

## 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>
14  #include <opencv2/highgui/highgui.hpp>
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)
32      << endl;
33      cout << "2.equalizeHistogramImage-gray:" <<
34      equalizeHistogramImage(str,0) << endl;;
35      cout << "3.equalizeHistogramImage-color:" <<
36      equalizeHistogramImage(str,1) << endl;
37
38  }
39
40  /*计算灰度图像的归一化直方图
41  *
42  * 具体内容:
43  * 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
44  *
45  * */
46  bool normalizeHistogramImage(string &src, int flag){
47      Mat image = imread(src,flag);
48      HisGray(image);
49
50      image = imread(src,1);
51      HisRGB(image);
52      His2D(image);
53
54      return true;
55  }
56
57  /*图像直方图均衡处理

```

```

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.
58  * 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。
59  *
60  * */
61  bool equalizeHistogramImage(string &src,int flag){
62
63      if(flag == 0){
64          Mat image = imread(src,flag);
65          imshow("input", image);
66
67          Mat res = equalizeHistogram(image);
68          imshow("equalizeHistogramImageGray", res);
69          waitKey(0);
70          destroyAllWindows();
71          HisGray(res);
72      } else if (flag == 1){
73          Mat image = imread(src,flag);
74          imshow("input", image);
75          Mat res;
76          vector<Mat> t;
77          split(image, t);
78          for (int i = 0; i < 3; i++){
79              t[i] = equalizeHistogram(t[i]);
80          }
81          merge(t, res); //对RGB三通道各自均衡化后,再组合输出结果
82
83          imshow("equalizeHistogramImageColor", res);
84          waitKey(0);
85          destroyAllWindows();
86          HisRGB(res);
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
100     int gray_equalized[256] = {0}; //均衡化后的灰度值
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);
105         for(int j = 0; j < res.cols; j++){
106             //统计每个灰度下的像素个数
107             gray[p[j]]++;
108         }
109     }
110
111     for(int i = 0; i < 256; i++){
112         //统计灰度频率
113         gray_prob[i] = (double)gray[i]/gray_sum;
114     }
115
116     gray_count[0] = gray_prob[0];
117     for(int i = 1; i < 256; i++){
118         //计算累计密度
119         gray_count[i] = gray_count[i-1] + gray_prob[i];
120     }
121     for (int i = 0; i < 256; i++){
122         //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
123         gray_equalized[i] = (int)(gray_count[i] * 255 + 0.5);
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++){
129             //直方图均衡化,更新原图每个点的像素值
130             p[j] = gray_equalized[p[j]];
131         }
132     }
133     return res;
134 }
135
136 bool HisRGB(Mat &image)
137 {
138     imshow("input", image);
139     int bins = 256;
140
141     int hist_size[] = { bins };
142     float range[] = { 0, 256 };
143     const float* ranges[] = { range };
144
145     MatND hist_r, hist_g, hist_b;
146     int channels_r[] = { 0 };
147     calcHist(&image, 1, channels_r, Mat(), // do not use mask
148             hist_r, 1, hist_size, ranges,

```

```

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,
155             true, // the histogram is uniform
156             false);
157
158     int channels_b[] = { 2 };
159     calcHist(&image, 1, channels_b, Mat(), // do not use mask
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;
165     minMaxLoc(hist_r, 0, &max_val_r, 0, 0);
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);
172     for (int i = 0; i < bins; i++)
173     {
174         float bin_val_r = hist_r.at<float>(i);
175         float bin_val_g = hist_g.at<float>(i);
176         float bin_val_b = hist_b.at<float>(i);
177         int intensity_r = cvRound(bin_val_r*hist_height / max_val_r);
178         //要绘制的高度
179         int intensity_g = cvRound(bin_val_g*hist_height / max_val_g);
180         //要绘制的高度
181         int intensity_b = cvRound(bin_val_b*hist_height / max_val_b);
182         //要绘制的高度
183         rectangle(colorHis, Point(i*scale, hist_height - 1),
184                 Point((i + 1)*scale - 1, hist_height - intensity_r),
185                 CV_RGB(255, 0, 0));
186
187         rectangle(colorHis, Point((i + bins)*scale, hist_height - 1),
188                 Point((i + bins + 1)*scale - 1, hist_height -
189                 intensity_g),
190                 CV_RGB(0, 255, 0));
191
192         rectangle(colorHis, Point((i + bins * 2)*scale, hist_height -
193                 1),
194                 Point((i + bins * 2 + 1)*scale - 1, hist_height -
195                 intensity_b),

```

```

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

```

```

235     return true;
236 }
237
238 bool His2D(Mat &image){
239     imshow("input", image);
240
241     int hbins = 256, sbins = 256;
242     int histSize[] = { hbins, sbins };
243
244     float hranges[] = { 0, 256 };
245     float sranges[] = { 0, 256 };
246     const float* ranges[] = { hranges, sranges };
247     MatND hist;
248
249     int channels[] = { 0, 1 };
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;
255     minMaxLoc(hist, 0, &maxVal, 0, 0);
256     int scale = 2;
257     Mat histImg = Mat::zeros(sbins*scale, hbins*scale, CV_8UC3);
258     for (int h = 0; h < hbins; h++)
259         for (int s = 0; s < sbins; s++)
260             {
261                 float binVal = hist.at<float>(h, s);
262                 int intensity = cvRound(binVal * 255 / maxVal);
263                 rectangle(histImg, Point(h*scale, s*scale),
264                         Point((h + 1)*scale - 1, (s + 1)*scale - 1),
265                         Scalar::all(intensity),
266                         FILLED);
267             }
268     imshow("His2D", histImg);
269     waitKey(0);
270     destroyAllWindows();
271     return true;
272 }

```