CS5680/CS6680 – Fall Semester 2023
Assignment 6 – Color Image Processing and Simplified Research Problems
Due: 11:59 p.m. Saturday, November 11, 2023
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 obtained 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 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 visible blue 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 (e.g., 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. This function has four input parameters **im**, **hBinNum**, **sBinNum**, and **vBinNum**, where **im** is the original color image, and **hBinNum**, **sBinNum**, and **vBinNum** are the bin numbers in the Hue, Satuation, and Value color space, respectively. It has one output parameter **hist**, which is 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 the **CalNormalizedHSVHist** function to compute the normalized 64-bin HSV color histogram (i.e., 4 bins in the Hue color space, 4 bins in the Saturation color space, and 4 bins in the Value color space) 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 copyright-related information (e.g., a watermark in the form of a random binary sequence) into an original image to produce a watermarked image, which looks like the original image without any noticeable changes. This simple algorithm can extract the embedded watermark from the watermarked image. Make sure that you implemented one embedding function and one extraction function.

1. [8 points] **Embedding Procedure:** Apply a 3-level "db9" wavelet decomposition on ***Lena.jpg*** by using an appropriate function. Apply an appropriate built-in function to generate **b**, a random sequence of 0's and 1's whose length equals 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 $\beta = 30$:

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

   Perform the inverse wavelet transform after the above operation to obtain the reconstructed image (i.e., the watermarked image), which stores or 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 small differences, you need to apply a scaling operation for a proper display. Note: The reconstructed image should be the type of uint8.

2. [5 points] **Extraction Procedure:** For the reconstructed (watermarked) image, perform the following operation: Apply a 3-level "db9" wavelet decomposition. For each approximation coefficient **newH** scanned in the raster scan 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 and the original **b** sequence. Display the appropriate message for each condition and show the percentage of the matched 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. Display the original image, its watermarked image, and the difference image (i.e., the difference between the original and watermarked images) in Figure 11. Display the appropriate message for each condition and show the percentage of the matched bits.