

Automatic classification of images on the web

Alexander Hartmann and Rainer Lienhart

Intel Lab, Intel Corporation, 2200 Mission College Blvd., Santa Clara, CA 95052-8119, USA

Rainer.Lienhart@intel.com

ABSTRACT

Numerous research works about the extraction of low-level features from images and videos have been published. However, only recently the focus has shifted to exploiting low-level features to classify images and videos automatically into semantically meaningful and broad categories. In this paper, novel classification algorithms are presented for three broad and general-purpose categories. In detail, we present algorithms for distinguishing photo-like images from graphical images, true photos from only photo-like, but artificial images and presentation slides from comics. On a large image database, our classification algorithm achieved an accuracy of 97.3% in separating photo-like images from graphical images. In the subset of photo-like images, true photos could be separated from ray-traced/rendered image with an accuracy of 87.3%, while with an accuracy of 93.2% the subset of graphical images was successfully partitioned into presentation slides and comics.

1. INTRODUCTION

Today search engines on the web allow to search for text contained in web pages. However, more and more people are also interested in finding images and videos on the net. Some search engines have already started to offer the possibility to search for images and videos such as Altavista, however, they often only allow to search for textual hints, which are taken from the image's filename, ALT-tag and/or associated web page.

Altavista also offers the possibility to search for images, which look similar to one that the user has already found using textual hints. However, the similarity search is only possible for some images, maybe because either not all images are analyzed yet or there are certain criteria that an image must meet before it can be used for a similarity search. However those criteria are explained nowhere.

The next generation of search engines will be media portals, which allow to search for all kinds of media elements. For instance, [13] indexes web images based on the text, faces and registered trademark logos appearing visually in them. There is a high demand for search engines which can index beyond textual descriptions. Those media portals of tomorrow need to automatically classify their media content without manual classification. Image libraries of tens of millions of images cannot be classified by humans.

Contributions: In this paper we present novel classification algorithms for three broad categories. In detail, we present algorithms for distinguishing

1. photo-like images from graphical images,
2. true photos from only photo-like images such as raytracing images or screen shots from computer games, and
3. presentation slides from comics.

With the exception of photo-like versus graphical images, we are not aware of any directly related work.

Our choice for these four classes is the result of a thorough analysis of the image classes we could find most often in our database of web images. Over a period of four months about 300000 web images, which did not represent buttons or navigational elements, were crawled and downloaded from the web. A large percentage of these images fell into the above four categories.

2. RELATED WORK

Only recently automatic semantic classification of images based on broad and general-purpose classes has been the topic of some research, i.e., automatic classification of images into semantic classes, which are meaningful to normal people. You do not have to be an expert in a specific field in order to do the classification. Examples of broad and general-purpose semantic classes are outdoor scenes versus indoor scenes and city scenes versus landscape scenes

In [5] and [11] Vailaya et al. describe a method to classify vacation images into classes like indoor/outdoor, city/landscape, and sunset/mountain/forest scenes. They use a Bayesian framework for separating the images.

Gorkani et al. propose a method of separating city/suburb versus country/landscape scenes using the most dominant orientation in the image texture [6]. The dominant orientation differs between city and landscape images. The authors state that it takes humans almost no time or ‘brain power’ to distinguish between those image classes, so there should exist an easy and fast to calculate feature.

Yiu et al. classify pictures into indoor/outdoor scenes using color histograms and texture orientation [7]. For the orientation they use the algorithm by Gorkani and Picard [6]. The vertical orientation serves as the discriminant feature, because indoor images tend to have more artifacts, and artifacts tend to have strong vertical lines.

Bradshaw proposes a method for labeling image regions as natural or man-made. For instance, buildings are man-made, while mountains in the background are natural. He also propose how this feature can be used for indoor/outdoor classification [8].

Swain et al. describe how to separate photographs and graphics on web pages [9,10]. They only search for ‘simple’ graphics such as navigation buttons or drawings, while our work deals with artificial but realistic-looking images, which would be classified as being natural by their algorithm. The features that Swain et al. use are: number of colors, most frequent color, farthest neighbor metric, saturation metric, color histogram metric, and a few more [9,10].

3. GRAPHICAL VERSUS REALISTIC-LOOKING IMAGES

One of the first decisions a user has to make when searching for a particular image is whether the image in search should be graphical or realistic-looking. Examples of graphical images are buttons and navigation elements, scientific presentations, slides, and comics; examples of realistic-looking images are photos, raytracing images, and many images of modern computer games.

3.1 Features

Image features, which could be distinctive for this separation and of which some have been proposed by Swain et al in [9] and [10], are:

- the total number of different colors, because graphics tend to have a lower number of colors.
- the relative size of the largest region and/or the number of regions with a relative size bigger than a certain threshold, because graphics tend to have larger uniformly colored regions.
- the sharpness of the edges, because edges in graphics are usually sharper than edges in photos.
- the fraction of pixels with a saturation greater than a certain threshold because colors in graphics are usually more saturated than those in realistic-looking images.
- the fraction of pixels having the prevalent color because graphics tend to have less colors than photos, and thus the fraction of pixels of the prevalent color is higher in graphics.
- the farthest neighbor metric, which measures the color distance between two neighbor pixels. The distance is defined as $d = |r1-r2| + |g1-g2| + |b1-b2|$, the absolute difference of both pixels’ RGB values. Three sub-feature can be derived:
 - the fraction f_1 of pixels with a distance greater than zero. Graphics usually have larger single-colored regions. So this metric should be lower for graphics.
 - the fraction f_2 of pixels with a distance greater than a high threshold. This value should be high for graphics.

- the ratio between f_2 and f_1 . As f_1 tends to be larger for photographs a low value of f_2/f_1 indicates a photo-like image.

3.2 Training

All these features were implemented and tested on a large number of training- and test images. In the end, only four of them proved to be useful:

- the total number of colors c_n after truncating each color channel to only its 3 most significant,
- the prevalent color c_p ,
- the fraction f_1 of pixels with a distance greater than zero and
- the ratio between f_2 and f_1 .

All other features were not distinctive enough partly due to the fact that all our images were JPEG-compressed. Some of the characteristic features of graphics are destroyed by JPEG's lossy compression. Note that in [9,10] most graphical images were GIF-compressed.

The following very simple, but sufficient hierarchical classifier was derived from a training set of about 1500 images:

```

if ( $c_n \leq thres_{cn}$ )
  -> graphical image
else if ( $c_p \geq thres_{cp}$ )
  -> graphical image
else if ( $f_1 \leq thres_{f1}$ )
  -> graphical image
else if ( $f_2/f_1 \geq thres_{f1f2}$ )
  -> graphical image
else
  -> photo-like

```

The optimal threshold values were derived from the training set.

3.3 Experimental Results

On a test set of 1568 graphical images (comics and slides, the same as in Section 5) and 6327 photographic images (raytracing images and photographs, the same as in Section 4) 93.8% of the graphical images and 98.1% of the photo-like images were classified correctly, resulting in an overall accuracy of 97.26%. The four-tuple of thresholds used were (50, 0.15, 0.8, 0.15).

The misclassified photo-like images were mostly photos made up of only a few colors (such as the right image in Figure 1), or raytracing images, which did not look realistic at all, but were put into this class because they were part of an image archive of raytracing images (see the two left most images in Figure 1).

Misclassified graphical images were either slides containing large photographs (left image in Figure 2) or very colorful comics (right image in Figure 2).

4. COMPUTER GENERATED, REALISTIC LOOKING IMAGES VERSUS REAL PHOTOS

The algorithm proposed in this section for distinguishing between real photos and computer-generated, realistic-looking images can be applied to the set of images which has been classified as being realistic-looking by the algorithm described in Section 3.

The class of real photos encompasses all kinds of images taken from nature. Typical examples are digital photos and video frames. In contrast, the class of computer-generated images encompasses raytracing images as well as images from graphic



Fig. 1. Examples of realistic looking images, which were misclassified as being graphics. The misclassified images were either photos made up of only a few colors (right image) or raytracing images, which did not look realistic at all, but were put into this class because they were part of an image archive of raytracing images (the two left most images).



Fig. 2. Examples of graphical images misclassified as realistic-looking images

tools such as Photoshop and computer games. Figure 3 shows three examples for each class.

4.1 Features

Every real photo contains noise due to the process of converting an analog image into digital form. For computer-generated, realistic-looking images this conversion/scanning process, however, is not needed. Thus, it can be expected that computer-generated images are far less noisy than digitized images. By designing a feature that measures noise it should be possible to distinguish between scanned images and images, which were digital right from the beginning.

A second suitable feature that can be used is the sharpness of the edges, because computer-generated images are supposed to show sharper edges than photographs. However, due to lossy JPEG-compression this feature gets less reliable since sharp edges get blurred, and blockiness may be added, i.e., sharp edges might be added, which were not there before.

In practice, we measure the amount of noise by means of the histogram of the absolute difference image between original and its de-noised version. The difference values can vary between 0 and 255. Two simple and fast filters for de-noising are the median filter and the Gaussian filter [3,14]. The core difference between both filters are that they assume different noise sources. The Median filter is more suitable if individual pixels are outliers, while the Gaussian filter is better for Gaussian additive noise [3].

Both de-noising filter were applied at with a radius of 1, 2, 3, and 4. Thus, the resulting feature vector consisted of 2048 values: $4 \cdot 256$ from the median filter and $4 \cdot 256$ from the Gaussian filter.

4.2 Training

We used the Linear Vector Quantization (LVQ) package from the Helsinki University of Technology [1] to train our classifier. In LVQ, a number of codebook vectors m_i are calculated, each of which describes a larger number of training vectors x_i and belongs to one class, for example, the class of computer generated images. The codebook vectors are then used to classify a number of test vectors y_j . A vector y_j is classified as belonging to the same class to which the nearest code vector m_i belongs. Thus, the class c to which the test vector y_j belongs is determined by:

$$c = \operatorname{argmin}_i \{ \|y_j - m_i\| \}$$

The optimal set of code vectors is the one which classifies the most training vectors correctly.

The learning process for getting the code vectors is an iterative process. Let $m_i(t)$ be the value of code vector i after t iterations and x a training vector. The next iteration of m_i is then:

- $m_c(t+1) = m_c(t) + \alpha(t)[x(t) - m_c(t)]$ if x and m_c belong to the same class,
- $m_c(t+1) = m_c(t) - \alpha(t)[x(t) - m_c(t)]$ if x and m_c belong to different classes, and
- $m_i(t+1) = m_i(t)$ for $i \neq c$.

$\alpha(t)$ determines the weight that the new sample has on the value of the code vector. A good value to start with is 0.1. This process is repeated for a certain number of iterations. suggest to makes the following recommendation. For nc different classes and training vectors of dim dimensions [1] suggest to calculate the number of codebook vectors (NOC) and iterations needed as follows:

$$NOC \equiv 0.4 \times nc \times \left(nc - 1 + \frac{dim}{2} \right)$$

$$iterations \equiv 40 \times NOC$$

Raytracing



Scanned



Fig. 3. Examples of raytracing and scanned images

4.3 Experimental Results

The overall image set consisted of 3681 real images (3114 scenic photographs from the 'Master Clips 500.000'-collection and 567 family photos) and 3807 raytracing images from <http://www.irtc.com>, the Internet Ray Tracing Competition. The overall image set was randomly partitioned into 4481 training images and 3007 test images.

LVQ training was performed with the following combinations of feature values

- All 2048 feature elements
- only the median values (1024 values)
- only the Gaussian values (1024 values)

and the following aggregations:

- the first 12 values (0-11) of each 'block' of 256 values were left untouched,
- from the next 116 values (12-127) the sum of four successive values were calculated, and
- the last 128 values (128-255) were added into one value.

This reduced the dimensionality of the input by a factor of 256/42. Also, the following alternative aggregation method was tested, too:

- from the first 128 values (0-127) the sum of each four successive values was taken, and
- the last 128 values were added into one value.

This reduced the dimensionality of the input vectors by a factor of 256/33. These combinations and aggregations resulted in the following training vectors:

- 336 values: median and Gaussian, aggregated with method 1
- 168 values: median only, aggregated with method 1
- 168 values Gaussian only, aggregated with method 1
- 264 values median and Gaussian, aggregated with method 2
- 132 values: median only, aggregated with method 2
- 132 values: Gaussian only, aggregated with method 2

The best classification results were achieved using all 2048 values along with 800 codebook vectors. The recognition accuracy for the test set was 87.33%. When using only one set of features (1024 values, 400 codebook vectors) the accuracy for the training set was 84.93% for median values and 82.30% for Gaussian values. Using the aggregated values the accuracy further decreases to 81.97% for 386 median and Gaussian values and 130 codebook vectors, and fell below 80% for just one feature.

A closer inspection of the misclassified raytracing images revealed that the presence of textures and artifacts due to image compression were misinterpreted for noise. Figure 4 shows an enlarged part of a raytracing image with a texture (diagonal lines) at the bottom and another almost random texture at the top. Both regions were misinterpreted as noise regions and, therefore, responsible for the misclassification.

We also tried to use only some of the 'packets' of 256 values, for example only the median-values with a radius of 1, 2 and 3, or with a radius of 2, 3 and 4, but all the results were worse than those using all 2048 values.

5. PRESENTATION SLIDES VERSUS COMICS

The algorithm we are proposing in this section for distinguishing between presentation slides and comics can be applied to the set of images, which have been classified as being graphical by the algorithm in Section 3.



Fig. 4. Example of an artificial image misclassified as being natural

The class of presentation slides includes all images showing slides independently of whether they were created digitally by presentation programs such as MS® PowerPoint® or by hand. Many scientific posters are designed like a slide, and, therefore, fall into this class, too. However, as we will see later scientific posters containing multiple slides syntactically look like a comic and will be (mis-) classified as such.

The class of comics includes cartoons from newspapers, most of which are available on the web, and books as well as other kinds of comics. Note that occasionally a comic may syntactically look like a slide and will be (mis-) classified as such. In this case, the classification is ambiguous and ill-defined from a purely visual point of view. Figure 7 shows two such ambiguous images.

Member images of both classes can be colored or black and white. Three examples for slides are shown in Figure 5, while the comics can not be shown for copyright reasons.

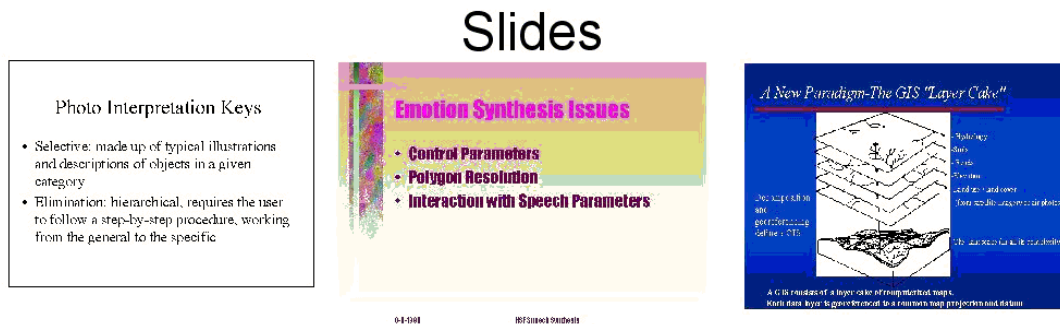


Fig. 5. Examples for slides

5.1 Features

We observed the following three main differences between presentation slides and comics:

1. In general, the size and/or alignment of text line occurrences differ for comics and slides. Thus, images of both classes can be distinguished by means of
 - the relative width of the topmost text line, i.e., the ratio between the width of the topmost text line and the width of the entire image,
 - the average width and height of all text lines and their respective standard deviation, and
 - the average position and the standard deviation of the center of mass over all text lines.

These features are motivated by the following observations:

Slides usually have a heading, which almost fills the entire width of the image. The subsequent text lines are wider than they are in comics. Moreover, the text lines in slides either have only one center in about the middle of the image and thus lead to a small standard deviation over the locations of their center of mass, or they all start in the same col-

umn and therefore have different centers of mass, but all those centers of mass are still near each other and thus resulting in a small standard deviation over the average center location, too.

The relative width of the topmost text line in comics is usually smaller than in slides, as are all other text lines. Slides in general use larger fonts than comics do. Therefore, the larger the average relative height of the text lines, the more probable it is that the image represents a slide.

Further, text in two or more columns are uncommon for slides. Comics on the other hand usually consist of more than one image resulting in more than just one visual center of text blocks. Thus, the standard deviation over the average text line center location will be large.

2. Images containing multiple smaller images aligned on a virtual grid and framed by rectangles are very likely to be comics. These borders can easily be detected by edge detection algorithms.

In comics, the length of those border lines is usually an integral fraction of the image's width or height. For instance, they are either about a fourth of the image's width or a third of the image's height in the left image in Figure 5. The more lines of such length can be found, the higher is the probability for the image to be a comic instead of a presentation slide.

This criterion can be made more precise by checking for the presence of the other $n-1$ lines in the same row/column if a line with a length of one n -th of the image's width/height was found. By means of this procedure lines, which are just by chance of the correct length but have nothing to do with the typical borders in comics, are eliminated.

3. Slides very often have a width-to-height ratio of 4:3. If the aspect ratio differs from 4:3, it is very unlikely that the image is a slide.

5.2 Feature Calculation

We used the algorithm developed by Wernicke et al. to find all text lines in the image under analysis [12]. For improved text line detection performance the text detector was re-trained with text samples from slides and comics.

Edges were extracted by means of the Canny edges detection algorithm and then vectorized [2]. All non-horizontal or non-vertical edges were discarded. Two vertical or horizontal lines were merged if and only if they had the same orientation and the end-point of one line was near the start-point of the other one. This procedure helped to overcome accidental break-ups in the borderlines. Next the lengths of all remaining edges were determined and checked whether each of them was about one, one half, one third or one fourth of the width/height of the image. If not, the respective edge was discarded.

Edges in the same row/column of roughly the same length and with a length close to the n -th fraction of the image's width/height ($n \in \{1, 2, 3, 4\}$) were merged. If the correct number of lines could be combined to form an entire row or column, it was counted as a strong evidence of being a comic by means of increasing the 'strong edge' counter by one. Otherwise, it was counted as a only a weak evidence of being a comic, and the 'weak edge' counter was increased by one (see Figure 6).

5.3 Training

All features in Figure 6 were used to determine whether an image represented a slide or a comic. Qualitatively it can be said that the lower the number of strong or weak border lines the higher the probability that the image represents a slide. If the aspect ratio is near 4:3, the probability for being a slide is high. A large width of the topmost text line as well as a large average text line width also indicate a slide, as does a low standard deviation of the center of mass over all text lines.

Since individual features we use are very diverse, we decided not to use a standard pattern classifier on the combined feature vector. Standard pattern classifiers such as Bayesian, LVQ or support vector machine classifiers usually work best if the individual vector elements are homogeneous, because scaling diverse vector elements appropriately is always a daunting task. Instead, a fuzzy member function was derived from a training set of 2500 images for each feature independently. The overall probability of an input image for being a comic/presentation slide was defined as the sum over the respective member values of all features, and the image was classified according to the most likely one.

We also implemented a simple reject option. If the total probability score for both classes differed less than a small amount, the image was rejected for classification.

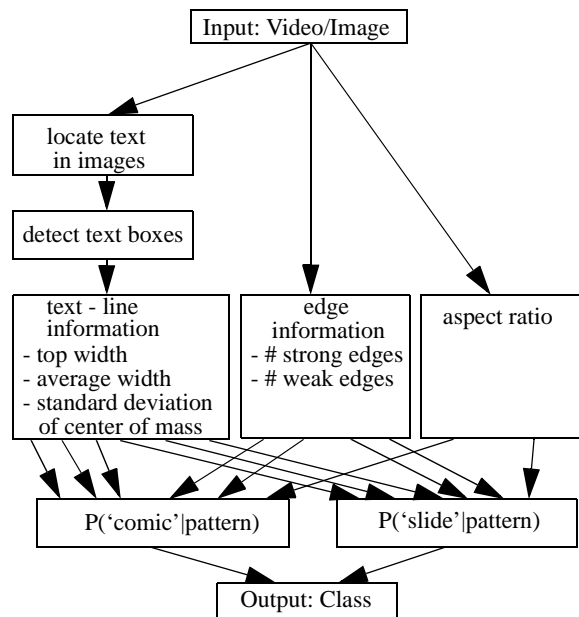


Fig. 6. Feature extraction for slides versus comics

5.4 Experimental Results

During our experiments we observed that in comics the neural network-based text localizer detected a significant number of false text blocks of small width, but large height. However, the text blocks in slides were recognized very well. This stark contrast in the false alarm rate of our text localizer between comics and slides can partly be explained by the fact that the usage of large fonts are prevalent in slides, but not in comics and that our detector worked very well on large text lines. In addition, the kinds of strokes used in comics to draw people and objects in the scene sometimes have similar properties as the strokes used for text, and thus result in false alarms. Despite these imperfections of our text detector [12], all our features except the average height of the text lines could still be used.

Our novel classification algorithm was tested on a large test set containing 1954 comic images and 637 presentation images with the following results:

	Comics	Slides
classified correctly	94.83%	88.08%
classified correctly with reject option	96.81%	92.27%

Table 1: Classification performance

Almost all of the misclassified comic images consisted of one big drawing without a surrounding borderline and had an aspect ratio of about 4:3.

Misclassified slides either consisted of tables, which led to a large standard deviation of the center of mass of the text lines, or included several images, where the borders were taken as a strong evidence for being a comic. One slide also had an unusual aspect ratio of 5:4. Another source of errors were slides, which were rotated by 90 degrees. Since we had no algorithm at hand that could detect the orientation of images, the rotation could not be compensated and, thus, the text lines could not be

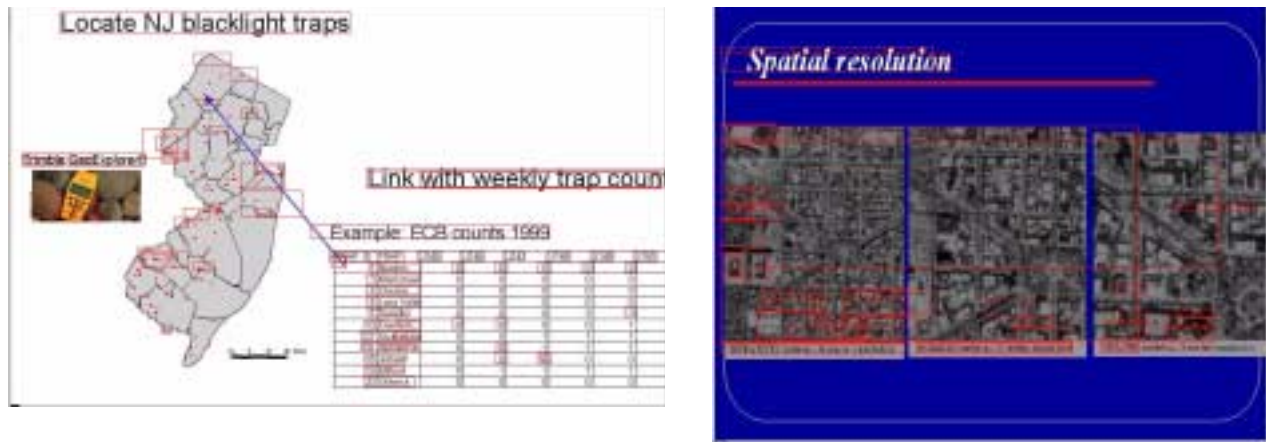


Fig. 7. Example of slides which were misclassified as being comics (including detected text blocks) detected correctly. Consequently the classification failed. Examples of misclassifications are given in Figure 7.

In general, our results show that the positions of the text lines are a strong feature for distinguishing between slides and comics. If the text location algorithm is improved, either by replacing with a new algorithm or by adding an OCR-module, the classification accuracy will further increase.

6. CONCLUSION

Automatic semantic classification of images is a very interesting research field. In this paper, we presented novel and effect algorithms for two classification problems, which have not been addressed before: comics versus slides and real photos versus realistic-looking but computer generated images. On a large image database, true photos could be separated from ray-traced/rendered image with an accuracy of 87.3%, while with an accuracy of 93.2% presentation slides were successfully distinguished from comics. We also enhanced and adjusted the algorithms proposed in [9,10] for the separation of graphical images from photo-like images. On a large image database, our classification algorithm achieved an accuracy of 97.3%.

References

- [1] The Learning Vector Quantization Program Package, <ftp://cochlea.hut.fi>
- [2] J. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 6, pp. 34-43, Nov. 1986.
- [3] Bernd Jaehne. *Digital Image Processing*. Springer-Verlag, 1997.
- [4] Tom M. Mitchell. *Machine Learning*. WCB/McGraw-Hill, 1997
- [5] Aditya Vailaya, Mario Figueiredo, Anil Jain, HongJiang Zhang. Bayesian Framework for Hierarchical Semantic Classification of Vacation Images. *Proceedings of the IEEE International Conference on Multimedia Computing and Systems (ICMSC)*, pp. 518 - 523, Florence (Italy), 1999.
- [6] M. M. Gorkani and R. W. Picard. Texture Orientation for Sorting Photos 'at a Glance'. *Proc. ICPR*, Oct. 1994.
- [7] Elaine Yiu and Federico Girosi. Image Classification Using Color Cues and Texture Orientation, <http://www.ai.mit.edu/projects/cbcl/res-area/current-html/ecyiu/project.html>.
- [8] Ben Bradshaw. Semantic Based Image Retrieval: A Probabilistic Approach. *ACM Multimedia 2000*, Oct. 2000.
- [9] Vassilis Athitsos, Michael J. Swain, Charles Frankel. Distinguishing Photographs and Graphics on the World Wide Web. *IEEE Workshop on Content-Based Access of Image and Video Libraries*, June 1997
- [10] Charles Frankel, Michael J. Swain, and Vassilis Athistos. WebSeer: An Image Search Engine for the World Wide Web, University of Chicago Department of Computer Science Technical Report TR-96-14, August 1996, <http://www.infolab.nwu.edu/webseer/>.
- [11] Aditya Vailaya. Semantic Classification in Image Databases, PhD thesis, Department of Computer Science, Michigan State University, 2000, <http://www.cse.msu.edu/~vailayaa/publications.html>.
- [12] Axel Wernicke and Rainer Lienhart. On the Segmentation of Text in Videos. *IEEE International Conference on Multimedia and Expo (ICME2000)*, New York, July 2000.
- [13] www.visoo.com
- [14] Mark R. Banham and Aggelos K. Katsaggelos. Digital Image Restoration. *IEEE Signal Processing Magazine*, Vol. 14, Issue 2, pp. 24-41, March 1997.