...b\11. Image Analysis with Microcomputer\exercises\Ex2-CV-functions.cpp

```cpp
/*********************************************************************************************
*                                    EXERCISE 2                                              *
*                                   ------------                                             *
*                                  BINARY IMAGES                                            *
*********************************************************************************************/

#include <opencv2/opencv.hpp>
#include <iostream>
#include <math>


#define LINES 'L'
#define RECTS 'R'
#define PI 3.14159265358979323846

using namespace std;
using namespace cv;




/*********************************************************************************************
* From the pointer to a greyscale image, draws a histogram image and returns a pointer      *
* to it. Two possible styles can be selected:                                               *
* LINES (a line for each bin)                                                               *
* RECTS (a rectangle for each bin)                                                          *
*********************************************************************************************/

IplImage * getHistogram(IplImage * gray , char style) {
    int dep = 256;
    int hist[dep];              // Histogram for grayscale image
    int st = 4;                 // To separate lines in the histograms, for better look ;)

    /// Initialize histograms
    for (int i = 0 ; i < dep ; i++)
        hist[i] = 0;

    /// Calculate histogram
    for(int i = 0 ; i < gray->height ; i++) {
        char * ptr = gray->imageData + i * gray->widthStep;
        for(int j = 0 ; j < gray->width ; j++) {
                if( *ptr < 0 ) {
                    char c = *ptr;
                    uchar x = (uchar) c;
                    hist[x] += 1;
                }
                else
                    hist[(unsigned)(*ptr)] += 1;
                ptr++;
        }
    }

    /// Create image for the histograms
    IplImage * his = cvCreateImage(cvSize(st*dep , 600) , IPL_DEPTH_8U , 1);
    cvSet(his , 0);                             // Initialize image (all black)
    his->origin = IPL_ORIGIN_BL;                // Set the origin in the bottom left corner
```

```
64      /// Draw a line/Rectangle for each bin
65      if (style == LINES)
66          for (int i = 0 ; i < dep ; i++)
67              if (hist[i] != 0)
68                  cvLine(his , cvPoint(i*st , 0), cvPoint(i*st , hist[i] / 10) , 150 , 1 , 4);
69      else if (style == RECTS)
70          for (int i = 0 ; i < dep ; i++)
71              if (hist[i] != 0)
72                  cvRectangle(his , cvPoint(i*st , hist[i] / 10),
73                              cvPoint((i+1)*st , 0), 150, -1, 4);
74      else {
75          printf("\nError. No style selected.\n");
76          cvReleaseImage(&his);
77          return NULL;
78      }
79
80      return his;
81  }
82
83
84
85
86  /*****************************************************************************
87  * Input: a pointer to a GREY SCALE image, a threshold, 3 pointers to int for the        *
88  * necessary computation to determine the center of mass.                                *
89  * Forms a binary image, counts the number of pixels set to 0, sums the values of the 2  *
90  * coordinates for the pixels set to 0 (used later to calculate the center of mass       *
91  * coordinates).                                                                         *
92  *                                                                                       *
93  * PS. It is possible to calculate the coordinates even in this function but it's done   *
94  *     later to respect the structure of the exercise                                    *
95  *****************************************************************************/
96
97  IplImage * getBinImagePlus(IplImage * img, unsigned t, int * tot, int * xcm, int * ycm) {
98      IplImage * I = cvCreateImage(cvGetSize(img) , IPL_DEPTH_8U , 1);
99      for(int i = 0 ; i < img->height ; i++) {
100         char * ptr = img->imageData + i * img->widthStep;
101         char * p = I->imageData + i * I->widthStep;
102         for(int j = 0 ; j < img->width ; j++) {
103                 if ((uchar)(*ptr) >= (uchar)t)
104                     *p = 255;
105                 else {
106                     // In this case we are on the object
107                     *p = 0;
108                     (*tot)++;
109                     (*ycm) += i;
110                     (*xcm) += j;
111                 }
112             ptr++;
113             p++;
114         }
115     }
116     return I;
117 }
118
119
120
121
122
123
124
125
126
```

```
127 /*****************************************************************************************
128 * Interactively allows to select a threshold and see the position in a histogram.       *
129 * When the user is satisfied, it shows the binary image built with the selected         *
130 * threshold and if the result is not satisfying, allows to select a new threshold again.*
131 * It returns a pointer to the final binary image that the user confirmed.               *
132 *                                                                                       *
133 *                                                                                       *
134 * The function needs:                                                                   *
135 *    - A pointer to a GREY SCALE image to apply the threshold to.                       *
136 *    - A pointer to a preliminary histogram image (necessary to restore the histogram   *
137 *      when changing the threshold).                                                    *
138 *    - A pointer to a 3-channel image for the histogram to be updated when a new        *
139 *      threshold is selected (it must be passed as input because the function must       *
140 *      return another pointer).                                                         *
141 *    - 3 pointers to int to compute the center of mass coordinates                      *
142 *****************************************************************************************/
143
144 #define NEW 'N'
145 #define OK 'O'
146
147 IplImage * interactiveThresholding(IplImage * G, IplImage * Chis, IplImage * his, int *tot, int
148                                    *xcm, int *ycm) {
149     unsigned int t = 0;                    // Threshold
150     char st = 4;                           // To place correctly the threshold line in the image
151     IplImage * bin;                        // To contain the binary image
152
153     printf("\nInsert Threshold: ");
154     scanf("%u", &t);
155
156     while(1) {
157         while(1) {
158             if ( !(t >= 0 && t <= 255) ) {
159                 printf("\nInvalid value!\n");
160                 printf("\nInsert Threshold: ");
161                 scanf("%u", &t);
162                 continue;
163             }
164             cvCvtColor(his , Chis , COLOR_GRAY2RGB);
165             cvFlip(Chis , Chis , 0);
166             cvLine(Chis , cvPoint(t*st , 0) , cvPoint(t*st , 600) , cvScalar(0,0,255) , 2 , 4);
167             cvShowImage("histogram" , Chis);
168
169             printf("\nPress 'O' to continue or 'N' to select a new threshold (in the histogram
170                    image window)\n");
171
172             int c = cvWaitKey(0);
173
174             if (c == OK)
175                 break;
176             else if (c == NEW) {
177                 printf("\nInsert Threshold: ");
178                 scanf("%u", &t);
179                 continue;
180             }
181             else {
182                 printf("\nERROR! Press 'O' or 'N' next time.\n");
183                 printf("\nInsert Threshold: ");
184                 scanf("%u", &t);
185                 continue;
186             }
187         }
188
189
```

```
190         // Note: in this context, tot, xcm, ycm are pointers
191         bin = getBinImagePlus(G, t, tot, xcm, ycm);
192         cvNamedWindow("Binary Image" , 0);
193         cvShowImage("Binary Image" , bin);
194         printf("\nPress 'O' if you are satisfied, or 'N' to select a new threshold (in the
195                binary image window)\n");
196
197         int k = cvWaitKey(0);
198
199         if (k == OK)
200             break;
201         else if (k == NEW) {
202             printf("\nInsert Threshold: ");
203             scanf("%u", &t);
204             continue;
205         }
206         else {
207             printf("\nERROR! Press 'O' or 'N' next time.\n");
208             printf("\nInsert Threshold: ");
209             scanf("%u", &t);
210             continue;
211         }
212     }
213     return bin;
214 }
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
```

```
253 /*****************************************************************************************
254 * This function allows the user to select a threshold and draws a new histogram with a  *
255 * visual indication of the threshold. At the same time computes the necessary numbers   *
256 * for the centre of mass.                                                               *
257 *                                                                                       *
258 * The function needs:                                                                   *
259 *    - a pointer to a GREY SCALE image to apply the threshold to.                       *
260 *    - a pointer to a preliminary histogram image (starting point to draw the new       *
261 *      histogram with the threshold).                                                   *
262 *    - 3 pointers to int to compute the center of mass coordinates.                     *
263 *****************************************************************************************/
264
265 IplImage * simpleThresholding(IplImage * G, IplImage * his, int *tot, int *xcm, int *ycm) {
266     unsigned int t = 0;              // Threshold
267     char st = 4;                     // To place correctly the threshold line in the image
268     IplImage * bin;                  // To contain the binary image
269
270     printf("\nInsert Threshold: ");
271     scanf("%u", &t);
272
273     IplImage * Chis = cvCreateImage(cvGetSize(his) , IPL_DEPTH_8U , 3);
274
275     // Represent a grey scale histogram in a 3-channel image.
276     // It will be useful later to draw the threshold on the histogram.
277     cvCvtColor(his , Chis , COLOR_GRAY2RGB);
278     cvFlip(Chis , Chis , 0);
279
280     cvLine(Chis, cvPoint(t*st , 0) , cvPoint(t*st , 600) , cvScalar(0 , 0 , 255), 2, 4);
281     cvShowImage("histogram", Chis);
282     cvSaveImage("histogram.png", Chis);
283     cvReleaseImage(&Chis);
284
285     bin = getBinImagePlus(G, t, tot, xcm, ycm);
286     cvNamedWindow("Binary Image", 0);
287     cvShowImage("Binary Image", bin);
288
289     return bin;
290 }
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
```

```
316 /*****************************************************************************
317 * Given a pointer to a binary image and the coordinates of the centre of mass, and 3    *
318 * pointers to location where the results will be stored, it calculates the reduced      *
319 * central moments.                                                                      *
320 * It prints out the central moments as well as the reduced central ones.                *
321 *****************************************************************************/
322
323 void reducedCentralMoments(IplImage * bin, double xcm, double ycm, double * M_20, double * M_02,
324                            double * M_11) {
325     long tot = 0;
326     double b, r;
327
328     for(int i = 0 ; i < bin->height ; i++) {
329         char * p = bin->imageData + i * bin->widthStep;
330         for(int j = 0 ; j < bin->width ; j++) {
331             if ((uchar)(*p) == 0) {
332                 tot++;
333                 b = j - xcm;
334                 r = pow(b, 2);
335                 *M_20 += r;
336                 b = i - ycm;
337                 r = pow(b, 2);
338                 *M_02 += r;
339                 *M_11 += (i - ycm) * (j - xcm);
340             }
341             p++;
342         }
343     }
344
345     printf("\nCentral Moments:\n\tm_11 = %.2f\n\tm_02 = %.2f\n\tm_20 = %.2f\n", *M_11, *M_02,
346            *M_20);
347
348     /// Reduced Central Moments
349     *M_02 /= tot;
350     *M_20 /= tot;
351     *M_11 /= tot;
352
353     printf("\nReduced Central Moments:\n\tm_11 = %.2f\n\tm_02 = %.2f\n\tm_20 = %.2f\n", *M_11,
354            *M_02, *M_20);
355 }
```