CSE 455 Homework 5
FEATURE DETECTION, DESCRIPTION AND MATCHING
Name: David Wong

1. Introduction

This project is designed around implementing the Harris corner detector and performing feature matching between a base image and similar-looking sibling images. First we use the Harris corner detector to identify points of interest in an image, and these features are extracted to create a feature set. After we have feature sets for two images, we compare them to match features between the images. The program returns the match percentage, with a higher percentage indicating better performance.

2. Feature detection, descriptor and matching

The first step for this project is to implement the feature detection. In the Harris detector, the original image is transformed into its x- and y-derivative images I_x and I_y. So after applying a Gaussian filter over the image, we convolve the subsequent blurred image with the provided dx- and dy-kernels. We then calculate the corner strength R for every pixel in the image. R is determined by the function $R = Det(M) – k*Tr(M)^2$, where $Det(M) = I_x^2*I_y^2–(I_xI_y)^2$ and $Tr(M) = I_x^2+I_y^2$. If the R-value is above a certain defined threshold (in this case threshold=0.00001) and is the local maximum in a 3x3 neighborhood, then we select the feature for extraction.

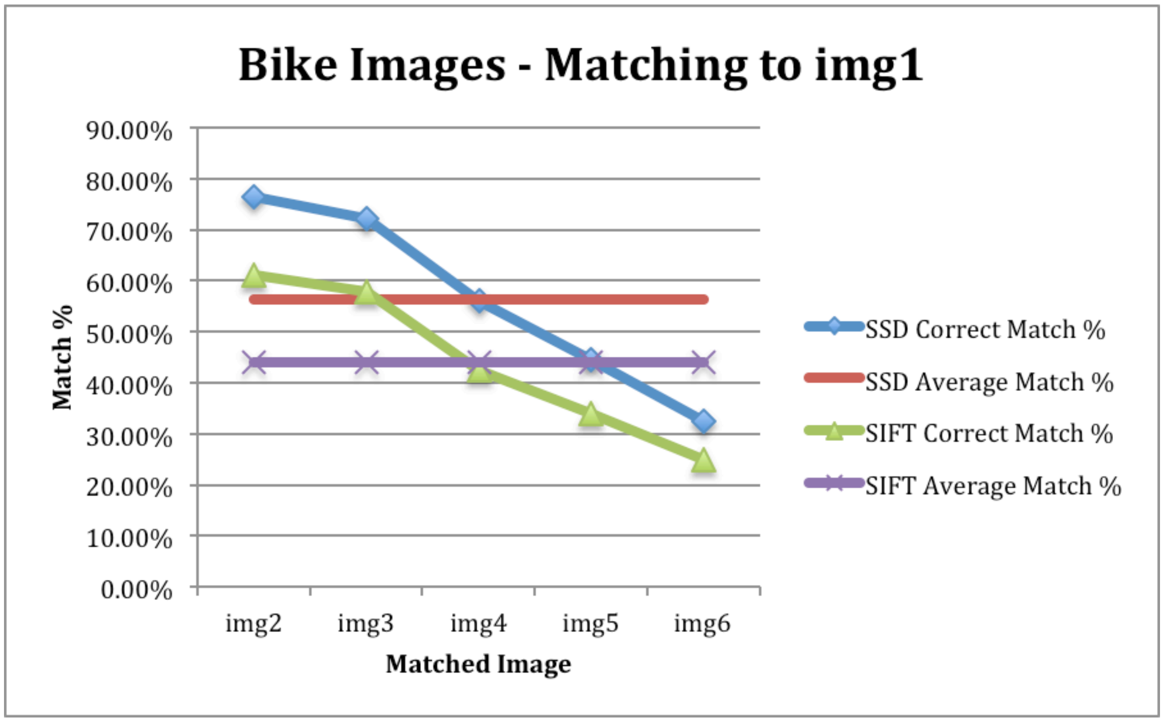
The feature detector creates a descriptor of 243 values computed from a 45x45-pixel window around the feature point. The 45x45 window is divided into 9x9 sections of 5x5 pixels each. After applying a 5x5 Gaussian filter over each of the channels (R,G,B) of these pixels, the resulting values are summed for each channel and added to the feature descriptor vector. The vector is normalized by the L2 norm, which is equal to the square root of the sum of the square of all the values in the vector. The resulting feature descriptor is added to the feature set.

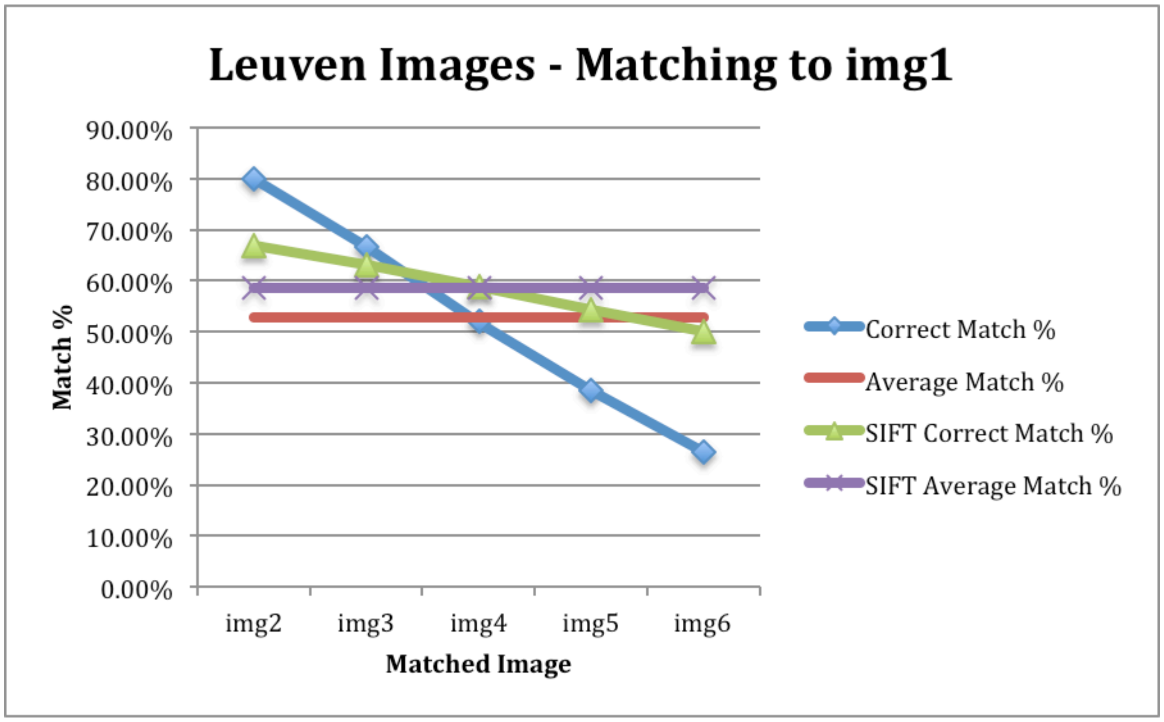
Feature matching is already implemented in the provided code, and the process involves taking two feature sets and matching features of the first image with features extracted from the second image. Every feature of the first image is paired with the most similar feature of the second image (having the shortest Euclidean distance between the two feature descriptors). These pairings are returned and evaluated to determine matching effectiveness.

Some changes were made to the features.cpp code beyond the ComputeHarris() and ExtractDescriptor() functions. The dxKernel and dyKernel functions were modified to reflect the kernels used in the Harris detector examples posted on the class webpage (-1, 0, 1 → 1, 0, -1). Additionally, the border value in dummyComputeFeatures() has been changed from 20 to 22 to accommodate the 45x45 window needed in feature descriptor extraction. The 5x5 Gaussian filter is also applied in this function with the resulting image being passed into ExtractDescriptor(). Having the filter applied at this step means that the filter only needs to be applied once. Lastly, the R-value threshold is checked in the if() statement right before ExtractDescriptor() to determine if the feature should be considered.

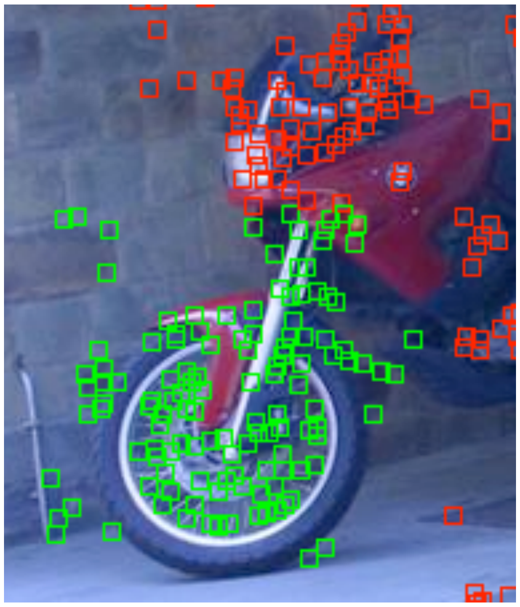
3. Experiments

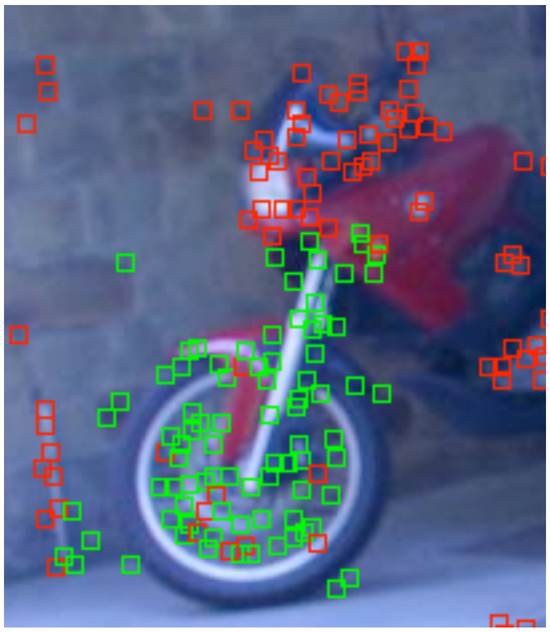
To test the matching performance of the feature detection, I ran the benchmark for both the “bikes” and “leuven” image sets. As predicted, matching performance decreased as the image number increased. I also ran the provided SIFT descriptor for comparison, and I graphed the results against the numbers from my feature descriptor.

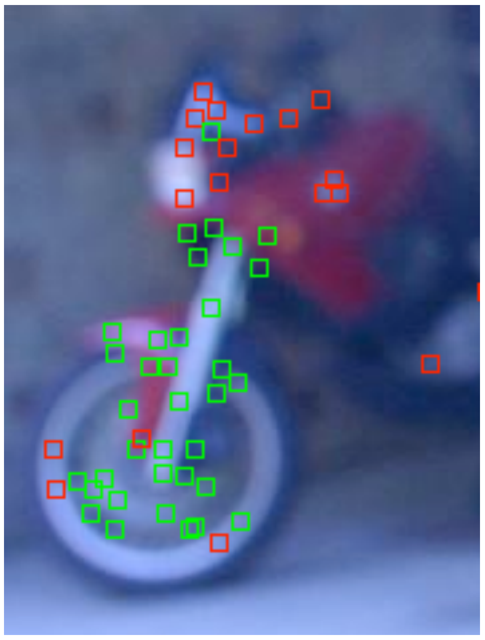




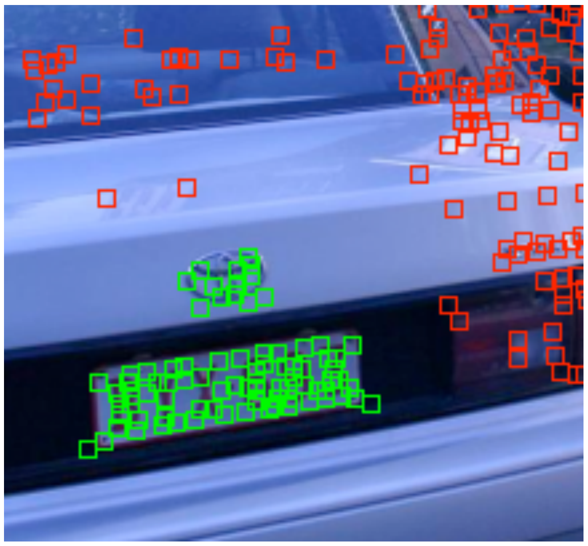
The results show that my feature descriptor matched better on the “bikes” image set, but worse on the “leuven” image set. My descriptor performed much better in leuven/img2, similarly on leuven/img3 and leuven/img4, and significantly worse on leuven/img5 and leuven/img6.

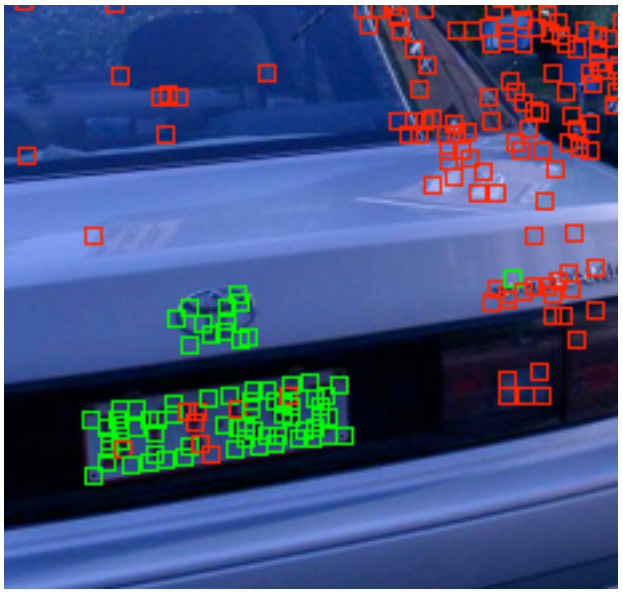

bikes/img1.ppm

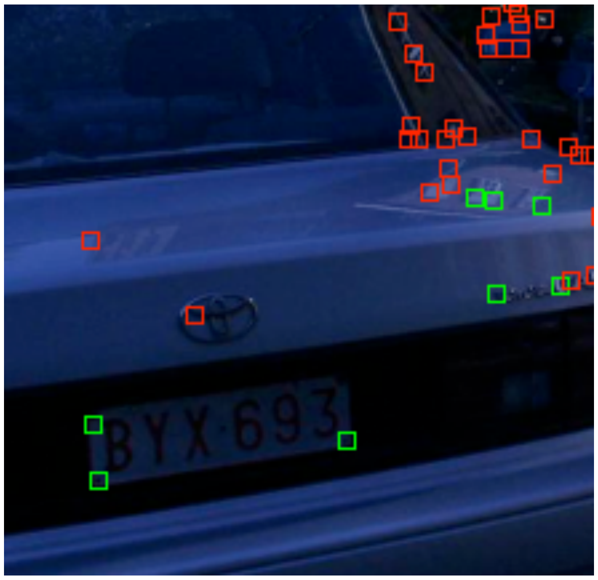

bikes/img2.ppm with simple matching

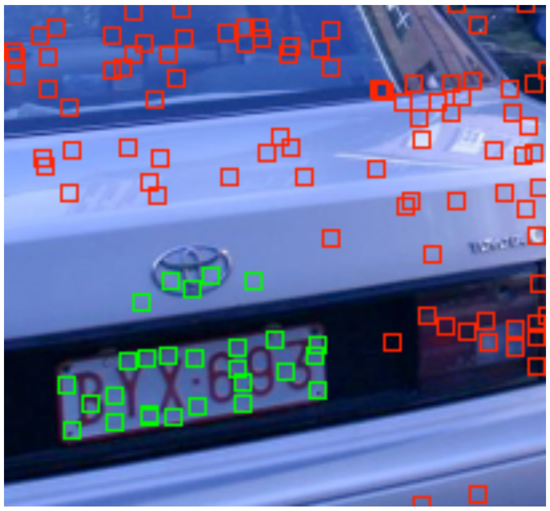

bikes/img6.ppm with simple matching

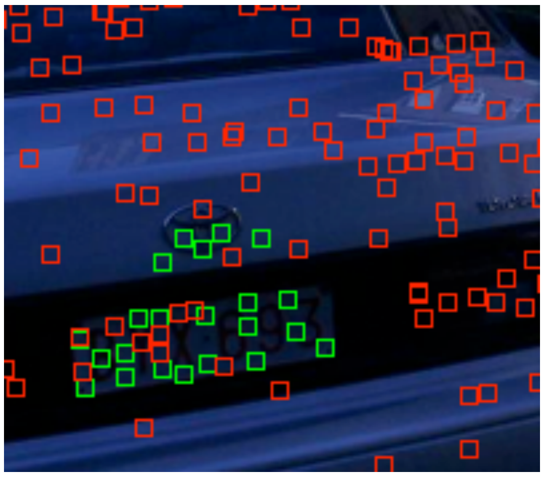
Good feature detection and matching despite bikes/img6 being a severely blurred-out version of bikes/img1.


leuven/img1.ppm


leuven/img2.ppm with simple matching


leuven/img6.ppm with simple matching


leuven/img1.ppm provided features


leuven/img6.ppm with sift matching

It appears that the SIFT descriptor is better at detecting and matching on images that are significantly different from the initial image (i.e. leuven/img1 with leuven/img6). The detector I implemented does not detect many features in the selected region (license plate region), thus performing poorly in the matching.