CMPEN 455: Digital Image Processing

Project 2

Connected-Component Labeling and Set Operations

Jonathan Lausch, Qiong Li, Zekai Liu

9/27/2019

A. Objectives

The main purpose of this project was to isolate bright connected components in an image, label them, and find the biggest regions from these. In addition, another goal was to visualize the effects unary and binary set operations had on images.

B. Methods

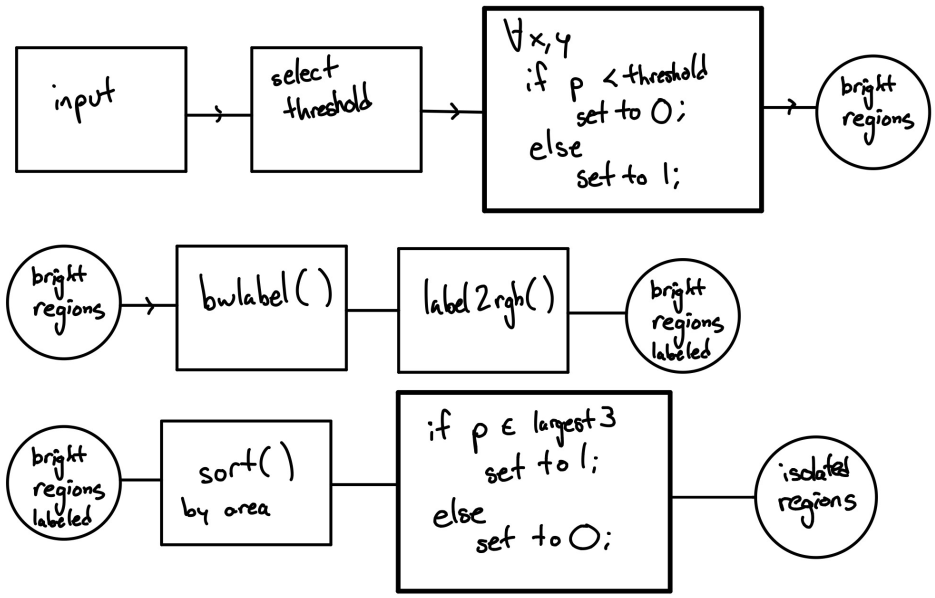
To run code for Question 1, please navigate to Q1>Code and open the 'output_q1.m' file in Matlab and press Run. Similarly for Question 2, navigate to Q2>Code and open 'output_q2.m' in Matlab and press Run. Make sure that Matlab is in the correct directory as the input files are also in each 'Code' folder.

Question 1 was about finding bright regions in within an image. To do so, we scanned through each pixel of the input image and depending on the grayscale value of that pixel, the new image would be assigned a value of 1 or 0 in the corresponding position. The value '1' indicates that the original value was greater than or equal to a predetermined threshold value, and 0 indicates a value less than that of the threshold. Figure 1 shows the result of the described operation with a threshold of 188. We saved this image as "fthresh".

For part b we used the specified bwlabel() function to find connected components of the threshold image "fthresh" that were 8-connected. Figure 2 shows each of these components separated by color, which was done using the label2rgb() function. Each unique color represents pixels that are 8-connected.

To find the largest 3 connected components, we wrote the **largest3reg** function, which generates the 3 largest regions from the color-labeled image. Using regionprops(), we found the area of every connected component in the image and then sorted them by their sizes. All pixels not in the largest 3 components were zeroed out. The result is shown in Figure 3.

Flowchart of Question 1 Process:

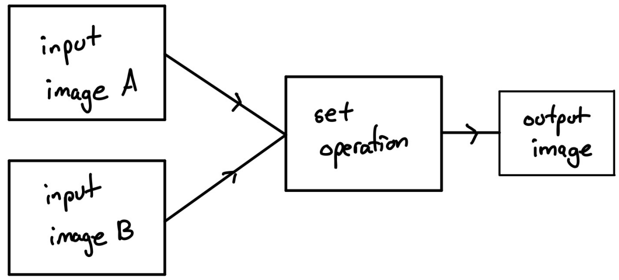


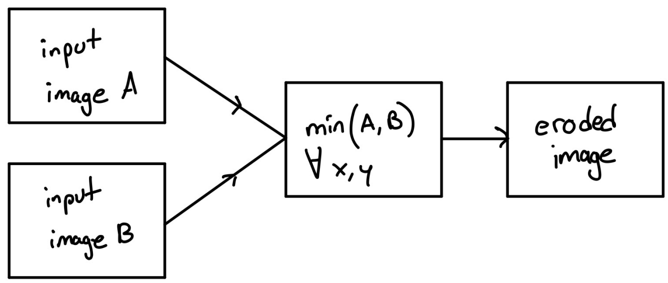
For the binary image operations A AND B, A OR B, A XOR B, and NOT A, their analogous set operations are $A \cap B$ (intersection), $A \cup B$ (union), $(A^c \cap B) \cup (A \cap B^c)$(A complement intersection B unioned with A intersection B complement), and B^c(complement). respectively (Q2_a).

To code each binary-image operation, we essentially translated the truth table for each into a series of if-else statements to avoid using the built in Matlab operations. Figures 4, 5, & 6 show the resulting images with each corresponding operation labeled beneath. Figure 7 shows the result of the unary NOT operation, which was coded in the same fashion as the binary operations, using the corresponding truth table as if-else statements.

The final part of Question 2 was the binary operation imageMIN, which takes the minimum pixel value from two images to create a new image that is composed of just these minimum values. The result is seen in Figure 8 and is visually an 'intersection' of the two images.

Flowchart of Question 2 Processes (Set operations above, imageMIN below)





C. Results

Figure 1 - Pixels greater than the threshold value identified. (White is greater than the threshold, and black is less than the threshold)

Figure 2 - Color-labeled 8-connected components

Figure 3 – Largest connected components from Figure 2

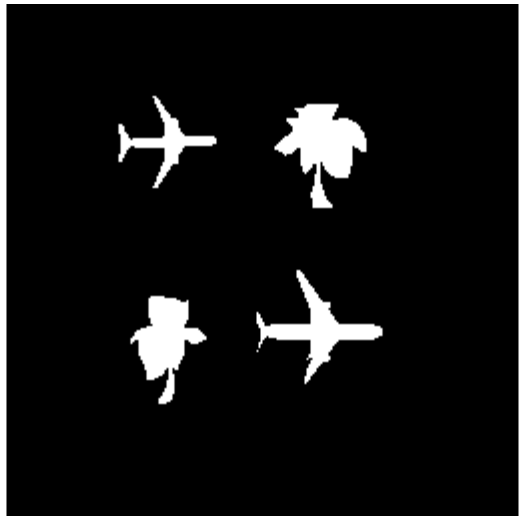


Figure 4 – AND Result

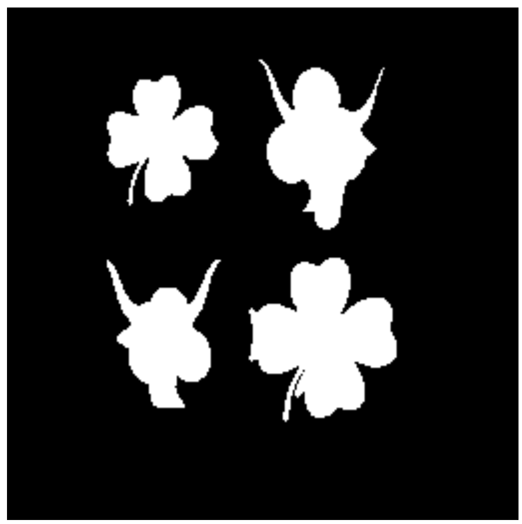


Figure 5 – OR Result

Figure 6 – XOR Result

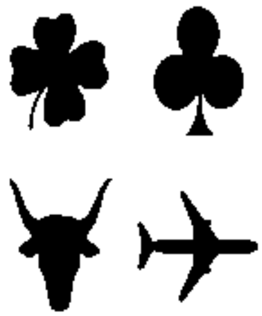


Figure 7 – NOT Result

Figure 8 – Image Minimum (Erosion)

D. Conclusion

In this project, we were tasked with isolating 3 bright components from an image. It is easy to see how this same process of selecting a threshold to isolate certain elements could be applied to finding pixels of a certain color, or all pixels below a given threshold. This project also gave us the opportunity to visualize set operations, by treating images as sets. In addition, we were able to visualize the combination of the minimum-valued pixels from two images, which retains the main characteristics from each photo. This appears to have a similar effect of lowering the opacity of an image and then placing it on top of another in a program such as Photoshop.