高级图像处理与分析课程实验



高级图像处理与分析课程实验报告

学号 JG15225001

姓名 祁振东

日期 2016年4月1日

|  |  |
| --- | --- |
| **实验名称** | 直方图均衡 |
| **实验内容** | * 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   * 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。 * 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** | [核心代码见下面](#d)。 |
| （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） |
| **实验中的问题** | RGB的彩色图像均衡若用单个通道分别均衡再merege与转换到HSV空间再均衡效果会有不同。 |
| （包括在实验 |
| 中遇到的问 |
| 题，以及解 |
| 决问题的方 |
| 法） |
| **实验结果** | 试验中主要用到了两个函数cvEqualizeHist和cvMerge。 |
| （实验完成后 |
| 的源码和打 |
| 包文件的说 |
| 明） |

**1、灰度****图像**

|  |
| --- |
| //图像的灰度直方图均衡化  //By MoreWindows  #include <opencv2/opencv.hpp>  #include <opencv2/legacy/compat.hpp>  using namespace std;  #pragma comment(linker, "/subsystem:\"windows\" /entry:\"mainCRTStartup\"")  void FillWhite(IplImage \*pImage)  {  cvRectangle(pImage, cvPoint(0, 0), cvPoint(pImage->width, pImage->height), CV\_RGB(255, 255, 255), CV\_FILLED);  }  // 创建灰度图像的直方图  CvHistogram\* CreateGrayImageHist(IplImage \*\*ppImage)  {  int nHistSize = 256;  float fRange[] = { 0, 255 }; //灰度级的范围  float \*pfRanges[] = { fRange };  CvHistogram \*pcvHistogram = cvCreateHist(1, &nHistSize, CV\_HIST\_ARRAY, pfRanges);  cvCalcHist(ppImage, pcvHistogram);  return pcvHistogram;  }  // 根据直方图创建直方图图像  IplImage\* CreateHisogramImage(int nImageWidth, int nScale, int nImageHeight, CvHistogram \*pcvHistogram)  {  IplImage \*pHistImage = cvCreateImage(cvSize(nImageWidth \* nScale, nImageHeight), IPL\_DEPTH\_8U, 1);  FillWhite(pHistImage);  //统计直方图中的最大直方块  float fMaxHistValue = 0;  cvGetMinMaxHistValue(pcvHistogram, NULL, &fMaxHistValue, NULL, NULL);  //分别将每个直方块的值绘制到图中  int i;  for (i = 0; i < nImageWidth; i++)  {  float fHistValue = cvQueryHistValue\_1D(pcvHistogram, i); //像素为i的直方块大小  int nRealHeight = cvRound((fHistValue / fMaxHistValue) \* nImageHeight); //要绘制的高度  cvRectangle(pHistImage,  cvPoint(i \* nScale, nImageHeight - 1),  cvPoint((i + 1) \* nScale - 1, nImageHeight - nRealHeight),  cvScalar(i, 0, 0, 0),  CV\_FILLED  );  }  return pHistImage;  }  int main(int argc, char\*\* argv)  {  // 从文件中加载原图  IplImage \*pSrcImage = cvLoadImage("..\\..\\pic\\h.jpg", CV\_LOAD\_IMAGE\_UNCHANGED);  IplImage \*pGrayImage = cvCreateImage(cvGetSize(pSrcImage), IPL\_DEPTH\_8U, 1);  IplImage \*pGrayEqualizeImage = cvCreateImage(cvGetSize(pSrcImage), IPL\_DEPTH\_8U, 1);  // 灰度图  cvCvtColor(pSrcImage, pGrayImage, CV\_BGR2GRAY);  // 直方图图像数据  int nHistImageWidth = 255;  int nHistImageHeight = 150;  int nScale = 2;  // 灰度直方图及直方图图像  CvHistogram \*pcvHistogram = CreateGrayImageHist(&pGrayImage);  IplImage \*pHistImage = CreateHisogramImage(nHistImageWidth, nScale, nHistImageHeight, pcvHistogram);  // 均衡化  cvEqualizeHist(pGrayImage, pGrayEqualizeImage);  // 均衡化后的灰度直方图及直方图图像  CvHistogram \*pcvHistogramEqualize = CreateGrayImageHist(&pGrayEqualizeImage);  IplImage \*pHistEqualizeImage = CreateHisogramImage(nHistImageWidth, nScale, nHistImageHeight, pcvHistogramEqualize);  //cvNamedWindow("灰度图", CV\_WINDOW\_KEEPRATIO);  cvShowImage("灰度图", pGrayImage);  cvSaveImage("灰度图.jpg", pGrayImage);  //cvNamedWindow("灰度图-均衡化后", CV\_WINDOW\_KEEPRATIO);  cvShowImage("灰度图-均衡化后", pGrayEqualizeImage);  cvSaveImage("灰度图-均衡化后.jpg", pGrayEqualizeImage);  //cvNamedWindow("直方图", CV\_WINDOW\_KEEPRATIO);  cvShowImage("直方图", pHistImage);  cvSaveImage("直方图.jpg", pHistImage);  //cvNamedWindow("直方图-均衡化后", CV\_WINDOW\_KEEPRATIO);  cvShowImage("直方图-均衡化后", pHistEqualizeImage);  cvSaveImage("直方图-均衡化后.jpg", pHistEqualizeImage);  cvWaitKey(0);  return 0;  } |

2、彩色图象

|  |
| --- |
| #include <cv.h>  #include <highgui.h>  #include <iostream>  using namespace cv;  using namespace std;  #pragma comment(linker, "/subsystem:\"windows\" /entry:\"mainCRTStartup\"")  //彩色图像的直方图均衡化  void CalcHistRGB(IplImage \*);  IplImage\* EqualizeHistColorImage(IplImage \*);  //main  int main(int argc, char\*\* argv)  {  IplImage \*pSrcImage = cvLoadImage("..\\..\\pic\\h.jpg", CV\_LOAD\_IMAGE\_UNCHANGED);  IplImage \*pHisEquaImage = EqualizeHistColorImage(pSrcImage);  CalcHistRGB(pHisEquaImage);  cvNamedWindow("彩色原图", CV\_WINDOW\_AUTOSIZE);  cvNamedWindow("彩色均衡化后", CV\_WINDOW\_AUTOSIZE);  cvShowImage("彩色原图", pSrcImage);  cvShowImage("彩色均衡化后", pHisEquaImage);  cvWaitKey(0);  return 0;  }  void CalcHistRGB(IplImage \*pImage)  {  IplImage\* img\_source = pImage;  if (img\_source)  {  IplImage\* RedChannel = cvCreateImage(cvGetSize(img\_source), 8, 1);  IplImage\* GreenChannel = cvCreateImage(cvGetSize(img\_source), 8, 1);  IplImage\* BlueChannel = cvCreateImage(cvGetSize(img\_source), 8, 1);  // IplImage\* alphaChannel = cvCreateImage(cvGetSize(img\_source), 8, 1);  //IplImage\* gray\_plane = cvCreateImage(cvGetSize(img\_source), 8, 1);  //分割为单通道图像  cvSplit(img\_source, BlueChannel, GreenChannel, RedChannel, 0);  // 显示图像  //cvCvtColor(img\_source, gray\_plane, CV\_BGR2GRAY);  //然后为这三幅图创建对应的直方图结构。  int hist\_size = 100;  int hist\_height = 100;  float range[] = { 0, 255 };  float\* ranges[] = { range };  CvHistogram\* r\_hist = cvCreateHist(1, &hist\_size, CV\_HIST\_ARRAY, ranges, 1);  CvHistogram\* g\_hist = cvCreateHist(1, &hist\_size, CV\_HIST\_ARRAY, ranges, 1);  CvHistogram\* b\_hist = cvCreateHist(1, &hist\_size, CV\_HIST\_ARRAY, ranges, 1);  //CvHistogram\* gray\_hist = cvCreateHist(1, &hist\_size, CV\_HIST\_ARRAY, ranges, 1);  //接下来计算直方图，创建用于显示直方图的图像，略去了一部分重复代码，以下也是  cvCalcHist(&RedChannel, r\_hist, 0, 0);  cvCalcHist(&GreenChannel, g\_hist, 0, 0);  cvCalcHist(&BlueChannel, b\_hist, 0, 0);  //cvCalcHist(&gray\_plane, gray\_hist, 0, 0);  //cvNormalizeHist(gray\_hist, 1.0);  cvNormalizeHist(r\_hist, 1.0);  cvNormalizeHist(g\_hist, 1.0);  cvNormalizeHist(b\_hist, 1.0);  int scale = 2;  IplImage\* hist\_image = cvCreateImage(cvSize(hist\_size\*scale, hist\_height \* 3), 8, 3);  cvZero(hist\_image);  //然后开始显示，这里对直方图进行了标准化处理，不然的话无法观察到明显的变化。  float r\_max\_value = 0;  float g\_max\_value = 0;  float b\_max\_value = 0;  //float gray\_max\_value = 0;  cvGetMinMaxHistValue(r\_hist, 0, &r\_max\_value, 0, 0);  cvGetMinMaxHistValue(g\_hist, 0, &g\_max\_value, 0, 0);  cvGetMinMaxHistValue(b\_hist, 0, &b\_max\_value, 0, 0);  //cvGetMinMaxHistValue(b\_hist, 0, &gray\_max\_value, 0, 0);  for (int i = 0; i<hist\_size; i++)  {  float r\_bin\_val = cvQueryHistValue\_1D(r\_hist, i);  int r\_intensity = cvRound(r\_bin\_val\*hist\_height / r\_max\_value);  cvRectangle(  hist\_image,  cvPoint(i\*scale, hist\_height - 1),  cvPoint((i + 1)\*scale - 1, hist\_height - r\_intensity),  CV\_RGB(255, 0, 0));  float g\_bin\_val = cvQueryHistValue\_1D(g\_hist, i);  int g\_intensity = cvRound(g\_bin\_val\*hist\_height / g\_max\_value);  cvRectangle(  hist\_image,  cvPoint(i\*scale, 2 \* hist\_height - 1),  cvPoint((i + 1)\*scale - 1, 2 \* hist\_height - g\_intensity),  CV\_RGB(0, 255, 0));  float b\_bin\_val = cvQueryHistValue\_1D(b\_hist, i);  int b\_intensity = cvRound(b\_bin\_val\*hist\_height / b\_max\_value);  cvRectangle(  hist\_image,  cvPoint(i\*scale, 3 \* hist\_height - 1),  cvPoint((i + 1)\*scale - 1, 3 \* hist\_height - b\_intensity),  CV\_RGB(0, 0, 255));  }  cvNamedWindow("直方图", WINDOW\_NORMAL);  cvShowImage("直方图", hist\_image);  }  }  IplImage\* EqualizeHistColorImage(IplImage \*pImage)  {  IplImage \*pEquaImage = cvCreateImage(cvGetSize(pImage), pImage->depth, 3);  // 原图像分成各通道后再均衡化,最后合并即彩色图像的直方图均衡化  const int MAX\_CHANNEL = 4;  IplImage \*pImageChannel[MAX\_CHANNEL] = { NULL };  int i;  for (i = 0; i < pImage->nChannels; i++)  pImageChannel[i] = cvCreateImage(cvGetSize(pImage), pImage->depth, 1);  cvSplit(pImage, pImageChannel[0], pImageChannel[1], pImageChannel[2], pImageChannel[3]);  for (i = 0; i < pImage->nChannels; i++)  cvEqualizeHist(pImageChannel[i], pImageChannel[i]);  cvMerge(pImageChannel[0], pImageChannel[1], pImageChannel[2], pImageChannel[3], pEquaImage);  for (i = 0; i < pImage->nChannels; i++)  cvReleaseImage(&pImageChannel[i]);  return pEquaImage;  } |