Übungsaufgaben IV, SBV1

Lukas Fiel, Lisa Panholzer

December 27, 2018

4 Übungsaufgaben IV

4.1 Region Growing

a) Manuelles Image Growing

Der Algorithmus zu dieser Übung wurde aus der Vorlesung übernommen. Es waren lediglisch N4 und N8 Nachbarpixelregionen zu unterscheiden. Diese wurden einfach durch Variable der Funktion mitgegeben und in einer *if* Abfrage abgefragt.
Figure 1 und Figure 2 vergleichen die zu unersuchenden Nachbarschaftspixel.

Regionsvergleich

x/y	-1	0	1
-1	0	x	0
0	x	0	x
1	0	x	0

Table 1: N4 Region

x/y	-1	0	1
-1	x	x	x
0	x	0	x
1	x	x	x

Table 2: N8 Region

Listing 1: RegionGrowing-Algorithmus.

```
      ; columns
1
2  import java.awt.Point;
3  import java.awt.Rectangle;
4  import java.util.Stack;
5
6  import ij.*;
7  import ij.gui.GenericDialog;
8  import ij.gui.PointRoi;
9  import ij.plugin.filter.PlugInFilter;
10 import ij.process.*;
11
12 public class RegionGrowing_ implements PlugInFilter {
13
14         ImagePlus imp1;
15
16         public int setup(String arg, ImagePlus imp) {
17                 if (arg.equals("about")) {
18                         showAbout();
19                         return DONE;
20                 }
21
22                 imp1 = imp;
23                 return DOES_8G + DOES_STACKS + SUPPORTS_MASKING + ROI_REQUIRED;
24         } // setup
25
26         public static int[][] performRegionGrowing(int[][] inImgArr, int width, int height, int lowerThresh, int upperThresh
               ↪ , int seedX, int seedY, String region) {
27                 // constants
28                 int BG_VAL = 0;
29                 int FG_VAL = 255;
30                 int UNPROCESSED_VAL = -1;
31
32                 int[][] returnArr = new int[width][height];
33
34                 for( int x= 0; x < width; x++) {
35                         for (int y = 0; y < height; y++) {
36                                 returnArr[x][y] = UNPROCESSED_VAL;
37                         }
38                 }
39
40                 Stack<Point> processingStack = new Stack<Point>();
41
42                 //first check if seed point is valid
```

```
43 |                 int seedVal = inImgArr[seedX][seedY];
44 |                 if(seedVal >= lowerThresh && seedVal <= upperThresh) {
45 |                         processingStack.push(new Point(seedX, seedY));
46 |                         returnArr[seedX][seedY] = FG_VAL;
47 |                 }
48 |
49 |                 while(!processingStack.empty()) {
50 |                         Point nextPos = processingStack.pop();
51 |
52 |                         //check all children in N4
53 |                         for( int xOffset= -1; xOffset <= 1; xOffset++) {
54 |                                 for (int yOffset = -1; yOffset <= 1; yOffset++) {
55 |                                         int nbX = nextPos.x + xOffset;
56 |                                         int nbY = nextPos.y + yOffset;
57 |
58 |                                         // check if N4 region
59 |                                         boolean isRegion = false;
60 |                                         if(region.equals("N4") && (xOffset*yOffset == 0 && xOffset+yOffset != 0)) isRegion =
                                                ↪  true;
61 |                                         if(region.equals("N8") && (xOffset != 0 || yOffset != 0)) isRegion = true;
62 |
63 |
64 |                                         if(isRegion) {
65 |
66 |                                                 // check if valid range ==> position within image boundaries
67 |                                                 if(nbX >= 0 && nbY >= 0 && nbX < width && nbY < height) {
68 |
69 |                                                         int nbVal = inImgArr[nbX][nbY];
70 |
71 |                                                         //if current pixel was not processed yet (check if pixel is
                                                                ↪ unprocessed and if vlaue in threshold range)
72 |                                                         if(returnArr[nbX][nbY] == UNPROCESSED_VAL) {
73 |
74 |                                                                 //if range valid
75 |                                                                 if(nbVal >= lowerThresh && nbVal <= upperThresh) {
76 |                                                                         returnArr[nbX][nbY] = FG_VAL;
                                                                                ↪          //set current pixel to foreground
77 |                                                                         processingStack.push(new Point(nbX,nbY));        //
                                                                                ↪ push current pixel to the stack
78 |                                                                 }
79 |                                                                 else {
80 |                                                                         returnArr[nbX][nbY] = BG_VAL;
81 |                                                                 }
82 |                                                         }
83 |                                                 }
```

```
                                        } // if N4 region
                                }//for yOffset
                        }// for xOffset

                } //while

                System.out.println(processingStack.size());

                //cleanup - all values still unprocessed - get assigned the background value BG_VAL
                for( int x = 0; x < width; x++) {
                        for (int y = 0; y< height; y++) {
                                if(returnArr[x][y] == UNPROCESSED_VAL) {
                                        returnArr[x][y] = BG_VAL;
                                }
                        }
                }


                return returnArr;
        } //performRegionGrowing


        public void run(ImageProcessor ip) {
                byte[] pixels = (byte[]) ip.getPixels();
                int width = ip.getWidth();
                int height = ip.getHeight();

                int[][] inDataArrInt = ImageJUtility.convertFrom1DByteArr(pixels, width, height);

                //request seed point
                PointRoi pr = (PointRoi)imp1.getRoi();
                Rectangle rect = pr.getBounds();
                int xStart = pr.getXCoordinates()[0] + rect.x;
                int yStart = pr.getXCoordinates()[0] + rect.y;

                System.out.println("xStart : " + xStart +  ", yStart : " + yStart);

                // user input - default
                int lowerThresh = 100;
                int upperThresh = 255;
                // user dialog
                GenericDialog gd = new GenericDialog("thresh_params");
                gd.addSlider("lower_thresh", 0, 255, lowerThresh);
                gd.addSlider("upper_thresh", 0, 255, upperThresh);
                gd.showDialog();
```

```
129 |
130 |                 if (!gd.wasCanceled()) {
131 |                         lowerThresh = (int) gd.getNextNumber();
132 |                         upperThresh = (int) gd.getNextNumber();
133 |                 }
134 |
135 |                 //finally calling function
136 |                 int[][] resultImg = performRegionGrowing(inDataArrInt, width, height, lowerThresh, upperThresh, xStart,
    |                     ↪ yStart,"N4");
137 |
138 |
139 |                 ImageJUtility.showNewImage(resultImg, width, height, "region_coin_result");
140 |
141 |         } // run
142 |
143 |         void showAbout() {
144 |                 IJ.showMessage("About_Template_...", "this_is_a_PluginFilter_template\n");
145 |         } // showAbout
146 |
147 | } // class FilterTemplate_
```

b) Image Growing mit Labeling

Für die Implmentierung wurde der Code aus Aufgabe **??** kopiert und erweitert. Muss das gesamte Bild untersucht werden um alle Objekte zu finden. Wird ein passendes Pixel gefunden, wird der Region-Growing Algorithmus herangezogen. Mittels der Fordergrundfarbe werden die Objekte eingeteilt und unterschieden.

Listing 2: RegionGrowing-Algorithmus.

```
      ; columns
1
2  import java.awt.Point;
3  import java.awt.Rectangle;
4  import java.util.Stack;
5
6  import ij.*;
7  import ij.gui.GenericDialog;
8  import ij.gui.PointRoi;
9  import ij.plugin.filter.PlugInFilter;
10 import ij.process.*;
11
12 public class AutoRegionGrowing_ implements PlugInFilter {
13
14    public int setup(String arg, ImagePlus imp) {
15                  if (arg.equals("about"))
16                          {showAbout(); return DONE;}
17                  return DOES_8G+DOES_STACKS+SUPPORTS_MASKING;
18          } //setup
19
20
8 21         public static int[][] performRegionGrowing(int[][] inImgArr, int width, int height, int lowerThresh, int upperThresh
                ↪ , String region) {
22                  // constants
23                  int BG_VAL = 0;
24                  int FG_VAL = 255;
25                  int UNPROCESSED_VAL = -1;
26
27                  int[][] returnArr = new int[width][height];
28
29
30                  // prepare -> set every pixel to unprocessed state
31                  for( int x= 0; x < width; x++) {
32                          for (int y = 0; y < height; y++) {
33                                  returnArr[x][y] = UNPROCESSED_VAL;
34                          }
35                  }
36
37                  Stack<Point> processingStack = new Stack<Point>();
38
39                  // for whole image
40                  for ( int x = 0; x < width; x++) {
41                          for ( int y = 0; y < height; y++) {
42
```

```
43                  //first check if seed point is valid
44                  int seedVal = inImgArr[x][y];
45                  if(seedVal >= lowerThresh && seedVal <= upperThresh && returnArr[x][y] == UNPROCESSED_VAL) {
46                          processingStack.push(new Point(x, y));
47                          FG_VAL = FG_VAL -20;
48                          System.out.println("next foreground will be: " + FG_VAL);
49                          //returnArr[x][y] = FG_VAL;
50                  }
51
52
53                  while(!processingStack.empty()) {
54                          Point nextPos = processingStack.pop();
55
56                          //check all children in N4
57                          for( int xOffset= -1; xOffset <= 1; xOffset++) {
58                                  for (int yOffset = -1; yOffset <= 1; yOffset++) {
59                                          int nbX = nextPos.x + xOffset;
60                                          int nbY = nextPos.y + yOffset;
61
62                                          // check if N4 region
63                                          boolean isRegion = false;
64                                          if(region.equals("N4") && (xOffset*yOffset == 0 && xOffset+yOffset
                                                ↪ != 0)) isRegion = true;
65                                          if(region.equals("N8") && (xOffset != 0 || yOffset != 0)) isRegion =
                                                ↪  true;
66
67                                          if(isRegion) {
68
69                                                  // check if valid range ==> position within image boundaries
70                                                  if(nbX >= 0 && nbY >= 0 && nbX < width && nbY < height) {
71
72                                                          int nbVal = inImgArr[nbX][nbY];
73
74                                                          //if current pixel was not processed yet (check if
                                                                ↪ pixel is unprocessed and if vlaue in
                                                                ↪ threshold range)
75                                                          if(returnArr[nbX][nbY] == UNPROCESSED_VAL) {
76
77                                                                  //if range valid
78                                                                  if(nbVal >= lowerThresh && nbVal <=
                                                                        ↪ upperThresh) {
79                                                                          returnArr[nbX][nbY] = FG_VAL;
                                                                                ↪                       //set
                                                                                ↪  current pixel to foreground
80                                                                          processingStack.push(new Point(nbX,
                                                                                ↪ nbY));        // push current
```

```
                                                                                        ↪ pixel to the stack
                                                                        }
                                                                        else {
                                                                                returnArr[nbX][nbY] = BG_VAL;
                                                                        }
                                                                }
                                                        }
                                                } // if N4 region
                                        }//for yOffset
                                }// for xOffset

                        } //while processed all pixels of growing region



                } // for hight -> y
        } //ffor width -> x


        System.out.println(processingStack.size());


        //cleanup - all values still unprocessed - get assigned the background value BG_VAL
        for( int x = 0; x < width; x++) {
                for (int y = 0; y< height; y++) {
                        if(returnArr[x][y] == UNPROCESSED_VAL) {
                                returnArr[x][y] = BG_VAL;
                        }
                }
        }


        return returnArr;
} //performRegionGrowing


public void run(ImageProcessor ip) {
        byte[] pixels = (byte[]) ip.getPixels();
        int width = ip.getWidth();
        int height = ip.getHeight();

        int[][] inDataArrInt = ImageJUtility.convertFrom1DByteArr(pixels, width, height);

        // user input - default
        int lowerThresh = 100;
```

```
                int upperThresh = 255;
                // user dialog
                GenericDialog gd = new GenericDialog("thresh_params");
                gd.addSlider("lower_thresh", 0, 255, lowerThresh);
                gd.addSlider("upper_thresh", 0, 255, upperThresh);
                gd.showDialog();

                if (!gd.wasCanceled()) {
                        lowerThresh = (int) gd.getNextNumber();
                        upperThresh = (int) gd.getNextNumber();
                }

                //finally calling function
                int[][] resultImg = performRegionGrowing(inDataArrInt, width, height, lowerThresh, upperThresh,"N8");


                ImageJUtility.showNewImage(resultImg, width, height, "region_coin_result");

        } // run

        void showAbout() {
                IJ.showMessage("About_Template_...", "this_is_a_PluginFilter_template\n");
        } // showAbout

} // class FilterTemplate_
```

4.2 Optimaler Threshold

a) Adaptiver optimar Threshold

todo

b) Image Growing mit Labeling

todo

4.3 Objekterkennung mittels mathematischer Morphologie

a) Rechtecks Erkennung - binär

mittels Erosion Dilation bzww arithmetische Operationen todo

b) Image Growing mit Labeling - Graustufen

todo