CS 417 Homework 1

out: Friday, January 24, 2003
due: Friday, January 31, 2003

Problem 1: Gamma correction.

A.

In class we discussed how to *gamma correct* an image I to produce a new image I', so that when we display I' on the monitor the light intensity $L(x, y)$ on the monitor will be proportional to $I(x, y)$. We assumed that the monitor produces exactly zero light when we send the number 0 to it.

In real life, any kind of display reflects some light when it is turned off. For better-quality displays the amount is lower, but there will always be some. Assuming that the light emitted by the display follows the usual power law approximation we have:

$$L(x, y) = k + I'(x, y)^{\gamma} \tag{*}$$

1. What formula do we need to use to correct an image for this display? That is, give an expression for I' as a function of I such that when I' is sent to the display L will be proportional to I across as much of the intensity range as possible. Both I and I' range from 0 to 1.
2. Briefly, what is the visual effect of this operation on the image?

B.

On the course web site you will find a simple Java applet that displays a region of alternating black and white lines next to a gray patch. A slider lets you set the pixel value of the gray patch. Using this applet you can determine the display gamma on your computer.[1] If you stand back from the display, the alternating bars will average out to 50% gray (that is, $(L(0) + L(1))/2$, whereas the gray patch will show you $L(n/255)$ for your chosen value of n. By observing what value of n causes the gray bar to blend into the stripes (when you stand far enough away that you can't see the stripes) you can deduce γ.

[1]Or at least the display gamma as seen by a user of the Java graphics API, which may or may not be trying to do some gamma correction on your system.

1. What type of display are you using (CRT, LCD, ...)?
2. What pixel value matches for your display? If you have trouble answering this, explain why and choose a value that seems typical.
3. Assuming that the equation (*) above accurately models your display, what is the display gamma for your system? Show the derivation of your answer.

Problem 2: Image formats.

This problem has to do with converting images from one representation to another. Assume we have stored in memory a representation of a grayscale image I as an m by n 8-bit raster image I_8; that is, $I_8[i, j]$ is a number between 0 and 255, for $i < m$ and $j < n$.

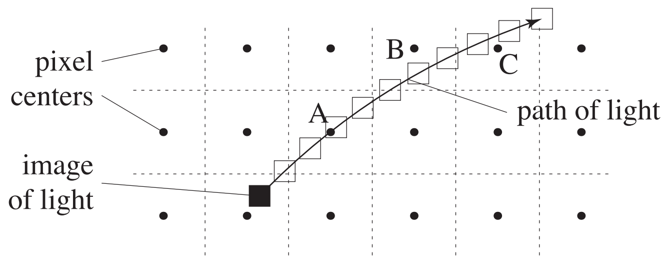
1. How much memory is occupied by I_8?
2. Give pseudocode for the simplest way you can think of to compute I_4, a representation of the same image I but with 4-bit precision. How much memory is occupied by I_4? What is the difference between I_8 and I_4, visually?
3. Give pseudocode for the simplest way you can think of to make I_{12}, a representation of the same image I but with 12-bit precision. How much memory is occupied by I_{12}? What is the difference between I_8 and I_{12}, visually?
4. Now we want to compute J_8, an 8-bit representation of I that has $m/2$ by $n/2$ pixels. Give pseudocode for the simplest way you can think of to compute J_8 from I_8. How much memory is occupied by J_8?
5. Assume we have lost the original data for I_8 and have kept only J_8. Give pseudocode for the simplest way you can think of to compute I_8^{new} from J_8. What is the difference between I_8 and I_8^{new}, visually?
6. For each of the previous parts that resulted in degraded image quality, name a better technique that would not degrade image quality as much. Describe how the images would differ visually with these better techniques. (You don't need to give pseudocode for these.)

Assume that for images of any precision the maximum representable pixel value always maps to the brightest available white on the display.

Problem 3: Antialiasing.

You are working on the graphics code for a flight simulator, and your task is to choose an antialiasing filter. During nighttime landings, the runway lights, which appear from the pilot's point of view as very tiny dots, are the most important objects on the simulator's raster display—so the simulator's rendering of sub-pixel objects is important.

Assume we can approximate the shape of one of the lights as a square, and on a particular simulation run the image of this light takes the path shown here across the pixel grid:



The runway light moves from the start to the end of the path shown in the drawing between times $t = 0$ and $t = 1$, and it does not rotate (it stays axis aligned). The width of the square is $1/4$ of the pixel spacing, and the intensity is uniformly equal to 1 over its area.

For the pixels marked A, B, and C, sketch graphs of pixel value against t for the following three cases (you can draw three curves on one graph for each case). The graphs need not be precisely to scale, but the overall shape should be right (in particular, make sure that the presence or absence of discontinuities or corners is correct).

1. Point sampling–the sampling filter is an impulse function at the pixel location.
2. Box filtering—the sampling filter is a square with width equal to the pixel spacing.
3. Gaussian filtering—the sampling filter is a Gaussian with standard deviation equal to half the pixel spacing.

Note that pixel filters should always be normalized.

Now sketch the same graphs again for the case where the distance to the light has increased by a factor of 10, so that the width of its image decreases by a factor of 10.

Briefly, what is the overall visual effect of applying each of these filters to a very small dot? (Answer in 5 or fewer words for each one.)