**CSC 545/645 Computer Speech, Music and Images**

**Exercise No. 2a, Week 6, due February 21, 2021**

**Calculating image histograms**

**Goals**

Learn to calculate red, green, and blue histograms.

**Background**

The image histogram is the basis for several simple but effective image modifications. In addition, looking at the histogram of an image can often clarify problems that can be corrected through further processing.

**Procedure**

Write a Processing program (based on *Ex02a\_histograms*, in the Exercises folder) to calculate red, green, and blue histograms.

Three global arrays are declared to hold the red, green, and blue pixel counts. Complete the definitions so that memory is allocated for these arrays. Remember that there are 256 values for each color channel.

The setup() function is written. Complete calcHists to populate the arrays with pixel counts. First initialize the arrays to ensure that all bin values are 0 (use a loop). Next use a nested loop to visit each pixel in the image. Get the red, green, and blue values for each pixel. Cast these values to int because you will use them to index the count arrays (rCount, gCount, and bCount). Think about what the count arrays tell you – for example, rCounts[0] holds the number of pixels with a red value of 0, rCounts[105] holds the number of pixels with a red value of 105, and so forth. So, when you get the red value for a pixel, increment the corresponding index. In other words, if the red value is 125, increment rCounts[125]. Do the same for gCounts and bCounts.

Complete makeTestImage() so you can generate an image in which you know the pixel counts. This will enable you to check your histogram counts so see if they seem to be correct. I have assumed makeTestImage will create an image that is all one color. For example, if you generate a red 10 x 10 test image, then all values of rCounts, gCounts, and bCounts should be 0 except for rCounts[255], which should have a value of 100 (all 100 pixels have a red value of 255). If your 10 x 10 test image is white, then gCounts[255] and bCounts[255] should also have a value of 100. You might want to make a more sophisticated test image but the key is that you know what the red, green, and blue pixel counts should be.

Complete the function alterBrightness() to change the brightness of an image. You will add the amount argument to the red, green, and blue values of each pixel then set the pixel to the color corresponding to the new red, green, and blue values. A positive value for amount will brighten the image; a negative value will darken the image.

Write the printHists() function to print the current histograms to the console. On each line, print the loop counter followed by the corresponding red, green, and blue counts.

Write the keyReleased() function to calculate and print the histogram for the correct image (call calcHists() followed by printHists()). If the key is ‘h’ calculate and print the histograms for img; also set currentImg to img. If the key is ‘b’ calculate and print the histograms for the brightened image and set currentImg to the brightened image. If the key is ‘d’ calculate and print the histograms for the darkened image and set currentImg to the darkened image.

Once you’re convinced your program is working, try it with some images. You might want to try some of the low contrast and underexposed images from the images stored on Blackboard (in Useful Files). You could try larger values of amount for alterBrightness, too.

**Deliverables**

Submit your .pde file on Blackboard by midnight on the due date.