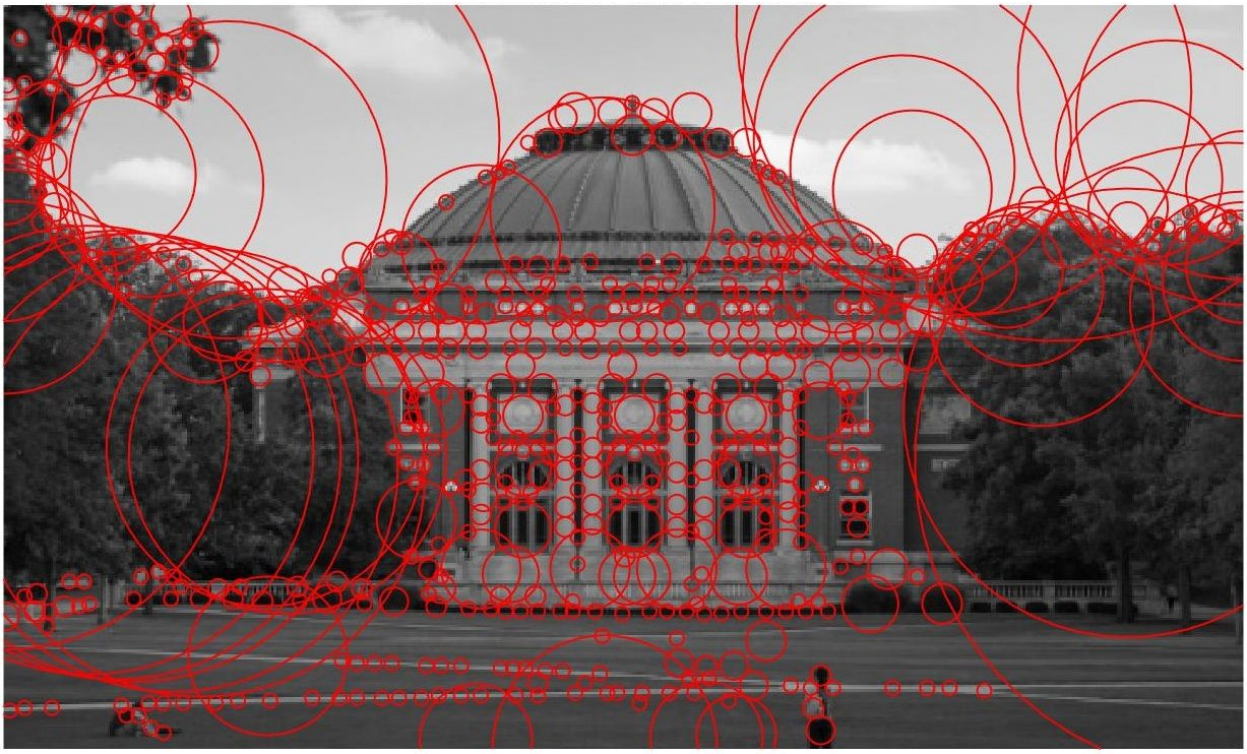
Scale-Space Blob Detection

Machine Problem 2 : CS 543 Computer Vision UIUC



Naman Shukla

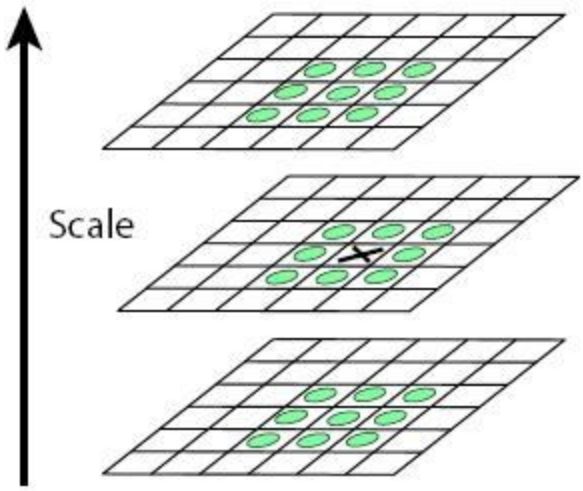
namans2@illinois.edu

INTRODUCTION

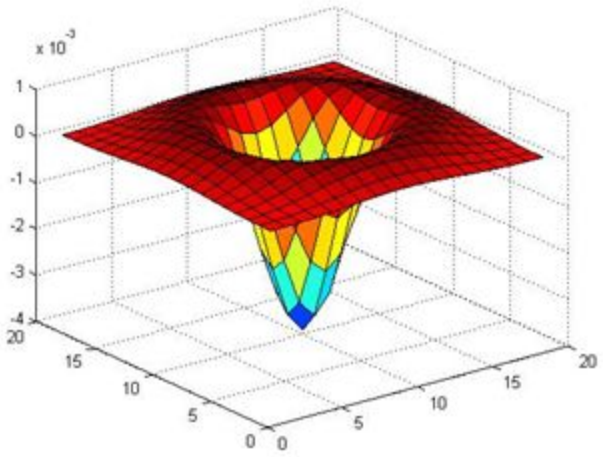
This machine problem is based on application of the scale and rotation invariance in finding the features (corners) through laplacian of gaussian filter. The blob detectors detect the image corners and represent as blobs which plays an important role in tracking and object detection. This report is the description of the implementation of machine problem 2. Implementation is done on MATLAB by following the algorithm layout provided in the course.

IMPLEMENTATION

1. Some hyper parameters are necessary to define for a single implementation. Here, initial ***sigma*** value for gaussian function used for filtering in laplacian of gaussian, number of ***step*** in the space scale and the ***threshold*** values for maximum suppression are those hyper parameters. Each combination of these parameters would result in different outcome and can be tuned according to user preferences.
2. The input image provided is raw and need to preprocess. The image is first converted to ***grayscale*** and then converted its data type to ***double*** followed by normalization.
3. After preprocessing, the blob detection algorithm is applied as follows:
    a. Scaling the filter operation (inefficient):



i. Convolve the image with filter of laplacian of gaussian with given sigma value. For this process MATLAB ***fspecial*** function is used (with keeping an odd filter as a constraint via sigma value) to generate filter.

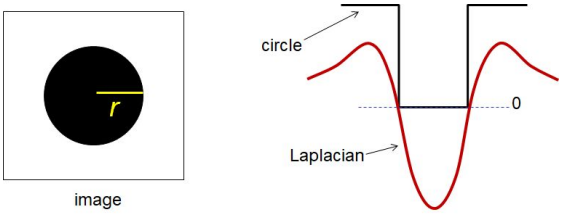


ii. Compare each pixel in the convolved image with a set of neighbourhood and selection of the best is performed.

iii. Repeat the above 2 step to create space scale (with hyperparameter of steps) with different values of sigma obtained by factor multiplication of K and thereby generating different size of filter to convolve an image with. As, we are only concerned with the maximum response of a pixel to LOG filter, non- maximum suppression is performed in 2D slices.

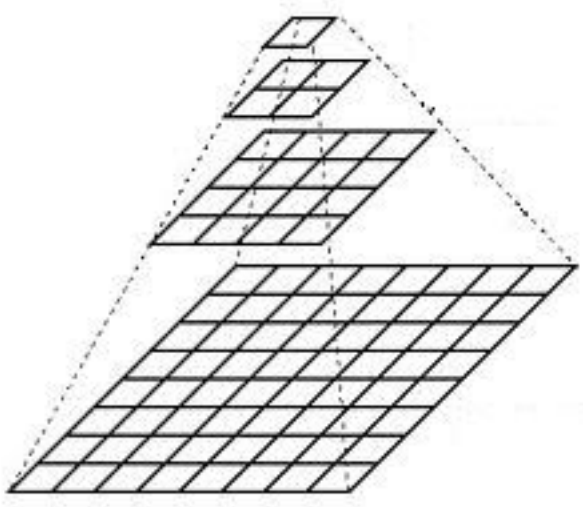
iv. Maximum value across the scale space is then taken as non-maximum suppression in 3D.

v. Now we consider only those pixel which passes survival threshold. The threshold value is the same value form hyperparameter. Then the characteristic radius and the center is calculated with the following equation: $r = \sigma . \sqrt{2}$



b. Scaling the image operation (efficient):

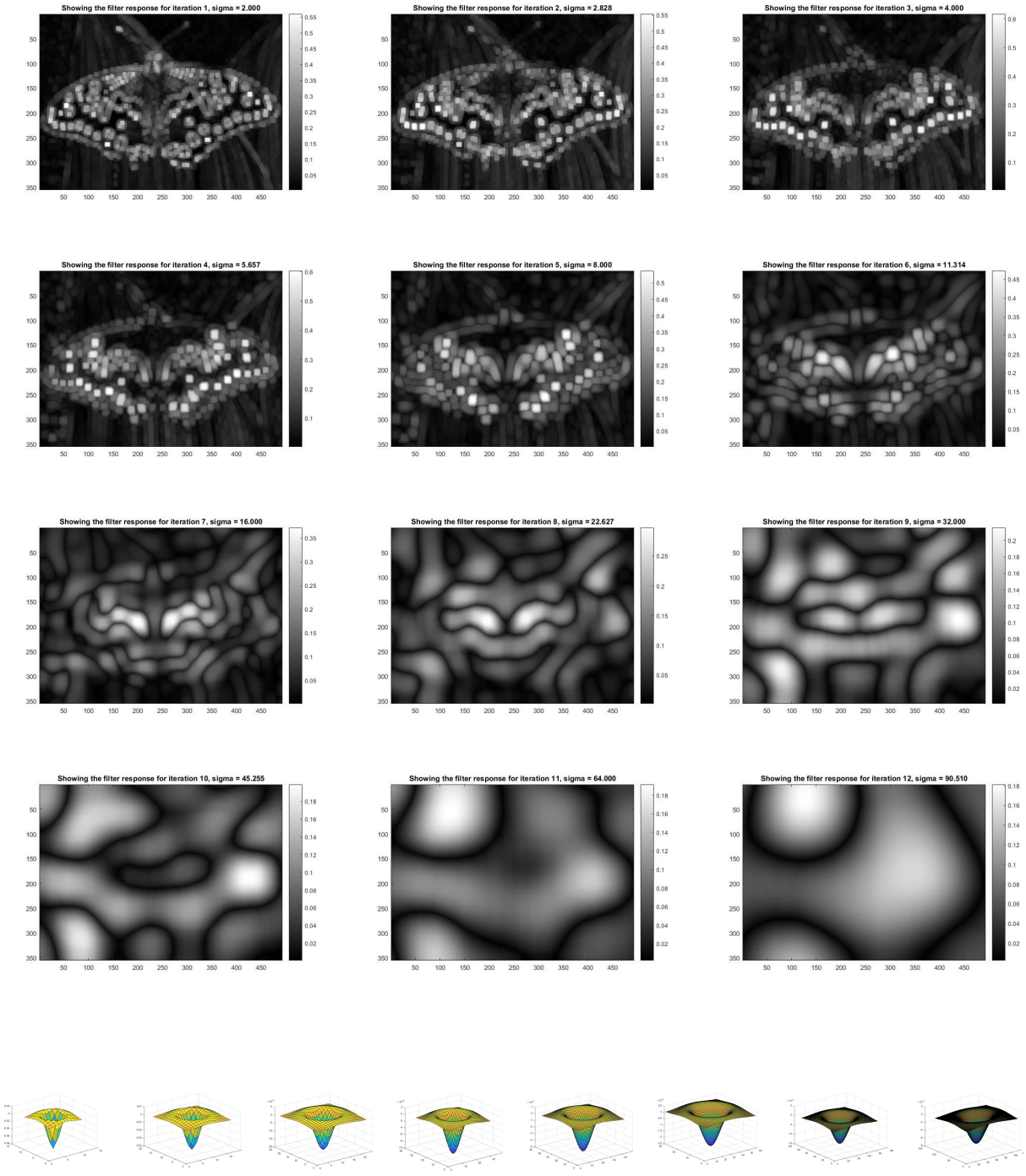
i. Given an image and a sigma value, the laplacian filter is created to convolve with the image.
ii. The given image is then resized with factor of inverse of k .



iii. The similar operation of the LoG convolution is applied to the resized image. The image is then rescaled again to normal dimensions.
iv. Similar steps starting from 2D non maximum suppression is applied to this procedure as well.

RESULTS

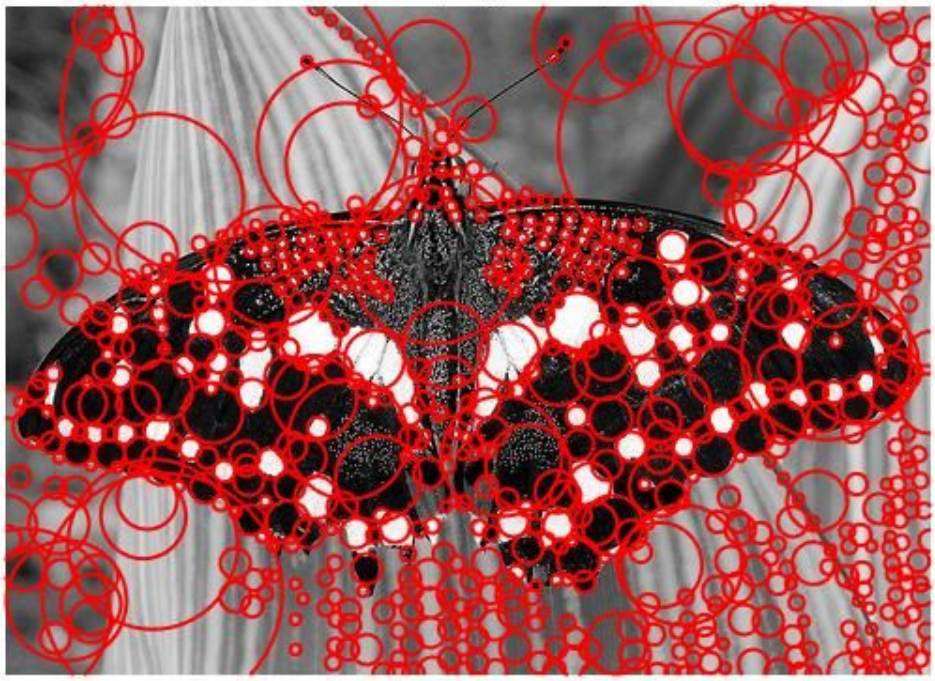
Shown below is the intermediate step in the overall process of LoG convolution with the image. I have created a visualization supporting function to display the process for better understanding.



The above image also displays the evolution of the LoG filter and its geometric view.

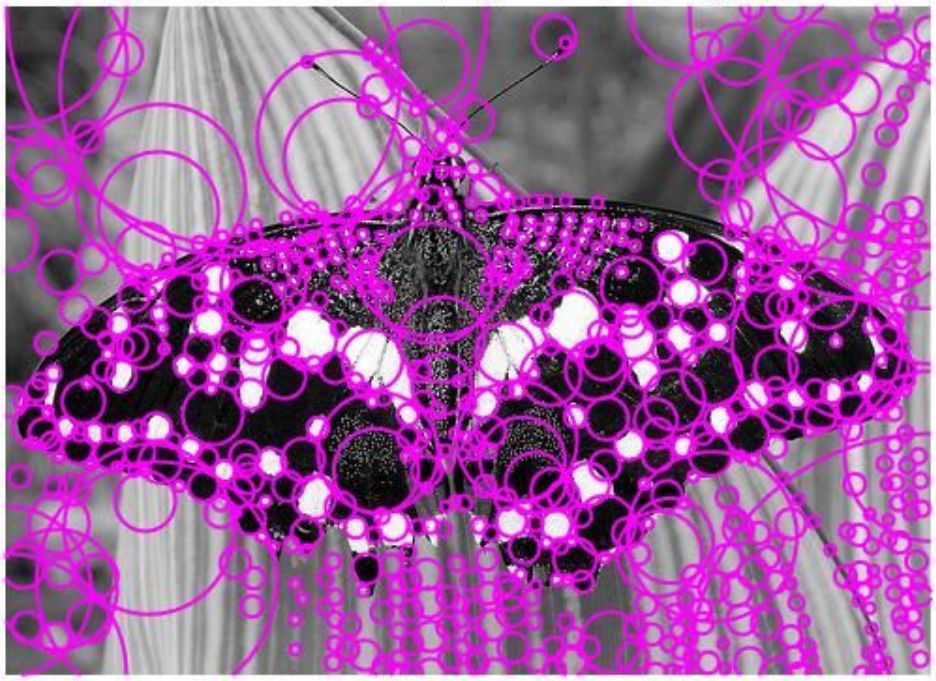
The figure below shows the ‘up scale’ implementation (inefficient) in red circles and ‘down scale’ implementation (efficient) in magenta circles.

Blobs: 706 , threshold: 0.001 , sigma: 2.0 , scale-space size: 12



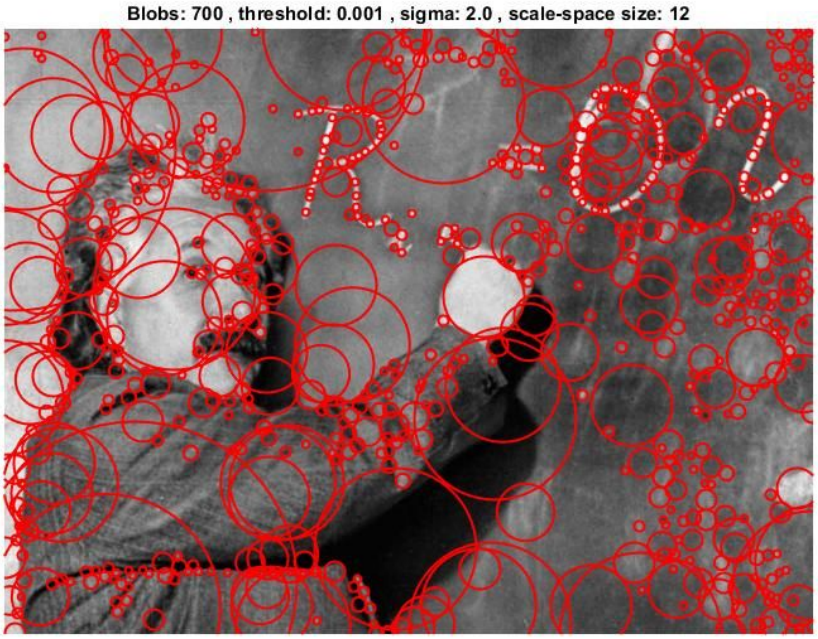
The circles are detecting the corners. Each blob visually represent geometric distinctiveness in the image space.

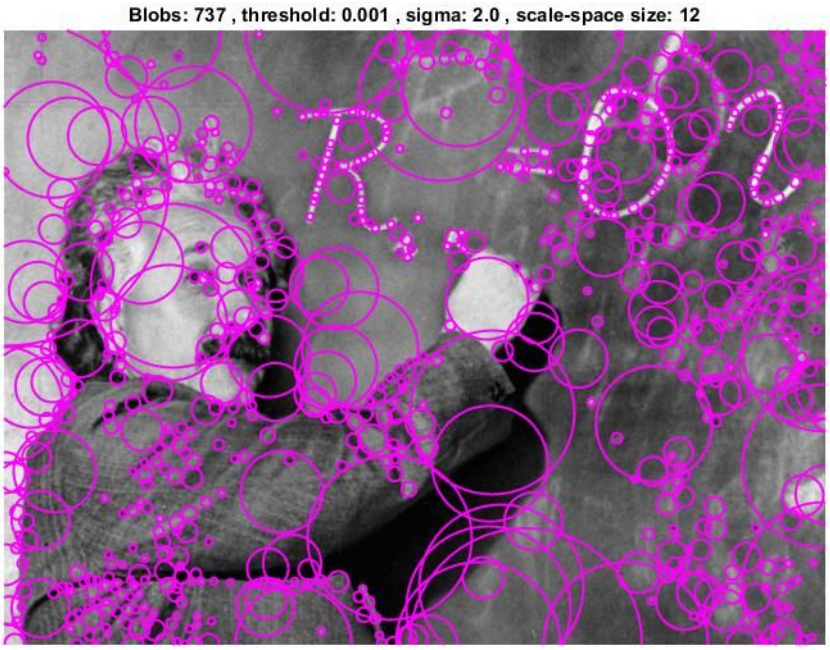
Blobs: 669 , threshold: 0.001 , sigma: 2.0 , scale-space size: 12



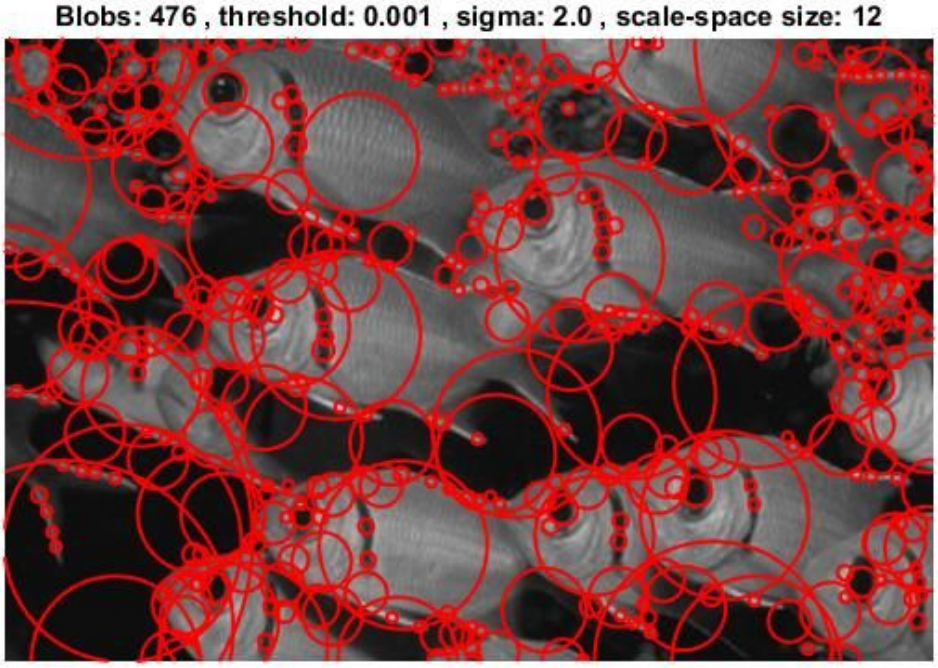
All the blobs both of the images are accurately capturing both white and black pattern in the butterfly (due to squaring the filter response)

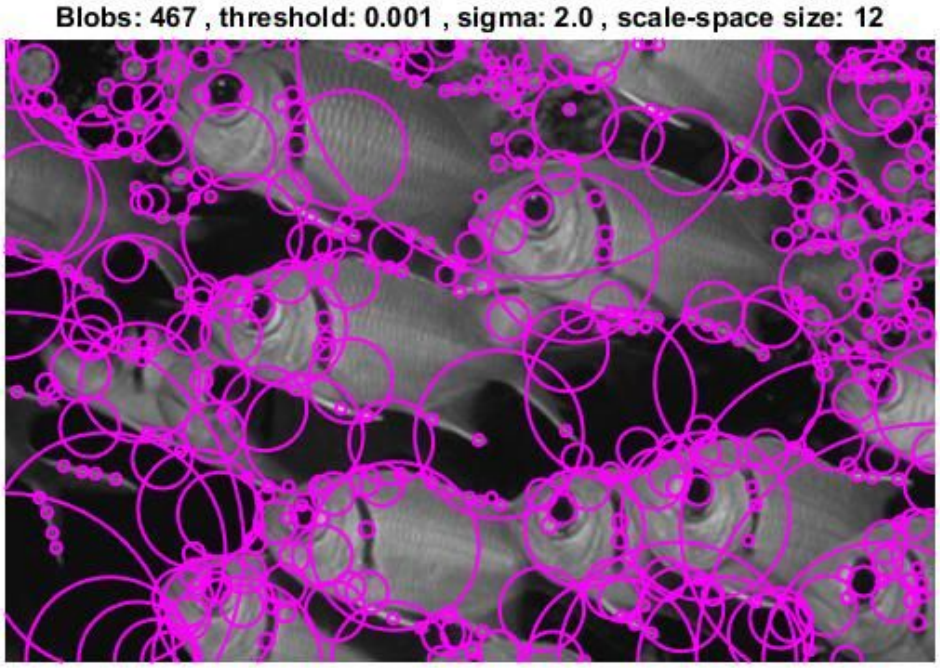
The Einstein's image capture the blobs as expected around his hand, equation, eyes and other corner points. By observing 'R', we can also see it captures the edge points.





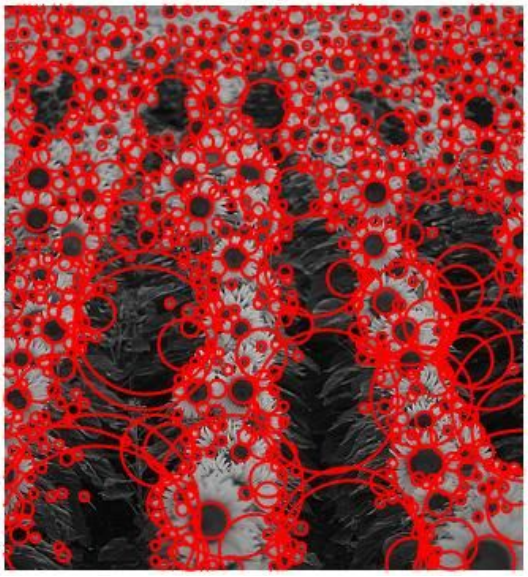
The gills and fishes eyes are captured perfectly by the blob detector. There are some bigger blobs that present all over the image which majorly connects two or more corners in the image.



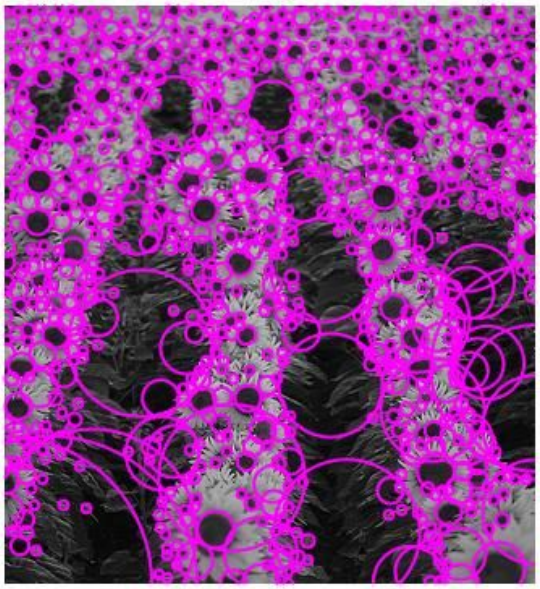


The center of the sunflowers are consistently inside the blob and lob detector works well to capture them.

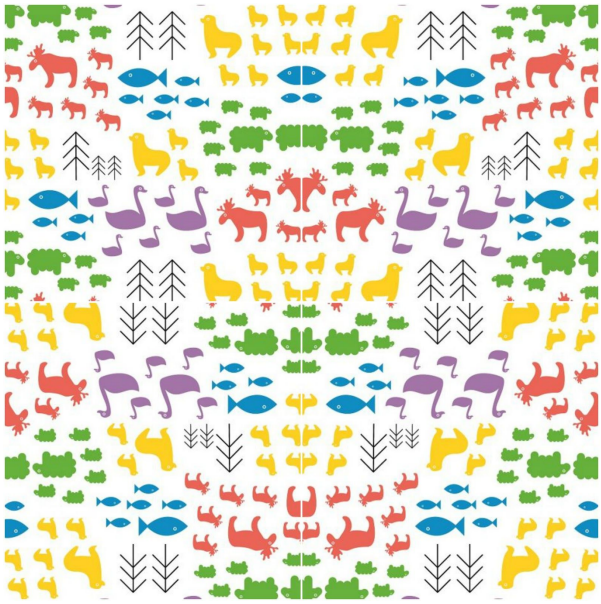
Blobs: 905 , threshold: 0.001 , sigma: 2.0 , scale-space size: 12



Blobs: 887 , threshold: 0.001 , sigma: 2.0 , scale-space size: 12



For other implementations, following four images are chosen. The reasons for each of the images are given below.

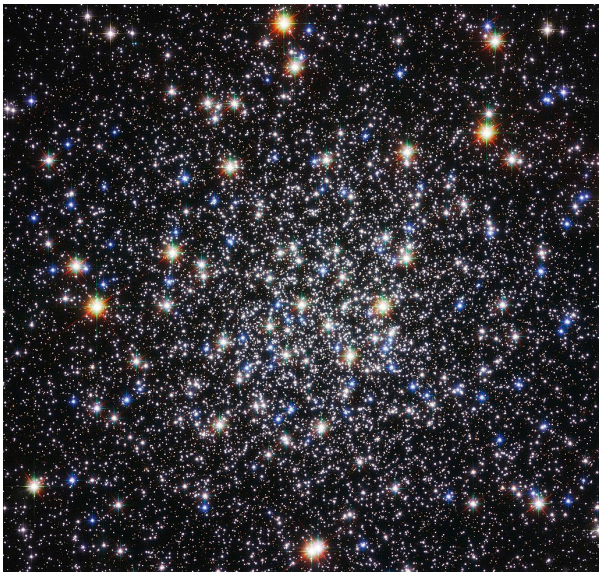


Animal Pattern

I have chosen this image to evaluate the orientation change of a pattern (here pattern is in the form of animals) and its response by blog detector.

The image contains minimalistic view of animals mirrored in all four directions.

This analysis would help in understanding the invariance of blob detector to orientation.

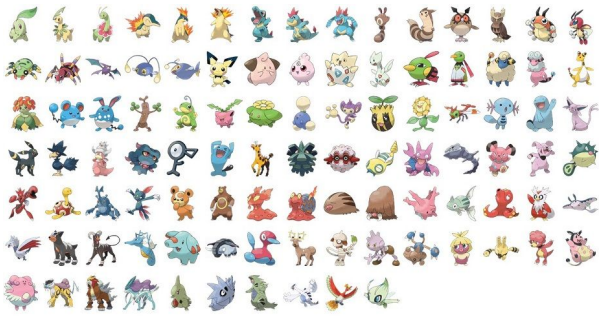


Hubble Telescope

This image is the original iconic picture from hubble space telescope. This contains variety of celestial bodies in outer space.

Noise in the image is high and each point in the image is some star or planet. The bigger clusters are galaxies.

This analysis would help in figuring out how blob detection work in noisy data.



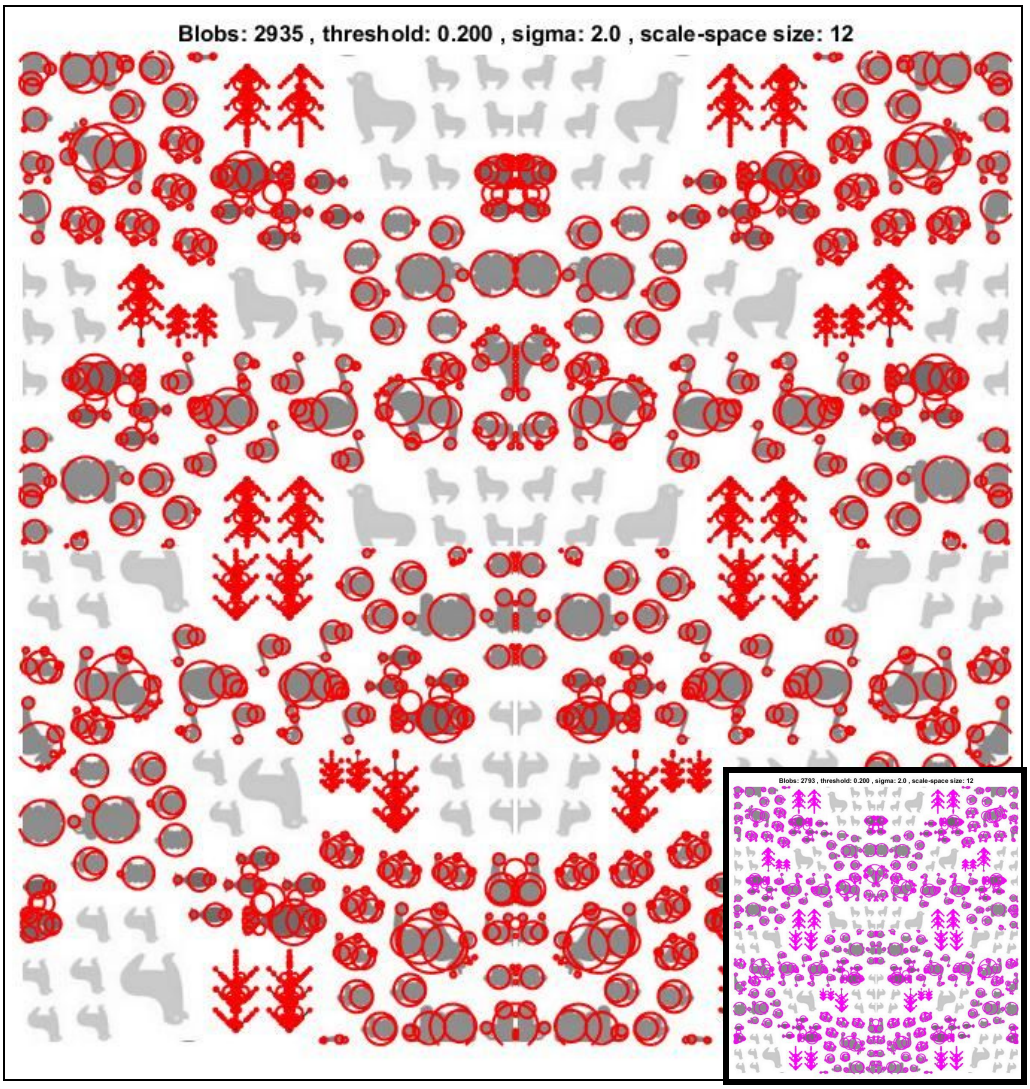
Pokemon

Collection of some pokemon in this image would help in figure out the blob detector response to images with plane background with non repeatable pattern.

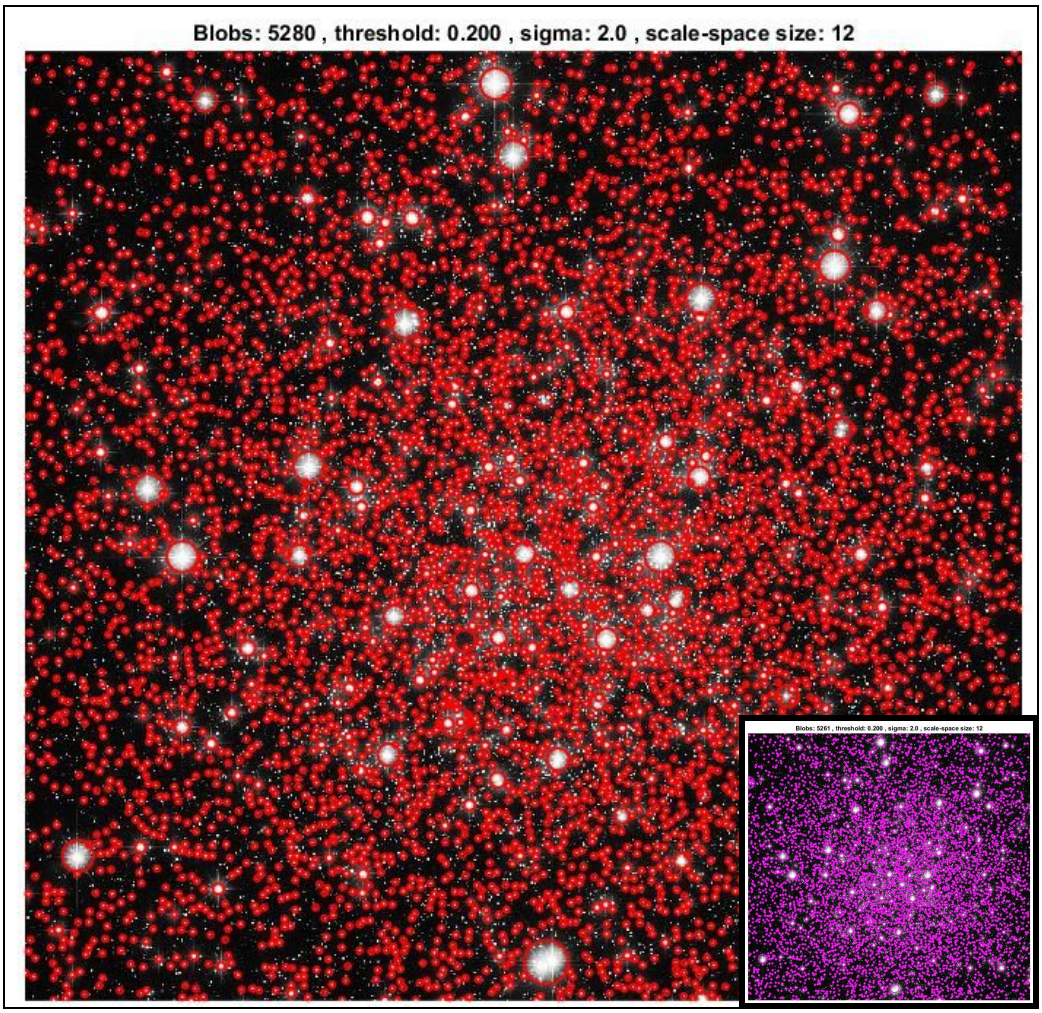
Grand Theft Auto 5

This image have 3 different animated characters from famous video game. Through this image we could find the blob detector response on human like figures.

Outputs of each implementations are given below. The images shown below have a single image with red circles representing the up scale (inefficient) implementation and in-doc small image with magenta circles representing the down scale (efficient) implementation.

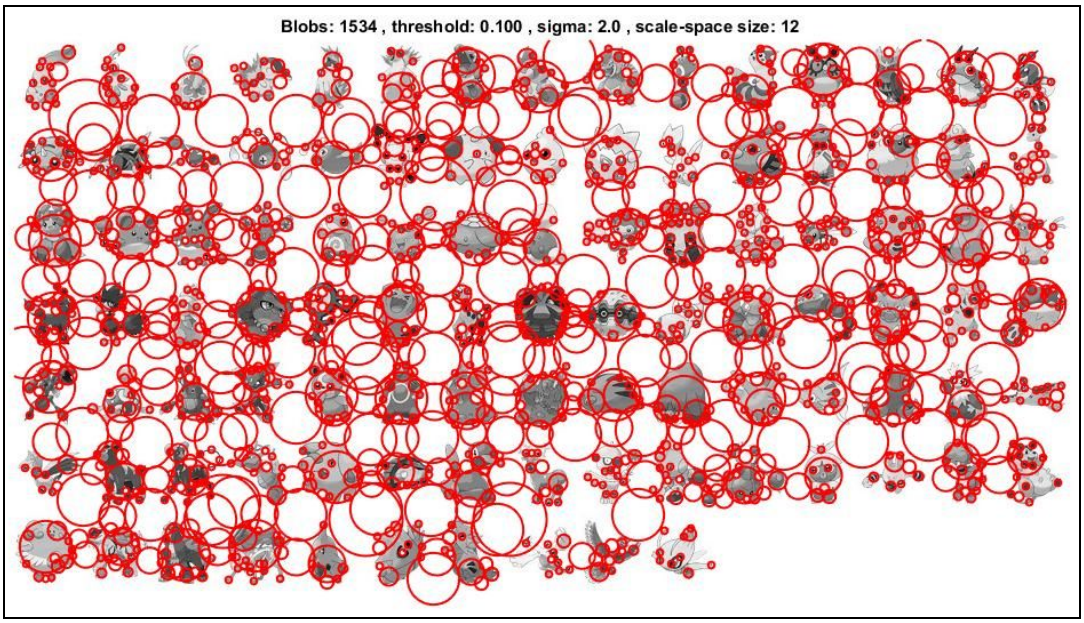


The animal pattern image above is obtained by setting sigma at 2 and threshold at 0.2. This image is one of the largest image in size, hence the hyperparameters are chosen to reduce the running time and give reasonable results. We can observe the repeatable patterns in output image. The blobs are also mirrored as the image. This implies that the blob detector is invariant to the orientation. There are certain portions in the image containing no blobs. This could be the result of the high threshold value. Further analysis is done in the analysis section.



The output of hubble telescope image is shown above. I have used a high threshold of 0.2 to reduce the number of circles. We can observe that mostly all stars are captured inside the blob. This might reflect some kind of overfitting of the detector response. This could be explained as the detector interpret these stars as invariant location in the image space.

The figure below is the detector response for the pokemon image, The detector works well with threshold value of 0.1. Almost all of the pokemon are perfectly captured with their characteristic shape in one of the circle by blob detector.



The grand theft auto 5 (video game) image depicts circles capturing key corner points by using blob detector with threshold of 0.2. The circles also captures the strong edge points present in the image. The logo has the most number of corners (as expected).

ANALYSIS OF THE RESULTS

The table given below is detailed description of each image. The table is color coded, representing orange as given images and green as chosen images.

Number	Image Title	Height [pixels]	Width [pixels]	Size [KB]
1	Fishes	500	335	40
2	Einstein	640	480	93
3	Sunflowers	328	357	141
4	Butterfly	493	356	135
Number	Image Title	Height [pixels]	Width [pixels]	Size [KB]
5	AllPokemon	920	496	181
6	HubbleOriginal	1024	974	501
7	GTA5	2880	1800	1381
8	AnnimalPattern	3988	3988	4278

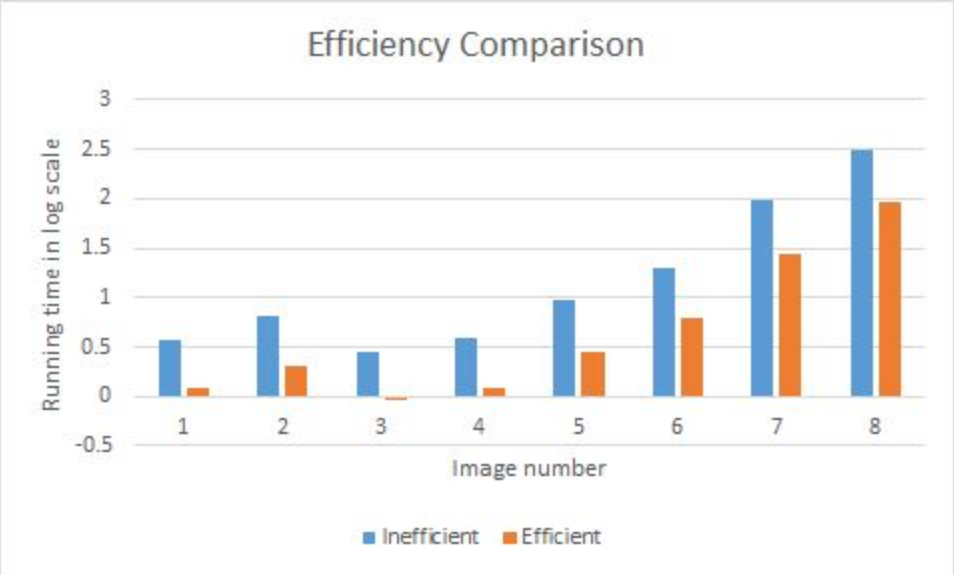
The overall comparison table is given below.

Number	Image	Sigma	SpaceStep	K	Threshold	Method	Blobs	Run_Time (sec)
1	Fishes	2	12	sqrt(2)	0.001	inefficient	476	3.79
1	Fishes	2	12	sqrt(2)	0.001	efficient	467	1.2
2	Einstein	2	12	sqrt(2)	0.001	inefficient	700	6.38
2	Einstein	2	12	sqrt(2)	0.001	efficient	737	1.99
3	Sunflowers	2	12	sqrt(2)	0.001	inefficient	905	2.78
3	Sunflowers	2	12	sqrt(2)	0.001	efficient	887	0.912
4	Butterfly	2	12	sqrt(2)	0.001	inefficient	706	3.87
4	Butterfly	2	12	sqrt(2)	0.001	efficient	669	1.24
5	AllPokemon	2	12	sqrt(2)	0.1	inefficient	1534	9.29
5	AllPokemon	2	12	sqrt(2)	0.1	efficient	1518	2.81
6	HubbleOriginal	2	12	sqrt(2)	0.2	inefficient	5280	19.91
6	HubbleOriginal	2	12	sqrt(2)	0.2	efficient	5261	6.14
7	GTA5	2	12	sqrt(2)	0.2	inefficient	3941	98
7	GTA5	2	12	sqrt(2)	0.2	efficient	3596	28
8	AnnimalPattern	2	12	sqrt(2)	0.2	inefficient	2935	312.8
8	AnnimalPattern	2	12	sqrt(2)	0.2	efficient	2793	91.1

We can clearly observe that the inefficient method have high running time as comparison to the efficient method. One of the key feature that I have observed is that sigma is one of the reason for high computation time.

As the images listed from 1 to 8 are arranged in ascending order of image size. Hence, the figure below depicts the running time (in sec) with respect to the image size.

As the running time varies from 0.9 sec to 300 seconds, a direct comparison is not prudent. Therefore a log-scale comparison is made in the running time as efficiency in the graph below.



DISCUSSION

Here are few points that are essential to discuss this implementation.

1. PARAMETERS SELECTION :

- **Batch :** I experimented with quite a few values of the matrix size used for 2D non maximum suppression. I varied this parameter from 3x3 to 11x11 and noted few observations. There is a tradeoff between accurate blobs and running time. 3x3 runs very fast but produce unnecessary blobs. Whereas higher size filter, say 11x11, takes exponentially more time. Hence, I found the optimal to be 7.
- **Threshold :** This parameter majorly decides the number of circles. I changed this parameters depending on size of image. I have used 0.001 for most of the images.
- **Sigma :** Larger sigma values detects larger features in the image whereas smaller sigma value detect smaller features in the image. I used sigma of 2 and found to give better results over others.
- **Space Step :** This is a major contributor to running time as well. I realized any value from 10 to 15 is good for implementation.
- **Hyperparameter k :** I also played around with this hyper parameter. Some pictures are giving me better results with value of $k = 2^{0.3}$. But, I used $k = 2^{0.5}$ as conventional.

2. RESULTS, FINDINGS AND COMMENTS :

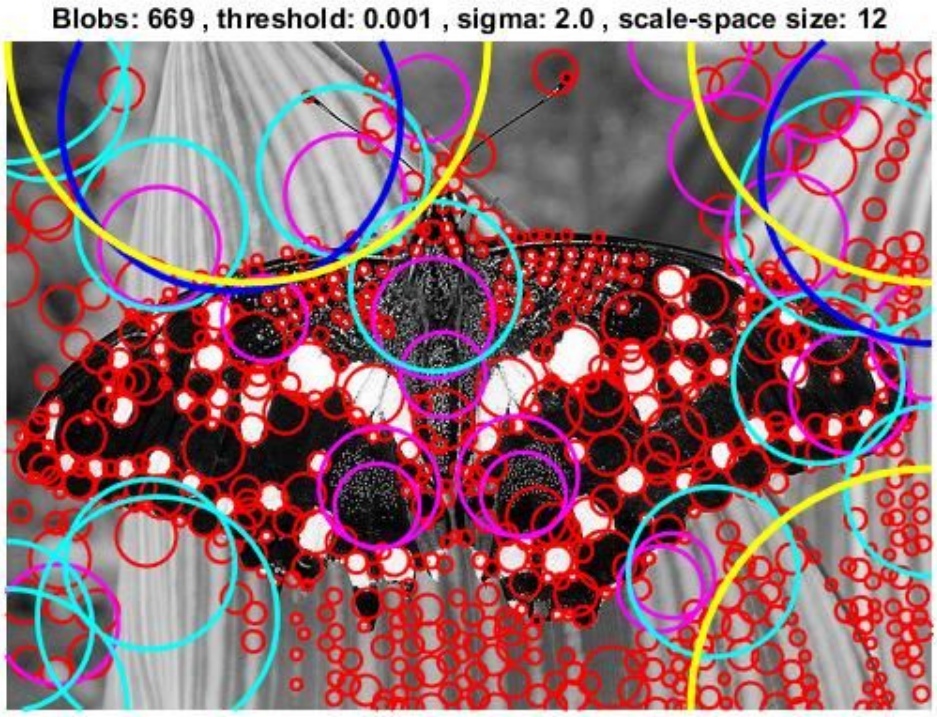
- Instead of using conventional squaring method for LoG filtering, I have used absolute value instead i.e. I have converted the negative values into the positive values using ‘abs’ function of the matlab. I found that similar results are obtained by slightly decreasing the threshold value. This is understandable as squaring makes the values of pixels much smaller. I noted absolute function is more efficient and there is a slightly better performance with this.
- One interesting point that I noticed is that we need to use harris detector to remove the strong edge points for removal purposes as they are also getting detected by the blob detector.

EXTRA CREDIT

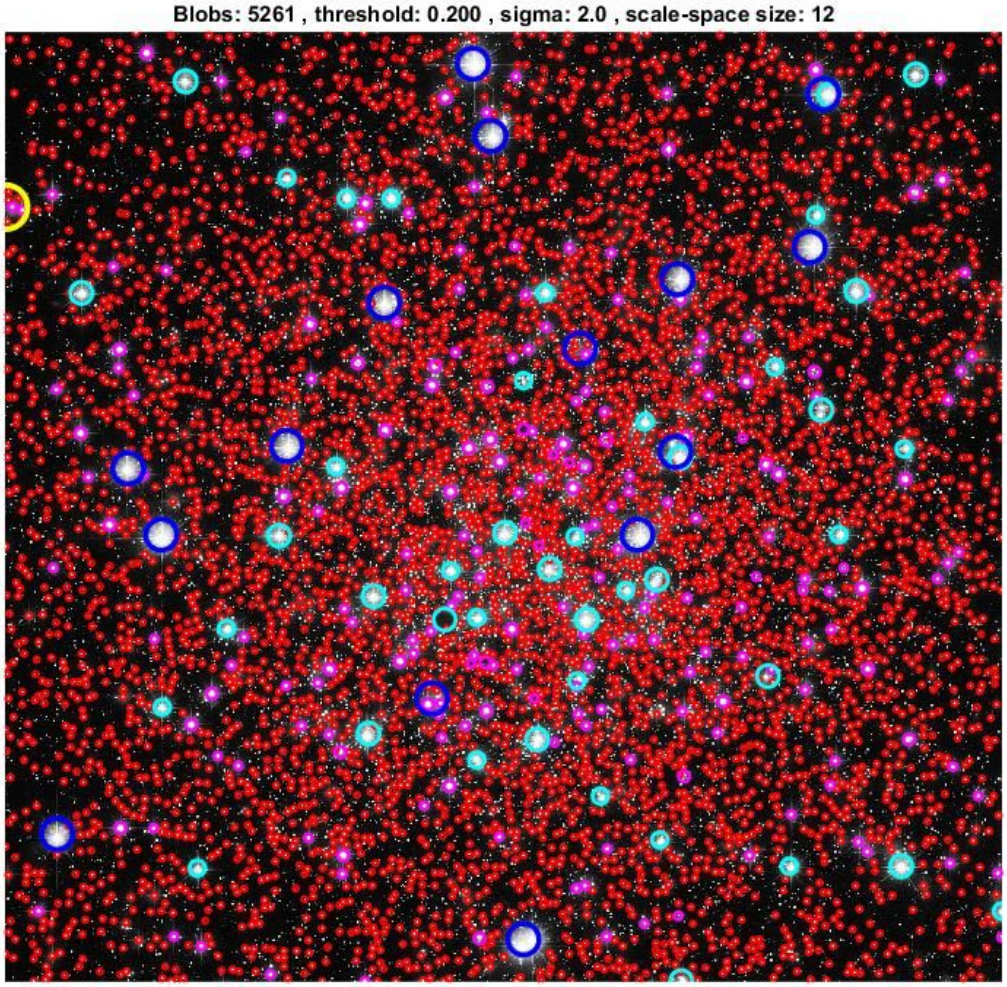
Additional implementation : Categorizing the blobs into levels in image space.

Motivation : After implementing hubble telescope's image, blob detector produces bunch of circles that correctly fits the logic behind the blob detection, but could be useless to extract useful information. I realize that there should be a process of classification on levels of corner according to the size of blob.

Concept : The size of the blobs represent information of bigger edges and inverse is true for smaller blobs. This information could be useful for some image analysis. Hence, I performed 6 level categorization of blobs based on the size of the blob and its distribution. [This is a hyper parameter in the code]. I have also color coded each of the category to make it visually apparent. Here are some of the results.

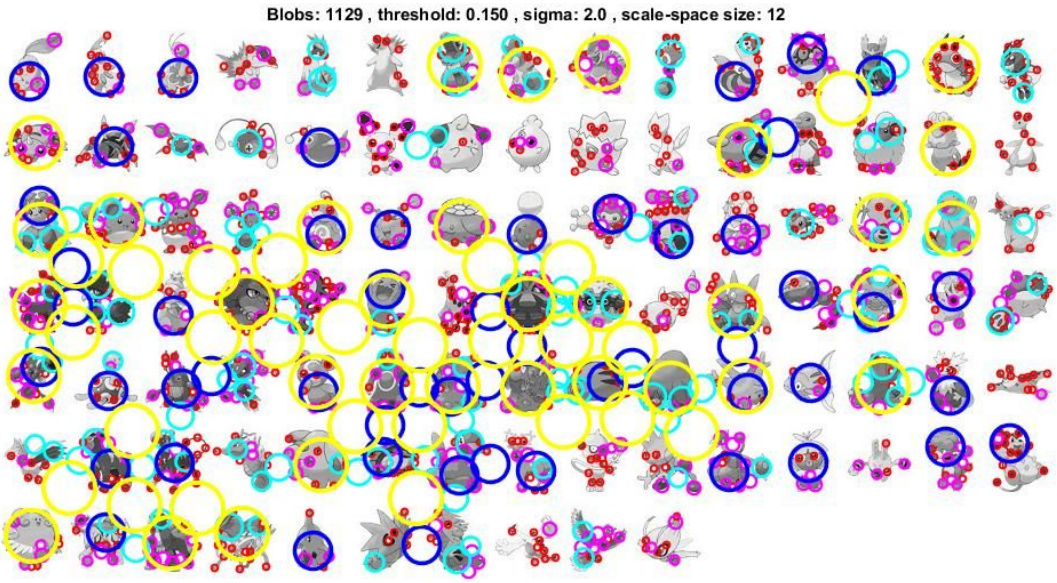


The image below is the classification on hubble telescope's picture.



We can clearly see that now information are more apparent than before. Mostly brighter stars are getting captured in green circles. According to hubble's image release documentation, the galaxies are bigger and brighter in shape. Most of the blobs with blue circles are able to capture those galaxies (with some exceptions with green circle overlap).

This demonstrate that categorical analysis is important for both data retrieval and visual analysis.



I also experimented with pokemon image (as an area of interest of researcher). Most pokemons with circular shape as a dominant part of their body are within blue and yellow circles. This provides visual and categorical description from image data.

[other images are presented in .zip file]

REFERENCES

1. UIUC CS 543 - Computer Vision lecture notes.
2. David G Lowe. Distinctive image features from scale-invariant keypoints.
3. Sample Harris detector code