電腦視覺作業二

Computer Vision HW2

R04525092 工科碩二 鄭力文

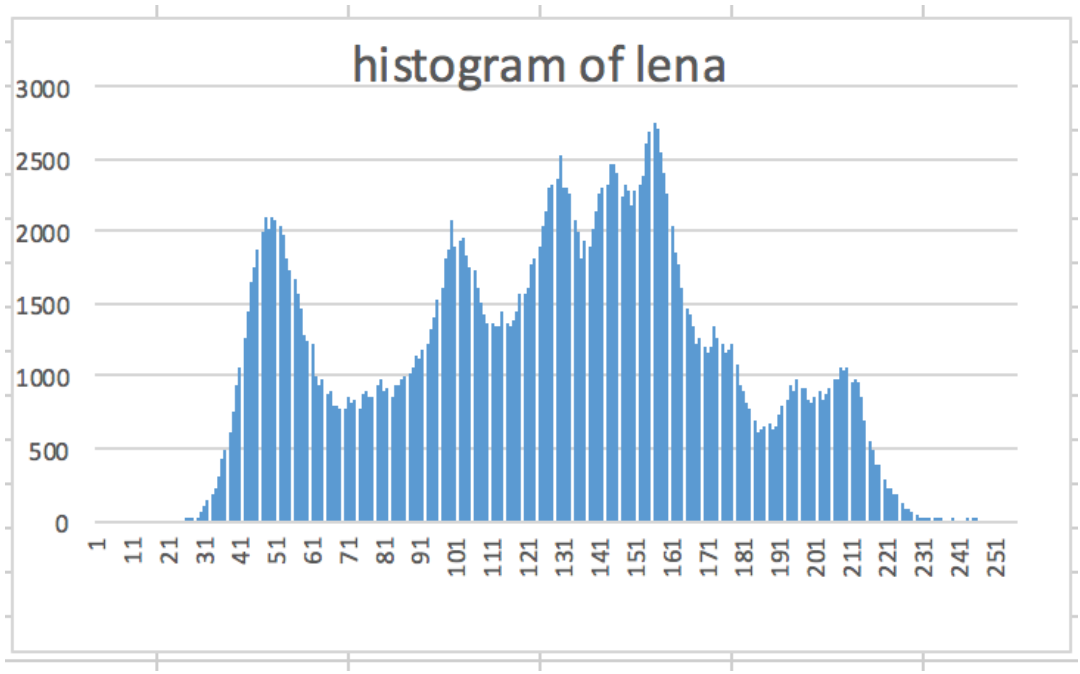
使用語言 C++ with openCV

1. a binary image (threshold at 128)

   二值化取 128

2. a histogram

   使用C + + 輸出 csv，後使用 excel 進行圖表繪製

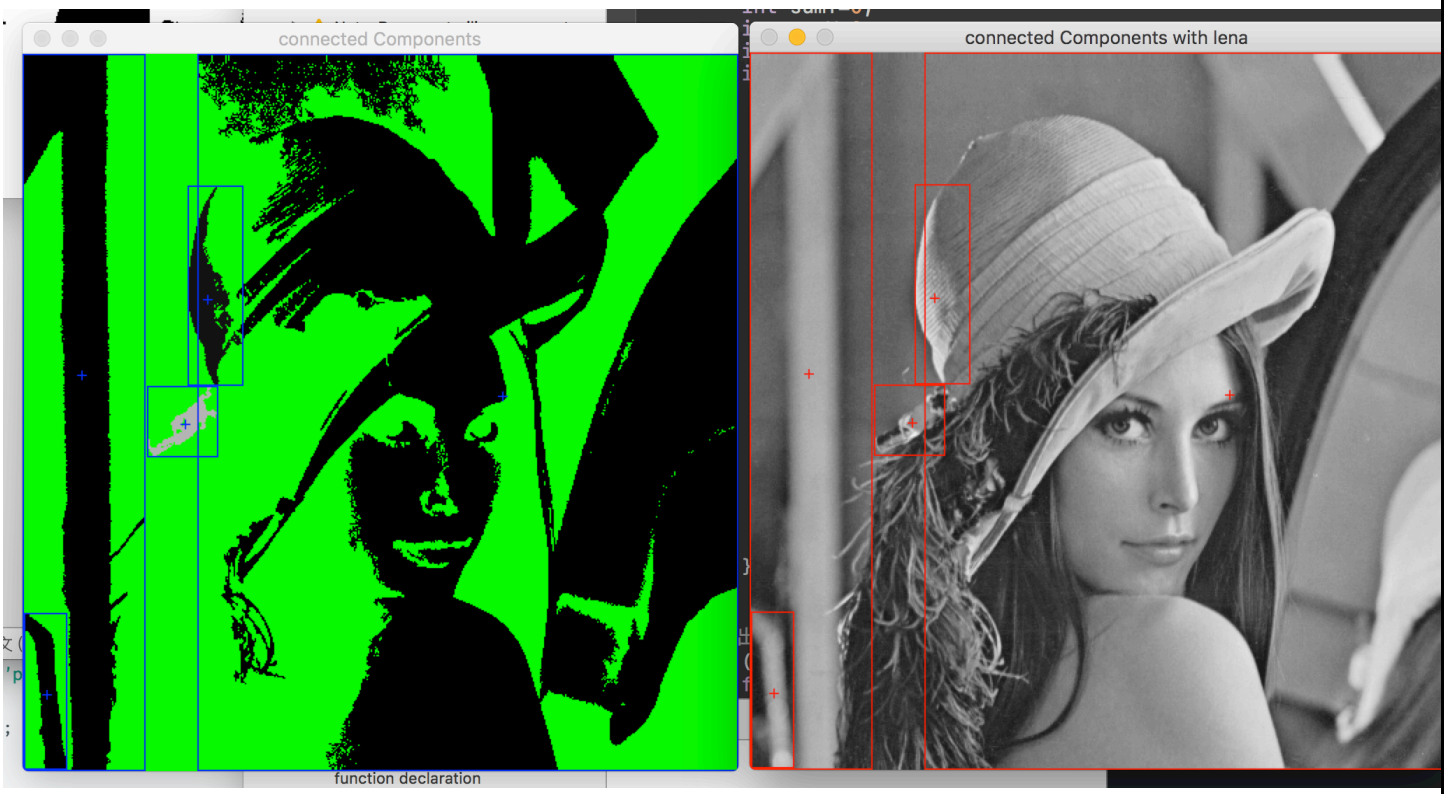


3. connected components (regions with + at centroid, bounding box

使用 The Classical Algorithm + 8-connected

並進算其相鄰同編號點的重心進行繪製中心點

結果



變數

```
//histogram array
int histogram [256]={0};
//讀取lena圖
Mat image = imread("lena.bmp");
//讀取lena圖 的寬高
cv::Size s = image.size();
int rows = s.height;
int cols = s.width;
//2值化 畫布
Mat thresholdingImg = image.clone();
//相鄰元件 畫布
Mat connectedComponents(rows, cols, CV_8UC3);
//相鄰元件+lena 畫布
Mat cpWithLena=image.clone();
//相鄰元件 array
int componetsArray[1024][1024]={0};
//DFS 等價表
int table[3000]={0};
// 數相鄰是否有500個pixel連結的紀錄表
int getCountTable[3000]={0};
```

```
//fuction getNewValue
//input pixel內的灰階值
//output 取2值化thresholding後的直
//取2值化
int getNewValue(int value){
    int newValue=0;
    int thresholding=128;

    if(value>=thresholding)
        newValue=255;

    return newValue;
}
```

```
//fuction drawLine
//input 4點座標
//畫出4點座標外的 bounding box外框圖＋中心點
void drawLine(int x1,int y1,int x2,int y2,int x3,int y3,int x4,int y4){

    int arr_x[]={x1,x2,x3,x4};
    int arr_y[]={y1,y2,y3,y4};

    int left_x=3000;
    int left_y=3000;
    int right_x=0;
    int right_y=0;

    for(int i=0;i<4;i++){
        if(arr_x[i]<left_x)left_x=arr_x[i];
        if(arr_x[i]>right_x)right_x=arr_x[i];
        if(arr_y[i]<left_y)left_y=arr_y[i];
        if(arr_y[i]>right_y)right_y=arr_y[i];
    }

    for(int i=left_x;i<=right_x;i++){
        componetsArray[i][left_y]=-1;
        componetsArray[i][right_y]=-1;
    }
    for(int i=left_y;i<=right_y;i++){
        componetsArray[left_x][i]=-1;
        componetsArray[right_x][i]=-1;
    }

    //畫中心
    for(int i=-3;i<=3;i++){
        componetsArray[left_x+((right_x-left_x)/2)+i][left_y+((right_y-left_y)/2)]=-1;
    }
    for(int i=-3;i<=3;i++){
        componetsArray[left_x+((right_x-left_x)/2)][left_y+((right_y-left_y)/2)+i]=-1;
    }

}
```

```
//fuction getCount
//數相鄰點的數量並紀錄進getCountTable
//為了<500不計算
void getCount(){
    int n=0;
    for(int i=0;i<3000;i++){
     n=0;
        for (int y=0;y<cols;y++){
            for(int x=0;x<rows;x++){
                if(componetsArray[x][y]==i){
                    n++;
                }
            }
        }
        getCountTable[i]=n;
    }

}
```

```
//fuction checkTable
//輸入 classical 演算法 遇衝突 pixel內編號的值
// a<b
// 將衝突的點進行紀錄並且最佳化以便於對表
void checkTable(int a,int b){
    if(table[a]==0&&table[b]==0){
        table[b]=a;
        table[a]=a;
    }
    else if(table[a]==0&&table[b]!=0){
        if(table[b]>a){

            for(int i=0;i<3000;i++){
                if(table[i]==table[b])
                    table[i]=a;
            }
            table[b]=a;
            table[a]=a;
        }
        else {
            table[a]=table[b];
        }
    }
    else if(table[a]!=0&&table[b]==0){
        table[b]=table[a];

    }
    else if(table[a]!=0&&table[b]!=0){

        for(int i=0;i<3000;i++){
            if(table[i]==table[b])
                table[i]=table[a];
        }
         table[b]=table[a];
    }

}
```

```
//function getBeside
//input x值 y值
//檢查隔壁的數字 上與左 topdown下來後 是否有值進行運算
//ruten 值或是 0
int getBeside( int x, int y){
        //上而下
        if(x==0&&y==0){
            return 0;
        }
        else if(x>0&&y==0){
            if(componetsArray[x-1][y]!=0){
                return componetsArray[x-1][y];
            }
            else{
                return 0;
            }
        }
        else if(x==0&&y>0){
            if(componetsArray[x][y-1]!=0)
                return componetsArray[x][y-1];
            else{
                return 0;
            }
        }
        else if(x>0&&y>0)
        {
            if(componetsArray[x-1][y]!=0 && componetsArray[x][y-1]!=0){

                if(componetsArray[x-1][y] == componetsArray[x][y-1]){

                    return componetsArray[x][y-1];

                }
                else{


                    if(componetsArray[x][y-1]>componetsArray[x-1][y])
                    {
                        checkTable(componetsArray[x-1][y],componetsArray[x][y-1]);
```

```
                    }
                    else{
                        checkTable(componetsArray[x][y-1],componetsArray[x-1][y]);
                    }
                    return componetsArray[x-1][y];
                }

            }

            else if(componetsArray[x][y-1]!=0)
                return componetsArray[x][y-1];
            else if(componetsArray[x-1][y]!=0)
                return componetsArray[x-1][y];
            else if(componetsArray[x-1][y-1]!=0){
                return componetsArray[x-1][y-1];
            }
            else{
                return 0;
            }
        }
        else{
            return 0;
        }

    return 0;
}
```

```
//主函式
int main(int argc, const char * argv[]) {
    cout<<"R04525092 工科 鄭力文"<<endl;
    //取2值化 thresholding image
    for (int y=0;y<cols;y++){
        for(int x=0;x<rows;x++){
            histogram[thresholdingImg.at<Vec3b>(x,y)[0]]+=1;
            for(int z=0;z<3;z++){
              thresholdingImg.at<Vec3b>(x,y)[z] =getNewValue(thresholdingImg.at<Vec3b>(x,y)[z]);
            }
        }
    }
    //初始化
    for (int y=0;y<cols;y++){
        for(int x=0;x<rows;x++){
            if(thresholdingImg.at<Vec3b>(x,y)[0]==255){
                componetsArray[x][y]=1;
            }
            else{
                componetsArray[x][y]=0;
            }
        }
    }

    int cpIndex=1;
    //top down運算 classial 演算法
        for (int y=0;y<cols;y++){
            for(int x=0;x<rows;x++){
                if(componetsArray[x][y]==0){
                     componetsArray[x][y]=0;
                }
                else if(getBeside(x,y)!=0){
                    componetsArray[x][y]=getBeside(x,y);

                }
                else{
                    componetsArray[x][y]=cpIndex;
                    cpIndex++;
                }
           }
        }
    // 對表
    for (int y=0;y<cols;y++){
        for(int x=0;x<rows;x++){
            if(componetsArray[x][y]!=0){
                componetsArray[x][y]=table[componetsArray[x][y]];
```

```
    //做出邊界圖
    for (int y=0;y<cols;y++){
        for(int x=0;x<rows;x++){
            for(int z=0;z<3;z++){

                if(componetsArray[x][y]==-1){
                    connectedComponents.at<Vec3b>(x,y)[0]=255;
                    cpWithLena.at<Vec3b>(x,y)[0]=0;
                    cpWithLena.at<Vec3b>(x,y)[1]=0;
                    cpWithLena.at<Vec3b>(x,y)[2]=255;

                }
                else if(componetsArray[x][y]!=0){
                    connectedComponents.at<Vec3b>(x,y)[z] =(componetsArray[x][y])%255;

                }
                else{
                    connectedComponents.at<Vec3b>(x,y)[1]=255;
                }
            }

        }
    }


    //顯示結果
    namedWindow("thresholding", WINDOW_AUTOSIZE);
    imshow("thresholding", thresholdingImg);
    namedWindow("connected Components", WINDOW_AUTOSIZE);
    imshow("connected Components",  connectedComponents);
    namedWindow("connected Components with lena", WINDOW_AUTOSIZE);
    imshow("connected Components with lena",  cpWithLena);
    waitKey(0);

    return 0;
}
```

```
        for(int x=0;x<rows;x++){
            if(componetsArray[x][y]!=0){
                componetsArray[x][y]=table[componetsArray[x][y]];
            }
         }
    }
    //數相鄰的數量
    getCount();
    //將小於500的相鄰數量元去除
    for (int y=0;y<cols;y++){
        for(int x=0;x<rows;x++){
            if(componetsArray[x][y]!=0&&getCountTable[componetsArray[x][y]]<500){
                componetsArray[x][y]=0;
            }

        }
    }
    //2值化 輸出csv
    ifstream fsInput;
    fsInput.open("histogram.csv");
    ofstream fsOutput("histogram.csv", ios::app);

    for(int i=0;i<256;i++){
        //cout<<i<<" "<<histogram[i]<<endl;
        fsOutput <<i<<","<<histogram[i] << endl;
    }
    fsOutput.close();
    //畫邊線圖
    for(int i=1;i<3000;i++){
        if(getCountTable[i]>500){
            int minX=999999,minX_y=0;
            int minY=999999,minY_x=0;
            int maxX=0,maxX_y=0;
            int maxY=0,maxY_x=0;
            for (int y=0;y<cols;y++){
                for(int x=0;x<rows;x++){
                    if(componetsArray[x][y]==i){
                        if(x<minX){minX=x;minX_y=y;}
                        if(x>maxX){maxX=x;maxX_y=y;}
                        if(y<minY){minY=y;minY_x=x;}
                        if(y>maxY){maxY=y;maxY_x=x;}


                    }
                }
            }
            drawLine(minX,minX_y,minY_x,minY,maxX,maxX_y,maxY_x,maxY);

        }
    }
```