MTRE 2610 Engineering Algorithms and Visualization – Dr. Kevin McFall

Laboratory – Image Processing

Introduction

The goal of this laboratory exercise is to use become familiar with digital representation of images and using basic image processing techniques to distinguish colors in an image.

Image Pixels

It may not be obvious in the modern age with high resolution cameras, but digital images are composed of discrete picture elements called pixels. Identifying individual pixels is possible with repeated digital zooms or in images such as the one in Figure 1 from Minecraft. The resolution of an image is the number of pixels in each direction of the image, i.e. a 480×640 image contains pixels organized in 480 rows and 640 columns. Figure 2 shows a small portion of a grayscale image with pixels of varying shades of gray and their numerical representation. Pixels with zero value are black and white pixels are assigned the largest value, which is 255 in this example. The bit depth of an image is not related to its resolution but rather the number of different values a pixel can take. The maximum pixel value of 255 in Figure 2 uses 8 bits since $2^8 = 256$. Somewhat counter-intuitively, image rows and columns are measured down and to the right from the top left corner.

Figure 1: Image from Minecraft with its purposely pixelated graphics.

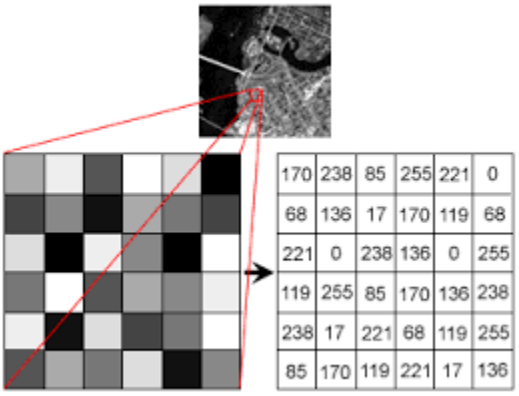

Figure 2: Digital zoom on a grayscale image showing individual pixels and their numerical values.

Several methods exist for representing color images, with the most common considering red/blue/green (RGB) components separately. An RGB image consists of three matrices with the same resolution, one each for strengths of the red, green, and blue components. The grayscale equivalent of a color image is created simply by taking the average of the RGB values for each pixel. In MATLAB, `im = imread('pic.jpg')` reads the `pic.jpg` file and creates a 3-dimensional matrix `im` with the lengths of the first two dimensions indicating the image resolution. The third dimension has a length of 3, which each slice holding the RGB values; for example `im(:,:,2)` is a matrix containing the green content of each pixel.

Image segmentation

The process of identifying a region of interest in an image is called segmentation. This could be identifying road boundaries for a self-driving algorithm or locating the eyes for facial recognition. Segmentation can be accomplished by searching for edges, shapes, colors, textures, etc. and is often a challenging image processing task. Segmentation in this assignment will be accomplished by the user manually clicking on the image to define a region of interest, but automatic segmentation will be covered in the next laboratory exercise. An image can be displayed with the `imshow` command and `[x y]=ginput(4)` will return four x and y image indices where in the image the user clicks with the mouse. Including `axis on` will explicitly display the pixel row/column positions. These four indices are to mark a 4-sided polygon bounding the image region to be considered as indicated by the blue area in Figure 3. The polygon boundary edge line

$$y = mx + b \tag{1}$$

passes through any two consecutive indices such as (x_1,y_1) and (x_2,y_2) where

$$m = \frac{y_2 - y_1}{x_2 - x_1} \text{ and } b = y_1 - mx_1 \tag{2}$$

Note that the y-axis in Figure 3 points down to be consistent with `imshow` axes. The inequality

$$y_c > mx_c + b \tag{3}$$

holds for point (x_c,y_c) as long it lies on that side of the line defined by m and b whereas

$$y_c < mx_c + b \tag{4}$$

would be true if (x_c,y_c) fell on the other side, i.e. the red region in Figure 3.

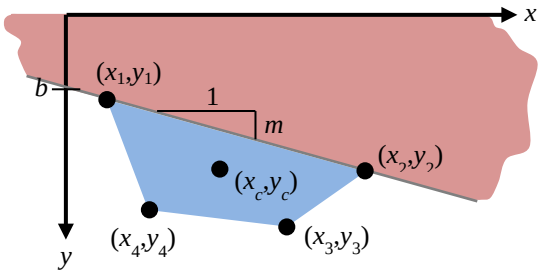

Figure 3: Geometry of a line defined by two points in relation to other points in the domain.

To identify the blue area in Figure 3 bounded by the four clicked points, first create a `meshgrid` of `x` and `y` integer values from 1 up to the resolution of the image, i.e. the `size` of `x` and `y` should be match the image size. The expression `y<m*x+b` returns a logical matrix where pixels in the red area in Figure 3 are true. All the RGB intensities at these locations should be set to 0 to eliminate pixels outside the blue bounded area. Repeat this process for the other three lines comprising the polygon to remove everything but the blue region. The choice of using `>` or `<` in the inequality will depend on which side of the line the center point lies. The appropriate choice is the opposite of whether y_c is greater or less than $mx_c + b$. The area center point x_c and y_c is located by taking the average of the four x_i and y_i points, respectively.

Automatically identifying colors

The task in this exercise is for the user to choose one of the colored file folders in Figure 4 by clicking on its corners in the image, and for the program to return the folder color, either yellow, blue, green, orange, or purple. Once the image is segmented, extract all nonzero entries for each of the RGB components and display a `hist`ogram of their values in a single figure with 3 `subplot`s as in Figure 5 for the green folder. Histograms plot the frequency a given value occurs in series of data. For an image, values will range from 0 to 255 for an 8 bit-depth. The green component is the strongest in Figure 5 with an average value around 175 compared with the red and blue averages which are both less than 125.

The `hist` command accepts a vector of values, whereas the pixel values are in matrix form. Some options for overcoming this include using `reshape` to change the matrix to a vector or using the fact that `find` returns linear indices when used with a single return argument. Although MATLAB dynamically type casts variables, casting must sometimes be performed manually. The `imread` function creates matrices of the unsigned 8-bit integer data type which for some reason cause errors using `hist`. For this reason, convert the extracted non-zero pixel integer values to `double` before passing them to `hist`. Unlike `plot`, `hist` does not return a handle for the displayed bar graph. The handle can be returned using `get` where the commands

```
set(get(gca,'children'),'facecolor',[1 0 0])
set(get(gca,'children'),'edgecolor',[1 0 0])
```

set the color for both the fill and edge lines in the bar graph to red, as the last argument is the RGB strengths of the desired color. Adjust the horizontal `axis` in the histograms to range from 0 to 250 as in Figure 5 to allow easier comparison of RGB components.

Calculate the average RGB values for the segmented area and use them to create an `if else` structure to classify the color of the selected area. Ensure that all folders are correctly classified for all sample images available on D2L.

Figure 4: Sample image with file folders (left) whose colors are to be automatically identified when segmented manually (right) by the user clicking on the four corners of a folder.

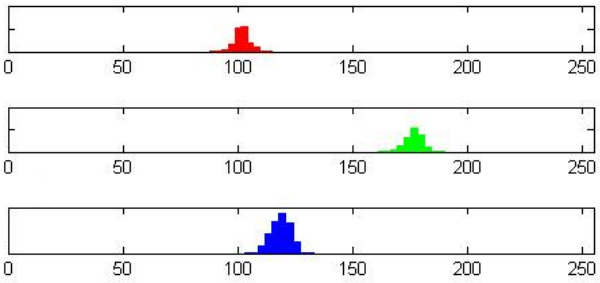

Figure 5: Histogram of RGB values when selecting the green folder in Figure 4.

Laboratory exercise procedure

Rather than attempting to solve the entire problem in one pass, start small and incrementally add complexity until the problem is solved. Before eliminating pixels on all four sides of the segmented folder and concerning on which side the polygon center lies, black out pixels on one side of the line defined by two points identified by the user. Complete the following code fragment to accomplish this task.

```
im = imread(   );
R = im(  ,  ,  );
G = im(  ,  ,  );
B = im(  ,  ,  );
figure(1); imshow(im); axis on;
[ ] = ginput(2)
m =
b =
[x y] = meshgrid(  ,  );
R(y > m*x+b) = 0;
G( ) =
B( ) =
imNew(  ,  ,  ) = R;
imNew(  ,  ,  ) = G;
imNew(  ,  ,  ) = B;
figure(2); imshow(imNew);
```

Now modify the code to request four clicks from the user, which are assumed to be adjacent corners defining a four-sided polygon. Compute the center of the polygon and use it to determine which side to keep for each of the four lines defined by each side of the polygon. Display the segmented image keeping only the region within the mouse clicks.

Grading rubric

1. 20 points: Black out all pixels on one side of a line defined by clicking twice in an image
2. 40 points: Display the segmented image
3. 20 points: Produce the R, G, and B histograms for the segmented image
4. 20 points: Demonstrate correct classification for all folders