EECS 332 Introduction to Computer Vision

Machine Problem 1: Connected Components Labeling

Name: Runlong Lin   ID:3075255

1. Introduction

In fields of image recognition, it's important to detect or label the isolated regions in an image. There are different types of Connected Components Labeling (CCL) algorithms to solve the problem. One is the recursion solution , which would be time consuming if the size of image is very large. Another efficient algorithm is the sequential solution. It's implemented by scanning each pixel and checking neighbor pixels' labels to determine the current label.

2. Algorithm Description

● **CCL algorithm**

In MP1, I use **Matlab** scripts to implement sequential CCL algorithm.

The core function of CCL algorithm is shown below:

```
function[label_img,num_of_label]=CCL(img)
```

where img is the input binary image, label_img is the output image containing labels, and num_of_label shows the total number of different labels (except the background).

The main steps in **CCl algorithms** can be divided into 2 parts:

■ **First Scanning**

1) We use 2 for-loops to iterate(scan) each pixels in the image. The iterating direction is from left to right, from top to down.
2) We design several iterating conditions. If the pixel is the foreground, we divide the current pixel into 4 cases according to its neighbor pixels' value. Considering our iterating direction, we assign the upper pixel and left pixel as the neighbor of current pixel. In each case, we have different ways to set the value of labels(as the code shows).

```
 %case1: Lu=Ll and both of them are non-zero
if (Lu==Ll) && (Lu~=0)&&(Ll~=0)
    L(u,v)=Lu;
%case2: either Lu or Ll is zero
elseif (Lu~=Ll)&&(~(Lu&&Ll))
    L(u,v)=max(Lu,Ll);
%case3: both of them are non_zero and Lu~=Ll
%in this case, we set up the E-table
%call the recursion_Etable to help us build up Et
elseif ((Lu~=Ll)&&Lu>0&&Ll>0)
    L(u,v)=min(Lu,Ll);
    ET(max(Lu,Ll))=recursion_Etable(ET,min(Lu,Ll));
%case4:Lu=Ll=0, we create a new label
%set the new label in E-tabel
else
    flag=flag+1;
```

```
        L(u,v)=flag;
        ET(flag)=flag;
    end
```

In CCl algorithm, we also build an assist function named recursion_Etable:

```
function[x]=recursion_Etable(arr,x)
    if  arr(x)~=x
        x=arr(x);
        recursion_Etable(arr,x)
    end
end
```

We use recursion to find the equivalent value of a certain label.

3) In certain cases (3 and 4), we set up equivalent table(E-table) by recursion to store labels which correspond to the same region. In case 4, there are no neighbors. So we uniquely label the current pixel and append the label in E-table. In case 3, we find the neighbor with the smallest label and assign it to the current pixel. Meanwhile we store the equivalence relation between two neighbor labels.

- **Second Scanning**

1) After the first scanning, we have labeled all the pixels. In the second scanning, we use the E-table to join the pixel in the same connected region.
2) We iterate each element. If the pixel is the foreground, we replace its label with the value in E-table.
3) Until now, we have successfully labeled most of the pixels. But in some images with "concave" shape, it seems that we can't set all the connected pixel in the same label. It's because the iterating direction is from left to right, from up to down. So if we scan in this way, we can't set Lup's and Lleft's label as equivalent labels. But actually they are equivalent as they are connected. Therefore, in the concave shaped image, we may have two connected components (Fig.1.) if we don't add some conditions in the iterating process.

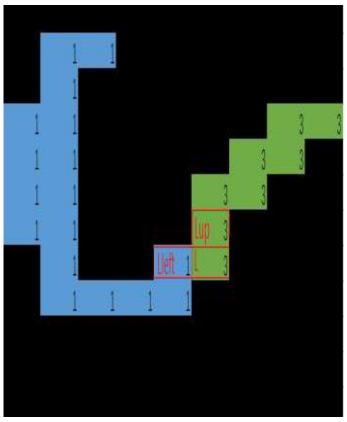

Fig.1. "concave" shaped image

4) So we add the iterating condition to optimize our E-table.

```
% optimize the E-table
for u=1:r-1
    for v=1:c-1
        if L(u,v+1)~=L(u+1,v)&&L(u,v+1)>0&&L(u+1,v)>0
            ET(max(L(u,v+1),L(u+1,v)))=recursion_Etable(ET,min(L(u,
        v+1),L(u+1,v)));
        end
    end
end
```

At this moment, we use the right pixel and the down pixel as the neighbor pixel. In this way, we can find all the equivalent labels.

5) We have the completed E-table. Then repeat step 2 in Second Scanning.

- **Size_filter algorithm**

For binary images with too much noise, we use **size_filter function** to filter the noise:

```
function[filter_img]=size_filter(threshold,img)
```

where img is the input binary image, threshold is the threshold area of the noisy region and filter_img is the output filtered image.

After we use CCL algorithm, we divided the image into several labeled regions. The size_filter function removes the small noise region if the area of the region is smaller than the threshold area.

3. Result Analysis

Here is the result of some labeled images.

- **test.bmg**

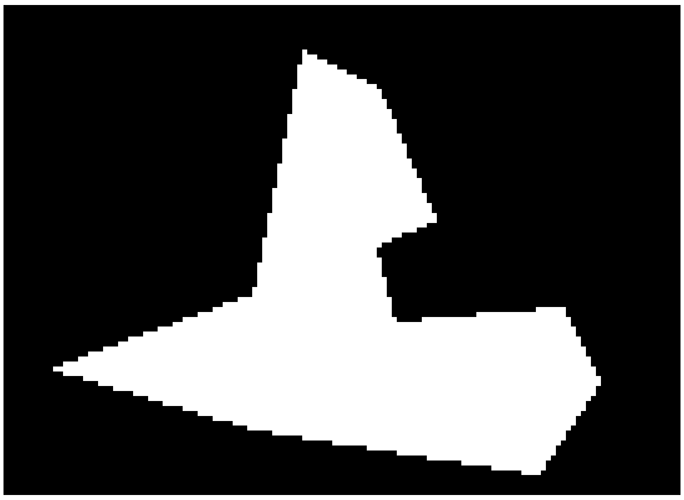

Fig.2. original test.bmg

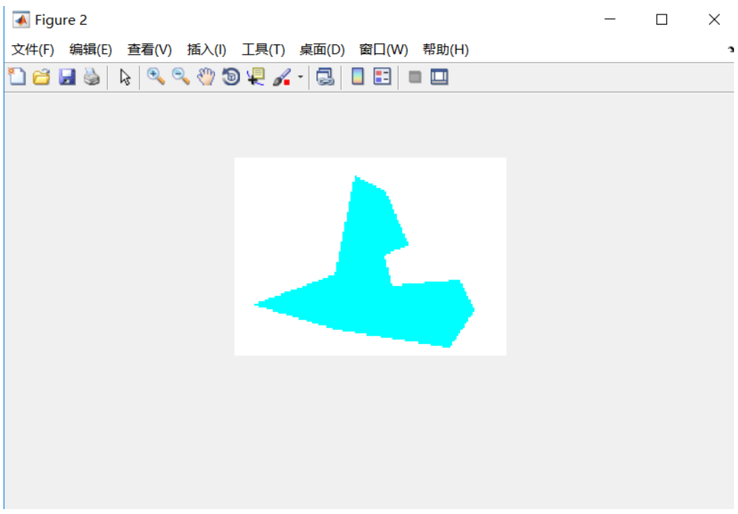
Figure 2
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)

Fig.3. labeled test.bmg

- **face.bmg**

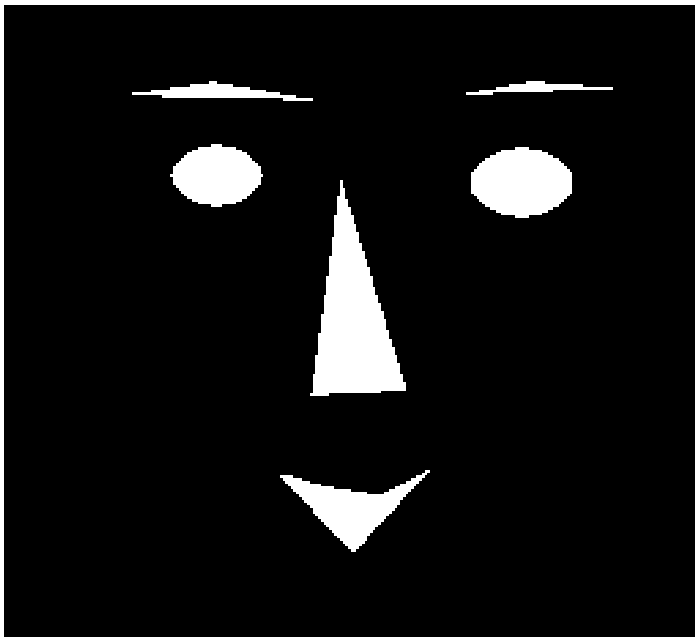
Fig.4. original face.bmg

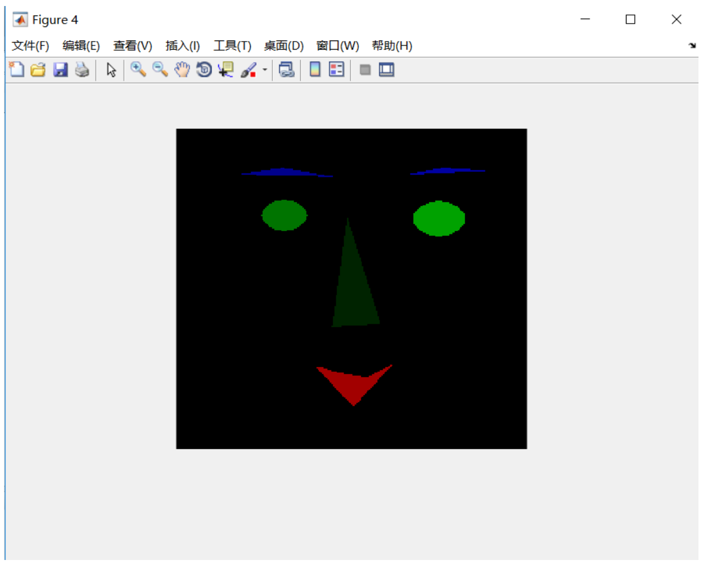
Figure 4
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
Fig.5. labeled face.bmg

- **gun.bmg**

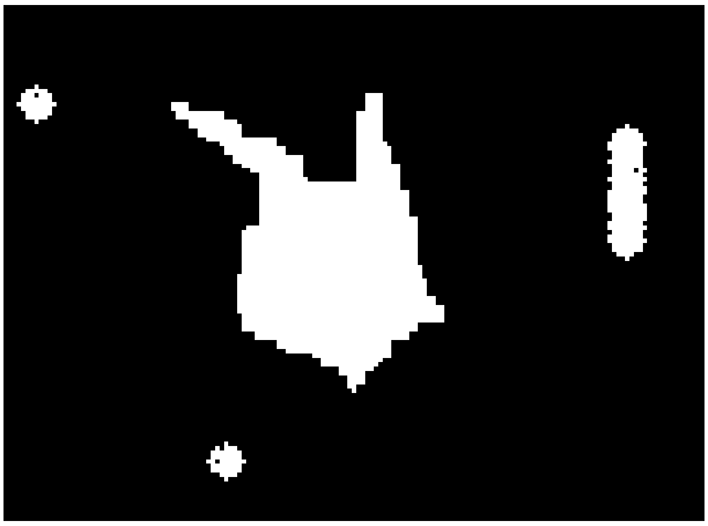

Fig.6. original gun.bmg

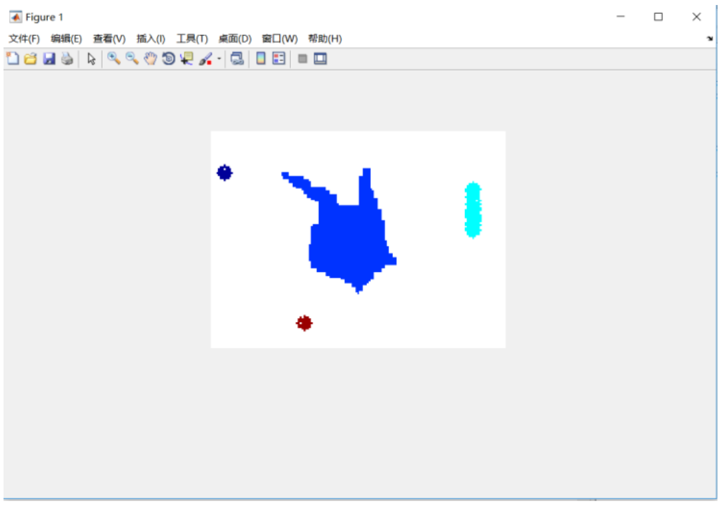
Figure 1
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)

Fig.7. labeled gun.bmg

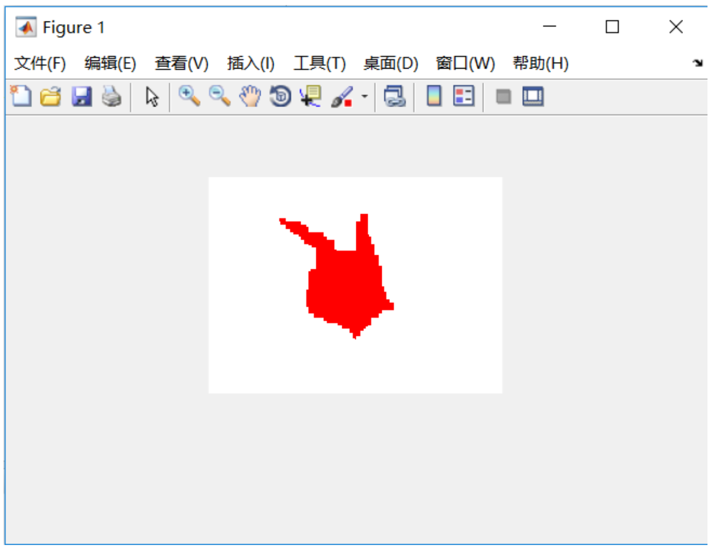
Figure 1
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)

Fig.8. filtered gun.bmg

- **NU.bmg**

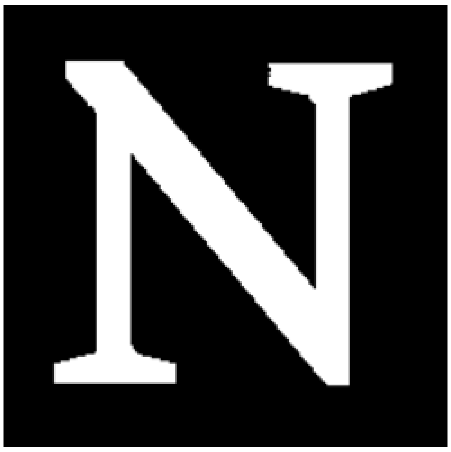
Fig.9. original NU.bmg

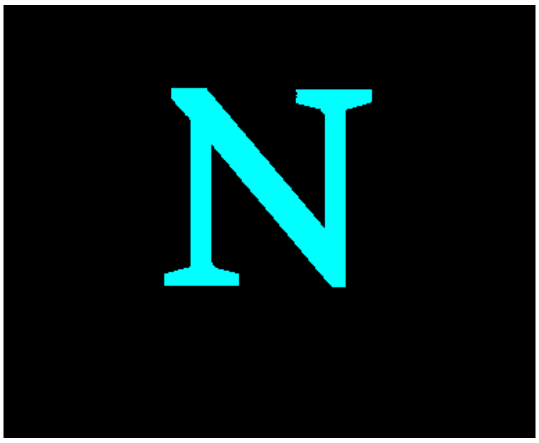
Fig.10. labeled NU.bmg

Analysis:

1) In test.bmg and NU.bmg, there is only one connected components. After we use CCL algorithm, this region can be labeled.

In face.bmg, it has totally 6 connected components. We can detect 6 components by CCL and display them with different color. However, the color of two eyes is similar because the value of their label is very close. Actually we can scale up the value to enlarge the difference between them.

In gun.bmg, after we label the connected components in the image, we use size_filter function to filter the noise and get figure8. Here I set the threshold as 500 roughly.

I also use another image NU.bmg as a test. It seems that the result is within our expectation.

2) In CCL and size_filter function, we use Matlab function **tabulate** to count the unique value in a labeled image:

```
aa=tabulate(img(:));
```

It will return the unique values of img, the number of instances of each value and the percentage of each value. Using tabulate, we can conveniently calculate the total number of labels and the area of each label.

4. Summary

In MP1 of EECS 332, we implement the CCL algorithm to label the connected components in the image by Matlab. The algorithm has passed several tests and worked well. However, we can still improve the algorithm. For example, we can use link list or other data structure to build the equivalent table so that we can save the compiling time. We can also use Python or C++ to implement CCL algorithm.