Feature Detection and Matching

Feature detection and matching is an important task in many computer vision applications, such as structure-from-motion, image retrieval, object detection, and more. In this series, we will be talking about local feature detection and matching.

Introduction:

1. Feature.
2. Component of Feature Detection And Matching.
3. Applications of Feature Detection And Matching.
4. Techniques of Feature Detection And Matching.

1. Feature.

A feature is a piece of information which is relevant for solving the computation task related to a certain application. Features may be specific structures in the image such as points, edges or objects. Features may also be the result of a general neighborhood operation or feature detection applied to the image. The features can be classified into two main categories:

- The features that are in specific locations of the images.
- The features that can be matched based on their orientation and local appearance are called edges.

2. Component of Feature Detection And Matching.

- Detection: Identify the Interest Point.
- Description: The local appearance around each feature point is described in some way that is invariant under changes in illumination, translation, scale, and in-plane rotation.
- Matching: Descriptors are compared across the images, to identify similar features.

3. Applications of Feature Detection And Matching.

- Automate object tracking.
- Point matching for computing disparity.
- Stereo calibration (Estimation of the fundamental matrix).
- Motion-based segmentation.
- Recognition.
- 3D object reconstruction.
- Robot navigation.
- Image retrieval and indexing.

4. Techniques of Feature Detection And Matching.

Harris corner detector:
Harris Corner Detector is a corner detection operator that is commonly used in computer vision algorithms to extract corners and infer features of an image. It was first introduced by Chris Harris and Mike Stephens in 1988 upon the improvement of Moravec's corner detector. we take the whole squared distinction (SSD) of the pixel esteems when the move and recognizing pixel windows where the SSD is enormous for shifts in each of the 8 directions. The features in the image are all pixels that have large values of E(u, v), as defined by some threshold.

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u,y+v) - I(x,y)]^2$$

Applying Taylor Expansion :

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \left(\sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix}$$

Let M be the summed-matrix:

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

We have score R:
with $\lambda 1$ and $\lambda 2$ are the eigenvalues of M.

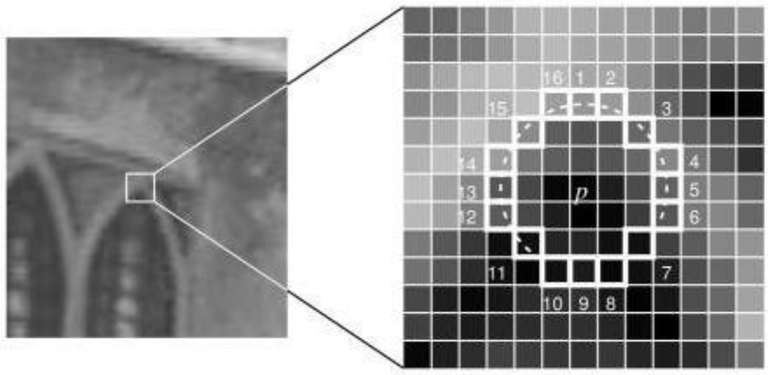
$$R = \det M - k(trace\, M)^2$$

$$\det M = \lambda_1 \lambda_2$$

$$trace\, M = \lambda_1 + \lambda_2$$

- When |R| is small, which happens when λ1 and λ2 are small, the region is flat.
- When R<0, which happens when λ1>>λ2 or vice versa, the region is an edge.
- When R is large, which happens when λ1 and λ2 are large and λ1~λ2, the region is a corner.

FAST (Features from Accelerated Segment Test):
The FAST corner detector was originally developed by Edward Rosten and Tom Drummond and was published in 2006. Features from accelerated segment test (FAST) is a corner detection method, which could be used to extract feature points and later used to track and map objects in many computer vision tasks. The FAST corner detector is very suitable for real-time video processing application because of this high-speed performance.



12 point segment test corner detection in an image patch. The highlighted squares are the pixels used in the corner detection. The pixel at p is the center of a candidate corner. The arc is indicated by the dashed line passes through 12 contiguous pixels which are brighter than p by more than the threshold.

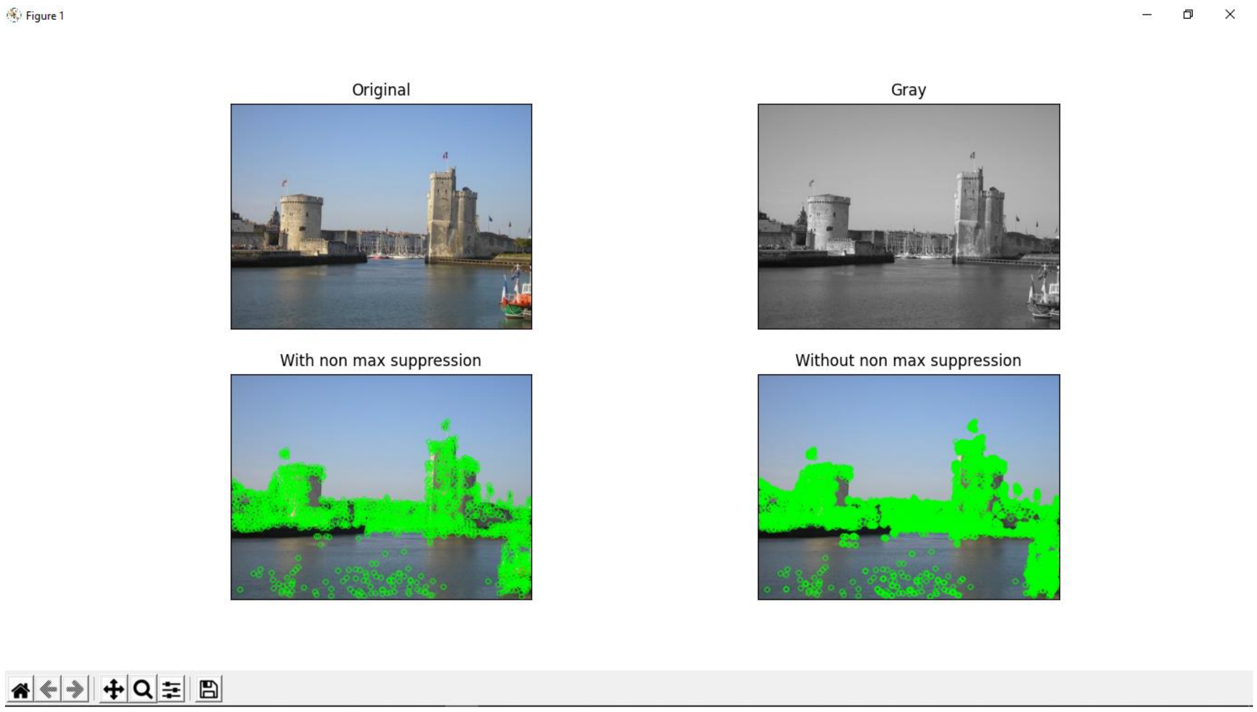
- Select a pixel p in the image which is to be identified as an interest point or not. Let its intensity be Ip.
- Select appropriate threshold value t.
- Consider a circle of 16 pixels around the pixel under test. (This is a Bresenham circle of radius 3.)
- Now the pixel p is a corner if there exists a set of n contiguous pixels in the circle (of 16 pixels) which are all brighter than Ip + t, or all darker than Ip - t. (The authors have used n= 12 in the first version of the algorithm)
- To make the algorithm fast, first compare the intensity of pixels 1, 5, 9 and 13 of the circle with Ip. As evident from the figure above, at least three of these four pixels should satisfy the threshold criterion so that the interest point will exist.
- If at least three of the four-pixel values — I1, I5, I9, I13 are not above or below Ip + t, then p is not an interest point (corner). In this case reject the pixel p as a possible interest point. Else if at least three of the pixels are above or below Ip + t, then check for all 16 pixels and check if 12 contiguous pixels fall in the criterion.

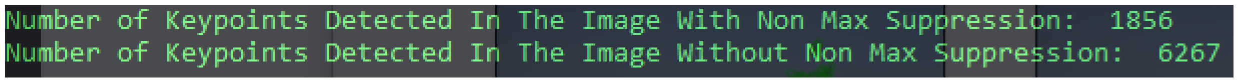
- Repeat the procedure for all the pixels in the image.

A few limitations: or ***n***<12, the algorithm does not work very well, the order in which the 16 pixels are queried determines the speed of the algorithm.

For this example, we will use the FAST algorithm to detect the features:

```python
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np

#Load file
img = cv.imread('LaRochelle.jpg')

#Convert image to RGB
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)

#Convert image to gray scale
gray = cv.cvtColor(img, cv.COLOR_RGB2GRAY)

fast = cv.FastFeatureDetector_create()

#Detect keypoints with non max suppression
keypoints_with_nonmax = fast.detect(gray, None)

#Disable nonmaxSuppression
fast.setNonmaxSuppression(False)

#Detect keypoints without non max suppression
keypoints_without_nonmax = fast.detect(gray, None)

image_with_nonmax = np.copy(img)
image_without_nonmax = np.copy(img)

#Draw keypoints on top of the input image
cv.drawKeypoints(img, keypoints_with_nonmax, image_with_nonmax, color=(0, 255, 0), flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv.drawKeypoints(img, keypoints_without_nonmax, image_without_nonmax, color=(0, 255, 0), flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

#Print the number of keypoints detected in the training image
print("Number of Keypoints Detected In The Image With Non Max Suppression: ", len(keypoints_with_nonmax))

#Print the number of keypoints detected in the query image
print("Number of Keypoints Detected In The Image Without Non Max Suppression: ", len(keypoints_without_nonmax))

#Display images
titles = ['Original', 'Gray', 'With non max suppression', 'Without non max suppression']
images = [img, gray, image_with_nonmax, image_without_nonmax]

for i in range(4):
    plt.subplot(2, 2, i+1), plt.imshow(images[i], 'gray')
    plt.title(titles[i])
    plt.xticks([]), plt.yticks([])

plt.show()
```





```
Number of Keypoints Detected In The Image With Non Max Suppression:  1856
Number of Keypoints Detected In The Image Without Non Max Suppression:  6267
```