R05922096 資工所 碩 1 李哲安

Computer Vision Hw2 Report

- Discription
    - Binarize Lena with the threshold 128
    - draw the histogram
    - connected components with

      threshood 500, 4-connected component
- Algorithm
    - Binarize - 取得 pixel 灰階，若>threshood 則設為黑

      否則白色。
    - Histogram – 遍歷 pixel，計算每個灰階出現的次數。
    - connected components – classical algorithm with

      threshood 500, 4-connected component
- Parameters (if any)
    - no

- Principal Code Fragment

  Binarize

  (file – /src/cv1.util.cv/ImgUtil.java)

```java
public static BufferedImage imgBinarize(BufferedImage bi ,int threshold){
    BufferedImage source= toGrayImage(bi);

    BufferedImage result = new BufferedImage(source.getHeight(), source.getWidth(), source.getType());
    for (int y = 0; y < source.getHeight(); y++) {
        for (int x = 0; x < source.getWidth() ; x++) {
            int rgb=source.getRGB(x, y);
            int r= rgb&0xff;
            //System.out.printf("%x ",rgb);
            int gray=r;
            int newBinarizeValue = gray>=threshold?0xffffffff:0xff000000;
            result.setRGB(x , y , newBinarizeValue);
        }
    }
    return result;
}
```

  Histogram

  (file – /src/cv1.util.cv/ImgUtil.java)

```java
public static int[] getImgHistogramMatrix(BufferedImage bi ){
    /*
     * return image binarize histogram matrix
     */
    BufferedImage source = toGrayImage(bi);
    int result[] = new int[256];
    for (int y = 0; y < source.getHeight(); y++) {
        for (int x = 0; x < source.getWidth() ; x++) {
            int gray=source.getRGB(x, y)&0xff;
            result[gray]++;
        }
    }

    return result;
}
```

Connected Components

(file – /src/cv1.util.cv.cclabeling/ClassicalAlgorithm.java)

Step1: find connected componet by classical algorithm.

and union set by union-find algorithm

Setp2: relabeling after union all set

and check threshood

Step3: output component , bounding box and centroid

```
public ClassicalAlgorithm(int binaryImgMatrix[][] , int boxThreshold , int componentType){
    this.componentType=componentType;
    this.boxThreshold=boxThreshold;
    this.binaryImgMatix = binaryImgMatrix;
    this.labelMatrix=new int[binaryImgMatrix.length][binaryImgMatrix[0].length];

    this.labelNum=0;
    this.parent=new ArrayList<Integer>();
    this.setCount=new ArrayList<Integer>();
    parent.add(0);
    setCount.add(0);

    findConnectedComponet();
    relabeling();
    createComponent();
    findBoundingBox();
}
```

```
private void findConnectedComponet(){
    for (int y = 0; y < binaryImgMatix.length; y++) {
        for (int x = 0; x < binaryImgMatix[0].length; x++) {
            if (binaryImgMatix[y][x]==1) {
                checkNeighbor(x,y);
            }
        }
    }
}
```

```
private void relabeling() {
    for (int y = 0; y < binaryImgMatix.length; y++) {
        for (int x = 0; x < binaryImgMatix[0].length; x++) {
            labelMatrix[y][x]=find(labelMatrix[y][x]);
            if (setCount.get(labelMatrix[y][x]) < boxThreshold) {
                labelMatrix[y][x]=0;
            }
        }
    }
}
```

```
private void createComponent(){
    biComponent = new BufferedImage(labelMatrix.length, labelMatrix[0].length, BufferedImage.TYPE_INT_ARGB);
    for (int y = 0; y < labelMatrix.length; y++) {
        for (int x = 0; x < labelMatrix[0].length; x++) {
            if (labelMatrix[y][x]!=0) {
                biComponent.setRGB(x, y, 0xffffffff);
            }else {
                biComponent.setRGB(x, y, 0xff000000);
            }
        }
    }
}
```

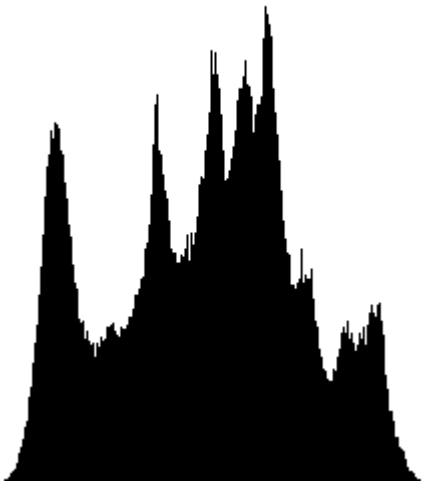
```
private void findBoundingBox(){
    boundingBox=new ArrayList<int[]>();
    HashMap<Integer, int[]> setRect = new HashMap<Integer, int[]>();
    for (int y = 0; y < labelMatrix.length; y++) {
        for (int x = 0; x < labelMatrix[0].length; x++) {
            if (labelMatrix[y][x]!=0 ) {
                if (!setRect.containsKey(labelMatrix[y][x])) {
                    setRect.put(labelMatrix[y][x], new int[]{x,y,x,y,x,y,1});
                }else {
                    int temp[] = setRect.get(labelMatrix[y][x]);
                    temp[0]=Math.min(temp[0], x);//startX
                    temp[1]=Math.min(temp[1], y);//startY
                    temp[2]=Math.max(temp[2], x);//endX
                    temp[3]=Math.max(temp[3], y);//endY
                    temp[4]+=x;//centX
                    temp[5]+=y;//centY
                    temp[6]++;//pixel number
                    setRect.put(labelMatrix[y][x], temp);
                }
            }
        }
    }
    for (int i :setRect.keySet()) {
        int tempBox[] = setRect.get(i);
        tempBox[4]/=tempBox[6];
        tempBox[5]/=tempBox[6];
        boundingBox.add(setRect.get(i));
    }

}
```

- Resulting Images

Binarize

Histogram



Connected Conponent (threshood = 500 ,4-connected)