Content Based Image Search over the World Wide Web

Vibha Rathi
Indian Institute of Technology,
Kharagpur.
vrathi@cse.iitkgp.ernet.in

A.K.Majumdar
Indian Institute of Technology,
Kharagpur.
akmj@cse.iitkgp.ernet.in

Abstract

Most web pages typically contain both images and text. However, most current search engines index documents based on text only. In order to facilitate effective search for images on the web, we need to complement text with the visual content of the images. We often look for images containing specific objects having some particular spatial and topological relations among them. In this paper, we describe a system which enables the user to effectively search for images using the image content information including color, component objects and their relations in addition to associated text.

1. Introduction

Considering the highly visual and graphical nature of the world wide web, the number of image search engines is very limited. Several image search engines like Google Image Search, WebSEEk [7], WebSeer [8] etc. have been developed in the last few years. While earlier image search engines used text only, WebSEEk and WebSeer use content based information like color, texture, shape etc. for indexing and query.

But we often look for images containing specific objects having specific spatial and topological relations among them. For instance, consider an advertising professional looking for images of a red car flanked by a tree on its left or an archaeology student looking for images of monuments having arches or domes. The currently available search engines cannot cater to such queries as they lack semantic description of an image as a collection of objects with relations amongst them. In this paper, we present a system that indexes images based on its component objects and supports queries based on them.

The main components of the system include (1) a crawler that works offline and scours the web for images, automatically extracts features like color, associated text, size, etc. and then relies on an administrator for manually identifying the objects and labeling, (2) the image database which is a repository of the collected information and (3) a query module which searches the database of images matching a given query image/ sketch.

The rest of the paper is organized as follows. Section 2 describes the process of image collection from the web. Section 3 describes the model used to describe an image as a collection of objects and relations between them. Section 4 describes the design of the image database. The image loading and querying mechanism is detailed in Section 5. In Section 6, we provide the results of preliminary evaluation of the system.

2. Image Collection from the Web

The image collection process is carried out by a semi-automated crawler or web robot which autonomously traverses the hypertext structure of the web following links and downloading images. The basic idea is to recursively traverse referenced links found in HTML pages.

A crawler or spider is the essential backend of any search engine. It collects information from the web, classifies it and creates appropriate directories or databases which can be later searched when there is a query from the user. This enables faster response to the query rather than searching the web afresh every time there is a query.

2.1 Focused Crawling

The sheer diversity and volume of content on the internet coupled with the growing community of intelligent users who use the web for serious search, requires specialized search tools that delve deeply into a particular topical area.

For the above reasons, our system allows the crawler to focus on a particular topical area or subject class (for eg. Landscapes or Animals) to a certain extent using the text associated with a given URL. We use a static set of keywords to characterize a subject class. The relevance of a URL is calculated by matching the text associated with it with the keywords specified for the subject class and determining its relevance index. The link denoted by the URL is followed if and only if the relevance index is greater than a predetermined threshold value. The process of keyword extraction and relevance index calculation is detailed in the sections below. Thus, we can easily change the focus of the crawler by using different sets of keywords corresponding to different subject classes.

2.1.1 Key term extraction

When the spider finds a URL it extracts the text associated with it which includes –

- The URL itself
- The alternate text associated with it
- The hyperlink text (if any)

Key terms are extracted from this text by chopping the text at non alpha characters. For instance consider the HTML tag:

<A HREF="http://www.photonet.com/images-2/tn31.jpg" ALT="Temple in India"> Contains gold statue </A>

The following key words can be extracted from it –
photonet, tn, Temple, in, India, Contains, gold, statue.

Common keywords like http, jpeg, www etc. are eliminated.

2.2.2 Relevance Index Calculation

Each of the keywords k_e extracted above is matched against each keyword k* of the keyword set after ignoring case. We define the keyword match value of k_e and k* as –

Keyword Match (k_e, k*) = Number of characters in the same position in both strings/ Max (Length(k_e), Length(k*))

The total match value of a keyword is –

$$\text{Total\_Match}(k_e) = \sum_{k^*} \text{Keyword\_Match}(k_e, k^*)$$

The relevance index is then given by –

$$\text{Relevance Index} = \sum_{k_e} \text{Total Match}(k_e) / \text{Number of extracted keywords}$$

The URL is processed further only if its relevance index is greater than 0.1. This threshold was determined after a lot of experimentation. This approach though simple, seems to work well. More detailed approaches to focused crawling are illustrated in [2].

2.2 Crawler Implementation

The chief components of the crawler are illustrated in the figure below. Starting with the seed URL(s), the crawler takes one URL at a time from the URL buffer. It classifies the URL into a HTML page, an image file or others using its MIME type.

If it is an image URL, the image is downloaded, its content information is extracted, and after manual object identification by the administrator it is added to the database.

If it is an HTML page, it is downloaded using the *HTML Scanner*. This is basically a specialized input stream that also looks for specific tags (containing URLs embedded in them) like <IMG SRC="">, <A HREF=""> etc. Each instance of a Scanner has an *HTML Observer* associated with it. This is a call back interface that is implemented by the Spider. It defines one function for each specific tag. Once a tag is encountered the corresponding function in the Spider is invoked and the appropriate action is taken. In most cases it is just to check the relevance of the URL found and add it to a *to do list*. Finally, before adding this URL to the URL Buffer, we check if it has already been processed. This is done by maintaining a Hashtable of processed URLs – the *Done table*. It has a fixed size and flushes out old entries keeping only the most recently encountered URLs.

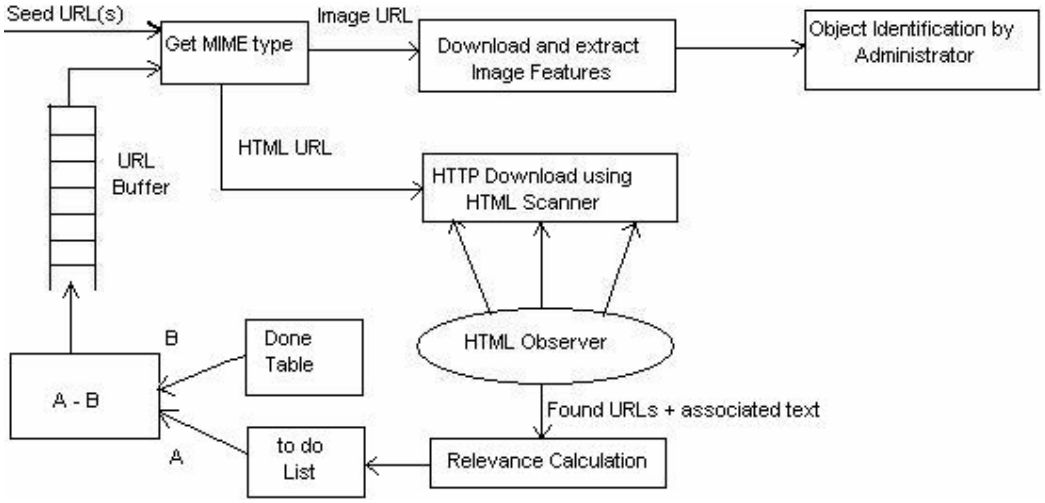

Figure 1: Basic components of the Crawler

3. The Image Retrieval Model

An image is basically described as a collection of objects having certain spatial and topological relations amongst them. The spatial and topological relations that may exist between two objects of an image are described below. These are automatically calculated by the system after the user identifies the objects.

3.1 Spatial Relations

The directional relations used in our model are the strict positional relations **left, right, above** and **below**. An object A is said to be to the left of another object B if and only if each pixel of A is to the left of each pixel of B. The same applies for each of the other spatial relations. Two objects may or may not have a spatial relation (if they overlap or touch). Also, the same pair of objects may also satisfy two spatial relations. For instance, object A may be to the left of and above object B. *(left, right)* and *(above, below)* pairs of relations are duals.

3.1 Topological Relations

Topological relations always exist between two objects and exactly one relation may exist between any two objects. It can be shown that in a 2-D planar region, a binary topological relation, existing between two regions (without holes), can be determined uniquely by analyzing the intersections (or non-intersections) between their interiors and boundaries. The framework for describing the topological relation between two objects A and B is the ordered set of these four intersections, called the ***4-Intersection Model*** [4]. It is represented as a 2x2 matrix as shown below.

$$R(A, B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \delta B \\ \delta A \cap B^\circ & \delta A \cap \delta B \end{pmatrix}$$

The notation A° stands for the interior area of the region A, while the notation δA stands for the boundary of the region A. Thus eight different topological relations may be defined between two regions as shown below:

Disjoint
$$\begin{pmatrix} \emptyset & \emptyset \\ \emptyset & \emptyset \end{pmatrix}$$

Meet
$$\begin{pmatrix} \emptyset & \emptyset \\ \emptyset & \neg\emptyset \end{pmatrix}$$

Equal
$$\begin{pmatrix} \neg\emptyset & \emptyset \\ \emptyset & \neg\emptyset \end{pmatrix}$$

Overlap
$$\begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset \end{pmatrix}$$

Contains
$$\begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset \end{pmatrix}$$

Inside
$$\begin{pmatrix} \neg\emptyset & \emptyset \\ \neg\emptyset & \emptyset \end{pmatrix}$$

Covers
$$\begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \emptyset & \neg\emptyset \end{pmatrix}$$

Covered By
$$\begin{pmatrix} \neg\emptyset & \emptyset \\ \neg\emptyset & \neg\emptyset \end{pmatrix}$$

It is easy to see from the above definitions that the relations *disjoint, meet, overlap* and *equal* are symmetric, while the (*covered-by, covers*) and (*inside, contains*) pairs of relations are duals.

3.2 The Graph Model

Using the above definitions, we may describe an image as a labeled graph with each image corresponding to an object in the image. An edge between two nodes, say A and B, is labeled by the positional (if any) and topological relations that hold between the objects corresponding to A and B. We also associate with an edge, the Euclidean distance between the centroids of the two objects it relates. Thus, the problem of finding image similarity is reduced to one of graph matching.

3.3 Image Similarity Measure

With the logical representation of images thus defined, we are now in a position to examine the problem of associative retrieval of images from image databases. In order to support such retrievals, it is necessary to define a suitable similarity measure between the query image and the target image stored in the database. The objects of the query image may be assigned weights to specify their relative importance. In image database literature, several such similarity measures have been proposed for associative retrieval of images [5]. In our model, based on [6], we define the similarity measure (SIM) between two images I_1 (query image), and I_2 (an image stored in the database), as follows.

Here, the notations used are:

P = {left-of, right-of, above, below}, the set of positional relations between the objects,

T = {disjoint, meet, equal, overlap, covered-by, covers, inside, contains}, the set of topological relations between the objects,

$G(O_I, E_I)$ = the graph of an image I, where O_I is the set of nodes and E_I is the set of edges.

An edge $e \in E_I$ between two objects $A, B \in O_I$ is associated with the values of the positional $P_e \in P$, and topological relations $T_e \in T$ that hold between the objects A and B, and the Euclidean distance D_e between the centroids of the objects A and B.

Given two images I_1 and I_2 with the associated graphs $G(O_1, E_1)$ and $G(O_2, E_2)$, to find the similarity between them, we first need to find a suitable mapping between the objects in the images. For example, if both the images have two objects, we may match any of the objects from the first image to any object in the second image. Such an

association can be expressed in terms of a graph homomorphism:

$$\Psi: G(O_1, E_1) \rightarrow G(O_2, E_2)$$

The graph homomorphism ψ maps a node (object) $A \in O_1$ to a node $\psi(A) \in O_2$ and similarly maps an edge $e_1 \in E_1$ to an edge $\psi(e_1) \in E_2$. Once a mapping is selected, we can measure the similarity between I_1 and I_2. Such a similarity measure is assumed to consist of following four components:

3.4.1 Object Similarity

It determines the extent of similarity between the objects A and $\psi(A)$, for all $A \in O_1$. Accordingly, it is,

$$Obj\_Sim(O_1, \psi(O_1)) = \sum w_a \chi(A, \psi(A)) / \Sigma w_a$$

where $\chi(A, B)$ is a similarity between two objects A and B, which is based on the similarity in color of the two objects, and w_a is the weight of object A. The color similarity is computed by comparing the average R, G and B color values of the objects A and B.

3.4.2 Positional Similarity

This determines to what extent the positional relations between objects in I_1 match those between the corresponding objects in $\psi(I_1)$. It is given by,

$$Positional\_Sim_\psi(E_1, \psi(E_1)) = \sum_{e \in E1} \sum_{s \in S} equal(\chi_s(A, B), \chi_s(\psi(A), \psi(B)) / \sum_{e \in E1} \sum_{s \in S}$$

where the edge $e \in E_1$ is assumed to be between the nodes(objects) A and B in O_1 and $\chi_s(A, B)$ is the positional relation between the objects A and B, while the equal function is true only if its parameters are equal, that is, the positional relation between A and B is the same in I_1 and $\psi(I_1)$.

3.4.3 Topological Similarity

It determines to what extent topological relations between the objects in I_I match with those between the corresponding the objects in $\psi(I_I)$.

$$Topo\_Sim_\psi(E_l, \psi(E_1)) = \sum_{e \in E1} \sum_{t \in Te} equal(\chi_t(A, B), \chi_t(\psi(A), \psi(B)) / \sum_{e \in El} \sum_{t \in Te}$$

where $\chi_t(A, B)$ is the topological relation between the objects A and B.

3.4.4. Distance Similarity

It gives the similarity in distance between a pair of objects in I_1 and that between the corresponding objects in $\psi(I_1)$.

$$Dist\_Sim_\psi(E_1, \psi(E_1)) = \sum_{e \in E1} \chi_{dist}(e,\psi(e)) / \sum_{e \in E1}$$

where for a pair of edges e_1 and e_2 between the pair of objects (A_1, B_1) and (A_2, B_2) respectively, $\chi_{dist}(e_1, e_2) = e^{-d}$ and d is the absolute value of the difference in Euclidean distance between the centroids of A_1, B_1 and A_2, B_2.

3.4.5 Final Similarity Measure

The four similarity measures defined above are combined to find the similarity between the image I_1 and its homomorphic map $\psi(I_1)$ according to the following expression.

$$SIM_\psi(I_1,\psi(I_1)) = \alpha\ Obj\_Sim_\psi(O_1, \psi(O_1)) + \beta\ Spatial\_Sim_\psi(E_1, \psi(E_1)) + \gamma\ Topo\_Sim_\psi(E_1, \psi(E_1)) + \delta\ Dist\_Sim_\psi(E_1, \psi(E_1))$$

where $\alpha + \beta + \gamma + \delta = 1$ whereas α, β, γ and δ are the Object Factor, Spatial Factor, Topological Factor and Distance Factor respectively.

Finally, the similarity between any pair of images I_1 and I_2 is defined with respect to the best mapping by:

$$SIM(I_1, I_2) = \max_\psi (SIM_\psi(I_1,\psi(I_1)))$$

The problem of associated retrieval of images has thus been reduced to finding a best possible mapping between the graph of the query image and that of the target images in the database using the above functions.

4. The Image Database

The information extracted form the image including its color histogram, size and file type (gif/jpeg) is stored along with the identified objects and a thumbnail (130 x 130 pixels) in the image database. The image itself is not stored, it is only downloaded for feature extraction. The design of the database is basically object-oriented.

The database consists of the following tables –

- **Images** – Stores the image information including color, URL, width, height and number of objects.
- **ObjRelns** – Stores the relations among two objects ofan image and the Euclidean distance between their centroids.
- **Category –** The object categories are organized in a hierarchical tree-like structure. This table basically stores each category and its immediate parent. For e.g. Category Nature is the parent of Category Tree and Category Sky. The user may dynamically add a new category or delete an existing one during the image loading phase.
- **Obj < category > tables** – for each of the object categories. These tables store the objects belonging to the particular category. The tables are dynamically created/deleted as and when a new category

is added/ removed. The object attributes like parent image, color and total number of pixels is stored along with each object.

In order to make the retrieval process faster, indexes are defined on the database.

5. Image Loading and Query

These are the two front ends of the system. The Image Loader is just an interface to the crawler and is used for populating the database while the Image Query system forms the user interface of the search engine.

5.1 Image Loading

The crawler works offline and periodically scours the web for URLs. As and when it finds an image URL it extracts some features of the image automatically and then presents it to an administrator for object identification. The user identifies the objects by drawing contours on the image and assigns categories to each of the identified objects. The system then automatically computes the spatial and topological relations among the identified objects and adds all this information to the database.

5.2 Image Query

The system supports two query modes – query using a hand sketch and query using a sample image. In the first mode, the user draws the objects in appropriate colors on a drawing board using a toolbar, keeping in mind the spatial and topological relations he requires. Each object is also assigned a category. The relative importance of different objects may be specified by assigning integer weights to them. In the second query mode the user identifies the objects on a sample image file or URL instead of sketching them manually. Figure 2 shows a sample query using an image file. The user identifies three objects and assigns them the categories – sunrise, lake and rock. Here we are looking for objects having sunrise *above* a lake *containing* a rock *inside* it. All objects in the image are given equal weight. Also, all four similarity factors are given equal importance.

5.2.1 Query Parameters

The user can specialize the search further by specifying several query parameters including maximum height and width of the result images, relative weights of the four similarity factors namely object color matching, spatial relations, topological relations and distance between objects. Moreover, the spatial relations are broken up into horizontal and vertical positional relations, for which weights can be given separately. Finally, the number of images that the user wishes to see as the result of the query or the *Result Size* can also be changed.

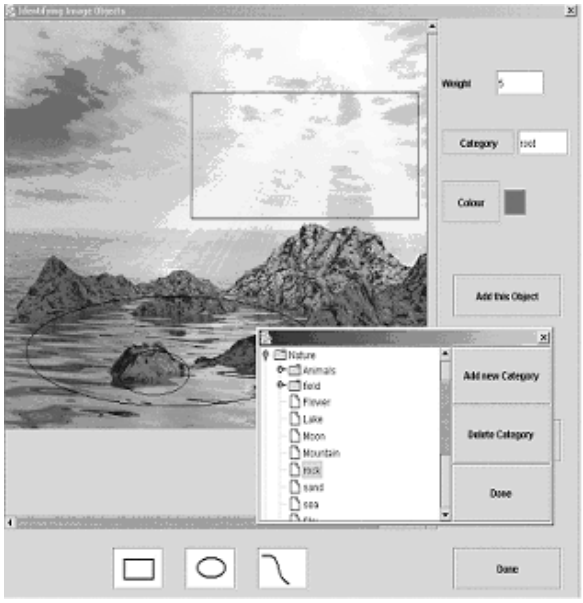

Figure 2: A sample query using an image file.

5.2.2 Search and Retrieval

The images are first filtered based on the maximum height and width constraints. Then, the image histograms are matched. Next, the images in the database are matched with the query image/sketch in accordance with the weights assigned to the different similarity factors and the final similarity measure is computed for each image. These are then ranked by maximum match values and *Result Size* image URLs and their thumbnails (which were earlier stored in the database) are presented to the user as the result set. The user may then click on any of the result images for a quick download. The result images for the query in Figure 2 are shown in Figure 3 below.

6. Results

In this section, we present the results of running some sample queries. By changing the search parameters we can alter the results. For instance, one of the query objects may be given greater weight, or object matching may be given greater importance than spatial and topological matching. The following are the results of a query by sketch containing two objects namely a Monument and the Sky. 15% of the images in the database were retrieved (i.e. had at least one of these two objects). The effect of varying the query parameters is shown in Table 1 below.

The main bottleneck to scaling up the system is that it requires an administrator to manually identify the objects. Hence, the database can only be built up gradually. To test

Figure 3: Results of the query shown in Figure 2

the scalability, we automated this process by random object selection and assignment to one of 45 categories.

We catalogued 10096 images. The storage requirement of the image database (created using Microsoft SQL 2000) including 4 KB thumbnails of each image, was found to be 780 MB. The response time for a query containing three objects was around 7.5 seconds.

Table 1: Effect of varying query parameters

Object Weights	Similarity Factor Weights	Search results
Both objects have equal weight	Object Color Matching has more weight than other factors	Both objects - 6% Sky – 12% Monument – 8%
Monument has twice the weight		Both objects - 6% Sky – 6% Monument – 10%

7. Conclusions

We have presented an effective content based image search engine that catalogues the plethora of images on the web based on text, visual data and component objects and relations between them. Any common user of the internet looking for images containing specific objects will find this system very useful.

In its current version, the system requires manual object identification and labeling which cannot be fully automated. This problem can be partially alleviated by using image segmentation algorithms to automatically divide the image into its main component objects which can then be manipulated by the administrator. The BLOBWORLD system [1], for instance uses automatic segmentation to divide the image into regions or blobs which are then matched. Another approach described in [3], also attempts automatic categorization or object recognition using template matching and EM based learning.

Also, the text associated with the image may be used to group images into hierarchical subject classes. That is, as in text based engines like Google and Yahoo, we can allow the user to focus his search by providing an effective directory-based navigation.

Finally, we also plan to add texture and shape to the content information about each image.

8. References

[1] Chad Carson, Megan Thomas, Serge Belongie, Joseph M. Hellerstein, Jitendra Mallik, "Blobworld: A System for region-based image indexing and retrieval" *Visual Information Systems. 1999*

[2] Soumen Chakrabarti, Martin Van Den Berg, Dom Byron, "Focussed Crawling: A New Approach to Topic Specific Web Resource Discovery*", WWW8 Conference*,1999.

[3] P. Duygulu, K. Barnard, J.F.G. de Freitas and D. A. Forsyth, "Object Recognition as Machine Translation :Learning a lexicon for a fixed image vocabulary", *ECCV ,2002.*

[4] M. Egenhofer, J. Herring, "Categorizing Binary Topological Relations between Regions, Lines and Points in Geographic Databases", *Department of Survey Engineering, Technical Report, University of Maine*, 1990.

[5] E. A. El-Kwae, M. R. El-Kwae, "A Robust Framework for Content-Based Retrieval by Spatial Similarity in Image Databases", *ACM Transactions on Information Systems*, Vol. 17, No. 2, April 1999.

[6] A. K. Majumdar, Indrajit Bhattacharya, A. K. Saha, "An Object Oriented Fuzzy Data Model for Similarity Detection in Image Databases" – to appear in *IEEE Transactions on Knowledge Engineering.*

[7] J. R. Smith, S. F. Chang "An Image and Video Search Engine for the World-Wide Web", *IS&T/SPIE Proceedings*, *Storage & Retrieval for Image and Video Databases V*, February 1997.

[8] M. J. Swain, Charles Frankel, Vassilis Athitsos, "WebSeer - An Image Search Engine for the World Wide Web, *IEEE Conference on Computer Vision and Pattern Recognition*, 1997.