MOBILE SENSING LEARNING

Mobile Sensing Video Assignment Three: Visualize It

You are to complete this assignment in **groups of up to three**. Start with the unmodified Xcode project from the video lecture. Use the branch titled *OpenCVExamples*.

Part One: Replace the processImage Function in OpenCVBridge

1. **[0.5 points]** In the `OpenCVBridge` class, create a new function called `processFinger` that returns a boolean value. Make sure this function is public. For now, just return false. You will update this function in the next steps.
2. **[0.5 points]** Update `ViewController.swift` to (1) use the back camera and (2) call the `processFinger` method instead of the `processImage` method (that is, do not call the `processImage` function). Remove any function calls or code inside `processFromCamera` that will be unneeded for processing if a finger is over the camera (such as face detection).

Part Two: Process the Mean and Verify

1. In `OpenCVBridge.mm`, notice that the `processImage` function has a section of code that takes the average of the Red, Green, and Blue channels and displays the values on the image using the function `cvPutText`. Copy the code for channel averaging from `processImage` into `processFinger`. Be sure any dependent operations are also copied. You might be interested in example code on the next page of this document.
2. **[0.5 points]** Does the current implementation of `cvtColor` and `avgPixelIntensity` correctly capture and interpret the red, green, and blue channels of the image? **If not, troubleshoot the problem and fix it** in the `processFinger` method. Hint: look closely at the color channels.

Part Three: Detect a finger and Update the UI

In this part of the assignment you will create an array and save the values from the `avgPixelIntensity` data that gets the mean of each color channel.

1. **[1 points]** Look at the values of each color channel when you place your finger over the camera. Come up with a way of detecting when someone places his/her finger over the camera (or anything over the camera, not just a finger). **Implement it in the `processFinger` method. Return true if a finger from `processFinger` is discovered.**
2. **[0.5 points]** When a finger is over the camera, save the average red, average green, and average blue values into three separate float arrays of length 100 points. That is, each element in the arrays will be the average color of a new camera frame. When you have captured 100 frames' color, print something to the image using `cvPutText`.
3. **[0.5 points]** When someone places their finger over the camera, disable the camera position toggle button and the torch toggle button. Re-enable them after taking the finger off of the camera. You will need to use the boolean return value in the `processFinger` method.
4. **[1 points]** Update your code to turn on the torch whenever someone places their finger over the camera. Turn off the torch when they remove their finger. The function should not be intermittent (i.e., it should not turn on and off multiple times when a finger is placed/removed).
5. **[0.5 points] For thought:** Given that each float array is 100 points: how many milliseconds of data has been collected? Please show your work.

What to turn in:

- Be sure to indicate who is part of your team.
- The Xcode project of the updated file (as a zipped/compressed file). **DO NOT** include the OpenCV framework in your zipped file.
- Answer the questions posed from above, include written answers in the upload.

SMU | Bobby B. Lyle School of Engineering

```
cv::Mat image_copy;
char text[50];
Scalar avgPixelIntensity;

cvtColor(_image, image_copy, CV_BGRA2BGR); // get rid of alpha for processing
avgPixelIntensity = cv::mean( image_copy );
sprintf(text,"Avg. B: %.0f, G: %.0f, R: %.0f", avgPixelIntensity.val[0],
       avgPixelIntensity.val[1],
       avgPixelIntensity.val[2]);

cv::putText(_image, text, cv::Point(0, 10), FONT_HERSHEY_PLAIN, 0.75,
              Scalar::all(255), 1, 2);
```