Ukrainian Catholic University

Faculty of Applied Science

SIFT FOR IMAGE STITCHING

Linear Algebra final project report

Nazariy Bachynsky

MAY 22, 2019

Abstract

IMAGE STITCHING IS A PROCESS OF COMBINING MULTIPLE PHOTOGRAPHIC IMAGES WITH OVERLAPPING FIELDS OF VIEW TO PRODUCE A SEGMENT PANORAMA OR HIGH-RESOLUTION IMAGE. IT IS USED IN MANY MODERN APPLICATIONS, FOR EXAMPLE, HIGH-RESOLUTION PHOTOMOSAICS IN DIGITAL MAPS AND SATELLITE PHOTOS, MEDICAL IMAGING, MULTIPLE IMAGE SUPER-RESOLUTION, VIDEO STITCHING, OBJECT INSERTION. THE FIRST PAPER RELATED TO THIS TOPIC CAME TO US FROM 1981 (MORAVEC, STEREO MATCHING USING A CORNER DETECTOR). IN 2007, LOWE AND BROWN [1] DESCRIBED ALGORITHM WHICH IS NOW IMPLEMENTED IN OPENCV LIBRARY. THIS ALGORITHM IS COMPLICATED AS IT CONSISTS OF MANY OTHER ALGORITHMS FOR FEATURE DETECTION, THEIR MATCHING, PERSPECTIVE TRANSFORMATION AND OTHERS (WE'LL DISCUSS). IN MY WORK I WILL MAKE AN OVERVIEW ON THIS ALGORITHM WITH DETAIL REVIEW OF SOME PARTS.

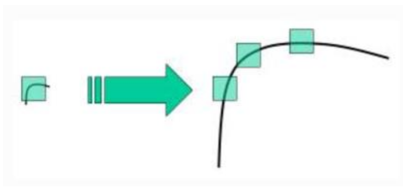
1. Introduction

Simply put, to stitch images we need to detect common parts of images and merge them (probably with perspective warp). But how it can be done in practice? Let's discuss. Finding common fields on images is the main part of algorithm (but not the most complicated), so in this work I'll pay it more attention than other ones. It can be splitted in two steps (which also consists of many tasks) – feature detection and feature matching between images. The last thing to do is bundling which is not a topic of this work.

2. Features detection using Scale Invariant Feature Transform (SIFT)

2.1 Scale-space extrema detection

We want algorithm to stitch images even if they have different rotation, perspective or scale. The question is how to detect corners in such cases. That is when we observe some part through the "window" (marked green) of image the corner can be not corner anymore when image is scaled.



It obvious, we need larger windows for larger corners. Scale-space filtering is needed. It has been shown by Koenderink (1984) and Lindeberg (1994) that under a variety of reasonable assumptions the only possible scale-space kernel is the Gaussian function. The good assumption is to use Laplacian of Gaussian (LoG). It is found for image with different σ. Depends on its value, gaussian kernel fits well with corners of different size. However, the LoG is a little costly in computations, so SIFT uses Difference of Gaussian. But firstly, we have a function

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

be the scale space of an image, where $I(x, y)$ is an input image, $*$ means a convolution and $G(x, y, \sigma)$ is a variable-scale Gaussian:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}.$$

Then, using scale-space extrema in DoG, we have:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma).$$

Here we see obvious advantage of this method – after getting smooth image L, all we need to do is just subtract less smoothed image from more smoothed one.

That is, the main point of this method is to compute difference of Gaussian blurring of an image with different $sigma$, say σ and $k\sigma$. We should repeat it scaling σ. Octave contains images with the same σ but different k. Each octave down-samples σ by 2 and use the same list of values of k.

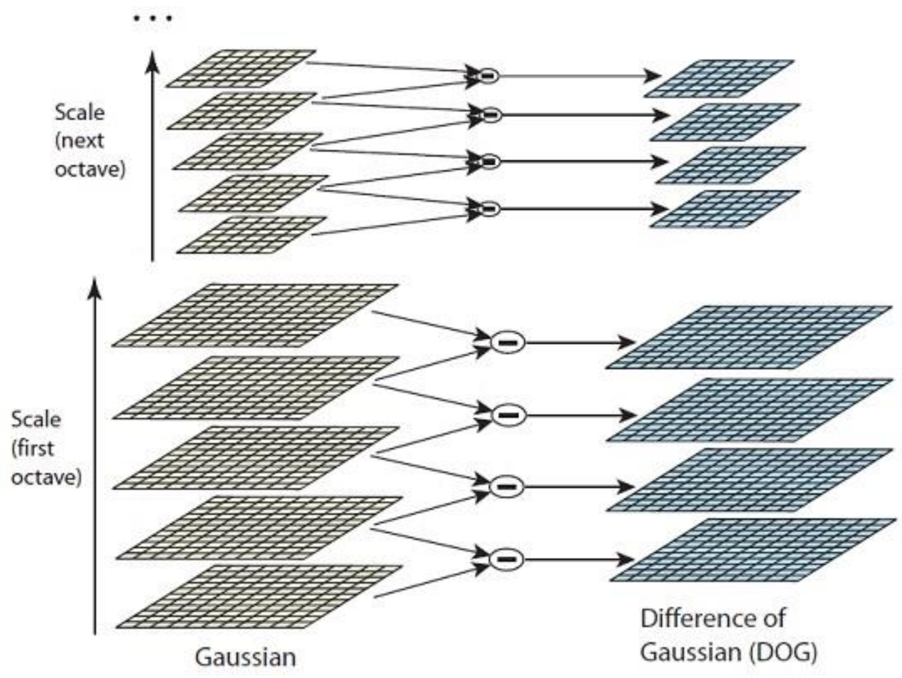
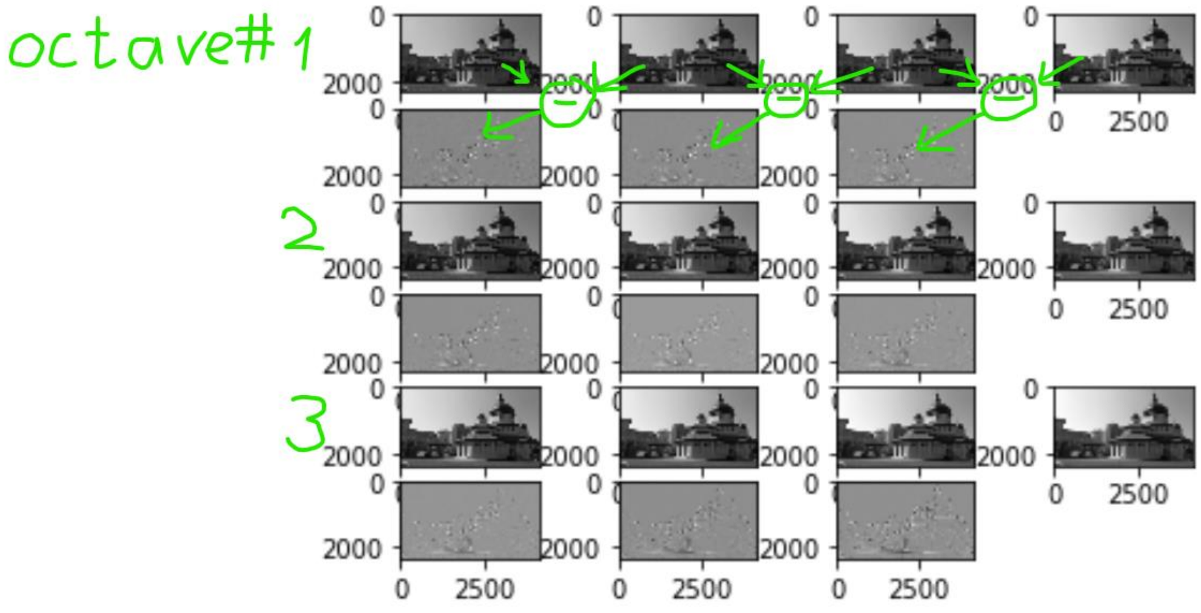


Figure 1. Image from [Lowe4]



Here is the example with one octave, $\sigma = 4$ …

By the way, Lindeberg (1994) showed that DoG provides a close approximation to the scale-normalized LoG.

The next step is to compare every pixel with its 8 neighbors in the same scale, and 9 from next and previous scales. The task is to find minima and maxima, that is pixel X is taken only if it is local minimum or maximum.

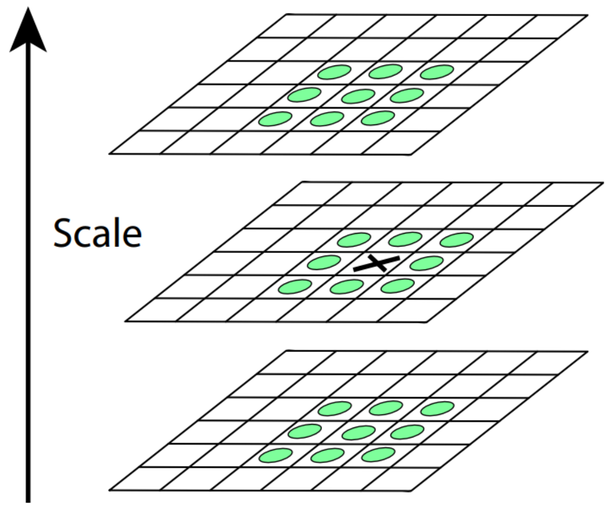


Figure 2. Pixel X is compared with its 26 neighbors in 3x3 region. [Lowe4]

2.2 Keypoint localization

DoG function and filtering by comparison with neighbors give us many candidates of keypoints. Thus, we need to discard excessive points.

Brown (Lowe and Brown, 2002) developed a method which calculates the interpolated location of the extremum, which substantially improves matching and stability. The interpolation is done using the quadratic Taylor expansion of the Difference-of-Gaussian scale-space function, $D(x, y\ \sigma)$ shifted so that the origin is at the sample point:

$$D(\mathbf{x}) = D + \frac{\delta D^T}{\delta \mathbf{x}} \mathbf{x} + \frac{1}{2} \mathbf{x}^T \frac{\delta^2 D}{\delta \mathbf{x}^2} \mathbf{x}$$

Where $\boldsymbol{x} = (x, y, \sigma)^T$ is the offset of that sample point. As we need extremum $\hat{\boldsymbol{x}}$, we can get it by computing derivative of this function and setting it to zero. The result is

$$\hat{\mathbf{x}} = -\frac{\delta^2 D^{-1}}{\delta \mathbf{x}^2} \frac{\delta D}{\delta \mathbf{x}}$$

2.2.1 Interpolation of nearby data for accurate position

In case $\hat{\mathbf{x}}$ is larger than 0.5 in any direction, it means it's close to another candidate keypoint and then the next potential keypoint is taken

2.2.2 Discarding low-contrast keypoints

Taking function value at the extremum $D(\hat{\mathbf{x}})$ gives us contrast level of point in this location, thus we can discard keypoints with function value at them less than 0.03 as they are low contrast. It is easy to see, that if we substitute $\hat{\mathbf{x}}$ in D, we'll get:

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\delta D^T}{\delta \mathbf{x}}\hat{\mathbf{x}}.$$

However, it is still not enough accurate. DoG method provides too much points on edges. We use 2x2 Hessian matrix **H** to compute the principal curvature:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix}$$

The derivatives are estimated by taking differences of neighboring sample points. To avoid computing eigenvalues which are proportional to the principal curvatures of D, we can use approach provided by Harris and Stephens in 1988. Let α be the eigenvalue of **H** with the greatest absolute value and β the other one. Using eigenvalue properties, we have:

$$Tr(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$

$$Det(\mathbf{H}) = D_{xx}D_{yy} - \left(D_{xy}\right)^2 = \alpha\beta.$$

Sometimes, when determinant is less then zero, eigenvalues have different signs, then curvatures are so and the point is discarded, because it's not an extremum.

It is also needed to check whether the ratio of principal curvatures is below some threshold. And we can do it without their individual values. Let r be such that $\alpha = r\beta$, then

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}.$$

It is easy to see, $(r+1)^2/r$ will be minimum when eigenvalues are equal.

And to compare the ratio of principal curvatures with some threshold t, we use

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} < \frac{(t+1)^2}{t}.$$

David Lowe in his experiments took r to be equal to 10.

2.3 Orientation Assignment

Now an orientation is assigned to each keypoint to achieve invariance to image rotation. A neighborhood is taken around the keypoint location depending on the scale, and the gradient magnitude and direction are precomputed using pixel differences.

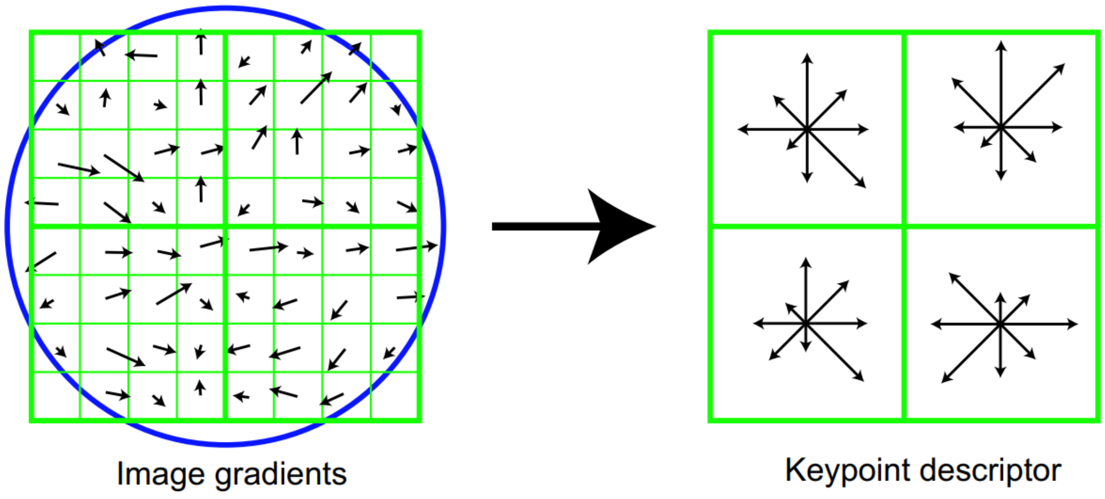
$$m(x,y) = \sqrt{\left(L(x+1,y) - L(x-1,y)\right)^2 + \left(L(x,y+1) - L(x,y-1)\right)^2},$$

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right)$$

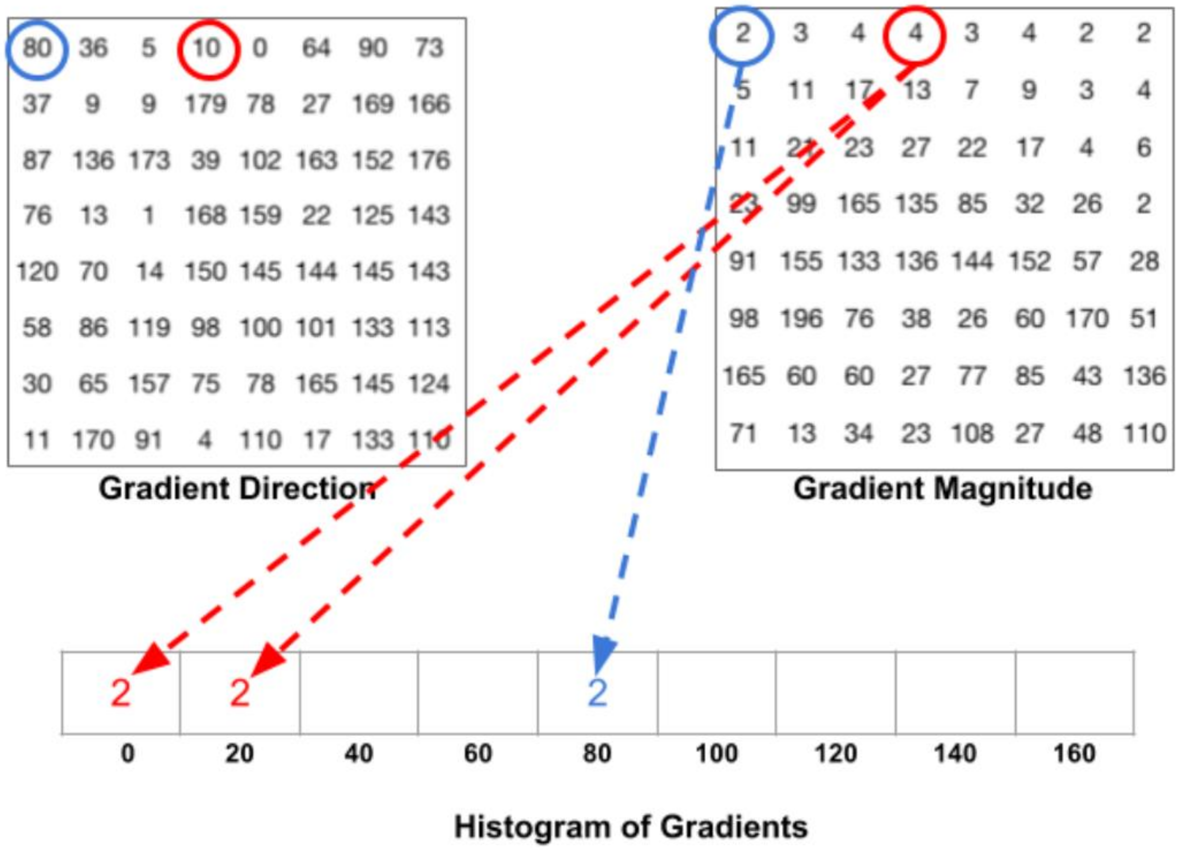
An orientation histogram with 36 bins covering 360 degrees is created. It is weighted by gradient magnitude and gaussian-weighted circular window with \sigma equal to 1.5 times the scale of keypoint. The highest peak in the histogram is taken and any peak above 80% of it is also considered to calculate the orientation. It creates keypoints with same location and scale, but different directions. It contributes to stability of matching.

2.4 Keypoint Descriptor

Keypoints descriptors are computed using magnitudes and orientations of image gradients at each point in a region around the keypoint location.



Each entry of keypoint descriptor is accumulation of 4x4 subregion from source. Length of each arrow is sum of gradients magnitudes near the corresponding direction. The calculations are made like the following. Here we have 2x2 descriptor from 8x8 region but in practice 16x16 => 4x4 size is used.



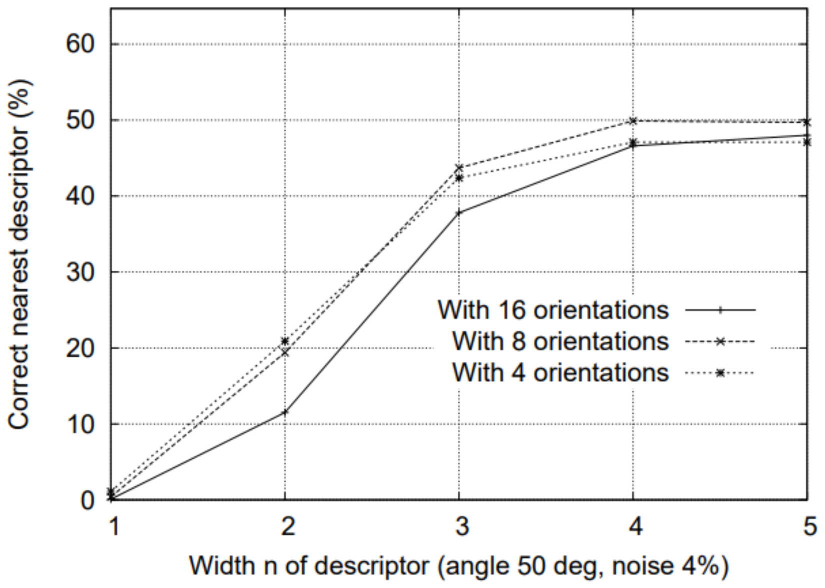
80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

2	2			2				
0	20	40	60	80	100	120	140	160

The first entry of magnitude matrix (2) corresponds to direction 80 degrees, so it goes in cell “80” in histogram of gradients. However, for example, red entry of orientation matrix (10) lies between 0 and 20, so the half of magnitude goes to cell ”0” and the other part to “20”. The same logic 4 from the third entry (from magnitudes) will give 1 to “0” and 3 to “20”.

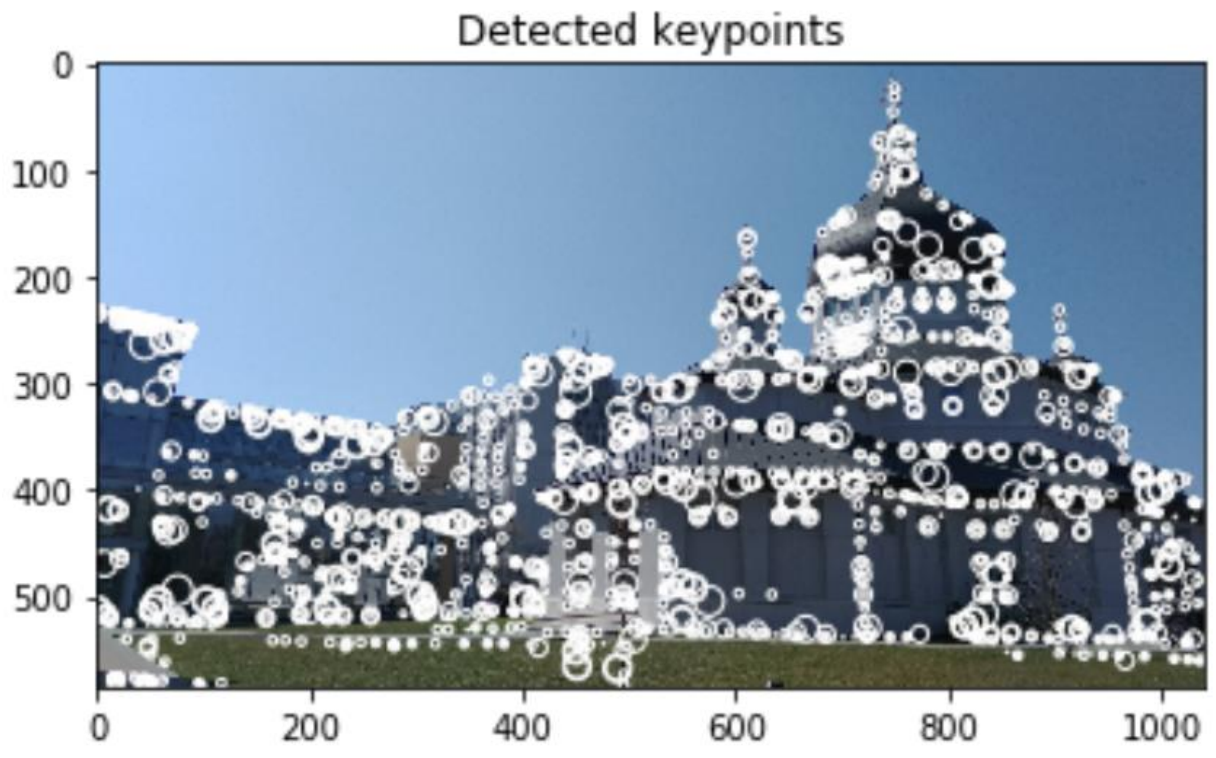
Lowe tested accuracy of histograms with different number of orientations by comparing keypoints with database. Here is the result

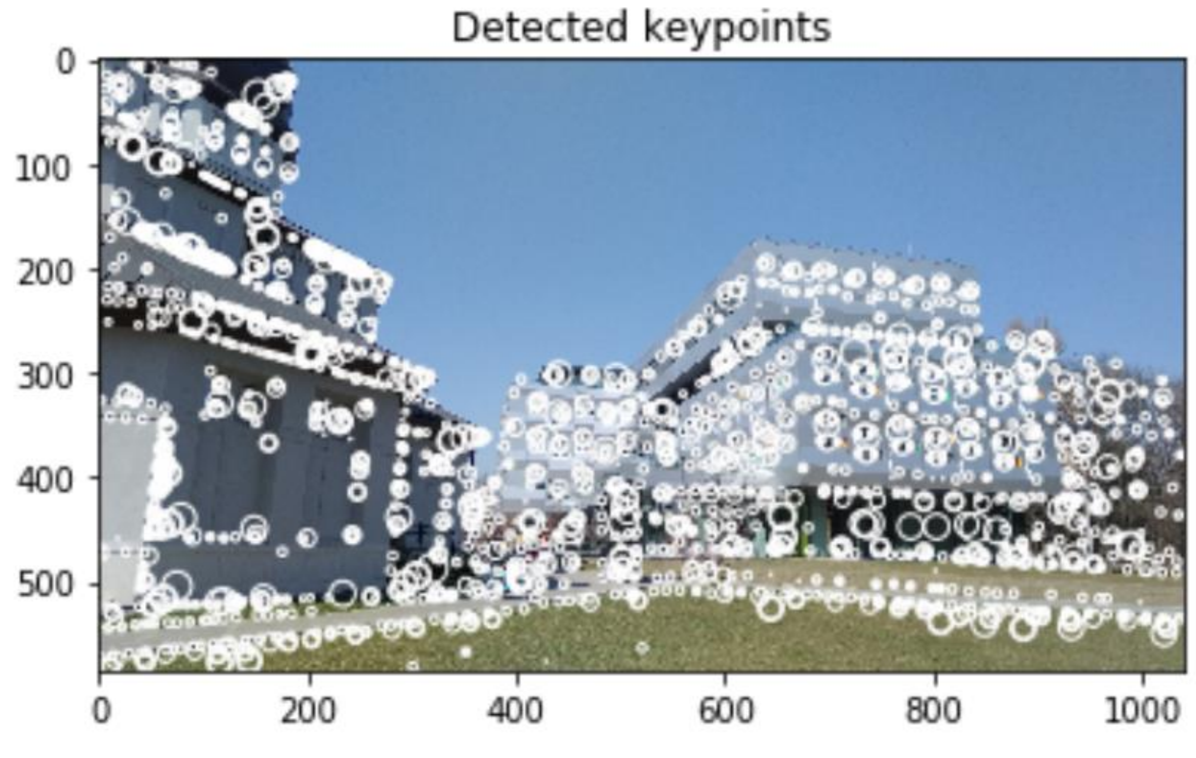


Thus, the number 8 for orientations is taken.

2.5 Examples

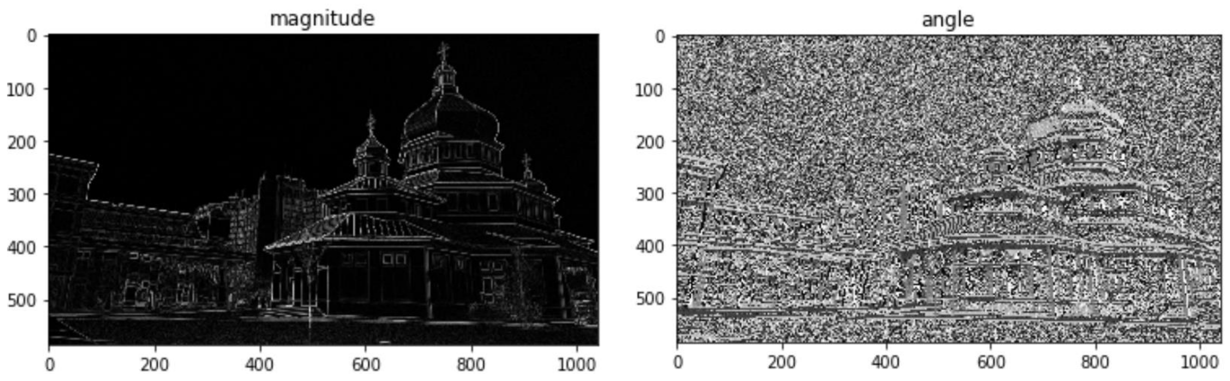
Here are some results of my experiments. The resolution of original image was over 4000x2000. I downscaled it 4 times to make computing faster. I took number of octaves be equal to 5, level of scaling sigma per each octave 2 (it should be integer) and $k = 2^{(1/scale\_per\_octave)}$ as Lowe suggested. Initial sigma = 1.6 and number of iterations in octave = scale_per_octave + 3. Threshold for discarding low contrast points is 0.01 instead of 0.03 in Lowe. In my case, these numbers work better than numbers suggested by Lowe. Here is the result of finding candidate keypoints.


Detected keypoints
0
100
200
300
400
500
0
200
400
600
800
1000


Detected keypoints
0
100
200
300
400
500
0
200
400
600
800
1000

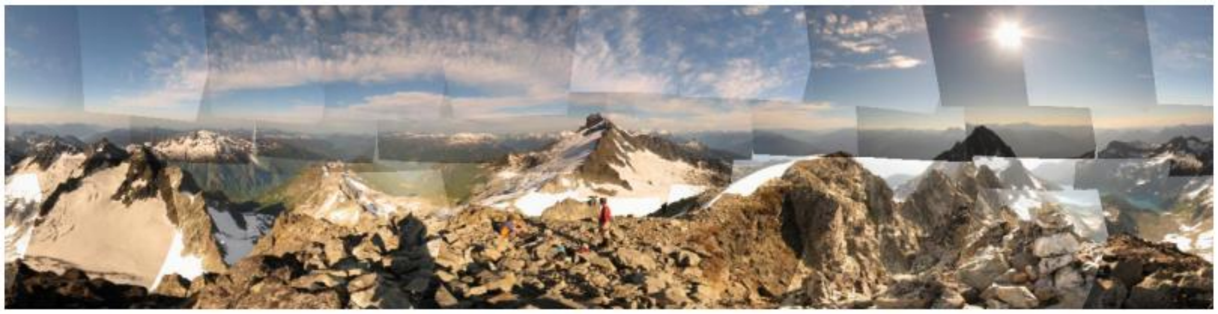
Remark

Besides of this, I also implemented gaussian filter and derivative function for matrix using Sobel matrix, using them, here are magnitudes and orientations of keypoints.

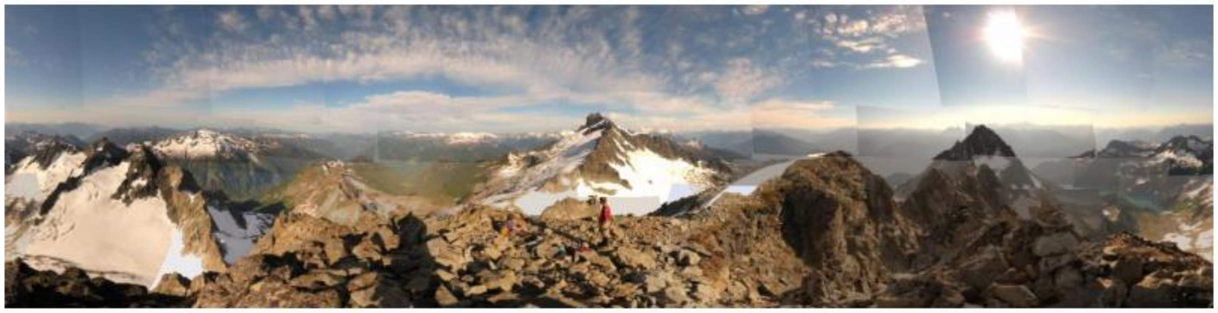


3. The next steps

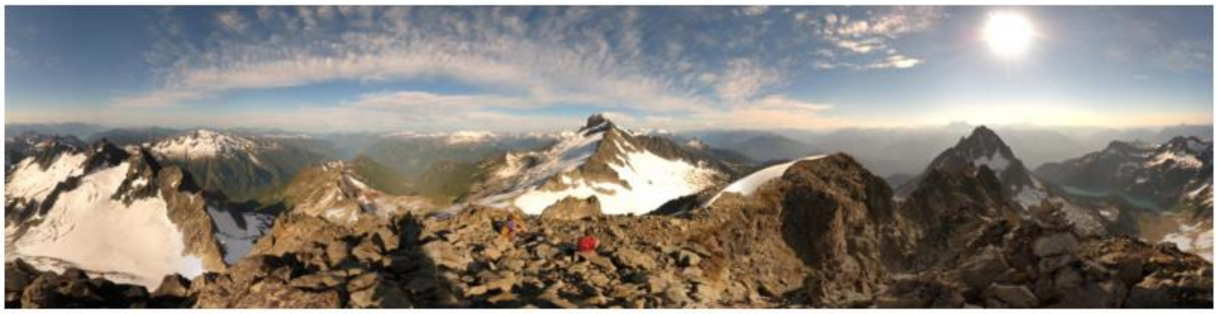
Then we need to find feature descriptors based on histograms of keypoints (from their magnitudes and orientations). After that keypoints of images will be matched to each other and homography matrix will be found using RANSAC (to discard outliers). Lowe and Brown in their work used k-d tree for matching keypoints between images. It allows to work with images and don’t think about their rotation and order. After this is done the task is to merge images. Thanks to feature descriptors it will be done even if images have different scale and orientation. And the last step which is also looks as complete independent task is to bundle the resulting image. (Example from Lowe and Brown, 2007)



(b) Without gain compensation



(c) With gain compensation

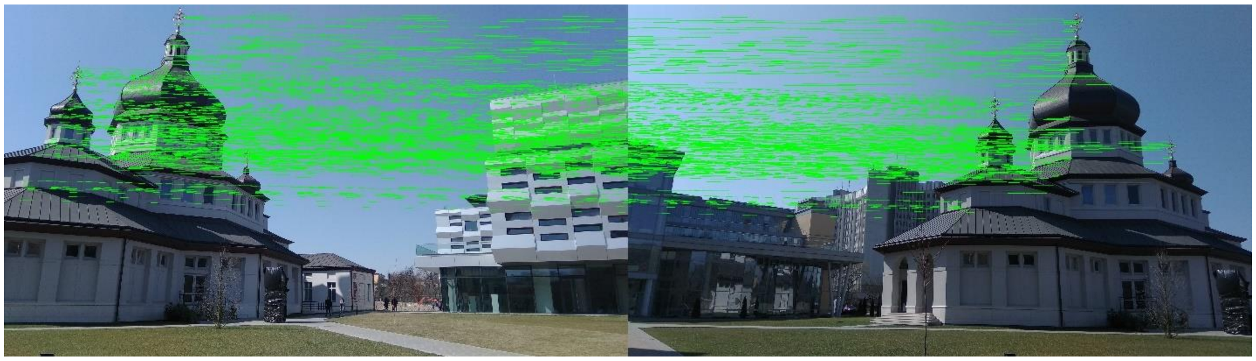


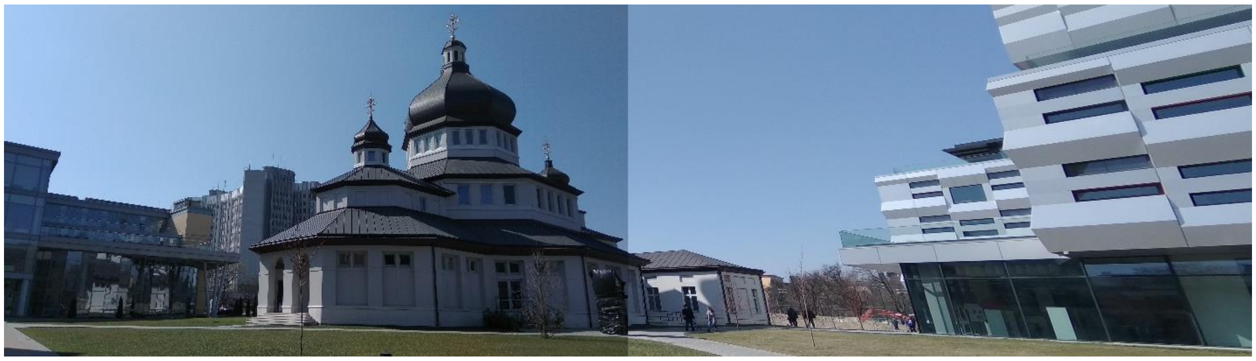
(d) With gain compensation and multi-band blending

4. Conclusions

Within this project I have learned to use basic operations with images. I also implemented some algorithms by my own: Gaussian blur, Difference of Gaussians, computing local extremas, computing derivatives of images and magnitude and orientation matrices based on it. This is partly implementation of SIFT-algorithm. However, as I mentioned earlier, I carried out experiments and picked up coefficients for algorithms which give better results.

I have also implemented program for full image stitching (see examples/ImageStitcherOpenCV). It's just for seeing how the result could look like, so code is based on code example from article from pyimagesearch . I use already implemented functions for find features and describe them and for stitch images. I manage them to do work. The disadvantage is that it works only with two images which are given in the right order. Here are the results of its work.





References:

[1] Automatic Panoramic Image Stitching using Invariant Features, M. Brown and D. G. Lowe, 2007

[2] Distinctive Image Features from Scale-Invariant Keypoints, D. Lowe, 2004

[3] Image stitching, Wiki

[4] OpenCV SIFT tutorial

[5] Difference of Gaussians, Wiki

[6] Other OpenCV SIFT tutorial

[7] Implementation of HDR panorama stitching algorithm

[8] Feature matching and Homography using OpenCV

[9] Honography theory, OpenCV

[10] Introduction to SIFT, Medium