CS 5420 **Digital Image Processing**
Assignment # 4 Due Date: November 14, 2018

Image Operations in Spatial Domain

1. Write a program to add Gaussian noise into an image. The gaussian noise is defined as addition or subtraction from the actual value of a pixel. The easiest way to create this noise will be to flip random bits in pixels. You can write code to flip $n\%$ of bits (not pixels) in the image where n is a command line parameter. Use this program to generate at least ten images from a given image. These ten images will be saved as separate files, possibly in a vector of type `cv::Mat`. Add the ten generated images together and compute the distance of the added image from the original image. The executable should be called `add_rm_noise`. Display the original, the intermediate images, and the added image.

2. Write a program that will allow you to zoom into a selected area of a given image with a specified zoom factor. The zoom factor is specified as an integer in the range $[1, \max(r, c)]$ where r and c are the number of rows and number of columns in the image, respectively. Zoom is to be achieved by pixel replication starting at the specified point (center of zoom) and growing outwards, within a specified window. The area outside of zoom window should be left undisturbed. Let us illustrate the command as follows, on a 640×480 image, named `image.jpg`.

   ```
   zoom -z 5 -R 200 -C 300 -H 100 -W 100 image.jpg out.jpg
   ```

   The options R and C specify the center of zoom window and H and W specify the height and width of zoom window. Thus, the coordinates of zoom window are $(150, 250)$ for bottom left and $(250, 350)$ for top right (specified as (r, c)).

   The algorithm will proceed as follows. Replace the pixels in the 5×5 block between $(200, 296)$ and $(204, 300)$ with the pixel at $(200, 300)$. Replace the pixels in the block to the left of this block by the pixel at $(200, 299)$. And so on. Effectively, you will replace the pixels with the specified window by replicating the pixels as needed.

   Alternatively, you can implement this by using the mouse and trackbar callbacks instead of specifying the command line parameters.

3. Write a smoothing spatial filter with standard averaging using the standard averaging method in a filter of specified size (make sure it is odd). Thus, a filter size of 7 will result in a smoothing kernel of size 7×7 with each coefficient as 1. Call the executable `sm_filter`. Implement using trackbar callback where the trackbar will allow the specification of kernel dimension.

4. Write a program to generate a circular kernel of specified size and resolution. The resolution will be specified as a power of 2. Thus, if the resolution is 1024, the coefficients of kernel will add to 1024, with each coefficient weighted by its distance from center. Obviously, the center pixel will have the highest weight, and corner pixels will have low weight, possibly 0. Apply this kernel to blur an image. You will actually be better off writing a convolution function that can apply the kernel to the image in this problem as well as the problem above. Take care of the edges by using reflected pixels. Call the executable `blur`.

5. Write a program to blur the borders of an image by making them pastel. The border should be such that it is all white at the edge and starts to fade n pixels from the border where n is a user specified parameter. The fading should happend in a linear fashion. If the pixel is valued at p, the pastel value of pixel at a distance d from border will be specified as

   $$p + \frac{n-d}{n}(\texttt{MaxRGB} - p)$$

   Specify n using a trackbar, making sure that $n < \min(h, w)$ where h and w are the height and width of the image, respectively.

6. Implement the morphological operators erosion and dilation and test them with a binary image generated by thresholding a grayscale image. Use the image mean for threshold value. Make sure that you define a structuring element in a text file in the following format

```
width height
c00 c10 c20 ...
c01 c11 c21 ...
 .   .   .
 .   .   .
 .   .   .
```

What to handin

Handin an electronic copy of all the sources, README, Makefile(s), and results. Create your programs in a directory called *username*.4 where *username* is your login name on `hoare`. Once you are done with everything, *remove the executables and object files*, and issue the following commands:

```
% cd
% chmod 755 .
% ~sanjiv/bin/handin cs5420 4
% chmod 700 .
```