Team 15

Team Members:

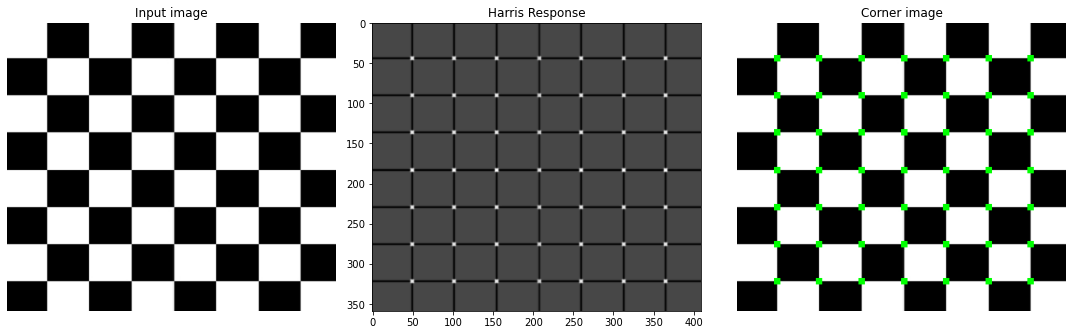
* Ahmed Hossam Mohammed Sedky Sec:1 Bn:2
* Ahmed Mohammed Abd elftatah sec:1 Bn:5
* Ehab Wahba Abdelrahman sec:1 Bn:22
* Mo’men Maged Mohammed sec:2 Bn:12
* Mohanad Alaa Ragab sec:2 Bn:31

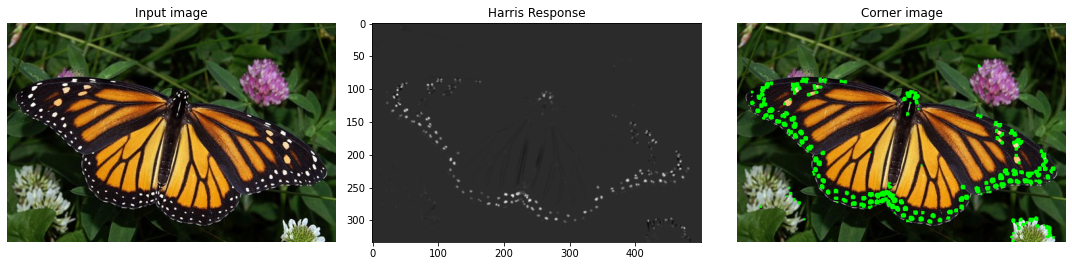
Emai to contact: [hossedky@gmail.com](mailto:hossedky@gmail.com)

**Harris Operator**

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

* Results
* First image computation time = 0.5 sec



* Second Image computation time = 0.5 sec 
* Discussion

First, we compute the gradient of the image in x-direction and y-direction using Sobel.

Second, we compute the second derivative of x,y, and x with respect to y.

Then we apply a Gaussian filter to it.

Then we compute the determine and the trace of this matrix



Finally, compute harris response “R”

Where R = det - k \*(trace)^2

Where k: corner sharpness = 0.04

Get pixels above specific threshold and color it on the

original image

**Scale Invariant Feature Transform (SIFT)**

1. Compute the image Pyramid

* Generate the base image by upscaling the input image x2 and applying a gaussian blur with a sigma of 1.6.
* Compute the number of octaves in the image pyramid
* Generate gaussian kernels which creates a list of the amount of blur for each image in a particular layer.
* Next we generate the gaussian images in each octave by blurring it with the sigma generated in gaussian kernels.

1. Find the scale-space extrema

* We just iterate through each layer, taking three successive images at a time. In each triplet of images, we look for pixels in the middle image that are greater than or less than all of their 26 neighbors: 8 neighbors in the middle image, 9 neighbors in the image below, and 9 neighbors in the image above.

1. Keypoints orientations

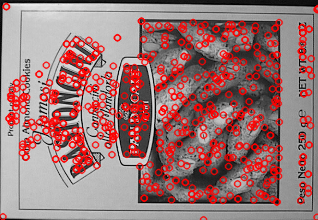
* Create a histogram of gradients for pixels around the keypoint’s neighborhood.
* Next, we compute the magnitude and orientation of the 2D gradient at each pixel in this neighborhood. We create a 36-bin histogram for the orientations 10 degrees per bin.
* We smooth the histogram. The smoothing coefficients. correspond to a 5-point Gaussian filter.

1. Cleaning Up Keypoints

* We sort and remove duplicates and convert our keypoints from base image coordinates to input image coordinates, which we can accomplish by simply halving the relevant attributes.

1. Generating Descriptors

* For each keypoint, our first step is to create another histogram of gradient orientations. We consider a square neighborhood (different side length this time) around each keypoint, but now we rotate this neighborhood by the keypoint’s angle.
* We’ll smooth the weighted gradient magnitude for each neighborhood pixel by distributing it among its eight neighbors in three dimensions: row bin, column bin, and orientation bin.
* Our last step is to flatten our smoothed 3D array into a descriptor vector of length 128. Then we’ll apply a threshold and normalize.



**Matching**

* Generate keypoints and descriptors of the two images by SIFT
* Match keypoints of the two images by SSD or NCC
* In SSD get the difference between the keypoints of the two images
* Least difference means best match
* In NCC we get the correlation between the keypoints
* Then sort the output from SSD or NCC
* Draw matches between the best correlated keypoints

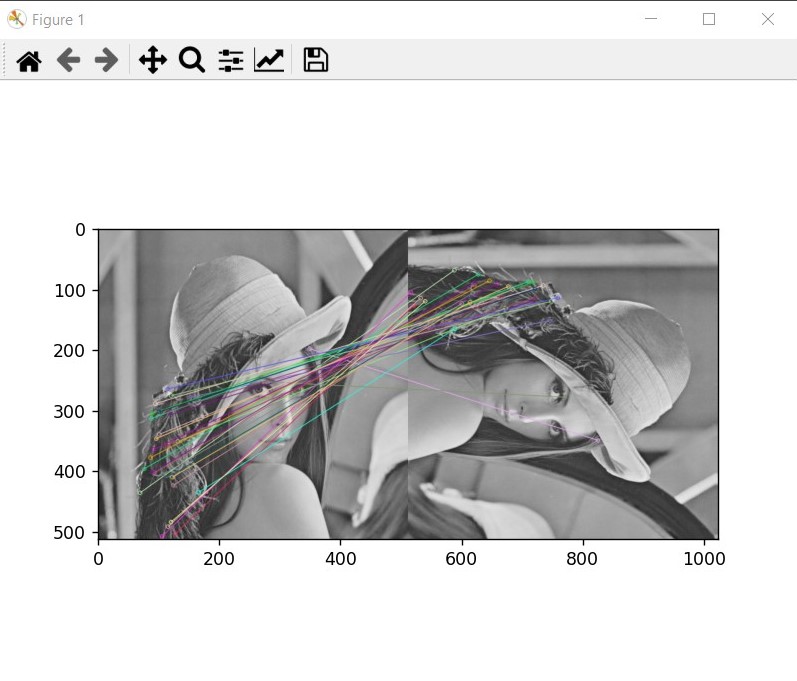


Image 1 NCC match

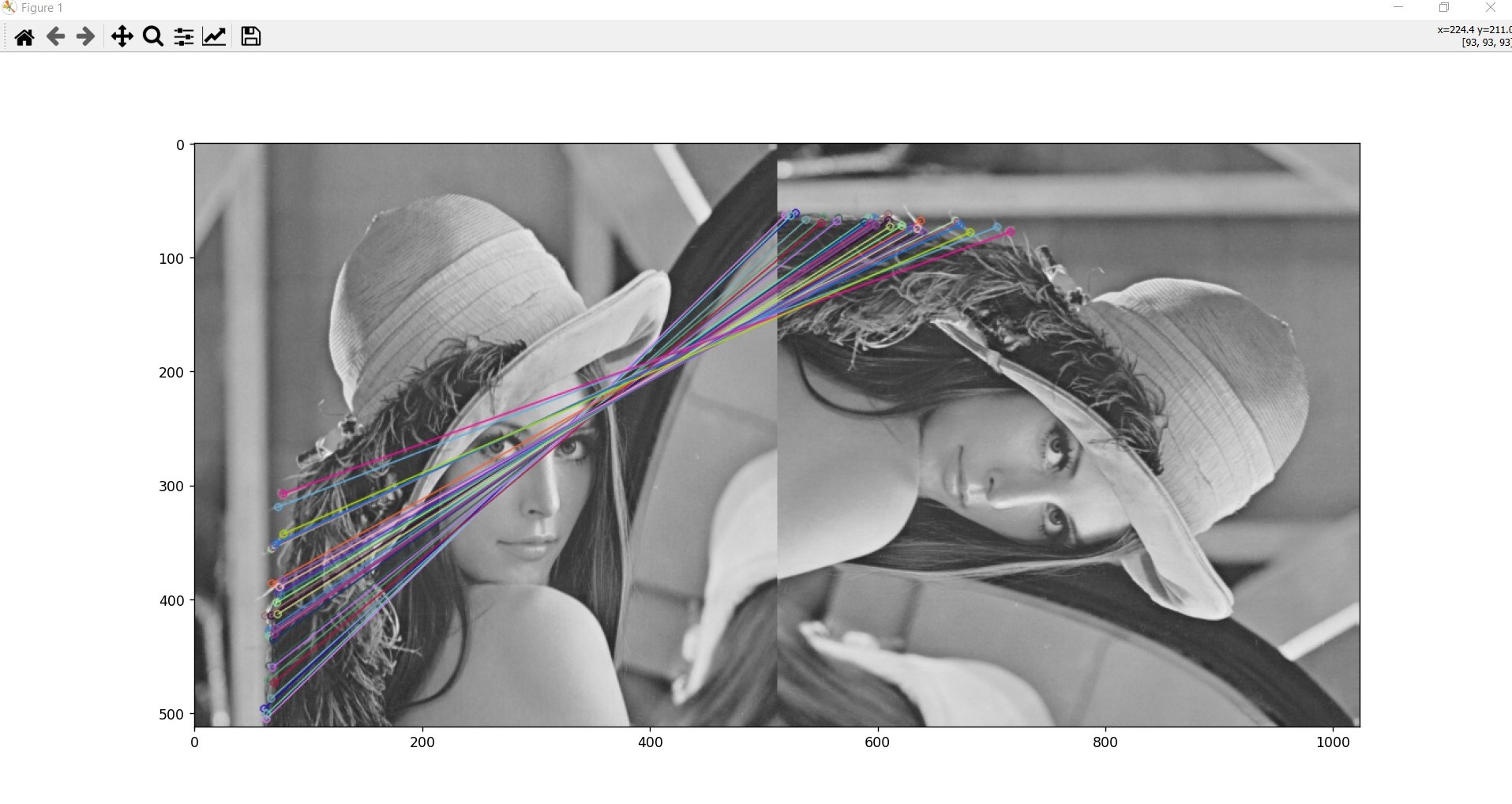


Image 2 SSD match

**Result**

The output is almost the same between the two methods

As the size of the image increases , computation time increases