... with Microcomputer\exercises\Ex4\Ex4\Ex4-CV-functions.cpp

```cpp
/*********************************************************************************************
*                                    EXERCISE 4                                              *
*                                   ------------                                             *
*                                    CONTOURS                                                *
*********************************************************************************************/

#include <opencv2/opencv.hpp>
#include "Ex4-CV-header.h"
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <cmath>

#define LINES 'L'
#define RECTS 'R'

using namespace std;
using namespace cv;



void gradSearchContour(IplImage * img , IplImage * contour) {
    for(int i = 0 ; i < img->height - 2 ; i++) {

        char * p_img = img->imageData + (i + 1) * img->widthStep + 1;
        char * p_contour = contour->imageData + i * contour->widthStep;

        for(int j = 0 ; j < img->width - 2 ; j++) {
            if ( gradCheck(p_img , p_img - 1 - img->widthStep , img->widthStep) )
                    *p_contour = 255;
            p_img++;
            p_contour++;
        }
    }
}


////////////////////////////////////////////////////////////////////////////////////////////


bool gradCheck(char * c , char * p , int step) {
    for(int i = 0 ; i < 3 ; i++) {
        char * lp = p + i * step;
        for(int j = 0 ; j < 3 ; j++) {
            if (i == 1 && j == 1) {
                lp++;
                continue;
            }
            float diff = (float)((uchar)(*lp) - (uchar)(*c));
            if(diff > 23)
                return true;
            lp++;
        }
    }
    return false;
}


////////////////////////////////////////////////////////////////////////////////////////////
```

... with Microcomputer\exercises\Ex4\Ex4\Ex4-CV-functions.cpp

```
65  void searchContour(IplImage * bin , IplImage * contour) {
66      for(int i = 0 ; i < bin->height - 2 ; i++) {
67
68          char * p_bin = bin->imageData + (i + 1) * bin->widthStep + 1;
69          char * p_contour = contour->imageData + i * contour->widthStep;
70
71          for(int j = 0 ; j < bin->width - 2 ; j++) {
72              if ( ((uchar)*p_bin == 0) && check(p_bin - 1 - bin->widthStep , bin->widthStep))
73                      *p_contour = 255;
74              p_bin++;
75              p_contour++;
76          }
77      }
78  }
79
80
81  ////////////////////////////////////////////////////////////////////////////////////////
82
83
84
85  bool check(char * p , int step) {
86      for(int i = 0 ; i < 3 ; i++) {
87
88          char * lp = p + i * step;
89
90          for(int j = 0 ; j < 3 ; j++) {
91              if (i == 1 && j == 1) {
92                  lp++;
93                  continue;
94              }
95              if((uchar)(*lp) == 255)
96                  return true;
97              lp++;
98          }
99      }
100     return false;
101 }
102
103
104
105 /*********************************************************************************
106 * This function allows the user to select a threshold and draws a new histogram with a  *
107 * visual indication of the threshold. At the same time computes the necessary numbers   *
108 * for the centre of mass.                                                               *
109 *                                                                                       *
110 * The function needs:                                                                   *
111 *    - A pointer to a GREYSCALE image to apply the threshold to.                        *
112 *    - A pointer to a preliminary histogram image (starting point to draw the new       *
113 *      histogram with the threshold).                                                   *
114 *    - 3 pointers to int to compute the center of mass coordinates.                     *
115 *********************************************************************************/
116
117 IplImage * simpleThresholding(IplImage * G, IplImage * his) {
118     unsigned int t = 0;             // Threshold
119     char st = 4;                    // To place correctly the threshold line in the image
120     IplImage * bin;                 // To contain the binary image
121
122     printf("\nInsert Threshold: ");
123     scanf("%u", &t);
124
125     IplImage * Chis = cvCreateImage(cvGetSize(his) , IPL_DEPTH_8U , 3);
126
127     // Represent a greyscale histogram in a 3-channel image.
128     // It will be useful later to draw the threshold on the histogram.
```

```
129
130         cvCvtColor(his, Chis, COLOR_GRAY2RGB);
131         cvFlip(Chis, Chis, 0);
132
133         cvLine(Chis, cvPoint(t*st, 0), cvPoint(t*st, 600), cvScalar(0,0,255), 2, 4);
134         cvShowImage("histogram", Chis);
135         cvSaveImage("histogram.png", Chis);
136         cvReleaseImage(&Chis);
137
138         bin = getBinImagePlus(G , t);
139         cvNamedWindow("Binary Image" , 0);
140         cvShowImage("Binary Image" , bin);
141
142         return bin;
143     }
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160     /*********************************************************************************
161     * From the pointer to a greyscale image, draws a histogram image and returns a pointer  *
162     * to it. Two possible styles can be selected:                                    *
163     * LINES (a line for each bin).                                                   *
164     * RECTS (a rectangle for each bin).                                              *
165     *********************************************************************************/
166
167     IplImage * getHistogram(IplImage * gray, char style) {
168         int dep = 256;
169         int hist[dep];              // Histogram for gray scale image
170         int st = 4;                 // To separate lines in the histograms, for better look ;)
171
172         /// Initialize histograms
173         for (int i = 0 ; i < dep ; i++)
174             hist[i] = 0;
175
176         /// Calculate histogram
177         for(int i = 0 ; i < gray->height ; i++) {
178
179             char * ptr = gray->imageData + i * gray->widthStep;
180
181             for(int j = 0 ; j < gray->width ; j++) {
182                     if(*ptr < 0) {
183                         char c = *ptr;
184                         uchar x = (uchar) c;
185                         hist[x] += 1;
186                     }
187                     else
188                         hist[(unsigned)(*ptr)] += 1;
189                     ptr++;
190             }
191         }
192
```

... with Microcomputer\exercises\Ex4\Ex4\Ex4-CV-functions.cpp

```
193         /// Create image for the histograms
194         IplImage * his = cvCreateImage(cvSize(st * dep  ,600) , IPL_DEPTH_8U , 1);
195         cvSet(his , 0);                     // Initialize image (all black)
196         his->origin = IPL_ORIGIN_BL;      // Set the origin in the bottom left corner
197
198         /// Draw a line/Rectangle for each bin
199         if (style == LINES)
200             for (int i = 0 ; i < dep ; i++)
201                 if (hist[i] != 0)
202                     cvLine(his, cvPoint(i * st, 0), cvPoint(i * st, hist[i] / 10), 150, 1, 4);
203         else if (style == RECTS)
204             for (int i = 0 ; i < dep ; i++)
205                 if (hist[i] != 0)
206                     cvRectangle(his, cvPoint(i*st , hist[i] / 10), cvPoint((i+1)*st,0), 150, -1, 4);
207         else {
208             printf("\nError. No style selected.\n");
209             cvReleaseImage(&his);
210             return NULL;
211         }
212
213         return his;
214     }
215
216
217
218
219
220
221     /*****************************************************************************************
222     * Input: a pointer to a GREY SCALE image, a threshold, 3 pointers to integer for the    *
223     * necessary computation to determine the center of mass.                                *
224     * Forms a binary image, counts the number of pixels set to 0, sums the values of 2      *
225     * coordinates for the pixels set to 0 (used later to calculate the center of mass       *
226     * coordinates).                                                                         *
227     *                                                                                       *
228     * PS. It is possible to calculate the coordinates even in this function but it's done   *
229     * later to respect the structure of the exercise                                        *
230     *****************************************************************************************/
231
232     IplImage * getBinImagePlus(IplImage * img, unsigned t) {
233         IplImage * I = cvCreateImage(cvGetSize(img), IPL_DEPTH_8U , 1);
234         for(int i = 0 ; i < img->height ; i++) {
235
236             char * ptr = img->imageData + i * img->widthStep;
237             char * p = I->imageData + i * I->widthStep;
238
239             for(int j = 0 ; j < img->width ; j++) {
240                     if ((uchar)(*ptr) >= (uchar)t)
241                         *p = 255;
242                     else {
243                         // In this case we are on the object
244                         *p = 0;
245                     }
246                 ptr++;
247                 p++;
248             }
249         }
250         return I;
251     }
```