...1. Image Analysis with Microcomputer\exercises\Ex1\Ex1CV-functions.cpp

```cpp
#include <opencv2/opencv.hpp>
#include <iostream>
#include <string>
#include <stdlib.h>

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

using namespace std;
using namespace cv;



/*****************************************************************************
* Prints relevant information about the uploaded image in standard output.
*****************************************************************************/

void infoDisplay(IplImage * img , char name[]) {
    /// File Name
    printf("File Name: %s\n" , name);

    /// File Size
    long imgSize = img->imageSize;                  // in Bytes, before compression
    imgSize = imgSize / 1000;                       // conversion to KB

    printf("File Size (no compression): %ld KB\n" , imgSize);

    /// Image Format - Stupid but works!
    char * ext = strrchr(name , '.');
    if (!ext)
        printf("No extension found\n");
    else
        printf("Image format: %s\n", ext + 1);

    /// Image Size
    printf("Image Size: %d x %d\n" , img->width , img->height);

    /// Color Format
    if (img->nChannels == 3)
        printf("Colour format: Coloured\n");
    else if (img->nChannels == 1)
        printf("Colour format: Gray Scale\n");
    else
        printf("Colour format: Unknown\n");
}

/*
PROBLEM:
The size stored in iplImage structure doesn't take into account the compression to get a
final jpg image so the size obtained is different. I can’t find any solution that actually
works...
*/
```

```
69  /*********************************************************************************
70  * Takes a pointer to a GREY SCALE image, a style identifier, and a string.
71  * The function draws the histogram by drawings lines or rectangles for each bin
72  * (as specified by 'style') then saves the histogram in an image named as specified by the
73  * string and also display it in a window whose title is also specified by the string.
74  *
75  * NB. Having to do everything by itself, there is some clumsy processing to manage the
76  * name of the file and the title of the window.
77  *********************************************************************************/
78
79  void drawHistogram(char name[] , IplImage * gray , char style) {
80      int dep = 256;
81      int hist[dep];              // Histogram for gray scale image
82      int st = 4;                 // To separate lines in the histograms, for better look ;)
83
84      /// Initialize histograms
85      for (int i = 0 ; i < dep ; i++)
86          hist[i] = 0;
87
88      /// Calculate histogram
89      for(int i = 0 ; i < gray->height ; i++) {
90          char * ptr = gray->imageData + i * gray->widthStep;
91          for(int j = 0 ; j < gray->width ; j++) {
92                  func((uchar)(*ptr) , hist);
93                  ptr++;
94          }
95      }
96
97      /// Create image for the histogram
98      IplImage * his = cvCreateImage(cvSize(st*dep , 600), IPL_DEPTH_8U , 1);
99
100     cvSet(his , 0);                             // Initialize image (all black)
101     his->origin = IPL_ORIGIN_BL;                // Set the origin in the bottom left corner
102
103     cvNamedWindow(name , 2);                    // Create window to contain the image
104
105     /// Draw the histogram - 2 different styles
106     if (style == LINES)
107         for (int i = 0 ; i < dep ; i++)
108             if (hist[i] != 0)
109                 cvLine(his, cvPoint(i*st , 0), cvPoint(i*st , hist[i] / 10) , 150, 1, 4);
110     else if (style == RECTS)
111         for (int i = 0; i < dep; i++)
112             if (hist[i] != 0)
113                 cvRectangle(his, cvPoint(i*st,hist[i] / 10),
114                             cvPoint((i+1)*st, 0), 150, -1, 4);
115     else {
116         printf("\nError. No style selected.\n");
117         cvReleaseImage(&his);
118         return;
119     }
120
121     /// Save Image in png format (next 4 lines to add the extension)
122     int l = strlen(name);
123     char filename[l + 4];
124     strcpy(filename , name);
125     strcat(filename , ".png");
126
127     cvSaveImage(filename, his);
128
129     cvShowImage(name, his);
130
131     cvReleaseImage(&his);
132 }
133
134
135
136
```

```
137 /*
138 NOTE: In the C++ version of OpenCV (maybe also in the C version, but I'm not sure) there
139 are tools to produce a histogram. I decided to write my own function from scratches to
140 get acquainted with the library.
141 */
142
143
144 /*********************************************************************************
145 * Just a support function to 'drawHistogram', 'drawHistChannels', and  'getHistogram'.
146 *********************************************************************************/
147
148 void func(uchar c , int * h) {
149     unsigned x = c;
150     (*(h + x))++;
151 }
152
153 /* In C++ this can be an inline function. */
154
155
156
157
158 /*********************************************************************************
159 * From a BGR image extracts the three channel (grey scale images) and draws and saves
160 * the images and relative histograms.
161 * Two possible styles to draw the histograms: LINES or RECTS
162 *********************************************************************************/
163
164 void drawHistChannels(IplImage * img , char style) {
165     IplImage * R = cvCreateImage(cvSize(img->width , img->height) , IPL_DEPTH_8U , 1);
166     IplImage * G = cvCreateImage(cvSize(img->width , img->height) , IPL_DEPTH_8U , 1);
167     IplImage * B = cvCreateImage(cvSize(img->width , img->height) , IPL_DEPTH_8U , 1);
168
169     cvSplit(img, B, G, R, NULL);
170
171     int dep = 256;
172     int histB[256], histG[256], histR[256];       // Histograms
173     int st = 4;                                   // To separate lines in the histograms
174
175     /// Initialize histograms
176     for (int i = 0 ; i < dep ; i++) {
177         histB[i] = 0;
178         histG[i] = 0;
179         histR[i] = 0;
180     }
181
182     /// Calculate histogram
183     for(int i = 0 ; i < img->height ; i++) {
184         char * ptrB = B->imageData + i * B->widthStep;
185         char * ptrG = G->imageData + i * G->widthStep;
186         char * ptrR = R->imageData + i * R->widthStep;
187         for(int j = 0 ; j < img->width ; j++) {
188                 func((uchar)(*ptrB) , histB);
189                 ptrB++;
190
191                 func((uchar)(*ptrG) , histG);
192                 ptrG++;
193
194                 func((uchar)(*ptrR) , histR);
195                 ptrR++;
196         }
197     }
198
199     /// Create image for the histograms
200     IplImage * hisB = cvCreateImage(cvSize(st * 256, 600) , IPL_DEPTH_8U , 1);
201     IplImage * hisG = cvCreateImage(cvSize(st * 256, 600) , IPL_DEPTH_8U , 1);
202     IplImage * hisR = cvCreateImage(cvSize(st * 256, 600) , IPL_DEPTH_8U , 1);
203
204
```

```
205         /// Initialize image (all black)
206         cvSet(hisB , 0);
207         cvSet(hisG , 0);
208         cvSet(hisR , 0);
209
210         /// Create window to contain the image
211         cvNamedWindow("Blue channel" , 2);
212         cvNamedWindow("Green channel", 2);
213         cvNamedWindow("Red channel"  , 2);
214
215         /// Set the origin in the bottom left corner
216         hisB->origin = IPL_ORIGIN_BL;
217         hisG->origin = IPL_ORIGIN_BL;
218         hisR->origin = IPL_ORIGIN_BL;
219
220         /// Draw histograms - 2 different styles
221         if (style == LINES)
222             for (int i = 0 ; i < dep ; i++) {
223                 if (histB[i] != 0)
224                     cvLine(hisB, cvPoint(i*st, 0), cvPoint(i*st , histB[i] / 10), 150, 1, 4);
225                 if (histG[i] != 0)
226                     cvLine(hisG, cvPoint(i*st, 0), cvPoint(i*st , histG[i] / 10), 150, 1, 4);
227                 if (histR[i] != 0)
228                     cvLine(hisR, cvPoint(i*st, 0), cvPoint(i*st , histR[i] / 10), 150, 1, 4);
229             }
230         else if (style == RECTS)
231             for (int i = 0 ; i < dep ; i++) {
232                 if (histB[i] != 0)
233                     cvRectangle(hisB, cvPoint(i*st , histB[i] / 10),
234                                 cvPoint((i+1)*st , 0), 150, -1, 4);
235                 if (histG[i] != 0)
236                     cvRectangle(hisG, cvPoint(i*st , histG[i] / 10),
237                                 cvPoint((i+1)*st , 0), 150, -1, 4);
238                 if (histR[i] != 0)
239                     cvRectangle(hisR, cvPoint(i*st , histR[i] / 10),
240                                 cvPoint((i+1)*st , 0), 150, -1, 4);
241             }
242         else {
243             printf("\nError. No style selected.\n");
244             cvReleaseImage(&hisB);
245             cvReleaseImage(&hisG);
246             cvReleaseImage(&hisR);
247             return;
248         }
249
250         /// Show on window and save on disk
251         cvShowImage("Blue Hist"  , hisB);
252         cvShowImage("Green Hist" , hisG);
253         cvShowImage("Red Hist"   , hisR);
254
255         cvSaveImage("Blue Channel Histogram.png"  , hisB);
256         cvSaveImage("Green Channel Histogram.png" , hisG);
257         cvSaveImage("Red Channel Histogram.png"   , hisR);
258
259         cvReleaseImage(&B);
260         cvReleaseImage(&G);
261         cvReleaseImage(&R);
262         cvReleaseImage(&hisB);
263         cvReleaseImage(&hisB);
264         cvReleaseImage(&hisB);
265     }
266
267
268
269
270
271
272
```

```
/*****************************************************************************
* Alternative function to extract the channels of a BGR image and to draw the
* histograms.
* This version uses another function to deal with each single channel one at a time.
* PROS: it's easier to understand, maintain, there is not so much repeated code, there
*       are no issues with names, exploits functional organization.
*
* CONS: for each channel it has to scan an entire image, so 3 images of the size must be
*       scanned compared to 'drawHistChannels' which computes the 3 histograms in
*       parallel... It could be nice if one decides to treat only a subset of channels
*       but at present state there is not such flexibility.
*****************************************************************************/

void drawHistChannelsAlt(IplImage * img , char style) {
    IplImage * R = cvCreateImage(cvSize(img->width , img->height) , IPL_DEPTH_8U , 1);
    IplImage * G = cvCreateImage(cvSize(img->width , img->height) , IPL_DEPTH_8U , 1);
    IplImage * B = cvCreateImage(cvSize(img->width , img->height) , IPL_DEPTH_8U , 1);
    IplImage * hisR, * hisG, * hisB;

    cvSplit(img, B, G, R, NULL);

    hisB = getHistogram(B , style);
    hisG = getHistogram(G , style);
    hisR = getHistogram(R , style);

    cvNamedWindow("Blue Hist"  , 2);
    cvNamedWindow("Green Hist" , 2);
    cvNamedWindow("Red Hist"   , 2);

    cvShowImage("Blue Hist"  , hisB);
    cvShowImage("Green Hist" , hisG);
    cvShowImage("Red Hist"   , hisR);

    cvSaveImage("Blue Channel Histogram.png"  , hisB);
    cvSaveImage("Green Channel Histogram.png" , hisG);
    cvSaveImage("Red Channel Histogram.png"   , hisR);

    cvReleaseImage(&B);
    cvReleaseImage(&G);
    cvReleaseImage(&R);
    cvReleaseImage(&hisB);
    cvReleaseImage(&hisB);
    cvReleaseImage(&hisB);
}
```

```
341 /*********************************************************************************************
342 * Forms the histogram for a GREY SCALE image and returns a pointer to the histogram image.
343 * The calling function has to deal with visualization and storing. This function does not
344 * deal with these aspects.
345 *********************************************************************************************/
346
347 IplImage * getHistogram(IplImage * gray , char style) {
348     int dep = 256;
349     int hist[dep];              // Histogram for gray scale image
350     int st = 4;                 // To separate lines in the histograms, for better look ;)
351
352     /// Initialize histograms
353     for (int i = 0 ; i < dep ; i++)
354         hist[i] = 0;
355
356     /// Calculate histogram
357     for(int i = 0 ; i < gray->height ; i++) {
358         char * ptr = gray->imageData + i * gray->widthStep;
359         for(int j = 0 ; j < gray->width ; j++) {
360                 func((uchar)(*ptr) , hist);
361                 ptr++;
362         }
363     }
364
365     /// Create image for the histograms
366     IplImage * his = cvCreateImage(cvSize(st*dep,600) , IPL_DEPTH_8U , 1);
367     cvSet(his , 0);  // Initialize image (all black)
368
369     /// Create window to contain the image
370     //cvNamedWindow("Histogram", 2);
371
372     /// Set the origin in the bottom left corner
373     his->origin = IPL_ORIGIN_BL;
374
375     /// Draw a line/rectangle for each bin
376     if (style == LINES)
377         for (int i = 0 ; i < dep ; i++)
378             if (hist[i] != 0)
379                 cvLine(his, cvPoint(i*st, 0) , cvPoint(i*st , hist[i] / 10), 150, 1, 4);
380     else if (style == RECTS)
381         for (int i = 0 ; i < dep ; i++)
382             if (hist[i] != 0)
383                 cvRectangle(his, cvPoint(i*st , hist[i] / 10),
384                             cvPoint((i+1)*st , 0), 150, -1, 4);
385     else {
386         printf("\nError. No style selected.\n");
387         cvReleaseImage(&his);
388         return NULL;
389     }
390
391     /// Show on window
392     //cvShowImage(name, his);
393     //name = strcat(name, ".png");
394     //cvSaveImage(name, his);
395     //cvReleaseImage(&his);
396     return his;
397 }
```