Pablo Medrano

A20410758

cs512

Assignment 3

# Problem statement

The problem chosen to solve in this assignment is corner detection.

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

* 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 where detected.

## Using the feature vectors you computed match the feature points. Number corresponding points with identical numbers in the two images.

## Interactively controlled parameters should include: the variance of the Gaussian (scale), the neighborhood size for computing the correlation matrix, the weight of the trace in the Harris corner detector and a threshold value.

# Proposed solution

First of all, two images are loaded and converted to grayscale but keeping three channels so we can add colors in the future.

Then, to calculate image gradients a Sobel filter is applied both in x and y axis. Afterwards the values for Ix\*Ix, Iy\*Iy and Ix\*Iy are computed for each pixel in the image. These values are used to calculate the correlation matrix in each window of the image.

Once the correlation matrix is computed, its determinant and trace can be calculated to be used to compute the C(G) values for each pixel which are stored in list of lists along with each pixel position.

Afterwards, the maximum value for C(G) in the list is found to be multiplied by a constant to use it as threshold to find the corners.

The pixels selected as corners are stored in a new list and then the rectangles are drawn around those pixels.

In order to compute the feature vector, a brute-force matching with ORB descriptors using SIFT is used.

This method calculates the feature vectors for both images matches similar pixels in both images depending on the distance.

They are sorted with those with less distance first that are the ones that are drawn in the images connected with lines between them.

The parameters are controlled through track bars in the image. The variance of the Gaussian is the only one that it is not controlled since it is applied when the Sobel filter is applied and it can be controlled in that function.

# Implementation details

I had a few problems with the eigenvalues I was getting because they were too small and I did not know why since I was using an openCV function called cornerEigenValsAndVecs that returned the eigenvalues and eigenvectors of the correlation matrix for each pixel given a window size.

I was stuck with this problem for a couple days because I did not know how to fix it.

At the end, Xu, the TA told me it was because I had to transform the image to float32 but I had already programmed a different method without using that function.

After that I was able to find some corners and create the track bars that control the parameters.

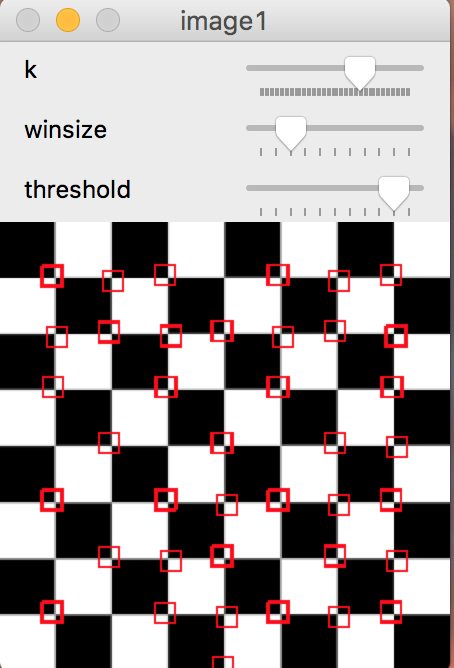
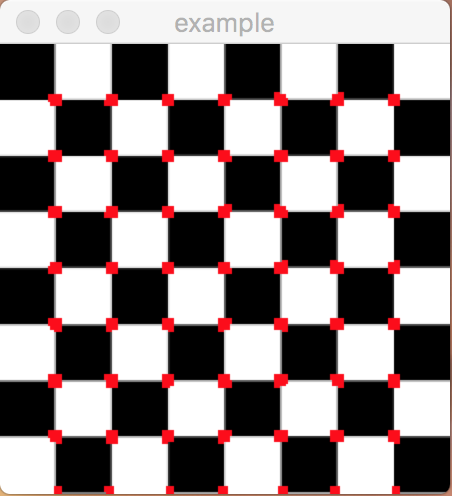
As I said I did not create a track bar to control the Gaussian smoothing because I apply a Sobel filter on the image that already has a Gaussian in it and the smoothing cannot be control in it. I could have applied a Gaussian filter myself as in the assignment before to be able to control the smoothing but it seemed redundant to me.

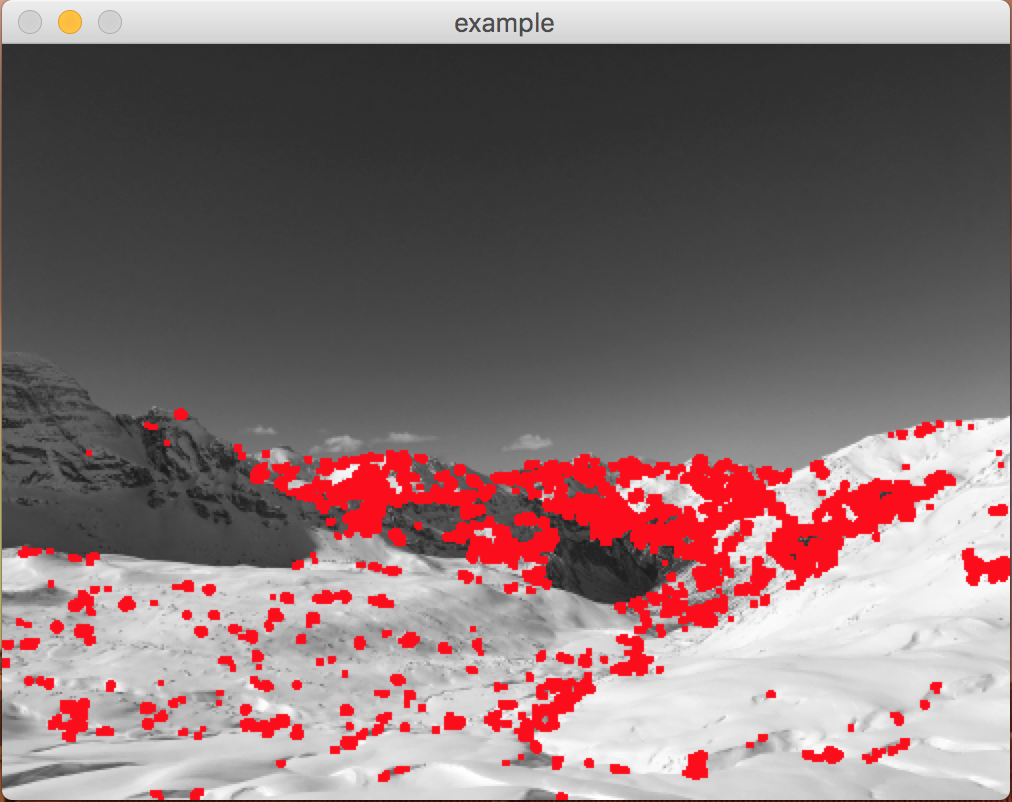
Finally, I was looking for a way to compute the feature vectors myself so they could be more similar to the corners found before with my coding but I could not find information on the internet on how to do it. Then I saw on blackboard that we could use the openCV functions to compute the feature vectors and it is what I did.

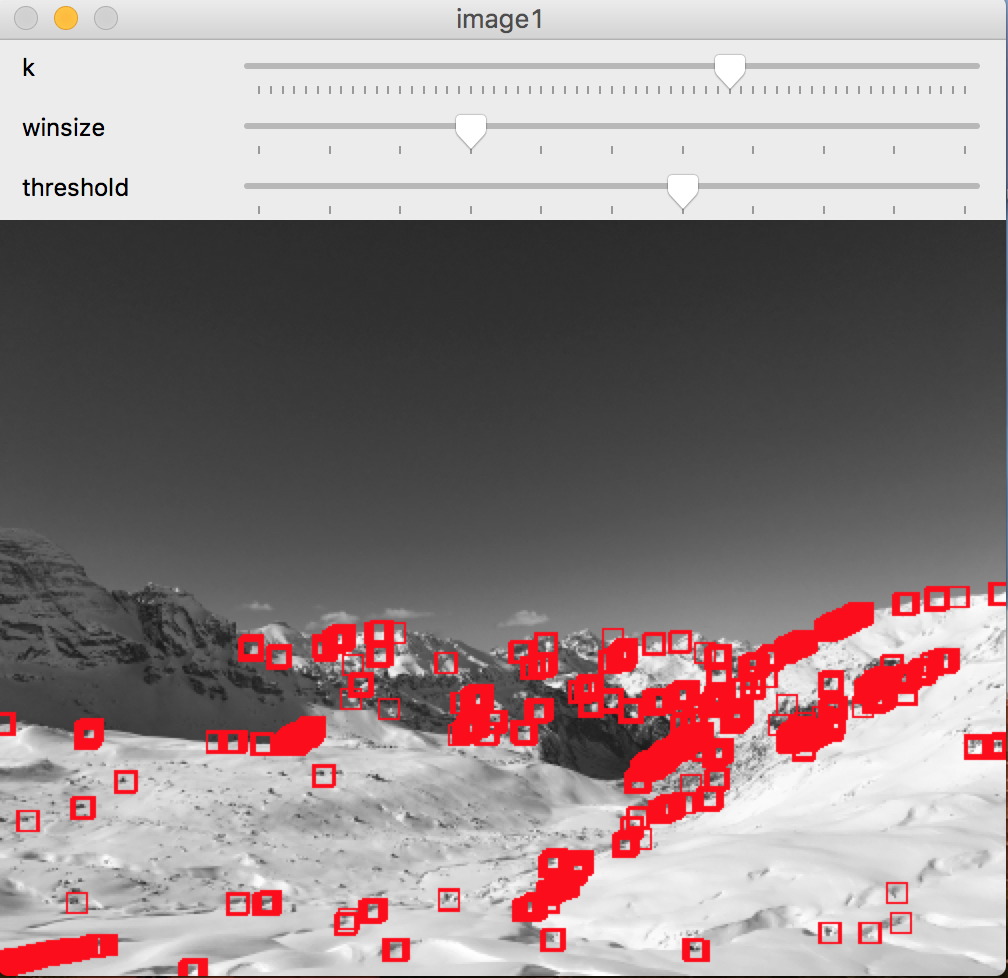
# Results and discussion

At the beginning I started computing the corners with my algorithm for several images and comparing the result to the actual Harris corner detection openCV function to see how was the performance of my function.

Here are some examples, the ones without the track bars are the ones computed using the openCV function and the ones with the track bars are the ones computed using my algorithm. I used more than these examples but I just show a couple of them.







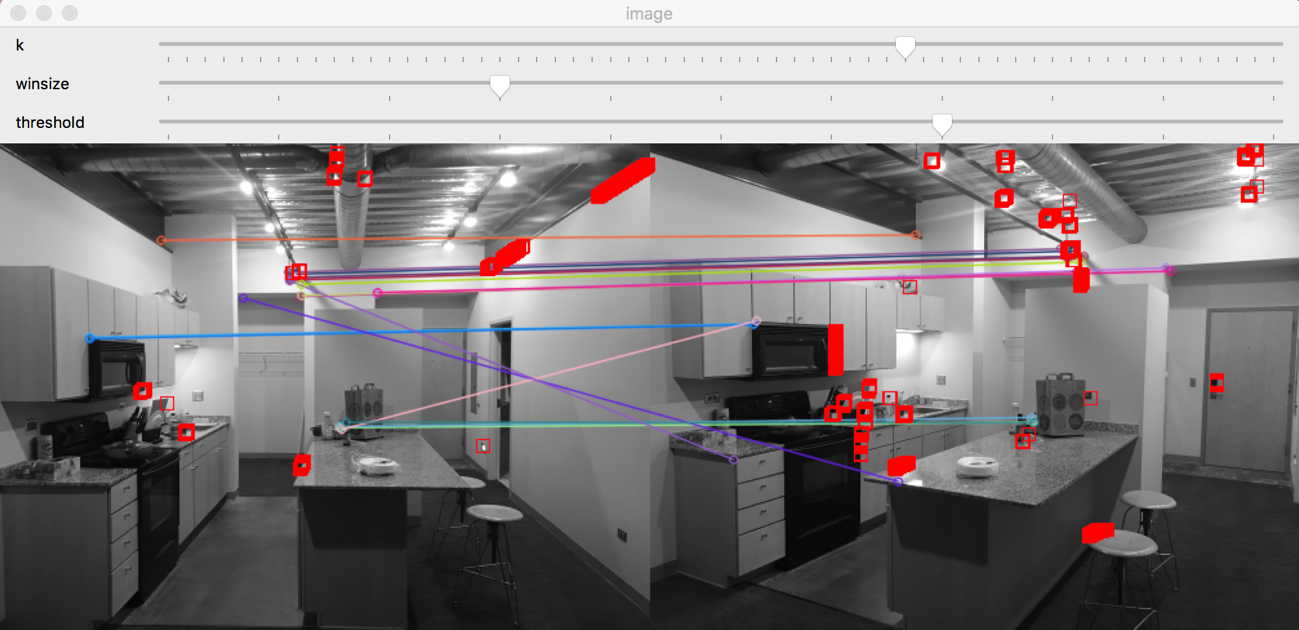
As it can be seen there are some differences but my algorithm does not perform too bad.

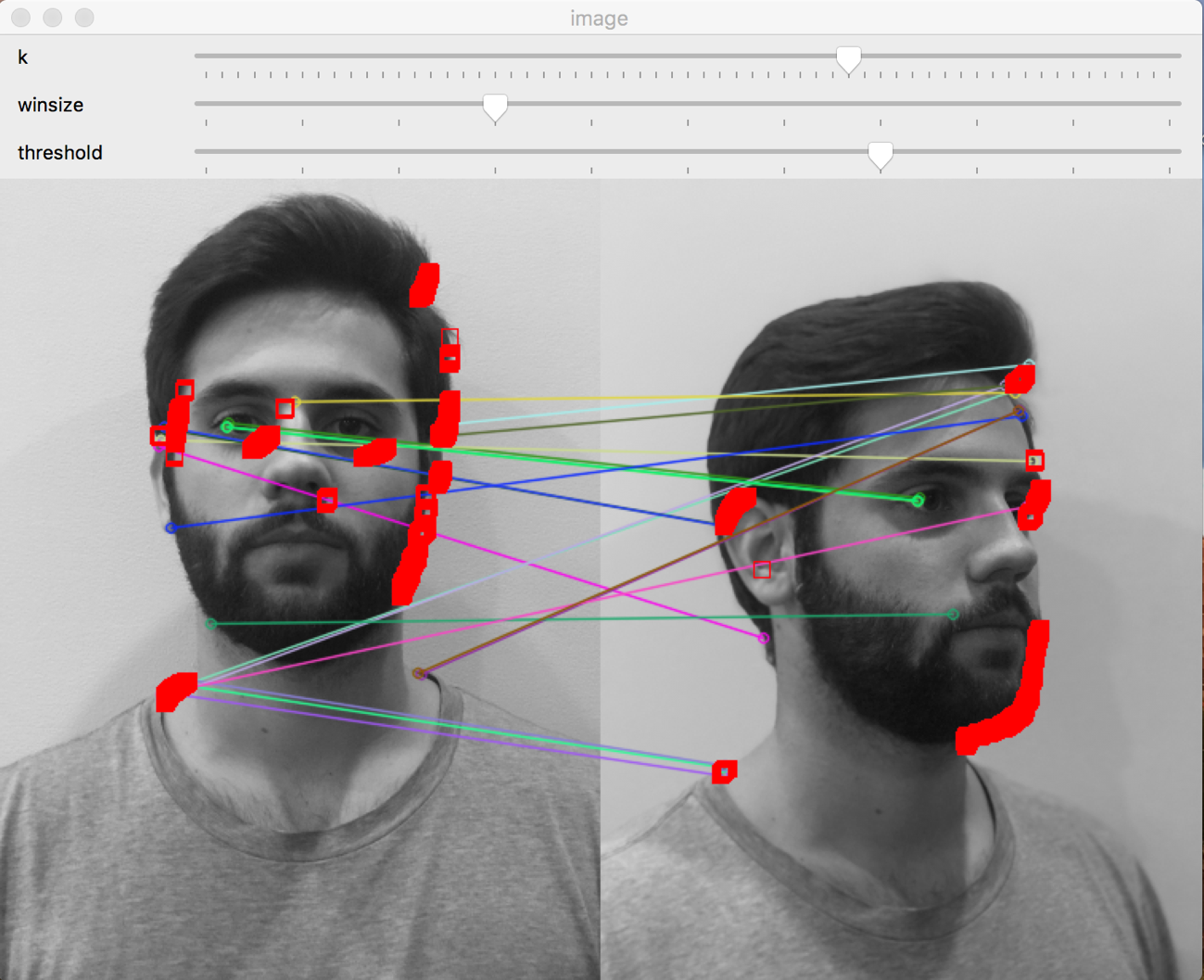
One thing I noticed is that the threshold used with the openCV function is quite smaller than the one I used in my algorithm and I do not really know why if the Harris corner detection openCV function should be similar to what I am doing to find the corners.

Also varying both the three parameters the numbers of corners found can be adjusted

After checking that my algorithm worked I went on and compute the corners and the feature vectors for two similar images and displayed them in the same window.

Here are a couple examples:





Most of points shown match by the feature are the same ones that my algorithm found but some others are not since openCV function can compute the corners a little different. However, I am showing much more corners in the image than the ones matched so not all of them should be matching others.

I think in overall the performance of the algorithm is good compared to the openCV functions.

# References

<https://docs.opencv.org/2.4/modules/highgui/doc/user_interface.html>

<https://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html>

<https://pythonprogramming.net/feature-matching-homography-python-opencv-tutorial/>

<http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html>

<https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_matcher/py_matcher.html>

<https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html>

<https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.concatenate.html>

<https://docs.opencv.org/3.0-beta/modules/imgproc/doc/feature_detection.html>