Project: Laplacian Blob Detector

-------- ECE 558 Fall 2018 -------

Project Report

by

Deepayan Bardhan
(dbardha)
(200266399)

North Carolina State University

Department of Electrical and Computer Engineering

Overview Of The Project

This project has been intended for implementation of blob detection using Laplacian scales. The objective of the project was to detect the patches/blobs in the given images and encircle them. As we know, Laplacian filter can be used to detect step-like pattern in scale-space and the blob location can be detected from those scale spaces using non-maximum suppression (NMS) method.

Algorithm For Blob Detection

There can be various ways of approaching the problem. The method that has been used in this project is noted below.

Step 1: Generate the Laplacian of Gaussian Filter

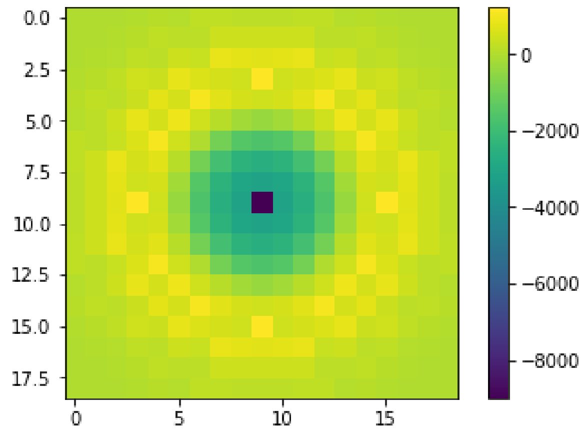
Provided the *sigma(σ)* and the *kernel size* the Laplacian of Gaussian Filter is being generated using the formula $LoG(x,y) = (x^2+y^2-2\sigma^2)e^{-(x^2+y^2)/2\sigma^2}$.

Using this we get a normalized Laplacian of Gaussian Filter which we can apply on the given images. The filter values are very small initially and hence would lead to small convolution values. Thus, they need to be multiplied/scaled by a certain factor to get proper convolution results.

Step 2: Build a Laplacian Scale Space

The filter that we generated in the previous step needs to be applied to get a Laplacian image at the current scale (say 1). To go to the next scale, we chose a particular value '*k*' with which we scale the *σ* of the preceding scale space and thereby generate a new filter which upon convolving with the given image we get the next Laplacian Scale Space image.

Similarly, using this same method we create a column of convolved images with varying *σ*. This is known as the Laplacian Scale Space.

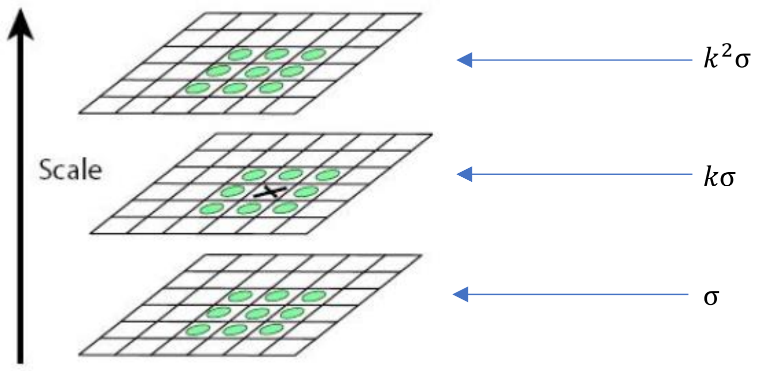


This is how the Laplacian of Gaussian (LoG) kernel looks like. The center has a very low value and it rises sharply to small positive values and then slopes to 0.

Step 3: Perform Non-Maximum Suppression (NMS)

The scale space that is generated is subjected to non-maximum suppression. This method is used to detect the maxima-s (for squared Laplacian responses) and extrema-s for simple Laplacian response. In this project squared Laplacian responses have been recorded so the maxima-s are the point of interest.

In 3D NMS a particular pixel value is compared with all the surrounding 26-pixel (top – 9, bottom – 9 & same level – 8) values. If it is the absolute maximum then that is our desired point. We do this process for all the layers of images.



The green pixels are compared with the X marked pixel

Step 4: Displaying the resulting blobs at their characteristic scales

The points which we get by NMS are picked up from individual layers and a circle of radius $\sqrt{2}$ times the σ of the layer is drawn at that co-ordinate. After doing this for all the layers we get an image with all the blobs detected.

IMPLEMENTATION

Chosen parameters:

a) Scaling factor of LoG filter = 500
b) σ = 2 (Starting layer)
c) k = 1.25
d) Number of layers/levels = 10
e) Threshold = 0.1

The value of the parameters were chosen on trial basis. The points considered while choosing the parameters values are:

i) The number of blobs being detected – Having a very low σ to start with and increasing the number of layers will lead to excess number of blobs detecting the smallest of patches in the images which is redundant on a large scale

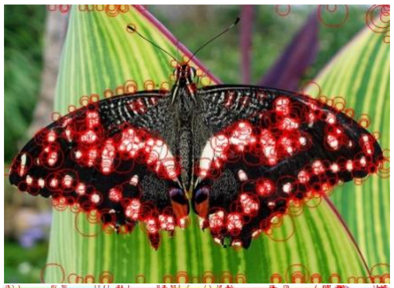
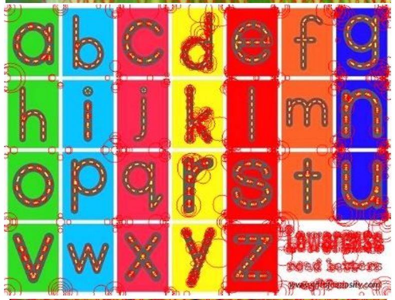
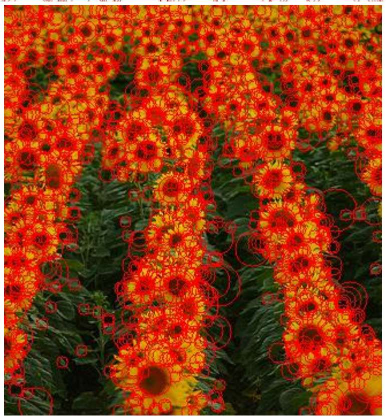
ii) The size of blobs – Having a large σ to start with would lead to less computation but would cause in missing of smaller blobs which might be critical in some images with small patches.

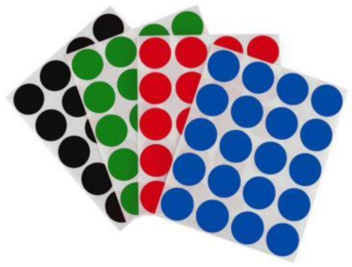
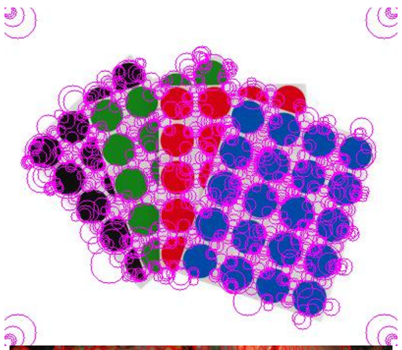
iii) Location of the blob – If too many blobs are being detected for a particular location that is because of the close spacing of the sigma's or a very low value of k. Also choosing a very high value of k would lead to missing out of many major patches.

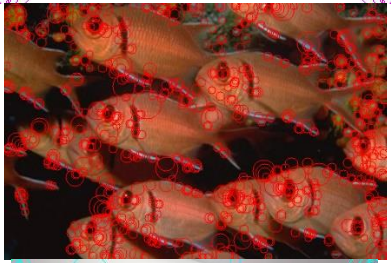
iv) Computation time – Increasing the number of layers or choosing a very large value of sigma, which leads to having a larger kernel size for LoG filter, consumes a lot of time in convolution.

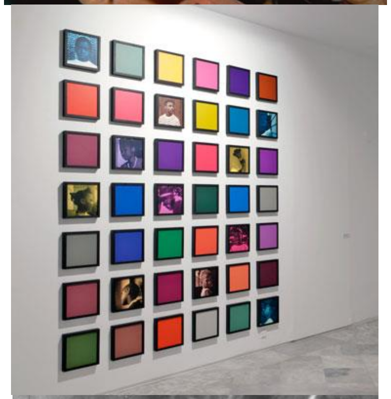
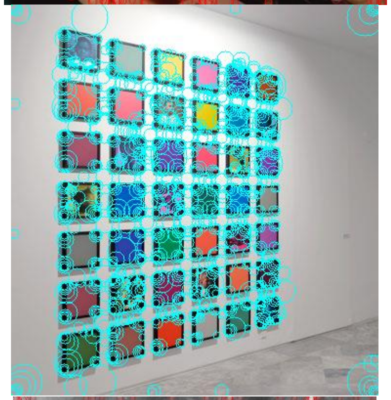
Keeping the above points in mind the parameters have been chosen optimally such that the output is not much affected and the computation time is in a reasonable range.

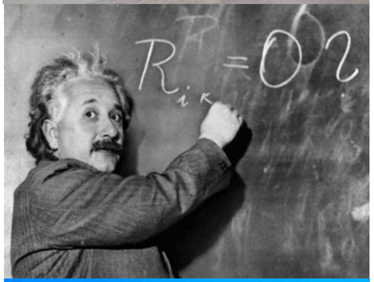
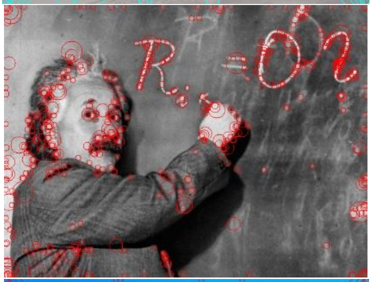
OUTPUT

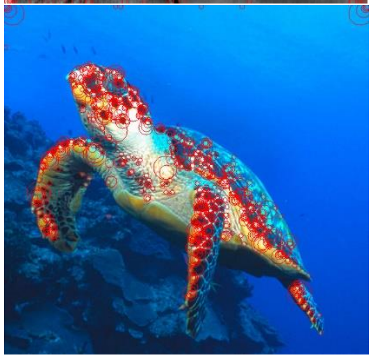
Original Image	*Image after Blob detection*	*Computation time*
		76 seconds
Lowercase road letters www.giftofcuriosity.com		94 seconds
		53 seconds



115 seconds


134 seconds



96 seconds



110 seconds


83 seconds

Detailed images are provided in the folder.

Scopes Of Improvement

A faster approach to this would be to implement this using octave where the computational time for convolution would reduce by a significant amount since in that case instead of increasing the σ value by k times (thereby increasing kernel size), the image is down-sampled by k times and then perform the convolution and then up-scale it to the same size thereby reducing the convolutional complexity.

Explanation Of The Code Provided

The convolution(img, kernel) function takes 2 parameter – the image and the kernel and returns the 'same' size convolution matrix such that the values of the pixels are scaled in a range of 0 to 255 (i.e. the smallest value after convolution is mapped to 0 and the highest to 255) so that it can be viewed properly as an image.

The genlog(sigma, size) method takes 2 parameters – the sigma of the Laplacian of Gaussian matrix and the kernel size of the matrix. It computes the matrix using the previous mentioned formula and then scales it and returns the same.

Inside the main code first a clock is started to note the running time of the code. 2 images – the image in gray scale on which the convolutions are performed and also the original colored one over which the circles are drawn, are kept in 2 variables. The value of k and σ are put and then the LoG kernels are generated and convolutions are performed and stored in a list consecutively.

After the scale space is generated NMS is performed over those matrices and then at the corresponding maximas, circles are drawn with the before mentioned formula on the original colored image.