MP1 - CCL

Logan Boswell

Algorithm Description

This MP is focused on connected-component labeling, a technique used to identify regions in an image. In order to implement this technique, I wrote a function that takes in a bitmap image, labels the regions in the image, and returns both the labeled image and number of regions in the image. Before looping through the pixels in the image, some empty lists are defined to store the following: sets of tuples representing pixel coordinates in an image, sets of region labels representing equivalent labels, redundant labels, indices of sets to remove from region equivalences list, and indices for which regions made it through the filtering process. Sets were used where I could add items without worrying about having duplicates and use boolean algebra like unions and intersections. After these are defined, the first step taken is to loop through each pixel in the image, starting from the top row and moving from left to right. If a pixel is white, then two of its neighbors are inspected to see if they are also white and labeled. The neighbors of interest are the pixels directly to the left and directly above the current pixel. There are 5 cases for the neighbor labels shown in the table below.

Table 1: Neighbor Label Cases

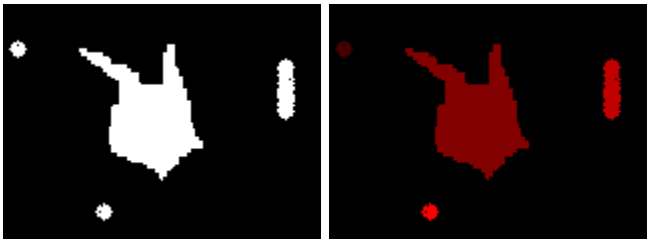
Both neighbors have not been labeled	Pixel coordinates are appended to the list of region sets as a tuple in a set
Only top neighbor has been labeled	Pixel coordinates are added as a tuple to the set containing the top neighbor coordinates
Only left neighbor has been labeled	Pixel coordinates are added as a tuple to the set containing the left neighbor coordinates
Both neighbors have the same label	Pixel coordinates are added as a tuple to the set containing the neighbor coordinates
Both neighbors are labeled, but the labels do not match	Pixel coordinates are added as a tuple to the set containing coordinates of the neighbor with the minimum label - equivalences must be updated

For the case where the neighbors are labeled differently, the region equivalences list is updated. This process begins with checking if the region equivalences list is empty. If so, the neighbor labels are appended to the region equivalences list as a set. If not, the list is parsed to check if the labels are included in any set in the list, where they will be added to the list if so and appended to the list as a new set if not. This includes all of the steps taken in the first pass through the image. Before the second pass, a few steps are taken to combine the region equivalences in the minimum number of sets possible. This starts with looping through the list of sets representing label equivalences. Starting with the first set in the list, the intersection of the current set is calculated with each subsequent set. If the intersection with a subsequent set is not equal to an empty set, the current set is replaced with the union of the current set and subsequent set, and the index of the set is added to the list of indices to remove. This list is sorted in descending order and then used to remove elements from the list of region equivalence sets at the specified indices. From here, the second pass starts by looping through sets in the region equivalences list. For each set, the minimum label is determined and removed from the set, and then for each label, replace the set in the regions list corresponding to minimum label with the union of it and the rest of the sets corresponding to the labels in the equivalences set while saving the corresponding indices in the redundant indices list. This list is sorted in descending order and then used to remove unnecessary elements in the regions list, which is the final step in the second pass. After the second pass is completed, all that is left to be done is labeling the regions in the image. In order to label the regions, different shades of red are determined by using linspace to make a list of equally spaced unsigned 8-bit integers between the bounds of 70 and 255. The list is the same length as the regions list. Using these shades, the regions list is parsed and the value of pixels in a certain region are replaced with the corresponding shade. Darker shades of red correspond to regions that were labeled earlier in the labeling process.

Results on Test Images

One region identified

Six regions identified



Four regions identified

Filtering

An optional filtering step is also included in this algorithm which executes based on two optional arguments in the function - a boolean value if filtering should be done (default value of False) and an integer representing the pixel threshold (default value of 0). This optional step occurs right before the shades of red are determined and assigned to the regions. In the filtering step,

each set in the regions list is checked to see if it contains more than the threshold amount of pixels, and if not, the set is removed from the regions list.

Filtering Results

The corresponding thresholds (from left to right) are 47, 50, and 244