ECE 160: Multimedia Systems
Homework 1
Due: April 9, in class

Section 1. Short Answer (10 pts each)

For each question, answer them in your homework report which you will submit as a PDF file online on Gauchospace. This applies to Section 2 below as well.

1. Identify three novel multimedia applications. Discuss why you think these are novel and their potential impact.
2. Color inkjet printers use the CMYK model. When the color ink cyan is sprayed onto a sheet of white paper,
   (a) why does it look cyan under daylight?
   (b) what color would it appear to be under a blue light. Why?

Section 2. Programming (40 pts each)

For each programming question, submit the written answers in the same PDF report as above, and also submit your source code and other result files on Gauchospace as indicated below.

3. **Median-cut Algorithm**: The median-cut algorithm is a popular algorithm for use in color quantization to build a color lookup table. It was originally published in 1980 as part of the Bachelors of Science thesis for Paul Heckbert. His description is reproduced:

   > It starts by finding the minimum and maximum red, green, and blue among all N colors (N usually 307200).
   >
   > Iteration step: Split a box. For each box, the minimum and maximum value of each component is found. The largest of these three determines the dominant dimension - the dimension along which we would want to split the box. The box with the largest dominant dimension is found, and that box is split in two. The split point is the median point - the plane which divides the box into two halves so that equal numbers of colors are on each side. The colors within the box are segregated into two groups depending on which half of the box they fall.
   >
   > The above step is repeated until the desired number of boxes is generated. Then the representative for that cell (the quantization output) is computed by averaging the colors contained. The list of reconstruction levels is our colormap.

   (a) Implement the median cut algorithm as described. Use the function prototype *color_table = median_cut(input_image_filename)* where *input_image_filename* is a filename for an image and *color_table* is the table of colors generated. Submit the code on Gauchospace.
   (b) For the color image, generate at list of 16 based on the median cut algorithm. List the 16 colors in the homework report.
   (c) For the 16 colors generated above, display the image by replacing each color with its closest match. Print the image in your homework. Submit to Gauchospace the color quantized image as a PNG.
   (d) For the same color image, create a color lookup table of 256 colors, and print the image based on 256 colors in your homework. Submit to Gauchospace the color quantized image as a PNG.

4. **Floyd-Steinberg Dithering**: In class we covered two methods of dithering, pattern dithering and ordered dithering. One of the best performing methods is the Floyd-Steinberg Dithering algorithm. This algorithm achieves dithering through error diffusion, meaning it spreads the quantization error (real value - quantized value) at the current pixel to neighboring pixels.

Floyd-Steinberg Dither scans through the image replacing each pixel with a quantized value, and then distributing the quantization error to the pixels have that yet to be scanned according to the following matrix, where $*$ is the current pixel:

$$\begin{bmatrix} & * & \frac{7}{16} \\ \frac{3}{16} & \frac{5}{16} & \frac{1}{16} \end{bmatrix}$$

Floyd-Steinberg Dithering pseudocode:

for y row from top to bottom **do**
  for x column from left to right **do**
    oldpixel = Image[x][y]
    newpixel = find_closest_palette_color(oldpixel)
    Image[x][y] = newpixel
    quant_error = oldpixel - newpixel
    Image[$x+1$][y] = Image[$x+1$][y] + quant_error $\times$ 7/16
    Image[$x-1$][$y+1$] = Image[$x-1$][$y+1$] + quant_error $\times$ 3/16
    Image[x][$y+1$] = Image[x][$y+1$] + quant_error $\times$ 5/16
    Image[$x+1$][$y+1$] = Image[$x+1$][$y+1$] + quant_error $\times$ 1/16
  end for
end for

(a) Implement Floyd-Steinberg Dithering algorithm as described above for converting grayscale images to black and white. Use the following prototype (i.e. function call): *output_image = floyd_steinberg_dither(input_image_filename)*, where *input_image_filename* is a filename for an image and *output_image* is a matrix representing the dithered output. Submit the code on Gauchospace.

(b) Using the included grayscale image, run the the dithering algorithm to transform it into a 1-bit grayscale image. In your homework, include the original image and dithered image, side by side. Submit to Gauchospace the dithered image as a PNG.

(c) The distribution matrix has been designed with the specific property that so that it would produce a checkerboard pattern in areas with intensity of 1/2 (or 128 in a 256 image). Implement following diffusion matrix:

$$\begin{bmatrix} & * & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}$$

Use the prototype, *output_image = floyd_steinberg_dither_uniform_distribution(input_image_filename)*. (Reuse code/helper function as reasonable with the above function.)

(d) Run the uniform distribution on the included grayscale image. In your homework, include the original and dithered image, and describe, qualitatively, how the result is different than the standard dithering method. Submit to Gauchospace the dithered image as a PNG.