Bachelor in Data Science and Engineering - 2021/2022
University Carlos III Madrid
Audio processing, Video processing and Computer vision

SERGIO AIZCORBE (**100406602**) | BERNARDO BOUZAS (**100406634**) | LS2 REPORT

1. Which is faster, to use filters at different scales or to down-sample the image? Why?

Both functions we have implemented to address this comparison, ***increase_log_filter_size()*** and ***downsample()***, calculate the squared Laplacian response for each scale.

In the first case, the filter domain is larger when we increase sigma. Specifically, for test values of sigma, we update at each iteration such that $\sigma = \sigma^* k^i$. Since the scale grows exponentially from lower to higher level, the filter has a larger domain to compute the response, hence the computational cost is very high.

In contrast, in our second method we down-sample the image first, and then the response to the same filter is up-sampled. The size of the filter is not modified at any time, plus the image domain gets shrinked; the size of the image reduces in each iteration, reducing the number of calculations needed when performing the convolution.

Downsampling the image, then, results in a computationally cheaper, and hence faster, approach.

2. 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?

When using the LoG filter, the first thing to do is to consider the scale, studying the **sigma parameter**, which is fundamental to obtain an optimal kernel function. The more we increase sigma, the wider the radius over which we will blur, which implies that we are forced to use a larger kernel matrix to capture enough of the function's energy. In this case, we need a sufficiently large kernel for the blur to be adequate, i.e., to cover a sufficiently satisfactory part of the image. Also, we must be careful not to overshoot, or it will start overlapping multiple neighbors at a time. We get the optimum sigma value by looking at the one which maximizes the magnitude of the Laplacian response.

3. 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?

In the approach in which we increase the scale of the filter, the response of the derivative of the filter to a perfect step edge decreases as σ increases. That is why we need to keep the response invariant, normalizing it. However, when we downsample the image without changing the filter, the response across different scales **doesn't actually decay**, as we don't need to change sigma. As it is stated, the response is already scale-invariant in the down-sampling approach, so we don't need to normalize.

4. Explain how the nms function works.

Our given nms function implements a **non-maximum suppression method**, which consists of excluding those pixels that, compared to their neighbors, have a lower filter response. All of these possible neighborhood zones are the candidate regions. In our case, the first part of the function preserves the greatest value among the region and converts to 0 the rest. This ensures that we only take into account pixels, whose filter response value is significant.

Once we have chosen only the candidates along all the scales, the given function obtains their positions, i.e., their coordinates, as well as the values of their responses. In order to correctly implement the proposed non-maximum suppression algorithm, it sorts these candidates. Then, it identifies the candidates with the highest filter response value and compares them with the rest. For this, the function receives as input a minimum distance, and so we can define the overlapping threshold to take into account when comparing. Those candidates with high overlap are discarded. This concludes the filtering of the candidates according to their response value and overlap, and the function returns the list of coordinates of candidates that optimize the criteria.

5. Explain how you implemented the 3D scale-space non-maximum suppression.

In our 3D scale-space nms function, non-maximum suppression is performed across scales. We apply a **maximum filter function to obtain the maximum values among the region (3 x 3 x No. of scales)**. Essentially, this allows us to compare the candidates of each level that present optimal conditions of magnitude of response with those that are in similar situations in the rest of the scales. The result will be, similar to the nms for pixels: keep the extremas that, this time across all scales, will provide the most accurate blob representation, and discard those that are already enclosed in the optimal 3D regions.

Finally, blobs are returned by comparing non-zero squared responses to the threshold in 3D space. Moreover, a blob will offer the maximum response to a filter only when the zeros of the Laplacian are aligned with it. This happens when the radius of the blob is approximately equal to $r=\sigma\sqrt{2}$, so this is the radius we define.

6. What did you find most difficult about this lab exercise?

In addition to learning the necessary knowledge to tackle the problem with the correct approach, we have found special difficulty regarding the **choice of parameters** and the lack of a standardized methodology for it. We have found a multitude of opinions and approaches to decide the size of the scale space, the image downsampling resize factor, sigma or threshold. This made us doubt if the results had more to do with these parameters or if they were due to our implementation. Even so, by doing research and understanding the contents of the course we believe we have done a good job designing and fine-tuning our methods.