

高级图像处理与分析课程实验报告

学号：SA23225226 姓名：郭浩天 日期：2023.10.23

实验名称	直方图均衡
实验内容	<p>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。</p> <p>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。</p> <p>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。</p>
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)	<pre>/*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 * 灰度级出现次数/图像的像素总数，得到各个灰度级出现的频率 */ void normalizeHistogramImage(string& src) { //灰度图方式读入 Mat image = imread(src, 0); //显示灰度直方图 showHisGray(image); //彩色图方式读入 //image = imread(src, 1); //显示RGB直方图 //showHisRGB(image); }</pre>

```

/*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.
* 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。
* 统计直方图每个灰度级出现次数，累计归一化的直方图，计算新的像素值
* */
void equalizeHistogramGrayImage(string& src) {
    //灰度图方式读入
    Mat image = imread(src, 0);
    //imshow("输入图片", image);
    //res为进行直方图均衡后的图像
    Mat res = equalizeHistogram(image);
    imshow("灰度图直方图均衡", res);
    waitKey(0);
    destroyAllWindows();
    //显示灰度直方图
    showHisGray(res);
}

/*彩色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.
* 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。
*
* */
void equalizeHistogramRGBImage(string& src) {
    //彩色图方式读入
    Mat image = imread(src, 1);
    //imshow("输入图片", image);
    //定义输出mat图res
    Mat res;
    //mat向量t，用来装rgb图的三个通道
    vector<Mat> t;
    //将image的三个通道分割到t向量中
    split(image, t);
    //单独对BGR三个通道进行直方图均衡
    for (int i = 0; i < 3; i++)
        t[i] = equalizeHistogram(t[i]);
    //将均衡后的t向量合并到res彩色图中
    merge(t, res);
    imshow("彩色图直方图均衡", res);
    waitKey(0);
    destroyAllWindows();
    //显示RGB直方图
    showHisRGB(res);
}

```

```

/*直方图均衡函数*/
Mat equalizeHistogram(Mat& input) {
    //定义均衡后的图res
    Mat res = input.clone();

    //记录每个灰度级别下的像素个数
    int gray[256] = { 0 };
    //记录灰度密度
    double gray_prob[256] = { 0 };
    //记录累计密度
    double gray_count[256] = { 0 };
    //均衡化后的灰度值
    int gray_equalized[256] = { 0 };
    //像素总数
    int gray_sum = res.rows * res.cols;

    //统计每个灰度下的像素个数
    for (int i = 0; i < res.rows; i++) {
        auto* p = res.ptr<uchar>(i);
        for (int j = 0; j < res.cols; j++) {
            gray[p[j]]++;
        }
    }
    //统计灰度频率
    for (int i = 0; i < 256; i++) {
        gray_prob[i] = (double)gray[i] / gray_sum;
    }

    //计算累计密度
    gray_count[0] = gray_prob[0];
    for (int i = 1; i < 256; i++) {
        gray_count[i] = gray_count[i - 1] + gray_prob[i];
    }

    //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
    for (int i = 0; i < 256; i++) {
        gray_equalized[i] = (int)(gray_count[i] * 255 + 0.5);
    }

    //直方图均衡化,更新原图每个点的像素值
    for (int i = 0; i < res.rows; i++) {
        auto* p = res.ptr<uchar>(i);
        for (int j = 0; j < res.cols; j++) {
            p[j] = gray_equalized[p[j]];
        }
    }
    return res;
}

```

```

/*
 * 彩色RGB直方图显示函数
 */
void showHisRGB(Mat& image)
{
    //imshow("输入图片", image);
    //bin代表区间256
    int bins = 256;

    int hist_size[] = { bins };
    float range[] = { 0, 256 };
    const float* ranges[] = { range };

    //有三个通道对应三个直方图数组
    MatND hist_r, hist_g, hist_b;
    int channels_r[] = { 0 };
    //计算r通道
    calcHist(&image, 1, channels_r, Mat(), // do not use mask
            hist_r, 1, hist_size, ranges,
            true, // the histogram is uniform
            false);

    int channels_g[] = { 1 };
    //计算g通道
    calcHist(&image, 1, channels_g, Mat(), // do not use mask
            hist_g, 1, hist_size, ranges,
            true, // the histogram is uniform
            false);

    int channels_b[] = { 2 };
    //计算b通道
    calcHist(&image, 1, channels_b, Mat(), // do not use mask
            hist_b, 1, hist_size, ranges,
            true, // the histogram is uniform
            false);

    double max_val_r, max_val_g, max_val_b;
    //minMaxLoc寻找矩阵(一维数组当作向量,用Mat定义) 中最小值和最大值的位置.
    minMaxLoc(hist_r, 0, &max_val_r, 0, 0);
    minMaxLoc(hist_g, 0, &max_val_g, 0, 0);
    minMaxLoc(hist_b, 0, &max_val_b, 0, 0);
    int scale = 1;

    int hist_height = 256;
    Mat colorHis = Mat::zeros(hist_height, bins * 3, CV_8UC3);
    for (int i = 0; i < bins; i++)
    {
        float bin_val_r = hist_r.at<float>(i);
        float bin_val_g = hist_g.at<float>(i);
        float bin_val_b = hist_b.at<float>(i);
        int intensity_r = cvRound(bin_val_r * hist_height /
max_val_r); //要绘制的高度
        int intensity_g = cvRound(bin_val_g * hist_height /
max_val_g); //要绘制的高度
    }
}

```

```

        int intensity_b = cvRound(bin_val_b * hist_height /
max_val_b); //要绘制的高度
        //绘制r通道
        rectangle(colorHis, Point(i * scale, hist_height - 1),
            Point((i + 1) * scale - 1, hist_height - intensity_r),
            CV_RGB(255, 0, 0));
        //绘制g通道
        rectangle(colorHis, Point((i + bins) * scale, hist_height -
1),
            Point((i + bins + 1) * scale - 1, hist_height -
intensity_g),
            CV_RGB(0, 255, 0));
        //绘制b通道
        rectangle(colorHis, Point((i + bins * 2) * scale, hist_height
- 1),
            Point((i + bins * 2 + 1) * scale - 1, hist_height -
intensity_b),
            CV_RGB(0, 0, 255));

    }
    namedWindow("彩色RGB直方图", WINDOW_AUTOSIZE); // Create a window for
display.
    imshow("彩色RGB直方图", colorHis);
    waitKey(0);
    destroyAllWindows();
}

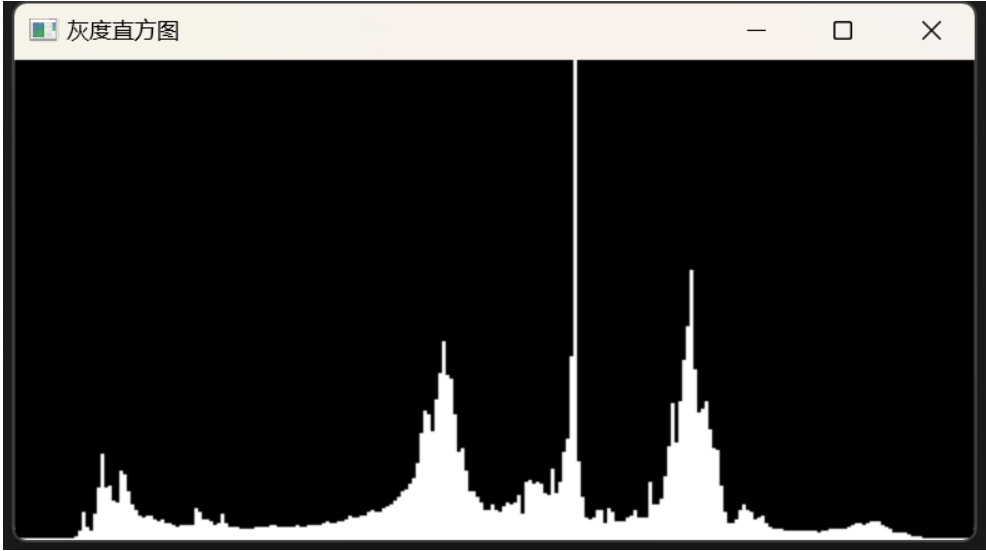
/*
 * 灰度直方图显示函数
 */
void showHisGray(Mat& image) {

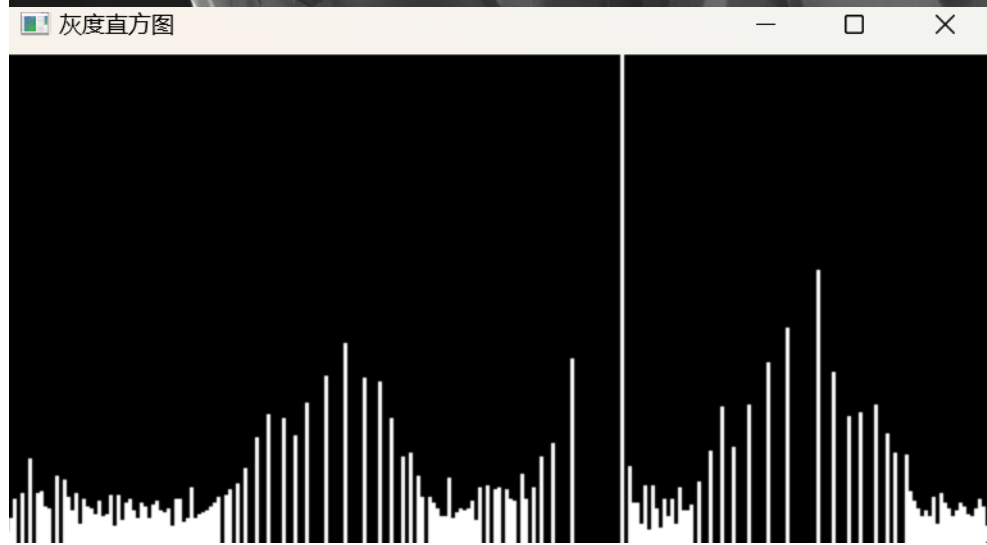
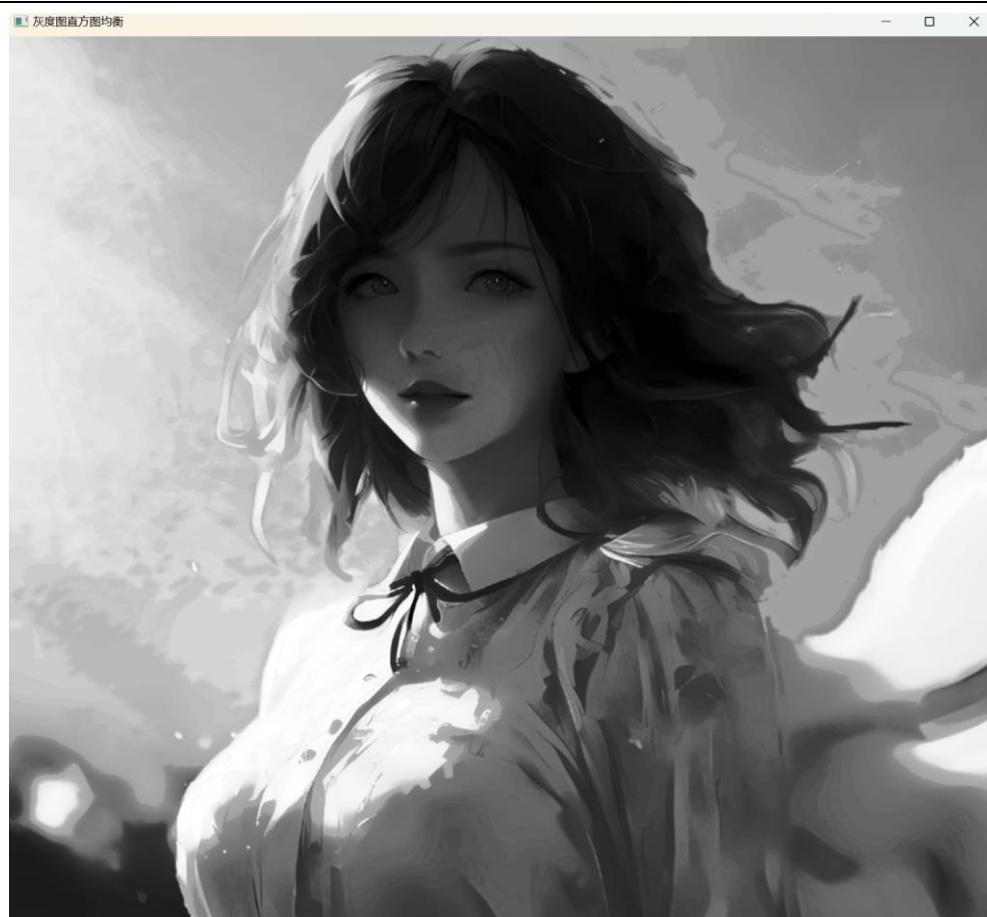
    //imshow("输入图片", image);
    //bin代表区间256
    int bins = 256;

    //hist的尺寸就是区间数
    int hist_size[] = { bins };
    float range[] = { 0, 256 };
    const float* ranges[] = { range };

    //多维mat型变量hist，用来输出的直方图数组
    MatND hist;
    //只用一个灰度通道需要读取
    int channels[] = { 0 };
    calcHist(&image, 1, channels, Mat(), // do not use mask
        hist, //输出的直方图数组
        1, //需要统计的直方图通道个数
        hist_size, //直方图分成多少个区间，bin的个数
        ranges, //统计像素值的区间
        true, // 进行归一化处理
        false); //多个图像不统计像素值的个数

```

	<pre>double max_val; //minMaxLoc寻找矩阵(一维数组当作向量,用Mat定义)中最小值和最大值的位置. minMaxLoc(hist, 0, &max_val, 0, 0); //一个bin2格 int scale = 2; int hist_height = 256; //直方图mat型变量hist_img Mat hist_img = Mat::zeros(hist_height, bins * scale, CV_8UC3); for (int i = 0; i < bins; i++) { float bin_val = hist.at<float>(i); int intensity = cvRound(bin_val * hist_height / max_val); // 要绘制的高度 rectangle(hist_img, Point(i * scale, hist_height - 1), //矩形框的左上角 Point((i + 1) * scale - 1, hist_height - intensity), // 矩形框的右下角 CV_RGB(255, 255, 255)); //填充颜色纯白 } imshow("灰度直方图", hist_img); waitKey(0); destroyAllWindows(); }</pre>
<p>实验中的问题</p> <p>(包括在实验中遇到的问题, 以及解决问题的方法)</p>	<p>1、计算灰度图像归一化直方图, 灰度级出现次数/图像的像素总数, 得到各个灰度级出现的频率, 然后实现直方图显示函数。</p> <p>2、灰度图像均衡化处理就是统计直方图每个灰度级出现次数, 累计归一化的直方图, 计算新的像素值。</p> <p>3、在灰度图像直方图均衡处理的基础上在R、G、B三个通道上实现彩色直方图均衡处理。</p>
<p>实验结果</p> <p>(实验完成后的源码和打包文件的说明)</p>	



彩色直方图均衡



彩色RGB直方图

