CV SECTIONS 6-7

- An interest point (key point, salient point) detector is an algorithm that chooses points from an image based on some criterion. Typically, an interest point is a local maximum of some function, such as a "cornerness" = (R) metric.

- A descriptor is a vector of values, which somehow describes the image patch around an interest point. It could be as simple as the raw pixel values, or it could be more complicated, such as a histogram of gradient orientations.

- Together an interest point and its descriptor is usually called a local feature.

  Local features are used for many computer vision tasks, such as image registration, 3D reconstruction, object detection, and object recognition

- Harris, Min Eigen, and FAST are
  interest point detectors, or more specifically, corner detectors.

- SIFT includes both a detector and a descriptor.
  The detector is based on the difference-ofGaussians (DoG), which is an approximation of the Laplacian.
  The DoG detector detects centers of blob-like structures.
  The SIFT descriptor is a based on a histogram of gradient orientations.

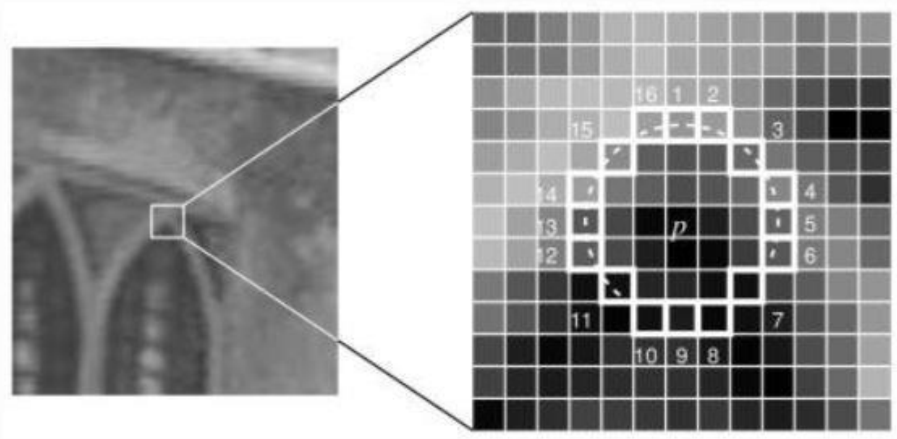
- SURF is meant to be a fast approximation of SIFT.

- BRISK, like SIFT and SURF, includes a detector and a descriptor.

  The detector is a corner detector.
  The descriptor is a binary string representing the signs of the difference between certain pairs of pixels around the interest point.

FAST ALGORITHM

1. Select a pixel p in the image which is to be identified as an interest point or not. Let its intensity be I_p.
2. Select appropriate threshold value t.
3. Consider a circle of 16 pixels around the pixel under test. (See the image below)



P is corner if BRIGHTER THAN Intensity of point + threshold (Ip + t) or DARKER THAN (ip - t)

IN MACHINE LEARINING PRESPECTIVE

- Select set of images to train
- Run FAST Algorithm
- For every feature point store 16-pixel around It as vector
- Do it for all images until get feature vector "P"
- EACH PIXEL HAVE ONE OF 3 STATES: DARK , Simillar , Bright
- Depend on these states → feature vector sub divided into 3 subset p(dark) , p(simillar) , p(Bright)
- Kp → boolean variable define that p is corner or not
- Do this for all images untill → Entropy = 0
- MAKE DECISION TREE

Non Maximum suppression

Detecting multiple interest points in adjacent locations is another problem. It is solved by using Non-maximum Suppression.

1. Compute a score function, V for all the detected feature points. V is the sum of absolute difference between p and 16 surrounding pixels values.
2. Consider two adjacent keypoints and compute their V values.
3. Discard the one with lower V value.

FAST ALGORITM CODE

// Initiate FAST object with default values

fast = cv.FastFeatureDetector_create()

// find the key point

kp = fast.detect(img, None) // find

img2 = cv.drawKeypoints(img, kp, None, color=(255, 0, 0)) //draw   // 255,0,0 → BLUE

// drawKeypoint (img , keypoint variable , NONE , BGR COLOR)

// get all parameters of FAST as Threshold , Non-Maximum Suppression , neighbourhood

- fast.getThreshold()   // return threeshold
- fast.getNonmaxSuppression()   // return Non maximum suppression (Boolean)
- fast.getType()   // return Neighbourhood

// Total Key Points without maximum suppression

len(kp) // length of key point variable

// return image with non maximum suppression

NewImage = (“New.jpg”, img2)

// return Image without non-maximum Suppression

fast.setNonmaxSuppression(0)   // more corners

kp = fast.detect(img, None)

SIFT (SCLAE INVARIANT Feature Transform)   // uses LoG

Harris → rotation invariant // لا يتأثر بال روتيشن   Harris → scale variant // يتأثر بالتكبير و التصغير

⇨ SIFT IS FOUNDED FOR THIS PROBLEM
  - Scale space extrema detection
  - Key point localization
  - Orientation assignment
  - Key point descriptor // 16x16 → 4x4
  - Key point matching

SIFT CODE

```
 gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

// CREATE SIFT OBJECT
 sift = cv.SIFT_create()

// Get Key Point
 kp = sift.detect(gray, None)

// Draw KeyPoint
 img1 = cv.drawKeypoints(gray, kp, None)

علي حسب  حجم الفيتشر ارسم دايرة اكبر
img2 = cv.drawKeypoints(gray, kp, None,
flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)


// return keypoint and descriptor
Kp , desc = Sift.detectAndCompute(img,None)
// return an array its size  = number of Keypoint X 128
```

SURF ALGORITHM SPEEDED UP ROBUST FEATURES // uses box filter + integral images

It is a speeded-up version of SIFT.
SURF goes a little further and approximates LoG with Box Filter.
Convolution with box filter can be easily calculated with the help of integral images.
And it can be done in parallel for different scales.

SURF CODE

Create surf object : surf= cv2.xfeatures2d.SURF_create(threshold)
Find keypoints using surf: kp= surf.detect(img)
Compute the descriptors with surf : des= surf.compute(img, kp)
Draw only keypoints : cv2.drawKeypoints(img, kp, None, color)
Number of features (KEY POINTS) = len(kp)
Number of discriptors = surf. descriptorSize()

To Convert image into grayscale

img = cv2.imread('img path', cv2.IMREAD_GRAYSCALE)   or   img = cv2.imread("img path",0)

BRIEF (Binary Robust Independent Elementary Features)

- BRIEF is a faster method feature descriptor calculation and matching. It also provides high recognition rate unless there is large in-plane rotation.
- One important point is that BRIEF is a feature descriptor, it doesn't provide any method to find the features.
- So you will have to use any other feature detectors like SIFT, SURF etc.

BRIEF CODE

ϒ Create STAR detector : star = cv2.xfeatures2d.StarDetector_create()
ϒ find the keypoints : kp = star.detect(img,None)
ϒ Initiate BRIEF extractor : brief = cv2.xfeatures2d.BriefDescriptorExtractor_create()
ϒ compute the descriptors with BRIEF : kp, des = brief.compute(img, kp)
ϒ Draw only keypoints : cv2.drawKeypoints(img,kp,None)

ORB (Oriented FAST and Rotated BRIEF)

- **ORB: An efficient alternative to SIFT or SURF** .
- it is a good alternative to SIFT and SURF in computation cost, matching performance and mainly the patents.
- ORB is basically a fusion of FAST keypoint detector and BRIEF descriptor with many modifications to enhance the performance.

ORB CODE

-Create ORB object : orb= cv2. ORB_create(max_num_feature) #500
- Find keypoints using orb: kp= orb.detect(img)
- Compute the descriptors with orb : des= orb.compute(img, kp)
-Draw only keypoints : cv2.drawKeypoints(img, kp, None, color)