**ESE 358 Computer Vision / ESE 568 Computer and Robot Vision, Fall 2025**

**Electrical and Computer Engineering, Stony Brook University**

**Murali Subbarao**

**Project 1. Getting Started: Basic operations on RGB pixel values**

**Weight 2%. Due 9/9/2025**

All computations below must be done by scanning the images, accessing the value at each pixel, and using it to compute the output at each pixel. Do not use library functions to compute the output. You are permitted to use AI coding tools. You must include the prompts that you used, and the output the AI tool gave.

1. Read an RGB image (8 bits/pixel) in jpeg format. Call this image **A**. The size of **A** should be 512x512. Display image **A** at runtime.
   * Can use shed1-small.jpg or shed2-small.jpg on Blackboard as test images
2. Isolate the three RGB channels of **A.** Call these channel isolations **RC, GC,** and **BC** respectively. Display these three images and save them as jpegs at runtime.
3. Compute the grayscale image **AG** of **A**. Do this by explicitly taking the average of the three channels R,G, and B, at each pixel. Display **AG** at runtime.
4. Compute the histograms of the images **RC, GC, BC**, and **AG**, by explicitly scanning the image to access the value at pixel and counting the frequency of occurrence of each value, and display the histogram plots at runtime.
   * To derive the histogram of each image, count the occurrences of each pixel brightness from 0 to 255.
5. (1 point) Binarizing the image: Threshold the image **AG** to obtain a binary image **AB**. To do this, prompt the user to enter a threshold brightness **TB** as input at runtime (e.g. **TB**=100). You should access each pixel value and compute the new thresholded value.
   * All pixels less than **TB** are assigned brightness 0 (black) and others are assigned brightness 255 (white). Display the output image **AB** at runtime.
6. (1 point) Simple edge detection: Derive the edge image **AE** of **AG** as follows:

Firstly, prompt the user to enter a threshold value **TE** (e.g. **TE** =15).

Next, derive **Gx**, which is the “gradient along rows”:

* For each pixel at indices (m,n) compute:
* Note on edge cases: at the last pixel/column in each row, set **Gx**(m, n) to zero.

Next, compute **Gy**, which is the “gradient along columns” :

* For each pixel (m,n):
* Note on edge case: at the last pixel/row in each column; set **Gy**(m, n) to zero.

Next, compute **GM**, which is the gradient magnitude:

* For each pixel (m,n):

Threshold **GM** using **TE** to compute the edge image **AE**:

* For each pixel (m,n): **AE**(m,n)=255 if **GM**(m,n) > **TE**; else **AE**(m,n)=0.

Display **AE** at runtime.

1. (2 points) Image Pyramid: An image pyramid is a representation of an image where the original image size is recursively down sampled by a factor of 2 for each successive level of the pyramid.
   * To downsample by a factor of 2, take the average brightness of 2x2 image blocks to obtain the corresponding pixels in the successive images. Recursively do this for each successive layer of the image pyramid. Access each of the 4 pixel values and compute the average to get the value of the pixel in the next level.
   * Compute and display the first three levels for the image pyramid of **AG**:
   * Call the three resulting images **AG2**, **AG4**, and **AG8** respectively where **AG2** should is half the size of **AG**, **AG4** is a quarter of the size of **AG**, and **AG8** is 1/8 the size of **AG** (in linear dimensions of height and width, not area).
2. Submit the following on Brightspace: Create a Microsoft docx format file that contains-- source code, and input and output images for each part above. Also, submit a zip file with all files including source files, input and output image files.

SKIP THE FOLLOWING TWO PARTS

1. Read another RGB image B (8bits/pixel) in jpeg format similar to A (e.g. A and B can a stereo image pair with cameras at different positions and directions), and compute its gray-level image BG by taking the average of the three channels R,G, and B, at each pixel.
2. Divide AG uniformly into 16x16 image blocks. Each block is identified by the row and column indices of the pixel at its top left corner. For each block in AG, find the most similar block in BG using the minimum of the sum of squared-differences of corresponding pixels.

Create an output image given by the absolute difference between the corresponding blocks of AG and BG, and display the result as a gray-level image.