高级图像处理与分析课程实验报告2

学号	姓名	日期
SA18225428	许强	2019.04.02

实验名称	灰度变换
实验内容	1、计算灰度图像的归一化直方图。具体内容:利用OpenCV对图像像素进行操作,计算归一化直方图.并在窗口中以图形的方式显示出来 2、灰度图像直方图均衡处理具体内容:通过计算归一化直方图,设计算法实现直方图均衡化处理。3、彩色图像直方图均衡处理具体内容:在灰度图像直方图均衡处理的基础上实现彩色直方图均衡处理。
实验完成情况(包括完成的 实验内容及 每个实验的完成程度。注意要贴出 每个实验的核心代码)	3个模块全部完成
实验中的问题 (包括在实验 中遇到的问题,以及解决问题的方法)	rectangle绘制方形图像用于显示直方图。
实验结果(实验完成后的源码和打包文件的说明)	代码注释中含有部分说明

```
1  //
2  // Created by XQ on 2019-03-28.
3  //
4  #include<iostream>
5  #include<string>
6  #include<cmath>
7

8  #include <opencv2/imgcodecs.hpp>
9  #include <opencv2/highgui.hpp>
10  #include <opencv2/imgproc/imgproc.hpp>
```

```
11
     #include <opencv2/opencv.hpp>
12
     #include <opencv2/core/types_c.h>
13
     #include <opencv2/core/core_c.h>
     #include <opencv2/highgui/highgui.hpp>
14
15
16
     using namespace std;
17
     using namespace cv;
18
19
20
21
     bool normalizeHistogramImage(string &, int flag = 0);
22
     bool equalizeHistogramImage(string &,int flag = 0);
23
     Mat equalizeHistogram(Mat&);
24
     bool HisRGB(Mat &);
25
     bool HisGray(Mat &);
26
     bool His2D(Mat &);
27
28
29
     int main(){
30
         string str = "/Volumes/数据/图片/2k/lostwall.jpg";
31
         cout << "1.normalizeHistogramImage:" << normalizeHistogramImage(str)</pre>
     << endl;
32
         cout << "2.equalizeHistogramImage-gray:" <<</pre>
     equalizeHistogramImage(str,0) << endl;;
         cout << "3.equalizeHistogramImage-color:" <<</pre>
33
     equalizeHistogramImage(str,1) << endl;</pre>
34
35
     }
36
37
     /*计算灰度图像的归一化直方图
38
      * 具体内容:
39
      * 利用 OpenCV 对图像像素进行操作,计算归一化直方图.并在窗口中以图形的方式显示出来
40
41
      * */
42
43
     bool normalizeHistogramImage(string &src, int flag){
44
         Mat image = imread(src,flag);
         HisGray(image);
45
46
47
         image = imread(src,1);
48
         HisRGB(image);
49
         His2D(image);
50
51
         return true;
52
     }
53
     /*图像直方图均衡处理
54
```

```
55
 56
       * 具体内容:
 57
       * 通过计算归一化直方图,设计算法实现直方图均衡化处理.
         在灰度图像直方图均衡处理的基础上实现彩色直方图均衡处理。
 58
 59
      *
 60
      * */
61
     bool equalizeHistogramImage(string &src,int flag){
62
 63
         if(flag == 0){
             Mat image = imread(src,flag);
 64
 65
             imshow("input", image);
 66
 67
             Mat res = equalizeHistogram(image);
             imshow("equalizeHistogramImageGray", res);
68
 69
             waitKey(0);
 70
             destroyAllWindows();
             HisGray(res);
 71
 72
          } else if (flag == 1){
 73
             Mat image = imread(src,flag);
             imshow("input", image);
 74
75
             Mat res;
             vector<Mat> t;
 76
 77
             split(image, t);
             for (int i = 0; i < 3; i++){
78
 79
                 t[i] = equalizeHistogram(t[i]);
 80
             merge(t, res); //对RGB三通道各自均衡化后, 再组合输出结果
 81
 82
             imshow("equalizeHistogramImageColor", res);
 83
 84
             waitKey(0);
 85
             destroyAllWindows();
             HisRGB(res);
 86
 87
 88
          return true;
     }
 89
 90
 91
92
     /*函数*/
93
     Mat equalizeHistogram(Mat& input){
 94
          Mat res = input.clone();
95
          int gray[256] = {0}; //记录每个灰度级别下的像素个数
96
97
          double gray_prob[256] = {0}; //记录灰度密度
98
          double gray_count[256] = {0}; //记录累计密度
99
         int gray_equalized[256] = {0}; //均衡化后的灰度值
100
101
          int gray_sum = res.rows * res.cols; //像素总数
```

```
102
103
          for(int i = 0; i < res.rows; i++){
104
              auto * p = res.ptr<uchar>(i);
              for(int j = 0; j < res.cols; j++){
105
                  //统计每个灰度下的像素个数
106
107
                  gray[p[j]]++;
108
              }
          }
109
110
          for(int i = 0; i < 256; i++){
111
112
              //统计灰度频率
113
              gray_prob[i] = (double)gray[i]/gray_sum;
114
115
          gray_count[0] = gray_prob[0];
116
117
          for(int i = 1; i < 256; i++){
              //计算累计密度
118
119
              gray_count[i] = gray_count[i-1] + gray_prob[i];
120
          for (int i = 0; i < 256; i++){
121
              //重新计算均衡化后的灰度值,四舍五入。参考公式:(N-1)*T+0.5
122
              gray_equalized[i] = (int)(gray_count[i] * 255 + 0.5);
123
124
          }
125
126
          for(int i = 0; i < res.rows; i++){
127
              auto * p = res.ptr<uchar>(i);
128
              for(int j = 0; j < res.cols; j++){
                  //直方图均衡化,更新原图每个点的像素值
129
                  p[j] = gray_equalized[p[j]];
130
131
              }
132
          }
133
          return res;
134
135
136
      bool HisRGB(Mat &image)
137
138
          imshow("input", image);
139
          int bins = 256;
140
          int hist_size[] = { bins };
141
142
          float range[] = { 0, 256 };
          const float* ranges[] = { range };
143
144
          MatND hist_r, hist_g, hist_b;
145
          int channels_r[] = { 0 };
146
147
          calcHist(&image, 1, channels_r, Mat(), // do not use mask
                   hist_r, 1, hist_size, ranges,
148
```

```
149
                   true, // the histogram is uniform
150
                   false);
151
152
          int channels_g[] = { 1 };
153
          calcHist(&image, 1, channels_g, Mat(), // do not use mask
154
                   hist_g, 1, hist_size, ranges,
                   true, // the histogram is uniform
155
156
                   false);
157
          int channels_b[] = { 2 };
158
          calcHist(&image, 1, channels_b, Mat(), // do not use mask
159
160
                   hist_b, 1, hist_size, ranges,
161
                   true, // the histogram is uniform
162
                   false);
163
164
          double max_val_r, max_val_g, max_val_b;
          minMaxLoc(hist_r, 0, &max_val_r, 0, 0);
165
166
          minMaxLoc(hist_g, 0, &max_val_g, 0, 0);
167
          minMaxLoc(hist_b, 0, &max_val_b, 0, 0);
168
          int scale = 1;
169
170
          int hist_height = 256;
171
          Mat colorHis = Mat::zeros(hist_height, bins * 3, CV_8UC3);
          for (int i = 0; i < bins; i++)
172
173
          {
174
              float bin_val_r = hist_r.at<float>(i);
              float bin_val_g = hist_g.at<float>(i);
175
              float bin_val_b = hist_b.at<float>(i);
176
              int intensity_r = cvRound(bin_val_r*hist_height / max_val_r);
177
       //要绘制的高度
178
              int intensity_g = cvRound(bin_val_g*hist_height / max_val_g);
       //要绘制的高度
179
              int intensity_b = cvRound(bin_val_b*hist_height / max_val_b);
       //要绘制的高度
              rectangle(colorHis, Point(i*scale, hist_height - 1),
180
                        Point((i + 1)*scale - 1, hist_height - intensity_r),
181
182
                        CV_RGB(255, 0, 0));
183
              rectangle(colorHis, Point((i + bins)*scale, hist_height - 1),
184
185
                         Point((i + bins + 1)*scale - 1, hist_height -
      intensity_g),
                        CV_RGB(0, 255, 0));
186
187
              rectangle(colorHis, Point((i + bins * 2)*scale, hist_height -
188
      1),
189
                         Point((i + bins * 2 + 1)*scale - 1, hist_height -
      intensity_b),
```

```
190
                         CV_RGB(0, 0, 255));
191
          }
192
          namedWindow("HisRGB", WINDOW_AUTOSIZE); // Create a window for
193
      display.
          imshow("HisRGB", colorHis);
194
          waitKey(0);
195
196
          destroyAllWindows();
197
          return true;
198
199
      }
200
201
      bool HisGray(Mat &image){
202
203
          imshow("input", image);
204
          int bins = 256;
205
206
          int hist_size[] = { bins };
207
          float range[] = { 0, 256 };
          const float* ranges[] = { range };
208
209
          MatND hist;
210
211
          int channels[] = { 0 };
212
          calcHist(&image, 1, channels, Mat(), // do not use mask
213
                    hist, 1, hist_size, ranges,
214
                    true, // the histogram is uniform
215
                    false);
216
217
          double max_val;
218
          minMaxLoc(hist, 0, &max_val, 0, 0);
219
220
          int scale = 2;
221
          int hist_height = 256;
222
          Mat hist_img = Mat::zeros(hist_height, bins*scale, CV_8UC3);
          for (int i = 0; i < bins; i++)
223
224
          {
225
               float bin_val = hist.at<float>(i);
226
              int intensity = cvRound(bin_val*hist_height / max_val); //要绘制
      的高度
               rectangle(hist_img, Point(i*scale, hist_height - 1),
227
                         Point((i + 1)*scale - 1, hist_height - intensity),
228
229
                         CV_RGB(255, 255, 255));
230
          }
231
232
          imshow("HistogramGray", hist_img);
233
          waitKey(0);
          destroyAllWindows();
234
```

```
235
          return true;
236
      }
237
238
      bool His2D(Mat &image) {
          imshow("input", image);
239
240
          int hbins = 256, sbins = 256;
241
          int histSize[] = { hbins, sbins };
242
243
244
          float hranges[] = \{0, 256\};
245
          float sranges[] = { 0, 256 };
246
          const float* ranges[] = { hranges, sranges };
          MatND hist;
247
248
          int channels[] = { 0, 1 };
249
250
          calcHist(&image, 1, channels, Mat(), // do not use mask
251
                   hist, 2, histSize, ranges,
252
                   true, // the histogram is uniform
253
                   false);
254
          double maxVal = 0;
          minMaxLoc(hist, 0, &maxVal, 0, 0);
255
          int scale = 2;
256
257
          Mat histImg = Mat::zeros(sbins*scale, hbins*scale, CV_8UC3);
          for (int h = 0; h < hbins; h++)
258
              for (int s = 0; s < sbins; s++)
259
260
              {
261
                  float binVal = hist.at<float>(h, s);
262
                  int intensity = cvRound(binVal * 255 / maxVal);
                   rectangle(histImg, Point(h*scale, s*scale),
263
264
                             Point((h + 1)*scale - 1, (s + 1)*scale - 1),
265
                             Scalar::all(intensity),
                             FILLED);
266
267
268
          imshow("His2D", histImg);
          waitKey(0);
269
270
          destroyAllWindows();
271
          return true;
272
```