Faculty of Engineering
Cairo University

Systems & Biomedical
Engineering Department

Scale Invarient Feature Transform (SIFT) - Harris Corners

Computer Vision Task 3

Submitted to:
Dr. Ahmed badawei
Eng. Laila Abbas
Eng. Mohamed Adel

Submitted by:

Name	Sec	BN
Ahmed Mohamed Mohamed	1	7
Mohamed Ahmed Abdelaziz	2	14
Mohamed khaled Galloul	2	15
Mohamed Abdelkareem Seyam	2	18

Cairo University

Contacts:
mohamed.ahmed997@eng-st.cu.edu.eg

please let us know if we need to provide any additional Adjustments

Contents

TASK3 - SIFT - Harris Corners

The work is based on the following papers:

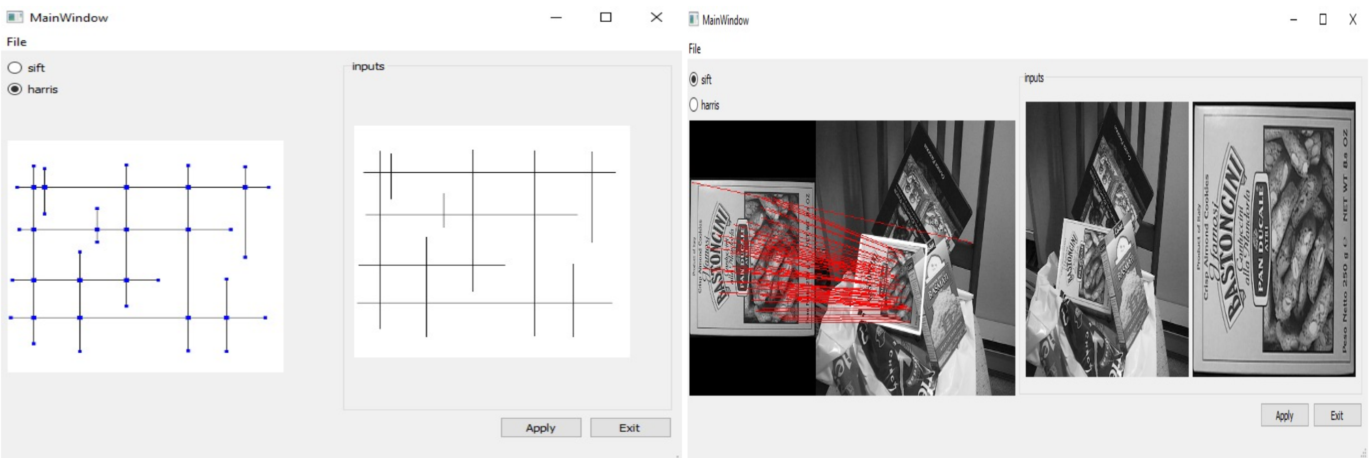
- **AOS** : Anatomy of the SIFT Method. Otero, Ives Rey. Diss. École normale supérieure de Cachan-ENS Cachan, 2015.
- **Lowe** Sift-the scale invariant feature transform. Lowe. Int. J, 2(91-110), 2, 2004.
- harris Corners paper

SIFT

Image matching is a fundamental aspect of many problems in computer vision, including object or scene recognition, image stitching, solving for 3D structure from multiple images, stereo correspondence, and motion tracking. The problem was how to describes image features that have many interesting properties suitable for matching differing images of an object or scene. The features are invariant to image scaling and rotation, and partially invariant to change in illumination and 3D camera viewpoint.

Following are the major stages of computation used to generate the set of image features:

1. **Scale-space extrema detection:** The first stage of computation searches over all scales and image locations. It is implemented efficiently by using a difference-of-Gaussian function to identify potential interest points that are **invariant to scale** and **orientation**.
2. **Keypoint localization:** At each candidate location, a quadratic model is fit to determine the accurate location and scale. Keypoints are selected based on measures of their stability by **rejecting points with low contrast neighbourhood** and **large principal curvature ratio**.
3. **Orientation assignment:** One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing **invariance to these transformations**.
4. **Keypoint descriptor:** The local image gradients are measured at the selected scale in the region around each keypoint. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination and hence account for **misregisteration** and **local positional shift**.
5. **Matching:** The purpose of detecting and describing keypoints is to find matches (pairs of keypoints) between images that minimizes the Euclidean distance between descriptors.



(a) Harris Corners (b) SIFT features matching

Figure 1: GUI Output

1 Scale-space extrema detection

To be able to extract extremum points we do the following steps:

1.1 Octaves

octaves: simulate different scales by **pyramidal concept with images blurred at multiple sigmas** to extract features from multiple scales. Detecting locations that are invariant to scale change of the image can be accomplished by searching for stable features across all possible scales, using a continuous function of scale known as scale space.

This Gaussian scale-space is a family of increasingly blurred images with sampling rate that is iteratively decreased by a factor of two, these subfamilies are called **octaves** each octave has number of layers by changing blur sigma.

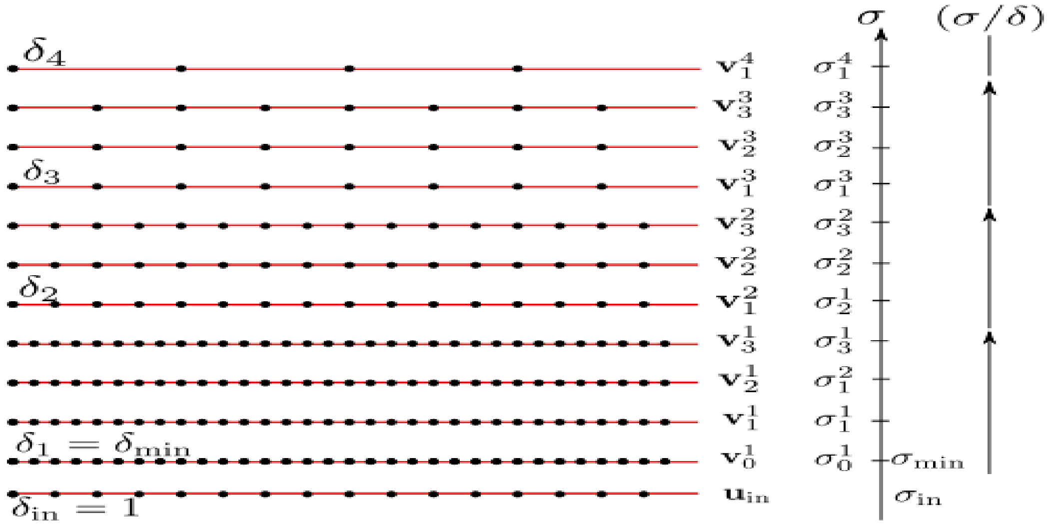


Figure 2: Convention adopted for the sampling grid of the digital scale-space v. The blur level is considered with respect to the sampling grid of the input image. The parameters are set to their default value, namely $\sigma_{min} = 0.8, n_{spo} = 5, n_{oct} = 8, \sigma_{in} = 0.5$.

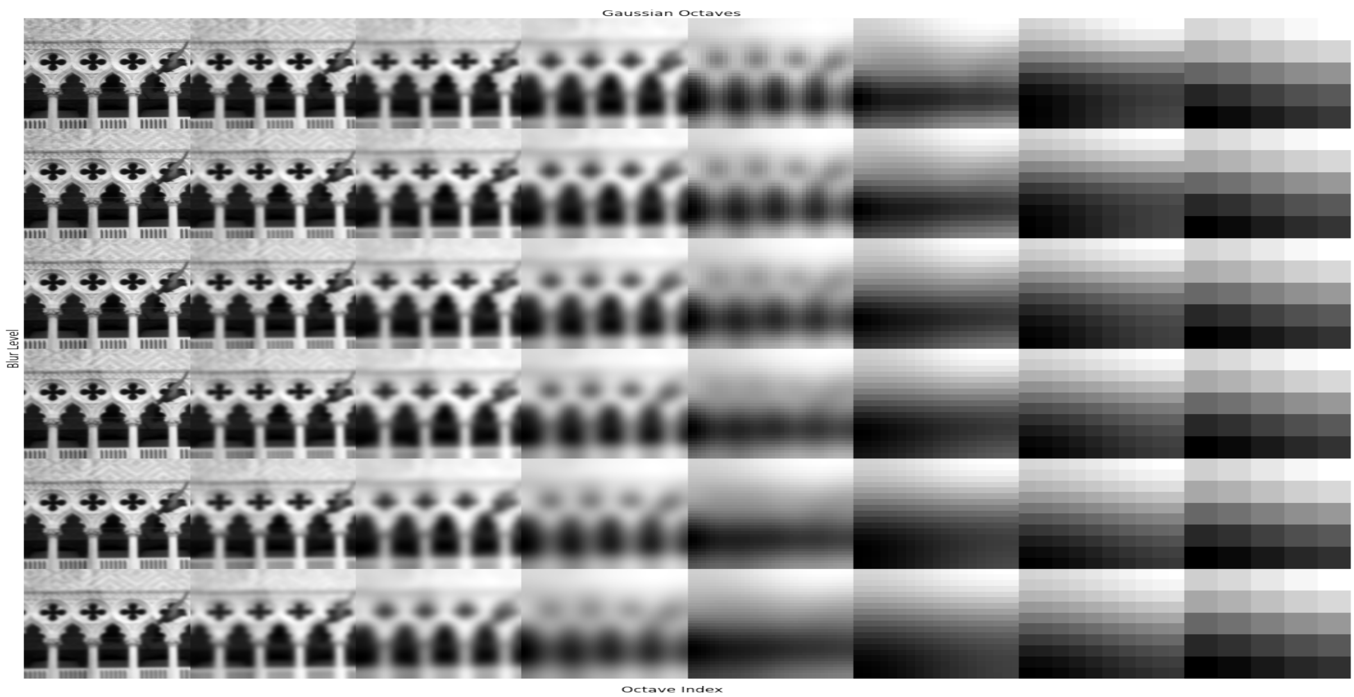


Figure 3: Sample of octave at different scales and different blurs

1.2 Difference of Gaussian (DOG)

Diffrence of Gaussian: Efficient to compute edge features by simple image subtraction **(two subsequent images at different sigma from the same octave)** which is a fast approximation to Normalized Laplacian of gaussian (NLOG) which require slow convolution computations.

$$G(x, y, k\sigma)G(x, y, \sigma) \approx (k1)\sigma^2\nabla^2 G$$

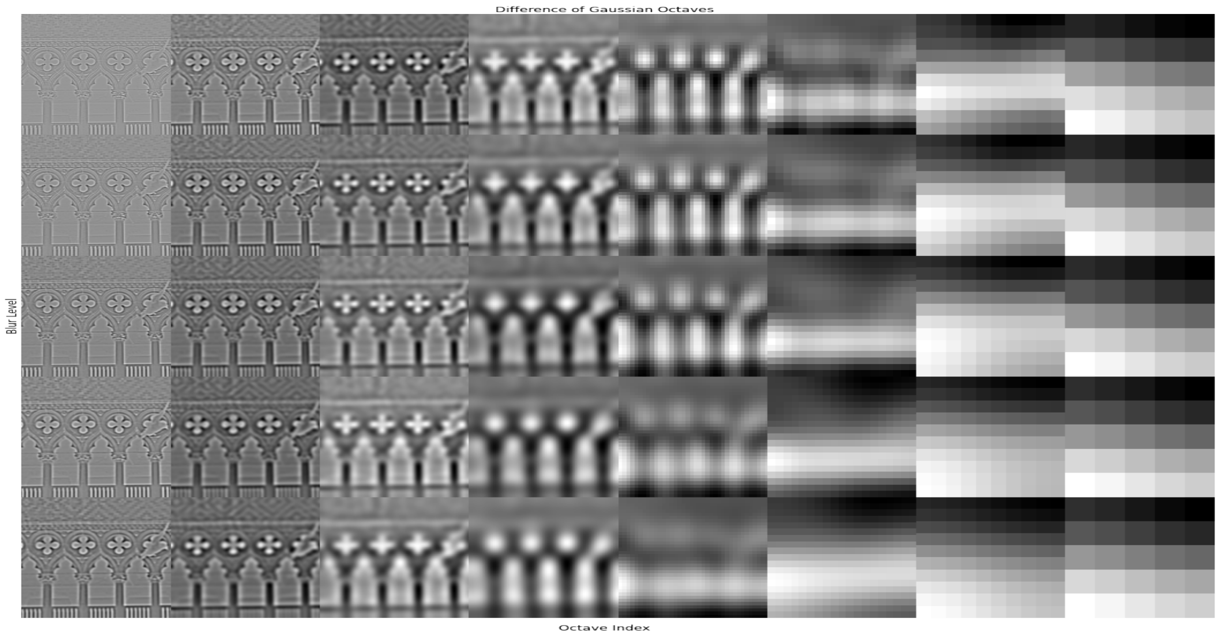


Figure 4: Sample of DOGs

1.3 Local Extremum Detection

Local Extremum Detection: Finds extrema in a DOG octave achieved by subtracting a cell by all it's direct (including diagonal) neighbors, and confirming all differences have the same sign. This could be visualized as searching for extremum in each 3d cube in each octave. A lot of extremum will be generated using this step; relying on them will affect the matching stability and hence we will reject points with low contrast neighbourhood and large principal curvature ratio.

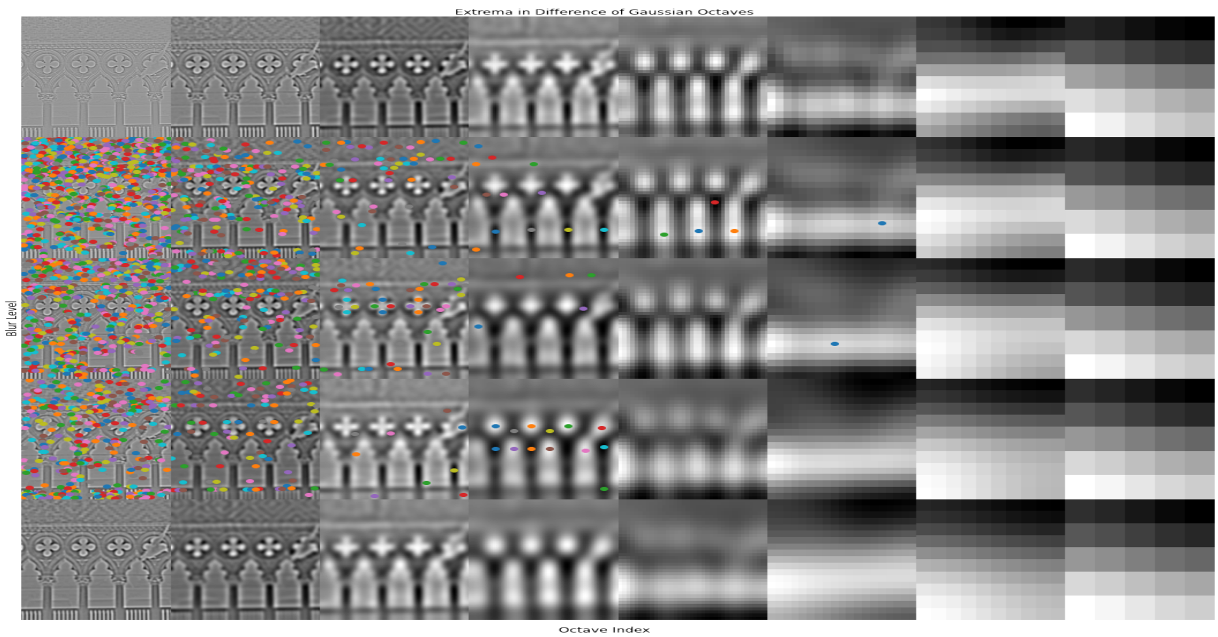


Figure 5: Extremum points from DOGs

2 Keypoints (Selection from DOG Extremum)

2.1 Accurate Keypoint Localization

The extremum location found in D(sigma, y, x) is not greatly localized as we are only considering a subset of scales. To precicely localize where a keypoint is present we will use quadratic model fit as follows:

$$D(X) = D + \frac{\partial D^T}{\partial X} X + \frac{1}{2} X^T \frac{\partial^2 D}{\partial X^2} X$$

$$\hat{X} = -\frac{\partial^2 D^{-}1}{\partial X^2} \frac{\partial D}{\partial X}$$

- X : is vector of x, y, sigma.
- D : is Diffrence of Gaussian.
- $\hat{x}$: keypoint offset

2.2 Contrast Threshold

The function value at the extremum, $D(\hat{x})$, is useful for rejecting unstable extrema with low contrast. This can be obtained by substituting in:

$$D(\hat{X}) = D + \frac{1}{2} \frac{\partial D^T}{\partial X} \hat{X}$$

all extrema with a value of $|D(\hat{X})|$ less than 0.03 were discarded (assume image pixel values in the range [0,1]).

2.3 Eliminating Edge Response (Principle Curvature Ratio)

A poorly defined peak in the difference-of-Gaussian function will have a large principal curvature across the edge but a small one in the perpendicular direction. The principal curvatures can be computed from a 2x2 Hessian matrix, H, computed at the location and scale of the keypoint:

compute the sum of the eigenvalues from the trace of H and their product from the determinant:

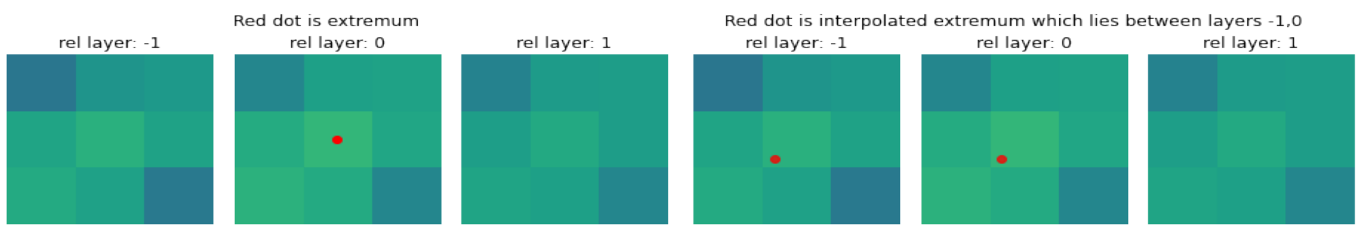
$$Tr(H) = D_{xx} + D_{yy} = \lambda_1 + \lambda_2$$

$$Det(H) = D_{xx} D_{yy} - (D_{xy})^2 = \lambda_1 \lambda_2$$

then,

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}$$

A value of r = 10, which eliminates keypoints that have a ratio between the principal curvatures greater than 10.



(a) Maximum from DOG (b) Interpolated Maximum

Figure 6: Accurate Keypoint Localization

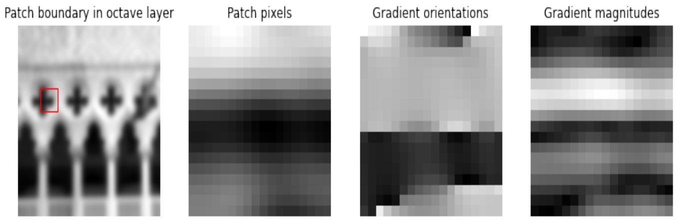
3 Keypoint Orientation

The orientation of the keypoint is important as it is used to translate the local neighbourhood region with reference to its principal orientation hence provide **orientation invariant** features.

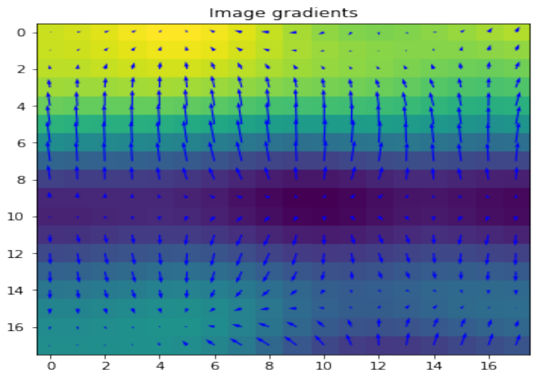
Gradient Direction is used to weight the magnitude of each orientation in generating the histogram.

Orientation Histogram (36 bins) Maxima

if there is another orientaion that lie within 80% of the maximum a new key point is generated similar to the original, but with another orientation.



(a) Image patch magnitude and gradient



(b) Gradient Image

Figure 7: Gradient Orientation

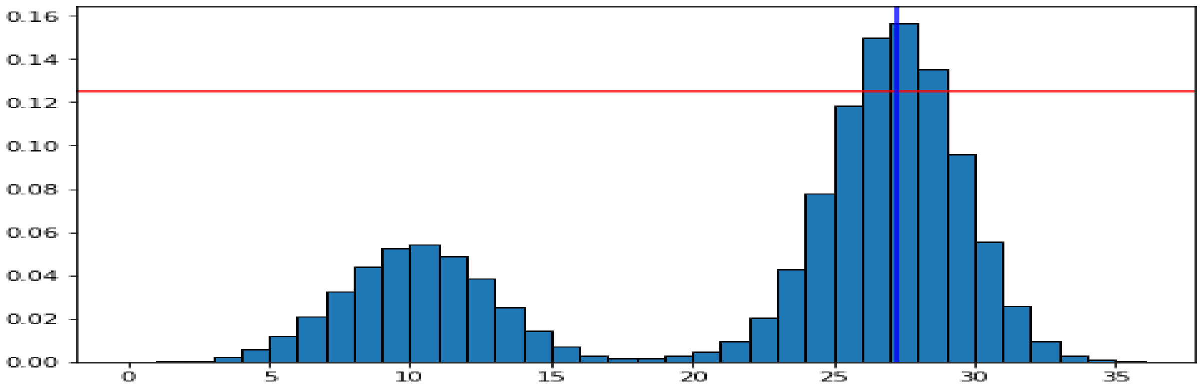


Figure 8: Gradient Orientation Weighted Smoothed Histogram

4 Descriptor Representation

A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location.

These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms (8 bins) summarizing the contents over 4x4 subregions. [4x4x8 = 128 element feature vector]

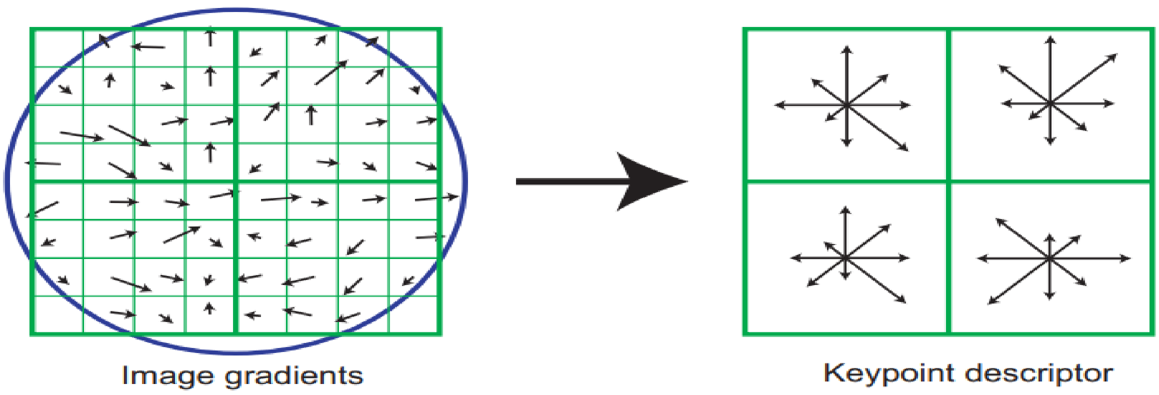


Figure 9: Gradient Orientation Weighted Smoothed Histogram

A goof descriptor used in matching should account for misregistration, distortion, noise and illumination changes. Simple correlation of image patches is highly sensitive to changes that cause misregistration of samples, such as affine or 3D viewpoint change or non-rigid deformations.

To solve this problems we could mimic the complex neurons in primary visual cortex. These complex neurons respond to a gradient at a particular orientation and spatial frequency, but the location of the gradient on the retina is allowed to shift over a small receptive field rather than being precisely localized.

Mimicing Neurons could be achieved by 3 steps:

4.1 A Gaussian weighting function (solve misregistration, distortion)

The purpose of this Circular Gaussian window is to - Avoid sudden changes in the descriptor with small changes in the position of the window - Give less emphasis to gradients that are far from the center of the descriptor, as these 'edges are most affected by misregistration errors'.

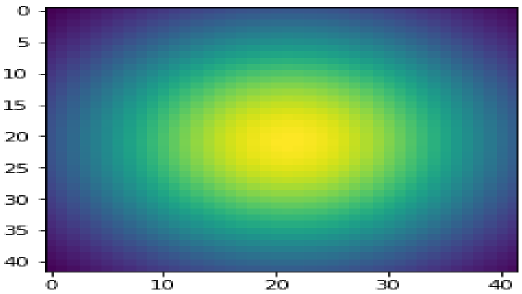
4.2 (4 x 4 x 8 bin) histograms concatenation (solve distortion)

using only 8 bins allows for significant shift in gradient positions by creating orientation histograms over 4x4 sample regions. A gradient sample can shift up to 4 sample positions while still contributing to the same histogram, thereby achieving the objective of allowing for larger 'local positional shifts'.

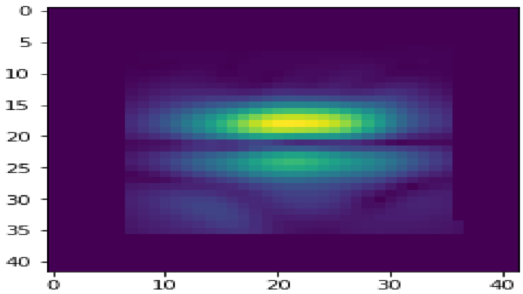
4.3 Normalized Vector (solve for illumination changes)

A change in image contrast in which each pixel value is multiplied by a constant will multiply gradients by the same constant, so this contrast change will be canceled by vector normalization.

A brightness change in which a constant is added to each image pixel will not affect the gradient values, as they are computed from pixel differences. Therefore, the descriptor is 'invariant to affine changes in illumination'.



(a) Gaussian Weighting Function



(b) Gradient Patch

Figure 10: Descriptor Patch Weighting

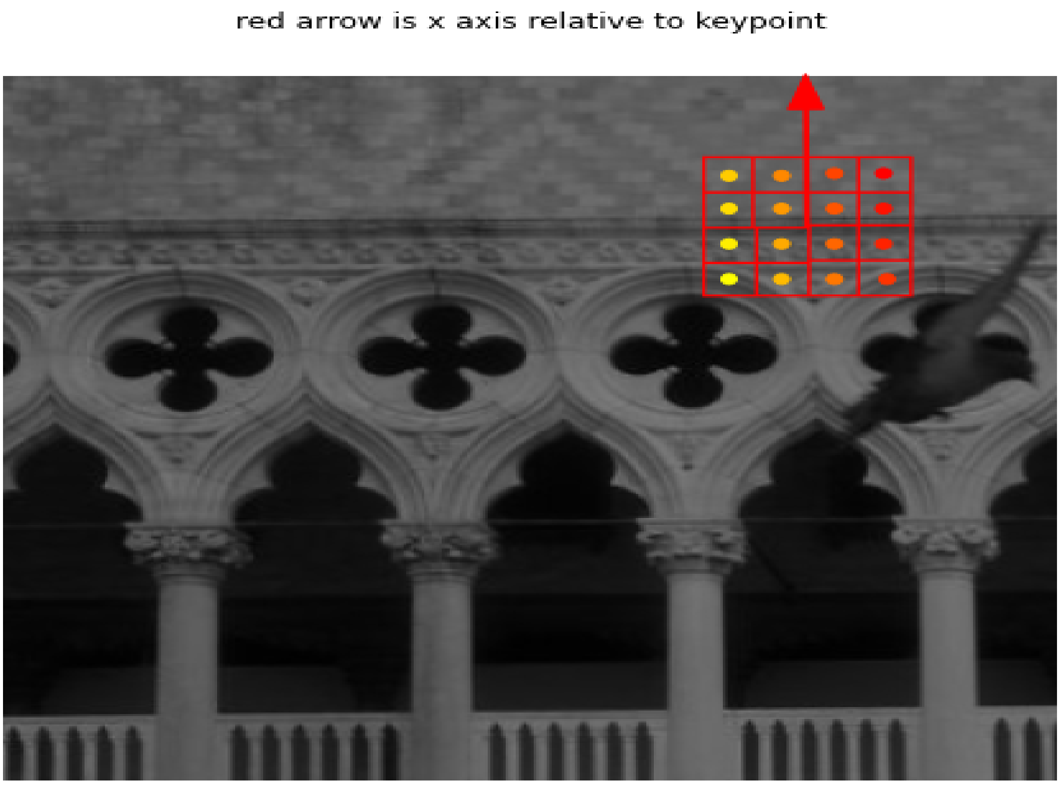


Figure 11: keypoint with its surrounding patch 4x4 sub-regions along with its principal orientation

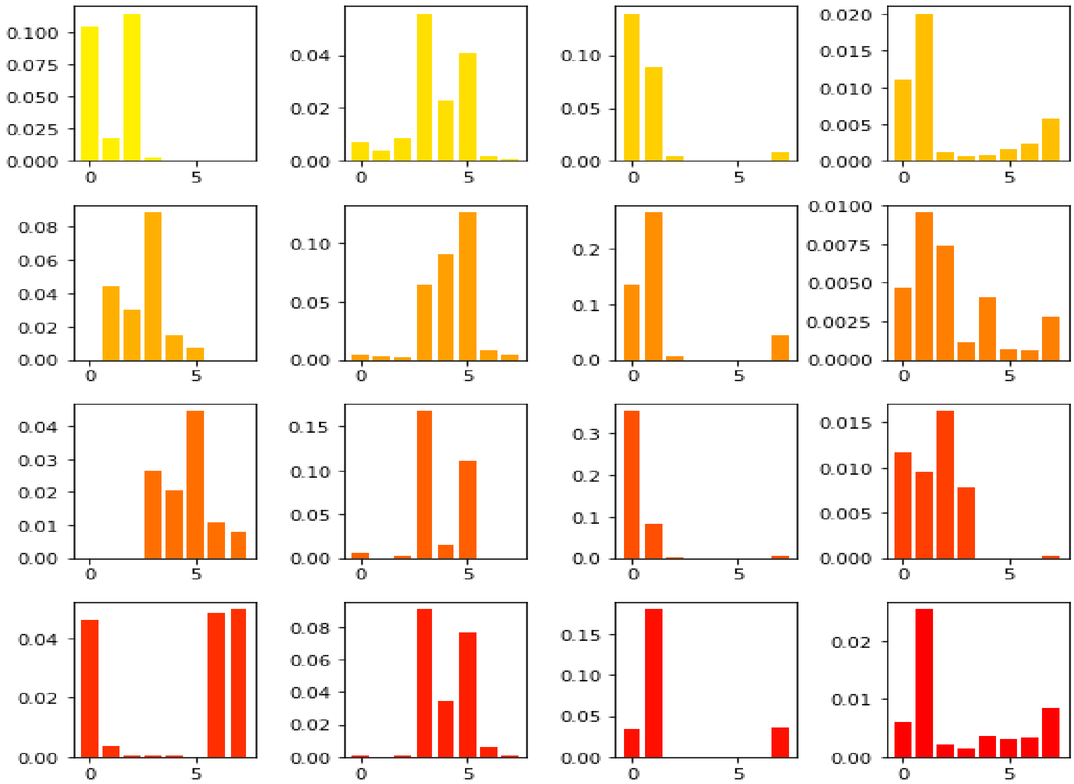


Figure 12: Histogram of each sub-region

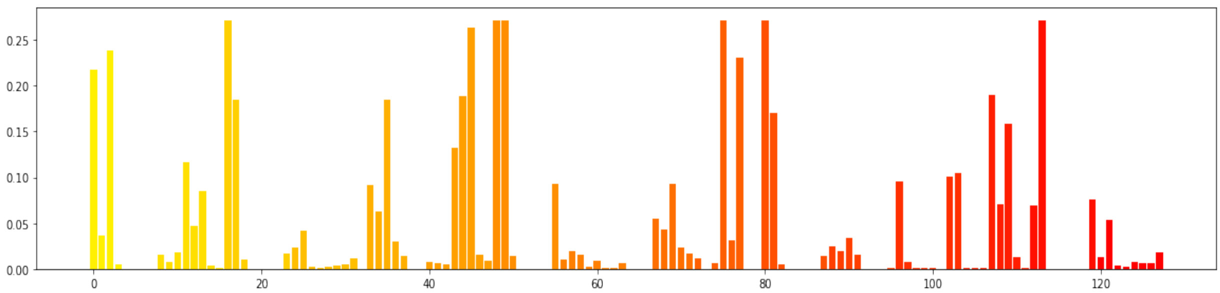


Figure 13: Concatenation of 4x4 sub-regions histogram to form a descriptor feature vector

5 Matching

To match features of two images we simply search for each descriptor in image1 to find the neareast neighbour from the image2 descriptors by using sum of squared differences (SSD).

img	UnVectorized	Vectorized
box	148 s	14 s
london	296s	54 s
san-marco	13+ min	4 min 14 s

Test on hard images: **different 3d views**, **different scales** , **occluded objects**.

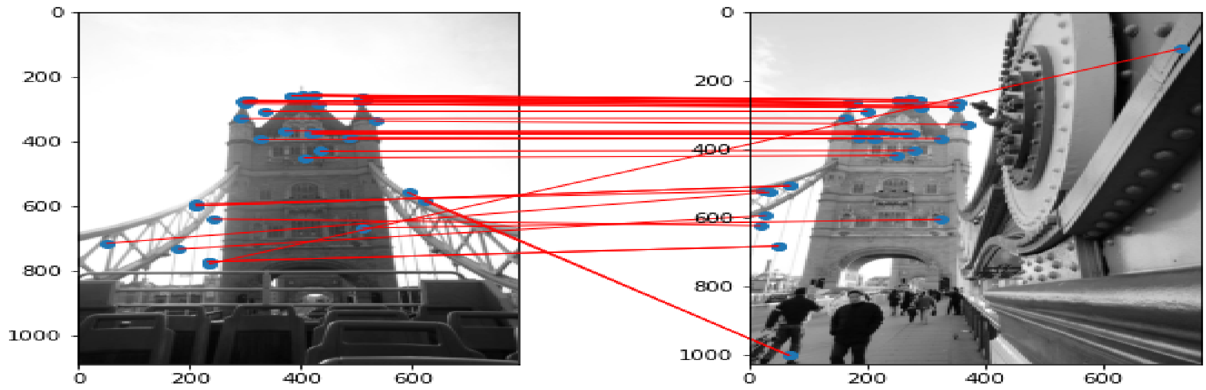


Figure 14: London Bridge

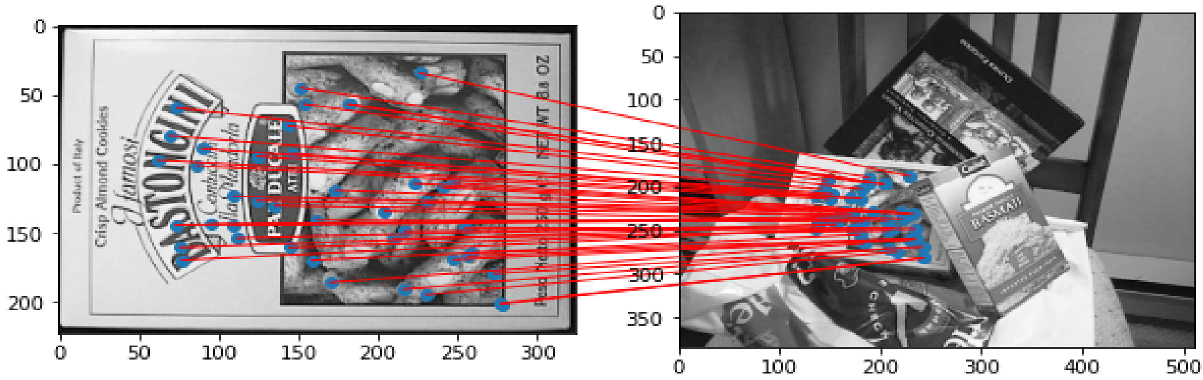


Figure 15: Box in Scene

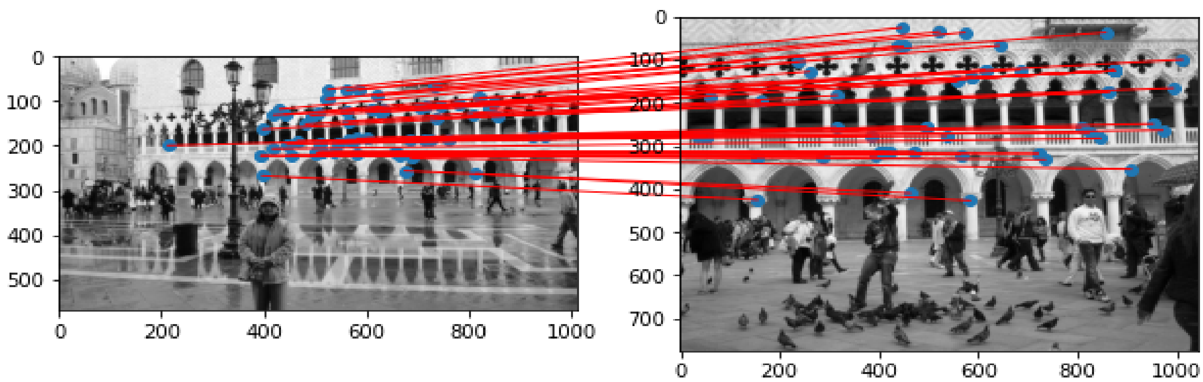


Figure 16: San Marco

6 Harris Corner Detection

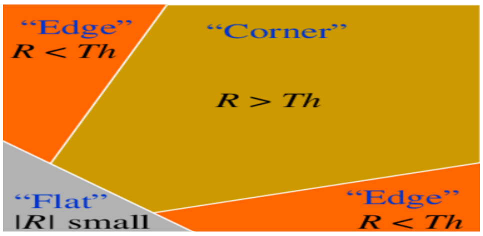
The Harris corner detector is a corner detection operator that is commonly used in computer vision algorithms to extract corners and infer features of an image.

A corner is a point whose local neighborhood stands in two dominant and different edge directions. In other words, a corner can be interpreted as the junction of two edges, where an edge is a sudden change in image brightness.

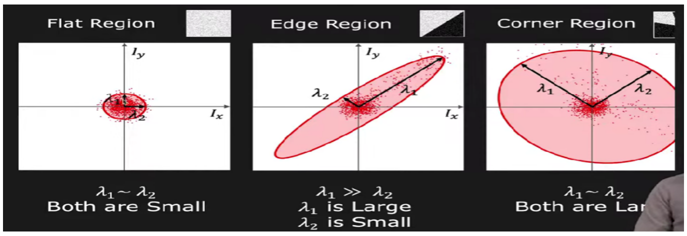
Corners are the important features in the image, and they are generally termed as interest points which are invariant to translation, rotation and illumination. Although corners are only a small percentage of the image, they contain the most important features in restoring image information, and they can be used to minimize the amount of processed data for motion tracking, image stitching, building 2D mosaics, stereo vision, image representation and other related computer vision areas.

Also in the SIFT part we discussed how to select interesting keypoints and we reject low contrast regions and regions with large curvature ratio this technique is actually borrowed from harris

for the math behind the Harris check section 2.3 above.



(a) harris space



(b) Eigen Values (λ_+, λ_-)

Figure 17: Harris Corner