

1 State of the art

Color image processing has become a major issue since a few years, most of the colour texture discrimination having been explored using the marginal colours way. The issue is that we are now able to do colour image recognition on digital images but the results on nature pictures are rather mediocre.

The CLEF contest has been created as an answer to that problematic, making universities' and laboratories' own solutions compete against each other in order to find the best colour texture feature.

In this document we will introduce key-points and their use in the various descriptors. We will go first with the standard ones which are SIFT SURF and opponent SIFT. The last one being the descriptor used by FINKI, the laboratory from the last year contest we chose as reference to compare our results. We will then use a new descriptor offered by Noel Richard, the C₂O.

1.1 Key-points

1.1.1 Classical

For this type of keypoint detection, the aim is to obtain only the useful keypoint (which are characterizing the image the best) without taking an "abstract" of the whole image as the dense grid method. To do it, the first thing that is done is to create a pyramidal tree containing some copies of the image at different resolutions. These copies will be blur by different Gaussian filter.

$$L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y) \quad (1)$$

The images are grouped by octaves. The resolution is divided by 2 between each consecutive octave : for the difference of Gaussians, it's equivalent to multiply σ by 2).

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma) \quad (2)$$

For each octave, it's made a difference between each two consecutive images blur by the filter at scale $k * \sigma$. This difference result is that the objects that are remaining on the difference image are thus which size is include between σ and $k * \sigma$. The parameter k is a constant of the algorithm that is fixed following a precise rule which depend on the precision that the user would. Because of the different resolutions, for each octave the difference result will keep higher and higher object that will allow us to detect approximately all the sizes of important features on the image. With all these difference calculated, the algorithm will take the maximas of each one as keypoints.

After doing that, the algorithm must discriminate and precise the coordinate of a part of the keypoint. Indeed, using different resolutions give some imprecise coordinate so it's necessary to make an interpolation to obtain the coordinate corresponding to the original image for keypoint extracted from the most reduced resolutions images. It is necessary to remove some kind of point too. The points that have not enough contrast comparing to the other are removed as the points that are on ridges (these points are really unstable and could move or disappear for many reasons so it's better to remove them).

When these operations have been computed, we obtain a set of keypoints that characterize the image. This method allows to obtain the keypoints to characterize an image without taking a predefined set of point. So the set of points obtained is potentially lighter than the one obtained by the dense grid method and it is more accurate because only important points wherever they are are conserved.

1.1.2 Dense Grid

The dense grid method is the easiest way to extract key-points. The image has first to be divided into k sub-images and the intersections of the sub-images' outlines become the key-points.

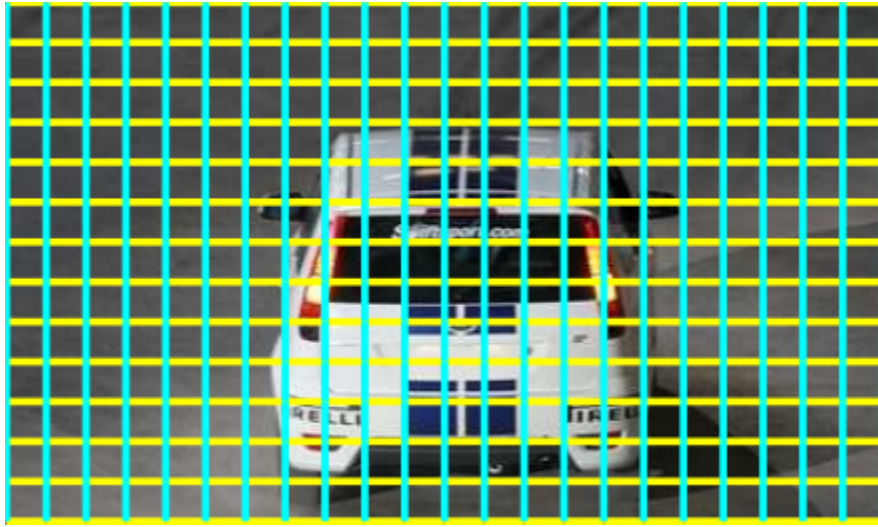


Figure 1: Dense grid

1.2 Descriptors

1.2.1 SIFT

Scale-invariant feature transform (or SIFT) is an algorithm in computer vision to detect and describe local features in images. The algorithm was published by David Lowe in 1999. Applications include object recognition, robotic mapping and navigation, image stitching, 3D modeling, gesture recognition, video tracking, individual identification of wildlife and match moving.

The algorithm is patented in the US; the owner is the University of British Columbia.

1.2.1.1 SIFT keypoints of objects are first extracted from a set of reference images and stored in a database. An object is recognized in a new image by individually comparing each feature from the new image to this database and finding candidate matching features based on Euclidean distance of their feature vectors. From the full set of matches, subsets of keypoints that agree on the object and its location, scale, and orientation in the new image are

identified to filter out good matches. The determination of consistent clusters is performed rapidly by using an efficient hash table implementation of the generalized Hough transform. Each cluster of 3 or more features that agree on an object and its pose is then subject to further detailed model verification and subsequently outliers are discarded. Finally the probability that a particular set of features indicates the presence of an object is computed, given the accuracy of fit and number of probable false matches. Object matches that pass all these tests can be identified as correct with high confidence.

1.2.2 SURF

In computer vision, Speeded Up Robust Features (SURF) is a local feature detector that can be used for tasks such as object recognition or 3D reconstruction. It is partly inspired by the scale-invariant feature transform (SIFT) descriptor. The standard version of SURF is several times faster than SIFT and claimed by its authors to be more robust against different image transformations than SIFT.

SURF uses an integer approximation of the determinant of Hessian blob detector, which can be computed with 3 integer operations using an integral image. For features, it uses the sum of the Haar wavelet response around the point of interest. These can also be computed with the aid of the integral image.

SURF descriptors can be used to locate and recognize objects, people or faces, make 3D scenes, track objects and extract points of interest.

SURF was first presented by Herbert Bay et al. at the 2006 European Conference on Computer Vision. An application of the algorithm is patented in the US.

1.2.3 SIFT vs SURF

The recognition of images or objects, is one of the most important applications of computer vision, becomes a comparison of local descriptors SIFT (Scale Invariant Feature Transform) and SURF (Speeded-UP Feature transform). These two local descriptors detect structures or very significant points in an image for a discriminating description of these areas from its neighboring points, in order to compare them with other descriptors using similar measures.

1.2.3.1 SIFT Algorithm:

- Detection of edges on the scale: the first stage performs a search in different scales and dimensions of the image by identifying possible points of interest, invariant to changes in orientation and scaling. This procedure is carried out with the function (DoG (Difference-of-Gaussian) giving different values to the σ , in the following equation):

$$D(x, y, \sigma) = L(x, y, k_i \sigma) - L(x, y, k_j \sigma) \quad (3)$$

where $L(x, y, k\sigma)$ is the convolution of the original image $I(x, y)$ with the Gaussian blur $G(x, y, k\sigma)$ at scale $k\sigma$, i.e., $L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y)$.

- Location of the keypoints: key points are chosen on the basis of stability measures, removed the key points with low.

- Contrast or are located in the margins-orientation assignment: invariance to rotation is achieved by assigning to each of the points of orientation based on the local properties of the image and which represents the descriptor for this orientation.
- Descriptor keypoints: the local gradients of the image are measured in the region surrounding the key point. These are processed by means of a representation which allows distortion levels and changes in lighting locally.

1.2.3.2 SURF Algorithm:

- Detection of points of interest, keypoints.
- Assignment guidance.
- Extraction of those described. More information in the sections of features.

1.2.3.3 Differences between algorithms

The main differences between the algorithms SIFT and SURF are the speed of implementation of respect for one another, but with both points of interest of the image keep invariants, scale and orientation, and changes in the lighting. The main differences of data in multiple experiments are that SIFT is kept the position, scale and orientation, since it is possible that in a same position (x, and), we find several points of interest to different scale and / or orientation σ . But another side in the SURF in a position (x, and) appears only a single point of interest, by what is not saved the scale and orientation, however if that registers the matrix of second order and the sign of the Laplacian.

1.2.4 Opponent SIFT

The opponent SIFT descriptors is an algorithm which use the same method than the classical SIFT descriptor. The only difference is that we calculate three descriptors for each Key-points. There are obtained from the color opponent channel, defined as

$$O_1 = \frac{R - G}{\sqrt{2}}, O_2 = \frac{R + G - 2B}{\sqrt{6}}, O_3 = \frac{R + G + B}{\sqrt{3}}. \quad (4)$$

The opponent SIFT describes the three opponent color spaces, we can see that the two first channels O_1, O_2 contain some intensity information, but they are not invariant to changes of lights intensity. The last channel will contain the intensity information.

The strength of these method is that that use a color spaces, and we can see directly information of that with the algorithm of the SIFT descriptor. The weakness is that, in this approach we use the RGB space to make this computation.

1.2.5 C₂O

C2O feature is a color descriptor which aim is to characterize an image by its color and texture characteristics. Indeed, the currents descriptors previously presented are some satisfying solutions to characterize current images in gray levels or in color levels but pretty

weak for highly textured images like nature images. So the university of Poitiers has worked on a descriptor named C2O (Color Contrast Occurence) which is based on a vector that include the texture and color information separately. To compute it, there is two steps to respect : the calculation of the Color Contrast Occurence Matrix and the feature (descriptor) from the matrix.

1.2.5.1 The Color Contrast Occurence Matrix

To compute this descriptor, the aim is to calculate a matrix which represents each keypoint by a probability. This probability represents (a revoir). To compute it, the image has to be used in a color space which is able to separate best the color and the luminance information. Tests have shown that the CIE $L^* a^* b^*$ space separate "has minimum correlation between luminance and chrominance information" ("Color Contrast Occurence, a full vector for color and texture"). So the image is passed in the CIE $L^* a^* b^*$ color space before the calculation of the descriptor. Before the calculation of the $L^* a^* b^*$ space, its needed to transform our image through the XYZ space that is a perceptual space based on a linear transformation of the RGB space.

$$A = \begin{pmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{pmatrix}$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = A * \begin{pmatrix} R \\ G \\ B \end{pmatrix} \quad (5)$$

With A a matrix which coefficients are dependind on the choosen standard illimuninant. When this XYZ space has been computed, we have to compute the following transformation to get our image in the $L^* a^* b^*$ space.

$$L^* = \begin{cases} 116 * (\frac{Y}{Y_0})^{\frac{1}{3}} - 16 & si \frac{Y}{Y_0} > 0.008856 \\ 903.3 * (\frac{Y}{Y_0}) & si \frac{Y}{Y_0} < 0.008856 \end{cases} \quad (6)$$

$$a^* = 500 * [f(\frac{X}{X_0}) - f(\frac{Y}{Y_0})] \quad (7)$$

$$b^* = 300 * [f(\frac{Y}{Y_0}) - f(\frac{Z}{Z_0})] \quad (8)$$

After that, we can calculate the descriptor. The principle is simple : for each keypoint, we have to calculate the probability to have a specific color difference between two pixels separated by a spatial vector (a voir si le dit vecteur est vecteur type). The color difference is

calculate by considering the angles created by the $L^* a^* b^*$ representation and a perceptual distance (probablement sur la luminance mais a vrifier).

(insrer ET COMPRENDRE les formules en syntaxe LaTeX)

So we define the color contrast occurence value as : $\overrightarrow{\Lambda(C_i, C_j)}$

$$\overrightarrow{\Lambda(C_i, C_j)} : prob(\overrightarrow{\Lambda(C_i, C_j)} = \overrightarrow{\Lambda_\chi}) \quad (9)$$

$$with \|\overrightarrow{\Lambda(C_i, C_j)}\| = \Delta E_\chi \quad (10)$$

$$and (\overrightarrow{Oa}, \overrightarrow{c_i c_j}) = (\alpha, \beta) = \overrightarrow{\Lambda_\chi} \quad (11)$$

This computation gives us a cloud of point which characterize the keypoint by its color and texture neighborhood (see below).

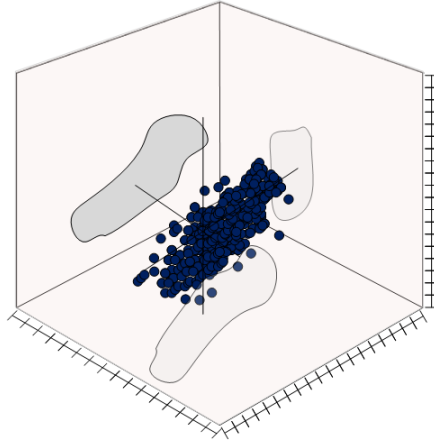


Figure 2: Color Contrast Occurence Matrix

On the figure shown above, we can see an example of the cloud of points that we expect to obtain. There to thing which characterize the image :

- the size and the form of the cloud which characterize the texture around the keypoint.
- the projections on the three plans of the representation which characterize the color information around the keypoint.

1.2.5.2 The Color Constrast Occurence feature

With the cloud of point obtained by the computation of the Color Contrast Occurence matrix, we have a 3 dimensional representation of our keypoint. To reduce the quantity of data to store and to facilitate the distance calculation, we have to represent this matrix by at least a 2-dimensional feature. To do that, the solution used is to realize a spherical quantization on the cloud of point to have a histogram which will represent our keypoint on two dimensions. Mathematically, this quantization is expressed as follows :

$$Sig_{C_2O}(I) = h_{\Delta_{i\alpha j \beta k}} = prob(\Delta_i \leq \|\overrightarrow{\Lambda(C_i, C_j)}\| < \Delta_j + \Delta E_{step}) \quad (12)$$

$$\text{and } \frac{2\pi}{n_\alpha}j \leq \alpha < \frac{2\pi}{n_\alpha}(j+1) \quad (13)$$

$$\text{and } 0 \leq \beta < \frac{2\pi}{n_\beta}(k) \quad (14)$$

Each sphere will include a number of points of the cloud, but to have a better distribution, each sphere will be split in some part as shown below :

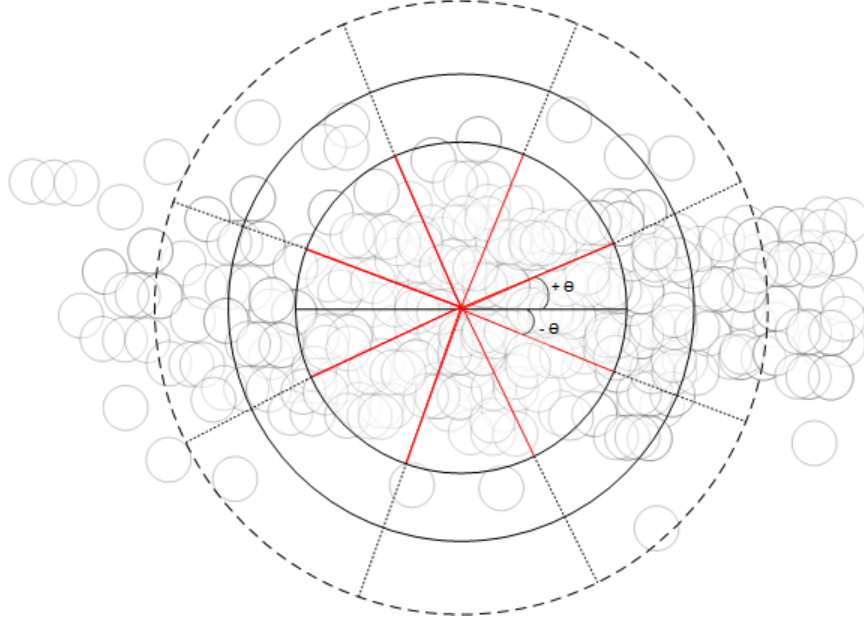


Figure 3: Spheric quantization

Here we can see a sectionnal view of our spherical quantization. Each sphere is divided by n parts as show above, and the number of points in each part are concatenates one by one in the description vector (quarter after quarter and sphere after sphere).