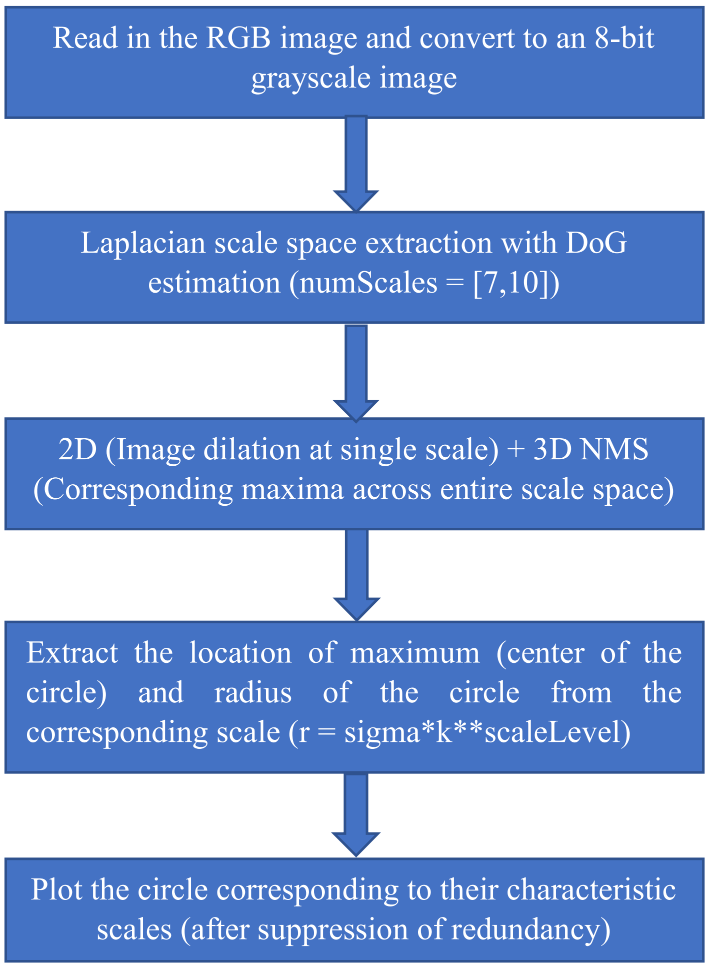
Introduction

Blob detection is an algorithm in computer vision to detect anomalies or change in properties such as brightness, color compared to surrounding regions. The procedure for detecting blobs can be split into three basic steps, extraction of Laplacian scale space, non-maximal suppression and displaying circles based on their scales.

The extraction of the Laplacian scale space done by repeated convolutions. These convolutions can either be done by keeping the image constant and increasing the LoG (Laplacian of Gaussian) kernel size, or by keeping the kernel constant and successively downscaling the image. An optimal method of the scale space computation is the estimation of LoG operation using DoG (Difference of Gaussian), which can be extremely corelated, if the kernel sigma value is appropriately chosen.

The next step is the non-maximal suppression which is employed to suppress many overlapping circles. It is done across all the scales so that there exists only one maximum corresponding to a particular pixel over the entire scale space. The final step is to plot the characteristic circle to their corresponding scales and displaying the result on the original image.

Algorithm implementation


Read in the RGB image and convert to an 8-bit grayscale image
Laplacian scale space extraction with DoG estimation (numScales = [7,10])
2D (Image dilation at single scale) + 3D NMS (Corresponding maxima across entire scale space)
Extract the location of maximum (center of the circle) and radius of the circle from the corresponding scale (r = sigma*k**scaleLevel)
Plot the circle corresponding to their characteristic scales (after suppression of redundancy)

a) **Scale space computation**

In the algorithm, the first step of the scale space computation is the repeated convolution of image with increased kernel sizes (LoG kernel). The kernel size is again determined by the choice of sigma and scale multiplier (k) to build the scale space. For optimization purposes, we are estimating the LoG using DoG. We basically generate a Gaussian pyramid for the image, but only this time, we don't downscale the image, rather we successively increase the Gaussian kernel size. The Laplacian scale space elements are extracted by the subtraction of successive images in the Gaussian pyramid. I have implemented the convolution operation using FFT (multiplication in frequency domain), which makes repeated computations much faster.

The final scale space array is a list of arrays where each member array of this list corresponds to a squared Laplacian scale space response. This then gets the scale space ready for non-maximal suppression and scale maxima computation.

b) **Non-maximal Suppression (NMS)**

This is done in two steps, namely, NMS-2D and NMS-3D. The 2D NMS is we take the neighboring pixels [x-1:x+1, y-1:y+1] of the pixel of interest (in the same scale) and extract the maximum of the 9 values. This is similar to image dilation and hence, for optimization, we can directly dilate the image once to retain only the maxima at the pixels. We then perform a 3D NMS across all the scales of the scale space. For the 3D NMS, we take into consideration the corresponding pixels from all the scales and extract the maximum. Only the pixels that have the maximum value in the region will retain the original value, and the others will be reduced to 0 to ensure that there is no other feature extracted from that region. Once we have maxima values, we can extract the pixel locations and the layer at which the maximum is coming from. The radius of the circle to be plotted around the circle at center (x, y) is chosen based on the sigma value for that scale. The circle parameters (x, y, radius) are then appended to a list which can be plotted on the figure once all the maxima have been characterized.

Another important step in the 2D NMS is the image thresholding. This is a very crucial hyperparameter which has to tuned specifically to an image. This parameter determines the number of maxima in the 2D NMS actually deserve to go through for comparison in the 3D NMS. This is done to ensure that only the most prominent features in a given region get captured in the final blob output.

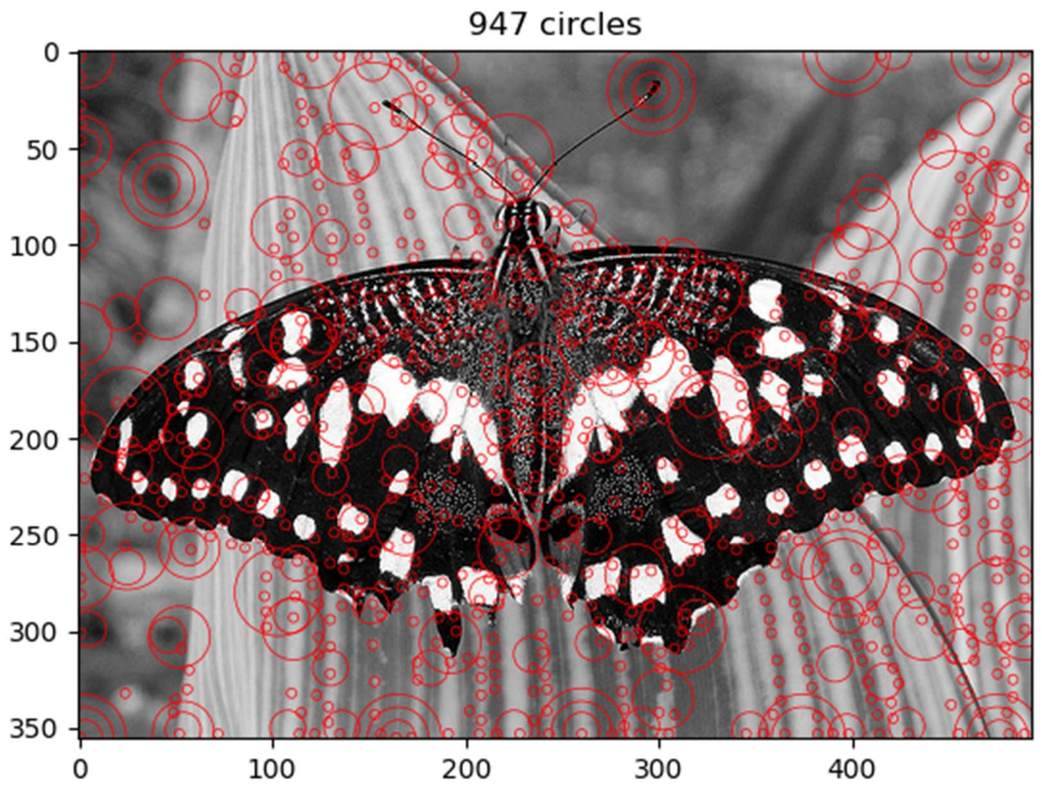
c) **Plotting characteristic circles**

Each circle has 3 parameters, namely, (x, y, radius) which can be superimposed on the original image for getting the blob output. To retrieve the blob markers, we find out where the maxima are present in the 3D NMS output array and calculate the radius of the circle corresponding to that scale. Before plotting the circles, we check the redundancy in the detected characteristic circles. The redundancy is characterized by the amount of overlap between two circles. The threshold for the amount of overlap is again a hyperparameter which can be tuned specific to an image. If the amount of overlap is greater than a certain threshold, the circle with larger radius is retained, leaving behind the other smaller ones. This redundancy has to be checked for every possible pair of circles. Since this is computationally expensive to implement off-the-shelf, we have used the in-built binary search algorithm to retrieve circle pairs for overlap checking. The output of the redundancy check function is the list of circles which can be plotted on the original image.

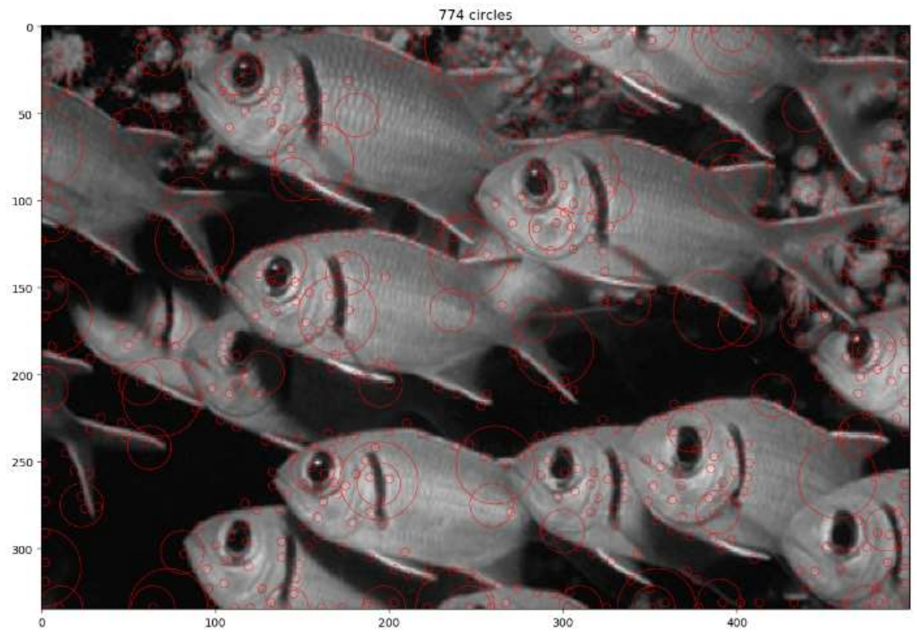
For plotting each of the circles, we traverse through the list and add each of the circular patches to a figure window which has the original image in the background.

Results and Figures

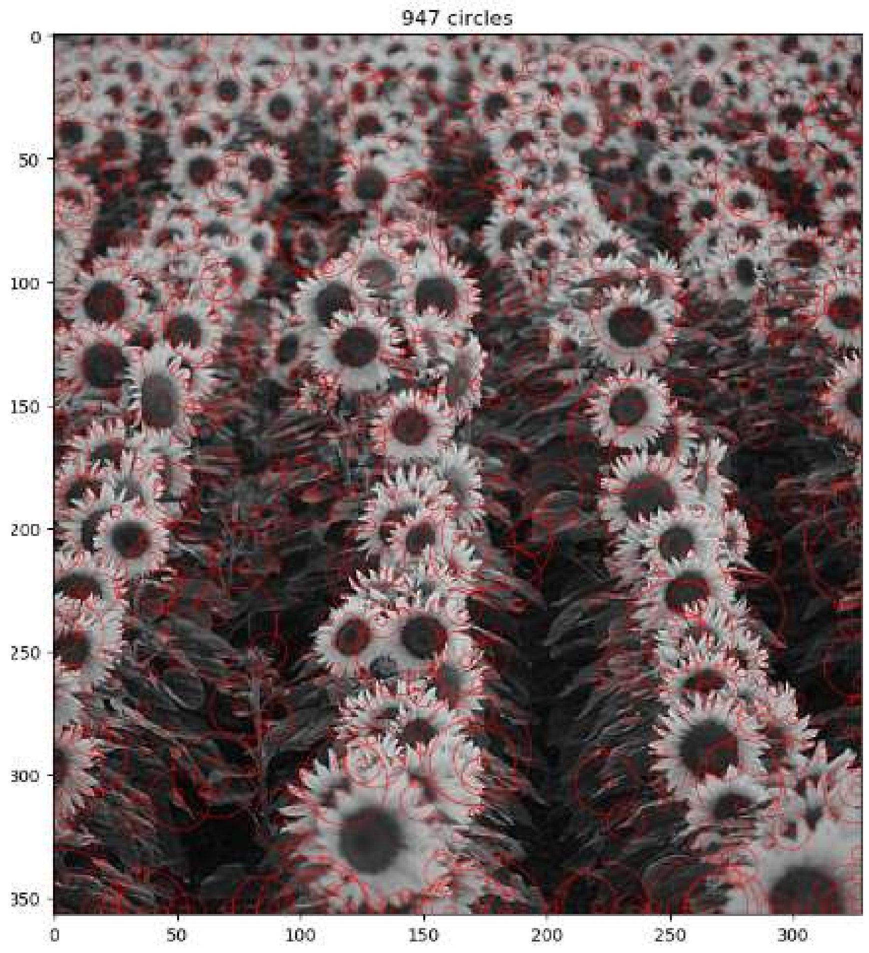
Run-time = 2.0709006786346436s



Run-time = 1.6827878952026367s



Run-time = 10.489546537399292s



Run-time = 2.99402117729187s