McGill University
ECSE 415 - Intro to Computer Vision
Assignment #1 - Due 11:59pm - October 10th, 2017

Preface

The purpose of this assignment is to give you an introduction to some fundamental image processing tools. Please submit your assignment solutions electronically via the myCourse assignment dropbox. The solutions should be in **Jupyter Notebooks** format, see tutorial #2 for a reference. The complete assignment submission should be a zip file containing your Jupyter code (.ipynb) in addition to all output images. All code should be written without the use of openCV's high-level image processing methods, i.e. do not use *cv2.filter2D(), cv2.Blur(), cv2.Threshold*, etc. Attempt all parts of this assignment. The assignment is out of a total of **60 points**. The student is expected to write his/her own code. Assignments received up to 24 hours late will be penalized by 30%. Assignments received more than 24 hours late will not be marked.

Problem 1 - Image Negatives [10 points]

In image processing, reversing intensity levels of an image produces the equivalent of a photographic negative. This type of processing is particularly suited for enhancing images with white or gray details embedded in dark regions. For this question you will be creating your own image negative. The negative of an image with intensity levels in the range $[0, 255]$ is obtained by using the transformation

$$s = 255 - r$$

where s and r are the new and old intensity levels respectively. Using nested for-loops to visit each pixel individually, transform *BreastDigitalMammogram.jpg* [1] to an image negative. You should display the original input image in addition to the new negative image. The figures below demonstrate a sample output.

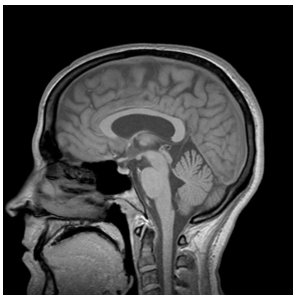

Fig. 1 - Input Image [1]

Fig. 2 - Sample Output

Problem 2 - Power-law (Gamma) Transformations [10 points]

Power-law transformations are used to help expand the values of pixels in a specific region while compressing the others. Supposing again you have r and s as the input's and output's respective

intensity value, power-law transformations have the basic form

$$s = cr^{\lambda}$$

where c and λ are positive constants. Power-law transformations with fractional values of λ map a narrow range of dark input values into a wider intensity range. The opposite is true for non-fractional values. Using nested for-loops, use gamma correction to enhance the image *LumbarSpineMRI.jpg* [2]. Provide the output images for $\lambda = 0.3, 0.6$.

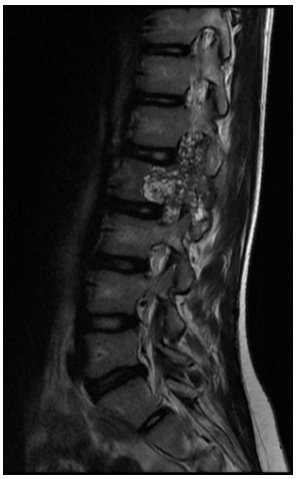

Fig. 3 - Input Image [2]

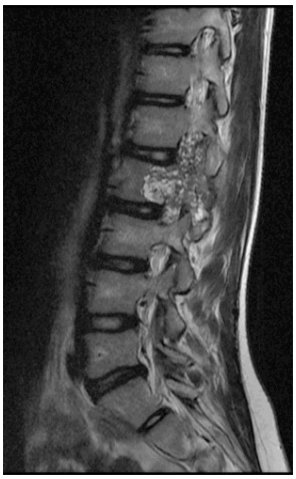

Fig. 4 - Sample Output

Problem 3 - Intensity Level Slicing [10 points]

In tasks such as image segmentation it is interesting to highlight or pull-outän interesting range of pixels. This task can be implemented in several ways, but most are variations of two basic themes. One approach is to remap all values of interest onto one value (white) and all other intensities onto another (black). This process creates an image commonly called a *Binary Image*. The second approach brightens the desired range of intensities but leaves all other intensity levels in the image unchanged. Using nested for-loops, implement and display these two approaches on *AroticAneurysmRadiograph.jpg* [3]. The intensities you should highlight are $120 <= t <= 240$.

Fig. 5 - Input Image [3]

Fig. 6 - Sample Output (Method 1)

Problem 4 - Smoothing Linear Spatial Filters [10 points]

Smoothing filters are used for tasks such are blurring and noise reduction. Blurring is generally used in preprocessing tasks, such as removal of small details from an image or bridging small gaps in lines or curves. The output of a smoothing, linear spatial filter is simply the average of the pixels contained in the neighborhood of the filter mask. These filters are called *averaging filters*. Using nested for-loops, implement the box-filters discussed in class on *HubbleSpaceTelescope.jpg* [4]. (Reference Lecture 4 - Slide 21 for averaging filter). You may need to zero-pad the input image prior to convolutions, i.e. adding zeros on the borders of the image to help with convolutions. (See Lecture 4 - Slides 13 to 20 for an example of zero-padding). Your averaging filter should be of size $15x15$. Once you have obtained the filtered image, you should then threshold the resultant to create a binary image. You should try threshold values of $t = 20, 70, 150$ and display the image you think is most optimal.

Fig. 7 - Input Image [4]

Fig. 8 - Sample Output (Binary)

Problem 5 - First-Order Derivatives [10 points]

First-order derivatives in image processing are represented using image gradients. Image gradients are extremely useful in image processing as they provide the user with gradient directions and strengths. For this question you will need to compute the edge strength given by the gradient magnitude. Using nested for-loops, find the gradient magnitude of *SatelliteImage.jpg* [5] using the Sobel filters discussed in class. (Reference Lecture 5 - Slide 11 for Sobel filters). You should first convert the image from RGB to grayscale. You may use openCV's *cv2.cvtColor* method. Once you have the magnitude image you should threshold by selecting an appropriate threshold value to create a binary image. Any threshold value that produces reasonable results will be adequate. You should display the gradient magnitude image in addition to the final binary image.

Fig. 9 - Input Image [5]

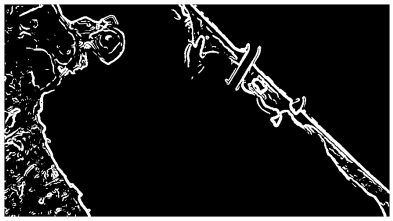

Fig. 10 - Sample Output (Binary)

Problem 6 - Sharpening Spatial Filter [10 points]

The objective of image sharpening is to highlight transitions in intensity. In problem 4 we saw that image blurring could be accomplished in the spatial domain by pixel averaging in a neighborhood. Because averaging is analogous to integration, one can conclude that sharpening can be accomplished by spatial differentiation. For this question, we consider the implementation of 2-D, second-order derivatives and their use for image sharpening. Seeing that the Laplacian is a derivative operator, its use highlights intensity discontinuities in an image and deemphasizes regions with slowly varying intensity levels. This will tend to produce images that have grayish edge lines and other discontinuities, all superimposed on a dark, featureless background. Background features can be *recovered* while still preserving the sharpening effect of the Laplacian simply by adding the Laplacian image to the original. Thus, the basic way in which we use the Laplacian for image sharpening is:

$$g(x, y) = f(x, y) + c[\nabla^2 f(x, y)]$$

where $f(x, y)$ and $g(x, y)$ are the input and sharpened images, respectively. Using nested for-loops, apply image sharpening to *Moon.jpg* [6]. You should use the Laplacian masks as discussed in class. (Reference Lecture 5 - Slide 24 for Laplacian filter). You should try different values of c and pick a value which you think is best. Finally, display the resultant image of applying the Laplacian mask in addition to the final sharpened image.

Fig. 11 - Input Image [6]

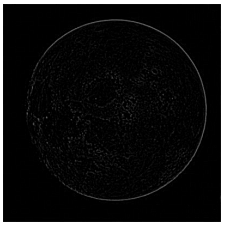

Fig. 12 - Laplacian Image

Fig. 13 - Sharpened Image

References

1. http://www.3timaging.com/
2. https://www.cedars-sinai.edu/Patients/Programs-and-Services/Imaging-Center/For-Patients/Exams-by-Procedure/MRI/MRI-Spine—Lumbar-or-Thoracic-.aspx
3. https://radiopaedia.org/cases/abdominal-aortic-aneurysm-mra-mri
4. https://www.space.com/15235-hubble-space-telescope-latest-photos.html
5. http://spacenews.com/retired-boeing-satellite-exec-to-head-eo-startup-hera-systems/
6. http://science.howstuffworks.com/moon.htm