Webcrawling Using Sketches

Michael S. Lew Kim Lempinen Nies Huijsmans

Department of Computer Science
Leiden University, Postbus 9512
2300 RA Leiden, The Netherlands
mlew@wi.leidenuniv.nl

Topics: distributed databases, intelligent agents, imagecrawling, compressed image databases

Abstract

The current breed of WWW search engines are systems which use agents to find and download text based information from the distributed multimedia database known as the World Wide Web. These search engines are well developed with respect to text, but typically ignore image and video content. In this paper we discuss a system which uses agents to search distributed multimedia databases over the world wide web. Operational goals of the system include the following: accessibility by anyone with a World Wide Web connection; low Response time; and usable by non-artists. Toward these ends, we developed the ImageScape system (http://ind134a.wi.leidenuniv.nl:2001) which includes MPEG based methods for compressing large image databases, optimal agents for analysis and downloading based on utility of the site information, and information theoretic techniques for sketch and representative image icon matching.

1 Introduction

Image and video databases are growing quickly due to a multitude of reasons. International digital library initiatives for digitizing and organizing paper and film libraries exist throughout the world. The WWW is becoming ubiquitous if it is not already. CD-ROMs are standard on all PCs and Digital Versatile Disks (DVD) have reached the store shelves.

However, the standard WWW search engines such as AltaVista and Lycos ignore nontext information. In this paper, we describe a search engine for images on the WWW. We cover the web robot which downloads the images, the Java based sketch interface, and the algorithm used to index the WWW color image database at Leiden. In Figure 1, we show

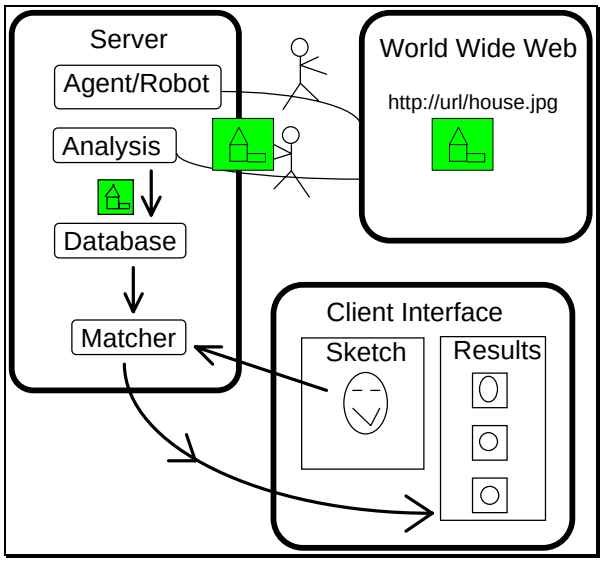


Figure 1. A diagram of ImageScape, a multimedia WWW search engine.

the fundamental relationships between the server, client, and the WWW. Agents are continuously sent to the WWW to retrieve images and videos with associated text information. When an image is brought to the server, it is analyzed for useful features, and then a thumbnail of the image and the features are stored in a relational database. When a user sends an image query from a WWW browser/client program, the query is compressed, sent to the server, and matched against the database. The best matches are then sent back to the WWW browser/client program to be displayed to the user.

In this section we gave an overview of the entire system. However, there are some interesting questions: (i) How can we maximize the download rate; (ii) When the images are downloaded, how can we maximize storage per gigabyte?; and (iii) For a given image database, how can we search for images? These questions are addressed in the next sections.

2 Optimal UtilityAgents

What is an agent? There are several types of agents which are classified based upon functionality and sophistication. A mobile agent is a software program which can hop between computers, retaining its internal state. An intelligent agent is a software robot which can perform sophisticated missions based on a knowledge base. These agents vary in their complexity from reflexive agents, which simply follow a rule base, to utility agents which perform analysis of the environmental input with their own database and assess the utility or importance of different actions. Note that a mobile agent does not have to be intelligent and an intelligent agent does not have to be mobile. For more information, see Petrie [1996] and also Braham and Comerford [1997].

Multiple agent programs need to have their activities coordinated. The program which does this is called the manager or *M*. The tasks are split up between M and the agents in that M assigns a list of goals for each agent. The agent can decide the order in which to achieve the goals depending upon the current circumstances. M does the global planning while the agents perform local planning based upon the current state of the environment.

One significant problem in searching distributed databases such as the WWW is how to maximize the download rate of useful information. Our approach was to optimize the use of the network, WWW site servers, the local workstation servers, and the need for the information. So, the question is what relevant information or inputs does M have and what choices can it make? One important factor is the bandwidth at a particular date/time in the past. A heuristic use of this factor would mean that the agent downloads information from a WWW site when the bandwidth is near maximum. Note that the heuristic uses are given to show the relevance of the factor, but they are not used in our system. However important the bandwidth is, it is also necessary to account for the previous growth of the site. Some sites are growing rapidly, while others are stable for long periods of time. Another heuristic is to give less importance to more stable sites. A variation on the growth of the site is the change in the site information. For example, newspaper related sites have the same general site structure over months, but the content in terms of articles changes daily. Furthermore, we need to address the question of how useful is the information at a site. A rough measurement of utility is made by counting the number of times a site occurs in WWW lists which describe the Top N sites. Heuristically, sites which are voted as top sites most often should be checked more frequently.

There are three states which an agent can be in

S1 Agent doing nothing & without data
S2 Agent waiting or downloading data
S3 Agent doing nothing & with data

And three processes which link S1, S2, and S3

P1 Send agent
P2 Agent waiting or downloading data
P3 Read agent

Figure 2 describes the relationships between states and processes. M's actions/controls are

(1) Choose P1;
(2) Choose P3;
(3) Terminate an agent

while an agent can choose which URL to download from a list specified by M.

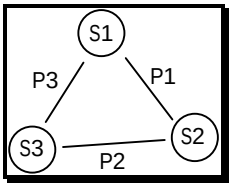


Figure 2. The relationship between the states and processes.

What is the optimal action given the input information? Since this is essentially a maximization problem, both M and the agents were modeled after a conjugate gradient maximization algorithm in that they greedily maximize the download rate of useful information.

In summary, the process of searching the WWW was split between a manager, M and agents. M has knowledge of global information about the other agents as well as the previous bandwidth to a site and the previous site changes. The agents do not know the

activities of the other agents, but they do have access to the current bandwidth and site changes in addition to the previous bandwidth and site growth information. Both M and the agents were cast as optimization problems where the goal is to maximize the download rate. This eliminated the need to create complex heuristics.

2.2 Image Database Compression

When the agents bring back images, we reduce them to thumbnails and delete the originals. This brings to mind the problem of how to maximize the number of thumbnails per gigabyte? Our general approach was modeled after MPEG video compression. In our coverage, we assume familiarity with JPEG and MPEG encoding. Please refer to Tekalp [1995] for extensive descriptions. Specifically, we implemented three database compression algorithms. First, if the new image is a copy of a database image, then we only store a pointer to the database image. Second, if the new image is a copy of a database image, but with small changes such as company logos or writing, we subtract the images and store a pointer to the database image and the JPEG compressed difference image. These two methods are particularly useful for the WWW because of the large number of image copies between sites, especially icons.

The third and most interesting method is similar to MPEG video compression. In MPEG video compression, the consecutive frames are often redundant, so the frames are often split into NxN blocks, and if a previous or future frame has a similar block, then only a pointer is stored. In a very large image database of hundreds of thousands to millions of images, it is not difficult to conceive that there could be significant redundancy between the database images. Whether it is the color of the sky or the texture of the trees, there do seem to be commonly occurring features in images. Thus, for the database compression, we split the image into NxN blocks, and search for similar blocks in other images. If a similar block is found then we only store the pointer to the block. This compression technique is useless for small picture collections (unless they are very similar images), but becomes increasingly useful for larger picture databases.

The thumbnails in our collection are designed on average to take approximately 1K per image using the JPEG quality control. For our WWW database of 7 million images, we were able to reduce the disk usage from approximately 7 gigabytes to 2.6 gigabytes. The current limitation in our system is the hard disk bandwidth. In order to reduce the image loading, we only checked images which have similar aspect ratios and similar color histograms. Thus, the 2.6 gigabytes is only an upper bound since we did not cross-index the entire database.

2.3 Matching

There are two fundamentally different ways of specifying image content in our interface. The user can draw a sketch or place representative image icons which refer to objects such as human faces, sand, trees, water, and sky. In order to find these objects, an algorithm for detection of objects in complex backgrounds is necessary. Representative work includes Yang and Huang [1994], who used a constraint based image pyramid. This method was especially computationally efficient due to the pyramidal image representation. Rowley and Kanade [1995] compared different strategies in using neural nets for detection of faces. Sung and Poggio [1995] synthesized 6 face and 6 nonface clusters using elliptical *k-means* clustering.

The object detection method used for this system was from Lew and Huijsmans [1996]. The candidate features for our system included the color, gradient, Laplacian, and texture information from every pixel of multiple scales. The method uses the Kullback relative information to find the 256 features which will minimize the misdetection rate. The primary reason why we used the Kullback method is that it is easily scalable with respect to computational efficiency. If a faster or slower server is used, then the number of features can be adjusted to reach an acceptable response time.

Shape matching between the contours of the image database and the query image is one of the fundamental areas in computer vision and thus there are a large number of useful interesting techniques. Reviewing all of the shape matching algorithms is beyond the scope of this paper. For a good but not necessarily comprehensive coverage, see Gudivada and Raghavan [1995]; Del Bimbo and Pala [1997]; and Flickner, et. al. [1995]. Some recent shape matching techniques are directed toward multiscale shape matching [Mokhtarian, Abbasi, and Kittler 1996; and Del Bimbo and Pala 1996] In our image search engine, we need to emphasize computational efficiency and roughness of the sketches.

There are a variety of problems with respect to the sketch matching. These include but are not limited to

(1) Is spatial placement of the objects in the sketch essential? If we draw a circle in the lower part of the canvas, are we looking for any circle or just circles in the lower section? and (2) How important is the background relative to the foreground (the sketch)? Regarding question (2) Suppose that the user sketches a ball, and that there exist an image with a ball against a plain background; and an image with a ball next to a little boy. Which image should be considered to be the better match? A less intuitive side-effect of question (2) is that if we implement a straightforward sum of squared difference error, then blank images would often match to the sketch image because most of the sketch image is blank. Note that many of the previous shape matching methods such as FFT coefficients assume that the image has been segmented, which is not currently a reliable automatic procedure.

There is no single answer to these questions because each user will have different preferences. In our matching algorithm, we assume that the user is drawing objects near to where he would like them to be, thus we enforce spatial positioning.

One interesting possibility would be to directly use the gradient magnitude instead of the contours. This possibility has the problem that it is not known at what edge strength the user is drawing the contours of the sketch. Thus we chose not to persue this direction until we enhance the sketch interface with variable width lines.

Assuming that the user is specifying the contour of an object, then it is reasonable to try to match the sketch with the contour map of an image. For this implementation we chose to use the Sobel operator [Ballard and Brown 1982] in conjunction with a Gaussian blurring filter for finding the edge/contour maps. Edges have the advantages that they reduce dependency on image contrast and varying lighting effects.

Another consideration is which scale should be used for the matching? Regarding computational efficiency, as the resolution of the image is halved, the amount of computational resources required is quartered. Furthermore, as the resolution of the image decreases, we gain more translation invariance. However, as the resolution increases, the discriminability of the contour maps increases giving us potentially more accurate results. From our own feasibility experiments, the resolution of 20x20 was the minimal resolution in which a wide variety of sketches could be detected. However, the computational aspects still required further enhancement.

Trigrams [Huijsmans, et al. 1996] were postulated as the image equivalent of textual trigrams in fast text matching. They are related to Linear Binary Patterns (LBP) [Ojala, et al. 1996, Wang and He 1990], but are founded on the edge map of an image instead of intensity space. Trigrams are based on 3x3 texel patterns which occur in edge maps of images. If each pixel in the 3x3 texel is considered to be a single binary digit, then there are 9 binary digits which corresponds to 2^9 or 512 unique trigrams. Intuitively, each trigram corresponds to one unique 3x3 edge mask.

The discriminability power of trigrams was found to be sufficient even on specialized image databases such as the Leiden 19th Century Portrait Database [Huijsmans, et al. 1996]. Furthermore, with respect to sophitication of the method, the trigrams were found to have comparable accuracy to the Virage datablade[Huijsmans, Lew, and Denteneer 1997], which is an eminent commercial product for finding similar images. Thus, we used the trigram method as a front-end to the 20x20 template matching by finding the top 1 percent matches (in our database this was 1000 matches) using the trigrams and then resorting using the Kullback relative information matching described next.

In order to allow the user to adjust the effect of the background, we examined the error due to mismatching sketch pixels, E_s, and the error due to mismatching background pixels, E_n. A user selectable weight, w, is selected so that the effect of the background can be completely removed if desired by the user. The formula is shown below:

$$E = w*E_s + (1-w)*E_n$$

How do we determine E_s and E_n? One possibility is to count the number of sketch pixels which coincide with the database edge image, divide by the total number of edge pixels and subtract from one. This would be the percentage sketch pixel error. However, this method has the drawback that it does not have account for uniformity of matching coverage. Uniformity of matching coverage refers to the human visual system preference for a database image which covers the query sketch uniformly as oppose to partially. An example is shown in Figure 3 where DB Image 1 would match better to the query if we only count matching pixels, but DB Image 2 “looks” more like a V than DB Image 1

It has already been found that application of the Kullback [1959] relative information [Lew and Huang 1996], toward finding the most informative pixels of an image results in a weighting matrix which reduces the weight of neighboring pixels. Intuitively, this occurs

because on average, pixels are highly correlated, so if the value of a particular pixel is known, then less information is gained by searching its neighbors. The weighting matrix from the integration of a Markov random field with the Kullback relative information was found to be a logarithmic weighting of the pixel values [Lew and Huang 1996], and this weighting matrix was implemented for E_s.

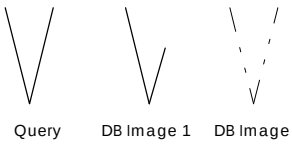


Figure 3. Two different matches for a query: partial and uniform covering

In summary, the indexing algorithm integrates the trigram method with Kullback weighting. The trigrams for the database images are first computed and stored. When the user sketch arrives at the server, the band-pass trigrams are computed [see Huijsmans, et al. 1996] and used to compute the top 1% matches. At this point, we had to decide whether spatial placement of the object was essential. We chose to implement weighting by spatial placement by resorting using Kullback templates. The top 20 matches are shown to the user.

Note that there is no segmentation done except for the calculation of the trigrams. The algorithm returns images which have roughly the same placement of objects as the sketch. If the user draws a circle in the lower part of the canvas, then we assume he is interested in images which have circles in the lower section.

3 Experiments

In this section, we describe experiments which were performed to test the efficacy of the different modules of our system. For the image database compression, we generated a graph of the probability of finding a similar 16x16 block with respect to the size of the database in Figure 4. For this experiment, we counted a block as similar if the average absolute difference was less than 3% in each of the color components.

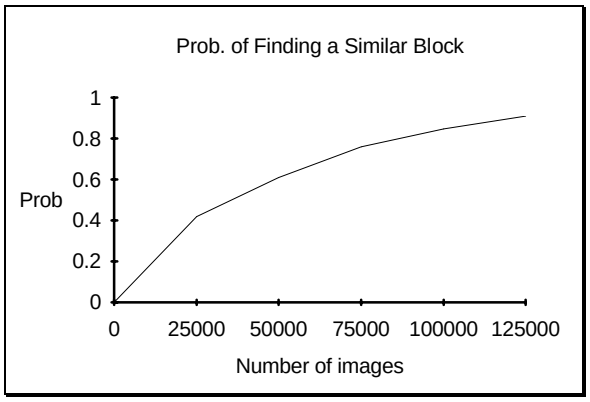


Figure. 4. Probability of finding a similar (16x16) block with respect to the number of images.

This graph states that even for image databases near 40,000 images, the probability is greater than half of finding a suitable block for compression, and near 120,000 images, the probability is greater than 0.9.

Any object detection method creates an operating characteristic between false alarms and misdetections. One can have few false alarms and many misdetections, or few misdetections and many false alarms. As a design decision, we chose to emphasize few false alarms. Note that selecting a point on the operating characteristic is necessary in every machine detection task. Using 1,400 ground truthed images, and a false alarm rate of 0.05 per image, our misdetection rates were 0.08 for sky, 0.14 for water, 0.19 for human faces/skin, 0.26 for trees/grass, , and 0.29 for stone/sand. In Figures 5 and 6, image queries using the representative image icons and their results are shown.

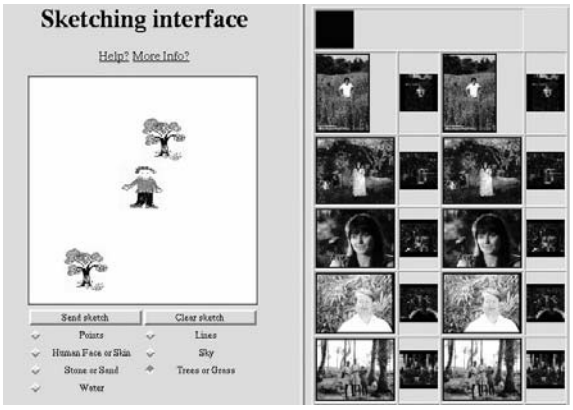


Figure 5. An image query using representative image icons, human face and trees/grass.

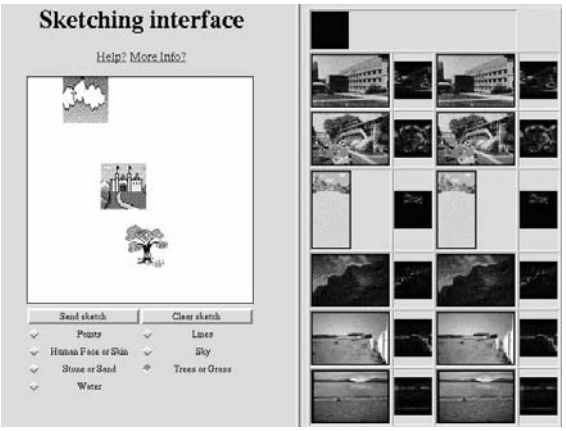


Figure 6. An image query using representative image icons, sky, sand/stone, and trees/grass.

For the sketch experiments, we created a standard sketch probe gallery of 15 images of varying complexity. Four of these probe images are shown in the results below in Figures 7 and 8. Thirty users were selected and each user was given one minute to sketch each image. Thus 450 test queries were used in the experiment. Since the sketch interface was designed for nonskilled users, we only selected nonartists. The time limit of one minute was selected because the sketch interface was meant for drafts, not finished nor precise drawings. We counted a sketch as a misdetection if it failed to appear in the top 20 result images. For the results in Figures 7 and 8 (see end of article for figures), the integrated trigram/Kullback algorithm was used to process the queries.

The misdetection rate for our probe gallery was 0.14. The majority of the misdetections occurred because the sketch probes were not aligned near the center of the image. To compensate for this we are adding an option to have the indexer automatically center the sketch.

The response time for a test database containing 100, 000 images (the full WWW database which is indexable by text is 7 million images. The test database for the sketch and image icon search algorithms is 100,000 images) and is for a single user on an SGI Indy (R5000, 150 MHZ) was 41 seconds for the Kullback template algorithm, and was reduced to less than 3 seconds for the integrated trigram/Kullback algorithm.

In general, the subjective quality of the user sketches was very low. Also, when the query image was not detected, the results were usually intuitive via the thresholded version of the image. For instance, in Figure 7(b), the planet in the 2nd row, 1st col, does look similar to the thresholded query image.

4 Conclusions

Current internet search engines do not allow the inclusion of image information for queries. This paper describes a system for searching large distributed multimedia databases using agent technology. In the course of developing the system, several problems were addressed: optimal utility agents, image database compression, and visual information retrieval using sketches and representative image icons. The techniques for the agents, database compression, and sketch/icon retrieval are enhancements or adaptations of current or published techniques. In this article, we presented the working system and tested it on databases from the WWW.

World Wide Web Demo

For the sketch based search engine, use Netscape 3.0 at URL

http://ind134a.wi.leidenuniv.nl:2001/

References

Ballard, D. and C. Brown, Computer Vision, Prentice Hall, 1982.

Braham, R. and R. Comerford, "Sharing Virtual Worlds: Avatars, Agents, and Social Computing," IEEE Spectrum, pp. 18-51, March 1997.

Flickner, M., H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by Image and Video Content: The QBIC System," Computer, IEEE Computer Society, pp. 23-32, Sept. 1995.

Del Bimbo, A., and P. Pala, "Shape Indexing by Multi-scale Representation," Proc. of the First Int. Workshop on Image Databases and Multi-Media Search, August, Amsterdam, pp. 43-50, 1996.

Del Bimbo, A., and P. Pala, "Visual Image Retrieval by Elastic Matching of User Sketches," IEEE Trans. Pattern Analysis and Machine Intelligence, February, pp. 121-132, 1997.

Gudivada, V. N., and V. V. Raghavan, "Finding the Right Image, Content-Based Image Retrieval Systems," Computer, IEEE Computer Society, pp. 18-62, Sept. 1995.

Huijsmans, D. P., S. Poles and M. Lew, "2D Pixel Trigrams for Content-Based Image Retrieval," Proc. of the Int. Workshop on Image Databases and Multi-Media Search, Amsterdam, August 22-23, 1996, pp. 139-145.

Huijsmans, D. P., M. Lew, and D. Denteneer, "Quality Measures for Interactive Image Retrieval with a Performance Evaluation of Two 3x3 Texel-based Methods," (to be published in) International Conference on Image Analysis and Processing, Florence, Italy, September, 1997.

Kullback, S. "Information Theory and Statistics," Wiley, New York, 1959.

Lew, M. and N. Huijsmans, "Information Theory and Face Detection," Proceedings of the International Conference on Pattern Recognition, Vienna, Austria, August 25-30, 1996, pp.601-605.

Lew, M. and T. Huang, "Optimal Supports for Image Matching," Proc. of the IEEE Digital Signal Processing Workshop, Loen, Norway, Sept. 1-4, 1996, pp. 251-254.

Mokhtarian, F., S. Abbasi, and J. Kittler, "Efficient and Robust Retrieval by Shape Content through Curvature Scale Space," Proc. of the First Int. Workshop on Image Databases and Multi-Media Search, August, Amsterdam, pp. 35-42, 1996.

Ojala, T., M. Pietikainen and D. Harwood, "A Comparative Study of Texture Measures with Classification Based on Feature Distributions," vol. 29, no. 1, pp. 51-59, 1996.

Petrie, C. "Agent-based Engineering, the Web, and Intelligence," IEEE Expert, Dec. 1996.

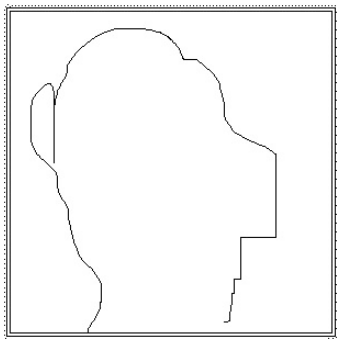
Rowley, H, and T. Kanade, "Human Face Detection in Visual Scenes," *Carnegie Mellon Computer Science Technical Report CMU-CS-95-158R*, November, 1995.

Sung, K. K., and T. Poggio, "Learning Human Face Detection in Cluttered Scenes," *6th International Conference on Computer Analysis of Images and Patterns*, Prague, pp. 432-439, 1995.

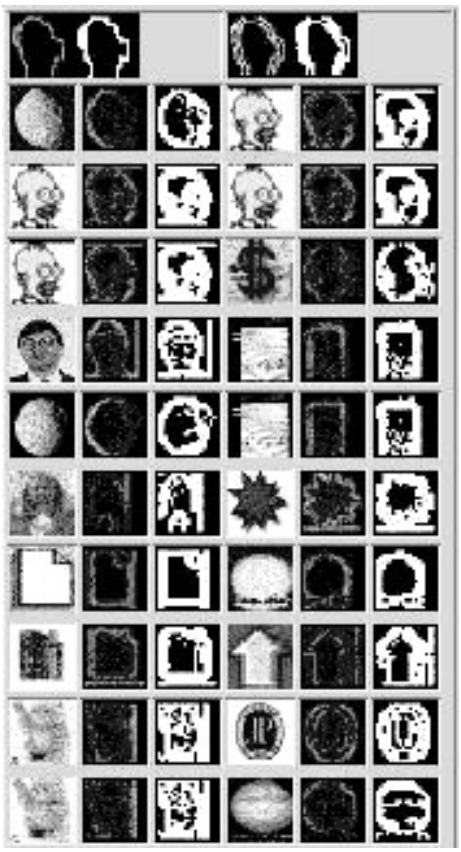
Tekalp, A. M., **Digital Video Processing**, Prentice Hall, New Jersey, 1995.

Wang, L. and D. C. He, "Texture Classification Using Texture Spectrum," Pattern Recognition 23, pp. 905-910, 1990.

Yang, G., and T. S. Huang, "Human Face Detection in a Complex Background," *Pattern Recognition*, 27(1):53-63, 1994.

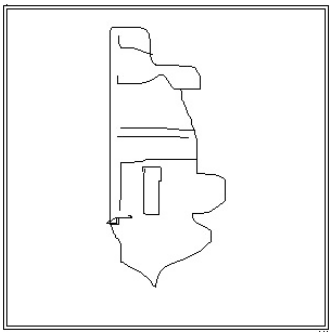


(a)

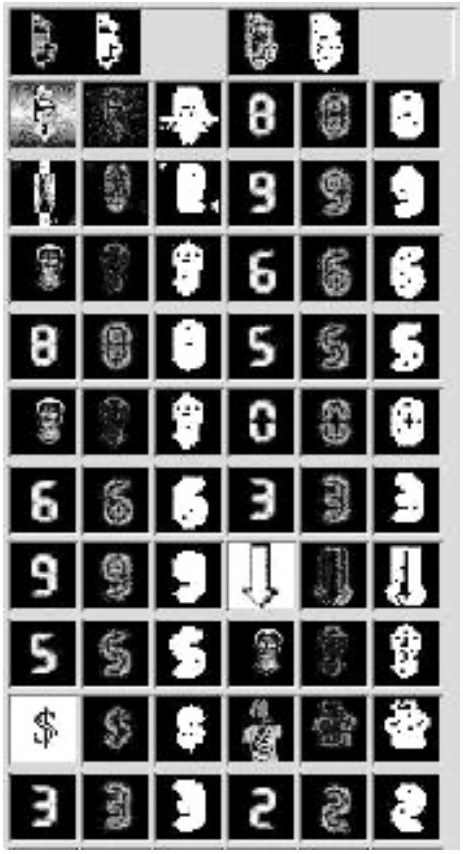


(b)

Figure 7. Search for Image: Homer (3rd row, 1st col. in (b)). Sketch (a) and results (b).



(a)



(b)

Figure 8. Search for Image: Opus (2nd row, 1st col. in (b)). Sketch (a) and results (b)