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 using the Laplacian of Gaussian (LoG) filter.

### Non-maxima supression

Before diving into the development of the scale-space blob detector, let's first explore the non-maximum suppression algorithm and how to implement it.

Follow this pseudocode:

1.- Design a function to create a 2D Gaussian blob:

```
def gaussian(x, y, x0, y0, sigma):
```

where (x0, y0) represents the center of the Gaussian and sigma the standard deviation.

2.- Create a coordinate grid for the image, with x and y values ranging from -10 to 10.

3.- Generate an image by combining three Gaussian blobs with standard deviations of 1.2, 1.6, and 2.2. The Gaussians should be centered at:

(x0=-4, y0=-4) with a height of 5,
(x0=4, y0=-4) with a height of 10, and
(x0=0, y0=4) with a height of 8.

4.- Non-Maximum Suppression using `rank_filter`:

- Apply `rank_filter` to the image with a filter size of 3 and rank=-1 and visualize the result.

```
max_filter_output = rank_filter(image, rank=-1, size=3)
```

- Combine the original image with the output from `rank_filter` to retain only the local maxima and suppress all other values.

Display the original image, the rank_filter output, and the final result after non-maximum suppression.

## Exercise 1

To begin, we will filter the image "Sunflowers.jpg" using a Laplacian of Gaussian (LoG) filter. Adjust the filter's variance to match the size of the largest sunflowers in the image. The pseudocode for this process is as follows:

```
Filter the image using the LoG filter
Apply thresholding to the filtered image
Perform non-maximum suppression
Display the results
```

Here are a few hints to assist you:

- Use `scipy.ndimage.gaussian_laplace` to implement LoG filter.
- To display the results, you may use the `plot_circles (image, cx, cy, rad)` function, which is included in the materials.

## Exercise 2

Now, let's start building the blocks of the scale-space blob detector. You can follow this high-level script structure:

```
# Build the scale-space
for sc in scales
        Filter image with the corresponding LoG filter
        Increase the scale (variance of the filter)

# Detection in the scale-space
Thresholding and non-maximum suppression

# Displaying the results
Display the results
```

Keep using the image "Sunflowers.jpg".

Here are a few hints to assist you:

*# Build the scale-space*

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

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

  Select $\sigma_0$, $s$ and $k$ such that you can detect the smallest and the largest sunflower with a reasonable number of scales (10-15).
- Remember to normalize the filter to ensure the filter responses are comparable across different scales.
- Visually inspect some detections by examining the corresponding filter responses.

*# Detection in the scale-space*

- To implement non-maximum suppression, start by performing 2D non-maximum suppression independently at each scale. Then, extend the process to 3D scale-space.
  - You can represent the scale-space as a 3D array of size $(M, N, Ns)$, where $MxN$ is the image dimension and $Ns$ is the number of scales.
  - Consider using the function `scipy.ndimage.rank_filter` to assist with the implementation of non-maxima suppression.

*# Displaying the results*

Here you have an example of how the results may look like. Try to select the parameters of the algorithm to achieve a similar result.

## Exercise 3

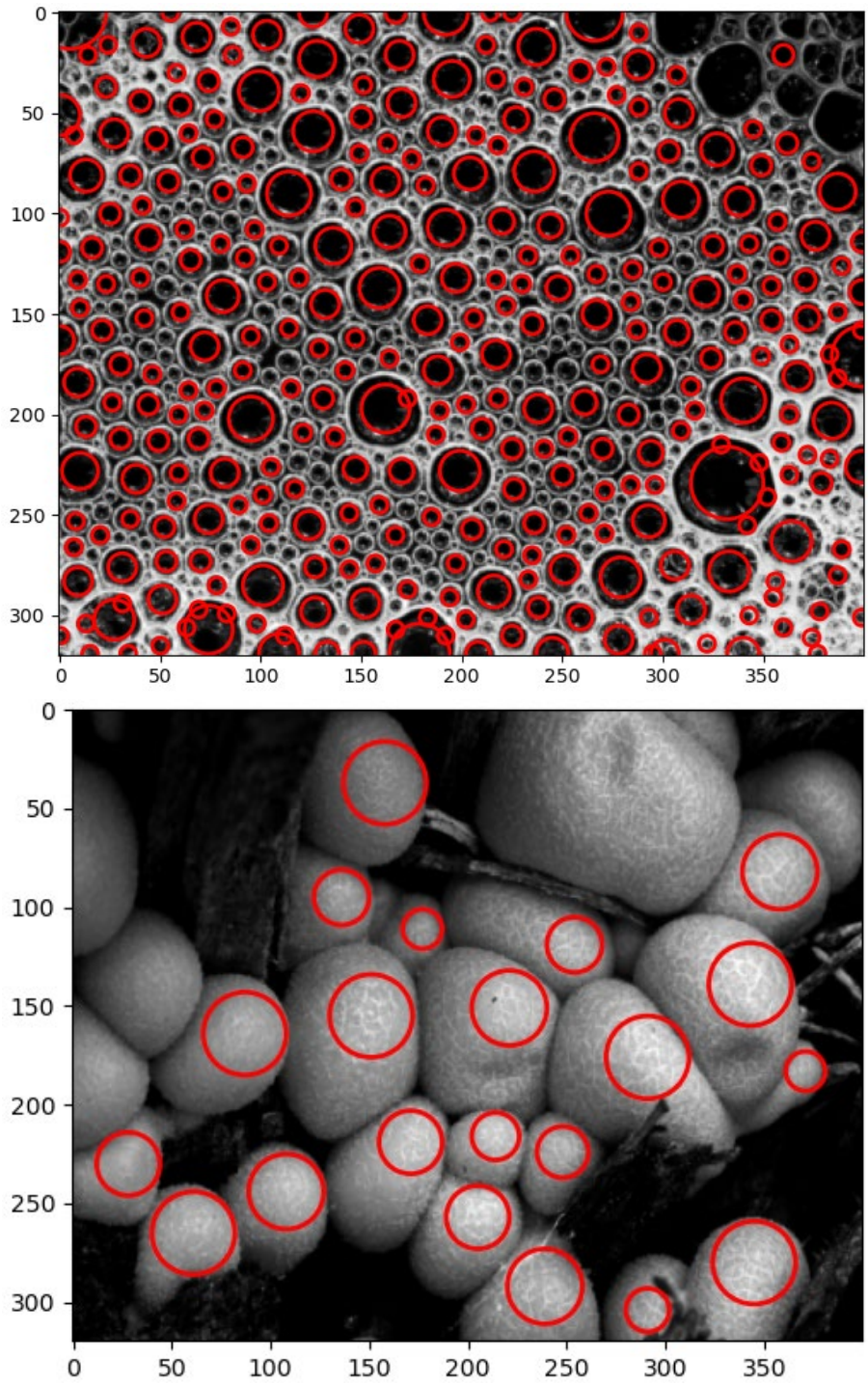
We have an alternative approach to creating the scale-space: instead of changing the filter size, you can downsample the image while keeping the filter the same. Let's try this second option:

- 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 filter response at different scales?
- Use `skimage.transform.resize` for downsampling (and upsampling, if necessary).

Which of the two approaches—changing the filter width or downsampling the image—is faster, and why? Use `time.time()` to measure the execution time.

## Exercise 4

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




## Evaluation

The evaluation for this lab will take place on **October 7th (G96)** and **October 9th (G97)**. You will be required to complete a 15-minute quiz related to this lab. Please bring your laptop, as the quiz will be conducted online and may involve writing a small program on the spot.