**Creating Local Feature Descriptor**

**I**n this project I build and test Local FP (**F**eature **P**oint) descriptor.

First in order to localize FPs I use multi-scale corner detector. There are 2 options: 1) Harris corner detector ()

2) Harmonic Mean () (which, by my opinion, is more preferable). Also localization of FP is calculated with sub-pixel accuracy, with Taylor expansion/approximation.

**T**hen with Normalized Laplace operator I find 'characteristic scale' for each FP. If for FP there is no char. scale - it's rejected. (‘charac. scale' - it's a scale where Laplace operator receives local maximum, and also at this scale FP is a 'corner'. Characteristic scale gives to descriptor invariance to scale/zoom).

There are a lot of parameters, that allow to influence different factors: it's possible to define neighborhood for local maximum - it can be some constant or depends on scale, define range where to find FP (to avoid situation that FPs will be near borders), define range of scales where to search characteristic scale, dilate with some radius /parameter/ FP, so each FP will be only one in disk with given radius, disk center - this FP.

**A**fter that charac. scales were found - for all scales image is blurred with them. All FP with specific char. scale are taken with their neighborhood of size according to their scale in blurred image / image is blurred with this specific scale – which is standard deviation of blurring Gaussian /. For each FP in those neighborhoods calculated derivatives and with weighted histogram defined MO (**M**ain **O**rientation). This process is similar to Lowe's SIFT descr. It's possible to influence this process: define size of angle bins, if to take only those FP that have one very supported direction ( with thresh can be influenced this ) or to allow for FP to have more then one MO, that are bigger then some thresh.

FP without MO are rejected (in case if was chosen option - only one MO, possible that there won’t be one very supported direction, will be some).

**A**fter MO is calculated - SIFT-like descriptor calculated for each pair of FP and MO (in case if was chosen possibility for FP to have more then one MO). The difference with SIFT is - when were calculated derivatives and theirs’ orientations in neighborhoods of FPs, for description are taken samples of them, that are taken in direction of MO near each FP. Step of sampling was taken dependent on characteristic scale - / there is a function, that specifies step, as function of scale, it's possible to take regular SIFT descr. if specify function to be const 1 /. Also one of possible tunable options: can be specified number of windows for description, size of angle bin.

**F**or matching FPs there were implemented several options.

Matching can be done by: 1) Minimal distance 2) Best Match/Second Match

3) Best Match/Second Match and threshold distance of matched FP.

For search for matched FP, implemented several options:

1) There are 2 options for strict search / second is quicker, but more memory consuming /.

2) 'kmeans' - kmeans algorithm is run first, then matches are searched only in cluster of FP.

3) 'kNN' - k-nearest search, implementation was taken from Matlab file exchange site / haven't noticed that it's quick, decided to left it, as additional option /.

**A**lso was implemented descriptor **ASIFT**. **ASIFT** simulates viewpoints of image(s), calculates SIFT-like descr., and then compares all FP.

**F**or displaying results / of matching / there are 2 options. 'Aff' - show only location of FP in both images and connects matched with lines. 'HL' (Harris Laplace) - show FPs, their 'scale' (rectangle with edge length proportional to scale), MO (rectangles are rotated in direction of MO), connects matched FPs with lines.

Also there is a function **PlotFP.m** for displaying finded FPs.

**R**esults:

1. *Here example of FP that have been calculated*:

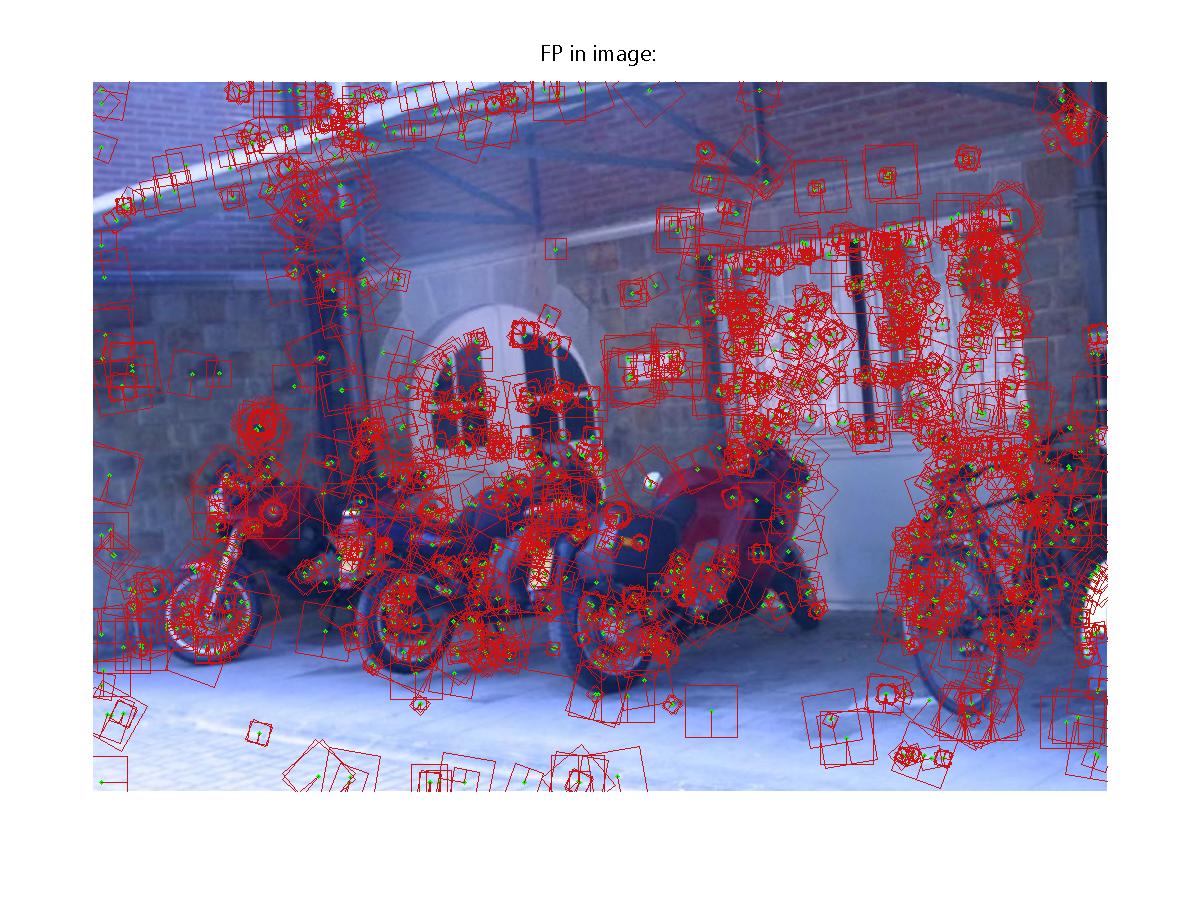


Figure 1: Fp’s in image with their characteristic scale and main orientation.

1. *Scale- blurring-rotation invariance*:



Figure 2: Original compared images.

Figure 3: Original correspondences.

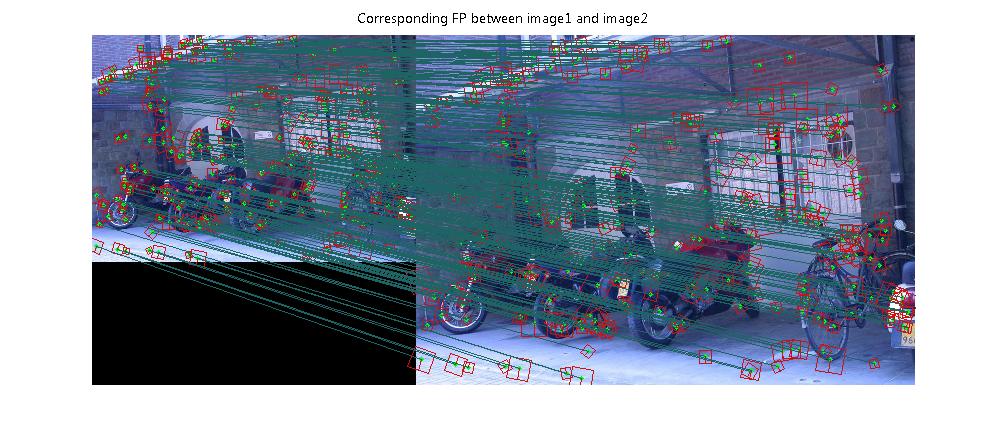


Figure 4: Scale invariance: second image scaled and compared with itself.

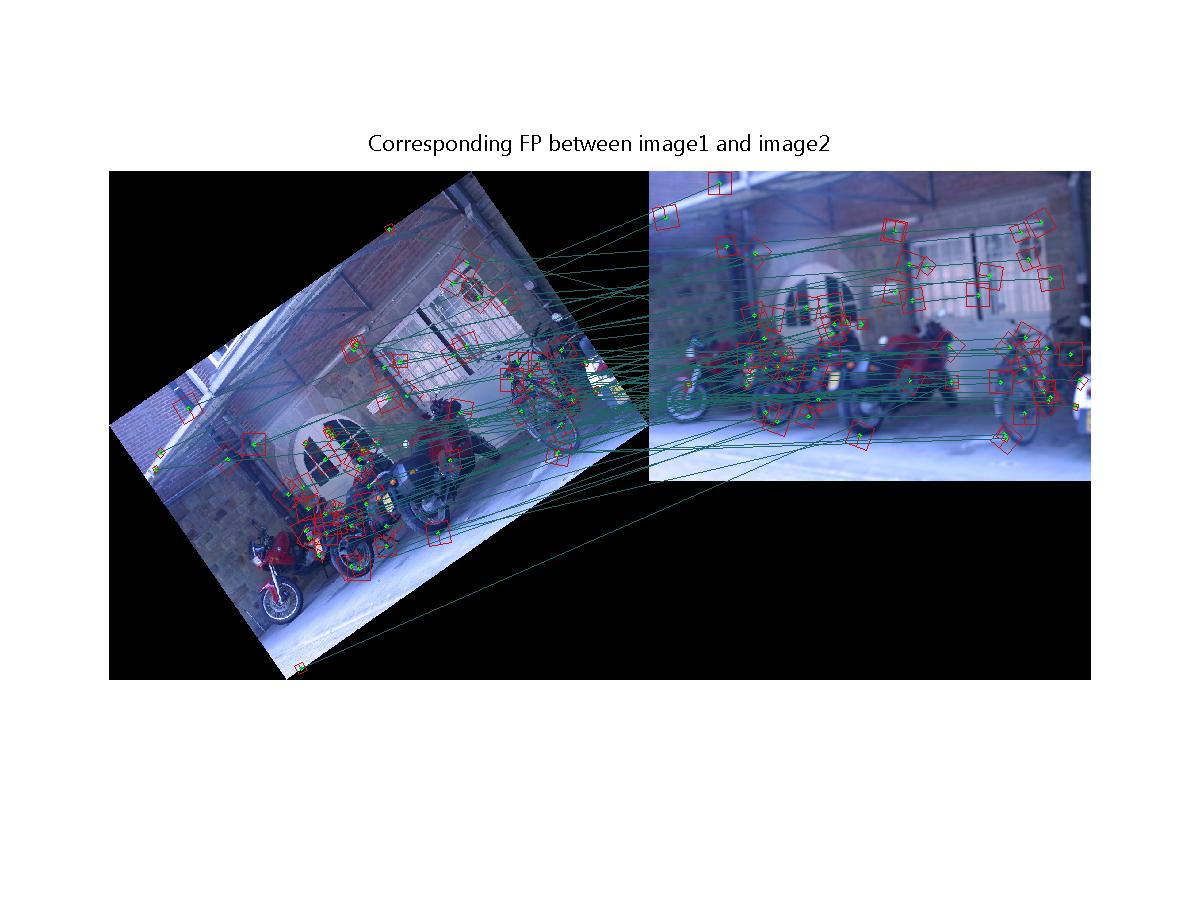


Figure 5: Rotation invariance: second image rotated and compared with first.



Figure 6: Compared images, first image scaled, rotated, blurred and slightly shifted.

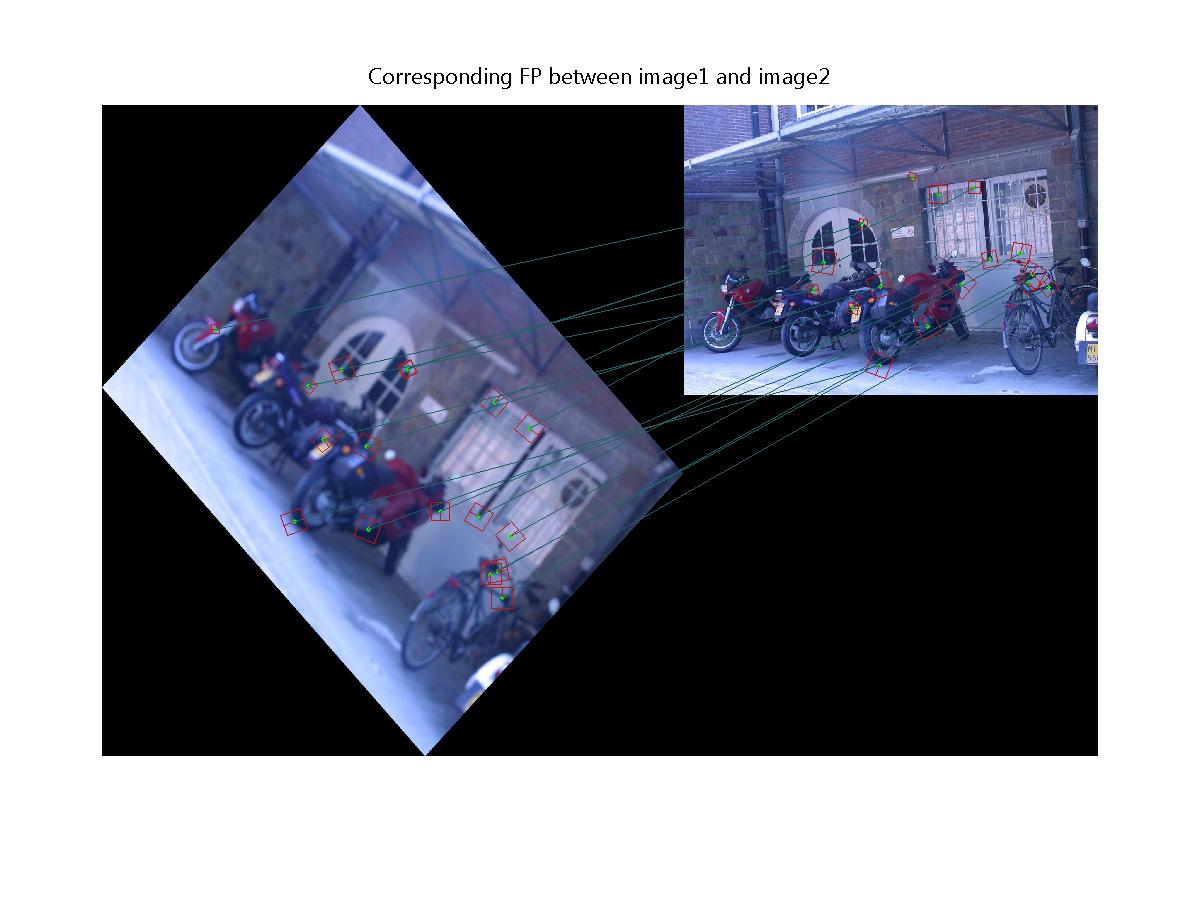


Figure 7: Result: matched FP’s.

1. *Shift invariance:*



Figure 8: Original compared images (taken from video sequence).

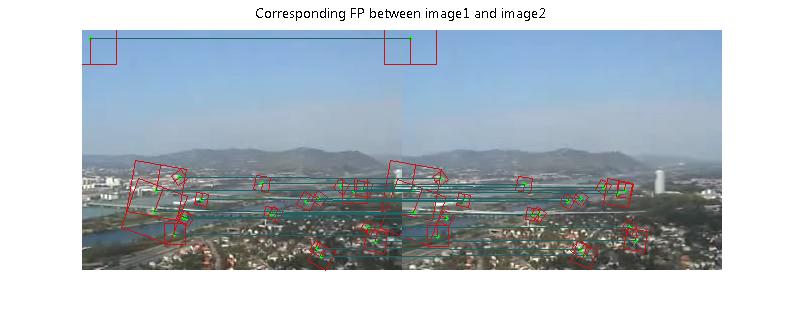


Figure 9: Result: matched FP’s.

1. *View point changes/object recognition:*



Figure 10: Original compared images (counting from left to right).



Figure 11: Result: matched FP’s of second and third images.

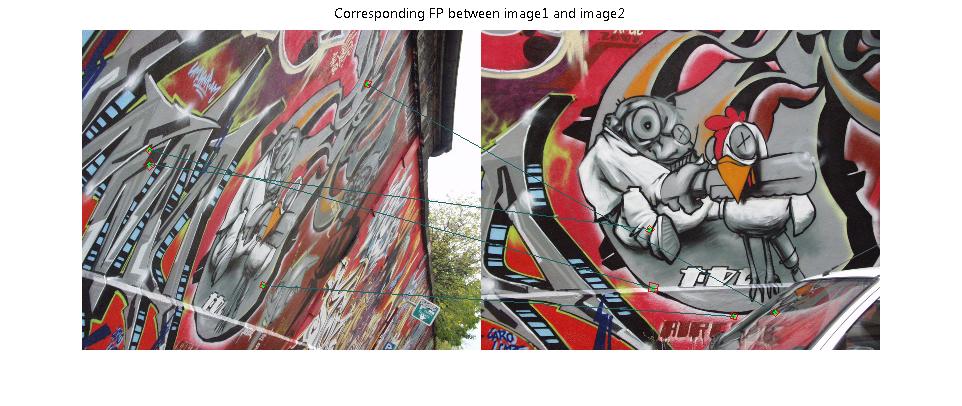


Figure 12: Result: matched FP’s of first and second images.

As can be seen, the built descriptor isn’t invariant under dramatic view point changes.

In order to solve this issue was implemented **ASIFT** descriptor. Here is the same comparison with **ASIFT** descriptor.



Figure 13: Result: matched FP’s of first and second images with **ASIFT** descriptor.

As can be seen there are a lot of true matches, few false (pixel on car connected with sky pixel) and some positive-false matches (were connected points that locally look the same – points on letters were connected to points on different letters).

*Using descriptor for recognition purposes:*



Figure 14: Original compared images.

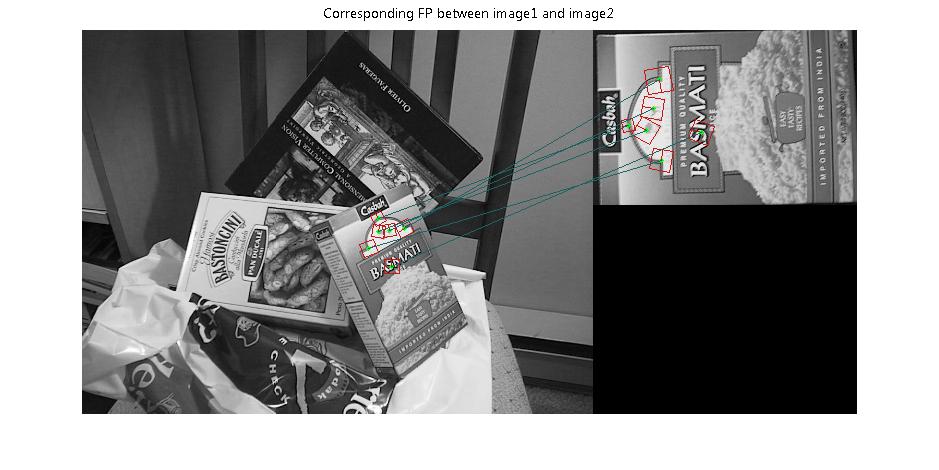


Figure 15: Result: matched FP’s.

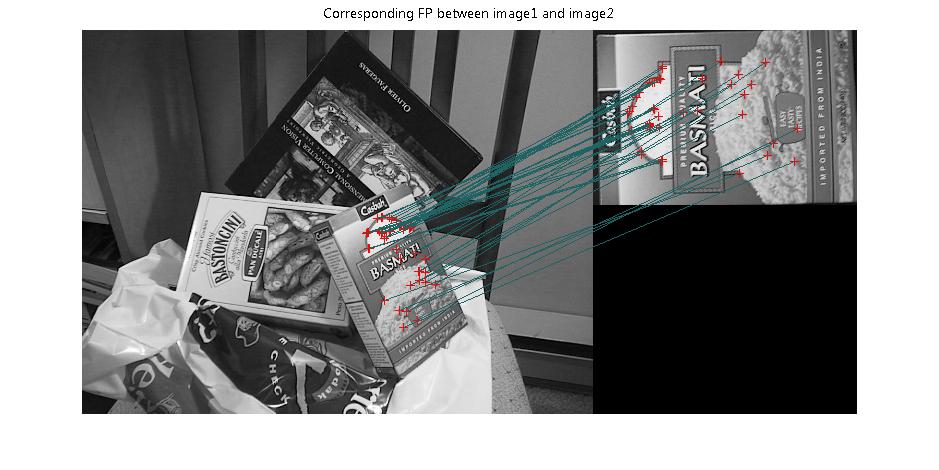


Figure 16: Result: matched FP’s with **ASIFT** descriptor.

1. *Conclusions:*

Was built SIFT-like descriptor, which is invariant under blur, rotation, shift, scale and moderate changes in viewpoint. Also as improvement was implemented **ASIFT** descriptor that shows very good results in all kind of transformations between images. The main (and the only) disadvantage of this descriptor comparing to a SIFT-like is running time. It takes a lot of time in order to compare two images with a lot of matches (the smaller viewpoint change between images – bigger number of correspondences, it takes more time to run).

The robustness of SIFT-like descriptor is sufficient for a range of purposes: from panorama making to object recognition.

\*[Here](http://rs689l3.rapidshare.com/files/281395804/Results.zip) can be download archive with all results of comparisons of images with SIFT-like descriptor. Also there are some results of run of **ASIFT** descriptor.