

... with Microcomputer\exercises\Ex4\Ex4\Ex4-CV-main

```
1  /*****
2  *                                     EXERCISE 4                                     *
3  *                                     -----                                     *
4  *                                     CONTOURS                                     *
5  *****/
6
7  #include <opencv2/opencv.hpp>
8  #include "Ex4-CV-header.h"
9  #include <iostream>
10 #include <stdio.h>
11 #include <math.h>
12
13 #define LINES 'L'
14 #define RECTS 'R'
15
16 using namespace std;
17 using namespace cv;
18
19
20
21 int main(int argc, char **argv) {
22     /*****
23     *                                     Gradient Method                                     *
24     *****/
25     IplImage * img = cvLoadImage(argv[1] , CV_LOAD_IMAGE_GRAYSCALE);
26
27     cvNamedWindow("Grey scale" , 0);
28     cvShowImage("Grey scale" , img);
29     cvSaveImage("Grey_Scale.png" , img);
30
31     /// Create a binary image
32     IplImage * his = getHistogram(img , LINES);
33
34     cvNamedWindow("histogram" , 0);
35     cvShowImage("histogram" , his);
36     cvSaveImage("Histogram.png" , his);
37
38     cvWaitKey(0);
39
40     IplImage * bin = simpleThresholding(img , his);
41
42     cvNamedWindow("Binary Image" , 0);
43     cvShowImage("Binary Image" , bin);
44     cvSaveImage("Binary_Image.png" , bin);
45
46     /// Doesn't produce a very good result and requires some priori knowledge about the scene
47     /// (e.g. if one expects darker objects on a less dark background, or viceversa).
48     IplImage * gradContour = cvCreateImage(cvSize(img->width - 2, img->height - 2), IPL_DEPTH_8U, 1);
49     cvSet(gradContour , 0);
50     gradSearchContour(img , gradContour);
51
52     cvNamedWindow("Gradient Method" , 0);
53     cvShowImage("Gradient Method" , gradContour);
54     cvSaveImage("Gradient.png" , gradContour);
55
56     IplImage * I = cvCreateImage(cvGetSize(img) , IPL_DEPTH_8U , 3);
57     cvCvtColor(img , I , COLOR_GRAY2RGB);
58     IplImage * m = cvCreateImage(cvGetSize(img) , IPL_DEPTH_8U , 1);
59     cvSet(m , 0);
60
61     cvSetImageROI(m , cvRect(1, 1, img->width - 2, img->height - 2) );
62     cvCopy(gradContour , m);
63     cvResetImageROI(m);
```

```
64     const CvMat M = cvarrToMat(m);
65
66     cvSet(I , cvScalar(255,0,0) , &M);
67     cvNamedWindow("Show Gradient" , 0);
68     cvShowImage("Show Gradient" , I);
69     cvSaveImage("Show_Gradient.png" , I);
70
71     /// Every black pixel that has an adjacent white pixel is a border
72     IplImage * binGradContour = cvCreateImage(cvSize(img->width-2, img->height-2), IPL_DEPTH_8U, 1);
73     cvSet(binGradContour , 0);
74     searchContour(bin , binGradContour);
75
76     cvNamedWindow("Binary-Gradient Contour" , 0);
77     cvShowImage("Binary-Gradient Contour" , binGradContour);
78     cvSaveImage("Binary_Gradient_Contour.png" , binGradContour);
79
80     cvCvtColor(img , I , COLOR_GRAY2RGB);
81
82     m = cvCreateImage(cvGetSize(img) , IPL_DEPTH_8U , 1);
83
84     cvSet(m , 0);
85     cvSetImageROI(m , cvRect(1 , 1 , img->width - 2 , img->height - 2));
86     cvCopy(binGradContour , m);
87     cvResetImageROI(m);
88
89     const CvMat M_1 = cvarrToMat(m);
90
91     cvSet(I, cvScalar(0 , 0 , 255) , &M_1);
92     cvNamedWindow("Show Binary Contour" , 0);
93     cvShowImage("Show Binary Contour" , I);
94     cvSaveImage("Show_Binary_Contour.png" , I);
95
96
97
98
99     /*****
100     *                               Gradient Method - the right way                               *
101     *****/
102
103     IplImage * Xim = cvCreateImage(cvGetSize(img) , IPL_DEPTH_8U , 1);
104     IplImage * Yim = cvCreateImage(cvGetSize(img) , IPL_DEPTH_8U , 1);
105     IplImage * gradientImage = cvCreateImage(cvGetSize(img) , IPL_DEPTH_8U , 1);
106
107     float Prewitt_x [3][3] = { { -1 , 0 , 1},
108                               { -1 , 0 , 1},
109                               { -1 , 0 , 1} };
110
111     float Prewitt_y [3][3] = { { -1 , -1 , -1},
112                               { 0 , 0 , 0},
113                               { 1 , 1 , 1} };
114
115     CvMat PrewittXK, PrewittYK;
116     cvInitMatHeader(&PrewittXK , 3 , 3 , CV_32F , Prewitt_x);
117     cvInitMatHeader(&PrewittYK , 3 , 3 , CV_32F , Prewitt_y);
118
119     cvFilter2D(img , Xim , &PrewittXK , cvPoint(-1 , -1) );
120     cvFilter2D(img , Yim , &PrewittYK , cvPoint(-1 , -1) );
121
122     for(int i = 0 ; i < Xim->height ; i++) {
123
124         char * px = Xim->imageData + i * Xim->widthStep;
125         char * py = Yim->imageData + i * Yim->widthStep;
126         char * pmag = gradientImage->imageData + i * gradientImage->widthStep;
127     }
```

... with Microcomputer\exercises\Ex4\Ex4\Ex4-CV-main

```
128     for(int j = 0; j < Xim->width; j++) {
129
130         float x = (float)(uchar)(*px);
131         float y = (float)(uchar)(*py);
132
133         *pmag = (uchar)sqrt(x * x + y * y);
134         px++;
135         py++;
136         pmag++;
137     }
138 }
139
140 cvNamedWindow("Prewitt" , 0);
141 cvShowImage("Prewitt" , gradientImage);
142 cvSaveImage("Prewitt.png" , gradientImage);
143
144
145 ///////////////////////////////////////////////////
146
147 float Sobel_x [3][3] = { { -1 , 0 , 1},
148                          { -2 , 0 , 2},
149                          { -1 , 0 , 1} };
150 float Sobel_y [3][3] = { { -1 , -2 , -1},
151                          { 0 , 0 , 0},
152                          { 1 , 2 , 1} };
153
154 CvMat SobelXK, SobelYK;
155 cvInitMatHeader(&SobelXK , 3 , 3 , CV_32F , Sobel_x);
156 cvInitMatHeader(&SobelYK , 3 , 3 , CV_32F , Sobel_y);
157
158 cvFilter2D(img , Xim , &SobelXK , cvPoint(-1 , -1) );
159 cvFilter2D(img , Yim , &SobelYK , cvPoint(-1 , -1) );
160
161 for(int i = 0 ; i < Xim->height ; i++) {
162
163     char * px = Xim->imageData + i * Xim->widthStep;
164     char * py = Yim->imageData + i * Yim->widthStep;
165     char * pmag = gradientImage->imageData + i * gradientImage->widthStep;
166
167     for(int j = 0 ; j < Xim->width ; j++) {
168
169         float x = (float)(uchar)(*px);
170         float y = (float)(uchar)(*py);
171
172         *pmag = (uchar)sqrt(x*x + y*y);
173         px++;
174         py++;
175         pmag++;
176     }
177 }
178
179 cvNamedWindow("Sobel" , 0);
180 cvShowImage("Sobel" , gradientImage);
181 cvSaveImage("Sobel.png" , gradientImage);
182
183 ///////////////////////////////////////////////////
184
185 IplImage * CannyIm = cvCreateImage(cvGetSize(img) , IPL_DEPTH_8U , 1);
186 cvLaplace(img , CannyIm , 5);
187 cvNamedWindow("Laplace" , 0);
188 cvShowImage("Laplace" , gradientImage);
189 cvSaveImage("Laplace.png" , gradientImage);
190
191 ///////////////////////////////////////////////////
```

```
192
193 // Resulting image very dark
194 float Roberts_x [2][2] = { { 0 , 1},
195                             { -1 , 0} };
196 float Roberts_y [2][2] = { { 1 , 0},
197                             { 0 , -1} };
198
199 CvMat RobertsXK, RobertsYK;
200 cvInitMatHeader(&RobertsXK , 2 , 2 , CV_32F , Roberts_x);
201 cvInitMatHeader(&RobertsYK , 2 , 2 , CV_32F , Roberts_y);
202
203 cvFilter2D(img , Xim , &RobertsXK , cvPoint(-1 , -1) );
204 cvFilter2D(img , Yim , &RobertsYK , cvPoint(-1 , -1) );
205
206 for(int i = 0 ; i < Xim->height ; i++) {
207
208     char * px = Xim->imageData + i * Xim->widthStep;
209     char * py = Yim->imageData + i * Yim->widthStep;
210     char * pmag = gradientImage->imageData + i * gradientImage->widthStep;
211
212     for(int j = 0 ; j < Xim->width ; j++) {
213
214         float x = (float)(uchar)(*px);
215         float y = (float)(uchar)(*py);
216
217         *pmag = (uchar)sqrt(x*x + y*y);
218         px++;
219         py++;
220         pmag++;
221     }
222 }
223
224 cvNamedWindow("Roberts" , 0);
225 cvShowImage("Roberts" , gradientImage);
226 cvSaveImage("Roberts.png" , gradientImage);
227
228
229 /*****
230 *                               Laplacian Method                               *
231 *****/
232
233 IplImage * LG_contour = cvCreateImage(cvGetSize(img) , IPL_DEPTH_8U , 1);
234
235 /*double LG_mat [9][9] = { {0 , 0 , 1 , 2 , 2 , 2 , 1 , 0 , 0},
236                             {0 , 1 , 5 , 10 , 12 , 10 , 5 , 1 , 0},
237                             {1 , 5 , 15 , 19 , 16 , 19 , 15 , 5 , 1},
238                             {2 , 10 , 19 , -19 , -64 , -19 , 19 , 10 , 2},
239                             {2 , 12 , 16 , -64 , -148 , -64 , 16 , 12 , 2},
240                             {2 , 10 , 19 , -19 , -64 , -19 , 19 , 10 , 2},
241                             {1 , 5 , 15 , 19 , 16 , 19 , 15 , 5 , 1},
242                             {0 , 1 , 5 , 10 , 12 , 10 , 5 , 1 , 0},
243                             {0 , 0 , 1 , 2 , 2 , 2 , 1 , 0 , 0}
244                             };*/
245
246 /*double LG_mat [7][7] = { {0 , 1 , 2 , 4 , 2 , 1 , 0},
247                             {1 , 7 , 24 , 31 , 24 , 7 , 1},
248                             {2 , 24 , 17 , -51 , 17 , 24 , 2},
249                             {4 , 31 , -51 , -248 , -51 , 31 , 4},
250                             {2 , 24 , 17 , -51 , 17 , 24 , 2},
251                             {1 , 7 , 24 , 31 , 24 , 7 , 1},
252                             {0 , 1 , 2 , 4 , 2 , 1 , 0}
253                             };*/
254
255
```

... with Microcomputer\exercises\Ex4\Ex4-CV-main

```
256     double LG_mat [5][5] = { { 1 , 9 , 19 , 9 , 1},
257                               { 9 , 58 , 12 , 58 , 9},
258                               {19 , 12 , -432 , 12 , 19},
259                               { 9 , 58 , 12 , 58 , 9},
260                               { 1 , 9 , 19 , 9 , 1}
261                               };
262
263     //////////////////////////////////////
264
265     /*double LG_mat [3][3] = { { 1, 4, 1},
266                               { 4, -20, 4},
267                               { 1, 4, 1}
268                               };*/
269
270
271     CvMat LG_kernel;
272     cvInitMatHeader(&LG_kernel , 5 , 5 , CV_64F , LG_mat);
273
274     cvFilter2D(img , LG_contour , &LG_kernel , cvPoint(-1 , -1) );
275
276     cvNamedWindow("Laplacian Contour" , 0);
277     cvShowImage("Laplacian Contour" , LG_contour);
278     cvSaveImage("Laplacian_Contour.png" , LG_contour);
279
280
281
282     /*****
283     *                               *
284     *          Contour Search          *
285     *                               *
286     *****/
287
288     /// To implement
289
290     cvWaitKey(0);
291
292     /*****
293     *                               *
294     *          Clean-Up          *
295     *                               *
296     *****/
297     cvReleaseImage(&img);
298     cvReleaseImage(&his);
299     cvReleaseImage(&bin);
300     cvReleaseImage(&gradContour);
301     cvReleaseImage(&binGradContour);
302     cvReleaseImage(&LG_contour);
303
304     cvDestroyAllWindows();
305
306     return 0;
307 }
```