Image Retrieval Using Resegmentation Driven by Query Rectangles

L. Cinque*, F. De Rosa*, F. Lecca*, S. Levialdi*
Dip. di Scienze dell’Informazione, Università di Roma
Via Salaria 113, 00198 Rome, Italy

S. Tanimoto+
+ Dept. of Computer Sci. and Engineering
Box 352350, University of Washington
Seattle WA 98195, USA

Author to which correspondence should be addressed:
Luigi Cinque
e-mail: cinque@dsi.uniroma1.it+

Abstract

In this paper we address two key issues in image retrieval: (1) the use of rectangles in queries to express properties of regions in the desired target images, and (2) the use of oversegmentation to build the index of images in the database. In our method, the rectangles in the user’s query are used to control a partial resegmentation of each candidate image. These query-driven partial resegmentations provide the features for determining the distance between the query and each candidate, so that the closest candidates can be determined and retrieved. This method enables the construction of image retrieval systems with completely automatic indexing.

1 Introduction

Most of the previous work on image retrieval by content uses image descriptions that are based upon statistical aspects (color histograms) or color and texture samples, taken at fixed locations in the images.

In order to allow better and more general kinds of retrieval, we employ descriptions of images in terms of regions that are obtained from a partial analysis of each image.

Some researchers have begun to use regions in recent years for image retrieval. Dimai and Stricker [7] use a small number of “fuzzy” regions in fixed positions to guide the computation of the image signatures; because the shapes and positions of the regions are not data-dependent, they do not obtain the benefits that regions based on segmentation could provide.

Another use of regions is that by Carson et al [8], in which a small set of regions is determined for each image using expectation maximization with color and texture features.

In this paper we address two key issues in image retrieval: first the use of oversegmentation when preparing the index of images in the database, and second the use of rectangles in queries to express properties of regions in the desired target images.

Our approach is to precompute a general structure for each image in the database. We call this general structure an oversegmentation. We show how to use the oversegmentation at query time to reduce the time needed to compute the “query to candidate” distance values.

In our method, first a subset of the images in the database are identified as candidates using an approximate, first-level comparison. Then the rectangles in the user’s query are used to control a partial resegmentation of each candidate image. The resegmentation is performed at relatively low computational cost because it uses the results of the oversegmentations produced during the indexing phase. These query-driven partial resegmentations provide the features needed for determining the distance between the query and each candidate, so that the closest candidates can be determined and retrieved.
The organization of the paper is as follows: section 2 outlines the basic technical ideas that underlie our method, section 3 describes the use of oversegmentation in order to build the index of images and the use of rectangles in queries, in section 4 results are presented, and finally in section 5 some further problems are discussed and conclusions are drawn.

2 Index Representation

2.1 Image Query by Content Problem

We define the image-query-by-content problem as follows. Given a query image \( I_1 \) and a database of images \( \{I_2, \ldots, I_n\} \), find the image \( I_i \) closest to \( I_1 \). The closeness is to be computed using a distance function \( D(I_i, I_j) \) which evaluates the shape, color, texture, position, and/or importance of the regions within their images.

2.2 Rich Region Descriptions

We use a new data structure called Rich Region Description, which is a list of regions obtained by performing different segmentations at different levels of detail. Since the candidate pool may contain many regions, a selection can be made to retain only those regions with an appropriate likelihood of being useful. This is made according to heuristics. The result of selection is a set of regions which is not necessarily a segmentation of the image. For example, it may cover some pixels more than once, and there may be some pixels that are not covered by any of the regions. This is not a disadvantage but actually an advantage, because this description is used for retrieval, not for reconstruction of the image. The description for an image, termed \( \text{desc}(I) \), is a set of region descriptions \( R_i \), where each of the regions is made up of pixels of \( I \). It is not necessary for every pixel to be represented in a region. Neither is it required that no pixel be represented more than once. Thus regions may overlap, and they don’t necessarily cover the entire image.

\[
R_i = \langle x_i, y_i, \text{red}_i, \text{green}_i, \text{blue}_i, \\
\text{color covariance}_i, \text{area}_i, \text{perimeter}_i, \\
\text{convexity}_i, \text{number of bays}_i \rangle
\]

In order to perform matching, the query image is “superimposed” to the oversegmentation of the image that is stored in the data base (Figures 1c and 1d), to obtain a new segmentation to be used for the matching algorithm: all the regions of the oversegmentation which are found under the area of each query rectangle are merged together to constitute a new region, which inherits the properties of the original regions (a new mean color and new shape information).

There is a criterion for deciding whether the region should be merged or not: the best results were obtained with the following heuristic: a region is merged if more that 60% of its area lies under the query region.

The new segmentation obtained is shown in Figure 1e: the justification for the use of this method is that whenever possible, the resulting new segmentations for the matching procedure will have a structure similar to the user query.

The query-driven partial resegmentations provide the features needed for determining the distance between the

3 Query-Driven Resegmentation

3.1 Computing Oversegmentations

As we already outlined, our approach to image retrieval in this work is towards the use of pre-calculated “coarse” descriptions of images, which can be obtained off-line with an automatic segmentation procedure.

The segmentation algorithm is described in [5], and is based on two main criteria: a multi-resolution approach and a target number of regions used as parameters for the algorithm.

The procedure works in two phases. In the first phase builds a reduced version of the image and performs simple region growing, which cycles until a target number of regions has been found. In the second phase, results from the first are scaled to the original size image; region merging is performed, based on the relative color values and sizes of neighboring regions. Subsequent iterations perform merging with progressively relaxed thresholds, until the target number of regions is obtained.

The regions resulting from this procedure form the descriptions for the images in the retrieval phase; we use “oversegmentations”, so that apparently unimportant details of images are not filtered out in the segmentation process. At the same time, using a relatively “higher level” description than the single pixels brings considerable speed improvements, as we’ll show in the results section.

3.2 Query-Driven Resegmentation

The main idea of this method is to be able to produce a segmentation (to be used for the matching process) that is dependent on the current query that the user specifies.

In our image retrieval system, the query is expressed using tools to draw rectangles with color, size and position constraints for the desired target images.

As an example, an user trying to find an image like Figure 1a will probably draw a query similar to Figure 1b.

The object \( \text{desc}(I) \) is what we term the rich region description for image \( I \).
The resegmentation process consists of the following steps:

1. storing over-segmentations of all the images in the database (this can be done off-line without user intervention)

2. each time a query is made to the system, a resegmentation is performed on the images of the database

3. the matching function is applied, to compare the new segmentation with the query, and the results are presented in relevance order

The process is fast because the resegmentation is performed at relatively low computational cost since it works with the over-segmentations produced during the indexing phase. A method to speed up the process has been implemented: only a subset of the images in the database are identified as candidates using an approximate, first-level comparison, using a k-d-tree data structure. In this way a computationally expensive part (ressegmentation and matching calculation) is performed only on images which really have a probability of answering the users’ queries. In our experiments, on a database of 350 images, the tree data structure enabled to prune the search and perform the matching on only the 10% of the database for typical queries.

4 Experiments

In order to perform tests, we developed an image retrieval system in Java, based on our resegmentation method. The user can build a database by selecting a folder with GIF or JPEG files and executing the command to index them: this run-once command builds the over-segmentations to be used later for retrieval, then builds the indexes on which the first-level comparison is made. There is a query interface by which the user creates colored rectangles in order to express his preferences about the images.

We tested our method on a 350 image database with performance comparisons that show the difference in processing times. In order to perform tests, we had our system record the history of the queries posed by the users, and then we selected some “typical” queries.

We analyzed the following cases:

a. computing complete segmentations at query time.

b. computing resegmentations at query time without index (performing matching on each image).

c. computing resegmentations at query time using an index.

In Figure 2 we have the timings for the segmentation phase of a typical 640x480 image of our database. since
it is a bottom-up process, images with less regions take more time to compute. The slowest part of the building of a database is the segmentation, while the indexing phase is relatively fast: in three hours the whole process, which is run-once, is completed and the system is ready to be used. Images can later be added without recomputing every segmentation.

During query processing, only about 10 percent of the images in the database are selected for resegmentation and matching, and both operations are very fast. This results in very fast response times: about 10 to 15 seconds for each query.

Case a) can not be used for user interaction because it is too slow, while the other two are acceptable and of course the best processing time is obtained with the use of both resegmentations and an index. The latter must be carefully planned to avoid affecting recall and precision of the system.

We also studied the influence of the granularity of the database (number of regions in the starting oversegmentation) on the processing times and on the results of the queries: we have found that there is not a big difference in processing times between 50 and 700 regions, and it would not be much useful to have a very high number of regions for "real-life" picture databases, so the best empirical setting would be to use databases made of 300 regions for each image.

<table>
<thead>
<tr>
<th>Regions</th>
<th>Segmentation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>700</td>
<td>26 sec.</td>
</tr>
<tr>
<td>300</td>
<td>28 sec.</td>
</tr>
<tr>
<td>100</td>
<td>28 sec.</td>
</tr>
<tr>
<td>50</td>
<td>30 sec.</td>
</tr>
</tbody>
</table>

| Time to build all segmentations (300 regions): | 9100 sec. (2 hours and a half) |
| Time to build the index: | 600 sec. (10 minutes) |
| Time to resegment an image: | less than 1 sec. |
| Time to compute the matching function between two images: | less than 1 sec. |
| Typical query processing time (45 images from the index): | 15 seconds |

Figure 2. Performance measurements of our system, taken on a Linux AMD 800Mhz PC with 128Mb of RAM and a data base of 350 images.

5 Conclusions and future work

In this paper, we addressed two key issues in image retrieval: (1) the use of rectangles in queries to express properties of regions in the desired target images, and (2) the use of oversegmentation when preparing the index of images in the database. In our method the rectangles in the user’s query are used to control a partial resegmentation of each candidate image. As we have seen in the results section, the use of the method enables completely automatic construction of image databases and the efficient computation of a matching function which reflects users’ needs.

References


