

BME 51:185 — Homework 2

DUE: September 20, 2004 at start of class

In this assignment you will look at two common image enhancement operations, the neighborhood averaging filter and histogram equalization.

Use the MILP environment for this assignment. Be sure your program handles common error conditions such as no image loaded, wrong image type loaded, and bad input parameter values by detecting these conditions and displaying an error dialog.

Because this problem requires that you devise an algorithm to solve a problem, part of your grade will be based on how well you comment the source code that describes the algorithm. If I can't understand your source code and comments, you will not receive full credit, even if the algorithm works.

How to turn in your assignment: All problems must be completed using the MILP framework discussed in class. Create a directory on your computer called `homework2`. Inside of `homework2`, put a copy of your `operation.cpp` and `operation.h` files, plus any other files that you added and are necessary to compile the project. Include a HTML file called `index.html` with a brief discussion of each part and example images showing your results. Please make sure all of the links in your HTML document work outside of the directory you created them in (use relative links)!

When you are finished with the homework, use `tar` or `zip` to turn your directory structure into a single archive. Submit the archive before the deadline using the `grade` program. **No late homework will be accepted!** The timestamp from the `grade` program will be used to determine which assignments are on time.

1. “Boxcar” neighborhood averaging filters. As we saw in the first homework, the sliding window filter can be efficiently implemented using the “boxcar” approach rather than the brute-force “nested for loops” approach. For this problem you will implement a square neighborhood averaging filter using the boxcar method. Make your function run as operation 01 in the MILP framework.

Your program should request the desired neighborhood window size from the user through a dialog box. Force the user to use odd sizes only, with the minimum size being at least 3 and the maximum size limited to one half of the smallest image dimension (truncated down to the nearest odd number). In your writeup document how you handle image border processing.

Demonstrate your results on any two of the noisy images in http://css.engineering.uiowa.edu/~bme185/P_185/noisy/.

2. In this part we will look at the effects of filtering on the noise level in the image. You may do this part in MATLAB.

Start with your boxcar algorithm and the image from http://css.engineering.uiowa.edu/~bme185/P_185/noisy/ct_heart_n50.tif. Apply your boxcar filter for some window size k , and then compute a new image d , which is the difference between the filtered image and the original, noise free image found in http://css.engineering.uiowa.edu/~bme185/P_185/noisy/ct_heart.tif. Compute the standard deviation $\sigma(k)$ of the pixels in d to estimate the noise level after filtering with window size k .

Plot $\sigma(k)$ vs. k for $k = 0, 3, \dots, 15$ (let $k = 0$ represent the no-filtered condition). Include the graph in your HTML document. Comment briefly on the graph and on the quality of the filtered images.

3. Histogram equalization: As described in class, histogram equalization can be used to improve the overall contrast of an image. However, since image pixel statistics are different in different parts of the image, global processing may not always yield the best results. One approach to addressing this problem is to use *local* histogram equalization (LHE).

LHE applies histogram equalization within a sliding window that is swept across the image in a manner similar to a convolutional mask (see GW section 3.3.3). For this problem, you should decide how to and handle boundary effects and document this decision as part of your solution.

Develop a LHE function for MILP. Make your function run as operation 02 in the MILP framework. Your program should request the desired neighborhood window size from the user through a dialog box. Force the user to use odd sizes only, with the minimum size being at least 3 and the maximum size limited to one half of the smallest image dimension (truncated down to the nearest odd number).

Demonstrate your results on the images in http://css.engineering.uiowa.edu/~bme185/P_185/lhe. Briefly examine the effects of changing LHE window size on the quality of the processed images.