

```

1  #include <opencv2/opencv.hpp>
2  #include <iostream>
3  #include <string>
4  #include <stdlib.h>
5
6  #define LINES 'L'
7  #define RECTS 'R'
8
9  using namespace std;
10 using namespace cv;
11
12  /*****
13  *                               EXERCISE 1                               *
14  *                               -----                               *
15  *                               IMAGE ACQUISITION AND DISPLAY          *
16  *                               *****/
17
18  void printing(IplImage *, char []);
19  void drawHistogram(char [], IplImage *, char);
20  void drawHistChannels(IplImage *, char);
21  void drawHistChannelsAlt(IplImage *, char);
22  void func(uchar, int *);
23
24  IplImage * getHistogram(IplImage *, char); // Not used in this exercise
25
26  int main(int argc, char *argv[]) {
27
28  /*****
29  *                               PART 1                               *
30  *                               -----                               *
31  *                               Loading an image from file            *
32  *                               *****/
33
34  // Load image from file specified from command prompt
35  IplImage * img;
36  img = cvLoadImage(argv[1], 1);
37
38  // Displays relevant information on command prompt
39  printing(img, argv[1]);
40
41
42  /*****
43  *                               PART 2                               *
44  *                               -----                               *
45  *                               Image Display & Histogram            *
46  *                               *****/
47
48  //_____Display_____
49
50  const char * wName = argv[1];
51  cvNamedWindow(wName, 0);
52  cvShowImage(argv[1], img);
53
54  //_____Conversion RGB to GS & Display_____
55
56  const char * wGray = "Gray Scale";
57  cvNamedWindow(wGray, 1);
58  IplImage * gray = cvCreateImage(cvGetSize(img), IPL_DEPTH_8U, 1);
59  cvCvtColor(img, gray, CV_RGB2GRAY);
60  cvShowImage(wGray, gray);
61
62  // Alternatively: Upload directly in GS & Display.
63  // BEWARE: The resultant image has different pixel values AFTER saving it
64  // compared to that obtained from a conversion!
65  /*-----
66  const char * wGray1 = "Gray Scale 1";
67  cvNamedWindow(wGray1, 0);
68
69  IplImage * gray = cvLoadImage(argv[1], CV_LOAD_IMAGE_GRAYSCALE);
70  cvShowImage(wGray, gray);
71  //-----
72  /**/
73
74  //_____Histogram_____
75
76  /**
77  NB: Although not required by PART 2, each channel of the colour image is
78  isolated and shown (in gray scale) and a histogram for each is shown.
79  It is required in part 3 for a frame captured with a webcam but I decided
80  to move it here because many computation can be done in parallel.
81  */
82
83  // Extracting and showing the channels
84  /*IplImage * R = cvCreateImage(cvSize(img->width, img->height), IPL_DEPTH_8U, 1);

```

```

85     IplImage * G = cvCreateImage(cvSize(img->width, img->height), IPL_DEPTH_8U, 1);
86     IplImage * B = cvCreateImage(cvSize(img->width, img->height), IPL_DEPTH_8U, 1);
87
88     cvSplit(img, B, G, R, NULL);
89
90     cvNamedWindow("Red", 0);
91     cvNamedWindow("Green", 0);
92     cvNamedWindow("Blue", 0);
93
94     cvShowImage("Blue", B);
95     cvShowImage("Green", G);
96     cvShowImage("Red", R);*/
97
98     /// histograms
99     char hName[] = "Histogram";
100    drawHistogram(hName, gray, LINES);
101    //drawHistChannels(img, LINES);
102    //drawHistChannelsAlt(img, LINES);
103
104
105    /*****
106    *
107    *
108    *
109    *****/
110
111    ///          Display stream from webcam on a window
112    ///          Capture an image from a webcam
113
114    /*****
115
116    /*          DOESN'T WORK ON MY PC
117    int c;
118    IplImage* color_img;
119    CvCapture* cv_cap = cvCaptureFromCAM(-1); // -1 = only one cam or doesn't matter
120    cvNamedWindow("Video",0); // create window
121    for(;;) {
122        color_img = cvQueryFrame(cv_cap); // get frame
123        if(color_img != 0)
124            cvShowImage("Video", color_img); // show frame
125        c = cvWaitKey(10); // wait 10 ms or for key stroke
126        if(c == 27)
127            break; // if ESC, break and quit
128    }
129    // clean up
130    cvReleaseCapture( &cv_cap );
131    cvDestroyWindow("Video");
132    /**/
133
134    /*****
135
136    /*          DOESN'T WORK ON MY PC - EMPTY WINDOW
137    const char * cName="Hello world!";
138    cvNamedWindow(cName, 0);
139    CvCapture* capture = 0;
140    double capProp=0;
141    IplImage *frame, *frame_copy=0;
142    capture = cvCaptureFromCAM(1);
143    if (capture) {
144        for (;;) {
145            if ( !cvGrabFrame( capture ) )
146                break;
147            frame = cvRetrieveFrame( capture );
148            if ( !frame )
149                break;
150            if ( !frame_copy ) {
151                printf("\nFrame settings:\n Width: %d\n Height: %d\n",
152                    frame->width, frame->height);
153                frame_copy = cvCreateImage (cvSize(frame->width,frame->height),
154                    IPL_DEPTH_8U, frame->nChannels);
155                cvResizeWindow(wName,frame->width,frame->height);
156            }
157            if ( frame->origin == IPL_ORIGIN_TL )
158                cvCopy ( frame, frame_copy, 0 );
159            else
160                cvFlip ( frame, frame_copy, 0);
161            cvShowImage(wName, frame_copy);
162            cvWaitKey(30);
163            //if (cvWaitKey(5)>0)
164                // break;
165        }
166    }
167    /**/

```

```

168 //*****
169 /*          WORKS IN THE LAB, DOESN'T WORK ON MY PC - CRASHES OR
170 //          TAKES AN EMPTY FRAME BUT ONLY IF THE CAMERA IS TURNED ON...
171
172 IplImage * frame;
173 CvCapture * capture = 0;
174 // initialize capture device
175 //capture = cvCaptureFromCAM(0);
176 capture = cvCreateCameraCapture(-1);
177 if(!cvGrabFrame(capture))
178     printf("error");
179 frame = cvRetrieveFrame(capture);
180
181 printf("\nFrame settings:\n Width: %d\n Height: %d\n", frame->width, frame->height);
182
183 cvNamedWindow("Cam", 2);
184 cvResizeWindow("Cam", frame->width, frame->height);
185 cvShowImage("Cam", frame);
186 cvWaitKey(30);
187 cvSaveImage("frame.png", frame);
188
189 /**/
190
191 //*****
192 //          STREAM FROM WEBCAM - WORKS ONLY IF THE CAMERA IS OFF
193 //          IF NOT, SHOWS A BLACK WINDOW...
194 //          ...AND IT'S C++
195 /*
196 VideoCapture cap(1);
197
198 /// cap is an object and to control the properties of the camera the methods set
199 /// and get can be used:
200 ///          cap.set(...);
201 ///          cap.get(...);
202
203 Mat frame;
204 if (!cap.isOpened()) {
205     printf("Cam could not be accessed\n");
206     return -1;
207 }
208 namedWindow("Cam");
209 while(cap.read(frame)) {
210     imshow("Cam", frame);
211     if (waitKey(10) >= 0) {
212         IplImage * im_cam = new IplImage(frame);
213         cvSaveImage("Cam.png", im_cam);
214         break;
215     }
216     if(frame.empty()) {
217         printf("End of stream\n");
218         break;
219     }
220 }
221
222 */
223 //*****
224 /*          CAPTURE AN IMAGE FROM WEBCAM - WORKS ONLY IF THE CAMERA IS OFF.
225 //          IF NOT, CAPTURES A BLACK IMAGE...
226 //          ...AND IT'S C++
227
228 VideoCapture cap(1);
229 Mat frame;
230 if (!cap.isOpened()) {
231     cout <<"Cam could not be opened"<<endl;
232     return -1;
233 }
234 namedWindow("Cam");
235 cap.read(frame);
236 imshow("Cam", frame);
237 IplImage * im_cam = new IplImage(frame);
238 cvNamedWindow("CamIPL", 0);
239 cvShowImage("CamIPL", im_cam);
240 /**/
241
242 //*****
243
244 //*****
245 *          PART 4          *
246 *          -----          *
247 *          Image storage          *
248 //*****
249 /*cvSaveImage("Grey Scale.png", gray);
250 cvSaveImage("Blue.png", B);
251 cvSaveImage("Red.png", R);

```

```

252 cvSaveImage("Green.png", G);
253 //cvSaveImage("Cam.png", im_cam);
254 //imwrite("frame.png", frame); // If you're using the C++ solution without
255 // without converting to IplImage.
256
257 */
258 //_____Clean-up_____
259 cvWaitKey(0);
260
261 cvReleaseImage(&img);
262 cvReleaseImage(&gray);
263 //cvReleaseImage(&R);
264 //cvReleaseImage(&G);
265 //cvReleaseImage(&B);
266 //cvReleaseImage(&im_cam);
267 //cvReleaseImage(&frameCopy);
268 //cvReleaseImage(&frameC);
269
270 cvDestroyAllWindows();
271
272 return 0;
273 }
274
275
276
277 /*****
278 * Simply shows in the command prompt relevant information about the uploaded *
279 * image.
280 *****/
281 void printing(IplImage * img, char name[]) {
282     /// File Name
283     printf("File Name: %s\n", name);
284
285     /// File Size
286     long imgSize = img->imageSize; // in Bytes, before compression
287     imgSize = imgSize / 1000; // conversion to KB
288     printf("File Size (no compression): %ld KB\n", imgSize);
289     /** PROBLEM: The size stored in iplImage structure doesn't take into account
290     the compression to get a final jpg image so the size obtained is different.
291     I can't find any solution that actually works...*/
292
293     /// Image Format - Stupid but works!
294     char * ext = strrchr(name, '.');
295     if (!ext)
296         printf("No extension found\n");
297     else
298         printf("Image format: %s\n", ext + 1);
299
300     /// Image Size
301     printf("Image Size: %d x %d\n", img->width, img->height);
302
303     /// Color Format - Not very nice...
304     if (img->nChannels == 3)
305         printf("Colour format: Coloured\n");
306     else if (img->nChannels == 1)
307         printf("Colour format: Gray Scale\n");
308     else
309         printf("Colour format: Unknown\n");
310 }
311
312
313 /*****
314 * Takes a pointer to a GREY SCALE image, a style identifier, and a string. *
315 * The function draws the histogram by drawings lines or rectangles for each *
316 * bin (as specified by 'style') then saves the histogram in an image named *
317 * as specified by the string and also display it in a window whose title is *
318 * specified by the string as well.
319 *
320 * NB. Having to do everything by itself, there is some clumsy processing to *
321 * manage the name of the file and the title of the window.
322 *****/
323 void drawHistogram(char name[], IplImage * gray, char style) {
324     int dep = 256;
325     int hist[dep]; // Histogram for gray scale image
326     int st = 4; // To separate lines in the histograms, for better look ;)
327
328     /// Initialize histograms
329     for (int i = 0; i < dep; i++)
330         hist[i] = 0;
331
332     /// Calculate histogram
333     for(int i = 0; i < gray->height; i++) {
334         char * ptr = gray->imageData + i*gray->widthStep;
335         for(int j = 0; j < gray->width; j++) {

```

```

336         func((uchar)(*ptr), hist);
337         ptr++;
338     }
339 }
340
341     /// Create image for the histogram
342     IplImage * his = cvCreateImage(cvSize(st*dep,600), IPL_DEPTH_8U, 1);
343     cvSet(his, 0); // Initialize image (all black)
344     his->origin = IPL_ORIGIN_BL; // Set the origin in the bottom left corner
345
346     cvNamedWindow(name, 2); // Create window to contain the image
347
348     /// Draw the histogram - 2 different styles
349     if (style == LINES)
350         for (int i = 0; i < dep; i++)
351             if (hist[i] != 0)
352                 cvLine(his, cvPoint(i*st, 0), cvPoint(i*st, hist[i]/10), 150, 1, 4);
353     else if (style == RECTS)
354         for (int i = 0; i < dep; i++)
355             if (hist[i] != 0)
356                 cvRectangle(his, cvPoint(i*st, hist[i]/10), cvPoint((i+1)*st, 0), 150, -1,
4);
357     else {
358         printf("\nError. No style selected.\n");
359         cvReleaseImage(&his);
360         return;
361     }
362
363     /// Save Image in png format (next 4 lines to add the extension)
364     int l = strlen(name);
365     char filename[l + 4];
366     strcpy(filename, name);
367     strcat(filename, ".PNG");
368
369     cvSaveImage(filename, his);
370
371     cvShowImage(name, his);
372
373     cvReleaseImage(&his);
374 }
375
376
377     /*****
378     * Just a support function to 'drawHistogram', 'drawHistChannels', and
379     * 'getHistogram'.
380     *****/
381 void func(uchar c, int * h) {
382     unsigned x = c;
383     (*(h + x))++;
384 }
385
386
387     /*****
388     * From a BGR image extracts the three channel (grey scale images) and draws
389     * and saves the images and relative histograms.
390     * Two possible styles to draw the histograms: LINES or RECTS
391     *****/
392 void drawHistChannels(IplImage * img, char style) {
393     IplImage * R = cvCreateImage(cvSize(img->width, img->height), IPL_DEPTH_8U, 1);
394     IplImage * G = cvCreateImage(cvSize(img->width, img->height), IPL_DEPTH_8U, 1);
395     IplImage * B = cvCreateImage(cvSize(img->width, img->height), IPL_DEPTH_8U, 1);
396     cvSplit(img, B, G, R, NULL);
397     int dep = 256;
398     int histB[256], histG[256], histR[256]; // Histograms
399     int st = 4; // To separate lines in the histograms
400
401     /// Initialize histograms
402     for (int i = 0; i < dep; i++) {
403         histB[i] = 0;
404         histG[i] = 0;
405         histR[i] = 0;
406     }
407
408     /// Calculate histogram
409     for (int i = 0; i < img->height; i++) {
410         char * ptrB = B->imageData + i * B->widthStep;
411         char * ptrG = G->imageData + i * G->widthStep;
412         char * ptrR = R->imageData + i * R->widthStep;
413         for (int j = 0; j < img->width; j++) {
414             func((uchar)(*ptrB), histB);
415             ptrB++;
416
417             func((uchar)(*ptrG), histG);
418             ptrG++;

```

```

419         func((uchar)(*ptrR), histR);
420         ptrR++;
421     }
422 }
423
424
425     /// Create image for the histograms
426 IplImage* hisB = cvCreateImage(cvSize(st * 256, 600), IPL_DEPTH_8U, 1);
427 IplImage* hisG = cvCreateImage(cvSize(st * 256, 600), IPL_DEPTH_8U, 1);
428 IplImage* hisR = cvCreateImage(cvSize(st * 256, 600), IPL_DEPTH_8U, 1);
429
430     /// Initialize image (all black)
431 cvSet(hisB, 0);
432 cvSet(hisG, 0);
433 cvSet(hisR, 0);
434
435     /// Create window to contain the image
436 cvNamedWindow("Blue channel", 2);
437 cvNamedWindow("Green channel", 2);
438 cvNamedWindow("Red channel", 2);
439
440     /// Set the origin in the bottom left corner
441 hisB->origin = IPL_ORIGIN_BL;
442 hisG->origin = IPL_ORIGIN_BL;
443 hisR->origin = IPL_ORIGIN_BL;
444
445     /// Draw histograms - 2 different styles
446 if (style == LINES)
447     for (int i = 0; i < dep; i++) {
448         if (histB[i] != 0)
449             cvLine(hisB, cvPoint(i*st, 0), cvPoint(i*st, histB[i] / 10), 150, 1, 4);
450         if (histG[i] != 0)
451             cvLine(hisG, cvPoint(i*st, 0), cvPoint(i*st, histG[i] / 10), 150, 1, 4);
452         if (histR[i] != 0)
453             cvLine(hisR, cvPoint(i*st, 0), cvPoint(i*st, histR[i] / 10), 150, 1, 4);
454     }
455 else if (style == RECTS)
456     for (int i = 0; i < dep; i++) {
457         if (histB[i] != 0)
458             cvRectangle(hisB, cvPoint(i*st, histB[i]/10), cvPoint((i+1)*st, 0), 150,
-1, 4);
459         if (histG[i] != 0)
460             cvRectangle(hisG, cvPoint(i*st, histG[i]/10), cvPoint((i+1)*st, 0), 150,
-1, 4);
461         if (histR[i] != 0)
462             cvRectangle(hisR, cvPoint(i*st, histR[i]/10), cvPoint((i+1)*st, 0), 150,
-1, 4);
463     }
464 else {
465     printf("\nError. No style selected.\n");
466     cvReleaseImage(&hisB);
467     cvReleaseImage(&hisG);
468     cvReleaseImage(&hisR);
469     return;
470 }
471
472     /// Show on window and save on disk
473 cvShowImage("Blue Hist", hisB);
474 cvShowImage("Green Hist", hisG);
475 cvShowImage("Red Hist", hisR);
476 cvSaveImage("Blue Channel Histogram.png", hisB);
477 cvSaveImage("Green Channel Histogram.png", hisG);
478 cvSaveImage("Red Channel Histogram.png", hisR);
479
480 cvReleaseImage(&B);
481 cvReleaseImage(&G);
482 cvReleaseImage(&R);
483 cvReleaseImage(&hisB);
484 cvReleaseImage(&hisG);
485 cvReleaseImage(&hisR);
486 }
487
488
489 /*****
490 * Alternative function to extract the channels of a BGR image and to draw
491 * the histograms.
492 * This version uses another function to deal with each single channel one at
493 * a time.
494 *
495 * PROS: it's easier to understand, maintain, there is not much repeated code
496 * there are no issues with names, exploits functional organization.
497 * CONS: for each channel it has to scan an entire image, so 3 images of the
498 * size must be scanned compared to 'drawHistChannels' which computes
499 * the 3 histograms in parallel... It could be nice if one decides to
500 */

```

```

500      *      treat only a subset of channels, but at present state there is not      *
501      *      such flexibility.                                                         *
502      *****/
503 void drawHistChannelsAlt(IplImage * img, char style) {
504     IplImage * R = cvCreateImage(cvSize(img->width, img->height), IPL_DEPTH_8U, 1);
505     IplImage * G = cvCreateImage(cvSize(img->width, img->height), IPL_DEPTH_8U, 1);
506     IplImage * B = cvCreateImage(cvSize(img->width, img->height), IPL_DEPTH_8U, 1);
507     IplImage * hisR, * hisG, * hisB;
508
509     cvSplit(img, B, G, R, NULL);
510
511     hisB = getHistogram(B, style);
512     hisG = getHistogram(G, style);
513     hisR = getHistogram(R, style);
514
515     cvNamedWindow("Blue Hist", 2);
516     cvNamedWindow("Green Hist", 2);
517     cvNamedWindow("Red Hist", 2);
518
519     cvShowImage("Blue Hist", hisB);
520     cvShowImage("Green Hist", hisG);
521     cvShowImage("Red Hist", hisR);
522
523     cvSaveImage("Blue Channel Histogram.png", hisB);
524     cvSaveImage("Green Channel Histogram.png", hisG);
525     cvSaveImage("Red Channel Histogram.png", hisR);
526
527     cvReleaseImage(&B);
528     cvReleaseImage(&G);
529     cvReleaseImage(&R);
530     cvReleaseImage(&hisB);
531     cvReleaseImage(&hisG);
532     cvReleaseImage(&hisR);
533 }
534
535
536
537      *****/
538      *      Forms the histogram for a GREY SCALE image and returns a pointer to the      *
539      *      histogram image.                                                         *
540      *      The calling function has to deal with visualization and storing. This      *
541      *      function does not deal with these aspects.                               *
542      *****/
543 IplImage * getHistogram(IplImage * gray, char style) {
544     int dep = 256;
545     int hist[dep]; // Histogram for gray scale image
546     int st = 4;    // To separate lines in the histograms, for better look ;)
547
548     /// Initialize histograms
549     for (int i = 0; i < dep; i++)
550         hist[i] = 0;
551
552     /// Calculate histogram
553     for (int i = 0; i < gray->height; i++) {
554         char * ptr = gray->imageData + i*gray->widthStep;
555         for (int j = 0; j < gray->width; j++) {
556             func((uchar) (*ptr), hist);
557             ptr++;
558         }
559     }
560
561     /// Create image for the histograms
562     IplImage * his = cvCreateImage(cvSize(st*dep, 600), IPL_DEPTH_8U, 1);
563     cvSet(his, 0); // Initialize image (all black)
564
565     /// Create window to contain the image
566     cvNamedWindow("Histogram", 2);
567
568     /// Set the origin in the bottom left corner
569     his->origin = IPL_ORIGIN_BL;
570
571     /// Draw a line/rectangle for each bin
572     if (style == LINES)
573         for (int i = 0; i < dep; i++)
574             if (hist[i] != 0)
575                 cvLine(his, cvPoint(i*st, 0), cvPoint(i*st, hist[i]/10), 150, 1, 4);
576     else if (style == RECTS)
577         for (int i = 0; i < dep; i++)
578             if (hist[i] != 0)
579                 cvRectangle(his, cvPoint(i*st, hist[i]/10), cvPoint((i+1)*st, 0), 150, -1,
580 4);
581     else {
582         printf("\nError. No style selected.\n");
583         cvReleaseImage(&his);
584         return NULL;
585     }
586 }

```

```
583     }
584
585     /// Show on window
586
587     //cvShowImage(name, his);
588     //name = strcat(name, ".png");
589     //cvSaveImage(name, his);
590     //cvReleaseImage(&his);
591     return his;
592 }
593
```