**Speeded Up Robust Features(SURF)**

The Speeded Up Robust Features method a.k.a the SURF method is a remarkable research paper authored by Herbert Bay, Andreas Ess, Tinne Tuytelaars and Luc Van Gool. It has been developed into an algorithm that is being used in many computer vision applications like object detection, object classification, 3-D construction, object tracking etc. This is a lightning-fast and robust algorithm, also used for real-time object detection. It is described to be a ‘local feature detector and descriptor ’.

A ‘local feature’ is a pattern or structure in an image that distinguishes it from every other image or combination of images in its immediate surrounding, such as a point, an edge or a small image patch. They can be used to detect image patches that are different in colour, texture or intensity. For example, an algorithm to detect all corners in an image. Local features also facilitate image correspondence (finding points in an image and identifying them as the same points in the same image taken from a different point of view) regardless of occlusion or presence of clutter. These properties make them suitable for image classification and feature extraction- a method used to select features that best define the image patch or interest point.

The detector-descriptor scheme is coined SURF (Speeded-Up Robust Features). The detector technology is based on the Hessian matrix(a second-order derivatives matrix of a particular pixel) but uses a very basic approximation. It is basically working on the integral images approach to reduce the computation time and therefore it is called the ‘Fast-Hessian’ detector which is a key component of SURF.

**Feature Detector**

‘Feature detectors’ have the ability to recognize and detect shapes, angles or other unique qualities (eg: corners) to find required points in an image. Different detectors can be used corresponding to our data. For example, if the image contains an image of cancer or bacteria cells, the blob detector is used, instead of the corner detector. Point Detectors The most widely used detector probably is the Harris corner detector is used for feature detection in making SURF model

**What is a feature descriptor?**

A feature descriptor can be thought of as an algorithm that encodes the features or information about an image; such that every feature can be differentiated from another, and outputs a ‘feature vector’. Example- Histogram of Oriented Gradients or HOG.

Feature extraction is one step in the SURF process. And for this, a very basic Hessian-matrix technique is used.

**Integral Images**

The Integral Image method is used as a fast and efficient way of calculating the sum of pixel values of the image in question if we visualize the image as a grid with each pixel value filling the grid, with each pixel containing the cumulative sum of the corresponding input pixel with every pixel above and every pixel left of the input pixel. This technique is mainly used for getting an idea about the average intensity enclosed within an image.

For calculating integral of an image at pixel(x,y):

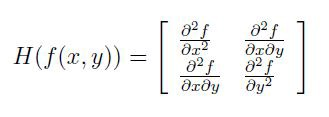
**s(x,y)= i(x,y)** [current pixel] **+ s(x-1,y)** [pixel to the immediate left] **+ s(x,y-1)** [pixel immediately above] **- s(x-1,y-1)**



**Hessian matrix-based points detector**

The blob detector in SURF relies on the Hessian matrix for locating or detecting the points of interest. The Hessian matrix is being used as a detector in SURF because of its excellent performance in computation time and accuracy. However, rather than using a different measure for selecting the location and the scale (Hessian-Laplace detector), surf relies on the **determinant of the Hessian matrix** for both.

**Hessian matrix:-**

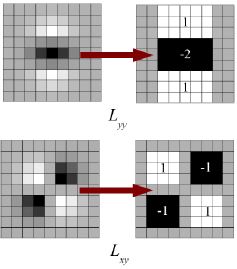
****

For selecting the scale, SURF uses the determinant of the Hessian matrix. Considering a point p=(x,y) in an image I, if σ is used to represent the scale, then the Hessian matrix H(p, σ) at p is:

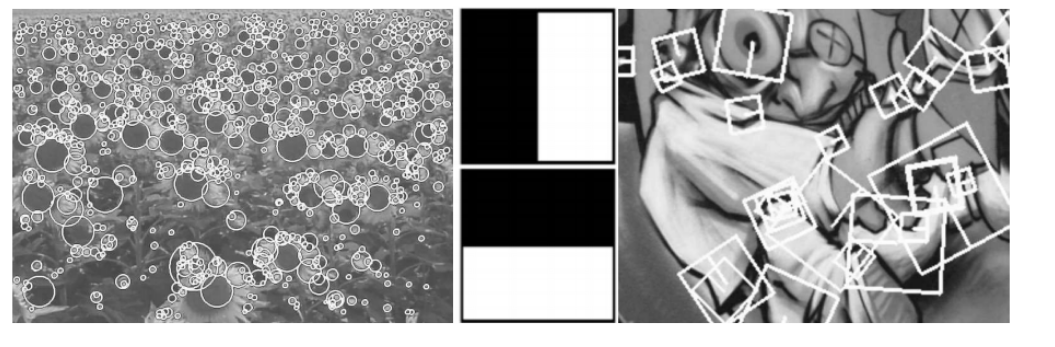


Lxx(p, σ) is the convolution of the Gaussian second-order derivative ∂2 ∂x2 g(σ) with the image I in point x, and similarly for Lxy(p, σ) and Lyy(p, σ).

Gaussians are optimal for scale-space analysis



Gaussian second-order partial derivatives in y-direction and XY-direction, and the approximations thereof using box filters. The grey regions are equal to zero. Gaussian second-order derivatives with σ = 1.2



Detected interest points for a Sunflower field. This image clearly illustrates feature detection using Hessian-based detectors. The Graffiti image depicts the different sizes of the descriptor window at various scales.

**Scale-space representation**

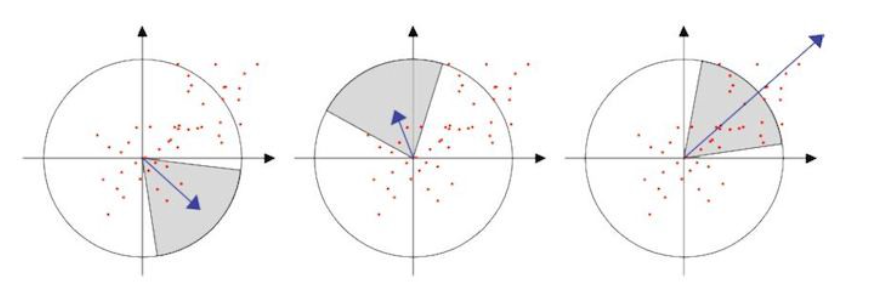
An image should be processed such that correspondence is achieved and the interest points can be found even at different scales or viewing angles.

A scale-space is a theoretical way of handling and representing image structure at different scales (or ratios). They are usually implemented as image pyramids, with the base of the pyramid representing a larger scale. By using a Gaussian filter, images are smoothed repetitively by up-scaling the filter at a constant rate, and at each stage, the image acts as a sample for developing the higher level of the pyramid.

**Feature description**

The SURF descriptive is created step-wise.

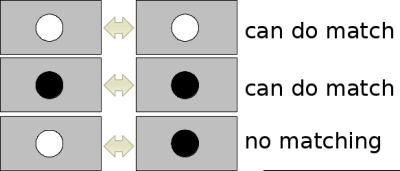
Firstly, a circular region around the key points or points of interest is described. A reproducible orientation based on this circular region is fixed. The square region in alignment with the orientation is constructed and the SURF descriptor is extracted from it.



SURF tries to achieve invariance to rotation. For doing so, Haar-wavelet (square-shaped wavelets representing a constant function) responses are calculated in x and y-direction. If ‘s’ is the scale at which the keypoint was detected, these responses are calculated in a circular neighbourhood of radius 6s.

Finally, the sum of the horizontal and vertical responses in the scanning area is calculated; the scanning orientation is changed, and the sum is re-calculated until the largest sum value is obtained. This particular orientation, with the largest value of the sum, is the main orientation of the feature descriptor.

The sign of Laplacian (trace of Hessian Matrix) was used for improving SURF that helps in finding the underlying interest points. With the addition of the sign of laplacian, no computation cost is increased since it is already computed during detection. The sign of the Laplacian distinguishes between the bright blobs on dark backgrounds even if the image is in the reverse situation. In the matching stage, only features are compared whether they have the same type of contrast. Using this feature helps in faster matching without reducing the performance of the descriptor.



**Implementing surf using OpenCV library on python**

**import cv2 as cv**

**import matplotlib.pyplot as plt**

**import NumPykey point as np**

**img= cv.imread("cat/cat.png")**

**# Create SURF object.**

**# Here set Hessian Threshold to 500**

**surf = cv.xfeatures2d.SURF\_create(500)**

**# Find keypoints and descriptors directly**

**kp, des = surf.detectAndCompute(img,None)**

**# Hessian threshold**

**surf.setHessianThreshold(5000)**

**# Again compute keypoints and check its number.**

**kp, des = surf.detectAndCompute(img,None)**

**img2 = cv.drawKeypoints(img,kp,None,(255,0,0),4)**

**cv.imshow("cat",img2)**

**#U-SURF, so that it won't find the orientation**

**surf.setUpright(True)**

**# Recompute the feature points and draw it**

**kp = surf.detect(img,None)**

**img3 = cv.drawKeypoints(img,kp,None,(255,0,0),4)**

**cv.imshow("cat",img3)**

**#The descriptor size and change it to 128 if it is only 64-dim.**

**surf.getExtended()**

**# So we make it to True to get 128-dim descriptors.**

**surf.setExtended(True)**

**kp, des = surf.detectAndCompute(img,None)**

**print( surf.descriptorSize() )**

**# Convert image to RGB**

**training\_img = cvtest\_gray = cv.cvtColor(img, cvtest\_gray = cv.COLOR\_BGR2RGB)**

**# Convert image to gray scale**

**training\_gray = cvtest\_gray = cv.cvtColor(training\_img, cvtest\_gray = cv.COLOR\_RGB2GRAY)**

**# Create test image**

**test\_img = cvtest\_gray = cv.pyrDown(training\_img)**

**test\_img = cvtest\_gray = cv.pyrDown(test\_img)**

**rows, cols = test\_img.shape[:2]**

**rotation\_matrix = cvtest\_gray = cv.getRotationMatrix2D((cols/2, rows/2), 30, 1)**

**test\_img = cvtest\_gray = cv.warpAffine(test\_img, rotation\_matrix, (cols, rows))**

**test\_gray = cv.cvtColor(test\_img, cvtest\_gray = cv.COLOR\_RGB2GRAY)**

**cv.imshow("training image", training\_img)**

**cv.imshow("test image", test\_img)**

**surf = cv.xfeatures2d.SURF\_create(400)**

**train\_kp, train\_descriptor = surf.detectAndCompute(training\_gray, None)**

**test\_kp, test\_descriptor = surf.detectAndCompute(test\_gray, None)**

**kp\_without\_size = np.copy(training\_img)**

**kp\_with\_size = np.copy(training\_img)**

**cv.drawKeypoints(training\_img, train\_kp, kp\_without\_size, color = (0, 255, 0))**

**cv.drawKeypoints(training\_img, train\_kp, kp\_with\_size, flags = cv.DRAW\_MATCHES\_FLAGS\_DRAW\_RICH\_KEYPOINTS)**

**cv.imshow("Train keypoints With Size",kp\_with\_size, cmap='gray')**

**cv.imshow("Train keypoints Without Size",kp\_without\_size, cmap='gray')**

**# Create a Brute Force Matcher object.**

**bf = cv.BFMatcher(cv.NORM\_L1, crossCheck = False)**

**matches = bf.match(train\_descriptor, test\_descriptor)**

**matches = sorted(matches, key = lambda x : x.distance)**

**final\_img = cv.drawMatches(training\_img, train\_kp, test\_gray, test\_kp, matches, test\_gray, flags = 2)**

**cv.imshow("cat",final\_img)**

**Conclusion:-**

In short, SURF adds a lot of features to improve the speed in every step. Analysis shows it is 3 times faster than SIFT while performance is comparable to SIFT. SURF is good at handling images with blurring and rotation, but not good at handling viewpoint change and illumination change.

**GitHub link for code reference:-**

[**https://github.com/DhruvBajaj01/SURF**](https://github.com/DhruvBajaj01/SURF)

**References:-**

* [**https://docs.opencv.org/4.5.2/index.html**](https://docs.opencv.org/4.5.2/index.html)
* [**https://www.sciencedirect.com/science/article/abs/pii/S1077314207001555**](https://www.sciencedirect.com/science/article/abs/pii/S1077314207001555)
* [**https://docs.opencv.org/4.5.2/df/dd2/tutorial\_py\_surf\_intro.html**](https://docs.opencv.org/4.5.2/df/dd2/tutorial_py_surf_intro.html)
* [**https://people.ee.ethz.ch/~surf/eccv06.pdf**](https://people.ee.ethz.ch/~surf/eccv06.pdf)