CMPEN/EE455: Digital Image Processing I

Computer Project # 2

Connected-Component Labeling and Set Operations

Ya Mei, Ziheng Fu, Songmeng Wang

Date: 09/27/2019

Objectives

The objective of this project is to get familiar with the application of a sequence of simple image-processing operations to an image. Both connected-component labeling and logical (set) operations would be used. This project also helps students learn more about binary images and relative MATLAB functions.

Methods

1): Bright-Region Extraction

a): Basic Idea:
The basic idea to realize bright-region extraction is to distinguish some regions with bright pixels and eliminate all other regions. The first step is to filter the image by attributing value 1 to the pixels whose original gray value is larger than the threshold and attributing value 0 to others. Apparently, the decision of setting the threshold is critical for the following operations. By doing this, the image is transformed to a binary gray-level image with the background of 0 and the foreground of 1. Subsequently, all the remained regions are labeled with a particular value so that three largest regions (three visually largest regions) could be distinguished by performing sort and *find* function in MATLAB.

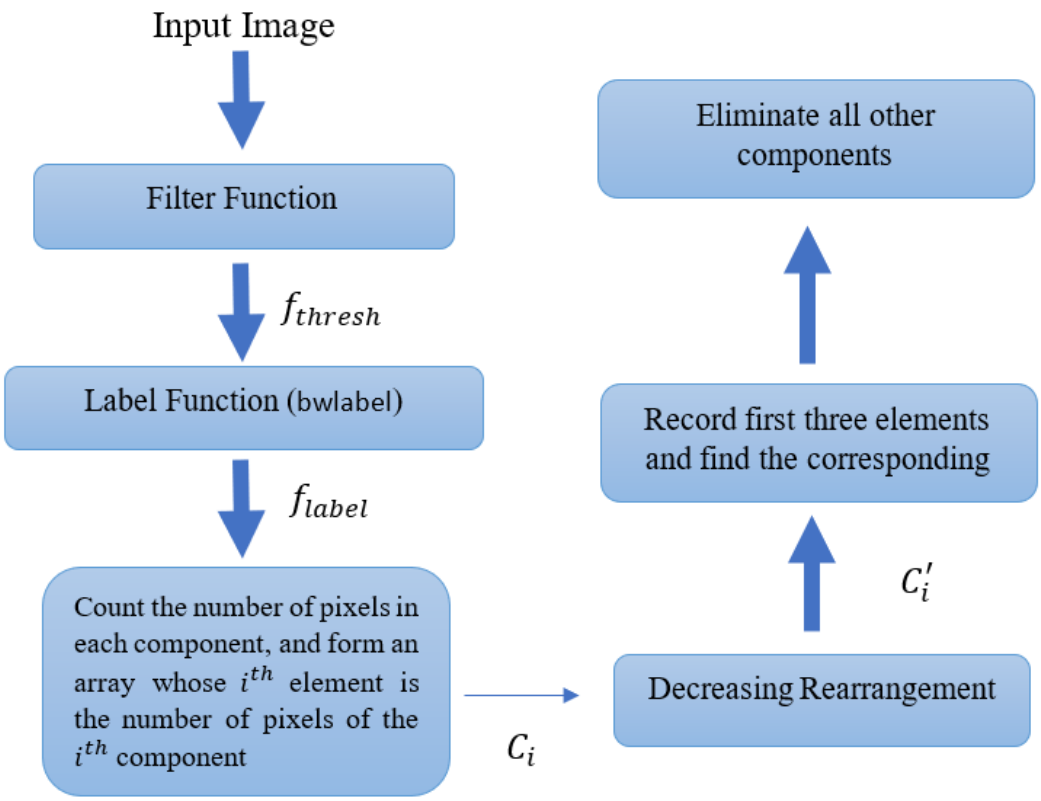

Figure 1. Flowchart of performing the algorithm

b) Explanation of MATLAB Codes

The filtering is realized by setting the threshold value to make sure that there will be more than 5 distinct bright regions obviously appearing on the image. It's absolutely clear that the remaining bright regions will decrease as the threshold value is larger.

$$f_{thresh}(x,y) = \begin{cases} 1\ , & if\ f(x,y) > 150 \\ 0\ , & otherwise \end{cases} \tag{1}$$

The threshold value was set to 150 as shown in equation (1), and it turns out that there are 50 components.

For the next step, all the components should be labeled with a distinct value and three largest components need to be distinguished. To realize this, *bwlabel* function in MATLAB is applied to label the components, as shown in equation (2).

$$f_{label}(x,y) = \begin{cases} 2\ , & if\ f_{thresh}(x,y) \in C_2 \\ 3\ , & if\ f_{thresh}(x,y) \in C_3 \\ & \dots \\ 0\ , & otherwise \end{cases} \tag{2}$$

Then, a for loop is written to count the number of pixels in each component and components are decreasingly sequenced by the size of them by using *sort* and *fliplr* function in MATLAB. Finally, the three largest components could be tracked by using *find* function in MATLAB to find the components corresponding to the sizes.

2): Logical (Set) Operations

Recall the truth table for basic logical operation, as shown in the table 1.

X	Y	AND	OR	XOR	NOT
0	0	0	0	0	1,1
0	1	0	1	1	1,0
1	0	0	1	1	0,1
1	1	1	1	0	0,0

Table 1: Truth Table of Basic Logical Operations

The similar operation can also be performed on the images, for example, the AND operation can be performed as indicated in equation (3).

$$f_{A\cap B}(x,y)=\begin{cases}1, & f_A(x,y)=1\ AND\ f_B(x,y)=1\\ 0, & otherwise\end{cases} \tag{3}$$

Similarly, performing other the logical operations on binary gray-level images could result in the corresponding images and the results of these operations are shown in the next section.

Additionally, the minimum operator can be build upon the idea which is shown by equation (4).

$$f_{\min\,(A,B)}(x,y)=\begin{cases}f_A(x,y), & f_A(x,y)>f_B(x,y)\\ f_B(x,y), & otherwise\end{cases} \tag{4}$$

Results

Part 1:

As the result of selecting 150 as the gray-level threshold, there are 7 distinct "bright" components appear to be visually obvious. The actual number of components is 80 according to MATLAB computation, but most of these 80 components could hardly be observed by human eyes, as shown in figure 1 and figure 2.

Figure 2: Original lenna.gif

Figure 3: lenna.gif after setting gray-level threshold, 80 bright components in this image

Changes are almost invisible after using the function *flable*. After labeling 80 bright components with different gray-level, we could still see same amount of distinct components with similar brightness.

Figure 4: lenna.gif after using flable, 80 bright components labeled with different gray-level

Components are clearly distinguished after using *imshow(fRGB)*. Each component is now labeled with different color. All the other pixels that not in any component is now displayed as white.

Figure 5: lenna.gif after using imshow(fRGB), 80 components with different colors

Among all 80 components, the 3 biggest components were selected and all the pixels in other components were set to 0(black). Only three clear bright components are now showed in the image below.

Figure 6: lenna.gif with only 3 biggest "bright" components

Part 2:

a)

A AND B is the intersection of A and B.
A OR B is the union of A and B.
A XOR B is the intersection of A union B and the complement of A intersection B
NOT A is the complement of A.

b)

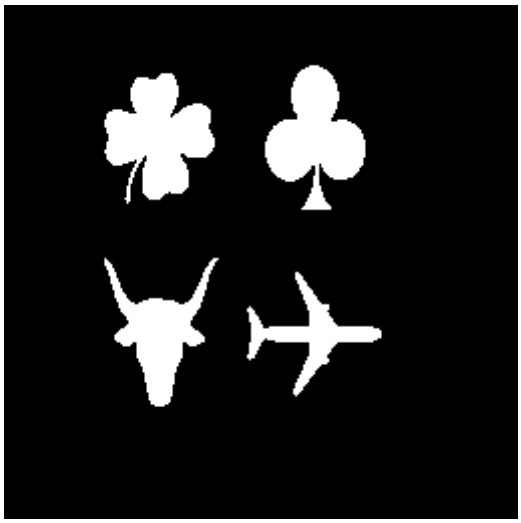

Figure 7: Original Image A (match1)

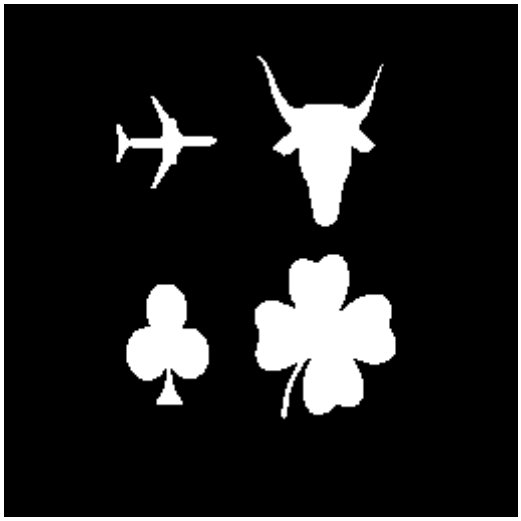

Figure 8: Original Image B (match2)

Figure 9: A(match1) AND B(match2)

This result image only displays white pixel wherever both A and B have white pixels on a specific coordinate. It displays black pixel otherwise.

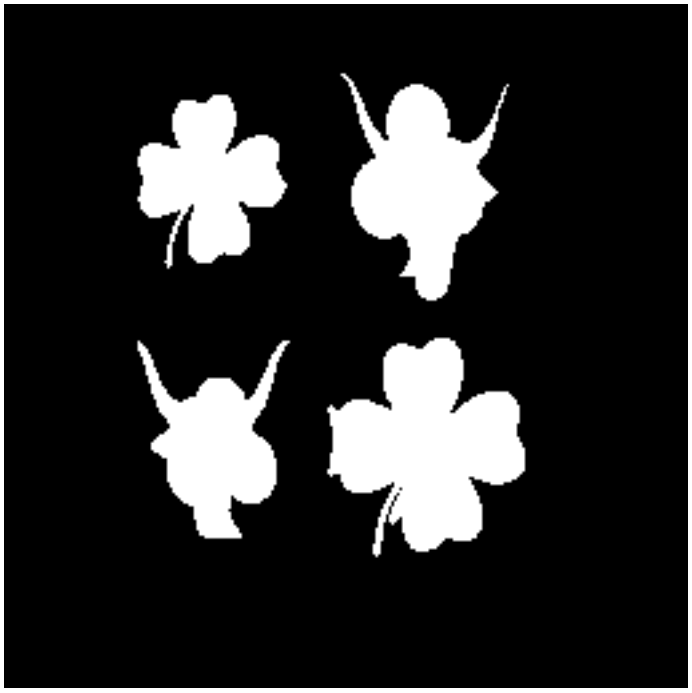

Figure 10: A(match1) OR

This result image displays white pixel wherever either A or B has white pixel on a specific coordinate.

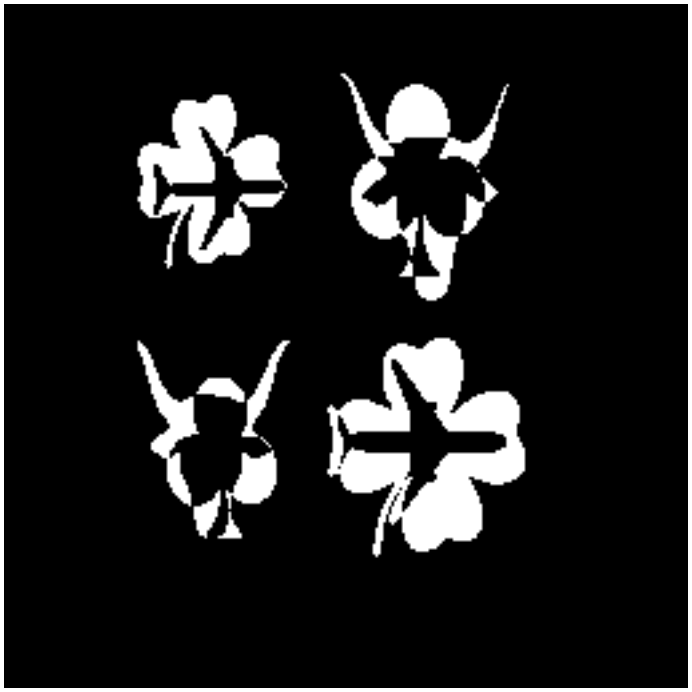

Figure 11: A(match1) XOR B(match2)

This result image only displays black pixels wherever both A and B have white pixels on a specific coordinate. If A and B have different color pixels on a specific coordinate, the result image displays white pixel.

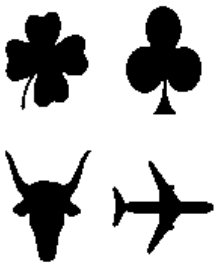

Figure 12: NOT A(match1)

This result image displays black pixel wherever A has white pixel on a coordinate and white pixel wherever A has black pixel on a coordinate.

c)

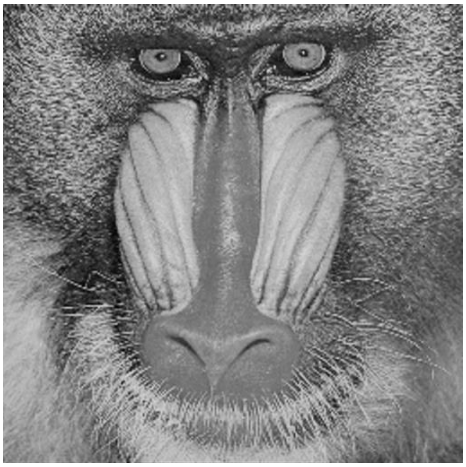

Figure 13: Original Image C

Figure 14: Original Image D (cameraman)

Figure 15: Result image displaying the smaller pixel value of C and D.

This result image displays the pixel of either image C or image D depending on which one has a smaller value on a specific coordinate.

Conclusion

The connect-component labeling is realized by applying pixels filtering followed by flabel function, as it suggested in the result, the size of the components vary from one digit value of pixels to the three digits, and accordingly it can be summarized that the larger the size of the component is, the more it contributes to the visual effect of the image.

Additionally, the logical operations performed on images can also be a significant part of digital image processing. As completed in the project, some binary images are overlapped by applying OR operator, projected upon another by applying AND operator and flipped by applying NOT operator. Moreover, the minimum operator which is also built in the project is able to result in the gray-scale analog of set intersection.