Scalable Methods for Content-Based Image Retrieval

GUSTAV BROBERG

KTH Computer Science and Communication

Master of Science Thesis
Stockholm, Sweden 2011
Scalable Methods for Content-Based Image Retrieval

GUSTAV BROBERG

Master’s Thesis in Computer Science (30 ECTS credits)
at the School of Computer Science and Engineering
Royal Institute of Technology year 2011
Supervisor at CSC was Stefan Carlsson
Examiner was Stefan Carlsson

TRITA-CSC-E 2011:066
ISRN-KTH/CSC/E--11/066--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication
KTH CSC
SE-100 44 Stockholm, Sweden
URL: www.kth.se/csc
Abstract

The largest and most used image search engines on the Internet today use textual information from either context or annotations to index images. Content-based Image Retrieval (CBIR) denotes the research efforts to instead use information contained in the images’ themselves for indexing. Though major advances in this field have been made in recent years there has been little focus on the problems of scaling CBIR methods to a point where they can be used for Internet image search, i.e. for data sets with billions of images.

In this project, three methods with potential for building scalable content-based similarity image search engines were identified and implemented. The implementations’ retrieval accuracy was tested and their scalability potential evaluated. It was found that few CBIR methods allow satisfactory scalability and the three selected methods’ retrieval performance was fairly poor. The result shows that two simpler methods outperformed a sophisticated spatial method. It was further suggested that one of the simple methods could be used to improve accuracy of text-based image retrieval.
Referat

Metoder för innehållsmässig bildsökning med hög skalbarhet

De största och mest använda bildsökmotorerna på Internet idag använder sig av textsträngar från uppmärkning eller omgivande text för att indexerar bilder. Inom området innehållsmässig bildsökning (CBIR) försöker man istället nyttja bilders egen data för indexering. Trots att stora framsteg med att utveckla korrektheten hos CBIR-metoder gjorts de senaste åren så har förhållandevis lite kraft lagts på att utvärdera dessa metoders skalbarhet vid användning på bilddatabaser av Internets storlek, d.v.s. med flera miljarder bilder.

I det här projektet har tre metoder som skulle kunna vara möjliga som grund för jämförelsebaserade bildsökmotorer med hög skalbarhet identifierat och implementerat. Implementationernas träffsäkerhet testades och deras respektive skalbarhet utvärderades. Det visade sig att få innehållsmässiga bildsökningsmetoder har tillfredsställande skalbarhet och att de tre metoder som utvärderades gav mindre tillfredsställande resultat. Resultaten i den här rapporten visar att två enklare metoder gav högre korrekthet än en sofistikerad, spatiell metod. Vidare föreslogs det att en av de enkla metoderna kan användas för att öka träffsäkerheten hos textbaserade sökmetoder.
# Contents

## Contents

1 Background 1
   1.1 Content-based Image Retrieval .......................... 1
   Query data .................................................. 2
   1.2 Image Similarity .......................................... 3
   1.3 Problem definition ....................................... 3
   Scope ......................................................... 4
   Motivation .................................................... 4
   1.4 Prior and related work ................................... 5

2 Methods 7
   2.1 A CBIR System’s Design ................................... 7
   2.2 Image Features ............................................ 9
       Color .................................................... 9
       Texture and shape ...................................... 11
       Spatial/Non-spatial features ......................... 12
   2.3 Scalability considerations .............................. 12
       Extraction complexity ................................. 13
       Feature vectors ....................................... 13
   2.4 Evaluated methods ....................................... 13
       Criteria .................................................. 13
       Color moments ........................................... 14
       JPEG-based (DCT) ....................................... 15
       Color sets ................................................. 17
   2.5 Evaluation procedure ................................... 20
       Implementation details ................................. 21
       Test data ................................................ 21
       Performance metrics ................................. 22

3 Results 25
   3.1 Performance comparison ................................. 25
       The UCID database ....................................... 25
<table>
<thead>
<tr>
<th>CONTENTS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>The PE for CBIR database</td>
<td>27</td>
</tr>
<tr>
<td>3.2 Example queries</td>
<td>30</td>
</tr>
<tr>
<td>Photographic image, from the UCID database</td>
<td>30</td>
</tr>
<tr>
<td>Drawn image, isolated object, from the PE for CBIR database</td>
<td>31</td>
</tr>
<tr>
<td>3.3 Complexity/scalability comparison</td>
<td>32</td>
</tr>
<tr>
<td>Feature extraction complexity</td>
<td>32</td>
</tr>
<tr>
<td>Feature vectors</td>
<td>32</td>
</tr>
<tr>
<td>4 Conclusions and summary</td>
<td>35</td>
</tr>
<tr>
<td>5 Bibliography</td>
<td>37</td>
</tr>
</tbody>
</table>
Chapter 1

Background

The purpose of this project is to find and evaluate methods for image querying based on the images’ actual content that give acceptable query times even when image databases are of massive scale. To understand the problems that arise it is necessary to give a background on what Content-based Image Retrieval means, its use and its theoretical limitations. This allows a discussion on image similarity, which is the measure on which image querying is based in this project. The distinction between semantic and visual similarity is made and the choice to focus on the latter in this project is motivated. A look at what has been accomplished in the field in regard to the project’s problem is also given.

1.1 Content-based Image Retrieval

*Information retrieval* is the general term for the science of searching for documents in a set of data matching a query. Here “documents” can be anything from journal articles and web pages, to images and videos. The set of data to search in is usually some sort of database that stores metadata about, or data derived from, the set of documents to query.

Traditionally the research focus in information retrieval has been on the domains of textual information retrieval. Indicative of this focus are ambitious projects that have been realized to provide the research community with collections of test data, evaluation methods and performance measures that are now well-establish “tools”. This has given the possibility to compare research results and quantify improvements in the field. [5] Examples of practical results of the advances yield in this field that are widely available to the public include web search engines, libraries’ books catalogs and e-mail spam filtering.

In the 1980’s and 90’s, the advent of large-scale digital image databases called for efficient ways to query them. Image content is however different from text in that cannot be readily analyzed or *parsed* as sentences in a text — it is inherently multi-dimensional and hard to split into parts without losing information. An obvious solution is to let a human *annotate* each image in the target database and thereby
turning it into textual content for which retrieval methods exist. Alternatively, if relevant textual context to the images is available, for example in the form of web pages they are placed in, it might be used for image annotations. Manual annotation could work well for small databases, but it is hardly scalable — labeling of all images in a large-scale database becomes time consuming or even infeasible. In real-life, the relevance of contextual information is often hard to estimate and might not be homogeneous in the data set. Clearly, a way of automatically retrieving information from the images themselves is a worthwhile research goal.

The collective term used for describing problems related to information retrieval from images themselves is Content-based Image Retrieval or CBIR. The problems it denotes have primarily been sorted under the field of computer vision. This is because it fundamentally seeks to answer how vision and visual interpretation can be modeled formally.

Query data

Whereas a string of characters is the “normal” input, or query data, in textual information retrieval, CBIR does not have any given equivalent. Depending on the intended use of the system, different types of data may be suitable as input for a query. Three types of query data that are commonly used in CBIR system and occur in literature are:

Description/keywords

Compact textual descriptors of the image. Often abstract in the sense that they try to describe high-level information sought in an image’s semantic content. Example of descriptors might be “child kicking ball” (description), “tree” (concrete keyword), or “spring” (abstract keyword).

Visual example

One or many images serve as an example of the requested content. The expected hits should thus be considered “similar” to the provided image.

Visual features

A set of visual properties that each queried image must have to be considered a match. The required set of visual properties could be ”must have at least 50% black pixels and a circular area of yellow pixels in the center”. These properties are often expressed by the user by using a graphical drawing interface.

In addition to the query data some CBIR systems allow user input in the form of relevance feedback. The user judges how relevant (or “correct”) the result of each query is, and this assessment is fed back to the system that uses it for learning or tuning itself. [17, 6.4]
1.2 Image Similarity

*Similarity* is an ambiguous term. When we state that two images are similar it could mean everything from that they have just about the same pixels, that they depict something similar, or that they have a resemblance in colors and patterns. It is easy to see that formal definition of *image similarity* can not be made while keeping the general notion of it — its meaning depend on who we ask and the context in which we do it. It is therefore of interest to narrow down the term when analyzing its meaning.

The focus of interest in this project is how humans perceive similarity of images. Somewhat simplified, this perception can be divided into *semantic similarity* and *visual similarity*. [4], [25]

The semantic content of an image here refers to how the subject interprets the image — her understanding of what the image depicts. Hence, semantic similarity for images is the perceived similarity of what the images portray. For example, the subject finds picture A similar to B because both contain an oak tree. More specifically, what we see in an image and how we interpret it is essential for evaluating its semantic similarity to other images.

Visual image similarity, on the other hand, describes how alike images are in regards to how they appear without any semantic interpretation. This means resemblance in layout, colors, patterns, texture; properties of the image content that does not need to be “understood”.

It should be made clear that neither of the two types of similarity described here allows stricter (and more adequate) definitions to be formed. For instance we cannot, without losing meaning, say that image B is more semantic similar to A, than C to A because they depict more objects of the same kind. Likewise, defining visual similarity as the correlation of some visual feature(s), is far too simplified to represent the perception of similarity.

What makes it difficult to formulate their meanings is not only the complexity of our understanding of image similarity, but the subjectivity of it. [10] What can and is read from an image depends on the individual. Furthermore it has been shown that in practice it is difficult to consciously separate semantics from visual attributes when estimating similarity. [13]

1.3 Problem definition

The aim of this project is to research the potential for using Content-based Image Retrieval methods for querying large-scale image databases. More specifically, the project seeks to identify image features that serve as accurate, yet low-dimensional/compact, descriptors. In extension it should find methods that have general good retrieval performance that are well suited for scaling. That means that they must be efficient not only in terms of query time but also extraction complexity and storage demands.
A key part of the project is to provide proof-of-concept level implementations of the identified methods that make an actual retrieval performance comparison possible. Furthermore this should allow the derived data presented in this report to be reproduced. Based on results and conclusions drawn from it, directions for further research should be given in the scope of this project.

**Scope**

This project is limited to the problem of image retrieval by visual example; the input data for a query is an image. The retrieval should thus, given an image set and an example image, identify the to the example image most similar images in the set.

Although perceived image similarity depend on semantic similarity, this project only encompasses methods based on visual similarity. Due to what is often referred to as the *semantic gap* [17], extraction of semantic information from an image is considered a non-trivial, fundamental problem in computer vision. CBIR methods extracting and using semantic information are generally domain-specific and complex; hence, evaluations of such methods lie outside the scope of this project.

It is important to note that what is sought here are general-purpose methods where the image data set could contain a wide range of images. If the target data sets have common, known characteristics, domain-specific methods will most certainly be better suited both in terms of accuracy and efficiency.

The implementations of the selected methods should be functioning well enough to serve an accurate picture of retrieval performance of the methods. They are however not required to be optimal or even efficient implementations in terms of computational complexity and memory use. The scalability comparison will instead be based on theoretical scalability limits of each method.

Outside the scope of this project is an in-depth discussion on techniques for indexing descriptors and dimension reduction. Mechanisms for user feedback to improve the methods’ accuracy are not researched in this project; they are interesting but it would require too much work to extend the implementations to support this.

**Motivation**

With the rapid growth of the Internet the number of publicly available digital images has exploded in the last years. Internet sites such as Facebook and Flickr allow end-users to upload images and now store billions of images. The value of effective search methods for this visual content should be obvious. Whereas text search engines have improved in accuracy by using the actual text content and linguistic knowledge, the corresponding existing image search engines are based not on actual image content but on textual context or human-provided annotations. But textual context is generally too inexact and manual annotations are subjective and the approach is often infeasible considering the amount of labor it demands.
1.4. PRIOR AND RELATED WORK

Dramatic improvements in image search performance can only take place if data derived from content itself are indexed. To be successful, the methods for CBIR must be both accurate and efficient. If they are inefficient, their scalability is poor, and it is unrealistic to use them as methods for large data set; this is where automated methods are most valuable. Even if truly scalable methods are too primitive to provide acceptable accuracy on their own, they may be used to improve text-based searching.

1.4 Prior and related work

The field of CBIR has grown considerably since the term was first used in 1992. [6] A somewhat unscientific study by R. Datta et al. [2, 1.] gives an indication of this growth. They searched Google Scholar for papers using the phrase “Image Retrieval” published in 1995–2004 and found an immense growth in the number of publications since 2000.

CBIR research in its early years has been mostly focused on theoretical basics — what visual features to use, how to represent them and the choice of distance metrics. In later years the semantic gap has been noted and a discussion of how it may be “bridged” is now a common starting point in papers.

Though the need for highly scalable CBIR systems is often acknowledged in research, surprisingly few feature descriptors and distance measures suggested in papers are evaluated for their scalability potential. A reason might be that it is sometimes seen as a practical problem in system design rather than a fundamental problem when designing feature descriptors. The relatively small part of research occupied with scalability problems is apparent in a number surveys summarizing CBIR research papers. [17, 7.1], [14, 3.], [2, 3.] The problem of making CBIR scalable has primarily been attacked by development of multi-dimensional indexing techniques and use of dimension reduction.

This project uses CBIR methods stemming from three different feature descriptors first suggested by M. Stricker and M. Orengo [21], Z. M. Lu et al. [9], and J. R. Smith and S. F. Chang [19], respectively. The descriptions in these papers were used as a base for the actual method implementations.
Chapter 2

Methods

A CBIR method is in this report defined as a set of image descriptors, the way in which they are extracted and how they are compared. The methods are selected based on a number of criteria. They should be general-purpose, allow fast queries and reasonable extraction complexity. Three such methods, here called Color moments, JPEG-based (DCT) and Color sets are selected to be evaluated both for retrieval performance and scalability potential. Two ground-truth data sets, each with a corresponding performance metric, Average Match Percentile and Precision and Recall, are used for evaluating retrieval performance. The scalability evaluation is based on estimates of each method’s extraction complexity and feature vector properties.

2.1 A CBIR System’s Design

The design of a system that provides querying of an image database by content can typically be divided into certain single function components.

Given a set of images the first step is to reduce the information these images hold to a data set that is possible to query. This data set should contain the values of the features that “best” represent each image in the set. The process of retrieving this data is commonly called Feature extraction. More formally put, the component making this extraction is a function that maps the data of an image to the desired feature space.

Unless the query image is part of the target data set, it must be analyzed using the same feature extraction at the search moment. Data from this extraction should then be compared to the collected feature vectors in the image data set.

To compare how similar, or rather how dissimilar, two images are some sort of distance measure is needed. The component responsible for this should take as input a pair of feature vectors (one for each image) and give as output a value of their dissimilarity. Or formally: it should give the distance between two points in the feature space.
Having a component doing feature extraction and one measuring distance between feature vectors is enough to query an image database for the most similar images to a given example image. First the feature vectors of all images in the target database are extracted and stored. Then the feature vector of the query image is extracted and compared to each set of feature vectors stored in the previous step. Sorting the distance values from all comparisons in descending order clearly gives the best matches, that is the answer to the query.

Although it is possible to implement a CBIR system using just these two components, it is hardly efficient in terms of computational cost. Assuming the extracted feature vectors can be stored, each query still requires at least an exhaustive search of the stored feature vectors, that is, linear time complexity ($O(n)$, $n =$ number of images).

It should thus be clear that some better way of searching is required as the number of images in the database grows. In practice the standard way of improving query time for databases is to build a search index and this is also applicable in this case. However, because indexing techniques and the use of them have been thoroughly researched and applying them is reasonably trivial, an elaborate discussion about them lies outside the scope of this project. Indexing techniques are therefore not evaluated in the study but use of them is assumed when reasoning about the evaluated methods’ scalability.

A common way of improving the accuracy of CBIR systems is by using a feedback mechanism. A human test subject can teach the system by classing the hits of
queries as either relevant or irrelevant. Use of feedback training is not included in the study, the methods are assumed to be fully “autonomous”. This is both due to the problem of estimating how well relevance feedback works for large-scale image databases (when only performing tests on small-size databases) and in order to limit the scope of this project.

To summarize: a component for extracting feature vectors from images and a component to quantify the distance is enough to implement a CBIR system; it is the implementations of these that are evaluated. Hence, for convenience, a “method” in this report is defined as a set of image features, the way in which they are extracted, and the distance measure to compare them.

2.2 Image Features

The “Content” in Content-Based Image Retrieval in essence refers to the image data, or in extension, what may be extracted from the image data. It should not be hard to see that the data of image in its raw form (all pixels and their color values) is poorly suited for comparing similarity of images. It is much too detailed (very high-dimensional vectors) and not uniform among images—for example, all images do not have the same number of pixels, dimensional proportions or use the same representation of color. It is therefore of interest to find features describing an image in a compact, yet expressive and uniform way. Most importantly, the features to extract should capture how the image is perceived and the differences that can be distinguished by a human.

Image features evaluated for image retrieval in the research field can broadly be divided into low- and high-level features. Low-level features are features that are somewhat directly available from the image data; examples of such are colors, texture and shapes. High-level features, on the other hand, refer to concepts that are closer to the image’s semantics. They are generally domain-specific due to the fundamental difficulty of reading semantic information from images [4]. Examples of high-level feature extraction include face, scene and pattern recognition and analysis.

Only methods based on low-level features were evaluated in this project. The most essential ones for this project are described below.

Color

One of the first [22] and most researched [2] features for image retrieval is color. Clearly, an image’s colors are fundamental for how we understand it and they are also trivial to extract data on. How to best represent color for application in the domain of image retrieval is however not obvious and it requires some understanding of how humans see color.
CHAPTER 2. METHODS

Color space

Representing a color numerically requires a mathematical model of color. The human eye has three types of color photoreceptors and therefore any model of color must be expressed of at least three components [12, q1]. A color space is the space defined by such a model, an interpretation (typically in terms of human vision) of each component’s values, and the possible value ranges for components. Color spaces are generally spanned by three or four components but may also consist of a finite set of enumerable colors (one example is Pantone). The common way of coding a color in a color space is as tuples of component values.

To give a concrete example using the terminology above: sRGB is a color space using the RGB color model. The components of RGB are Red, Green, and Blue, each component allows values in [0.0, 1.0]. sRGB stipulates an interpretation in form of reference display, viewing, and observer conditions [10]. An example of a color in sRGB is (0.5, 0.5, 1.0) (a light-blue color).

The notion of perceptual uniformity for color spaces describes how changes in a color value relate to the perceived change. A color space is said to be perceptually uniform if any two equal distances in the color space are also perceptually equally distant. [12, q35] Color spaces with the RGB model serve as good examples of perceptually non-uniform color spaces. A small step in a RGB space at low intensity (low values) will not be noticed whereas a step of the same distance at high intensity will.

Perceptual uniformity is a desirable property for color spaces in the field of CBIR. Feature vectors based on color should reflect perceivable differences (but not non-perceivable ones) and that requires that they are extracted from images in a perceptually uniform color space. Color spaces with models such as CMYK, RGB and HSV were primary designed for display devices or print and are inaccurate in modeling human color vision. Several attempts to design a color model to reflect how color is perceived; the most recognized and used is probably the CIE model with the color spaces CIELAB and CIELUV.

Expressing color information

A color histogram describes the distribution of colors used in an image. The histogram consists of bins—one bin for each color in the image’s color space. The value of each bin is the number of pixels in the image with the bin’s color. This means that a color histogram expresses the probability of a pixel being of each color in the color space. Given a color space with \( m \) colors and an image with \( n \) pixels, a color histogram \( h = (h_1, \ldots, h_m) \), has the following property:

\[
    n = \sum_{i=0}^{m} h_i
\]

For typically used color spaces the color histograms of images are feature vectors of high dimension and are generally sparse— they have a high percentage of empty
2.2. IMAGE FEATURES

bins. To aid this, the “width” of the bins may be increased so that a single bin spans a range of colors. This can clearly be suboptimal as it might discard valuable information of the color distribution.

![Figure 2.2. A photographic image (reduced to 256 colors) and its color histogram](image)

Several alternative, generally more compact ways of capturing the color content of an image have been suggested.

- **Color moments** (see Section 2.4) use the first three statistical moments of the color probability distribution.
- **Dominant colors** describe only the “dominant” colors used in the image in the form of a normalized histogram. Color clustering is performed in order to determine what colors are dominant.
- **Color coherence vectors** (CCV:s) [11]
- **Correlograms** [7]
- **Color sets** (see Section 2.4)

**Texture and shape**

An image can be seen as a mosaic of patches with different *textures*. Even though the idea of texture is easy to grasp there is no established definition of texture in computer vision. An informal definition is that a region has a certain texture if it has, without considering its local shape or colors, properties that are approximately constant or periodic. A texture can be characterized by low-level statistical descriptors such as edge frequency, direction and randomness. Different types of filtering are normally utilized to extract texture properties.

An adequate specification of the shapes appearing in an image could tell much about the image’s content. There are unfortunately two problems with using shape
as a feature. First, recognizing semantically relevant shapes is hard. Secondly, describing the shape of a region in formal terms is not trivial.

Shape identification relies on segmentation where an image is partitioned into segments of “connected” pixels. Here, connected means that they share some visual characteristic—a possible indication that they depict the same object.

Regardless of the inherent difficulties with shape in CBIR, methods using it has successfully been applied in a number of domains.

**Spatial/Non-spatial features**

When discussing what feature vectors to extract for CBIR it is relevant to make a distinction between *spatial* and *non-spatial* features. A feature is here said to be spatial if it expresses any information in terms of location in the image. Examples of such are texture and shape descriptors, CCV’s and correlograms. Color histograms, on the other hand, are non-spatial; they contain no information on where colors are used in the image.

![Figure 2.3. Two images with the “same” pixels but different spatial distributions](image)

How colors are distributed in the image is obviously fundamental for our interpretation of it. Methods that do not keep such information have a clear disadvantage in CBIR. They do, however, have a benefit: non-spatial features are insensitive to affine transformations (such as rotation, scaling and mirroring) of the image.

Non-spatial features may nevertheless be used to express spatial properties if used on a region instead of globally. An example is the *Local color histogram* approach [23] where an image is segmented and histograms are extracted for each region.

**2.3 Scalability considerations**

The choice of method sets the limit on how fast a query can be processed and how feature extraction and query times change with the size of the data set.
2.4 EVALUATED METHODS

Extraction complexity

If feature vectors can be persisted and the target image database is available before the query, feature extraction can be done “in advance” or offline. Furthermore, feature vectors are typically based only on a single image’s properties. Therefore multiple feature extractions may independently be done in parallel.

For some methods, in order to extract the sought feature vectors, a conversion from one color space to another for the image is required. This kind of pre-processing may in some cases be more computationally expensive than the extraction itself.

Furthermore, most image format must be decompressed before any information considered salient for the method can be read. In this study it is assume that all “input” images are 8-bit JPEGs as it is the most common format for photographic images on the Internet today. This means that the cost of JPEG decompression and color space conversion was included in the comparison for methods that required it.

Feature vectors

To efficiently query a set of feature vectors, the data must be indexed. That is, a search index must be built which allows lookups with significantly less comparisons than doing an exhaustive search. Different techniques for indexing multi-dimensional data (commonly called multi-dimensional indexing) have been developed; many of them stem from research in indexing for spatial data in domains like GIS (Geographic information system). These techniques can create efficient indices up to a limited and then “degrade” to linear complexity. Indexing high-dimensional data is considered to be a fundamentally hard problem.

As important for scalability as the dimensionality of a feature vector is the size of its components. The size limits how much of the index that can be held in memory at any moment and also the cost of a single comparison between a pair of components. The difference in size for storing vector components of single bytes, integers or floating point numbers is significant (factors of one, two and four) even with small data sets.

2.4 Evaluated methods

A number of methods were considered for evaluation for this project, but only three were chosen. In the brief history of CBIR, much research has been put into exploring image features and improving distance measures, but the focus has generally been on retrieval performance rather than scalability [14]. As a result, a great deal of methods could be rejected right away due to computational complexity.

Criteria

For the selection of methods to evaluate, the following requirements were set:
• **Reasonable computational complexity of feature extraction**
  The method should allow extraction of feature vectors to be done in parallel and
  the cost of each extraction should be low enough to make indexing of very large
  databases (with millions of images) possible.

• **Low dimensionality of feature vectors and a small total amount of data**
  To build an index that allows sub-linear lookups and to hold many vectors in
  memory.

• **Fast to compute distance measure**
  Given a pair of feature vectors, finding the difference (or distance) between them
  should be a “trivial” operation.

**Color moments**

As explained in Section 2.2 a histogram is a commonly used form of capturing the
color statistics of an image. From a color histogram, **color moments** can be deduced — a compact form of describing the color characteristics of an image.

**Description**

In *Similarity of Color Images* Stricker and Orengo described [21] a compact form to
express the color distribution of the image. This simple approach uses the idea from probability theory of how distributions can be characterized by their **moments** and **central moments**. Specifically, the first moment and the second and third central moments of the color distribution are used in the Color moments method. In more common terms the first moment is the mean color; the second and third central moments are the distribution’s variance and skew. To make these values more comparable, the square root of variance (i.e. the standard deviation) and the third root of the skew is used.

As colors in our target image are expressed in components, the three moments will be given for each component/channel. If we denote an image’s pixels by $I_{ij}$
where $i$ is the color channel index and $j$ is the pixel index, the feature vectors are
given by:

\[
\bar{h}_i = \frac{1}{N} \sum_{j=1}^{N} I_{ij} \quad \sigma_i = \sqrt{\frac{1}{N} \sum_{j=1}^{N} (I_{ij} - \bar{h}_i)^2} \quad \gamma_i = \sqrt[3]{\frac{1}{N} \sum_{j=1}^{N} (I_{ij} - \bar{h}_i)^3}
\]
2.4. EVALUATED METHODS

Given two images $I$ and $J$, each using $r$ color channels, we name their color moments feature vectors $(\bar{h}_i, \sigma_i, \gamma_i)$ and $(\bar{h}'_i, \sigma'_i, \gamma'_i)$ respectively, where $i = 1, 2, \ldots, r$. The images’ dissimilarity can then be expressed using the function

$$dissimilarity(I, J) = \sum_{i=1}^{r} \left( w_{i1} |\bar{h}_i - \bar{h}'_i| + w_{i2} |\sigma_i - \sigma'_i| + w_{i3} |\gamma_i - \gamma'_i| \right)$$

in which $w_{ij}$ are adjustable (positive) weights that allow amplifying/dampening of moment or color channel differences based on their importance. An example of their use is given by Stricker and Orengo [21] — if the data set consists of images with similar lighting conditions, differences in mean color is likely to be of high value, and thus $w_{i1}$ should have higher value than the other weights.

Analogously, depending on the choice of color space, performance might be improved by amplifying changes in some components but not others. One example is increasing the effect of changes in hue when using an $HSV$ color space.

JPEG-based (DCT)

The JPEG (Joint Photographic Experts Group) image format has become the de facto standard format for photographic images on the Internet and elsewhere. It is a compressed format using the discrete cosine transform (DCT) to encode data. Several ways of using DCT data for CBIR have been suggested in the last decade [17, 4.2].

Motivation

Methods using DCT data directly avoid the cost of decompressing the image before analysis; the target features can be directly, or using just minor processing, read from the JPEG file. This gives a low total feature extraction cost but still provide salient information of the image.

Description

Even though the JPEG standard allows different types of color spaces, compressions and encodings, the so-called baseline format will be synonym with JPEG here unless other is specified. The baseline format implies that image data is encoding in the JFIF (JPEG File Interchange Format) format with $Y'CBCR$ color space using lossy compression.

Somewhat simplified, the following steps are carried out to encode a color image as JPEG:

1. The image is transformed to a luminance/chrominance color space $Y'CBCR$ color space. This is somewhat lossy due to rounding.

2. Downsampling is performed on the chromatic component; its resolution is often halved. This is highly lossy, but the perceived quality is roughly unchanged.
3. The image’s pixels are grouped in $8 \times 8$ blocks and each of them is transformed with DCT resulting in $8 \times 8$ frequency components. They form a spectrum from lower to higher frequency, which allows discarding information in the high frequencies.

4. The 64 frequency components are quantized by a chosen coefficient. Accuracy in amplitudes of high-frequency components (i.e. where brightness and color change is high) is less visible than accuracy at low-frequency. Because the larger coefficients of high-frequency components, quantization will cause a rougher approximations than for lower frequencies. This is usually the step where the greatest information loss takes place. Often, the “quality setting” in JPEG compression software corresponds to the quantization coefficient.

5. Encoding is done of the blocks’ now quantized coefficients using the loss-less Huffman encoding.

   In *A Content-based Image Retrieval scheme in JPEG compressed domain* Lu, Li and Burkhardt proposed [9] a method for CBIR based on both color and texture information from JPEG images.

   In the method, color information is retrieved by reading the four coefficients in the upper left corner of each $8 \times 8$ DCT block. From these coefficients the mean value of each color component for each $4 \times 4$ sub-block in the block can be calculated. Calculating these for every block and for each $Y'CBR$ component gives a good approximation of the color information in the image. The feature vector is then formed by calculating the normalized (accumulative) histograms for each mean value of each component, in total 12 histograms (3 color components and 4 sub-blocks per block).

   The texture feature vector described in the paper is based on energy, frequency and directional information retrieved from the DCT blocks. Six different groups of coefficients in each $8 \times 8$ block are summed respectively and mean and standard deviation for all blocks in an image are calculated using these sums.

   As apparent by the results in the original paper [9, 4.] and also supported by the experiments in this study, the color information adds little to the retrieval accuracy of this method. Due to little improvement in performance and large increase in feature vector size (and to some extent extraction cost), only the texture properties were considered in this comparison. By extension, only the texture feature is described in detail below.

   A texture feature vector is based on the following coefficient groups in each $8 \times 8$ block. The list below states each group’s “visual” meaning.
2.4. EVALUATED METHODS

1. **DC coefficient** — energy information
2. **AC coefficient** — frequency information
3. *Same as 2*
4. Vertical direction information
5. Horizontal direction information
6. Diagonal direction information

A dissimilarity measure for any two images $I$ and $J$ may then be formed using the Euclidean distance between their normalized texture feature vectors $\bar{f}^I$ and $\bar{f}^J$.

$$\text{dissimilarity}(I, J) = \sqrt{\sum_{i=1}^{12} (\bar{f}_{Y_1}^I - \bar{f}_{Y_1}^J)^2} + \sqrt{\sum_{i=1}^{12} (\bar{f}_{C_{H1}}^I - \bar{f}_{C_{H1}}^J)^2} + \sqrt{\sum_{i=1}^{12} (\bar{f}_{C_{R1}}^I - \bar{f}_{C_{R1}}^J)^2}$$

Here, $\bar{f}_i$ denotes the normalized feature value $f_i$, the value in the feature vector with index $i$. That is, $\bar{f}_i = \frac{f_i - \mu_i}{\sigma_i}$, where $\mu_i$ and $\sigma_i$ are the mean and standard deviation for the feature value $f_i$ of all images in the target database.

**Color sets**

As clear by the discussion in Section 2.2, methods utilizing spatial information have higher potential for good retrieval performance in many domains than those that discard it. Unfortunately summarizing the spatial properties of image features is seldom trivial. The **Color sets** method aims to retain basic information of spatial color distribution of an image in a form that allows efficient indexing.

**Motivation**

It is of interest to see how the relatively more sophisticated color sets method performs compared to the others. In other words, does the retrieval performance motivate increased complexity and computational cost? Furthermore, this is the only method that fully relies on spatial properties.
Description

The method referred to here as “Color sets” was developed at Columbia University as a part of VisualSEEk, a visual feature search engine [19]. In the VisualSEEk software a user may query an indexed image database by giving visual descriptors in form of colored regions with different shape and location. For instance, to search for images depicting large areas of tree foliage, the user might input (by drawing) a query for images with a green region in the upper part of the image. This query would then be translated to a color set used to search the index of color sets extracted from the target image set. How this method uses color sets, what they are and how they can be acquired from images will be the topic of this section.

The goal of the method is to identify and classify regions that are of probable interest to the user. [18] It is based on the assumption that coherently colored areas of a significant size tend to have a semantic meaning—for example, they might depict isolated objects or background. As noted in Section 2.2, segmentation can be used for this type of identification, but it has the disadvantage of being pure partitioning that cannot identify correct boundaries overlapping regions. Instead the method uses automatic region extraction through back-projection which allows for the same pixels to be part of multiple regions.

Back-projection in image processing means re-application of extracted data onto an image. The perhaps most common use for this technique is back-projection of histograms for locating objects in an image. [22] Given the color histogram of a query image $M$ (often portraying just a single object or a region of interest) and the histogram of the image to query $I$, a ratio histogram is computed as $R_i = \min(M/I, 1)$. Each pixel in the image is then set to the ratio histogram value, looked up with the pixel’s color as index, this is the back-projection. The resulting image is then convolved with a blurring mask and the peak value responds to the location of the best match of the query image, if there is one.

It is possible to use histogram back-projection for queries by visual example in small image sets, but for this study it is unsuitable. Here, the query histogram is unknown before the query moment; it should hence be obvious that the scalability of this method is poor, as it requires one to do the back-projection for all images in the database at each query. As a result the VisualSEEk method uses color sets instead of histograms for back-projection.

A color set is a compact representation of color information. Formally it is a binary vector in space where each axis corresponds to a color in a quantized color space. Thus, each element in the binary vector represents a color in the space; it is either included (the value is 1) or not included (the value is 0).

Back-projecting a color set onto an image having the same quantized color space is simple. For each pixel, if the pixel’s color is present in the color set, give it value 1, otherwise 0.

The idea of using color sets instead of histograms for back-projection is to make offline processing possible; a reasonable sized set of color sets may be generated and back-projected on each image in advance. But as it is only valuable to back-project
2.4. EVALUATED METHODS

a color set for which regions are expected to be extracted, a subset of color sets are selected based on the histogram of the image to be processed.

Once back-projection has been performed, regions are extracted using filtering and region labeling. Valuable data on regions, aside from their color set, include their location, bounding box and area (pixel count). This data, the regions’ feature vectors, can be indexed efficiently in the form of a binary tree using color sets.

In more detail, these are the stages of the region extraction:

1. Image color processing

   Given the input image \(I\) its color is transformed to the HSV (Hue, Saturation, Value) color space. HSV is chosen because its separation is closer to human perception of color than RGB, but still is relatively simple to transform to and from RGB.

   The HSV color space is then quantized. That is, each component is split into a number of degrees/levels. From experiments in this study and in the original paper, a good quantization was found to be 18 hues, 3 saturations 3 values and 4 grays. The human vision is most sensitive to small changes in hue, so a significantly higher accuracy is needed for that component. This gives a color space \(Q_M\) with a total of 166 (= 18 · 3 · 3 + 4) separate colors. An index is assigned to each color, and its binary color space is denoted \(B_{166}^c\).

   A median filter is thereafter applied on the color space transformed image to “smooth” out irregularities.

2. Color set back-projection

   The regions for each color set \(\hat{c}\) should then be extracted using back-projection, starting with all color sets using one color, then two colors, and so on.

   As mentioned, extraction resulting in zero regions or regions that are not big enough to be considered salient can be avoided by inspecting the global color histogram. More specifically, the two thresholds \(\tau_0\) and \(\tau_1\) are defined. First, the total number of pixels that are to be kept when \(\hat{c}\) is back-projected must be at least \(\tau_0\); this is evaluated by reading the representation of each color present in \(\hat{c}\). Secondly, any color in \(\hat{c}\) must have a value of at least \(\tau_1\) in the global residue histogram. The global residue histogram is defined as the global histogram where after every extraction the region’s contribution is subtracted—in other words it corresponds to pixels that have not occurred in any already extracted region.

   The result of each back-projection is a bi-level image and once again a median filter is applied to smooth out insignificant regions.

3. Region feature extraction

   Sequential region labeling is performed to identify the resulting isolated regions in the bi-level image. For each region, the number of pixels, its area,
is counted. Using another set of thresholds, \( \tau_a, \tau_b \) and \( \tau_c \) it is decided if a region should be kept or assigned zero-label (is background). For any region to be assigned a non-zero label it must have more at least \( \tau_a \) pixels, every color in its color set must be represented by at least \( \tau_b \) pixels and each such contribution must equal at least \( \tau_c \) (\%) of the whole area. Filtering with these thresholds asserts that the extracted areas of the images will be assigned to the best matching color set.

Each non-zero labeled region’s meta-data (area and bounding box) are stored along with its color set.

![Figure 2.4. Steps in feature extraction: 1. Original image, 2. Color space transform and filtering, 3. Color set back-projection, 4. Thresholding and region labeling.](image)

The dissimilarity between two images with the Color sets method can be defined as a sum of differences between each non-zero labeled region’s feature vectors. In turn, the difference between regions’ feature vectors can be defined as a weighted sum of the distance between their color sets and the distance between the regions’ spatial properties. The distance between two color sets is the Manhattan distance between the colors in the quantized HSV color space. By assigning weights to the terms in the sum, each of the properties color, area and location can be may be decreased or increased in “importance”.

### 2.5 Evaluation procedure

To evaluate the methods’ retrieval performance, an implementation of each method’s feature extraction part was developed. In addition a simple test framework was written to allow feature vectors to be extracted with each methods and persisted in a flat file database. Along with simple dissimilarity functions for each method, this made it possible to performed queries on test image sets. The remainder of this section will go into details on the implementations, the test data and evaluation methods used.
2.5. EVALUATION PROCEDURE

Implementation details

Feature extraction procedures and distance functions for each method was implemented in the C++ language. To carry out the tests in this study two separate programs were written: one for extraction and one for querying. The extraction program takes as input a list of image files and outputs a file with feature vectors of each image in serialized form. The query program takes as input a query image file and a file with serialized extracted feature data. As output it gives the “hits” in form of a list with image filenames and their dissimilarity to the query image sorted in ascending order on the dissimilarity value. Simple Ruby scripts were used to automate the process of testing (extraction and querying) for this study.

For the implementations of the Color moments and the Color sets method Adobe’s Generic Image Library [8] was used. Essentially, this library was utilized for basic image file operations, image data read/write access and color space transformations. For the implementation of the JPEG-based method, libjpeg [3] was used for partial decoding to allow reading DCT coefficients.

As a performance evaluation, rather than an evaluation of potential for scalability, was the goal of the experiments, the methods were implemented using naïve, or “least-effort” algorithms and without optimization.

Test data

To conduct a formal evaluation of retrieval performance for information retrieval, ground-truth has to be established. In image retrieval this ground-truth usually consists of queries along with a selection of images, the expected matches, from a given data set. As of 2010 the field of CBIR does not have any widely accepted ground-truth data or even an image database to evaluate the performance of its achievements. CBIR researchers have used a plethora of different image data sets for evaluation of “general-purpose” query by visual example methods. This poses a problem as the data sets seldom are comparable and are sprung from widely different sources.

Assuming that a data set has been selected, ground-truth queries are to be derived from it. Various approaches to this has been used in the field; if the images are categorized one might form ground-truth by choosing any image in the data set as the query and all images with the same categorization as the “correct” result. Another approach is to create a data set by grabbing frames from video and define a ground-truth query/result as the adjacent frames in a sequence with little visual variation. The most direct way to acquire ground-truth data is however to let human subjects form queries by judging the similarity of images in a given data set. The obvious advantage of this approach is that at least one human subject will agree that the retrieval performance of a method is good if its query results correlate with the ground-truth. The disadvantage of this approach is that it is time demanding and subjectivity among humans might be high. [20]
CHAPTER 2. METHODS

For the performance evaluation in this project, data sets from two sources were used, both acquired through direct human judging. The reason for this choice was two-fold. First, the images in such sets commonly represent a wide range of photographic content with “normal” visual variation, as opposed to for example grabbed video frames. Secondly, this study’s focus on visual similarity is captured in human similarity judgment as opposed to ground-truth based on categorization, which is often purely semantic.

To increase the possibility of cross-study comparisons, the goal has been to use ground-truth based on image database that are common and preferably freely available — the selected data sets are described here in more detail.

The UCID database
UCID (Uncompressed Colour Image Database) is a freely available image database consisting of 1,338 photos and 262 ground-truth queries. [16] The intention of its compilation was to study the effect of image compression in CBIR (most available data sets have compressed images); nevertheless, in this study JPEG compressed versions of the images were used. The ground-truth data has varying number of matches per query (from one to seventeen), each result set without any internal image ranking and with the query image considered not to be part of the target query set.

The “PE for CBIR” ground-truth data set
As part of the development of a new performance evaluation protocol for CBIR, Liu Wenyin, et al. suggested a ground-truth data set acquired through direct-labeling by seven college students. [24] The ground-truth consists of 200 queries in a database with 10,009 images. As with the UCID data, the queries have a varying number of matches, though here the matches have a ranking based on the agreement among the test subjects. The images used in the data set originate from various CDs in Corel’s image Gallery collection. Though selections of images from that vast collection have been very common in CBIR research, their availability is decreasing and as a result only a subset of the images was available and could be used in this study.

Performance metrics
Just as with CBIR test data and ground-truth, several different performance indicators have traditionally been used to describe how well a CBIR method correlates with the ground-truth. As a result, there is a lack of widely accepted metric used by all researchers. A reason for this is that the ground-truth data’s properties set constraints on the applicability of metrics. Key properties of the ground-truth data to consider are:
2.5. EVALUATION PROCEDURE

- Query-to-matches ratio — the number of matching images to any given query image.

- Result ranking — given that the query-to-matches ratio is $1 : n$ and $n > 1$, if the matches are sorted in order of relevance.

- The total size of the database and if the performance metrics is meant to be comparable between databases of different sizes.

For this study, two different metrics were used.

Precision and Recall

The combination of Precision and Recall is a commonly used measure in the domain of information retrieval, and so also in evaluation of image retrieval systems. Furthermore, the majority of the papers referenced in this study publish their results in form of precision vs. recall graphs.

If ground-truth data have been established for a test query, i.e. which the relevant images in response are, precision and recall is defined as:

$$ R = \text{relevant images}, \quad P = \text{retrieved images} $$

$$ \text{Precision} = \frac{|R \cap P|}{|P|} $$

$$ \text{Recall} = \frac{|R \cap P|}{|R|} $$

This means that if a retrieval has precision 1.0, all returned images are relevant, but it does not imply that all relevant images are returned. Analogously, if a retrieval has recall 1.0, all relevant images are returned, but it does not imply that only relevant images are returned. Thus, it is more interesting to talk about the precision at a recall level. To illustrate the retrieval performance the precision at recall levels from 0.0 to 1.0 may be plotted in a precision vs. recall graph (or PR graph). This is usually done in steps, for example the precisions at recall levels 0.1, 0.2, 0.3, . . .

As the retrieval performance is evaluated for a number of queries, the performance for each query should be taken into account when using a PR graph. This may be done simply by averaging the precisions at each recall level [1]:

$$ \text{Precision}(l) = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{\text{Precision}_i(l)}{|Q|} $$

Where $|Q|$ is the number of queries and $\text{Precision}_i(l)$ is the retrieval precision of query $i$ at recall level $l$.  

23
Average Match Percentile

While PR graphs give a detailed view of a method’s retrieval correctness, a single value performance metric allows direct comparison of methods without further interpretation. To compare results for tests on different databases it is necessary to use a metric based on database size. An early example of such is the match percentile, MP, first suggested by Swain and Ballard. [22, 3.1.2] For any query $Q$ on a database of size $N$, the match percentile is:

$$MP_Q = 100 \frac{N - R_Q}{N - 1}$$

Where $R_Q$ is the rank of the correct image (according to ground-truth) in the test result. This definition clearly assumes a 1:1 query-to-matches ratio, which is not the case with the ground-truth data of this study. Thus, the original paper on UCID made an extended definition of MP that takes the number of correct model images for each query into account. [16]

$$MP_Q = \frac{100}{S_Q} \sum_{i=1}^{S_Q} \frac{N - R_i}{N - 1}$$

Here, $S_Q$ is the number of images in the ground-truth and $R_i$ ($i = 1 \ldots S_Q$) are the increasing ranks of the right matches, i.e. $R_i < R_{i+1}$.

An average match percentile (AMP) of 100 corresponds to an ideal image selection — only the $S_Q$ correct images are selected in the first $S_Q$ positions of each query. If the AMP of a method is 98, on average, the correct images are in the first 2% of the retrieved images. A method that selects images completely at random will have an AMP of 50.
Chapter 3

Results

3.1 Performance comparison

The UCID database

The 262 queries from the UCID ground-truth were performed on the 1,338-image database for each method. Each image had a resolution of either $512 \times 384$ pixels or $384 \times 512$ pixels. As the UCID image database uses images in an uncompressed format (TIFF), the images were converted to JPEG with a Q-factor of 92 (but no other post-processing was done). The table below shows the calculated average match percentile for all queries and the number of queries that resulted in a correct image being in the first, second or third place.

Table 3.1: Average Match Percentile and best ranks count for the methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Color Moments (RGB)</th>
<th>Color Moments (HSV)</th>
<th>DCTs</th>
<th>Color Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMP</td>
<td>78.988</td>
<td>89.416</td>
<td>86.289</td>
<td>68.347</td>
</tr>
<tr>
<td>1st</td>
<td>37</td>
<td>64</td>
<td>62</td>
<td>13</td>
</tr>
<tr>
<td>2nd</td>
<td>11</td>
<td>28</td>
<td>13</td>
<td>5</td>
</tr>
<tr>
<td>3rd</td>
<td>9</td>
<td>10</td>
<td>15</td>
<td>3</td>
</tr>
</tbody>
</table>

To study the effect of JPEG conversion, the database images’ were converted with a number of different Q-factors. The queries for each method were then performed on every data set.
### CHAPTER 3. RESULTS

Table 3.2: Average Match Percentile with different JPEG quality factors

<table>
<thead>
<tr>
<th>JPEG quality</th>
<th>Color moments (RGB)</th>
<th>Color moments (HSV)</th>
<th>DCTs</th>
<th>Color sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>79.000</td>
<td>89.620</td>
<td>86.922</td>
<td>70.418</td>
</tr>
<tr>
<td>95</td>
<td>78.999</td>
<td>89.510</td>
<td>86.138</td>
<td>69.905</td>
</tr>
<tr>
<td>90</td>
<td>78.988</td>
<td>89.384</td>
<td>86.501</td>
<td>69.551</td>
</tr>
<tr>
<td>85</td>
<td>79.012</td>
<td>89.212</td>
<td>87.456</td>
<td>70.102</td>
</tr>
<tr>
<td>80</td>
<td>78.997</td>
<td>89.164</td>
<td>87.486</td>
<td>69.132</td>
</tr>
<tr>
<td>75</td>
<td>79.013</td>
<td>89.238</td>
<td>87.510</td>
<td>71.489</td>
</tr>
<tr>
<td>70</td>
<td>78.982</td>
<td>89.153</td>
<td>87.357</td>
<td>70.221</td>
</tr>
</tbody>
</table>
3.1. PERFORMANCE COMPARISON

The PE for CBIR database

For this comparison 172 queries from the “PE for CBIR” ground-truth were performed for each method. 28 queries from the ground-truth were not performed due to missing images. The database consisted of 8,476 images in total. All images in the database had a resolution of either $384 \times 256$ pixels or $256 \times 384$ pixels and were converted from Wavelet Image format to JPEG. No other post-processing was done. The precision for each method was measured at the 41 recall levels $0, 0.025, 0.05, \ldots, 1$ for each query.

In the graph below averaged measured precisions of all queries for each method at the given recall levels are shown.

![Graph showing averaged precision vs. recall for all queries using each method.](image)

Figure 3.1. Averaged precision vs. recall for all queries using each method.
CHAPTER 3. RESULTS

Figure 3.2. Precision vs. recall for each query using the Color moments method (RGB)

Figure 3.3. Precision vs. recall for each query using the Color moments method (HSV)
3.1. PERFORMANCE COMPARISON

Figure 3.4. Precision vs. recall for each query using the DCTs method

Figure 3.5. Precision vs. recall for each query using the Color sets method
CHAPTER 3. RESULTS

3.2 Example queries

To give an idea of ground-truth data sets’ image content and the retrieval performance of the implemented methods, the output of some example queries are shown below. The matches of each the query using the methods are listed with increasing rank from left to right.

Photographic image, from the UCID database

Query results for image ucid00647.tif (converted to JPEG) from the UCID database. The match percentile (MP) should give an idea of the methods performance for this particular query.

![Query image](image)

**Figure 3.6.** Query image:

Color moments method (HSV), $MP = 91.430$:

![Color moments method](images)

DCTs method, $MP = 78.928$:

![DCTs method](images)

Color sets method, $MP = 57.906$:

![Color sets method](images)
3.2. EXAMPLE QUERIES

**Drawn image, isolated object, from the PE for CBIR database**

Query results for image Disk08/Photos/Dino_Art/644001.JPG from the “PE for CBIR” database. The precision at recall level 0.5 for this particular query is displayed for each method.

![Query image](image)

**Figure 3.7.** Query image:

**Color moments method (HSV),**  $P@0.5 = 0.304$:

![Color moments method images](image)

**DCTs method,**  $P@0.5 = 0.927$:

![DCTs method images](image)

**Color sets method,**  $P@0.5 = 0.028$:

![Color sets method images](image)
3.3 Complexity/scalability comparison

Feature extraction complexity

Though it is generally possible to perform almost all image feature extraction offline (see Section 2.3), there is reason to consider the extraction cost. With large-scale image databases, or databases that grow fast, the total cost of extraction might be too high to be practical, even considering high parallelism. Furthermore, if the query image is not part of the target data set, its features vectors are not known in advance, and thus feature extraction must be done at query time.

Rough estimates of time complexity for extracting feature data for each method are presented here. If \( n \) is the number of pixels and \( m \) is the number of distinct colors used in the image:

**Color moments**

The JPEG image must first be decoded and color converted to either HSV or RGB. A color histogram for the image is extracted; this requires each pixel to be visited. For each color present in the histogram \( (h_i(x) > 0) \), calculate the three moments for each color component. Estimated complexity: \( O(n + 9m) \).

**JPEG-based (DCT)**

Visit each JPEG block \( (= n/64) \) and for each color component \( (= 3) \) calculate mean and standard deviation for each DCT group \( (= 6) \). Estimated complexity: \( O(2 \cdot 6 \cdot 3 \cdot \frac{n}{64}) = O(\frac{n}{2}) \).

**Color sets**

The JPEG image is decoded and color converted to quantized HSV. Then a median filter with a \( 5 \times 5 \) pixel kernel is applied, \( O(5^2 n) \). A histogram is extracted to find relevant color sets, \( O(n) \). Then for each such color set, back-projection is performed, \( O(n) \), and median filtering, \( O(5^2 n) \). The regions are labeled, (connected component labeling, \( O(2n) \) or theoretically best \( O(n) \)), and the salient ones are detected. In total \( O(n + 25n + 2n) \) per color set. As color sets are generated first using one color, then two colors, its theoretical possible that a large number of color sets must be back-projected. In practice this is rarely the case though if the thresholds are selected carefully, about five per image.

**Feature vectors**

What feature data are required for a CBIR method is essential for its scalability potential. The vectors’ dimensionality sets constraints on how efficiently the data can be indexed. The data size is important for knowing the number of vectors that can be stored in memory.

For each method, the feature vector properties for one image are listed below. This assumes a floating point can be represented in four bytes and an (short) unsigned integer in two bytes.
3.3. COMPLEXITY/SCALABILITY COMPARISON

Color moments
Three moments (mean, variance, skew) for three color channels (either RGB or HSV) 9-dimensional gives a 9-dimensional vector of floating point values: 36 bytes.

JPEG-based (DCT)
Two moments (mean and standard deviation) for six coefficient groups and three color components ($Y'$, $C_B$ and $C_R$) gives a 36-dimensional vector of floats: 144 bytes.

Color sets
For each region a color set represented as a 166-dimensional binary vector (21 bytes), the spatial data, which consist of location (2 integers), dimension (2 integers) and area (1 integer) must be stored. This means 31 bytes per region and assuming an average of three regions per image gives a total of 93 bytes.
Chapter 4

Conclusions and summary

The outcome of this project is essentially implementations of three potentially scalable methods for CBIR by image similarity and a comparative evaluation of them. This section discusses the results of the evaluations to conclude and give directions of future research.

The easiest way to get a sense of the methods’ retrieval performance is by looking at the number of queries with correct images in three first places in the UCID database tests. It should be evident that the retrieval performance in this test of the implemented methods for this data set is poor. Even the method with the most accurate results, Color Moments (HSV), only place a correct image in the first place for 64 of the 262 queries (≈ 24%). Including the second and third place only increases the percentage of “hits” to about 39% for the best method.

To further interpret the results produced by the UCID test, the Schaefer and Stich’s evaluation [15] of methods using the same data set is used as a reference. Comparing the AMP for the three methods to the reference methods shows that all evaluated methods have lower AMP, between 65 (color sets) and 89 (color moments, HSV), where the reference methods have AMPs between 92 and 97.

The UCID tests also compared the results for queries where the source image database had been converted to JPEGs with different Q-factors. Looking at Table 3.2 it should be evident that compression only had a marginal effect on the retrieval performance. For the Color moments and the Color sets method a lower Q-factor (higher compression) had a negative impact (≤ 0.5 in AMP). Using DCTs however, the effect was positive (≤ 1.7 in AMP).

The least expected results in this comparison are the dismal retrieval performance numbers of the Color sets method. Despite being the most sophisticated method, using multiple spatial features of the images, it turned out to be the least accurate in both retrieval tests. The Color sets method’s intended use was for queries by visual descriptors (as part of the VisualSEEk software) but the use here are for queries by image. There are test performed in the original paper [19] on query by example image and they give very good results. Those tests are however not extensive enough to allow any conclusions to be drawn on the method’s fitness.
for this use. While less likely, the possibility that choices in implementation of the method affected the retrieval performance negatively cannot be ruled out. Details of algorithms used for color space quantization and region labeling in the papers the method are not given, and might thus be different in the implementation evaluated in this study.

The PE for CBIR results are mostly consistent with the UCID results. From the averaged precision recall graph (Figure 3.1) it is apparent that the results are as poor as the UCID tests indicated. The DCTs method scored best, but only slightly better than Color moments (HSV). Here too, the Color sets method performed much worse than the others. Some query images resulted in satisfactory precisions at low recalls. One such image was the drawn example query image (Figure 3.7); querying it gave reasonable results using all methods, even using Color sets.

The retrieval performance of the Color moments method was greatly improved by having the images in HSV. This can be explained by the use of HSV which allows weights to be applied on hue, saturation and brightness independently — each having different value for the perceived similarity.

In order to estimate the scalability potential of the methods, asymptotic complexity of feature extraction was computed for each of them. From these rough estimates it is indicated that both Color moments’ and DCTs’ vectors are computational cheap to extract, with DCTs’ being the cheapest. Extraction of feature vectors for the Color sets method, on the other hand, is costly and the complexity varies widely with the input image.

The dimensionality and size of the feature vectors determines how efficiently they can be queried. With the Color moments method’s 9-dimensional feature vectors it is possible to build reasonably efficient multi-dimensional index. For the DCTs method, however, the dimension of the feature vectors is too high (36 dimensions) to expect an efficient index without any further dimension reduction. It should thus be further researched how dimension reduction could be applied to allow DCTs and other CBIR methods not included in this study to be indexed efficiently.

Though none of the methods in this project, on its own, showed a retrieval performance that makes it suitable for image search, it is still of interest to study how the Color moments and DCTs can be used to improve text-based image search.
Chapter 5

Bibliography


