# 1 SIFT(scale-invariant feature transform)

SIFT is an algorithm used in sector of computer vision for detection and identification of similar elements between different numeric images. The principal method proposed by the author David Lowe is to calculate the SIFT descriptors on images studied. These descriptors are numeric information which derived of local analysis of the image and they characterize the visual content of this image so that this one is independent of the scale, the framing, the angle of observation and the luminosity.

## 1.1 Stages of the SIFT algorithm

The first stage is to detect key-points on the image. Each key-point is characterize by coordonates x,y on the image and a scale factor $\sigma$. After that it's necessary to ameliorate the precision of the key-point localisation.

- Scale-space extrema detection
  The gradient scale factor is calculated for smooth the original image. This operation allow to delete details which radius are inferior at $\sigma$.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{1}$$

Figure 1: Illustration of scale factor gradient (octave 7x7;$\sigma_0$ =0; $\sigma_1$ =1.6; $\sigma_2$ =0.4)

After that, the key-points which have dimension approximately equal to $\sigma$ are detected by using the Difference of Gaussian(DoG) given by the following equation.

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, k\sigma) \tag{2}$$

Where k is the fixed parameter of the algorithm and depend to the fineness of the scale desired.

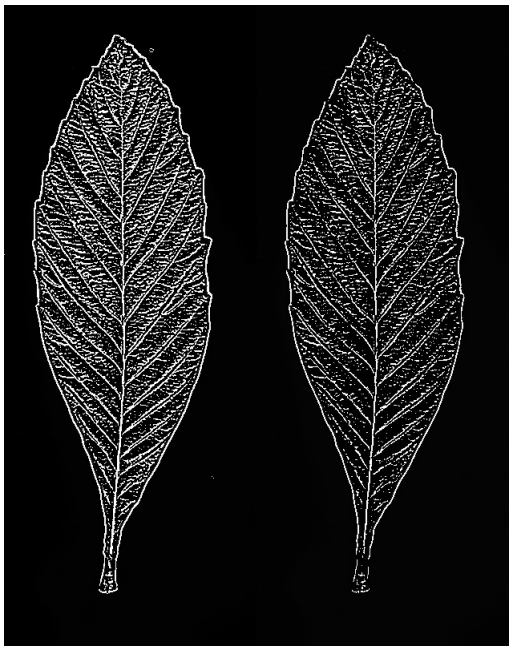


Figure 2: Illustration of the DoG

- Orientation assignment
  On the base of local image gradient detections each key-point detected is assigned to one or many orientations.

  With the symmetric finite difference, the gradient and the amplitude are calculated for each position around the key-point. The calculation of these two factors is given by the following relations:

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

$$\theta(x,y) = atan2(L(x,y+1) - L(x,y-1), L(x+1,y) - L(x-1,y))$$

  A histogram with 36 intervals is realized on the vicinity and each interval covering an angle of 10 degrees.

  The peaks in this histogram correspond to the dominant orientations. All dominant orientations allow to have at least 80% of the maximum value are taking in consideration and other additional key-points are detected. These new key-points detected are only different by the principal orientation.

- key-point descriptor
  An area of 16x16 pixels is taken around the key-point; subdivide in 4x4 zones of 4x4 pixels each one. A histogram including 8 intervals is calculated on each zone.

  After that, the 16 histograms with 8 intervals each one, are concatenated and normalized. In object to reduce the sensibility of the descriptor to the changes of the luminosity, the values are fixed to 0.2 and the histogram is calculated again for finally give the descriptor of the key-point of 128 dimension.

# 2 SIFT test and results

In our project we used the SIFT of Opencv library in python. This function allow to detect key-points on images and compute descriptors for each one. The script is adapted to our program in order to caculate descriptors in whole database of images. Before integreting the script in the process flow, test steps were done in order to verify if the results obtained with the program are satisfied.

Tests consist to apply the sift on image according to different configurations to notice the invariance of the function to the configurations. In a first phase, the tests were done on a basic image design on paint. Here we have the illustration images of the results

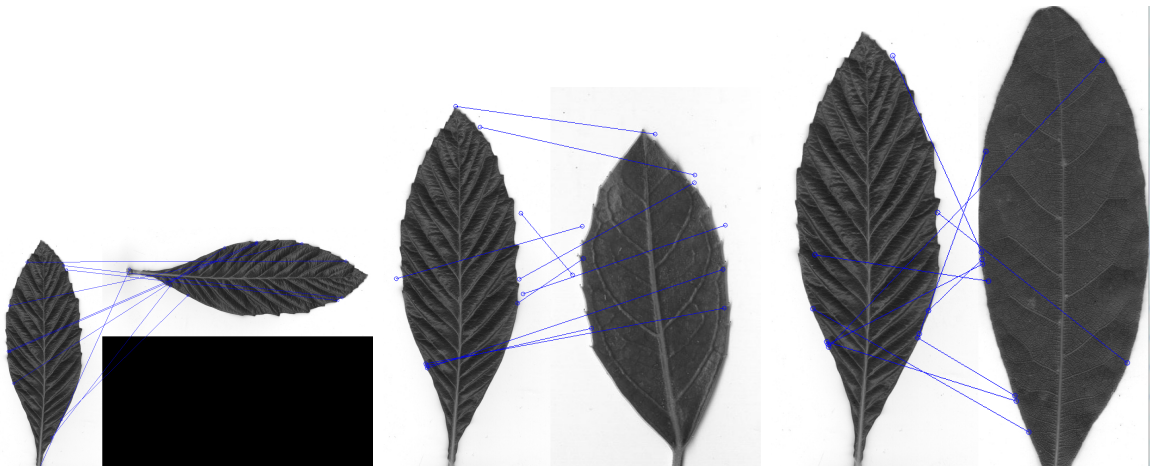
(a) Key-points detected

(b) Key-points detected after luminosity change

Figure 3: Illustration 1

Figure 4: Key-points detected after rotation

After that, the test was also done on three images of the database. Three images which look alike are taken in the database and key-points are calculated on each one. These images are matched two by two with the key-points in order to find similarity between images.



(a) Illustration 1 (b) Illustration 3 (c) Illustration 3

According to our relsults obtained through the images above, we concluded that the SIFT descriptor is satisfied and it can be used to compute all descriptors in whole the database.

# 3 Parallelism test

This test consist to use multiple processors to make a computation faster. It will permit multiple tasks to proceed without waiting for each other. The algorithme consist to decompose the complete task into independent subtasks and the data flow between them and distribut the subtasks over the processors. In python we have different posibilities to do this. The test was done by using the module multiprocessing because it was the only avaible in our python version. We can see on the following images the time computation before and after using the parallelism.

```
>>> runfile('C:/Users/RIMA/Desktop/Nouveau dossier/main.py', wdir=r'C
MA/Desktop/Nouveau dossier')
Reloaded modules: function_sift
0.00499987602234
```

(d) Illustration with parallelism

```
>>> runfile('C:/Users/RIMA/Desktop/Nouveau dossier/main.py', wdir=r'C:/Users
MA/Desktop/Nouveau dossier')
Reloaded modules: function_sift
6.16100001335
>>>
```

(e) Illustration without parallelism