Features and Image 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.

Objective

- What is feature
- Applications of features in image matching
- Techniques

1. What is feature

A feature is a piece of information which is relevant for solving the computational 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 several categories:

- **Edge**: Edges are points where there is a boundary (or an edge) between two image regions. In general, an edge can be of almost arbitrary shape, and may include junctions. In practice, edges are usually defined as sets of points in the image which have a strong gradient magnitude
- **Interest points**: point which is expressive in texture. Interest point is the point at which the direction of the boundary of the object changes abruptly or intersection point between two or more edge segments.
- **Blob**: Blobs provide a complementary description of image structures in terms of regions, as opposed to corners that are more point-like

2. Applications of features

- Automate object tracking
- Point matching for computing disparity
- Stereo calibration(Estimation of the fundamental matrix)

- Motion-based segmentation
- Recognition
- 3D object reconstruction
- Robot navigation

3. Techniques

Harris corner detector

Harris corner detector is an algorithm designed by Chris Harris and Mike Stephens. The idea of the algorithm is rely on the transformation of a region. A region around intertest point will have significant changes in intensity when the window is moving for a distance (u,v) from (x,y) in any direction. We take the sum squared difference (SSD) of the pixel values before and after the shift and identifying pixel windows where the SSD is large for shifts in all 8 directions. The sum of all the sum squared differences (SSD) can be defined as a function:

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

Applying Taylor series expansion to the above equation, we get:

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

Let M be the summed-matrix, the equation now becomes:

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

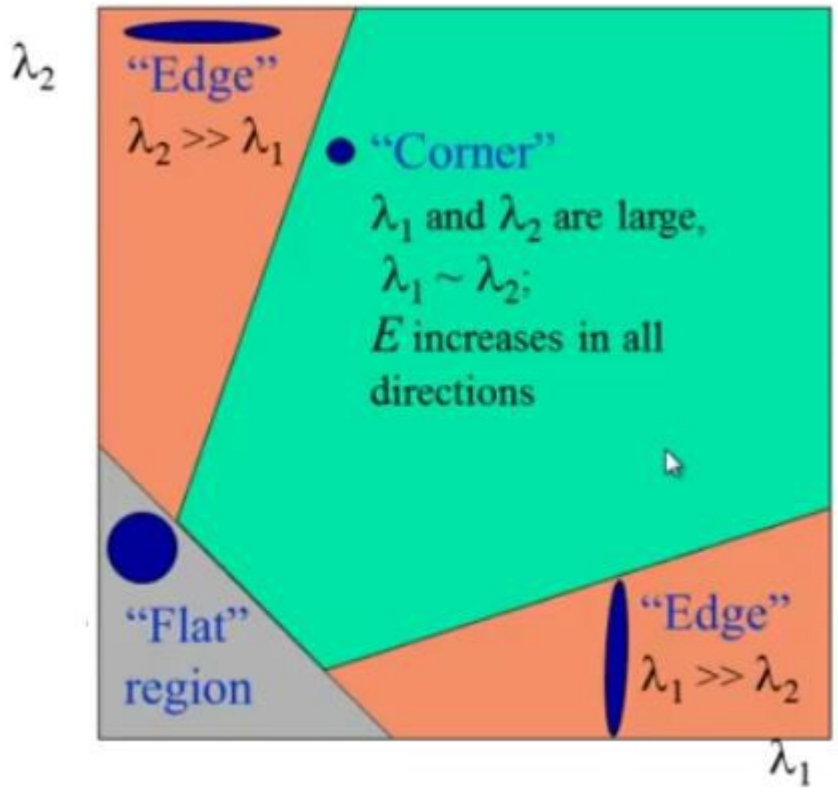
The job now is to classify and find out all the pixels whether it's a corner of not based on the value R :

$$R = detM - k(traceM)^2$$

where:

- $detM = \lambda_1 \lambda_2$
- $traceM = \lambda_1 + \lambda_2$

λ_1, λ_2 are the eigenvalue of matrix M. So the values of these eigenvalues decide whether a region is a corner, edge or flat.



Scale Invariant Feature Transform (SIFT)

Another features detection algorithm is SIFT. This algorithm will detect the keypoint and in order to distinguish between keypoint, we need a value call descriptor. When we apply the SIFT algorithm, for each keypoint we will get the coordinate of the keypoint, its scale and orientation. The advantages of this method is that the keypoint isn't affect by the light intensity, noise or rotation and it's very efficiency, close to real-time performance

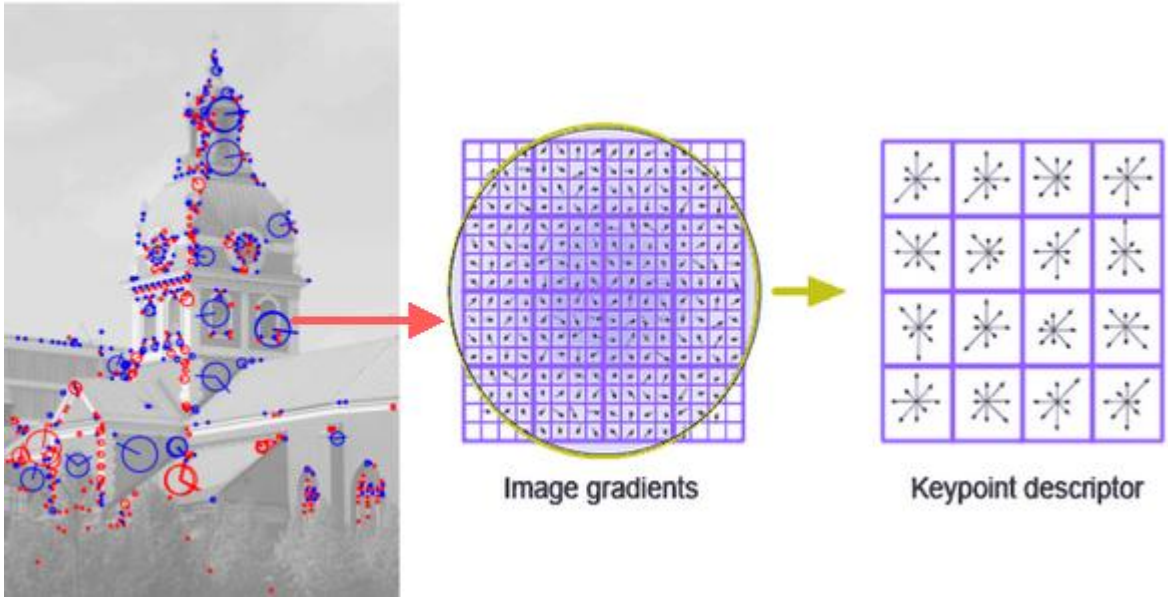
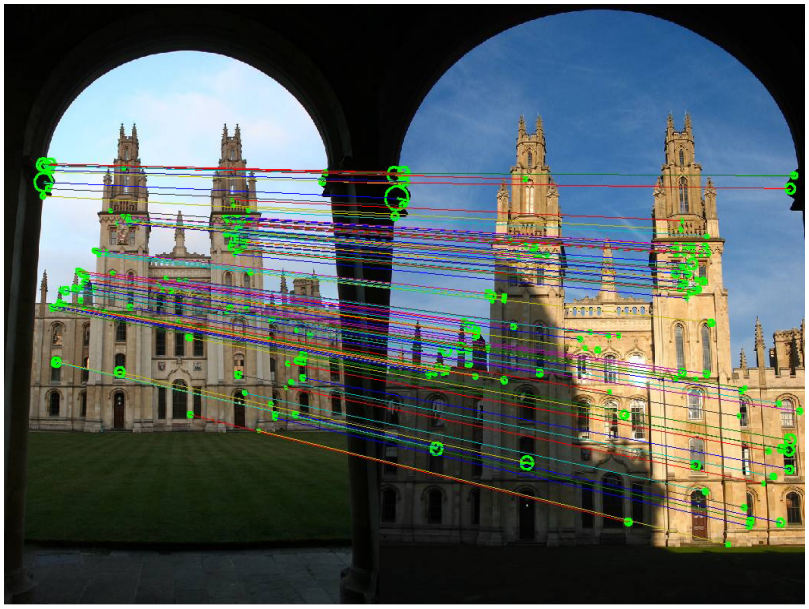


Image matching

A common approach to image matching consists of detecting a set of interest points each associated with image descriptors from image data. Once the features and their descriptors have been extracted from two or more images, the next step is to establish some preliminary feature matches between these images.



For this example, we will use the SIFT algorithm to detect the features and using Brute-force matcher to match descriptors

Code:

```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
img2 = cv.imread('hanoi-1940-paul-doumer-bridge-2-small.jpg',cv.IMREAD_GRAYSCALE) #
img1 = cv.imread('LB_pont_2_small.jpg',cv.IMREAD_GRAYSCALE) # trainImage

sift = cv.SIFT_create()
kp1, des1 = sift.detectAndCompute(img1,None)
kp2, des2 = sift.detectAndCompute(img2,None)
bf = cv.BFMatcher()
matches = bf.knnMatch(des1,des2,k=2)
good = []
for m,n in matches:
   if m.distance <  0.75*n.distance:
          good.append([m])
img3 = cv.drawMatchesKnn(img1,kp1,img2,kp2,good,None,flags=cv.DrawMatchesFlags_NOT_
plt.imshow(img3),plt.show()
```

Result: