**Pixel-Based Processing  
ECE 847 Assignment #2**

**Purpose:** In this assignment you will learn to apply a number of basic pixel-based processing algorithms such as thresholding, floodfill, connected components, and morphological operations.

**Before you start:** Watch the Lecture 1 series and the Lecture 2 series.

**Background info:** As mentioned in the introduction, machine vision is a mature field. In machine vision, the goal is to extract useful information from images in a controlled environment such as a factory setting. One common scenario is that of parts arriving on a conveyor belt, with an overhead camera positioned to analyze the parts. Typically the goal is to detect the individual objects, segment them from one another, compute their spatial position and orientation, classify them, and detect defects. This assignment involves building an end-to-end system that achieves some of these real-world goals in a restricted scenario. Although the assignment involves problems such as segmentation and classification, it does so in a restricted context without requiring knowledge of more advanced algorithms in these areas.

**Instructions:**

1. In this assignment you will write an application (that is, you will modify the code in *homework*) to automatically detect and classify fruit on a dark background. The code should perform the following steps in order when it is run:
   1. Reads 1 command-line parameter, which we will call *filename*.
   2. Loads *filename* from the *blepo/images* directory into a Grayscale or BGR image and displays it in a figure window.
   3. Performs double thresholding using two thresholds that you determine by trial and error, which are hardcoded in your code.
   4. Perform noise removal (if needed) using mathematical morphology (i.e., some combination of erosion / dilation / opening / closing) either before or during thresholding.
   5. Runs connected components (either the classic union-find algorithm or by repeated applications of floodfill) to detect and count the foreground regions in the cleaned, thresholded image, distinguishing them from the background.
   6. Computes the properties of each foreground region, including
      1. zeroth-, first- and second-order moments (regular and centralized)
      2. compactness (To compute the area, simply count the number of pixels. To compute the perimeter, apply the logical XOR to the thresholded image and the result of eroding this image with a 3x3 structuring element of all ones; the result will be the number of 4-connected foreground boundary pixels.)
      3. eccentricity (or elongatedness), using eigenvalues
      4. direction, using either eigenvectors (PCA) or the moments formula (they are equivalent)
   7. Automatically classifies each piece of fruit into one of three categories: apple, grapefruit, or banana, using a combination of these foreground properties.
   8. Detects the banana stem. There is no one right way to do this. A variety of techniques have been developed by students over the years such as eroding until the stem disappears; rotating to align the major axis with the horizontal axis, then projecting onto the horizontal axis and finding the thinnest columns; or finding the further point from the centroid then growing the region. Feel free to be creative if you can think of another way.
2. Your output should look like this:
   1. One figure window should show the original image. Three additional figures should show the result of thresholding the image with the low and high thresholds, along with the output of double-thresholding. Be sure to set the title of each figure to an appropriate human-readable string that indicates what is being displayed. Feel free to display additional intermediate results in other figures if you like.
   2. In a final figure, display the original image with a one-pixel-thick boundary overlaid on each object, the color of the boundary indicating the type of fruit: Red indicates apple, green indicates grapefruit, and yellow indicates banana. For each object, draw a cross at its centroid and draw two perpendicular lines (with appropriate lengths) to indicate the major and minor axes. Indicate the banana stem by coloring with magenta the boundary pixels in that portion of the banana.
   3. Print out all the region properties you computed on the console window (using printf or cout).
3. The grader will test your code on the images fruit1.pgm and fruit2.pgm (or, in BMP format, fruit1.bmp and fruit2.bmp), as well as for other similar images. The other images will have the same scale and lighting conditions, but the image dimensions, rotation, and number of fruit instances may change. The same algorithm parameters will be used for all images.
4. For this assignment, you may use any functions in the Blepo library. In particular, you will find functions for dilation, erosion, floodfill, and connected components in ImageOperations.h and ImageAlgorithms.h. None of these functions is hard to write, and you are free to implement your own versions instead.
5. Submit your code to the grader as described in the first assignment.