CS5680/CS6680 – Fall Semester 2024
Assignment 6 – Color Image Processing and Simplified Research Problems

Due: 11:59 p.m. Friday, November 8, 2024
Total Points: 60 points

You can call built-in functions to solve these problems.

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

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

1. [12 points] Load ***ball.bmp*** and convert it to the Hue, Saturation, and Value (HSV) or Hue, Saturation, and Intensity (HSI) color space by calling an appropriate function. In the H channel (i.e., Hue image), apply suitable image processing techniques to separate the ball from the background. Please display your intermediate results in Figure 1. Determine the centroid of the ball and mark its position with a visible blue cross on the original color image in Figure 2.

2. [13 points] Identify the shadow of the ball and mark it on the original image with a distinct color. Please display your intermediate results in Figure 3 and present 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 that uses an image as a query to search for similar images in a large database. Below is a simple algorithm to search for a small image database using low-level visual features, specifically a normalized color histogram.

1. [10 points] Compute Normalized Color Histogram: An image histogram represents the probability distribution of image intensities. For color images, its histogram extends to capture the joint probabilities of intensities across the three color channels. 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 denote the three color channels (e.g., R, G, B or H, S, V). Computationally, the color histogram is created by discretizing colors within an image, counting the number of pixels for 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. This function should have four input parameters **im**, **hBinNum**, **sBinNum**, and **vBinNum**, where **im** is the original color image, and **hBinNum**, **sBinNum**, and **vBinNum** are the bin counts in the Hue, Saturation, and Value color channels, respectively. It should have one output parameter **hist**, which is a 1-D vector storing the percentage of pixels of each color. The length of this vector is **hBinNum×sBinNum×vBinNum**.

2. [10 points] Assuming a small image database with four images, namely ***Elephant1.jpg, Elephant2.jpg, Horse1.jpg, and Horse2.jpg***, call the **CalNormalizedHSVHist** function to compute a 64-bin normalized HSV color histogram (i.e., 4 bins for the Hue color channel, 4 bins for the Saturation color channel, and 4 bins for the Value color channel) for each image in this small database. Plot the four histograms in Figure 5 titling each subplot appropriately. Respectively use each image as a query to search this small database and display the retrieval results in Figures 6 to 9. Make sure that each figure displays all retrieved images ranked by similarity, with rankings of 1, 2, 3, and 4 (where 1 is the top

match) and its associated similarity score, which is computed by the histogram intersection defined as follows:

$$d(h,g) = \frac{\sum_{A}\sum_{B}\sum_{C} min(h(a,b,c) \times |h|, g(a,b,c) \times |g|)}{min(|h|,|g|)}$$

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

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

Digital watermarking techniques offer viable solutions for copyright protection. Below is a simple algorithm to embed copyright-related information (e.g., a watermark in the form of a random binary sequence) into an original image, producing a watermarked image that appears unchanged. This algorithm can also extract the embedded watermark from the watermarked image. Make sure that you implement one embedding function and one extraction function.

1. [8 points] **Embedding Procedure:** Perform a 3-level "db9" wavelet decomposition on ***Lena.jpg*** using an appropriate function. Use a built-in function to generate **b**, a random sequence of 0's and 1's with a length equal to the size of the approximation image (i.e., the top-left subband LL_3). Ensure this random sequence remains consistent each time the program is run. Sequentially embed each value of **b** into the approximation subband **H** in a raster scanning order (left to right, top to bottom). For each paired **b** and **H**, apply the following operation using $\beta = 30$:

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

   After embedding, perform an inverse wavelet transform to obtain the reconstructed image (watermarked image), which hides the binary sequence (watermark). Display the original image, its watermarked image, and the difference image (i.e., the difference between the original and watermarked images) in Figure 10. Due to the subtle differences, you need to apply a scaling operation for better visibility. Note: The reconstructed image should be of type uint8.

2. [5 points] **Extraction Procedure:** For the reconstructed (watermarked) image, repeat the 3-level "db9" wavelet decomposition.
   For each approximation coefficient **newH** scanned in raster order,

   if **newH**(i, j) mod β > β/2 (here β = 30, the same value used in the embedding procedure),
   set its paired **newb**(k) = 1.
   Otherwise, set its paired newb(k) = 0.

   Use "if-else" statements to compare the extracted **newb** sequence with the original **b** sequence. Display the appropriate message for the if-else condition and show the percentage of matching bits.

3. [2 points] Repeat steps 1 and 2 to embed the same random sequence **b** in the original image ***Lena.jpg*** using β = 60. Specifically, display the original image, its watermarked image, and the difference image in Figure 11. Display the appropriate message for the if-else condition and show the percentage of matching bits.