EECS 442: Computer Vision, Winter 2018
Homework 3: Scale-space blob detection
Due Date: February 15 by 11:59 P.M.

Basic Project Outline

1. Generate and apply a Laplacian of Gaussian filter at a variety of scales. You will develop two separate implementations:
   a. First, try increasing the size of the filter to detect blobs at different sizes.
   b. Next, write code to scale the image instead before applying the filter.
2. Perform nonmaximum suppression in scale space.
3. Display resulting circles.

Test images

The data folder contains four images to test your code, and output folder contains sample output images for your reference. Keep in mind, though, that your output may look different depending on your threshold, range of scales, and other implementation details.

Running the code

Download the code and data and run evalCode.m. It runs a dummy implementation of the code and draws the blob (a circle in the center of the image). On running the code the output should be something like this (exact times will vary based on the machine):

```
>> evalCode
Elapsed time is 0.007286 seconds.
Elapsed time is 0.002676 seconds.
```

And two identical images that look like this one:
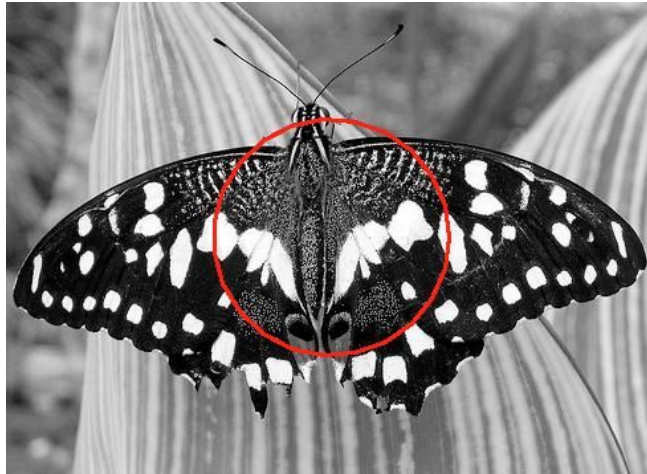
Detailed instructions

Here are the key steps to implement the blob detector:
- Don't forget to convert images to grayscale (rgb2gray command) and double datatype(im2double command).
- To create 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. **For full credit, you should turn in both implementations: one that increases filter size** detectBlobsScaleFilter.m**, and one that downsamples the image** detectBlobsScaleImage.m. In your report, list the running times for both versions of the algorithm and briefly explain the differences. For timing, use the tic and toc commands.
    - **Hint 1**: Think about whether you still need scale normalization when you downsample the image instead of increasing the scale of the filter.
    - **Hint 2**: For the efficient implementation, pay attention to the interpolation method you're using to upsample the filtered images (see the options of the imresize function). What kind of interpolation works best?
- 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. I typically 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:

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

Then scaleSpace(:,:,i) would give you the i-th 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:

```
scaleSpace = cell(n,1); %creates a cell array with n "slots"
scaleSpace{i} = myMatrix; % store a matrix at level i
```

- To perform nonmaximum suppression in scale space, you should first do nonmaximum suppression in each 2D slice separately. For this, 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 drawBlobs.m . You should display the highest scoring 1000 detected blobs for each image. If your code returns fewer than 1000 blobs then you may have to lower your threshold. **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.

Helpful resources

- [Blob detection](#) on Wikipedia.
- D. Lowe, "[Distinctive image features from scale-invariant keypoints](#)," International Journal of Computer Vision, 60 (2), pp. 91-110, 2004. This paper contains details about efficient implementation of a Difference-of-Gaussians scale space.
- T. Lindeberg, "[Feature detection with automatic scale selection](#)," International Journal of Computer Vision 30 (2), pp. 77-116, 1998. This is advanced reading for those of you who are really interested in the glory mathematical details.

Report instructions

As before, you must turn in both your report and your code. Your report should have the following sections and will be graded on these items:

1. The "inefficient" implementation
   a. Output of your circle detector on **ALL** provided images.
   b. Running time.
   c. Briefly explain how you relate the size of filter, sigma and blob radius.
   d. Briefly explain how you store different levels of the scale space.
   e. Report the parameters you use, including the initial scale, the number of level, and the threshold on the squared Laplacian response, etc.
2. The "efficient" implementation
   a. Output of your circle detector on **ALL** provided images.
   b. Running time.
   c. Briefly explain how you store different levels of the scale space.
   d. Briefly explain how you resize the image for detecting blobs of different scales, what interpolation method you use to upsample the filtered images.
   e. Briefly explain whether you still need scale normalization when you downsample the image instead of increasing the scale of the filter.
   f. Report the parameters you use, including the initial scale, the number of level, and the threshold on the squared Laplacian response, etc.
3. Nonmaximal suppression
   a. Briefly explain how you perform nonmaximal suppression across **scales**.
   b. Briefly explain how you perform nonmaximal suppression across **space**.
4. Briefly explain how scaling image helps improve the speed over scaling filter.

Instructions for submitting the homework

On **Canvas** submit the following files:
- detectBlobsScaleFilter.m
- detectBlobsScaleImage.m

On **Gradescope** submit the following files:
- report.pdf

Acknowledgements