Feature Descriptor

My feature descriptor (PLUS) is designed to be robust in aspects of illumination and orientation.

At each pixel location determined to be a Harris-maxima, I sample the 5x5 square grid (with the said pixel in the center), and determine the values of the least R, G, and B values which could be from a similar or multiple pixels. The 5x5 square is my pixel window.

Then, I subtract the RGB values of all pixels within this pixel window with the values of the least intensities which were previously determined. I then sum the RGB intensity values of each pixel, and then multiply it with its Gaussian ratio as described below:

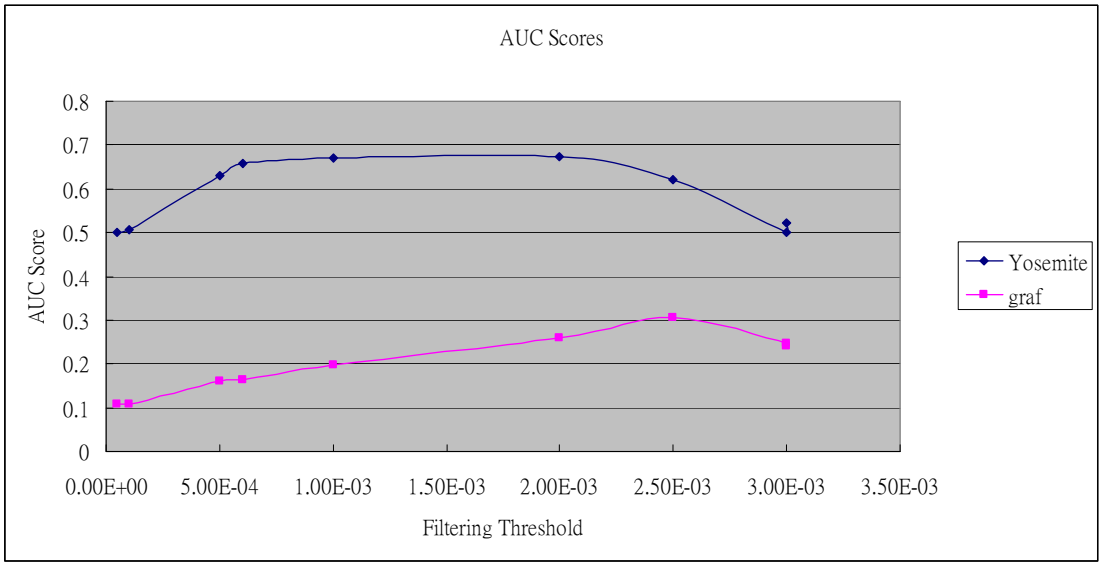
$$Value[i,j] = (R[i,j] - R_{min} + G[i,j] - G_{min} + B[i,j] - B_{min})*Gaussian5x5[i,j];$$

where *Value* is the feature value of a pixel at (i,j), $X[i,j]$ is the intensity value of color channel X, X_{min} values are minimums determined for each 5x5 feature window and $Gaussian5x5[i,j]$ gives the Gaussian approximation for pixel (i,j) relative to the Harris-maxima as the center.

Subtracting away the minimum intensity values for each of the color channel allows my feature descriptor to be robust in terms of illumination in any of the three color channels, and using a Gaussian blur allows my features to be more robust against slight changes in orientation/rotations.

Design Decisions

There is a great deal of flexibility in determining various threshold values as well as how one could design the feature descriptor. The design of PLUS and the establishment of various thresholds are determined empirically by testing the effects of various features, like robustness in illumination, are set against the AUC score of the picture sets *Yosemite* and *graf*. The threshold set to filter-off insignificant local maximas in *computeLocalMaxima(…)* is a good example; I have determined the value of 2.2E-3 gives the best performance compromise between the AUC scores of the two picture sets from the following graph:



AUC Scores vs. Filter-threshold

It is not hard to see that the choice of this threshold value ought to be set between 2E-3 and 2.5E-3.

As mentioned, the design of PLUS is also determined by this metric. Certain features are favored/disfavored by certain image matches paired with different matching algorithms. For example, the *Yosemite* set is Gaussian-blur adverse, and favors the use of gray-scale values instead of RGB values in the feature-descriptor window. Eventually I decided to use RGB as the *graf* set showed significant average improvement with the use of RGB values instead of gray-scale.

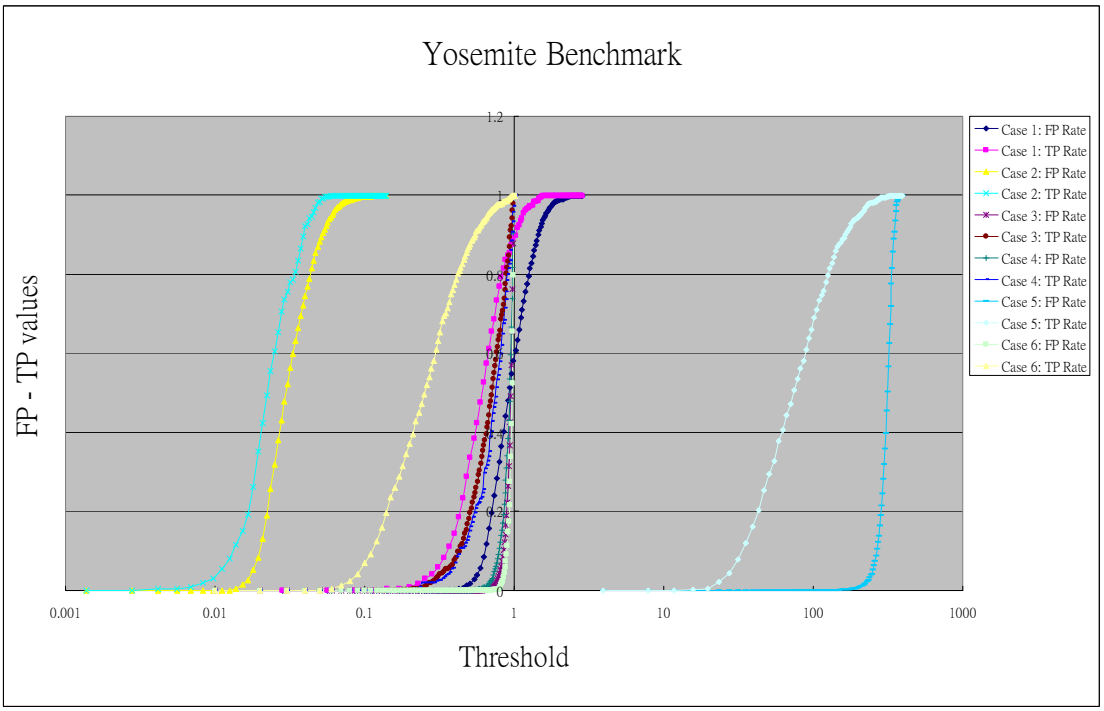
I considered serious between the use of RGB values and or just a single gray-scale value for the feature-descriptor. Reading just the gray-scale value from the already produced gray-scale image is significantly cheaper than reading 3 color channels and

also having to detect the minimum R, G, and B values separately in PLUS. However, given that a gray-scale image is composed of the weighted intensities of the RGB channels using weights which are most probably aesthetically determined, a gray-scaled intensity implemented feature descriptor will, in a sense, be only robust against a fixed composition of RGB values. Using all three colors at the expense of some computation, I believe is worth it.

From the average AUC scores, PLUS seems to perform pretty well.

Benchmark Graphs

Yosemite

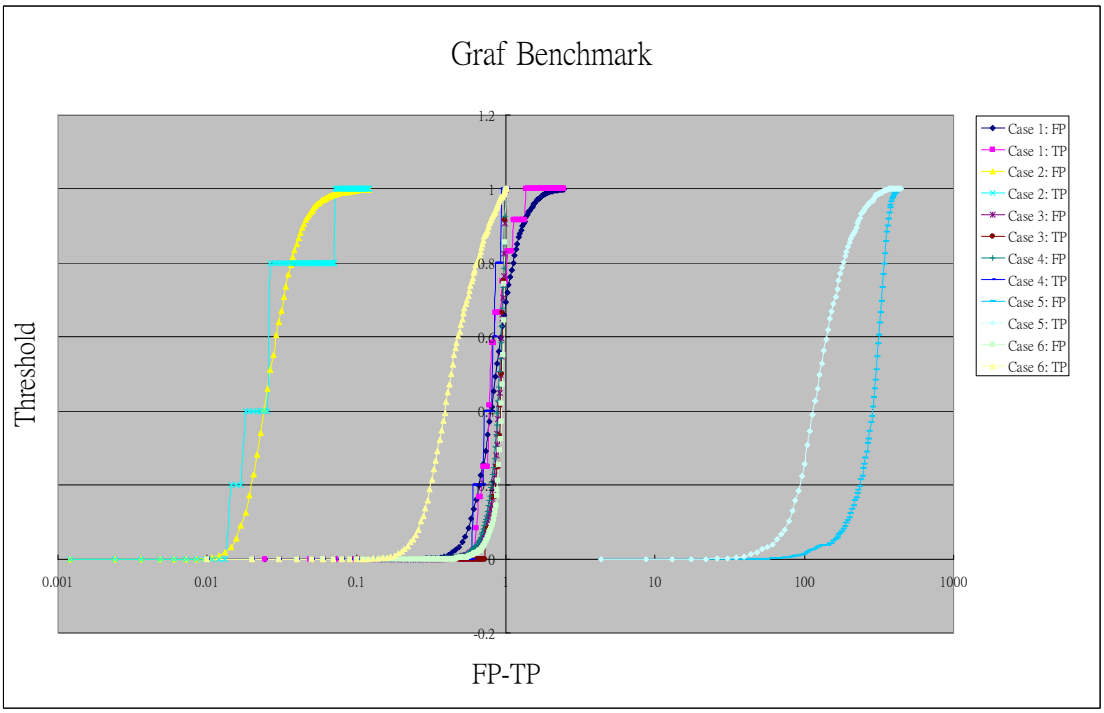


Case 1: Simple window descriptor + SSD distance
Case 2: PLUS descriptor + SSD distance
Case 3: Simple window descriptor + ratio-test distance
Case 4: PLUS descriptor + ratio-test distance
Case 5: SIFT + SSD distance
Case 6: SIFT + ratio-test distance

AUC Score:

Simple window descriptor + SSD distance	:	0.807894
PLUS descriptor + SSD distance	:	0.706741
Simple window descriptor + ratio-test distance	:	0.877379
PLUS descriptor + ratio-test distance	:	0.811823
SIFT + SSD distance	:	0.994692
SIFT + ratio-test distance	:	0.995494

graf



Case 1: Simple window descriptor + SSD distance
Case 2: PLUS descriptor + SSD distance
Case 3: Simple window descriptor + ratio-test distance
Case 4: PLUS descriptor + ratio-test distance
Case 5: SIFT + SSD distance
Case 6: SIFT + ratio-test distance

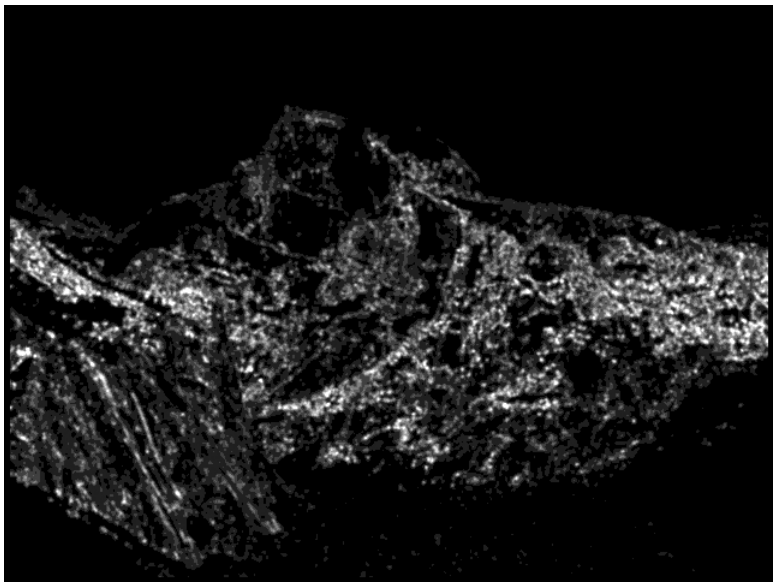
AUC Score:

Simple window descriptor + SSD distance	:	0.668725
PLUS descriptor + SSD distance	:	0.597490
Simple window descriptor + ratio-test distance	:	0.664417
PLUS descriptor + ratio-test distance	:	0.561607
SIFT + SSD distance	:	0.932023
SIFT + ratio-test distance	:	0.967779

* I chose my heuristic parameters using the *graf* and *Yosemite* sets. My descriptor, combined with the ratio-test matcher gets a score of 0.763966 for matching img1.ppm to img3.ppm in the *graf* set, up from a score of 0.455041 using a regular window descriptor.

Harris Images

Yosemite2.jpg



Harris (+ Auto-contrast)

MaxHarris (+ Auto-contrast)

img4.ppm (graf set)

Harris (+ Auto-contrast)

MaxHarris (+ Auto-contrast)

Average AUC Scores

graf:

Simple window descriptor + SSD distance	:	0.63
PLUS descriptor + SSD distance	:	0.59
Simple window descriptor + ratio-test distance	:	0.51
PLUS descriptor + ratio-test distance	:	0.66

leuven:

Simple window descriptor + SSD distance	:	0.28
PLUS descriptor + SSD distance	:	0.23
Simple window descriptor + ratio-test distance	:	X
PLUS descriptor + ratio-test distance	:	X

bikes:

Simple window descriptor + SSD distance	:	0.25
PLUS descriptor + SSD distance	:	0.15
Simple window descriptor + ratio-test distance	:	0.49
PLUS descriptor + ratio-test distance	:	0.68

wall:

Simple window descriptor + SSD distance	:	0.23
PLUS descriptor + SSD distance	:	0.44
Simple window descriptor + ratio-test distance	:	0.61
PLUS descriptor + ratio-test distance	:	0.66

The program crashes when my script gets to some matches of the *leuven* set. This is very peculiar given that all other benchmarks worked. I am unable to trace the source of this problem, but I suspect it is that standard-library issue once again.

Strengths and Weaknesses

There is some incompatibility between my system and FLTK. I am not unable to run the solution executable provided, nor am I able to fully utilize the GUI from a program that was built on my computer (Jiun-Hung identified some standard library issues). This also implies that I cannot meaningfully create visuals to demonstrate that my program works. ☹ - I cannot create artifacts.

I do not have a benchmark to compare against; if the solutions EXE ran, I would be able to say with greater confidence that my program performs pretty well (the SIFT features were matched to a very high score, that much is certain). From the average results, it is interesting to note the difference in score between the SSD-distance match and the ratio-test match; PLUS seems to be very compatible with the ratio-test algorithm.

Overall I think it works. ☺

Thanks.