ELEC 474

Machine Vision

Lab 1

September 10th 2019

1. Introduction

The purpose of this lab is to introduce you to OpenCV, and to give you some experience in accessing and manipulating images. The lab is divided into five sections, each of which requires a particular program to be written. All of the programs follow a similar pattern: First, one or more images is loaded. Next, each pixel of the image is accessed using a nested for loop, and a new image is formed. Finally, the result is displayed or written to disk.

The basic structure of the code is provided in the file **`imshow.cpp`**. You are allowed to use any standard OpenCV functions to achieve your goal, so long as the above pixel-wise processing format is adhered to. Useful functions are:

```
Mat img = imread("imagefile.jpg" ) ;

namedWindow("image A", CV_WINDOW_AUTOSIZE);

imshow("image A", img);

waitKey();
```

Also, pixel access can be achieved as follows:

```
int row, col ;

Mat img ;

// ..

char val = img.at<char>(row,col) ;
```

At the end of the lab, once all of the sections are complete, have a TA or instructor evaluate your work and assign a grade.

2.1 Image Statistics

A gray-scale image can be considered to be a 2D array of pixels, where each pixel takes on a value between 0 and 255, where 0 represents pure black, 255 pure white, and any other value a shade of gray. (Depending on the image data type and access, this range may appear shifted to [-128,127].)

Write a program that first loads and displays an image. Next, loop through each pixel in the image and calculate the min, max, and average pixel value, and print these values to the console. Test your program on image **`lena2.jpg`**.

2.2 Image Inversion

Write a program to invert a gray-scale image and display the result. Any input pixel with value 0 will be inverted to 255, value 1 will invert to 254, etc. The output image with therefore look like a "negative" of the input.

2.3 Image Flip

An image is considered to be a 2D array of OpenCV class **`Mat`**. Each pixel is uniquely identified by its row and column location. The indexing for an image is different than for standard arrays, as the origin is in the top left corner. Also, in OpenCV, the row (i.e. y-coordinate) is indexed first, e.g.:

```
Mat img ;

int row, col ;

// …

char val = img<char>(row,col) ;
```

Write a program that loads and displays an image, and then transforms and displays a version of this image that is a mirror reflection in the y-axis. Do the same by for the x-axis. Test your program on image **`lena2.jpg.`**

Q/ Why is this an easy image to test this function?

2.4 Image Merge

Write a program that loads and displays two images of the same dimension. Then, create and display a new image which is the merger of the two input images. Rather than doing a straight arithmetic sum, for each pixel location, assign that pixel value which is the maximum value from either input image. In this way, the result will be similar to a logical OR of the two images.

Test your program on images **`octo1.jpg`** and **`octo2.jpg`**.

2.5 Image Difference

Image differencing is a method of determining subtle changes in a scene. Two images are acquired of a scene, and then the difference between each respective pixel value is calculated. If the pixel difference is larger than some threshold, then there has been some change in that image location, and that pixel is considered to be *different*. Alternately, if the difference is small, then the pixels are considered to be the *same*.

Write a program that takes two images as input, and classifies each pixel as either different or same. The output will be a binary image, where the different pixels are displayed as pure white (value 255), and the same pixels are displayed as pure black (value 0). Test your program using input images **`messy_desk_A-1.jpg`** and **`messy_desk_B-1.jpg`**. Try altering the value of the difference threshold to see what effect it has.

Q/ What are some of the possible causes of the differences between the two images?