CS 485/685 – Computer Vision

Programming Assignment 1

(Due February 20)

Note: Before displaying your results, normalize all pixel values to make sure they are in the range [0, 255]. To read image files you may use the code provided on the course website or OpenCV functions.

1. (30 pts) Gaussian Smoothing

   (a) Implement 1D Gaussian smoothing. Apply a 1D Gaussian mask on the 1D data in the "Rect_128.txt" file. Show (as a graphic plot) and discuss your results using σ = 1, 5, and 11. You may use the code provided on the course website to generate the 1D Gaussian masks needed in your experiments (mask size = 5σ).

   (b) Implement 2D Gaussian smoothing. Extend the 1D code to generate 2D Gaussian masks. Apply a 2D Gaussian mask on the *lenna* image using σ = 1, 5, and 11. Show and discuss your results.

   (c) Implement 2D Gaussian smoothing using 1D Gaussian masks as discussed in class. Show your results on the *lenna* image using σ = 1, 5, and 11. Compare your results with those from (b).

   Submit the following:

   - (hardcopy) A report describing your results; it must include the masks used, all original and smoothed images, a discussion of results and comparisons; for all figures provide captions with a brief description.
   - (Canvas) One ZIP file "PA1_pb1.zip" containing:
     - The report file "PA1_pb1_report.pdf" mentioned above.
     - The source code files in C/C++.
     - A "README.txt" file with instructions on how to compile and run the program and any parameters that need to be set.

2. (30 pts) Edge Detection

   Implement edge detection using Sobel masks for horizontal (S_y) and vertical (S_x) edges. Given an image I(x,y), your program should compute and display the following images:

   (a) the gradient image in the x direction: $I_x(x,y) = I(x,y) * S_x(x,y)$,

   (b) the gradient image in the y direction: $I_y(x,y) = I(x,y) * S_y(x,y)$,

(c) the gradient magnitude image: $M(x,y) = | I_x(x,y) | + | I_y(x,y) |$

(d) the edge image by thresholding M(x,y) (the threshold should be specified by the user).

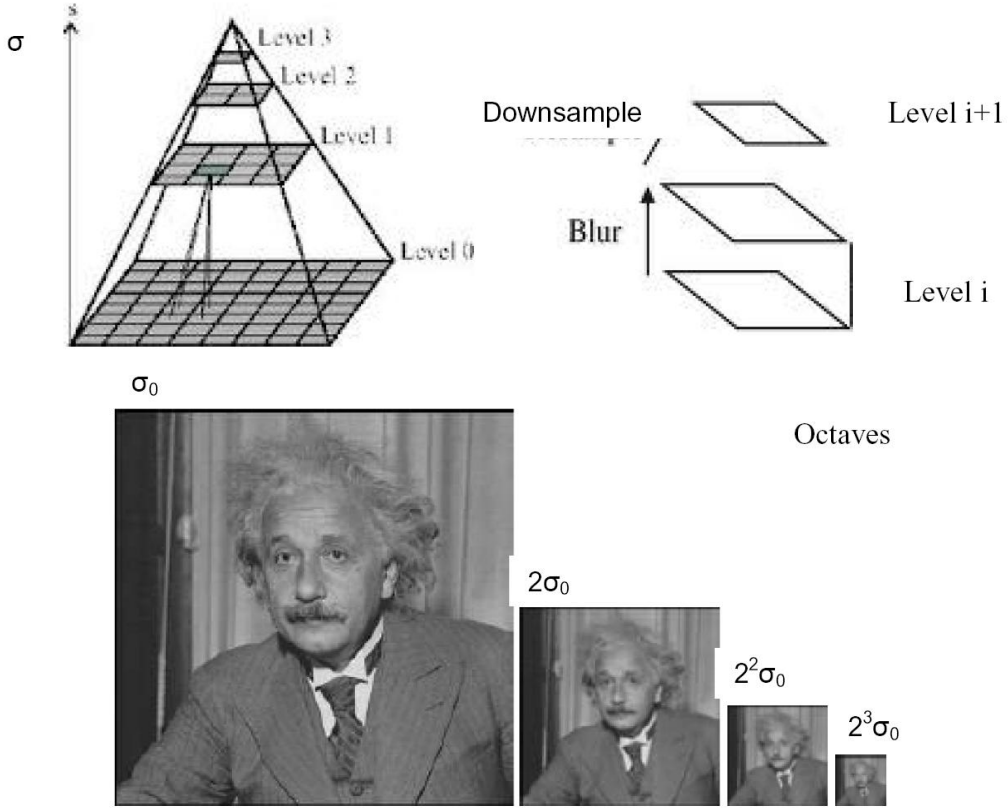
Show your results on the *lenna* and *sf* images.

Submit the following:

- (hardcopy) A report describing your results; for all figures provide captions with a brief description.
- (Canvas) One ZIP file "PA1_pb2.zip" containing:
  - The report file "PA1_pb2_report.pdf" mentioned above.
  - The source code files in C/C++.
  - A "README.txt" file with instructions on how to compile and run the program and any parameters that need to be set.

3. (40 pts) Gaussian Pyramid

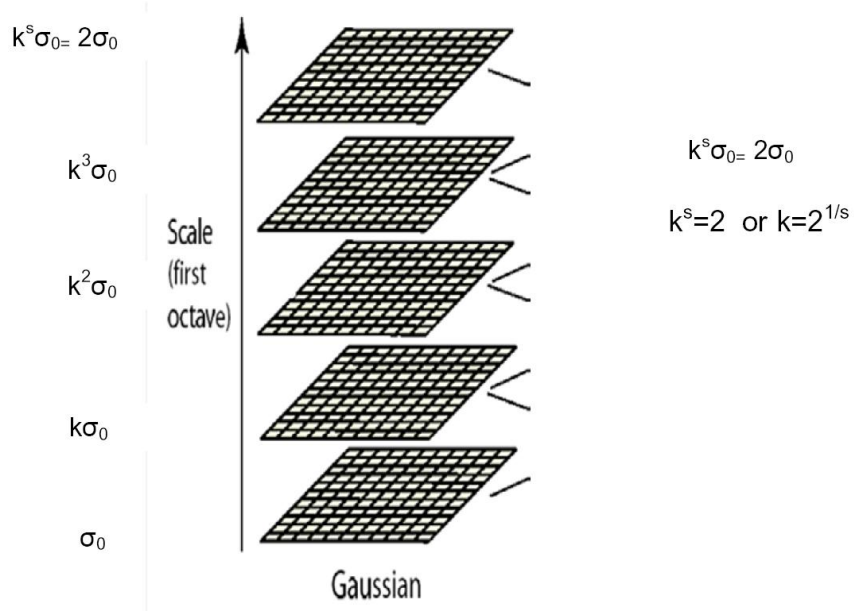
Write code to compute the Gaussian pyramid of an image. There are n levels (octaves) in the pyramid, each consisting of s intermediate levels. Within each level, smoothing (blurring) with a different scale occurs at each intermediate level. Before proceeding to the next level, a step of down-sampling by 2 is performed (see figure below).



An initial scale σ_0 is given. Each level (octave) should consist of s intermediate levels; the scale step k between intermediate levels in the same octave should be $k=2^{1/s}$. For example:

- In the first octave shown in the figure below, smooth the original image with σ_0, $k\sigma_0$, $k^2\sigma_0$, …, $k^s\sigma_0 = 2\sigma_0$ to generate the intermediate levels. Then down-sample by 2 the last image generated; the result becomes the starting image of the second octave.
- In the second octave, continue by smoothing the starting image of this octave with σ_0, $k\sigma_0$, $k^2\sigma_0$, …, $k^s\sigma_0 = 2\sigma_0$. Then down-sample by 2 and move to the next octave.

The initial scale σ_0, the number of levels (octaves) in the pyramid n and the number of intermediate levels s in each octave should be parameters of your program. Show your results on the *lenna* image.



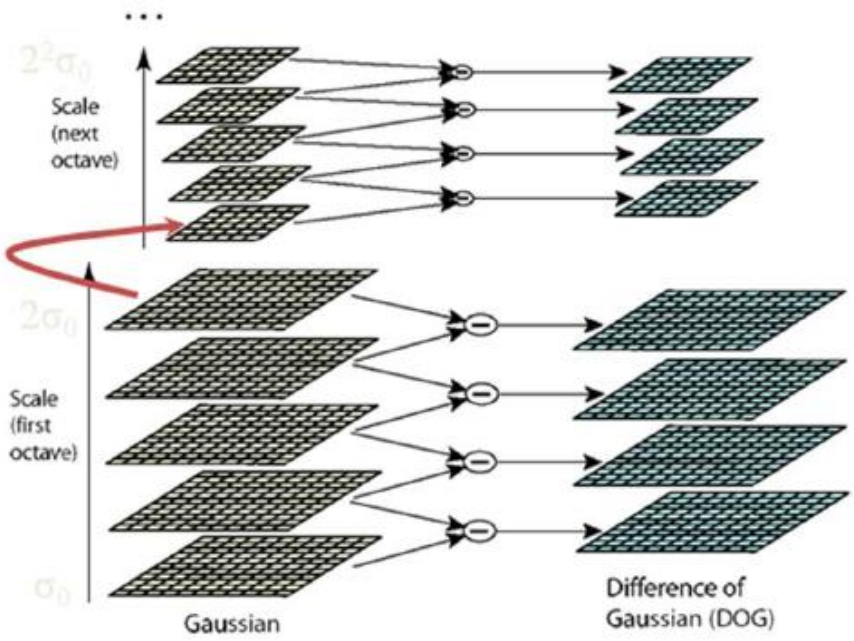
Submit the following:

- (hardcopy) A report describing your results; it must include all parameters used, all images at all levels (and intermediate levels), a discussion of results; for all figures provide captions with a brief description.
- (Canvas) One ZIP file "PA1_pb3.zip" containing:
  - The report file "PA1_pb3_report.pdf" mentioned above.
  - The source code files in C/C++.
  - A "README.txt" file with instructions on how to compile and run the program and any parameters that need to be set.

4. (10 pts – mandatory for graduate students, optional as extra credit for undergraduate students) Difference-of-Gaussians (DoG) Pyramid

   Write code to compute the DoG pyramid of an image. The DoG pyramid can be built by subtracting consecutive levels of the Gaussian pyramid (see figure below). As discussed in

class, the difference between two Gaussian functions approximates the Laplacian of Gaussian (LoG). Therefore, the DoG pyramid approximates the LoG pyramid. Note that in order to build a DoG pyramid that uses all levels of the Gaussian pyramid, you would need to create some extra levels in the Gaussian pyramid (i.e., one below the bottom level and one above the top level). Show your results on the *lenna* image.



Submit the following:

- (hardcopy) A report describing your results; it must include all parameters used, all images at all levels (and intermediate levels), a discussion of results; for all figures provide captions with a brief description.
- (Canvas) One ZIP file "PA1_pb4.zip" containing:
    - The report file "PA1_pb4_report.pdf" mentioned above.
    - The source code files in C/C++.
    - A "README.txt" file with instructions on how to compile and run the program and any parameters that need to be set.