CSE 573 HOME WORK 2 SANTOSH PHANI VISHNU AITA Person. NO : 50246573

1.The output of your circle detector on all the images (four provided and four of your own choice), together with a comparison of running times for both the "efficient" and the "inefficient" implementation. (The comparison is graded, not the running times.)

Notation:

Method 1 :

In this case the last parameter suggests that method 1 is opted. i.e. filter size is increased after every iteration.

>> blobDetection(IM,1.3,1.3,15,0.0017,1)

Method 2 :

In this case the last parameter suggests that method 2 is opted. i.e. filter size is kept constant and image is down sampled after every iteration.

>> blobDetection(IM,1.3,1.3,15,0.0017,2)

OUTPUTS

Butterfly.jpg Method 1

>> blobDetection(IM,1.3,1.3,15,0.0017,1)

Elapsed time is 7.591153 seconds.

Butterfly.jpg Method 2

>> blobDetection(IM,1.3,1.3,15,0.0017,2)

Elapsed time is 0.203602 seconds.

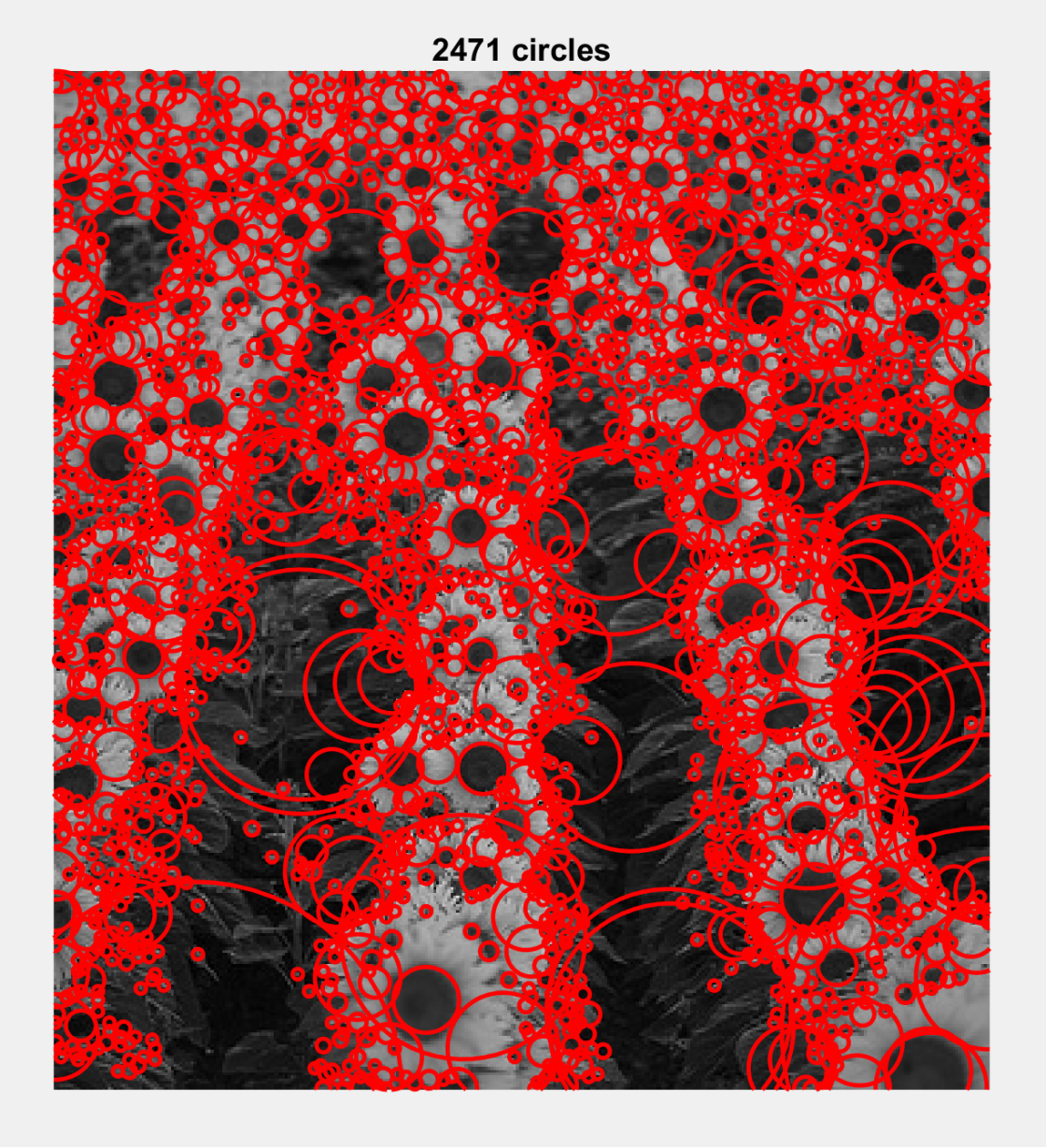
By above computation we can say that Method 2 is more efficient as it took less time and it is because applying filter on the down sampled image takes less resources and also computation for smaller images is much faster.

Values sigma = 1.3, k = 1.3, N = 15, threshold= 0.0017, method = 1 and 2. (efficient when method 2).

sunflowers.jpg Method 1

>> blobDetection(IM,1.3,1.3,15,0.0017,1)

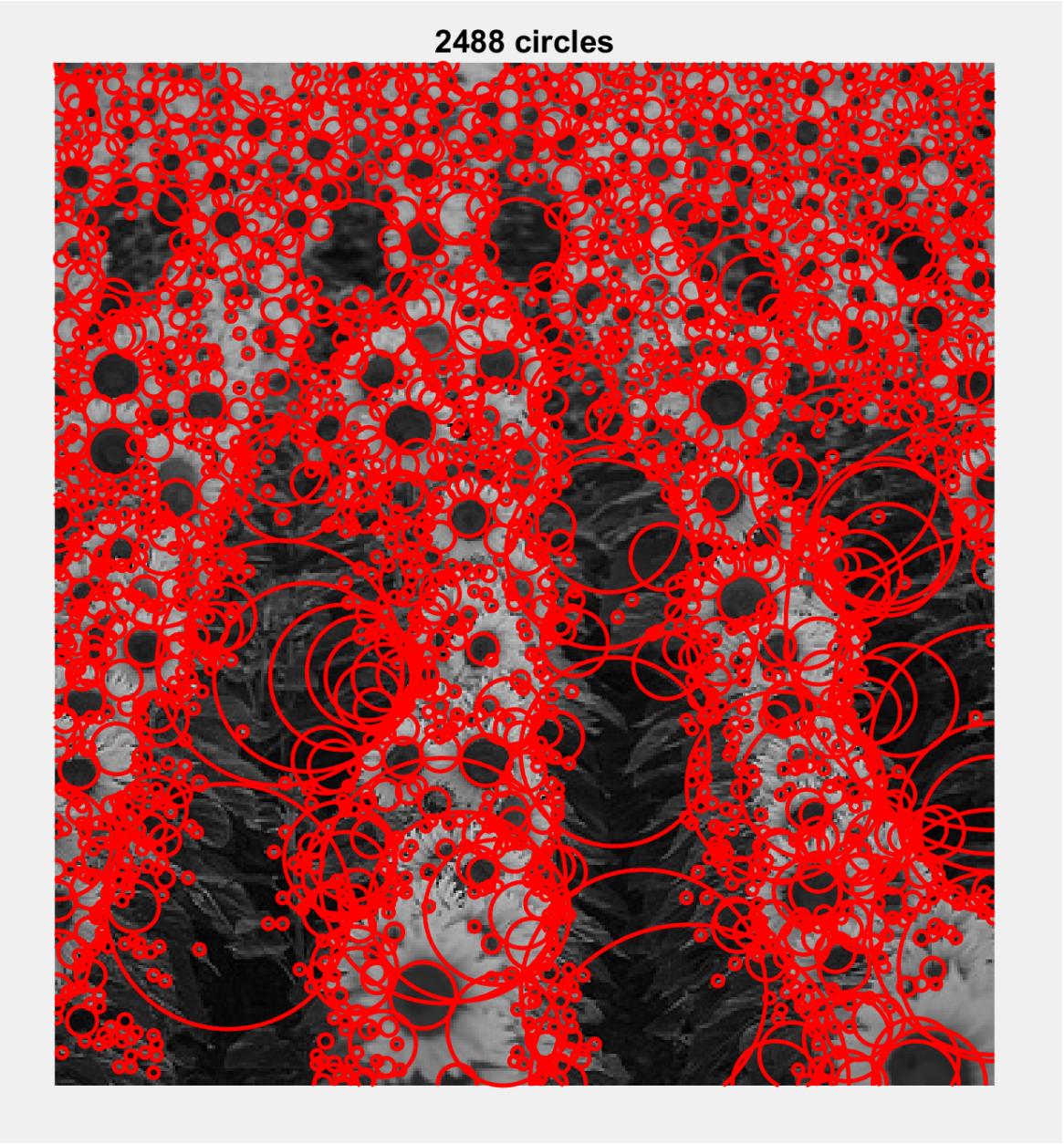
Elapsed time is 4.917522 seconds.



sunflowers.jpg Method 2

>> blobDetection(IM,1.3,1.3,15,0.0017,2)

Elapsed time is 0.169994 seconds.



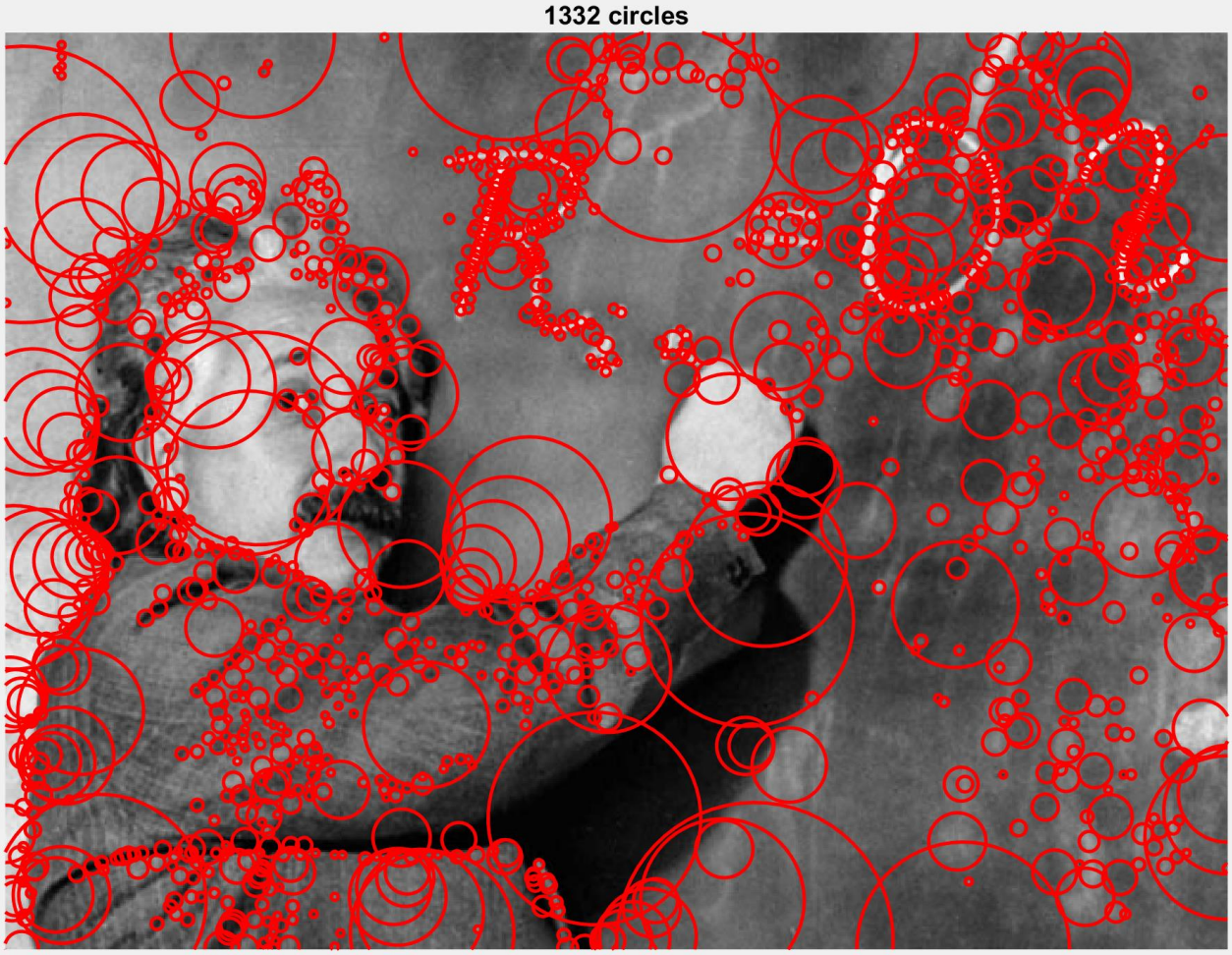
By above computation we can say that Method 2 is more efficient as it took less time and it is because applying filter on the down sampled image takes less resources and also computation for smaller images is much faster.

Values sigma = 1.3, k = 1.3, N = 15, threshold= 0.0017, method = 1 and 2. (efficient when method 2).

einstein.jpg Method 1

>> blobDetection(IM,1.3,1.3,15,0.0017,1)

Elapsed time is 10.708645 seconds.



Einstein.jpg Method 2

>> blobDetection(IM,1.3,1.3,15,0.0017,2)

Elapsed time is 0.278363 seconds.

By above computation we can say that Method 2 is more efficient as it took less time and it is because applying filter on the down sampled image takes less resources and also computation for smaller images is much faster.

Values sigma = 1.3, k = 1.3, N = 15, threshold= 0.0017, method = 1 and 2. (efficient when method 2).

fishes.jpg Method 1

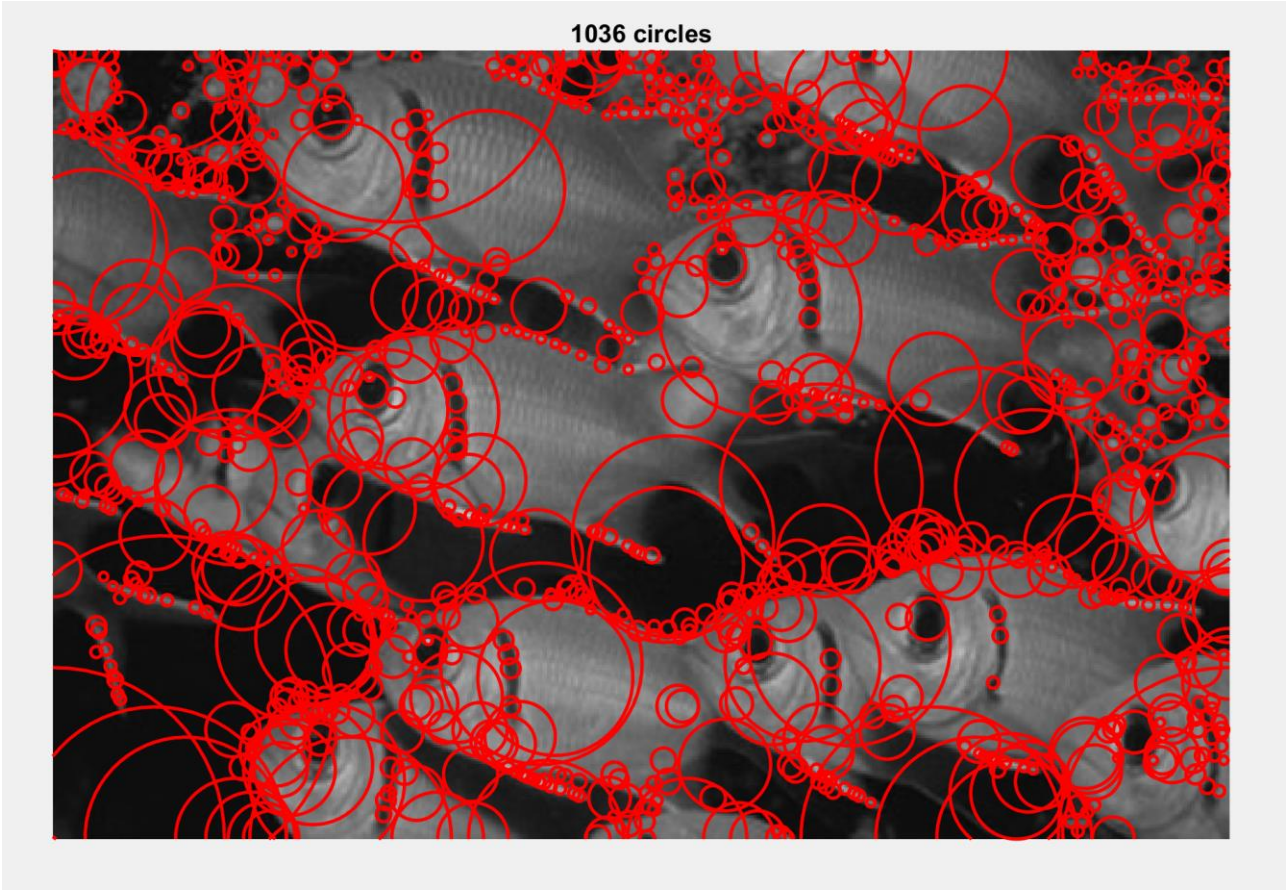
>> blobDetection(IM,1.3,1.3,15,0.0017,1)

Elapsed time is 7.041538 seconds.

fishes.jpg Method 2

>> blobDetection(IM,1.3,1.3,15,0.0017,2)

Elapsed time is 0.192894 seconds.



By above computation we can say that Method 2 is more efficient as it took less time and it is because applying filter on the down sampled image takes less resources and also computation for smaller images is much faster.

Values sigma = 1.3, k = 1.3, N = 15, threshold= 0.0017, method = 1 and 2. (efficient when method 2).

The following four images are really large. So, it took more time comparatively.

i_001.jpg Method 1

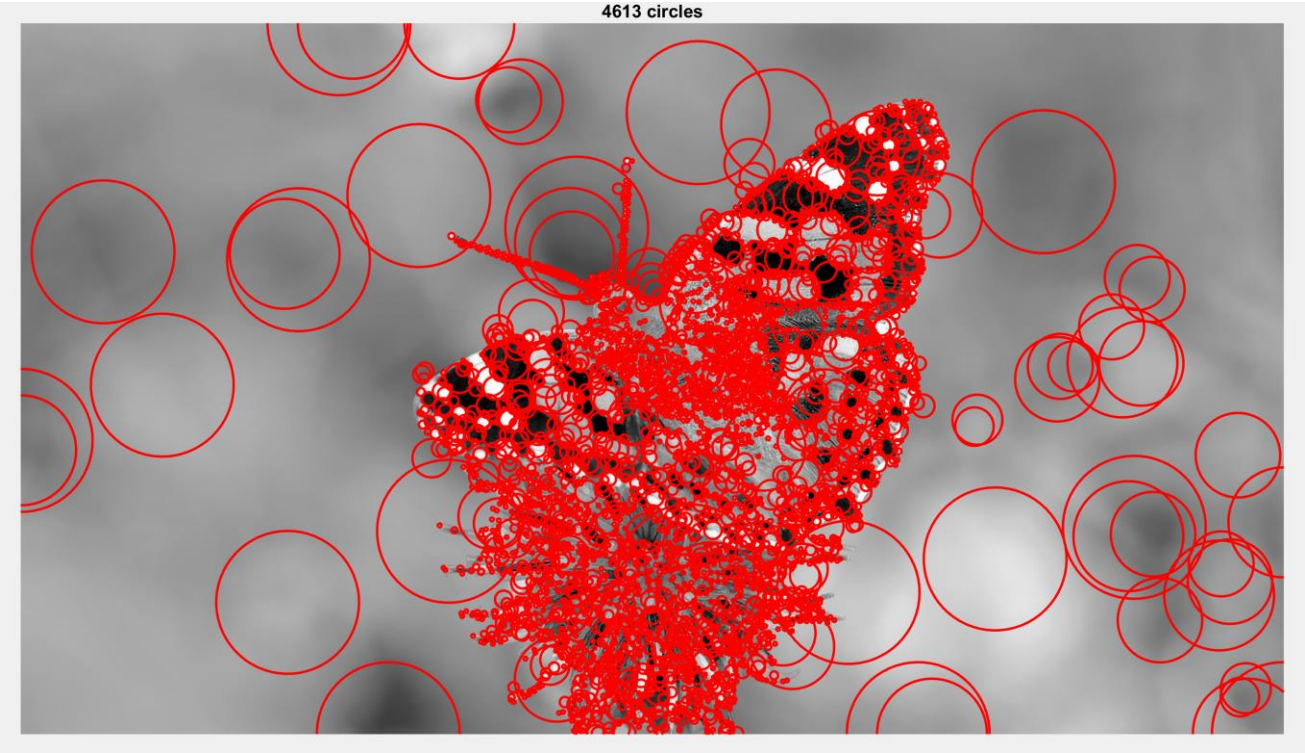
>> blobDetection(IM,1.3,1.3,15,0.0017,1)

Elapsed time is 32.880955 seconds.

i_001.jpg Method 2

>> blobDetection(IM,1.3,1.3,15,0.0017,2)

Elapsed time is 0.573536 seconds.



By above computation we can say that Method 2 is more efficient as it took less time and it is because applying filter on the down sampled image takes less resources and also computation for smaller images is much faster.

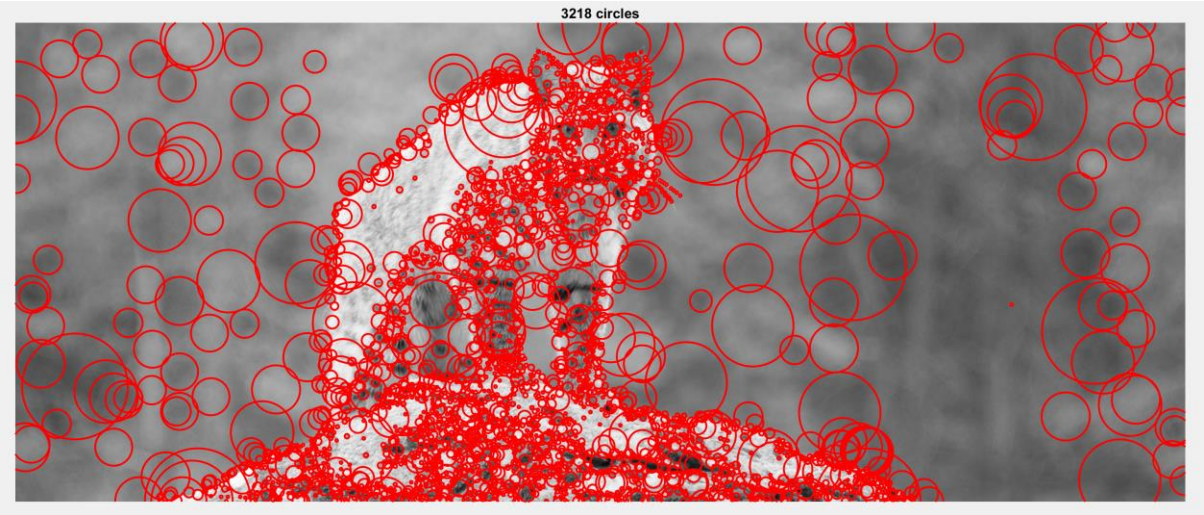
Values sigma = 1.3, k = 1.3, N = 15, threshold= 0.0017, method = 1 and 2. (efficient when method 2).

The following three images are really large. So, it took more time comparatively.

i_002.jpg Method 1

>> blobDetection(IM,1.3,1.3,15,0.0017,1)

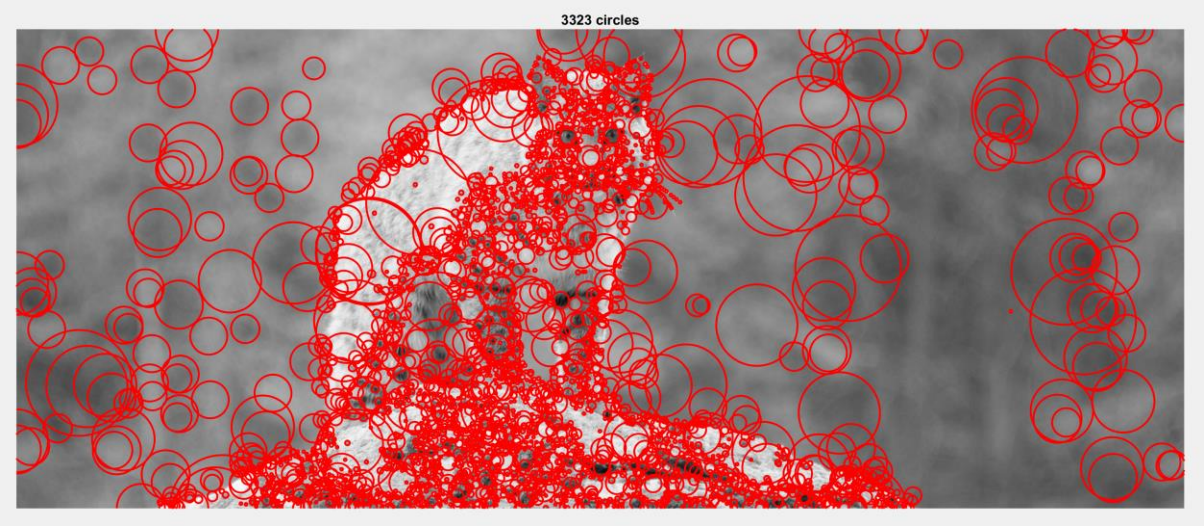
Elapsed time is 37.764858 seconds.



i_002.jpg Method 2

>> blobDetection(IM,1.3,1.3,15,0.0017,2)

Elapsed time is 0.631240 seconds.



By above computation we can say that Method 2 is more efficient as it took less time and it is because applying filter on the down sampled image takes less resources and also computation for smaller images is much faster.

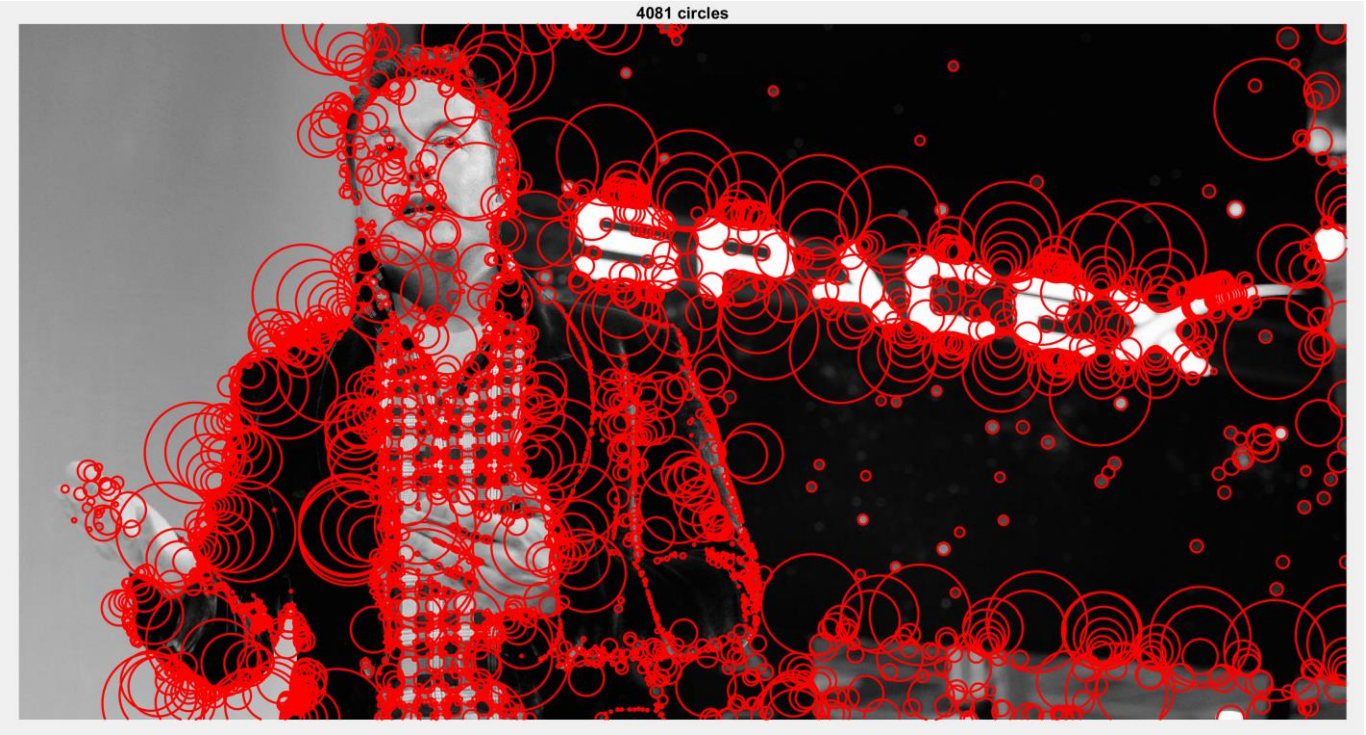
Values sigma = 1.3, k = 1.3, N = 15, threshold= 0.0017, method = 1 and 2. (efficient when method 2).

The following two images are really large. So, it took more time comparatively.

i_003.jpg Method 1

>> blobDetection(IM,1.3,1.3,15,0.0017,1)

Elapsed time is 66.825546 seconds.



i_003.jpg Method 2

>> blobDetection(IM,1.3,1.3,15,0.0017,2)

Elapsed time is 0.967643 seconds.

By above computation we can say that Method 2 is more efficient as it took less time and it is because applying filter on the down sampled image takes less resources and also computation for smaller images is much faster.

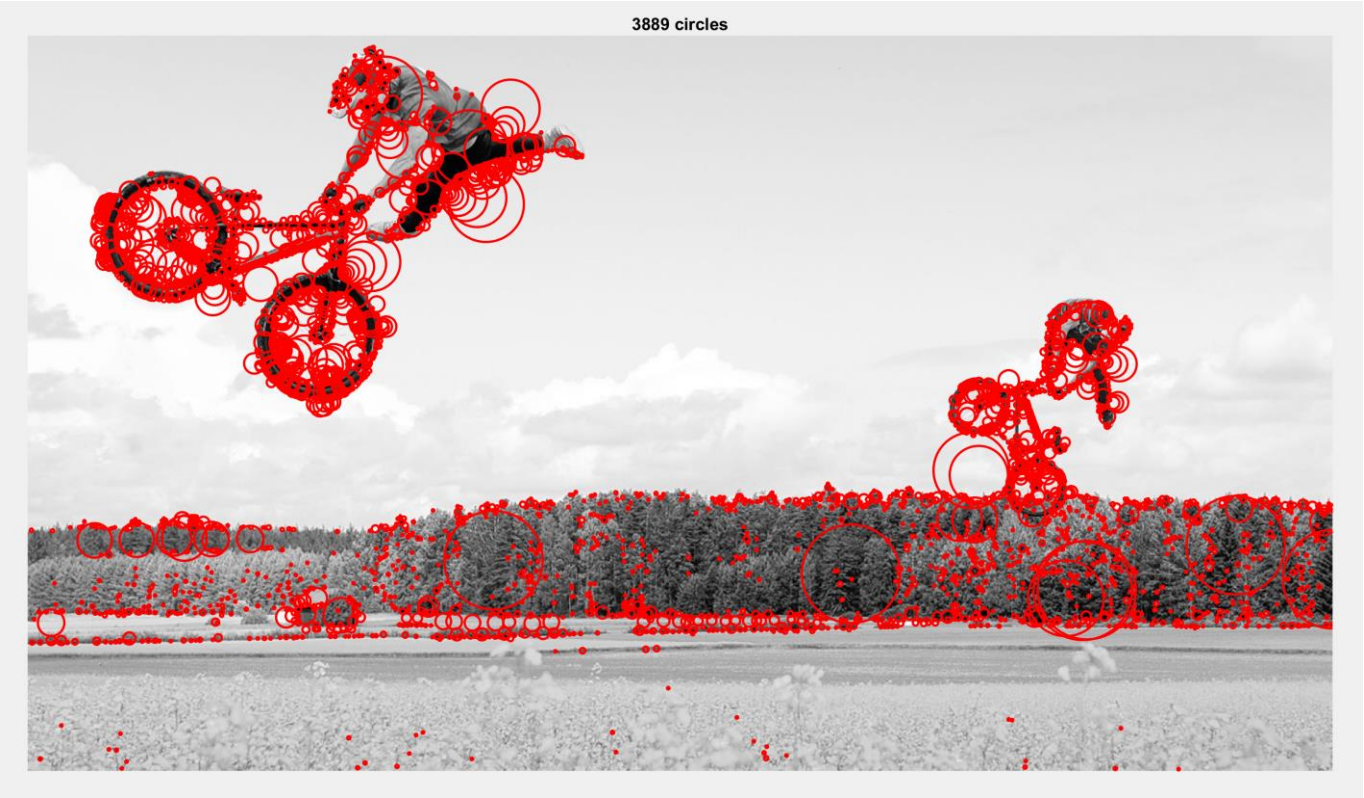
Values sigma = 1.3, k = 1.3, N = 15, threshold= 0.0017, method = 1 and 2. (efficient when method 2).

The following image is really large. So, it took more time comparatively.

i_004.jpg Method 1

>> blobDetection(IM,1.3,1.3,15,0.0017,1)

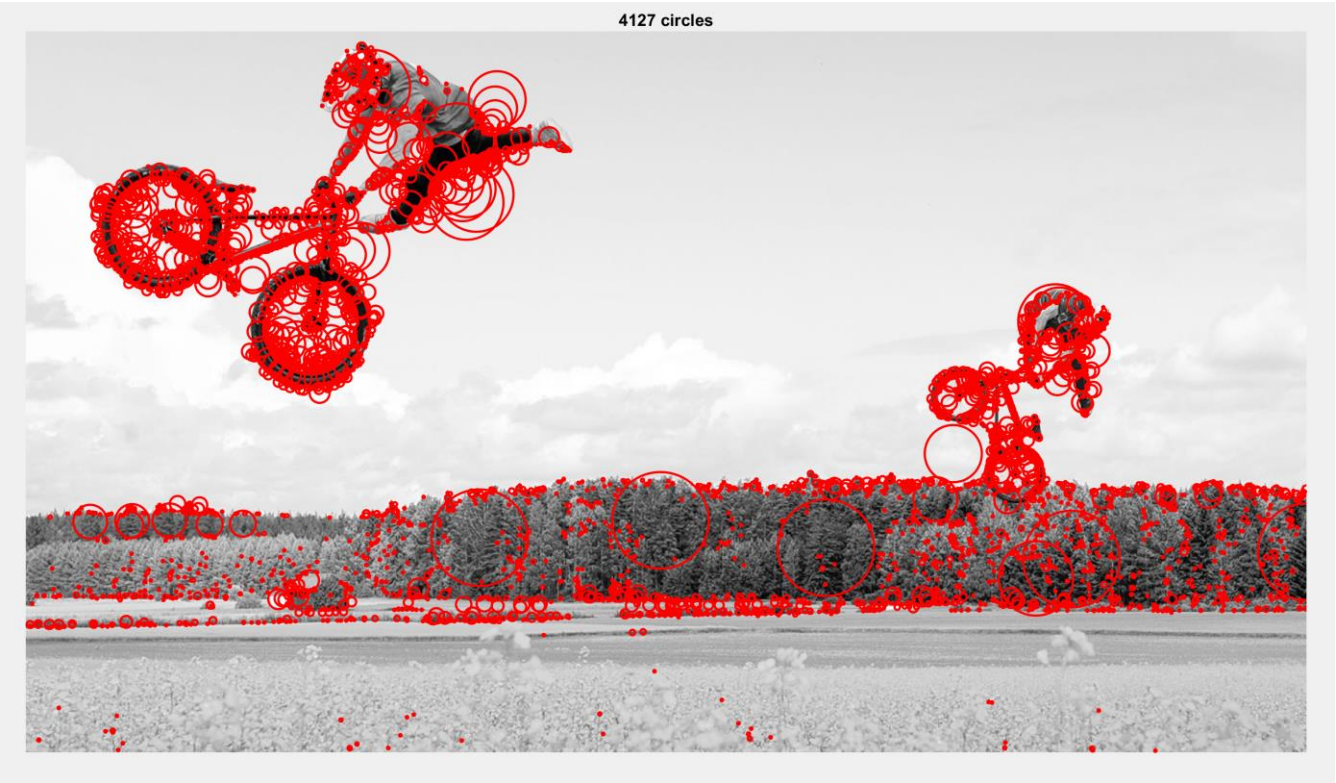
Elapsed time is 76.737888 seconds.



i_004.jpg Method 2

>> blobDetection(IM,1.3,1.3,15,0.0017,2)

Method 2: Elapsed time is 1.203127 seconds.



By above computation we can say that Method 2 is more efficient as it took less time and it is because applying filter on the down sampled image takes less resources and also computation for smaller images is much faster.

Values sigma = 1.3, k = 1.3, N = 15, threshold= 0.03, method = 1 and 2. (efficient when method 2).

2.An explanation of any "interesting" implementation choices that you made.

Out of the two methods, the down sampling method is very optimal as it comparatively took less resources and time.

While down sampling passing 'bicubic' as a parameter to imresize() helped smoothen the images to generate better output as it retains spatial detailing in a better way.

While applying filter using imfilter(), passing 'replicate' as a parameter helped replicate the borders. So that blobs around the borders can be reduced.

Ordfilt2() is a more efficient method to perform non maxima suppression on the 2D scales and it doesn't need any other max function like nlfilter().

My method to evaluate max is really simple. I just found the maximum value of each pixel in the entire 3D scale_space. Then retained those values as a 2D matrix called maximas

```
Maxima_space = maximas.*(maximas == scale_space);
```

I simply calculated the final maxima scale space as shown above.

Then I computed centers and radius using find() and finally output the circles on the images.

3.An explanation of parameter values you have tried and which ones you found to be optimal.

Initially when the sigma value and k value were taken as 2,2 respectively and the total no. of levels taken was 15. So, it was evident that after each iteration the value of sigma increased phenomenally as it was getting multiplied by k= 2 every time, thus increasing the size of the filter and leading to more computational time and using more processing power.

After trying out different values for k. It was obvious that it was optimal when k ranged between 1 and 2. (also if k<1 then the filter size is getting reduced each time. This is wrong).

Tuning threshold properly to generate desired no. of circles on the output which is nearly accurate took some work. Initially when threshold was 0.5 not many circles were displayed. So after reducing the value of threshold in small parts. I found the optimal threshold value suitable for my system as 0.0017.

This combination of parameters took about 8 seconds to generate output in the method 1 and drastically less time of about 0.15 seconds for the second method to generate output.