Computer Vision and Image Processing: Homework 1

Scale Space Blob Detection

**Name**: Manpreet Dhanjal

**UBIT Name**: DHANJAL

**Person Number**: 50248990

# Introduction:

Feature extraction from digital images is one of the fundamental requirements for any Computer Vision and Image processing maneuver. Some of the basic feature extraction techniques include edge detection, corner detection, blob detection, ridge detection etc. Edge detection only provides information about the edges in an image but it doesn’t help in recognizing a surfaces. Corner detection fails to detect corners in a zoomed in image since at the pixel level the corner will actually be curved and the filter might not be able to recognize it. Hence, to aid the edge and corner detectors we generally use blob detectors with them.

Blobs, in general sense, are regions in an image where certain properties like brightness, color etc. are approximately constant as compared to their surroundings. Unlike corner detector, a blob detector will always be able to detect blobs even if the image is enlarged or made very small and it gives information about the objects in the images.

Generally, blob detection is also used in object recognition and object tracking.

# Analysis & Results:

In order to be able to detect blobs of all the sizes in a given image or detect the same blobs in resized image, we need to make a scale invariant blob detector.

The scale space needed for blob detection can be calculated using 2 methods:

1. **Increasing filter size**: In this method, the size of image is kept constant but filters of different sizes are applied onto that image in each iteration.
2. **Down sampling the image**: In this method, the filter size is kept constant but the image is resized in each iteration and filter is applied to it.

## Outputs on images:

Following is the output of blob detector on images using the above 2 methods:

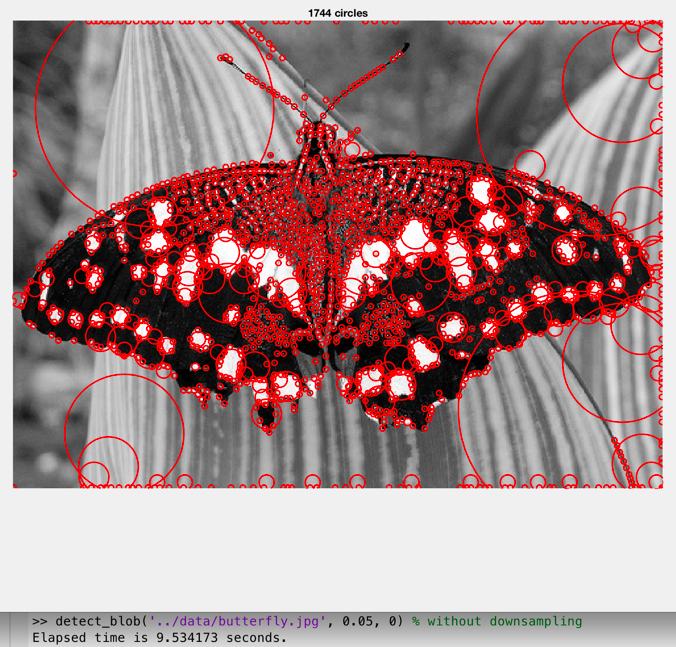


Figure 1: Left – variable filter size(9.5 s), Right – variable image size(1 s)

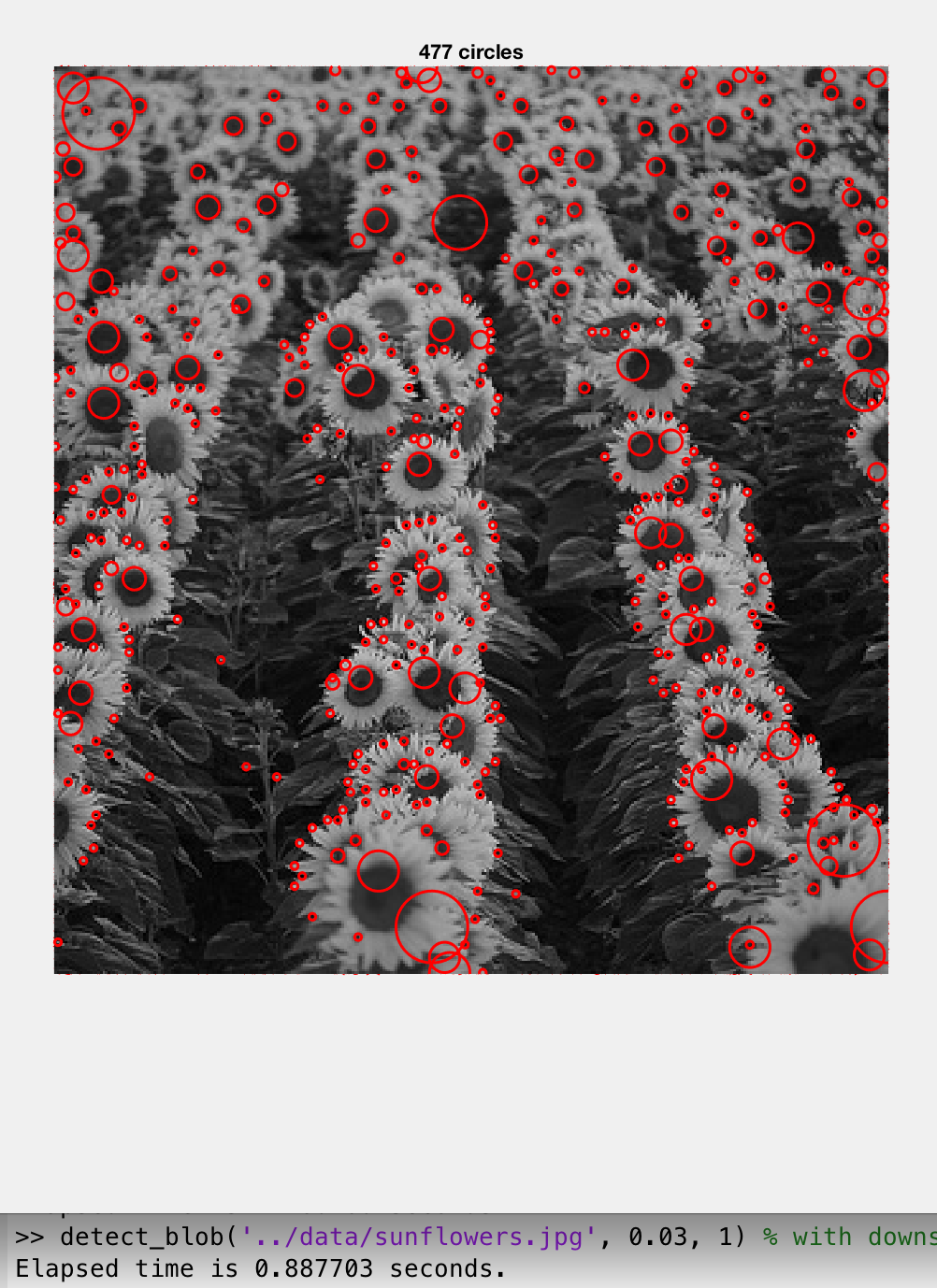
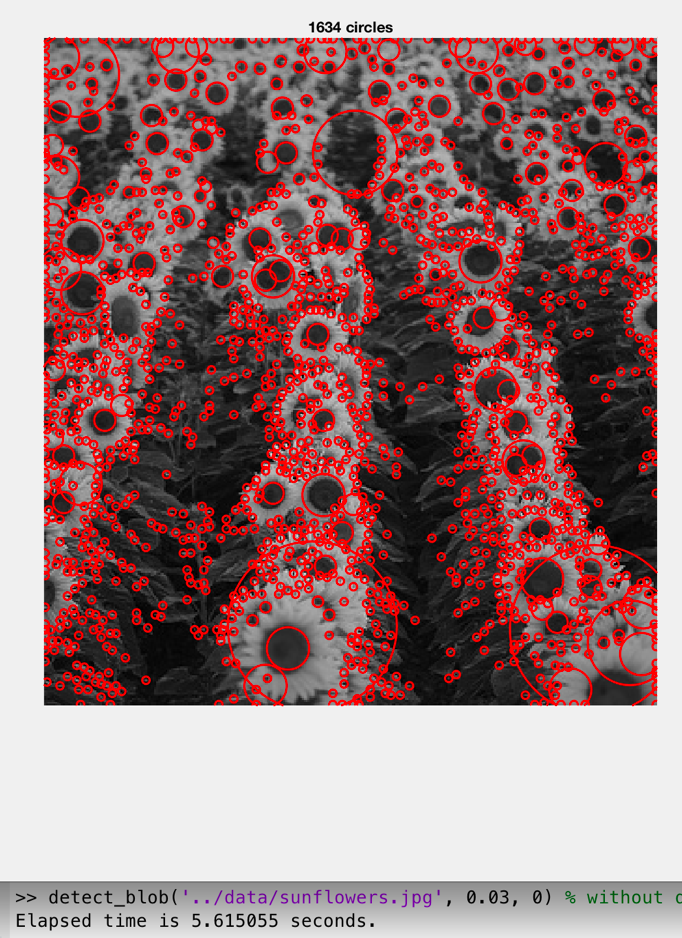


Figure 2: Left – variable filter size(5.6 s), Right – variable image size(0.8 s)

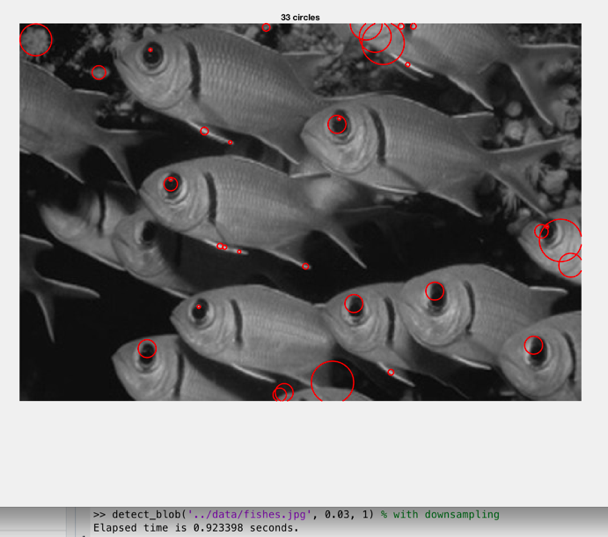
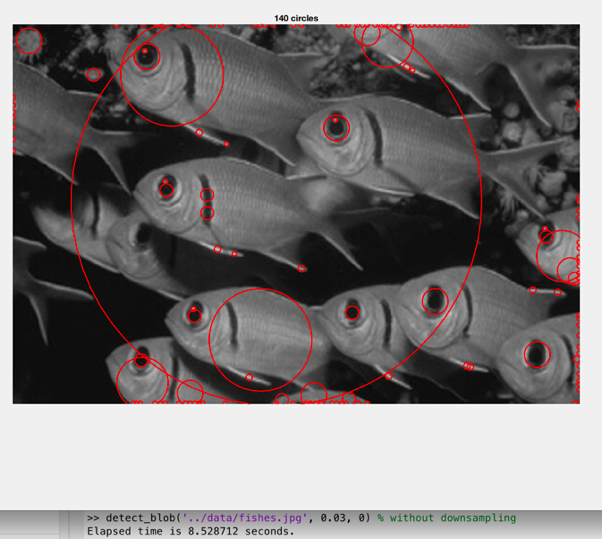


Figure 3: Left – Variable filter size(8.5 s), Right – Variable image size(0.9 s)

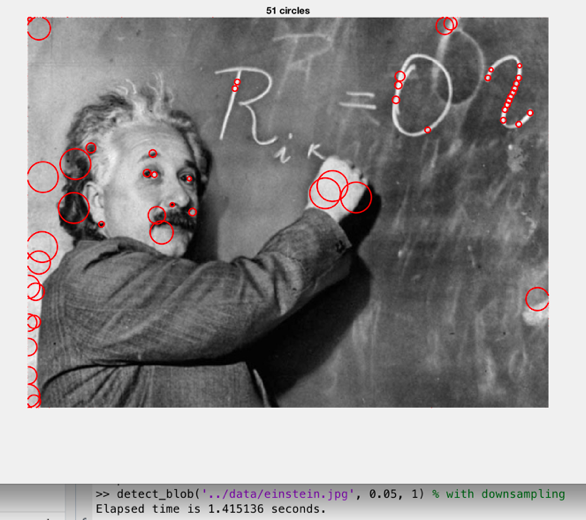
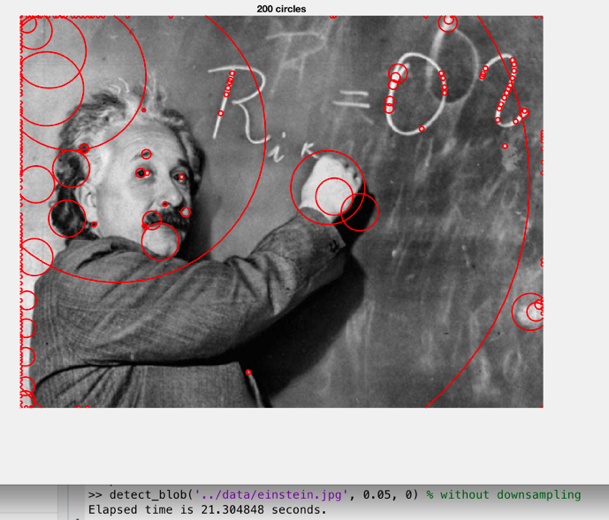


Figure 4: Left – Variable filter size(21.3 s), Right – Variable image size(1.4 s)

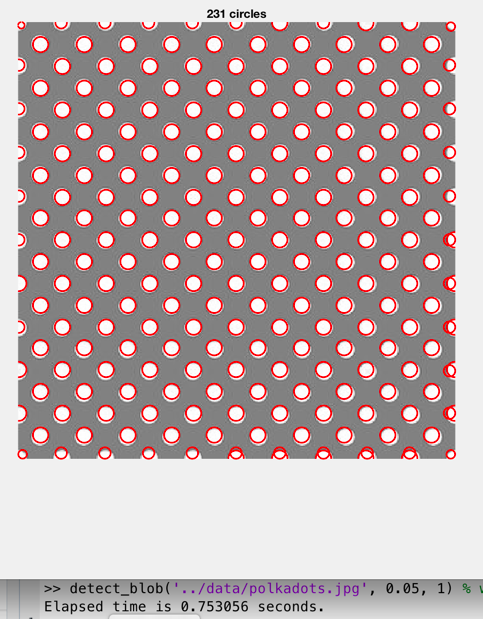
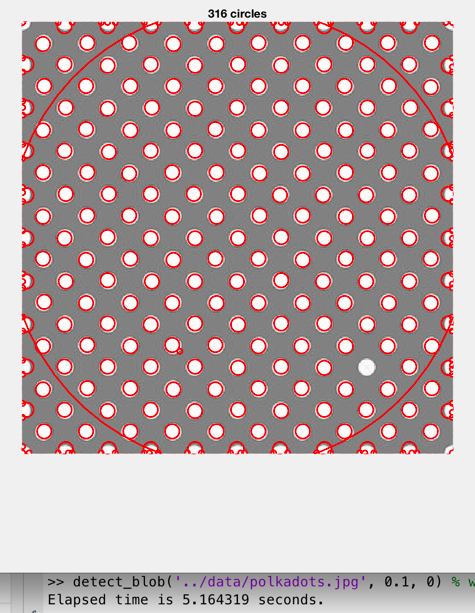


Figure 5: Left – Variable filter size(5.1 s), Right – Variable image filter(0.75 s)

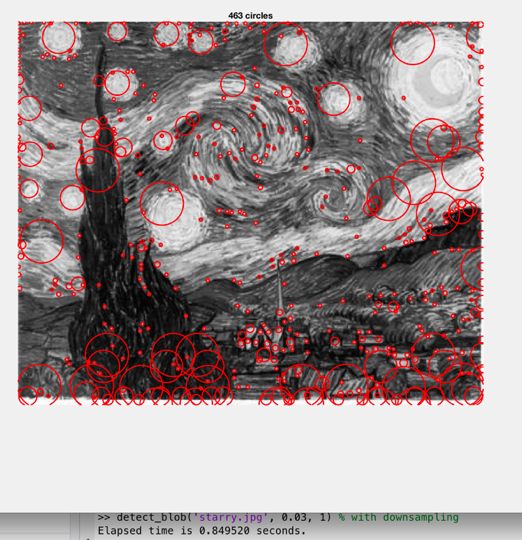
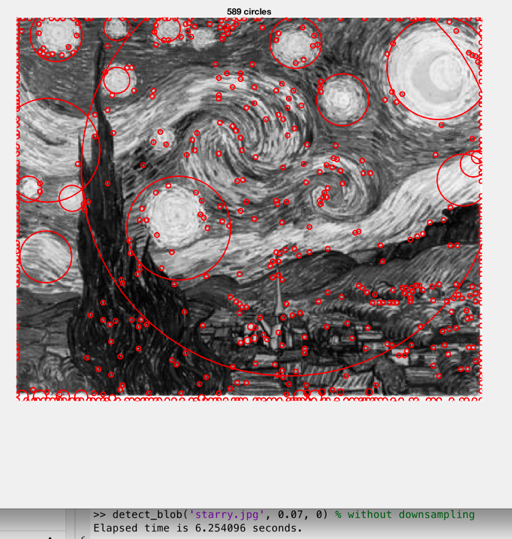


Figure 6: Left – Variable filter size(6.25 s), Right – Variable image size(0.8 s)

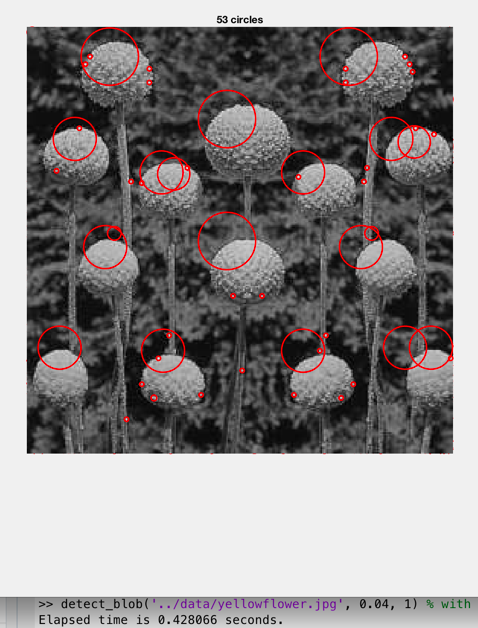
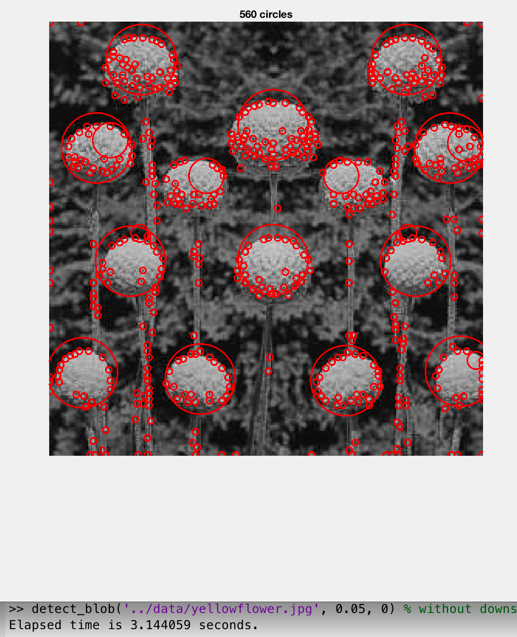


Figure 7: Left – Variable filter size(3.1 s), Right – Variable image size(0.4 s)

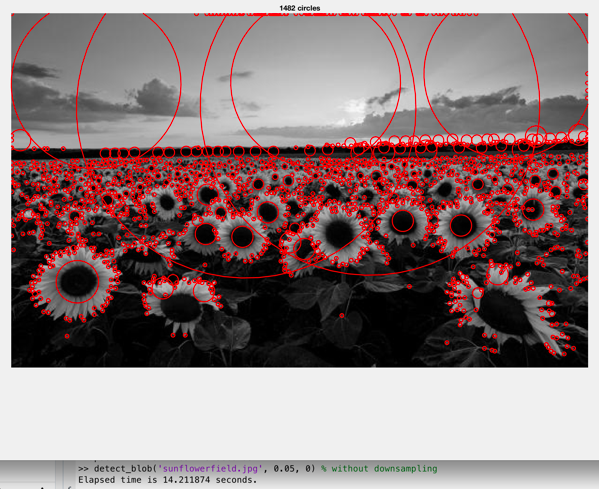


Figure 8: Left – Variable filter size (14.2 s), Right – Variable image size(1.2 s)

A comparison of the time taken by the 2 methods to detect blobs in a given image shows that down sampling the image is the efficient method. Based on the image size the time difference may vary from 4 seconds to 20 seconds. This could be attributed to the fact that the size of image reduces in each iteration and hence the number of calculations needed when performing convolution is far less as compared to when we are increasing the size of the filter and keeping the size of the image constant.

## Interesting Implementation choices:

Following are my interesting Implementation choices:

* **Method of resizing for image**: For resizing the image we have different interpolation methods like bicubic, bilinear and nearest. Bicubic interpolation is much better at retaining the intensity values in the image as compared to the other methods. Since, in this project the intensity values are of at most importance to us, hence bicubic interpolation should always be preferred.
* **Threshold values:** Deciding the threshold values is one of the most important part of this project. I tried to figure out a relationship between the maximum value of filter response at a particular scale and the threshold and used formulas similar to threshold = max(max(filter\_response)) \* 0.01. This formula resulted in different thresholds for different scale space values. But using a constant value of threshold for all the scales resulted in better results as compared to using variable threshold values.
* **Number of neighbors used in non-maximum suppression:** For non-maximum suppression in the scale space, I have first done the non-maximum suppression in 2D. This is done by comparing the value of pixel with all of its neighbors in a 5x5 window and assigning it the maximum value. To perform the non-maximum suppression in the scale space, I took the maximum for each pixel value in all the scale space and got the centers for the blobs. But this technique resulted in multiple circles side by side. Hence, I used another method to perform the non-maximum suppression in the scale space by comparing the pixel values with 2 layers above and 2 layer below it. This gave better outputs.

## Optimal parameter values:

Following are the parameter values that I tried along with the values that I found to be optimal:

* **Size of scale space**: This parameter indicates the number of times we resize the image incase of down sampling method and the number of times we increase the filter size incase of the increasing filter size method. According to my analysis a scale space of 10 is enough given the size of image data provided to us.
  + Increasing filter size method: In this method, if we go beyond 10, the filter size becomes greater than the size of the image. This leads to unnecessary calculations.
  + Down sampling image method: In this method, if we down sample the image more than 10 times, the size of image becomes negligible compared to the size of filter, which is the same case as above. So no useful output is obtained from this.
* **Factor k of resizing the filter or image:** k is the factor by which we will down sample the image or increase the size of the filter.
  + Increasing filter size method: For this project, I have used k = 2, and this gives good results.
  + Down sampling image method: I am down sampling the image by a factor of 0.75, hence k = 4/3 for this method. I initially tried down sampling the image by a factor of 0.5 i.e. k = 2 but it seemed to skip many of the blobs while filtering, hence I found k=4/3 to be an optimal value for this method.
* **Value of sigma:** The spread or sigma of Laplacian of Gaussian filter should increase when we increase the size of the filter. Hence, we need to derive a formula to get the size of filter according to the sigma or vice versa. Typical formulas for calculating the filter size using sigma are 10\*sigma or 2\*(3\*sigma)+1. According to my analysis, we need an odd sized filter so I have used the formula, 2\*ceil(sigma\*2.5)+1. This always returns an odd value and doesn’t contain redundant values as in the case when we choose the formula to be 10\*sigma.
* **Threshold:** According to my analysis, the threshold value varies from image to image but typically the threshold value lies between 0.03 to 0.05. My implementation of the blob detector expects a threshold value from the user but if it is unspecified it takes 0.04 as default value.
* **Radius at a given scale**: For a particular filter the radius of the blob can be decided according to the formula sigma \* square\_root(2). This is because, a blob will give maximum response to a filter only when the radius of the blob is approximately equal to the above mentioned formula and hence the radii are calculated to various scales. The same formula is used for the case of image down sampling also.

# Conclusion:

Although applying variable size filter on image takes more time compared to when we resize the original image and apply constant filter, the output results of the first case is better than the second one. Moreover, sometimes the centers of the detected blobs are shifted in the case of down sampling of image. This issue might be arising because of the change in the pixel location during up sampling.

# References:

* <https://en.wikipedia.org/wiki/Blob_detection>
* <http://www.cs.unc.edu/~lazebnik/spring11/lec08_blob.pdf>
* David G Lowe. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2):91–110, 2004.