CSE 573: Computer Vision and Image Processing

HW 2: Scale-space blob detection

Instructor: Kevin Keane
TAs: Radhakrishna Dasari, Yuhao Du, Niyazi Sorkunlu

Due Date: October 18, 2017

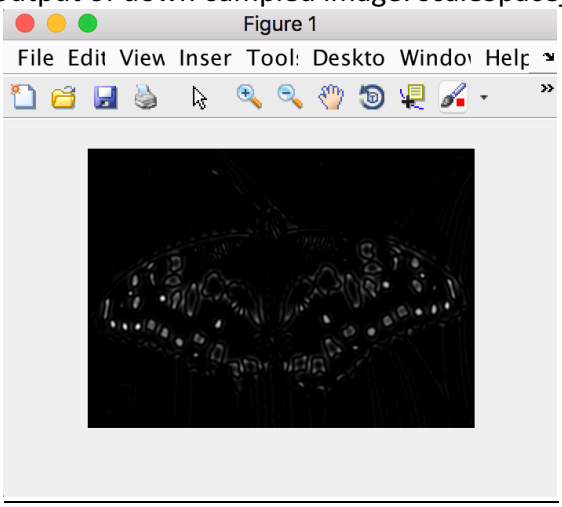
Submitted by,
Muthuvel Palanisamy,
muthuvel@buffalo.edu
Person # - 50246815

Algorithm Implemented:

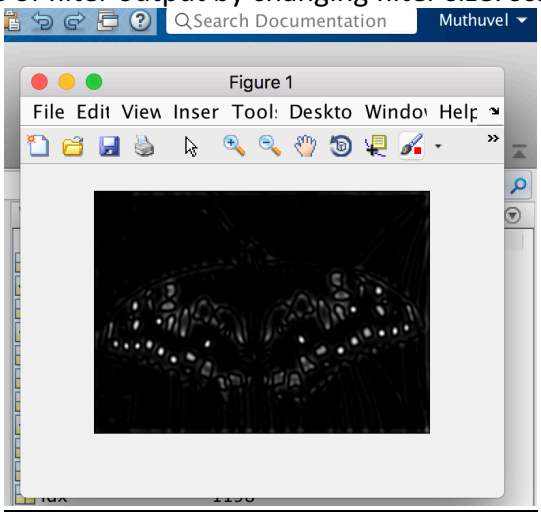
1. Building Scale Space

- Image conversion to grey scale and then to double
- Image is down sampled with a factor k=1.5 for obtaining normalized scale space
- Imresize() is used for interpolation : 'bicubic' interpolation is to get the scale space
- Bicubic interpolation helps because it can produce values even outside the pixel range
- Another normalized scale space is obtained by changing filter size and sigma
- Filter size is calculated = 2*ceil(3*sigma)+1
  Sigma is varied from 2 to 50 for 10 iterations in total
  Filter size is chosen to be an even value

Sample output of down-sampled image: scaleSpace_ds(:,:,2)



Sample Image of filter output by changing filter size: scaleSpace(:,:,2)



sigma=4

- We could see that there is not significant change in the output in two different method

2. Non-Maximum Suppression

- Non-maximum suppression on 2D slices individually and the across the scales
- It took around 0.35 second to for the process to complete
- Ordfilt2 function is used to replicate maxima across pixel values and then on comparison with the original matrix the non-maxima values are suppressed
- The same procedure is repeated for 3D suppression. A pixel is compared with 26 of its neighbours to find the maxima
- Max() function is used to find the maximum values

3. Finding co-ordinates

- Co-ordinates of the non-maxima suppressed scale space is found using max() function
- The appropriate radii of each blob is give by the relation

  Radii= sigma/power(2,1.1)
- Threshold is chosen to as below for different images

  → sunflower = 0.01

  → butterfly = 0.01

  → einstein = 0.005

  → fishes = 0.0075

4. Showing circles using show_all_circles function

Runtime values

(parameter = constant)

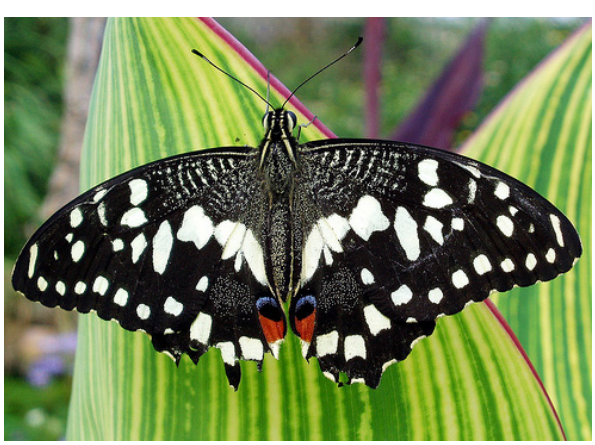
Image	Downsampling (Efficient)	Changing filter size (Inefficient)
Butterfly	0.102 seconds	2.800 seconds
Fishes	0.104 seconds	2.657 seconds
Einstein	0.210 seconds	4.819 seconds
Sunflowers	0.104 seconds	2.000 seconds
Bluerose	0.396 seconds	12.01 seconds
Rocks	1.548 seconds	53.933 seconds
Smile	0.462 seconds	15.844 seconds
Eyes	1.788 seconds	62.900 seconds

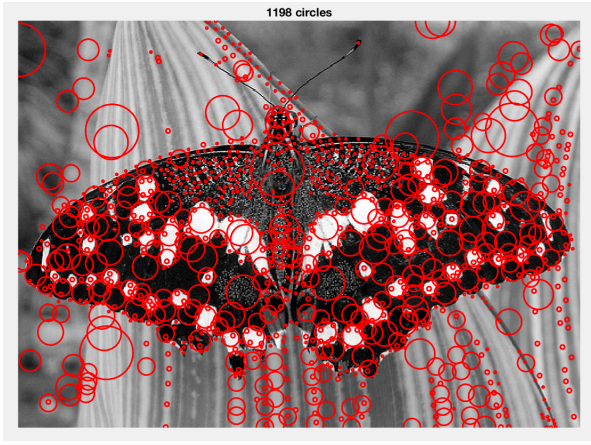
Runtime comparison

Downsampling	Changing filter size
Remains almost the same for all images	Varies greatly for different images
Not greatly affect by high sigma values	High sigma values take longer time, some as long as 30 to 40 seconds
Handles unwanted blobs on the edges efficiently using bicubic interpolation	Not much efficient in handling blobs around the edges
Gives around 1198 circles for butterfly image	Gives around 1100 blobs for butterfly image for similar parameters

Output for different images:

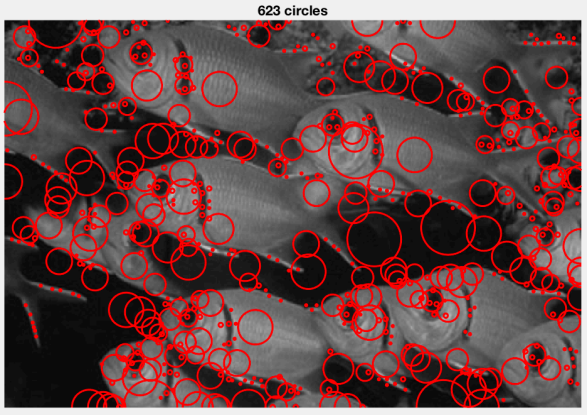
1. Butterfly





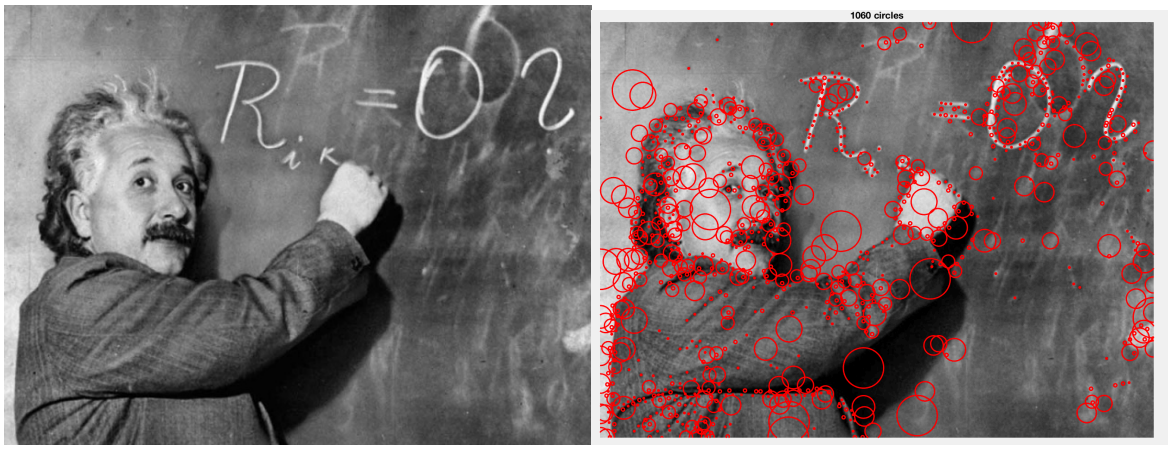
Maximum sigma value = 50

2. Fishes



Maximum sigma value = 50

3.Einstein



Maximum sigma value = 50

4.Sunflowers

Maximum sigma value = 50

5.Bluerose

Maximum sigma value = 50

6.Rocks

Maximum sigma value = 128

7. Smile

Maximum sigma value = 128

8. Eyes

Maximum sigma value = 256

Take away

- Threshold limits the number of blobs detected.
- The bigger the size of the blob you want to detect, bigger should be the maximum sigma value – also depends on the resolution of the image
- Appropriate filter size should be chosen when changing the sigma value
- **Sigma is chosen from 2 to 50 for many images** because the the maximum blob size to be detected is covered in this region and any values above that will be fully black for images with comparatively lesser resolution
- **Filter size is choses as 2*ceil(sigma*3)+1.** This size is suitable to for filtering for different sigma values and if the size of filter is even, the images is not proper
- **Ordfilt2** is fast and efficient for non-maximum suppression. The run time is mostly similiat to **colfilt** function

Additional features tried

- Tried creating a 3D ordfilt function. But, I was not able to verify if the logic is correct – function name : **ordfilt3d –** included in code
- Tried non maximum suppression across scales for a pixel across all the scales instead of just 26 neighbours. The later seemed comparatively efficient
- Found co-oridinates of maxima without using conventional for loops which is faster and took only less than 0.005 seconds to complete (Check %% find co-ordinates of the maxima section in the main code.
- 'bicubic' parameter is chosen as interpolation method since it can produce values even outside pixel are thus reducing unwanted blobs on the end

---