CS 559: Computer Graphics

Homework 2

This homework must be done individually. Submission date is Tuesday, February 20, 2001, in class.

Question 1: The RGB color space is not perceptually uniform, while the LUV space (defined below) is approximately uniform. Hence, one way to decide whether two pairs of colors in RGB space, say a, b and c, d, are separated by the same perceptual distance is to first convert all the colors into LUV space then compute their relative distances there using a standard distance metric. Use this idea to find 9 shades of RGB grey that are perceptually roughly equally separated. One shade should be black $grey_0 = (0, 0, 0)$ and one should be white $grey_8 = (1, 1, 1)$. Give the other shades $g_i = (r_i, g_i, b_i), 1 \leq i \leq 7$ to two significant figures. Plot the grey intensity as a function of i, to see how far from perceptually uniform RGB space really is.

To get from RGB to XYZ, use the following matrix:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0.73467 & 0.27376 & 0.16658 \\ 0.26533 & 0.71741 & 0.00886 \\ 0.00000 & 0.00883 & 0.82456 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} \tag{1}$$

LUV coordinates, (L^*, u^*, v^*) are computed in several steps. First compute (X_n, Y_n, Z_n) which are the XYZ coordinates of white. Then compute the following four values:

$$u' = \frac{4X}{X + 15Y + 3Z} \tag{2}$$

$$v' = \frac{9Y}{X + 15Y + 3Z} \tag{3}$$

$$u'_n = \frac{4X_n}{X_n + 15Y_n + 3Z_n} \tag{4}$$

$$v'_n = \frac{9Y_n}{X_n + 15Y_n + 3Z_n} \tag{5}$$

Finally, compute:

$$L^* = 116 \left(\frac{Y}{Y_n} \right)^{\frac{1}{3}} - 16 \tag{6}$$

$$u^* = 13L^*(u' - u'_n) \tag{7}$$

$$v^* = 13L^*(v' - v'_n) \tag{8}$$

When $Y/Y_n < 0.01$, $L^* = 903.3Y/Y_n$, rather than the equation above. Note that when $r = g = b$, $u' = u'_n$ and $v' = v'_n$ and hence $u^* = v^* = 0$. This means that you only need to compute L^* and you can use the difference between the L^* values for two colors as their perceptual distance. You might like to write a program to do the color conversions.

Question 2: Consider an image format that uses indexed color (it stores a table of colors and then each pixel is an index into the table). Let w and h be the width and height of the image, let k be the number of bits for the index at each pixel, and assume that the table uses 24 bits for each color it stores.

1. Write an expression for the number of bits required to store the color table and pixel data (in total).

2. Assume that the total number of pixels is $n = wh$ and that $k = 8$. Consider indexed color as a compression scheme that takes an image requiring $24n$ bits and compresses it, through color quantization, to one requiring fewer bits. Plot a graph of the compression rate as a function of n, starting at $n = 10000$ and going up to $n = 1000000$. The compression rate is the compressed file size divided by the original file size. What is the limit compression rate as $n \to \infty$?

3. Now fix the number of pixels at $n = 1000000$, and allow k to vary. Show the graph of compression ratio as a function of k, for k between 1 and 24. At what value of k is no compression achieved (to one decimal place)? At what value of k is 0.5 compression achieved?

4. What can you say about the relative effectiveness of indexed color as a compression scheme in cases where you want only a small amount of compression (reducing the file size by less than a half) and in cases where high compression is desired (compression ratios around 0.1)? Consider in particular the probable loss of image quality at various compression ratios.

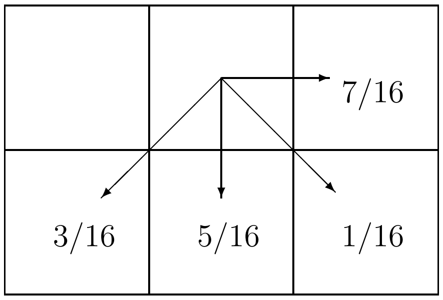
Question 3: The variant of Floyd-Steinberg dithering given in class distributed the error in one pixel to three of its as yet unvisited neighbors.

1. What happens if you run the version given in class on an image with constant intensity of 0.5? What are the artifacts?

2. One possibility is to only distribute error to the pixel ahead of the current one on the same row. Why is this a bad idea? (Hint: Consider what would happen with the vertically striped image used in the demo program, then extend to smoothly varying intensity gradients).

3. Another possibility is to only distribute error to the pixels below the current one. Why is this a bad idea?

4. An improved variant (the one most commonly used) distributes error to the four unvisited neighbors, as shown below. It also scans from left to right along one row, then right to left along the next, and so on. Why might these improvements produce better dithered images?


7/16
3/16
5/16
1/16