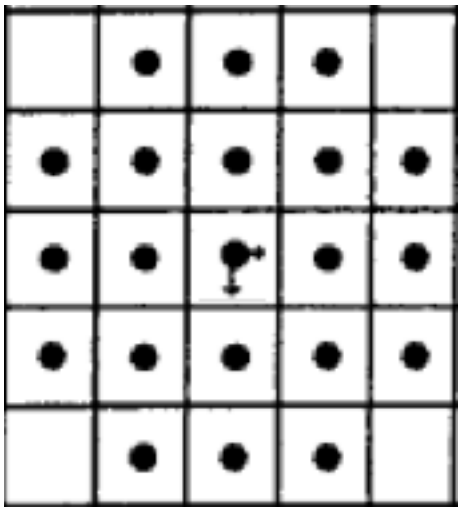
R05922096 資工所 碩 1 李哲安

Computer Vision Hw5 Report

- Discription
    - Mathematical Morphology – Gray Scaled Morphology
- Algorithm
    - Kernel of dilation, erosion, opening and closing

      Value = 0



- Parameters (if any)
    - no

- Principal Code Fragment

Main

(file – /src/hw5/ DemoGrayLevelMorphology.java)

```
//read image
System.out.println("reading img ...");
BufferedImage lena = FileUtil.readImg(inputFolder+inputFileName);
lena = ImgUtil.toGrayImage(lena);
//ImgUtil.showImg(lena, "init");

// dilation
System.out.println("image dilation ...");
BufferedImage dilImg = GrayLevelMorphology.dilation(lena, "3-5-5-5-3", 0);
//ImgUtil.showImg(dilImg, "dilation");

// erosion
System.out.println("image erosion ...");
BufferedImage eroImg = GrayLevelMorphology.erosion(lena, "3-5-5-5-3", 0);
//ImgUtil.showImg(eroImg, "erosion");

// opening
System.out.println("image opening ...");
BufferedImage openImg = GrayLevelMorphology.opening(lena, "3-5-5-5-3", 0);
//ImgUtil.showImg(openImg, "opening");

// closing
System.out.println("image closing ...");
BufferedImage closImg = GrayLevelMorphology.closing(lena, "3-5-5-5-3", 0);
//ImgUtil.showImg(closImg, "closing");
```

Dilation

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

```
public static BufferedImage dilation(BufferedImage bi, String kernelShape, int kernelValue){
    ArrayList<int[]> kernelLogic = getKernelLogic(kernelShape, kernelValue);
    BufferedImage source = bi;
    BufferedImage result = new BufferedImage(bi.getHeight(), bi.getWidth(), bi.getType());

    for (int y = 0; y < source.getHeight(); y++) {
        for (int x = 0; x < source.getWidth() ; x++) {
            int maxGrayValue = Integer.MIN_VALUE;

                for (int logic[] : kernelLogic) {
                    int checkX = x + logic[0];
                    int checkY = y + logic[1];
                    try {
                        int checkGrayValue = source.getRGB(checkX, checkY)&0xff;
                        maxGrayValue = Math.max(maxGrayValue, checkGrayValue);
                    } catch (Exception e) {
                        //ignore
                    }
                }
            result.setRGB(x, y, 0xff000000 + (maxGrayValue<<16) + (maxGrayValue<<8) + maxGrayValue);
        }
    }
    return result;
}
```

Erosion

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

```
public static BufferedImage erosion(BufferedImage bi, String kernelShape, int kernelValue){
    ArrayList<int[]> kernelLogic = getKernelLogic(kernelShape, kernelValue);
    BufferedImage source = bi;
    BufferedImage result = new BufferedImage(bi.getHeight(), bi.getWidth(), bi.getType());

    for (int y = 0; y < source.getHeight(); y++) {
        for (int x = 0; x < source.getWidth() ; x++) {
            int minGrayValue = Integer.MAX_VALUE;

            for (int logic[] : kernelLogic) {
                int checkX = x + logic[0];
                int checkY = y + logic[1];
                try {
                    int checkGrayValue = source.getRGB(checkX, checkY)&0xff;
                    minGrayValue = Math.min(minGrayValue, checkGrayValue);
                } catch (Exception e) {
                    //ignore
                }
            }
            result.setRGB(x, y, 0xff000000 + (minGrayValue<<16) + (minGrayValue<<8) + minGrayValue);
        }
    }
    return result;
}
```

Opening and Closing

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

```
public static BufferedImage opening(BufferedImage bi, String kernelShape, int kernelValue){
    return dilation(erosion(bi, kernelShape, kernelValue), kernelShape, kernelValue);
}
public static BufferedImage closing(BufferedImage bi, String kernelShape, int kernelValue){
    return erosion(dilation(bi, kernelShape, kernelValue), kernelShape, kernelValue);
}
```

- Result Image

Dilation

Erosion

Opening

Closing