ECE 558 Project

Digital Imaging Systems

Rahul Mishra

[rmishra4 / 200267922]

Introduction

In this project we are trying to find out the blobs or circular patches present in the images. The patches are of various sizes and we need to detect them all. We use Laplacian of Gaussian filter to localize these patches. There are two ways in which this has been done. The steps used, and the results obtained are described below.

Fixed Image Size, Varying Filter Size

Steps:

1. Read the input image. Convert it to grayscale using cv2.imread('image-path', 0). Then normalize it.

2. LoG filter. The function used for this is:

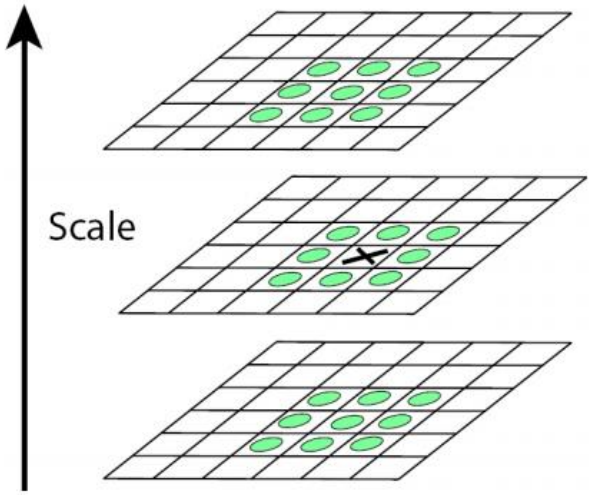
$$\text{LoG}(x, y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\text{Normalized LoG} = \sigma^2 LoG(x, y)$$

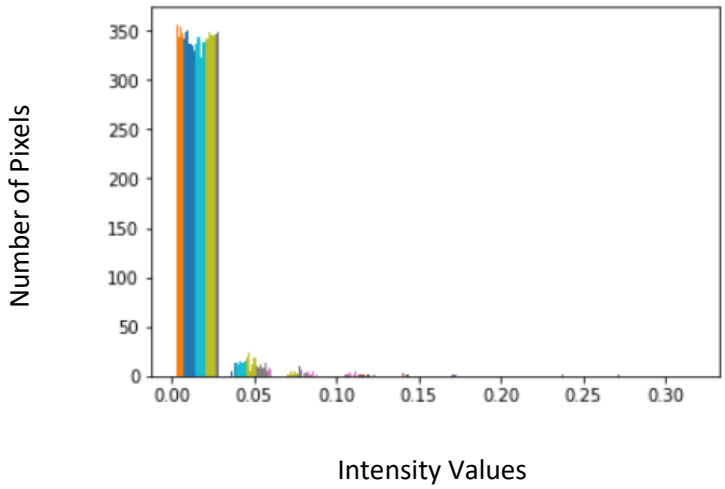
Function Name: makeLOGfilter (size, sigma)

We will create filters of different sigma which vary like $\sigma, k\sigma, k^2\sigma, k^3\sigma, k^4\sigma$ … , where we have chosen
k = 1.24, $\sigma = \frac{1}{\sqrt{2}}$
Number of layers/scales = 15
Size of filter = 6σ which is rounded off to the upper odd number. The reason behind this is that 6σ captures more than 99% of the variation of LoG, leaving out the values which are essentially not going to affect the results on account of being close to zero.

3. Now we will convolve the image separately with each filter and square the response of filtering. This will result in non-negative values for each pixel.
   Function Name: conv (f1, f2)
4. Non-Maxima Suppression: After squaring the convolution result, we are only left with maxima in the image.



In the figure show, we see that there are 26 neighbors to the center pixel- 8 on same image, 9 on the above image and 9 on the bottom image.
Similarly, the image which is at the top and the bottom of the scale don't have 26 neighbors. Rather, they have only 17 neighbor pixels- 8 on the same image, and 9 on bottom or upper image.
The pixel intensity value distribution after convolution and squaring for the first image with minima sigma is:



We will suppress, i.e., make zero, the pixel values which are not maxima among its neighbors and below some threshold. For selecting threshold, we observe the distribution of intensity values. We don't want every pixel to be considered blob. We will only pick the higher values. On examining the histogram, we can choose 0.1 as the threshold.

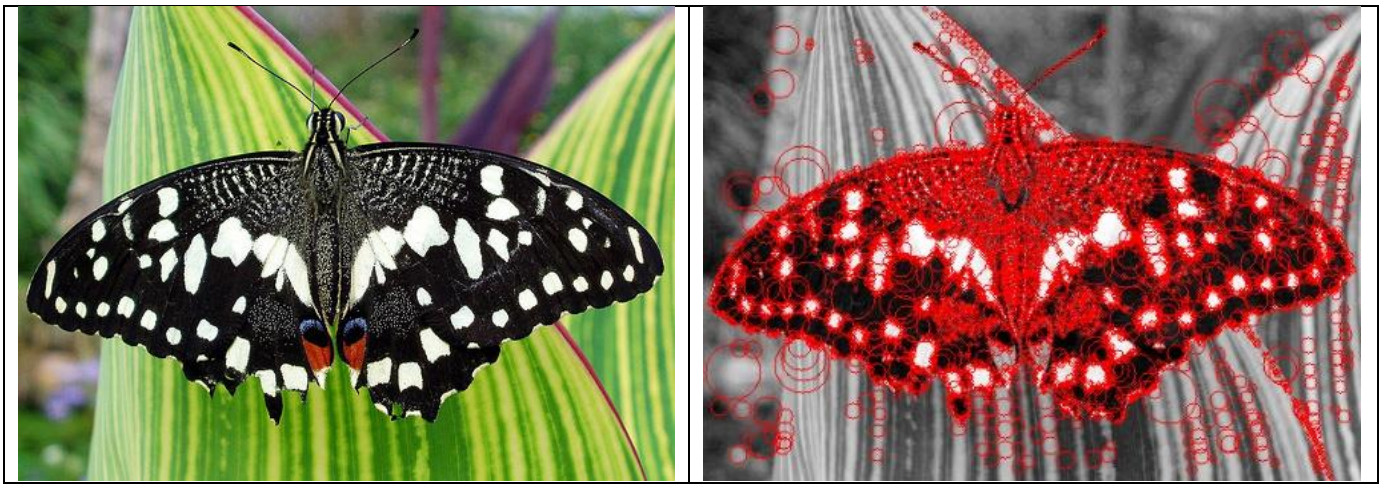
5. Finally, we draw the circle around all the non-zero-pixel values. Radius of the circle corresponding to each scale is derived from the sigma of the filter which detected that blob. $Radius = \sqrt{2}\sigma$

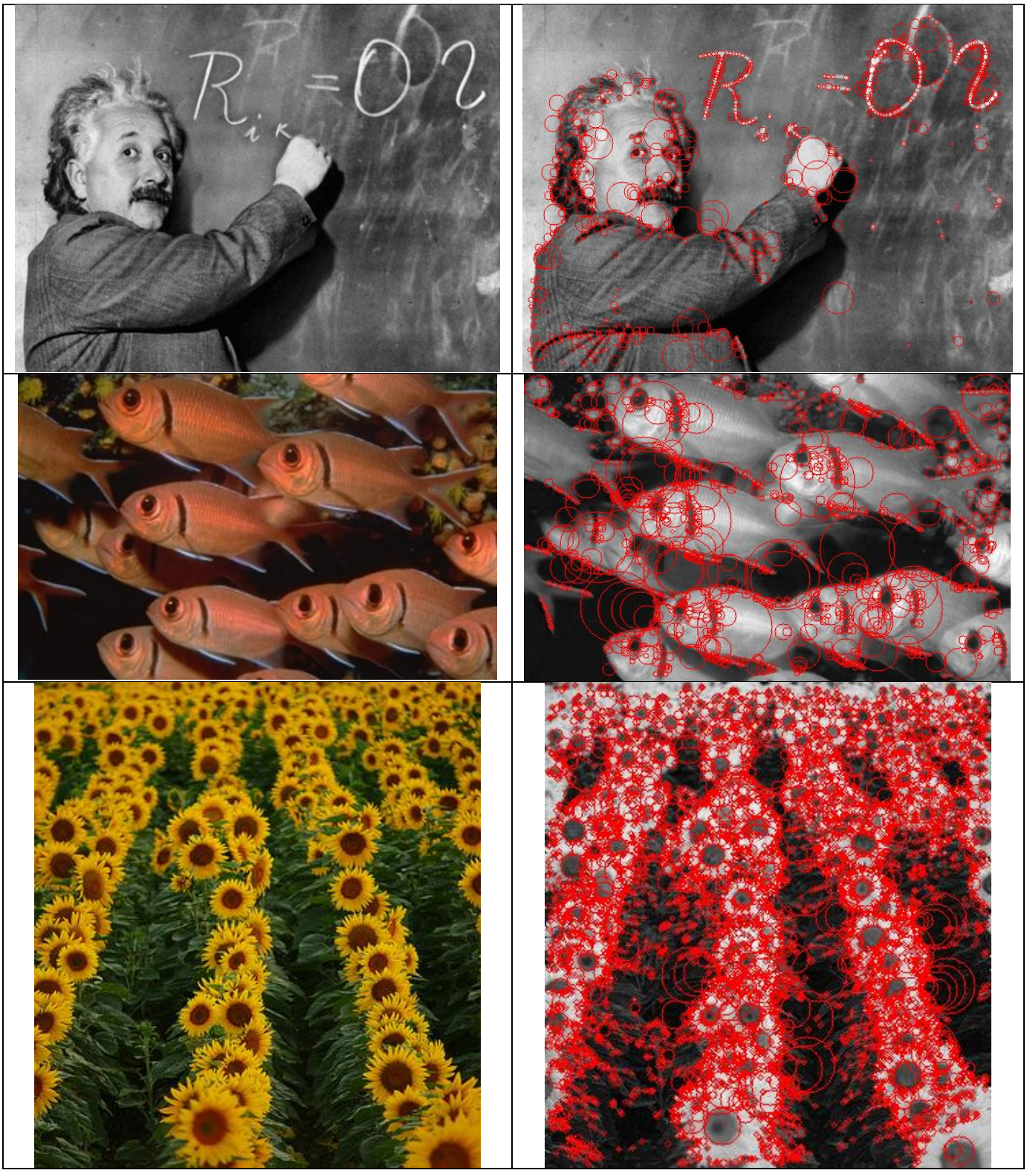
Varying Image Size, Fixed Filter Size

1. Read the input image. Convert it to grayscale using cv2.imread('image-path', 0). Then normalize it.
2. Make LoG filter of fixed σ and fixed size.
3. Reduce the image size by a factor of k in each iteration and make a scale space by convolving each image with the above filter. Take the square of the response.
4. Bring back the image to its original size and perform non-maxima suppression.
5. Draw the circles around the non-zero pixels.

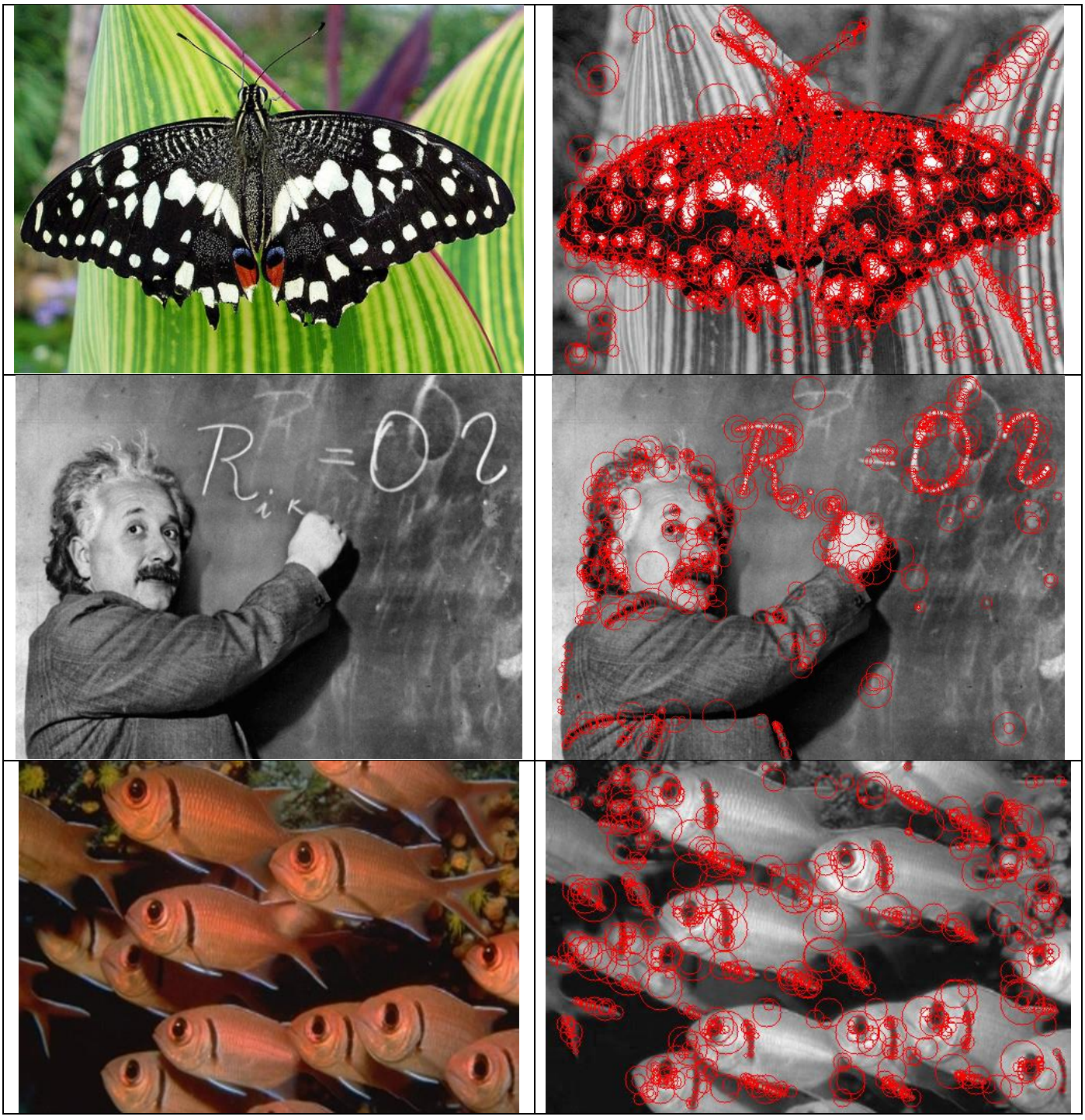
Results:

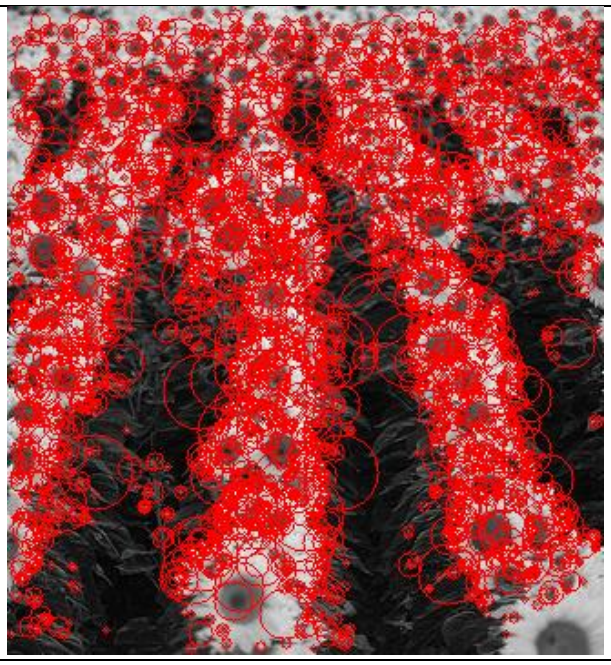
Varying Image Size, Fixed Filter Size




$R_{ik} = 0$
$R_{ik} = 0$

Varying Image Size, Fixed Filter Size

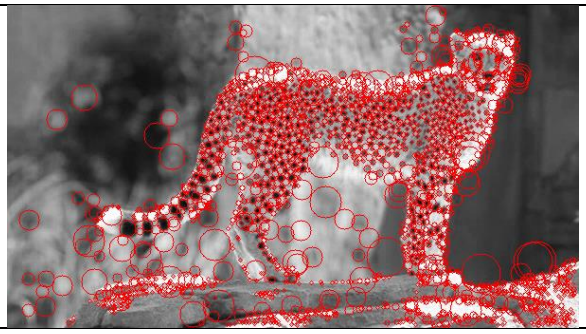


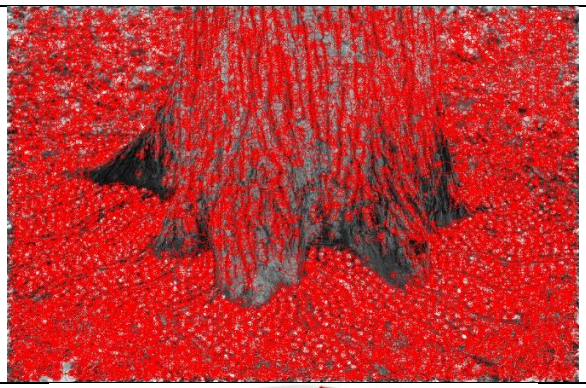


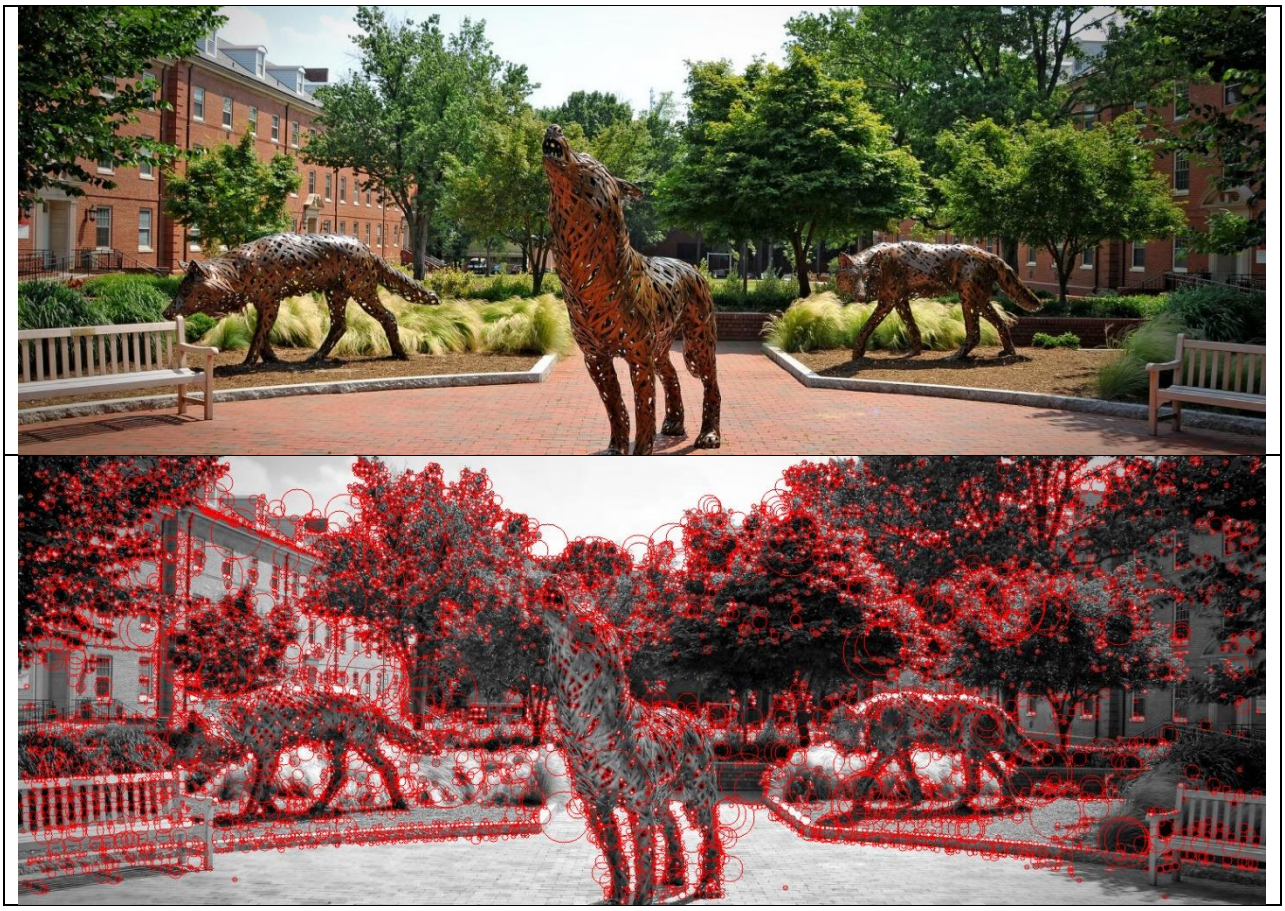
Runtime Comparison:

	Time (Fixed Image Size)	Time (Varying Image Size)
Butterfly	21.11s	5.83s
Einstein	32.54s	5.96s
Fishes	17.71s	3.15s
Sunflowers	13.53s	3.22s

More Images Output:







Conclusion:

We see that with little tuning we can localize the blobs present in the images. The method where we are reducing the size of the image is better in runtime. But overall, their result are almost similar.