Boston University
Graduate School of Arts and Sciences
Computer Science Department

Masters Project

Image Feature Extraction Subsystem of the ImageRover WWW Image Search System.

by
Leonid Taycher

August 1, 1997

Approved by:

____
Professor Stan Sclaroff, Computer Science Department

Submitted in partial fulfillment of the requirements
for the degree of Masters of Arts in Computer Science

Abstract

The focus of this project was to modify the image feature extraction subsystem of the ImageRover system[11]. The subsystem was extended to use color and texture measures which more closely correspond to the human perception. The feature implemented for color is the color histogram in $L^*u^*v^*$ color space. The texture measure implemented is $2D$ Wold decomposition, which incorporates the three most important for human perception measures, “directionality”, “periodicity” and “randomness.” Texture module performance was evaluated on the Brodatz texture album. The resulting modules are now part of the ImageRover system.

1 Introduction

In the last several years, as more and more information became available on the Internet (especially on World Wide Web), the main problem of the users shifted from existence of needed information to the ability to find it.

One of the solutions for this problem is the creation of a comprehensive index of all the available documents. Manual creation of such an index is impossible because the number of the documents on the WWW is estimated at 50-100 million. Therefore an automatic indexing scheme is needed. Such indices can be created by using software robots or spiders that traverse parts of the Web storing the information about the documents found.

There exist several well known indices such as Lycos, HotBot, Altavista etc, that provide keyword text search on the collected documents. They do so by precomputing the keywords for each document processed, creating an index and then searching it for the keywords entered by user. While having an index for the text documents on the Web is very useful, there are about 20 to 30 percent of the Web documents that cannot be indexed using text based techniques: images, sounds and videos.

Image search engines would enable simpler access to such collections of non-text data as museums, medical and other scientific data collections, on-line catalogs, etc.

While some information about the image content can be extracted from the associated text (such as image's **URL**, text from the **ALT** attribute of the **IMG** tag, and the text surrounding an image in the HTML document) it is not enough to provide a good text-based image search engine, since the complete information about the images cannot be extracted from such text, and the image usually may not be completely described in words. Therefore, we need an engine that provides ability to dynamically search based on image content.

The main problem with creating the image search engine is that the user has to provide the description of what is needed. This description may be in words, but the image content may not always be easily described using words. Another problem with this approach is that to provide the keyword based search using automated indexing system would require solving the image understanding problem.

Alternative is to allow user to specify what he is looking for using a weighted sum of precomputed image features which roughly correspond to human perception, which are computed offline (just as keywords for the text documents are). The commonly used features are color and texture measures. This way, the image search would be reduced to finding images with the feature vectors which are close to the feature vectors of the examples, an analog of "image grep".

One problem with this approach is that sometimes user may want to find images which are similar to the example only in part of the features. Alleviating this problem requires introducing relevance feedback, where user may interactively steer search in the correct direction by selecting more and more positive and negative examples. Thus on each iteration the system is looking for the images having features that are present in the positive example and not having features of negative examples.

Such a system, named ImageRover, was previously built and tested in the BU Image and Video Computing Group. It currently provides the ability to search a sample database of about 15,000 images on the Web using QBE (Query By Example) interface with similarity measures based on color and texture orientation distributions.

The focus of this project was to improve the color and texture measures to provide for the better correspondence between human perception and the features computed. To this end, the RGB based color histogram calculation was replaced by $L^*u^*v^*$ based one, since distances in $L^*u^*v^*$ color space correspond to human perception more closely than those in RGB. The simple texture orientation histogram measure was supplemented by periodicity and "randomness" measures obtained by using by 2-D Wold transform[5].

2 Related work

There already exist several popular systems which provide QBIC (Query by Image Content) searches. These systems fall into one of two categories.

- Locally based indices, which provide index for a relatively small number of images (several thousands);*e.g.*, IBM QBIC, Virage, Cypress, VisualSeek, Jacob and Excalibur Visual RetrievalWare.

- WWW based indices, which attempt to allow users to search parts of the Web;*e.g.*, Lycos media search tool, Yahoo ImageSurf, WebSEEk and WebSeer. ImageRover belongs to this category as well.

The systems of the second kind provide mainly *keyword based search*, *supplementing* it with search by the color distribution (WebSEEk, WebSeer, Yahoo Image Surfer), by the number of faces and some file information (WebSeer) or by the texture information. Both WebSEEk and Yahoo ImageSurf provide a category based tree (*e.g.* in the Yahoo Image Surfer, to looks for images of actresses, you can select *Entertainment:Movies and Films:Actors and Actresses* category), which allows users to narrow the range of images to be searched by the machine vision techniques used by the system(color, texture, faces, etc.) to a manageable number. On the other hand Lycos Media Search and WebSeer provide straight keyword search, which is supplemented by the image content data in WebSeer.

ImageRover falls into the second category, but in contrast to other systems it does not currently provide keyword based search, relying entirely on image content based one.

2.1 Interface

Most of the systems mentioned above (with the exception of Lycos Media search) use the general image content search interface methods.

The common interface is as follows: first the user narrows down the search using keyword based techniques, then he is provided by either a random (Yahoo and WebSEEk) or presorted (WebSeer) list of the images which correspond to the chosen keywords/category. The user can then select one of the images from the list to search for the images which are similar to it using image content data. The user can also request that a new selection of random images from the category be presented(Figure [2.1].)

3 ImageRover

We have already implemented a pilot system called ImageRover. The goal of the ImageRover project was to create a WWW search engine in which the searches would be based mainly on the image content, and possibly supplemented by the text data.

A system of this magnitude can be logically separated into several stages (or subsystems) that can be implemented separately:

- Image Collection
- Feature Extraction

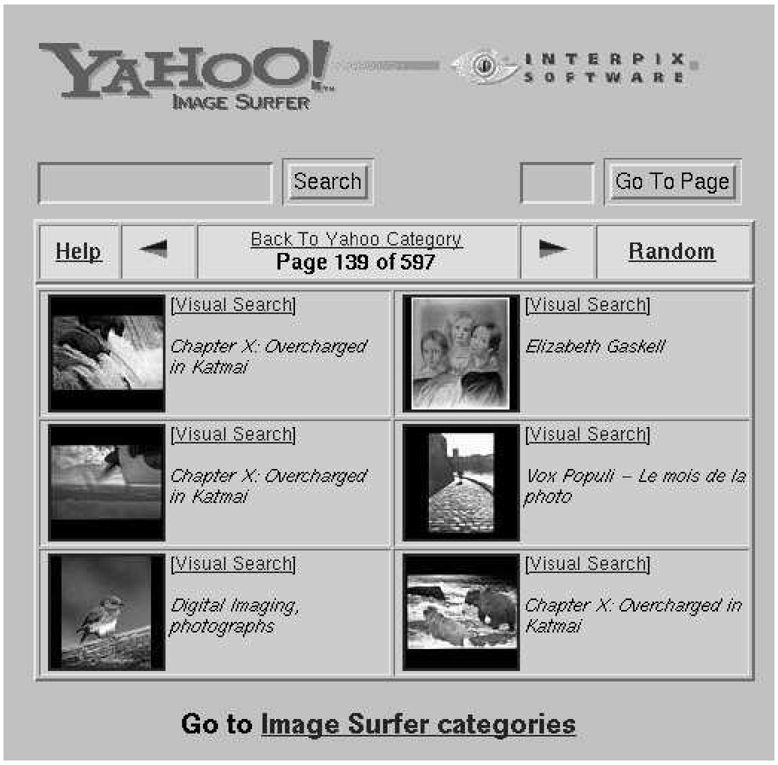

Figure 2.1: Yahoo ImageSurf Interface

- Image Database
- Query Server
- User Interface

The first three stages can be performed offline. The **Image Collection** stage consists of robots which traverse the WWW and collect the images for further processing. The **Feature Extraction** stage computes the decompositions on the raw image data and produces feature vectors which are then stored in the **Image Database**. Indexing and search have to performed online. They are performed by **Query Server** which provides fast access to the database index and **User Interface** (currently HTML form and CGI script) which processes user queries using Query Server and provides relevance feedback. These parts of ImageRover system are explained in more detail below.

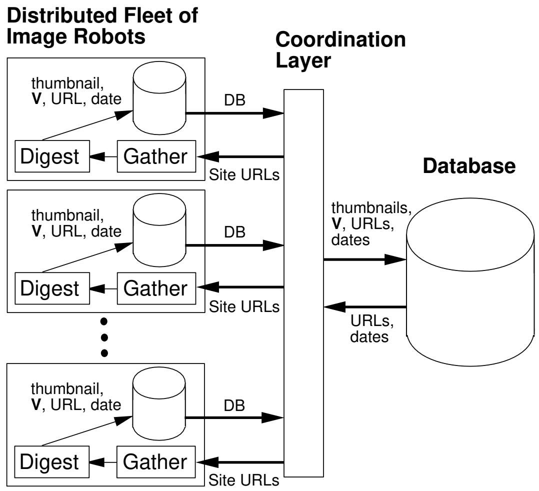

Figure 3.1: Image robot subsystem diagram. The image collection subsystem utilizes a distributed fleet of WWW robots running on different computers. Robots can contain *collection modules*, *digestion modules*, and a local database. The collection modules recursively parse and traverse WWW documents, gathering images. The digestion modules then process these images to extract needed image indexing information and to compute a reduced resolution thumbnail image. The robots are dispatched and coordinated via a separate *coordination layer*, which also manages updates of the image index database.

3.1 Image Collection Subsystem

The image collection subsystem has the task of retrieving images from the remote servers. Since the estimated time to retrieve one image is 83 seconds (the estimate includes time necessary to process non-image documents), it is impossible to retrieve a significant portion of WWW images using a single computer. To alleviate this the subsystem constists of multiple, possibly multithreaded robots running on multiple computers. The work of the robots is coordinated by the coordination layer, which dispatches them and maintains image index database. The structure of the subsystem is shown in Figure 3.1. The output of this subsystem is the raw image data and the text data from the HTML documents to be used for creating text based index. The first version of robots was implemented in `Perl` and the current version is in `C++`. Both versions were implemented by me.

3.2 Image Feature Extraction Subsystem

This subsystem takes as input raw image data and produces feature vectors which are further produced in the image indexing subsystem. Implementation of this subsystem was the focus of the project and is explained in detail in Section 4.

3.3 Image Database Subsystem

This subsystem takes as an input a collection of the feature vectors of the images. It then performs dimensionality reduction on the vectors, since the *intrinsic dimension* of the data is significantly less than the length of the feature vectors (for example many of the bins in the color histogram are almost never used). This is done using principal components analysis(PCA)[9] on each of the subvectors. A subvector is an elementary feature *e.g.* a color histogram for the center region of the image. Then, an indexing structure is constructed based on optimized k-d tree[1]. The goal is to minimize the expected number of records examined with the search algorithm. The output of the subsystem is an optimized index to be accessed by the user interface subsystem.

3.4 Query Server and User Interface Subsystems

The user queries are processed by an CGI script and the Query Server. The Query Server process holds in its memory (preferably in core memory) the index for the image database, and performs searches on it given the original vector and norms to be used and weights to be given to different subvectors.

A CGI script works the following way:

1. A selection of the images is presented to user. It can be either random, or a result of a previous search.

2. When user selects some of the images, the average feature vector is calculated based on them. Also, the script calculates standard deviations for each subvector. The norms supplied to the Query Server are selected to minimize the standard deviations, and the weights are selected to inverse of standard deviations.

3. The new search is performed using the Query Server and the new selection is presented to the user, consisting of the images which were

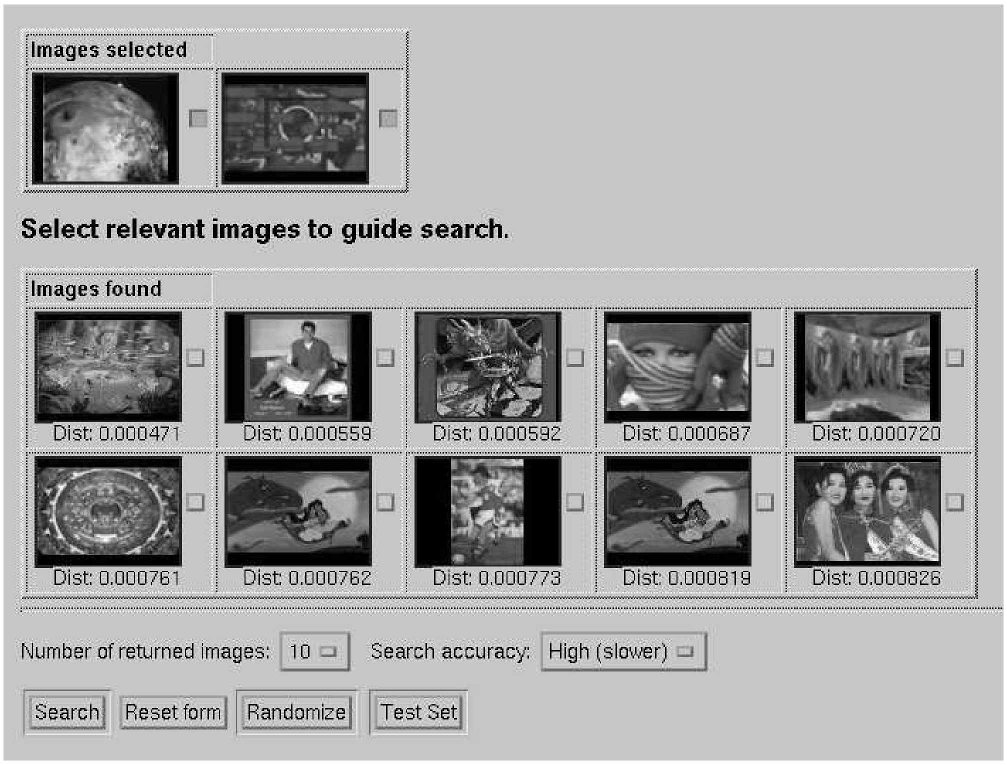

Figure 3.2: ImageRover Interface

returned by Query Server. An example of the search is shown in (Figure [3.2]).

The these two subsystems were written by Marco La Cascia.

4 Image Feature Extraction Subsystem Implementation

The current implementation of ImageRover uses the distances between the feature vectors produced by this subsystem to determine the similarity of the images. So in order for the system to be useful for humans, it should utilize the features for which the distance between extracted feature vectors corresponds to the human notion of similarity.

Both color and texture measures used in the previous implementation of the subsystem were not optimal for this purpose. The purpose of this project was to introduce color and texture features which more closely correspond to human perception. Thus the RGB color space based measure was replaced by CUE $L^*u^*v^*$[3] color space based one, distances in which closely correspond to human perception[16]. Also the directionality based texture measure was supplemented by periodicity and statistical measures using the 2-D Wold transform[5], since these features were identified as most important for humans[12].

The subsystem uses WebVision library written by me for this project, which allows reading images in GIF and JPEG formats (the two most popular image formats on the WWW), and provides such image operations as color space conversion as well as operations on separate planes, such as convolution. It also contains template based vector and matrix implementations with most of the common operations defined.

The subsystem is implemented as a collection of modules which take as input an image (grayscale or color depending on the feature) and the collection of the region definitions, and output the features computed for each of the regions specified. Currently six overlapping regions are used: the whole image, upper left, upper right, lower left, lower right and the center. The details of computation of the features are explained below.

4.1 Color Measure

The color feature extraction module computes the color histogram of the image in $L^*u^*v^*$ color space. RGB values are first transformed into CIE XYZ values and then into $L^*u^*v^*$ by the following equations:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.607 & 0.174 & 0.200 \\ 0.299 & 0.587 & 0.114 \\ 0.000 & 0.066 & 1.116 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{4.1}$$

$$L^* = \begin{cases} 25 * (100 * \frac{Y}{Y_0})^{\frac{1}{3}} - 16 & \text{if } \frac{Y}{Y_0} \geq 0.008856 \\ 903.3 \frac{Y}{Y_0} & \text{otherwise} \end{cases} \tag{4.2}$$

$$u^* = 13L^*(u' - u'_0) \tag{4.3}$$

$$v^* = 13L^*(v' - v'_0) \tag{4.4}$$

$$u' = \frac{4X}{X + 15Y + 3Z} \tag{4.5}$$

$$v' = \frac{9X}{X + 15Y + 3Z} \tag{4.6}$$

where the $RGB \rightarrow XYZ$ conversion matrix corresponds to CIE Illuminant C (Overcast sky at noon), which was deemed most appropriate[16], and values of X_0, Y_0, Z_0, u'_0 and v'_0 (X_0, Y_0, and Z_0 are used to compute u'_0 and v'_0) are calculated for the reference value of white under the same illuminant.

The ranges of L^*, u^*, and v^* (based on [0..1] RGB range) are: L^* : $[L^*|_{(0,0,0)} L^*|_{(1,1,1)}]$, u^* : $[u^*|_{(0,1,0)} u^*|_{(1,0,0)}]$, and v^* : $[v^*|_{(0,0,1)} v^*|_{(0,1,0)}]$.

In order to obtain a histogram, each axis was split into 4 equal sized bins (64 bins total). After calculating the number of pixels which fall into each bin, the histogram was normalized to have unit sum and smoothed. Over the six regions currently defined, the computed feature vector has the length of 384.

4.2 Texture Measures

The texture measure implemented in this project is 2-D Wold decomposition. The feature vector extracted by this transform is the combination of the feature vectors from structured (directionality and periodicity), and unstructured (MRSAR) texture measures, and the weighting between them based on the importance of each feature for the image.

To determine the relative importance of structured and unstructured features we use the fact that autocorrelation functions of periodic patterns are also periodic, while most of the power of autocorrelation of random texture is concentrated near zero displacement. To determine if the texture is structured, we compute the "small displacement" region[5](Figure 4.1).

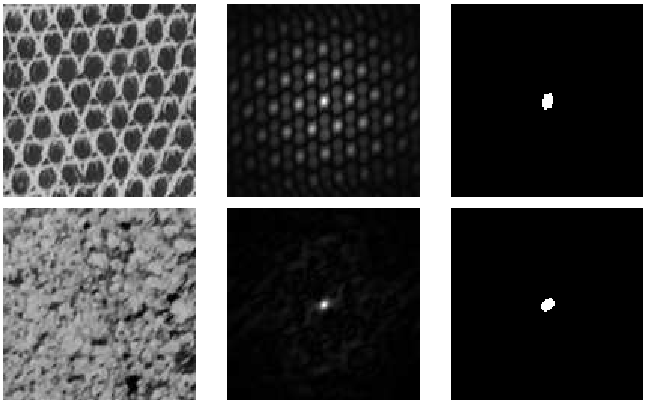

Figure 4.1: From left to right: the original Brodatz textures; their autocorrelation function; their "small displacement" regions. First row D22:. The high energy regions are spread over the entire displacement plane. Second row D29:. Most of the energy concentrated in the small region around the zero displacement.

The region is computed on the image autocovariance plane (obtained by the inverse DFT of the image power spectrum) by expanding from the zero displacement, until the value of the autocovariance function is lower then a small portion of the function range(in our system 10%). The energy of the region is then computed. If the texture is structured, then the ratio between the energy contained in the "small displacement" region and the total autocorrelation power will be small, but it will be large for unstructured textures (Figure 4.1).

In our system, all textures with the ratio less then 0.10 were considered to be completely structured, textures with the ratio greater the 0.32, to be completely unstructured, and for the textures with the ratio between these two values both structured and unstructured matter.

The thresholds were determined as follows. A collection of textures was hand classified into structured and unstructured categories. The distribution in each category was assumed to be Gaussian, and the mean and the variance for them were computed. For the structured textures the results were $\mu_s = 0.1$, $\sigma_s = 0.14$, and for the unstructured textures, $\mu_u = 0.32$, $\sigma_u = 0.42$. The weights of structured and unstructured features for the textures with the energy ratios between μ_s and μ_u were set to be the probabilities of the texture being structured or unstructured obtained from above distributions, and then scaled to have a unit sum.

Directionality Texture Measure

The directionality measure for a region of an image is computed using steerable pyramids[2]. For each level of the pyramid a histogram of orientation strengths of all pixels in the level is computed. The orientation and its strength at each pixel is calculated as the combination of outputs of three G_2 and four H_2 filters, which together constitute a steerable basis[2]. The filters used are:

$G_{2a} = 0.9213(2x^2 - 1)e^{-(x^2+y^2)}$	$H_{2a} = 0.9780(-2.254x + x^3)e^{-(x^2+y^2)}$
$G_{2b} = 1.843xye^{-(x^2+y^2)}$	$H_{2b} = 0.9780(-0.7515 + x^2)e^{-(x^2+y^2)}$
$G_{2c} = 0.9213(2y^2 - 1)e^{-(x^2+y^2)}$	$H_{2c} = 0.9780(-0.7515 + y^3)e^{-(x^2+y^2)}$
	$H_{2d} = 0.9780(-2.254y + y^3)e^{-(x^2+y^2)}$

Then the oriented energy at the pixel is $E_{G_2H_2} = C_1 + C_2\cos(2\theta) + C_3\sin(2\theta) + h.o.t$, where

$$\begin{aligned}
C_1 &= 0.5[G_{2b}]^2 + 0.25[G_{2a}][G_{2c}] + 0.375([G_{2a}]^2 + [G_{2c}]^2) + \\
&\quad + 0.3125([H_{2a}]^2 + [H_{2d}]^2) + 0.5625([H_{2b}]^2 + [H_{2c}]^2) + \\
&\quad + 0.375([H_{2a}][H_{2c}] + [H_{2b}][H_{2d}]) \\
C_2 &= 0.5([G_{2a}]^2 - [G_{2c}]^2) + 0.46875([H_{2a}]^2 - [H_{2d}]^2) + \\
&\quad + 0.28125([H_{2b}]^2 - [H_{2c}]^2) + 0.1875([H_{2a}][H_{2c}] - [H_{2b}][H_{2d}] \\
C_3 &= -[G_{2a}][G_{2b}] - [G_{2b}][G_{2c}] - 0.9375([H_{2c}][H_{2d}] + [H_{2a}][H_{2b}]) - \\
&\quad - 1.6875[H_{2b}][H_{2c}] - 0.1875[H_{2a}][H_{2d}]
\end{aligned}$$

The dominant orientation angle at a point is $\theta_d = \frac{\arg[C_2,C_3]}{2}$, and the strength of this orientation is $S_{\theta_d} = \sqrt{C_2^2 + C_3^2}$

The direction histogram is then computed by ignoring the pixels where the strength is less than some threshold (0.005 in the current implementation). The histogram range $[-\frac{\pi}{2}..\frac{\pi}{2}]$ is quantized to 16 bins. The histogram is then circularly smoothed. Over the six image regions currently defined(Figure 8.1), the computed feature vector over four scales has the length of 384.

The approach used in ImageRover differs from that proposed by Gorkani and Picard [4], in that they first extract peaks from the histogram and compute image similarity based on peak-to-peak distances. In ImageRover the histograms are compared directly via histogram distance, thus avoiding problems with direct peak extraction and matching.

Periodicity Texture Measure

The measure of the texture's periodicity is computed as a histogram of power of the harmonic peaks in the discrete Fourier transform of an image at different orientations[5]. The histograms are computed at four scales. This is accomplished by converting the input image to grayscale, computing a Gaussian pyramid for it, and then analyzing the harmonic structure of each level. Each level in the pyramid is normalized to zero mean, and its discrete Fourier transform magnitude image is computed. Next, a list of peak locations and magnitudes is computed. Peaks are defined as the local maxima of DFT magnitudes, and are found by searching a 5×5 neighborhood of each sample.

The peak list is then cleaned up by deleting all peaks which do not have supporting peaks (peaks of the same angular direction but different frequency). Following [5], a peak is considered to be harmonic if it is either (a) fundamental *i.e.* its frequency can be used to linearly express frequencies of other peaks, or (b) harmonic *i.e.* its frequency can be expressed as a linear combination of fundamental peaks.

The histogram of peak powers is then computed over 16 direction from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$. It is normalized by the number of pixels in the image region, and circularly smoothed. Over the six image regions currently defined (Figure 8.1), the computed feature vector over four levels has the length of 384.

Unlike the original implementation in [5] based on inter-peak distances for comparison, ImageRover compares the normalized histograms of harmonic peak magnitudes, which offers robustness to changes in scale.

Unstructured Texture Measure

The unstructured texture measure used in this system are the average MRSAR[10] coefficients.

The value of the pixel $p_{i,j}$, at image position (i, j) is modeled as linear function of neighboring pixels:

$$p_{i,j} = c_1(p_{i-l,j}+p_{i+l,j})+c_2(p_{i,j-l}+p_{i,j+l})+c_3(p_{i-l,j-l}+p_{i+l,j+l})+c_4(p_{i-l,j+l}+p_{i+l,j-l}),$$

where c_k are the coefficients and l determines the resolution of the pixel neighborhood. This can be rewritten as $p_{i,j} = P_{i,j}[c_1 c_2 c_3 c_4]^T$

The solution for MRSAR coefficients at the pixel $p_{i,j}$ is under-determined, so coefficients are estimated over an $n \times n$ window centered at (i, j) using least square fitting. In ImageRover system, $n = 21$ and the neighborhood sizes used are 2, 3, and 4. Since computing MRSAR coefficients is computationally expensive, they are computed over 400 evenly distributed windows, and the coefficients in the rest of locations are estimated from them.

For each region, at each neighborhood size, four MRSAR coefficients are estimated as the average of MRSAR coefficients of all pixels in the region. Also the least square error (LSE) is estimated. Since for each region the module provides a 15 element vector (a, b, c, d, and σ at 3 resolutions), for six image regions currently defined the overall length is 90.

The code used in this module was adapted from work by Thomas P. Minka and Rosalind Picard (MIT Media Lab).

5 Performance Evaluation

5.1 The Experiment Setup

It is difficult to evaluate the performance of the image feature extraction subsystem in the context of ImageRover, since there is no simple way to quantify image similarity in the large database. So the texture feature extraction modules were evaluated on the subsets of the standard Brodatz texture album.

The test databases were created by selecting a subset of Brodatz textures (512×512 images were used), and splitting each image into 16 128×128 nonoverlapping parts. The parts of the same original were considered to be similar, and parts of different originals to be dissimilar. So the subset used in the experiments could not contain nonhomogenous textures, such as `D72:Tree stump used as a chopping block` or similar textures such as `D3:Reptile skin` and `D22:Reptile skin` without effecting test results.

For each image the feature vector was computed over five image regions and the whole image as described in Section 8. The measure of similarity between two images was chosen to be the sum of weighted Euclidian distances between corresponding subvectors.

The results of experiments were evaluated using standard Recall vs. Precision graphs, where

$$\text{Recall} = \frac{\text{Number of relevant items retrieved}}{\text{Number of relevant items in database}}, \tag{5.1}$$

$$\text{Precision} = \frac{\text{Number of relevant items retrieved}}{\text{Number of items retrieved}} \tag{5.2}$$

and $AVRR/IAVRR$ ratio[14]. The $AVRR$ measures the average rank of the relevant item in the result of the query, and $IAVRR$ is the ideal average rank of the relevant item, which in the case of all relevant sets having the same size n is $n/2$, in our case $16/2 = 8$.

The conducted experiments are described in detail below.

5.2 Periodic Texture Experiment

In this experiment I tested the performance of the system on the structured textures. The 12 textures used for the experiment are shown in Figure 5.1. The number of images in the texture database was 192. In general the system performed well with the $AVRR/IAVRR$ ratio 3.1($AVRR/IAVRR = 12$ for random retrieval), and the recall vs. precision graph shown in Figure 5.3. The system failed in several cases. An example of failure is mixing up of parts of D101 and D52 (Figure 5.3, this happened because both are highly structured, and have very similar structured properties (both have four similar dominant directions: vertical, horizontal and two diagonal and both have same periodic structure).

5.3 General Texture Experiment

The second experiment tested the performance of the system on a more general subset of textures. The 40 textures used in this experiment are shown in Figure 5.2. The number of images in the database was 640. The performance of the system was generally worse then in the first experiment with $AVRR/IAVRR$ ratio of 15.0 ($AVRR/IAVRR = 40$ for random retrieval), and the recall vs. precision graph shown in Figure 5.4.

Some of the failures in this experiment may be explained by incorrect classification of the textures as structured/unstructured (*e.g.* D19 and D110 were misclassified as highly structured and D1 was misclassified as having strong structured and unstructured components). The source of these errors is the current procedure for formation of “small displacement” region. The

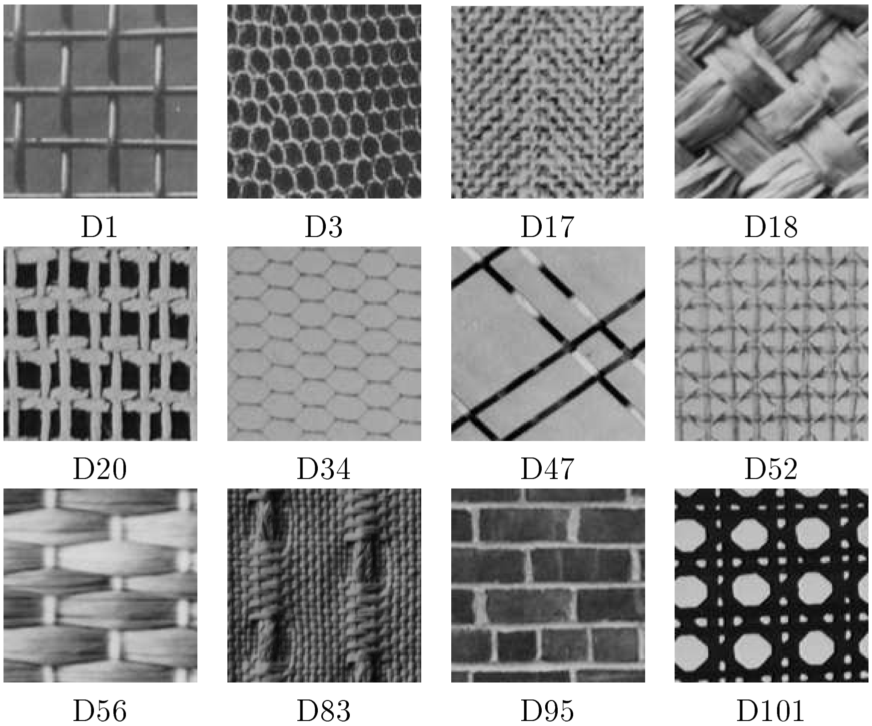

Figure 5.1: The subset of Brodatz Textures used for periodic texture experiment.

algorithm used in the project assumes the existence of deep (with values smaller than some threshold) valleys between the peaks of the autocorrelation function. Therefore if the valley around the central peak has values larger than this threshold, the small displacement region is extended to include this valley and the neighboring peaks. In this case the power ratio will be high, and the texture will be assumed to be unstructured. This can happen in the images similar to D1, with high contrast high texture areas, and the low contrast background. On the other hand if an autocorrelation function is very flat with one small peak, then only a small part of the central peak will be extracted as a "small displacement" region, the power ratio will be low, and the texture will be assumed to be structured. This may happen is the image is unstructured but has very high contrast. If this happens, the autocovariance function will have a very sharp peak at the origin, and the rest of the function will be essentially flat.

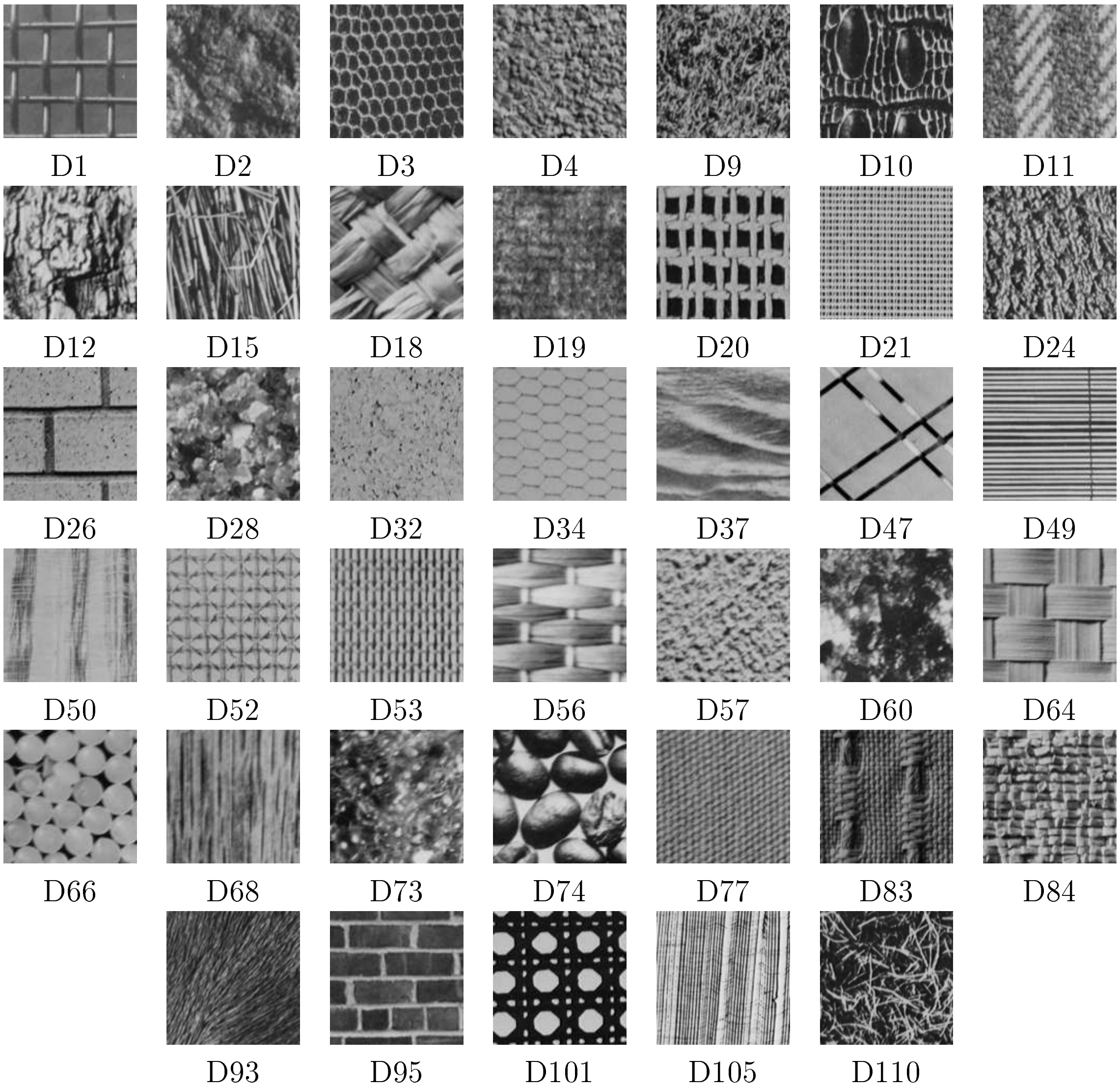

Figure 5.2: The subset of Brodatz Textures used for General Texture Experiment.

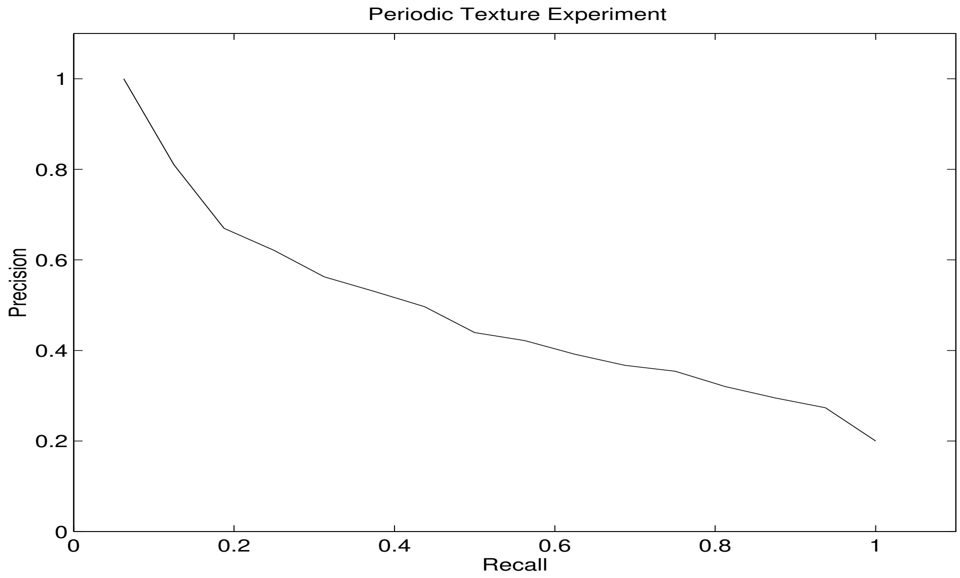

Figure 5.3: The Recall vs. Precision graph for Periodic Texture Experiment.

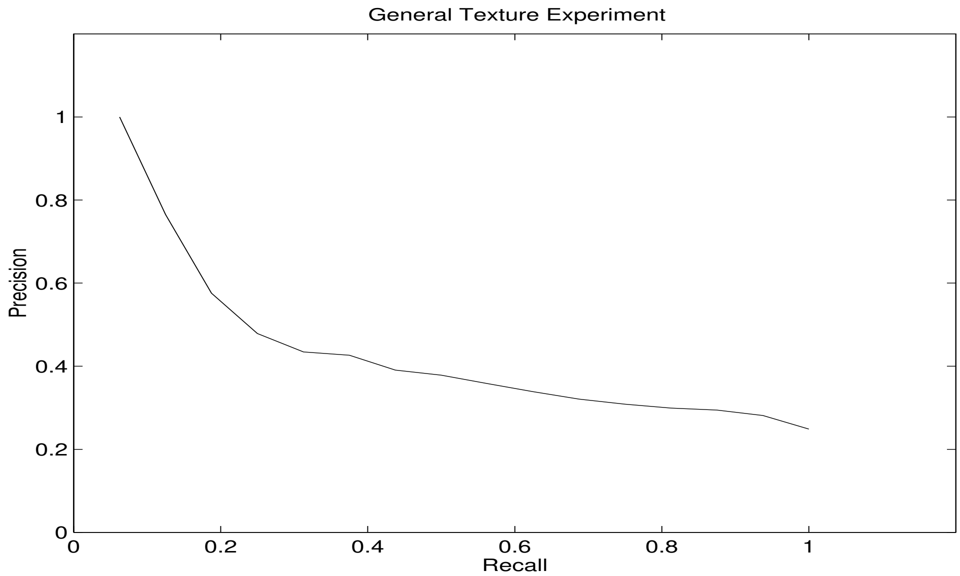

Figure 5.4: The Recall vs. Precision graph for General Texture Experiment.

6 Future work

The image feature extraction system in it's current form extracts only global features which do not correspond to objects in the image (which are very important for humans). Thus the system can be improved in different ways.

- The current feature extractors may be improved by using the same pyramid based structure in all texture feature modules.

- The $L^*u^*v^*$ color histograms may be supplemented by graylevel histograms. The importance of each histogram may be determined offline by the amount of "grayness" (relative density on pixels around black-white axis) in image.

- Instead of using average MRSAR coefficients, their histogram may be computed (if it is possible to obtain practical bounds on their size)

- A number of new decompositions which do not "know" about the object composition of the image can be introduced such as **symmetry features measure** based on [13] which allows to better classify color textures, since it associates texture and color features of the patch. Another such decomposition **color correlogram feature**[17] which summarizes the color distributions in the image.

- The system may also be extended by including "object aware" decompositions such as blob-world-like segmentation with relevant blobs being selected based on the relevance feedback based on [15].

- Another "object aware" module which may be included in the ImageRover system is eigen-faces based module which provides either a vector of number of faces and their positions in the image, or, if a face takes up more than 30% of the image (a portrait) a vector which characterizes the face.

- The interface may be improved to include a sketch tool to allow users to draw the image which is going to be used as a search seed.

- A supplemental keyword search, based on the words from surrounding hyper-text may be added to improve the search.

7 Summary

This project extends the prototype ImageRover system's image feature extraction subsystem by replacing old color and texture measures. The measure used for color is $L^*u^*v^*$ based histogram (replacing RGB based one). The measure used for texture is $2D$ Wold transform, which incorporates MRSAR[10] based "randomness" measure, Fourier peak based[5] "periodicity" measure in addition to previously used "directionality" measure which uses steerable filters[4]. The texture measure also incorporates the structured vs. unstructured weighting based on [5].

The performance of the texture measures was tested using the images from standard Brodatz Texture Album (Section 5). The experiments show the viability of the subsystem.

8 Appendix

The regions defined on the image are as follows (Figure 8.1):

0 Whole image

1 Middle quarter

2 Top Left quarter

3 Top Right quarter

4 Bottom Left Quarter

5 Bottom Right quarter

The output feature vector of the subsystem is (with the size in floating point numbers shown in parentheses):

Color data(384)	Texture data(870)

,

where

Color Data			
Region 1(64)	Region 2(64)	...	Region 6(64)

,

Texture Data			
Region 1(145)	Region 2(145)	...	Region 6(145)

,

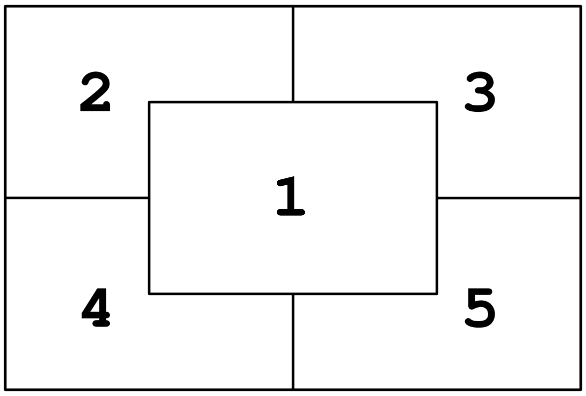

Figure 8.1: The image subregions defined in ImageRover.

Texture Region Data				
sw(1)	uw(1)	MRSAR Coeffs(15)	Directionality histos(64)	Periodicity histos(64)

,
where *sw* is the relative "structuredness" of the region and *us* is its relative "unstructuredness", $sw + uw = 1.0$.

The multiscale data is located in the feature vectors from highest to lowest resolution.

Bibliography

[1] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A.Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pages 573–582, 1994.

[2] W.T. Freeman and E.H. Adelson. Steerable filters for early vision, image analysis and wavelet decomposition. In *ICCV90*, pages 406–415, 1990.

[3] T. Gevers and A.W.M. Smeulders. A comparative study of several color models for color image invariant retrieval. In *Image Databases and Multi-Media Search*, pages 17–26, 1996.

[4] M.M. Gorkani and R.W. Picard. Texture orientation for sorting photos at a glance. In *ICPR94*, pages A459–464, 1994.

[5] F. Liu and R.W. Picard. Periodicity, directionality, and randomness: Wold features for image modeling and retrieval. *PAMI*, 18(7):722–733, July 1996.

[6] B.S. Manjunath and W.Y. Ma. Texture features for browsing and retrieval of image data. *PAMI*, 18(8):837–842, August 1996.

[7] M.J. Swain and D.H. Ballard. Color indexing. *IJCV*, 7(1):11–32, November 1991.

[8] H. Tamura, T. Mori, and T. Yamawaki. Textural features corresponding to visual perception. *SMC*, 8:460–473, June 1978.

[9] D. White and R. Jain. Algorithms and strategies for similarity retrieval. tr vcl-96-101. In *U. of California, San Diego*, 1996.

[10] J. Mao and A.K. Jain. Texture classification and segmentation using multiresolution simultaneous autoregressive models *Pattern Recognition*, 25(2):173-188, 1992.

[11] S. Sclaroff and L. Taycher and M. La Cascia. ImageRover: A Content-based Image Browser for the World Wide Web *Proc. IEEE Workshop on Content-Based Access of Image and Video Libraries*, June 1997.

[12] H. Tamura and S. Mori and T. Yamawaki. Textural Features Corresponding to Visual Perception *IEEE Transactions on Systems, Man and Cybernetics*, 8(6):460-1473, 1978.

[13] B. Thai and G. Healey Extracting Symmetry Features from Color Images *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* June 1997.

[14] S. Sclaroff Deformable Prototypes for Encoding Shape Categories in Image Databases *Pattern Recognition, Special Issue: Image Databases*, April 1997.

[15] C. Carson and S. Belongie and H. Greenspan and J. Malik Region-Based Image Querying *Proc. IEEE Workshop on Content-Based Access of Image and Video Libraries*, June 1997.

[16] U. Gargi and R. Kasturi An evaluation of color histogram based methods in video indexing *Proc. of International Workshop of Image Databases and Multi-media search*, August 1996.

[17] J. Huang and S.R. Kumar and Mandar Mitra and W.-J. Zhu and R. Zabih Image Indexing Using Color Correlograms *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* June 1997.