```
1 #include <iostream>
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <string.h>
5 #include <opencv2/opencv.hpp>
  typedef struct THR {
7
8
     int B;
     int x;
9
10 } THR;
11
12
13 using namespace std;
  using namespace cv;
15
16
17 void preprocessing (const char *);
18 void thresholding (const char *);
19 void multi(const char *);
20 int findBackground(int *, int);
21 void drawHistogram(IplImage *, int *);
22 IplImage * simpleThresholding(IplImage *, int);
23 IplImage * getBinImage(IplImage *, unsigned);
24
25
26
  int main(int argc, char ** argv) {
27
     const char * imgName = argv[1];
28
     preprocessing(imgName);
29
     return 0;
30 }
31
32
33
34
35
36
37
38
39
40
  42
43
44
45
46
47
48
49
50
51
  void preprocessing(const char * imgName) {
     IplImage * img = cvLoadImage(imgName, CV_LOAD_IMAGE_GRAYSCALE);
52
53
54
     MORPHOLOGICAL TRANSFORMATION
55
     56
```

```
...nalysis with Microcomputer\Project\4. Calibration\Cal1\main.cpp
```

```
57
58
     IplImage * CL = cvCreateImage(cvGetSize(img), IPL_DEPTH_8U, 1);
59
     cvSet(CL, 0);
60
61
     int IT = 12;
62
63
     cvMorphologyEx(img, CL, NULL, NULL, CV_MOP_OPEN, IT);
64
     cvErode(CL, CL, NULL, 10);
     cvDilate(CL, CL, NULL, 3);
65
66
67
     68
     ///
                      DISPLAY AND SAVE
69
     70
71
72
     cvSaveImage("temp.png", CL);
73
74
     thresholding("temp.png");
75
76
77
     78
                         CLEAN-UP
79
     80
81
     cvDestroyAllWindows();
82
83
     cvReleaseImage(&CL);
84 }
85
86
87
88
89
90
91
92
93
94
95
   97
98
99
100
101
102
103
104
105
  void thresholding(const char * imgName) {
106
107
                  ORIGINAL IMAGE UPLOAD
108
     IplImage * img = cvLoadImage(imgName, CV_LOAD_IMAGE_GRAYSCALE);
109
     ///
                       HISTOGRAM
110
     int dep = 256;
111
     int hist[dep];
112
113
     drawHistogram(img, hist);
114
```

```
...nalysis with Microcomputer\Project\4. Calibration\Cal1\main.cpp
```

```
3
```

```
115
                        COMPUTE THE BACKGROUND = B
116
       ///
117
       int B = findBackground(hist, img->height * img->width);
118
119
120
       int t = 8, T;
121
       IplImage * bin;
122
       char imNum[2];
123
       char c;
124
       int i = 0;
125
       while(t <= 20) {</pre>
126
           char imName[30] = "";
127
128
           T = t + B;
           bin = simpleThresholding(img, T);
129
130
           strcat(imName, "bin");
           imNum[0] = 49 + i;
131
132
           imNum[1] = ' \ 0';
133
           strcat(imName, imNum);
           strcat(imName, ".png");
134
           cvSaveImage(imName, bin);
135
136
           multi(imName);
137
           i++;
138
           t += 2;
       }
139
140
       ///
                                 CLEAN-UP
141
       cvReleaseImage(&img);
142
       cvReleaseImage(&bin);
143
       //cvReleaseImage(&bin1);
144
       //cvReleaseImage(&bin2);
145
146 }
147
148
149
150
151
152
153
154
155
156
158
159
160
161
162
163
164
165
166
167
    void func(uchar c, int * h) {
168
169
       unsigned x = c;
170
       (*(h + x))++;
171 }
172
```

```
173
174
175
176 void drawHistogram(IplImage * gray, int hist []) {
177
        int st = 4;
                       // To separate lines in the histograms, for better look;)
        int dep = 256;
178
179
           /// Initialize histograms
        for (int i = 0; i < dep; i++)</pre>
180
181
           hist[i] = 0;
182
183
           /// Calculate histogram
184
        for(int i = 0; i < gray->height; i++) {
185
           char * ptr = gray->imageData + i*gray->widthStep;
186
           for(int j = 0; j < gray->width; j++) {
187
                  func((uchar)(*ptr), hist);
188
                  ptr++;
           }
189
190
        }
191
192
           /// Create image for the histogram
        IplImage * his = cvCreateImage(cvSize(st*dep,600), IPL_DEPTH_8U, 1);
193
194
        cvSet(his, 0);
                                     // Initialize image (all black)
195
       his->origin = IPL_ORIGIN_BL;
                                     // Set the origin in the bottom left corner
196
197
       //cvNamedWindow("histogram", 2);
                                             // Create window to contain the image
198
199
           /// Draw the histogram - 2 different styles
200
        for (int i = 0; i < dep; i++)</pre>
201
           if (hist[i] != 0)
               cvLine(his, cvPoint(i*st, 0), cvPoint(i*st, hist[i]/10), 150, 1, 4);
202
203
        cvSaveImage("Histogram.png", his);
204
205
        //cvShowImage("histogram", his);
206
207
        cvWaitKey(0);
208
209
       cvReleaseImage(&his);
210 }
211
212
213
214
215
216
217
218
219
220
221
224
225
226
227
228
229
230
```

```
...nalysis with Microcomputer\Project\4. Calibration\Cal1\main.cpp
```

```
231
232
233
234
     int findBackground(int hist[], int tot) {
235
         bool fmin = true, fmax = true;
236
         int m, M;
237
238
         for (int i = 0; fmin || fmax; i++) {
239
240
             if (fmin)
241
                  if(hist[i]){
242
                      m = i;
243
                      fmin = false;
244
                 }
             if (fmax)
245
246
                  if(hist[255 - i]){
247
                      M = 255 - i;
248
                      fmax = false;
249
                 }
250
         }
251
252
253
         THR res_t;
254
         res_t.B = m + (M - m) / 16;
255
256
         int frac = 45, cnt = 0;
257
258
         for(int i = m; i <= res_t.B; i++) {</pre>
259
             cnt += hist[i];
         }
260
261
262
         if( (res_t.x = ((cnt * 100) / tot)) == frac)
263
             return res_t.B;
264
         else if (res_t.x < frac) {</pre>
             while(res_t.x < frac) {</pre>
265
266
                 res t.B++;
267
                 cnt += hist[res_t.B];
268
                 res t.x = (cnt * 100) / tot;
             }
269
270
             THR t1, t2;
271
             t1.B = res_t.B;
             t1.x = res t.x;
272
273
             t2.B = res_t.B - 1;
             t2.x = ((cnt - hist[t2.B])* 100) / tot;
274
275
             if (abs(t2.x - frac) < abs(t1.x - frac) ) {</pre>
                 printf("\nPercentile: %d%%
276
                                                       Background: %d\n", t2.x, t2.B);
277
                 return t2.B;
278
             }
279
280
             else {
                 printf("\nPercentile: %d%%
281
                                                       Background: %d\n", t1.x, t1.B);
282
                 return t1.B;
283
             }
284
285
286
         else { // x > frac
287
             while(res_t.x > frac) {
288
                 res_t.B--;
```

```
...nalysis with Microcomputer\Project\4. Calibration\Cal1\main.cpp
                                                                                6
289
              cnt -= hist[res t.B];
290
              res_t.x = (cnt * 100) / tot;
291
          }
292
          THR t1, t2;
293
          t1.B = res_t.B;
          t1.x = res_t.x;
294
          t2.B = res_t.B + 1;
295
296
          t2.x = ((cnt + hist[t2.B])* 100) / tot;
297
          if (abs(t2.x - frac) < abs(t1.x - frac) ) {</pre>
298
              printf("\nPercentile: %d%%
                                            Background: %d\n", t2.x, t2.B);
299
              return t2.B;
          }
300
301
          else {
              printf("\nPercentile: %d%%
302
                                            Background: %d\n", t1.x, t1.B);
303
              return t1.B;
304
          }
       }
305
306
   }
307
308
309
310
311
312
313
314
315
316
317
   318
   319
     ///
320
321
322
323
324
325
326
327
    IplImage * simpleThresholding(IplImage * G, int th) {
328
329
       IplImage * bin;
330
331
       bin = getBinImage(G, th);
       cvNamedWindow("Binary Image", 0);
332
       cvShowImage("Binary Image", bin);
333
334
335
       return bin;
336 }
337
338
339
340
341
```

```
...nalysis with Microcomputer\Project\4. Calibration\Cal1\main.cpp
```

```
7
```

```
345
   IplImage * getBinImage(IplImage * img, unsigned th) {
346
       IplImage * I = cvCreateImage(cvGetSize(img), IPL_DEPTH_8U, 1);
       for(int i = 0; i < img->height; i++) {
347
348
           char * ptr = img->imageData + i*img->widthStep;
349
           char * p = I->imageData + i*I->widthStep;
           for(int j = 0; j < img->width; j++) {
350
351
                  if ((uchar)(*ptr) >= (uchar)th)
352
                      *p = 255;
353
                  else
354
                      *p = 0;
355
              ptr++;
356
              p++;
357
           }
358
       }
359
       return I;
360 }
361
362
363
364
365
366
367
368
    369
370 /// ///////// >
     ///
371
372
373
374
375
376
377
378
379
380
    void multi(const char * imgName) {
381
       Mat img;
382
       img = imread(imgName, CV_LOAD_IMAGE_GRAYSCALE); // Already processed externally
383
384
       vector<vector<Point> > segm;
       findContours(img, segm, CV_RETR_TREE, CV_CHAIN_APPROX_NONE, Point(0, 0));
385
386
       cout << endl << "objects: "<< segm.size();</pre>
387
388
       if(segm.size() > 1) {
389
           cout << " -> multiple objects in the same ROI!" << endl;</pre>
390
       img.release();
391
392 }
393
394
```