T. K. Lau and I. King. Montage: An image database for the fashion, clothing, and textile industry in Hong Kong. In *Proceedings of the Third Asian Conference on Computer Vision (ACCV'98), also as Lecture Notes in Computer Science # 1351*, volume I, pages 410–417, January 4-7, 1998. Springer Verlag Berlin Heidelberg.

Montage : An Image Database for the Fashion, Textile, and Clothing Industry in Hong Kong

Tak Kan Lau and Irwin King

Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, New Territories, Hong Kong
{tklau,king}@cse.cuhk.edu.hk

Abstract. The fashion, textile, and clothing industry is a main constituent in Hong Kong. In this industry, handling a large amount of images is an important task in various phases, for example, the designing, sourcing, and merchandising phase. We develop an image database system called, Montage for managing and retrieving these visual information efficiently and effectively. Montage is an image database supporting content-based retrieval by *color histogram*, *sketch*, *texture*, and *shape*. One important feature of Montage is the *Open Architecture* design which makes the system *extensible*, *customizible*, and *flexible*. There are two aspects of this open architecture design: (1) Open DataBase Connectivity (ODBC) and (2) plug-in framework which we will discuss in more details. Moreover, we describe an experimental Java system enabling internet access to Montage. In the paper, we also present an experiment to evaluate the performance of several query methods.

1 Introduction

The fashion, textile, and clothing industry is a main constituent in Hong Kong's manufacturing sector which is one of the largest employers in Hong Kong. In this industry, handling a large number of images is an important issue in various phases such as the designing, sourcing, and merchandising phase. For example, fashion designers often refer to previous designs from a large collection of designer sketches when creating new designs; consumers and retailers have to purchase apparel merchandise from catalogs.

In the past, we use traditional databases to handle images. These databases use keywords for image retrieval which poses difficulties for the end users without special training. First, different users may use different words to describe an image. Second, even when a standardized vocabulary is used, it is still hard to depict the image clearly and precisely.

In order to manage such a large amount of images efficiently and easily, image databases [4, 1, 5] are emerged. These databases support content-based retrieval which lets us retrieve images by image content. By using visual information, we can describe images more accurately and easily.

We develop an image database system named Montage for the fashion, textile, and clothing industry in Hong Kong on the Windows NT and Windows 95 platforms. The goals of the system are:

- *Open Architecture Design*: Open architecture allows users to customize or extend the system to their own needs. *Open DataBase Connectivity* (ODBC) and *plug-in framework* are the two main aspects in the design.
- *Internet Solution*: An experimental Java system enables internet access to Montage. The system becomes platform independent by using Java applet programming. Moreover, users from all over the world may retrieve images in the server database.
- *Content-Based Retrieval*: Montage supports content-based image retrieval by using color histogram, sketch, texture, and shape. Users can retrieve images easily and efficiently in the system by using image content.

In the rest of this paper, we discuss the system architecture and the internet solution of Montage in Section 2 and Section 3 respectively. Section 4 describes the details of the query methods. Section 5 shows how to use classification trees to pre-organize the images. We present an performance experiment on the query methods in Section 6. In section 7, we discuss some possible applications of Montage for the industry. A conclusion is drawn in Section 8.

2 System Architecture

Montage uses an open architecture system design. It contains three main layers: *Core Layer*, *Extension Layer*, and *Application Layer*. Each layer consists of different modules as shown in Fig. 1(a). First, the core layer contains all the necessary components of the system for basic image processing and retrieval. Second, the extension layer contains modules for system customization. Third, application programs can be built on top of the system in the application layer.

2.1 Catalog Module

Catalog module is responsible for database manipulation such as creating new catalog, inserting images into a catalog, and retrieving images from a catalog. It uses a master file to store all the records of an image catalog. Moreover, the master file also keeps the header information of the file of the thumbnail images extracted from the database images for later display.

2.2 Image Processing Module

Image processing module is responsible for the image processing operations for the system. It provides basic image processing functions such as image transformation, filtering, resizing, color adjustment, 2D object drawing, and the object-oriented operations. With these functions, users can edit images stored in the database and create new images.

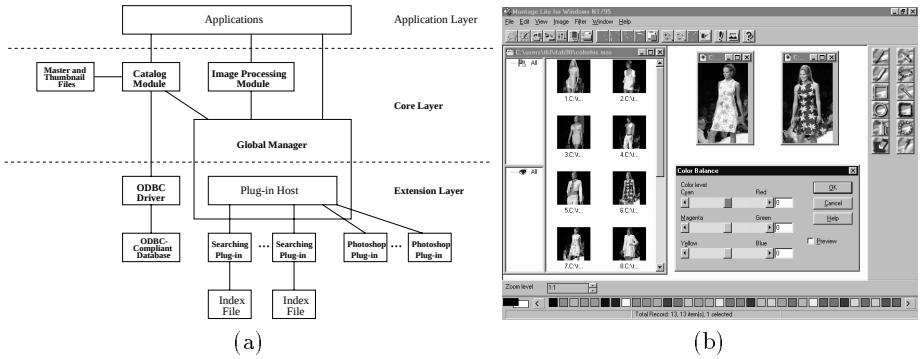

Fig. 1. (a) The system architecture. (b) Image editing and image browsing.

2.3 Global Manager

Global manager communicates with the main modules of the system. It is in charge of global resource management. It also takes part in the plug-in framework of the system. The main tasks of global manager are: (1) keeping the global status of the system such as current foreground color, (2) managing the global utilities such as MeasureUnit, (3) supplying global user interface such as tool boxes and color palettes, and (4) providing a plug-in host environment.

2.4 ODBC driver and the ODBC-compliant Database

Open DataBase Connectivity (ODBC) is a uniform interface standard used to access ODBC-compliant databases. By using a specific ODBC driver, ODBC-compliant databases can be accessed by the same set of ODBC calls.

In Montage, we use an ODBC-compliant commercial database for alphanumeric data manipulation. We use ODBC to access the database which is used for non-image related processing and image query by keywords. ODBC is useful for the users who already have their own databases to manage image data. By using ODBC, those users can retain all the data in our system and are able to retrieve images by features. Moreover, our system gains a set of text-related operations. In short, Montage complements other ODBC-compliant databases.

2.5 Plug-in Host and Plug-in Modules

We use a plug-in framework to make Montage extensible, maintainable, and flexible. The plug-in framework has two main components: *Plug-in Host* and *Plug-in Modules*. We can enhance or customize the system by means of plug-in modules. Plug-in host is used to provide the necessary system interfaces for the plug-in modules to interact with Montage.

Montage supports two main kinds of plug-ins :

- *Photoshop Plug-ins* : Many Photoshop plug-ins can be found on the market and they provide various image processing functions. We can enhance the image processing module by adding these plug-ins to Montage. As a result, we can reduce the functionality of the image processing module and leave most of the operations to plug-ins. Hence, the system becomes more flexible.
- *Searching Plug-ins* : A searching plug-in corresponds to an image searching method (see Section 4 for details). By adding searching plug-ins, new searching methods can be used by the system. Basically, each searching plug-in is in charge of the following tasks:
  - *Extracting feature*: When a new image is inserted into the database, the catalog module calls all the loaded searching plug-ins to extract specific features from the image for later retrieval.
  - *Indexing*: In order to reduce access time and narrow down the search space in image searching, different indexing methods such as R-tree [3] are used in different searching plug-ins.
  - *Searching images*: The searching plug-in calculates search keys for the queries of the searching method. It compare the features of the images in the database with the search keys to produce the results of the queries.
  - *Providing user interface*: Each searching plug-in provides the user interface to specify queries for the searching method (see Fig. 2).

2.6 An Application Program

There is an application program built in the application layer. Basically, it tightly integrates two sub-programs: *Image Catalog* and *Image Editor*. Image catalog provides a user-friendly interface to access the image database whereas image editor allows users to modify images in the database and create new images. Using this application program, users can browse and edit images at the same time (see Fig. 1(b)). This feature is very essential. For example, fashion designers can refer to the previous designs while they are creating new designs.

3 Internet Solution

The Internet is growing rapidly in recent years and has become one of the best ways for people to communicate around the world. Hence, it is essential to make Montage internet accessible.

Apart from the main system described in Section 2, we are prototyping an experimental Java system to enable internet access to Montage. By using Java applets, the system becomes platform independent. In the experimental system, the searching engine and the image database are located at the server side. Users from the client sides may use Java-compliant WWW browsers to execute the Java applets and specify queries for remote image retrievals. The queries are then sent to the server for processing and the results are returned and displayed on the browsers.

4 Query by Image Contents

4.1 Query-by-Color-Histogram

Query-by-color-histogram uses the global color distribution of an image for image retrieval (see Fig. 2(a)). We make use of RGB and Hue values of the pixels to calculate a 46-bucket color histogram for each image in the database. Then, the histogram will be indexed by R-tree. After pre-processing all the images, we can specify a query image and perform nearest-neighbor search to find images with overall color distribution similar to the query image.

4.2 Query-by-Sketch

The query-by-sketch method makes use of the regionalized color information of the images for retrieval. For each image in the database, we first partition the image into 400 non-overlapping (20×20) same-size partitions. Then, we quantize the RGB color space evenly from 16.7 million to 4096 colors. We find the modes R_m, G_m, B_m, and a Hue value calculated from the three modes for each partition. An 1,600 feature vector is then formed by collecting the modes and the Hue values from all the partitions of the image.

A sketch pad is used to specify sketch queries as shown in Fig. 2(b). Based on the color assigned to each cell, we calculate a key feature vector for an query according to the method mentioned in last paragraph. We then compare the key feature vector with the feature vectors of the images in the database to find out the result of the query.

4.3 Query-by-Texture

The query-by-texture method uses statistical methods to compute textural features for texture matching (see Fig. 2(c)). For each image in the database, we first quantized the intensity space to 16 gray levels. Then, we compute six features: *smoothness* SM, *angular second moment* ASM, *contrast* CON, *correlation* COR, inverse element difference moment $IEDM$, and *entropy* EN form the image [6, 2]. Based on the six values, we calculate a representative feature value F for the image by the formula $F = \lambda_1 \cdot SM + \lambda_2 \cdot ASM + \lambda_3 \cdot CON + \lambda_4 \cdot COR + \lambda_5 \cdot IEDM + \lambda_6 \cdot EN$ where $\lambda_1, ..., \lambda_6$ are the pre-defined weighting factors. Given a query texture, its extracted feature value will compare to the feature values of the images in the database. Images having smaller differences in feature values are selected as the result of the query.

4.4 Query-by-Shape

The query-by-shape method uses the shape information in an image for retrieval (see Fig. 2(d)). For each image, the method uses the user-specified boundary points of the shapes in the image to calculate a feature vector by using Fast Fourier Transform. A user may then draw a polygon as an query to find the images with shapes similar to the query polygon.

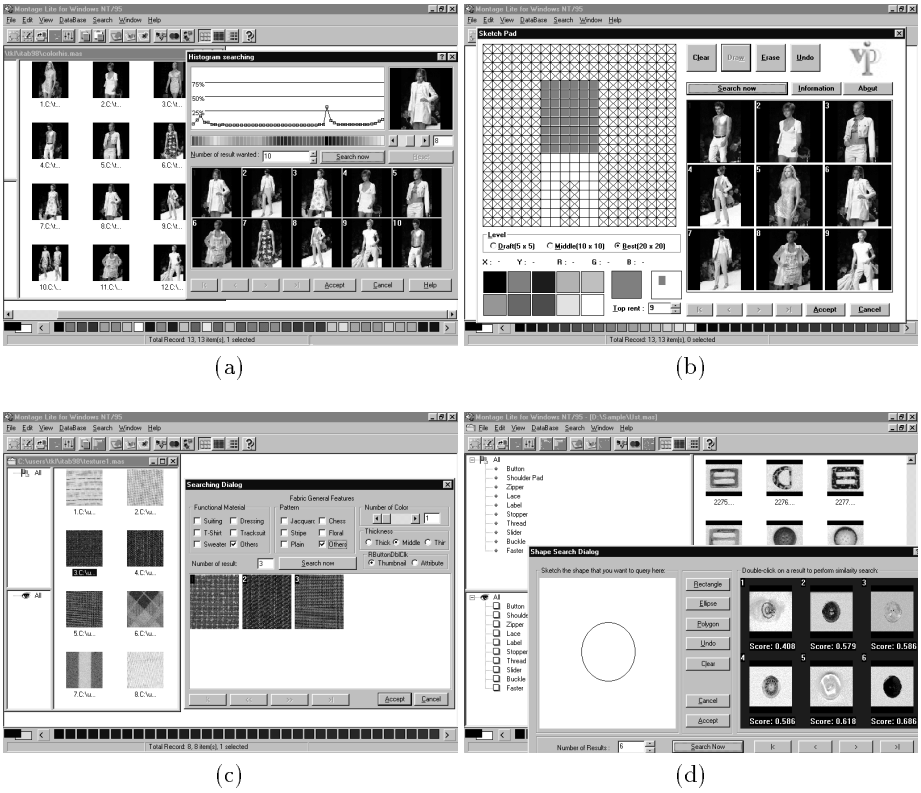

Fig. 2. (a) Query by color histogram. (b) Query by sketch. (c) Query by texture. (d) Query by shape.

5 Classification Tree

One way to allow users to pre-organize the image information is the use of classification trees (see Fig. 3(a)). Each image catalog keeps a classification tree for image classification. A classification tree is a hierarchy of image classes. Every image in the catalog belongs to one or more user-defined classes. With the classification tree, images retrieval become more efficient. If we want to find some images belonging to certain classes, we can first specify the classes in the classification tree in order to reduce the search space. Searching can then be performed on a smaller database to obtain an accurate result faster.

6 Performance Experiment

We present an experiment on Montage to evaluate the performance of several query methods on a fabric database. It was conducted on a 32M-RAM Pentium-100 PC using the Windows 95 operating system. We used two performance

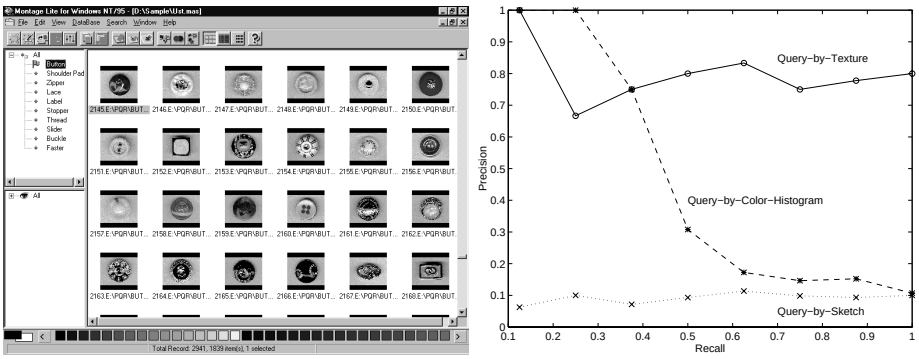

Fig. 3. (a) A classification tree. (b) Recall and Precision results of the experiment.

measures, namely *Recall* and *Precision*, to evaluate the accuracy of the query methods for image retrieval. These measures are given by :

$$\text{Recall} = \frac{\text{Number of target images retrieved}}{\text{Number of target images}} \tag{1}$$

$$\text{Precision} = \frac{\text{Number of target images retrieved}}{\text{Number of images retrieved}} \tag{2}$$

We used a database of 100 fabric images to evaluate the performance of three query methods: query-by-texture, query-by-color-histogram, and query-by-sketch for texture searching. We did not test the query-by-shape method because it was obviously not suitable for this kind of database. All the images were in GIF format with 128×128 pixels. We tested the methods to retrieve 8 pre-selected target fabric images. In the experiment, we selected one of the target images as the query for the first two query methods. Based on this target image, we specified a sketch query for query-by-sketch. It took 130 seconds to pre-processing the 100 images. Their Recall and Precision performance was shown in Fig. 3(b).

For an ideal case, the precision would be 1 for every recall. For recall values over 0.5, both query-by-color-histogram and query-by-sketch have precision values less than 0.4. Therefore, they are not good for texture searching. For query-by-texture, we can see that the precision is up to 0.8 even for high recall values. Hence, it was a good method for locating similar fabric images.

We did similar experiments on different kinds of databases and found that different query methods are good in different cases. Therefore, Montage included the four query methods for different applications and users.

7 Applications

Montage is a content-based image retrieval database and it is potentially useful for the fashion, textile, and clothing industry. Two possible applications are:

- *Office Automation*: Companies may manipulate and manage their visual information in electronic format rather than in hard copy format. In the fashion industry, for example, fashion designers may store their previous designs in Montage so that they may refer to these designs quickly and easily. Moreover, they may create new designs at the same time.
- *Global Sourcing*: Global sourcing is an important issue in the textile and apparel industry. Manufacturers often need to find different suppliers from a large printed catalog for different resources used in the manufacturing phase. Using Montage to organize the suppliers' information, manufacturers can make use of the graphical interface to specify queries for effective retrievals.

8 Conclusion

We have developed an image database system called, Montage for the fashion, textile, and clothing industry in Hong Kong. It supports content-based retrieval by color histogram, sketch, texture, and shape. The system provides an open architecture for the third-party developers by means of ODBC and plug-in framework. We have also implemented an experimental Java system for internet access to Montage. With these features, Montage is potentially useful for the industry.

9 Acknowledgment

This work is supported in part by Hong Kong's Industry Grant #AF/17/95 (2427 01300) and a RGC Grant #CUHK 485/95E(CU95513). The authors would like to thank Prof. Ada Fu, Prof. Laiwan Chan, Prof. Lei Xu, and all the members of the VIP Lab (`http://www.cse.cuhk.edu.hk/~viplab`) for their contributions to the system described here.

References

1. J.R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. Jain, and S. Chiao-Fe. The Virage Image Search Engine: an open framework for image management. In *Proceedings of the SPIE - The International Society for Optical Engineering*, volume 2670, pages 76–87, 1996.
2. R.C. Gonzalez. *Digital Image Processing*. Addison Wesley, 1992.
3. A. Guttman. R-tree: A Dynamic Index Structure for Spatial Searching. *ACM SIGMOD*, 14(2):47–57, 1984.
4. W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The QBIC project: querying images by content using color, texture, and shape. In *Proceedings of the SPIE - The international society for optical engineering*, volume 1908, pages 173–87, 1993.
5. J.R. Smith and S. F. Chang. VisualSEEk: a fully automated content-based image query system. In *ACM Multimedia '96*, November 1996.
6. F. Tomita and S. Tsuji. *Computer analysis of visual textures*. Kluwer Academic Publishers, Boston, 1990.