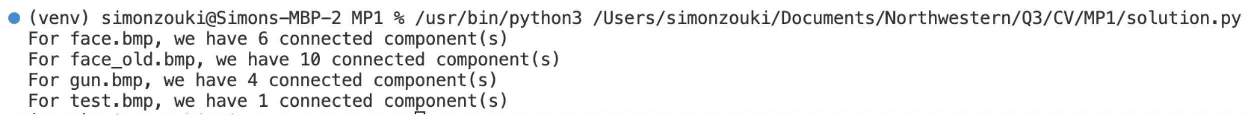
HW1 Report

Problem Description

This homework covers an important problem in Computer Vision as described in class. As described in the assignment description, it is also an important aspect of the course project. To repeat what is needed, given an image, the algorithm developed should detect the different components that are
The connected component labeling problem involves identifying and labeling individual connected regions in the given binary image. In our case, the connected component is a group of neighboring pixels (4-neighbors in this case) that are white and have a pixel value of 255. The output of the algorithm should be assigning unique numbers for every connected component.

Proposed Algorithm

The proposed algorithm goes through the image row-by-row. When it finds a pixel of value 255 (white), it checks the left and top neighbor, because checking the right and bottom one is useless since they haven't been reached yet due to the nature of the traversal. If any of the neighbors are already part of a component, we propagate it to the pixel we are on. If the 2 neighbors have different values for the components, we select the top one, and note the conflict (which will be resolved later).
After getting through the image, we go through the image one more time to solve the conflicts and get the final labeling. We return the label mapping and the number of labels. The results shown below show the number of labels for every image.

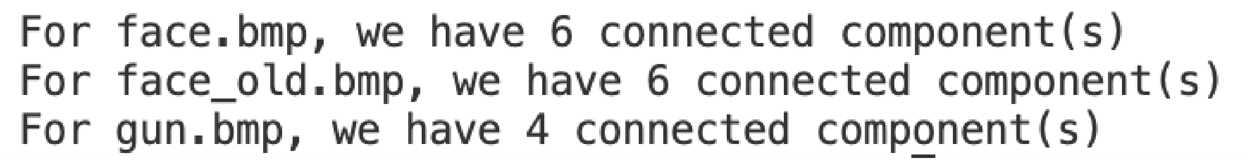


```
● (venv) simonzouki@Simons-MBP-2 MP1 % /usr/bin/python3 /Users/simonzouki/Documents/Northwestern/Q3/CV/MP1/solution.py
  For face.bmp, we have 6 connected component(s)
  For face_old.bmp, we have 10 connected component(s)
  For gun.bmp, we have 4 connected component(s)
  For test.bmp, we have 1 connected component(s)
```

The results are straight forward for the test, gun, and face images as we can see the connected components very clearly in the images. For the face_old, there are a few small components that can't be seen directly on the image, but after further analysis and zooming in we can see that the return number of 10 is correct. This all suggests that the algorithm is working as desired.

Another solution would have been the recursive solution, which we showed in class to be less efficient when it comes to space complexity, so the proposed solution is the better one.

I also implemented a size filter to disregard the component with less pixel than a chosen threshold. I chose 32 as a value for the threshold and we get the following result, which makes sense. The code is attached for reference.



```
For face.bmp, we have 6 connected component(s)
For face_old.bmp, we have 6 connected component(s)
For gun.bmp, we have 4 connected component(s)
```

I also returned the output images by mapping the labels to different shades of binary colors. One of the examples can be seen here. It was also attached to the solution.