# ECE 5273
# Homework 2

Spring 2025 Dr. Havlicek

Obtain the images `Suzi1.bin` and `ct_scan.bin` from the course web site. Each image has 256 × 256 pixels and each pixel has 8 bits.

In this assignment you will perform object extraction (target extraction) by using simple thresholding followed by connected components labeling (blob coloring) with minor region removal. This is a special case of two classical image processing problems known as *image segmentation* and *classification.* Throughout the assignment, including the printing of your results, use a value of 255 (Hex 0xFF) for LOGIC ONE and a value of zero (Hex 0x00) for LOGIC ZERO.

**Objectives:**

1. *Suzi1:* the first objective is to produce a binary image **J** that is LOGIC ONE at pixels contained in the "girl" object of the original image and LOGIC ZERO at pixels contained in the background of the original image. The second objective is to produce a grayscale image **K** of the segmented "girl" object. At pixels where **J** is LOGIC ONE, **K** should be equal to the original *Suzi1* image. At pixels where **J** is LOGIC ZERO, **K** should be 255 (to make a "white background" for the segmented object image).

2. *ct_scan:* the first objective is to produce a binary image **J** that is LOGIC ONE at pixels contained in the "torso section" object of the original image and that is LOGIC ZERO at pixels contained in the background of the original image. The second objective is to produce a grayscale image **K** of the segmented "torso section" object. This should be done exactly the same way it was for the *Suzi1* image.

For each image, use the following procedure:

A) Study the image and select an appropriate threshold that will discriminate between the desired object and the background.

B) Form a binary image **J** by applying the threshold so that pixels likely to be part of the desired object are assigned the value LOGIC ONE, while those likely to be part of the background are assigned the value LOGIC ZERO.

   **Hint:** for the *Suzi1* image, this means that $J(i, j)$ should be LOGIC ONE if the corresponding input pixel is *below* threshold. For the *ct_scan* image the opposite is true: you should set $J(i, j) =$ LOGIC ONE if the corresponding input pixel is *above* threshold. This is because the object of interest in the *Suzi1* image is *darker* than the background, whereas the object of interest in the *ct_scan* image is *brighter* than the background.

C) Apply connected components labeling with minor region removal (as described at pp. 2.40-2.47 of the course notes) to refine the segmentation in **J**,

D) Construct the segmented object grayscale image **K**.

**Turn In:**

- code listing.
- brief explanation of how the threshold value was determined for each image.

- for each original image (*Suzi1* and *ct_scan*):
  - the original grayscale image;
  - binary image of the initial thresholding result (show the threshold value as a header, caption, or annotation);
  - grayscale image of the blob coloring result before minor region removal (Note that this basically means displaying the "R" image of labels or "blob colors." Try to use a separate grayscale for each blob – if there are more than 256 blobs then some of them will have to share grayscales. Optionally, if you have time, you might want to consider adapting the "approximate contour generation" algorithm given on pp. 2.104-2.107 of the course notes to draw an outline around each blob);
  - binary image of the blob coloring result after minor region removal;
  - the final binary image obtained after the full procedure (including blob coloring with minor region removal, inversion, re-application of blob coloring with minor region removal to the inverted binary image, and then the final inversion step – see notes p. 2.47);
  - the segmented object grayscale image **K**.

**NOTE:** if you are using Matlab to display your images as shown in the example on p. 1.5 of the course notes, then you need to **make sure** to issue the command `axis('image');` right after you call `image();` to display your image. This is because the default behavior of the Matlab `image();` command is to display with an *unequal aspect ratio* where each pixel is displayed as a rectangle instead of a square. This will distort the appearance of your image, which is undesirable for our square images. To correct this, you simply issue the command `axis('image');` immediately after calling `image();` as shown on p. 1.5 of the course notes. Alternatively, you can avoid this aspect ratio problem by calling `imshow()` instead, e.g., `imshow(Suzi1,[0,255]);`.

**DUE: 2/4/2025**