



HELSINKI UNIVERSITY OF TECHNOLOGY  
Department of Engineering Physics and Mathematics

# **Content-based retrieval of hierarchical objects with PicSOM**

Mats Sjöberg

Master's thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology

Supervisor    Professor Erkki Oja  
Instructor    Docent Jorma Laaksonen

Espoo, Finland, 2nd June 2006

Author:	Mats Sjöberg
Department:	Department of Engineering Physics and Mathematics
Major subject:	Computer and Information Science
Minor subject:	Engineering Physics, Computational Physics
English title:	Content-based retrieval of hierarchical objects with PicSOM
Swedish title:	Innehållsbaserad sökning av hierarkiska objekt med PicSOM
Number of pages:	77
Chair:	T-61 Computer and Information Science
Supervisor:	Professor Erkki Oja
Instructor:	Docent Jorma Laaksonen
Abstract:	<p>The amounts of multimedia content available to the public has been increasing rapidly in the last decades and it is expected to grow exponentially in the years to come. This development puts an increasing emphasis on automated content-based information retrieval (CBIR) methods, which index and retrieve multimedia based on its contents. Such methods can automatically process huge amounts of data without the human intervention required by traditional methods (e.g. manual categorisation, entering of keywords). Unfortunately CBIR methods do have a serious problem: the so-called semantic gap between the low-level descriptions used by computer systems and the high-level concepts of humans.</p> <p>However, by emulating human skills such as understanding the contexts and relationships of the multimedia objects one might be able to bridge the semantic gap. To this end, this thesis proposes a method of using hierarchical objects combined with relevance sharing. The proposed method can incorporate natural relationships between multimedia objects and take advantage of these in the retrieval process, hopefully improving the retrieval accuracy considerably.</p> <p>The literature survey part of the thesis consists of a review of content-based information retrieval in general and also looks at multimodal fusion in CBIR systems and how that has been implemented previously in different scenarios.</p> <p>The work performed for this thesis includes the implementation of hierarchical objects and multimodal relevance sharing into the PicSOM CBIR system. Also extensive experiments with different kinds of multimedia and other hierarchical objects (segmented images, web-link structures and video retrieval) were performed to evaluate the usefulness of the hierarchical objects paradigm.</p> <p>Keywords: content-based retrieval, self-organizing map, multimedia databases</p>

Utfört av:	Mats Sjöberg
Avdelning:	Avdelningen för teknisk fysik och matematik
Huvudämne:	Informationsteknik
Biämne:	Teknisk fysik, beräknings fysik
Arbetets namn:	Innehållsbaserad sökning av hierarkiska objekt med PicSOM
Title in English:	Content-based retrieval of hierarchical objects with PicSOM
Sidoantal:	77
Professur:	T-61 Computer and Information Science
Övervakare:	Professor Erkki Oja
Handledare:	Docent Jorma Laaksonen
Sammandrag:	<p>Mängden av audiovisuellt material som står till allmänt förfogande har ökat snabbt under de senaste decennierna, och förväntas fortsätta öka exponentiellt under de kommande åren. Denna utveckling ställer ett allt ökande krav på automatiska och innehållsbaserade metoder för informationssökning (eng. content-based information retrieval – CBIR), som indexerar och söker multimedia baserat på dess verkliga innehåll. Dyliga metoder kan automatiskt bearbeta stora mängder data utan mänskligt ingripande vilket traditionella metoder kräver (t.ex. manuell kategorisering, inmatning av nyckelord). Tyvärr har CBIR-metoderna ett grundläggande problem: den s.k. semantiska klyftan, dvs. den vida klyftan mellan den maskinella representationen hos datorsystem och de semantiska begrepp som människor använder.</p> <p>Men genom att efterlikna mänskliga färdigheter, t.ex. förmågan att beakta kontext och förhållanden mellan olika objekt, kan man försöka överbrygga den semantiska klyftan. Detta är utgångspunkten för denna avhandling, som föreslår en ny metod som använder sig av hierarkiska objekt kombinerat med relevansfördelning. Den föreslagna metoden kan ta tillvara naturliga förhållanden mellan multimedieobjekt och dra nytta av dessa i sökningsprocessen, vilket potentiellt kan förbättra precisionen avsevärt.</p> <p>Avhandlingens litteraturöversikt består av en genomgång av innehållsbaserad informationssökning i allmänhet samt multimodal fusion inom CBIR system och hur dyliga system har realiserats i olika sammanhang inom tidigare forskning.</p> <p>Arbetet som utfördes i samband med denna avhandling omfattar bl.a. vidareutveckling av PicSOM CBIR systemet för att omfatta även hierarkiska objekt och multimodal relevansfördelning. Utöver detta genomfördes utförliga experiment med olika typer av multimedia och olika former av hierarkiska objekt (segmenterade bilder, webblänkstrukturer samt videosökning) för att utvärdera nyttan i praktiska sammanhang.</p> <p>Nyckelord: innehållsbaserad sökning, självorganiserad karta, multimedia databaser</p>

# Acknowledgements

All the research work presented in this Master's thesis has been conducted at the Laboratory of Computer and Information Science at the Helsinki University of Technology.

First, I would like to thank my instructor and leading researcher in the PicSOM project, Doc. Jorma Laaksonen, for always guiding this thesis into the right direction. In addition to providing inspiring research ideas and a constant willingness to help with every conceivable technical problem that has arisen, he has also tirelessly proof-read the thesis and suggested many improvements in both language and content.

I owe my gratitude to Prof. Erkki Oja for supervising this thesis and for enabling it by providing excellent research facilities. I also want to thank the entire staff of the laboratory for providing a great working environment with a relaxed atmosphere. A special thanks goes to co-workers Markus Koskela and Ville Viitaniemi for much assistance and advice during the different stages of the research work.

Finally, I am grateful to my parents for sparking my interest in science and supporting me during my studies.

Espoo, 2nd June 2006

Mats Sjöberg

# Contents

<b>1</b>	<b>Introduction</b>	<b>8</b>
1.1	Motivation . . . . .	8
1.2	Background . . . . .	9
1.3	Thesis organisation . . . . .	10
<b>2</b>	<b>Content-based information retrieval and hierarchical objects</b>	<b>11</b>
2.1	Content-based information retrieval . . . . .	11
2.1.1	Object similarity . . . . .	12
2.1.2	Feature extraction . . . . .	14
2.1.3	Database indexing . . . . .	15
2.1.4	The semantic gap . . . . .	16
2.1.5	Query by example, relevance feedback . . . . .	17
2.2	Multimodal information, hierarchical objects . . . . .	18
2.2.1	Multi-part hierarchical objects . . . . .	19
2.2.2	Segmented images . . . . .	22
2.2.3	Video and audio content . . . . .	25
2.2.4	Web-link structures . . . . .	25
2.2.5	Multimedia messages . . . . .	26
<b>3</b>	<b>PicSOM CBIR system</b>	<b>28</b>
3.1	SOM algorithm . . . . .	29
3.2	Tree-structured SOMs . . . . .	30
3.3	PicSOM architecture . . . . .	32
3.4	Implementing hierarchical relevance feedback in PicSOM . . . . .	33
3.4.1	Original algorithm . . . . .	33
3.4.2	Using hierarchical objects . . . . .	36
3.5	Feature extraction framework . . . . .	37
3.6	User interface . . . . .	39

<b>4</b>	<b>Implementation</b>	<b>41</b>
4.1	Image segmentation and features . . . . .	41
4.1.1	$k$ -means segmentation . . . . .	41
4.1.2	Region merging . . . . .	42
4.1.3	Average colour feature . . . . .	44
4.1.4	Texture neighbourhood feature . . . . .	45
4.1.5	Colour moments feature . . . . .	46
4.2	Video features . . . . .	46
4.3	Web-link feature . . . . .	47
4.4	Textual features . . . . .	49
4.4.1	Character and word $n$ -grams . . . . .	49
4.4.2	Word histogram . . . . .	50
4.4.3	Binary keyword features . . . . .	50
4.5	External feature extraction . . . . .	51
4.5.1	MPEG-7 content descriptions . . . . .	51
4.5.2	Mel cepstrum . . . . .	52
<b>5</b>	<b>Experiments</b>	<b>54</b>
5.1	Performance evaluation . . . . .	55
5.1.1	Ground truth classes . . . . .	55
5.1.2	Recall-relative precision . . . . .	55
5.1.3	Average precision . . . . .	57
5.2	Retrieval with segmented images . . . . .	58
5.2.1	Experiment setting . . . . .	58
5.2.2	Results . . . . .	59
5.3	Retrieval with web-link structures . . . . .	61
5.3.1	Database collection . . . . .	61
5.3.2	Experiment setting . . . . .	62
5.3.3	Results . . . . .	62
5.4	TRECVID 2005 automatic search . . . . .	63
5.4.1	Video multi-part structure . . . . .	63
5.4.2	Semantic class models . . . . .	64
5.4.3	Text query processing . . . . .	66
5.4.4	Experiment setting . . . . .	67
5.4.5	Results . . . . .	68

<b>6</b>	<b>Conclusions and future prospects</b>	<b>69</b>
6.1	Conclusions . . . . .	69
6.2	Future prospects . . . . .	70

# Chapter 1

## Introduction

### 1.1 Motivation

Information in its every form is becoming more and more important in the world of today. Modern computer systems can store huge amounts of data, and new data is acquired at an ever increasing rate. In a recent study (Lyman & Varian 2003) it was estimated that we collectively produced around 5 exabytes\* of new information in the year 2003! All this, coupled with the fact that we are becoming more and more dependent on finding relevant information quickly in our daily private and professional lives, calls for new and better information retrieval methods to be developed.

A typical situation in information retrieval might be searching a large database of multi-media content, like text, pictures, music or video. Such a database could for example be the World Wide Web or a subset of it. The traditional indexing approach of manually entering meta-data, like textual descriptions and keywords associated with the stored data, quickly becomes infeasible when we have thousands or millions of data objects. Manually entering meta-data is not only labour intensive but also error prone. Additionally the descriptions provided by humans can vary from person to person, from language to language, and can depend on different interpretations and points of view.

The contemporary solution to the indexing problem is *content-based information retrieval* (CBIR), where the data objects are indexed by feature vectors automatically calculated from the data objects themselves. This allows us to search by the actual contents and we are no longer constrained by keywords and meta-data descriptions.

---

\* $5 \times 10^{18}$  bytes

Automatic feature extraction is of course also much faster than manual indexing, and in some cases even the only possibility.

On the other hand, creating a good feature extraction algorithm is not a trivial task. A CBIR system cannot “understand” the data that it is indexing in the sense that humans can, and therefore the retrieval results can never be perfect as judged by a human being. This means that a CBIR system is most effective as a semi-automatic tool, with a human expert pointing it in the right direction. Used in the correct manner we can get a best-of-both-worlds solution where the computational system calculates and compares low-level features, and the human user provides the high-level abstract overview. In reality the CBIR systems are far from perfect, and the human interaction with the system, trying to point it in the right direction, might be tedious and frustrating.

To improve existing CBIR systems we have looked at how humans do recognition and association. Humans do not only rely on immediate low-level data, but also benefit from *a priori* knowledge and an understanding of the context and relationships of the data items. The latter is something that has been chosen as the focal point of this thesis; because data objects in a database are seldom unrelated, it might be fruitful to use any existing object dependencies in an information retrieval system. Such dependencies might be the context of words, images found in the same web page, links to other web pages, or an e-mail message containing attachments.

Our approach is to model dependency relationships as multi-part hierarchical objects and share the user-given relevance assessments between related objects. In this way, for example an image attachment of an e-mail message that has been deemed relevant (e.g. on the basis of its colour properties) can also increase the relevance of the e-mail message that it is attached to. Thus objects that are related, but not similar in the view of low-level features can still be found in the retrieval process.

## 1.2 Background

The PicSOM system (Laaksonen et al. 2002) developed at the Laboratory of Computer and Information Science at Helsinki University of Technology is a system for content-based information retrieval. The PicSOM project was started in 1998 by Prof. Erkki Oja and Dr. Jorma Laaksonen. It was inspired by the earlier WEB-SOM (Honkela et al. 1997) text retrieval system of Academician Teuvo Kohonen and his group.

The unique approach used in PicSOM is to have several *Self-Organising Maps*

(Kohonen 2001) in parallel to index and determine the similarity of data objects. Through *query by example* a PicSOM query becomes an iterative process where the user can improve the search results by giving *relevance feedback*.

The PicSOM CBIR system was initially designed to index and retrieve images only. Segmentation was introduced into PicSOM (Viitaniemi 2002), and later we have used image segments in parallel with entire images to improve retrieval results (Sjöberg et al. 2003). This algorithm was then generalised to be used with multi-part messages (Muurinen 2003). In this thesis we generalise the concept further to include any multi-part hierarchical object relationships. Additionally we describe some experiments done with the improved PicSOM version, using MPEG-7 image features and segmented multi-part images. We also discuss an experiment with web pages, using links and web page images to form hierarchical objects. Finally we review our contribution to the TRECVID 2005 evaluations where automatic queries of video clips were performed using video, image, textual and audio features combined with semantic class models. Our results compared very favourably with the other participating groups: PicSOM had the third best group-wise results of the automatic search task.

### **1.3 Thesis organisation**

The rest of the thesis is organised as follows. In Chapter 2 content-based information retrieval is introduced together with a review of the hierarchical object paradigm. We also present some example usages of this idea. In Chapter 3 we describe the original PicSOM system and how the use of hierarchical objects was added to the PicSOM algorithm. In Chapter 4 we describe the implementation details of our experiments, such as the specific segmentation and feature extraction algorithms used. The experiments themselves are reviewed in Chapter 5, and the thesis ends with the final conclusions in Chapter 6.

# Chapter 2

## Content-based information retrieval and hierarchical objects

In the first section of this chapter we will shortly review the central content-based information retrieval (CBIR) concepts used throughout this text. Notice that by CBIR we mean content-based *information* retrieval rather than the more common *image* retrieval. This is because we also consider objects of other types than images, for example text, video, and multi-media objects in general.

As hierarchical objects are the special focus of this text, the rest of the chapter is used for presenting this concept and the different types of hierarchical objects that we have used for our experiments with the PicSOM CBIR system.

### 2.1 Content-based information retrieval

In content-based information retrieval one usually has a large database of objects from which one wishes to retrieve a smaller set of objects relevant to a specific query. In general there exist many different types of search scenarios, depending on the needs of the user. A common classification for CBIR search tasks is described in Cox et al. (2000):

*Target search.* In a target search the user is looking for a specific object in the database that he or she knows in advance. For example looking for a specific photo of the Big Ben in a database of holiday photos.

*Category search.* In a category search the user is looking for objects of a certain type. The classification of relevant objects is up to the user, but he or she

has a specific class in mind. For example if searching a database of videos, a user might search for videos containing flying airplanes, but not have a specific video in mind.

*Open-ended search or browsing.* When interactively browsing the object database, the goal of the user can be very vague, and can change radically during the search process. An example would be an artist looking for inspiration in an image database, exploring different types of images.

A good CBIR system should support most or even all of the search types given above. To be able to perform an open-ended search and browse the database intuitively we need a CBIR system that provides tools with good visualisation capabilities. Also grouping similar objects nearby in the user interface would assist the browsing. The PicSOM system supports all of these mentioned features. The difficulty of performance evaluation of CBIR systems generally increases with more abstract search tasks, with target search considered to be the least abstract and open-ended search the most abstract.

### 2.1.1 Object similarity

A fundamental operation in a traditional database search is object matching. The user makes an exact database search query, using for example a query language like SQL(Date 2003), and the retrieval system returns all matching objects. In CBIR a different approach is needed because it is very unusual to know the exact properties of the object or objects one wishes to retrieve. Objects are ranked using some kind of similarity measure, and the search query must be transformable into the same form so that the similarity of the search query with the database objects can be measured. The CBIR database queries can then, for example, be *k-nearest neighbour*, thus choosing the  $k$  most similar objects, or  *$\alpha$ -cut query* where all objects within a certain similarity threshold  $\alpha$  are chosen.

Obviously the similarity measure must be chosen so that it correlates at least to some degree with human judgement of similarity. This is only possible if there is enough overlap between the machine and human measures of similarity (Squire et al. 1999). A common way to approach the similarity measure problem is to use the *vector space model* (VSM). In the VSM we use the distance between vectors, representing the database objects, as a dissimilarity measure. So a smaller distance between the vectors of two objects means higher similarity.

A distance measure, or *metric*,  $d$  is a mapping from a pair of points in vector space  $(\mathbf{p}, \mathbf{q}) \in \mathbb{R}^K \times \mathbb{R}^K$ , where  $K$  is the dimension of the vector space, to the set of non-negative real numbers  $\mathbb{R}^+$ , such that the following equations are all satisfied:

$$d(\mathbf{p}, \mathbf{q}) = 0, \quad \text{if and only if } \mathbf{p} = \mathbf{q} \quad (2.1a)$$

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) \quad (2.1b)$$

$$\forall \mathbf{r} \in \mathbb{R}^K : d(\mathbf{p}, \mathbf{q}) \leq d(\mathbf{p}, \mathbf{r}) + d(\mathbf{r}, \mathbf{q}). \quad (2.1c)$$

A common distance measure is the *generalised Euclidean distance*

$$d_{GE}(\mathbf{p}, \mathbf{q}) = \sqrt{(\mathbf{p} - \mathbf{q})^T \mathbf{A} (\mathbf{p} - \mathbf{q})}, \quad (2.2)$$

where  $\mathbf{A}$  is a matrix with dimension  $K \times K$ . This simplifies to the normal Euclidean distance measure when we set  $\mathbf{A} = \mathbf{I}$ :

$$d_E(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\| = \sqrt{\sum_{n=1}^K (p(n) - q(n))^2}, \quad (2.3)$$

where  $p(n)$  and  $q(n)$  are the  $n$ th components of the vectors.

Another generalised distance measure is the *Minkowski distance*

$$d_{L_\lambda}(\mathbf{p}, \mathbf{q}) = \left( \sum_{n=1}^K |p(n) - q(n)|^\lambda \right)^{1/\lambda}, \quad (2.4)$$

where  $\lambda$  is a parameter specifying the base norm  $L_\lambda$ . With  $\lambda = 1$ , we get the *Manhattan* or *city-block distance*, and with  $\lambda = 2$  we get the Euclidean distance  $d_{L_2} = d_E$ .

A distance measure that gives the same ordering as the Euclidean distance for normalised vectors is the *cosine measure*

$$d_{\cos}(\mathbf{p}, \mathbf{q}) = \frac{\mathbf{p} \cdot \mathbf{q}}{\|\mathbf{p}\| \|\mathbf{q}\|} = \frac{\sum_{i=1}^K p(i)q(i)}{\sqrt{\sum_{i=1}^K p(i)^2} \sqrt{\sum_{i=1}^K q(i)^2}}. \quad (2.5)$$

In some situations it is more appropriate to use *binary features*, where the component values of the vectors are restricted to 0 and 1. A value of 1 then means that a certain feature is present, and 0 that it is not. An example would be if a certain word is present in a text or not. Binary feature vectors can be handled like real-numbered feature vectors, but there are also some special distance measures that can be used only with binary features. Note that such measures are not necessarily metric distances because they do not always satisfy the triangle inequality in (2.1c).

For a binary feature vector  $\mathbf{p}$ , we define  $P$  to be the set of features present in the corresponding object, i.e. for which  $p(n) = 1$  where  $n$  is the index of the feature. By  $|P|$  we mean the number of features present in the corresponding object, which is the same as the  $L_1$  norm of the vector  $\mathbf{p}$ .

Now we can, for example, define the following distance measures between feature sets  $P$  and  $Q$  (and their corresponding binary vectors of length  $K$ ):

*Absolute difference*

$$D_m(P, Q) = \frac{|P| - |Q|}{K}. \quad (2.6)$$

*Logical AND*

$$D_{and}(P, Q) = 1 - \frac{|P \cap Q|}{K}. \quad (2.7)$$

*Logical XOR*

$$D_{xor}(P, Q) = \frac{|(P \setminus Q) \cup (Q \setminus P)|}{K}, \quad (2.8)$$

where  $P \setminus Q = \{x \in P : x \notin Q\}$  is the set minus operator.

*Tversky's contrast model* (Tversky 1977) gives a distance measure which can be seen as a kind of combination of the above:

$$D_T = \alpha f(P \setminus Q) + \beta f(Q \setminus P) - f(P \cap Q), \quad (2.9)$$

where  $f$  is a non-negative function and  $\alpha$  and  $\beta$  are some constants.

## 2.1.2 Feature extraction

In order to utilise the vector space model presented in the previous section, we need to form a vector representation for each database object. The objects in a typical CBIR database often contain a lot of data, for example a digital image with  $1280 \times 1024$  pixels or a text with hundreds of pages. In a typical database we can have thousands of objects, so it would simply be computationally infeasible to deal directly with all this information in an on-line query. Also because of the *curse of dimensionality* (Bellman 1961) one could not even expect very good results by directly dealing with data of such high dimensionality.

A common approach to address the problem of high dimensionality in CBIR is to reduce each one of the objects in the database into one or many *low-dimensional feature vectors* by using different automatic feature extraction methods. The extracted features should describe the contents of the objects in a concise, meaningful and discriminative manner. Ideally the feature description should also correlate well

with human judgement of what is distinctive of the object. The reduction of objects into low-dimensional feature vectors, or *feature extraction*, is a very crucial point in a CBIR system. If the feature extraction produces bad, non-descriptive feature vectors, then the information retrieval system will almost certainly give poor results.

The use of automated feature extraction methods as opposed to manually entering descriptive information, like textual annotations and keywords, is in most cases inevitable due to the large amounts of data involved. Automated feature extraction is also subject-independent, while manual annotation is not. A person manually analysing the data might not be sensitive to properties that seem irrelevant to him or her, but might be relevant to some other person, or in some other situation. Additionally the mood of the annotator and other environmental factors might affect the result, giving feature descriptions that might not be very useful.

### 2.1.3 Database indexing

The index is a central data structure in an efficient CBIR system. Slow interaction frustrates users and might cause them to entirely reject the retrieval system (Miller 1968, Nielsen 1999). The index creation is often a very slow process and is thus usually performed off-line before the retrieval. The index should preferably still be able to add new objects on-line if needed, for example objects uploaded by the user during the retrieval process.

When choosing the data structure for indexing in CBIR, there are several aspects that need to be taken into account. The databases are typically very large, which means that a query implemented as a basic linear search becomes too slow. This calls for data structures that support efficient search methods. On the other hand, because the feature space in the vector space model can be of high dimensionality, it is impossible to create an index for all imaginable queries and some compromises need to be made.

In the vector space model, an essential operation of the index is to find the  $k$  nearest feature vectors (and thus their corresponding objects) of a given query, represented by a point in feature space. The resulting objects of this *k-nearest-neighbour* ( $k$ NN) search are considered to be similar to the given query in the sense that their, hopefully descriptive, feature vectors are near to the query point representation in that feature space.

There are several possible indexing structures that can be used in CBIR. Koskela (2003) divides the structures into two broad categories. The first approach transforms the feature space into a new space (usually with lower dimensionality)

where the database operations can be implemented more efficiently. Dimensionality reduction can be done for example by using principal component analysis (PCA) (Hotelling 1933), singular value decomposition (SVD), random mapping (Kaski 1998) or the Self-Organising Map (see Section 3.1).

The second indexing approach is to use divide-and-conquer strategies. This means dividing the data or feature space into clusters or subspaces, ideally so that only a few of these need to be processed per query. Methods such as Tree-structured Vector Quantisation (Chen et al. 1997) and Tree-structured Self-Organising Maps (see Section 3.2) can be used for this.

### 2.1.4 The semantic gap

Automated feature extraction methods do have a crucial problem which is very central in CBIR today: the so-called *semantic gap*. Eakins (2002) identifies three semantic levels of image queries, which can be applied more generally for any information query:

**Level 1** retrieval by *primitive* features;

**Level 2** retrieval by *logical* features or *semantic* attributes;

**Level 3** retrieval by *abstract* attributes.

Level 1 means low-level features, such as colours of an image or words or character combinations in a text. Level 2 can be more high-level descriptions such as “a picture of a car” or “a medical text”. Level 3 contains more abstract concepts such as “a picture depicting happiness” or “an ironic text”.

This division into semantic levels also clearly illustrates the semantic gap, i.e. the large separation between the high-level semantic description used by humans and the low-level features used by computer systems (Rui et al. 1999). Object descriptions in CBIR systems are mostly of semantic level 1, while humans naturally use levels 2 or 3 when describing their query target. For example in the case of a digital image, the automatically extracted low-level features only “see” local phenomena, such as colours, patterns and textures. A human analysis of the same image might be more holistic, describing for example the objects seen in the image and their relationships, not necessarily even noticing particular colours or textures.

The core of the problem lies in the difficulty of automatically extracting semantic content from a data object. The same semantic concept might have many totally

different low-level representations, and also objects with similar low-level features might have distinct semantic meanings. This was also pointed out by Gupta & Jain (1997) in the domain of digital images. Humans on the other hand are experts in recognising semantic content. And from what we know, low-level information, such as specific words in a text or colours in an image, is only one part of the information used in the recognition process of humans. A lot of *a priori* knowledge is involved, such as previous experience of similar situations, cultural context and so on. Such information is very hard to incorporate into a computer system.

To achieve something analogous to our human understanding of information content we would need to incorporate other sources, such as the aforementioned *a priori* knowledge, into our CBIR systems. One way to introduce a certain amount of learning from previous similar query situations was demonstrated by Koskela et al. (2003) using the PicSOM CBIR system. In that work the system stores the relevance history of each object, i.e. how relevant the object has been in different past queries. This information is then used as a separate statistical feature in future queries, giving a measure of semantic similarity based on previous experience. But this is of course still a long way from the wide spectrum of information, cultural knowledge and superb recognition and association skills that humans possess.

The fundamental gap between semantic and low-level information is certainly a serious limitation to the CBIR approach. But even so CBIR systems can serve as very useful semi-automatic information mining and browsing tools when looking for information in huge databases. However, in specific or *narrow* application domains, such as identifying errors in industrial products, CBIR methods can even perform as good as humans.

### 2.1.5 Query by example, relevance feedback

A common paradigm in CBIR is *query by example* which means that the queries are based on example objects from the database or some external source. The user can then grade these objects, for example, as relevant or non-relevant. The user thus becomes a part of the query process because the retrieval system is dependent on the feedback from him. This idea is discussed specifically for picture searches in (Chang & Fu 1980).

As a CBIR system usually cannot return all the relevant objects correctly in its first response, the retrieval process becomes an iterative process. The user refines the results in each query round by providing *relevance feedback* (Salton & McGill 1983, Rui et al. 1998) resulting in an iterative refinement of the query. This entire process

can be seen as a form of supervised learning, where the user steers the system by providing feedback.

A CBIR system implementing relevance feedback essentially tries to learn the optimal correspondence between the high-level human concepts and the low-level internal features used in the system. This learned correspondence is usually stored only per query session because the relevance of the different objects will change from query to query.

Relevance feedback is often implemented by adjusting internal weights, for example the influence of specific features, to better correspond to the given feedback. Relevance feedback can thus also relieve the user from the burden of manually specifying the internal weights of the system as they are automatically tuned in each query. This is a desirable property as the internal weights given to different features often do not correspond well with the human conceptual understanding of the contents of the object (Picard 1996).

The prerequisites for relevance feedback can be condensed into three minimum requirements that need to be fulfilled by a system implementing it:

1. The system must present a series of new example objects and never give the same object more than once. Thus the system will eventually present all objects and will not end up in a loop.
2. The user must judge the relevance of each presented object. Often this is done by explicitly picking the relevant ones, while the system automatically marks the other ones as non-relevant.
3. The system must change its behaviour depending on the feedback from the user. After each query round the system will have more graded objects, i.e. more information, and should therefore be able to improve the retrieval results in the next round.

The last point is really the tricky part in relevance feedback: how to use this feedback information to find the most relevant objects most efficiently.

## **2.2 Multimodal information, hierarchical objects**

As mentioned in the introduction, the focal point of this thesis is how to express and take advantage of known relationships and context of data items in information retrieval. This problem, sometimes called multimodal information fusion, is

not trivial as there is no obvious general solution. The information coming from different modalities, like text, audio or video content, may often correlate in some manner which can be beneficial to the information retrieval. But in other cases the information from different sources may even contradict each other. A general multimodal information retrieval system needs to be able to handle all such situations and combine the information from different modalities in a useful way.

There are several ways to implement multimodal fusion in information retrieval, and the problem can be addressed on different levels. On the feature level, feature vectors from different modalities can simply be concatenated, essentially creating a new feature. This is the approach taken for example in ImageRover (Cascia et al. 1998) where visual and textual features are combined into one unified vector. Another strategy is to process the different modalities separately and then merge the retrieval results in some manner in the end. Finally, the most promising technique is to implement cross-modality into the information retrieval process itself. This usually results in associating objects across different modalities. This cross-modality can for example be based on context that has been stored beforehand, for example that a certain sound clip was recorded at the same time and place as a certain video shot was taken. Another way to implement cross-modality is by statistical correlation, using for example latent semantic indexing (LSI) or cross-modal factor analysis (Li et al. 2003) and Bayesian network models (Rehg et al. 1999). Most of the existing multimodal information retrieval systems are highly specialised to work with specific types of media, for example audio and video, or even very specific domains, e.g. videos of sporting events.

In the following section, relevance sharing with multi-part hierarchical objects is presented as a novel technique for multimodal fusion. After that, different domains are presented where the hierarchical object approach has been successfully implemented. Then we shortly review some other systems solving similar retrieval problems.

### **2.2.1 Multi-part hierarchical objects**

We have chosen to model relationships between objects in a database by grouping the objects involved into *multi-part objects*. A multi-part object is a collection of objects in the database and does not have any intrinsic properties other than those of the contained objects.

This text focuses mostly on multi-part objects that can be represented in a hierarchical manner, and thus can be organised in an object tree in the database. In some situations an object in the tree can have many parents, and then the structure

is technically no longer a tree but rather a graph. Still, we usually use the term “object tree”. Also, if we consider each multi-part object as being formed of one parent plus its children (on one or more levels), a graph can be considered as a tree that only happens to share some children with other object trees. These object trees or *hierarchical objects* can be multi-modal, i.e. consist of objects of different types, and the object tree can be of any depth.

The parent objects and the sub-objects in the hierarchical object tree can be real data objects, from which we can calculate feature vectors, but in general they do not have to be. They can also be logical objects that just contain other objects. The objects that are leaf nodes in the tree, on the other hand, must be real data objects with indexable feature vectors.

Two examples of real-world multi-media objects are shown in Fig. 2.1. On the left side we have a typical media-rich e-mail message with image, audio and video attachments, where the different parts have been highlighted and numbered. On the right there is an example of a web page with textual content, embedded images and links to other web pages. Examples of hierarchical object trees created from these examples are shown in Fig. 2.2. For technical reasons the web addresses, or universal resource locators (URLs), of the web page and its containing links are collected in one separate “links” object numbered 00000004:0. The specifics of these hierarchical objects will be discussed in the following sections. The feature extraction methods used with the different media types will be discussed in detail in Chapter 4.

## Relevance sharing

The properties of each object in the hierarchical tree, i.e. the calculated feature vectors (using one or many different feature extraction methods), can be considered to be characteristic not only of the object itself, but to some extent also of its parents, children and siblings in the tree structure. We call this idea *relevance sharing*, which means that the relevance assessments originally received from user feedback will be transferred from the object to its parents, children and siblings. This means, for example, that if a certain e-mail message is considered relevant by the user in a query, its attachments would also get elevated relevance values. Additionally, when we retrieve new attachments which are similar to the previous ones, this relevance will in time also propagate to e-mail messages with similar attachments.

Fig. 2.3 shows an illustration of relevance sharing in the e-mail example. On the left the relevance goes upwards from a child video-clip object, which has perhaps been marked as relevant by the user. On the right the relevance is spread downwards from

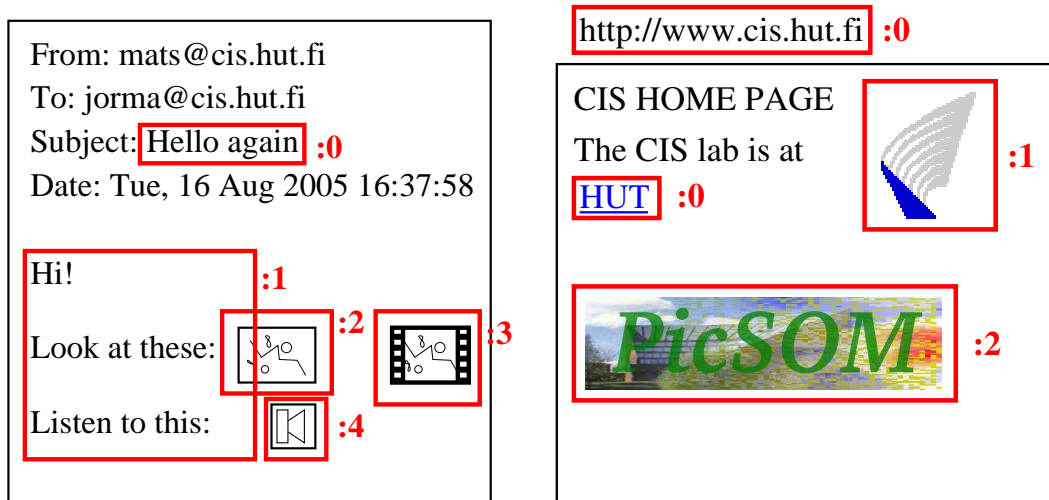


Figure 2.1: A typical e-mail message with inline attachments (left), a web page with text, link information and images (right).

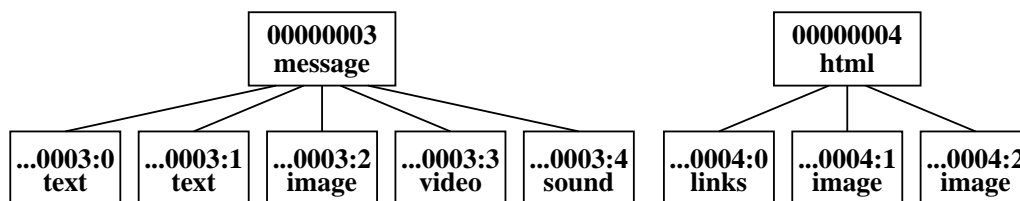


Figure 2.2: Hierarchical object trees for an e-mail message (left) and a web page (right).

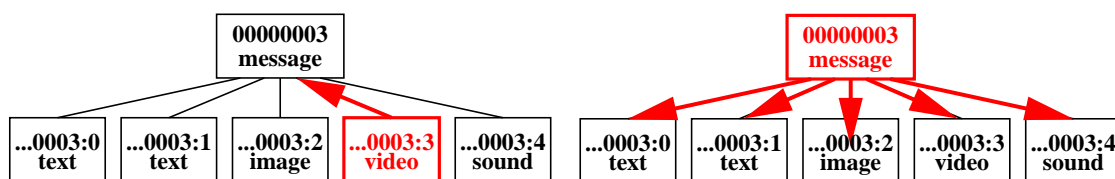


Figure 2.3: Relevance sharing in a multi-part e-mail object tree, going up from a child to its parent (left) or down from the parent to its children (right).

the parent to its children. (The originating child object does not get its relevance value elevated further if the relevance returns to it.) This process will result in the multimodal fusion of the information concerning different object types.

In the following sections some specific applications of hierarchical objects in CBIR will be described in more detail: *segmented images*, *video and audio content*, *web-link structures* and *multimedia messages*.

### 2.2.2 Segmented images

In this section we concentrate on the use of segmented images in a hierarchical object structure. Image segmentation essentially means dividing a digital image spatially into smaller disjoint parts according to some rule. The rule could, for example, simply be to divide the image into two by splitting it in the middle. To obtain more meaningful results most segmentation methods use some visual features of the image, for example colour, to determine the segments.

Intuitively one can easily understand the importance of segmentation in content-based image retrieval. Typically a picture contains several real-world objects, of which some are not relevant to a certain query. One might, for example, be searching a large image database looking for pictures of cars, but the surroundings of the car in a specific picture are not interesting. Then it would not matter if the retrieved image has a lot of grass or asphalt or if one can see a lot of the blue sky in the image or not, as long as there is a car in it. In this case it would thus be useful to be able to automatically segment the car images into at least two parts: the car and its surroundings.

Also if we consider the general problem of image understanding, we find that it is intrinsically linked to the problem of image segmentation. That is, if one understands an image, one can also tell what the distinct parts of it are. Segmentation thus seems to be a natural part of image understanding, which of course is one of the main problems in computer vision. A solution to the image understanding problem, however far away that may seem, will almost certainly contain segmentation in some form.

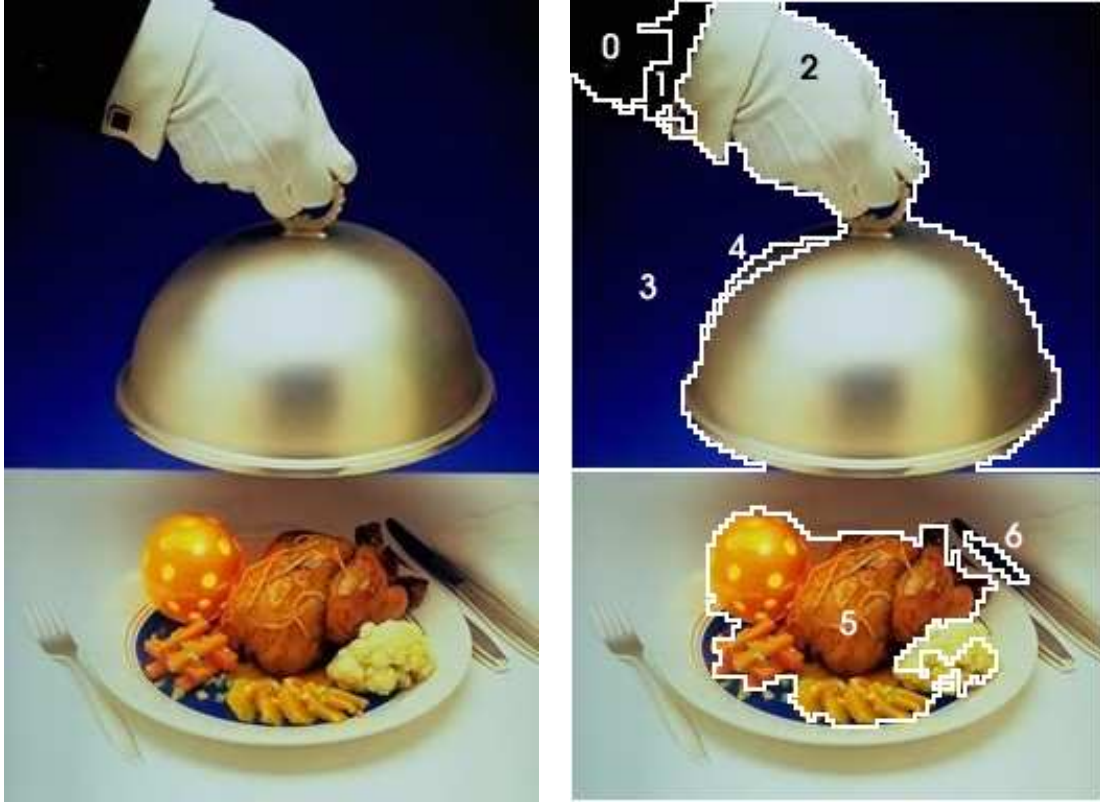
With this in mind, our segmentation algorithm should ideally produce segments that correspond to our high-level semantic understanding of the image. This means that the segments should correspond directly to what humans see as different objects in the image, like the car and its surroundings, or even smaller elements, like the wheels of the car, the sun, etc.

In reality the automatically generated segments seldom correspond directly to our understanding of the image because they are created using only low-level visual features (e.g. colour or texture). So for example, if colour is used for segmentation, two nearby but different objects with similar colour might end up in the same segment. This is actually an instance of the problem discussed in Section 2.1.4, i.e. the semantic gap between the high-level semantic description used by humans and the low-level features used by computer systems. A computer vision system simply cannot understand an image based on only low-level feature information.

But even so we think that segmentation is useful in image retrieval because different, visually homogeneous regions somehow characterise the objects and scenes in the image. That is, we believe that the use of segmentation can give more information of the composition of the image than calculating features from the entire image alone. So for example, if you say that an image has three red segments and a green one, it is much more informative than saying that its average colour is yellow.

When we have segmented an image, we can create an object hierarchy where the original unsegmented entire image is the parent object and the segments, i.e. parts of the image, are child objects in a tree. Thus relevance feedback, together with the relevance sharing in hierarchical objects discussed in the previous section, gives us a system where both similar images and images with similar segments contribute to each others' relevance values.

In Fig. 2.4 one can see an example of the use of segments in feature extraction. On the left we have calculated the average RGB colour feature (the average of the red, green and blue colour components of the image pixels) for the entire image. On the right we have segmented the image using  $k$ -means combined with region merging (see Section 4.1) and calculated the feature for each segment separately. The segmented image provides much more information than by averaging over the entire image. One could argue that the same result could be achieved by simply concatenating the colour features calculated from each segment to generate one long feature vector with all the results. And furthermore, that the extra information gained is just because we have a feature vector with higher dimensionality. But this is not the case, as each segment is treated as an object in its own right with its own low-dimensional colour feature. So image similarity increases not only from images having similar features, but also from images having similar segments. For example red segments might be found to be crucial to the retrieval in a particular case, but other colours might not be very important. Such a situation would be hard to benefit from in the retrieval process if all the colour feature data is simply concatenated in some more or less arbitrary order.



red	green	blue	segment
0.4263577	0.4061035	0.386114	0
0.0086115	0.0086115	0.0084055	1
0.0301994	0.0261996	0.0217827	2
0.6279142	0.6435601	0.5504292	3
0.0186285	0.0175655	0.2216980	4
0.1802614	0.1599783	0.2017541	5
0.6643879	0.3807732	0.0891608	6
0.0406402	0.0320094	0.0552036	6

Figure 2.4: On the left we have calculated the average RGB colour feature of an image. On the right we have segmented the image using  $k$ -means combined with region merging and calculated the average colour feature for each segment separately. Adapted from Viitaniemi (2002).

Segmentation techniques have been employed in many CBIR systems, including e.g. NetRa (Ma & Manjunath 1999), VisualSEEk (Smith & Chang 1996), Blob-World (Carson et al. 2002) and SIMPLicity (Wang et al. 2001). The PicSOM system (Laaksonen et al. 2002, Viitaniemi 2002) uses segmentation together with the hierarchical objects approach described in this section. Additionally, such methods as Unified Feature Matching (UFM) (Chen & Wang 2001) and the use of point configurations in the feature space (Dimai 1999) have been presented. These approaches differ mainly in the fashion the segment-wise similarities are combined to form image-wise similarities used in the retrieval.

### 2.2.3 Video and audio content

A video clip consists of a series of still images shown rapidly in a sequence. Typical frame-rates are around 24 or 30 images per second. There can also be a sound track associated with the video sequence. Videos can thus be stored hierarchically so that the image frames and sound tracks are sub-objects in an object tree. Alternatively we can store only important key frames of the video as separate images, and have the entire video clip as the parent object. If the video itself is very long it might also be a good idea to segment it into shorter segments, for example into different scenes. Then the entire video would be the parent, with the shorter segments as sub-objects. Additionally, key frames and sounds can be assigned as children to the video segments creating a tree with three levels. In this way we can, for example, share relevance between two different videos having similar sounds or frames.

One of the most interesting currently available video retrieval systems is the Informedia project (Wactlar et al. 1996). Informedia tries to provide full content search and browsing of video clips by integrating speech, closed captioning and image recognition. It also has good video data visualisation capabilities. Another system is proposed by Castro & Muntz (1999) that uses “context-based” indexing of video-conference recordings. It uses additional sensors to acquire for example the location and identities of people as an additional context to the actual video. Bayesian networks are used for probabilistic analysis of the sensor information.

### 2.2.4 Web-link structures

Web-link structures have previously been studied in the PicSOM system strictly from an image retrieval point-of-view in (Laakso et al. 2001). In that study the images were the focal point, and the web-link information was considered only as

auxiliary feature vectors of the images. The hierarchical object structure however, fits perfectly for storing web pages with their contents and links. We have chosen a model where each web page with its links, images and other media content is one tree with the web page itself as the root object. The links between pages are stored in separate link sub-objects, and the normal vector-based similarity approach is used to gain the connection between linked pages and pages with similar links. An example of a hierarchical web page object is shown on the right in Fig. 2.2.

A special property of the web is that images and other media types embedded in the web pages can be linked to from several different contexts. This leads to a hierarchical object structure that is not purely tree-like as certain sub-objects (images) can have many parent objects (web pages). Using the hierarchical approach together with feature-based indexing means that similarity in embedded media, links and text content can be used not only for searching web pages, but also for images and other embedded data. This information is used in all searches so that, for example, in an image search, one gains relevance information also from links and texts of related web pages.

Most systems today either use only the image data itself or only surrounding textual information and meta-data. Popular systems such as Google Image Search\* and Yahoo! Image Search† belong to the latter class. See the TASI (2004) review of image search engines for more details. The systems described in a recent review of current CBIR World Wide Web image retrieval systems (Kherfi & Ziou 2004) describes mostly engines that belong to the former class, i.e. using mostly image data, but some also combine image data with textual data from the enclosing web pages. ImageRover (Cascia et al. 1998) for example combines the visual and textual features into one unified vector. WebMars (Ortega-Binderberger et al. 2000) is the only system in the review that allows multimodal browsing and it uses a hierarchical object model. Another interesting system, which is not mentioned in the review, is AMORE (Mukherjea et al. 1999) that uses multiple media types as well.

### 2.2.5 Multimedia messages

A promising area where it is very natural to use the multi-part hierarchical object paradigm is *multimedia messages*. Multimedia messages can potentially combine all the media types that we have discussed: text, images, audio and video. Multimedia messages can for example be e-mails, SMS mobile text messages or MMS

---

\*<http://images.google.com/>

†<http://images.search.yahoo.com/>

mobile multimedia messages with both text, images and perhaps even video or sound content.

Such multi-part messages form a tree hierarchy, with the abstract message object itself as the parent object and the headers, such as sender and receiver, text and different media content as child nodes. See Fig. 2.2 (left) for an example of how this structure might look like for an e-mail message. In this case the header information is stored in the parent object. Alternatively one might have a “header” child object with its own children containing the different e-mail header fields, like sender and receiver, subject, date, content-types etc. In (Muurinen 2003) the retrieval of e-mail messages using hierarchical objects was studied by using the PicSOM CBIR system.

Again, with the hierarchical object approach we can gain relevance information from all the different media types used in the object tree. Even if we restrict our search to only one media type, for example to return only images, we can still gain from the relevance sharing of the other media types. For example, two image attachments might be deemed as similar because their e-mails discuss the same subjects, even though the images themselves do not have much in common in terms of low-level visual features.

# Chapter 3

## PicSOM CBIR system

The content-based information retrieval system PicSOM (Laaksonen et al. 2001, Laaksonen et al. 2002) has been used as a framework for the research described in this thesis. The PicSOM project was started in 1998 by Prof. Erkki Oja and Dr. Jorma Laaksonen at the Laboratory of Computer and Information Science at the Helsinki University of Technology. It was inspired by the text retrieval system WEBSOM (Honkela et al. 1997) of Academician Teuvo Kohonen and the WEBSOM group\*. PicSOM has been developed or applied in seven master's degree theses (Koskela 1999, Brandt 1999, Laakso 2000, Pakkanen 2002, Viitaniemi 2002, Rummukainen 2003, Rautkorpi 2005) and one PhD thesis (Koskela 2003).

PicSOM uses the *Self-Organising Map* (SOM) algorithm (Kohonen 2001), developed by Academician Teuvo Kohonen, to index and determine the similarity of database objects. The name PicSOM derives from the words “picture” and “SOM”. The PicSOM system was initially used only for indexing pictures, hence the name.

The novel idea in PicSOM is to use several SOMs in parallel to retrieve objects relevant to a query. These parallel SOMs have been trained with separate data sets obtained by using different feature extraction algorithms on the same objects. So each SOM arranges the same objects differently, according to its particular multi-dimensional feature vectors.

PicSOM uses the principles of *query by example* and *relevance feedback* described earlier in Section 2.1.5. This means that the system shows the user a set of database objects, which the user then indicates as relevant or non-relevant to the current query, i.e. close to or far from what he is looking for. Based on this relevance feedback information PicSOM changes its configuration so that it will display better objects in the next round. This is done by increasing the influence of those SOMs that give

---

\*<http://websom.hut.fi/websom/>

the most valuable similarity evaluation according to the current relevance feedback information. The PicSOM query process is iterative as the results are refined in each query round by using the relevance feedback given by the user. The process can thus be seen as a form of supervised learning.

The PicSOM web page, with a list of publications on PicSOM and a working demonstration is located at <http://www.cis.hut.fi/picsom>.

### 3.1 SOM algorithm

The SOM algorithm can be viewed as an elastic two-dimensional grid of artificial neuron units that is fitted to the input space in an optimal manner, thus placing similar vectors in nearby neurons on the map. Each neural unit is represented by a model vector  $\mathbf{m}_i$ . In general the SOM grid can have any dimension, but the PicSOM system uses a two-dimensional grid as this is the easiest one to visualise.

The training of the map, i.e. fitting the grid to the input vectors, is carried out as a sequential process with a step index  $t = 0, 1, 2, \dots, t_{max} - 1$ . In each step we introduce a new input vector  $\mathbf{x}(t)$  to the map, and seek the index  $c(\mathbf{x})$  of the best-matching unit (BMU), or “winner neuron”, which is the map unit that is closest to the input vector in the Euclidean sense. The model vector  $\mathbf{m}_{c(\mathbf{x})}(t)$  in this unit thus satisfies

$$\forall i : \|\mathbf{x}(t) - \mathbf{m}_{c(\mathbf{x})}(t)\| \leq \|\mathbf{x}(t) - \mathbf{m}_i(t)\|. \quad (3.1)$$

When we have found the BMU index  $c(\mathbf{x})$  we update the model vectors  $\mathbf{m}_i$  of the map for the next time step  $t + 1$  as

$$\mathbf{m}_i(t + 1) = \mathbf{m}_i(t) + h(t; c(\mathbf{x}), i)(\mathbf{x}(t) - \mathbf{m}_i(t)), \quad (3.2)$$

where  $h(t; c(\mathbf{x}), i)$  is the *neighbourhood function*. This function is usually a bell-shaped curve that is centred on the BMU index  $c(\mathbf{x})$  and decreases gradually when the distance of index  $i$  from the centre increases.

Eq. (3.2) changes the values of the map units towards the value of the input vector. The BMU is modified the most and the other map units are modified in smaller amounts as the neighbourhood function decreases with increasing distance to the BMU. This process is iterated over all the available input samples and  $h(t; c(\mathbf{x}), i)$  is let to decrease over time  $t$  so that the model vectors  $\mathbf{m}_i$  converge. The large modifications in the beginning of the iteration determine the large-scale topology of the map, whereas the small values in the end fine-tune it.

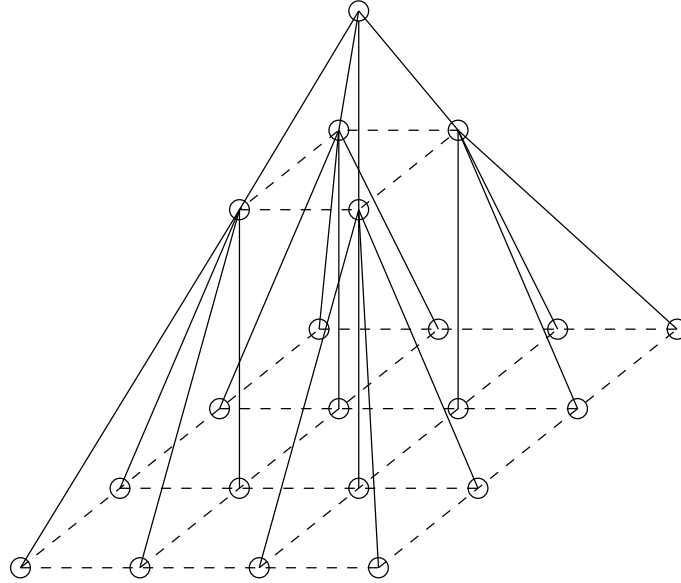


Figure 3.1: The structure of a three-layer two-dimensional TS-SOM, adopted from (Koskela 1999).

## 3.2 Tree-structured SOMs

Rather than the ordinary SOM, PicSOM uses the *Tree-structured Self-Organising Map* (TS-SOM) (Koikkalainen & Oja 1990, Koikkalainen 1994). A TS-SOM has several layers of normal SOMs with increasing size. Each unit, except those in the lowest layer, has an area of child units in the larger SOM below.

When the BMU has been found in one layer it is only necessary to search through its child units and their closest neighbours to find the BMU in the layer below. This scheme resembles the conventional tree-search algorithm, and reduces the complexity when searching for the BMU from  $O(n)$  to  $O(\log n)$ . Fig. 3.1 depicts a very small two-dimensional TS-SOM with with three SOM layers. The circles represent individual map units and the dashed lines span the individual SOM layers. A typical TS-SOM configuration for PicSOM is four layers with  $4 \times 4$ ,  $16 \times 16$ ,  $64 \times 64$  and  $256 \times 256$  units.

In the PicSOM system several feature vectors may be extracted for each object in the database, each providing a different “view” of the object. Each feature extraction algorithm gives a different similarity assessment of the objects and result in a differently organised TS-SOM. For a given object type, say images, we thus get one TS-SOM per feature extraction algorithm performed for that object type.

Each TS-SOM layer will be trained separately and each vector is used several times in the training, for example a hundred times is typical in PicSOM. After a layer has

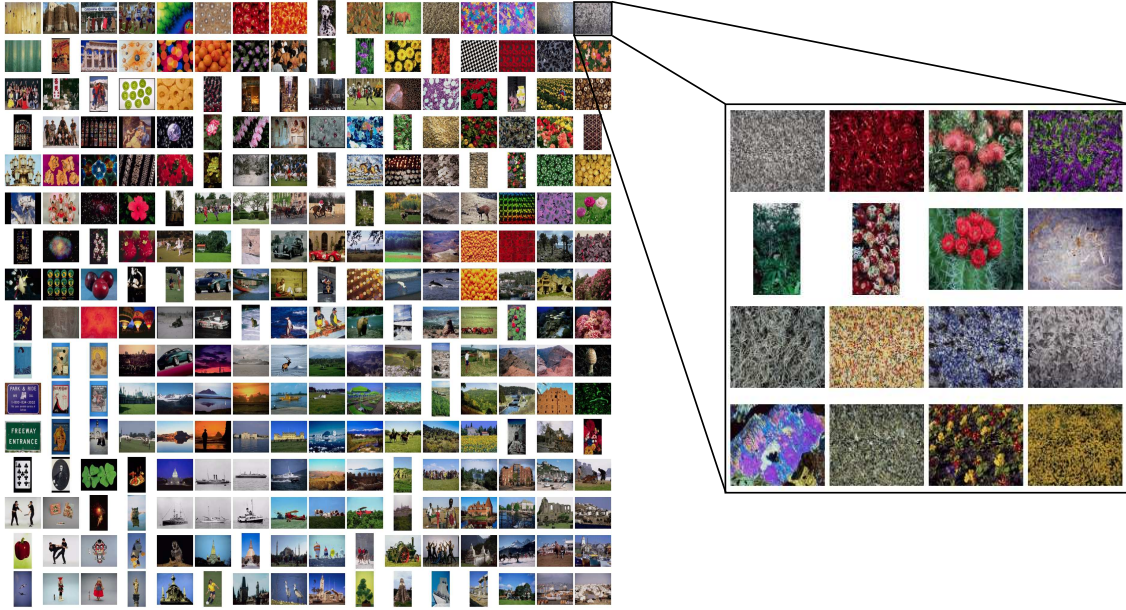


Figure 3.2: A  $16 \times 16$  layer of a map trained with MPEG-7 Edge histogram features of images from the Corel database (left). The map unit in the upper right corner is “zoomed in” and the box (right) shows the  $4 \times 4$  area of child units in the layer below.

been trained, it is fixed, and each neural unit is given an *object label* corresponding to the database object with the nearest feature vector.

The object labels combined with the hierarchical layout of the TS-SOM can be used in interactive browsing of the database. For visualisation the object labels should ideally be represented in a visual form; for example a representative image frame of a video object, or a small thumbnail image of an image object. We can then plot a visual map of a TS-SOM layer, where each neural unit will be represented by its corresponding object label. Then the map layers provide views of the database with different resolutions. When one has found an interesting object on a high layer one can “zoom in” on that object and inspect its child units on the layer below, which can be expected to be similar and give a higher resolution view of that map area.

An example of visual browsing with TS-SOMs is shown in Fig. 3.2. The figure shows a  $16 \times 16$  TS-SOM map layer created from a database of images using a texture-based visual feature (MPEG-7 Edge histogram). The map unit in the upper right corner has been “zoomed in” and the child units from the TS-SOM layer below are shown. These units (or their corresponding visual labels) tend to have a texture similar to their parent unit.

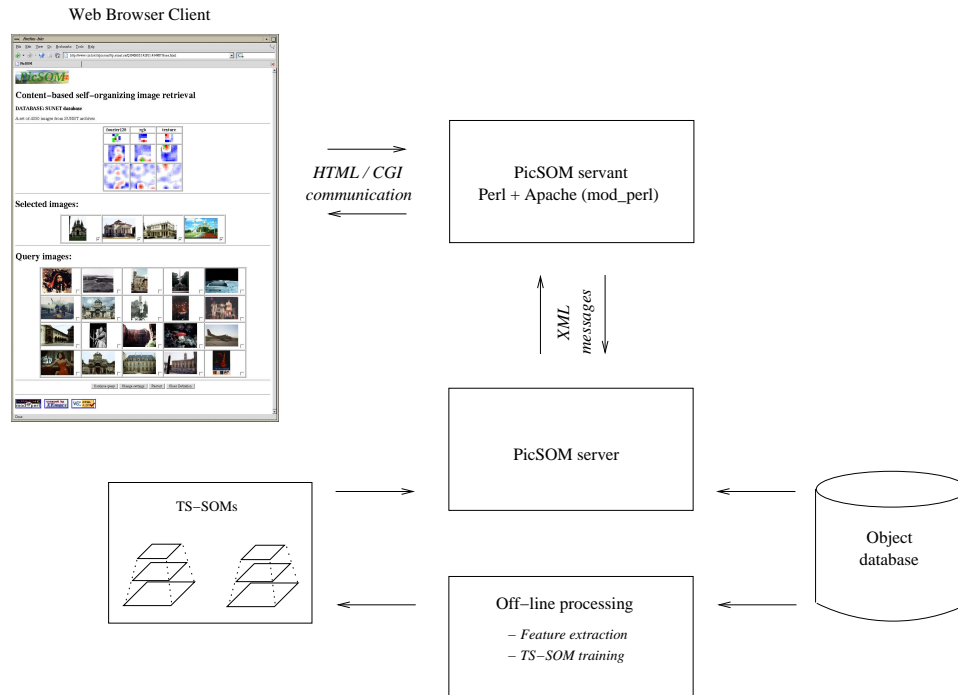


Figure 3.3: PicSOM architecture with web browser and server subsystems.

### 3.3 PicSOM architecture

The PicSOM system architecture is illustrated in Fig. 3.3. The system can be accessed via any standard World Wide Web browser, its client-side is thus platform independent. PicSOM consists of three subsystems:

1. off-line processing,
2. the PicSOM server,
3. the PicSOM servant.

The off-line processing takes care of feature extraction from the database objects and training the TS-SOMs from the feature data. The resulting TS-SOMs are stored in PicSOM's internal database. The off-line processing subsystem consists of many smaller programmes, mostly implemented in C++ and Perl.

The PicSOM servant, implemented as a CGI script\* in Perl, takes care of generating the web user interface. The servant is normally run by the Apache World Wide Web server software† using the common mod\_perl module‡, which is a fast Perl interpreter

\*The Common Gateway Interface, <http://hoohoo.ncsa.uiuc.edu/cgi>

†<http://httpd.apache.org/>

‡<http://perl.apache.org/>

running inside Apache. The servant generates web pages shown to the human user, receives input from him and communicates with the PicSOM server using special XML messages\*.

The PicSOM server receives the user input via the servant. The user feedback is a set of database objects marked as relevant or non-relevant, which are then mapped to the TS-SOMs. Using the PicSOM relevance feedback algorithm (described in Section 3.4) the server calculates a set of new objects for the next round which are then sent to the servant to be presented to the user. The PicSOM server is implemented in ISO/ANSI C++ and currently runs on Silicon Graphics systems with IRIX64, AlphaServer with Tru64 UNIX or 32- and 64-bit Linux PC systems.

All subsystems in PicSOM are completely independent and communicate through a set of predefined interfaces. This means that each part can be changed internally without disturbing the others. This principle of modularity is carried through the entire PicSOM implementation down to the internal object-oriented structure of the PicSOM server and the off-line processing implementation. This gives an open and easily extensible architecture.

The object databases are implemented directly as multi-level hierarchical directory structures in the Unix file system. The objects themselves and the features calculated from them are stored in the leaf directories of the directory tree. The top-level directories contain the trained TS-SOMs, log files of the performed queries and other information.

## **3.4 Implementing hierarchical relevance feedback in PicSOM**

In this section we first shortly present the original algorithm used in PicSOM before hierarchical objects were implemented. Then we review the changes that were made to incorporate hierarchical objects. The original PicSOM CBIR algorithm is illustrated in Fig. 3.4, the new one in Fig. 3.5.

### **3.4.1 Original algorithm**

In the query process of PicSOM the user evaluates each shown object by marking it either as relevant or non-relevant. For each object type, all relevant objects get a

---

\*<http://www.w3.org/XML/>

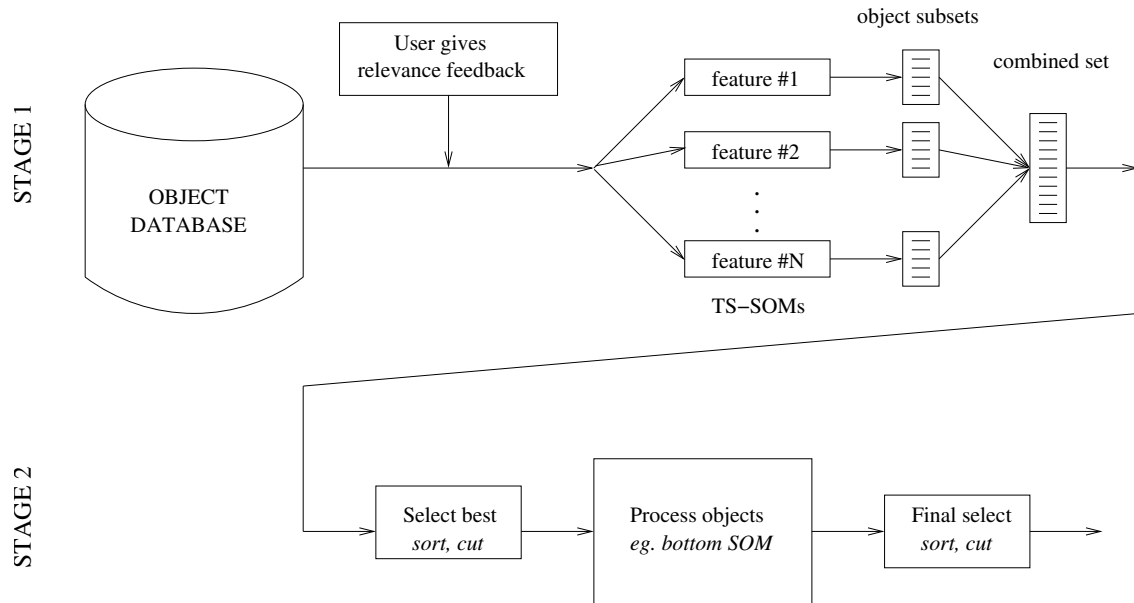


Figure 3.4: PicSOM CBIR stages, the original algorithm.

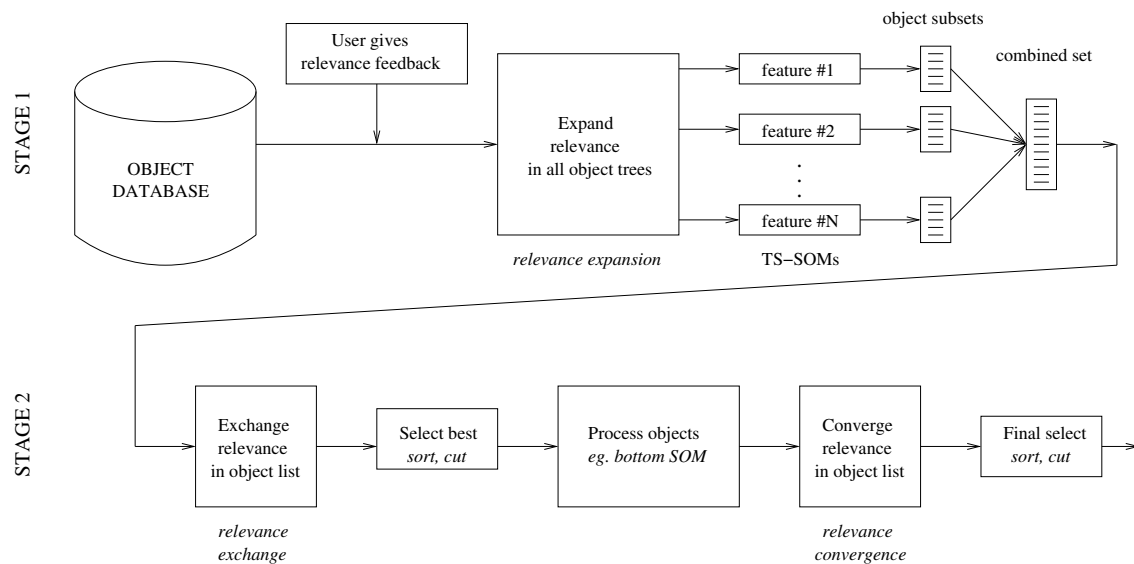


Figure 3.5: PicSOM CBIR stages with hierarchical objects.

positive weight inversely proportional to the total number of relevant objects of that type. Similarly the non-relevant objects get a negative weight inversely proportional to their total number. The grand total of all weights is thus always zero for a specific type of objects. In each layer of each TS-SOM, these values are summed into the best-matching units (BMUs) of the objects, which results in sparse value fields on the maps.

There can be several situations in any spatial neighbourhood on any particular SOM: there can be units corresponding to a) mostly relevant objects, b) mostly non-relevant objects, c) relevant and non-relevant objects mixed or d) no rated objects. The first two cases, a) and b), indicate that the feature used in creating this particular map is good at separating relevant objects from non-relevant ones. Case c) on the other hand means that the feature is not very useful in this query. Case d) does not give any information on the usefulness of the feature.

The value fields on the maps are low-pass filtered or “blurred” to spread the relevance information between neighbouring map units. This is because neighbouring map units have similar properties and it is probable that neighbours of relevant objects are relevant too. In this way all the units in the maps, and thus also the objects mapped to the units, get a qualification value depending on the local density of relevant objects. Maps with very mixed distributions of relevant and non-relevant objects get low qualification values as a result of the low-pass filtering and therefore they are automatically less influential in the object selection process.

After this a fixed number of objects from each TS-SOM with the highest qualifications values are selected. We then have  $N$  sets of object lists, where  $N$  is the number of features used. This means that in each list the objects are the most relevant for that specific feature, but not necessarily for any other feature. The lists are combined to one large list by summing the object qualification values from all SOMs. This also automatically removes any duplicates. These steps end stage 1 in Fig. 3.4, which is characterised by the fact that we consider the feature-wise object relevance values only, whereas, in stage 2 the objects are considered together with respect to all features.

In stage 2 we first select the best objects, i.e. the ones with the highest total qualification values. Then in some cases we might reconsider all objects by the sum of their bottom-most qualification values on all feature TS-SOMs. A specific number of objects that have not been shown before, usually 20, with the highest total qualification values are selected to be used as the new example objects in the next query round.

### 3.4.2 Using hierarchical objects

The implementation of relevance feedback with hierarchical objects in PicSOM was done by modifying the original algorithm described in the previous section. The old PicSOM algorithm considered only single objects, usually digital images. To this we added an object hierarchy so that the objects could have sub-objects, thus creating an object tree with parents and children. Initially this was done only for images, using their segments as sub-objects, creating a tree with a depth of one (Sjöberg et al. 2003). But in the work described in (Muurinen 2003) this was generalised further to include other multimedia objects and retrieval of multi-part messages was implemented. For this thesis, the algorithm was generalised even further to include such situations where a child can have many parents. Additionally many more scenarios were implemented, such as web-link structures and video retrieval. The processing stage of the new PicSOM algorithm is illustrated in Fig. 3.5.

Feature vectors are calculated from each object, including all objects in the hierarchical object trees, and separate TS-SOMs are trained from these as before. Normally there will be one map for each combination of the feature and its appropriate object types. So for example with segmented images, there would be separate maps calculated from the images and from the segments, even though the feature extraction algorithms may be the same.

As we now have many different object types, the user must select the *target type*, i.e. which type of objects he or she is looking for. This means that the PicSOM system uses all object types internally, but only returns objects of the specified target type. For example we might want to see only the image attachments when searching through a database of e-mail messages. But at the same time we still benefit from the *relevance sharing* between for example the texts and headers of the e-mail messages.

The relevance feedback process described in the previous section was modified so that when an object is marked as relevant the relevance is expanded from parents to children, from children to parents and possibly also to siblings depending on the type of the objects. This stage is called *relevance expansion* in the algorithm diagram.

Qualification values are then calculated for all the objects on all the TS-SOMs, and the value fields are low-pass filtered on each map as before. After combining the best object lists from all features, we get to the stage *relevance exchange*. Here the qualification values are again shared between parents and children in the hierarchical trees.

After the final processing stage, the relevance values are finally converged in the

hierarchical trees by sharing the values as before. This stage, *relevance convergence*, forms the final object list containing only objects of the desired target type. From this the 20 objects with the highest qualification values that have not been shown before, are chosen as the example objects for the next query round.

In the new selection process described above, the relevance values are thus shared both up-wards and down-wards in the object trees in three stages of the processing. As a result, the qualification values of the objects are not only determined by their own features, but also depend on the parent, child and sibling objects in their hierarchical trees.

### 3.5 Feature extraction framework

The current PicSOM system contains a separate programme that handles the feature extraction. It consists of a driver programme and a feature extraction framework that provides many extraction methods and is easy to expand in the future. The feature extraction algorithms are implemented as subclasses derived from the base class `Feature` that provides some basic facilities common to all features. The complete class structure of the feature extraction framework is illustrated in Fig. 3.6 in UML diagram form\*. The leaf nodes in the UML tree are all concrete implementations that can be instantiated, the intermediate classes are virtual ones that represent different groups of feature extraction methods. Each concrete class derived from `Feature` provides an implementation for a specific feature extraction method. The classes also contain an internal data storage class that encapsulates the feature vector data in some form.

The class `ContourFD` calculates Fourier descriptors of contours in an image. The class `PixelBased` is a virtual class that provides methods for doing pixel-based feature extraction. This means that we iterate over each pixel in the image and extract some properties that depend on this pixel alone, or possibly including its immediate surroundings. The resulting feature vector is a function of these pixel-based properties. An example of this is the Average RGB colour feature (see Section 4.1.3), implemented in the `Rgb` class. Class `ColM` calculates the Colour moments feature (see Section 4.1.5) which performs feature extraction using the three first central moments of the colour distribution in the HSV colour space.

The virtual class `TemplateBased` is a subclass of `PixelBased` that uses a template that is placed on each pixel in turn. The template is simply a collection of positions

---

\*Unified Modelling Language (UML) is a specification by the Object Management Group, see <http://www.uml.org/>

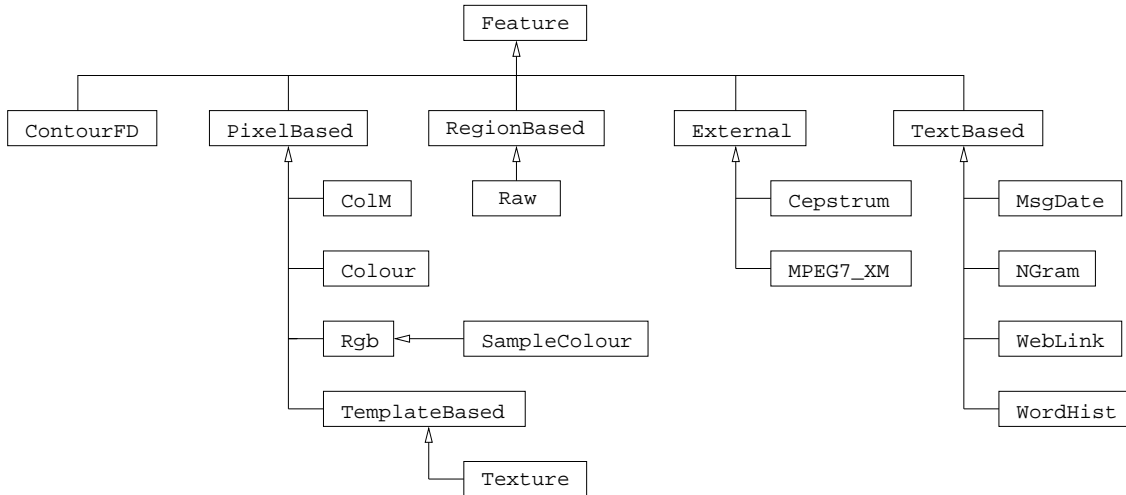


Figure 3.6: The feature extraction class structure in UML.

around the current pixel that should be considered in the feature extraction. The Texture neighbourhood feature is implemented in the **Texture** class. It uses a template that covers the 8-neighbourhood of each pixel and calculates the estimated probabilities that the neighbour pixels in each position are brighter (have higher luminance values) than the central pixel (see Section 4.1.4).

The class **RegionBased** is a virtual class representing region-based computing of features. This means that the image is separated into one or many rectangular regions for which some properties are calculated separately. The **Raw** class in the UML diagram in Fig. 3.6 is an example of this.

The **External** virtual class is for using external feature extraction software. This makes it possible to use external systems transparently through the familiar interface. Users need not even be aware that the processing is done externally. **MPEG7\_XM** which runs the external MPEG-7 XM programme (Section 4.5.1) is an example of this. **Cepstrum** creates Mel cepstrum audio features (Section 4.5.2) using an external software created at our laboratory.

**TextBased** is a base class for textual features. **NGram**, which creates character or word n-grams (Section 4.4.1), and **WordHist**, which creates word histograms (Section 4.4.2), are suitable for general text documents. **MsgDate** creates a feature vector based on the date of a message. **WebLink** creates a feature from a file containing web links (Section 4.3).

## 3.6 User interface

The screen shot in Fig. 3.7 shows an example of how the web user interface of PicSOM can look like during an interactive image query. In the top-part of the picture we can see three TS-SOMs representing three different features extracted from the images. The TS-SOMs are in three layers, with the bottom-most, largest layer last. Each layer in the TS-SOMs is represented by a coloured two-dimensional map. Notice how the resolution increases with lower SOM-layers.

The colours in the TS-SOM layers represent the relevance values of the different areas on the SOM maps. The blue spots in a map represent areas with negative relevance values, the red spots are areas with positive relevance, white means unmarked. The maps have been low-pass filtered, as discussed in Section 3.4.1. As expected, some maps have distinct red or blue clusters, while others are more uniform. More relevant images are likely to be found in the areas with high concentrations of red.

In the middle section of the user interface we can see the images that have been found, i.e. the images that have so far been marked as relevant by the user. In this example the user is apparently looking for pictures of old buildings. In the lower part we can see the query images, that is the new example images for this query round given by the PicSOM server. As we can see, even from just a few relevant images, many of the examples given by the PicSOM system seem to be correct.

Another important feature of the PicSOM user interface is the *browsing mode*, in which one can navigate a TS-SOM map layer. The browsing is initialised by clicking on some interesting area of a TS-SOM map in the normal query mode. The browsing mode will start by showing the selected area of the TS-SOM layer as a grid of neurons represented by their corresponding object labels. Fig. 3.2 gives an indication of what this might look like. In the browsing mode one can then navigate up, down, left and right in the two-dimensional TS-SOM layer by clicking on a set of arrows displayed to the right of the maps. This mechanism facilitates the open-ended search task described in Section 2.1, for example an artist looking for inspiration by more or less randomly exploring the database.

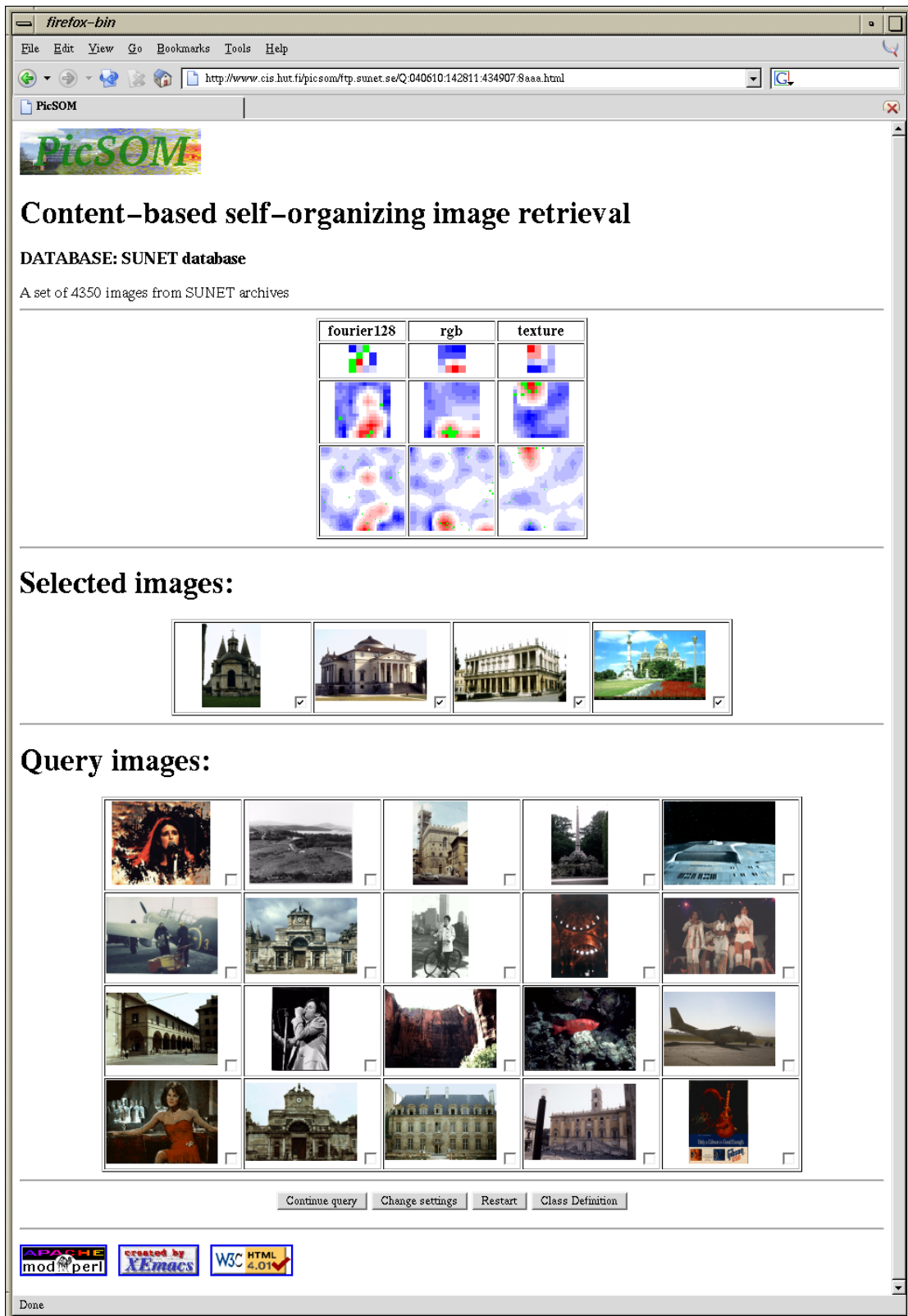


Figure 3.7: PicSOM web user interface.

# Chapter 4

## Implementation

This chapter describes the details of how hierarchical objects were implemented, and experimented with, in the content-based information retrieval system PicSOM. The actual set-up and execution of the experiments are described in Chapter 5. This chapter is separated into four sections according to feature types: segmented images, web-link features, textual features and external feature extraction. Each section describes the particular feature extraction techniques and hierarchical objects models used.

### 4.1 Image segmentation and features

The PicSOM CBIR system started out as a purely image-based retrieval system. In (Viitaniemi 2002) segmentation was introduced into PicSOM, and the next step has been the development of the hierarchical object system, described in this thesis, where the segments can be used as sub-objects to the entire images. In the experiments with segmented images we used the  $k$ -means and region merging algorithms for automatic image segmentation, as these methods were readily available and produced reasonably good results. In this section, we also introduce some feature extraction methods that were used with images and their segments.

#### 4.1.1 $k$ -means segmentation

$k$ -means is a clustering algorithm that partitions data into  $k$  clusters. In our experiments we used the *isodata* variant of the  $k$ -means algorithm (Theodoridis & Koutroumbas 1999). In image segmentation the data points of the general algorithm represent either single image pixels or disjoint groups of neighbouring pixels

(for example rectangular blocks of pixels). A localised visual feature vector is calculated for each image data point. This feature vector is used as the value of the data point in the  $k$ -means algorithm, as given in the following steps:

1. Fix the number of clusters  $k$ . Choose the initial cluster centres  $\mu_i$  in some way, for example randomly.
2. Partition the data points into clusters. Each data point belongs to the cluster with the closest centre.
3. Calculate new cluster centres according to the mean value of the associated data points from the previous step.
4. If the clustering is self-consistent, i.e. the clusters have not changed since the last step, then stop. Otherwise go to step 2.

$k$ -means partitions the image data according to the visual features only. It does not consider the spatial locations of the image data points at all. Real objects in an image usually consist of a spatial neighbourhood of similar pixels, however the  $k$ -means algorithm might consider two distant points with the same colour as belonging to the same cluster (object). One remedy to this problem is to segment the image using  $k$ -means with a large  $k$  value and then merge nearby segments, using for example region merging, as explained in the next section. Choosing the number of segments  $k$  is of course problematic as the ideal value depends on the contents of the image.

In Fig. 4.1 one can see an image of a car (top left) together with its  $k$ -means segmentation with  $k = 10$  (top right). The  $k$ -means segmentation produces a clearly over-segmented image, i.e. there are many separate segments that belong to the same object.

### 4.1.2 Region merging

In general an image region is defined as a set consisting of several adjacent pixels (Sonka et al. 1998). In the language of set theory an image region is a contiguous set of pixels.

In contrast to the  $k$ -means algorithm, region merging starts with an image that already has a large number of small segments and then merges these into larger ones. Region merging is typically used in situations where one has first segmented



Figure 4.1: An image of a car (top left) with  $k$ -means segmentation (top right) and  $k$ -means combined with region merging (bottom left). The final segmentation is visualised (bottom right) by colouring each segment with its average colour.

an image with a method producing an over-segmented image. That is, one has more segments than needed and wishes to merge some of them into larger ones.

To perform the merging one needs to define some conditions (Sonka et al. 1998). First a region  $R_i$  must satisfy some appropriate homogeneity condition:

$$H(R_i) = \text{TRUE}, \quad \forall i = 1, 2, \dots, S, \quad (4.1)$$

where  $S$  is the number of regions. This condition could for example be that the colours of the region's pixels must be sufficiently similar.

Also any two adjacent regions  $R_i$  and  $R_j$  must in the end satisfy the following:

$$H(R_i \cup R_j) = \text{FALSE}, \quad \forall i \neq j. \quad (4.2)$$

A general outline of the region merging algorithm would be as follows:

1. Start from an initially over-segmented image (e.g. from an earlier segmentation method) with many small regions satisfying condition (4.1). Note that this initial segmentation usually does not satisfy condition (4.2).

2. Define a criterion for selecting two adjacent regions to be merged.
3. Merge all adjacent regions satisfying the merging criterion.
4. If no two adjacent regions can be merged maintaining condition (4.1) then stop, otherwise continue with step 3. In the end, condition (4.2) should hold.

Our implementation of region merging differs slightly from the general algorithm. The merging criterion used in PicSOM has been based on feature vectors calculated separately for each region. More specifically we have used the HSV Colour moments feature where we treat the values in the different colour channels as separate probability distributions. The Colour moments feature is described in more detail in Section 4.1.5.

First we merge all regions with their neighbours, if the regions are smaller than a given threshold and their feature vectors are sufficiently close to each other in the feature space. After merging the smallest regions we calculate the distances between the feature vectors of adjacent regions and place the region pairs into a priority queue with the shortest distances in the beginning. Then we repeatedly merge the pairs at the beginning of the queue. After each merging we insert the distances between the new region and its neighbours in the queue. The stopping condition is fulfilled when a pre-specified maximum number of regions has been reached.

In Fig. 4.1, the image of the car (top left) that was initially  $k$ -means segmented (top right) has been additionally processed with region merging (bottom left). The resulting segmentation is much better with the combined method, but it is still not perfect. For example the ground and the wheels belong to the same segment, although they are clearly different real-world objects. A nicer visualisation of the final segmentation is also shown in Fig. 4.1 (bottom right). Each segment has been coloured with its average colour. This also illustrates the reduced feature representation of the image we would get if we represented each segment with its Average colour value.

### 4.1.3 Average colour feature

The Average colour feature is a three dimensional vector containing the average RGB values (red, green and blue) of all the pixels within a specific region. The region can be the entire image, a segment generated from the image or one of the five fixed zones seen in Figure 4.2. The zones represent different areas of the image: the upper (1), the lower (5), the left hand side (2), the right hand side (4) and the

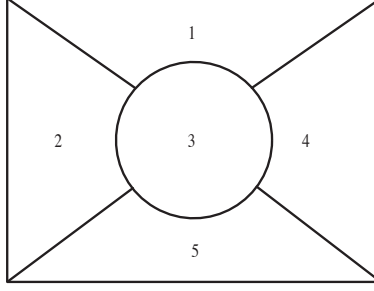


Figure 4.2: The five predefined zones used in some feature extraction algorithms in PicSOM.

central zone (3). When using such fixed zones the features are usually calculated for each zone in an image and concatenated into one 15-dimensional vector.

#### 4.1.4 Texture neighbourhood feature

Texture neighbourhood is a simple textural feature that examines the Y-values (luminance) of the YIQ colour representation of the 8-neighbourhood of each inner pixel in an image. The YIQ colour components are calculated from the RGB components in the following manner (Gonzales & Woods 1992):

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \quad (4.3)$$

The values of the feature vector are then the estimated probabilities  $\bar{P}_i$  that the neighbour pixel in position  $i$  is brighter (higher Y-value) than the central pixel  $k$ . The position indices  $i$  are designated as shown in Figure 4.3. The Y-value of the inner pixel  $k$  is  $y_k$ , its neighbour in position  $i$  has the Y-value  $y_{k,i}$ . The probability estimate  $\bar{P}_i$  is calculated as

$$\bar{P}_i = \frac{1}{n_{inner}} \sum_{k=1}^{n_{inner}} s(y_{k,i}, y_k), \text{ where } s(a, b) = \begin{cases} 1 & \text{if } a > b, \\ 0 & \text{otherwise,} \end{cases} \quad (4.4)$$

where  $n_{inner}$  is the number of inner pixels. The feature vector for one image or image segment is then

$$[\bar{P}_0 \ \bar{P}_1 \ \dots \ \bar{P}_7]^T. \quad (4.5)$$

<i>0</i>	<i>1</i>	<i>2</i>
<i>3</i>	<i>central pixel</i>	<i>4</i>
<i>5</i>	<i>6</i>	<i>7</i>

Figure 4.3: The position indices  $i$  in the 8-neighbourhood of pixel  $k$  used in the Texture neighbourhood feature.

### 4.1.5 Colour moments feature

If we treat the values in the different colour channels  $i$  as separate probability distributions we can calculate the three first central moments: mean  $M_i$ , variance  $\sigma_i$  and skewness  $s_i$ . Given  $N$  pixels with colour channel values  $c_{i,j}$ ,  $j = 1, \dots, N$ , the Colour moments components are calculated as

$$M_i = \frac{1}{N} \sum_{j=1}^N c_{i,j}, \quad (4.6)$$

$$\sigma_i = \sqrt{\frac{1}{N} \sum_{j=1}^N (c_{i,j} - M_i)^2}, \text{ and} \quad (4.7)$$

$$s_i = \sqrt[3]{\frac{1}{N} \sum_{j=1}^N (c_{i,j} - M_i)^3}. \quad (4.8)$$

The colour channels can be for example red, green and blue (RGB) or hue, saturation and brightness value (HSV). In these cases the Colour moments feature will give a 9-dimensional feature vector.

## 4.2 Video features

In addition to the MPEG-7 video features that will be discussed in Section 4.5.1 we can also calculate so called temporal versions of the three still image features introduced in the previous section, giving three new video features.

A temporal video feature is calculated as follows. Each frame of the video clip is divided into the five spatial zones of Figure 4.2: upper, lower, left, right and

centre. A still image feature vector is calculated separately for each zone and then concatenated to form frame-wise vectors. The video clip is temporally divided into five non-overlapping video sub-clips or slices of equal length. All the frame-wise feature vectors are then averaged within the slices to form a feature vector for each slice. The final feature vector for the entire video clip is produced by concatenating the feature vectors of the slices. For example using the 3-dimensional Average colour still image feature we would get a final vector with a dimensionality of  $3 \times 5 \times 5 = 75$ . Average colour (Section 4.1.3), Texture neighbourhood (Section 4.1.4) and Colour moments (Section 4.1.5) have all been used as basis feature extraction algorithms to produce such temporal features. The idea is to capture how the averaged still image features change over time in the different spatial zones.

### 4.3 Web-link feature

Web-link structures were studied as a collection of hierarchical objects in our experiments. The database was collected from real web pages using a web robot programme developed at our institution. A special web-link feature was developed to incorporate the special characteristics and properties of web links.

To explain the web-link feature it is easiest to go through the stages of its calculation. Our goal was to generate a numerical feature vector from each of the special “links” objects in the object database of PicSOM (see Fig 2.2). A links object contains the web address, i.e. Universal Resource Locator (URL), of the web page itself and the URLs of the external links to other web pages that it refers to. It also contains the URLs of embedded images.

We start by creating a separate vector for each URL contained in the links object. Later the final feature vector for the entire links object will be created by combining these vectors. The goal is to construct a vector that contains information of the URL itself and also its domain and directories. So, for example, if two pages belong to the same domain, but have different URLs, their corresponding vectors should be closer than if they did not have a common domain. To achieve this we will prune each URL by successively removing one directory level. Each pruned URL is also an URL on its own. For example, if we have this URL in the database:

<http://www.cis.hut.fi/research/IA/index.shtml>

we would prune it into the following URLs:

<http://www.cis.hut.fi/research/IA/>

<http://www.cis.hut.fi/research/>

<http://www.cis.hut.fi/>

Now each URL can be regarded as one dimension in a very high-dimensional binary space of all valid URLs. We can then construct a vector from an URL by summing together the binary vectors of the original URL and its pruned components. This will be an extremely sparse vector where the nonzero components correspond to the URLs. If we calculate such a vector from the above example and compare it to a vector from a different URL, for example:

<http://www.cis.hut.fi/research/som-research/index.shtml>

they will have at least two nonzero components in common, as the URLs share the same domain and the first directory level. Thus we have now gained the property that URLs with same domains or directories will have vectors that are closer to each other.

The described approach would, however, be infeasible due to the enormous dimensionality of the vectors. The solution, as initially suggested in (Laakso et al. 2001), is to reduce the dimensionality of the vectors by performing *random mapping* (Kaski 1998) using the *Secure Hash Algorithm* (SHA-1) (FIPS 1995).

Random mapping replaces the original high-dimensional orthogonal base with a new base of lower dimensionality that is almost orthogonal, reducing the dimensionality in a way that preserves the similarities between the vectors in an approximate manner. The SHA-1 produces a condensed 160-bit representation of a text string or message, called a *message digest*. The algorithm has its origins in cryptography and data security, and has been designed as a one-way function. This means that it is intended to be computationally infeasible to find the corresponding message to a given digest, or indeed two different messages which would produce the same digest. The latter property makes the SHA-1 useful for indexation.

The URL-wise vector is then constructed as follows:

1. The URL is pruned as described above; the original URL, the web page directory and each higher level directory up to and including the domain part.
2. We calculate the SHA-1 message digest for each of these URL parts and generate a 1024-dimensional random vector for each by looking at the first 32 bits from its SHA-1 message digest, interpreted as four 8-bit values. The first value is used as an index in the range  $[0, 255]$ , the second in  $[256, 511]$ , the third in  $[512, 767]$  and the fourth in  $[768, 1023]$ . These indices specify the four components of the 1024-dimensional vector that are set to 1. The 1020 other components of the random vector are set to 0.

3. The random vectors are then weighed so that the original URL gets the largest weight, reducing the weight for each successive directory level with the domain having the lowest weight.
4. The weighted vectors are summed and the Euclidean length of the resulting vector is normalised to unity. We thus get a normalised vector for each original URL, containing components not only for the URL itself but also for its directory parts and domain.

The final feature vector for the entire links object itself is given as the sum of these normalised vectors calculated for each URL contained in the links object. The URL of the web page itself is weighted in the sum so that its contribution is more prominent than that of the linked and embedded URLs. The vector sums are implemented according to the maximum-norm, i.e. for each component index we choose the maximum value as the component sum. This produces approximately the same results as normal addition because the vectors are sparse.

The resulting web-link feature vector now has the property we initially described, i.e. that pages with URLs having the same domain or directories would have vectors that are close. The dimensionality is also fixed at 1024 which is computationally sound.

## 4.4 Textual features

Some simple but sufficient textual features have been developed for the PicSOM CBIR system to be used in describing the contents of textual objects e.g. in multimedia messages. Some of these features are shortly presented in the following sections.

### 4.4.1 Character and word $n$ -grams

The character or word  $n$ -gram feature is calculated by first extracting every  $n$ -gram, i.e. each set of  $n$  consecutive letters or words existing in the text. The text might be pre-processed and stemmed beforehand. So for example from the text string “Hello World” we can extract the following character  $n$ -grams with  $n = 3$ , also known as trigrams:

␣␣H, ␣He, Hel, ell, llo, lo␣, o␣W, ␣Wo, Wor, orl, rld, ld␣, d␣␣.

For each  $n$ -gram we calculate an SHA-1 message digest and form a 1024-dimensional random vector from the first 32 bits in the same manner as with the web-link feature (see Section 4.3). The final feature vector is the sum of all these vectors from each  $n$ -gram in the text, normalised by their number.

#### 4.4.2 Word histogram

The Word histogram feature is calculated in three stages. First a histogram is calculated for each textual object (document) in the database giving the frequencies of all the words in that text. Then the document-specific histograms are combined into a single histogram or dictionary for the whole database. The final Word histogram feature vectors are calculated for each document by comparing its word frequencies to the dictionary, i.e. the words in the database-wide histogram. For each non-stop word (i.e. not commonly used words such as “the”) in the dictionary we calculate the tf-log-idf weight (Salton & McGill 1983) for the document. The resulting feature vector then gives the tf-log-idf values for all dictionary words in that document.

The tf-idf weight is commonly used in information retrieval and is given as the product of the *term frequency* and the *inverse document frequency*. The term frequency for a word  $k$  in one document is calculated as

$$\text{tf}_k = \frac{n_k}{\sum_{j \in K_D} n_j}, \quad (4.9)$$

where  $n_k$  is the number of occurrences of the word  $k$  and the denominator gives the number of occurrences of all dictionary words  $K_D$  in the document. The corresponding document frequency is calculated as

$$\text{df}_k = \frac{N_k}{N}, \quad (4.10)$$

where  $N_k$  is the number of documents where the word  $k$  appears, and  $N$  is the total number of documents. The tf-log-idf is then given as the product of Eq. (4.9) and the log-inverse of Eq. (4.10):

$$\text{tf-log-idf}_k = \frac{n_k}{\sum_{j \in K_D} n_j} \log \frac{N}{N_k}. \quad (4.11)$$

#### 4.4.3 Binary keyword features

An extension of the PicSOM system allows the usage of an inverted file as an index instead of the SOM (Koskela et al. 2004). The binary keyword feature is such a

feature, where the inverted file contains a mapping from words to the database objects containing them. The inverted file contains only non-stop words which have been stemmed using the Porter stemming algorithm (Porter 1980).

When a text query  $q$  is used in PicSOM, the words of the query are stemmed and stop words are removed. The system then removes those words that do not exist in the inverted file and a short list of keywords  $K_q$  characterising the search query remains. A measure  $S_{i,q}$  is calculated for each textual object  $i$  in the database as

$$S_{i,q} = \sum_{k \in K_q} \frac{\delta_{i,k}}{N_k}, \text{ where } \delta_{i,k} = \begin{cases} 1 & \text{if } k \text{ exists in } i, \\ 0 & \text{otherwise,} \end{cases} \quad (4.12)$$

and where  $N_k$  is the total number of textual objects that contain the keyword  $k$ . The higher the value of  $S_{i,q}$  for a specific textual object is, the closer it is deemed to be to the search query. Using this measure we can sort a list of the most qualified objects as used in the PicSOM algorithm (see Section 3.4).

## 4.5 External feature extraction

Many feature extraction methods have already been implemented outside of PicSOM and these external algorithms can be used transparently through the familiar PicSOM feature extraction interface. A new class structure for calling external programmes was added to our feature extraction framework (see Fig. 3.6). Extensions for new external programmes can easily be added to this structure.

### 4.5.1 MPEG-7 content descriptions

We have previously relied only on our own self-implemented features in experiments with PicSOM, which has made comparisons with other systems difficult. However, recently we have adopted the standardised content descriptions for multimedia data provided by the MPEG-7 international standard (MPEG 2002) devised by the Moving Picture Experts Group (MPEG).

Many video and still image descriptors of MPEG-7 have been integrated into the PicSOM framework as a part of the work for this thesis. These descriptors are listed in Table 4.1, with the dimensionality of each given in parentheses. Some of these MPEG-7 descriptors were used in the experiments described in Chapter 5.

Some features had to be rejected from use with PicSOM. For example Contour-based shape was very non-Euclidean and could thus not be used in PicSOM where the

<b>Texture descriptors</b>		<b>Colour descriptors</b>	
Edge histogram	(80)	Colour layout	(12)
Homogeneous texture	(62)	Colour structure	(256)
		Dominant colour	(6)
<b>Shape descriptors</b>		Scalable colour	(256)
Region-based shape	(35)		
		<b>Video descriptors</b>	
		Motion activity	(11)

Table 4.1: MPEG-7 video and still image descriptors used in our experiments, dimensionality is given in parentheses.

vector similarity comparisons are made using the Euclidean distance. As a matter of fact, the Euclidean distance is not always optimal even for the other MPEG-7 features, but in most cases it is good enough. Homogeneous texture seemed to work badly with very small segments, Colour structure did not work with segments at all. So these two methods were not used in experiments with segmented images.

Feature extraction using the standardised content descriptors provided by the MPEG-7 international standard has already been implemented in the MPEG-7 Experimentation Model (XM) Reference Software (MPEG 2001). This software (MPEG-7 XM) is freely downloadable from their web site\*.

To run the external XM programme a number of things need to be done beforehand. A few temporary input files need to be set up with the appropriate information. The segmentation data format used in PicSOM has to be converted to PBM bitmap files – one file for each image or segment. The feature extraction output from XM is saved in an XML-based format. This needs to be interpreted and stored in the internal data structures of our system. All this is done automatically on-the-fly by the `MPEG7_XM` class during the execution of the feature extraction driver programme. The XML parsing is done using the open source Libxml toolkit<sup>†</sup>.

## 4.5.2 Mel cepstrum

The Mel-scaled cepstral coefficient (MFCC), or shortly Mel cepstrum, is an audio feature commonly used for speech recognition, but can be used with other sounds as well (Davis & Mermelstein 1990). Mel cepstrum is the discrete cosine transform

\*[http://www.lis.ei.tum.de/research/bv/topics/mmdb/e\\_mpeg7.html](http://www.lis.ei.tum.de/research/bv/topics/mmdb/e_mpeg7.html)

<sup>†</sup><http://www.xmlsoft.org/>

(DCT) applied to the logarithm of the mel-scaled filter bank energies.

The discrete sampled audio signal  $\tilde{x}(n)$ , where  $n$  is the time step, is first processed into frames, i.e. small chunks over which the signal is assumed to have stationary spectral characteristics. We get  $x_t(n)$ , where  $t$  is the frame number, which is then Fourier transformed to produce the power spectrum as  $S_t(k) = |\text{DFT}(x_t(n))|^2$ . The filter bank  $\{H_j(k)\}$  is Mel-scaled, which means that it is linearly spaced at frequencies below 1 kHz and logarithmically spaced above 1 kHz. We calculate the energies from each filter  $j = 1, \dots, P$ , where  $P$  is the number of filters:

$$e_t(j) = \sum_{k=0}^{N-1} H_j(k) |S_t(k)|^2, \quad (4.13)$$

where  $N$  is the size of the DFT frame.

The Mel cepstrum feature is then calculated as the DCT of the logarithm of  $e_t(j)$ :

$$\text{MFCC}_t(i) = \sqrt{\frac{2}{P}} \sum_{j=1}^P \left\{ \log(e_t(j)) \cos\left(\frac{\pi i}{P}(j - 0.5)\right) \right\}. \quad (4.14)$$

The number of coefficients taken is usually 12, i.e.  $i = 1, \dots, 12$ , and these are organised as a vector. Finally the total power of the signal is appended to the vector giving a feature vector of length 13. The Mel cepstrum feature is calculated using an external programme created by the Speech recognition group at the Laboratory of Computer and Information Science at the Helsinki University of Technology\*.

---

\*<http://www.cis.hut.fi/projects/speech/>

# Chapter 5

## Experiments

A series of experiments were run with different types of hierarchical objects to see the effect of relevance sharing on the retrieval performance. The objective of these experiments is to see how well the system can find members of a certain class of objects in a large database, for example finding images depicting horses in a database containing a wide array of different images. In the first two experiments we compare the results of using hierarchical objects with using only single objects. In this way we can evaluate the advantage that hierarchical objects give us in information retrieval. The third experiment was the TRECVID 2005 evaluations where the goal was to compare the PicSOM system using hierarchical objects with other systems in the area of video retrieval.

This chapter starts by discussing the issue of performance evaluation in content-based information retrieval systems and how we have implemented that in our PicSOM CBIR system. This is followed by a presentation of the following experiments and their results:

*Segmented images*, with images and their segments as sub-objects using MPEG-7 still-image features.

*Web-link structures*, with web pages and their embedded information such as text, images and links represented as sub-objects. The set of features is wide because of the many different types of objects: the MPEG-7 image features, the web-link feature and the  $n$ -gram text feature.

*TRECVID 2005 automatic search*, video retrieval evaluation including video clips, key frames and text from automatic speech recognition and translation. Intelligent text query processing and positive and negative class models were also used.

## 5.1 Performance evaluation

Evaluating the performance of a content-based information retrieval system is not a trivial task. The evaluation should compare the retrieval results of the system with the “correct” results as specified by a human user. In target search (see Section 2.1), the human user has a *target class* in mind, i.e. a class of objects that he or she wishes to find as a result of the retrieval process. The target class is seldom well specified, even in the mind of the user, and can even evolve during the interactive retrieval process. Furthermore, the interpretation of the contents of an object might differ between humans; some objects might clearly belong to a certain class, while others might be unclear.

### 5.1.1 Ground truth classes

A common method of evaluating performance in CBIR is to use *ground truth classes*. A ground truth class is a preselected subset of the database that consists of objects belonging to a certain semantic class. This class is usually hand picked by human experts, for example according to some well defined agreed-upon criteria. This reduces differences of interpretation, and the class will be fixed during the retrieval process.

If the ground truth class is available in an electronic format, the query task can be performed automatically, speeding up the evaluation process substantially. The same ground truth class can also be used in different situations as long as the database remains the same. For example when using different features or competing retrieval methods, the ground truth class provides a neutral and impartial way to compare the results.

### 5.1.2 Recall-relative precision

To evaluate the performance of a CBIR system in an exact and automated manner we need a mathematical formulation. Such a formulation can be developed as follows.

In a given query we are looking for objects of the class  $\mathcal{C}$ , and the system presents the example objects for evaluation in the order  $\{I_1, I_2, \dots, I_N\}$ , where  $N$  is the total number of objects in the database. We define  $h_t = h(I_t; \mathcal{C})$  as the Boolean membership value of the object  $I_t$ , where  $t = 1, \dots, N$ , that is,

$$h_t = h(I_t; \mathcal{C}) = \begin{cases} 1, & \text{if } I_t \in \mathcal{C}, \\ 0, & \text{if } I_t \notin \mathcal{C}. \end{cases} \quad (5.1)$$

It is reasonable to assume that the correctness evaluation is independent of the order in which the objects are presented. Additionally, this formulation does not take into account that the system usually shows more than one example object per query round. However, that will merely cause a lag in the relevance feedback mechanism (because the system will not get feedback after each object) and does not qualitatively alter our reasoning. We denote the number of objects belonging to class  $\mathcal{C}$  with  $N_{\mathcal{C}}$ . The *a priori* probability of class  $\mathcal{C}$  is then  $\rho_{\mathcal{C}} = N_{\mathcal{C}}/N$ .

The evaluation then usually proceeds as follows:

1. The ground truth class  $\mathcal{C}$  is selected, for example “images of cars” in an image database.
2. The evaluation process is initialised by giving the CBIR system an object of the class  $\mathcal{C}$  as an example of what we are looking for. Because of this the effective size of the ground truth class is actually  $N_{\mathcal{C}} - 1$  and the *a priori* probability is given by  $\rho_{\mathcal{C}} = (N_{\mathcal{C}} - 1)/(N - 1)$ .
3. The evaluation then proceeds with a specified number  $N_Q$  of normal query rounds where  $N_R$  example objects are given by the CBIR system in each round. Objects belonging to our selected ground truth class  $\mathcal{C}$  are marked as relevant and the others as non-relevant, mimicking the normal human-driven interactive feedback process.
4. Steps 2. and 3. are then repeated so that each object in the class  $\mathcal{C}$  is used once as the initialising object, and the results for each time instance  $t = 1, \dots, N_Q N_{\mathcal{C}}$  are averaged over all cases. This is to assure that the results do not depend on a specific object.

With large databases we are not always interested in finding all the objects belonging to a specific class simply because there are so many of them. Also if we look at a normal human-driven query situation, browsing thousands of objects to find all objects of a given class is simply not feasible. The goal is thus often just to find a small set of objects belonging to that class. That is why the automatic evaluation process is run only for a specified number of query rounds, typically showing a total of a few thousand example objects.

Looking at the retrieval results we then determine how accurately and how exclusively the system finds the relevant objects up to different points in time. This can be presented compactly by plotting the *relative precision* against the *recall*.

Recall  $\mathcal{R}$  expresses how large a portion of the relevant object class has already been shown up to time instance  $t = 1, 2, \dots, N - 1$ :

$$\mathcal{R}(t) = \frac{\sum_{i=1}^t h_i}{N_C - 1} \in [0, 1]. \quad (5.2)$$

Precision  $\mathcal{P}$  indicates the accuracy of retrieval, i.e. how exclusively only relevant objects have been retrieved:

$$\mathcal{P}(t) = \frac{\sum_{i=1}^t h_i}{t} \in [0, 1]. \quad (5.3)$$

Furthermore, we use relative precision  $\mathcal{P}_r(t)$ , which equals precision divided by the *a priori* probability  $\rho_C$  of the class:

$$\mathcal{P}_r(t) = \frac{\mathcal{P}(t)}{\rho_C}. \quad (5.4)$$

The resulting recall–relative precision plot first shows the initial accuracy of the CBIR system, i.e. how well it finds relevant objects with very little or no user feedback. The evolution of the curve after that indicates how well the relevance feedback mechanism works. With a good relevance feedback mechanism the curve should initially rise and then turn to a slow decline when a sufficiently large portion of the relevant objects has been shown.

### 5.1.3 Average precision

In the TRECVID 2005 experiments non-interpolated average precision was used. This measure is formed by calculating the precision after each retrieved relevant object. The precision is defined to be zero for all non-retrieved relevant objects. The average precision measure is then obtained by averaging these precisions over the total number of relevant objects. If the total number of relevant objects found exceeds the specified maximum results size (2000 in TRECVID), the average precision is averaged over the maximum results size instead. Average precision gives us a single value which makes it easy to compare the retrieval performance of different systems.

## 5.2 Retrieval with segmented images

The PicSOM CBIR system was initially designed to index and retrieve only images, thus using image segments in combination with the entire images was a natural experiment to start with. The idea, as described in Section 2.2.2, is to form the object tree with the entire image as the parent and its segments as children. We also compare this to results with the original PicSOM system, where we considered only the entire images, to see what effect the relevance sharing would have.

### 5.2.1 Experiment setting

We used 59 995 colour photographs from the Corel Gallery 1 000 000 product\*. These were converted to the JPEG format using a tool provided by Corel. The image sizes are  $384 \times 256$  or  $256 \times 384$  pixels. From this set we hand picked six sets of ground truth images. The sets and their verbal criteria are as follows:

- **faces**, 1115 images (*a priori* probability 1.85%), where the main target of the image has to be a human head which has both eyes visible and the head has to fill at least 1/9 of the image area.
- **cars**, 864 images (1.44%), where the main target of the image has to be a car, and at least one side of the car has to be completely shown in the image and it has to fill at least 1/9 of the image area.
- **planes**, 292 images (0.49%), where all airplane images have been accepted.
- **sunsets**, 663 images (1.11%), where the image has to contain a sunset with the sun clearly visible in the image.
- **houses**, 526 images (0.88%), where the main target of the image has to be a single house, not severely obstructed, and it has to fill at least 1/16 of the image area.
- **horses**, 486 images (0.81%), where the main target of the image has to be one or more horses, shown completely in the image.

We segmented all the images in the Corel database with the *k*-means algorithm (see Section 4.1.1) followed by region merging (Section 4.1.2). After that we extracted MPEG-7 still image features from both the segments and the entire images. As

---

\*<http://www.corel.com/>

colour descriptors we used Colour layout, Scalable colour and Dominant colour, as a shape descriptor Region-based shape, and as a texture descriptor Edge histogram.

For the images and segments we trained TS-SOMs with layer sizes  $4 \times 4$ ,  $16 \times 16$ ,  $64 \times 64$  and  $256 \times 256$  units for each feature. Every object was used 100 times to train each layer. The experiments were run on each of the ground truth classes in three different ways: 1) using images only, 2) using segments only, and 3) using both images and segments combined.

The first image shown to the system was selected from the correct class. Using the preselected ground truth classes we could automatically determine the relevance or non-relevance of each image. We performed  $N_Q = 50$  relevance feedback rounds with  $N_R = 20$  images shown at each iteration. The experiment was repeated so that each ground truth image was used once as the initialiser and the results were then averaged over all  $N_C$  experiments.

### 5.2.2 Results

Fig. 5.1 shows the recall–relative precision graphs for the six ground truth classes. The relative precision initially increases, but after the recall has reached a certain value, usually around 0.1 or 0.2, it starts to decrease. The only class where this is not true is “houses”. Its precision is much lower than in the other classes, and therefore the recall remains much lower as well. So we never reach the “turning point” where the curve starts to decrease.

Using only segments does invariably worse than using only images, the latter representing the performance of the old PicSOM algorithm. Results gained from some initial experiments using fewer features suggests that this difference is smaller when using a smaller set of features. Increasing the number of good features usually increases the precision as well, and then the segment–image differences in the curves grow larger.

However, using segments and images combined is always clearly better than using either one of them separately. This also shows that although the segments by themselves do not produce very good results, they do contain significant and distinct information that, when combined with images, improves the query performance as a whole.

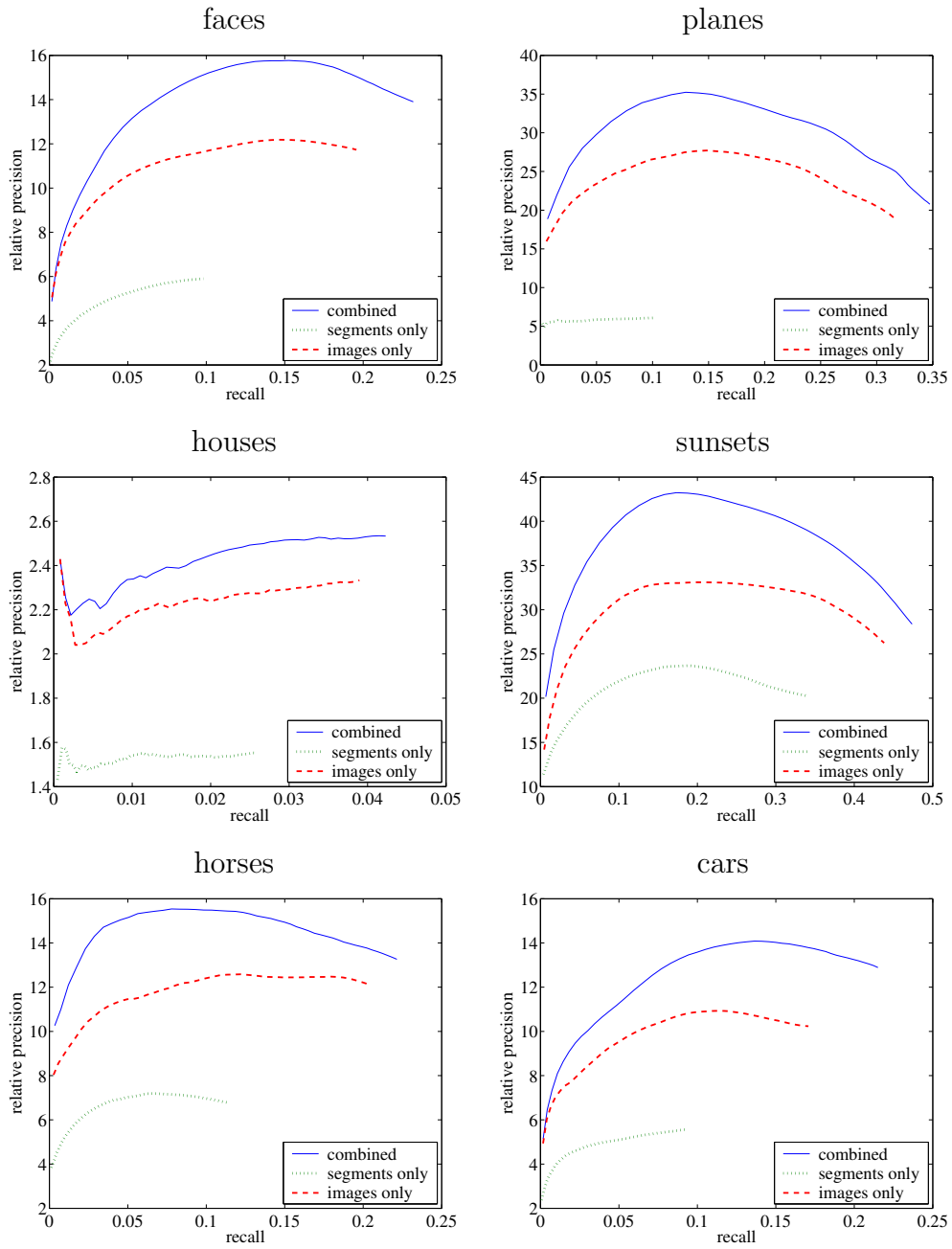


Figure 5.1: Recall–relative precision graphs of the segmented images experiment.

## 5.3 Retrieval with web-link structures

Web pages are basically text files with formatting and linking commands written in HyperText Markup Language (HTML)\*. The layout and graphics that one sees in a web browser are rendered according to these commands. Images are stored separately on the web server, but can be embedded in a web page by using a special HTML command that refers to them. Some web pages may have intricate programmed functionality, for example written in Perl, Java or PHP, but these are executed on the web server and the end result is still always plain HTML. One exception to this is JavaScript which is executed on the client-side, in the web browser, but this has no significance to our discussion.

### 5.3.1 Database collection

We created a database of web pages by collecting all the pages contained in our laboratory Internet site on the [www.cis.hut.fi](http://www.cis.hut.fi) server. This resulted in a database of a little over 7000 HTML pages and almost 2900 images. For the gathering of web pages we used the `ImgRobot` programme, originally created by Olli-Pekka Rinta-Koski (Rinta-Koski 1999), and later modified by Sami Laakso (Laakso 2000) and myself. `ImgRobot` is a C++ programme that collects web pages starting from a given initial page. It traverses the hyperlink structure by following the links of the web pages until no new pages can be found within the specified domain.

The HTML files, images and other embedded objects are downloaded by the `ImgRobot` programme and stored in their native formats in the `PicSOM` database. The embedded objects are stored as separate sub-objects of the web page object in the database (see Section 2.2.4). The links in the web page are collected and stored in a special “links” object.

From the images we extracted MPEG-7 still image features. As colour descriptors we used Colour layout and Scalable colour, as shape descriptor Region-based shape, and as texture descriptor Edge histogram. For the web links we calculated the web-link feature, described in Section 4.3. For the text contents of the HTML files we calculated the  $n$ -gram feature described in Section 4.4.1 with  $n = 3$ .

---

\*<http://www.w3.org>

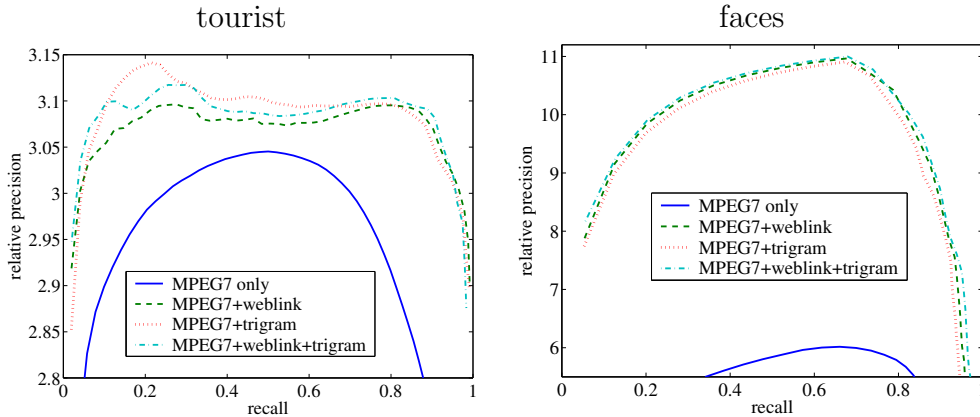


Figure 5.2: Recall–relative precision graphs for web-link experiment.

### 5.3.2 Experiment setting

Two ground truth classes containing images as the target type were selected manually:

- **tourist**, 907 images (*a priori* 31%), from a conference or vacation and mainly outdoor tourist-type photography with attractions like monuments and buildings.
- **faces**, 253 images (8.6%), such that the main target was a human head.

We trained a SOM for each feature type, four from the MPEG-7 features of the images, one from the web-link feature of the links, and one from the trigram feature of the text contents. The sizes of the SOM layers were  $4 \times 4$ ,  $16 \times 16$ ,  $64 \times 64$  and  $256 \times 256$  units. Every object was used 100 times to train each SOM layer.

The experiments were run in four ways: 1) using only the MPEG-7 image features, and then the MPEG-7 features combined with 2) the web-link feature, 3) the trigram feature, and 4) both non-image features. Each query was initialised by showing one image of the correct class. After this  $N_Q = 50$  query rounds were performed with  $N_R = 20$  example images at each round. The experiment was repeated so that each ground truth image was used as the initialiser once, and the results were averaged.

### 5.3.3 Results

In Fig. 5.2 one can see the recall–relative precision graphs for the two ground truth classes. In all plots the precision initially increases and then begins to decline when

a clear majority of the relevant images have been found. The additional non-image features can be seen to increase the precision of the retrieval in all three combinations. In the combined cases the recall level where the precision starts to decline is also substantially larger. Using non-visual features seems to bring the final recall to unity, or at least very close to it. Thus almost all relevant images have been found when only about one third of the images have been retrieved.

The overall precision in the faces class is better. This is probably because this class contains mostly a very narrow class of mug-shot-type photos that probably have very similar features. The tourist class is a much wider class including tourist attractions, monuments and cityscapes.

The differences between the different non-image feature combinations are relatively small. In the faces class, the combination of the web-link and trigram feature seems to be the best overall, although by a very small margin. In the tourist class the picture is more varied, trigram reaches the highest relative precision value, peaking around a recall of 0.2. Otherwise differences are minimal.

## **5.4 TRECVID 2005 automatic search**

The PicSOM group participated for the first time in the NIST TRECVID video retrieval evaluation experiments for the TRECVID 2005 workshop (Koskela et al. 2005). Our main objective was to implement all the necessary functionality into PicSOM to be able to use the provided TRECVID data and return results in the required XML format. We also wanted to combine multimodal features with a text query and both positive and negative class models. The parallel TS-SOMs of the PicSOM system were augmented with inverted file indices created from automatic speech recognition and machine translation (ASR/TR) data provided by NIST.

The TRECVID evaluations consisted of many different tasks, of which only the fully automatic search tasks will be described here. The automatic search tasks were run with the PicSOM system by using predefined search topics and no user interaction. The result of each run was a list of 2000 video clips ordered by their deemed relevance to the search topic. To our delight, the PicSOM results compared very well with the other systems taking part in the TRECVID 2005 evaluation.

### **5.4.1 Video multi-part structure**

The videos in the supplied TRECVID 2005 database were several hour long segments of news broadcasts in three languages: Arabic, Chinese (Mandarin) and English.

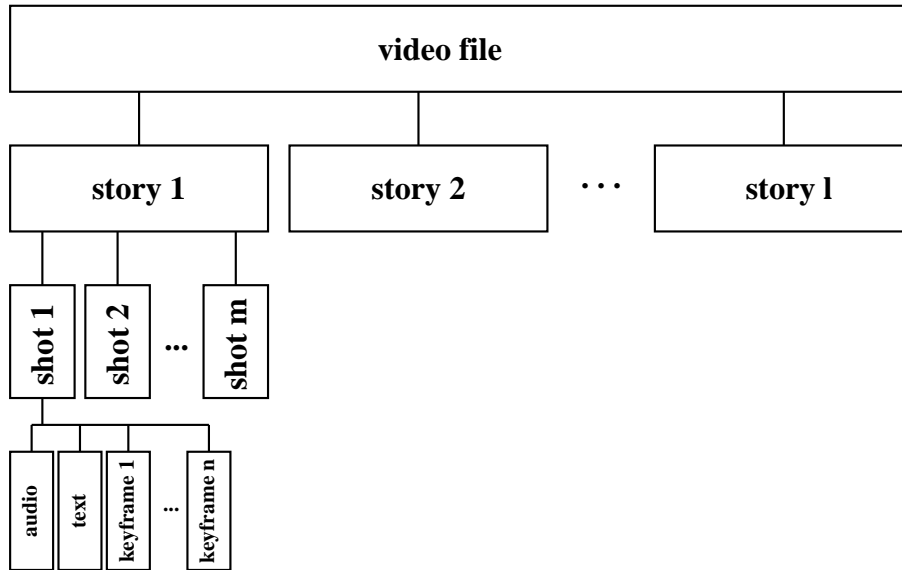


Figure 5.3: The hierarchical tree generated from the TRECVID 2005 videos.

The long videos were divided into “stories”, generally 1-2 minutes long, containing one news story or some other appropriate segment. These were held as sub-objects of the entire videos in a hierarchical object tree. The stories were further segmented into short “shots” of a few seconds each, containing one internally homogeneous scene. For example, an instantaneous change of the camera angle would usually indicate the beginning of a new shot, while a slow panning of the camera might be contained in the same shot. The shots belong to a new layer of sub-objects as children to the stories. From the individual shots, audio clips and key frame images were extracted as sub-objects. Finally textual content, created by automatic speech recognition and machine translation from the non-English languages was also added. The speech recognition and translation (ASR/MT) outputs were generated by off-the-shelf products and provided by NIST. This entire hierarchical structure is depicted in Fig. 5.3.

The predefined search topics were given with a textual instruction, a set of example images and a set of example videos. These were composed as a hierarchical object as shown in Fig. 5.4, and could thus easily be inserted in the system. Appropriate features were then calculated from the given objects.

#### 5.4.2 Semantic class models

Some semantic classifications of the video clips in the training set were provided by NIST, for example a list of clips showing an explosion or fire. A very informative

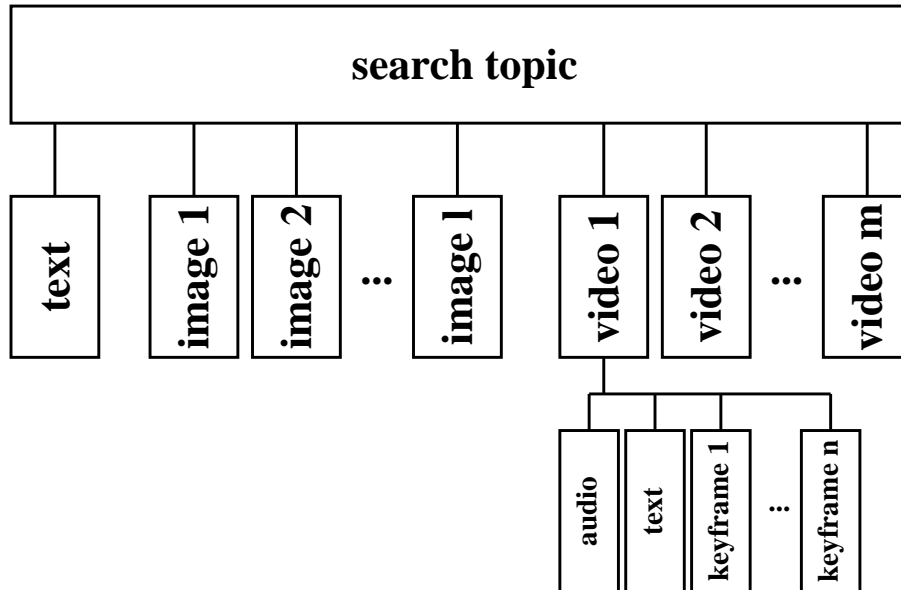


Figure 5.4: The hierarchical tree of a search topic used with the TRECVID 2005 database.

visualisation can be gained by mapping the feature vectors of the objects belonging to a specific semantic class as impulses on the SOM surfaces. This gives insight into how well a certain feature can cluster the vectors associated with that semantic concept. When used in the retrieval, the sign of the impulses can be adjusted to represent relevant (positive) and non-relevant (negative) concepts. The sparse value fields on the maps are low-pass filtered as usual to spread the information. This also helps visual inspection as the marked areas become larger and more uniform. An example of such a mapping of the concept *explosion/fire* on the MPEG-7 Colour layout SOM can be found in Figure 5.5. Areas corresponding to objects of the concept are shown in shades of grey. As can be seen, the objects cluster quite well on the map into nearby locations.

Theoretically the class distributions must be considered as estimates of the true distributions as they are finite and mapped on to the discrete two-dimensional grids of the SOMs, while the original feature space usually has a much higher dimensionality. The class model idea was initially used in PicSOM for comparing different features (Laaksonen et al. 2003) and for image group annotation in (Koskela & Laaksonen 2005).

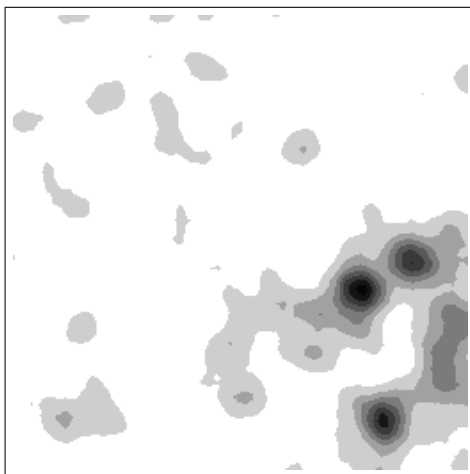


Figure 5.5: The *explosion/fire* class model mapped on the MPEG-7 Colour layout SOM, adapted from Koskela et al. (2005).

### 5.4.3 Text query processing

The ASR/MT output of non-English videos included additional information, such as if a certain proper name was a person, location or organisation. Of these we used the person and location information to create an index of “known” proper names and whether they referred to persons or locations. Furthermore, discriminative words were picked up from the ontology descriptions (provided by NIST) to create a word–concept index. For example the word “minister” would map to the semantic class *government\_leader*. This information was used in processing the text queries in the automatic search experiments before being used in the retrieval.

Proper names were initially identified in the text query by recognising single or consecutive words with a capitalised first letter. These proper names were then compared with the index of known proper names by using the Levenshtein distance (Black 1999). If the index name with the shortest distance was sufficiently close to the query name then the query name was deemed to be a misspelled version of the index name. The tolerance was dependent on the length of the query name, so that for short names a shorter Levenshtein distance was needed for acceptance. The identified misspelled words were corrected and the query string was cleaned, i.e. lowercased, dots and commas removed, and unnecessary text such as the preceding “Find shots of” discarded.

Additionally, the word–concept index was used to identify words that might indicate useful class models. The presence of negative words, like a preceding “not” negated the class model. Finally if a person’s name was identified previously, the class models

*face* and *person* were added automatically.

Table 5.1 shows the transformations that would be made to this example query string: “Find shots of Omar Karammi, the former prime minister of Lebannon” (spelling errors intentional). The first row in the table shows the original string, and the second row the identifications found by the system. “Omar Karammi” is identified as a person and “Lebannon” as a location (even though they are misspelled). The identification WORD-CONCEPT under the word “minister” signifies that the word has been found in the word–concept index. The third row shows the actions or transformations performed. The initial “Find shots of” is deleted and the misspelled names are corrected. The corrections are marked by CORR with the corrected version in parenthesis. The fourth row shows the class models added, the sign before the class name identifies a positive or negative class model. The identified person automatically adds the class models *face* and *person* and “minister” adds *government\_leader*. The last row shows the final processed text, capital letters, dots and commas removed, that was processed with the inverse file.

<b>original</b>	<i>Find shots of Omar Karammi, the former prime minister of Lebannon</i>		
<b>identification</b>	PERSON	WORD-CONCEPT	LOCATION
<b>actions</b>	DELETE	CORR(Omar Karami)	CORR(Lebanon)
<b>classes</b>	+face, +person		+government_leader
<b>processed</b>	<i>omar karami the former prime minister of lebanon</i>		

Table 5.1: An example of text query processing in automatic search.

#### 5.4.4 Experiment setting

The experiments were run automatically as batch runs with 24 different search topics, ranging from finding a specific person to specific situations or events (e.g. an object burning or exploding). In Fig. 5.6 an example query taken from the actual TRECVID 2005 search topics is shown. The entire text query was “Find shots of Mahmoud Abbas, also known as Abu Mazen, prime minister of the Palestinian Authority”. Two example images were given and nine video segments, of which only two are represented in the figure by their key frames.

In total the video shots were indexed using four video features (the MPEG-7 Motion activity and temporal versions of Average colour, Colour moments and Texture neighbourhood), three still image features from MPEG-7 (Colour layout, Edge histogram, and Homogeneous texture) and one audio feature (Mel cepstrum). These features were all explained in Chapter 4. The sizes of the SOM layers were  $4 \times 4$ ,  $16 \times 16$ ,  $64 \times 64$  and  $256 \times 256$  units. Only one query round was performed returning 2000 video objects.

## 5.4.5 Results

The average precision for each search topic, numbered from 149 to 172, is shown in Fig. 5.7 compared to the median and best results. For most topics the PicSOM system performs on or above the median. In only one case the result remains clearly under the median, whereas in seven cases it is clearly above it. By calculating the mean average precision over all topics we get an overall score for the retrieval accuracy. If we take only the best result from each research group into account, the PicSOM system ranked third out of 9 groups. That is a very good achievement taking into account that this was our first year in TRECVID and our main goal was just to make our system compatible with the evaluation interfaces.

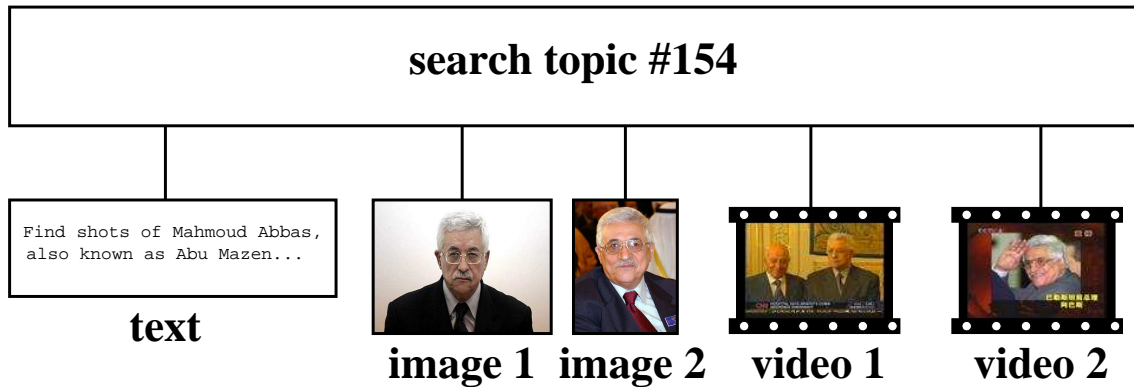


Figure 5.6: An example search topic from TRECVID 2005.

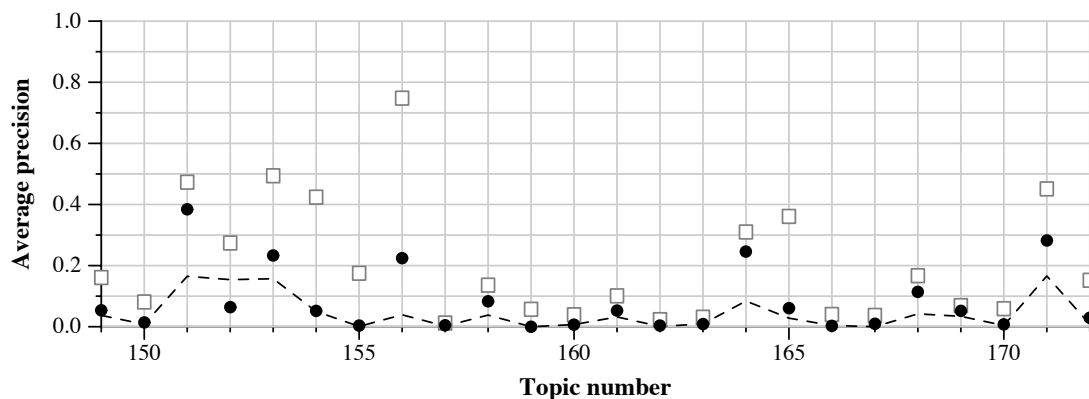


Figure 5.7: The average precision for each search topic. The score for PicSOM is shown (dot) versus the median (dotted line) and the best result (box).

# Chapter 6

## Conclusions and future prospects

### 6.1 Conclusions

This thesis started by discussing the increasing amounts of data that we produce and have to understand and use efficiently in the modern world. The goal of the research discussed here is to reduce this continuous stream of information into something that is manageable by humans by using content-based information retrieval methods. The purpose is to create a semiautomatic search tool to aid humans in finding relevant information quickly. This problem will probably be one of the defining questions in information science of the 21st century. Although this problem was in no way solved in this thesis, I would like to think that we have taken a step forward by slightly improving existing CBIR methods.

The novel idea in this thesis was to take advantage of known relationships between different data objects in content-based information retrieval. This was done by using hierarchical data objects where each object can have parents and children. In the relevance feedback process these hierarchies were used for relevance sharing, so that objects deemed relevant in a query would influence related objects as well.

Many examples of the application of this idea were given by using the PicSOM CBIR system: images with their segments, videos with image frames and audio, web-link structures and multi-part messages. In the case of segmented images and linked web pages we presented experimentation results showing the clear advantage of using hierarchical objects over using just plain objects. A working multi-part message database has been created (Muurinen 2003), but we do not yet have proper testing data to be able to produce any quantitative results. Video and audio experiments have been carried out in the framework of the TRECVID 2005 evaluations producing results that compared well with other participating retrieval systems.

The bulk part of the underlying and “hidden” work of this thesis was in implementing new extensions to the existing systems: the new relevance feedback method for hierarchical objects, integrating the MPEG-7 features into the PicSOM framework, implementing the storing of hierarchical objects, modifying the ImgRobot web traversing tool, and diverse minor tasks.

## 6.2 Future prospects

As CBIR is both computationally heavy and also very user-centric in its nature, there are two areas where improvements can be made. First, the internal processing, such as object similarity and the usage of relevance feedback. Second, the external user interface for interactive query and browsing. This thesis has mostly concentrated on the first area, but it is recognised that improvements in the latter can also have a large impact. What is the use of fancy indexing techniques if the end user cannot benefit from them?

Increased control and ability to steer the query process in an intuitive manner would be obvious improvements to the user interface. For example being able to select interesting image segments or other areas of interest in the user interface of PicSOM could be useful. The relevance given to the image could then be focused to selected segments and not given equally to all segments as currently is the case. A problem with this, though, is that the automatic segmentation algorithms seldom produce results that correspond to the human understanding of the image. This situation could be improved by using human driven semi-automatic segmentation, or even combining it with the focusing approach. The user could then somehow help the system along by specifying different points, or interesting areas in an image. For example, by picking out a car in an image by drawing its borders, the system would try to find similar segments with similar properties in other images.

Internally many improvements could still be made to the PicSOM system. More and better feature extraction methods should be implemented and experimented with. For many features the Euclidean distance metric is not always optimal. If this could be resolved some way, it might improve retrieval results. Alternative indexing techniques should also be tried out.

Although our method is taking advantage of relations between the database objects, there are often other higher level relationships that are not used. For example in the case of multi-part messages like e-mails, there often exist long chains of replies. The messages and their replies are closely linked because they discuss the same

subjects, but this information is currently not explicitly used. Another example is the web-link structure. Although our current implementation uses links objects that incorporate the links and domain and directory levels of URLs, the web of links is not used explicitly. Therefore pages that are not directly linked or share similar links do not benefit from this information.

Another interesting viewpoint is the semantic web paradigm (Berners-Lee et al. 2001), where semantic information is explicitly embedded into web documents. The hierarchical object structures used in PicSOM incorporate certain forms of semantic knowledge in an automated way, which can be seen to complement the semantic web idea. This could reduce the manual annotation work normally required when creating the semantic web. On the other hand, future developments in our system could utilise semantic web information as an additional feature in the hierarchical structure.

Further experimenting with different types of media is also an interesting topic. Video and multi-media messages seem to be especially promising areas. Combining content-based features with textual annotations would also be interesting. In some cases, annotations such as keywords could be available only for a subset of the database objects. The CBIR system should then be able to take advantage of this information as well.

The above ideas are only the tip of the iceberg when it comes to possible future improvements. So there are a lot of things that can be done in the coming years. In addition to the mentioned incremental improvements to existing methods, also more fundamental research must be made if we are ever to bridge the semantic gap. This is perhaps our ultimate, possibly unachievable, goal in this research area.

# Bibliography

- Bellman, R. (1961). *Adaptive control processes: a guided tour*, Princeton University Press, Princeton, NJ, USA.
- Berners-Lee, T., Hendler, J. & Lassila, O. (2001). The semantic web, *Scientific American* **284**(5): 28–37.
- Black, P. E. (1999). Algorithms and theory of computation handbook, *NIST Dictionary of Algorithms and Data Structures*, CRC Press LLC. <http://www.nist.gov/dads/HTML/Levenshtein.html>.
- Brandt, S. (1999). *Use of shape features in content-based image retrieval*, Master's thesis, Laboratory of Computer and Information Science, Helsinki University of Technology.
- Carson, C., Belongie, S., Greenspan, H. & Malik, J. (2002). Blobworld: Image segmentation using expectation-maximization and its application to image querying, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(8): 1026–1038.
- Cascia, M. L., Sethi, S. & Sclaroff, S. (1998). Combining textual and visual cues for content-based image retrieval on the world wide web, *IEEE Workshop on Content-Based Access of Image and Video Libraries*, Santa Barbara, CA, USA, pp. 24–28.
- Castro, P. & Muntz, R. (1999). Using context to assist in multimedia object retrieval, *ACM Workshop on Multimedia Intelligent Storage and Retrieval Management*, Orlando, FL, USA.
- Chang, N.-S. & Fu, K.-S. (1980). Query-by-Pictorial-Example, *IEEE Transactions on Software Engineering* **6**(6): 519–524.
- Chen, J.-Y., Bouman, C. A. & Allebach, J. P. (1997). Fast image database search using tree-structured VQ, *Proceedings of IEEE International Conference on Image Processing (ICIP '97)*, Vol. 2, Santa Barbara, CA, USA, pp. 827–830.

- Chen, Y. & Wang, J. Z. (2001). Looking beyond region boundaries: Region-based image retrieval using fuzzy feature matching, *Multimedia Content-Based Indexing and Retrieval Workshop, September 24-25*, INRIA Rocquencourt, France.
- Cox, I. J., Miller, M. L., Minka, T. P., Papathomas, T. V. & Yianilos, P. N. (2000). The Bayesian image retrieval system, PicHunter: Theory, implementation and psychophysical experiments, *IEEE Transactions on Image Processing* **9**(1): 20–37.
- Date, C. (2003). *Introduction to Database Systems*, 8 edn, Addison Wesley, New York.
- Davis, S. B. & Mermelstein, P. (1990). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences, in A. Waibel & K. Lee (eds), *Readings in speech recognition*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 65–74.
- Dimai, A. (1999). Invariant scene description based on salient regions for preattentive similarity assessment, *10th International Conference on Image Analysis and Processing (ICIAP), September 27-29*, Venice, Italy, pp. 957–962.
- Eakins, J. P. (2002). Towards intelligent image retrieval, *Pattern Recognition* **35**(1): 3–14.
- FIPS (1995). Secure hash standard, *Federal Information Processing Standards Publication 180-1*, NIST. <http://www.itl.nist.gov/fipspubs/fip180-1.htm>.
- Gonzales, R. C. & Woods, R. E. (1992). *Digital Image Processing*, Addison-Wesley, New York.
- Gupta, A. & Jain, R. (1997). Visual information retrieval, *Communications of the ACM* **40**(5): 70–79.
- Honkela, T., Kaski, S., Lagus, K. & Kohonen, T. (1997). WEBSOM—self-organizing maps of document collections, *Proceedings of WSOM'97, Workshop on Self-Organizing Maps, Espoo, Finland, June 4-6*, Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland, pp. 310–315.
- Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components, *Journal of Educational Psychology* **24**: 417–441.
- Kaski, S. (1998). Dimensionality reduction by random mapping: Fast similarity method for clustering, *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN98)*, Vol. 1, Anchorage, AK, USA, pp. 413–418.

- Kherfi, M. & Ziou, D. (2004). Image retrieval from the world wide web: Issues, techniques and systems, *ACM Computing Surveys* **36**(1): 35–67.
- Kohonen, T. (2001). *Self-Organizing Maps*, Vol. 30 of *Springer Series in Information Sciences*, third edn, Springer-Verlag, Berlin.
- Koikkalainen, P. (1994). Progress with the tree-structured self-organizing map, in A. G. Cohn (ed.), *11th European Conference on Artificial Intelligence*, European Committee for Artificial Intelligence (ECCAI), John Wiley & Sons, Ltd., Amsterdam, The Netherlands, pp. 211–215.
- Koikkalainen, P. & Oja, E. (1990). Self-organizing hierarchical feature maps, *Proceedings of International Joint Conference on Neural Networks*, Vol. II, San Diego, CA, USA, pp. 279–284.
- Koskela, M. (1999). *Content-based image retrieval with self-organizing maps*, Master’s thesis, Laboratory of Computer and Information Science, Helsinki University of Technology.
- Koskela, M. (2003). *Interactive Image Retrieval using Self-Organizing Maps*, PhD thesis, Laboratory of Computer and Information Science, Helsinki University of Technology. Available online at: <http://lib.hut.fi/Diss/2003/isbn9512267659/>.
- Koskela, M. & Laaksonen, J. (2005). Semantic annotation of image groups with Self-Organizing Maps, *Proceedings of 4th International Conference on Image and Video Retrieval (CIVR 2005)*, Singapore, pp. 518–527.
- Koskela, M., Laaksonen, J. & Oja, E. (2003). Inter-query relevance learning in PicSOM for content-based image retrieval, *Supplementary Proceedings of 13th International Conference on Artificial Neural Networks / 10th International Conference on Neural Information Processing (ICANN/ICONIP 2003)*, Istanbul, Turkey, pp. 520–523.
- Koskela, M., Laaksonen, J. & Oja, E. (2004). Use of image subset features in image retrieval with self-organizing maps, *Proceedings of 3rd International Conference on Image and Video Retrieval (CIVR 2004)*, Dublin, Ireland, pp. 508–516.
- Koskela, M., Laaksonen, J., Sjöberg, M. & Muurinen, H. (2005). PicSOM experiments in TRECVID 2005, *Proceedings of the TRECVID 2005 Workshop*, Gaithersburg, MD, USA, pp. 262–270.

- Laakso, S. (2000). *Implementation of content-based www image search engine*, Master's thesis, Laboratory of Computer and Information Science, Helsinki University of Technology.
- Laakso, S., Laaksonen, J., Koskela, M. & Oja, E. (2001). Self-organizing maps of web link information, *in* N. Allinson, H. Yin, L. Allinson & J. Slack (eds), *Advances in Self-Organising Maps*, Springer, Lincoln, England, pp. 146–151.
- Laaksonen, J., Koskela, M., Laakso, S. & Oja, E. (2001). Self-organizing maps as a relevance feedback technique in content-based image retrieval, *Pattern Analysis & Applications* **4**(2+3): 140–152.
- Laaksonen, J., Koskela, M. & Oja, E. (2002). PicSOM—Self-organizing image retrieval with MPEG-7 content descriptions, *IEEE Transactions on Neural Networks, Special Issue on Intelligent Multimedia Processing* **13**(4): 841–853.
- Laaksonen, J., Koskela, M. & Oja, E. (2003). Probability interpretation of distributions on SOM surfaces, *Proceedings of Workshop on Self-Organizing Maps (WSOM'03)*, Hibikino, Kitakyushu, Japan, pp. 77–82.
- Li, D., Dimitrova, N., Li, M. & Sethi, I. K. (2003). Multimedia content processing through cross-modal association, *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, ACM Press, New York, NY, USA, pp. 604–611.
- Lyman, P. & Varian, H. R. (2003). How Much Information, <http://www.sims.berkeley.edu/how-much-info-2003/>.
- Ma, W. Y. & Manjunath, B. S. (1999). NeTra: a toolbox for navigating large image databases, *Multimedia Systems* **7**(3): 184–198.
- Miller, R. (1968). Response time in man-computer conversational transactions, *AFIPS Spring Joint Computer Conference*, Atlantic City, NJ, USA, pp. 267–277.
- MPEG (2001). MPEG-7 visual part of the eXperimentation Model (version 9.0). ISO/IEC JTC1/SC29/WG11 N3914.
- MPEG (2002). MPEG-7 Overview (version 8.0). ISO/IEC JTC1/SC29/WG11 N4980.
- Mukherjea, S., Hirata, K. & Hara, Y. (1999). Amore: A world wide web image retrieval engine, *World Wide Web* **2**(3): 115–132.

- Muurinen, H. (2003). *Implementing Support for Content-Based Multimedia Message Retrieval in the PicSOM System*. Special assignment, Laboratory of Computer and Information Science, Helsinki University of Technology.
- Nielsen, J. (1999). User interface directions for the web, *Communications of the ACM* **42**(1): 65–72.
- Ortega-Binderberger, M., Mehrotra, S., Chakrabarti, K. & Porkaew, K. (2000). WebMARS: A multimedia search engine, *Proceedings of the SPIE Electronic Imaging 2000: Internet Imaging*, San Jose, CA, pp. 314–321.
- Pakkanen, J. (2002). *Sisältöpohjainen haku paperivirhetietokannassa PicSOM-järjestelmän avulla (in Finnish)*, Master's thesis, Laboratory of Computer and Information Science, Helsinki University of Technology.
- Picard, R. W. (1996). A society of models for video and image libraries, *IBM Systems Journal* **35**(3/4): 292–312.
- Porter, M. (1980). An algorithm for suffix stripping, *Program* **14**(3): 130–137.
- Rautkorpi, R. (2005). *Shape features in the classification and retrieval of surface defect images*, Master's thesis, Laboratory of Computer and Information Science, Helsinki University of Technology.
- Rehg, J. M., Murphy, K. P. & Fieguth, P. W. (1999). Vision-based speaker detection using bayesian networks, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99)*, Vol. 2, pp. 110–116.
- Rinta-Koski, O.-P. (1999). WWW-kuvahakurobotti (in Finnish). Special assignment, Laboratory of Computer and Information Science, Helsinki University of Technology.
- Rui, Y., Huang, T. S. & Chang, S.-F. (1999). Image retrieval: Current techniques, promising directions, and open issues, *Journal of Visual Communication and Image Representation* **10**(1): 39–62.
- Rui, Y., Huang, T. S., , M. O. & Mehrotra, S. (1998). Relevance feedback: A power tool in interactive content-based image retrieval, *IEEE Transactions on Circuits and Systems for Video Technology* **8**(5): 644–655.
- Rummukainen, M. (2003). *Implementing multimedia retrieval markup language for image retrieval systems' comparison*, Master's thesis, Laboratory of Computer and Information Science, Helsinki University of Technology.

- Salton, G. & McGill, M. J. (1983). *Introduction to Modern Information Retrieval*, Computer Science Series, McGraw-Hill, New York.
- Sjöberg, M., Laaksonen, J. & Viitaniemi, V. (2003). Using image segments in PicSOM CBIR system, *Proceedings of 13th Scandinavian Conference on Image Analysis (SCIA 2003)*, Halmstad, Sweden, pp. 1106–1113.
- Smith, J. R. & Chang, S.-F. (1996). VisualSEEk: A fully automated content-based image query system, *Proceedings of the 4th International ACM Multimedia Conference (ACM MM '96)*, Boston, MA, USA, pp. 87–98.
- Sonka, M., Hlavac, V. & Boyle, R. (1998). *Image Processing, Analysis and Machine Vision*, 2nd edn, Brooks/Cole Publishing Company, Pacific Grove, CA, USA.
- Squire, D., Müller, H. & Müller, W. (1999). Improving response time by search pruning in a content-based image retrieval system, using inverted file techniques, *Proceedings of IEEE International Workshop on Content-Based Access of Image and Video Libraries (CBAIVL '99)*, Fort Collins, CO, USA, pp. 45–49.
- TASI (2004). A review of image search engines. <http://www.tasi.ac.uk/resources/searchengines.html>.
- Theodoridis, S. & Koutroumbas, K. (1999). *Pattern Recognition*, Academic press, San Diego, CA, USA.
- Tversky, A. (1977). Features of similarity, *Psychological Review* **84**(4): 327–352.
- Viitaniemi, V. (2002). *Image segmentation in content-based image retrieval*, Master's thesis, Laboratory of Computer and Information Science, Helsinki University of Technology.
- Wactlar, H. D., Kanade, T., Smith, M. A. & Stevens, S. M. (1996). Intelligent access to digital video: Informedia project, *IEEE Computer* **29**(5): 46–52.
- Wang, J. Z., Li, J. & Wiederhold, G. (2001). SIMPLiCity: Semantics-sensitive integrated matching for picture libraries, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23**(9): 947–963.