R05922096 資工所 碩 1 李哲安

Computer Vision Hw9 Report

- Discription
    - General Edge Detection
- Algorithm
    - Robert, Prewitt, Sobel, Robinson, Frei & Chen, Kirsch , and Nevatia-Babu's edge detectors.
- Parameters (if any)
    - no
- Principal Code Fragment

  Main (file – /src/hw9/ DemoGeneralEdgeDetection.java)

```
public static void main(String[] args) {
    //read image
    System.out.println("reading img ...");
    BufferedImage lena = FileUtil.readImg(inputFolder+inputFileName);
    lena = ImgUtil.toGrayImage(lena);
    //ImgUtil.showImg(lena, "lena");
    BufferedImage robert = GeneralEdgeDetector.operate(lena, 30, MaskName.Robert);
    BufferedImage prewitt = GeneralEdgeDetector.operate(lena, 70, MaskName.Prewitt);
    BufferedImage sobel = GeneralEdgeDetector.operate(lena, 100, MaskName.Sobel);
    BufferedImage robinson = GeneralEdgeDetector.operate(lena, 100, MaskName.Robinson);
    BufferedImage freiAndChen = GeneralEdgeDetector.operate(lena, 80, MaskName.FreiAndChen);
    BufferedImage kirsch = GeneralEdgeDetector.operate(lena, 300, MaskName.Kirsch);
    BufferedImage nevatia = GeneralEdgeDetector.operate(lena, 25000, MaskName.Nevatia);
```

  GeneralEdgeDetector

  (file – /src/cv1.util.cv.edge /GeneralEdgeDetector.java)

```
public static BufferedImage operate(BufferedImage bi, int threshold, GeneralEdgeDetectorMasks.MaskName maskName){
    ArrayList<Mask> maskList = GeneralEdgeDetectorMasks.getMasksList(maskName);
    if (maskName == MaskName.Kirsch || maskName == MaskName.Robinson || maskName == MaskName.Nevatia) {
        return maxOperation(bi, threshold, maskList);
    }
    return generalOperation(bi, threshold, maskList);
}
```

```java
private static BufferedImage generalOperation(BufferedImage bi, int threshold, ArrayList<Mask> maskList){
    BufferedImage result = new BufferedImage(bi.getWidth(), bi.getHeight(), bi.getType());

    for (int y = 0; y < bi.getHeight(); y++) {
        for (int x = 0; x < bi.getWidth() ; x++) {
            double gradientMagnit = 0.0;
            for(Mask mask : maskList){
                double maskWeightValue = 0.0;
                for (MaskLogic logic : mask.logics) {
                    try {
                        int gray = bi.getRGB(x + logic.x, y + logic.y) & 0xff;
                        maskWeightValue += gray * logic.w;
                    } catch (Exception e) {
                        // Ignore out of bound
                    }
                }
                gradientMagnit += maskWeightValue * maskWeightValue;
            }
            gradientMagnit = Math.sqrt(gradientMagnit);
            int newGray = gradientMagnit >= threshold? 0: 255;
            result.setRGB(x, y, 0xff000000 + (newGray<<16) + (newGray<<8) + (newGray));
        }
    }
private static BufferedImage maxOperation(BufferedImage bi, int threshold, ArrayList<Mask> maskList){
    BufferedImage result = new BufferedImage(bi.getWidth(), bi.getHeight(), bi.getType());
    for (int y = 0; y < bi.getHeight(); y++) {
        for (int x = 0; x < bi.getWidth() ; x++) {
            double gradientMagnit = -Double.MAX_VALUE;
            for(Mask mask : maskList){
                double maskWeightValue = 0.0;
                for (MaskLogic logic : mask.logics) {
                    try {
                        int gray = bi.getRGB(x + logic.x, y + logic.y) & 0xff;
                        maskWeightValue += gray * logic.w;
                    } catch (Exception e) {
                        // Ignore out of bound
                    }
                }
                gradientMagnit = Math.max(gradientMagnit, maskWeightValue);
            }
            int newGray = gradientMagnit >= threshold? 0: 255;
            result.setRGB(x, y, 0xff000000 + (newGray<<16) + (newGray<<8) + (newGray));
        }
    }
    return result;
}
```

- Result Image

Robert (threshold = 30)

Prewitt (threshold = 70)

Sobel (threshold = 100)

Robinson (threshold = 100)

Frei & Chen (threshold = 80)

Kirsch (threshold = 300)

Nevatia-Babu's (threshold = 25000)