional feature space to the visualization space, neighbors are found using corresponding distance functions.

3.3.4 Evaluation criteria

The benefit of using SVM is clearly proved in previous work, our aim in this chapter is to prove that the combination of SVM and similarity based visualization improves the results even further.

In category search, performance of the system is commonly measured using precision and recall. Recall is the percentage of relevant images in the result set, with respect to the total set of images in the category searched for. Precision is the number of relevant images in the result set, relative to the size of the result set. To evaluate and compare different methods, both recall and precision are important. It is however easier to have one single measurement. Therefore, another popular measure is the average precision [119, 129] which calculates the area under the precision and recall curve. The values of those measurements are in the range $[0, 1]$.

Average precision *is the sum of the precision at each relevant hit in the result set divided by the total number of relevant images in the collection.*

Besides, high average precision, our aim is to reduce the number of user actions needed to get there. Hence, the average precision is computed as function of the interactive effort.

Interactive effort *is the total number of actions one needs to interact with the system to get a certain result.*

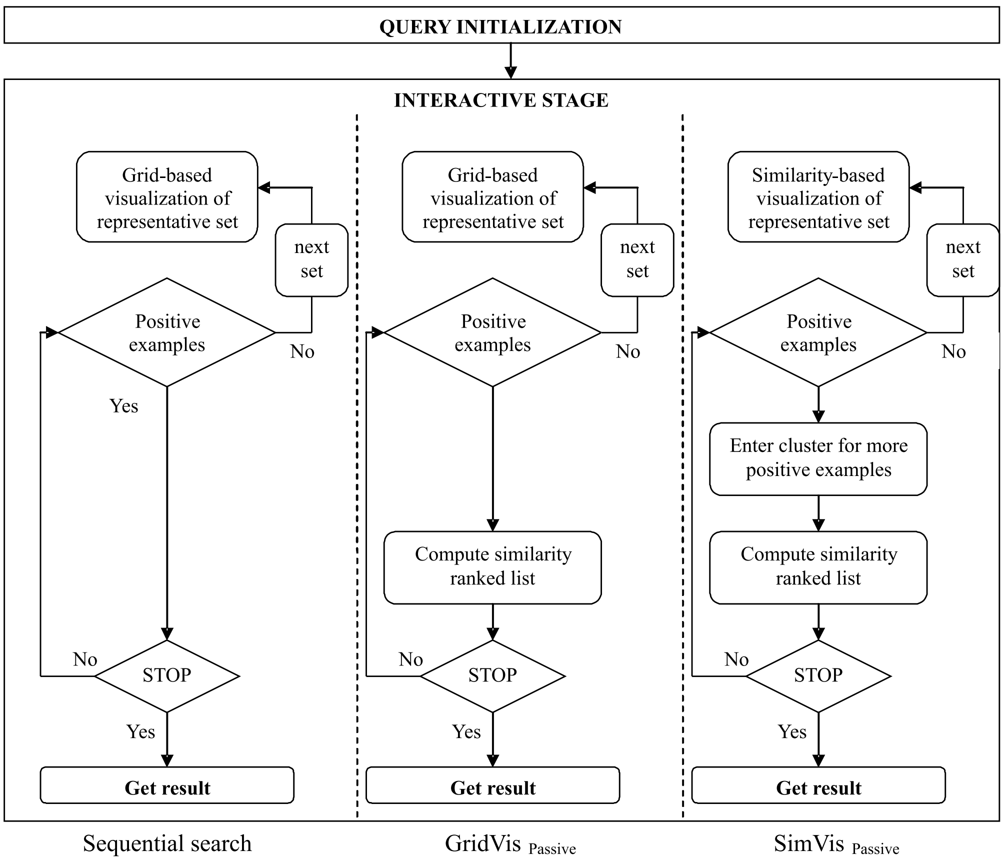

QUERY INITIALIZATION
INTERACTIVE STAGE
Grid-based visualization of representative set
next set
Positive examples
No
Yes
STOP
No
Yes
Get result
Grid-based visualization of representative set
next set
Positive examples
No
Compute similarity ranked list
STOP
No
Yes
Get result
Similarity-based visualization of representative set
next set
Positive examples
No
Enter cluster for more positive examples
Compute similarity ranked list
STOP
No
Yes
Get result
Sequential search
GridVis Passive
SimVis Passive

Figure 3.10: A brief scheme for three different searching approaches. From left to right: sequential search, **GridVis$_{\mathbf{Passive}}$**, **SimVis$_{\mathbf{Passive}}$**

For comparison, we consider the search system with or without SimVis and with or without active learning. First, to provide the baseline, without both of the optional steps, we get exhaustive search, without any system support. The first column in figure 3.10 is a sketchy version of the interactive stage of the baseline. In this search, the user gradually goes through the collection to find all relevant images. The system displays 100 random images in sequence (we name this visualization GridVis, i.e. grid-based visualization), the user selects positive images one-by-one from the images in the visualization. This search is called **sequential search**. When finished, another random set is shown. The process is repeated until all images of the category are found.

Secondly, the first optional step is again removed, i.e. visualization method is GridVis, and the second one is replaced by the standard interactive search algorithm, leading to **GridVis$_{\mathbf{Passive}}$**. Therefore, the interactive stage in figure 4 is skimmed down as shown in the second column of figure 3.10. In this search, after a number of interactions, the selected images are used as query set. Similarity values between all images in the collection to the query set are computed based on a predefined similarity function. A ranked list based on these similarity values is obtained. The top 100 most similar ones are returned. From there, average precision values are also computed. We report the mean average precision at interactive effort intervals of 5 over all the search categories for each collection.

As a third method, we apply the first optional step to the sequential search, to find what performance in search can be achieved without an advanced learning algorithm, **SimVis$_{\mathbf{Passive}}$**.The process is briefly presented in the last column of figure 3.10.

Finally, the proposed scheme in figure 3.4, called **SimVis$_{\mathbf{Active}}$**, is compared to all the others.

In summary, four different search strategies are considered

- **Sequential**: exhaustive search with gradual annotation of relevant images.
- **GridVis$_{\mathbf{Passive}}$**: interactive search using grid based visualization, no support of similarity based visualization.
- **SimVis$_{\mathbf{Passive}}$**: interactive search with similarity based visualization.
- **SimVis$_{\mathbf{Active}}$**: interactive search in the proposed scheme with combination of similarity based visualization and active learning.

3.3.5 Selection of active learning methods

In this experiment, the two SVM approaches in section 3.2.3 are considered. Parameters for the two methods are taken from [15]. Results are in figures 3.11.

This figure shows the results on three collections with two different features. In the ALOI collection, as the number of images in each category is rather small, using one-class SVM out-performs two-class SVM as the two-class SVM classifies many irrelevant ones as relevant. In the Corel collection, as the number of relevant ones is reasonable with 100 or more images, the performance of two-class SVM is improved. Overall, it is comparable with one-class SVM, but again the speed factor is a concern as the size of training data increases rapidly during interaction. The last two figures are for the TRECVID collection. Again, the performance of the one-class SVM is higher.

Results prove our expectation on the performance of the two techniques. Despite of the fact that more information is given in the training examples for two-class SVM than in one-class SVM, it turns out the one-class SVM works better in general. One of the reasons is that the number of non-relevant images is very large while the relevant set is much smaller. A way to gain a better performance in this particular problem

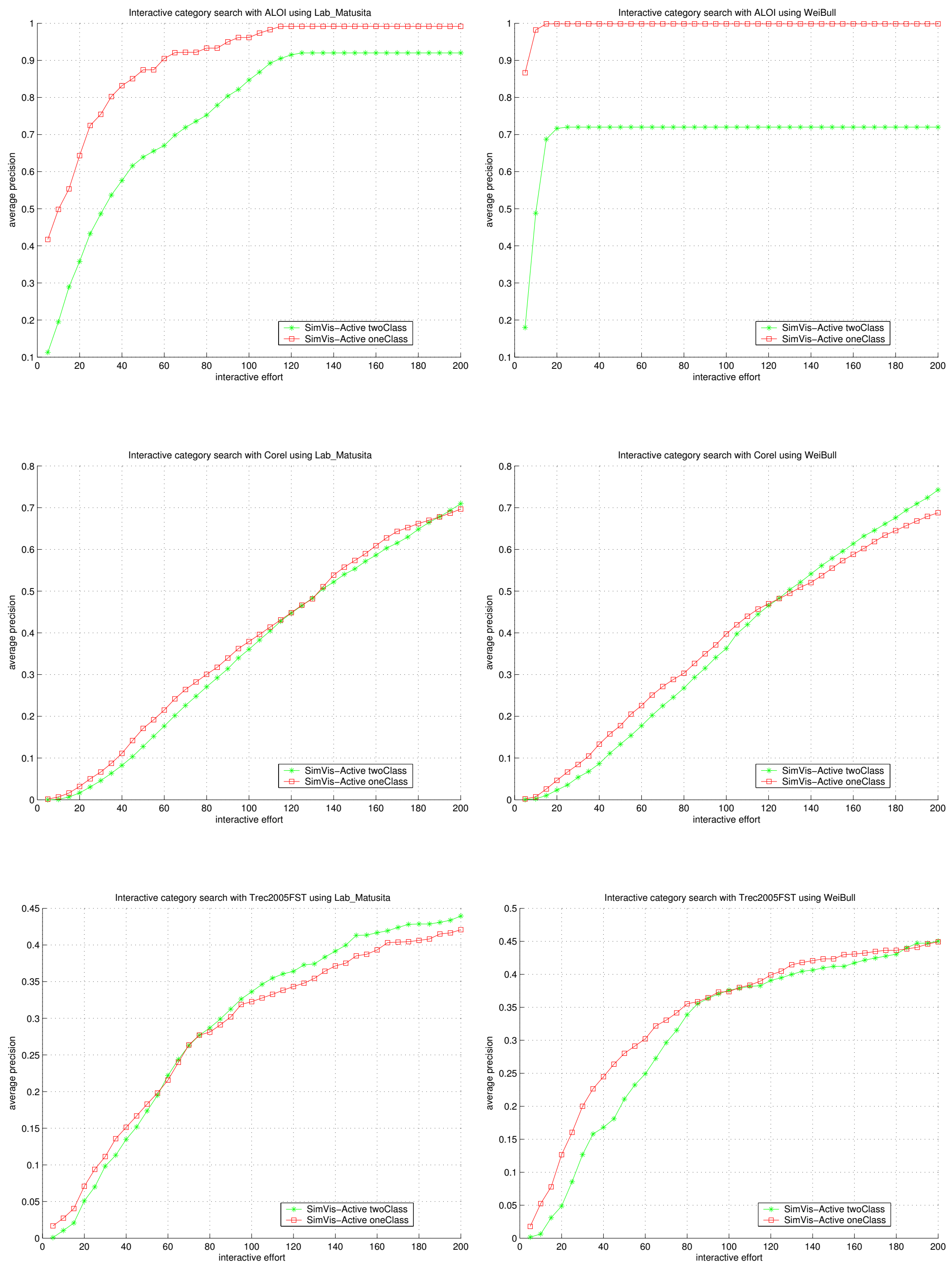


Figure 3.11: Comparison of SVM performance. The first column gives results with L*a*b, top-down the ALOI, Corel, and TRECVID collection. In a similar way, the second column gives results for the Wiccest features.

of the two-class SVM is to adjust the parameters such as setting the penalty for the negative class higher. However, doing this iteratively according to the user relevant feedback is very difficult. Moreover, after one feedback iteration the size of the training data is increased especially the number of negative examples, hence the training time takes longer. In interactive search, processing speed is an important element. All the computation and display should be in acceptable time as the user does not have the patience to wait long for the next iteration. One-class SVM requiring much less examples will be a better option.

One-class SVM does have the disadvantage that when relevant images fall into different clusters or they are scattered, it may concentrate on one part of the feature space containing relevant images, ignoring other potentially relevant areas. Nevertheless, in the given search scenario, within a certain searching time, the number of correct ones returned are more important than how diverse the distribution of correct ones is. Therefore, we select the one-class SVM.

3.3.6 Experimental results on the full scenario

For each query, we simulate the user search actions following the proposed scheme. The search stops when the average precision reaches 1.

The ALOI results are presented in figure 3.12. Since images of the same category are close in information space, even GridVis$_{\text{Passive}}$ performs well in the beginning. However, at some point, it cannot find more relevant images, while the average precision of the proposed approach still increases. Secondly, results show that using Wiccest features gives better performance in this dataset than L*a*b. The average precision reaches 1 very fast especially with our approach. The search time is improved significantly. Where in the baseline the user needs on average 80 actions to get all the relevant images, only 20 actions are needed in SimVis$_{\text{Active}}$.

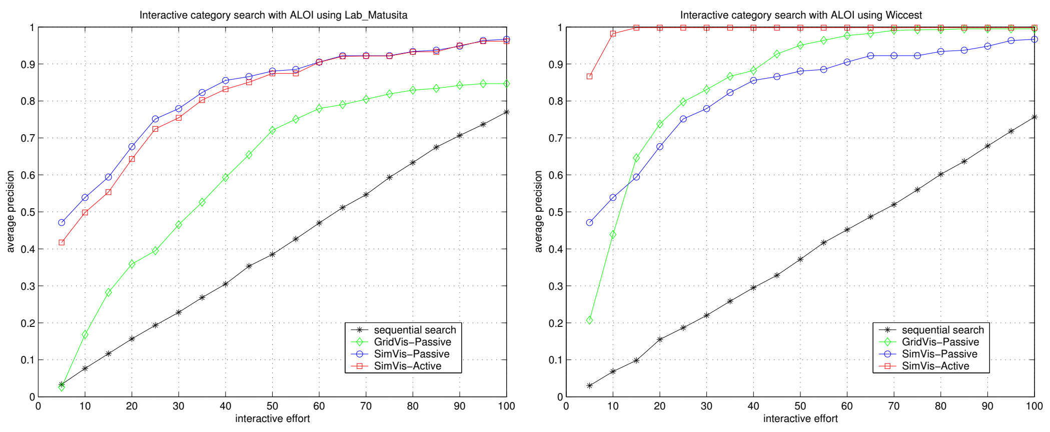


Figure 3.12: Results of *interactive effort* vs. average precision with the ALOI data. Left to right: with the L*a*b, with the Wiccest features.

Results in figure 3.13 are for the Corel collection. The GridVis$_{\text{Passive}}$ performs worst, even worse than the sequential baseline. This is expected, as the color histogram is too poor to distinguish high-level concepts. With similarity based visualization, the result of SimVis$_{\text{Passive}}$ is improved significantly. At the beginning, when only limited interactive effort has been made, i.e. small number of training samples, the GridVis$_{\text{Passive}}$ even yields better results than the SimVis$_{\text{Passive}}$ and SimVis$_{\text{Active}}$. When sufficient examples are provided, performance of the proposed system becomes the best. Overall, Wiccest features do not improve the performance upon simple L*a*b histograms for this dataset.

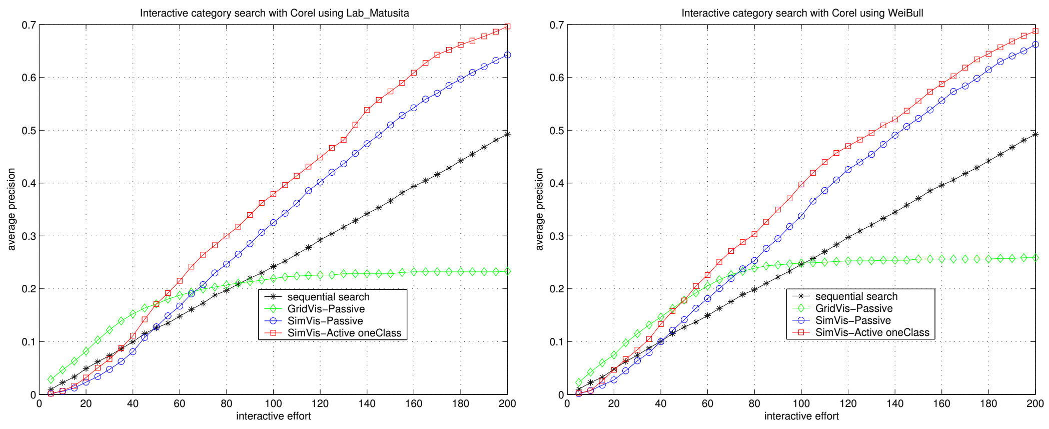


Figure 3.13: Results of *interactive effort* vs. average precision with the Corel data. Left to right: with the L*a*b, with the Wiccest features.

Figure 3.14 gives the results for search topics in TRECVID data. It is obvious that the baselines are very low as the search tasks are semantic. The global histogram L*a*b is too simple to obtain the relevant images, meaning that the relevant images are easily scattered into different clusters. Therefore, though the performance of the proposed system is improved, on average the different between SimVis$_{\text{Passive}}$ and SimVis$_{\text{Active}}$ is not as significant as in case of the ALOI and Corel collection. To search through this collection, filtering based on a textual query or a semantic concept query, as for example in [106], should be applied first to reduce the whole collection to a limited set of more homogeneous images. From there, our approach can be employed to do the search.

In summary, the proposed approach works well in all three different image collections with different image features. It shows that the combination of advance visualization and active learning does lead to a better search performance. Results with the Wiccest features in general are better than with the simple L*a*b color histogram, especially with the ALOI collection where images of the same search category are very similar. Figure 3.15 show an example of the system interface, where interactions between the user and the system take place.

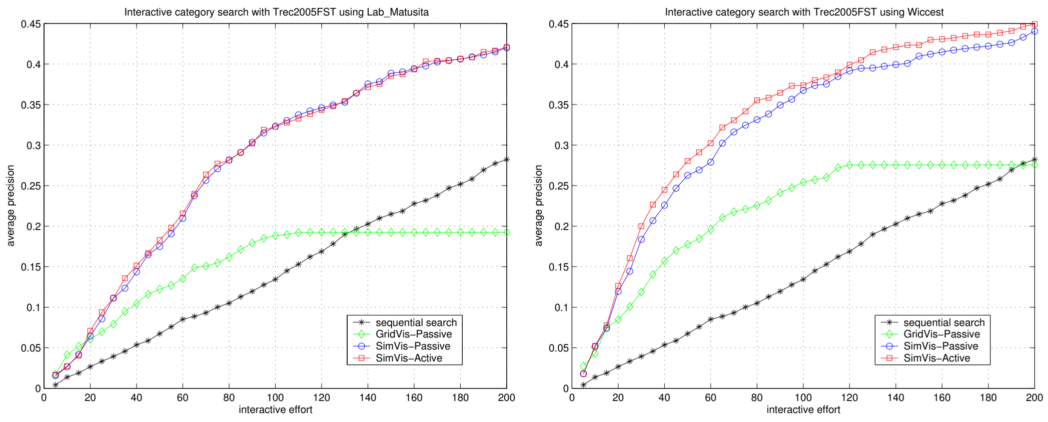


Figure 3.14: Results of *interactive effort* vs. average precision with the TRECVID data. Left to right: with the L*a*b, with the Wiccest features.

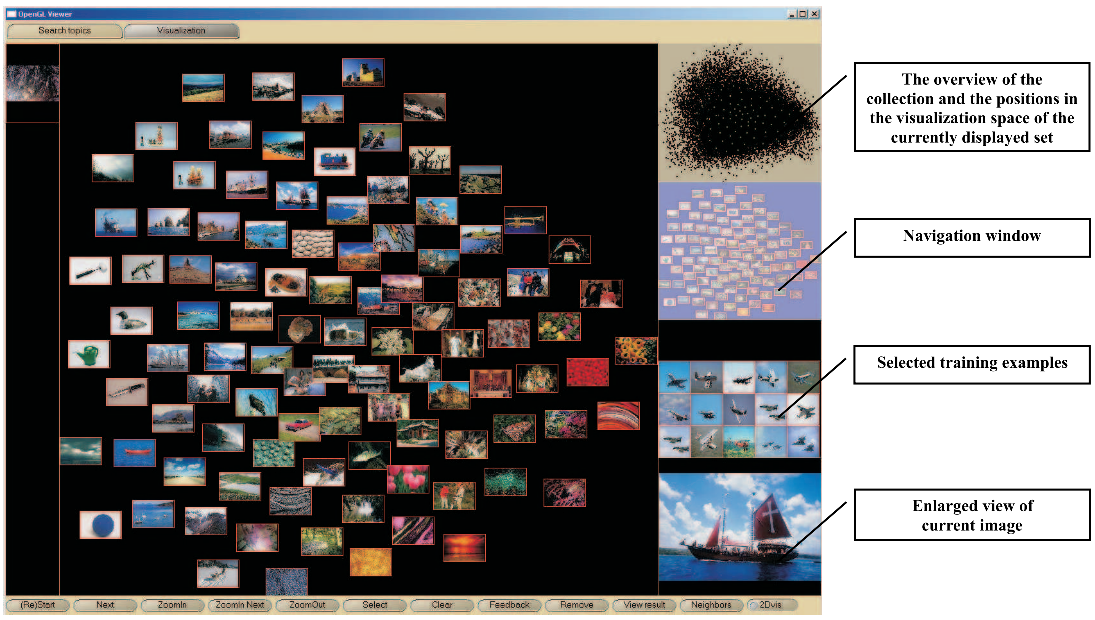

OpenGL Viewer
Search topics
Visualization
The overview of the collection and the positions in the visualization space of the currently displayed set
Navigation window
Selected training examples
Enlarged view of current image
(Re)Start
Next
ZoomIn
ZoomIn Next
ZoomOut
Select
Clear
Feedback
Remove
View result
Neighbors
2Dvis

Figure 3.15: Screen-dump of the interactive search system with the Corel dataset.

3.4 Conclusion

In interactive search, most systems only concentrate on improving the performance of active learning based on user feedback. Therefore, the way images are shown to the user has received little attention. On the other hand, in evaluating the interaction interface between the user and the system related work is mostly interested in the visualization. A visual search system combining similarity based visualization and active learning using support vector machines has been proposed. In this scheme, we analyze possible relations between visualization and relevance feedback. From there, an optimal solution is presented.

On the way of building an optimal system, we have implemented different distance functions for comparing two feature vectors. Our experiments show that the selection of distance function is strongly dependent on the chosen feature. The default function may not be the best option, for instance for L*a*b histograms, the Matusita function is certainly better than the commonly used Euclidean function. Comparison between one-class and two-class SVM is also presented. Our conclusion is that for interactive search, the two methods are generally having equal performance, but one-class SVM is much faster, therefore, better suited for interactive search.

To prove the efficiency of our method, we have strategically selected three different classes of image collections. The first collection, ALOI, has high quality and exhibits structure in the high dimensional feature space as images in a category are of the same object. The Corel collection also has high quality but semantic categorization; hence, it is difficult in search. The last one, a collection of broadcast news shots used in TRECVID 2005 from CNN, Chinese, and Arabic broadcasters, is the most challenging for any search system with varying semantics in each category. By selecting these collections, we show that our proposed approach gives the best performance compared to common systems.

Applying the system with objective evaluation where all possible actions are simulated is also one of our contributions in this chapter. This choice overcomes the limitation of having real users or subjective evaluation because of its difficult and expensive set up. The proposed simulated actions can be easily applied to other systems.

Experiments show that for the ALOI collection, our system reduces the user interactive effort by a factor of 4 compared to the search baseline, for example, after 10 actions it reaches an average precision of almost 1.0, whereas it needs up to 70 actions in the baseline to get similar results. In case of Corel, on average, the performance of the proposed approach is 3 times better with the same number of user actions, and almost 2 times for the TRECVID collection. Our system, therefore, eases user search tasks such that with much less number of actions it is possible for the user to get more images that are relevant.

The performance of the new class of color invariant features, the Wiccest features, is still domain dependent. It works well, specifically for the ALOI collection, when images of the same category search are very similar. For collections where similarity of images is more abstract such as in the TRECVID collection, its performance does not outperform the simple color histogram feature. That means for a general search system, using color histogram is still a good option

In this chapter, we concentrate on visual exploration of the collection as a whole while ignoring the query specification step. However, the proposed scheme can still easily apply to systems which use a query as filtering step. Especially in cases where the search is based on semantic categorization, for instance in the TRECVID collection, our visual based search scheme works well [106].

With the tremendous growth of image collections, a very time consuming user interaction process will result. Our approach can be considered as the beginning of a new generation of advanced search mechanisms. The scenario-based evaluation method is a general scheme for evaluating interactive retrieval systems. Many other search scenarios can be objectively evaluated in order to find effective integrated methods.

3.5 Acknowledgments

The work is within the ImIk project (Interactive disclosure of Multimedia Information and Knowledge) sponsored by IOP MMI (Man-Machine Interaction) and the BSIK MultimediaN project. Dr. J.M. Geusebroek is thanked for providing the ALOI collection as well as the Wiccest features.

Chapter 4

Interactive search by direct manipulation of dissimilarity space

4.1 Introduction

Interactive search tasks in content-based image retrieval (CBIR) are classified into three types: association search, target search, and category search [104]. Search by association is a class of search where the user starts with no specific aim other than interesting findings. Target search aims at finding one specific image. And category search looks for all images belonging to a specific class. In any of the three tasks, during the search process, the system aims at finding relevant images while discarding irrelevant ones. To do so, a dissimilarity measure is needed to compare images. However, dissimilarity between images strongly depends on the context of the search. Prior to the interaction, the system's definition of dissimilarity is based on the objective image content, whereas during the interaction the user judges similarity based on a subjective interpretation of the semantic content. This contrast is known as the semantic gap [104]. Imagine we have two sets of pictures, one of "dogs" and the other of "birds". In a search task looking for images of the "animal" category, images in these two sets are to be taken as members of the same class. However, if the task is searching images of "pets", the pictures with "dogs" are relevant, but the pictures with "birds" may not be relevant as it depends on what type of bird appeared in the pictures. Only the user knows exactly what she is searching for and the systems needs to learn the dissimilarity based on the user's relevance feedback.

In literature, many different methods have been developed to learn dissimilarity measures from relevance feedback. For an overview see [94, 128]. To provide feedback, users enter their input and receive feedback in the *manipulation space*. When

This chapter is submitted to the IEEE Transactions on Multimedia feedback is given, the dissimilarity is commonly learned via feature space [7, 71, 58]. To effectively learn dissimilarity, a large set of features are needed or a small set of specific features. A small selection of specific features usually works for a narrow domain only. In contrast, a large set of generic features, provides the possibility to find dissimilarities among images of any kind. However, such a large set of features leads to a high computational load, especially when the dimension of the feature space goes to thousands of features. And what is more, a large set needs a large set of examples. For interactive search, immediate response is important, hence for interactive search in broad domains learning dissimilarity based on the feature space is not ideal.

Let us reconsider the above example. It is difficult to define effective features to assign two images to the "animal" group. However, if the user points out that target images are similar to an example picture of a "dog" and an example of a "bird" we might group them based on the observation that they are close to either one of them. Defining concepts on the basis of examples is also far more intuitive for the user than defining it in terms of the features of the images. In our approach, rather than considering the feature space, we will focus on the *dissimilarity space*, where images are represented by their relations to other images.

A similar approach has been applied in [10]. In this reference, the authors also consider dissimilarity space as a substitute for the feature space. They create dissimilarity spaces following the technique of Duin and Pekalska [29, 81] for different classes of features such as color or texture. They first select a set of images, named prototypes. The dissimilarity space is created such that images are represented by their relative dissimilarities to the prototypes. They then explore the optimal way of fusing these spaces over the feature spaces. Although the retrieval process is done on the dissimilarity spaces, these spaces are not updated, hence the interactive learning of dissimilarity still strongly depends on the initial feature spaces. As indicated this makes it difficult to learn the semantic target classes.

In feature space, the individual features have no meaning to the user so the feature space can not be mapped to manipulation space in an intuitive manner. Using dissimilarity instead of features is more intuitive for the interacting user. In the manipulation space, images are displayed showing the current interpretation of dissimilarity of the system. The user may interact in the space by labeling images as relevant and/or irrelevant, by giving dissimilarity scores, or by moving relevant images close to one another. By doing this, the user gives direct feedback to adjust dissimilarity among images.

In this chapter, we integrate the information available from interaction as deep into the definition of (dis)similarity as one can. We do not only update dissimilarity but also iteratively update the dissimilarity space. This means that when the user changes the layout of the image set displayed in the manipulation space, the layout of images on the dissimilarity space is also adjusted to fit the feedback. Images are presented to the user with their relations reflecting the system's definition of dissimilarity. This definition is obtained from the dissimilarity space. The user either agrees or not with the current relations. If not, he can give feedback to show his opinion on how the relations should be. The changes on the manipulation space will be directly mapped to update the dissimilarity space.

The chapter is organized as follows. In section 4.2, we will describe in more detail existing research in learning dissimilarity. Next, in section 4.3, we present our approach to learn the dissimilarity through updating the dissimilarity space in manipulation space. Results of the system with two different image collections are shown in section 4.4. Finally, conclusions are presented in section 4.5.

4.2 Background and related work

In this section, we introduce notation. We also give an overview of the literature on learning dissimilarity.

4.2.1 Basic notation

Given a collection of images $\mathcal{I} = \{I_1, I_2, ..., I_n\}$, an r-dimensional feature space $\mathcal{F}$ is defined in which each image $I_i \in \mathcal{I}$ is represented by a feature vector $\vec{F_i}$ of length r. Dissimilarities $S(.)$ are derived from the feature vectors. They are computed between every pair of images I_i and I_j and stored in a matrix $\mathcal{S} = \{S(I_i, I_j)\}_{i=1,n;j=1,n}$. Let $\vec{W} = \{w_1, w_2, \ldots, w_r\}$ denote a set of r values weighing the dimensions in $\mathcal{F}$. Let $\vec{\zeta}$ denote a set of parameters steering the dissimilarity function.

When the user is interacting with the system he has a goal which can be defined as a set of desired images $\mathcal{I}_+ \subset \mathcal{I}$ to be found. Given this goal, learning of dissimilarity can be viewed as an iterative process where the system learns to identify the set $\mathcal{I}_+$ from feedback given by the user.

4.2.2 Methods for learning dissimilarity

In most existing methods, learning is done via feature space either by feature selection [7, 71], feature weighing [58, 123], or using a parameter-based function of features [97, 39].

In general, for existing methods, learning the dissimilarity in iteration $t+1$ from analyzing the feedback of the user in iteration t can be formulated:

$$S^{t+1}(I_i, I_j) = f(\vec{F_i}, \vec{F_j} | \vec{W}^t, \vec{\zeta}^t), \tag{4.1}$$

where $\vec{W}^0$ and $\vec{\zeta}^0$ are start values.

In general, not all features are equally important. Hence, in the feature weighting approach weights are set for each feature. Feature selection is a special case of feature weighting, where the weights of the eliminated features are set to 0. As the update changes $\vec{W}$ only, Eq.4.1 can be rewritten as:

$$S^{t+1}(I_i, I_j) = f(\vec{F_i}, \vec{F_j} | \vec{W}^t). \tag{4.2}$$

In [123], the authors concentrate on exploring the distribution of the data set. A subspace of the feature space is found, and a quadratic similarity functions is learnt. From there, the dissimilarity matrix between images is updated. A similar approach is presented in [7], where the authors propose a weighted Minkowski similarity that continuously learns the weight of each feature based on positive and negative examples. The dissimilarity matrix is adjusted on the basis of the new set of weights. In [71], the similarity is also recalculated by selecting a subspace. Updating is based on the configuration of images on the screen resulting from the user's manipulation of the position of images on the screen. The system re-estimates the layout of the images, such that the similarity function gets closer to the user's desire. A similar approach but with dynamic selection of the feature subspace followed in [58]. In this reference, starting with a number of features, a dynamic function is proposed where at each step the optimal number of features is found. From there, the weighted and non-weighted perceptual dynamic function is built based on the Minkowski distance. In [31], the authors propose a system which allows the user to score similarity between given pairs of images. The system then learns the similarity coefficient from the user feedback predicting the similarity among the images which were not displayed. Within the same school of thought, work has been reported in [13, 41, 98, 107]. In general, this approach requires a large set of features in order to select a subset best representing the semantic similarity between images. However, the selection of a large number of features has a major disadvantage for interactive search because of its computational expense, especially with complicated dissimilarity functions. In addition, the need for a large a number of examples in learning leads to tedious interaction.

Another class of methods is formed by the parameter-based approaches. In these approaches, the matrix in the feature space does not change during learning, but rather a parameterized function of the features is adjusted to fit the user's feedback. Eq.4.1 is specialized as:

$$S^{t+1}(I_i, I_j) = f(\vec{F_i}, \vec{F_j} | \vec{\zeta^t}), \tag{4.3}$$

For example, in [97], the authors introduce an interface where the user adjusts the similarity between images in the manipulation space by moving them around. New positions of images displayed are used as relevance feedback. Using Fuzzy Feature Contrast and Tversky's similarity measure, the authors define a similarity function where $\vec{\zeta}$ contains around 100 different parameter dimensions. Based on user feedback, the system then adjusts the set of parameters in the dissimilarity function such that the dissimilarity decreases between images which according to the user are close. The large number of parameters requires a large number of training examples and thus substantial user interaction. In [39], given a set of images as query examples, a restricted similarity measure is formulated which recalculates dissimilarity between all images and queries depending on their positions compared to the classification boundary. The boundary is characterized by a parameter set. Given a set of positive and negative examples, SVM and AdaBoost are used to learn a classification boundary. The top ranked images are then labelled as positive and negative examples to repeat the refinement of dissimilarity.

The parameter based approaches do not require a large set of features. However, to effectively learn the dissimilarity matrix either the features should be well chosen or the system should have a wide range of parameters.

The use of features in learning dissimilarity has a major limitation as it strongly depends on the choice of features. With the lack of efficient features for general search tasks, a new approach that learns dissimilarity with less influence of features should be considered.

4.3 Direct manipulation of dissimilarity

In this section, we present our approach in learning dissimilarity by updating the dissimilarity space based on user's relevance feedback without going back to the feature space. Using the same notation as in eq.4.1 our method can be described as:

$$S^{t+1}(I_i, I_j) = f(S^t(I_i, I_j)), \text{ with } S^0(I_i, I_j) = f(\vec{F_i}, \vec{F_j}). \tag{4.4}$$

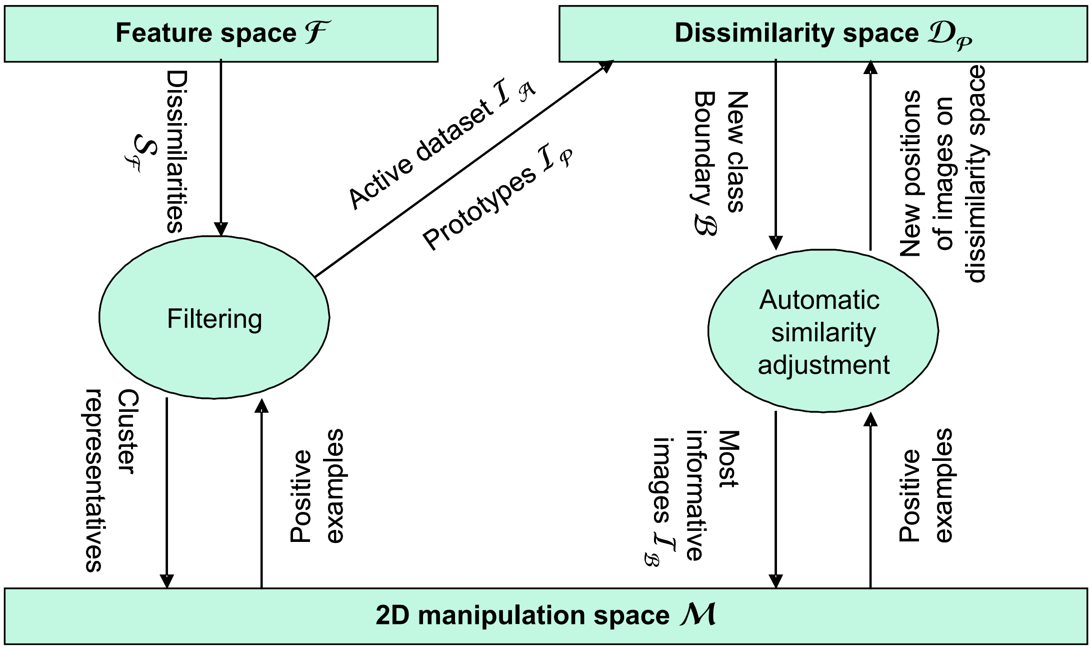

Feature space $\mathcal{F}$
Dissimilarity space $\mathcal{D}_{\mathcal{P}}$
Dissimilarities $S_{\mathcal{F}}$
Active dataset $\mathcal{I}_{\mathcal{A}}$
Prototypes $\mathcal{I}_{\mathcal{P}}$
New class Boundary $\mathcal{B}$
New positions of images on dissimilarity space
Filtering
Automatic similarity adjustment
Cluster representatives
Positive examples
Most informative images $\mathcal{I}_{\mathcal{B}}$
Positive examples
2D manipulation space $\mathcal{M}$

Figure 4.1: Schematic overview of the proposed approach.

An overview of our proposed approach is in Figure 4.1. First, a dissimilarity matrix is obtained by comparing feature vectors in the feature space $\mathcal{F}$. A projection from the high dimensional space is used to create a manipulation space $\mathcal{M}$ (to be discussed in section 4.3.2). Images are presented in $\mathcal{M}$ to the user for interaction and feedback. A set of images, named the prototype set $\mathcal{I}_{\mathcal{P}}$, is selected (section 4.3.1). When sufficient prototypes have been found, a dissimilarity space $\mathcal{D}_{\mathcal{P}}$ is then created (section 4.3.1). The learning process is then started. The manipulation space $\mathcal{M}$ is now a direct projection of $\mathcal{D}_{\mathcal{P}}$. In this learning phase, $\mathcal{D}_{\mathcal{P}}$ is iteratively adjusted with user feedback and active learning. The adjustment is presented in section 4.3.3. At each iteration, a set of most informative images is returned. The user then labels positive images for another round of feedback. The learning phase is finished when the user stops the search.

4.3.1 Creation of the dissimilarity space

4.3.1.1. Prototype-based dissimilarity space

To create a dissimilarity space, we employ the method proposed by Pekalska [81]. In the reference, the goal is data classification with no user interaction or relevance feedback, we extend it to interactive search.

The first step is to select a set of p images $\mathcal{I_P} \subset \mathcal{I}$, called the *prototypes*:

$$\mathcal{I_P} = \{I_{P_1}, I_{P_2}, \ldots, I_{P_p}\} \tag{4.5}$$

The role of the prototypes is to create a dissimilarity space where relevant and irrelevant images are well separated. Hence, careful selection of prototypes is important. The mapping from dissimilarity matrix to dissimilarity space by selection of prototypes is equivalent to choosing a set of columns (or rows) in the dissimilarity matrix. $\mathcal{D_P}$ denotes the dissimilarity space, and Φ the mapping from a dissimilarity matrix $\mathcal{S}$ to $\mathcal{D_P}$:

$$\Phi : \mathcal{S} \xmapsto{\mathcal{I_P}} \mathcal{D_P}. \tag{4.6}$$

This means that for each image I_i, we have a p-dimensional vector

$$\mathcal{D}_i = \big\{S(I_i, I_{P_1}), S(I_i, I_{P_2}), \ldots, S(I_i, I_{P_p})\big\} \tag{4.7}$$

Therefore, dissimilarities between all images in $\mathcal{I}$ to $\mathcal{I_P}$ are represented by a matrix with size $n \times p$. The collection $\mathcal{I}$ then builds up a p-dimensional dissimilarity space $\mathcal{D_P}$, named *prototype-based dissimilarity space*:

$$\mathcal{D_P} = \begin{bmatrix} S(I_1, I_{P_1}), & S(I_1, I_{P_2}), & \ldots, & S(I_1, I_{P_p}) \\ S(I_2, I_{P_1}), & S(I_2, I_{P_2}), & \ldots, & S(I_2, I_{P_p}) \\ \vdots & & \vdots & \\ S(I_n, I_{P_1}), & S(I_n, I_{P_2}), & \ldots, & S(I_n, I_{P_p}) \end{bmatrix} \tag{4.8}$$

An illustration of creating a dissimilarity space is shown in figure 4.2.

In $\mathcal{D_P}$ the similarity $S_\mathcal{P}(I_i, I_j)$ of two images is defined by the Euclidean distance between D_i and D_j.

4.3.1.2. Selection of prototypes with hierarchical clustering

For selecting prototypes, it is argued in [82] that systematic selection of prototypes gives a better representation of the dissimilarity space than random selection. We therefore aim to find a set of prototypes $\mathcal{I_P}$ such that the mapping Φ preserves the information in the similarity matrix $\mathcal{S}$ as good as possible.

The method in [81] aims for classification, where training set and test set have been defined beforehand. In interactive search, we are not able to select a set of prototypes as we do not know which images the user will search for. In practice, there are cases where the user starts the search with relevant and/or irrelevant images. In general

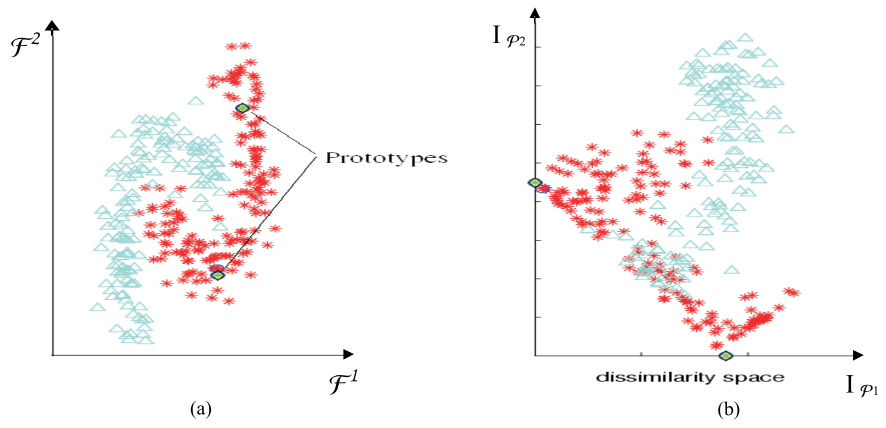


Figure 4.2: An example to illustrate the creation of a dissimilarity space. (a) In this example, for simplicity, images are represented in a 2D feature space $\mathcal{F} = \{\mathcal{F}^1, \mathcal{F}^2\}$, with dissimilarities among them obtained by the unweighed Euclidean distance between feature vectors. Two images are selected as prototypes. (b) Based on distances between all images to the prototypes $I_{\mathcal{P}_1}$ and $I_{\mathcal{P}_2}$, we create a 2D dissimilarity space.

this set of images is not a good set of prototypes. For CBIR, a strategy is needed for finding a good set of prototypes.

The browsing for prototypes is performed in manipulation space, where the user directly interacts with images. The selection of relevant images as prototypes makes the learning on dissimilarity space simpler [10]. We therefore focus on images selected as relevant by the user. In particular, a set of images is shown to the user, she will select relevant images when they appear. Because of the limitation of the display screen, only a small set of images can be displayed at a time. We denote the set of images displayed in iteration t as $\mathcal{I}_D^t$. This set should be carefully chosen as it affects the resulting $\mathcal{I}_{\mathcal{P}}$. Important is that $\mathcal{I}_D^t$ gives an adequate overview of the whole collection [78]. First, the collection is divided into a set of clusters $\{I_{C_k}\}_{k=1,M}$ using a clustering algorithm. The system selects an image from each cluster, which is called the representative element of that cluster, for display. M is chosen such that the representatives are giving an overview of the collection $\mathcal{I}$, while assuring that the M images still fit the screen. If a relevant image (i.e., an element of $\mathcal{I}_+$) is presented in $\mathcal{I}_D$, the user will select that image as a prototype, i.e.,

$$\mathcal{I}_{\mathcal{P}}^{t+1} = \mathcal{I}_{\mathcal{P}}^t \cup (\mathcal{I}_D^t \cap \mathcal{I}_+), \mathcal{I}_{\mathcal{P}}^0 = \emptyset \tag{4.9}$$

Once images currently on display are inspected, the system will choose a new set of representatives to display.

From a practical point of view, the size n of the image collection is usually large. Hence, sequentially visiting all images is a time consuming procedure and should be avoided when there are clusters not containing any relevant image. To speed up the process, we add a filter to the browsing. Rather than visiting all elements in cluster I_{C_i}, the cluster is divided into m sub-clusters using the same clustering algorithm. The value of m is selected such that:

$$m = \left\lceil \frac{||\mathcal{I}_{C_i}||}{n^*} \right\rceil, \tag{4.10}$$

with $||\mathcal{I}_{C_i}||$ the size of the cluster. For example, if the cluster $\mathcal{I}_{C_i}$ contains 100 elements and n^* is set equal 20, the number of sub-clusters is $m = 5$. Increasing the value of n^* will give a finer clustering and hence more browsing time. Coarse clustering with a high value of n^* increases the chance of missing relevant images.

As in the above, centers of the sub-clusters are chosen as the representatives for that cluster, i.e., one of the m sub-centers is sequentially selected to represent the cluster. Therefore, instead of visiting all n images, the user only considers $M * m$ representative images. For each cluster, when m representatives have been visited and no relevant image is found, the cluster has a high probability of being irrelevant to the search. Hence, it can be eliminated from the collection. On the contrary, if one of the representatives is relevant, with the expectation that more relevant images could be inside the corresponding cluster, the system keeps the cluster.

When a cluster is removed, there is space for other representative images to go on display. The unexplored part of the collection $\mathcal{I}_U$ contains images which are not in the removed clusters $\mathcal{I}_R$ and not in the kept clusters $\mathcal{I}_K$:

$$\mathcal{I}_U^{t+1} = \mathcal{I}_U^t \backslash (\mathcal{I}_K^t \cup \mathcal{I}_R^t), \mathcal{I}_U^0 = \mathcal{I}. \tag{4.11}$$

In each iteration, the system needs to cluster $\mathcal{I}_U^t$. This step cannot be done offline as it depends on the user actions. Hence, a fast and computationally inexpensive clustering algorithm is used namely competitive learning [95]. With new clusters available, the browsing is continued until a predefined number of prototypes has been found, denoted by iteration ∞. At that point, we have found a set of prototypes $I_{\mathcal{P}}^\infty$ and effectively reduced the collection to an active set $\mathcal{I}_A$:

$$\mathcal{I}_A = \mathcal{I} \backslash \mathcal{I}_R^\infty \tag{4.12}$$

with increased chance of finding relevant images in a later stage. For notational convenience the ∞ is dropped in the following sections.

4.3.2 The manipulation space

To provide relevance feedback a 2-dimensional manipulation space $\mathcal{M}$ is needed in which the user interacts with the images. Ideally, there is a direct relation between the similarities defined in the high dimensional space, being it feature space or dissimilarity space, and the manipulation space.

A projection from the high dimensional space to a 2-dimensional manipulation space is needed. Similarity based visualization, [71, 91, 78, 87] which aims to preserve the dissimilarities between every pair of images in the manipulation space is highly appropriate for this task.

Let x_i denote the position of image I_i in manipulation space. Furthermore, let $\mathcal{S}_{\mathcal{M}}(I_i, I_j)$ be the Euclidean distance between the images in manipulation space $\mathcal{M}$.

In similarity based visualization, to faithfully represent the dissimilarity space, $\mathcal{S}_{\mathcal{M}}$ should reflect the similarity $\mathcal{S}_{\mathcal{P}}$ in dissimilarity space. We define:

$$\Psi : \mathbb{S} \mapsto \mathcal{M} \tag{4.13}$$

With $\mathbb{S}$ a matrix containing dissimilarities. In chapter 2, we have compared four different projection techniques Ψ. The projection method giving the best performance in terms of preserving original relations is called ISOSNE, a combination of isometric mapping and stochastic neighbor embedding. ISOSNE is chosen as our Ψ.

ISOSNE contains two main steps (see chapter 2 for more details). A graph-based distance between images is first computed using k nearest neighbors. For each image I_i, the algorithm creates links to its k nearest neighbors based on Euclidean distance. Based on the graph, distances between images are redefined. If an image I_j is not in the k nearest neighbor list of I_i, i.e., there is no direct link between them, their distance will be computed via the intermediate links. Dijkstra's algorithm is employed to compute the shortest path between I_i and I_j. The second step is to project relations obtained from the graph based distance to the 2D manipulation space. To preserve the relations, the algorithm optimizes a cost function $\mathbf{C}$ measuring the difference between the probability distribution in $\mathcal{M}$ and the distribution in the original space, denoted as $P^{\mathcal{M}}$ and $P^{\mathcal{O}}$, respectively. Based on Kullback-Leibler distance, $\mathbf{C}$ is computed as:

$$\mathbf{C} = \sum_i \sum_j P_{ij}^{\mathcal{O}} \log \frac{P_{ij}^{\mathcal{O}}}{P_{ij}^{\mathcal{M}}} \tag{4.14}$$

where the probability distributions are calculated as follows:

$$P_{ij}^{(.)} = \frac{exp(-S_{(.)}^2(I_i, I_j))}{\sum_{l \neq i} exp(-S_{(.)}^2(I_i, I_l))} \tag{4.15}$$

with $S_{(.)}$ denoting similarity in the high dimensional original space, or a Euclidean distance in 2D between image I_i and I_j in $\mathcal{M}$.

To find the optimal placement of images in manipulation space, they are first initialized at random positions. These positions are then adjusted after each gradient descent iteration such that it reduces the cost function $\mathbf{C}$. When $\mathbf{C}$ is optimized, with positions found, distances between images $\mathcal{S}_{\mathcal{M}}$ are the ones optimally preserving the original relation in collection. Figure 4.3 shows two examples of displaying images in manipulation space with similarity preservation.

4.3.3 Automatic adjustment of dissimilarity space

At this point, the initial dissimilarity space is in place. The user has selected the set of prototypes $\mathcal{I}_{\mathcal{P}}$ treated as query examples to start the search process. The search task is now to find other relevant images using these examples.

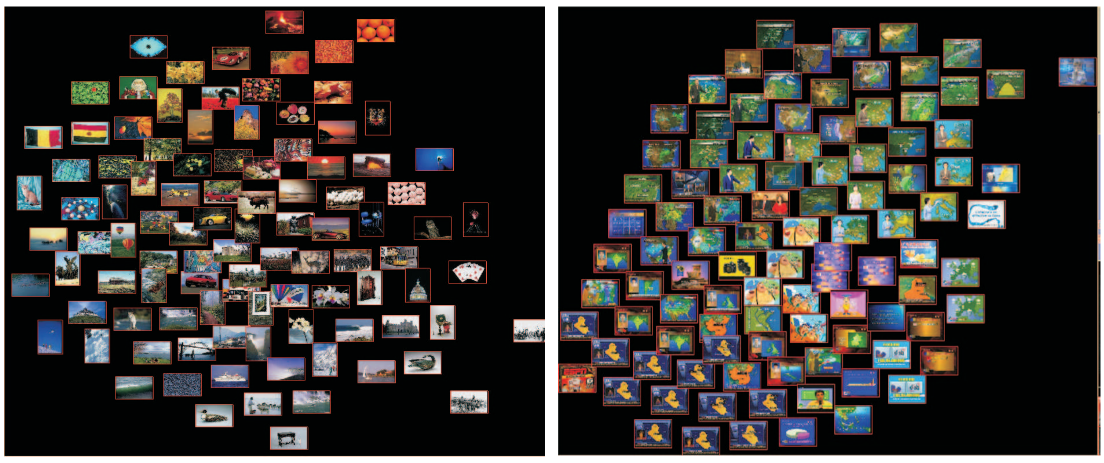


Figure 4.3: Two examples of similarity based visualization of images in the manipulation space. The layout of images is such that similar images are close. This allows for efficient interaction as images of the same class are likely to be grouped on the screen and can be selected by one user interaction.

4.3.4 Active learning

Different learning strategies can be employed [128]. We select active learning with support vector machines (SVM) for its capability of boosting retrieval results [113, 128, 79]. In interactive search, there is an unbalance between the size of the category searched for and the size of the collection. We therefore follow [62, 17, 77] and use one-class SVM.

The prototypes I_{P_i} are used as positive examples. From there, the one-class SVM defines a boundary $\mathcal{B}$ covering as much as possible the positive examples. Let us denote:

$\mathcal{B}_+$ the set of images inside $\mathcal{B}$ predicted to be relevant to the search.

$\mathcal{B}_-$ the set of images outside $\mathcal{B}$ predicted to be irrelevant to the search.

$\mathcal{I}_\mathcal{B}$ the set of images closest to $\mathcal{B}$ according to distance function $d_\mathcal{B}(.)$.

For SVM, $d_\mathcal{B}(.)$ is computed as a decision function that decides the probability of an image belonging to the relevant or irrelevant class.

In the next iterations, to improve the search, the system aims at refining $\mathcal{B}$. The refinement is such that it eliminates irrelevant images from $\mathcal{B}_+$ and adds new irrelevant images to $\mathcal{B}_-$. To do so, the images in $\mathcal{I}_\mathcal{B}$ are chosen as display set I_D^t. Thus, the images displayed are the ones for which classification is most uncertain. Feedback on those uncertain images yields the most information for improving the classification boundary. This is known as the “close-to-boundary” feedback approach [113, 79]. Figure 4.4 shows an example of images closest to the boundary, and figure 4.5 shows images in $\mathcal{B}_+$.

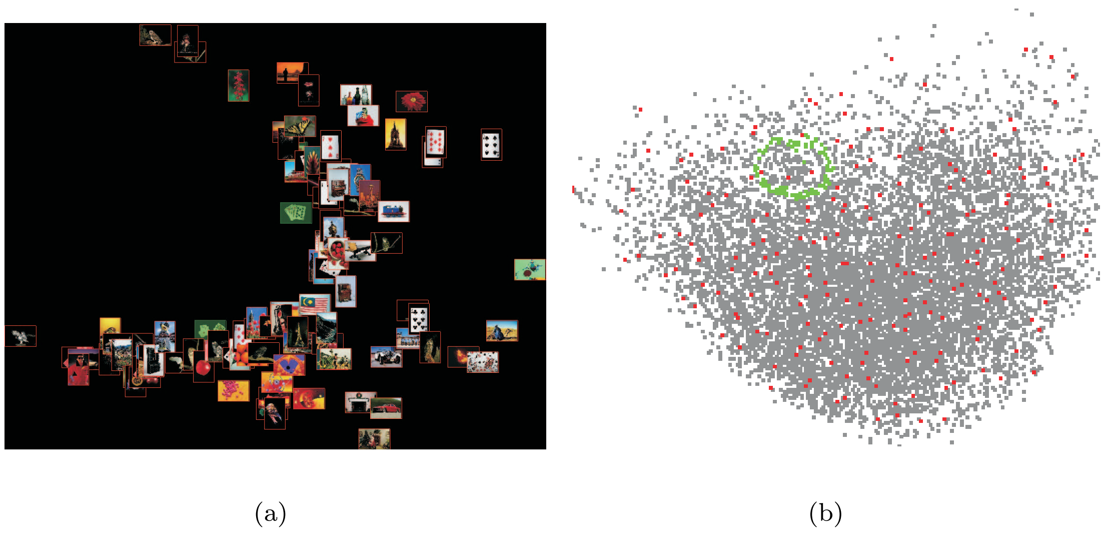


Figure 4.4: (a) An example of images closest to the border. (b) Idem represented by green dots in the full manipulation space.

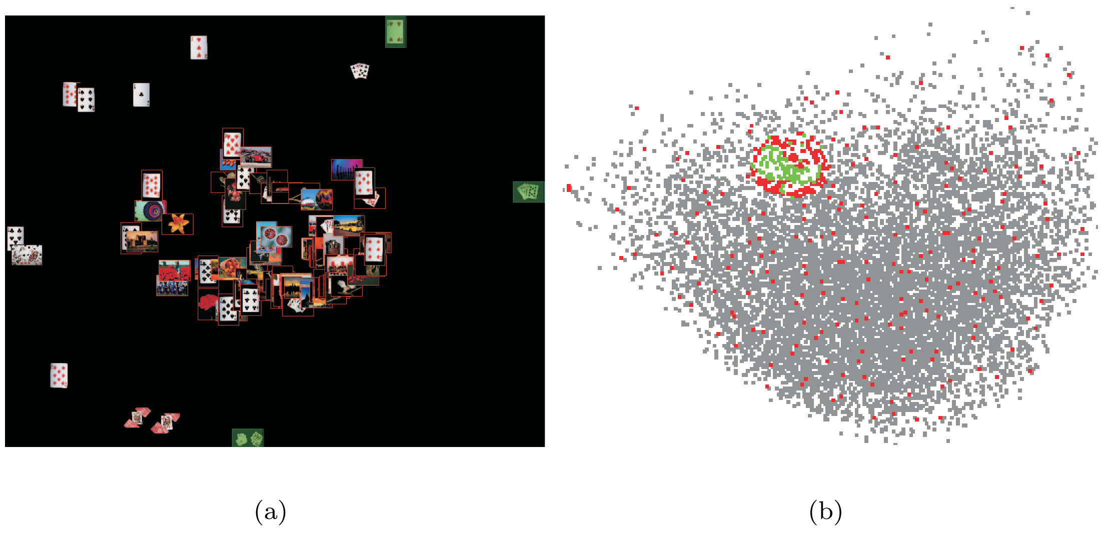


Figure 4.5: (a) An example of images inside the boundary. (b) Idem represented by green dots in the full manipulation space.

The user will label relevant images if they exist. Unlabelled images are treated as irrelevant and removed from the collection. Let

$$I_{\mathcal{B}^t_+} = I^t_D \cap I_+, \tag{4.16}$$

$$I_{\mathcal{B}_-^t} = I_D^t \backslash I_{\mathcal{B}_+^t} \tag{4.17}$$

When new feedback is given the SVM is recomputed on the new set of positive examples to update the boundary:

$$\mathcal{B}_+^{t+1} = (\mathcal{B}_+^t \cup I_{\mathcal{B}_+^t}) \backslash I_{\mathcal{B}_-^t} \tag{4.18}$$

The process is repeated until the user stops the search.

4.3.5 Similarity update

Initially, $\mathcal{D}_\mathcal{P}$ is the result of the projection of the dissimilarity matrix $\mathcal{S}_\mathcal{F}$ obtained in the feature space $\mathcal{F}$ (Eq.4.6). Hence, in general it does not correspond to the user desired semantic similarity. In that case, $\mathcal{D}_\mathcal{P}$ needs to be adjusted using user feedback. Therefore, in an iterative process the similarity is updated to better reflect the user's target similarity. To do so we adapt the update method from [39].

As indicated, $\mathcal{B}$ defines the currently predicted class boundary. For the dissimilarity update we make the assumption that the prediction is correct. For all images $I_i \in \mathcal{B}_+$ the representation $\mathcal{D}_\mathcal{P}(I_i)$ is kept constant. In contrast, for $I_i \in \mathcal{B}_-$ the representation is altered. In the ideal case, where $\mathcal{B}$ perfectly covers all relevant images, all other irrelevant images should be pushed away from the boundary. Of course, in practice, relevant images can be miss-classified. Pushing these images far away from the boundary makes it difficult to retrieve them later. As the likelihood of being relevant depends on the distance to the boundary $d_\mathcal{B}(.)$ we require that

$$\forall I_i, I_j \in \mathcal{B}_-\colon \text{ if } d_\mathcal{B}(I_i) < d_\mathcal{B}(I_j) \Rightarrow \mathcal{S}_\mathcal{P}(I_i, I_P) < \mathcal{S}(I_j, I_P) \tag{4.19}$$

and use this in the gradual change of the dissimilarity space based on the user feedback.

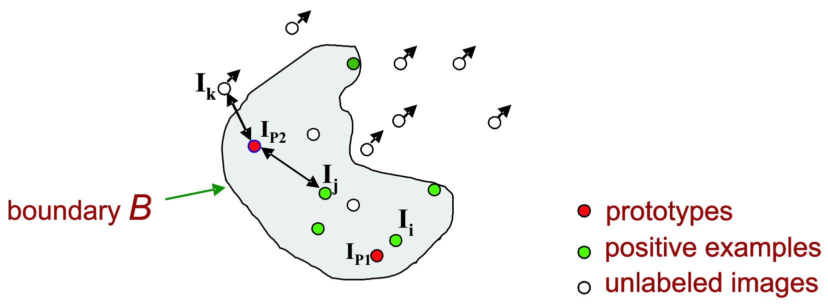


Figure 4.6: Illustration of similarity update. Assume a dissimilarity space created by 2 prototypes I_{P_1}, I_{P_2}. Let image I_j be less similar to $\mathcal{I}_\mathcal{P}$ compared to image I_i. After the update, image I_i will be more similar to $\mathcal{I}_\mathcal{P}$ than I_j. When simply using distance to the boundary, this is not the case. Images $I_k \notin \mathcal{B}_+$ will be pushed away from I_{P_1} and I_{P_2}.

The method in [39] (see fig. 4.6) satisfies the above constraints by using the following update function:

$$\mathcal{S_P}^{t+1}(I_i, I_P) = \begin{cases} \mathcal{S_P}^{t}(I_i, I_P) & \text{if } I_i \in \mathcal{B}_+, \\ \max\limits_{I_j \in \mathcal{B}_+} \mathcal{S_P}^{t}(I_j, I_P) + d_{\mathcal{B}}(I_i) & \text{otherwise.} \end{cases} \tag{4.20}$$

Hence, after learning with SVM and based on the new positions of images with respect to the boundary, their distances to the prototypes are changed. This leads to the adjustment of the dissimilarity space. These changes assure that irrelevant images will be pushed away, while the system keeps relevant images in the vicinity of the prototypes. In the next iteration the new similarity helps in better classification.

4.4 Experiments

4.4.1 Setup

4.4.1.1. Overview of experiments

We now present experiments to show the performance of our proposed approach.

The first experiment concentrates on the filtering component in the browsing phase of the proposed system. We will evaluate whether adding this component to the system will speed up the browsing through the collection for finding relevant images.

The second experiment considers the creation of the dissimilarity space. As described in section 4.3.1, to create $\mathcal{D_P}$ we need to determine the prototype set $\mathcal{I_P}$ and the dissimilarity between images and prototypes $\mathcal{S}(I_i, I_{P_i})$. At the beginning of the search, two spaces are available, namely the feature space $\mathcal{F}$ and the manipulation space $\mathcal{M}$.

To create a dissimilarity space both spaces can be used. We have the following two options for creating a dissimilarity space:

$$\Phi^1 : \mathcal{S_F} \xmapsto{\mathcal{I_P}} \mathcal{D}^r_{\mathcal{P}} \tag{4.21}$$

$$\Phi^2 : \mathcal{S}_{\mathcal{M}=\Psi(\mathcal{S_F})} \xmapsto{\mathcal{I_P}} \mathcal{D}^2_{\mathcal{P}} \tag{4.22}$$

where $\mathcal{D}^r_{\mathcal{P}}$ is the dissimilarity space based on r-dimensional prototypes, and $\mathcal{D}^2_{\mathcal{P}}$ the dissimilarity space based on 2-dimensional prototypes. This experiment leads to the choice of the proper dissimilarity space $\mathcal{D_P}$.

In the final experiment, the goal is to compare the search performance on the selected dissimilarity space against the feature space. For a fair comparison, the starting points are the same for both approaches.

4.4.1.2. Image collections

We select two different image collections. The first one is the well-known Corel collection. We select a set of 10000 images containing 100 non-overlapping categories of size 100 each.

The second collection is obtained from the TrecVid 2005 benchmark [105]. This set contains 43907 images, extracted from news video archives. We take the 29 different categories defined in [114] such as boat, basketball, car, chair to classify the collection . Other than the Corel collection, an image in this collection may be in more than one category. The number of images in each category varies from tens to thousands.

4.4.1.3. Features

For these two image collections, we extract the contexture feature set introduced in [114]. This feature set is evaluated as effective in learning the categorization of images. The authors define 15 proto-textures such as water, sky, snow. They learn the probability that an image contains the proto-textures. For the two collections in our experiment, 8 different parameter settings are used (spatial scale $\sigma = 1, \sigma = 3$ and different region sizes with ratios of $\frac{1}{2}$ and $\frac{1}{6}$ of the x and y dimensions of the image). For each image, we extract a feature vector containing 15 probabilities for 8 parameter values leading to a feature space of 120 dimensions. To obtain the manipulation space, ISOSNE is applied to project the feature space to 2D space. The Euclidean distance is used for comparing two feature vectors.

4.4.1.4. Evaluation criteria

For comparison, we define a baseline based on displaying pictures without any dissimilarity (or feature) computation where the system in each iteration displays a set of n_D randomly chosen images. Relevant images are selected if they are present in the displayed set. The baseline is calculated as the number of relevant images n^{t+1} likely to be found at iteration $t+1$. We have:

$$n^{t+1} = n^t + \frac{n_+ - n^t}{n - n^t} * n_D \tag{4.23}$$

where n is the size of the collection and n_+ is the total number of relevant images.

For the Corel collection, we have $n = 10000$, $n_D = 100$, and the value $n_+ = 100$ for each category. At the first iteration, the user will find one relevant image out of hundred on average. For the TrecVid2005 collection we average over all categories to obtain the baseline.

For evaluating the performance of the system, we report recall values R of the top ranked 100 images $\mathcal{I}^{100}$:

$$R = \frac{\|\mathcal{I}^{100} \bigcap \mathcal{I}_+\|}{\|\mathcal{I}_+\|}. \tag{4.24}$$

where $\|.\|$ denotes the size of a set. From there, we calculate the relative improvement measuring the improvement of a method over the baseline. Assume, a method X at iteration t yields a recall value R_X^t, the baseline at the same iteration returns a recall R_B^t. The relative improvement is:

$$\phi^t(X, B) = \frac{R_X^t - R_B^t}{R_B^t} * 100 \tag{4.25}$$

4.4.2 Experiments on prototype selection

To see the efficiency of the filtering, we test with the two given collections above. We report at each iteration the number of images removed, number of images kept, and number of missing relevant images when they fall into removed clusters. Finally, we average results over all categories, which is 100 for the Corel, and 29 for the TrecVid. The comparisons are between browsing through the collection with and without the filtering component. Selection of representative samples for a cluster by random selection is also examined.

We select $n^* = 10$ for our experiments. Table 4.1 and 4.2 show results.

	iterations	elements kept	missing relevant
no filtering	184	100%	0%
random subcluster	20	60%	13%
selected subcluster	23	61%	5%

Table 4.1: Results for browsing the Corel collection.

	iterations	elements kept	missing relevant
no filtering	> 500	100%	0%
random subcluster	46	67%	17%
selected subcluster	41	64%	8%

Table 4.2: Results for browsing the TrecVid collection.

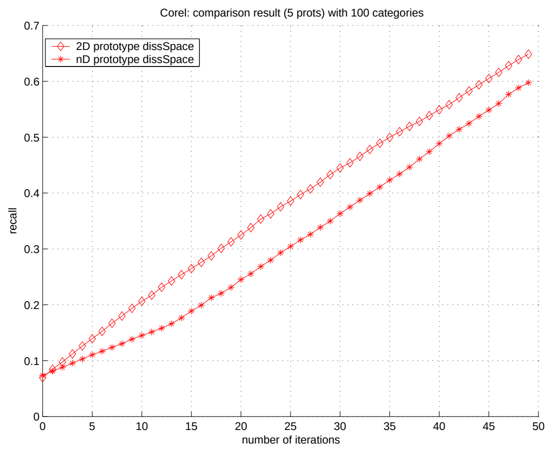
Of course, browsing without filtering will not miss any relevant image as all are visited. The drawback is that the total number of iterations needed is 10 times higher compared to the other two approaches. On average, with the Corel collection, normal browsing requires 184 iterations to check the whole collection, while the new approach needs only 23 iterations. With the TrecVid collection, the difference is even more significant with the number of iterations reducing to 41 while without filtering over 500 iterations are needed.

For a fair comparison, we average 10 different runs for the random approach. With the proposed approach, the number of missed relevant images is always smaller than random selection of representatives. Moreover, it is observed that the filtering can reduce the size of the collection significantly without loosing many of the relevant images. In the Corel collection, the size is reduced by 39%, while missing 5% of the relevant images. Reducing the size of the collection will certainly speed up the search. We believe it will also increase the precision and recall values. The same holds for the TrecVid collection, with a reduction of 36% of the size of the collection on average missing 8% of relevant images.

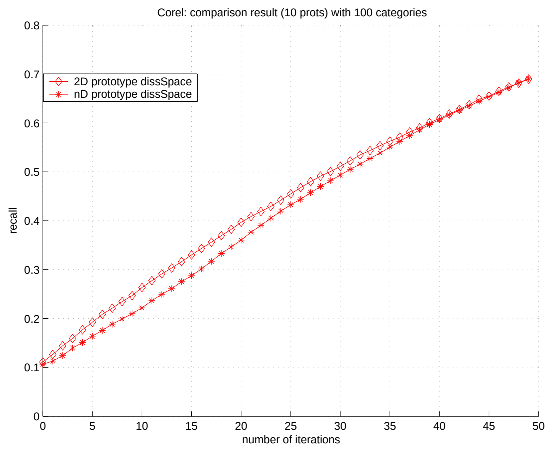
From this experiment, the conclusion is that adding filtering during browsing does indeed support the search process by reducing the size of the collection while keeping the chance of missing any relevant images to a minimum.

4.4.3 Experiment on the creation of dissimilarity space

To create the projection of the 120 dimensional feature space $\mathcal{F}$ to $\mathcal{M}$, the 2D manipulation space, we apply ISOSNE. We compute two dissimilarity spaces $\mathcal{D}_{\mathcal{P}}^{120}$ and $\mathcal{D}_{\mathcal{P}}^{2}$. To do so, first the prototype set $\mathcal{I}_{\mathcal{P}}$ is selected. We test with two sets of prototypes in the creation of $\mathcal{D}_{\mathcal{P}}$ with $p = 5$ or $p = 10$.



(a)



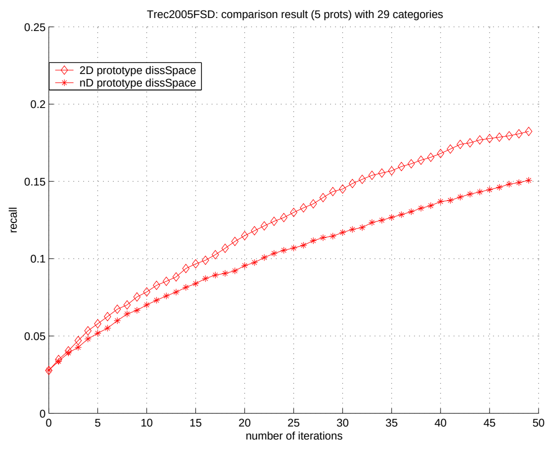
(b)

Figure 4.7: Using different dissimilarity spaces with the Corel collection averaged over 100 categories. (a) dissimilarity space created by 5 prototypes. (b) dissimilarity space created by 10 prototypes.

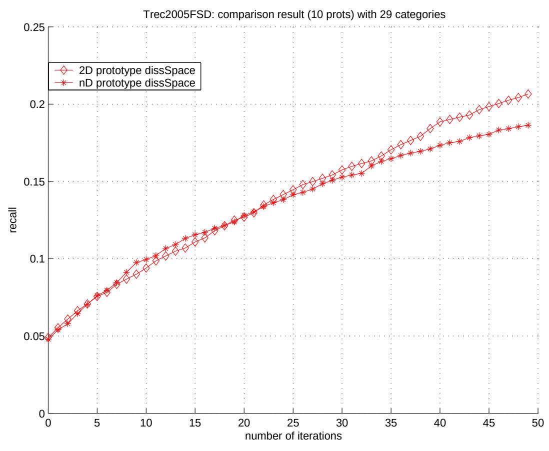
On each dissimilarity space, prototypes I_{P_i} are used as initial positive examples. The SVM produces a ranked list based on distances to the boundary. Recall values are reported based on the 100 top ranked images.

Figure 4.7 and 4.8 show the performance on $\mathcal{D}_{\mathcal{P}}^{120}$ and $\mathcal{D}_{\mathcal{P}}^{2}$. Based on equation 4.25, for both collections, the performance of learning on dissimilarity space is on average 60% relative improvement over the baseline.

Because of the projection of $\mathcal{I}$ from 120 dimensions to 2 dimensions for creating the manipulation space, relations between images cannot be kept perfectly even though the projection is optimal in keeping these relations. If the relation is not preserved on $\mathcal{M}$, performance of the $\mathcal{D}_{\mathcal{P}}^{2}$ will get worse when compared to $\mathcal{D}_{\mathcal{P}}^{120}$. However, it is interesting to observe from the results that the search performance on $\mathcal{D}_{\mathcal{P}}^{2}$ is always better than on $\mathcal{D}_{\mathcal{P}}^{120}$ whether having 5 or 10 prototypes. These results show that the ISOSNE performs very well in preserving relations between images. Moreover, because of ISOSNE extracts the structure of the collection by first computing the graph-based distance, it takes an advantage over the direct distance computation on the feature space. In other words, dissimilarity between images in the feature space is computed by directly comparing two feature vectors, whereas in the manipulation space, as a results of ISOSNE, dissimilarity is obtained by preserving a graph-based



(a)



(b)

Figure 4.8: Using different dissimilarity spaces with the TrecVid collection averaged over 29 categories. (a) dissimilarity space created by 5 prototypes. (b) dissimilarity space created by 10 prototypes.

distance on the feature space. That explains why the performance of learning on $\mathcal{D}^2_{\mathcal{P}}$ is better than learning on $\mathcal{D}^{120}_{\mathcal{P}}$.

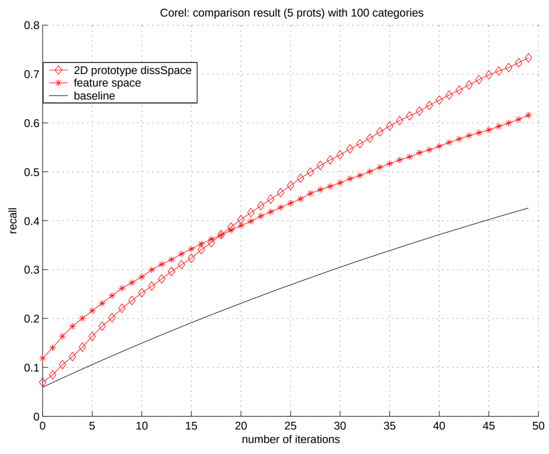
For the selection of a dissimilarity space, we prefer using $\mathcal{D}^2_{\mathcal{P}}$. Moreover so because it establishes a direct link between distances in dissimilarity space and manipulation space.

4.4.4 Experiment on direct manipulation of dissimilarity space vs. indirectly via feature space

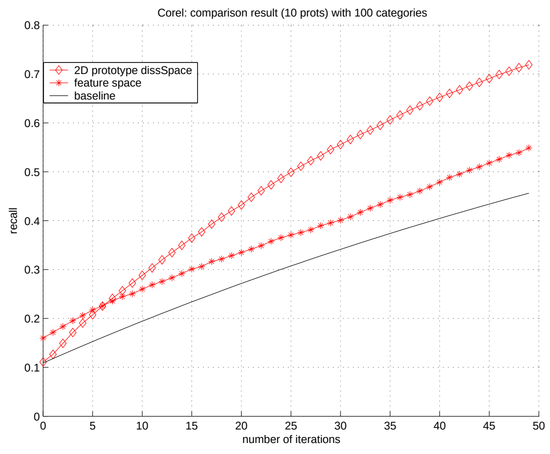
We compare two ways of updating the dissimilarity matrix, via $\mathcal{D}^2_{\mathcal{P}}$ as we propose, or via feature space $\mathcal{F}$ [7, 71, 58, 123].

Results are shown in figure 4.9 and 4.10 for the Corel and TrecVid. The figures show that with small number of prototypes, the dissimilarity space is not able to maintain the relations between images. Therefore, the improvement of learning on the dissimilarity space is smaller than learning via the feature space. With 10 prototypes, the dissimilarity space covers the image collection better. Hence, on average it gives a higher improvement.

With a smaller number of prototypes, i.e., smaller number of initial positive examples, the performance of the baseline is worse than the one with higher number of examples. From the figures, average performance of learning via feature space will get worse when starting from more initial examples. Because the prototype set $\mathcal{I}_{\mathcal{P}}$ is chosen such that it distributes over the collection when p gets higher, this set better covers the collection. In the feature space, this means that the more prototypes, the broader the boundary $\mathcal{B}$. This leads to a higher number of irrelevant images inside

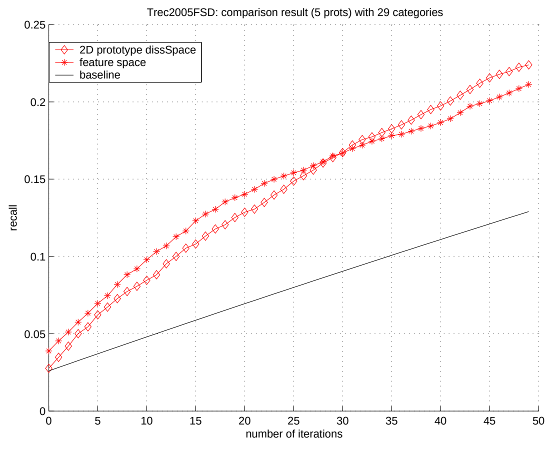


(a)

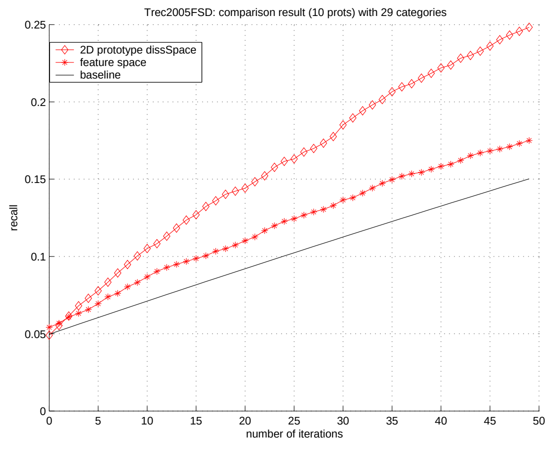


(b)

Figure 4.9: Comparison of direct learning on dissimilarity spaces and learning via feature space with the Corel collection averaged over 100 categories. The results are evaluated by recall. (a) dissimilarity space created by 5 prototypes. (b+d) dissimilarity space created by 10 prototypes.



(a)



(b)

Figure 4.10: Comparison of direct learning on dissimilarity spaces and learning via feature space with the TrecVid collection averaged over 29 categories. The results are evaluated by recall. (a) dissimilarity space created by 5 prototypes. (b) dissimilarity space created by 10 prototypes.

$\mathcal{B}$. This is a main disadvantage of using a feature space since they are not capable of capturing semantic categorization. When a dissimilarity space is created from $\mathcal{I}_{\mathcal{P}}$, the set of initial examples groups positive images together. Therefore, the performance of learning on dissimilarity space is improved.

From this experiment, we conclude that number of prototypes should not be too small. Ten prototypes is reasonable for creating the dissimilarity space as well as reasonable in the interactive search where few positive examples are provided. The learning performance on the dissimilarity space is better than updating dissimilarity via feature space.

4.5 Conclusion

In this chapter, we have proposed a new approach in interactively learning dissimilarity. Different from existing techniques [7, 71, 58, 123] we directly learn the dissimilarity space from user's feedback. This means that instead of collecting a large set of features or choosing problem-specific features, only the relations between images are used. By doing so, we avoid the computational problem occurring for large sets of features and the difficulty in selecting effective features in interactive category search. Representing images by their relations to others is close to the perceptual meaning of those images, which is difficult to obtain using feature representations.

We have demonstrated by experiment that learning in this dissimilarity space $\mathcal{D}_{\mathcal{P}}$ in general gives a better performance than the learning on feature space $\mathcal{F}$, when a reasonable number of initial prototypes $\mathcal{I}_{\mathcal{P}}$ is being used.

For the selection of prototypes, we present a browsing technique with hierarchical clustering and filtering components. This browsing strategy can speed up searching for relevant images, and in the mean time filter the collection by removing irrelevant clusters. This means that we are able to get a better chance of finding relevant images. In our experiments, we showed that the number of iterations needed to browse through the collection reduces by a factor of 10 for both the Corel and the TrecVid collection.

We present a method of updating the dissimilarity space using relevance feedback and active learning with one-class SVM. This adjustment assures closeness of relevant images to the classification boundary, while pushing away irrelevant ones. Experimental results show that our proposed approach outperforms the others there the relative improvement over the baseline is on average 60% for both the Corel and the TrecVid collection.

Finally, we have implemented an interactive search system, based on the proposed methodology (this system is part of the MediaMill search system [106]). Figure 4.11 shows a screenshot of our system.

In conclusion, interactive learning on dissimilarity space $\mathcal{D}_{\mathcal{P}}$ rather than via feature space $\mathcal{F}$ is very promising. The simplicity in creating dissimilarity space and its performance has great potential for many tasks in content based retrieval. In this chapter, our proposed approach in improving the interactive search in CBIR by updating the dissimilarity space gives the best performance. Moreover, as the manipulation space $\mathcal{M}$ reflects the structure of the collection on the dissimilarity space $\mathcal{D}_{\mathcal{P}}$, user interaction on $\mathcal{M}$ will directly influence the adjustment of $\mathcal{D}_{\mathcal{P}}$ allowing for intuitive interaction.

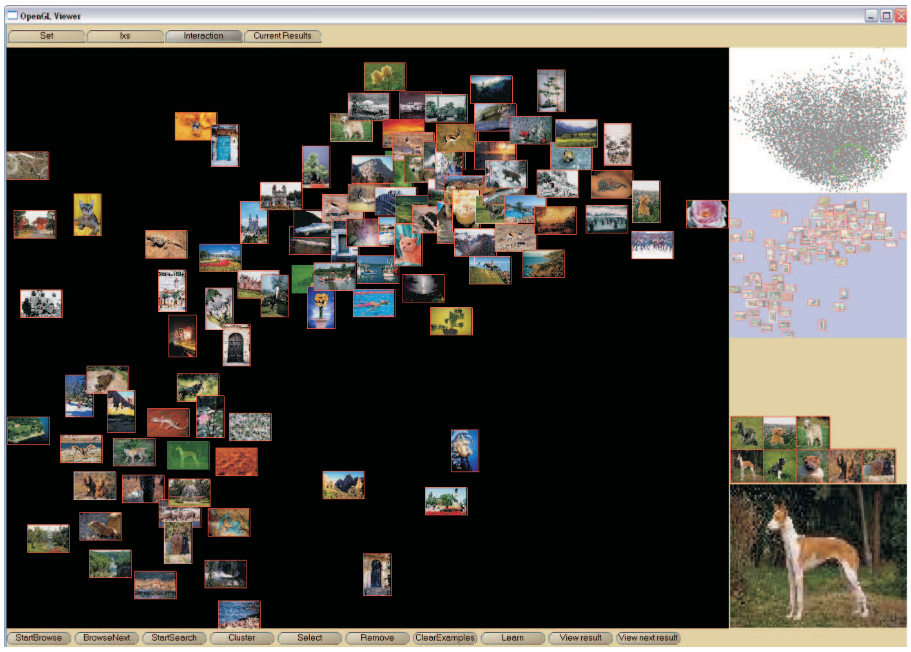


Figure 4.11: A screen shot of the system when searching the Corel collection. The system contains 4 parts. The main window represents the part of manipulation space displayed, where the user interacts with images. This screenshot captures one step of the learning stage with displayed images being the ones closest to the classification boundary. The top-right window gives an overview of the whole manipulation space. The bottom-right corner window shows the full size of the current image. During the interaction, relevant images selected by the user will be kept in the middle window on the right side. For selection of positive examples, the 2D similarity based visualization shows the advantage over the grid based display. For example, instead of selecting one image at a time, in 2D similarity-based visualization similar images stay close together, therefore the user can select a group of images at a time. Furthermore, as distances among images on the screen have a direct relation to distances in the dissimilarity space the interaction is very intuitive.

Acknowledgments

The work is within the ImIk project (Interactive disclosure of Multimedia Information and Knowledge) sponsored by IOP MMI (Man-Machine Interaction). The authors thank J. van Gemert for providing the contexture features.

Chapter 5

Relevance feedback based saliency adaptation in CBIR

5.1 Introduction

Content based image retrieval (CBIR) has been under investigation for a long time and many systems were built to cater for the varying demands posed by different applications. An extensive review can be found in [104]. Despite of the large number of systems developed, a gap remains between the user's expectation and the system's retrieval capabilities. The main reason for this is the semantic gap in [104] defined as:

"The semantic gap is the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for the user in a given situation."

The gap can be limited by using even more advanced features [104], [93], by introduction of ontologies for structuring the possible user queries [102], or by using additional resources like associated textual information [4]. However, as the interpretation is subjective, a gap will remain. Thus, user interaction is an essential part of any practical CBIR system.

In [104], a framework for interactive CBIR called query space is defined capturing all of the state-of-the-art interaction mechanisms. The query space is defined as $\mathcal{Q} = \{I_{\mathcal{Q}}, F_{\mathcal{Q}}, S_{\mathcal{Q}}, Z_{\mathcal{Q}}\}$. The first component is the set of active images $I_{\mathcal{Q}}$. The second one is a selection of features $F_{\mathcal{Q}}$. The third one is a similarity function $S_{\mathcal{Q}}$ used to compare images in the database. The last one is a set of symbolic labels $Z_{\mathcal{Q}}$ with an associated probability for each image. The retrieval process in query space consists of five main steps. The first one is *initialization* in which the system initiates a query space. A query is then defined in the *specification* step. In the *visualization* step, retrieved results of the query are mapped into a n-dimensional display space.

This chapter is published in the ACM Multimedia Systems [76]

In the *feedback* step the user gives relevance feedback RF_i by changing one or more components in query space $\mathcal{Q}$:

$$\{I_Q^i, F_Q^i, S_Q^i, Z_Q^i\} \overset{RF_i}{\rightarrow} \{I_Q^{i+1}, F_Q^{i+1}, S_Q^{i+1}, Z_Q^{i+1}\} \tag{5.1}$$

The final results are returned in the *output* step.

Existing interactive systems mostly focus on changing I, S or Z. For example, Vendrig focusses on changing I [117]. In [92] positive and negative examples are used in the similarity function S to update the weighting of different features. In [97], the user changes the relative position of the images in visualization space to update similarity. Minka [70] learns labels in Z like grass, and brick from the user feedback. In [65], the user redefines the label of the object obtained by the system such as names of people, or special type of animals.

Few systems support user feedback on the features. QBIC [34] has the user select features such as color, texture, and shape. The Pictoseek system [37] allows users to decide between different color spaces like RGB, HSI, rgb. However, these selections have to be made beforehand by the user, they cannot be changed during the course of interaction except from starting all over again. Dynamic feature selection was proposed in [92], but only for global features.

An alternative to global features are salient details to represent the image content where salient details can be points [45], [68], [112], lines [60], [88], or regions [27], [121], [80].

These methods are focussed on automatically summarizing the image into a set of salient details. More precisely, they use a fixed definition of saliency. However, the saliency is user and context dependent and thus should be defined by the user through interaction.

Therefore, we aim at interactive definition of saliency. To that end, we need to analyze existing detail detection methods and summarize them into a unifying framework. The framework is presented in order to find out which elements are involved in changing the output results. These are called *tunable parameters*. The main idea of our method is to tune those parameters to best fit the user feedback. From there, we present an application of our approach in CBIR. Thus our approach follows the pattern:

$$\{I_Q, F_Q^i, S_Q, Z_Q\} \overset{RF_i}{\rightarrow} \{I_Q, F_Q^{i+1}, S_Q, Z_Q\} \tag{5.2}$$

To do so the system is composed of an off-line stage to precompute candidate salient details and an interactive stage in which the user is doing the interactive retrieval. It yields a general framework for interactive retrieval based on salient details. Thus it can be used to create an add-on to existing methods rather than replacing them.

This chapter is organized as follows. Section 5.2 describes the *off-line stage*. This section is also a review of existing salient detail detectors in order to find the tunable parameters for the *interactive stage*. Different sets of candidate salient details are extracted by changing those parameters. In section 5.3, this second stage of our framework is presented with methods to tune the parameters based on user feedback. Section 5.4 shows results to demonstrate the new approach.

5.2 Off-line salient detail detection

Although the salient detail detectors in literature follow different approaches, every method can be divided into five main steps: image processing, detail detection, feature computation, saliency computation and selection based on significance, which we will now define more precisely.

Image processing is the step where both the input I and output I^* are one or more images. Image processing removes irrelevant information like noise, or enhances specific image content like edges or contrast. An image processing operator is denoted by $e_{\vec{\sigma}}$ where $\vec{\sigma}$ is an element in the space Σ of parameters steering the process. Thus we denote the image processing step by

$$I^* = e_{\vec{\sigma}}(I). \tag{5.3}$$

Detail detection is the process where the image is decomposed into details $\mathcal{D}$, where the details can be regions, lines or points. By adjusting the set of parameters $\vec{\omega}$ in parameter space Ω, different sets of details are obtained. The detail detection step is given as:

$$\mathcal{D} = p_{\vec{\omega}}(I^*) \tag{5.4}$$

where $p_{\vec{\omega}}$ is a specific segmentation method.

Feature computation calculates the set of feature values $\mathcal{F}$ over all detected details. A feature can be any description of a detail, e.g., based on color, texture, shape or size. We denote feature computation for a set of details $\mathcal{D}$ as

$$\mathcal{F} = f_{\vec{\lambda}}(\mathcal{D}) \tag{5.5}$$

where $\vec{\lambda}$ is a set of parameters in parameter space Λ to compute the feature f.

Saliency computation calculates how salient the features of a detail are relative to other details within its spatial context. Thus it needs to define a local area $\pi \in \Pi$ in which feature values of other details are considered. In literature most of the methods consider π to cover effectively the whole image, thus ignoring local spatial context. We define it in the more general form where within the context defined by π this step typically uses an operator l to compute the local extrema. It could further enhance the saliency of the detail by normalizing the saliency inside the context. An example of this is shown in figure 5.2. We denote it by $l_{\vec{\pi}}(\mathcal{D})$ where $\vec{\pi}$ defines the interval or area used in determining the saliency of a certain detail. This step will select details standing out from the other details in the set $\mathcal{D}$. The general form of this step is

$$\mathcal{L} = l_{\vec{\pi}}(\mathcal{F}). \tag{5.6}$$

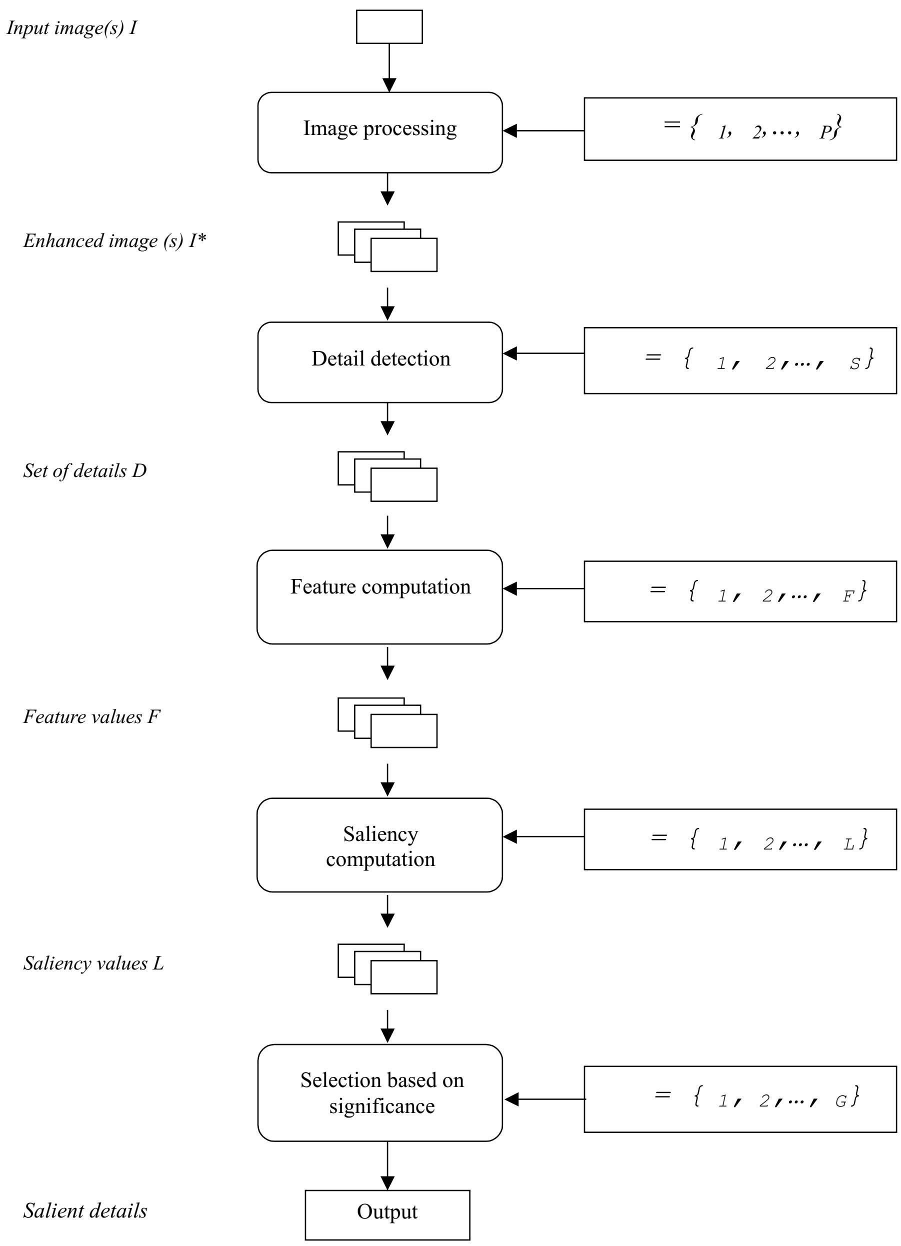


Figure 5.1: Framework for offline salient detail detectors.

Selection based on significance is the step which finally selects details $\hat{\mathcal{D}}$ based on the saliency values computed in the previous step. It reveals the most salient parts of the image, e.g., by defining a given threshold on the saliency values or by restricting the number of output salient details. The selection function is denoted by $g_{\vec{\gamma}}$ with $\vec{\gamma} \in \Gamma$. It gives:

$$\hat{\mathcal{D}} = g_{\vec{\gamma}}(\mathcal{L}). \tag{5.7}$$

Going through the general scheme, the whole process of automatic salient detail detection can be described as:

$$\hat{\mathcal{D}} = g_{\vec{\gamma}} \circ l_{\vec{\pi}} \circ f_{\vec{\lambda}} \circ p_{\vec{\omega}} \circ e_{\vec{\sigma}}(I) \tag{5.8}$$

where I is the input image, and $\hat{\mathcal{D}}$ is the final output set of salient details, $\hat{\mathcal{D}} \subseteq \mathcal{D}$.

As stated each step has its own set of parameters. So for each salient detail detector we have a set of parameters $\theta = (\vec{\sigma}, \vec{\omega}, \vec{\lambda}, \vec{\pi}, \vec{\gamma})$. By changing any of the parameters in θ, we get a different set of salient details.

Figure 5.1 shows the model for the salient detail extraction process. Each step is described in more detail in the sequel by considering how it is defined in existing methods:

5.2.1 Image processing

Image processing enhances specific image characteristics. Examples are smoothing, gradient computation, or enhancement of line-like structures. These methods all involve a scale parameter [108] [3], [60], [48].

Recognizing that the proper scale is dependent on the user need, scale based methods consider a range of parameter values. Different scale spaces have been defined. Sebe et al. [112] use the wavelet transform to represent image variations at different scales. Another similar approach considers the input image(s) at different resolutions, but with different features. As an example in [27], luminance, color and texture at several scales are computed at every position in the image. The strength of inhomogeneities of luminance, color and textures are used as indicators of edge evidence. An accumulated edge evidence map (AEEM) for different scales and different local measures is then created. Another example is in [46], where Itti builds a scale space where for every pixel in the image he computes color, intensity and orientation. Along the same line, Salah et al. [96] use line orientations as the features at different scales. In [120], Walker uses normalized Gaussian derivative kernels with different scales to construct the differential structure of the image.

In summary, the process of salient detail detection, at this step, represents the input image I as one or more images in which specific characteristics are emphasized, but with scale as the most important parameter..

5.2.2 Detail detection

Detail detection involves the segmentation of images into a set of details namely points, lines or regions. This step may use statistical classification [120], edge detection [45], region detection [27] or a combination of these techniques [51]. Each method has its own parameters.

Regarding point detection, Schmid [100] gives a comprehensive overview. Examples of methods segmenting images into points are also given in [45]. The author first segments the image into lines, and then finds the "cotermination" of pairs of lines under some constraints.

Also in [45], Qasim uses a perceptual grouping algorithm to segment the image into line based details such as: line segments, L-junctions, and U-junctions. Based on the intensity image, pixels are grouped into “line-support” regions if they have a similar gradient orientation. Other examples are in [60], [3].

For region detection, a typical example is the method in Blobworld [14]. After a set of features are computed at each pixel at a selected scale, the system then groups pixels into regions using the EM algorithm. Hoang in [44] uses k-means clustering to group pixels in the image feature space to find homogeneous regions. In [21], Cinque et al. segment the image into regions using a region-growing algorithm. Each time a pixel is added to a region, its four nearest neighbors that have not been processed are considered. A threshold on the color difference is used to decide whether a point belongs to the region or not. In the reference, they also merge two nearby regions if the difference between their mean colors is less than a threshold. These two thresholds determine the region segmentation results.

5.2.3 Feature computation

Features can be any description of the details. Examples are the color histogram of a region, the length of a line, or the curvature at a point on a curve. Changing the features used and steering their computation gives another possibility for changing the final set of salient details.

In [69], where the image with candidate points is represented in a scale-space, the scale invariance of points is computed as feature. In [112], for every points extracted in the previous step using wavelets, the feature computed at each point the sum of the coefficients along the trace from the coarse level to the finer one. This method thus depends on the wavelet function selected.

Another approach for computing features of points is described in [120]. Several vectors of invariants are formed at each candidate point in the image. These vectors build a multivariate distribution. At each point, they estimate the local density in a distribution by summing the contribution from a mixture of Gaussians. The density estimated for each vector of invariants is used as a feature at the appropriate scale.

In the case of lines, [3] uses the Gestalt’s principles. They calculate the following features of curves: the length of the curve, the total curvature or energy, and the amount of fragmentation. In [60], they use the expected number of closed contours that pass through an edge as its feature.

In [45], the length of a line is used. In [88], he presents some features of edges such as life-time, the time that an edge persists before disappearing during the blurring.

Feature computation for regions is presented in [27], [65], [80], [46]. Cinque [21] simply uses the mean color of a region. In [27], Dimai computes the strength of inhomogeneities of luminance, color and texture as feature of the region considered.

The feature computation step is very important to provide good candidates for the next step. This step depends on the segmentation step as it needs correctly segmented details. The most important parameter is the choice of the right feature to compute for each detail.

5.2.4 Saliency computation

For humans saliency is always defined relative to a neighborhood thus saliency should be computed locally. However, most of the methods compute saliency as global saliency based on the feature values. This leads to missing local salient details which in the whole image may not seem significant, but within a local area become salient. A particular example of locally defined saliency is given in figure 5.2.

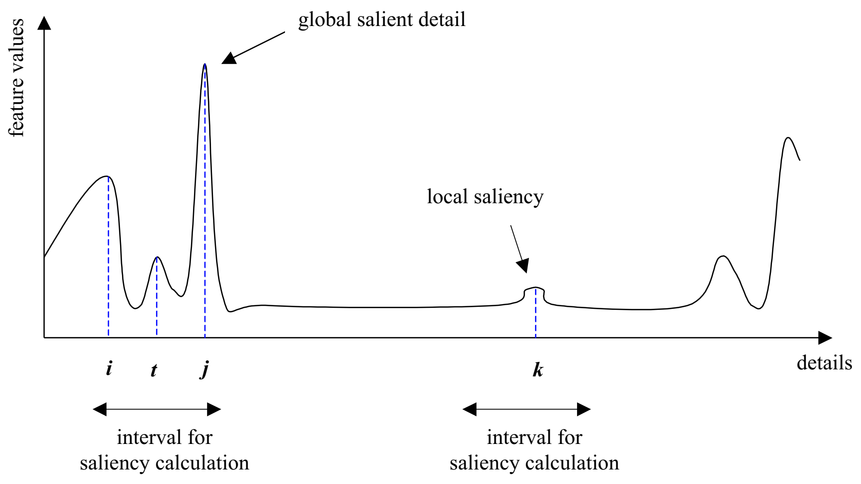


Figure 5.2: Example of local saliency. For global saliency computation, detail k will not be considered salient. Within the interval, the local maxima detector returns k as a salient detail since compared to its neighbor it exhibits a significant change in feature value.

It follows that the typical parameter for this step is Π determining the area or the interval where the saliency is considered [69], [100]. For example, in [100], they first apply the Harris detector for extracting corner points. The authors then use relatively small circles around each corner to compute the saliency of that point based on variance under rotations. In the special case where Π denotes the whole image, it boils down to global saliency computation.

Hence, in the output of this step, for each candidate detail d, the saliency value is $l_{\vec{\pi}}(d)$.

5.2.5 Selection based on significance

For describing an image one should select a limited set composed of the most relevant salient details. Given the saliency based on feature values, we classify existing methods in the following two types. The first type employs a simple way of selection by using a threshold on the number of details. This ignores the varying complexity of different images. The second type is more natural using a threshold δ on the saliency values [9], [21], [45]. For example, in [45], a threshold is used to select the longest lines. In general, this approach can be represented as follows:

$$\hat{\mathcal{D}} = \{ \ d \subset \mathcal{D} \mid l_{\vec{\pi}}(d) > \delta, \delta \in \mathbb{R} \ \}. \tag{5.9}$$

The following is the special case requiring no parameters where only the most salient detail will be returned:

$$\hat{\mathcal{D}} = \underset{d \in \mathcal{D}}{argmax} l_{\vec{\pi}}(d). \tag{5.10}$$

5.3 Interacting with salient details

In the *interactive stage*, the aim is to take the offline data as basis and find the details which are salient from the perspective of the user. At this point we should make a distinction between salient details for which visual evidence is present in the image and details for which this is not the case. The latter can only be found by the system using prior knowledge on the shape [3]. We focus on the former case, and hence make the assumption that for a user desired salient detail the evidence can be found for some parameter setting. However, due to the *"sematic gap"*, the system cannot define automatic methods for setting those parameters in a general domain. Support by the user to find the most appropriate parameters is required.

5.3.1 Processing steps

In current systems changing the features, if possible at all, is done by manipulating the parameters and visualizing the result. This is acceptable for a computer vision expert, but being able to control the system in such a manner is not feasible for the end-user. Therefore, rather than having the user manipulate the parameters directly we let the user give relevance feedback on the results obtained. Based on this user interaction, the system then has to select the optimal parameters accordingly.

Thus, the *interactive stage* consists of four steps: initialization of the result, visualization of the result by using default parameters, relevance feedback and parameter adjustment which will now be described. Figure 5.3 presents the whole framework composed of the two stages.

Initialization This step starts the interactive stage with default parameters θ_0 as suggested by the original methods. Those values are sent to the database and the system returns the default salient detail set $\hat{\mathcal{D}}^0$.

Visualization In this step, the details and their properties are visualized to help the user in giving useful relevance feedback. In the simplest case, points and lines are shown with their positions, regions with their boundaries overlaid on the image. A more advanced method would also show why these details are considered more salient than the others.

Relevance feedback By interacting with the objects visualized, the user gives relevance feedback RF_i to the system. A number of possibilities are indicating positive/negative examples, denoting a degree of (ir)relevance with respect to the target,

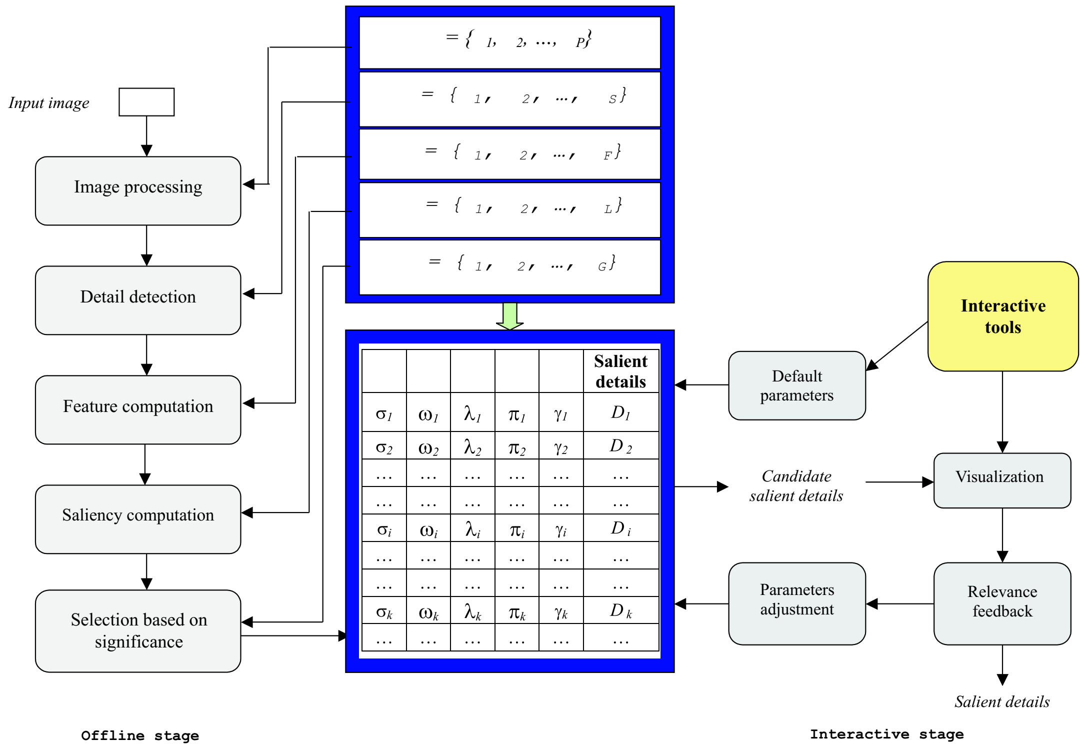


Figure 5.3: Framework of interacting with salient details

changing positions, or indicating region of interest, merging or splitting [94], [64], [73], [59]. The general form of this step is:

$$\hat{\mathcal{D}}_{\theta^i} \xrightarrow{RF_i} \hat{\mathcal{D}}_{\theta^{i+1}}, \tag{5.11}$$

where $\hat{\mathcal{D}}_{\theta^i}$ is the current set of salient details, and $\hat{\mathcal{D}}_{\theta^{i+1}}$ is the set returned after the relevance feedback.

Parameter adjustment Given the user's feedback RF_i the system should find a set of parameters $\hat{\theta}^{i+1}$ optimizing some criterion. Let us consider an example in the case of points. When the user selects a region of interest, the new set of salient points should contain as much points as possible in the interest region while limiting the number of points in other regions. A general form for this is given by:

$$\hat{\theta}^{i+1} = \underset{\theta}{argmax}\, \mathrm{E}_{RF_i}\left(\hat{\mathcal{D}}_{\theta^{i+1}}\right), \tag{5.12}$$

where $\hat{\mathcal{D}}_{\theta^{i+1}}$ is the set of salient details returned with parameter θ^{i+1}, E_{RF_i} is an evaluation function on the output salient details. The choice for this function depends on the specific application. A general criterion is that the new set of salient details should be closer to the user expected result than the previous one.

To avoid the user becoming disoriented by large changes a constraint $\mathcal{C}$ is added. In other words, the constraint assures a smooth path to the optimal query. For example, a constraint could be that the size of new set of salient details should not be much larger than the previous set. The general constraint is represented as:

$$\mathcal{C} = \mathcal{C}\left(\hat{\mathcal{D}}_{\theta^i}, \hat{\mathcal{D}}_{\theta^{i+1}}\right). \tag{5.13}$$

From 5.12 and 5.13 this parameter adjustment is described as:

$$\hat{\theta}^{i+1} = \underset{\theta}{argmax}\ \mathrm{E}_{RF_i}\left(\hat{\mathcal{D}}_{\theta^{i+1}} | \mathcal{C}\right), \tag{5.14}$$

After *parameter adjustment*, the system finds the optimal set of parameters. These values will be sent to the database, after which a new set of candidates is returned. The process is repeated until the user accepts the results.

5.3.2 Offline computation

We have shown that salient details can be computed using a five step process where each step is steered by a set of parameters. For interaction it is not feasible to do all the computation at run-time. Therefore for a given range of the parameter values we pre-compute the salient details and store them in a database in the offline computation step. At each step, we determine the most important parameter, while keeping the other parameters to their default values. Computing details beforehand leads to the problem of storing the offline data. As only the resulting details have to be stored the storage requirements are moderate compared to the space required for the image itself. For example in the case of salient points, only their coordinates are stored for each parameter setting. In the case of salient regions, the storage is also moderated as for each parameter setting an identification image is stored where the value corresponds to a region. As the number of regions is limited, the identification image of a RGB image is on average 3KB only (in PNG format with the size of an image is 384x256 or 256x384).

Assume the setting for a parameter is 10 different values. In the worst case when the method employs each of the five step a parameter, the size of Σ is then 10^5. Each segmented image has size 3KB, so for example with 10000 images, total storage is 3GB. It seems that the setting for each parameter is unlimited. In practice, those parameters should not be too different from the default ones since we meant to adjust them to suit a specific query. As in our experimental results, the setting takes the default parameters as a standard point then decreases and/or increases within certain range to get the new set of parameters. Hence, this will limit the explosion of possible combination between parameters.

In the following the pre-computed data for a given input image is called the *offline data* for that image. The size of the *offline data* is $O(N_{\mathrm{it}})$:

$$N_{\mathrm{it}} = (||\Sigma|| \times ||\Omega|| \times ||\Lambda|| \times ||\Pi|| \times ||\Gamma||),$$

where the $||\cdot||$ denotes the size of a set.

When the dataset grows beyond 10.000 images we should consider the use of special data structures to make the access to the offline data more efficient. This depends on the type of relevance feedback given by the user, the specific function chosen in equation 5.12 and the constraint function chosen in equation 5.13. With a carefully chosen constraint it is feasible to obtain a considerable pruning of the search region in parameter space. That is, for every image and for every parameter setting we can precompute which other parameter settings fall within the constraint. Links to those parameter settings can be stored in an index. Only if the initial query image is chosen outside the dataset, we have to go through the whole parameter space.

5.4 Instantiations of the framework

In the previous section a general framework for interacting with salient features has been defined. To show its validity we now introduce three example instantiations of the framework to show how it can be used to add an interaction step to existing methods. Each example illustrates different aspects of the framework to show how it can be used to add an interaction step to existing methods. These examples are respectively based on points, lines and regions. Finally, we show how to apply the latter as the query redefinition step in content based image retrieval.

5.4.1 Interacting with salient points

For offline salient point detection, the Harris detector is used [55]. The Harris detector has a set of parameters $\theta = \theta(\sigma, r, t)$ where σ is the standard deviation of the smoothing Gaussian used in the image processing step, r is the radius of mask's size considered for local maxima in the detail detection step, and t is the threshold for selecting output salient points.

The *off-line stage* as described above stores an archive of possible sets of salient points by computing results for $\sigma = \{1.0, 2.0, 3.0, 4.0\}$, $r = \{1.0, 2.0, 3.0, 4.0\}$, and $t = \{100, 200,$
$500, 800, 1000, 1100, 1200, 1500, 1800, 2000, 2500\}$. The default set of parameters is given by $\theta^0 = (1.0, 1000, 1.0)$ [55]. In this example, the relevance feedback is the selection of a region of interest.

Given an input image with a set of points extracted by the initialization step, the user then selects a region of interest $\mathcal{T}$ by drawing an rectangle with points inside taken as positive examples. This relevance feedback is used to search through the off-line database to find the optimal result.

The updating step searches for a θ^{i+1} satisfying equation 5.12. In this experiment, we aim at finding many points inside the region of interest, and few points outside. Hence, the system finds θ^{i+1} such that:

$$\theta^{i+1} = \underset{\theta}{argmax} \left(\frac{||P_{in}(\theta)||}{||P_{in}(\theta)|| + ||P_{out}(\theta)||} \right)$$

where

$$\begin{cases} P_{in}(\theta) & = \mathcal{T} \cap \hat{\mathcal{D}}_\theta \\ P_{out}(\theta) & = \hat{\mathcal{D}}_\theta \setminus \mathcal{T} \end{cases}$$

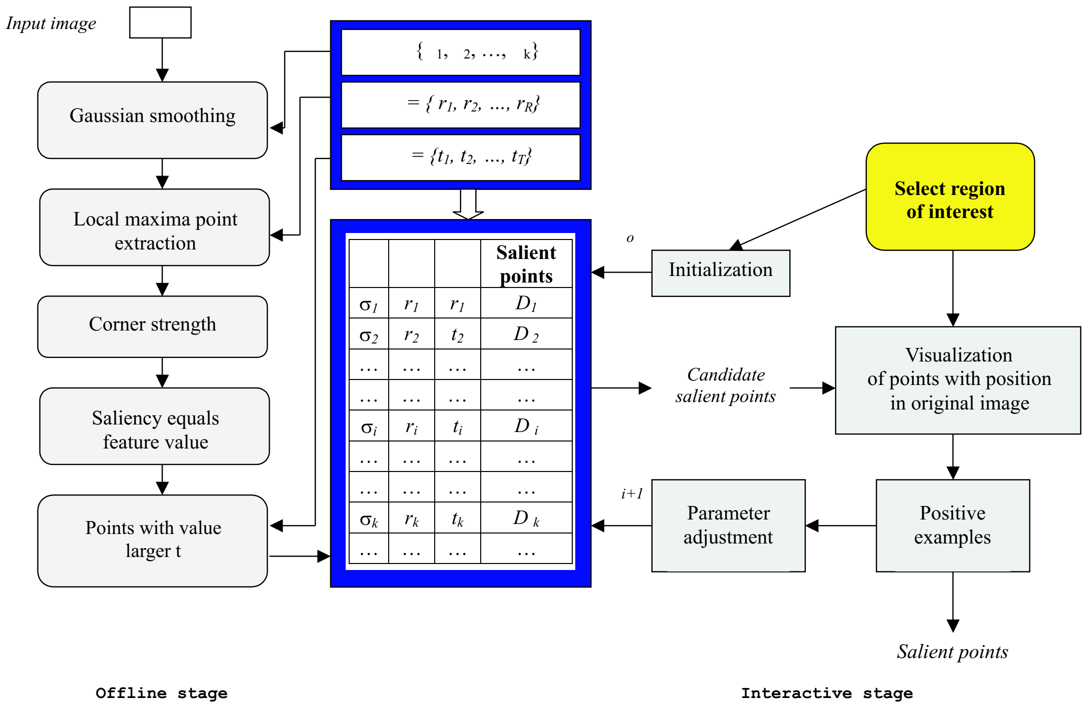


Figure 5.4: Instantiation of the framework for the Harris point detector

to assure that points are gradually added to $\mathcal{T}$, we use the constraint:

$$\mathcal{C} = \left\{ (1-\epsilon)||\hat{\mathcal{D}}_{\theta^i}|| < ||\hat{\mathcal{D}}_{\theta^{i+1}}|| < (1+\epsilon)||\hat{\mathcal{D}}_{\theta^i}|| \right\}.$$

with $\epsilon = 0.2$. This means that within the constraint $\mathcal{C}$, we find a parameter set θ that returns the maximum value of $\frac{P_{in}}{P_{in}+P_{out}}$. This is illustrated in figure 5.4.

We now show some results of using the method thus defined. In the first example, we consider the image depicted in figure 5.5a. With the default set of parameters, the Harris point detector returns 463 points mostly lying along the borders. In case the user wants to have points inside the petal area, the default parameter set fails to return the set of salient points inside that region. Within the region of interest, the number of positive points is 128/463. Based on the user feedback, the system finds the new values for parameters which return the total number of 475 points, with more points in the region of interest namely 162/475.

The second example is similar. The default parameter set returns 1904 points with 39 positive points. The system updates the parameters and 1568 points are found. The final set of salient points (107/1568) is still able to cover the main corners, borders, while better to representing the user selected region.

5.4.2 Interacting with salient lines

In this section, we show an instantiation of the framework for salient line extraction.

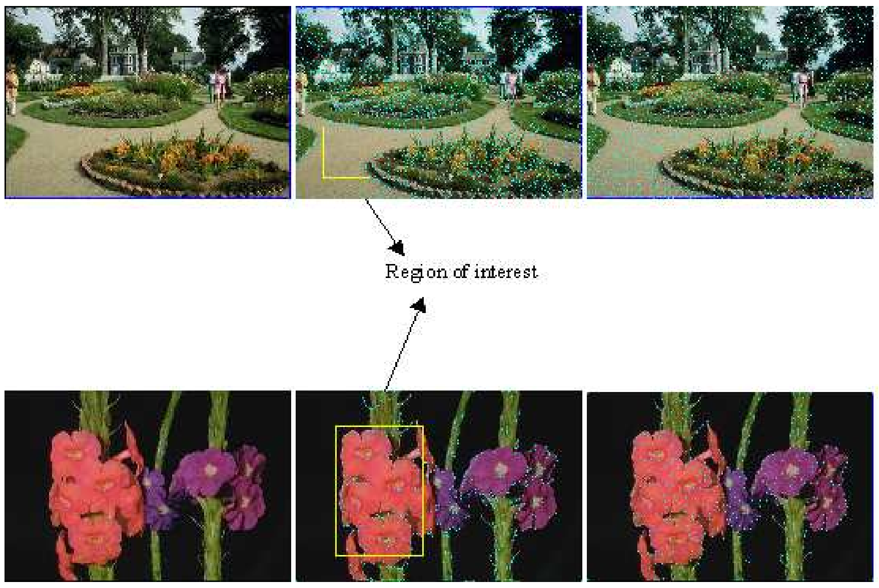


Figure 5.5: Examples of experimental results. The left images are query images. The middle ones are images with default automatic detected salient points superimposed, the rectangles show the regions of interest with positive examples given by the user. The right ones are the returned images after user-feedback.

In this example, we employ the Canny edge detector [12]. In the image processing step, the input image is smoothed using a Gaussian mask. Gradient magnitude and edge direction are then computed at each pixel to extract edges. In the final step, Canny uses a threshold over the average strength of candidate edges, which will yield a set of salient edges.

The output is given to the user to provide feedback. The final image should contain as much as possible the details which are salient from the user's perspective. The instantiation is shown in figure 5.6.

From the above we collect the set of parameters for this method as $\theta = \theta(\sigma, t)$, where σ is the standard deviation of the Gaussian, and t is the threshold to select whether there is an edge point or not. Default θ^0 is set to $(1.0, 3.0)$ as proposed in [33]. The *offline-stage* stores an archive results for parameters $\sigma = \{1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4, 4.5, 5\}$, and $t = \{1.0, 2.0, 2.5, 3, 3.5, 4, 5, 6, 6.5, 7\}$.

In this experiment, we let the user give feedback by selecting the line which they want to be removed. Here we make the assumption that these edges occur because the level of detail is too high, hence too many edges of limited length are present. Therefore, the length l_0 of the selected line is computed. The new θ^{i+1} will be searched through the *offline data* such that the new set of salient lines contains as little as possible lines with length smaller than l_0. Thus,

$$\theta^{i+1} = \underset{\theta}{argmax} \left(\frac{||\mathcal{D}_p(\theta)||}{||\mathcal{D}_p(\theta)|| + ||\mathcal{D}_n(\theta)||} \right)$$

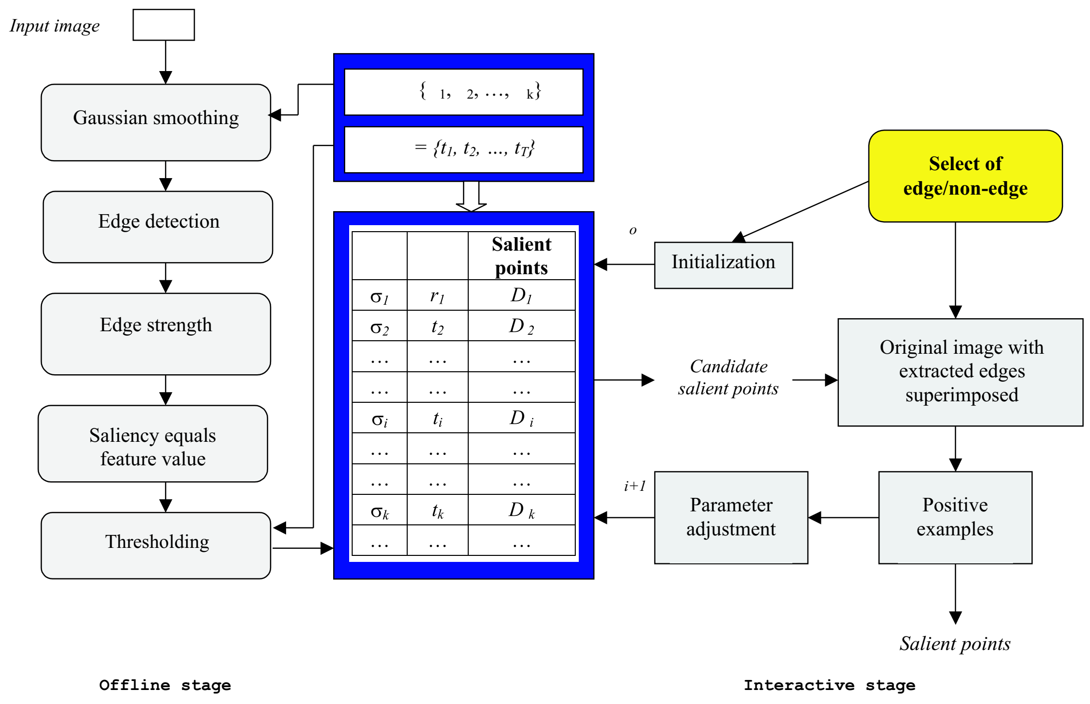


Figure 5.6: Instantiation of the framework for Canny edge detection

where

$$\begin{cases} \mathcal{D}_p(\theta) & = \{d : ||d|| > l_0\} \\ \mathcal{D}_n(\theta) & = \{d : ||d|| \leq l_0\}. \end{cases}$$

The constraint is:

$$\mathcal{C} = \left\{ ||(1-\epsilon)||\hat{\mathcal{D}}_{\theta^i}|| \leq ||\hat{\mathcal{D}}_{\theta^{i+1}}|| \leq (1+\epsilon)||\hat{\mathcal{D}}_{\theta^i}|| \right\},$$

where we use $\epsilon = 0.1$.

To start the system, we begin with the smallest value of σ and t to obtain all possible edges. Each time the unwanted lines are removed. This is illustrated in figure 5.7. We observe that the image after user's feedback has less petty lines compare to the result using default parameters. Of course, this experiment can also be done in the case where the user looks for more detailed edges. The user then gives feedback by pointing at regions in the image where edges should be found, but are not present. This is quite similar to the case of finding points of interest as presented in section 5.4.1.

5.4.3 Interacting with salient regions

In this example, we work with the region segmentation method from [44]. In this reference, starting with Gaussian smoothing, the image is filtered using a set of Gabor filters. All pixels are represented in the feature space given by the Gabor results and

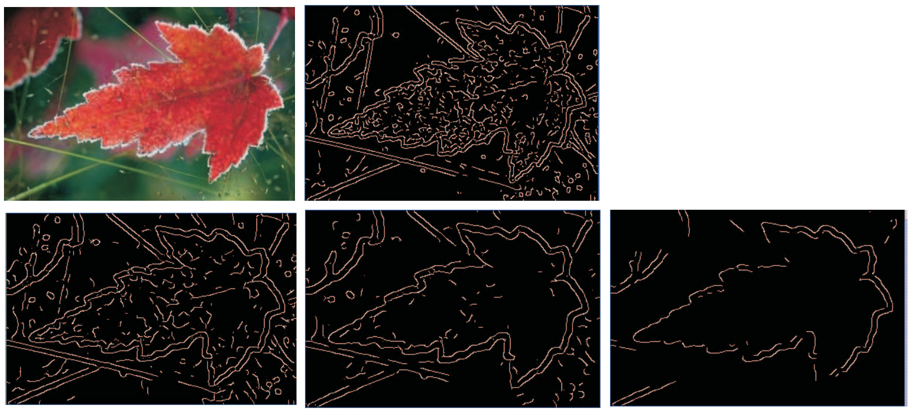


Figure 5.7: Examples of experimental results with interactive salient line detection. The first row contains the original image and an edge image with default parameters. Some results after user interaction are shown in the second row.

the color. A k-means clustering method is then used to group pixels into regions. They collect the same color and texture features from the previous step to find the similarities between extracted regions. Regions with similarities larger than a given threshold are then merged to give the final salient regions.

Different parameters are used in the process, we select two of them as tunable parameters: the scale σ used in the Gabor filters computation, and the similarity threshold t. Thus, we collect a set of parameters for this method as $\theta = \theta(\sigma, t)$. The remaining parameters for the Gabor filter namely the specific frequencies and orientations are taken following the guideline in [44]. The default values are $\theta^0 = (\{4, 3.5, 2.95, 2.35, 1.75\}, 7.5)$. In the experiment, we take $t \in [4.0, 8.0]$ with step 0.5 and $\sigma \in [1.0, 6.0]$ with step 0.05 and apply them to the method.

The resulting salient regions are visualized. The user gives feedback by asking the system to split or merge regions. When the user asks for a global split or merge, the system searches for a $\hat{\theta}$ which returns $\hat{\mathcal{D}}'_\theta$ with a number of regions larger or smaller respectively than the previous result $\hat{\mathcal{D}}$, i.e.,

$$\theta^{i+1} = \underset{\theta}{argmax} \left(||\hat{\mathcal{D}}_\theta|| \right),$$

$$\text{with } \begin{cases} \mathcal{C} = \left\{ ||\hat{\mathcal{D}}_{\theta^i}|| < ||\hat{\mathcal{D}}_{\theta^{i+1}}|| < (1+\epsilon)||\hat{\mathcal{D}}_{\theta^i}||,\ \epsilon \in \mathbb{R}^+ \right\} \\ \mathcal{C} = \left\{ ||\hat{\mathcal{D}}_{\theta^i}|| > ||\hat{\mathcal{D}}_{\theta^{i+1}}|| > (1-\epsilon)||\hat{\mathcal{D}}_{\theta^i}||,\ \epsilon \in \mathbb{R}^+ \right\} \end{cases}$$

The new θ^{t+1} returned with maximum number of regions or minimum, in case of splitting or merging respectively, within the constraint $\mathcal{C}$.

In our experiment, we simply select ϵ such that the number of salient regions will increase/decrease 20% in the new set. If $\epsilon * ||\hat{\mathcal{D}}_{\theta^i}|| < 1$ then the constraint will be ± 1

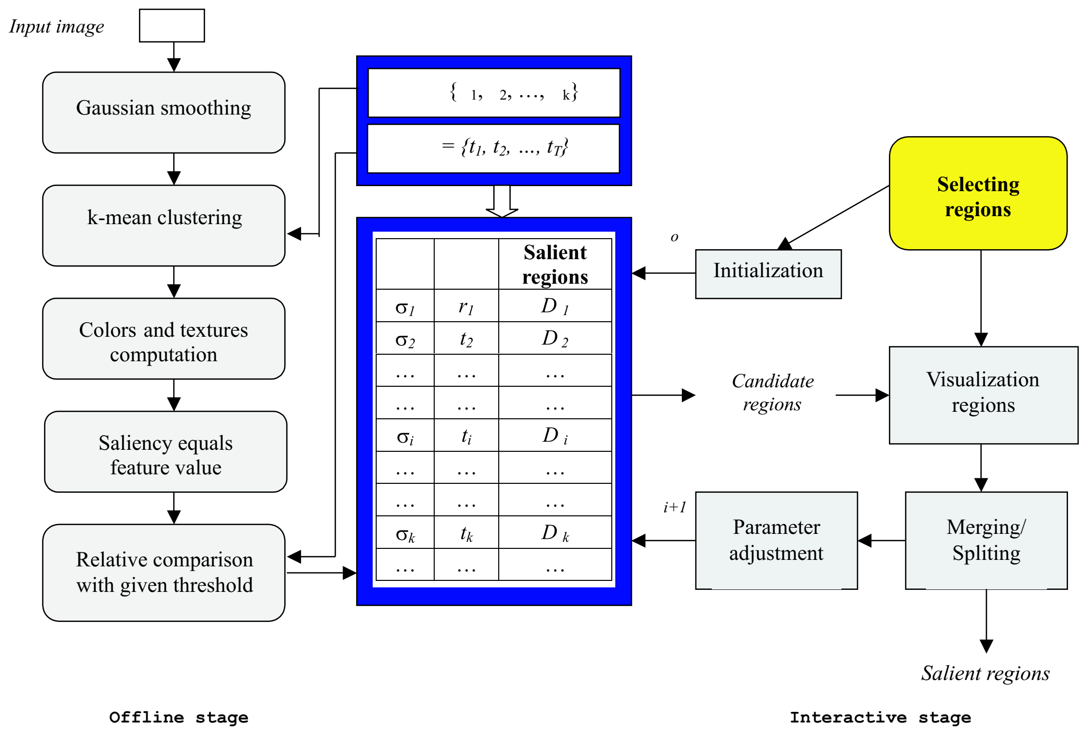


Figure 5.8: An instantiation of the framework using Hoang's region segmentation

region, respectively. Therefore,

$$\epsilon = \max(0.2, \frac{1}{||\hat{\mathcal{D}}_{\theta^i}||}).$$

In case of a local split the system tries to keep the outer boundary of the selected region, and introduces inner boundaries. With local merging of regions, the system keeps the outer boundary and removes inner ones. This leads to general criteria for selecting a set of parameter θ which will now be explained.

Assume the current set of salient regions is $\hat{\mathcal{D}} = \{\hat{d}_1, \cdots, \hat{d}_k\}$, the user decides to merge two regions $\hat{d}_i$ and $\hat{d}_j$. The system returns $\hat{\mathcal{D}}'_\theta = \{\hat{d}'_1, \cdots, \hat{d}'_t\}$ such that

$$\exists m : \hat{d}'_m \simeq \{\hat{d}_i \cup \hat{d}_j\},$$

where $\simeq$ denotes that the new region $\hat{d}'_m$ approximates the combination of the two regions $\hat{d}_i$ and $\hat{d}_j$ with an error $\tau \in \mathbb{R}^+$.

Therefore, let $\mathcal{A}(\hat{d})$ denotes the area of $\hat{d}$ then the evaluation function will be:

$$\mathrm{E} = \left(\left(\mathcal{A}(\hat{d}'_m) \cup \mathcal{A}(\{\hat{d}_i \cup \hat{d}_j\})\right) \setminus \left(\mathcal{A}(\hat{d}'_m) \cap \mathcal{A}(\{\hat{d}_i \cup \hat{d}_j\})\right)\right),$$

with constraint $\mathcal{C} = \left\{||\hat{\mathcal{D}}_{\theta^{i+1}}|| = ||\hat{\mathcal{D}}_{\theta^i}|| - 1\right\}$.

In the similar case for local splitting, we have $\hat{d}_m \simeq \{\hat{d}'_i \cup \hat{d}'_j\}$. Those formulas can easily be generalized for n regions. Repeating this process, the user will be able to find the meaningful segmentations. Figures 5.9 and 5.10 show some results of interactive segmentation of images.

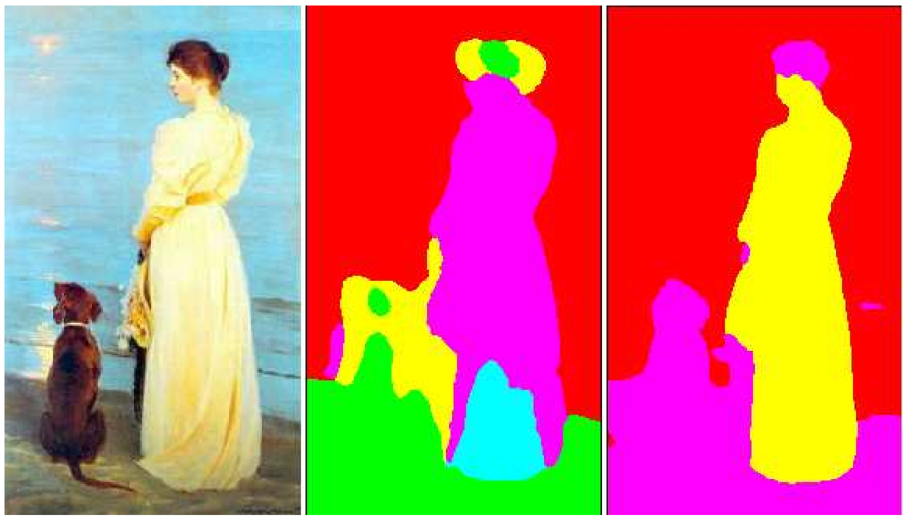


Figure 5.9: Example of an interactive segmentation result. The left image is the query image. The middle one is the segmented image with default parameters. The right image is the result after user-based splitting/merging process. The default set of candidate salient regions are shown to the user, a global merging command is given. The system then returns the segmented image with a number of regions lower than before.

The results depend on what the user is looking for, since the definition of “object” is at the semantic level. In the first example, the salient region represents the “woman”. In the second example, the user looks for the region of the sun.

5.4.4 Content based image retrieval with query refinement

In this section, we present two applications of the proposed framework for content based image retrieval using salient points and salient regions. We leave out the application with salient lines as it is in the middle of points and regions.

Some available examples of using salient points in CBIR are presented in [100], [112], for lines in [45], and for regions in [14]. For CBIR with salient details an appropriate similarity function has to be defined. For instance, the Hausdorff or Chamfer distance would be an appropriate choice to measure the similarity between images based on comparing the locations of salient points. With sets of lines, the similarity can be based on the comparison between two sets of lines on either the curvatures of lines, or their shapes, or we can view lines as the border between two regions and compare their features. With regions, features can be shapes, colors, textures.

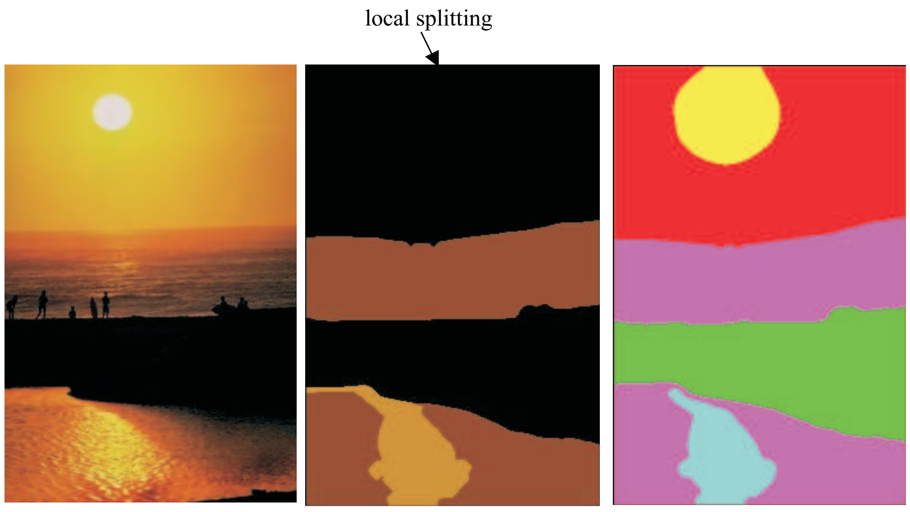


Figure 5.10: Example of an interactive segmentation result. In this case, a local splitting command is given to the default set. A local search is given within the selected region to find more regions in it, therefore the parameter θ is locally defined.

In the previous sections, we have used the framework to find the optimal parameters for one image. Now let us consider how to apply this for searching. If we perform t steps, we have (from e.q. 5.2)

$$\{I_Q, F_Q^0, S_Q, Z_Q\} \overset{RF_i}{\rightarrow} \{I_Q, F_Q^t, S_Q, Z_Q\}$$

Thus at every step, the whole query space is updated as the current parameters are applied to the whole dataset. Thus after initialization we iteratively search for a good query. The number of iterations during the user interaction depends on the value of ϵ in the constraint function $\mathcal{C}$. If ϵ is set to a small value, the system will take several steps to reach the final result.

5.4.4.1. Examples

For the experiment, a dataset of 1100 Corel images is used, which consists of different scenes and objects. The initial images can be selected by the user from a set of internal or external examples. For simplicity, we work on a single image at a time. Working with multiple examples can be done by combining the user's feedback on each image.

The system is based on the query details defined by default parameters and tuned parameters after relevance feedback as described in section 5.4.3 and 5.4.1.

In case of using salient points, for each image in the database, a set of salient points is extracted. For computing the feature, we use [109] to get color moments at each color channel. In our experiment, we use HSI color space. First, the color histogram HSI is computed at each salient point in a 3x3 neighborhood. Those values are sum up and normalized by the number of salient points in the images. Next, three color moments: mean, variance, and skewness are computed for each color channel. Hence, each image is then represent by a vector of 9 values. $\mathcal{L}_1$ is used as a similarity function between feature vectors.

For query region specification and refinement we follow the method of section 5.4.3. However, in this case, as features of the region, we take Hue and Saturation as they are simple and effective for the Corel dataset. With Hue and Saturation features, the function S_Q is based on histogram intersection [110]. The similarity value is defined by the best matching region of an image to the query one. Finally, returned images are ranked based on similarities.

As described in the previous section, we select $\epsilon = 0.2$. Then for each query, it takes 4 to 5 iterations to reach the user's desired result, i.e., $t = 4$ or 5.

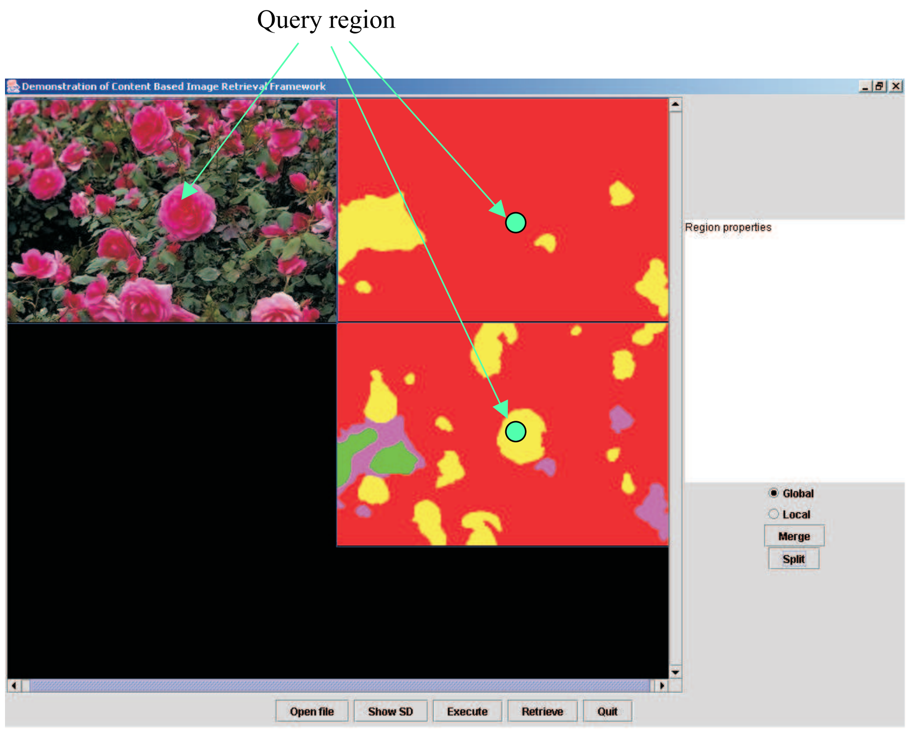


Figure 5.11: Example of a query region definition. The upper-left corner is the query image, the upper-right corner is the image with query region using default parameters. The last one is the image with redefined query region. The circle denotes the clicked point by the user.

An example of how the system works is given in figure 5.11. In this particular example, it follows that with default parameters, the query is almost the whole image, hence contains both red (flowers) and green (leaves). The system then returns images which contain regions with similar color distributions. The user shows that he/she is only interested in the red region, the results after adaptation therefore show all images with red regions also if they are on a different background. In figure 5.12, we show the results with the corresponding retrieved regions similar to the query region.

It is observed that the result with updated parameters (figure 5.12a) is closer to the desired results than the default (figure 5.12b) since retrieved regions are indeed similar red regions. In figure 5.12c, the updated parameters are only applied to the query image but not to the whole dataset. Since the new parameters define a smaller scale, the results show that images returned have red regions extracted at high scale but miss ones that only exist at smaller scale. When applying the updated parameters to the whole dataset, the regions at higher scale are subdivided into smaller regions, hence those contain smaller red regions, which are now retrieved by the system.

5.4.4.2. Experiments

As stated before, finding the user's desired results is a subjective problem, hence it is not easy to evaluate or compare our system to existing ones. However, we believe that using updated parameters, the user will be provided the choice to select the right details, which are missing using the default ones, for a query search. To see whether salient details performs better than global features, and to see whether it pays off to optimize the parameters specifically for each query, we compute the following comparisons:

(i) default query detail compared to details with default parameters in offline data.

(ii) updated query detail compared to details with default parameters in offline data.

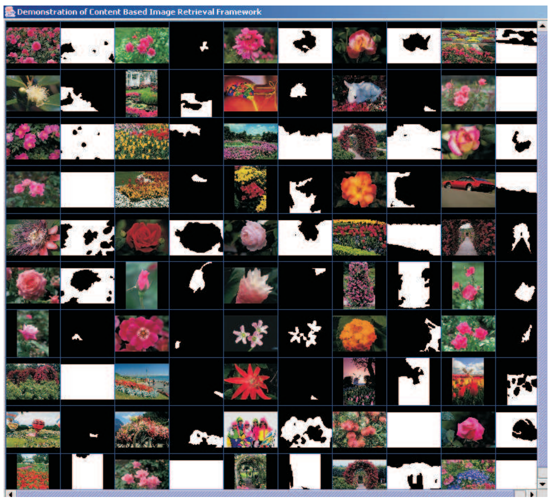
(iii) updated query detail compared to details with updated parameters in offline data.

(iv) using global features (i.e., features are computed for the whole image).

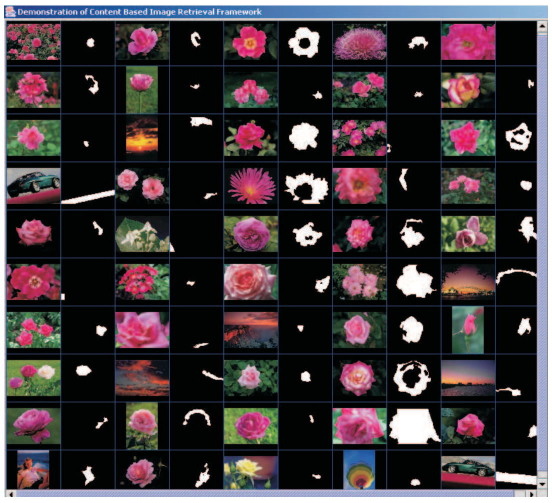
For that purpose, we built a model for the search task, which will be described as follows.

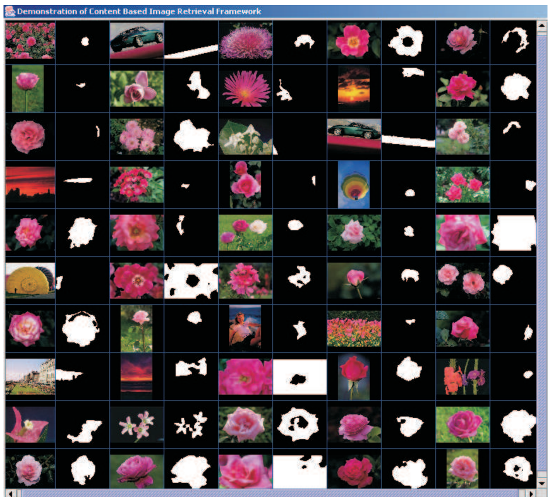
We used the pre-defined categories from Corel. The 10 categories are cars, surfing, sunset, flowers, roses, seasons, flower beds, balloons, summer, winter. The size of each category is 100 images, except the category "Roses" containing 200 images. Each image in the dataset is subsequently used as the query. Salient details are extracted using either default or updated parameters. With salient points, the feature vector of the query image is compared to all feature vectors in the database at the same parameter setting. In case of region based search, the comparison is more complicated. Each region in the image is selected as the query region. The search process looks into the offline data, and compares the query with all extracted regions at the same parameter setting. The similarity is the maximum value of all regions in one image compared to the query region. Hence, for each query region, we get a ranked list of images based on the similarity values (figure 5.13). We then compute the recall and precision. The final rank list of a query image is the one with highest values of recall and precision. Number of retrieved images in the experiments are 10, 20, 30, 50, 100, 150, 200 images. Figure 5.16 and 5.17 and 5.18 show average precision and recall of each category using salient regions and points, respectively.

From the results, we show that using salient details generally perform better than global features. This especially holds for region based search when finding red flowers, cars, and sun. Since the searched objects are specific, the user easily figures out the



(a)





(b) (c)

Figure 5.12: (a) Result images using region-based search where the query region is extracted using default parameters, at the right side of each picture the region found is indicated. (b) The same but now using the updated parameters. (c) Result images using region-based search where the query region is extracted using updated parameters but without applying those updated parameters to the whole dataset.

salient regions for the search. In case of general search, for example searching for seasons, it is hard to point out which regions are most representative for the term "Season". Besides, we show that finding the appropriate query details (ii) leads to an improvement of the search. However, updating the whole dataset with the new

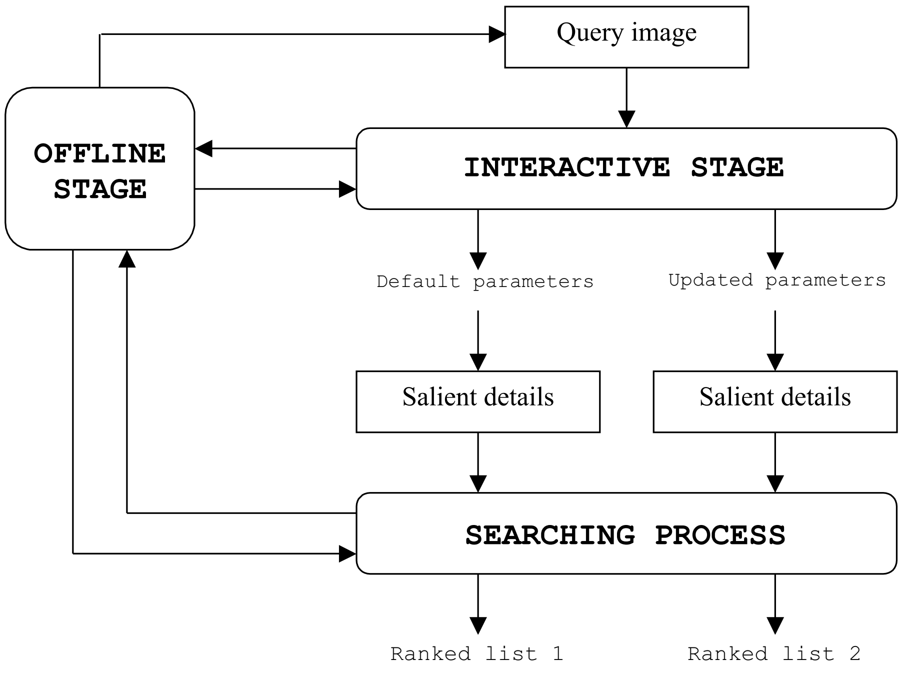
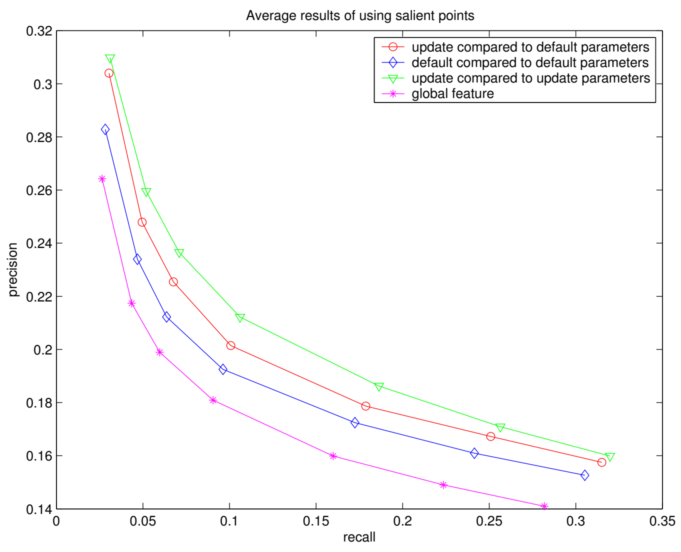


Figure 5.13: Experimental setup.

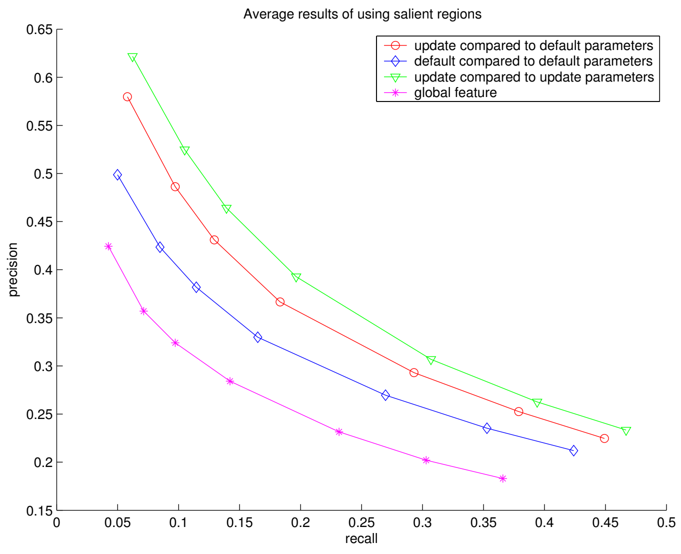
updated parameters (iii) will give the best results. From the experiments, we also prove that for each category there is an optimal parameter set for the search different from the default one.

5.5 Conclusion

In this chapter, we have proposed a user based framework for interacting with salient details. Using this framework we have identified that existing salient detail detections can be classified into the following five steps: image processing, detail detection, feature computation, saliency computation, and selection based on significance. Tunable parameters at each step are then found, from there we present efficient methods for updating those parameters via user relevance feedback. The instantiations of the framework show that adapting saliency of details can get closer to saliency in user's perspective. Moreover, based on a set of 1100 images from the Corel collection, the potential of applying the frame work to the image retrieval system is illustrated. Experimental results first prove our theory that using salient details perform better than global features, especially using salient regions. Secondly, it is demonstrated that using parameter optimization to update the query space remarkable improvements in retrieval performance can be obtained. Extending the proposed framework so that it can deal with larger datasets (e.g., more than 100000 images) is an interesting topic for future work. For such a large data set, a number of issues such as indexing for interaction and search as well as storing data must be studied thoroughly.



(a) Using salient points

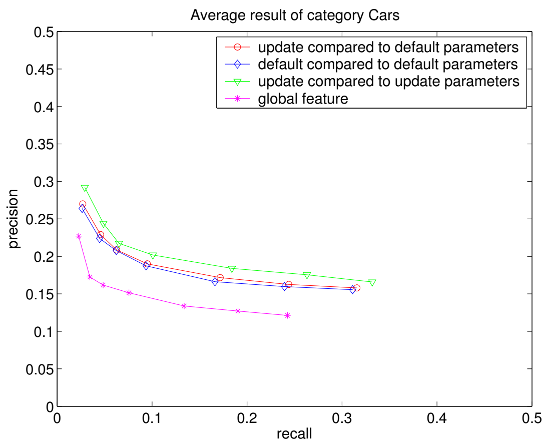


(b) Using salient regions

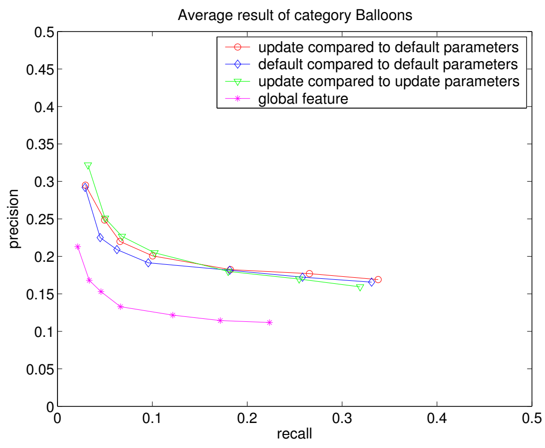
Figure 5.14: Average recall and precision comparison of 1100 Corel images.

Acknowledgment

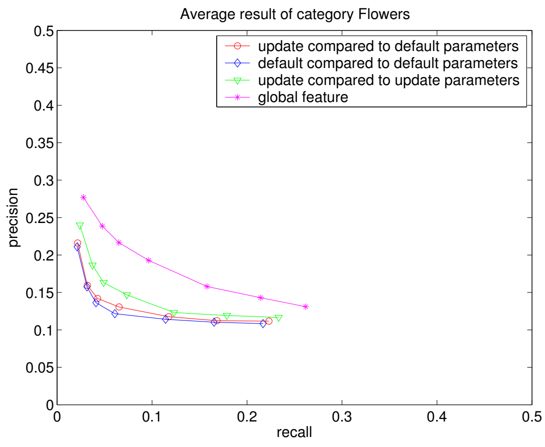
The work is within the ImIk project (Interactive disclosure of Multimedia Information and Knowledge) sponsored by IOP MMI (Man-Machine Interaction). The authors would like to thank anonymous reviewers for their valuable suggestions and comments.



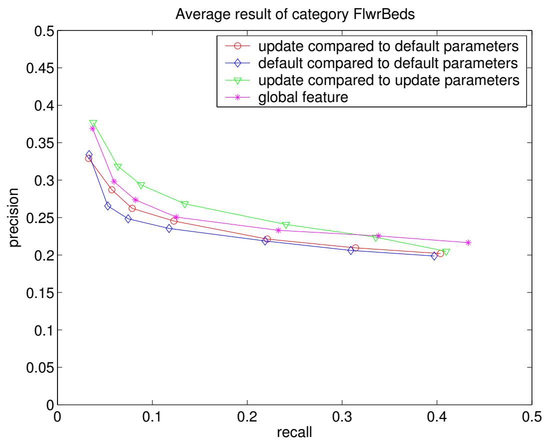
(a) Cars



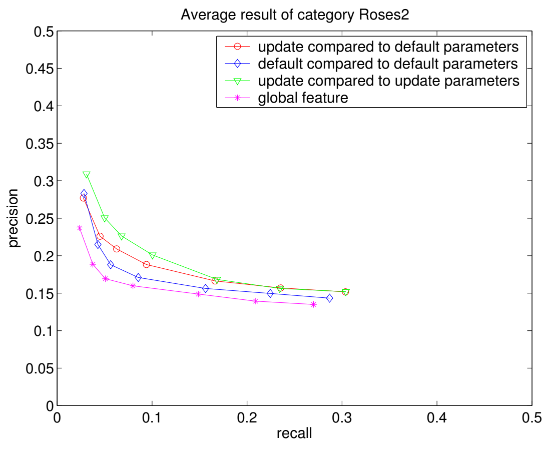
(b) Balloons



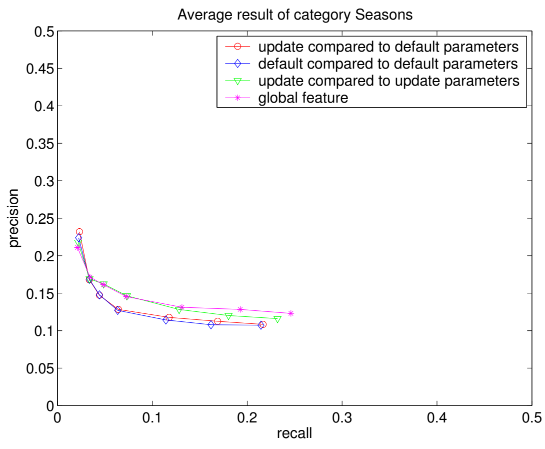
(c) Flowers



(d) Flowerbeds

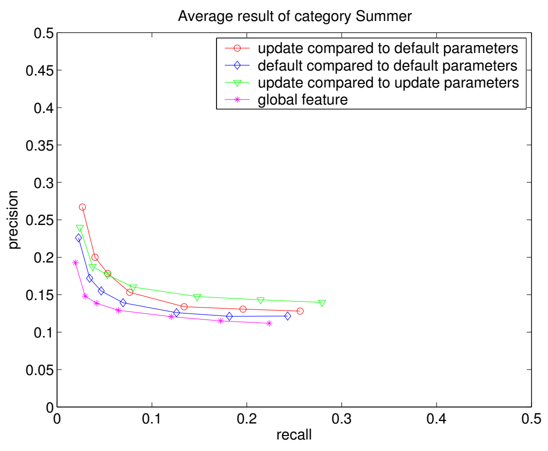


(e) Roses

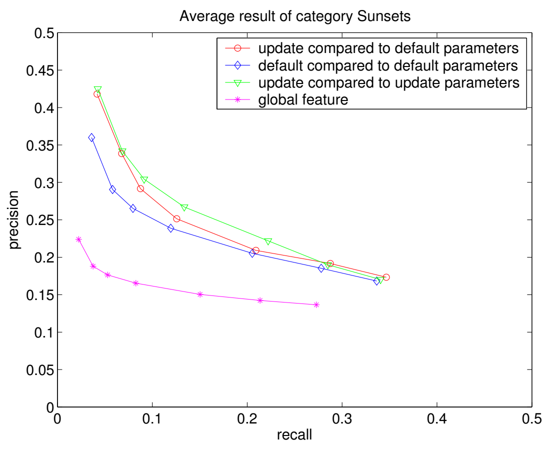


(f) Seasons

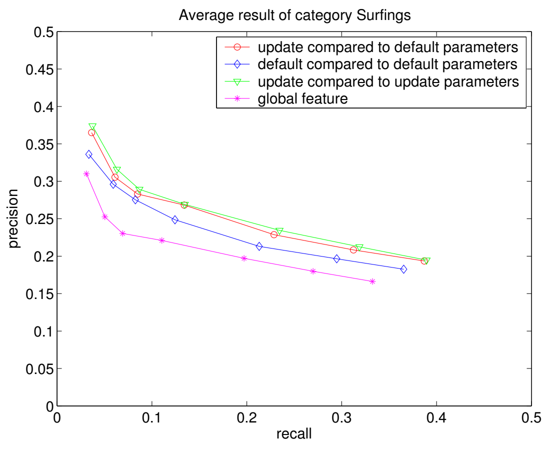
Figure 5.15: Recall and precision comparison using salient points.



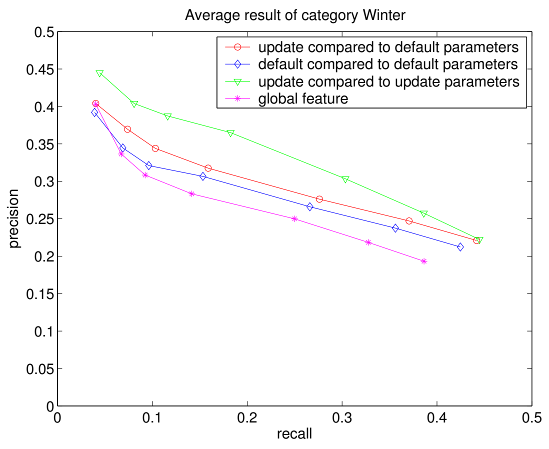
(a) Summer



(b) Sunset

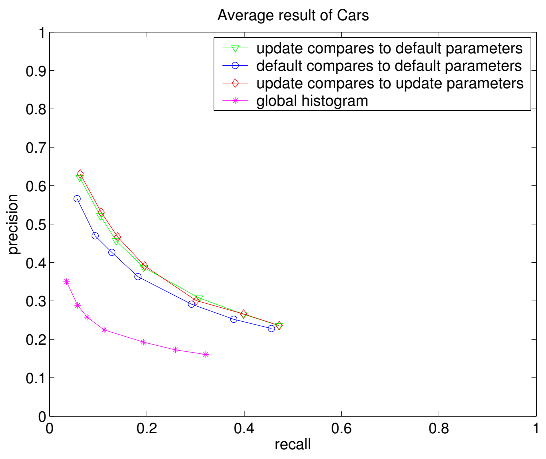


(c) Surfing

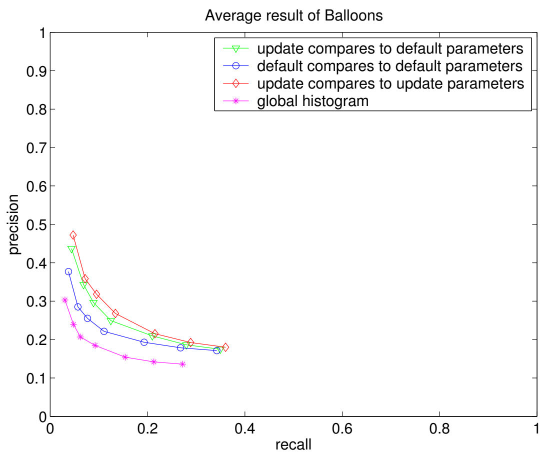


(d) Winter

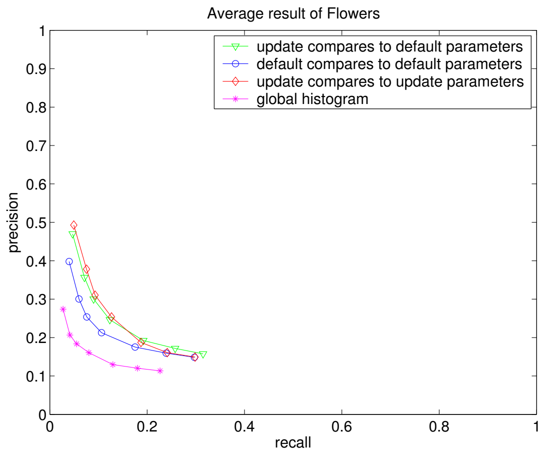
Figure 5.16: Recall and precision comparison using salient points (continue).



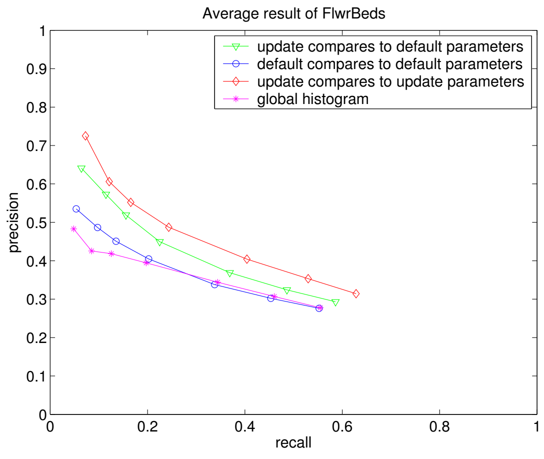
(a) Cars



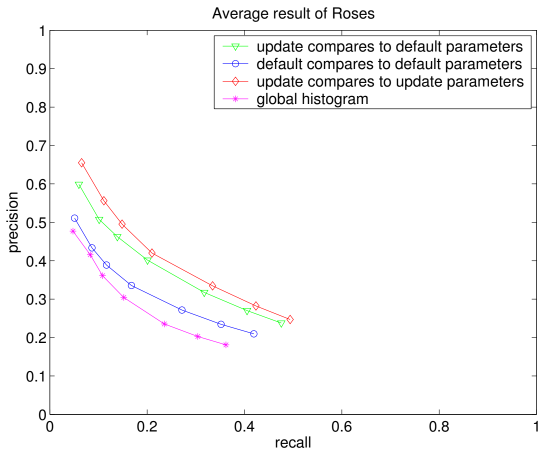
(b) Balloons



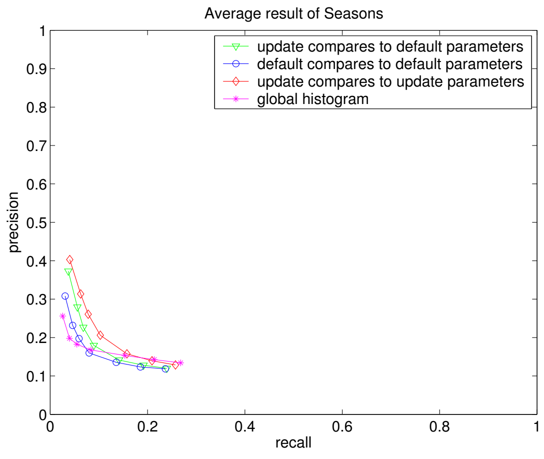
(c) Flowers



(d) Flowerbeds

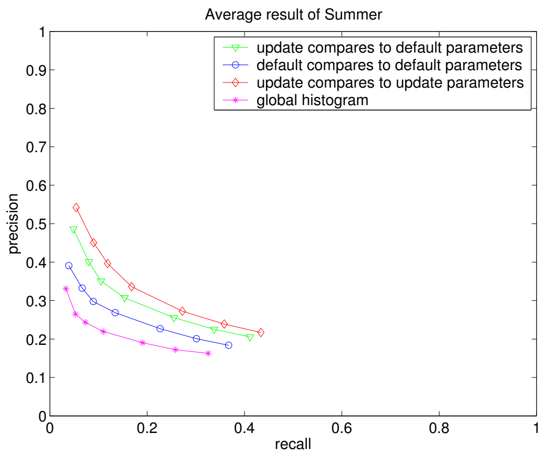


(e) Roses

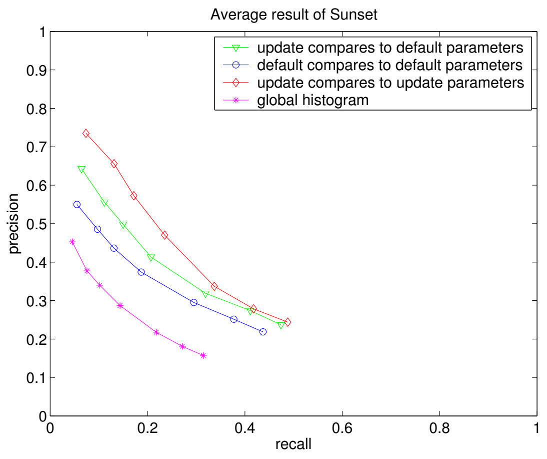


(f) Seasons

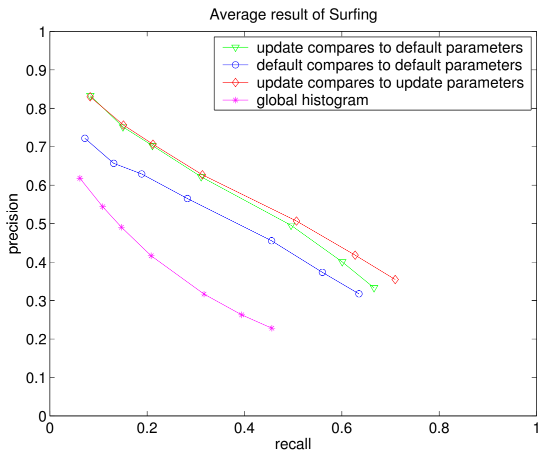
Figure 5.17: Recall and precision comparison using salient regions.



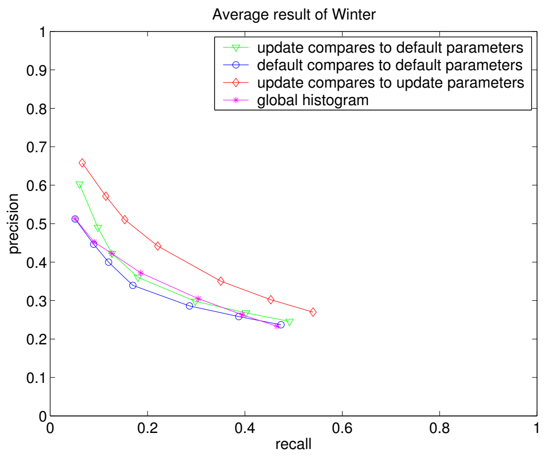
(a) Summer



(b) Sunset



(c) Surfing



(d) Winter

Figure 5.18: Recall and precision comparison using salient regions (continue).

Chapter 6

Conclusion

6.1 Summary of contributions

Content-based image retrieval (CBIR) has been under investigation for a long time. Many systems have been built to meet different application demands. In all systems, there is a strive towards semantic retrieval. As the exchange of semantic information requires concordance between any two parties on the context, user interaction is an essential component of any semantic CBIR-system. And, the interaction will be not easy to achieve as still there is the *semantic gap* between the user's expectation of what the system can do and the actual retrieval capabilities of the system. In this thesis, we have developed and evaluated several components for interactive CBIR to reduce the semantic gap. The research questions raised in the introduction have been analyzed and answered in four chapters.

To answer the question **Q1**: *What is the optimal way of visualizing images in obtaining useful feedback?*, we investigated various visualization techniques in **chapter 2**. We first analyzed the role of visualization in interactive CBIR for searching and browsing through large collections of images. Despite of their essential role, we observed that existing visualization systems do not sufficiently cope with the problems occurring when dealing with large visual collections. Therefore, we have made these problems explicit. From there, we established three general requirements that visualization should obey: overview, visibility, and structure preservation. Solutions satisfying each requirement were proposed as well as functions balancing the importance of the various requirements. Based on these requirements, we presented an optimal visualization scheme for supporting users in their interaction with large image collections. Experimental results on a collection of 10,000 Corel images, using simulated user actions, show that the proposed scheme significantly improves performance for the task of category search when compared to the 2D-grid based visualizations commonly used in CBIR.

Having defined an optimal visualization system, we considered the integration of visualization and user feedback to improve interaction further. In **chapter 3**, we first observed that in current research the visualization and feedback are often being considered as two independent elements. This means that on the one hand, some of the research concentrates on learning methods for effective use of relevance feedback; while on the other hand, another focus of research is on effective methods for conveying information to the user. We proposed an efficient interactive search system in which the two steps are truly integrated. In such an integrated system, there are many degrees of freedom like the similarity function, the number of images to display, the image size, different visualization modes, and possible feedback modes. It is unfeasible to find by user studies optimal values for all of these parameters. We therefore developed search scenarios in which tasks and user actions were simulated. This gave an answer to the question **Q4**: *How to objectively evaluate the performance of an integrated interactive CBIR-system?*. Given the scenario and user actions, the proposed scheme was optimized based on objective constraints and evaluation criteria. In such a manner, the degrees of freedom are being reduced and the remaining degrees can be evaluated in user studies. We performed extensive experimentation on interactive category search with different image collections. The results show that the use of advanced visualization and active learning indeed pays off on all of the datasets. The components are included in the MediaMill system, which received the *Best Technical Demo Award* at the ACM Multimedia conference 2005 [106].

The first set of chapters consider the problem at a system level. In the next two chapters, we concentrated on specific components of the interactive framework (see figure 1.1) namely the features $\mathcal{F}$ and similarity function $\mathcal{S}$. In **chapter 4**, we considered the learning of similarity between images (question **Q3**: *How to iteratively learn similarity on a higher level than the primitive level*). We introduced a new approach to learn dissimilarity for interactive search in CBIR. We used relevance feedback to adjust the dissimilarity without going back to the feature space, and this is different from what other techniques do. This has the great advantage that we can manipulate the dissimilarity directly. To create a dissimilarity space, a set of images was first selected as prototypes. Similarities between each image in the collection and each of those prototypes formed the new representation of an image [81]. After user feedback, we employed one-class SVM to adjust this space such that relevant images stayed close to one another while irrelevant ones were pushed away following the technique in [39]. Results on a Corel dataset of 10000 images and a TRECVID collection of 43,907 key frames show that our proposed approach is not only intuitive, it also significantly improves the retrieval performance.

Like the definition of similarity, the saliency of image features also is user and context dependent. So, to answer the question **Q2:** *How to define the saliency of features, namely points, regions, or lines, such that it is context and user interpretation dependent?*, in **chapter 5**, we considered user interaction with salient details in the image. Interaction with features in literature has mostly been focused on changing global image features. The interactive salient detail definition goes further than summarizing the image into a set of salient details. We dynamically updated the user- and context-dependent definition of saliency based on relevance feedback. To that end, we proposed an interaction framework for salient details from the perspective of the user. A number of instantiations of the framework were presented. Experimental results using salient points and regions proved the effectiveness of adapting the saliency from user feedback in the retrieval process.

In summary, we have introduced a new generation of advanced image search mechanisms. An optimal search system integrating both visualization and feedback steps has been proposed. The techniques to iteratively update dissimilarity and salient features in this thesis bring these abstract notions much closer to the user. With the proposed techniques and systems, the results of interactive CBIR improve in performance and intuition, building a bridge over the semantic gap.

6.2 Future directions

We consider similarity-based visualization and the associated feedback mechanisms as the most promising line of research to continue on. We discuss here how to generalize the current system to support other tasks, how to scale up to much larger collections of images, and how to arrange the evaluation.

Generalization to other tasks In this thesis we mainly concentrated on visual features, but the visualization system can in principle be applied to many other types of features as long as there is a well-defined similarity function. As long as visual similarity is used, relations between images may be evident by simply looking at them. This may not be all that trivial for other situations like the use of text-based relations. What are needed are visualization mechanisms which visualize these relations for the users to grasp their meaning. The proposed interaction scheme can be adapted to different scenarios such as target search, association search, and annotation. For instance, in target search the user looks for a specific image. In our system, hierarchical browsing of images where similar images are grouped will guide the user to the right search direction. For this task the system should be more conservative in deleting images from the active set as it might well contain the target image. For the annotation task, images have to be assigned a semantic meaning by the user. As similar images are located close to each other, the user's effort to annotate a large image collection will be reduced significantly. To adapt to this task the system should allow attaching multiple semantic labels to an image whereas in category search only one category is considered. In general, for new tasks, the adaptation is achieved by careful redefinition of the task and the user actions before optimizing the system.

Scalability in supporting the user One of the major challenges in interactive retrieval is the ever increasing size of image collections. The collections used in this thesis are big, but still small compared to image collections users might encounter in practice. At some point the combination of hierarchical search followed by similarity-based visualization will require too much interaction steps. The limitation of the display in combination with the visibility requirement will become a bottleneck. To optimize the use of the visualization space, 3D and dynamic similarity based visualization are to be considered. When following this direction, the balance between the three proposed requirements should be adapted as well. 3D-visualization can be treated as a layered set of 2D-visualizations, where the techniques in this thesis are applied to each individual layer. The question then is how the order of the layers relates to structure. When genuine 3D visualization is used with the size of images changes with depth, structure should be preserved in 3D, while visibility has to be dependent on depth. In dynamic visualization, visibility puts limitations on the speed with which images can be displayed, and this should be an additional constraint in the visibility term. Advanced visualization is even more needed for very large collections than it is for the datasets considered in thesis.

Scalability from a system perspective . To achieve interactive response times, our interactive system mostly deals with representations of the data which are computed offline. Offline computation makes the system less flexible. When images are added to the collection, re-computation of the offline data is required. For clustering the competitive learning method described in section 4.3.1 (chapter 4) scales well. But, when moving from offline to online computation different mapping techniques are needed. In chapter 2, we showed that the local linear embedding LLE (see table 2.1) gives a good approximation to the best performing mapping ISOSNE (isometric stochastic neighbor embedding), but requires an order of magnitude less computation time. Therefore, LLE is a good candidate for achieving online mapping of large collections. An alternative is a more recently published version of the ISOMAP algorithm [111]. This method provides an approximation of the optimal mapping of datasets by selecting a set of elements called landmark points. It means that for computing the mapping only a subset of the dataset needs to be considered. For small sets the landmarks will not be able to preserve the structure of the whole collection in the mapping. For large image collections, the problem of selecting appropriate landmark elements which preserve structure and in the mean time are fast enough for interactive system purpose is an open issue. Following these lines of fast, but sufficiently accurate, methods will allow scalability in the number of images the system can handle in an interactive fashion.

Objective evaluation Unlike fully automatic systems, an interactive system is designed to work with users. For evaluating the system user studies are essential. As we have discussed in the thesis, user studies should be preceded by evaluations with simulated users as it is easier and more flexible. In our experiments, the simulated actions are the ones most commonly used during interaction. The more complicated the system, the more difficult it becomes to simulate the user actions. To establish more elaborate objective evaluation models, a close relation between interactive search and the human-computer interaction field is to be established. Currently, we see a limited research effort in that community on interactive CBIR systems. And, in the CBIR-community limited attention is paid to usability aspects. What is needed is a common framework in which both technical and usability aspects are well defined. Then both can be jointly optimized using simulations similar to the ones we have proposed.

Extension on the methodology The use of a dissimilarity space for searching images in CBIR is a promising research direction. We have pointed out the advantages of this approach: the space is simple to create; it is intuitive for interaction and thus it is easier for the user to understand. Our method for adjustment of the dissimilarity representation already improves search, but it is still basic as we give all assumed dissimilar images the same adjustment direction. This is not an optimal solution as each image has relations with its neighbors. Moving an image affects those neighbors also. Hence, the movement of images in the collection should be made dependent on one another. Secondly, the requirement of having a reasonable number of positive prototypes may not be met for collections containing a few relevant images only. A careful selection of negative examples as prototypes can solve this problem. This means that prototypes now contain positive as well as negative examples. Current research in the pattern recognition field is considering how to deal with this issue [82], but no application to CBIR has found its way yet.

With the proposed directions, we believe the current small bridge across the semantic gap may be strengthened so that it will be able to support heavier traffic.

Bibliography

[1] C. Ahlberg and B. Shneiderman. Visual information seeking: tight coupling of dynamic query filters with starfield displays. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 313–317. ACM Press, 1994.

[2] D. Aigrain, H. Zhang, and D. Petkovic. Content–based representation and retrieval of visual media: A state of the art review. *Multimedia Tools and Applications*, 3(3):178–202, 1996.

[3] T.D. Alter and R. Basri. Extracting salient curves from images: an analysis of saliency network. *International Journal on Computer Vision*, 27(1):51–69, 1998.

[4] K. Barnard, P. Duygulu, R. Guru, P. Gabbur, and D. Forsyth. The effects of segmentation and feature choice in a translation model of object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 675–682. IEEE Computer Society, 2003.

[5] M. Basseville. Distance measures for signal processing and pattern recognition. *Signal Processing*, 18(4):349–369, 1989.

[6] B. Bederson. PhotoMesa: A zoomable image browser using quantum treemaps and bubblemaps. In *ACM Symposium on User Interface Software and Technology, CHI Letters*, volume 3, pages 71–80, 2001.

[7] B. Bhanu, J. Peng, and S. Qing. Learning feature relevance and similarity metrics in image databases. In *CBAIVL '98: Proceedings of the IEEE Workshop on Content - Based Access of Image and Video Libraries*, page 14. IEEE Computer Society, 1998.

[8] U. Brandes, M. Gaertler, and D. Wagner. Experiments on graph clustering algorithms. In *Proceedings of 11th European Symposium Algorithms*, volume 2832 of *Lecture Notes in Computer Science*, pages 568–579. Springer-Verlag, 2003.

[9] S. Bres and J. Jolion. Detection of interest points for image indexation. In *VISUAL '99: Proceedings of the Third International Conference on Visual Information and Information Systems*, Lecture Notes in Computer Science, pages 427–434. Springer-Verlag, 1999.

[10] E. Bruno, N.M. Loccoz, and S.M. Maillet. Learning user queries in multimodal dissimilarity spaces. In *Proceedings of the 3rd International Workshop on Adaptive Multimedia Retrieval*, Lecture Notes in Computer Science, pages 168–179. Springer-Verlag, 2005.

[11] G. Caene, G. Frederix, A.A.M. Kuijk, E.J. Pauwels, and B.A.M. Schouten. Show me what you mean! pariss: A CBIR-interface that learns by example. In *VISUAL '00: Proceedings of the 4th International Conference on Advances in Visual Information Systems*, Lecture Notes in Computer Science, pages 257–268. Springer-Verlag, 2000.

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

[13] A. Carkacioglu and F.Y. Vural. Learning similarity space. In *IEEE International Conference on Image Processing*, volume 1, pages 405–408. IEEE Computer Society, 2002.

[14] C. Carson. Blobworld: image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1026–1038, 2002.

[15] C.C. Chang and C.J. Lin. LIBSVM: a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/˜cjlin/libsvm.

[16] C. Chen. Generalised similarity analysis and pathfinder network scaling. *Interacting with Computers*, 10(2):107–128, 1998.

[17] C. Chen, G. Gagaudakis, and P. Rosin. Content-based image visualization. In *IV '00: Proceedings of the International Conference on Information Visualisation*, pages 13–18. IEEE Computer Society, 2000.

[18] J.Y. Chen, C.A. Bouman, and J.C. Dalton. Hierarchical browsing and search of large image databases. *IEEE Transactions on Image Processing*, 9(3):442–455, 2000.

[19] M. Chen, M. Christel, A. Hauptmann, and H. Wactlar. Putting active learning into multimedia applications: dynamic definition and refinement of concept classifiers. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 902–911, 2005.

[20] Y. Chen, X.S. Zhou, and T.S. Huang. One–class SVM for learning in image retrieval. In *IEEE International Conference on Image Processing*, volume 1, pages 34–37. IEEE Computer Society, 2001.

[21] L. Cinque, F. Lecca, S. Levialdi, and S. Tanimoto. Retrieval of images using rich region descriptions. In *ICPR '98: Proceedings of the 14th International Conference on Pattern Recognition-Volume 1*, pages 899–901. IEEE Computer Society, 1998.

[22] L. Cinque, S. Levialdi, A. Malizia, and K.A. Olsen. A multidimensional image browser. *Journal of Visual Language and Computing*, 9(1):103–117, 1998.

[23] M. Crucianu, M. Ferecatu, and N. Boujemaa. Relevance feedback for image retrieval: a short survey. In *"State of the Art in Audiovisual Content-Based Retrieval, Information Universal Access and Interaction, Including Datamodels and Languages", Report of the DELOS2 European Network of Excellence*, page 20, 2004.

[24] M. Das, E.M. Riseman, and B. Draper. FOCUS: Searching for multi-colored objects in a diverse image database. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 756–761. IEEE Computer Society, 1997.

[25] O. de Bruijn and R. Spence. Rapid serial visual presentation: A space-time trade-off in information presentation. In *Proceedings of the working conference on Advanced Visual Interfaces*, pages 189–192. ACM Press, 2000.

[26] S. Deb and Y. Zhang. An overview of content-based image retrieval techniques. In *IEEE International Conference on Advanced Information Networking and Application*, page 59. IEEE Computer Society, 2004.

[27] A. Dimai. Unsupervised extraction of salient region-descriptors for content based image retrieval. In *ICIAP '99: Proceedings of the 10th International Conference on Image Analysis and Processing*, page 686. IEEE Computer Society, 1999.

[28] R.C. Dubes. How many clusters are best? - an experiment. *Pattern Recognition*, 20(6):645–663, 1987.

[29] R.P.W. Duin, D. Ridder, and D.M.J. Tax. Experiments with a featureless approach to pattern recognition. *Pattern Recognition Letters*, 18:1159–1166, 1997.

[30] J.P. Eakins. Towards intelligent image retrieval. *Pattern Recognition*, 35(1):3–14, 2002.

[31] I. El-Naqa, Y. Yang, N.P. Galatsanos, R.M. Nishikawa, and M.N. Wernick. A similarity learning approach to content based image retrieval: application to digital mammography. *IEEE Transactions on Medical Imaging*, 23(10):1233–1244, 2004.

[32] M. Ferecatu, M. Crucianu, and N. Boujemaa. Sample selection strategies for relevance feedback in region-based image retrieval. In *K. Aizawa, Y. Nakamura, and S. Satoh (Eds.): Pacific-Rim Conference on Multimedia, Lecture Notes in Computer Science*, pages 497–504. Springer-Verlag, 2004.

[33] M.M. Fleck. Some defects in finite-difference edge finders. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(3):337–345, 1992.

[34] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: the qbic system. *IEEE Computer*, 28(9):23–32, 1995.

[35] J.M. Geusebroek, G.J. Burghouts, and A.W.M. Smeulders. The Amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112, 2005.

[36] J.M. Geusebroek and A.W.M. Smeulders. A six-stimulus theory for stochastic texture. *International Journal of Computer Vision*, 62(1/2):7–16, 2005.

[37] T. Gevers and A. Smeulders. Pictoseek: Combining color and shape invariant features for image retrieval. *IEEE Transactions on Image Processing*, 9(1):102–119, 2000.

[38] P.H. Gosselin and M. Cord. RETIN AL: an active learning strategy for image category retrieval. In *IEEE International Conference on Image Processing*, volume 4, pages 2219–2222. IEEE Computer Society, 2004.

[39] G.D. Guo, A.K. Jain, W.Y. Ma, and H.J. Zhang. Learning similarity measure for natural image retrieval with relevance feedback. *IEEE Transactions on Neural Networks*, 13(4):811–820, 2002.

[40] J. He, A.H. Tan, C.L. Tan, and S.Y. Sung. On quantitative evaluation of clustering systems. *In Clustering and Information Retrieval, W. Wu and H. Xiong and S. Shekhar eds.*, pages 105–134, 2003.

[41] X. He, O. King, W.Y. Ma, M. Li, and H.J. Zhang. Learning a semantic space from user's relevance feedback for image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(1):39–48, 2003.

[42] D. Heesch and S. Ruger. Three interfaces for content-based access to image collections. In *Proceedings of the International Conference on Image and Video Retrieval*, volume 3115 of *Lecture Notes in Computer Science*, pages 491–499. Springer-Verlag, 2004.

[43] G. Hinton and S. Roweis. Stochastic neighbor embedding. *Neural Information Processing Systems*, 15:833–840, 2002.

[44] M.A. Hoang, J. Geusebroek, and A.W.M. Smeulders. Color texture measurement and segmentation. *Signal Processing*, 85(2):265–275, 2005.

[45] Q. Iqbal and J.K. Aggarwal. Retrieval by classification of images containing large manmade objects using perceptual grouping. *Patter Recognition*, 35:1463–1479, 2001.

[46] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[47] Y. Ivory and M. Hearst. The state of the art in automating usability evaluation of user interfaces. *ACM Computing Surveys*, 33(4):470–516, 2001.

[48] C.E. Jacobs, A. Finkelstein, and D.H. Salesin. Fast multiresolution image querying. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 277–286. ACM Press, 1995.

[49] A.K. Jain, R. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Inteligence*, 22(1):4–37, 2000.

[50] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[51] A.H. Kam. *A general multiscale scheme for unsupervised image segmentation.* Phd thesis, University of Cambridge, August 2000.

[52] T. Kato. Database architecture for content-based image retrieval. In *Proceedings of the SPIE Image Storage and Retrieval Systems*, volume 1662, pages 112–123. A.A. Jambardino and W.R. Niblack eds, 1992.

[53] D.A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 7(1):100–107, 2002.

[54] M.L. Kherfi and D. Ziou. Image retrieval based on feature weighting and relevance feedback. In *IEEE International Conference on Image Processing*, volume 1, pages 689–692. IEEE Computer Society, 2004.

[55] P. Kovesi. Matlab code for salient points detector. Software available at http://www.cs.uwa.edu.au/~pk/Research/MatlabFns/Spatial/harris.m.

[56] J. Laaksonen, M. Koskela, S. Laakso, and E. Oja. PicSOM—content-based image retrieval with self-organizing maps. *Pattern Recognition Letters*, 21(13–14):1199–1207, 2000.

[57] W. Leeuw and R. Liere. Visualization of multidimensional data using structure preserving projection methods. In *Data Visualization: The State of the Art, F.H. Post and G.M. Nielson and G.P. Bonneau eds.*, pages 213–224. Kluwer, 2003.

[58] B. Li and E.Y. Chang. Discovery of a perceptual distance function for measuring image similarity. *ACM Multimedia Journal Special Issue on Content-Based Image Retrieval*, 8(6):512–522, 2003.

[59] S.D. MacArthur, C.E. Brodley, A.C. Kak, and L.S. Broderick. Interactive content-based image retrieval using relevance feedback. *Computer Vision and Image Understanding*, 88(2):55–75, 2002.

[60] S. Mahamud, L.R. Williams, K.K. Thornber, and K. Xu. Segmentation of multiple salient closed contours from real images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(4):891–897, 1999.

[61] M.K. Mandal, F. Idris, and S. Panchanathan. A critical evaluation of image and video indexing techniques in the compressed domain. *Image and Vision Computing Journal, special issue on Content-based Image Indexing*, 17(7):513–529, 1999.

[62] L.M. Manevitz and M. Yousef. One–class SVMs for document classification. *Journal of Machine Learning Research*, 2:139–154, 2004.

[63] B.S. Manjunath and W.Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842, 1996.

[64] O. Marques and B. Furht. MUSE: A content-based image search and retrieval system using relevance feedback. *Multimedia Tools and Applications*, 17(1):21–50, 2002.

[65] A.M. Martinez and J.R. Serra. A new approach to object-related image retrieval. *Journal of Visual Languages and Computing*, 11:345–363, 2000.

[66] B. M. Mehtre, M.S. Kankanhalli, and W. F. Lee. Shape measures for content based image retrieval: A comparison. *Information Processing Management*, 33(3):319–337, 1997.

[67] C. Meilhac and C. Nastar. Relevance feedback and category search in image databases. In *IEEE International Conference on Multimedia Computing and Systems*, volume 1, page 9512. IEEE Computer Society, 1999.

[68] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *IEEE International Conference on Computer Vision*, volume 1, pages 525–531. IEEE Computer Society, 2001.

[69] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part I*, pages 128–142. Springer-Verlag, 2002.

[70] T. Minka and R. Picard. Interactive learning using a "society of models". In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 447–452. IEEE Computer Society, 1996.

[71] B. Moghaddam, Q. Tian, N. Lesh, C. Shen, and T.S. Huang. Visualization and user-modeling for browsing personal photo libraries. *International Journal of Computer Vision*, 56(1–2):109–130, 2004.

[72] H. Muller, N. Michoux, D. Bandon, and A Geissbuhler. A review of content-based image retrieval systems in medicine - clinical benefits and future directions. *International Journal of Medical Informatics*, 73:1–23, 2004.

[73] H. Muller, W. Muller, S. Marchand-Maillet, T. Pun, and D.McG. Squire. Strategies for positive and negative relevance feedback in image retrieval. In *IEEE International Conference on Pattern Recognition*, volume 01, page 5043. IEEE Computer Society, 2000.

[74] M. Nakazato and T.S. Huang. 3D MARS: Immersive virtual reality for content-based image retrieval. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, page 12. IEEE Computer Society, 2001.

[75] M. Naphade, S. Basu, J.R. Smith, C. Lin, and B. Tseng. Modeling semantic concepts to support query by keywords in video. In *IEEE International Conference on Image Processing*, volume 1, pages 145–148, 2002.

[76] G.P. Nguyen and M. Worring. Relevance feedback based saliency adaptation in CBIR. *ACM Multimedia Systems*, 6(10):499–512, 2005.

[77] G.P. Nguyen and M. Worring. Scenario optimization for interactive category search. In *MIR '05: Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*, pages 159–166. ACM Press, 2005.

[78] G.P. Nguyen and M. Worring. Interactive access to large image collections using similarity based visualization. *Journal of Visual Languages and Computing*, 2006. To appear.

[79] H.T. Nguyen and A.W.M. Smeulders. Everything gets better all the time, apart from the amount of data. In *Proceedings of the International Conference on Image and Video Retrieval*, volume 3115 of *Lecture Notes in Computer Science*, pages 33–41. Springer-Verlag, 2004.

[80] E.J. Pauwels and G. Frederix. Fiding salient regions in images - nonparametric clustering for image segmentation and grouping. *Computer Vision and Image Understanding*, 75(1–2):73–85, 1999.

[81] E. Pekalska and R.P.W. Duin. Dissimilarity representations allow for building good classifiers. *Pattern Recognition Letters*, 23:943–956, 2002.

[82] E. Pekalska, R.P.W. Duin, and P. Paclik. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition*, 39(2):189–208, 2006.

[83] G. Peters, B. Zitova, and C. Malsburg. How to measure the pose robustness of object views. *Image and Vision Computing*, 20(4):249–256, 2002.

[84] R. Pless. Image spaces and video trajectories: Using Isomap to explore video sequences. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, pages 1433–1440. IEEE Computer Society, 2003.

[85] T. Randen and J. H. Husoy. Filtering for texture classification: A comparative study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):291–310, 1999.

[86] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood. Evaluating a visualization of image similarity as a tool for image browsing. In *INFOVIS '99: Proceedings of the 1999 IEEE Symposium on Information Visualization*, page 36. IEEE Computer Society, 1999.

[87] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood. Does organisation by similarity assist image browsing? In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 190–197. ACM Press, 2001.

[88] P.L. Rosin. Edges: saliency measures and automatic thresholding. *Machine Vision and Application*, 9:139–159, 1997.

[89] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[90] Y. Rubner, J. Puzicha, C. Tomasi, and J.M. Buhmann. Empirical evaluation of dissimilarity measures for color and texture. *Computer Vision and Image Understanding Journal*, 84(1):25–43, 2001.

[91] Y. Rubner, C. Tomasi, and L.J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[92] Y. Rui and T.S. Huang. Optimizing learning in image retrieval. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 236–245. IEEE Computer Society, 2000.

[93] Y. Rui, T.S. Huang, and S.F Chang. Image retrieval: Current techniques, promising directions, and open issues. *Journal of Visual Communication and Image Representation*, 10(1):39–62, 1999.

[94] Y. Rui, T.S. Huang, and S. Mehrotra. Relevance feedback: A power tool for interactive content-based image retrieval. *IEEE Transactions on Curcuits and Video Technology*, 8:644–655, 1998.

[95] D.E. Rumelhart and D. Zipser. Feature discovery by competitive learning. *Cognitive Science*, 9:75–112, 1985.

[96] A.A. Salah, E. Alpaydin, and L. Akarun. A selective attention-based method for visual pattern recognition with application to handwritten digit recognition and face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):420–425, 2002.

[97] S. Santini, A. Gupta, and R. Jain. Emergent semantics through interaction in image databases. *IEEE Transactions of Knowledge and Data Engineering*, 13(3):337–351, 2001.

[98] S. Santini and R. Jain. Similarity measures. *IEEE Transactions on Pattern analysis and machine intelligence*, 21(9):871–883, 1999.

[99] R. Schettini, G. Ciocca, and I. Gagliardi. Content-based color image retrieval with relevance feedback. In *IEEE International Conference on Image Processing*, pages 75–79. IEEE Computer Society, 1999.

[100] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):530–535, 1997.

[101] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *International Journal on Computer Vision*, 37(2):151–172, 2000.

[102] A.T. Schreiber, B. Dubbeldam, J. Wielemaker, and B.J. Wielinga. Ontology-based photo annotation. *IEEE Intelligent Systems*, 16(3):66–74, 2001.

[103] B. Shneiderman. Information visualization: Dynamic queries, starfield displays, and lifelines. white paper:
http://www.cs.umd.edu/hcil/members/bshneiderman/ivwp.html.

[104] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.

[105] C.G.M. Snoek, J. van Gemert, J.M. Geusebroek, B. Huurnink, D.C. Koelma, G.P. Nguyen, O. de Rooij, F.J. Seinstra, A.W.M. Smeulders, C.J. Veenman, and M. Worring. The mediamill trecvid 2005 semantic video search engine. In *Proceedings of the 3th TRECVID Workshop*, 2005.

[106] C.G.M. Snoek, M. Worring, J. van Gemert, J.M. Geusebroek, D. Koelma, G.P. Nguyen, O. de Rooij, and F. Seinstra. Mediamill: Exploring news video archives based on learned semantics. In *Proceedings of ACM Multimedia, Best Technical Demonstration Award*. ACM Press, 2005.

[107] D.M. Squire, W. Muller, and H. Muller. Relevance feedback and term weighting schemes for content-based image retrieval. In *VISUAL '99: Proceedings of the Third International Conference on Visual Information and Information Systems*, pages 549–556. Springer-Verlag, 1999.

[108] E.J. Stollnitz, T.D. DeRose, and D.H. Salesin. Wavelets for computer graphics: A primer, part 2. *IEEE Computer Graphics and Applications*, 15(4):75–85, 1995.

[109] M. Stricker and M. Orengo. Similarity of color images. In *SPIE Storage and Retrieval for Image and Video Databases, W. Niblack and R.C. Jain eds.*, volume 2420, pages 381–392, 1995.

[110] M.J. Swain and D.H. Ballard. Color indexing. *International Journal on Computer Vision*, 7(1):11–32, 1991.

[111] J.B. Tenenbaum, V.D. Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2322, 2000.

[112] Q. Tian, N. Sebe, M.S. Lew, E. Loupias, and T.S. Huang. Image retrieval using wavelet-based salient points. *Journal of Electronic Imaging*, 10(4):835–849, 2001.

[113] S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *MULTIMEDIA '01: Proceedings of the ninth ACM international conference on Multimedia*, volume 9, pages 107–118. ACM Press, 2001.

[114] J. van Gemert, J.M. Geusebroek, C.J. Veenman, C.G.M. Snoek, and A.W.M. Smeulders. Robust scene categorization by learning image statistics in context. In *Proceedings of the International Workshop on Semantic Learning Applications in Multimedia (SLAM)*, page 105. IEEE Computer Society, 2006.

[115] R.C. Veltkamp and M. Hagendoorn. State-of-art in shape matching. *Multimedia Search: State of the Art, Springer-Verlag*, 2000.

[116] R.C. Veltkamp and M. Tanase. Content–based image retrieval systems: A survey. Technical Report UU-CS-2000-34, University of Utrecht, The Netherlands, October 2000.

[117] J. Vendrig, M. Worring, and A.W.M. Smeulders. Filter image browsing: interactive image retrieval by using database overviews. *Multimedia Tools and Applications*, 15(1):83–103, 2001.

[118] C. Vertan, M. Ciuc, C. Fernandez-Maloigne, and V. Buzuloiu. Browsing image databases by 2D image similarity scatter plots: Updates to the IRIS system. In *Proceedings of the International Symposium on Communications*, pages 397–402, 2002.

[119] E. Voorhees and D. Harman. TREC-10 proceedings appendix on common evaluation measures, 2001. http://trec.nist.gov/pubs/trec10/appendices/measures.pdf.

[120] K.N. Walker, T.F. Cootes, and C.J. Taylor. Locating salient facial features using image invariants. In *FG '98: Proceedings of the 3rd. International Conference on Face & Gesture Recognition*, pages 242–247. IEEE Computer Society, 1998.

[121] K.N. Walker, T.F. Cootes, and C.J. Taylor. Locating salient object features. In *Proceedings of the 9th British Machine Vision Conference*, volume 2, pages 557–566. British Machine Vision Association, 1998.

[122] J.Z. Wang, J. Li, and G. Wiederhold. SIMPLIcity: Semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(9):947–963, 2001.

[123] H. Ye and G. Xu. Similarity measure learning for image retrieval using feature subspace analysis. In *ICCIMA '03: Proceedings of the 5th International Conference on Computational Intelligence and Multimedia Applications*, pages 131–136. IEEE Computer Society, 2003.

[124] H.J. Zhang, Z. Chen, W.Y. Liu, and M. Li. Relevance feedback in content-based image search. In *PRIS '01: Proceedings of the 1st International Workshop on Pattern Recognition in Information Systems (Invited keynote)*, page 1. ICEIS Press, 2001.

[125] L. Zhang, F. Lin, and B. Zhang. Support vector machine learning for image retrieval. In *IEEE International Conference on Image Processing*, volume 2, pages 721–724. IEEE Computer Society, 2001.

[126] Y. Zhao and G. Karypis. Criterion functions for document clustering: Experiments and analysis. Technical report tr 01–40, Department of Computer Science, University of Minnesota, Minneapolis, MN, 2001.

[127] X.S. Zhou and T.S. Huang. Relevance feedback in content-based image retrieval: some recent advances. *Information Sciences*, 148:129–137, 2002.

[128] X.S. Zhou and T.S. Huang. Relevance feedback in image retrieval: a comprehensive overview. *Multimedia Systems*, 8(6):536–544, 2003.

[129] M. Zhu. Recall, precision and average precision, 2004. Report available at http://www.stats.uwaterloo.ca/stats_navigation/techreports/04WorkingPapers/2004-09.pdf.

Samenvatting

Het veld dat zich bezighoudt met het ontsluiten van beelden op basis van de inhoud, beter bekend onder haar Engelse equivalent Content-based Image Retrieval en bijbehorende acroniem CBIR, heeft door jarenlange studie een grote verzameling aan toepassingen opgeleverd die voldoet aan een varirend eisenpakket. In al deze systemen bestaat er echter een semantische kloof tussen de verwachting van gebruikers enerzijds en de zoekbekwaamheid van systemen anderzijds. Gebruikersinteractie is daarom een essentile component van elk CBIR systeem. In dit proefschrift beschrijven en ontwikkelen we verschillende componenten van interactieve CBIR systemen om de semantische kloof te dichten. De onderzoeksvragen uit de introductie worden geanalyseerd en beantwoord in de vier hoofdstukken van dit proefschrift.

Het antwoord op vraag **Q1:** *"Wat is de optimale wijze om beelden te visualiseren voor het ontvangen van waardevolle terugkoppeling van gebruikers?"* wordt beantwoord in **hoofdstuk 2** door verschillende visualisatietechnieken te onderzoeken. Eerst analyseren we de rol van visualisatie in interactieve CBIR systemen voor het zoeken en bladeren door grote beeldverzamelingen. Ondanks haar essentile rol, zien we dat bestaande visualisatietechnieken niet alle problemen oplossen die een rol spelen bij het analyseren van grote beeldverzamelingen. Deze problemen maken we expliciet en we bepalen drie algemene eisen voor visualisatie van beeldverzamelingen: overzicht, zichtbaarheid, en structuurbehoud. Voor elke eis stellen we oplossingen voor, evenals functies om de verschillende eisen te balanceren. We presenteren een visualisatieschema dat gebruikers optimaal ondersteunt bij het interacteren met grote beeldverzamelingen. Voor de experimenten gebruiken we een verzameling van 10.000 beelden uit de Corel collectie in combinatie met gesimuleerde gebruikersacties. Uit deze experimenten blijkt dat het voorgestelde visualisatieschema de resultaten voor een gegeven zoektaak significant verbetert in vergelijking met reguliere 2D rastervisualisatie, zoals die veelvuldig benut wordt in bestaande CBIR systemen.

Na het bepalen van optimale visualisatieschema's, beschrijven we de integratie van visualisatie met terugkoppelingstappen. In **hoofdstuk 3** laten we eerst zien dat huidig onderzoek naar interactieve zoekmechanismen visualisatie en terugkoppeling als twee onafhankelijke elementen beschouwt. Dit betekent dat bestaand onderzoek zich enerzijds concentreert op leermethoden voor effectief gebruik van relevante terugkoppeling en zich anderzijds focusseert op effectieve methoden voor het overbren-

Summary in Dutch

gen van informatie naar de gebruiker. Wij stellen een efficint interactief zoeksysteem voor waarbij deze twee stappen daadwerkelijk gentegreerd zijn. In een dergelijk gentegreerd systeem bestaan verscheidene vrijheidsgraden, zoals de gelijkenisfunctie, het aantal zichtbare beelden, de beeldgrootte, de visualisatiewijze en mogelijke terugkoppelingmanieren. Het is onhaalbaar om via gebruikersstudies voor al deze variabelen een optimale waarde te bepalen. We ontwikkelen daarom zoekscenario's waarbij taken en gebruikersacties gesimuleerd worden. Dit geeft ons een antwoord op vraag **Q4:** *"Hoe de performance van gentegreerde interactieve CBIR systemen objectief te evalueren?"* Van daaruit wordt het voorgestelde integratieschema geoptimaliseerd, gebaseerd op objectieve randvoorwaarden en evaluatiecriteria. Op deze wijze worden de vrijheidsgraden gereduceerd. De resterende vrijheidsgraden kunnen met behulp van gebruikersstudies worden gevalueerd op een systeem dat geavanceerde gelijkenisgebaseerde visualisatie integreert met interactief leren. We voeren uitgebreide experimenten uit met verschillende beeldverzamelingen op het interactief zoeken van categorien. De resultaten op basis van het voorgestelde simulatieschema laten zien dat het gecombineerde gebruik van geavanceerde visualisatie en interactief leren loont op alle datasets. Ons systeem is gentegreerd in de MediaMill video zoekmachine, die als *beste technische demo* bekroond is op de ACM Multimedia conferentie in 2005 [106].

De bovenstaande hoofdstukken beschrijven het interactievraagstuk op een systeemniveau. In de volgende twee hoofdstukken focusseren we op specifieke componenten van het interactieraamwerk (zie figuur 1.1), namelijk de kenmerken $\mathcal{F}$ en de gelijkenis $\mathcal{S}$.

In **hoofdstuk 4** beschrijven we het leren van gelijkenis tussen beelden (vraag **Q3:** *"Hoe kunnen we iteratief gelijkenis leren op een niveau hoger dan het primitieve niveau?"*). We introduceren een nieuwe benadering om het verschil tussen beelden te leren voor interactief zoeken in CBIR. We demonstreren dat in de literatuur gelijkenis vaak geleerd wordt via de kenmerkruimte door kenmerkselectie, kenmerkweging of een geparameteriseerde functie van de kenmerken. Anders dan bestaande technieken gebruiken we relevante terugkoppeling om in plaats van de gelijkenisruimte de verschilruimte aan te passen, zonder terug te gaan naar de kenmerkruimte. Dit heeft het grote voordeel dat directe manipulatie op de verschillen plaatsvindt. Om de verschilruimte te creren gebruiken we de methode van Pekalska. Nadat de gebruiker terugkoppeling geeft passen we een n-klasse SVM classificatiemethode toe om de ruimte zo aan te passen dat relevante beelden bij elkaar blijven, terwijl irrelevante beelden weggeduwd worden volgens de techniek in [39]. Resultaten op de Corel dataset van 10.000 beelden en de TrecVid collectie van 43.907 beelden laten zien dat onze voorgestelde oplossing niet alleen intutief is, maar dat het de zoekresultaten ook significant verbetert.

Tot slot, om vraag **Q2:** *"Hoe de saillantheid van kenmerken, namelijk punten, regio's, of lijnen, te bepalen opdat het context- en gebruikersinterpretatie-afhankelijk is?"* te beantwoorden, beschrijven we in **hoofdstuk 5** interactie met in het oog springende details, in het bijzonder punten, regio's en lijnen in beelden. In de literatuur focusseert men vooral op interactie met veranderende globale beeldkenmerken. De interactieve saillante detail definitie gaat verder dan een samenvatting van het beeld in saillante details. We verbeteren de gebruikers- en contextafhankelijke definitie van saillante details dynamisch met behulp van relevante terugkoppeling. Om dat te bereiken stellen we een interactieraamwerk voor waarbij veelzeggende details vanuit het perspectief van de gebruiker worden bezien. Een aantal instantiaties van het raamwerk worden gepresenteerd. We passen onze benadering toe op het verfijnen van zoekvragen in detailgebaseerde beeldontsluiting met saillante punten en regio's. Experimentele resultaten bewijzen de effectiviteit van het aanpassen van saillantheid op basis van gebruikersterugkoppeling in het zoekproces.

Concluderend: in dit proefschrift hebben we een nieuwe generatie van geavanceerde zoekmechanismen voor beelden gentroduceerd. Een optimaal zoeksysteem dat zowel visualisatie als terugkoppelingstappen integreert is voorgesteld. We hebben kenmerken en gelijkenis naar de gebruiker gebracht en op die manier een smalle brug over de semantische kloof geslagen. Verder onderzoek langs de voorgestelde richtingen zal deze brug verder versterken zodat druk verkeer mogelijk wordt.

Acknowledgements

Doing a PhD is a hard, but rewarding task. I am very happy with the accomplishment of the PhD, and more importantly with what I have learned over the past four years. Besides the knowledge I have obtained, I have improved my confidence and independency in doing research. This success does not come merely by the effort of myself, but also from the support and help of many other people. I would like to take this opportunity to express my gratitude to them.

Firstly, I would like to thank my supervisor, Marcel Worring, for his openness, and excellent supervision. It has been a great opportunity for me to work with him. Many thanks to him for teaching me how to pose research questions, understand problems, organize schedules, and write scientific papers. I would like to thank my promoter, Arnold Smeulders, for having me as one of his PhD students. Although we only had few meetings, these were always valuable and yielded interesting discussions. With his academic expertise, advice was very helpful to my research. I thank him for providing me with an opportunity to be the ISIS Soos chairwoman. While doing this task, I have built up my confidence in research as well as in social activities.

I thank Dennis and Richard for their help in all kind of troubles caused by my unprofessional programming. With their zeal I have been able to create very nice systems. Thanks also to Jan-mark, Cor, Nicu, and Sjaak for their expert suggestions and discussions of my research.

I would like to thank everyone in the ISIS group. It has been a great pleasure to work with all of you. Informal conversations at lunch time, cake time, borrels, and outside trips will always be great memories. I thank Cees for his assistance, suggestions, and for always being nice to me even when I interrupted his concentration:. I thank Jan for his very nice attitude, and for a lot of interesting conversations. I would like to express my gratitude to Thang, Hieu, and Kien, who have been kind and helpful friends.

Thank to Virginie and Suna, who have helped me in handling many paper work and formality documents in a very prompt and efficient manner. I would like to thank the ICT group, with special thanks to Stephan, for all their assistance when I had trouble with my computer.

I would like to thank the people working in the ImIk project, Laura, Guus and Bob. The monthly meetings were very helpful with a lot of invaluable comments and suggestions.

I thank all my Vietnamese friends. With them I always have a feeling of being at home. I am thankful for everything we have shared together that I will always remember.

My profound thank are send to my family: my parents, my parents-in-law, my sisters, and sisters-in-law. Though they are very far away from me, they always stand by me and support me in any decision I make. Most importantly, I would like to thank my wonderful husband for his support over the years. Without his love, his encouragement and understanding, life would be much more difficult. To him, I send my deepest thankfulness.