LAB 1 Computer Vision

Josep Monclús Carrasco 250004
Xavier Akira Pavon Namba 217941

1.1 Sift

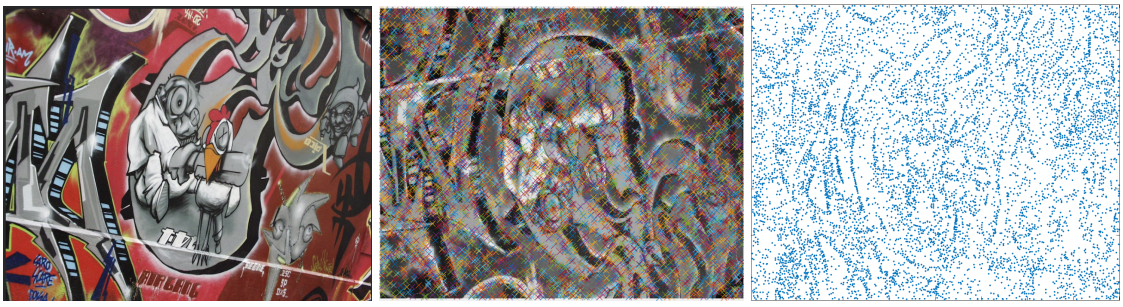
You need to understand what each function does in terms of functionality, input and output. Then you need to implement the missing instructions in the file find extremas.m. The location of the code that needs to be implemented is indicated by the text .

We created the function answered not by looping across all the rows and columns, we did it in a way that we think is more efficient by using logic operators.

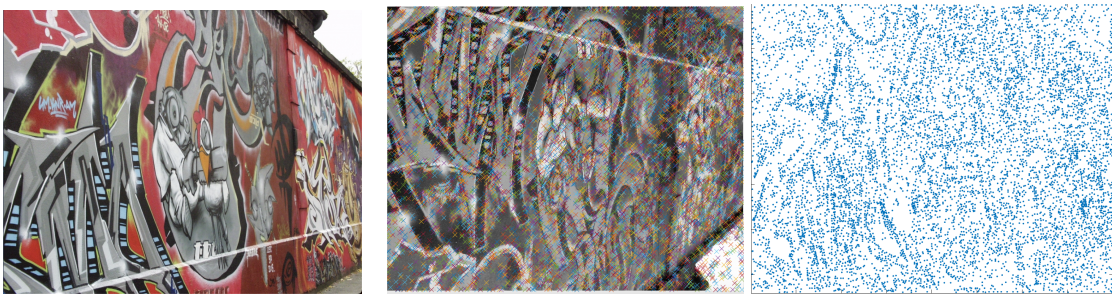
Try to run your code in several examples, by considering indoor and outdoor scenarios as well as textureless areas. You can acquire your own dataset here, or surf on the Internet. What can you comment about the results? Are there outliers in your estimation? If so, please explain some ideas to remove them.

We run it in some of the examples, in the left original photo, in the middle the image in grayscale with the key points marked on the rotated image, and in the right only plotting the key points.

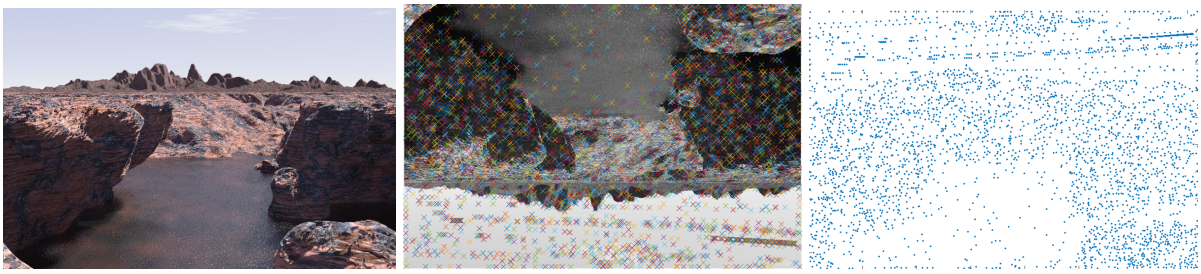
img3.jpg



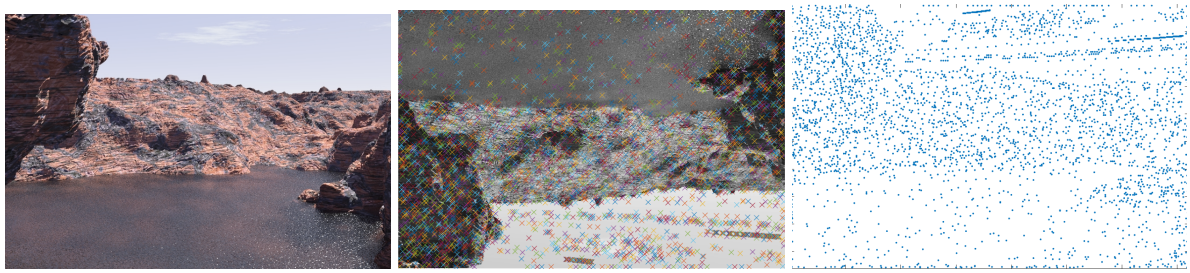
img5.jpg



landscape-a.jpg



landscape-b.jpg



person.jpg (own downloaded photo expecting key points only in the person)

ball.jpg (other downloaded photo)

RESULTS

As we can see, in the first 4 photos we can intuit more or less some of the variations on the image as the graffiti, and the key point plotting is very similar with the rotated graffiti, more or less the same for the landscape scenarios but we start seeing some of the problems with the algorithm like some outliers because of minimal variations on the image. To represent this problem, we decided to search for 2 simple photos (something in a white background) to see how it works and as we can see with the person the results were not as expected, it detects some variations in the person but it detect some areas on the white background that we do not want because are so small variations, for this we think that it is necessary to take into account low contrast, key points with small contrast in this image are not really relevant so it could be good idea to remove it. In the ball photo we arrive at an outlier case, we expected to detect in the perimeter of the hexagons that are the parts where the pixel change is more abrupt.

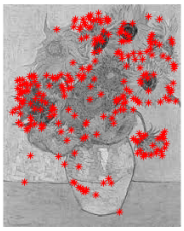
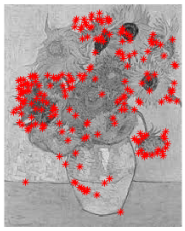
1.2 Algorithm fine-tuning

For each of the required algorithms, the optional parameters were programmatically fine-tuned in order to achieve good results on the input image (*sunflower.jpg*, in this exercise) and to examine the effect of each parameter on the algorithm's performance. This was achieved by running each algorithm on a wide range of parameter combinations, and selecting the best results in accordance with two different metrics:

- MAX: Among all of the algorithm calls, the one with the highest mean keypoint score and mean top (20) keypoint score was selected. They correspond with the first row in the subsequent tables.
- MEAN: Among all of the algorithm calls, the one with the highest *combined* mean keypoint score and mean top (20) keypoint score was selected. They correspond with the second row in the subsequent tables.

FAST

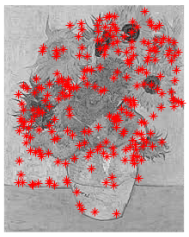
Mini Quality	Min Contrast	numFeat	mean Score	var(y)	var(x)	meanScoreTop	var(y)Top	var(x)Top	compTime
0.586	0.001	60	88	2613.7	2329.6	88.8	724.0	140.9	0.0046
0.379	0.001	250	63.84000015	2821.9	2587.1	61.0	670.1	10.9	0.0028
0.100	0.2	207	66.92753601	2845.1	2618.6	66.6	215.6	26.6	0.0111

detectFASTFeatures-MAX	detectFASTFeatures-MEAN	detectFASTFeatures-DEFAULTARGS
		
Min Quality: 0.586 Min Contrast: 0.001	Min Quality: 0.379 Min Contrast: 0.001	Min Quality: 0.1 Min Contrast: 0.2

FAST achieves satisfactory results on the MEAN metric choice, both qualitatively and quantitatively. Keypoints are well located along sharp edges and textured areas, with a relatively even distribution across the image (notice the high variance values). In this particular case, the parameter with the most influence on the results appears to be minQuality.

SIFT

Contrast Threshold	Edge Threshold	NumLayers In Octave	Sigma	numFeat	mean Score	var(y)	var(x)	meanScore Top	var(y)Top	var(x)Top	compTime
0	3	1	1.89	72	0.048	2590.126	4093.997	0.044	358.464	3788.705	0.010
0	3	1	2	85	0.047	2696.273	4234.133	0.049	263.247	4000.595	0.011
0.0133	10	3	1.6	238	0.025	2530.320	2918.019	0.027	21.849	1003.371	0.020

detectSIFTFeatures-MAX	detectSIFTFeatures-MEAN	detectSIFTFeatures-DEFAULTARGS
		
Contrast threshold: 0 Edge threshold: 3 Num. Layers in octave: 1 Sigma: 1.89	Contrast threshold: 0 Edge threshold: 3 Num. Layers in octave: 1 Sigma: 2	Contrast threshold: 0.0133 Edge threshold: 10 Num. Layers in octave: 3 Sigma: 1.6

Our fine-tuning approach on the SIFT algorithm did not achieve good results. Keypoints are detected on large texture-less areas, with little apparent correlation with the underlying geometry of the image. Further manual fine-tuning on ContrastThreshold, EdgeThreshold and NumLayersInOctave should be performed in order to improve the results.

SURF

Metric Threshold	Núm Octaves	N Scale Levels	numFeat	mean Score	var(y)	var(x)	ms Top	var(y)Top	var(x)Top	compTime
556	2	6	208	1526.247	2109.717	2892.680	6001.562	2477.165	2061.719	0.010
1000	2	6	94	2451.054	2328.897	2530.247	6001.562	2477.165	2061.719	0.007
1000	3	4	76	2346.193	2285.727	2590.908	4953.953	2652.173	2567.762	0.025

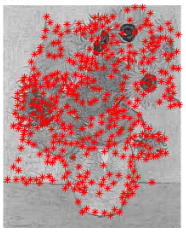
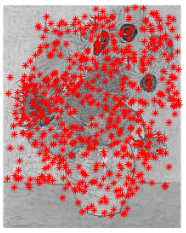
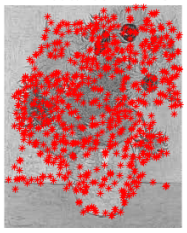
detectSURFFeatures-MAX	detectSURFFeatures-MEAN	detectSURFFeatures-DEFAULTARGS
		
Metric threshold: 556 Num octaves: 2 Num scale levels: 6	Metric threshold: 1000 Num octaves: 2 Num scale levels: 6	Metric threshold: 1000 Num octaves: 3 Num scale levels: 4

SURF performs relatively well on this particular case as well, as the detected keypoints are localized along clear areas of interest of the image. The fine-tuned results correlate with the default argument call (last row, image on the right).

KAZE

Diffusion	Threshold	Núm Octaves	N Scale Levels	numFeat	mean Score	var(y)	var(x)	meanScore Top	var(y)Top	var(x)Top	compTime

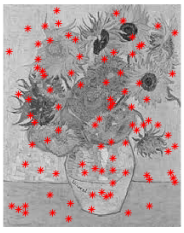
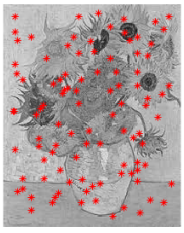
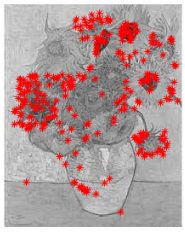
region	0.0001	1	4	391	0.00042	2490.071	3308.354	0.00034	1545.190	42.557	0.034
sharpedge	0.0001	4	3	555	0.00090	2542.271	3520.823	0.00028	1637.453	202.245	0.092
region	0.0001	3	4	734	0.00068	2504.004	3388.459	0.00034	1545.190	42.557	0.307

detectKAZEFeatures-MAX 	**detectKAZEFeatures-MEAN** 	**detectKAZEFeatures-DEFAULTARGS** 
Diffusion: region Threshold: 0.0001 Num octaves: 1 Num scale levels: 4	Diffusion: sharpedge Threshold: 0.0001 Num octaves: 4 Num scale levels: 3	Diffusion: region Threshold: 0.0001 Num octaves: 3 Num scale levels: 4

Keypoints are well distributed along key areas of the image, but perhaps the decision threshold should be increased in order to filter out some of the weakest detections.

BRISK

MinContrast	NumOctaves	MinQuality	numFeat	meanScore	var(y)	var(x)	msTop	var(y)Top	var(x)Top	compTime
0.001	1	0.89	89	310.483	2866.705	4690.948	310.800	3036.458	261.918	0.200
0.001	2	0.33	100	226.107	2747.834	4571.263	173.303	3962.332	2380.419	0.208
0.2	4	0.1	276	69.897	2911.463	2575.708	67.984	210.369	26.703	0.373

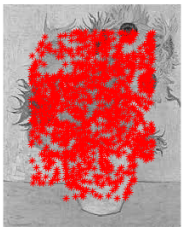
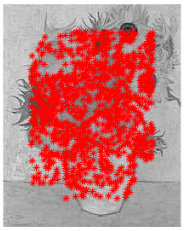
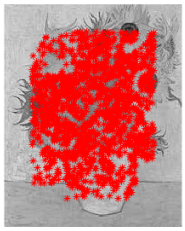
detectBRISKFeatures-MAX 	**detectBRISKFeatures-MEAN** 	**detectBRISKFeatures-DEFAULTARGS** 
Min. contrast: 0.001 Num. octaves: 1 Min. quality: 0.89	Min. contrast: 0.001 Num. octaves: 2 Min. quality: 0.33	Min. contrast: 0.2 Num. octaves: 4 Min. quality: 0.1

Once again, the fine-tuning approach should be improved, in this case to give more importance to Min Contrast.

ORB

Scale Factor	Núm Levels	numFeat	mean Score	var(y)	var(x)	meanScoreTop	var(y)Top	var(x)Top	compTime
2.53	5	1049	9.19E-05	1465.60	2333.98	0.00018	810.01	1.01	0.0031
3.21	5	1019	9.36E-05	1503.45	2369.82	0.00018	810.01	1.01	0.0046

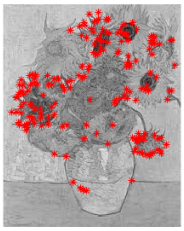
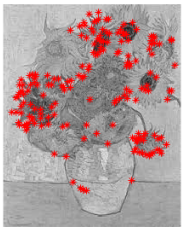
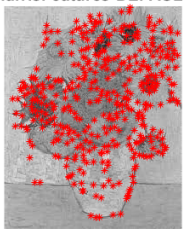
1.2	8	1817	7.32E-05	1340.66	2169.22	0.00018	810.01	1.01	0.0252

detectORBFeatures-MAX	detectORBFeatures-MEAN	detectORBFeatures-DEFAULTARGS
		
Scale factor: 2.53 Num. levels: 5	Scale factor: 3.21 Num. levels: 5	Scale factor: 1.2 Num. levels: 8

ORB properly recognises the object of interest, but at the expense of a low decision threshold.

Harris

Mini Quality	Filter Size	numFeat	mean Score	var(y)	var(x)	meanScoreTop	var(y)Top	var(x)Top	compTime
0.0526	3	181	0.0023	2830.89	2384.08	0.0026	19.605	309.803	0.139
0.0526	3	181	0.0023	2830.89	2384.08	0.0026	19.605	309.803	0.139
0.01	5	395	0.0007	2256.36	2867.14	0.0017	36.698	624.993	0.027

detectHarrisFeatures-MAX	detectHarrisFeatures-MEAN	detectHarrisFeatures-DEFAULTARGS
		
Min. quality: 0.0526 Filter size: 3	Min. quality: 0.0526 Filter size: 3	Min. quality: 0.01 Filter size: 5

The fine-tuned results on Harris are among the best, as they not only exhibit good detections, but they also outperform those obtained via the default argument call. This is especially noticeable along the right edge of the jar, on which the default call detects many similar-looking features along low curvature areas. In the fine-tuned calls, in contrast, only the clearly strongest keypoint (right at the intersection of the jar, floor and wall) is detected.

MSER

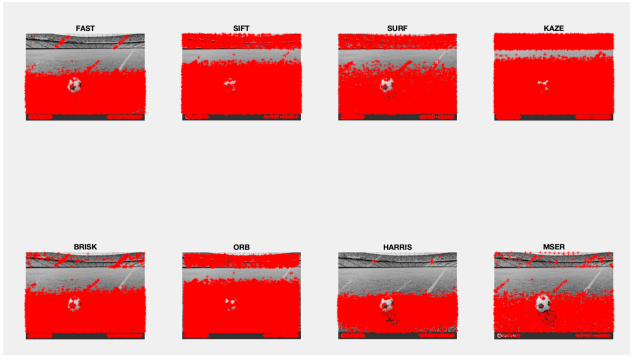
detectMSERFeatures-DEFAULTARGS

(default arguments)
Computation time: 0.042 seconds

Due to time constraints, we were not able to fully implement the fine-tuning function for MSER. By examining the default parameter choice, however, we can see that it's a relatively selective algorithm, although the detected keypoints are well located along the object of interest.

Are the analyzed algorithms scale independent?

Test:



Due to the time and the bad quality of the image we can not test exactly but we can see that more or less at least Harris and MSER are scale independent, because of as we can see in the key points obtained they do not point the ball, only the countorns of the ball, that’s why we can see the ball and not all red.