Solutions to Optional Homework (Lecture 2)

Problem 3.2

(a) $$s = T(r) = \frac{1}{1 + (m/r)^E} \tag{1}$$

(b) See Fig. P3.2.

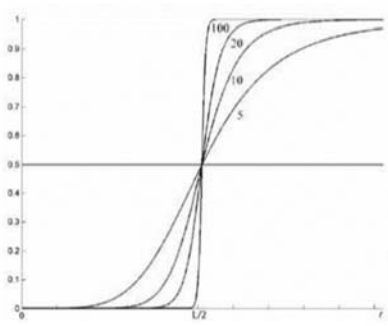

Figure P3.2

(c) We want s to be 0 for $r < m$, and s to be 1 for values $r > m$. When $r = m$, $s = 1/2$. But, because the values of r are integers, the behavior we want is

$$s = T(r) = \begin{cases} 0.0 & \text{when } r \le m-1 \\ 0.5 & \text{when } r = m \\ 1.0 & \text{when } r \ge m+1 \end{cases} \tag{2}$$

The question in the problem statement is to find the smallest value of E that will make the threshold behave as in the equation above. When $r = m$, we see from (a) that $s = 0.5$, regardless of the value of E. If C is the smallest positive number representable in the computer, and keeping in mind that s is positive, then any value of s less than $C/2$ will be called 0 by the computer. To find the smallest value of E for which this happens, we solve the following equation for E, using the given value $m = 128$ for any given value of C:

$$\frac{1}{1 + [m/(m-1)]^E} < C/2$$

Because the function is symmetric about m, the resulting value of E will yield $s = 1$ for $r \ge m+1$. Another solution is to implement Eq. (2) directly, but the problem statement specifically calls for using Eq. (1), in which case we need to know the value of E.

Problem 3.4

(a) Converting decimal numbers to binary automatically creates the bit planes. For example, the values of the pixels in an 8-bit image range from 0 to 255. If you converted those numbers to binary each value would be converted to 8 binary bits. The result would thus be a cube of size M-by-N-by-8. But this is exactly what Fig. 3.13 is. In other words, you can look at the conversion to binary of an n-bit image as being represented by a stack of n binary planes, descending from most significant to least significant. The trick in implementing this approach is being able to extract the individual planes. The approach would vary depending on the computer language used. For instance, in MATLAB we would use function **dec2bin** to covert an image with decimal values to a binary representation. After that, some additional manipulation of the resulting array would be needed to extract the individual bit planes.

Problem 3.5

(a) The number of pixels having different intensity level values would decrease, thus causing the count in the lower-order bins in the histogram to decrease. Because the number of pixels would not change, this would cause the numebr of counts of the higher order bins to increase, resulting in taller histogram peaks in the higer values. This will brighten the image but its tonality would decrease.

Problem 3.6

All that histogram equalization does is remap histogram components on the intensity scale. To obtain a uniform (flat) histogram would require in general that pixel intensities actually be redistributed so that there are L groups of n/L pixels with the same intensity, where L is the number of allowed discrete intensity levels and $n = MN$ is the total number of pixels in the input image. The histogram equalization method has no provisions for this type of intensity redistribution process.