CS 465 Solution 1

(revised September 20, 2004)

Problem 1: Image formats

1. Image Size
   (a) Float RGB Image: $1280 * 1024 \frac{\text{pixel}}{\text{image}} * 3 \frac{\text{channel}}{\text{pixel}} * 4 \frac{\text{byte}}{\text{channel}} = 15\text{MB}$
   (b) Int RGB Image: $1280 * 1024 \frac{\text{pixel}}{\text{image}} * 3 \frac{\text{channel}}{\text{pixel}} * 1 \frac{\text{byte}}{\text{channel}} = 3.75\text{MB}$
   (c) 16-bit Grayscale: $1280 * 1024 \frac{\text{pixel}}{\text{image}} * 1 \frac{\text{channel}}{\text{pixel}} * 2 \frac{\text{byte}}{\text{channel}} = 2.5\text{MB}$
   (d) 4-bit Grayscale: $1280 * 1024 \frac{\text{pixel}}{\text{image}} * 1 \frac{\text{channel}}{\text{pixel}} * 0.5 \frac{\text{byte}}{\text{channel}} = 640\text{KB}$
   (e) 1-bit Bitmap: $1280 * 1024 \frac{\text{pixel}}{\text{image}} * 1 \frac{\text{channel}}{\text{pixel}} * 1 \frac{\text{bit}}{\text{channel}} = 160\text{KB}$

   Note: 1 MB = 1024 KB = 1024*1024 B = 1024*1024*8 bits
2. Image processing required
   (a) Double RGB to Int RGB: Range conversion is necessary - the integer representation runs from 0 to 255, where 255 maps to full intensity. The double representation specifies that 1.0 maps to full intensity, so we have to scale before rounding.
   (b) Double RGB to 16-Bit Grayscale: Some sort of weighting/averaging mechanism is needed. As discussed in lecture, evenly weighting the color channels is not as preferable as other weights. The one used in lecture is grey = 0.2*R + 0.7*G + 0.1*B. A scaling is still necessary, but this time to 2^{16}-1.
   (c) Double RGB to 4-Bit Grayscale: This is pretty much the same process as described above, but using $2^4 - 1$ as the range scale. Dithering might be helpful here, but is not necessary.
   (d) Double RGB to 1-Bit Bitmap: After obtaining an intensity estimate as in part b, a cut off function is applied to see if each value is on or off. After that, the only reasonable way to store a 1-bit bitmap is to dither.
3. Image fidelity lost
   (a) Double to Float: Nothing is lost. There is a slight loss in range, but both doubles and floats cover ranges far beyond the real amount of light ever seen by human eyes.
   (b) Float to Int: Dynamic Range is lost. There is a practical upper bound now on the value that a simple 8-bit value can represent. The range is usually listed as being [0, 255], whereas the range for a floating point value is many orders of magnitude higher.

(c) Int to 16-bit Grayscale: Color is lost. Because there are more bits per channel in the grayscale image, color is the only information that is lost.

(d) 16 to 4-bit Grayscale: Tonal resolution is lost. We have gone from having 2^{16} shades to only 16 shades. At this point, banding should be visible in the image.

(e) 4-bit Grayscale to 1-bit Bitmap: Spatial resolution is lost. The only reasonable way to store a 1-bit bitmap is with dithering, which tends to blur the previously distinct boundaries of objects.

4. To gamma correct or not to gamma correct Gamma correction is important where we need to squeeze better quality out of fewer shades of color available to us. Note that the need for gamma correction is more visible in areas of lower intensity. This is because the human visual system sees percent change, not absolute change. As a result, a fine gradation between intensities is needed where the absolute intensity is lower.

(a) Float RGB Image: Gamma correction is completely unnecessary here as floating point values are able to get arbitrarily close to 0 (practically speaking).

(b) Int RGB Image: Gamma correction is useful here. While 256 intensities might not be distinguishable to the human eye, we do notice an improvement when these intensities are distributed non-linearly.

(c) 16-bit Grayscale: Gamma correction is not necessary. 16 bits allows for a much finer gradation than the human visual system can detect.

(d) 4-bit Grayscale: Gamma correction is very necessary in this case. 16 colors is very noticeable to the human eye, and there is a noticeable change in image quality when these colors are mapped in a non-linear fashion.

(e) 1-bit Bitmap: Doesn't apply. With only two values, the curve in between the two values can be completely arbitrary. The only values that will be used are 0 and 1.