Adapting Indexing Features to Scene Structures for Relevant Image Retrieval

G. Qiu and S. Sudirman
School of Computer Science, The University of Nottingham

ABSTRACT

Scene Structural matrices (SSMs) are 2-D tables that can be used conveniently to capture the organization of a scene in a color image when it has been adaptively partitioned and represented by a bintree. In this paper, we present a number of methods to populate the cells of SSMs. Well-established and popular image indexing features are used. Because the SSMs implicitly contain scene organization information and intermediate level concept, indexing the cells with these features is equivalent to adapting them to the structure of the scene. We present experimental results on retrieving one type of images which can be described by an intermediate level concept – the landscape images. Results showed that the new methods significantly outperformed traditional methods.

1. INTRODUCTION

In this paper, we present a method to incorporate low level vision features with intermediate level visual information of a scene for content-based image retrieval. Although low-level features, such as color histogram [1], color correlogram [2] and numerals others developed over the years, such as texture features [3], all have had various degrees of success in content-based image retrieval, it has been long recognized by researchers that relying on low-level features alone is not enough and results are often unsatisfactory. Two major reasons make low-level feature inadequate. Firstly, the end user of a content-based retrieval is a human, and therefore the criterion for judging the success and failure of a technique is subjective. The visual system of a human is a very complex and highly developed (much of it is still unknown to mankind) which will inevitably employ high level visual information in the judgement of the result of a retrieval technique. Secondly, it is known that low level visual features such as color histogram and others bear no direct correlation with high level similarity (a similarity judged by humans). It is therefore quite easy to understand why low level features alone will not provide satisfactory results. Recognizing these reasons, researchers has been trying to incorporate higher level visual information, the so called semantics, with low level feature together to improve retrieval results [4]. However, even though the idea of using high level semantics is sound, high level semantical descriptions of a scene are difficult to obtain. Current practices to extract high level knowledge of a scene fall into these approaches. Some tried to link words with low-level features. Work in [6] used text annotated image items and tried to establish the correlation of a certain terms (text descriptions used by humans) with low-level features. Others tried to link high level concepts with low-level features. In [7], the authors used a similar idea to model high level concepts for video indexing. We present a method, which employs intermediate level concepts associated with low-level processing without relying on manual annotation (a laborious task) by human.

We are motivated by two simple observations. We looked through a small section (randomly chosen) of the Corel color photo collection used by many researchers. Our collection contained 60,000 images and the portion of it we looked at had 7400 images. We found that one particular class of images, one we termed Landscape images (an image is classified as a landscape image if part of the image is sky and the image was taken in outdoor settings) had a large quantity, 1123 out of 7400 (or 15%) could be classified as landscape images, more than any other easily definable categories. This suggested to us that to be able to accurately retrieve this class of image would be very useful, as has been indicated by other researchers’ work in this direction [8]. Because landscape is a high level concept, simple use of low-level features alone will unlikely to be successful. Learning models such as those suggested in [8] or other more sophisticated object recognition techniques will have to be used.

Fig. 1. Images and their horizontal edges.

Our second observation will probably be best illustrated by Fig. 1. The left column of Fig. 1 shows two landscape images and one non-landscape images. The right column shows their corresponding edges. It is seen that the two landscape images’ edge images both have a horizontal line separating the image into upper and bottom two parts (or at least they give the viewer such an impression). We observed this was true for most of the landscape images (according to our definition). This phenomenon may be related to perceptual organization process [9]. This suggested to us that the edges contain very important information in distinguishing landscape and non-landscape images. However, how to use the information to correlate it with the high level concept is not immediately clear. Philosophically and intuitively, we should be able to exploit the edge orientation information for the purpose of separating landscape images from other classes of images because it is so obvious from Fig. 1. However, it is also well known that the obvious things to human

1 Supplementary figures of the paper can be viewed at:
are actually very hard to modern computer systems, especially for perceptual tasks as we are dealing with in the current paper.

In a recent paper, the authors introduced the scene structural matrix (SSM) [5], an image descriptor in the form of two 2D matrices. SSM originated from our work in developing an integrated approach to color image coding indexing and retrieval [10]. In [10], we used a binary space partitioning tree [11] method to achieve the combined purpose of image coding, indexing and retrieval. However, the process of constructing the tree, and the use of the tree for image content matching can be computationally very expensive. We therefore simplified the method by using a bintree [12] and from the bintree we developed the scene structural matrix, a simple and effective content descriptor. The segmentation process, although very coarse, was able to capture the characteristics of high level structures in landscape images. In the initial implementation of the technique [5], only the color ratio across the partitioning line was used. In this paper, we use a number of low-level features to improve the performance of the SSM.

The organization of the paper is as follows. In Section 2, we give a brief background overview of bintree image segmentation (representation) and the scene structural matrix architecture derived from it. Section 3 describes a number of methods for constructing the scene structural matrix using some popular low-level features in the literature. Section 4 presents experimental results and section 5 concludes the paper.

2. A BRIEF OVERVIEW OF BINTREE AND SCENE STRUCTURAL MAXTRIX

2.1 Bintree for Image Representation

Binary space partitioning (BSP) tree is a constrained adaptive image segmentation method which can be used for efficient image representation and compression [11, 13]. Compared to tree structure segmentation methods with a rigid grid, such as quadtree [12], BSP tree can represent an image with fewer piecewise smooth regions. Compared to representation methods based on nonconstrained image segmentation (i.e., segmentation according to the values of the pixels), BSP tree (and other tree, such as quadtree) structure can describe the regions with a more efficient data structure (the tree).

The basic idea of a BSP tree for image representation is illustrated in Fig. 2. An appropriately chosen straight line is used to partition the image into two subimages. For each of the subimages, another appropriately chosen straight line is used to partition the subimage into two subimages and the process is proceed recursively until a stopping criterion is met (this can be either the subimage is smaller than a certain size or the subimage is smooth enough). This representation was used by [11] and [13] for grayscale image coding and was later extended to color image coding in [10]. The authors of [10] also used the tree representation structure for content-based image retrieval. However, the computational burden of building the tree representation is very heavy. To simply the process, [5] introduced a special case of the general BSP tree representation, i.e., the line parameters were quantized into two sets of values: \( \Theta \), \( \rho \) = \{(0, N/2), (g2, M/2)\}, where \( M \times N \) is the subimage size. In other words, the subimages (including the whole image) are partitioned either horizontally or vertically into two equal halves. This partitioning scheme is also known as bintree representation [12]. To determine whether a subimage is divided vertically or horizontally, we find the overwhelming edge directions within the subimage. If majority of the edges is in horizontal direction then the image is partitioned horizontally, otherwise vertically. Examples of partitioned image can be viewed at: http://www.cs.nott.ac.uk/~qu/Online/ICME2002.html

As can be easily understood, for such a strategy, the partition will be meaningful if an image is a landscape image as shown in Fig. 1, but less meaningful (in terms of its relation to the high level concept) in nonlandscape images, and will be even less meaningful if it is a random textured image. Therefore, one must be careful how to actually interpret the meaning of the partitioned image. We will discuss this later.

Fig. 2. An \( M \times N \) image is first partitioned by a line \( \rho = x \cos(\theta) + y \sin(\theta) \) into two half planes. The half planes (subimages) are then subsequently partitioned by a straight-line recursively.

2.2 The Scene Structural Matrix

With bintree representation, an image content descriptor, which captures the structure of a scene, was derived in [5]. When a line partitions a sub-image, it intersects with the two borders of the sub-image which are perpendicular to it forming a \( T \) shape structure at different resolutions. It is based on this \( T \) shape structure that the scene structural matrix was constructed. The SSM is a two dimensional array indexing the \( T \) shape structures of the bintree segmented image. There are only two types of \( T \) shape structures and their conjugates. Therefore only the \( \downarrow \) and the \( \uparrow \) structures need to be indexed since (\( \downarrow \) and \( \downarrow \)) and (\( \uparrow \) and \( \uparrow \)) will always appear in pairs. It is clear, at different level and depending on how a subimage is cut, the two arms of the \( T \)-shape features will have different lengths. The SSMs capture this fact by indexing the \( T \)-shape features of various sizes. There are two SSMs, SSM\( ^\downarrow \) and SSM\( ^\uparrow \) indexing the two unique \( T \)-shape features. Each cell in the matrix corresponds to the \( T \)-shape with a certain arm lengths. Let \( h, w \) be the length of horizontal arm and vertical arm of the \( T \) shape (\( \downarrow \) and \( \uparrow \)) features. We have

\[
\text{SSM}^\downarrow(i,j) \equiv \text{SSM}^\downarrow(h, w) = K(I) \\
\text{SSM}^\uparrow(i,j) \equiv \text{SSM}^\uparrow(h, w) = K(I)
\]

Where \( K(I) \) denotes a function which computes low-level features from subimage \( I \). Therefore SSM\( ^\downarrow(i,j) \) is indexed by low level features calculated from a subimage which has a width of \( W/2 \) and a height of \( H/2 \) and which is partitioned by a horizontal line in the middle. Similarly, SSM\( ^\uparrow(i,j) \) is indexed by low level features calculated from a subimage which has a width of \( W/2 \) and a height of \( H/2 \) and which is partitioned by a vertical line in the middle. As can be understood easily, the matrices implicitly contain information about how the scene was partitioned hence intermediate level knowledge about the scene.
is preserved in such a descriptor. Notice that not all cells of the matrices will have features, some will be empty. In [5], the values of the cells in $SSM^T$ was set to the accumulated average color difference (CD) across the horizontal arm, and the values of the cells in $SSM^T$ was set to the accumulated average color differences across the vertical arm. We shall call this type of SSM CD-SSM in this paper. It was shown that SSM constructed in such a way demonstrated spectral and spatial invariant properties in image retrieval. However, in many cases of content-based retrieval, we actually want to distinguish spectral and spatial variation between images, and therefore we would like to include such features as color in the matrices. The idea is that the structure (spatial and orientation) of the image will be captured in the structure of SSMs and the contents of the SSM cells should be used to capture various image features such as color and texture which have been proved to be very useful in image retrieval. Such a combination should in principle better capture the contents of a scene for more relevant image retrieval. Based on such rationale, we present several implementations in the next section.

3. INDEXING SSM WITH LOW-LEVEL FEATURES

With the bintree representation of an image and its associated SSMs, we would like to use established image indexing features to populate the cells of SSM. There are some well-known and effective low-level features, such as color histogram [1], color correlogram [2] and texture descriptors [3]. One of the criticisms for local features was that they did not take into account the spatial relation and low-level similarity (Euclidean distance) does not measure high-level (perceptual) similarity. However, by incorporating the low-level features with SSMs, which capture the intermediate level information of the image, should be a useful way to make low-level features more effective.

3.1 Color Histogram SSM (CH-SSM)

Color histogram was one of the first features developed for content-based indexing and retrieval [1]. This is probably the best known and most often used. Our experiences are that it is an effective indexing feature despite its inherent weakness. Therefore the first feature we want to use is the color histogram. After constructing the bintree, we compute the color histogram for the subimages and populate the cells. For convenience, we use following notation:

$$SSM^H(i, j) = CH^H(i, j)$$
$$SSM^T(i, j) = CH^T(i, j)$$

Where CH represent color histogram, and the $R(\ )$ function in (1) computes color histogram.

3.2 Color Correlogram SSM (CC-SSM)

Another effective low-level feature we found worked well was the color correlogram [2]. This feature included local spatial relations of color. For computational simplicity, autocorrelogram is used in practice. With this indexing feature, we represent the matrices as

$$SSM^H(i, j) = CC^H(i, j)$$
$$SSM^T(i, j) = CC^T(i, j)$$

Where CC represent color correlogram, and the $R(\ )$ function in (1) computes color correlogram.

4. EXPERIMENTAL RESULTS

To evaluate the performance of the method, we have performed some extensive experiments. The experiment setting was as this. The image database consisted of 7400 color photo from the Corel collection. From the database, we labeled all the landscape images. We classified an image as a landscape image if part of the image contained sky features and the image was taken in an outdoor environment. There were a total of 1123 such landscape images, or 15%. In other words, if we randomly picked an image from the database, the chance of that image being a landscape was roughly 15%. From this group of landscape image, we randomly chose 280 as query examples. The objective was to retrieve as many landscape images as possible and in the highest ranks as possible. To measure the performance, we used an averaged cumulative retrieval rate. For a query $k$, let $N_k(i)$ be the number of landscape images returned which are within the first $i$ positions (ranks). The average cumulative landscape image retrieval rate (ACR) for the $i$th rank is defined as

$$ACR(i) = \frac{1}{K} \sum_{k=1}^{K} \left( \frac{N_k(i)}{i} \right)$$

K is the total number of queries performed (280 in our case). That is, ACR($i$) measures on average, the percentage of images within the first $i$ returns are landscape images, and therefore it measures the retrieval relevance.

As a benchmark, we also implemented MPEG-7 color structure descriptor [15] and color correlogram (the same way as it was done in [2]). The MPEG-7 CS used 256 colors and color correlogram used 64 colors. All SSMs were built for 5 level partition. We compared the performances of various methods. Fig. 3 shows the average cumulative retrieval rate of various methods. Of the two benchmarks, color correlogram had a better performance. Using the color difference SSM (CD-SSM) of [5], the performance is slightly worse than the benchmarks. With color histogram SSM (CH-SSM), the performance is much better than the benchmarks, on average outperformed the better benchmark by 12%. The color correlogram SSM (CC-SSM) significantly outperformed the benchmarks and on average is about 26% better. We then combined the color different features of [5] and other features, i.e., each cell in the SSM now consisted of two types of features. A combination of CD-SSM and CH-SSM (CD+CH)-SSM outperformed the benchmarks on average by 35%, and used the CC-SSM and CD-SSM together (CD+CC)-SSM, the performance was improved over the better benchmark by 43% on average. The last diagram of Fig. 3 summarized the improvements of four SSM schemes over the color correlogram method. It is seen that these methods all performed better. The best performance was achieved by the (CD+CC)-SSM, which improved the performance over the benchmark by as much as 61% for the first ranks. That is, for 280 queries, 85 queries returned a landscape image in the 1st rank for the color correlogram method, and 137 queries returned a landscape image in the 1st rank for the (CD+CC)-SSM method. An illustration of retrieved image examples can be viewed at URL: http://www.cs.nott.ac.uk/~qiu/Online/ICME2002.html
more weights to local features such as color correlogram. SSM is a convenient way to capture the structure of the partition, other methods should be developed to capture different intermediate to high level concepts.

6. REFERENCES

[16] J. Mundy, A. Zisserman and D. Forsyth, Editors, Applications of invariance in computer vision, Spinger-Verlag, 1994