

# Atlas WISE: A Web-Based Image Retrieval Engine<sup>1</sup>

M.L. Kherfi, D. Ziou, D. Brahmi,  
DMI, Université de Sherbrooke, Sherbrooke,  
Qc, Canada, J1K 2R1  
Email: {kherfi, ziou, brahmi@dmi.usherb.ca}

and A. Bernardi  
Laboratoires universitaires Bell, 2020  
rue University, 25<sup>e</sup> étage, Montréal, Qc,  
Canada, H3A 2A5  
Email : alan.bernardi@bell.ca

## Abstract

*The huge number of images available on the World Wide Web renders it necessary to possess retrieval engines that allow users to retrieve the sought images in a reasonable time. However, the most existing engines are devoted to text, omitting visual and multimedia information. We believe that image and visual media retrieval is very important and will take the place in the next years. In this paper, we present a new image retrieval system for the World Wide Web called Atlas WISE. We explain the important aspects of our system such as image gathering, feature extraction, optimization and retrieval, as well as its client-server architecture. Atlas WISE is based on a new feature selection algorithm that we proposed in a previous work and proved that it realizes a good performance. Furthermore, due to its extensible architecture, Atlas WISE allows to combine text-based image retrieval with content-based retrieval in a sake of retrieving images either by low-level descriptors or high level concepts.*

## 1. Introduction

Since its appearance in 1991, the World Wide Web has become a great source of information. This information takes different formats including text, sound, and visual data. The great paradox of the Web is that the more information is available about a given subject, the more difficult is to locate the needed information in a reasonable time [1]. To overcome this problem, many retrieval systems (called also retrieval engines) has appeared in the last years; however, the most of them are text-retrieval oriented. Concerning image and multimedia retrieval from the Web, few engines have appeared and most of them are primitive prototypes developed in universities. Known systems include Image Rover [3], developed in Boston university. The system Diogenes [4], developed in the university of Illinois at Chicago, has been especially designed for human facial images. WebSeer [5], developed in the university of Chicago, retrieves images and tries to distinguish between photographs and graphics. Another system that tries to distinguish between photographs and graphics is PicToSeek [6], developed in the university of Amsterdam. The system WebSeek [7], developed in Columbia university, addresses the issue of retrieving videos in addition to images from the World Wide Web. ImageScape [8], developed in Leiden university, Netherlands, introduces the concept of queries based on icons and user-drawn sketches. A more detailed study on visual information retrieval from the World Wide Web can be found in [1]. We believe that to make an effective use of the huge quantity of visual information available on the World Wide Web, more attention must be given to such an issue. This led us to develop a new Web image retrieval engine that we called Atlas WISE, as an abbreviation of Atlas Web Image Search Engine. Atlas WISE possesses many specificities that distinguish it from other existing engines. First, it is based on a new feature selection algorithm that we proposed in [2]. This algorithm allows to automatically determine the importance of each image feature without asking the user to specify it explicitly. Second, the open architecture of our system allows to add any other features without the need to modify the optimization/retrieval algorithm. Third, it allows the combination of low-level features such as color and texture with high-level concepts depicted by

---

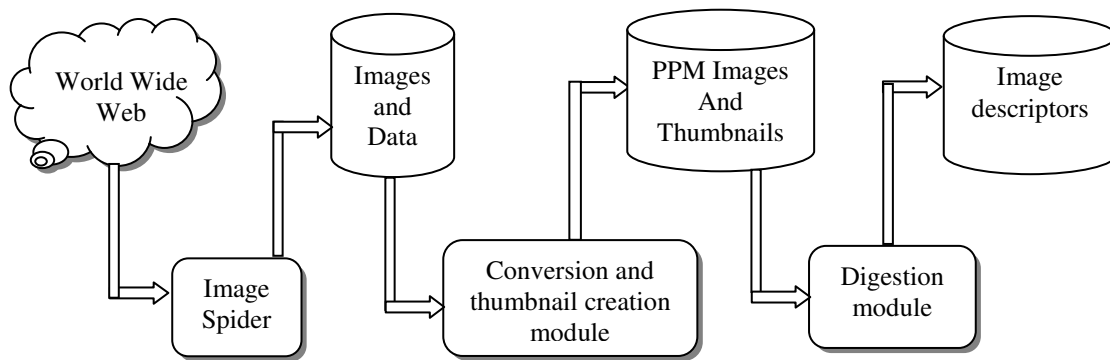
<sup>1</sup> The completion of this research was made possible thanks to NSERC and Bell Canada's support through its Bell University Laboratories R & D program.

images. This allows especially to narrow what is called the semantic gap, i.e., retrieving images using semantic concepts.

In this paper, we present the main aspects of our Web retrieval engine. The paper is organized as follows: in Section 2, we present the general structure of Atlas WISE with a brief explanation of each component. In Section 3, we detail the modules participating in preprocessing phase, namely Image Spider, Image conversion and thumbnail creation module, and Image digestion module. Details on retrieval phase are given in Section 4, where we give the steps of communication between client and server and we explain our retrieval algorithm. Basic principle of our feature selection algorithm, its mathematical formulation and optimal parameters are given in Section 5. In Section 6, we give some experimental results; and in Section 7, we introduce the semantic-based retrieval and how it can be integrated into the retrieval engine. We finish the paper with some concluding remarks.

## 2. The general structure of Atlas WISE

We can distinguish two main parts of the system: the preprocessing part, and the retrieval part. The preprocessing part (Figure 1) is constituted of three modules:



**Figure 1. The general structure of Atlas WISE: preprocessing part**

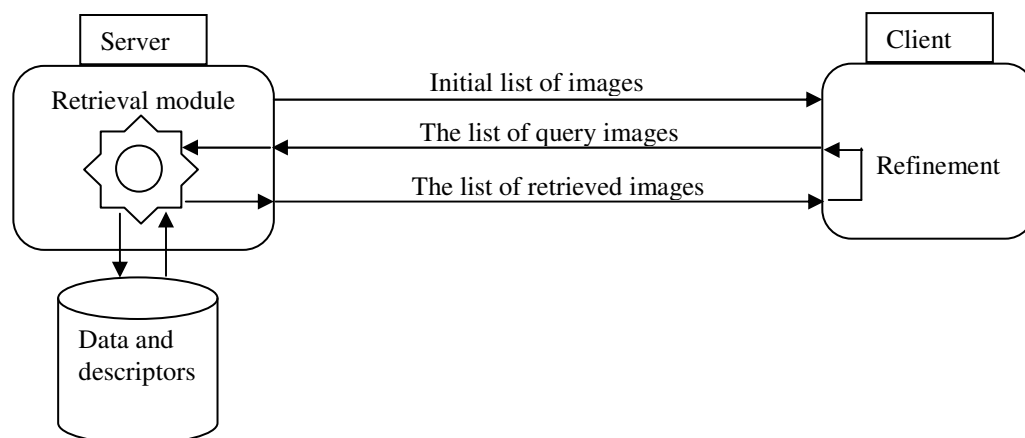
**Image Spider:** a crawler that scours the Web, identifies images and data that can be used for retrieval, and makes a copy of them in the local hard drive.

**Image conversion and thumbnail creation module:** for any collected image, this module convert it into PPM format which is the only format accepted by the digestion module. It also generates a small copy of each image (a thumbnail) to be used for viewing.

**Image digestion module:** for each collected image, this module computes a set of features (or descriptors) to be used in retrieval phase. After the features are computed, the images are destroyed from the local drive. We only keep the thumbnails, the feature vectors, and of course the URL of each image to be able to locate it in the future. We notice that URL means Unified Resource Locator, i.e., a Web address.

The retrieval part (Figure 2) is constituted of two main modules:

- The server: it receives queries from the clients, launches the retrieval process; then, when the resulting images are obtained, the server sends them to the corresponding client.
- The client allows the user to get connected to the system. It consists in an input/output interface where the user can first select images to participate in the query; then after the query is processed, it allows him to see the resulting images. The client also ensures the link between the user and the server. Many clients can be connected to the server at the same moment. More details on the communication between client and server are given in Section 4.1.



**Figure 2. The general structure of Atlas WISE: retrieval part**

### 3. Preprocessing

Data gathering and preprocessing must be performed offline. This is because the World Wide Web contains many millions of images, collecting all of these images and computing their descriptors can take several days even when the used machines are very powerful. Hence, It is inconceivable that the retrieval engine perform that operations for each query, they rather must be done previously. In this Section, we describe the main modules participating in preprocessing phase.

#### 3.1. Image Spider

The first module of our image retrieval engine Atlas WISE is called Image Spider. It is a crawler that regularly traverses the Web collecting images and data needed for retrieval. Many techniques can be used in Web crawling [1]. One possible technique consists in generating all the possible Web addresses, than visiting each of them. This technique allows a better coverage of the World Wide Web content; however, it is very time consuming. Furthermore, many of the possible URL addresses are not used. Hence we adopted the technique described below. We launch many copies of Image Spider, each copy starts with a seed URL to initiate exploration. We essentially use popular index pages such as *Yahoo!* and *Google* to start. By analyzing the content of every visited Web page, Image Spider:

- identifies all the valid images present in this page, and makes a copy of them into the local hard drive.
- extracts the URL's pointing to other pages, then visits the page pointed by each of these URL's.
- This operation is repeated recursively.

**Data updating:** Everyday, there are pages added to the World Wide Web or deleted from it. There are also data changed in existing Web pages. Hence, we launch regularly Image Spider in order to update the collected images and data. This allows us to keep the computed descriptors of images up to date.

**Image validity:** Not all the images present in a Web page are valid for retrieval. Indeed, some are very small. In general, they represent icons rather than images. Hence, Image Spider excludes every image whose dimensions are less than 64×64 pixels. Other kinds of images may be there for advertising or for decoration such as buttons and balls, for navigation such as arrows, or for information such as warnings [1]. Image Spider has to discard such images in the future to avoid that our system retrieves them.

#### 3.2. Conversion and thumbnail creation module

We use a Python script that performs the two following operations:

**Image conversion:** Atlas WISE is based on our image retrieval engine which accepts images in a unique format which is PPM. On the other hand, images present on the Web take different formats including JPG,

GIF and TIF. Hence, all these formats of images are converted into a single format PPM to be usable by the digestion module.

**Thumbnail creation:** Given the huge size of the World Wide Web, it is inconceivable to keep a permanent copy of each collected image and data. To save storage space, each image is deleted after it has been digested. However, to be able to present the retrieval results to the user, we generate a compact copy of each image which requires less storage space, called thumbnail. We mention that the URL of each image is kept together with its descriptors to be able to reach the original image in the future if needed.

### 3.3. Image digestion module

We use colour histograms and texture histograms as features. Furthermore, due to the modular structure of our system, we can easily add any other feature or combination of features. We begin by briefly explaining how we compute colour histograms in a method very similar to the one we use in [2] We then explain the computation of texture histograms.

**Colour histogram computation:**

For each image, we compute a colour histogram of 27 components. We first map all the pixels of the image into the 3-D HSI colour space. We then subdivide each axis of the HSI space into 3 bins, this gives us a total of  $3^3=27$  bins. Then, for each bin, we compute the number of pixels falling in it and we put the result in the corresponding cell in the colour histogram. See Figure 3.

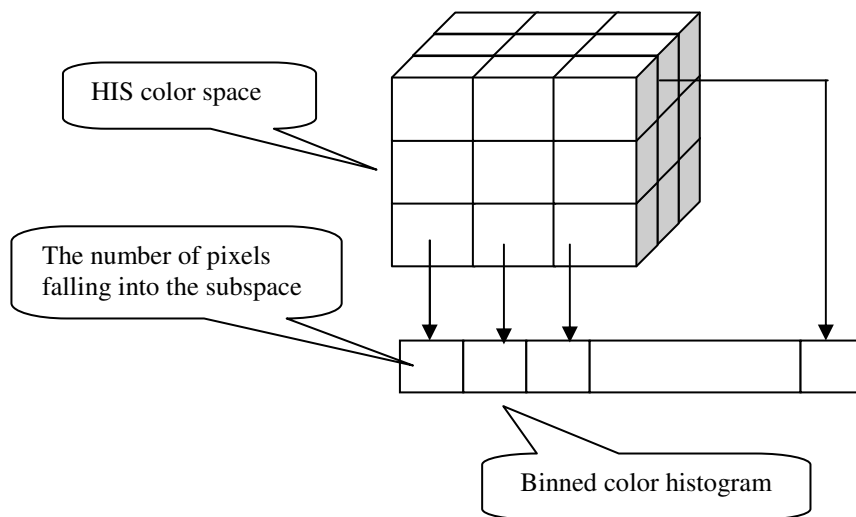


Figure 3. color feature computation

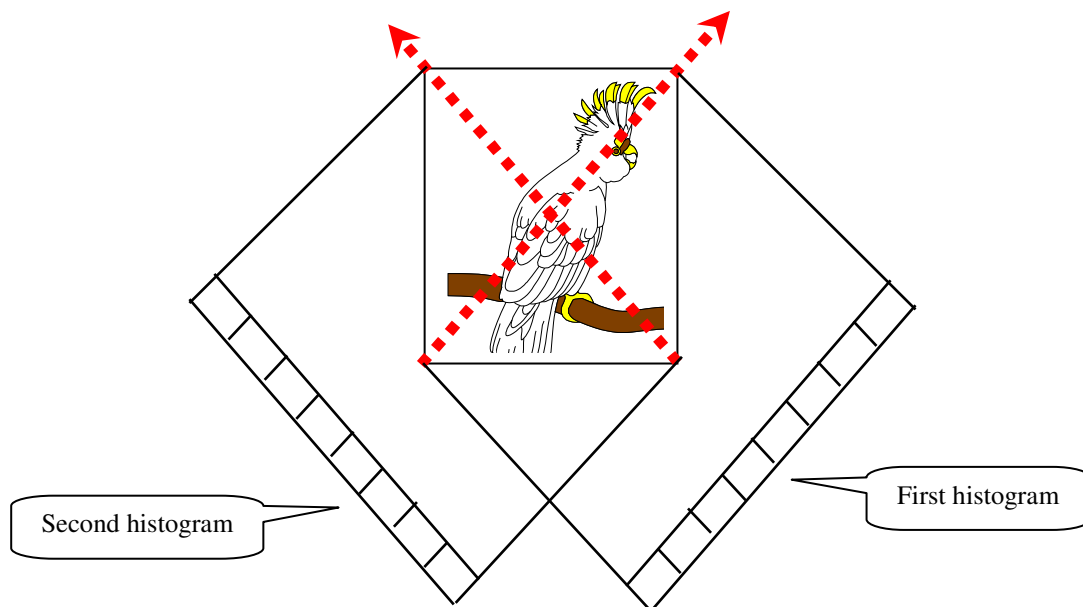
**Texture histogram computation:**

The texture features consist in two histograms computed for each image as follows: (See Figure 4)

- We detect the edges of the image, and we eliminate the noise
- We project the edge points on the two main axes of the image ( $X=Y$  and  $X=-Y$ )
- We construct one histogram corresponding to each axis, each cell of a histogram contains the number of edge points falling in its corresponding position.

## 4. Retrieval

After preprocessing is finished, retrieval process can start. It is ensured by two principal modules: the client and the server.



**Figure 4. Computation of texture histograms**

#### 4.1. Communication between client and server

We can summarize the communication between the client and the server as follows:

- Using an Internet navigator, such as *Internet Explorer*, the user can launch the client by typing its address in the address bar.
- The client sends a connection request to the server
- The server responds by sending to the client a set of sample images allowing the user to specify the initial query.
- The user constitutes his query using one or more among the images viewed by the client. The query can contain positive example images only or both positive and negative examples.
- The client sends the query list to the server.
- The server launches the retrieval module (detailed in Section 4.2) which identifies, using the feature vectors and some similarity measures, the images that answer the current query. This list is sent to the client.
- Having receiving the list of resulting images, the client view them to the user.
- The user can continue refining the retrieval by introducing new positive example/negative example images in the query. He can also visit any viewed image in its original location on the Web.

#### 4.2. Retrieval module

As an input, this module receives a query consisting in a list of images together with their similarity degrees with the sought image(s). The retrieval module performs the following:

- It extracts the descriptors of the query images from the database of descriptors.
- Computes the optimal parameters used for feature selection and retrieval as explained in Section 5.
- Computes the distance of each image in the collection from the query.
- Ranks the images increasingly according to this distance, and
- Returns the top ranked images.

In Section 5 we explain briefly our feature selection algorithm. More details can be found in [2].

### Query processing steps:

If the query contains positive example only, it is processed in one step. We compute the optimal query descriptor which is nothing but the average of the descriptors of all positive example images. We compute the feature selection parameters,  $u_i$  and  $W_i$  for each feature. Then, we compute the distance of each image in the collection from the query, we rank images increasingly according to this distance, then we return to the user the top ranked images.

If the query contains both positive and negative examples, it is processed in two steps. In the first step, only positive example is considered and images are sorted according to their resemblance to positive example. Only the top ranked images of the first step are considered in the second step where images are ranked according to their resemblance to positive example and dis-resemblance to negative example, see Figure 5. Justifications and more details about this retrieval method can be found in [2].

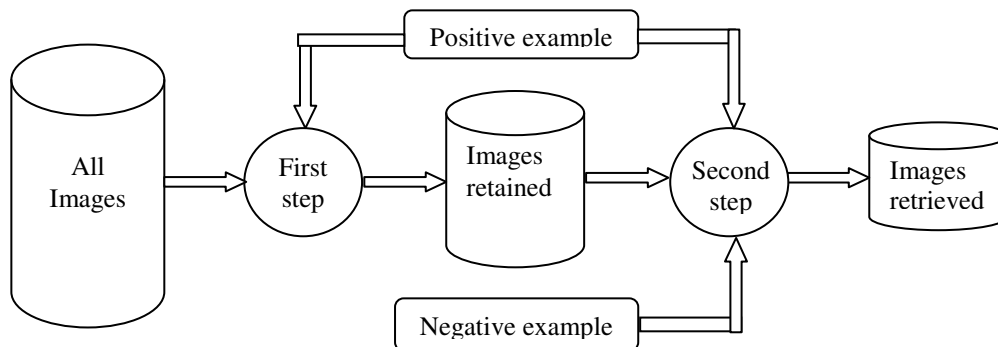


Figure 5. retrieval in two steps when the query contains negative example

## 5. Feature selection algorithm

In [2], we introduced a new approach using positive and negative examples for content-based image retrieval, that we use in our system Atlas WISE. We represent images with a two-level hierarchical model described in [2]. The distance between two images  $x_1$  and  $x_2$  is given by Equation (1)

$$D(x_1, x_2) = \sum_{i=1}^I u_i (\bar{x}_{1i} - \bar{x}_{2i})^T W_i (\bar{x}_{1i} - \bar{x}_{2i}) \quad (1)$$

where  $u_i$  is the global weight assigned to the  $i^{\text{th}}$  feature, and  $W_i$  is a weight matrix assigned to the components of the  $i^{\text{th}}$  feature.

A given query can contain one or many positive example images and one or more negative example images. The objective of our feature selection algorithm is to weight each feature as well as its components according to the following criteria:

- Their concentration: in a positive-example only query, features for which positive example images are close to each other are given more importance than dispersed features. In a positive-and-negative-example query, features for which positive example images are close to each other and negative example images are close to each other, are given more importance than other features.
- Their degree of discrimination: in a positive-and-negative-example query, features that distinguish well between positive and negative examples are given more importance than common features.

Feature selection is performed by computing, for each feature  $i$ , its scalar weight  $u_i$  and a matrix  $W_i$ . The problem is mathematically formulated as follows:

If the query contains positive example only, we use the same method as in [9], where we compute  $u_i$  and  $W_i$  that minimize the global dispersion of positive example given by Equation (2)

$$J_{\text{positive}} = \sum_{i=1}^I u_i \sum_{n=1}^{N_i} \pi_n^i (\bar{x}_{ni}^1 - \bar{x}_i^1)^T W_i (\bar{x}_{ni}^1 - \bar{x}_i^1) \quad (2)$$

where  $T$  denotes matrix transposition,  $\pi_h^1$  is the relevance degree given by the user to the  $n^{\text{th}}$  image,  $\bar{x}_{ni}^1$  is the  $i^{\text{th}}$  feature vector of the  $n^{\text{th}}$  image, and  $\bar{\bar{x}}_i^1$  is the average of the  $i^{\text{th}}$  feature of positive example images. Optimal parameters in such case are the same as in [9].

If the query contains both positive example and negative example images, then we compute  $u_i$  and  $W_i$  that simultaneously minimize the intra dispersion  $A$  (Equation (3)) and maximize the inter dispersion  $R$  (Equation (4))

$$A = \sum_{i=1}^I u_i \sum_{k=1}^2 \sum_{n=1}^{N_k} \pi_h^k (\bar{x}_{ni}^k - \bar{\bar{x}}_i^k)^T W_i (\bar{x}_{ni}^k - \bar{\bar{x}}_i^k) \quad (3)$$

$$R = \sum_{i=1}^I u_i \sum_{k=1}^2 \tilde{\pi}^k (\bar{x}_i^k - \bar{q}_i)^T W_i (\bar{x}_i^k - \bar{q}_i) \quad (4)$$

where  $k=1$  for positive example and  $k=2$  for negative example. Derivation gives us the optimal parameters of the model which are:

$W_i = (\det(C_i))^{-\frac{1}{H_i}} C_i^{-1}$  where  $H_i$  is the dimension of the  $i^{\text{th}}$  feature and  $C_i$  is the matrix  $[C_{irs}]$  such that

$$C_{irs} = R \sum_{k=1}^2 \sum_{n=1}^{N_k} \pi_h^k (x_{nr}^k - \bar{x}_r^k)(x_{ns}^k - \bar{x}_s^k) - A \sum_{k=1}^2 \tilde{\pi}^k (\bar{x}_r^k - q_r)(\bar{x}_s^k - q_s)$$

$u_i = \sum_{j=1}^I \sqrt{\frac{f_j}{f_i}}$  where

$$f_i = R \left[ \sum_{k=1}^2 \sum_{n=1}^{N_k} \pi_h^k (\bar{x}_{ni}^k - \bar{\bar{x}}_i^k)^T W_i (\bar{x}_{ni}^k - \bar{\bar{x}}_i^k) \right] - A \left[ \sum_{k=1}^2 \tilde{\pi}^k (\bar{x}_i^k - \bar{q}_i)^T W_i (\bar{x}_i^k - \bar{q}_i) \right]$$

These weights are then applied to define similarity measures which allow us to compute a kind of distance between each image in the Web and the query.

## 6. Experimental results

We collected more than 10000 images from the World Wide Web, and we used them to perform many experiments of retrieval. We constitute a query of two positive example images and we launch retrieval. In Figure 6, we can see the retrieved images where the query images have been returned in the top positions (1) and (2). We notice that the retrieval results are good in general. A more formal evaluation of our feature selection algorithm has been done in [2].

## 7. Improving retrieval by integrating semantic features

Despite the progress in content-based image retrieval, the current CBIR systems still have a major drawback, that is the semantic gap. Indeed, in many cases, users can be interested in retrieving images that depict a given event or convey certain messages. However, low-level features such as color and texture are still unable to describe such aspects. This can be seen in the example of Figure 6, where image (11) has been retrieved because it is visually similar to the query. However, if we consider semantic concepts, the query contains horses on the green grass; whereas, image (11) contains people in the sea.

One possible way to overcome the problem of retrieval with semantic concepts is to combine text with image content. In Web pages, images are surrounded with text that can be very relevant to them. Keywords that can be exploited come mainly from the image caption, the alternate text (displayed if the image can not be viewed), the image name, the page title, the page URL, and the text near the image in location. In [10], we propose a new method that allows to improve content-based image retrieval systems by integrating semantic features. Our method is somewhere similar to the one used in Information Retrieval approach. As in vector space model, we represent each image with a vector. If we have a total of  $N$  images and  $M$  possible terms, then we use a  $M \times N$  matrix where the  $n^{\text{th}}$  column vector represents the  $n^{\text{th}}$  image. See Figure 7. Each component  $P_{ij}$  of the vectors reflects a particular term (keyword)  $T_i$  associated with a given image  $I_j$ . The value assigned to that component reflects the importance of the keyword in representing the

semantics of the image. The main idea underlying the semantic-based relevance feedback is to modify  $P_{ij}$  from the user's perspective, which can be expressed mathematically by Equation (5).

$$P_{ij} = \text{Prob}(T_i, I_j) = \text{Prob}(I_j/T_i) \text{Prob}(T_i) \quad (5)$$

where  $P(I_j/T_i)$  is the probability to get the image  $I_j$  given the term  $T_i$ , and  $P(T_i)$  is the a priori probability to observe the keyword  $T_i$ .

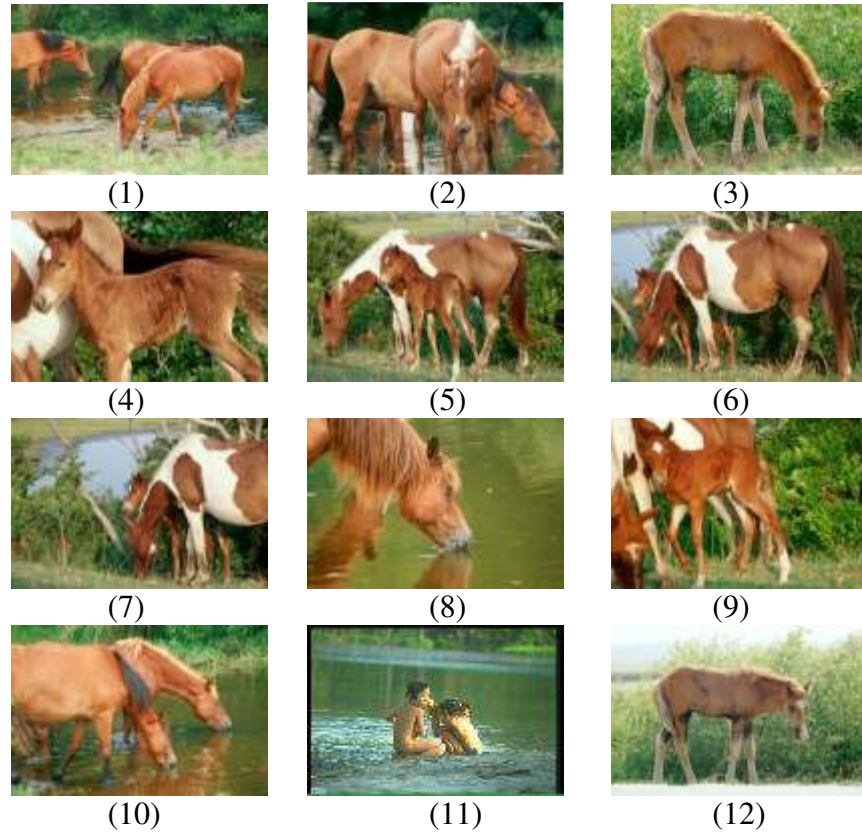


Figure 6. Example of retrieval.

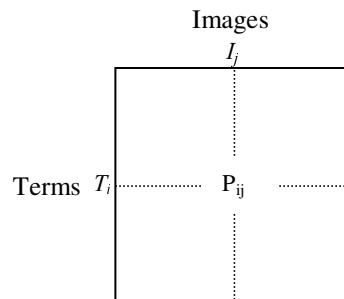


Figure 7. vector space model representation of images.

In the current version of Atlas WISE, keyword collection is ensured by our data gathering module. However, the integration of keywords in the retrieval process according to the method proposed above will be added soon. This method will allow users to perform retrieval using high-level semantics in addition to low-level features.

## 8. Conclusion

In this paper, We presented the main aspects of our Web image retrieval engine Atlas WISE. First, there is the gathering of images from the Web and the computation of their descriptors. Second, there is the client-server architecture of our systems that allows users to get connected to it from everywhere via the Internet. We also presented the feature selection algorithm we use in retrieval stage, and the semantic-based retrieval method that allows us to improve the retrieval results.

## 9. References

- [1] M.L. Kherfi, D. Ziou, and A. Bernardi. Image Retrieval from the World Wide Web: Issues, Strategies, and Systems. To Appear. 2003.
- [2] M.L. Kherfi, D. Ziou, and A. Bernardi. Content-Based Image Retrieval using Positive and Negative Examples. Submitted to Journal of Visual Communication and Image Representation, August 2002.
- [3] S. Sclaroff, L. Taycher, and M. La Cascia. Image Rover: A Content-Based Image Browser for the World Wide Web. In *IEEE Workshop on Content-Based Access of Image and Video Libraries*, pages 2-9. Puerto Rico 1997.
- [4] Y. A. Aslandogan, and Clement T. Yu. Diogenes: A Web Search Agent for Content-Based Indexing of Personal Images. In *ACM SIGIR*, Athenes, Greece, 2000.
- [5] M. J. Swain, C. Frankel, and V. Athitsos. WebSeer: An Image Search Engine for the World Wide Web. In *IEEE Conference on Computer Vision and Pattern Recognition*, Puerto Rico 1997.
- [6] T. Gevers, A. W. M. Smeulders. The PicToSeek WWW Image Search System. In *IEEE International Conference on Multimedia Computing and Systems*, pages 246-269, Florence, Italy, 1999.
- [7] J. R. Smith, and S.-F. Chang. An Image and Video Search Engine for the World Wide Web. In *SPIE Conference on Storage and Retrieval for Image and Video Databases V, IS&T/SPIE*, pages 84-95, San Jose, California, 1997.
- [8] M. S. Lew. Next Generation Web Searches for Visual Content. *IEEE Computer*, 33(1):46-53. 2000.
- [9] Y. Rui, and T. S. Huang, Optimizing Learning in Image Retrieval. In *IEEE International Conference on Computer Vision and Pattern Recognition*, Hilton Head, Sc, USA, 2000.
- [10] D. Brahmi, and D. Ziou. Improving CBIR systems by integrating semantic features. To Appear 2003.