ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Image Tag Propagation

MSc Semester Project

Leila Mirmohamadsadeghi

Under supervision of :

Prof. Touradj Ebrahimi, Ivan Ivanov

Multimedia Signal Processing Group

10/06/2010

With the growth of the interest of internet users to upload and annotate images, this semester project aims at developing a system to enrich images with social network tagging. Different visual features are exploited in order to extract the content from an image and use it for efficient image annotation.

Table of contents

Table of abbreviations

CIE: Commission Internationale de l'Eclairage
DOG: Difference Of Gaussian
EOH: Edge Orientation Histogram
GLDM: Gray Level Difference Method
GUI: Graphical User Interface
HOG: Histogram of Oriented Gradients
HSV: Hue-Saturation-Value
PDF: Probability Distribution Function
RGB: Red, Green, Blue
SIFT: Scale Invariant Feature Transform
SURF: Speeded Up Robust Features

1. Introduction

Over the last few years, social network systems have greatly increased internet users' involvement in content creation and annotation. These systems are characterized as easy-to-use and interactive. Users contribute with their opinion by annotating content (the so-called *tagging*), they add tags, comments and recommendations or rate the content.

This semester project aims at developing a system to enrich images with social network tagging. Different visual features are exploited in order to extract the content from an image and use them for efficient image annotation. More specifically, the user provides an image which he/she tags. The system performs image similarity search in order to find images having the same content in a large collection of images. Initial tags of the query image can then be propagated to new images. For instance, a tourist takes a picture of the Eiffel tower with his mobile phone. By querying a given dataset, the system can identify a number of images also depicting the Eiffel tower. The provided tags, e.g. "Eiffel tower", "Paris", are merged and finally assigned to other images in the dataset. This also allows determining the context of the picture. This approach offers a compelling new look at how metadata can be easily generated in order to provide efficient content-aware management and organization of image collections.

More specifically, the goal of this project is to study different approaches for content-based image retrieval and compare their performances for tag propagation in still images.

This report is structured in three major sections:

- Content-based Image retrieval: different techniques for retrieving similar images to a query from a data set are explained and explored. These content-based extraction methods are: local Scale invariant feature transform (SIFT) [1], Speeded up robust features (SURF) [2], dominant color histogram (in HSV and CIE Lab color spaces), Edge orientation histogram (EOH) [3], Histogram of oriented gradients (HOG) [4], Texture-analysis method Gray level difference method (GLDM) [5] and texture analysis based on Gabor filtering [6].
- Tag propagation: the concept of creating and propagating textual information about an image.
- User interface: A graphical user interface is developed to demonstrate the operation of image retrieval system along with the tag propagation system.
- Experiments: Image retrieval using different features are tested over different classes of images to assess performance of different methods for different data.

2. Content-based image retrieval

With the recent exponential growth of images on the internet and in social networks, the need to search images based on their content is growing in importance as well. Search engines such as Google images [7] search images based on their textual annotations. The drawback to this method is that images need to be annotated manually, and moreover the annotation could describe a given image in a way that doesn't describe its visual content. In contrast, content-based image retrieval searches images based on their visual content. This method of searching images is currently not commonly used by internet users; however there are engines available for specialized customers, such as Piximilar [8], a content-based image search engine on a given database based on visual color similarity.

For image retrieval using visual content, given a query image, similar images can be retrieved from a given dataset by using features to describe the contents of the image [9]. A feature histogram or vector is extracted from the query image and matched against previously extracted features of a dataset. Given a similarity measure (often a distance measure), most similar images can be retrieved from the dataset. In other words, the user interface created here will operate as a search engine based on the local or global content of an image.

2.1 SIFT

The scale invariant features (SIFT) [1] extracted in this method, as their name indicates, are invariant to scale changes. These features are particularly interesting in matching images of a same object in which target object appears with a different size or angle. In addition, these features are distinctive, which makes them suitable for matching images with a high probability.
The computation of the features is achieved in the following stages:

2.1.1 Detection of extrema in the scale-space

The first step in the SIFT algorithm is to find points which are stable for different views of an object. These points are found by searching for stable features across all possible scales using a continuous function of scale known as the scale-space. The scale-space kernel used here is the Gaussian kernel. The scale-space of a given image *I(x,y)* for scales σ is:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

Where

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

Keypoints are found by detecting extrema in the difference of Gaussian function:

$$L(x, y, \sigma) = \big(G(x, y, k\sigma) - G(x, y, \sigma)\big) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma)$$

The constant *k* is the separation between two scales.

The scale-space is created by applying the above operation at different scales for an octave, an octave corresponds to doubling the value of the scale. Each octave is chosen to be divided into an integer number of intervals. Images at adjacent scales are subtracted to create a difference of Gaussian pyramid (DOG) (see Figure 1). In order to detect the local extrema, each sample point is compared to its eight neighbors in the current image and nine neighbors in the scale above and below. It is selected only if it is larger than all of these neighbors or smaller than all of them.

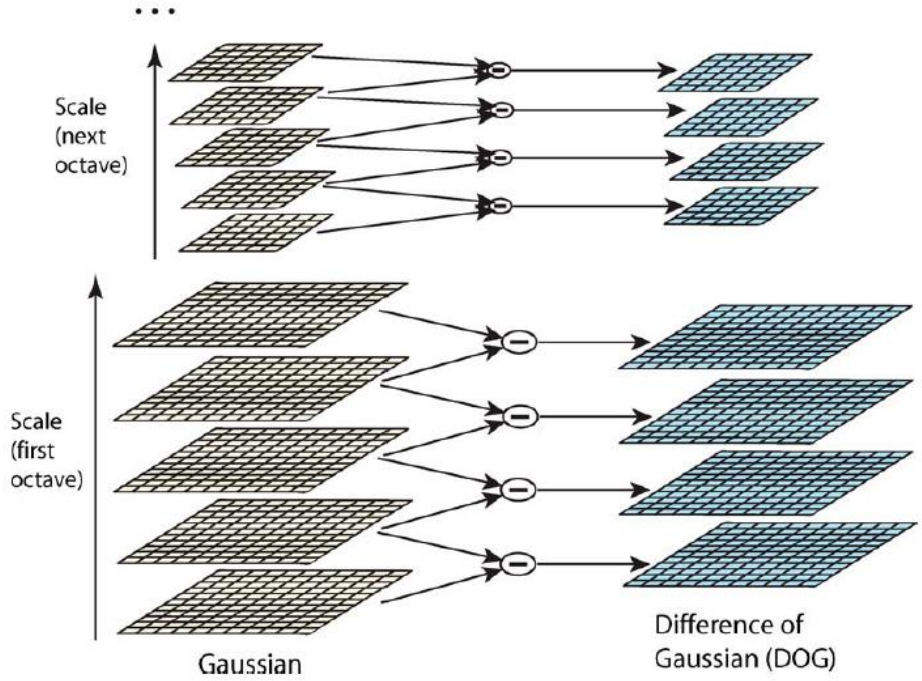


Figure 1: Adjacent scale-space images on the left are subtracted to create difference of Gaussian images on the right [1]

2.1.2 Keypoint localization

After detection of extrema of the scale-space, the points with low contrast or those located on edges need to be rejected. To do so, a 3D quadratic function is fitted to the local sample points to determine the interpolated location of the extremum. This approach uses the Taylor expansion up to the quadratic term of the scale-space function $D(x, y, \sigma)$, shifted so that the origin is at the sample point:

$$D(\boldsymbol{x}) = D + \frac{\partial D^T}{\partial \boldsymbol{x}}\,\boldsymbol{x} + \frac{1}{2}\,\boldsymbol{x}^T\,\frac{\partial^2 D^T}{\partial \boldsymbol{x}^2}\,\boldsymbol{x}$$

Where *D* and its derivatives are evaluated at the sample point, and $\boldsymbol{x} = (x, y, \sigma)^{\mathrm{T}}$ is the offset from this point. The location of the extremum $\hat{\boldsymbol{x}}$, is determined by taking the derivative of this function with respect to **x** and setting it to zero:

$$\hat{\boldsymbol{x}} = -\frac{\partial^2 D^{-1}}{\partial \boldsymbol{x}^2}\,\frac{\partial D}{\partial \boldsymbol{x}}$$

2.1.3 Orientation assignment

Each keypoint is assigned an orientation based on local image properties. The keypoint descriptor can be represented relative to this orientation and therefore achieve invariance to image rotation.
To do so, the Gaussian smoothed image at the closest scale to the keypoint scale is considered to allow computations in a scale invariant manner. For this image $L(x, y)$, the gradient magnitudes and orientations are computed for the neighboring pixels of the keypoint:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

An orientation histogram is formed from the gradient orientations of sample points within a region around the keypoint. Each sample added to the histogram is weighted by its gradient magnitude and by a Gaussian-weighted circular window with a σ that is 1.5 times that of the scale of the keypoint. This additional weight avoids small sudden changes to affect the descriptor. If there are local peaks within 80% of the highest peak in the neighborhood of a keypoint, additional keypoints are assigned with that orientation. This multiplicity of keypoints at one location improves stability of matching.

2.1.4 Creation of descriptor

The descriptor is formed from a vector containing the values of all the orientation histogram entries (see Figure 2). The figure shows a 2 × 2 array of orientation histograms, whereas in this project a 4 × 4 array of histograms with 8 orientation bins in each is used. Therefore, there are 4 × 4 × 8 = 128 elements in the feature vector for each keypoint.

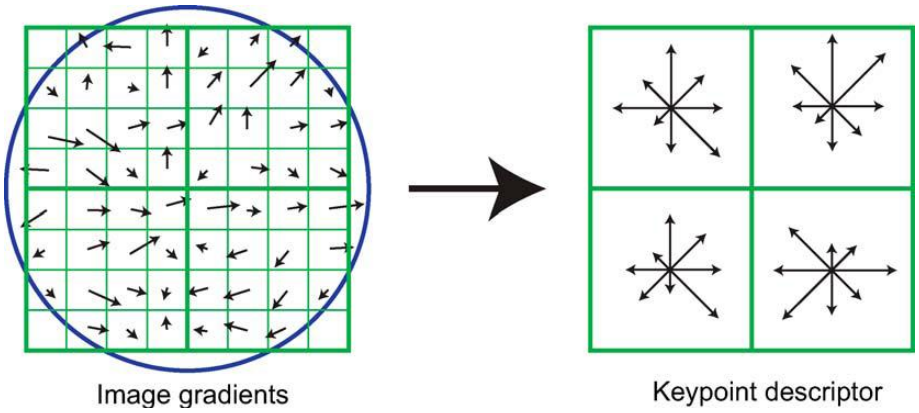


Figure 2: Creation of a keypoint descriptor from gradient magnitudes and orientations [1]

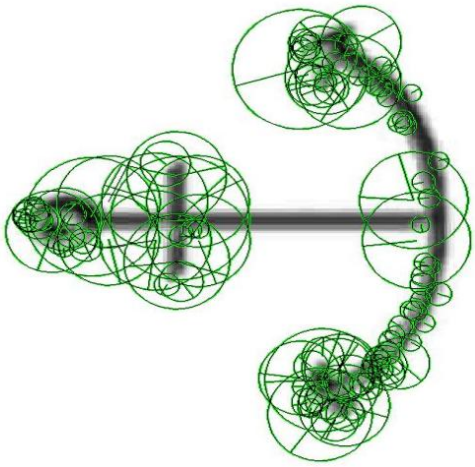


Figure 3: SIFT Keypoints in an image

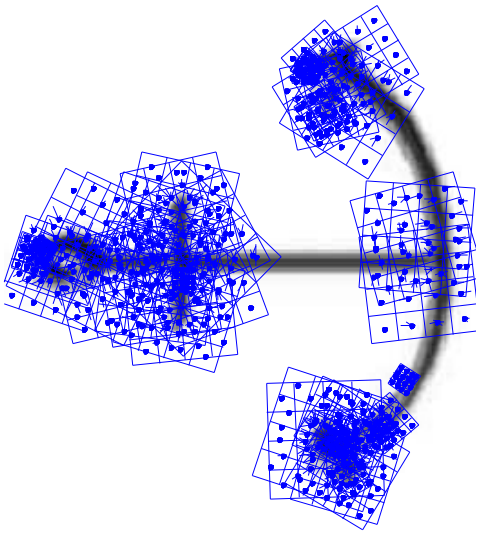


Figure 4: SIFT descriptors

2.1.5 Matching

For the purposes of this project, the SIFT feature vectors for all the images in the dataset are computed offline and stored as a feature space. This space is used to retrieve images similar to the query image by means of a similarity measure. Two descriptors (D1 for a descriptor from the query image and D2 for a descriptor from an image in the dataset) are matched to each other if their Euclidian distance multiplied by a certain threshold is not greater than the distance of D1 to all other descriptors [10]. The similarity measure computed here takes into consideration the mean distance between matched keypoints, weighted by the number of matches. The images with the most number of similar vectors are retrieved from the dataset.

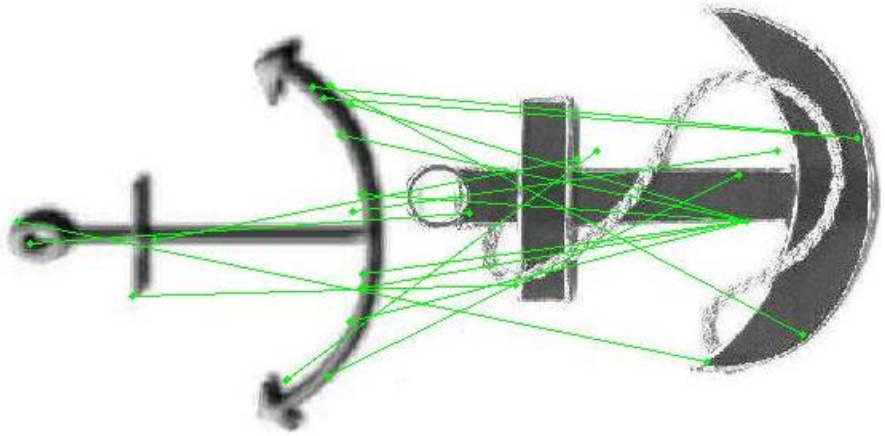


Figure 5: SIFT Keypoint matching between two images

2.2 SURF

The Speeded up robust features algorithm (SURF) [2] is a scale and rotation-invariant interest point detector/descriptor which is computationally fast. The detector is based on the Hessian matrix, but uses a very basic approximation, just as Difference of Gaussian is a very basic Laplacian-based detector. The descriptor, on the other hand, describes a distribution of Haar-wavelet responses within the interest point neighborhood. Integral images are exploited to improve speed.

2.2.1 Fast-Hessian Detector

The detector is based on the Hessian matrix because of its performance in terms of computation time and accuracy. The determinant of the Hessian matrix is used to determine the location and scale of the descriptor.
The Hessian matrix is written as $H(\boldsymbol{x}, \sigma)$ for a given point $\boldsymbol{x} = (x, y)$ in an image:

$$H(\boldsymbol{x}, \sigma) = \begin{bmatrix} L_{xx}(\boldsymbol{x}, \sigma) & L_{xy}(\boldsymbol{x}, \sigma) \\ L_{xy}(\boldsymbol{x}, \sigma) & L_{yy}(\boldsymbol{x}, \sigma) \end{bmatrix}$$

where $L_{xx}(\boldsymbol{x}, \sigma)$ denotes the convolution of the second derivative of the Gaussian $\partial^2/\partial x^2 g(\sigma)$ with the image at point $\boldsymbol{x}$. Since Gaussian filters are non-ideal in any case, for computational simplicity, the second order derivative of the Gaussian is approximated by a box filter. The determinant of the Hessian matrix is written as:

$$\det(H_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2$$

With the use of integral images and box filters, the same filter doesn't need to be applied iteratively to the output of the previous filtered layer, but rather all filters of same size can be directly applied on the original image, meaning that the scale-space is analysed by upsampling the filter size instead of reducing the image size. In order to localize interest points in the image and over scales, a non maximum suppression in a 3

$\times$ 3 $\times$ 3 neighborhood is applied. The maxima of the determinant of the Hessian matrix are then interpolated in scale and image space.

2.2.2 SURF Descriptor

The surf descriptor is extracted from an image in two steps: the first step consists of assigning an orientation based on information from a circular region around the interest point. Then, a square region aligned to the selected orientation is constructed, and the SURF descriptor is extracted from it.
To achieve the first step, the Haar-wavelet responses are calculated in horizontal x and vertical y direction in a circular neighborhood of radius $6s$ around the interest point, with s being the scale at which the interest point was detected. Once the wavelet responses are calculated and weighted with a Gaussian ($\sigma = 2.5s$) centered at the interest point, the responses are represented as vectors in a space with the horizontal response strength along the abscissa and the vertical response strength along the ordinate. The horizontal and vertical responses within an estimation window are summed. The two summed responses then yield a new vector. The orientation of the interest point is assigned to be same as the longest such vector.
To achieve the second step, the region is split up regularly into smaller square sub-regions. For each sub-region, a few simple features at regularly spaced sample points are computed. The horizontal and vertical wavelet responses are summed up over each sub-region and form a first set of entries to the feature vector. In order to bring in information about the polarity of the intensity changes, the sum of the absolute values of the responses is also extracted; therefore each sub-region has a four-dimensional descriptor vector $v = (\Sigma d_x, \Sigma d_y, \Sigma|d_x|, \Sigma|d_y|)$, where d_x denotes the horizontal wavelet response and d_y the vertical response. For an image divided into 4 $\times$ 4 sub-regions, the feature vector will be of overall length 64 [11].

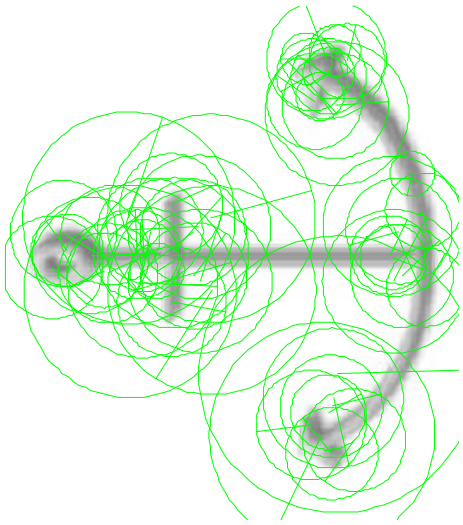


Figure 6: SURF interest points

2.2.3 Matching

To match a query image to images from the dataset, the same approach as for the SIFT vectors is used for the SURF vectors. The images with most similar matching interest points are retrieved (see section 2.1.5).

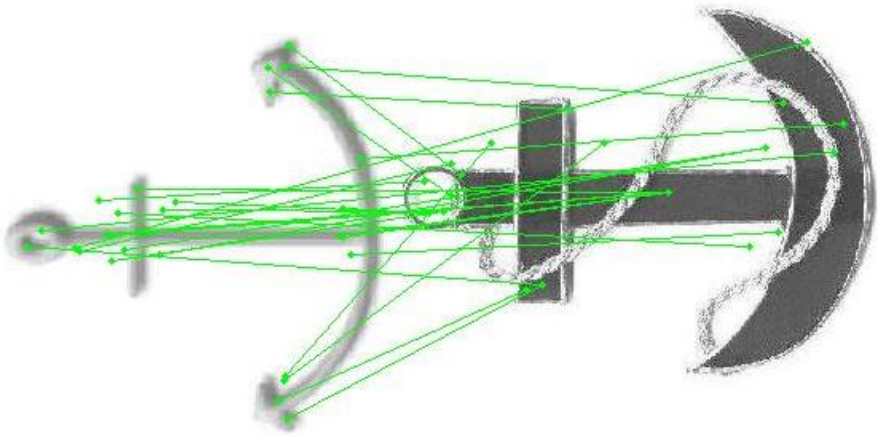


Figure 7: SURF interest point matching between two images of the class *anchor*

2.3 Color histograms

From a human visual point of view, color is a powerful descriptor that simplifies the recognition and extraction of objects from a scene. It is the way the human visual system measures a certain part of the electromagnetic spectrum [12]. A standard descriptor for the color of an image is the color histogram [13]. Given a discrete color space defined by some color axes (e.g. red, green, blue), the color histogram is obtained by discretizing the image colors and counting the number of times each discrete color occurs in the image array. Histograms are invariant under translation and rotation.
Given a color space, a 3D color histogram can be extracted for each image. In this project, histograms in HSV and CIE Lab spaces are used and compared to each other for image retrieval. Idée's multicolor search lab which searches the photo sharing website Flickr provides a nice demo on color-based image retrieval [14].

2.3.1 HSV color histogram

The HSV color coordinates are a cylindrical representation of point in the red, green, blue (RGB) color space. They describe color similarly to the human visual system description of color perception. In order to create the image histogram in this space, the image needs to be converted [12] from RGB color space (standard of the images of the data set used in this project):

$$S = \frac{\max(R,G,B) - \min(R,G,B)}{\max(R,G,B)}$$

$$V = \max(R,G,B)$$

If $S = 0$ then hue is undefined, otherwise:
If $R = max(R,G,B)$ and $G = min(R,G,B)$ then $H = 5 + B'$
If $R = max(R,G,B)$ and $G \neq min(R,G,B)$ then $H = 1 - G'$
If $G = max(R,G,B)$ and $B = min(R,G,B)$ then $H = 1 + R'$
If $G = max(R,G,B)$ and $B \neq min(R,G,B)$ then $H = 3 - B'$
If $B = max(R,G,B)$ then $H = 3 + G'$
Otherwise $H = 5 + R'$
With R', G' and B' defined as:

$$R' = \frac{\max(R,G,B) - R}{\max(R,G,B) - \min(R,G,B)}$$

$$G' = \frac{\max(R,G,B) - G}{\max(R,G,B) - \min(R,G,B)}$$

$$B' = \frac{\max(R,G,B) - B}{\max(R,G,B) - \min(R,G,B)}$$

Once the HSV values are obtained for a given image through conversion, a 3D histogram is created given these three layers [15]. This operation is done for every image in the dataset to create a feature space.

2.3.2 CIELab color histogram

The CIE Lab color space, created by the *Commission Internationale de l'Eclairage* is a perceptually uniform space in which the Euclidian distance between colors is strongly correlated with the human perception. This property makes the distance metric in this space interesting for image retrieval. However before creating a color histogram of the image, it needs to be converted into CIE Lab values. The conversion between RGB and CIE Lab uses the standardized CIE XYZ color space. The CIE XYZ color space is device independent contrary to RGB space and can be obtained by an affine transformation from the RGB values:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = [M] \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

where $[M]$ depends on the RGB working space. Given the XYZ values, the Lab values can be computed [12]:

$$L^* = 116(\frac{Y}{Y_n})^{\frac{1}{3}} - 16$$

$$a^* = 500[\left(\frac{X}{X_n}\right)^{\frac{1}{3}} - \left(\frac{Y}{Y_n}\right)^{\frac{1}{3}}]$$

$$b^* = 200[\left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} - \left(\frac{Z}{Z_n}\right)^{\frac{1}{3}}]$$

Once the CIE Lab values of a given image are computed, similar to the case of the HSV space, a 3D histogram of the Lab values is computed [16] and stored as a feature space.

2.3.3 Matching

In order to retrieve similar images to the query image, a distance metric is needed to compare the color histograms of the query image to those of each image in the dataset.

The distance L1 (also called city-block distance or Manhattan distance) and the distance L2 are considered to serve as a base for a distance metric:

$$dist_{L1}(\boldsymbol{a}, \boldsymbol{b}) = \|\boldsymbol{a} - \boldsymbol{b}\|_1 = \sum_i (a_i - b_i)$$

$$dist_{L2}(\boldsymbol{a}, \boldsymbol{b}) = \|\boldsymbol{a} - \boldsymbol{b}\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$$

where i goes through all indexes in the histograms a_i and b_i. The distance metric is obtained by thresholding out the extreme values from the normalized distance measure [15]. Images from the dataset having the smallest distance metric with the query image will be retrieved. In this project, it is experimentally discovered that the L1 distance metric yields better retrieval results, moreover, for each histogram layer, 11 bins are considered, a histogram for a given image will be of size 11 ×11 × 11.

2.4 Edge orientation histogram

In an image, an edge is defined as an abrupt change of the image intensity. Edges are an important descriptor of the content of an image as they provide a description of the overall shape of objects in an image; moreover the human visual perception is sensitive to edges. The edge orientation histogram (EOH) represents the frequency and directionality of edges in an image [3].

2.4.1 Edge extraction

To create a histogram of edge types in an image, it is first divided into 4 × 4 rectangular non-overlapping regions. Edges are extracted in each region and ordered in a histogram based on their type. Five types of edges are considered: vertical, horizontal, 45º, 135º and non-directional edges (see Figure 8).

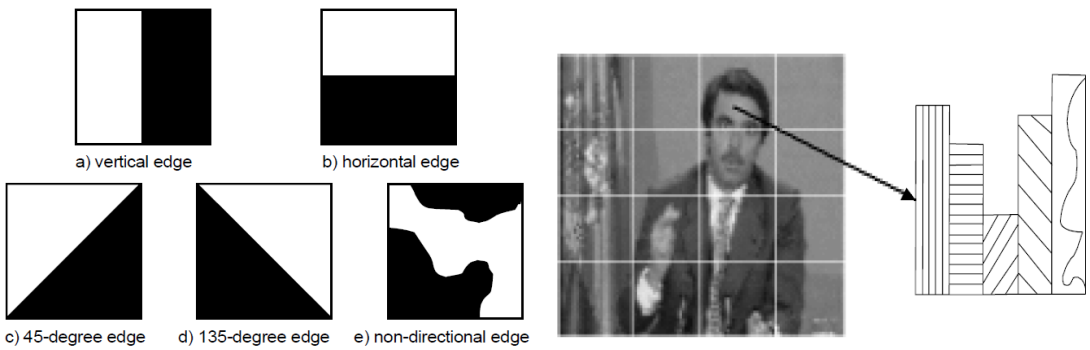


Figure 8: Five types of edges are considered for EOH [3]

Edge detection is achieved by using the Canny edge detector [17]. The resulting feature histogram is a 4 × 4 ×5 matrix [18] which is then normalized. For each image in the dataset used in this project, the edge orientation histogram is extracted and saved in a feature space for future matching.

2.4.2 Matching

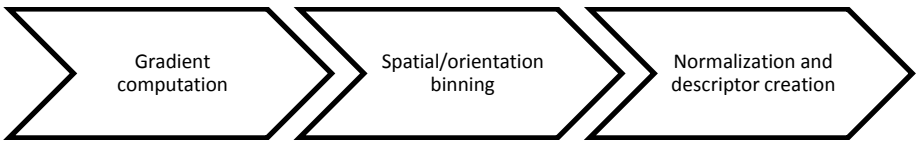
In order to retrieve similar images to a query from a dataset, the extracted feature vector of the query is compared to each saved histogram. To this effect, the Bhattacharyya measure is used [19]. This metric measures the approximate overlap of two populations:

$$\rho(\boldsymbol{a}, \boldsymbol{b}) = \sum_i \sqrt{\sum a_i \,.\, \sum b_i}$$

where i goes through all indexes in the histograms a_i and b_i. The images from the dataset, having the most overlap are selected for retrieval.

2.5 Gradient histogram

Grids of Histograms of Oriented Gradient (HOG) descriptors have been shown to be efficient in human detection [4]. The method is based on evaluating normalized local histograms of image gradient orientations in a dense grid. The basic idea is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions.
The extraction of a histogram of oriented gradients for a given image is achieved in the following major steps:



2.5.1 Gradient computation

The HOG features are extracted on the gray level version of an image. After this preliminary conversion the image is smoothed using a Gaussian filter to avoid small abrupt changes in intensity to affect the gradient computation. Gradient magnitude and orientations are computed for the image, as shown in Figure 9:

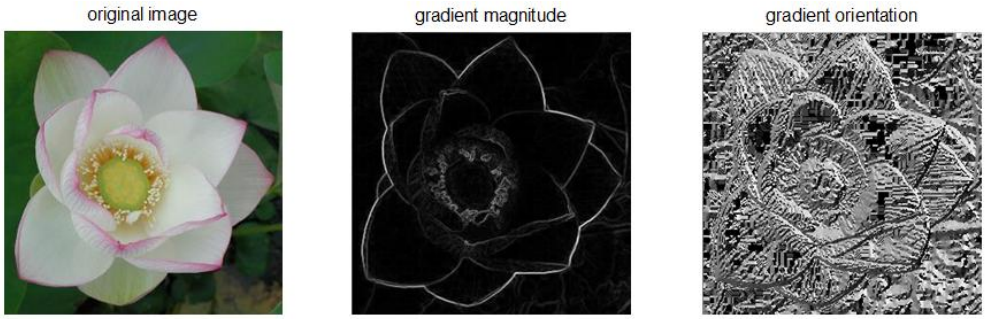


Figure 9: Gradient magnitudes and orientations for a given image

2.5.2 Spatial/orientation binning

To create a histogram of gradients, each pixel contributes a weighted vote for an edge orientation histogram channel based on the orientation of the gradient element centered on it. The votes are accumulated into bins over local spatial regions that are called cells. Cells can be either rectangular or radial (log-polar sectors). The orientation bins are evenly spaced over 0º-180º (unsigned gradient) or 0º-360º (signed gradient). In the implementation used here, there are 5 rectangular cells along each direction (horizontal and vertical) and signed gradients are considered. Histograms of the orientations weighted by the magnitudes are created for each cell. Cells are grouped in a block. Concatenation of histograms over blocks creates a feature vector. Blocks overlap in one cell in each direction (see Figure 10).

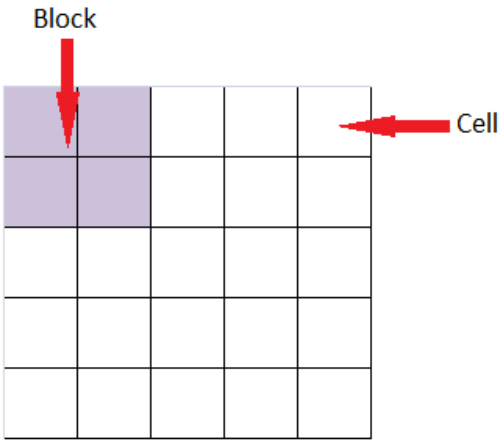


Figure 10: Image division for HOG computation

2.5.3 Normalization and descriptor creation

Since gradient values vary a lot due to foreground-background contrast, the gradient values in a cell are normalized over several blocks. In this implementation, the histograms are normalized with the L2 norm. For each cell, a 20 bin histogram of weighted gradients is created. There are 4 cells in a block, and 4 blocks in each column and row. Once all the histogram bins are concatenated, the resulting feature vector is of length $20 \times 4 \times 4 \times 4 = 1280$ elements [20].

2.5.4 Matching

The histograms are concatenated and linearized to create a vector of length 1280. Similar to the case of matching edge orientation histograms (see section 2.4.2), histograms of oriented gradients are compared to each other using the Bhattacharyya measure. Given a query image, images from the dataset whose feature vector most overlaps that of the query, are retrieved.

2.6 Texture

Texture is defined as differences in the spatial arrangement of gray values of neighboring pixels and is an important visual feature in image segmentation and scene

understanding. There are different ways to analyze and describe texture. In this project, two methods are implemented and tested:

- The Gabor texture feature method [6] is a texture segmentation algorithm inspired by the multi-channel filtering theory for visual information processing in the early stages of human visual system.
- The gray level difference method, which is proposed for detection of clustered microcalcifications on digitized X-ray mammograms [5]. Microcalcifications, one of the early indicators of breast cancer, are tiny granule-like deposits of calcium which have an average diameter of 0.3 mm. This method is based on gray level statistical features of the image.

2.6.1 Gabor texture features

Texture feature extraction by using Gabor filters is referred to as the multi-channel filtering approach [6]. It takes into account the fact that the dominant frequency of different textures is different. A systematic filter selection scheme is used, which is based on reconstruction of the input image from the filtered images. Texture features are obtained by subjecting each (selected) filtered image to a nonlinear transformation and computing a measure of "energy" in a window around each pixel.
The following steps are achieved in order to extract a Gabor texture feature:

1) Functional characterization of the channels and the number of channels: A bank of two dimensional Gabor filters is used to characterize the channels. A canonical Gabor filter in the spatial domain is:

$$G(x,y) = \exp\{-\frac{1}{2}[\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}]\}\cos(2\pi u_0 x + \Phi)$$

where u_0 and Φ are the frequency and phase of the sinusoidal plane wave along the z-axis (i.e. the 0º orientation), and σ_x and σ_y are the space constants of the Gaussian envelope along the x- and y-axis, respectively. In this project, a set of filters at six orientations $[0: pi/6: pi - pi/6]$ at ten different scales, [2:2:10] chosen experimentally, are used.

2) Extraction of appropriate texture features from the filtered images: for each scale and each orientation, two features μ and σ are computed:

$$\mu_{mn} = \frac{E(m,n)}{P \times Q}$$

where $P \times Q$ is the size of the image,

$$\sigma_{mn} = \frac{\sqrt{\sum_x \sum_y (|G_{mn}(x,y)| - \mu_{mn})^2}}{P \times Q}$$

With

$$E(m,n) = \sum_x \sum_y |G_{mn}(x,y)|$$

where m is the scale and n is the orientation, and G is the filtered image.

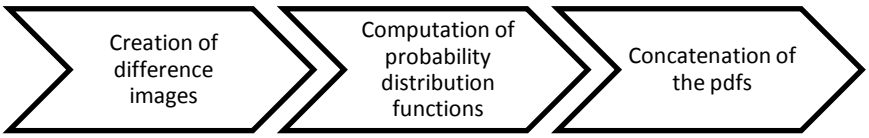
3) Integration of texture features from different channels to produce a segmentation: by gathering the μ and σ features computed for each scale and each orientation, the Gabor feature vector for a given image becomes:

$$f = (\mu_{00}, \sigma_{00}, \ldots, \mu_{mn}, \sigma_{mn})$$

In this project, the Gabor feature vector is of length $6 \times 20 = 120$.

2.6.2 Gray level difference method

In the gray level difference method (GLDM), difference images are created in four directions and the gray level histograms of these four new images are linearized to create a feature vector. The feature extraction is achieved in the following steps:



1) Creation of difference images: in the four directions north-east, north-west, south-east and south-west difference images are created: then the probability distribution function is created from the gray level histograms of these four new images. Given a step distance d, the four difference images are:

$Image(i,j) - Image(i,(j+d))$
$Image(i,j) - Image((i-d),(j+d))$
$Image(i,j) - Image((i+d),j)$
$Image(i,j) - Image((i-d),(j-d))$

2) The probability distribution function is created from the cumulative sum of the gray level histograms of these four new images. Each probability distribution function is given a length of 256.
3) The four probability distribution functions are concatenated to create a feature vector of length 1024 [21].

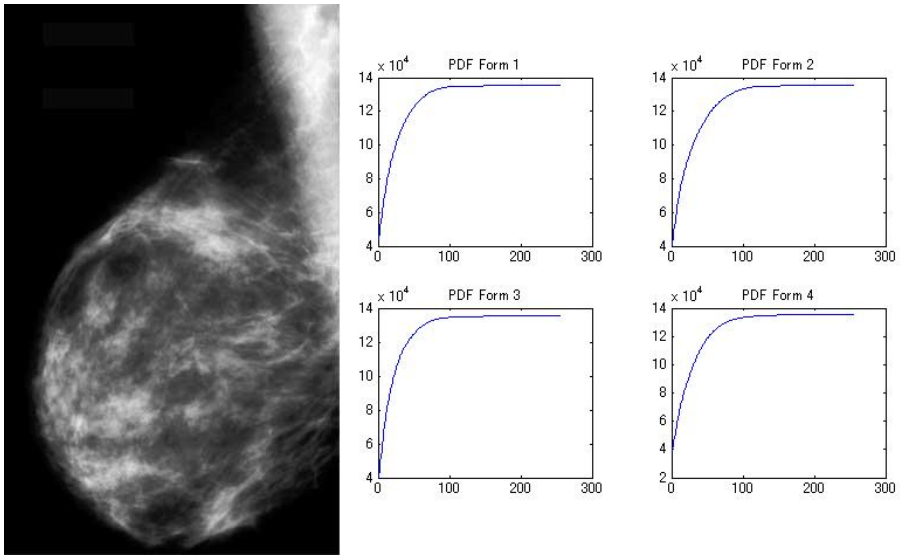


Figure 11: Four probability distribution functions in the GLDM [21]

2.6.3 Matching

Similar to the case of matching edge orientation histograms (see section 2.4.2), gray level difference feature vectors and Gabor feature vectors are compared to each other using the Bhattacharyya measure. Given a query image, images are retrieved from the dataset whose histogram most overlaps that of the query.

3. Tag propagation

Tags are textual information about an image added by a user. They may describe its visual or semantic content and allow quick reference to a given image. Tags allow efficient contet-aware management of a database. The main goal of this project is to allow for the tags of the query image to be propagated to similar images given to the user through content-based image retrieval. The user will select images from the results of the image retrieval system he/she thinks are relevant to the query image or its tag. The the query image tags, or newly entered tags are added to the tag collection of the selected images. The tag propagation operation consists in adding the tag to the collection of already existing tags of a chosen image. If the image doesn't have any tags, the tag propagation operation creates the tag support as well. To demonstrate this operation, a Matlab GUI interface is developed. It enables the choice of a query image, input tags to be propagated and a choice of features for image retrieval. Once similar images have been retrieved, the user chooses the relevant images for tag propagation.

3.1 Tag creation

The tag given to an image by a user might specify its visual content, or describe its semantic content. It is stored along with the image, in the case of this project in a text file carrying the same name as the image. The tag files can be stored in the same location as the image or such as the case of this project, a separate folder can be assigned as a tag-space. Once the user specifies a tag, and decided on propagation it will be propagated to the query image and the selected retrieved images. If the user doesn't specify a new tag, the tags of the query image will be propagated to the chosen relevant similar images.

Figure 12: Tag propagation segment of the developed demo

4. Developed user interface

A user interface is created to demonstrated content-based image retrieval, as described in this report, and tag propagation. To this effect, Matlab Graphical User Interface (GUI) is used. The interface enables the choice of the query image and tag, an image-retrieval scheme and the choice of the number of results to be shown. Once similar images are retrieved, a check-box along with each image enables it to be selected by the user as relevant to the search. The user can select an arbitrary number of relevant images to apply tag propagation on.

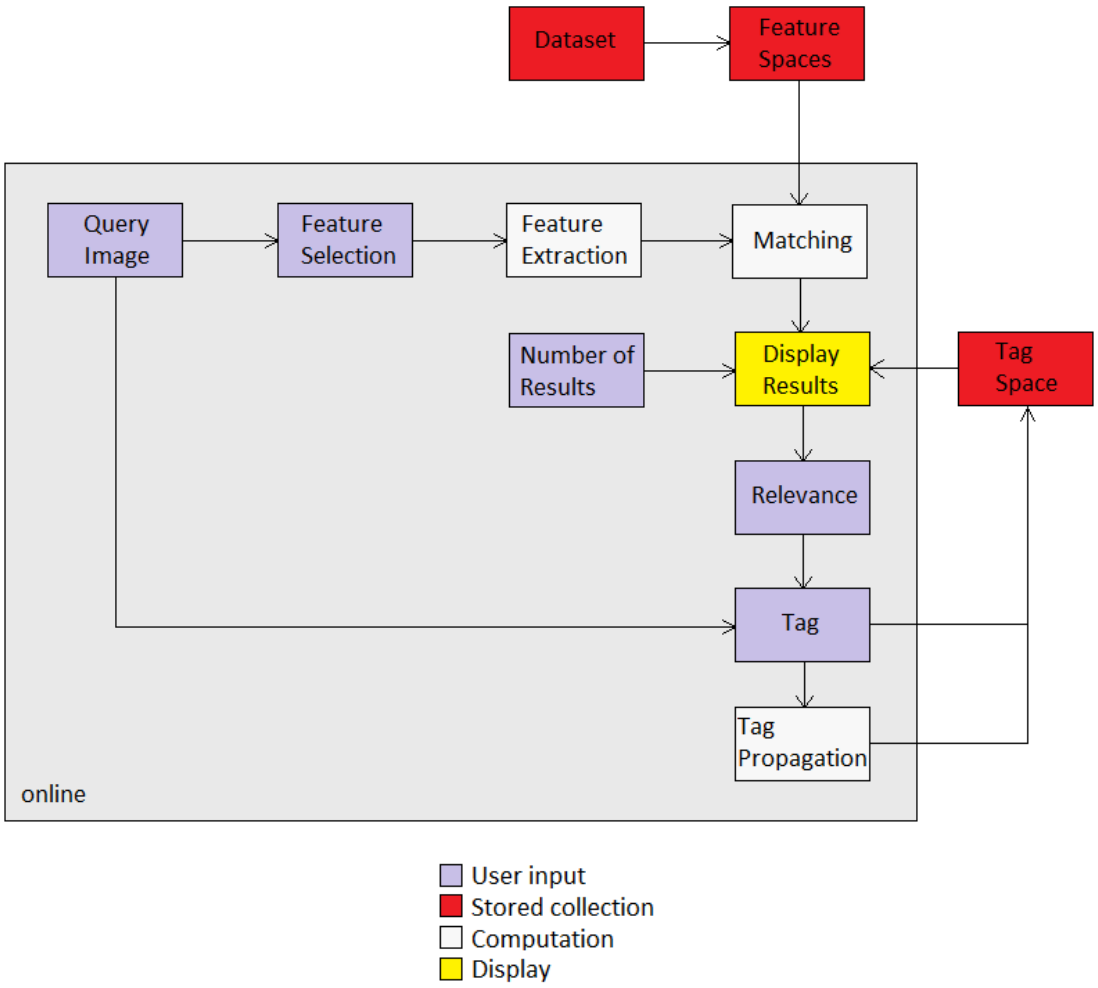


Figure 13: Flow diagram of developed system for tag propagation

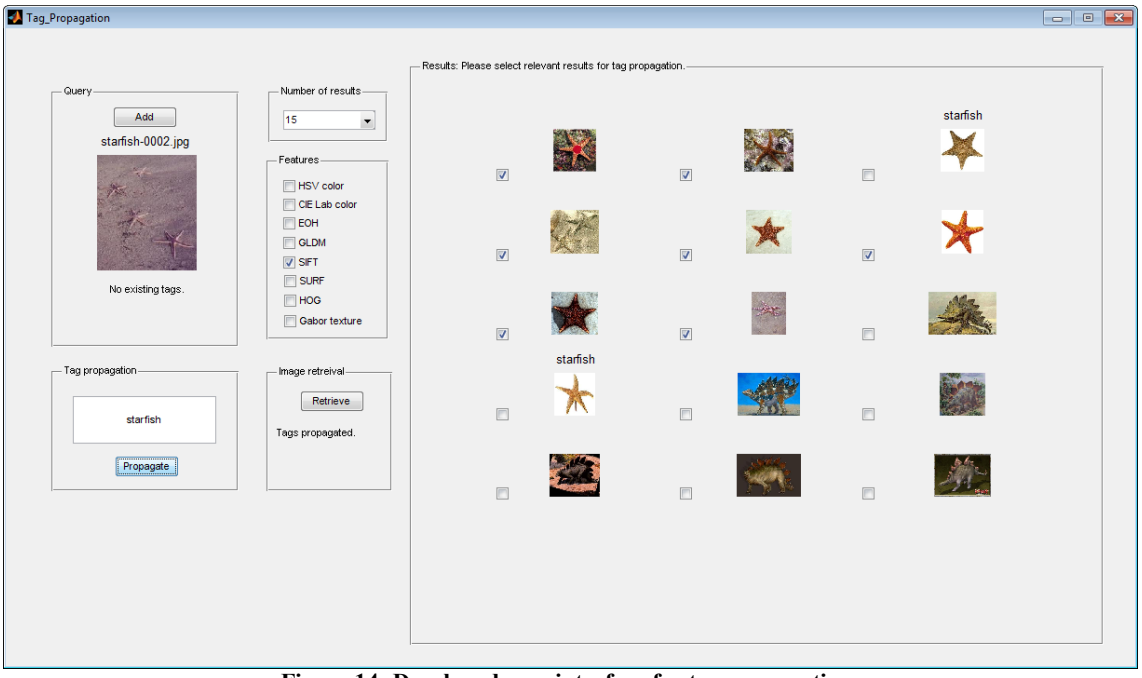


Figure 14: Developed user interface for tag propagation

If any of the retrieved images or the query image has tags, they are displayed along with the image. Only one image retrieval scheme can be selected. Here, the image retrieval scheme refers to the desired feature used for image similarity measurement and retrieval. The number of results to be displayed can be selected as well. Tags entered by the user are added to the existing tags of the images.

The query image can be selected from any location on the hosting computer. Once the number of results and image processing scheme are selected, the *Retrieve* button activates the image retrieval system. The image retrieval system extracts image features from the query and matches them against the features saved in the feature space. The selected number of results is displayed along with existing tags for each image. The user can now select images he/she judges as relevant. Then he/she can enter a tag, and by activating the *Propagate* button, the entered tags are propagated to the query and selected images. If no new tags are entered, the activation of the *Propagate* button will propagate the already existing tags of the query image to the selected images.

The dataset is stored in a folder whose path is specified internally for the image retrieval search schemes. The feature spaces are a set of matrices (*.mat* file) containing the previously extracted features of each image in the specified dataset. These features can be stored in these matrices under different forms, such as vectors or histograms. The tag space is a folder containing all existing image tags. In this application, tags are stored in a *.txt* file for each image. The tag file of a given image carries the exact same name as the image, except the file extension. Each time a tag is created through the interface, the corresponding image's tag file is updated and displayed.

5. Experiments

To test the image retrieval and tag propagation, with the developed interface, a set of images is needed as image and feature database. Experiments are then performed on this dataset. In the experiments described in this section, query images are chosen from the dataset. The experiments aim at providing a sample of the performance of the system through precision assessment.

5.1 Dataset

For the purposes of this project, the Caltech 101 [22] dataset is used for image retrieval. This dataset is available online[1] and contains 101 object classes along with a background class. It was compiled in 2003 with the intent to facilitate computer vision research and techniques. In this project, a selection of 11 images per class, dismissing the black and white images (and the entire black and white class of *car_side*), is used. The total number of images in this selection adds up to 1111. Query images can be chosen from the selection dataset or can be specified from another location by the user. Training models for each image retrieval scheme are created for the images in this selection.

Figure 15: Sample of images from the dataset

[1] http://www.vision.caltech.edu/Image_Datasets/Caltech101/

Figure 16: Sample images belonging to the same class

5.2 Performance evaluation

The tag propagation system doesn't need extensive evaluation, as it is a simple task and doesn't present any flaws. The entered tags added by the user (or the existing tags of the query image, in case no new tags are entered) are propagated to the selected relevant images. The image retrieval system, on the other hand, needs extensive evaluation and performance assessment. Different image retrieval schemes used here operate differently on different object classes, for example the GLDM for texture feature extraction was specifically designed for analyzing digitized mammograms [5] and may not operate in a satisfactory manner for matching airplanes for example. The EOH [3] may match flowers to each other but it might not make a difference between a lotus and a sunflower, however the color histogram will differentiate them. The interest of presenting many different features for image retrieval lies also in the ability to determine whether some classes of images are retrieved better using one feature rather than another. To evaluate the performance of the image retrieval system, a scheme is designed to assess performance of each image retrieval scheme on a given class of objects. For a given class and a given retrieval scheme, all the images belonging to that class are considered and similar images to each image in the given class are retrieved using the specified scheme. The overall performance of a retrieval scheme is objectively computed by a mean accuracy on retrieved images over a class. In information retrieval systems, the two metrics precision and recall [23] are widely used metrics to assess performance of a retrieval system. However, given the nature of retrieval in this project, it is preferred to define the accuracy of the system, over all images of a given class, for a given retrieval scheme as follows:

$$accuracy@N = \frac{number\ of\ correctly\ retrieved\ images\ out\ of\ N\ retrieved\ images}{N\ retrieved\ images}$$

The measures given in the following section give an assessment on the performance of the retrieval system over the $N = 5$ first retrieved images, and over all possible retrievable images in a class of objects, $N = 11$ in the case of the dataset used here. Users of retrieval systems often tend to look more at the very first results given to them, therefore it is interesting to judge the system over a small given number of fist results ($N = 5$ in this case). It is noted that the performance of an image retrieval scheme is directly linked to the quality of the image and the position of the interest region or object.

5.3 Results

The following classes are tested for all the different image retrieval schemes described previously using the measure $accuracy@N$ (in this project, N takes values 5 and 11):

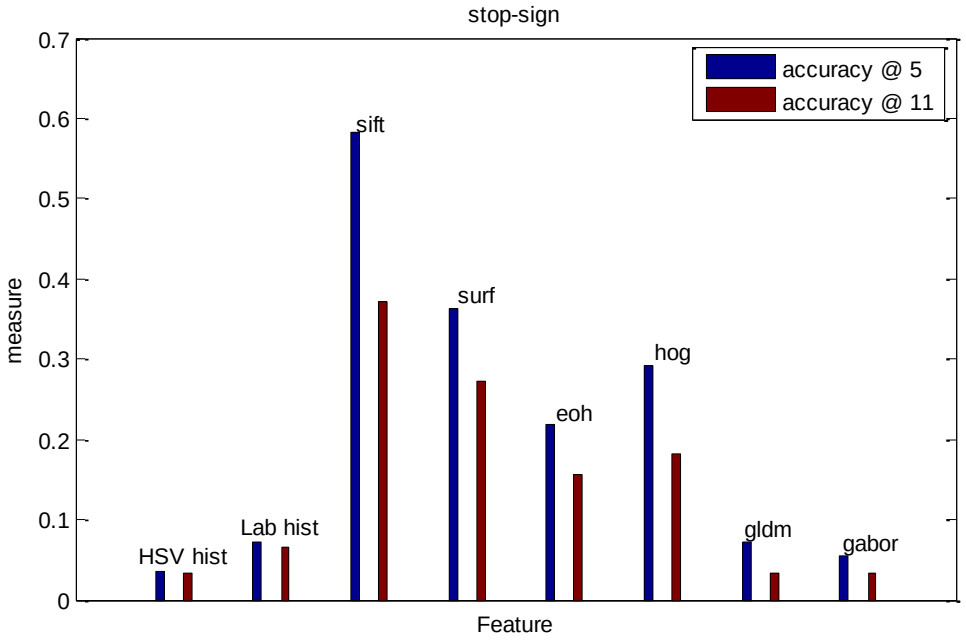


Figure 17: Evaluation for the *stop_sign* class

The *stop_sign* class presents images of signs under different angles, therefore image retrieval using SIFT perform best among the image retrieval schemes.

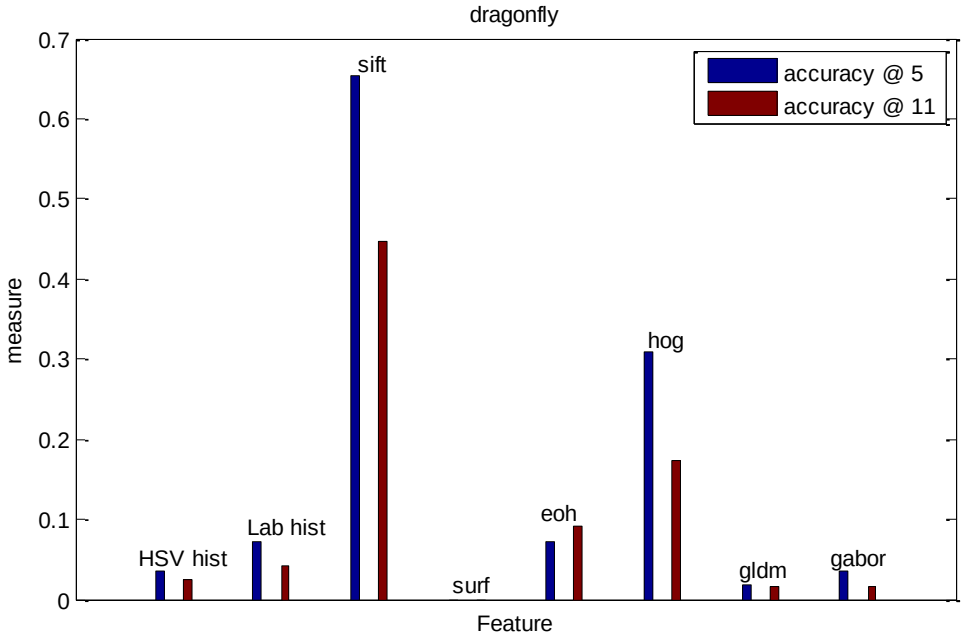


Figure 18: Evaluation for the *dragonfly* class

Similarly to the image in the class *stop_sign*, the images in the class *dragonfly* present the object of interest under different views, therefore the SIFT features perform best for image retrieval.

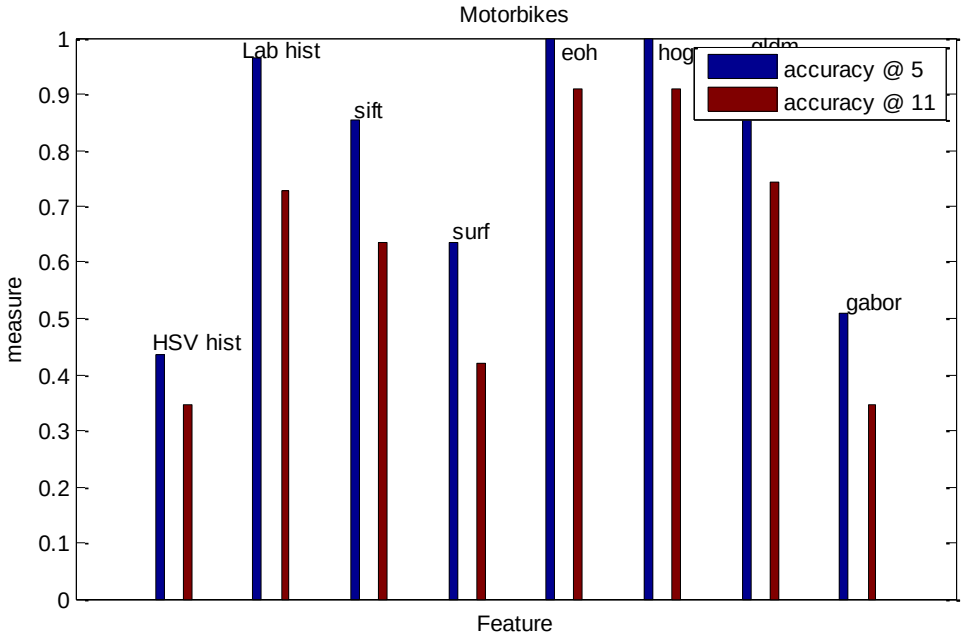


Figure 19: Evaluation for the *Motorbikes* class

The images in the *Motorbikes* class present motorbikes, as the object of interest, always on a white background, from the same angle. Therefore, all features perform in a satisfactory manner. However, they are retrieved best by using edge orientation histograms and histograms of oriented gradients.

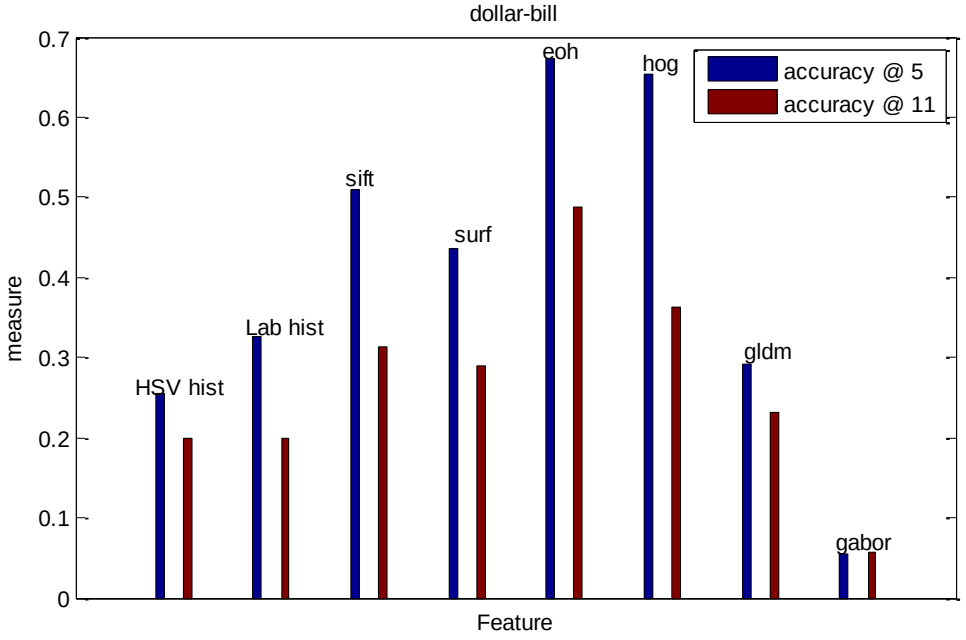


Figure 20: Evaluation for the *dollar_bill* class

For the images in the *dollar_bill* class, the dominant color is green, therefore it can be seen that color features perform better than for other classes. Also the shape of a bill is distinctive; therefore the edge features perform best for image retrieval.

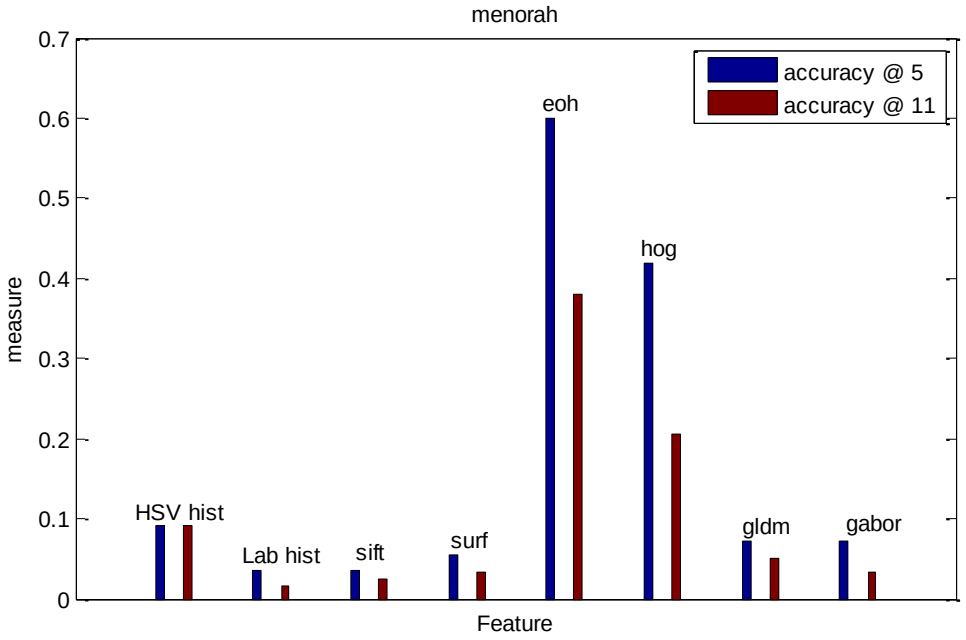


Figure 21: Evaluation for the *menorah* class

For the images in the *menorah* class, the shape and edges are the most distinctive features; therefore the edge features such as EOH and HOG perform best for image retrieval.

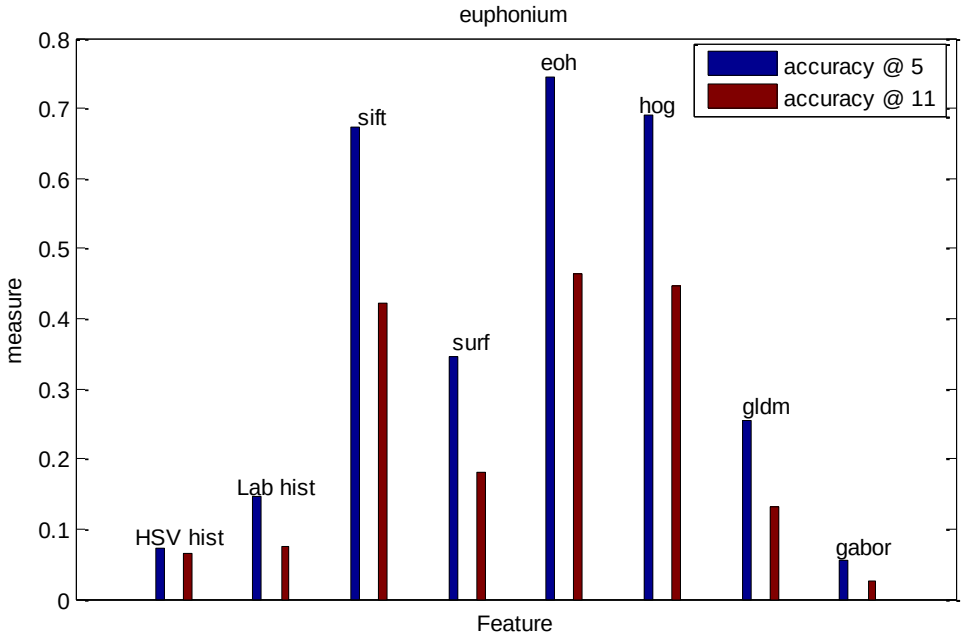


Figure 22: Evaluation for the *euphonium* class

For the images in the *euphonium* class, the edge is the most distinctive feature, therefore image retrieval using dominant edge features performs best, additionally SIFT features seem to perform in a satisfactory manner.

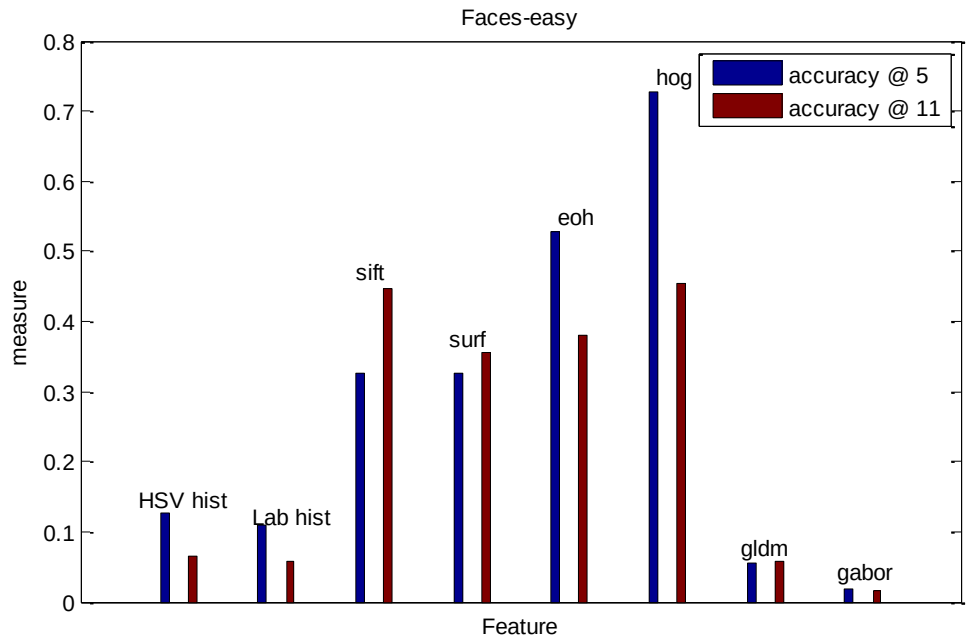


Figure 23: Evaluation for the *faces_easy* class

The images in the *faces_easy* class are retrieved best by the edge features HOG. In these images, the general shape of the face and elements inside are most discriminant. On the other hand, texture doesn't seem to be a distinctive feature of a face, which is understandable.

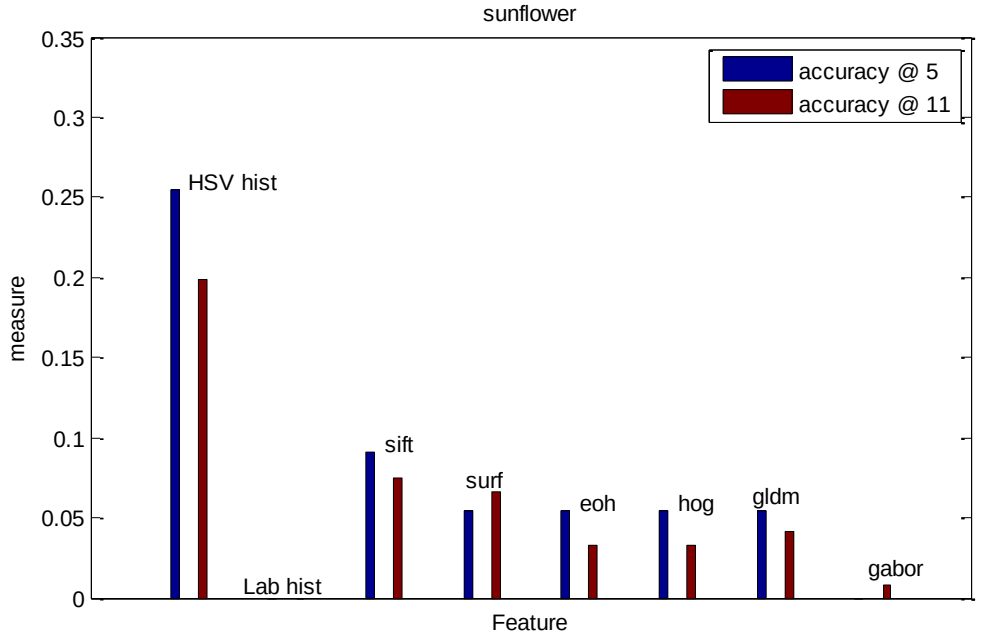


Figure 24: Evaluation for the *sunflower* class

For the images in the *sunflower* class, color seems to be the most discriminant feature; it is understandable, since most sunflowers are yellow and black with green leaves. Sift features operate better than edge or texture, since these images present sunflowers under different angles.

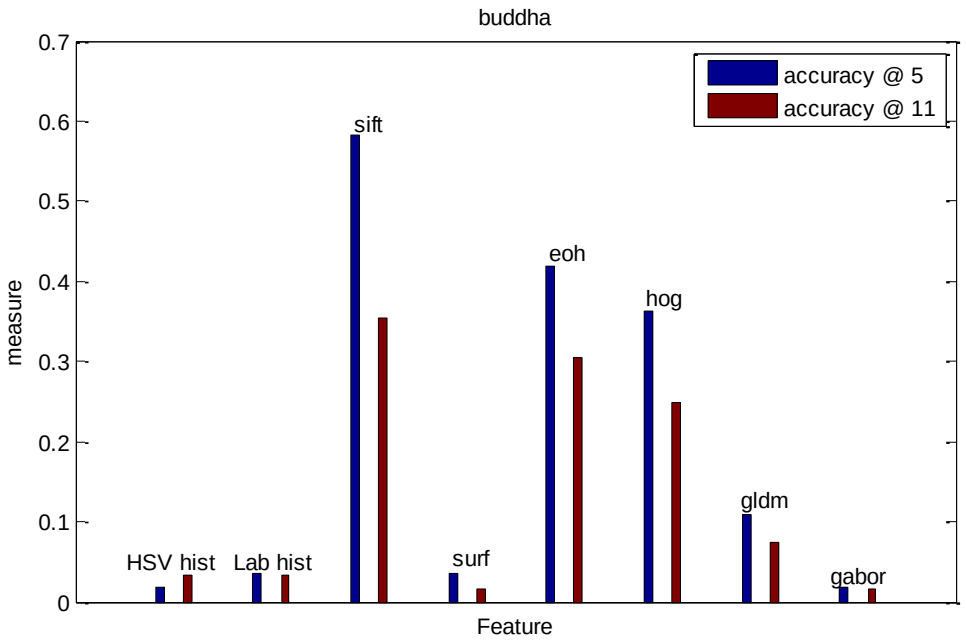


Figure 25: Evaluation for the *buddha* class

For the images in the *buddha* class, image retrieval using SIFT performs best, since the object of interest, the Buddha statue, changes shape and color often and is not always presented under the same angle. It can be seen that retrieval based on color and texture is not satisfactory, as well as retrieval with SURF features, which is an interesting and rather unexplained phenomenon.

An overall assessment of the performance of the presented image retrieval feature vector or histograms results in many more specific assessments. Indeed, based on the nature of the query image, it is best to search for similar images with a scheme taking into account it's most dominant features. Also, the accuracy directly depends on the quality of the interest object in an image. The size and position of the object with respect to the image, the background nature (color, size of background objects), and the lightening conditions, all affect the performance of image retrieval based on local or global visual features of the image. It is also noticed that accuracy over the five first results is often much better than accuracy over 11 results, which indicates that often the relevant images are in the very first results presented to the user.

6. Conclusion

In this project, different image retrieval schemes were used to perform content based retrieval in order to help efficient annotation propagation of images over a dataset. Color, edge, texture and local distribution based features were tested on images of the Caltech 101 dataset. The outputs of this content based image retrieval are used to propagate tags of the query image to similar images in content. Tags are defined as textual annotations a user makes for a given image. Through the system designed in this project, the tags of a given query image can be propagated to similar images retrieved through the image retrieval system. New tags can be added to the query image and propagated to its similar images as well.
The mentioned retrieval schemes along with the tag propagation system were implemented on Matlab in a GUI accessible to any user. After an objective evaluation of the system, it is noticed that the tag propagation system in itself is relatively simple and works as expected. It can be easily combined with the retrieval schemes to allow for a user to select the retrieval scheme and then among the outputs, select relevant images for tag propagation. It was established that different schemes operate differently on different images. Based on the nature of the query image, there may be one or several schemes that operate better than other for retrieval of similar images. Also the accuracy of the retrieval system highly depends on the quality of the interest object and the way it is depicted in the image. The following general observations are made on the obtained results:
The SIFT features are performant on images that present different views of the same object, such as the images in the *stop_sign* and the *buddha* classes. Retrieval using SIFT features is a lot more time and resource consuming than retrieval with SURF but on the positive side yields better results.
The color features are performant on images where color is a dominant feature for recognition, such as the *dollar_bill* class. However, a more specialized dataset is needed to correctly assess performance of the color histogram matching, as the images in the current dataset aren't particularly suitable to this effect.
Edge features are performant on images which present objects of the same shape, such as the *menorah*, *dollar_bill* and *euphonium* classes.
The image in the chosen database don't give satisfactory results on the texture feature-based image retrieval, a more specialized dataset should be used to assess performance of these schemes.

6.1 Further improvement

Different improvements can be brought to the current system. Among which, the following ideas:

- In the tag propagation segment, more advanced operations can be applied on the tags, for example the entered tag can be compared to existing tags to ensure such tag doesn't already exist. Text recognition and correction can be applied as well.
- Several of the described image retrieval features can be combined to take into account different discriminant properties of an image for retrieval of similar images. For example, to correctly retrieve similar images to a given flower, the dataset is first queried based on edge histogram, and then the retrieved images are queried based on the dominant color. With this logic, more than two schemes can be combined in steps.

- The color features can be improved by taking into account the spatial organization of colors within the object of interest.
- The implementation of such system without using Matlab might improve the speed and flexibility of the system for different purposes.

7. References

[1] **Lowe, D. G.** Distinctive Image Features from Scale-Invariant Keypoints. *Journal of Computer Vision.* 2004.
[2] **Bay, H., Tuytelaars, T. and Gool, L. Van.** SURF: Speeded Up Robust Features. *Proceedings of the ninth European Conference on Computer Vision.* 2006.
[3] **Won, C. S., Park, D. K. and Park, S. J.** Efficient Use of MPEG-7 Edge Histogram Descriptor. *ETRI Journal.* 2002.
[4] **Dalal, N. and Triggs, B.** Histograms of Oriented Gradients for Human Detection. *IEEE Computer Society Conference.* 2005.
[5] **Kim, J. K. and Park, H. W.** Statistical Textural Features for Detection of Microcalcifications in Digitized Mammograms. *IEEE Transactions on medical imaging.* 1999.
[6] **Jain, A. K. and Farrokhnia, F.** Unsupervised Texture Segmentation Using Gabor. *Pattern Recognition.* 1991.
[7] **Google.** Google Images. [Online] http://images.google.com/.
[8] **Idée inc.** Piximilar. [Online] http://www.ideeinc.com/products/piximilar/.
[9] **Deselaers, T.** Features for Image Retrieval. 2003.
[10] **Vedaldi, A.** SIFT implementation for Matlab. [Online] http://www.vlfeat.org/~vedaldi/code/sift.html.
[11] **Strandmark, P.** [Online] 2008. https://code.ros.org/svn/opencv/trunk/opencv.
[12] **Tkalcic, M. and Tasic, J. F.** Colour spaces - perceptual, historical and applicational background. *The IEEE Region 8 EUROCON.* 2003.
[13] **Swain, M. J. and Ballard, D. H.** Color Indexing. *International Journal of Computer Vision.* 1991.
[14] **Idée inc.** Idée Multicolr search lab flickr set. [Online] http://labs.ideeinc.com/multicolr/.
[15] **Giannakopoulos, T.** Image retrieval - Query by Example Demo. *Matlab Central.* [Online] 2008. http://www.mathworks.com/matlabcentral/fileexchange/22030-image-retrieval-query-by-example-demo.
[16] **Ivanov, I.** Implementation of CIELab histogram creation for Matlab. 2010.
[17] **Canny, J.** A Computational Approach To Edge Detection. *IEEE Trans. Pattern Analysis and Machine Intelligence.* 1986.
[18] **Conaire, C. Ó.** Computer vision source code. [Online] http://elm.eeng.dcu.ie/~oconaire/source/.
[19] **Derpanis, K. G.** The Bhattacharyya Measure. [Online] 2008. http://www.cse.yorku.ca/~kosta/CompVis_Notes/bhattacharyya.pdf.
[20] **Ivanov, I.** Implementation of HOG for Matlab. 2010.
[21] **Narayanan, A. S.** Texture Feature Extraction implementation - GLDM. *Matlab Central.* [Online] http://www.mathworks.com/matlabcentral/fileexchange/25057-texture-feature-extraction-gldm.
[22] **Fei-Fei, L., Fergus, R. and Perona, P.** Learning generative visual models. *IEEE. CVPR 2004, Workshop on Generative-Model.*
[23] **Makhoul, J., et al.** Performance measures for information extraction. *Proceedings of DARPA Broadcast News Workshop.* 1999.