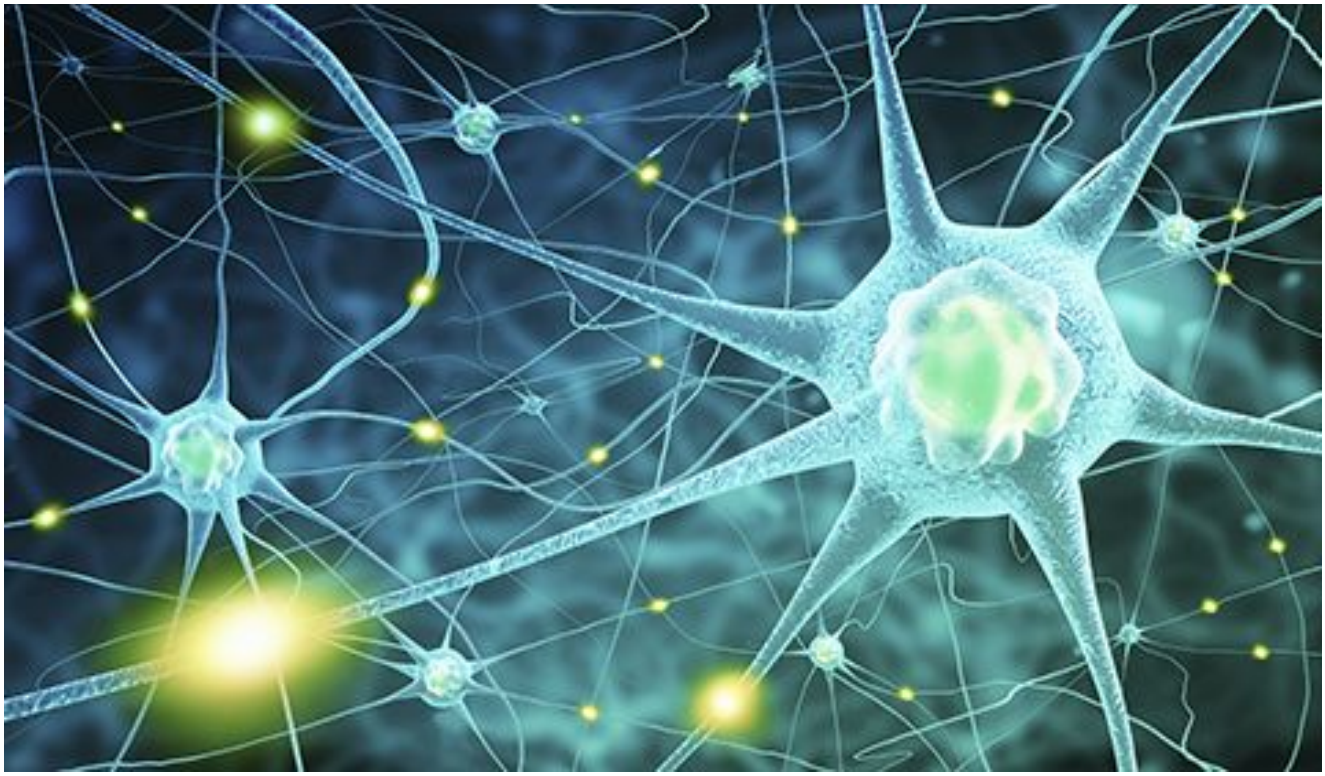
Assignment 2

Computer Vision - CS512



Akshay R

A20442409

Problem Statement:

Load and display two images containing two images containing similar content. In each image perform:

Corner detection:

- Estimate image gradients and apply the Harris corner detection algorithm.
- Obtain a better localization of each corner.
- Compute a feature vector for each corner point.
- Display the corners by drawing empty rectangles over the original image centered at locations where corners were detected.

Using the feature vectors, you computed match the features points. Number the corresponding points with identical numbers in the two images. Interactively controlled parameters should include: The variance of gaussian (scale), the neighbourhood size for computing the correlation matrix, the weight of the trace in the harris corner detector and a threshold value.

Proposed Solution:

The main method calls various functions to perform different activities when a keystroke is registered. We first input the images to be processed either by a file path or by using the camera. A preliminary check of the size of the image and converting the image to RGB channels will ensure the image can be processed to perform various actions. The image is converted to greyscale for smoother processing.

Gradients are calculated using the Sobel filters as computed in the previous assignment. The correlation matrix is calculated using the gradients. Determinant and trace of the correlation matrix are calculated to obtain cornerness measure using:

C(G) = determinant(C) - k * trace ^ 2(C)

This is stored in a list of lists. The pixels selected as corners are stored in a new list and rectangles are drawn for corner detection.

SIFT algorithm is used to detect and describe corners in the image . Feature vectors are calculated for both images and similar features are matched by taking a distance between both images. Similar features get too clumsy when they are close to each other, hence for better visualization lines are drawn to connect corresponding feature matches.

Parameters are controlled through trackbars.

Every special key is mapped to individual functions that in turn perform the required job functions. This mapping is through an infinitely running if else statement that breaks only when the escape key is pressed.

Implementation and methodology:

The program gives the user a choice to either input the images through a file path or directly from a webcam. The help bar displays the key bindings to perform particular actions:

Harris corner detection is computed through a function called **ownHarris,** I've used the openCV function to compare my implementation to the OpenCV implementation through a function called **cornerHarris.**

ownHarris is implemented as discussed in the proposed solution.

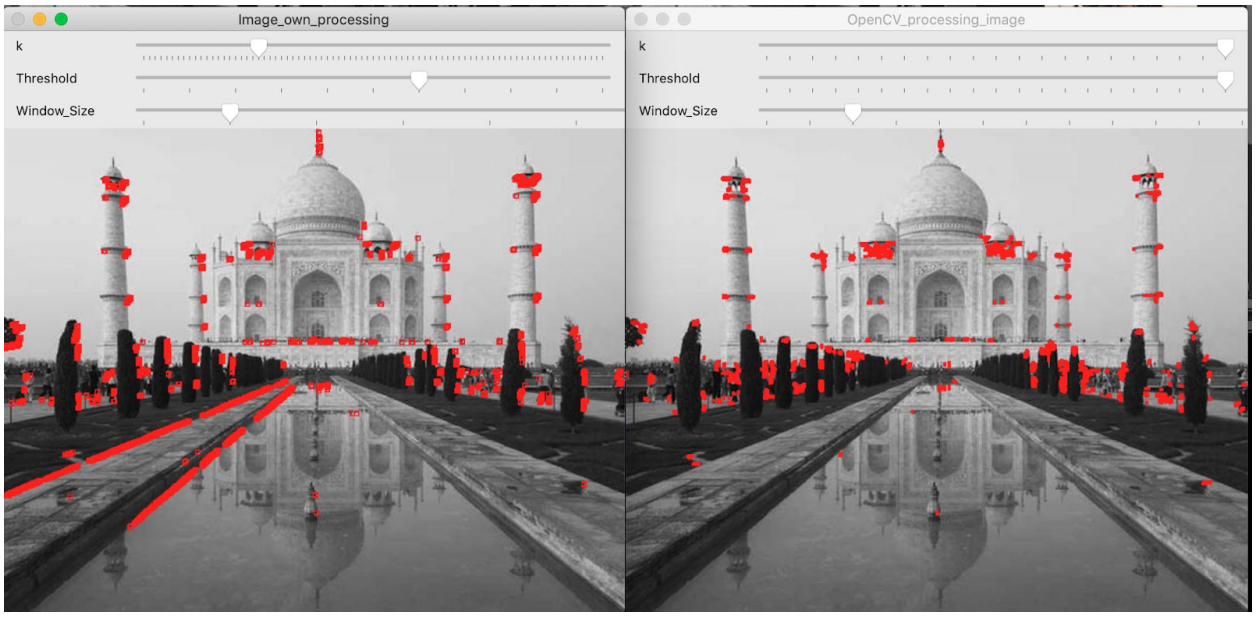
Better localization is computed through a function called **betterLocalization.** To perform better localization every point in the detected corner window is projected onto the gradient at that particular point where we test if it's a corner.This is done by using the formula. P = inverse(C)[gradient(x)gradient_transpose(x)]x. Where c is the correlation matrix.

Features vectors are computed and matching features in both images are displayed using a function called **fvec**. This method is implemented by first computing the gradients throughout the image and then performing a difference between both the images to obtain points that match. These points are highlighted and a line is drawn showing the pixels that match.

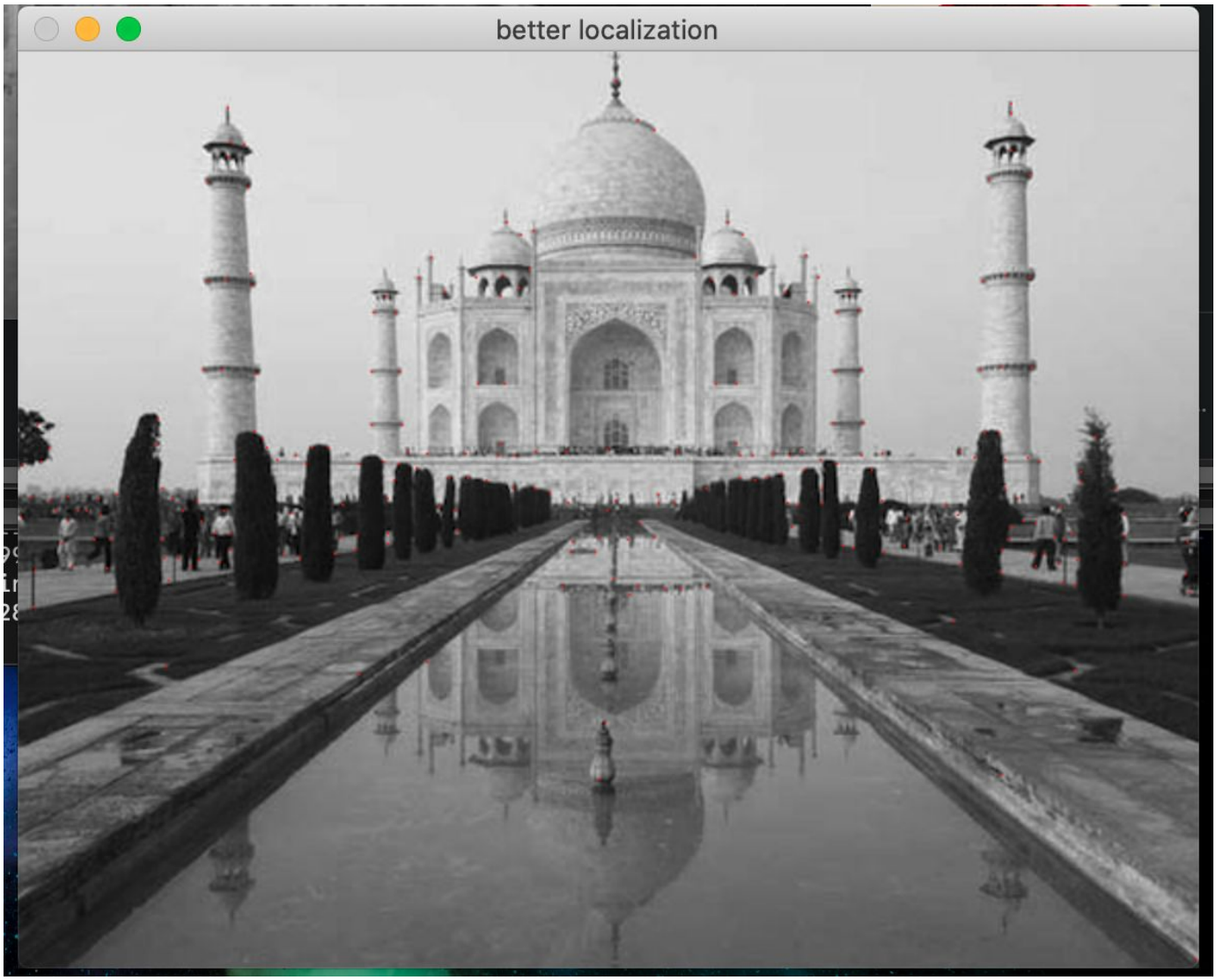
Difficulties faced: Viewing numbers while comparing features were difficult. So I implemented feature matching using lines. Gaussian smoothing must be changed in the program code and cannot be changed during runtime. I opted for this strategy due to the reason that selecting different gaussian smoothing filters were consuming a lot of time during runtime.

Results and discussion:

Capturing the results when ‘A’ and ‘a’ are pressed. The results of Own implementation and OpenCV are almost identical, although the variables seem to be different.

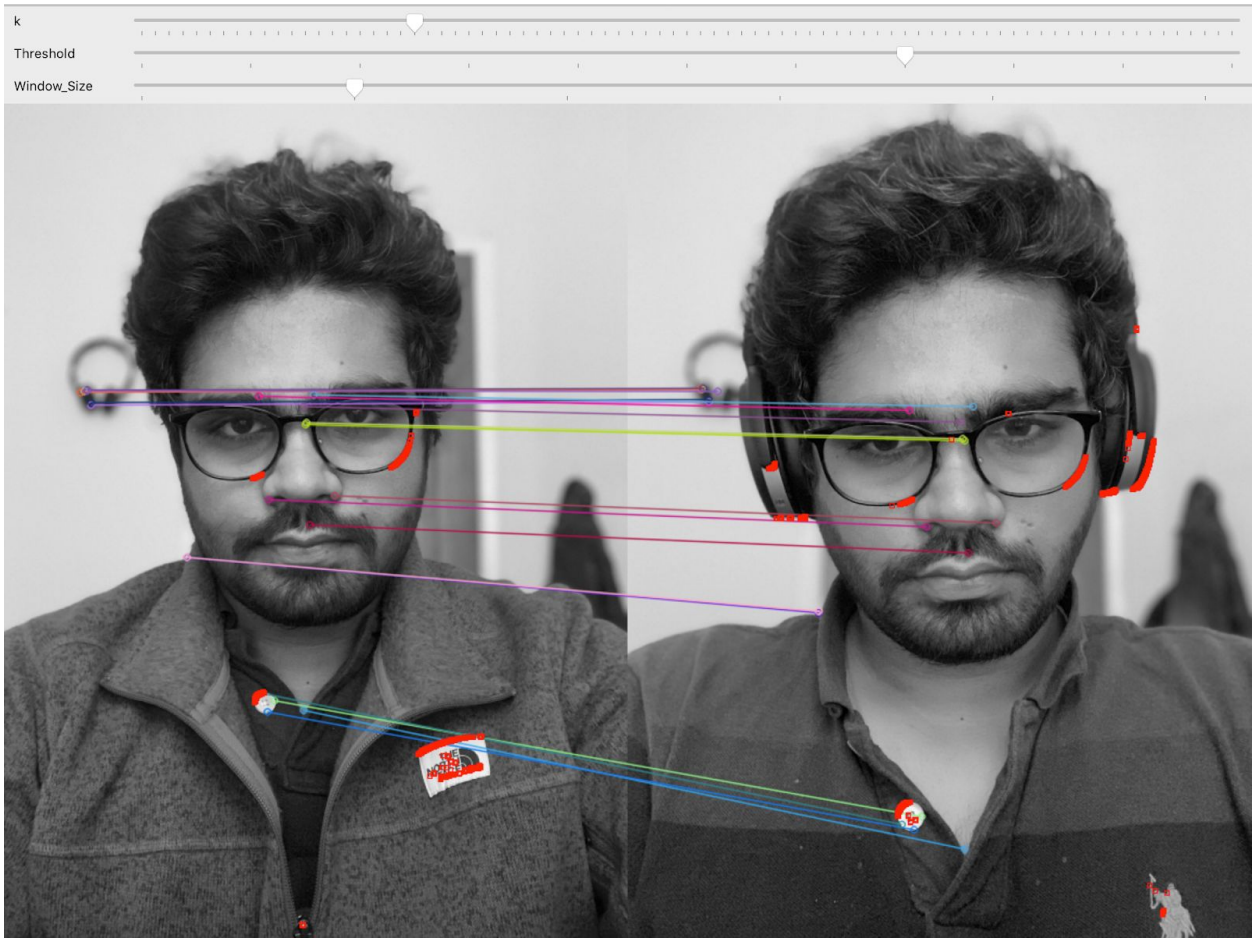


Better localization can be seen when ‘b’ is pressed. The red pixels represent better localized corners.



A zoomed in picture of better localized corners are as shown in the below image

Feature matching is done when ‘f’ is pressed. Similar colored lines represent matched features in both images. These are images taken on the webcam.



References:

- https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_core/py_basic_ops/py_basic_ops.html
- https://docs.opencv.org/3.3.0/da/d6e/tutorial_py_geometric_transformations.html
- https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/sobel_derivatives/sobel_derivatives.html
- https://docs.opencv.org/master/dc/dc3/tutorial_py_matcher.html
- https://kapernikov.com/object-localization-with-a-single-camera-and-object-dimensions/