Project 1 Image Processing

Name: Juan Minardi
ID: F11015122

Grayscale

Application of formula provided on slides for the intensity in each pixel

Uniform Quantization

As instructed by the slides, shifting reduces the amount of colors in an image uniformly. Because 256 colors were required, the shifting was done an equal amount of times to the amount of bits to be reduced on the channel.

Populosity Quantization

Since populosity requires frequency counting and sorting. I searched for structures that could perform those functions. After looking at the structure documentations, I decided to use a two dimensional vector which element frequency is counted on a map. The map is copied to a vector pair for sorting and the 256 most common colors are used for color approximation with euclidean distance.

“https://www.geeksforgeeks.org/map-associative-containers-the-c-standard-template-library-stl/”
“https://www.educative.io/answers/how-to-sort-a-map-by-value-in-cpp”

Dither Threshold, Random and Cluster

The implementation of this three dithers are similar. First, grayscale is applied. Dither threshold uses 0.5 as the value in range [0.0, 1.0], to determine if the intensity of the pixel will be 0 or 255. Dither random adds a random value to intensity from the range [-0.2, 0.2] before doing threshold dithering. Finally, cluster dithering consists on using a threshold box that will be moved throughout the image to do threshold dithering. Only reference used were the slides provided.

Dither Brightness

I couldn’t understand at first how to keep the average brightness constant, until I saw a professors explanation in a discussion. In which he explains that if you take the ith element based on brightness, the 1st to the ith element will all be black. So if the ith element represents the percentage of pixels that has to be black to maintain the average brightness, then you can use it as a threshold. The discussion has only written explanations with no coding.

“https://research.cs.wisc.edu/graphics/Courses/559-f2002/projects/project1/project-1-faq.html#bright”

Dither Floyd-Steinberg and Color

Dither floyd consists on doing threshold, calculating the error and transfer portions of the error to its neighbors that are yet to dither. Dither color is pretty similar. Unlike Dither-fs, color doesn’t apply grayscale at start. Instead in dither color, dither-fs is applied independently to each of the three channels. I created a function for finding closest color that works for both, taking the old pixel or channel intensity, the color palette and the size of the color palette as parameters, doing euclidean distance to each color in the palette and returning the closest color.No references were needed as the slides were clear enough and a pseudo code was provided.

Filters

For the filters, I created a function that could handle all cases except gaussian-n which used a modified version of the function using binomial coefficients to create an arbitrary sized filter. The filter function takes a 5x5 matrix, a float for filter division and an unsigned char dynamic array which could be either data or a copy of it that will be modified. The discussion mentioned before clarified that the box always has to take into account the original image, so a copy of data has to be made before starting.

https://research.cs.wisc.edu/graphics/Courses/559-f2002/projects/project1/project-1-faq.html#filter

Half Size

I took the Filter function and made a modified copy called Reconstruct which takes a 3x3 matrix instead of a 5x5 matrix. The looping is done by half the width and height instead. Finally the values of width and height are reduced by half. The slides plus the references previously mentioned on filters were enough information.