

...ce\repos\Assignment5-1-v2\Assignment5-1-v2\Assignment5-1-v2.cpp

```
1  /*****
2  *                                     EXERCISE 1                                     *
3  *                                     -----                                     *
4  *                                     IMAGE ACQUISITION AND DISPLAY                                     *
5  *****/
6
7  #include <opencv2/opencv.hpp>
8  #include <iostream>
9  #include <string>
10 #include <stdlib.h>
11 #include "Ex1CV-header.h"
12
13 using namespace std;
14 using namespace cv;
15
16 int main(int argc, char *argv[]) {
17
18     /*****
19     *                                     PART 1                                     *
20     *                                     -----                                     *
21     *                                     Loading an image from file                                     *
22     *****/
23
24     /// Load image from file specified from command prompt
25     IplImage * img;
26     img = cvLoadImage(argv[1] , 1);
27
28     /// Displays relevant information on command prompt
29     infoDisplay(img , argv[1]);
30
31
32     /*****
33     *                                     PART 2                                     *
34     *                                     -----                                     *
35     *                                     Image Display & Histogram                                     *
36     *****/
37
38     /// _____Display_____
39
40     const char * windowName = argv[1];
41     cvNamedWindow(windowName , 0);
42     cvShowImage(argv[1] , img);
43
44
45     /// _____Conversion RGB to Grayscale & Display_____
46
47     const char * windowGray = "Gray Scale";
48     cvNamedWindow(windowGray , 1);
49     IplImage * gray = cvCreateImage(cvGetSize(img) , IPL_DEPTH_8U , 1);
50     cvCvtColor(img , gray , CV_RGB2GRAY);
51     cvShowImage(windowGray , gray);
52
53
54
55
56
57     // Alternatively: Upload directly in GS & Display.
58
59     // BEWARE: The result image has different pixel values AFTER saving it
60     // compared to that obtained from a conversion!
61
62     /*
63     const char * wGray1 = "Gray Scale 1";
64     cvNamedWindow(windowGray1 , 0);
65     IplImage * gray = cvLoadImage(argv[1] , CV_LOAD_IMAGE_GRAYSCALE);
66     cvShowImage(windowGray , gray);
67     */
68 }
```

...ce\repos\Assignment5-1-v2\Assignment5-1-v2\Assignment5-1-v2.cpp

```
69
70     /// _____Histogram_____
71
72     /**
73     NB: Although not required by PART 2, each channel of the colour image is isolated and
74     shown (in grayscale) and a histogram for each is shown.
75     It is required in part 3 for a frame captured with a webcam but I decided to move it
76     here because many computation can be done in parallel.
77     */
78
79
80     /// _____Extracting and showing the channels_____
81     /*
82     IplImage * R = cvCreateImage(cvSize(img->width , img->height) , IPL_DEPTH_8U , 1);
83     IplImage * G = cvCreateImage(cvSize(img->width , img->height) , IPL_DEPTH_8U , 1);
84     IplImage * B = cvCreateImage(cvSize(img->width , img->height) , IPL_DEPTH_8U , 1);
85
86     cvSplit(img , B , G , R , NULL);
87
88     cvNamedWindow("Red" , 0);
89     cvNamedWindow("Green" , 0);
90     cvNamedWindow("Blue" , 0);
91
92     cvShowImage("Blue" , B);
93     cvShowImage("Green" , G);
94     cvShowImage("Red" , R);
95     */
96
97     /// _____Histograms_____
98
99
100     char histogramName[] = "Histogram";
101     drawHistogram(histogramName , gray , LINES);
102     //drawHistChannels(img , LINES);
103     //drawHistChannelsAlt(img , LINES);
104
105
106
107     /*****
108     *                               *
109     *           PART 3              *
110     *           -----            *
111     *           Image acquisition from cameras          *
112     *****/
113
114     /// _____Display stream from webcam on a window_____
115     /// _____Capture an image from a webcam_____
116
117     /*****
118     // DOESN'T WORK ON MY PC
119
120     int c;
121     IplImage * color_img;
122     CvCapture * cv_cap = cvCaptureFromCAM(-1); // -1 = only one cam or doesn't matter
123     cvNamedWindow("Video" , 0); // create window
124     for(;;) {
125         color_img = cvQueryFrame(cv_cap); // get frame
126         if(color_img != 0)
127             cvShowImage("Video" , color_img); // show frame
128         c = cvWaitKey(10); // wait 10 ms or for key stroke
129         if(c == 27)
130             break; // if ESC, break and quit
131     }
132     // clean up
133     cvReleaseCapture( &cv_cap );
134     cvDestroyWindow("Video");
135     */
136
```

...ce\repos\Assignment5-1-v2\Assignment5-1-v2\Assignment5-1-v2.cpp

```
137 //*****
138
139 // DOESN'T WORK ON MY PC - EMPTY WINDOW
140
141 /*
142 const char * cName = "Hello world!";
143 cvNamedWindow(cName , 0);
144 CvCapture * capture = 0;
145 double capProp = 0;
146 IplImage *frame, *frame_copy = 0;
147 capture = cvCaptureFromCAM(1);
148 if (capture) {
149     for (;;) {
150         if ( !cvGrabFrame( capture ) )
151             break;
152         frame = cvRetrieveFrame( capture );
153         if ( !frame )
154             break;
155         if ( !frame_copy ) {
156             printf("\nFrame settings:\n Width: %d\n Height: %d\n",
157                 frame->width , frame->height);
158             frame_copy = cvCreateImage(cvSize(frame->width , frame->height),
159                                     IPL_DEPTH_8U , frame->nChannels);
160             cvResizeWindow(wName , frame->width , frame->height);
161         }
162
163         if ( frame->origin == IPL_ORIGIN_TL )
164             cvCopy ( frame , frame_copy , 0 );
165         else
166             cvFlip ( frame, frame_copy, 0);
167         cvShowImage(wName, frame_copy);
168         cvWaitKey(30);
169         //if (cvWaitKey(5)>0)
170             // break;
171     }
172 }
173 */
174
175
176 //*****
177
178
179 // WORKS IN THE LAB; DOESN'T WORK ON MY PC... CRASHES OR TAKES AN EMPTY FRAME BUT ONLY
180 // IF THE CAMERA IS TURNED ON...
181
182 /*
183 IplImage * frame;
184 CvCapture * capture = 0;
185 // initialize capture device
186 //capture = cvCaptureFromCAM(0);
187 capture = cvCreateCameraCapture(-1);
188 if( !cvGrabFrame(capture) )
189     printf("error");
190 frame = cvRetrieveFrame(capture);
191
192     printf("\nFrame settings:\n Width: %d\n Height: %d\n", frame->width, frame-
193 >height);
194
195     cvNamedWindow("Cam" , 2);
196     cvResizeWindow("Cam" , frame->width , frame->height);
197     cvShowImage("Cam" , frame);
198     cvWaitKey(30);
199     cvSaveImage("frame.png" , frame);
200 */
201
202
203
204
```

...ce\repos\Assignment5-1-v2\Assignment5-1-v2\Assignment5-1-v2.cpp

```
//*****
205
206
207 // STREAM FROM WEBCAM - WORKS ONLY IF THE CAMERA IS OFF; IF NOT, IT SHOWS A BLACK
208 // WINDOW
209 // PS. IT'S C++.
210
211 VideoCapture cap(1);
212
213 // cap is an object and to control the properties of the camera the methods set
214 // and get can be used.
215 // Images are treated as objects of the class 'Mat' in the C++ version of OpenCV.
216
217 Mat frame;
218 if ( !cap.isOpened() ) {
219     std::cout << "Cam could not be accessed" << std::endl;
220     return -1;
221 }
222 namedWindow("Cam");
223 while( cap.read(frame) ) {
224     imshow("Cam" , frame);
225     if ( waitKey(10) >= 0 ) {
226         IplImage * im_cam = new IplImage(frame);
227         cvSaveImage("Cam.png" , im_cam);
228         break;
229     }
230     if( frame.empty() ) {
231         std::cout << "End of stream" << std::endl;
232         break;
233     }
234 }
235
236
237 //*****
238
239 // CAPTURE AN IMAGE FROM WEBCAM - WORKS ONLY IF THE CAMERA IS OFF. IF NOT, CAPTURES A
240 // BLACK IMAGE...
241 // PS. IT'S C++
242
243 VideoCapture cap(1);
244 Mat frameCaptured;
245 if ( !cap.isOpened() ) {
246     std::cout << "Cam could not be accessed" << std::endl;
247     return -1;
248 }
249 namedWindow("Cam");
250 cap.read(frameCaptured);
251 imshow("Cam" , frameCaptured);
252 IplImage * im_cam_captured = new IplImage(frameCaptured);
253 cvNamedWindow("CamIPL" , 0);
254 cvShowImage("CamIPL" , im_cam_captured);
255
256
257 //*****
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
```

...ce\repos\Assignment5-1-v2\Assignment5-1-v2\Assignment5-1-v2.cpp

```
273  /******
274  *                                PART 4                                *
275  *                                -----                                *
276  *                                Image storage                            *
277  *****/
278
279  /*
280  cvSaveImage("Grey Scale.png" , gray);
281  cvSaveImage("Blue.png" , B);
282  cvSaveImage("Red.png" , R);
283  cvSaveImage("Green.png" , G);
284  //cvSaveImage("Cam.png" , im_cam_capture);
285  //imwrite("frame.png" , frameCaptured);    // If you're using the C++ solution
286  // without converting to IplImage.
287  */
288
289
290  /// _____ Clean-up _____
291
292
293  cvWaitKey(0);
294
295  cvReleaseImage(&img);
296  cvReleaseImage(&gray);
297  cvReleaseImage(&im_cam_captured);
298  //cvReleaseImage(&R);
299  //cvReleaseImage(&G);
300  //cvReleaseImage(&B);
301  //cvReleaseImage(&im_cam);
302  //cvReleaseImage(&frameCopy);
303  //cvReleaseImage(&frameC);
304
305  cvDestroyAllWindows();
306
307  return 0;
308 }
```