CS5680/CS6680 – Fall Semester 2018
Assignment 6 – Color Image Processing and Simplified Research Problems
Due: 11:59 p.m. Thursday, November 8, 2018
Total Points: 60 points

General Assignment Instructions: The same as the previous assignments.

Warm-up Reading (Optional):
Read related materials on Chapter 6 "Color Image Processing" in the textbook "Digital Image Processing".

Problem 1: Color Image Processing [Total: 25 points]

The task here is to help a robot to identify a bright orange ball in its surrounding. The ***ball.bmp*** is an image obtained from a camera mounted on the robot.

1. **[12 points]** The Matlab toolbox provides functions for converting images in the RGB color space to the most common color spaces. Load ***ball.bmp*** and convert it to the HSV color space by calling an appropriate Matlab function. In **H-space** (i.e., Hue image), apply appropriate image processing techniques to separate the ball from the background. Please show your intermediate results in figure 1. Find the centroid of the ball and indicate its location by a cross on the original color image in figure 2.

2. **[13 points]** Find the shadow of the ball and indicate the located shadow using a distinct color on the original image. Please show your intermediate results in figure 3 and your final shadow result in figure 4.

Problem 2: Simple Color Image Retrieval [Total: 20 points]

Content-Based Image Retrieval (CBIR) is a challenging research problem. It uses an image as a query to search for similar images in a large image database. Below is a simple algorithm to search for a small image database using low-level visual features, i.e., normalized color histogram.

1. **[10 points]** Compute Normalized Color Histogram: An image histogram refers to the probability mass function of the image intensities. This is extended for color images to capture the joint probabilities of the intensities of the three color channels. More formally, the normalized color histogram is defined as:

$$h_{A,B,C}(a,b,c) = Prob(A=a, B=b, C=c)$$

where A , B, and C represent the three color channels (R, G, B or H, S, V). Computationally, the color histogram is formed by discretizing the colors within an image, counting the number of pixels of each color, and calculating the percentage of pixels of each color. Implement a **CalNormalizedHSVHist** function to compute the normalized HSV color histogram for a color image. The function prototype should be: hist = **CalNormalizedHSVHist** (im, hBinNum, sBinNum, vBinNum). Here, hist will be a 1-D vector for storing the percentage of pixels of each color. The length of this 1-D vector is hBinNum×sBinNum×vBinNum.

2. **[10 points]** Assuming that you have a small image database with four images, namely Elephant1.jpg, Elephant2.jpg, Horse1.jpg, and Horse2.jpg. Call **CalNormalizedHSVHist** function to compute the normalized 64-bin HSV color histogram (i.e., 4-bin H, 4-bin S, and 4-bin V) for each image in this small database. Plot these four histograms in figure 5 with the appropriate title for each subplot.

Respectively use each image as a query to search this small database and display the retrieval results in figures 6 through 9. Make sure that each figure displays all retrieved images for each query with the corresponding ranking (e.g., ranking of 1, 2, 3, and 4 with 1 indicating the top match, 2 indicating the second top match, 3 indicating the third top match, and 4 indicating the fourth top match) and **the associated similarity score**, which is computed by the histogram intersection defined as follows:

$$d(h,g) = \frac{\sum_A \sum_B \sum_C min\left(h(a,b,c) \times |h|, g(a,b,c) \times |g|\right)}{min\left(|h|, |g|\right)}$$

where ***h*** and ***g*** represent the normalized histograms of two images, respectively. |***h***| and |***g***| represent the magnitude of each histogram, which is equal to the number of pixels in each corresponding image.

Problem 3: A Simple Watermarking Technique in Wavelet Domain [Total: 15 points]

Digital watermarking techniques are viable solutions for copyright protection. Below is a simple algorithm to embed a watermark (a random binary sequence) into an original image and extract the embedded watermark from the unaltered watermarked image. Make sure that you implemented one embedding function and one extraction function.

1. **[10 points] Embedding Procedure:** Apply a 3-level "db9" wavelet decomposition on *Lena* by using an appropriate Matlab function. Apply an appropriate Matlab built-in function to generate ***b***, a random sequence of 0's and 1's whose length equals to the size of the approximation image (i.e., the top left subband LL_3). Make sure that this random sequence will be the same each time you re-run the program. Sequentially embed each value of ***b*** into the approximation subband ***H*** in a raster scanning order (from the left to the right and from the top to the bottom). For each paired ***b*** and ***H***, conduct the following operation using β = 30 and 90, respectively:

$$H'(i,j) = \begin{cases} H(i,j) - (H(i,j) \bmod \beta) + 0.75\beta & \text{if } b(k) = 1 \text{ and } (H(i,j) \bmod \beta) \geq 0.25\beta \\ [H(i,j) - 0.25\beta] - [(H(i,j) - 0.25\beta) \bmod \beta] + 0.75\beta & \text{if } b(k) = 1 \text{ and } (H(i,j) \bmod \beta) < 0.25\beta \\ H(i,j) - (H(i,j) \bmod \beta) + 0.25\beta & \text{if } b(k) = 0 \text{ and } (H(i,j) \bmod \beta) \leq 0.75\beta \\ [H(i,j) + 0.5\beta] - [(H(i,j) - 0.5\beta) \bmod \beta] + 0.25\beta & \text{if } b(k) = 0 \text{ and } (H(i,j) \bmod \beta) > 0.75\beta \end{cases}$$

   Independently perform the inverse wavelet transform after the operation with each of the two β values. For the reconstructed images (i.e., watermarked images) respectively generated by using β = 30 and β = 90, display the original image, its corresponding watermarked image, and the difference image (i.e., the difference between the original and watermarked images) in figure 10 and figure 11. Due to the small differences, you need to apply a scaling operation for a proper display. **Note: Here, the two reconstructed images are the watermarked images, which store/hide the binary sequence (watermark). These two reconstructed images should be the type of uint8.**

2. **[5 points] Extraction Procedure:** For each of the two reconstructed (watermarked) images, perform the following operation: Apply a 3-level "db9" wavelet decomposition. For each approximation coefficient ***H'*** scanned in the raster scan order, if *H'(i, j)* mod $\beta > \beta/2$, set its paired *b'(k)* = 1. Otherwise, set its paired *b'(k)* = 0. Use two sets of "if-else" statements to compare each extracted ***b'*** sequence and the original ***b*** sequence for β = 30 and β = 90, respectively. Display the appropriate messages for each condition and show the percentage of the number of the matched bits.