Report : Connect Component Labeling

Algorithm:

This algorithm identifies and labels connected foreground(non-zero pixel intensity) components in a grayscale image. The algorithm is implemented using Python. It consists of the following steps at high level

1. Reads an input grayscale image
2. Identifies and labels the connected components using **ccl(path)** function.
3. Filters out the noise in the image(by a percentage of the biggest connected component) using **filter_noise(img, threshold)** function.
4. Converts the labeled, filtered image to a colorized image(with each unique component displayed in a random color) using **colorize(img)** function.
5. Saves the colorized image to an output file
6. Displays the final segmented, colorized image

Entry point to the program is the **main()** function. It sets the input path, filter, calls the ccl(path) function, prints statistics and displays and saves the final output image.

The main function **ccl(path)** follows the below steps:

- The algorithm makes two sequential passes over the image, which identifies and labels all the connected components. The image traversal starts from [0,0] and is traversed from left to right and top to bottom.
- Uses a variable E_TABLE to keep track of the relationship between component labels.
- In the first pass, the algorithm assigns a label to each foreground pixel(pixel intensity more than 0) based on its upper and left neighboring pixels.
    - If both neighbors are background(pixel value 0), assign a new non-zero component label to the current pixel and add the label to the E_table.
    - If either of the neighbors(but not both) are background, assign the non-zero label from the neighbor to the current pixel
    - If both neighbors have same, non-zero label, then assign that label to the current pixel
    - If the neighbors have different, non-zero labels, assign the minimum label to the current pixel, and add a new relation in the E_TABLE.

- After the first pass the algorithm simplifies the E_TABLE, by using a depth-first search on it to group together the connected labels within the E_TABLE.
- It then does a second pass over the image to update the labels to the smallest value based on the table. During the pass, if the label value of the pixel is a key of the E_TABLE, no update is done as it is already the smallest possible component. But if the pixel label is a value in the E_TABLE, the pixel label is updated with its key in the E_TABLE, as they belong to the same connected component and the key is the smallest value possible for that component.

The function **filter_noise(img, threshold)** is added to help filter out any noise in the data. It filters out any connected component that is smaller than the threshold percentage of the biggest connected component. i.e, if we input a threshold of 10, any connected component less than 10 percentage of the size of the biggest connected component is filtered from the final image.

Results & Analysis:

The connected component labeling algorithm was able to successfully identify and label the connected components on the following test data. The filter function was useful in filtering out any unwanted noise in the data, so that we can obtain more meaningful images as the final result of connected component labeling. The filter had to be varied depending on the test data to get more accurate results. For example in the gun.bmp test image, the threshold value was increased so that all the noises in the image can be successfully filtered.

1. Face.bmp (filter threshold = .1)
   The algorithm identified six connected components on this image. After applying the filter also the connected components were the same, meaning there was no noise in the image smaller than the threshold.
2. test.bmp
   The algorithm identified one connected component on this image. After applying the filter also the connected components were the same, as there was no other component in the image to filter out.
3. gun.bmp (filter threshold = .1)
   The algorithm identified four connected components on this image. After applying the filter also the connected components were the same, which means the threshold was way too low for the noise to be filtered out. (gun-unfiltered.jpg)

4. gun.bmp (filter threshold = 5)
   The algorithm identified four connected components on this image. After applying the filter with an increased threshold the noise was filtered out.