WebSeer: An Image Search Engine for the World Wide Web[1]

Charles Frankel, Michael J. Swain, and Vassilis Athitsos

The University of Chicago
Computer Science Department
1100 East 58th Street
Chicago, Illinois 60637

Technical Report 96-14

August 1, 1996

Abstract

Because of the size of the World Wide Web and its inherent lack of structure, finding what one is looking for can be a challenge. *PC-Meter's* March, 1996, survey found that three of the five most visited Web sites were search engines. However, while Web pages typically contain both text *and* images, all the currently available search engines only index text. This paper describes *WebSeer*, a system for locating images on the Web. WebSeer uses image content in addition to associated text to index images, presenting the user with a selection that potentially fits her needs.

[1]This work was supported in part by ONR contract N00014-93-1-0332 and NSF Grant No. IRI-9210763-A01.

Introduction

The explosive growth of the World Wide Web has proven to be a double-edged sword. While an immense amount of material is now easily accessible on the Web, locating specific information remains a difficult task. An inexperienced user may find it next to impossible to find the information she wants; even an experienced user may miss relevant Web pages. Users searching the World Wide Web have a number of options presently available to them. Lycos, Excite, Alta Vista, and Yahoo! are but a few examples of useful search engines. All of these systems have been designed primarily to find text-based information on the Web. WebSeer is designed to find the *other* major source of information currently available on the Web: images.

In order to find specific images, a method of discovering and indexing their contents must first be developed. The pixels alone are insufficient; some additional information is needed to make it possible to search for specific images. Discovering image content has proven to be an extremely difficult problem. One approach has been to supplement the image content with other types of information associated with the image. Ogle and Stonebraker's Cypress system [1][2] uses information contained in hand-keyed database fields to supplement image content information. Srihari's Piction system [3] uses the captions of newspaper photographs containing human faces to help locate the faces. IBM's QBIC system [4] relies on the user specifying specific visual cues or providing an example image (e.g. a sketch) to begin a query for an image. In Picard and Minka's Foureyes system [5][6] close interaction with a human user supplements information derived from the image content.

WebSeer uses the textual information surrounding an image and the image header to supplement the information derived from analyzing the image content. This additional information is used to create a context in which image analysis algorithms can effectively operate. Image analysis algorithms are then used to classify the image within a taxonomy of types (drawing, photograph, portrait, etc.) and to extract useful semantic information.

Like text search engines, WebSeer does not have to access the original data to respond to a query; all analysis of the image and surrounding text is done off-line during the creation of the database. In this way, WebSeer will be able to give fast query responses to a possibly huge number of users.

The first part of this report describes the types of information that an image search engine such as WebSeer could extract from the Web and how they could be stored in a database. The second part illustrates a number of example queries and how images could be found that fit the queries. The final part details the prototype WebSeer program and the algorithms that have been implemented to date.

How to Find Images on the Web

Information for finding images on the World Wide Web can come from two sources: the relevant text and the image itself. Using information from both sources, a program should be able to successfully retrieve requested images.

Cues from the Text and HTML Source Code

An HTML document is a *structured* document. Understanding the structure of a particular document can reveal valuable information about the images contained on that page. There are several places relevant information about image content may be located within the document. These are ordered by the likelihood of the text being useful in an image search.

1. **Image File Names:** File names contain important clues as to the content of the file. In many cases the full URL of an image (e.g. “http://www2.int-evry.fr/~hug/images/marilyn/mm-trans.gif”) contains more information than a relative URL (e.g. “mm-trans.gif”), which may be included in the HTML source. We have found the file name and the name of the top-level directory containing the file are most useful for indexing the content of the image. A caveat is that filenames may be difficult to decipher because of abbreviations used for brevity or out of necessity (e.g. in the FAT file system used in DOS/Windows 3.1).
2. **Image Captions:** Often, images have captions that describe the picture. Although HTML does not have an explicit caption tag, captions are often signaled as text that is located within the same *center* tag as the image (http://www.lib.uchicago.edu/~mozart/Marilyn/monthly.html), within the same cell of a table as an image (http://studentweb.tulane.edu/~llovejo/marilyn/), or within the same paragraph (see Appendix A).
3. **`ALT=` Text:** Image fields in HTML documents may contain a special `ALT=` section for alternate text which is displayed in the event that the images can not be loaded. This alternate text may briefly describe the content of the image. For instance, the alternate text for the a photograph of a person may contain simply the name of the person in the photograph. The alternate text may also give hints as to the intended “goal” of the image. For instance, an image containing an ad for Coca-Cola may have the alternate text “Buy Coca-Cola.”
4. **HTML Titles:** The HTML title of the document (which is displayed above the viewed page by the browser, is used for history lists, and so on) often provides information about the content of the images contained within the document. For example, the page http://tomahawk.cf.ac.uk/~scotw/marilyn/page2/page2.html is entitled “Marilyn Monroe Images - The ventilator scene”, which is highly descriptive of the contents, and more descriptive than any other cues that can be found on the page (see Appendix A).
5. **HyperLinks:** The text of a hyperlink usually gives clues about what the link refers to. In many cases, images are referenced by hyperlinks rather than being imbedded within the text of the page. An example of such a link is “Marilyn JPG #21 (Jumping for joy)”, at

http://www.netins.net/showcase/agency/marilyn.html. The same photo is referenced as "Marilyn *jumping for joy*" at http://www.rtmol.stt.it/rtmol/onsite/stars/marilyn/m_imag~1.html, where only the italicized text is the text of the hyperlink. In many cases, text hyperlinks assume the reader understands that the pictures are of Monroe and does not give any information about who is in the picture. These links say "Simply gorgeous ....", "Leaning on a banister", and so on. Consequently, other text in the same sentence or phrase with the hyperlink can also be relevant to deciphering the image content.

6. **Other text:** In addition to those types mentioned above, other text can give cues about the images contained on the page. This text could be used to index the images on the page, with a lower likelihood of reference than the text categories listed above.

Cues from the Image Content

Although it is clear that the context provided by the surrounding text can produce an effective image search engine, examining the images themselves can substantially enrich the search. In this section we will summarize some of the relevant information that can be derived directly from the image content.

The following attributes can be easily obtained from the image header.

- Grayscale vs. color: Some may wish to search for only color images. While color images can be converted to grayscale, those in search of black-and-white photography will want the real thing.
- Image Size: This is usually useful for those who wish to screen out images smaller than a certain size.
- File Type (JPEG, GIF, GIF89, etc.): Some searchers may wish to avoid the compromises in quality introduced by constraints of these formats, especially the limited color palette in the GIF format. GIF89s are animations, and so will often be of separate interest than the other static image formats.
- File Size: For those with slow connections or limited RAM or hard drive space, overly large files may be of less interest
- File Date (when present): If someone is looking for newer images, the date can be used to screen out images that are certainly older than a certain date.

Obtaining information beyond that listed above generally requires some image analysis. The most useful sorts of image analysis can be grouped into three overlapping categories: 1. classifying the image into one of several types (e.g. photograph, hand-drawing, computer-generated drawing), 2. recognizing image structures (e.g. faces, horizons), and 3. distinguishing specific regions primarily by color/texture attributes (e.g. skin, sky, foliage).

Classifying Images

Creating a taxonomy of image types can facilitate separating images the user is interested in from those that she is not. However, the most important aspect of classification is creating the categories. The authors of an image search engine must either design a system that uses helpful categories, or include a mechanism to allow the user to guide her own search using relevant schema. Ideally, a system would use aspects of both types of categorization.

One important classification which can be made is between images that are photographs and hand-drawn and computer-created drawings. In many cases users may wish to find only photographs of faces, and ignore the many other images that match the search string. Some of these non-photographs may be weeded out because they are too small (e.g. buttons, dividers, arrows). Others, such as image-maps containing text, advertisements, or drawings, must be eliminated on other criteria. Of course, more realistic drawings will be likely to be mistakenly categorized as photographs. Likewise photographs can be thresholded and manipulated in other ways to make them look more like drawings. Nonetheless, we have found that many images can be successfully classified as either drawings or photographs. It may also be possible to separate computer-generated drawings (i.e. those that are artificial-looking, with precisely straight lines and precise color gradients) from hand drawings.

There are many other potentially useful categories. Images can be classified as portraits, which can be further classified by the number of people in the image and the amount of the body shown along with the face (close-up, head-and-shoulders, upper-body, etc.). Images can also be classified as indoor or out, and outdoor images can be classified as landscapes.

Some of these categorizations may be best determined by analyzing global statistics, or the statistics of relatively large portions of the image (e.g. photo vs. drawing). Others require other sorts of analysis, which are described below.

Recognizing Image Structures

There are many types of objects to identify in images, which would aid content-based indexing. We have identified two "objects" that are both important to the typical WebSeer user and within the capabilities of existing computer vision technology. They are:

1. **Faces:** Face-finding algorithms have recently become fairly successful at finding faces in grayscale photographs [7][8]. Faces in photographs are often seen upright, with the subject looking at the camera; the easiest situation for face-finding algorithms. The high quality of many photographs found on the Web, in many cases produced by professionals with careful control of the lighting and high quality film and optics, improves the reliability of face detectors. Combining face detection with color information from skin detection should make a reliable face detector. A primary role of a face detector is to distinguish images of people from other types of images that may show up on the same page. However, face detection can do more. Users may wish to find a portrait of a single person, a group photo, or pictures of a "crowd" containing many faces. Portraits can be close-ups, head-and-shoulders shots, upper-body, or full-body shots; the size and location of the face can differentiate among these choices.

2. **Horizon:** A number of systems search images for the presence of a horizon [9] (e.g. the Cypress System). Detecting a horizon in a photograph tells us that the image has been taken outdoors and allows us to roughly classify it as a *landscape* photograph.

Face recognition is another possible task one might think of assigning the system. For instance, WebSeer could have a database of people that users are likely to search for, and identify their faces when creating the index. Restricting possible matches to those people mentioned in the text would greatly reduce the number of false positives the face recognizer would return. Further research will likely define other categories of objects that could be searched for and indexed in the same way as faces and horizons.

Recognizing Regions by Color/Texture Analysis

Some color and texture analysis may be useful for finding image regions containing skin, sky, foliage, etc. in photographs. Skin may be used for finding faces and determining if images are portraits; it is in itself of interest to some image searchers. Sky and foliage can also be used in image classification. Other less common color textures, including sand, wood, etc., may also be identified. If more classes such as these are created, it may prove useful to look for them only when the context suggests they may be present.

The WebSeer Implementation

WebSeer was implemented with three guiding principles in mind:

First, WebSeer must have acceptable performance. We need to allow for extremely high speed searches, as we expect a large number of people to be using our system simultaneously. Since indexing occurs off-line, the performance of the indexing engine is less crucial than that of the search engine. Nonetheless, indexing the entire web in a reasonable amount of time requires the processing to proceed quite quickly. For example, assuming there are 10 million unique images on the web, indexing them in two weeks would require eight images to be processed every second. Crawler speed is also important. Preliminary results indicate that for every 100 HTML pages on the Web, there are 40 (unique) GIF images and one (unique) JPEG image. In contrast to the file size of HTML pages, which averages around 6k, the average file size for GIF files is 11k, and the average file size for JPEGs is 35k.

Second, we tried to incorporate standard commercial software and hardware whenever possible. Much work has been put into developing advanced database engines, and WebSeer's ability to leverage technology adds significantly to its power. Microsoft's Visual C++ development environment was used for much of this project. Microsoft's Foundation Classes in conjunction with the Standard Template Libraries (STL) provided many useful functions and abstractions during the development of this project. On the hardware front, the use of relatively inexpensive PCs allowed us to ramp up our storage and processing capacities quickly and within reasonable cost.

Third, the project should provide a basis for experimentation. We foresee WebSeer evolving in the following ways:

- better image understanding algorithms.
- advanced text indexing capabilities
- improved interactions between the image understanding and text indexing algorithms
- more complex transformations from form query submissions to search actions taken.

To facilitate this type of research, the project is divided in such a way that each component can be worked on independently. Additionally, we wish to facilitate the incorporation of relevant new technologies as they become available. The WebSeer project is composed of five major executables. With the exception of the URL Server, which is written in Java, all executables are written in C++ and run on a M.S. Windows NT 3.51 platform.

1) The WebSeer Crawler crawls the web downloading both HTML pages and images. The crawler is multi-threaded so that the delay downloading pages is spread over multiple threads. Each thread is connected to a database of previously visited (and waiting to be visited) URLs using the ODBC 2.0 database protocol.

2) The URL Server is a multi-threaded java application which receives requests to download URLs from the WebSeer Crawler. Separating the URL server from the Crawler application allows us to download pages from multiple machines (with different operating systems) simultaneously.

3) The WebSeer Indexer creates the index which is searched by the users. The indexer parses the HTML code and executes the appropriate image understanding applications.

4) The WebSeer CGI script is called when the user submits (POSTs) a query from the WebSeer form. This script opens a TCP/IP connection to the WebSeer Search Server, and formats the results for display to the user.

5) The WebSeer Search Server accepts requests from the WebSeer CGI script and performs the appropriate searches based on the form fields which the user has filled in.

The WebSeer Crawler largely obeys the Robot Exclusion Protocol[a]. The Protocol is dependent on system administrators including a `robots.txt` file on their web site, which lists areas of the web site which robots should not visit. Most robots are designed to download only HTML files, and so visiting a directory which includes images would be inappropriate. For this reason, some robots.txt files exclude directories which contain only image files (See Appendix B). Since the WebSeer Crawler is designed to download images, we obey the restrictions specified by the `robots.txt` file when deciding whether to download an html file, but not when downloading an image file.

[a] The Protocol is specified at http://info.webcrawler.com/mak/projects/robots/norobots.html

The WebSeer Indexer is perhaps the most interesting component to this system. The indexer processes a single HTML file at a time. As each image is encountered in the HTML file (either through an "<img src=…>" tag or an "<a href src="this_image.gif">" tag, the appropriate text is extracted in the form of whole words. These words may be present in any of the ways described above. Some of the words are more likely to contain information about the content of the image they refer to. For this reason, we *weight* the words according to the likelihood that they contain useful information. Words contained in the title of the HTML page, for example, have a lower weight than those in the ALT tag of an image. When a user performs a search the summed weights of the matching words are used as one criterion for sorting the resulting images. These words and their weights are stored in a single database table. Part of an example WebSeer text table is show below.

Image ID	Text Fragment	Fragment Weight
31308	kathy	1
31308	ireland	1
31308	vivek	1
31309	tn	3
31309	ki	3
31309	vivek	1
31309	kathy	1
31309	ireland	1
31311	marilyn	3
31311	gallery	2
31311	lovejoy	1
31311	marilyn	1
31311	monroe	1
31311	collection	1

Image understanding algorithms are run whenever an image is encountered. The figure below indicates the fields which WebSeer currently saves for each image.

Field Name	Sample Data
File Name	http://www.cdt.org/images/cdtlgo.gif
Source URL	http://www.cdt.org/index.html
Color Depth	8 bit color
File Size	3,129 bytes
File Type	gif
Image Width	204
Image Height	107
Is Image a Photograph?	No
Is Image an Image Map?	No
Is Image included as a Reference?	No
Number of Faces	0
Largest Face Size	0%

Three fields contain information about how the image was included in the HTML document. The "Is Image a Photograph" field contains the results of the photograph/drawing algorithm described below. "Is Image an Image Map" indicates whether the image appeared in the HTML code with an ISMAP tag. These images appear to the user as clickable images.

"Is Image included as a Reference?" indicates whether the image appears as part of an HTML document, or whether only a reference is included. If an image is part of an HTML document, users viewing that document will immediately be able to see that image. If a reference is included, the user may have to click on some part of the document in order to view the image. References to images are common when the images are large and/or slow to download.

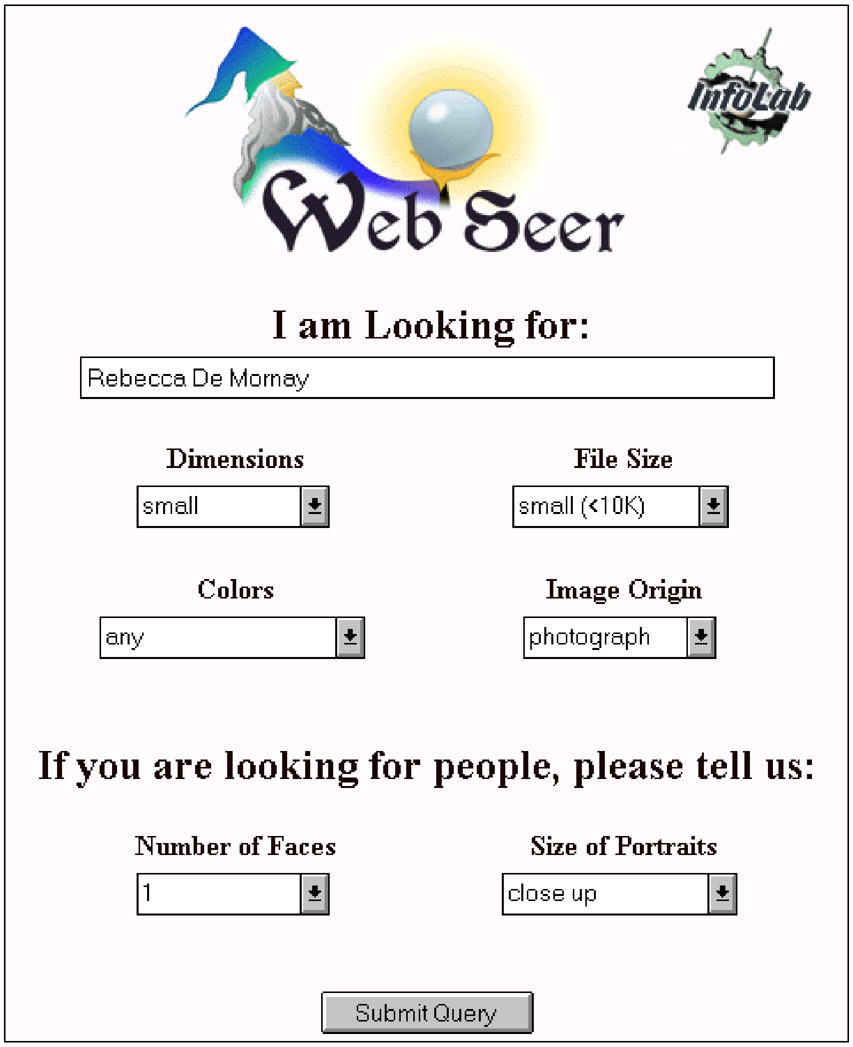
The "Number of Faces" field indicates the number of faces which were detected in the given image. Each detected face has four attributes associated with it: horizontal position, vertical position, height, and width. Since an image may contain a number of different faces, the attributes (fields) associated with each particular face are saved in a separate "face table".

The "Largest Face Size" field saves the height of the largest face detected in the given image, as a percentage of image height. Although this information is also contained in the face table, saving the information in the image table speeds some searches by eliminating the need to perform a JOIN of the image table with the face table. An example is a search for a "close-up" image. We define close-up images as images where the largest face has a height greater than 50% the size of the image.

An Example Search

A user is interested in finding small, close-up images of Rebecca De Mornay. They type "Rebecca De Mornay" as the search text, and make selections as displayed in Figure 1.

The results page interface, shown in Figure 2, works as follows. Thumbnails of the resulting images are displayed above a bar which indicates the size of the original image. Clicking on the image will launch your browser to the URL original *image*. Clicking on the page icon to the right of the image will launch your browser to the URL of the page which *contains* the image.



I am Looking for: Rebecca De Mornay

Dimensions: small

File Size: small (<10K)

Colors: any

Image Origin: photograph

If you are looking for people, please tell us:

Number of Faces: 1

Size of Portraits: close up

Figure 1: An image search query

The results were obtained as follows. Webseer searches for images which contain any of the words contained in the text, after eliminating words that are especially common (e.g. connecting words such as “and” and “the”). The results are sorted by the summed weight of the words associated with that image. For instance, if one image has associated words “Rebecca”, with a weight of 3, and “Mornay” with a weight of 1, that image would have a summed weight of 4. Only images with a height or width < 150 pixels, with file size < 10K, which were determined to be photographs, and which contain exactly one face whose height is at least 50% of the size of the image are included in the results. Lastly, the results are sorted, first by the weighted sum of associated words, and then, since “close up” was selected, by the detected face size (with the largest faces appearing first).

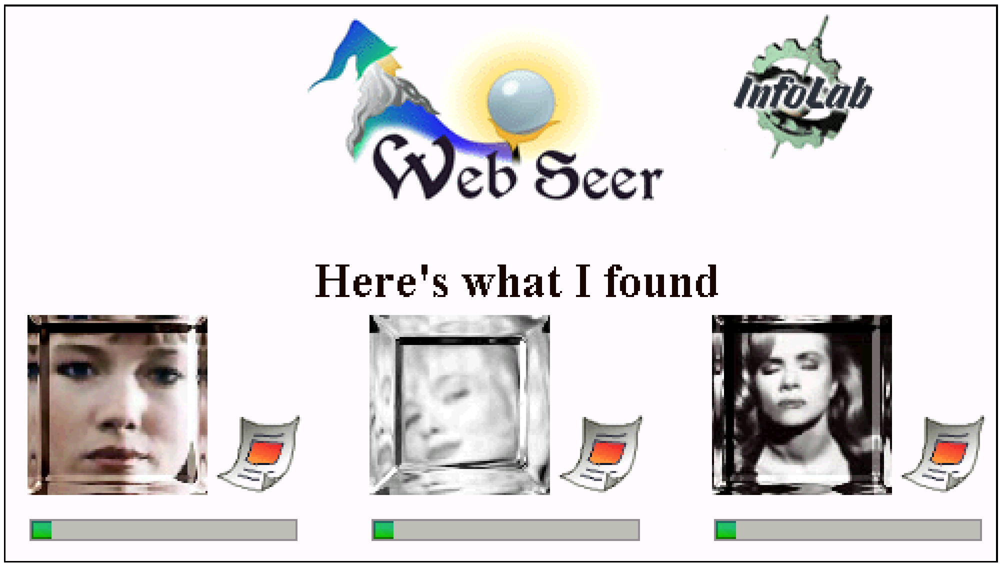


Figure 2: Results of the query shown in Figure 1.

Image Content Analysis: Classifying Photographs and Drawings

The algorithm classifies images into photographs and artificial images, where the terms "artificial images" and “drawings” are used to denote all images that are not photographs. There are some borderline cases. The most common is images that contain a photograph and an artificial part, which could be, for example, a computer-generated background or super–imposed text. Currently, the algorithm is not designed to handle such images consistently.

The algorithm can be split into two independent modules. One module consists of tests that separate photographs from drawings. After the image has been submitted to all the tests, the decision-making module decides how it should be classified.

Tests

In every test an image gets a positive real number as a score. The images are represented by three matrices, corresponding to their red (R), green (G), and blue (B) color bands, whose entries are integers from 0 to 255. Photographs and drawings tend to score in different ranges. These are the tests we currently use in the algorithm:

•The **band difference test**. We pick a threshold T between 0 and 255 and initialize the counter C to 0. For every pixel in the image, let r, g, b be its respective values in the R, G, B matrices. We look at $|r - g|$, $|g - b|$ and $|r - b|$, and we increase the counter C by one for each of those absolute values that is greater than T. The score of the image is $\frac{C}{S}$, where S is the number of pixels in the image. We expect drawings to score higher than photographs, because the colors used in drawings are usually highly saturated.

•The **farthest neighbor test**. We pick a threshold T between 0 and 255 and initialize counters C and S to 0. For each inner pixel P of the image, we consider its top, bottom, left, and right neighbor. For each neighbor, we calculate the absolute difference of its R value from the R value of P, and we look at the maximum M of those four absolute differences. If $M > 0$, we increase S. If, in addition, $M > T$, we increase C. The score of the image is $\frac{C}{S}$.

The rationale behind this test is to see how abrupt the color changes are in an image. In photographs they tend to be smoother than in drawings. Therefore, drawings tend to score higher than photographs in this test.

• The **color test.** We count the number of different colors in the image. Drawings tend to have fewer distinct colors than photographs.

• The **most common color test**. We find the color that occurs most frequently in the image. The score is $\frac{C}{S}$ where C is the number of pixels in the image that have that color, and S is the total number of pixels in the image. Again, artificial images usually score higher than photographs, because they tend to have a one-color background.

• The **narrowness test**. Let R be the number of rows and C be the number of columns in the image. If $R > C$, then $N = \frac{R}{C}$, otherwise $N = \frac{C}{R}$, where the score of the image is N. Photographs tend to score between 1 and 2, whereas drawings often score above 2.

For images saved in the JPEG format, the color test doesn't work because of the way JPEG compression works. Neighboring pixels that have the same color in the original image usually have similar but not identical colors after the compression. In addition, abrupt color changes in the original images usually become smoother after compression. This makes the farthest neighbor test less powerful.

Decision making

Since GIF and JPEG images tend to follow different scoring patterns in the tests we use, the decision maker needs to know how images tend to score based both on their category (photograph or drawing) and their compression format. So, for each of the four cases (GIF photographs, GIF drawings, JPEG photographs, JPEG drawings), we created a training set of at least 100 images, selecting representative images from the Web. For each test, we calculated and stored the scores of all images in the training set. The minimum score is set so that 1% of the images score below it. The maximum is defined in an analogous way. The set of scores together with the minimum and the maximum is called the distribution of scores for that test. In particular, if the training set contains photographs, the distribution is called natural, otherwise it is called artificial.

Suppose we have a GIF image and calculate its score S on a particular test T. Consider the natural distribution of scores for that test. If the score S is between the minimum and the maximum score in that distribution, L is the percentage of scores in the distribution that are less than or equal to S, and M is the percentage of scores that are greater than or equal to S. We define the *natural grade* N of the image to be the minimum of L and M.

If the score S is less than the minimum, then R = $\frac{S}{\text{minimum}}$ and the natural grade N is $\frac{R}{1000}$. If S = 0, however, we set N to 0.001. If S is greater than the maximum, then R = $\frac{\text{maximum}}{S}$ and N = $\frac{R}{1000}$. We calculate the artificial grade A of the image in a similar way.

Based on these scores, we assign an overall grade between 0 and 1 to the image. The higher the grade is, the more likely the image is to be a photograph. We fix a threshold T, and we classify all images whose grade is over T as photographs and all other images as drawings. This is how we calculate the grade:

• We initialize P and D to 1.

• For each test, we calculate the score of the image, and from the score we calculate the natural grade N and the artificial grade A for that test. We set P to PN and D to DA.

• After we are done with all tests, we set the grade of the image to $\frac{P}{P + D}$.

The reader may have recognized that if, for each test, A was the conditional probability that an image got the score S given that it was a drawing, and N was the conditional probability that an image got the score S given that it was a photograph, then the grade of the image would be the probability that it was a photograph given all its test scores and assuming that the different tests have independent score distributions. In that case, we should classify all images scoring over 0.5 as photographs and the rest as drawings. However, A and N are not defined to be the conditional probabilities. So, the optimal classification threshold is not necessarily 0.5.

Nevertheless, it remains true that the grade is more reliable when the tests we use are pairwise independent or approximately independent.

Results

The algorithm has been tested both on the training sets that we used, and on images that we didn't include in those sets. With the group of JPEG artificial images the algorithm was only tested on a training set. We have found that purely artificial images in the JPEG format occur less frequently than images in other formats.

The following tables give some experimental results:

• Table 1 gives results on GIF images. The first row gives the number of images for each group. After that, each entry gives the percentage of correct classifications of images in the given group when grades are calculated based on that test only. The group "GIF art1" is the training set of GIF drawings. “GIF nat1” is the training set of GIF photographs. “GIF art2” and “GIF nat2” are, respectively, sets of drawings and photographs that were not used in training.

• Table 2 gives results on JPEG images, in the same format as Table 1.

• Table 3 gives the values of all thresholds used.

Table 1: Results on GIF Images

	GIF art1	GIF art2	GIF nat1	GIF nat2
number of images	395	339	129	112
band difference test	50.9	45.4	94.6	90.2
farthest neighbor test	86.6	89.1	89.1	83.1
color test	60.8	79.4	83.7	85.7
most common color test	81.0	88.5	92.2	85.7
ratio test	71.4	74.6	77.5	77.7
All tests combined	96.2	94.1	88.4	81.2

Table 2: Results on JPEG Images

	JPEG art1	JPEG nat1	JPEG nat2
number of images	150	228	118
band test	13.3	98.7	97.5
farthest neighbor test	55.3	95.2	97.5
most common color test	66.7	98.2	91.5
ratio test	66.7	96.5	99.2
All tests combined	92.0	93.9	95.8

Table 3: Thresholds

	GIF	JPEG
band difference test	233	185
farthest neighbor test	100	150
grade	0.3	0.1

Current Work

We are currently working on improving the algorithm in three directions:

• Finding more tests and perfecting the ones we already have.

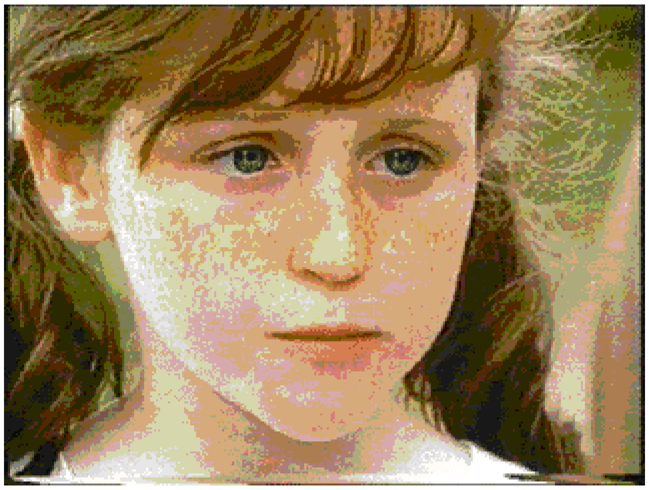
• Refining the decision-making module. In particular, we are interesting in finding a way to use information from tests that are not pair-wise independent.

• Increasing the number of categories for classifying images. As a first step, we have to create a category for images that contain both a photograph and an artificial part, and find a way of locating each part in such images.

Examples of Images

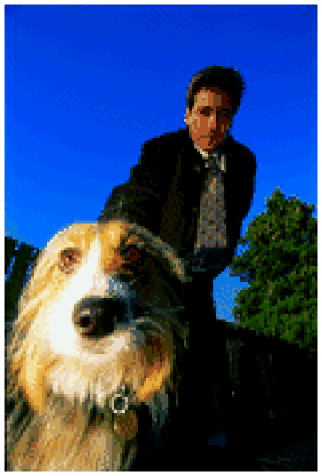
`http://www.nashville.com/~DavidFsFan/update.gif`

This is a typical GIF drawing with a grade of less than 0.001, which was classified correctly. It has only 3 colors, a ratio of columns to rows over 5, and abrupt color changes.



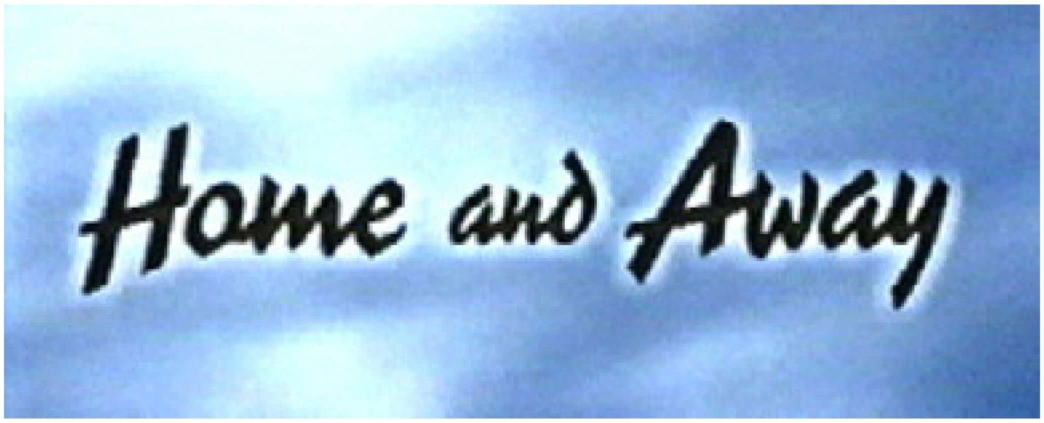
http://huizen.dds.nl/~evdmeer/kellie1.gif

This is a typical GIF photograph. There are no regions of constant color; the color transitions are smooth. The grade of the image was 0.999.



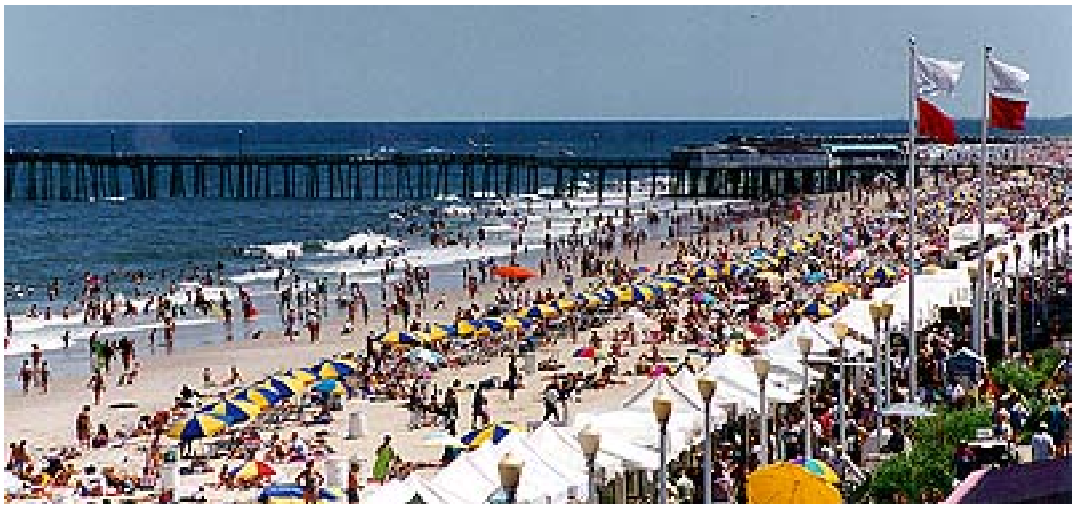
http://www.pathfinder.com/@@dKvzf*AZdQAAQAkt/ew/950310/xfiles/CT78.gif

This is a misclassified GIF photograph with a grade of 0.019. It scores very high in the band difference and most common color tests.



http://www.ozemail.com.au/~rellis/title.jpg

This is a misclassified JPEG drawing with an overall grade of 0.338. The color transitions are smooth, and the only test on which it scores below the threshold 0.1 is the narrowness test.



http://www.pilotonline.com/boardwalk/images/beach.jpg

This is a misclassified JPEG photograph with a grade of 0.001. The color transitions are very abrupt, and the image is unusually narrow.

Locating Faces

We are testing face finders written by Sung and Poggio [7] and Rowley, et. al. [8]. Both of these approaches look for upright faces facing the camera. The results shown for Rebecca De Mornay were obtained using Sung and Poggio's code. The major drawback of Sung and Poggio's algorithm is efficiency; finding all the faces in a large image can take minutes (on a Sun Sparc 20 with 125 MHz hyperSparc processors) whereas we can only afford an average of fractions of a second per image. We have also experienced problems with false positives using Sung and Poggio's code with its default settings.

Rowley et. al.'s code is considerably more efficient, as has been reported in [8], and so is more suitable for our application. Published results indicate that the effectiveness is about the same as Sung and Poggio's on grayscale images. We hope to increase the effectiveness and efficiency by looking for faces only in photographs, and using color to select regions of interest for the face finder, since most of the photographs on the Web are color.

Scaling Up

Techniques for content-based retrieval that work with a hundred, a thousand, or even ten-thousand images will not necessarily scale up to the task of indexing all the images on the World Wide Web. There are an estimated 30 million *Web pages* (HTML documents). Our preliminary experiments indicate there may be about one-third as many images as Web pages, meaning about 10 million images to index. Rough calculations suggest that WebSeer's database will be about 1.5 GB, and storing thumbnails will take up about 15 GB of disk space. Crawling the Web to index all the images will require downloading them all. Our current multi-threaded Web crawler can download many pages per second, running on a 200 MHz Pentium Pro PC attached to a dual T1 line shared with the rest of the University. Since projected improvements in networking infrastructure will improve this performance substantially, image processing, face finding in particular, is likely to be the bottleneck. We will expend considerable effort optimizing the image processing algorithms used in the crawler.

Future Work

We view the photograph vs. drawing distinction as a starting point toward a more general *image taxonomy*. We are working on identifying a taxonomy that fits users' needs and is constructed of image classes that can be reliably identified. Some of these categories may include advertisements, geographic maps, landscapes [9], city/country scenes [5][10], night scenes, sunsets, scenes with foliage, and so on.

Tomasi and Guibis descibe a system which classifies *types* of images based on image content alone [11]. We believe that we need a close interaction between the image understanding algorithms and the associated text indexing algorithms in order to successfully categorize images. Srihari's theory of "visual semantics" [12] provides useful insight into some of the challenges of integrating text indexing with image understanding algorithms.

A closer interaction between text and image processing will also provide significant improvements in indexing speed. Detecting faces in an image is an example of an image understanding algorithm which requires a relatively large amount of processing time. By only running those algorithms on images whose results are likely to be useful, we believe that we can save significant processing time per image.

As more functionality is added to WebSeer, the interface should get simpler, not more complicated. The selections of some image content combination boxes, for example, could be assumed from the text of the query that the user enters. Consider the query "Bill and Hillary Clinton." The capitalization of the first letters of the words "Bill" and "Hillary" provides a clue that the words are proper nouns, and the words “Bill” and “Hillary” can be found in a list of common first names. This information can be used to direct the search towards images in which two faces were detected.

The approach and a considerable fraction of technology from WebSeer should be applicable to other multimedia databases containing structured text and images, including newspaper or magazine archives, and video annotated by closed captions.

Appendix A:

HTML source code for several web sites demonstrating some of the ways in which image “captions” may be included in a structured HTML document.

```
<HTML><TITLE>Marilyn Monroe Picture of the Month</TITLE>

<BODY BGCOLOR="#FFFFFF">

<HR>

<center>

<H1>Picture of the Month<BR>

July 1996</H1>

<A HREF="mmps-1.jpg">

<IMG SRC="mmps-1.jpg" ALT="Picture from July 6, 1962"></A>

<BR>

On July 6, 1962, Allan Grant took these beautiful pictures of
Marilyn

in her home to accompany an interview that appeared in the August

3, 1962 issue of <em>Life</em>.  It was her last photo session.

<P>Photographs &copy;copyright Allan Grant.

</center>

<P>

<HR>

Return to: <A HREF="../Marilyn.html"> Marilyn Monroe</A>

<ADDRESS>Peggy L. Wilkins / (312)702-8780 / mozart@uchicago.edu

</ADDRESS>

</BODY>

<HTML>
```

http://www.lib.uchicago.edu/~mozart/Marilyn/monthly.html

```
<HTML>

<HEAD></HEAD>

<BODY   BGCOLOR=#000000   TEXT=#DB9370   LINK=#FF00FF   VLINK=#EAADEA
ALINK=#FF0000>

<BASEFONT SIZE=4>

<CENTER>

<IMG  SRC=../images/Marhead.jpg  HEIGHT=144  WIDTH=288  ALT="Lovejoy’s
Marilyn Monroe Image Gallery"><BR>

</CENTER>

<BR>

<FONT  SIZE=+1>Although  there  is  a  slim  possibility  that  Marilyn
Monroe  is  not  the  most  beautiful  woman  I  have  ever  seen,  she
posessed an ability to command attention greater than any other’s
and can still, after so many years, captivate me.  I have collected
images from fans and from around the net in one large archive of her
images.</FONT><P>

<A NAME=galleries>

<TABLE BORDER=0 CELLSPACING=30>

<TR VALIGN=BASELINE>

<TR VALIGN=BASELINE>

<TD>

<A HREF=../favorites/index.html TARGET="_parent">

Favorites from the net</A>

</TD>

<TR VALIGN=BASELINE>

<TD>

<A HREF=../beach/index.html TARGET="_parent">

Marilyn on the beach</A>

</TD>

</TR>

</TABLE>
```

Excerpt from http://studentweb.tulane.edu/~llovejo/marilyn/

```
<TITLE>Marilyn Monroe Images - The ventilator scene</TITLE>

</head><body>

<hr>

<h3><i>Marilyn Archive - The ventilator scene</i></h3>

<hr>

To retrieve an image click on the list below the thumbnails.

<hr>

<IMG SRC="one.gif">

<hr>

|   <a   href=mm-053.jpg>mm-053.jpg</a>   |   <a   href=mm-054.jpg>mm-
054.jpg</a> |

<hr>

<IMG SRC="two.gif">

<hr>

<a    href=mm-055.jpg>mm-055.jpg</a>    |    <a    href=mm-055a.jpg>mm-
055a.jpg</a>  <a href=N-mm49.jpg>N-mm49.jpg</a>

<hr>

<h3><b><i>Acknowledgement</b></i></h3><p>

These pictures are courtesy of a bunch of Film Magnates.
```

http://tomahawk.cf.ac.uk/~scotw/marilyn/page2/page2.html

Appendix B:

Below is an example of a robot exclusion file which restricts access to directories which contain only images. Note the entries containing `/acis/gif/`, `/cu/libraries/events/sw25/gifs/`, `/cu/libraries/gifs/`, and `/gif/`. The author of this file has informed us that these restrictions were included since he believes that current robots are only interested in indexing textual information, and these directories contain no text [13].

```
# this file is read by any robot that conforms to the WWW
robot
# guidelines described here:
#       http://www.nexor.co.uk/mak/doc/robots/norobots.html


User-agent: *
Disallow: /acis/eds/triarc
Disallow: /acis/gif/
Disallow: /acis/poetry/
Disallow: /cgi-bin/
Disallow: /cp/
Disallow: /cu/libraries/events/sw25/gifs/
Disallow: /cu/libraries/gifs/
Disallow: /cu/libraries/inside/
Disallow: /cu/spectator/
Disallow: /cu/distrib/
Disallow: /experimental/
Disallow: /gif/
Disallow: /httpd/reports/
Disallow: /imaging/
Disallow: /tc/
Disallow: /waissources/
```

http://www.columbia.edu/robots.txt

References

[1] Ogle and Stonebraker (1995). Chabot: Retrieval from a Relational Database of Images. IEEE *Computer,* 28(2), 49-56.

[2] Nancy A. Van House, Mark H. Butler, Virginia Ogle, and Lisa Schiff. User-Centered Iterative Design for Digital Libraries: The Cypress Experience. *D-lib Magazine*, Feb 1996, ISSN 1082-9873.

[3] Srihari, Rohini K. (1995). Automatic Indexing and Content-Based Retrieval of Captioned Images. IEEE *Computer,* 28(2), 49-56.

[4] Flickner, Myron, et. al. (1995). Query by Image and Video Content: The QBIC System. *IEEE Computer.* September, 1995 Volume 28, pgs. 23-32.

[5] R.W. Picard (1996). A Society of Models for Video and Image Libraries. *Media Laboratory Perceptual Computing Section Technical Report No. 360*.

[6] T.P. Minka and R.W. Picard (1995). Interactive Learning using a "society of models". *MIT Media Laboratory Perceptual Computing Section Technical Report No. 349*.

[7] Kah-Kay Sung and Tomaso Poggio (1994). Example-based learning for view-based human face detection. *Technical Report A.I. Memo 1521, CBCL Paper 112, MIT*, December 1994.

[8] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade (1995). Human Face Detection in Visual Scenes. *Carnegie Mellon University Technical Report CMU-CS-95-158*.

[9] Markus Stricker and Alexander Dimai (1996). Color Indexing with Weak Spatial Constraints. *SPIE Conference*, February 1996, San Jose, CA.

[10] R.W. Picard and T.P. Minka (1995). Vision texture for annotation. *Journal of Multimedia Systems,* vol. 3, pp 3-14.

[11] C. Tomasi and L. Guibas (1994). Image Descriptions for Browsing and Retrieval. *Proceedings of the ARPA Image Understanding Workshop*, November 1994, pp. 165-168.

[12] Srihari, Rohini K. (1995). Use of Multimedia Input in Automated Image Annotation and Content-Based Retrieval. *Presented at Conference on Storage and Retrieval Techniques for Image Databases, SPIE '95*, San Jose, CA, February 1995.

[13] Beecher, Ben (1996) Personal Communication. AcIS R&D Columbia University.