CSE 573 - Computer Vision and Image Processing (Report)

Blog-Detection

* **Name:** Sujit Singh
* **UBIT Name:** sujitjit
* **UBIT:** # 50247206

1. Comparision of the efficient and inefficient method

The comparision between the efficient and the ineffcient is done as per below images:

**Threshold** = 0.02

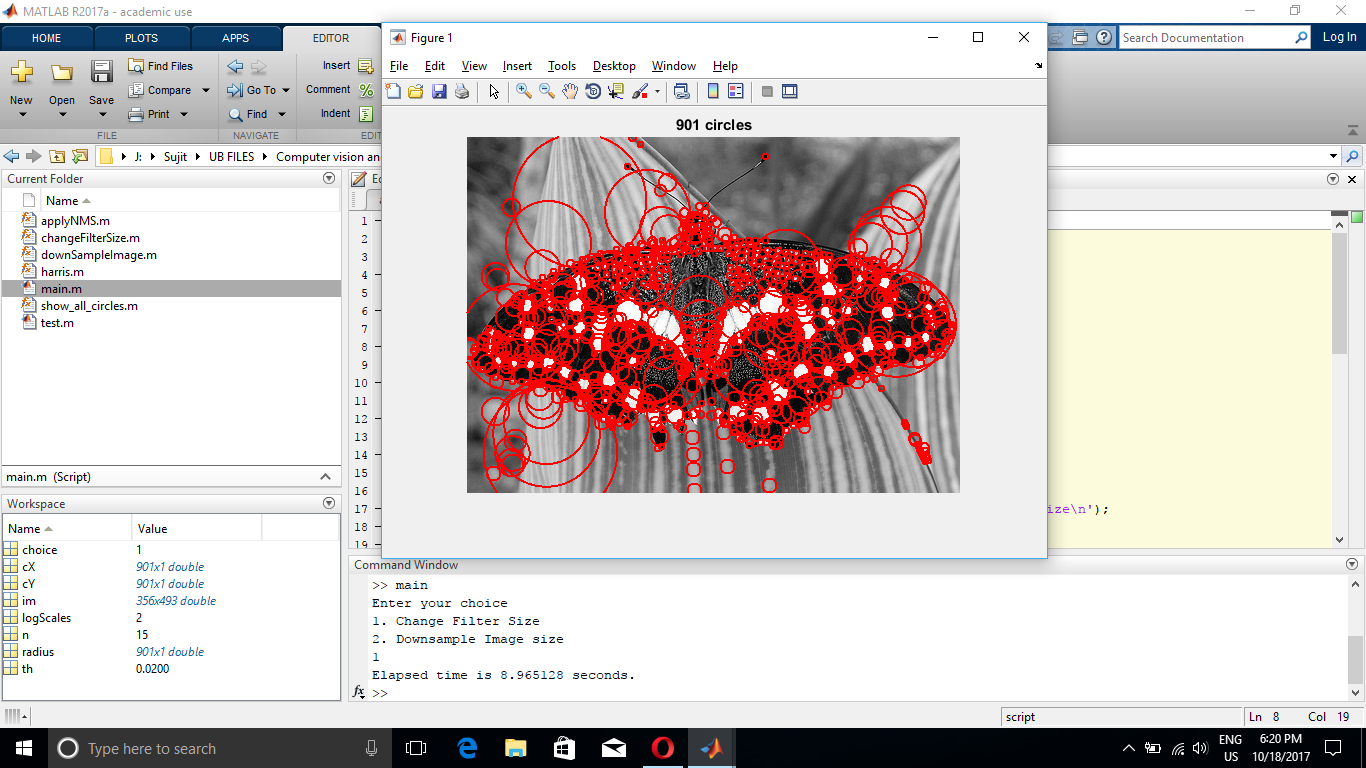
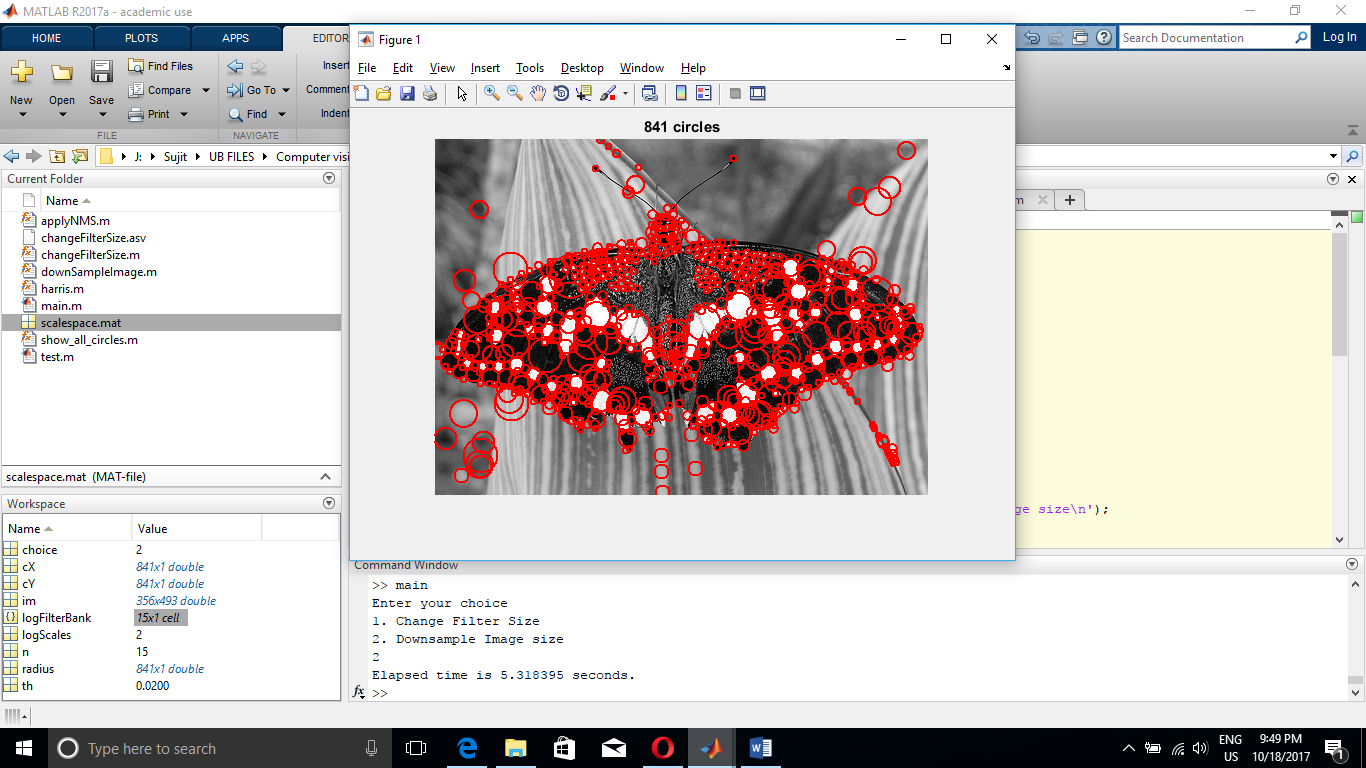
 

Fig 1.1 Butterfly (Filter upsize) Fig 1.2 Butterfly (Image down sample)

**Running Time:** 8.97s **Running Time:** 3.29s

**Threshold** = 0.0075

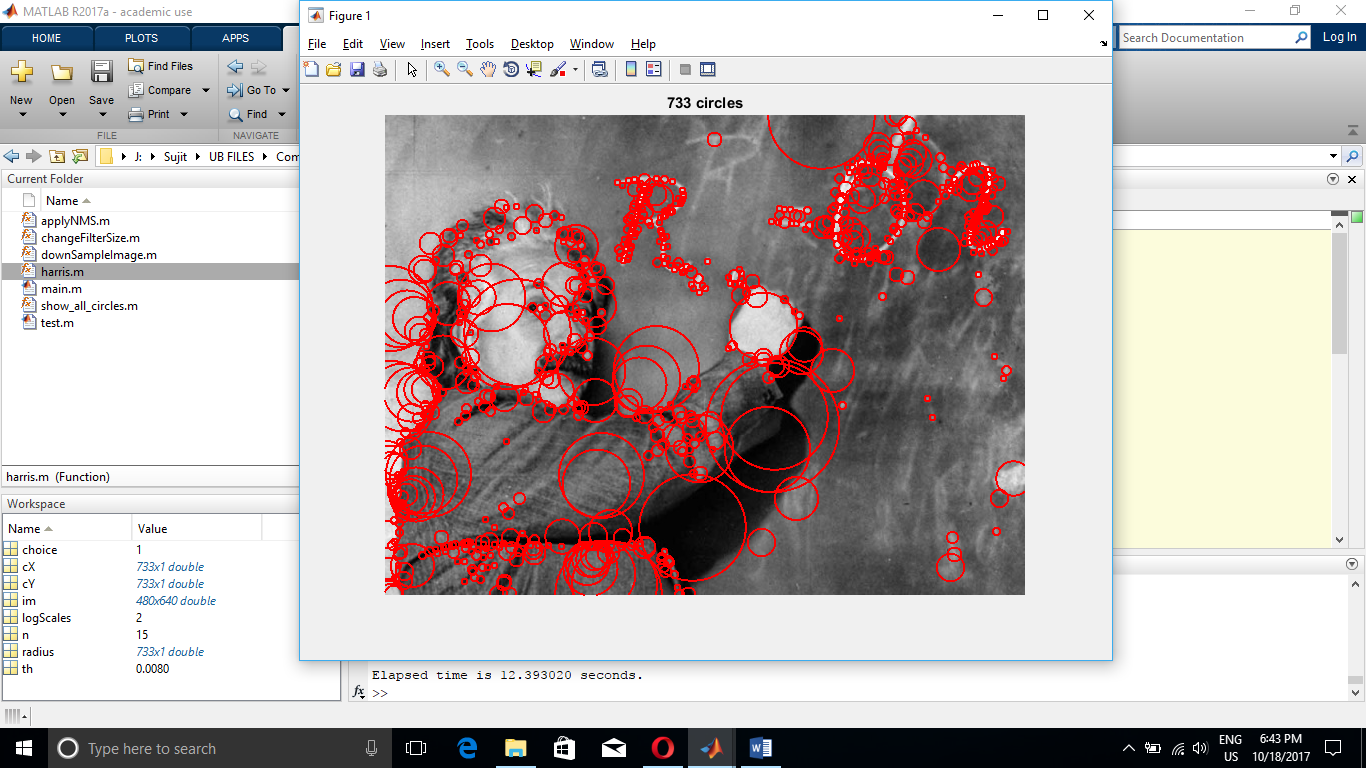
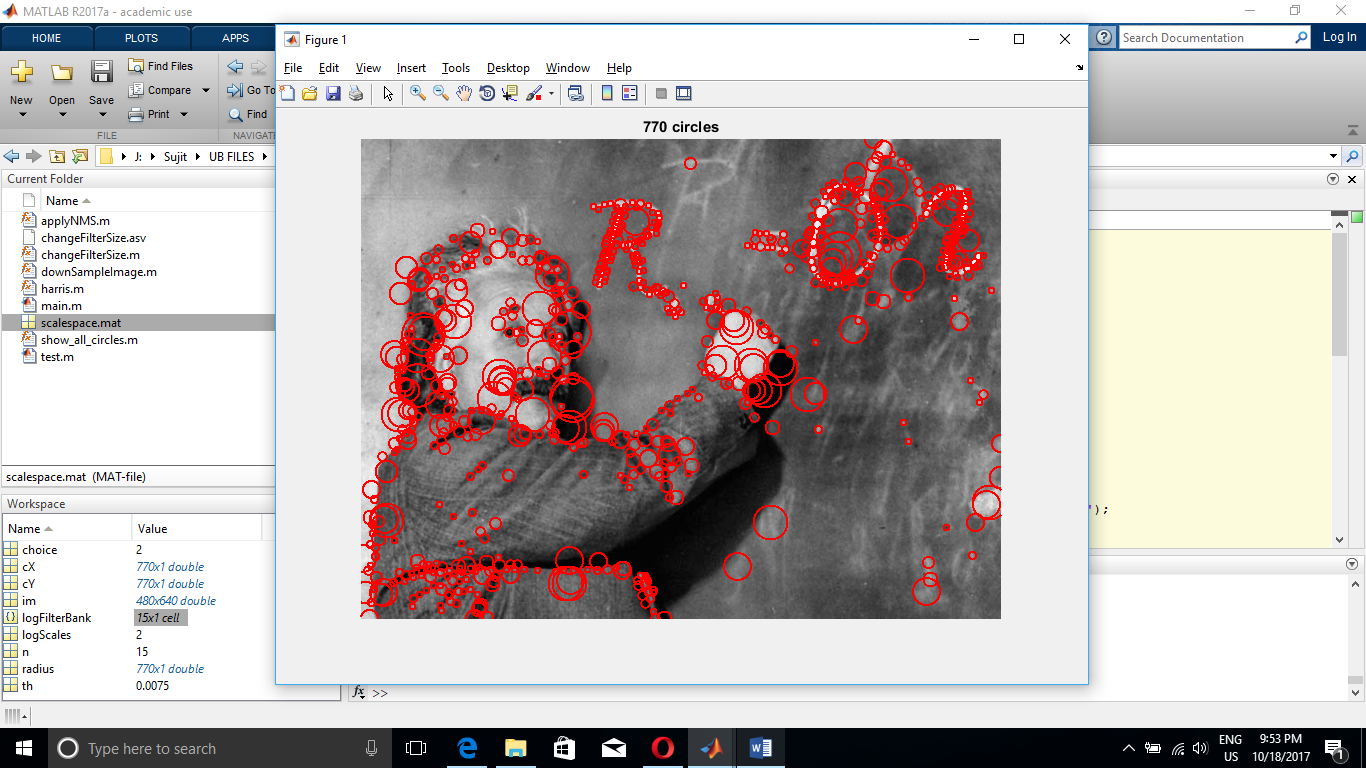
 

Fig 1.3 Einstein (Filter upsize) Fig 1.4 Einstein (Image down sample)

**Running Time:** 12.39s **Running Time:** 5.24s

**Threshold** = 0.01

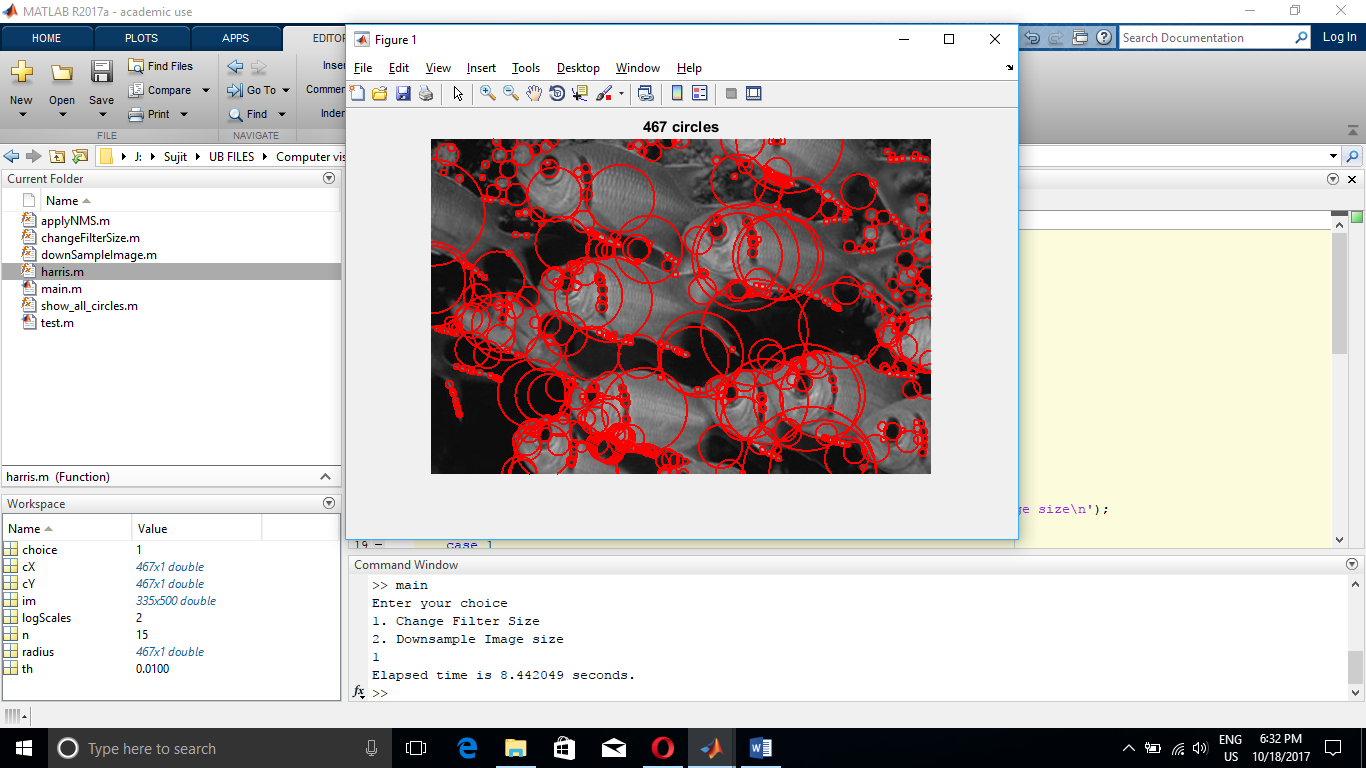
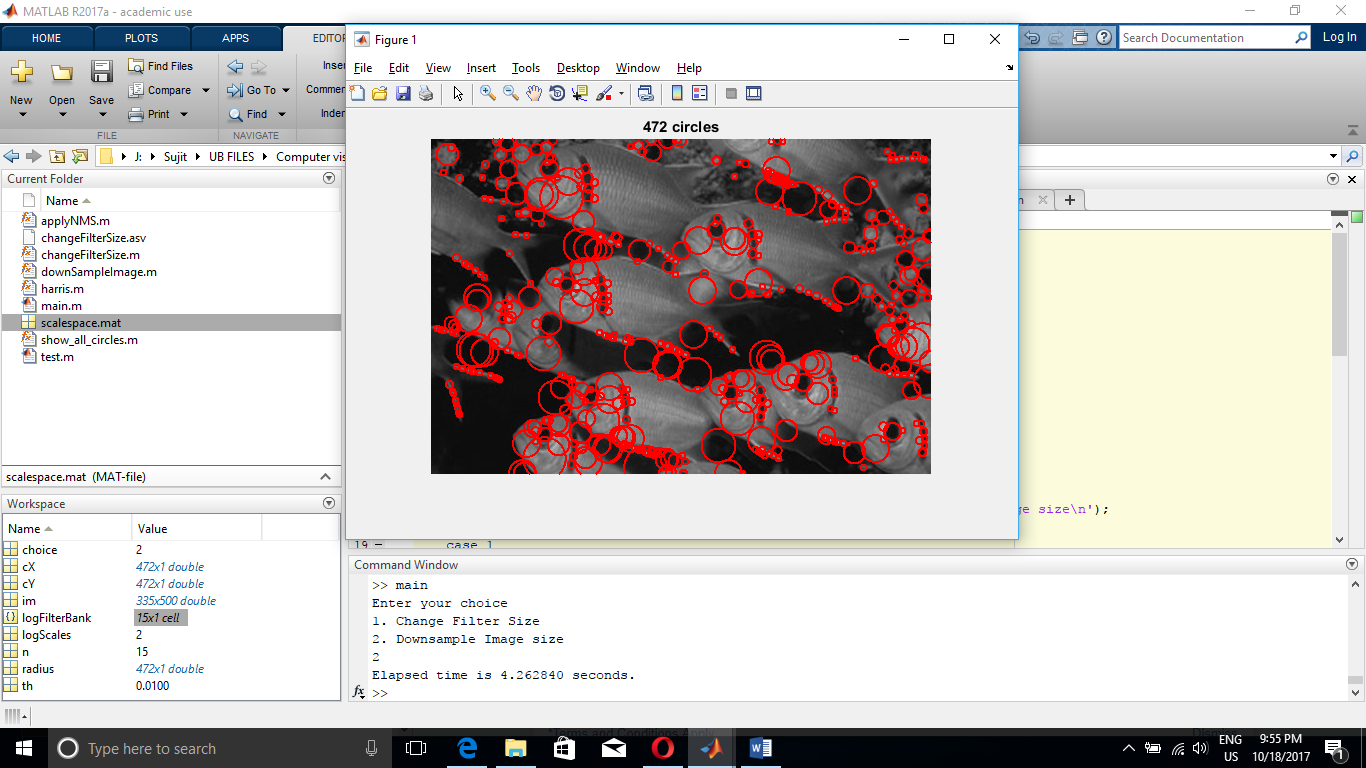
 

Fig 1.5 Fishes (Filter upsize) Fig 1.6 Fishes (Image down sample)

**Running Time:** 8.44s **Running Time:** 4.72s

**Threshold** = 0.0125

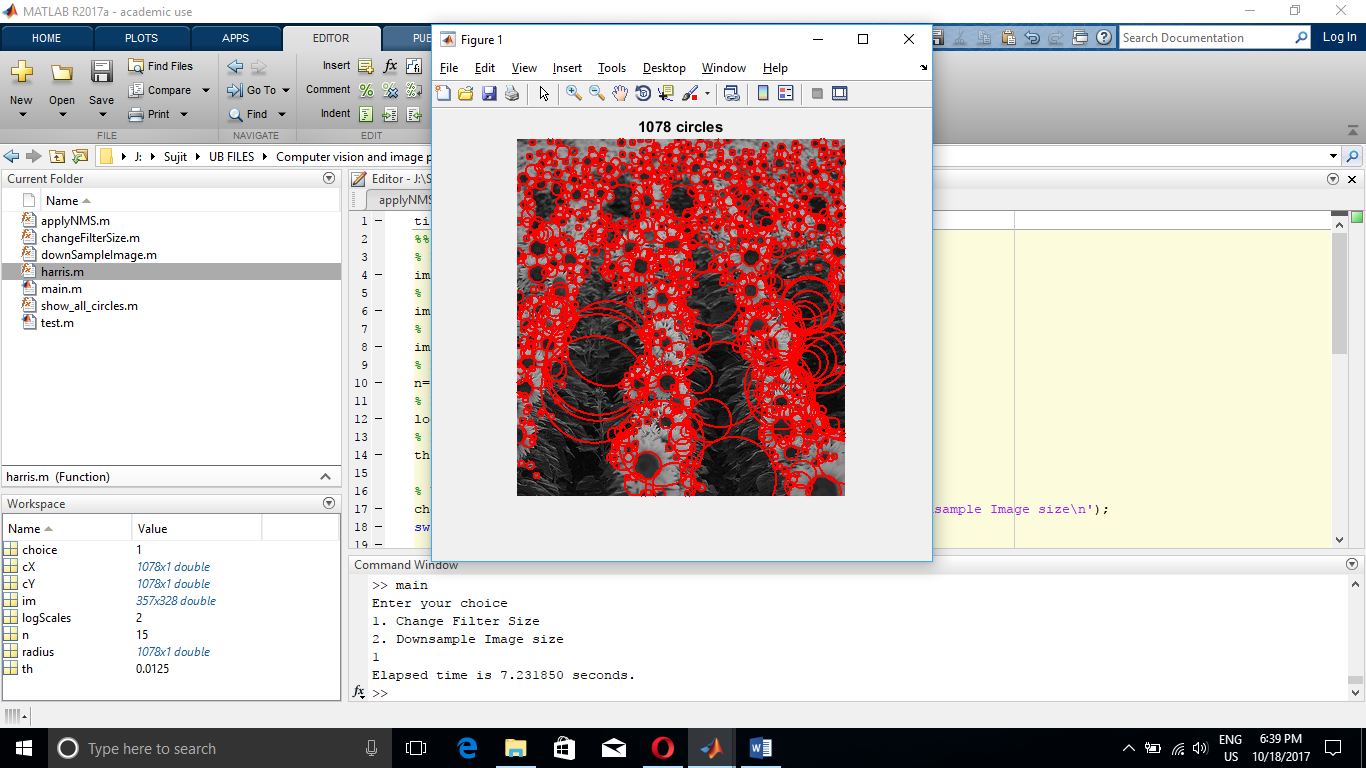
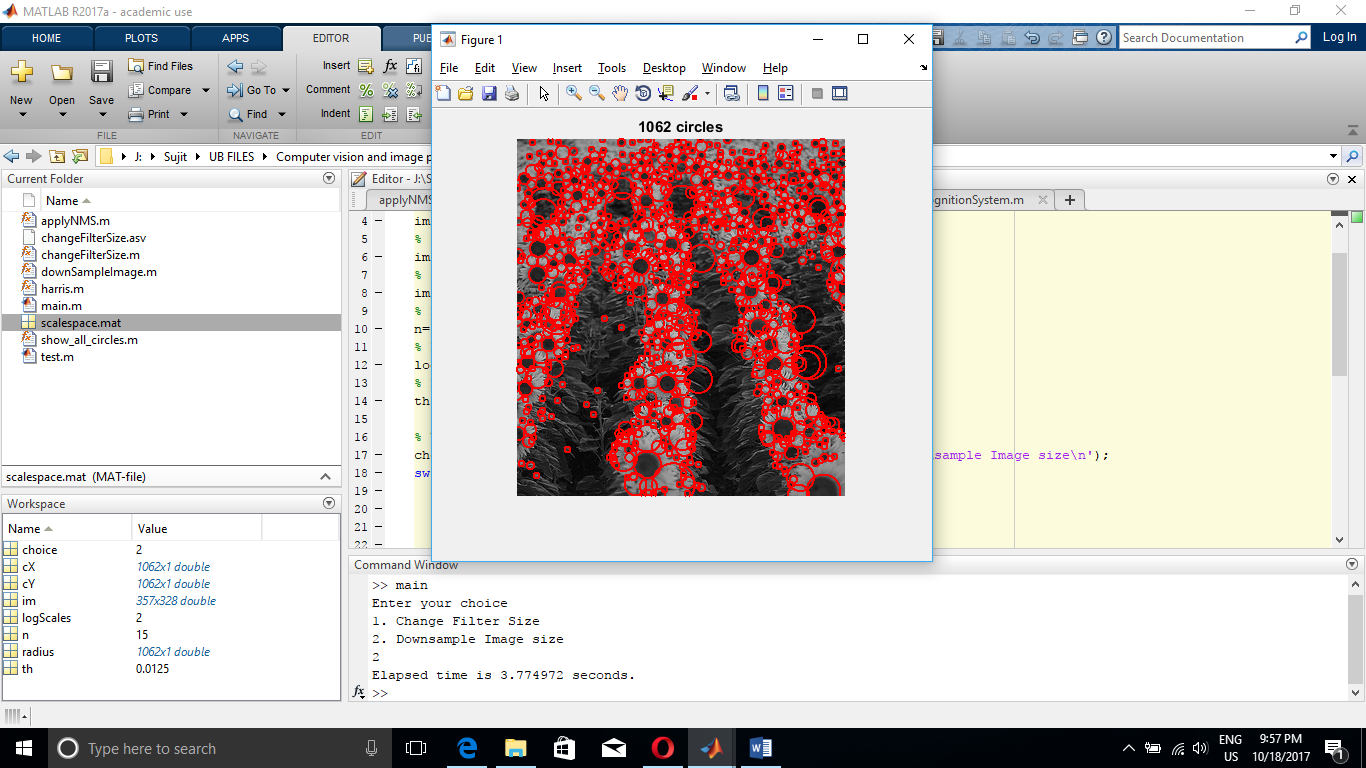
 

Fig 1.7 Sunflower (Filter upsize) Fig 1.8 Sunflower (Image down sample)

**Running Time:** 7.23s **Running Time:** 3.77s

**Threshold** = 0.02



Fig 1.9 Garden

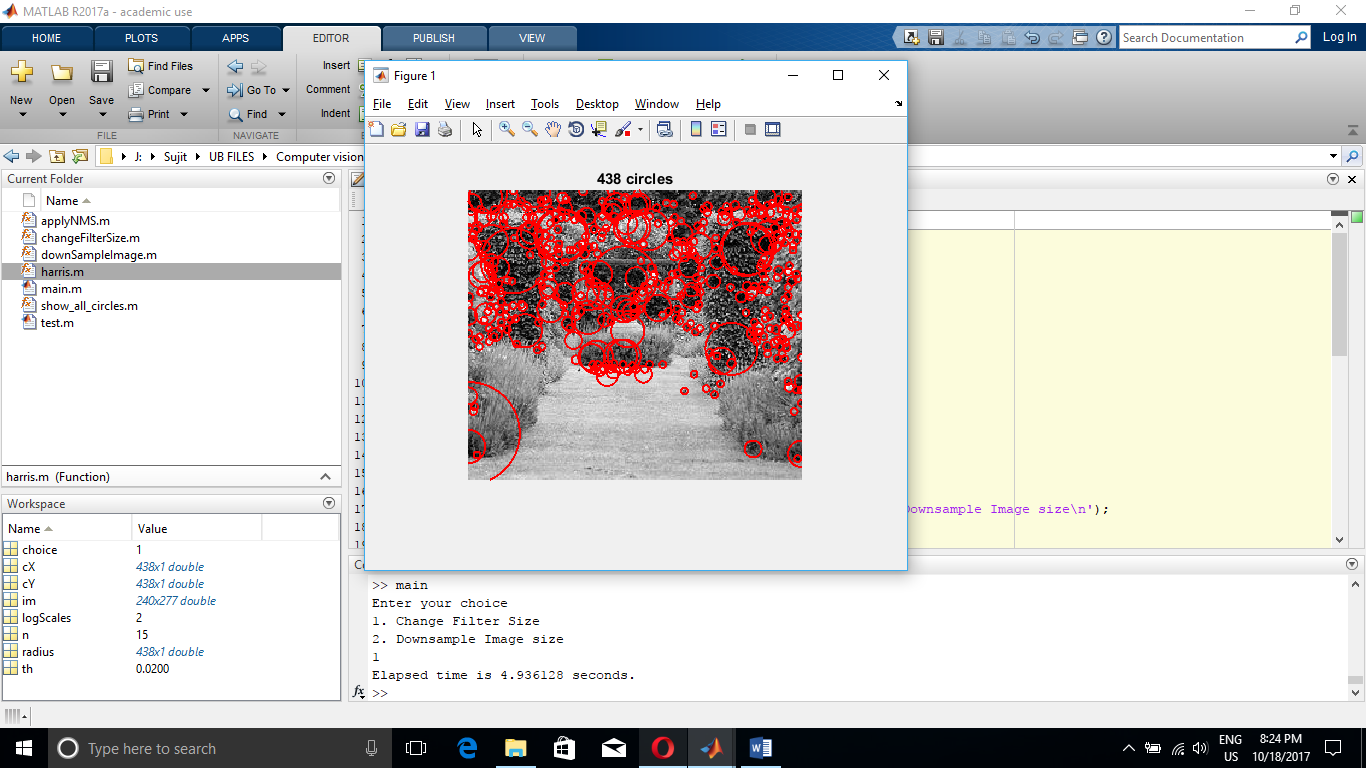
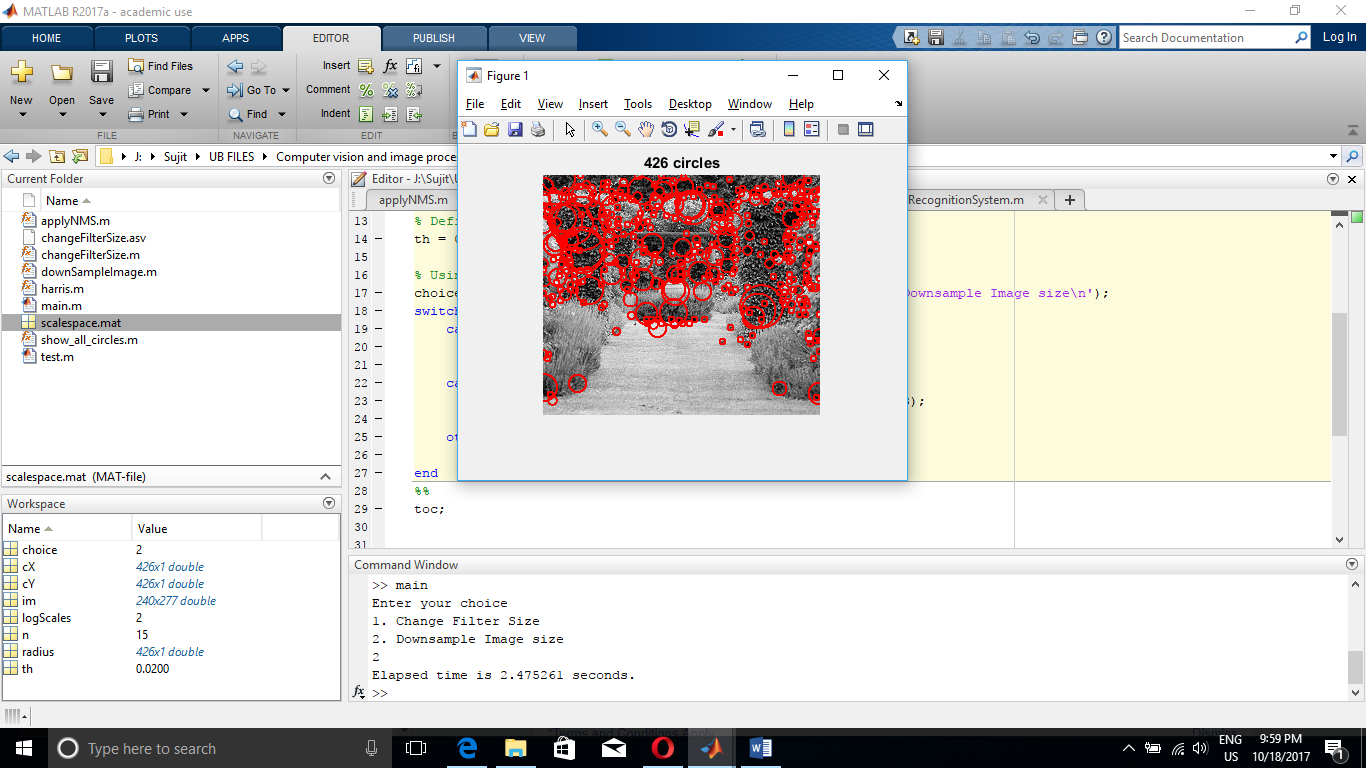
 

Fig 1.10 Garden (Filter upsize) Fig 1.11 Garden (Image down sample)

**Running Time:** 4.93s **Running Time:** 2.47s

**Threshold** = 0.02



Fig 1.12 Classroom

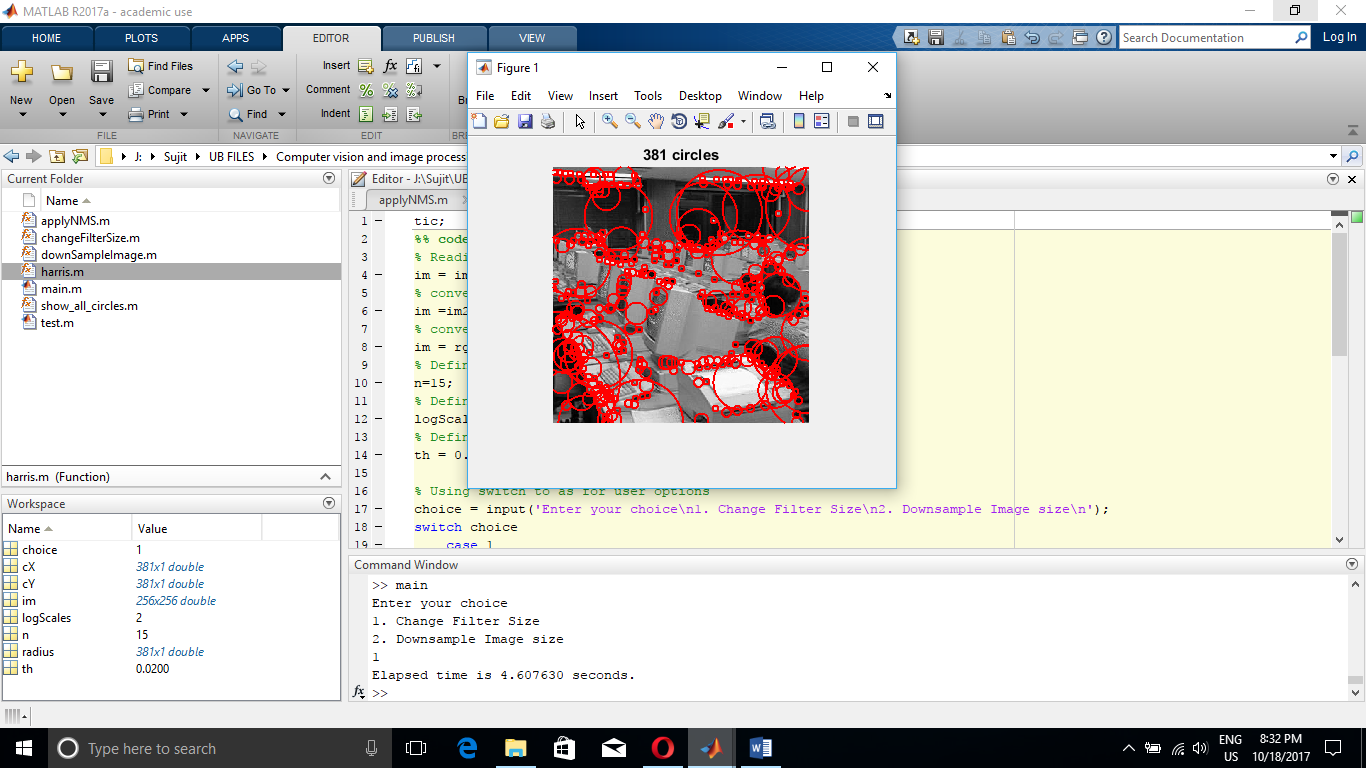
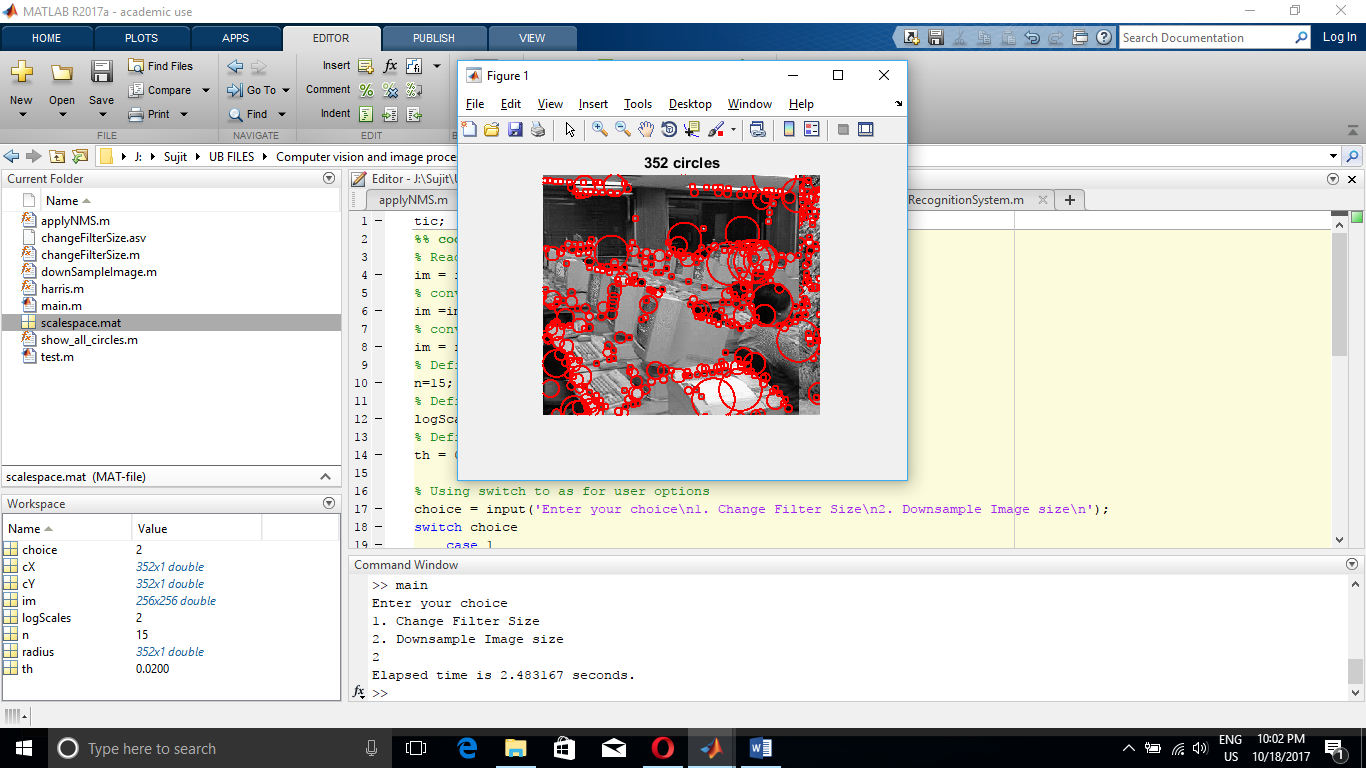
 

Fig 1.13 Garden (Filter upsize) Fig 1.14 Garden (Image down sample)

**Running Time:** 4.60s **Running Time:** 2.48s

**Threshold** = 0.005



Fig 1.15 Mountain

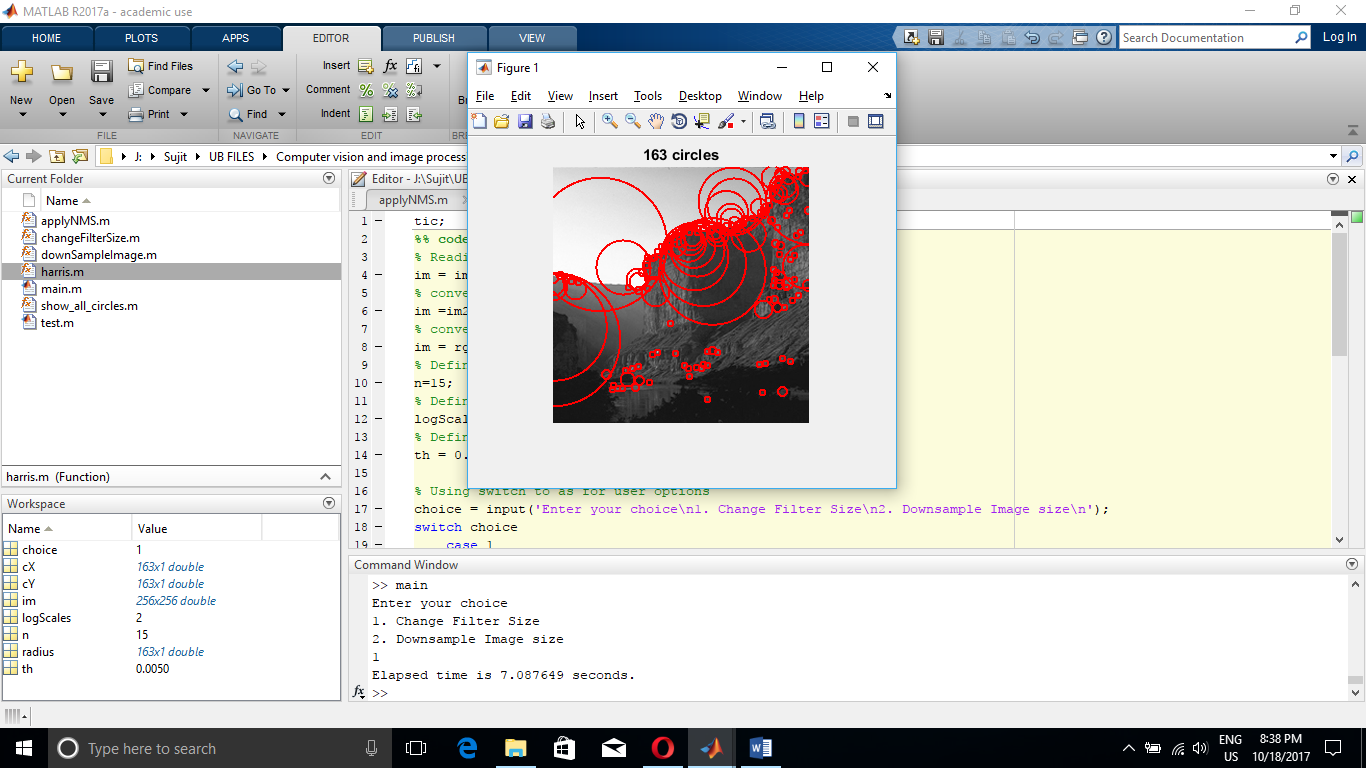
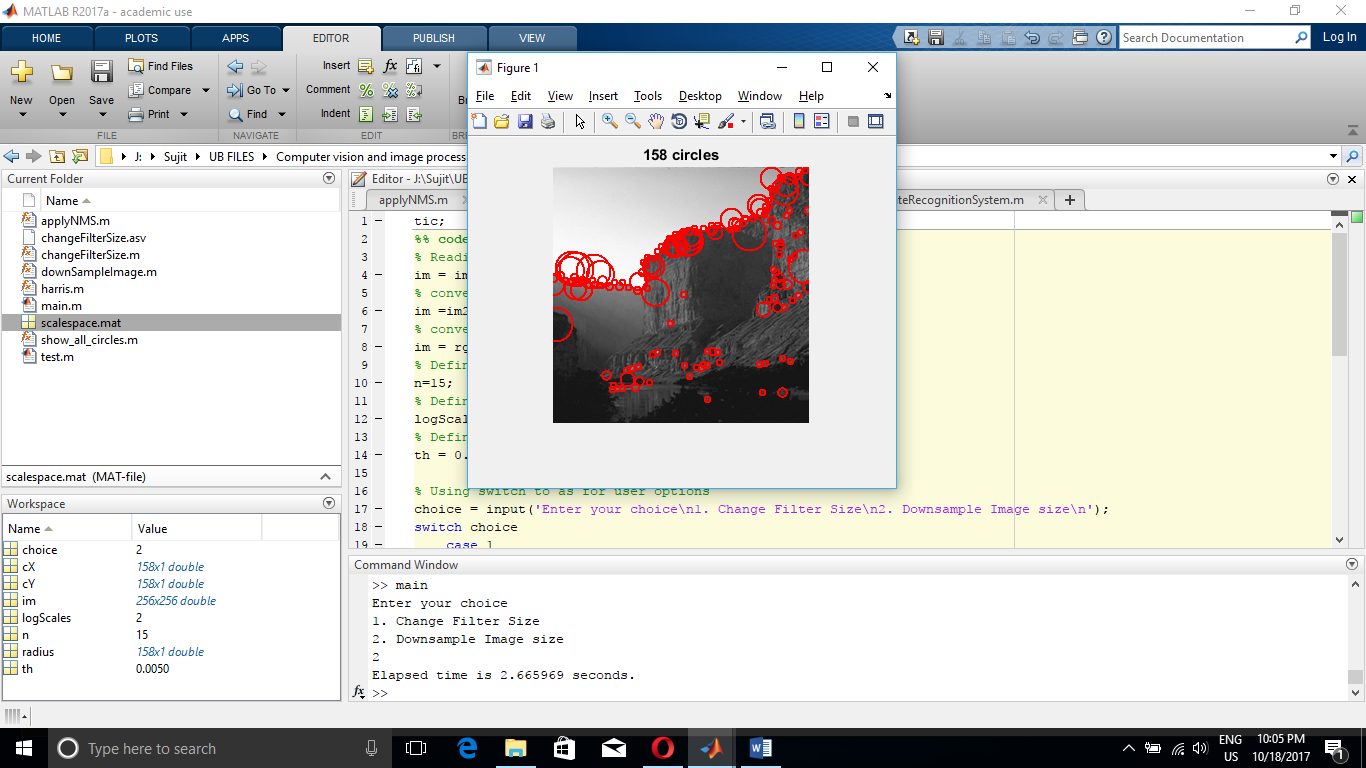
 

Fig 1.16 Mountain (Filter upsize) Fig 1.17 Mountain (Image down sample)

**Running Time:** 7.08s **Running Time:** 2.667s

**Threshold** = 0.055



Fig 1.18 Art Gallery

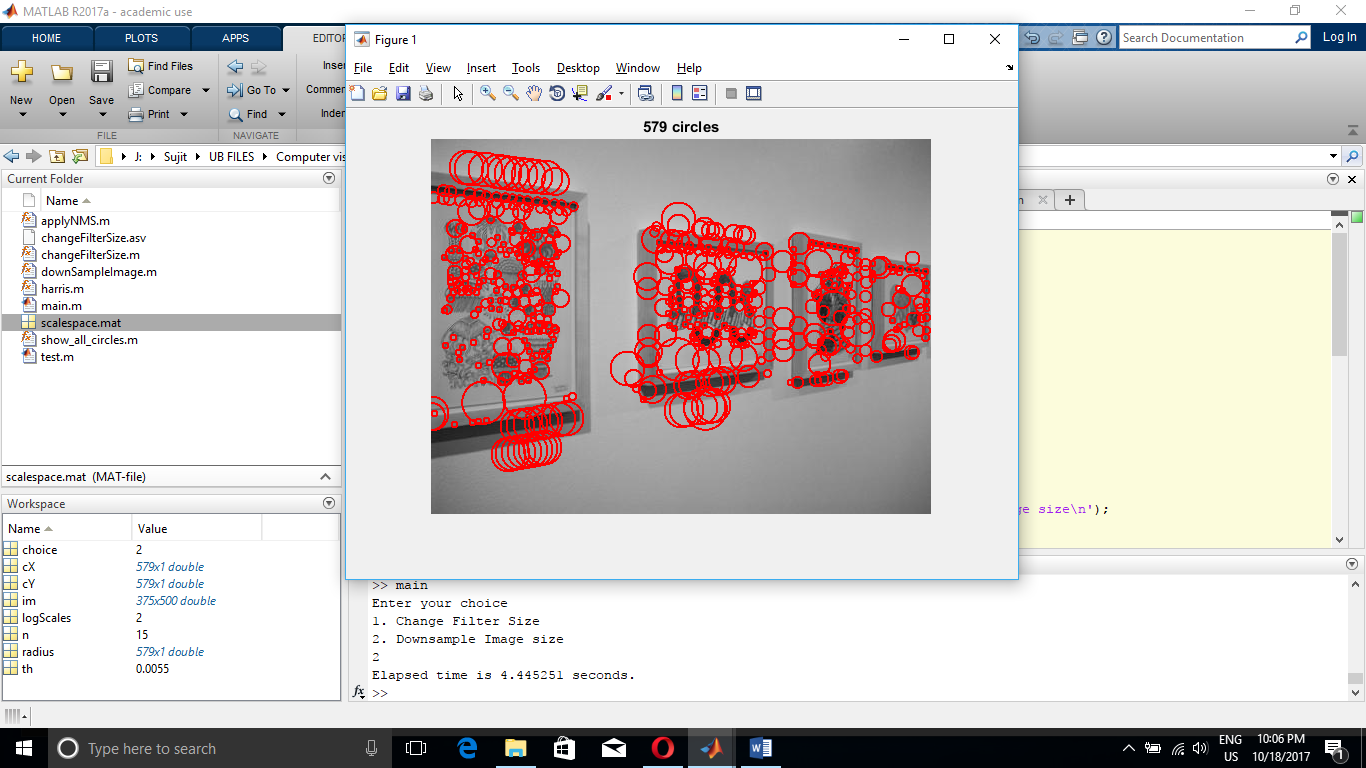
 

Fig 1.19 Art Gallery (Filter upsize) Fig 1.20 Art Gallery (Image down sample)

**Running Time:** 8.34s **Running Time:** 4.445s

From the above images, it is seen that the blob detection is executed quicker when we do image sampling rather than filter upsizing. This is because of the fact that increasing the kernel size by a factor of k will make an exponential increase in the running time. Although the number of convolution operation decreases, it takes a longer time for one convolution. The effect of improper convolution (which takes additional time) can be reduced by using odd filters. Ideally, odd filters give identical outputs.

1. Implementation using optimal choices .

The implementation choices made by me in this project are as follows:

1. **The usage of odd filter**: Ideally odd filters are used as a parameter to fspecial. While applying Laplacian of Gaussian, symmetry of application of filter are very important. Hence, odd filter ideally gives identical outputs on both the halves of the image with the center pixel coming right in between, i.e. (n/2)\*(n/2)th pixel of a n size filter(where n is odd).
2. **Selection of the fixed filter size for downsampling image approach**: In the downsampling the image approach, we can eliminate the larger blobs by appropriately selecting the filter size. The appropriate filter size will give more consized(of lesser radius) but specific blobs making it as an efficient method.
3. **Time complexity**: From the above comparision it can be seen that the running time of the program is nearly halved as the program is not upsizing the filter anymore.

Hence, we can state that the method of downsampling the image is much more efficient than the method of filter resizing.

\*\*\*