TECHNICAL UNIVERSITY OF CRETE

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

CS418: COMPUTER VISION

Instructor: Effrosyni Doutsi

Assignment 2: Blob Detection

Due date: Thursday, May 6, 2021

This assignment is adapted from Svetlana Lazebnik

The goal of this assignment is to implement a Laplacian blob detector. The main step of your algorithm should be briefly the following ones:

1. Generate a Laplacian of Gaussian filter
2. Build a Laplacian scale-space, starting with some initial scale and going for n iterations. You should filter the initial image with a scale-normalized Laplacian, keep the Laplacian response and generate the next scale by increasing the factor k.
3. Find the extrema in the scale-space
4. Display the resulting circles at their characteristic scales.

Dataset

You are given 2 images that you need to use for testing your code (use ALL of them). You should keep in mind that the result depends on the initial threshold, the range of scales and other implementation details thus reproducing exactly the same results will be awkward. In addition to the images provided, please run your code on at least 2 more images that you like and you think they are informative enough for this kind of a problem.

Implementation Instructions

- Don't forget to **convert images** to grayscale (rgb2gray command) and double (im2double).

- For creating the **Laplacian filter**, use the *fspecial* function (check the options). Pay careful attention to setting the right filter mask size. [Hint: Should the filter width be odd or even?]
- It is relatively inefficient to repeatedly filter the image with a kernel of increasing size. Instead of increasing the kernel size by a factor of k, you should downsample the image by a factor 1/k. In that case, you will have to upsample the result or do some interpolation in order to find maxima in scale space. [Hint: pay attention to the interpolation method you need to use] For extra credits, you should turn in both implementations: one that increases filter size, and one that downsamples the image. In your report, list the running times for both versions of the algorithm and discuss differences (if any) in the detector output. For timing, use tic and toc commands.
- You have to choose the **initial scale**, the factor k by which the scale is multiplied each time, and the number of levels in the scale space. A typical scenario is to set the initial scale to 2, and use 10 to 15 levels in the scale pyramid. The multiplication factor should depend on the largest scale at which you want regions to be detected.
- You may want to use a three-dimensional array to represent your **scale space**. It would be declared as follows:

  scale_space = zeros(h,w,n); where [h,w] - dimensions of image, n - number of levels in scale space

  Then scale_space(:,:,i) would give you the ith level of the scale space. Alternatively, if you are storing different levels of the scale pyramid at different resolutions, you may want to use a cell array, where each "slot" can accommodate a different data type or a matrix of different dimensions. Here is how you would use it:

  scale_space = cell(n,1); that creates a cell array with n "slots" scale_space{i} = my_matrix;, store a matrix at level i

- To find the **extrema** you may find functions *nlfilter*, *colfilt* or *ordfilt2* useful. Play around with these functions, and try to find the one that works the fastest. To extract the final nonzero values (corresponding to detected regions), you may want to use the *find* function.
- You also have to set a **threshold** on the squared Laplacian response above which to report region detections. You should play around with different values and choose one you like best.
- To **display** the detected regions as circles, you can use the function provided in the assignment package (or feel free to search for a suitable MATLAB function or write your own). Hint: Don't forget that there is a multiplication factor that relates the scale at which a region is detected to the radius of the circle that most closely "approximates" the region.

What to turn in:

You should turn in both your **code** and a **report** discussing your solution and results.

1. The report should contain a description of your algorithm and any decisions you made to write your algorithm a particular way. Then you will show and discuss the results of your algorithm. Show the output of your circle detector on all the images (2 provided and 2 of your own choice), together with running times for both the "efficient" and the "inefficient" implementation. An explanation of any "interesting" implementation choices that you made. An explanation of parameter values you have tried and which ones you found to be optimal. A good writeup doesn't just show results, it tries to draw some conclusions from your experiments.

Turning in the Assignment

Your submission should consist of the following:

- All your code and output images **in a single zip file**.
- A brief report **in a single PDF file** with all your results and discussion.

Academic integrity: Feel free to discuss the assignment with each other in general terms. Coding should be done a product of each individual team. If you use existing material, be sure to acknowledge the sources in your report.