A Feature-Based Image Retrieval Database for the Fashion, Textile, and Clothing Industry in Hong Kong

Irwin King and Tak Kan Lau
{king,tklau}@cs.cuhk.edu.hk
http://www1.cs.cuhk.hk:8080/ imdb/

Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, New Territories, Hong Kong

ABSTRACT

We present an image database system for the fashion, textile, and clothing industry in Hong Kong supporting feature-based retrieval by *color histogram*, *color sketch*, *shape*, and *texture*. The Query-by-color-histogram method uses feature vectors extracted from the color distribution of an image for retrieval. The Query-by-color-sketch method makes use of the regionalized color information of the image for retrieval. The Query-by-shape method uses a two-stage polygon representation for shape searching. The Query-by-texture method uses statistical texture analysis to compute textural features for texture matching. One important feature of our system is the *Open Architecture* design. This design allows the system to be *extensible*, *maintainable*, and *flexible*. There are two aspects of this open architecture: (1) Open DataBase Connectivity (ODBC) and (2) plug-in framework. Moreover, we describe a server/client system design enabling internet access to the system. Based on our system design, we demonstrate a fashion image database and a fabric image database.

1. INTRODUCTION

Manipulating a large number of images is one of the main tasks performed during the designing and merchandising phase in the fashion, textile, and clothing industry in Hong Kong. Fashion designers often refer to previous designs from a large collection of designer sketches when they are creating a new design. Often, consumers and retailers have to purchase apparel merchandise from fashion catalogs. In the past, the above tasks are done manually using printed catalogs. This task is laborious and inefficient. Some fashion companies have used traditional databases to store and retrieve images. However, these databases mainly use keywords or text descriptors for retrieval which poses difficulties for the end users without special training. In order to manage a large amount of fashion-related images efficiently and easily, we develop an image database system which supports feature-based retrieval by *color histogram*, *color sketch*, *shape*, and *texture*.

Many Content-based retrieval multimedia database have been developed in the past few years. For example, Query by Image Content(QBIC) [10] allows queries on database based on color, texture, and shape of image objects and regions. Other database systems which support content-based query include the Photobook system [12], the Chabot system [11] and the VisualSEEk system [14]. Moreover, ConQuest Software, Inc. and Virage are producing commercial quality image retrieval systems.

We develop a feature-based retrieval image database system for the Windows NT and Windows 95 platform. It is specially designed for fashion, textile and clothing industry. The features of our system are:

1. **System Customization** – To let users customize or extend the system to their own needs, an image database environment with open architecture for the third-party developers is provided. We use Open DataBase Connectivity (ODBC) [5] as the interface to other commercial databases that are ODBC compliant. Moreover, we use the Photoshop plug-in framework for image editing (Section 2.1 - 2.4).

2. **User-Friendly Interface** – The system provides two modules that interact with the users. The first module is the *Image Catalog Module* which provides image accessing functions from the database. The second is the *Image Editor Module* which lets the users to edit both bitmap and vector format images (Section 2.5).

3. **Internet Solution** – A server/client design is adopted for the system to enable internet access to the system. The system becomes platform independent by using Java applets and HyperText Markup Language (HTML) (Section 3).

4. **Feature-Based Retrieval** – The system supports feature-based query by color histogram, color sketch, shape, and texture to let users search images easily and efficiently (Section 4).

5. **Fashion-Related Image Databases** – A fashion image database and a fabric image database are built on our system for demonstration (Section 5).

Points 1, 3, and 4 show the new features of the feature-based image retrieval database. Points 2 and 5 demonstrate the uniqueness of our system specially for the fashion, textile, and clothing industry in Hong Kong.

2. SYSTEM ARCHITECTURE

The architecture of the system contains two main parts as shown in Figure 1(a). One is the **System Engine** and the other is the **Interface Module** of the system. The System Engine consists of several components: *Database Engine*, *Image Processing Engine*, *ODBC Driver*, *Text Database*, *Photoshop Plug-ins*, and *Search Plug-ins*. The first two components provide the basic functions and operations to the database system whereas the other components deal with the open architecture design of the system. In the Interface Module, it contains the *Image Catalog Module* and the *Image Editor Module*. These two modules act as the interface for the system.

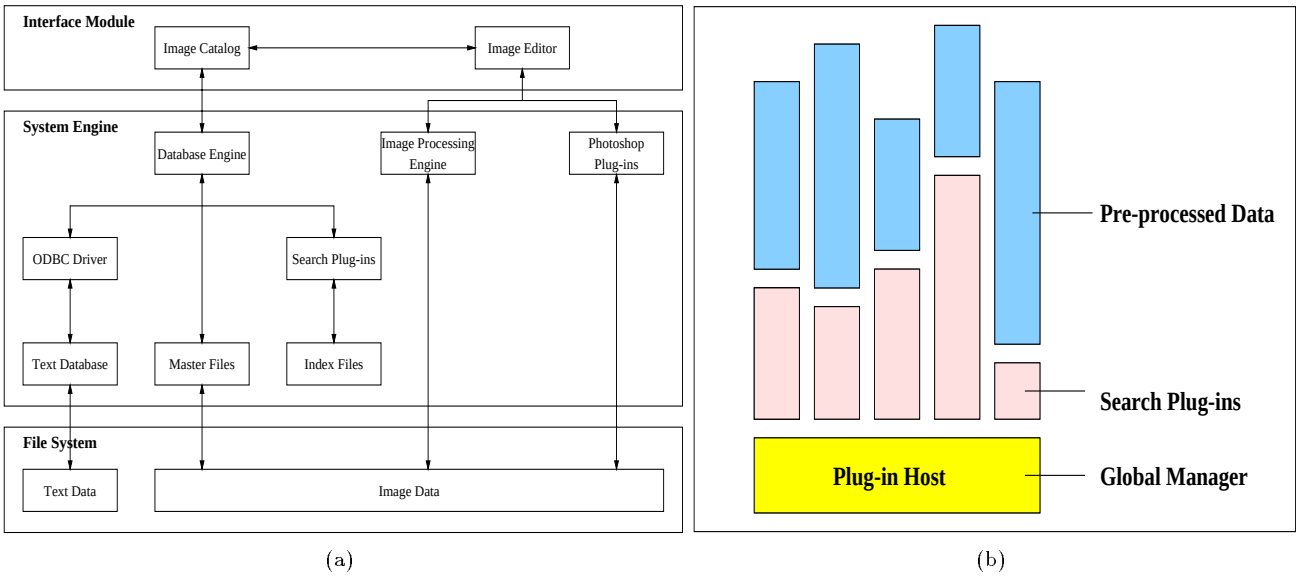


Figure 1: (a) The system architecture. (b) The structure of the Search Plug-ins. The Global Manager is the plug-in host. Pre-processed Data are the feature vectors in the index files.

2.1 Database Engine

The Database Engine manipulates the master files. Each master file contains the records of the images in the database. Each record contains the information of an image such as the location of the original image in the file system and the thumbnail image data of the image. The Database Engine is also responsible for image access of the database. When a new image is stored to the database, the engine creates an record for it and the text descriptors of the image are sent to the text database using the ODBC driver. It then calls the search plug-ins currently loaded into the system to perform pre-processing and indexing for the image. When there is an image retrieval request from a user, the engine sends the query to the corresponding Search Plug-ins. According to the result returned, the Database Engine finds the records of the images of the result and sends their information back to the user.

2.2 Image Processing Engine

Image Processing Engine is used to provide the basic image processing functions to the Image Editor such as image transformation, filtering, resizing, 2D object drawing, and color adjustment. The engine also helps the Database Engine

to extract thumbnail image data from the images.

2.3 ODBC and the Text Database

Open DataBase Connectivity (ODBC) [5] is a uniform interface standard used to access ODBC-compliant databases. By using a suitable ODBC driver, ODBC-compliant databases can be accessed by the same set of ODBC calls. In our system, we use ODBC to access the Text Database which is an ODBC-compliant database for non-image related processing and image query by keywords. This feature is provided since some users may already have their own database 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, by using ODBC our system gains a set of text-related operations. In short, our image database system complements other ODBC-compliant databases.

2.4 Plug-in Framework

To make the system extensible, maintainable, and flexible, we use the Photoshop Plug-in framework [9] in our system. The plug-in framework has two main components: *Plug-in host* and *Plug-in module*. We can enhance or customize application programs by means of plug-in modules. The *Global Manager Module* acts as our Plug-in host which loads the plug-in modules to the memory and calls them accordingly. Photoshop Plug-ins and the Search Plug-ins are in plug-in module format.

1. **Photoshop Plug-ins** : There are many Photoshop Plug-ins found on the market. They provide various image processing functions. We can enhance the image processing engine by adding these plug-ins to the system. As a result, we can reduce the functionality of the Image Processing Engine and leave most of the operations for the plug-ins. It makes the system more flexible because we can load suitable plug-ins to the system when we need them. Apart from those commercial plug-ins, we are able to write our own plug-ins to customize the Image Processing Engine.

2. **Search Plug-ins** : One search plug-in corresponds to one image searching method. By adding search plug-ins, new searching methods can be used by the system. Figure 1(b) shows the structure of the Search Plug-ins. Basically, each Search Plug-in is in charge of the following tasks:

   (a) **Extracting feature** : When a new image is stored to the database, the Database Engine calls all the loaded Search Plug-ins to extract specific features from the image for later retrieval.

   (b) **Indexing** : In order to reduce access time and narrow down the search space in image searching, each Search Plug-in is required to implement its own data structure for indexing of the feature vectors. In our project, we have implemented some indexing methods such as the R-tree [6], R*-tree [1], and VP-tree [17].

   (c) **Searching images** : The Search Plug-in calculates search keys for the queries of the searching method. By comparing the features of the images in the database with the search keys, the system produces the results of the queries.

   (d) **Providing user interface** : Each Search Plug-in provides a user interface to specify queries for the searching method. For example, a sketch pad dialog is used to enter a color-sketch-query as shown in Figure 5(a).

2.5 Image Catalog and Image Editor

The interface part consists of two main modules: *Image Catalog* and *Image Editor*.

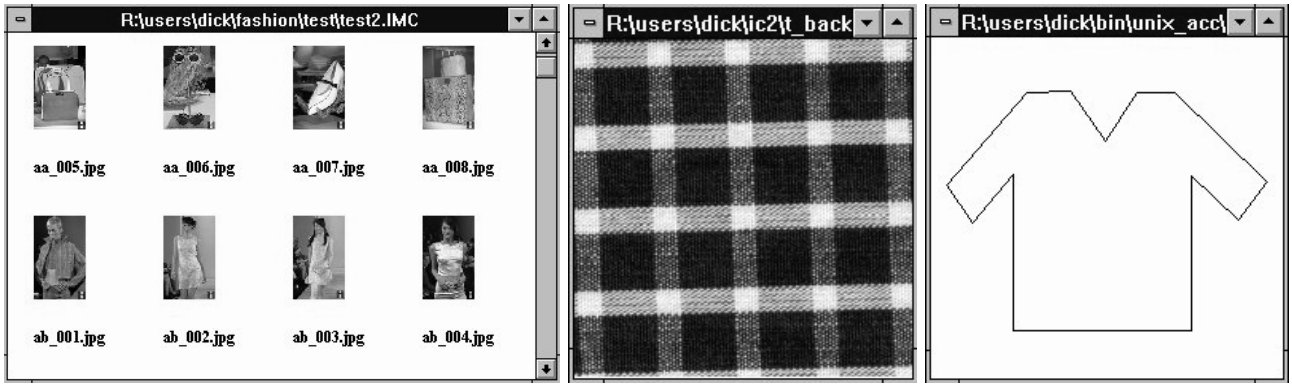
1. **Image Catalog** : Image Catalog supplies a user-friendly interface to access the image database. It provides functions and operations for manipulating images in the database. Images are displayed in catalog window and only the thumbnail images are shown on the screen in order to speed up the display. To create queries to retrieve images, Image Catalog calls the user interfaces query routine associated with the specific Search Plug-in. For example, a sketch pad dialog is called from the query-by-color-sketch Search Plug-in for the user to specify a color-sketch-query. The system creates a new catalog window for the result of the query.

2. **Image Editor** : Image Editor lets users to modify images in the database and create new images. All the functions provided by the editor are supported by the Image Processing Engine and the Photoshop Plug-ins.

   The Image Editor is different from other image-editing software since it supports both bitmap and vector images. Bitmap format images are made up of pixels. Images stored in the database are in bitmap format. On the other

hand, vector format images are made up of descriptions (attributes) of image elements. We use vector format to create new images because it is convenient to perform object-oriented operations such as ordering, selecting, and copying the graphical objects.

Under the global user interface of the system, we can open child windows for image catalog, bitmap image, and vector image simultaneously (Figure 2). This feature is very essential. For example, fashion designers can refer to the previous designs from the Image Catalog while they are creating new designs. We can also modify the images in the Image Catalogs. After we had finished editing the images, all the master files and the index files related to the images in the database system will be updated automatically.



(a) (b) (c)

Figure 2: Three types of child windows of the user interface. (a) An Image Catalog. (b) A bitmap format image. (c) A vector format image.

3. INTERNET ACCESS TO THE SYSTEM

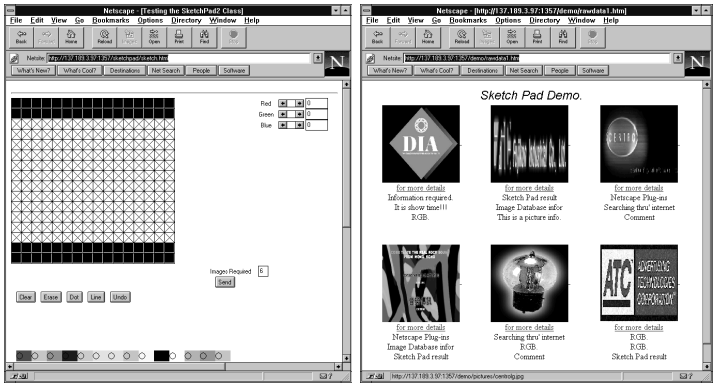
Apart from the system design described in Section 2, there is a server/client design enabling internet access to the system. By using Java applets and HyperText Markup Language (HTML), the system becomes platform independent. In any computer system, users may use a Java-capable World Wide Web browser to specify queries from the client side and retrieve images from the image database located at the server.

Based on the server/client design, the image database is located at the server side. We have written several Java applets for the user interface of each Search Plug-in. To input a query from the client side, the user can use a World Wide Web browser to execute the corresponding java applet and the user interface appears on the browser. The applet collects the query from client and sends it to the server. Then, the image database system searches the database to find out the result of the query. The result is produced in HTML format and is displayed back to the user. It includes the thumbnail images of the images which are similar to the target image of the query. If the user want to have the detail information of a particular image, he or she can click the image and the detail description of the image will be sent from the server to the client by means of HTML. Figure 3 shows the internet access to the system.

4. QUERY BY IMAGE FEATURES

4.1 Query-by-Color-Histogram

The system lets users search images by using color histogram. For each image from the database, we quantize its color space from $(2^8)^3$ to 512 levels by taking the three most significant bits of each color component (R, G, B). Then we partition the image into 9 non-overlapping partitions (3×3). For each partition, we build a color histogram and using the average of the histogram as the representative color. Then, each partition has 3 values R, G, and B. By collecting these values in all the partitions, a 27-dimensional feature vector of the images is obtained. We use the Nearest Neighbor Search [13] in the R-tree [6] for retrieving and indexing image feature vectors.



(a) (b)

Figure 3: Internet access to the system (a) Specifying query from the client side. (b) Result of the query.

4.2 Query-by-Color-Sketch

The query method described here is modified from the one proposed in [8]. It makes uses the regionalized color information of the image to search images. We first partition the image into 256 non-overlapping partitions (16×16). Then, we quantize the color space to 64 levels by taking the two most significant bits of each color component (R,G,B). For each partition, we find a representative color. There are two methods to find such color. One is using the average color of the partition and another is using the primary color. The average color c_{avg} of a partition with n pixels is given by $c_{avg} = \sum_{k=1}^{n} \frac{c_k}{n}$ where $c_1, ..., c_n$ are the quantized color of the pixels in the partition. The primary color of a partition is the mode quantized color of the pixels in the partition. By collecting the the average color for each partition, a 256-dimensional feature vector is obtained. Similarly, one other feature vector is obtained by using the primary colors.

The color sketch query is created from a sketch pad which is made up of 256 non-overlapping (16×16) small cells as shown in Figure 5(a). We calculate a 256-dimensional key vector of the query according to the color assigned in each cell. We then compare the key vector with the feature vectors of the images from the database to find out the result of the query. According to the users' choices, different sets of feature vectors (average or primary) are used in the searching. We use the Euclidean distance to measure the similarity of the image and the target image of the query. The shorter the distance is, the more the image is similar to the target image. The Euclidean distance D between the key vector $K(k_1, k_2, ..., k_{256})$ and the feature vector $F(f_1, f_2, ..., f_{256})$ is $D = \sqrt{\sum_{i=1}^{256}(k_i - f_i)^2}$.

4.3 Query-by-Shape

We use the two-stage polygon representation for shape matching proposed in [16]. It incorporates with two polygon matching techniques: Binary String Descriptor (BSD) [2] and Multi-Resolution Area Matching (MRAM) technique. In order to extract the object contour from an image, we have developed an object extraction technique based on Randomized Generalized Hough Transform [3]. The extracted object contour is then approximated into simple polygons for shape retrieval. For each pre-processed simple polygon defined in an image, we compute its Standardizing Binary String Descriptor (SBSD) and Multi-Resolution Area Information (MRAI) to obtain a feature vector as $(SBSD, MARI, image)$.

By comparing the SBSD of the polygons of the images in the database with the SBSD of the target polygon, those polygons having the same SBSD as the target polygon are selected as the candidate polygons. For every candidate polygon, we compare its MRAI with the one of the target polygon. The images containing objects with shape similar to the target polygon are chosen as the result of the query.

4.4 Query-by-Texture

This section describes how to extract features from texture images and how query-by-texture is carried out. For feature extraction, we use statistical methods to compute six features of each texture image in the database. We quantized the intensity space to 16 gray levels and find the intensity (gray level) for each pixel of the images. According to these

values, we find the six textural features of each image. Some of the features are computed from the the co-occurrence matrix [7]. For every image, we calculate four 16×16 normalized co-occurrence matrices with the step size = 1 and the direction $\theta = 45°$, $0°$, $-45°$, $-90°$, and the average of all four directions. For an image, these six textural features are defined in [15, 4] as below (Let $P(i,j)$ be the (i,j)-*th* entry in a co-occurrence matrix P).

1. **Smoothness** : The occurrence probability of gray level i in the image is given by $h(i) = \phi(i)/n$ where $\phi(i)$ is the number of pixels whose gray level is i in the image and n is the total number of pixels in the image. Then, the smoothness of the texture image is $SMOOTHNESS = (1 + \sum_{i=1}^{n}(i - \sum_{i=1}^{n} ih(i))^2 h(i))^{-1}$.
2. **Angular Second Moment (ASM)** : $ASM = \sum_{i=1}^{n}\sum_{j=1}^{n} P(i,j)^2$.
3. **Contrast** : $CONTRAST = \sum_{i=1}^{n}\sum_{j=1}^{n}(i-j)^2 P(i,j)$.
4. **Correlation** : $CORRELATION = (\sum_{i=1}^{n}\sum_{j=1}^{n} ijP(i,j) - \mu_x\mu_y)/(\sigma_x\sigma_y)$ where μ_x, μ_y, σ_x, σ_y are the means and standard deviations of the marginal-probability matrices p_x and p_y of P respectively.
5. **Inverse Element Difference Moment (IEDM)** : $IEDM = \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{P(i,j)}{(i-j)^2}, \quad i \neq j$.
6. **Entropy** : $ENTROPY = -\sum_{i=1}^{n}\sum_{j=1}^{n} P(i,j)\log P(i,j)$.

For each image, we calculate five feature values from the five co-occurrence matrices based on the six textural features associated with each matrix. The feature value F_k for an image with the k-*th* co-occurrence matrix is :

$$F_k = \lambda_1 \cdot SMOOTHNESS_k + \lambda_2 \cdot ASM_k + \lambda_3 \cdot CONTRAST_k + \lambda_4 \cdot CORRELATION_k + \lambda_5 \cdot IEDM_k + \lambda_6 \cdot ENTROPY_k$$

where $\lambda_1, ..., \lambda_6$ are the weighting factor of the above formula. By comparing the feature values of the images in the database with the one of the query texture image, images having smaller differences are selected as the result of the query.

5. SAMPLE DATABASES AND QUERIES

In this section, we demonstrate a fashion image database and a fabric image database on a 16M-RAM Pentium-90 PC using the Windows NT operating system.

We have implemented a fashion image database. Figure 4 shows a query by color histogram and Figure 5 demonstrates a query by color sketch. These two queries are performed on the fashion database which contains 340 images in JPEG format with sizes varying from 2K to 12K bytes. All the images are needed to be pre-processed to extract features first. It takes about 82 seconds for the query-by-color-histogram method and about 310 seconds for the query-by-color-sketch method to pre-process all the images. After the pre-processing step, any search can be done in a few seconds. For the query in Figure 4(a), the system spends only approximately 2 seconds to retrieve the 5 most similar images to the query image. For the query in Figure 5(a), it takes about 4 seconds in the searching of 5 most similar images.

We have also implemented a fabric database. Currently, the fabric database contains 120 fabric images. It is used for testing the query-by-texture method. We intend to collect more fabric samples from different textile manufacturers in Hong Kong to extend the size of this database so that it can be used for sourcing and merchandising of textile. Figure 6 shows a sample query-by-texture on the current fabric database. All the images in the database are with the same size (128×128 pixels) and same format (GIF). The system takes about 24 seconds to calculate all the textural features from the 120 images. After feature calculation, the system spends approximately 1 second to search 5 most similar images to the query texture.

6. CONCLUSION

We have designed and developed an image database system for the fashion, textile, and clothing industry in Hong Kong supporting feature-based retrieval by color histogram, color sketch, shape, and texture. The system provides an open architecture for the third-party developers by means of ODBC and plug-in framework. Moreover, we can edit both bitmap and vector format images by using the Image Editor Module. A server/client system design is used for internet access to our system. Based on the system design, we have implemented a fashion database and a fabric database to demonstrate feature-based image retrieval for managing a large number of images efficiently.

7. 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 Professor Ada Fu, Professor Laiwan Chan, Professor Lei Xu, Mr. Peter Chan, Mr. Tom Hung, Mr. Yan Zhang, and Mr. Eddie Cheng for their contributions to the system described here.

8. REFERENCES

[1] N. Beckmann, H. P. Kriegel, R. Schneider, and B. Seeger. "The R*-tree: an efficient robust access method for points and rectangles". *ACM SIGMOD*, 19(2):322–31, 1990.

[2] B. Bhavnagri. "A Method for representing Shape Based on an Equivalent Relation on Polygons". *Pattern Recognition*, 27(2):247–260, 1994.

[3] M. Fung, W. Lee, and I. King. "Randomized generalized Hough Transform for 2-D grayscale object detection". In *Proceedings of International Conference on Pattern Recognition*, volume II, pages B 511–515, 1996.

[4] R.C. Gonzalez. *"Digital Image Processing"*. Addison Wesley, 1992.

[5] R. Gryphon. *"Using ODBC 2"*. QUE, special edition, 1995.

[6] A. Guttman. "R-tree: A Dynamic Index Structure for Spatial Searching". *ACM SIGMOD*, 14(2):47–57, 1984.

[7] R. M. Haralick, K. Shanmugam, and I. Dinstein. "Textural features for image classification". *IEEE Trans.*, SMC-3:610–621, 1973.

[8] C. E. Jacobs, A. Finkelstein, and D. H. Salesin. "Fast Multiresolution Image Querying". In *Proceedings of SIGGAPH 95*, pages 277–286, 1995.

[9] T. Knoll. "Adobe Photoshop Software Development Kit Version 3.0.5", February 1996.

[10] 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.

[11] V.R. Ogle and M. Stonebraker. "Chabot: Retrieval from a Relational Database of Images". *Computer*, 28(9):40–8, 1995.

[12] A. Pentland, R. W. Picard, and S. Sclaroff. "Photobook: Content-Based Manipulation of Image Databases". Technical Report No. 255, M.I.T. Media Laboratory Perceptual Computing, November 1993.

[13] N. Roussopoulos, S. Kelley, and F. Vincent. "Nearest Neighbour Queries". *ACM SIGMOD*, 24(2):71–9, 1995.

[14] J.R. Smith and S. F. Chang. "VisualSEEk: a fully automated content-based image query system". [To be appeared in ACM Multimedia '96].

[15] F. Tomita and S. Tsuji. *"Computer analysis of visual textures"*. Kluwer Academic Publishers, Boston, 1990.

[16] L.H. Tung, I. King, P.F. Fung, and W.S. Lee. "Two-Stage Polygon Representation for Efficient Shape Retrieval in Image Databases". In *Proceedings of the First International Workshop on Image Databases and Multi-Media Search*, pages 146–153, 1996.

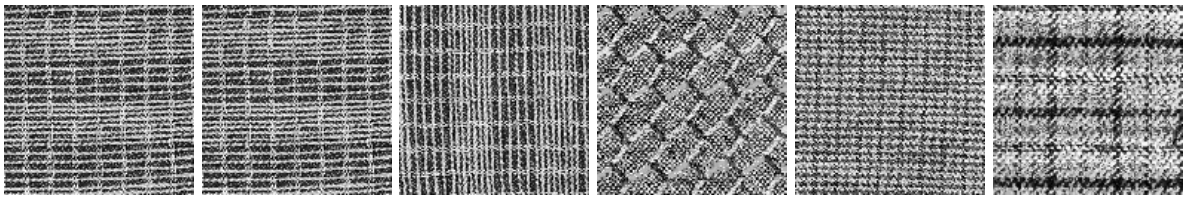
[17] P. N. Yianilos. "Data structures and algorithms for nearest neighbor search in general metric spaces". In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 311–321, January 1993.

(a) (b) (c) (d) (e) (f)

Figure 4: Query-by-color-histogram. (a) The query image. (b)-(f) The result of the query. (b) is the most similar image to (a) and then (c), (d), (e), and (f).

(a) (b) (c) (d) (e) (f)

Figure 5: Query-by-color-sketch. (a) A color sketch query. (b)-(f) The result of the query. (b) is the most similar image to (a) and then (c), (d), (e), and (f).



(a) (b) (c) (d) (e) (f)

Figure 6: Query-by-texture. (a) The query texture image. (b)-(f) The result of the query. (b) is the most similar texture image to (a) and then (c), (d), (e), and (f).