Blob Detection Report

Algorithm Outline:

1. Generate Laplacian of Gaussian Filter

The following equation is used to calculate Laplacian of Gaussian (LoG),

$$LoG(x, y) = -\frac{1}{\pi\sigma^4}[1 - \frac{x^2 + y^2}{2\sigma^2}]e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

After further solving the above equation,

$$\boldsymbol{LoG(x, y) = [\frac{x^2 + y^2 - 2\sigma^2}{2\pi\sigma^6}]e^{\frac{-(x^2+y^2)}{2\sigma^2}}}$$

The function log(x, y, sigma) calculates the Laplacian of Gaussian at a given point given the sigma/variance. The function uses math library and basic mathematical operations.

The function generate_log(sigma) calculates the LoG kernel of a particular size which depends on the given sigma value. The function log() is used to generate LoG. The kernel size should always be odd and thus, is handled in the function itself.

Both the functions are in generate_log.py file.

These functions are further used to generate a sigmalist. This list is generated after taking the input (number of sigma values) from the user and multiplying with a constant k = $\sqrt{2}$ at every step.

2. Build a Laplacian scale space starting with initial scale and going for n iterations

Laplacian scale space is a stack of input images convolved with LoG masks generated above and squared. The Laplacian scale space is displayed for a scale of 5 in project03.ipynb file. The convolution function was reused from the project01 and is implemented in convolution.py file.

3. Filter image with scale-normalized Laplacian at current scale

The scale-normalized Laplacian kernels/ masks are used for further calculation and these masks are calculated using $\sigma^2 * kernel$.

These masks are displayed in project03.ipynb file with a scale = **5**.

4. Perform non-max suppression in scale space

After convolution with scale-normalized Laplacian kernels/ masks, we have to make sure to suppress values less than a threshold. I used threshold = 0.008 which gave better output as compared to standard values. This threshold is passed to function nms_2d(slice, threshold) in nms_2d.py file. The slice here means one instance from Laplacian scale space. The non-max suppression is carried out in two stages,

a. Checking if the centre value is greater than the neighbouring 8 pixels (3*3 window) values and setting to 1 if true, else, 0. This operation is done in the same scale space instance/ slice and thus the name nms_2d. This modification is saved as binary_laplacian_scale_space.

b. Checking if the centre value is greater than the 9 values above and below (3*3*3 cubic window) a particular scale space instance/slice. In the edge cases, i.e. the first slice and the last slice, I checked with values below and values above only respectively. I used specific conditions while handling edge cases. I checked for the max value from above and below slice using np.max() function. The nms_3d(binary_laplacian_scale_space, laplacian_scale_space) function in nms_3d.py file.

5. Display resulting circles at their characteristic scales

After completing the above steps, using the cv2.circle() function, display the maxima detected using circles at the characteristic scales/ slices. Finally, saving the output images using cv2.imwrite().

Details:

The spalarp_project03 folder contains 8 output images after taking inputs from TestImages4Project folder.

The experimental_codes folder contains trial notebooks used for understanding, analysing and visualizing some of the above-mentioned functions.

My code, project03.ipynb code runs using the helper files – convolution.py, generate_log.py, nms_2d.py, nms_3d.py. All these files are in spalarp_code folder. The user may enter the number of scales they want to generate the blobs for. The scale value of **5** is used to generate the output images.