uc3m | Universidad Carlos III de Madrid

Departamento de Teoría de la Señal y Comunicaciones

Scale-Space Blob Detectors

Audio processing, Video processing and Computer Vision

Lab exercise 2

Scale-space blob detectors

The goal of this lab exercise is to implement a scale-space blob detector based on the Laplacian of Gaussian (LoG) filter. You can follow this high-level script structure:

```
For s in scales
    Filter image with the corresponding LoG filter
    Square the filter response
    Increase the scale
Perform scale-space non-maximum suppression
Display the results
```

Let us working with the image "Sunflowers.jpg".

In the following, there are some hints to help you.

Use `scipy.ndimage.filters.gaussian_laplace` to implement LoG filter.

Explore the size of the sunflowers in the image and propose an appropriate scale-space. Follow this formula:

$$\sigma_k = \sigma_0 s^k, \qquad k = 0,1,2,\dots$$

In order to create the scale space, you have two options: either a) filtering the image with LoG filters of different scales; or b) keeping the same filter and down-sampling the image. Implement both options.

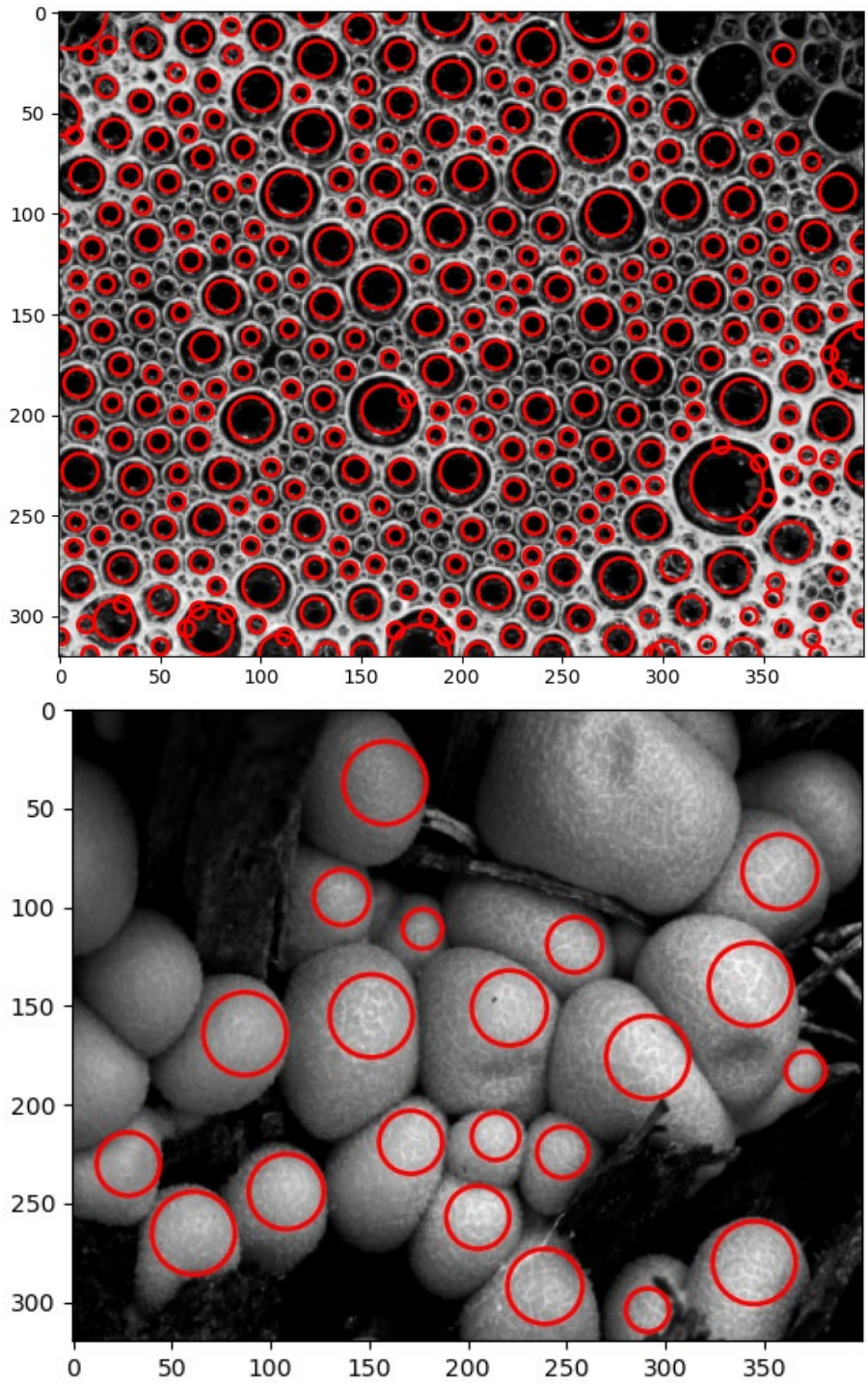
- Use `skimage.transform.resize` for down-sampling (and up-sampling, if needed).
- Which one is faster? Use `time.time()`.

To implement non-maximum suppression, compute a 2D non-maximum suppression independently for each scale and then extend the procedure to the 3D scale-space.

- You may use a 3D array to represent your scale-space (M, N, Ns), where MxN is the image dimension and Ns is the number of scales.
- You may use the function `scipy.ndimage.rank_filter`.

To display the results, you may use the function `plot_circles (image, cx, cy, rad)`, provided with the materials. Here you have an example of how the results may look like. In any case, your results may look different depending on your threshold, range of scales, and other implementation details.

Now, let's work two other images: `fruits.jpg` and `water_texture.jpg` (both photos by form PxHere). How would you choose the parameters of the algorithm to get these results? The parameterization should be different for each image.


0
50
100
150
200
250
300
0
50
100
150
200
250
300
350
0
50
100
150
200
250
300
0
50
100
150
200
250
300
350

2-page report

Include in you report the answers and explanations that follow:

- Which is the size of the LoG filter kernel?
- Which is faster, to use filters at different scales or to down-sample the image? why?
- How have you chosen the scale-space, $\sigma_k = \sigma_0 s^k$, when keeping the original size of the image?
- Assuming that you are using one LoG filter and down-sampling the image, how would you choose the sigma parameter of the LoG filter, i.e., the scale of the filter?
- Do you still need to normalize the scale of the filter response when you down-sample the image instead of increasing the scale of the filter?
- Explain how the non-maximum suppression function works.
- Explain your parameterization to get similar results to those shown with `fruits.jpg` and `water_texture.jpg`