CSCI 55700
Image Processing and Computer Vision
Spring 2024
Project 2
Low level image processing
Due Date: March 17, 2024, 11:59pm.

Introduction

The goal of the project is to get you started working with images and to implement (on your own) some point and neighborhood processing functions on images. You will have approximately two weeks to implement, test, and write a report on the results for this project.

Project Description

In this project, you will implement a few of simple image point transformation and spatial filtering operations:

- Implement a histogram equalization program that flattens the histogram of the input image as described in lecture by creating a mapping function $c(I)$ based on the histogram of the original image. Then apply this point transformation function to your image:

$$I_{new}(r,c) = c(I(r,c))$$

- Implement a function that does a log mapping of the input image (thus enhancing the detail in dark regions).

$$I_{new}(r,c) = \log(I(r,c) + 1)$$

  We add one to the original pixel value because $\log(0)$ is ill defined in case the pixel value is 0. If $I(r,c) = 0$, then $I_{new}(r,c) = \log(0+1) = \log(1) = 0$.

- Write a function that will take an input angle θ and produce an output image which is the input image rotated by θ around the image center. Normally θ is positive if the rotation is counter-clockwise (CCW) and negative otherwise. If the pixels in the output image do not correspond to a rotated pixel of the input image, then set their value to 0 (black). [**Hint**: The recommended way to do this is to start with the pixel (i,j) of the *output image* and figure out what input image pixel its value should come from. If you start with output image and get the pixel value from the corresponding source image, then you do not have holes in the output image due to rounding. Because you are

scanning the output image pixels, you will have a value for all valid pixels coming from somewhere in the input image.]

- Implement the Gaussian averaging filter. You will have to design the filter mask (integer approximation or floating point samples; it's up to you). You may want to have σ of the Gaussian filter as an input parameter that you can easily vary. This will let you experiment with different width Gaussian filters. The filter mask size may depend on the value of σ.
- Implement the median filter. Use 3×3 neighborhood.

Note that the range of your output image may change in some of these cases, so you may have to renormalize the range to [0,255]. Also note that both in the log transformation case and the Gaussian averaging case, you will be dealing with different data types (byte vs. float etc.), so you need to make sure you take care of this properly.

Run your programs on the following data from the intensity image directory image_data directory:

1. auto
2. building
3. tire
4. child
5. ct_scan

In the case of Gaussian filtering and median filtering, you may experiment by adding various types and levels of noise to the image and see how the two filtering approaches perform. The types of noise can be Gaussian noise with 0 mean and variance σ_n^2. Or it could be salt-and-pepper noise. You can experiment with different Gaussian filters with different σ's and see how well it cleans the image. You can use the standard tools in Matlab to add the different types and levels of noise to the image. You need not implement this yourself. But you need to implement the image processing methods yourself. The Matlab function to add noise to an image is `imnoise`. See http://www.mathworks.com/help/images/ref/imnoise.html for its documentation.

You can compare the results of your implementation with what a more standard package generates (e.g., Matlab image processing tool, openCV, or any of the GUI based tools such as ImageJ and GIMP).

Expected Implementation

You are expected to implement the processing required in this project yourself and not just use existing implementation in a library. Remember that the goal of these exercises is for you to learn the techniques yourself and not just use what someone else has already

implemented. So, in this project, you are to implement with your own code the histogram equalization, log transform, rotation of an image, Gaussian filtering, and median filtering. You may use existing library functions for image IO (i.e., reading and writing images from/to files), adding noise of various characteristics (Gaussien, salt and pepper noise etc). You may also use existing library functions of what you implemented in order to compare the results of your own implementations. For example, in order to compare how well your implementation of Gaussian filtering is doing, you can filter the image with an implementation by a function a library provides and compare the two results.

What to hand in

You will hand in a report and source code for this project. You can put everything except the report into a zipped file and upload it as one file. You should upload the report separately as a document (word or pdf). The report should contain the following:

1. A short introduction and a brief description of the theory and the algorithms you are implementing. (i.e., the histogram equalization, log transformation, rotation, etc.).
2. The results of running your implementations on the images given above. You are welcome to test other images as well.
3. An analysis/interpretation of your results:
    a. What does each function do?
    b. How does it affect the image?
    c. Where it may be useful and where not?
    d. If you have made the comparison, how does it compare with standard tools available?

In your report, if you are showing results (result images), you should include the following clearly labeled.

- Input image (name of the file).
- Result of your program (in the case of rotation by θ or Gaussian filtering with parameter σ, you need to have the parameter's value clearly labeled for each image). If you are showing added levels of noise, you need to specify the noise parameters as well.
- Result of running the same processes in standard image processing tools on the same input images.
- A short discussion and any conclusions you may draw. Anything else you can think of. For example, you may want to compute the difference between the results of your

program and the standard software in some quantitative way. For example, for this to work, you have to be sure that the resulting ranges are comparable.

Grading Rubric

Component	Percentage
Report	20%
Histogram Equalization	15%
Log transform	10%
Image rotation	15%
Gaussian filtering	15%
Median filtering	15%
Noise experiments	10%
Total	**100%**

Have Fun!