Computer and Robot Vision

Homework#2

R01944040 柳成蔭

這次的作業是基本的圖像處理，(a) 取 threshold=128 來將 lena 圖變成 binary image，(b)得到 lena 圖的 histogram，(c)得到 binary image 中的 connected components。

我使用 VS2012 編寫程式

(a) binary image

對圖中的每個 pixel 進行判斷，如果值小於等於 127，就設為 0；如果大於 127，就設為 255。

```
for(i=0;i<=imgBinary.rows-1;i++)
{
    for(j=0;j<=imgBinary.cols-1;j++)
    {
        if(imgBinary.at<uchar>(i,j)<=127)
            imgBinary.at<uchar>(i,j)=0;
        else
            imgBinary.at<uchar>(i,j)=255;
    }
}
```

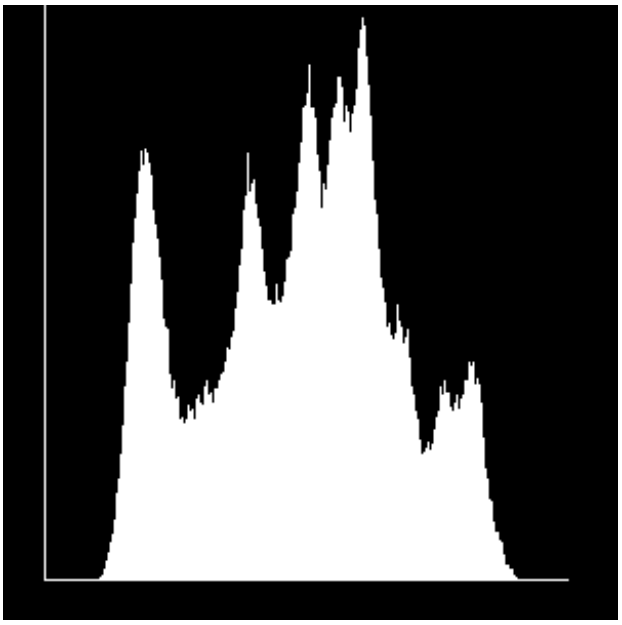
經過設 threshold 之後的結果：

(b) histogram

統計每個值對應的 pixel 的個數，記錄在 histogram[]中。

```
Mat_<uchar>::iterator iteK=img.begin<uchar>(),
                      iteKEnd=img.end<uchar>();
while (iteK!=iteKEnd)
{
    histogram[(*iteK)]++;
    iteK++;
}
```

畫出 histogram：



(c) connected components

我使用 4-connected neighborhood detection。

第一遍掃描將不是 0 的點進行 label。如果是左、上都為 0 的點，直接新增一個 label 到 linked 中。

```
if( (i==0 && j==0)
                    || ( i==0 && label[i][j-1]==0 )
                    || ( label[i-1][j]==0 && j==0 )
                    || ( label[i-1][j]==0 && label[i][j-
1]==0 ) ) //如果上和左是空的
                {
                    //把labelNumber加到linked[labelNumber]
                    newNode.root=0;
                    newNode.nodeLabel=labelNumber;
                    newNode.nodeCount=1;

                    newNode.top=i;
                    newNode.down=i;
                    newNode.left=j;
                    newNode.right=j;

                    linked.push_back(newNode);
                    label[i][j]=labelNumber;
                    labelNumber++;
                }
```

如果是左、上已經有label的點，就要找到左、上各自的root，比較之後歸到同一個set。

```
else
                    {
                        if(label[i-1][j]==0)
                        {
                            label[i][j]=label[i][j-1];
```

```
    linked[ label[i][j] ].nodeCount++;
                        }
                        else if(label[i][j-1]==0)
                        {
                            label[i][j]=label[i-1][j];

    linked[ label[i][j] ].nodeCount++;
                        }
                        else
                        {
                            label[i][j]=label[i-1][j];

    linked[ label[i][j] ].nodeCount++;

                            int p=label[i][j-1];
                            int q=label[i-1][j];

                            while(linked[p].root>0)
                            {
                                p=linked[p].root;
                            }
                            while(linked[q].root>0)
                            {
                                q=linked[q].root;
                            }

                            if(linked[p].nodeLabel >
linked[q].nodeLabel)
                                linked[q].root=p;
                            else if(linked[p].nodeLabel <
linked[q].nodeLabel)
                                linked[p].root=q;
                            else
                            {
                                linked[p].root=0;
                                linked[q].root=0;
                            }
```

```
                    }

                    if(i<linked[ label[i][j] ].top)
                        linked[ label[i][j] ].top=i;
                    if(i>linked[ label[i][j] ].down)
                        linked[ label[i][j] ].down=i;
                    if(j<linked[ label[i][j] ].left)
                        linked[ label[i][j] ].left=j;
                    if(j>linked[ label[i][j] ].right)
                        linked[ label[i][j] ].right=j;
                }
            }
```

之後將屬於同一個 set 的所有 label 的 count 數加總起來，并確定該 set 中上下左右位置值最大的點。如果 pixel 的數量大於 500，就輸出 pixel 的數量，并把連通的區域框出來：

```
size_t sizeLinked=linked.size();
    for (int k = 1; k < sizeLinked ; k++)
    {
        if(linked[k].root!=0)
        {

    linked[linked[k].root].nodeCount=linked[linked[k].root]
.nodeCount + linked[k].nodeCount;

            if(linked[k].top<linked[linked[k].root].top)
                linked[linked[k].root].top=linked[k].top;
            if(linked[k].down>linked[linked[k].root].down)
                linked[linked[k].root].down=linked[k].down;
            if(linked[k].left<linked[linked[k].root].left)
                linked[linked[k].root].left=linked[k].left;
            if(linked[k].right>linked[linked[k].root].right)

    linked[linked[k].root].right=linked[k].right;
        }
    }

    cvtColor(imgLabel,imgConCom,CV_GRAY2RGB);
```

```
    for (int k = 1; k < sizeLinked ; k++)
    {
        if(linked[k].root==0 && linked[k].nodeCount>=500)
        {
            cout << linked[k].nodeCount << endl;
            Point a,b,c;
            a.x=linked[k].left;
            a.y=linked[k].top;
            b.x=linked[k].right;
            b.y=linked[k].down;
            c=a+b;
            c.x=c.x/2;
            c.y=c.y/2;
            circle( imgConCom,c,6,CV_RGB(0,255,0),2 );
            rectangle(imgConCom,a,b,CV_RGB(255,120,120),2);
        }
    }
```

找出 4-connected neighborhood 的 connected components 後的結果：