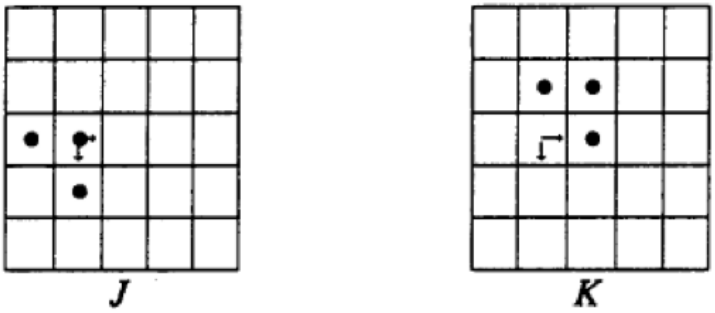
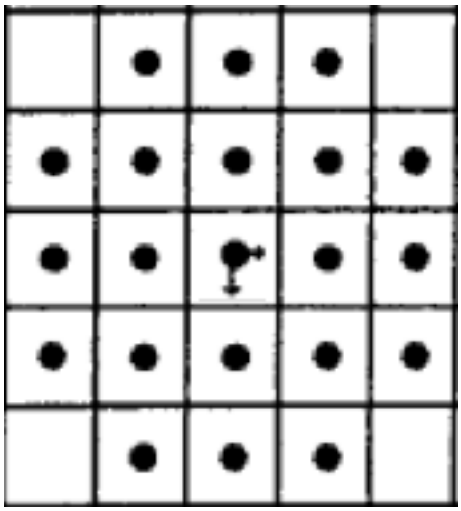
R05922096 資工所 碩 1 李哲安

Computer Vision Hw4 Report

- Discription
    - Mathematical Morphology - Binary Morphology
- Algorithm
    - Kernel of Hit and Miss



    - Kernel of dilation, erosion, opening and closing



- Parameters (if any)
    - no

- Principal Code Fragment

Main

(file – /src/hw4/ DemoBinaryMorphology.java)

```
public static void main(String[] args) {
    //read image
    System.out.println("reading img ...");
    BufferedImage lena = FileUtil.readImg(inputFolder+inputFileName);
    //ImgUtil.showImg(lena, "init");

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

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

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

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

    // Hig and Miss
    System.out.println("image Hit and Miss ...");
    BufferedImage hamImg = BinaryMorphology.hitAndMiss(lena, "L", "~L");
    //ImgUtil.showImg(hamImg, "hamImg");
```

Dilation

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

```
public static BufferedImage dilation(BufferedImage bi, String kernelShape){
    ArrayList<int[]> kernelLogic = getKernelLogik(kernelShape);
    BufferedImage source = ImgUtil.imgBinarize(bi, 128);
    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 binaryValue = source.getRGB(x, y)&0xff;
            if (binaryValue == 255) {
                for (int logic[] : kernelLogic) {
                    int checkX = x + logic[0];
                    int checkY = y + logic[1];
                    try {
                        result.setRGB(checkX, checkY, 0xffffffff);
                    } catch (Exception e) {
                        //ignore
                    }
                }
            }
        }
    }
    return result;
}
```

Erosion

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

```
public static BufferedImage erosion(BufferedImage bi, String kernelShape){
    ArrayList<int[]> kernelLogic = getKernelLogik(kernelShape);
    BufferedImage source = ImgUtil.imgBinarize(bi, 128);
    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++) {
            boolean flag = true ;
            for (int logic[] : kernelLogic) {
                int checkX = x + logic[0];
                int checkY = y + logic[1];
                try {
                    int binaryValue = source.getRGB(checkX, checkY)&0xff;
                    if (binaryValue!=255) {
                        flag = false; break;
                    }
                } catch (Exception e) {
                    flag = false; break;
                }
            }
            if(flag)result.setRGB(x, y, 0xffffffff);
        }
    }
    return result;
}
```

Opening and Closing

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

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

Hit and Miss

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

```
public static BufferedImage hitAndMiss(BufferedImage bi, String kernelShape1, String kernelShape2){
    BufferedImage source = ImgUtil.imgBinarize(bi, 128);
    BufferedImage source_c = new BufferedImage(source.getHeight(), source.getWidth(), source.getType());
    BufferedImage result1 = new BufferedImage(bi.getHeight(), bi.getWidth(), bi.getType());
    BufferedImage result2 = new BufferedImage(bi.getHeight(), bi.getWidth(), bi.getType());
    BufferedImage finalResult = new BufferedImage(bi.getHeight(), bi.getWidth(), bi.getType());
    /* calculate complement */
    for (int y = 0; y < source.getHeight(); y++) {
        for (int x = 0; x < source.getWidth() ; x++) {
            int rgb=source.getRGB(x, y);
            int binaryValue = rgb&0xff;
            if (binaryValue == 255) {
                source_c.setRGB(x, y, 0xff000000);
            }else{
                source_c.setRGB(x, y, 0xffffffff);
            }
        }
    }
    /* erosion */
    result1 = erosion(source, kernelShape1);
    result2 = erosion(source_c, kernelShape2);
    /* intersection */
    for (int y = 0; y < finalResult.getHeight(); y++) {
        for (int x = 0; x < finalResult.getWidth() ; x++) {
            int binaryValue1 = result1.getRGB(x, y)&0xff;
            int binaryValue2 = result2.getRGB(x, y)&0xff;
            if (binaryValue1 == 255 && binaryValue2 == 255) {
                finalResult.setRGB(x, y, 0xffffffff);
            }
        }
    }
    return finalResult;
}
```

- Result Image

Dilation

Erosion

Opening

Closing

Hit and Miss