Image Processing

Homework 2

Fall 2017

Kynan Justis

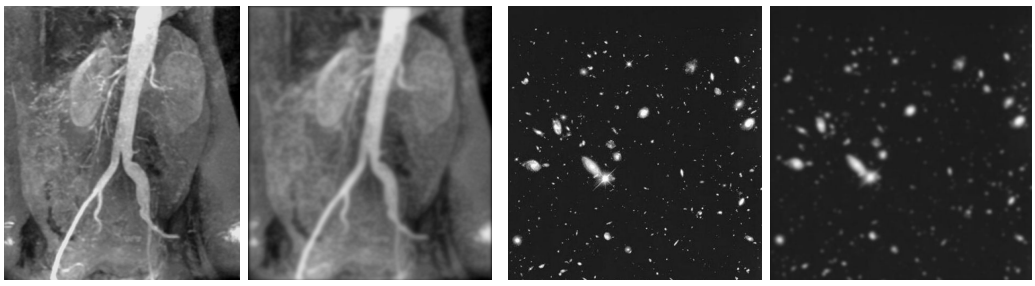
10/23/2017

Overview

There were several parts to this assignment. The first part focuses on implementing various smoothing algorithms. Mean filtering, median filtering, and gaussian filtering are all covered to accomplish this. In the second part, the focus is to implement a sharpening filter. Finally, the third part is to implement a gaussian pyramid for square images. While this requires the use of downsampling, an implementation for upsampling is also required to go along with the pyramid image. The results of these implementations will be covered in the same order.

Part 1 - Mean filter

The implementation for mean filtering is fairly straightforward. For any number of iterations, the main loop involves first padding the source image with a border of zeros. From there, the first non-border pixel is accessed. Since the border is present, it's possible to look at an entire 3x3 slice around that pixel. This slice is passed to *AverageMatrix()* where all of the pixel values are averaged. The average from this procedure is then returned and placed back into the center of the slice that was originally created. By repeating this series of steps for every pixel in the source image, the resulting image is the average from each slice, which gives it a smoothed or blurred effect.

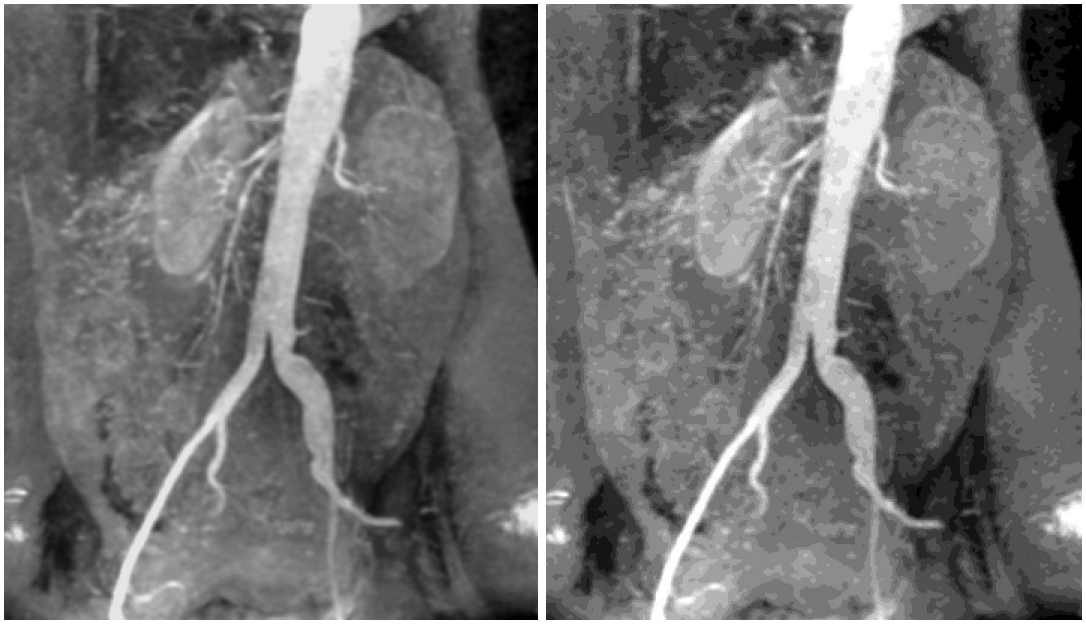


Part 1 - Gaussian filter

The implementation for the Gaussian filter is quite a bit similar to the averaging implementation. Like before, the same process of iterating over the source pixels and creating slices is carried out. However, the calculations carried out within the slices are different. Each new pixel is calculated with the following equation:

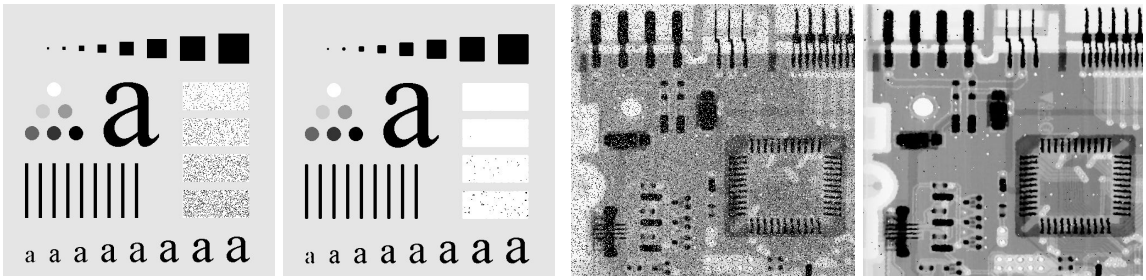
$$N(x, y) = \exp(-\frac{x^2 + y^2}{2\sigma^2})$$

These new pixels are then written to the final output matrix and returned. While the filter doesn't do terribly well with stray noise, it does smooth out some of the detail.



Part 1 - Median filter

Applying the median filter is also very similar to the mean and Gaussian filters. The same method of iterating through the source image pixel is used with 3x3 slices being created each time. From this slice, the median value is calculated. This is done by first re-shaping the 2D slice matrix into a 1D matrix. Then the 1D matrix is sorted. Using the length of the 1D matrix, the median value can be calculated by dividing the number of elements by two. If there are an even number of elements present, then the median is the average of the middle two elements. Otherwise, if there are an odd number of elements, then the value at the middle-most index is returned. This return value is placed at the center of the slice, which then gets written back into the source image. Unlike the Gaussian filter, stray noise is handled pretty well.



Part 2 - Sharpening filter

For the sharpening filter, a mean filter pass is first applied to the source image. The image from this pass is then stored in a separate matrix, which leaves two matrices in memory: the source image and the mean filtered image. A third matrix is then made by subtracting each pixel value in the mean filtered image from the source image to result in the difference between the two. Then, by adding the pixel values in this third image back into the pixel values in the original source image, a sharpened image is created. One problem, however, is that the addition from the pixel values can result in pixel value overflow. To combat this, basic clamping is applied to ensure that every pixel falls within the range 0-255 without wrapping.

Part 3 - Gaussian pyramid

The last part is to create a Gaussian pyramid from a grayscale, square-dimensioned image. Also, examples of upscaling should be provided as well. For this implementation, the Gaussian pyramid is displayed across the top. To save space, upscaled versions of each downsample are displayed across the bottom. As for actually creating the Gaussian pyramid, the method isn't too complex. First, every even row and column is removed from the source image and the in a matrix 1/4th the size of the source. Then, the downsampled matrix is given a mean filter pass and finally written back into the source image in the following pattern.

Note

As with the submission for Assignment 1, all of the results for the various filters and source images can be found here in the output folder: https://github.com/kfjustis/ip2 If you want to build the project with CMake, assuming there is a valid installation of OpenCV and CMake, download the project, make a folder called "build" inside. Then, change the directory into the build folder, run "cmake .." and then "make". The binary will be in "build/src".