Computer Vision: Task 3 Report

1- Extract the unique features in all images using Harris Operator and λ- Operator

pre processing :

1. apply gaussian blur to the image to reduce image
2. convert the blurred image to grayscale

steps to implement Harris corner detector:

1. Calculate image x and y derivatives:

   Using the Sobel operator to compute the derivatives of the image in the x and y directions.

3. Sum the derivatives for each pixel:

   For each pixel in the image, apply a 1-pixel shift of a window over the image and sum the derivative values within the window.

4. Define the H matrix for each pixel:

   Using the sums of derivatives computed in the previous step, define the H matrix for each pixel. The H matrix is a 2x2 matrix that represents the local structure of the image around each pixel.

5. Calculate the response of the detector:

   Compute the corner response function R for each pixel using the determinant and trace of the H matrix. The corner response function is defined as
   If the Harris flag is set, compute the Harris corner response function using the determinant and trace of M

$$R = \det(M) - k(\text{trace}(M))$$

where k is a constant parameter.

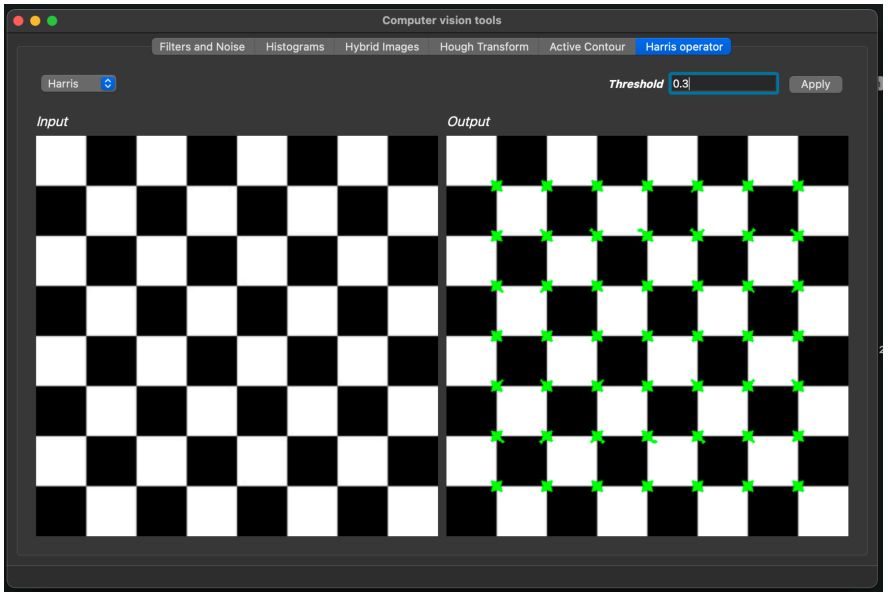
Otherwise, use the λ-based response function.

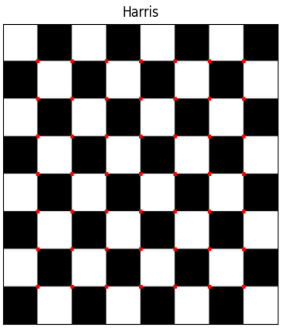
$$R = \det(M) / (\text{trace}(M))$$

6. Apply a threshold to exclude detections:

Finally, apply a threshold to the corner response values to identify significant corners. Pixels with a corner response value above the threshold are considered corners, while those below the threshold are discarded.

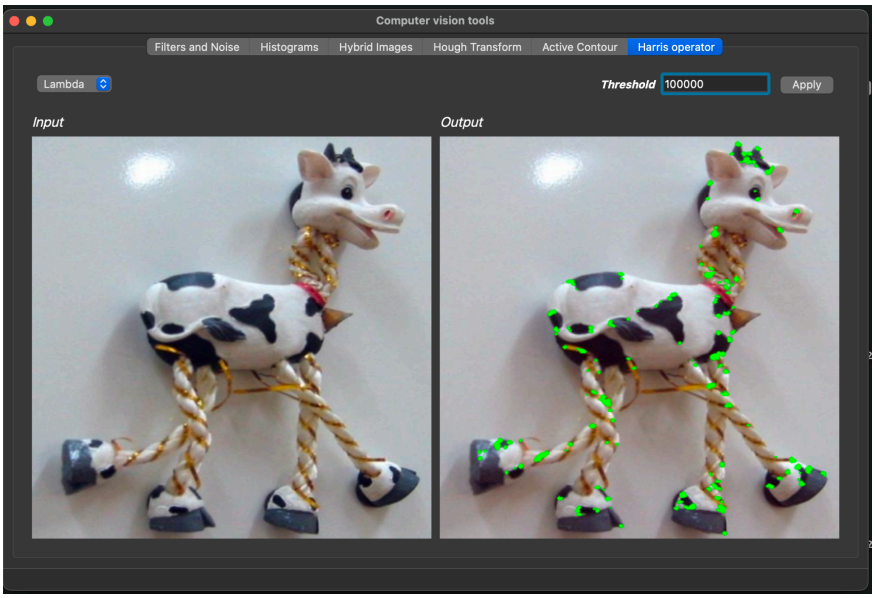
Examples compared with **opencv** function





Time taken: 2.6839559078216553 seconds

Time taken: 2683955.9078216553 microseconds



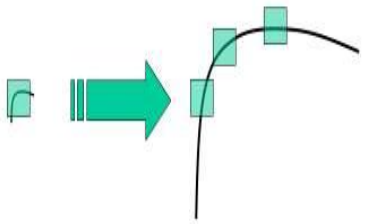
Time taken: 4.587117910385132 seconds

Time taken: 4587117.910385132 microseconds

2- Generate Feature Descriptors Using Scale-Invariant Feature Transform (SIFT)

The **scale-invariant feature transform (SIFT)** is a feature extraction method (among others, such as HOG feature extraction) where image content is transformed into local feature coordinates that are invariant to translation, scale and other image transformations, invented by David Lowe in 1999. The SIFT Detector has been said to be a close approximation of the system used in the primate visual system. Some of its applications include object recognition, image stitching, 3D modeling, and video tracking.

Unlike corner detectors, like the **Harris corner** detector, which are only rotation-invariant (i.e. the recognition or detection of a feature - in this case, a corner - in an image is unaffected by the rotation of the image), **SIFT** is both scale- and rotation-invariant. This is due to the fact that if an image is scaled, a corner may not "remain" a corner, causing the corner detector to vary with scale. This phenomenon is shown in the image below.



a b

Figure: (a)Original corner in a window; (b)The same corner scaled up used with windows of the same size

So, unlike the **Harris Detector**, which is dependent on properties of the image such as viewpoint, depth, and scale, **SIFT** can perform feature detection independent of these properties of the image. This is achieved by the transformation of the image data into scale-invariant coordinates. There are four main steps involved in the SIFT algorithm:

1. Scale-space Extrema Detection

From the figure above, it is clear that the same window cannot be used to detect corners of a different size, especially if the corner is larger than the original. For this reason, scale-space filtering is used. First, the image is upscaled to provide a higher resolution base that helps the detection of smaller features. Then, to create the scale-space, different octaves, a series of images in the scale-space pyramid where each image is a progressively blurred version of the base image, must be created where each octave a scaled down version of the previous octaves. **This helps the algorithm remain invariant to scaling, whether up or down.**

In the scale-space, the Laplacian of Gaussian (LoG), which is essentially the second derivative of a Gaussian filter in 2D, is found for the image with various σ values (or scales). LoG acts as a blob detector which detects blobs in various sizes due to change in the σ sigma parameter, i.e. different "strengths" or blur. In short, the σ parameter acts as a scaling parameter. For example, in the above figure, a gaussian kernel with a low σ gives high values for small corners while a gaussian kernel with a high σ fits well for larger corners. By

finding the local extrema (maxima and minima) across the scale-space, we obtain a list of **(x,y,σ)** values where there is a potential keypoint at position **(x,y)** at the σ scale.

However, LoG is an expensive operation, so in the SIFT algorithm, the Difference of Gaussians (DoG), an approximation of the LoG, is used instead. DoG is obtained as the difference of two successive Gaussian blurred images at different σ (or scales), assume σ and $k\sigma$. Specifically, a DoG image ***D(x,y,σ)*** is given by

$$D(x, y,\sigma) = L(x, y, \sigma) - L(x,y,k\sigma) ,$$

where **L(x,y,kσ)** is the convolution of the original image I(x,y) with the Gaussian blur **G(x ,y,kσ**) at scale **kσ** , i.e., $L(x,y,k\sigma) = G(x ,y,k\sigma) * I(x,y)$

This process is done for all octaves of the image in the scale-space Pyramid. It is represented in the image below:

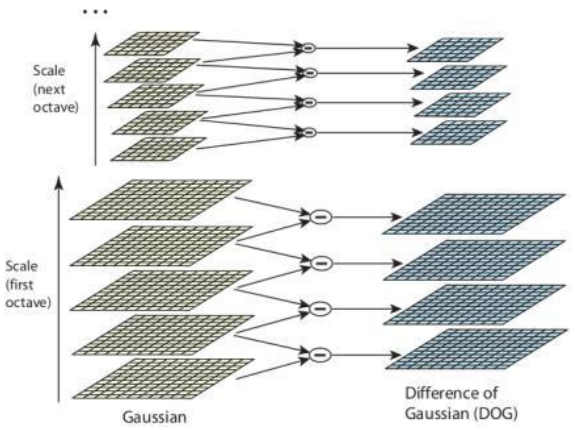


Figure: the process of calculating Difference of Gaussians or DoG

Once the DoGs are calculated, the scale-space is searched for local extrema over both scale and space. For example, by comparing a pixel's value in the image to the values of its neighbors in the 3x3x3 neighborhood in the scale space, the pixel is determined whether or not it is a local extrema. This neighborhood includes 8 neighbors in the current, same image, 9 neighbors in the previous scale, and 9 neighbors in the next scale. **If it is, in fact, a local extrema, it is a potential keypoint. It essentially means that the keypoint is best represented in that scale.** This process is shown in below image:

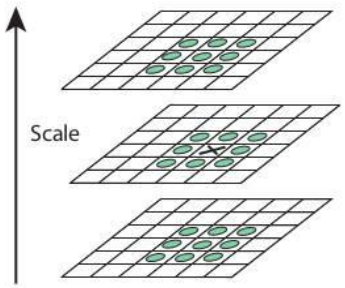


Figure: comparing center pixel in a 3x3x3 neighborhood to other pixels to determine if it is a local extrema

Regarding different parameters, David Lowe's 1999 paper gives some empirical data which can be summarized as, ***number of octaves = 4***, ***number of scale levels per octave = 5***, ***contrast threshold = 0.03***, ***initial*** $\sigma = 1.6$***,*** $k = \sqrt{2}$, etc as optimal values.

2. Keypoint Localization

Once potential keypoints locations are found, they must be refined to get more precise results. In the paper, the quadratic Taylor series expansion of scale space was used to get a more accurate location of the extrema. Additionally, if the intensity at these extrema is less than a contrast threshold value (**0.03** as per the paper), i.e. keypoint having low contrast, the keypoint is rejected.

The DoG function has strong responses along edges, even if the candidate keypoint is not robust to small amounts of noise. Therefore, in order to increase stability, we need to eliminate the keypoints that have poorly determined locations but have high edge responses.

For poorly defined peaks in the DoG function, the principal curvature across the edge would be much larger than the principal curvature along it. Similar to the Harris corner detector whereby edges had one eigenvalue that is larger than the other, the second-order Hessian matrix, ***H***, is used to find these principal curvatures amounts by solving for the matrix eigenvalues:

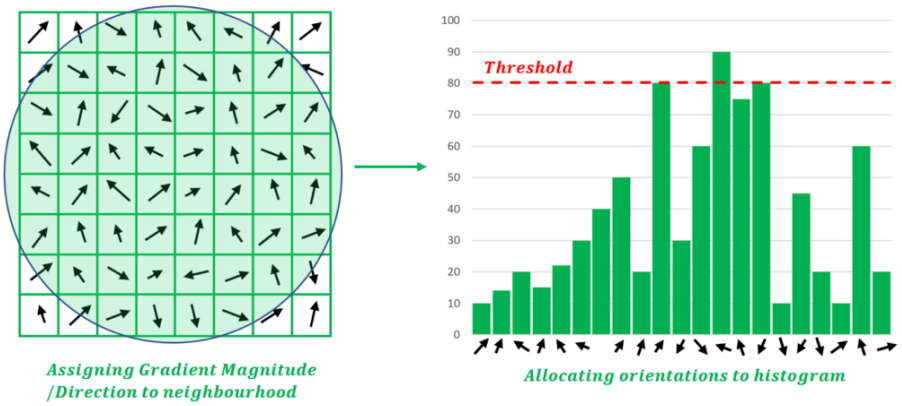
$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

It turns out that the ratio of the two eigenvalues, say **α** is the larger one, and **β** the smaller one, with ratio **r = α / β**, is enough for SIFT. Therefore, if this ratio is greater than the edge threshold or threshold eigenvalue ratio (equal to 10 in the paper), that keypoint is discarded.

Through keypoint localization, any low-contrast keypoints and poor edge-response keypoints are eliminated. The only keypoints that remain are strong interest points.

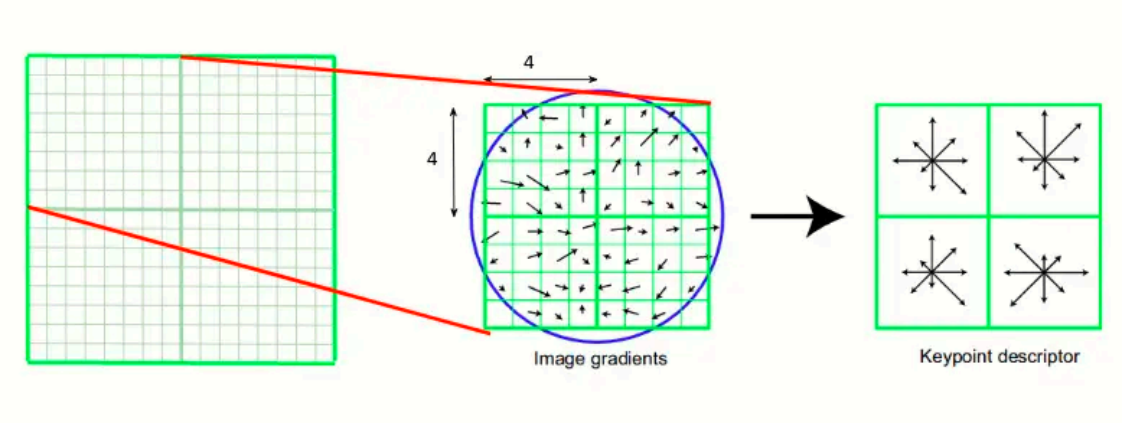
3. Orientation Assignment

Now, **an orientation is assigned to each keypoint to achieve invariance to image rotation**. A neighborhood is taken around the keypoint location depending on the scale, and the gradient magnitude and direction is calculated in that region. An orientation histogram with 36 bins to represent 360 degrees of orientation (10 degrees per bin) is created. The histogram is weighted by gradient magnitude and gaussian-weighted circular window with σ equal to 1.5 times the scale of the keypoint. The highest peak in the histogram is taken. Furthermore, any histogram peaks above 80% are also considered to the calculation of the orientation of a keypoint. These >80% peaks are used to create keypoints with the same location and scale, but different directions; **it contributes to the stability of matching.**



4. Keypoint Descriptor

At this point, each keypoint has a location, scale and orientation. Now, the keypoint descriptor is created using a normalized region around the key point. A 16x16 neighborhood around the keypoint is taken.This 16×16 block is further divided into 4×4 sub-blocks and for each of these sub-blocks, an 8 bin histogram is created using magnitude and orientation. **This results in a total of 128 bin values (16 sub-blocks * 8 bins per block) that are represented as a vector to form a keypoint descriptor.** In addition to this, several measures are taken to achieve robustness against illumination changes, rotation etc.

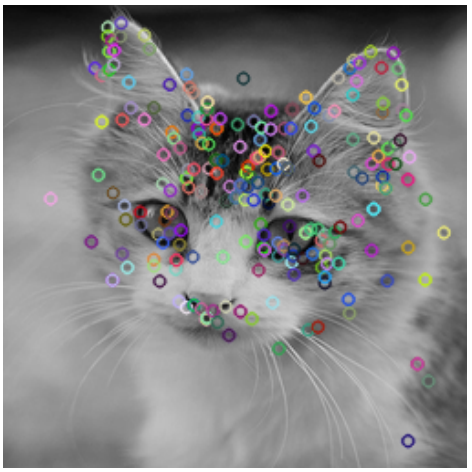


After explaining the SIFT algorithm used, here are our results in trying to extract SIFT Keypoints and Descriptors from images:

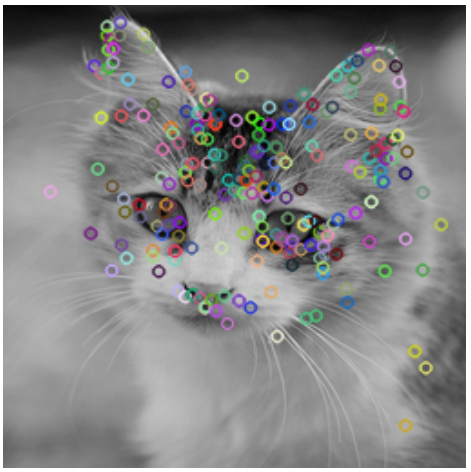
(Note: it is not used in the UI directly but rather it is used as a sub function in the feature matching discussed below)

SIFT Output with sigma=1.6, number of intervals = 3 (i.e. number of images per octave = 7), an assumed initial blur of 0.5, and an image border of 5:

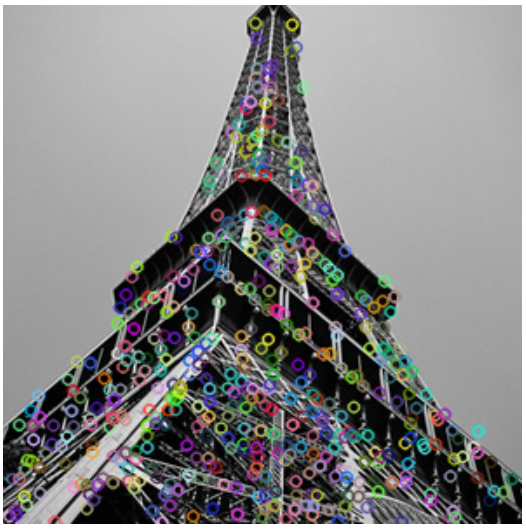
The output of our SIFT algorithm

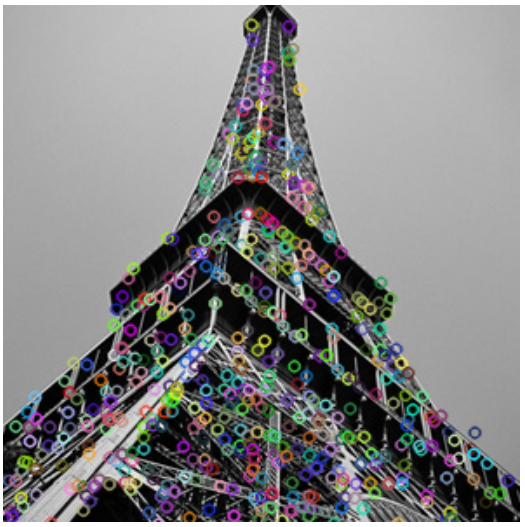


OPENCV output



→ ***Time taken = 18.032 seconds***





→ **Time taken = 21.88 seconds**

*Note: **Computation Times vary from one computer to another.**
→ Also, as image size and resolution increase, the computation time also increases since more SIFT Features and Descriptors are calculated.

3-Feature matching using SSD and NCC

The SSD and NCC techniques is utilized to assess the similarity between corresponding image regions and identify potential matches.

- **Algorithm steps:**
    1. Extract keypoints and descriptors using the SIFT algorithm from both images.
    2. For each descriptor in the reference image, compute its similarity with all descriptors in the secondary image.
    3. Calculate the Sum of Squared Differences (SSD) between the feature vectors of each pair of descriptors.
    4. Determine the descriptor in the secondary image with the lowest SSD score relative to the reference descriptor.
    5. Alternatively, compute the Normalized Cross Correlation (NCC) between the intensity differences of corresponding image regions.
    6. For each descriptor in the reference image, a matching descriptor in the secondary image is determined based on NCC scores that represent the similarity between the normalized intensity differences of the image regions. NCC scores are calculated by normalizing the cross-correlation between the descriptors.
    7. Select the descriptor in the secondary image with the highest NCC score exceeding a predefined threshold as the optimal match for the reference descriptor.

- **Comparing with the CV built-in function:**
    1. NCC matching

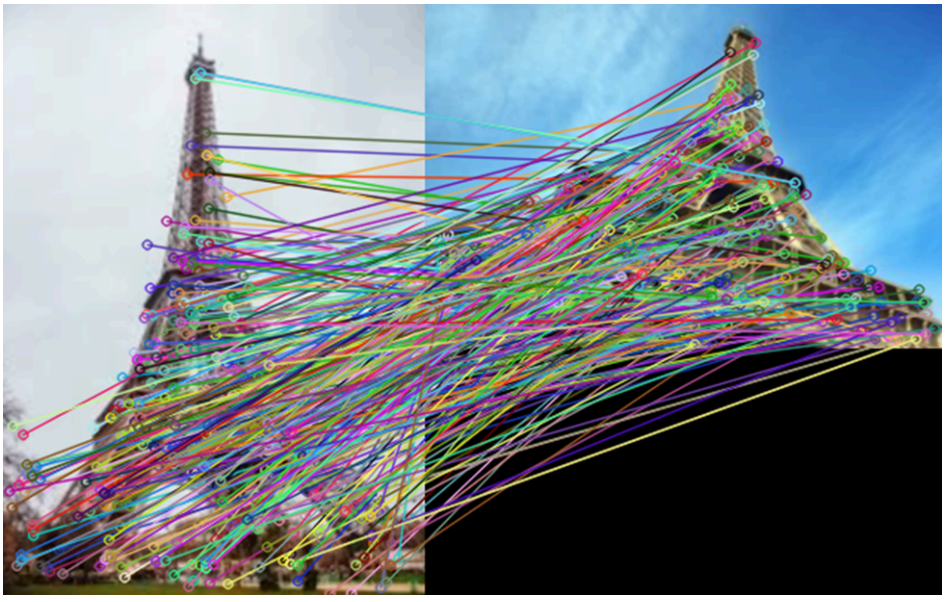


Figure 1. NCC built-in function

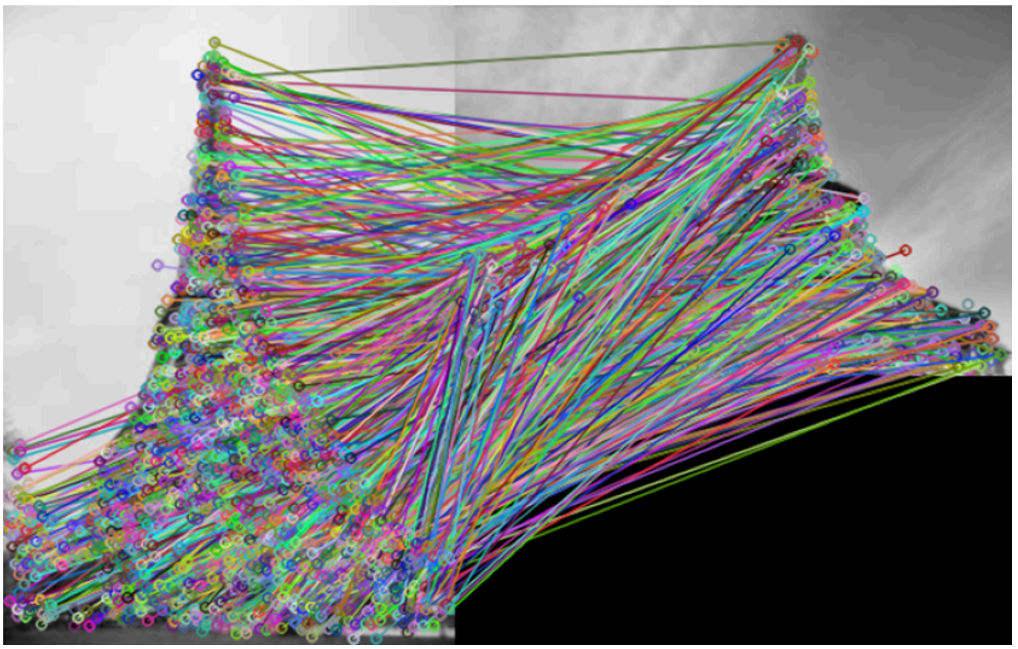


Figure 2. NCC implementation

2. SSD matching

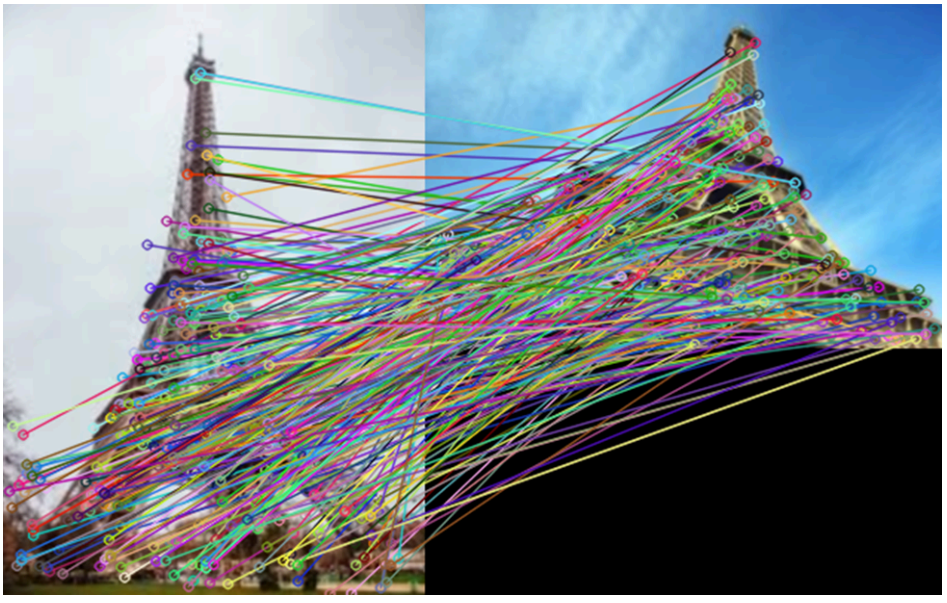


Figure 3. SSD built-in function

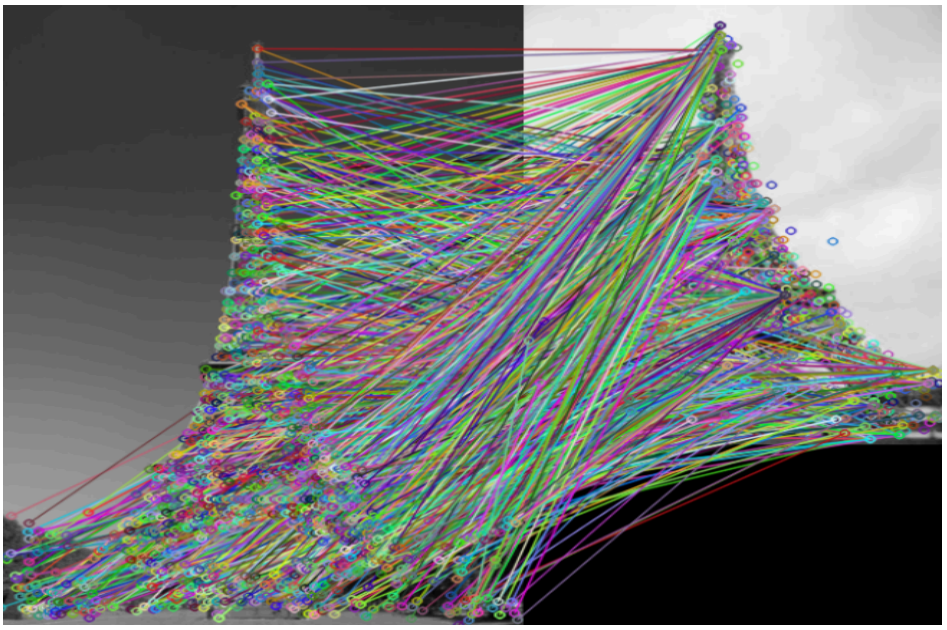


Figure 4. SSD implementation