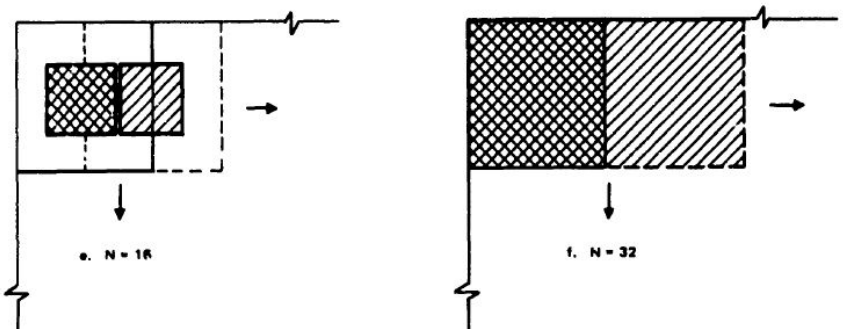
知乎 首发于 程序员的伪文艺




# 直方图均衡化

李新春
既可提刀立码，行遍天下；又可调参炼丹，卧于隆中。



直方图均衡化(Histogram Equalization)是一种增强图像对比度(Image Contrast)的方法，其主要思想是将一副图像的直方图分布变成近似均匀分布，从而增强图像的对比度。直方图均衡化虽然只是数字图像处理(Digital Image Processing)里面的基本方法，但是其作用很强大，是一种很经典的算法。

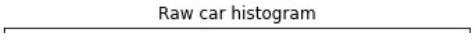
下面，本文会介绍一些直方图均衡化方面的知识和方法，包括以下几个部分：

1. 直方图均衡化与对比度增强
2. 直方图均衡化(HE)原理和实现
3. 自适应直方图均衡化(AHE)原理和实现
4. 限制对比度自适应直方图均衡化(CLAHE)原理和实现
5. 自适应局部区域伸展(Local Region Stretch)直方图均衡化原理和实现
6. 总结和参考文献

## 1、直方图均衡化与对比度增强

下面给出图像对比度增强的一个例子，Figure 1 是一张汽车图片(图片来自：CS6640 - Project 2)，图片是一张338 * 600的灰度图。可以看出汽车与背景都是雾蒙蒙的看不清楚，整张图片偏暗，并且汽车与背景(地面、房屋)区别不是很明显。将其直方图绘制出来之后得到Figure 2，可以看出其灰度绝大多数分布在100~180之间，而直方图均衡化要做的就是让直方图尽可能地均匀分布在0~255内。


Figure 1: car.jpg



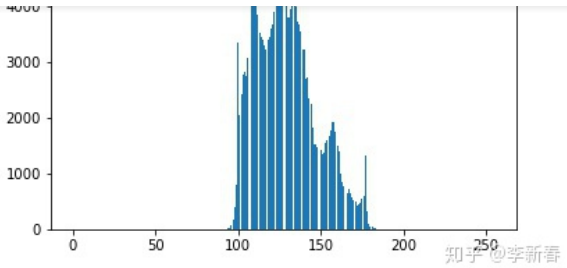



Figure 2 : car.jpg的直方图分布

下面使用Python库PIL中的ImageOps来完成直方图均衡化的过程，来看看结果图片如何。代码很简单，使用下面一行即可：

```
eq_img = ImageOps.equalize(img)
```

得到灰度直方图均衡化之后的图像及其直方图为Figure 3 & 4，可以看出汽车”锃亮“了许多，车身变得很清晰，背景房屋的纹理也显现了出来，总之图像质量不再是灰蒙蒙的了。同时，观察其直方图也可以看出，直方图的分布在0~255近似均匀了，稍后会解释为什么是近似均匀。


Figure 3 : ImageOps直方图均衡化后car.jpg

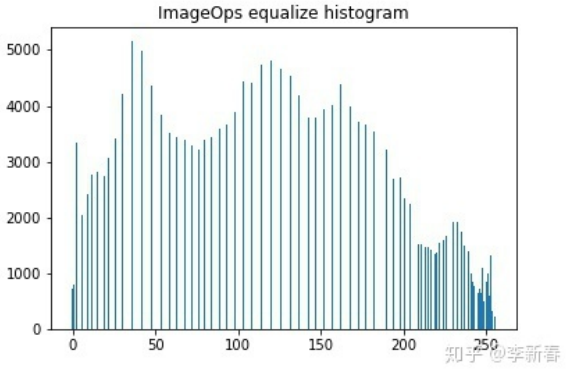



Figure 4 : 使用ImageOps对car.jpg直方图均衡化后直方图分布

## 2、直方图均衡化(HE)原理和实现

这里介绍直方图均衡化(Histogram Equalization)的基本原理。假设我们现在有一个图像A，其直方图分布 $H_A(D)$ ，我们想利用一个单调非线性映射 $f: R \rightarrow R$ ，将图像A变为图像B，即对图

是单调非线性变换函数 $f$，左上方式得到的图像B的直方图分布，其中有 $D_B = f(D_A)$，$D_B + \Delta D_B = f(D_A + \Delta D_A)$。即可以理解 $f$ 的作用是将A图像里面像素点灰度为 $D_A$ 的全部变为 $D_B$，那么则有：$\int_{D_A}^{D_A+\Delta D_A} H_A(D)dD = \int_{D_B}^{D_B+\Delta D_B} H_B(D)dD$

上面公式可以理解为对应区间内像素点总数不变。为了实现直方图均衡化，特殊地有：

$$\int_0^{D_A} H_A(D)dD = \int_0^{D_B} H_B(D)dD$$

因为目标是直方图均匀分布，那么理想的 $H_B(D) = \frac{A_0}{L}$，$A_0$ 是 像素点个数，$L$ 是灰度级深度，通常取256。那么得到：

$$\int_0^{D_A} H_A(D)dD = \frac{A_0 D_B}{L} = \frac{A_0 f(D_A)}{L}$$

那么，$f$ 就可以求出来了，结果为：

$$f(D_A) = \frac{L}{A_0}\int_0^{D_A} H_A(D)dD$$

离散形式为：

$$f(D_A) = \frac{L}{A_0}\sum_{u=0}^{D_A} H_A(u)$$

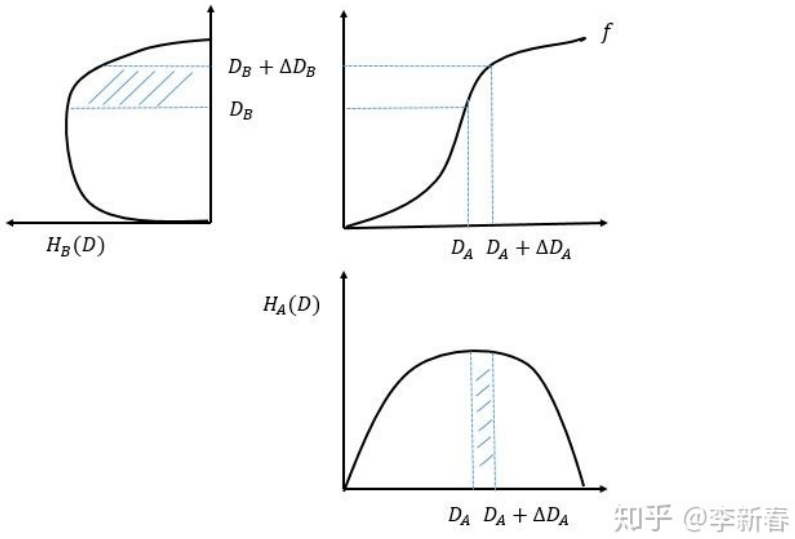



Figure 5 : 直方图均衡化示意图

通过上面推导过程可以看出，映射函数 $f$ 和CDF密切相关。并且，推导过程是在连续分布上作的处理，而实际上图像灰度级别通常是256，故是一种以离散情况近似了连续分布，如果灰度级别足够高的话产生的结果会是真正均匀分布的直方图，由于灰度级只有256，故而实际情况中得到的直方图往往不是均匀分布，而是近似均匀分布的。同时，假若A图像的灰度直方图变化剧烈，且某些灰度区间不存在像素点，这会造成CDF的剧烈变化，那么在灰度区间(x, x+1)内即可能发生剧烈的变化，从而导致最后产生的B图像的直方图也有较大的不均匀性。但是，实际使用过程中，得到的图像能近似均匀分布已经能达到比较好的对比度增强效果了。

直方图均衡化用代码实现也很简单，用Python的话下面几行就够了：

```
# calculate histogram
hists = histogram(img)
```

```
hists_cumsum = np.cumsum(hists)
const_a = level / (m * n)
hists_cdf = (const_a * hists_cumsum).astype("uint8")

# mapping
img_eq = hists_cdf[img]
```

### 3、自适应直方图均衡化(AHE)原理和实现

一般来说，在数字图像处理领域中，会经常见到“自适应”这个词，即根据图像的局部性质进行处理。那么自适应直方图均衡化(Adaptive Hisogram Equalization)是什么意思呢?

在前面介绍的直方图均衡化中，是直接对全局图像进行均衡化，是Global Histogram Equalization，而没有考虑到局部图像区域(Local Region)，自适应过程就是在均衡化的过程中只利用局部区域窗口内的直方图分布来构建映射函数 $f$ 。

最早的关于AHE的论文来自*Image enhancement techniques for cockpit displays*，一篇来自1974年的论文，很古老的文章了。这里面介绍了AHE的方法，其实原理很简单。

首先，最简单并且直接的想法，如Figure 6 展示(图来自论文截图)，对A图像每个像素点进行遍历，用像素点周围W * W的窗口进行计算直方图变换的CDF，然后对该像素点进行映射。这是一个很Naive的方法，计算复杂度高的离谱。下面简单分析一下，首先对窗口大小为W * W的图像计算CDF，复杂度为 $O(W*W+L)$ ，那么作一遍遍历下来，复杂度为 $O(M*N*(W*W+L))$ ，这个复杂度太高了。

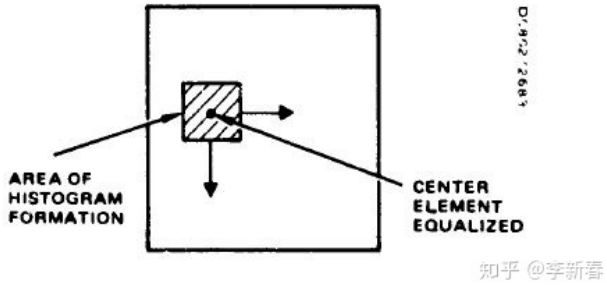
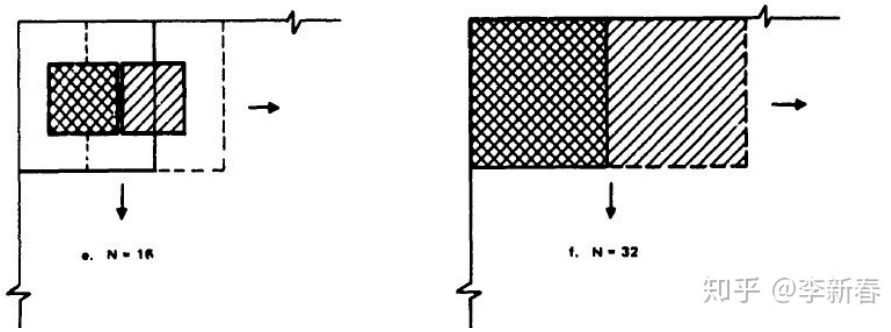



Figure 6 : Naive AHE

然后，作者 给出了一些改进的方法，即利用W * W的窗口计算直方图的CDF，然后不仅仅是对图像的一个像素点进行变换，而是对一系列(比如: 4, 9, 16...)的像素点进行变换，假设是利用64 * 64的窗口计算直方图CDF，然后对该窗口内中心区域的32 * 32个像素点进行变换，这样速度就是Naive AHE里面的32 * 32倍，下个窗口相对于本次窗口水平或者垂直移动32个像素点，示意图如Figure 7，图中右边是利用W * W的窗口对W * W个像素点进行变换，即相当于对图像分块，分别做直方图均衡化操作。



在实际实现过程中，根据窗口大小(W, W)和影响区域大小(A, A)，结合图像大小(M, N)，每次将窗口移动步长A即可。注意处理边界情况，比如当窗口位于图像左上角时，影响区域不是(A, A)，而是整个窗口，同理处理右上、右下、左下和四周的情况即可。

下面给出利用32 * 32的窗口大小计算直方图CDF，影响区域为16 * 16，即对窗口内16 * 16大小的区域进行变换，窗口每次移动步长为16。得到的结果为Figure 8 & 9，可以看出最后的直方图虽然也是近似均匀分布的，但是图像却产生了块状不连续的现象，图像中会出现以16 * 16为基本单位的块状区域，这就使得效果不是很好。

另外，图中还会出现一种块内“马赛克”现象，这是因为当W * W窗口内像素点近似一样时，即直方图只有一个灰度级，那么这样得到的CDF就会是一种“阶跃”的曲线，使变换后图像过度增强，一些噪声就可能被过度放大。


Figure 8 : AHE得到的car.jpg

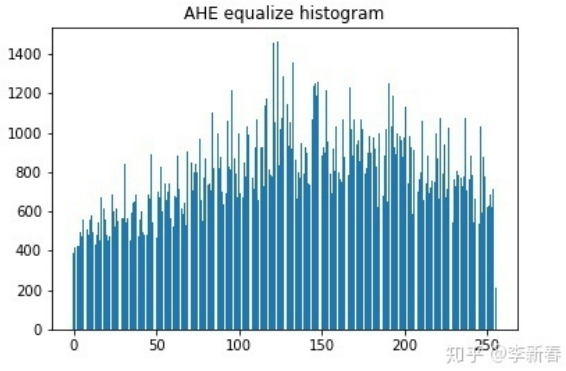



Figure 9 : AHE对car.jpg处理后直方图

## 4、限制对比度自适应直方图均衡化(CLAHE)原理和实现

为了避免由于AHE产生的图像不连续和过度增强的结果，引入一种限制直方图分布的办法，即限制对比度自适应直方图均衡化(Contrast Limited Adaptive Hitogram Equalization)。CLAHE来自文章*Adaptive Histogram Equalization and Its Variations*，是1987年 一篇论文，相对于AHE，提出了两个改进的地方。

第一，提出一种限制直方图分布的方法。考虑图像A的直方图，设定一个阈值，假定直方图某个灰度级超过了阈值，就对 之进行裁剪，然后将超出阈值的部分平均分配到各个灰度级，这个过程可以用Figure 10来进行解释。图中左上图是原来的直方图分布，可以看出有两处峰值，其对应的CDF为左下图，可以看出有两段梯度比较大，变化剧烈。对于之前频率超过了阈值的灰度级，那么就把这些超过阈值的部分裁剪掉平均分配到各个灰度级上，如右上图，那么这会使得CDF变得较为

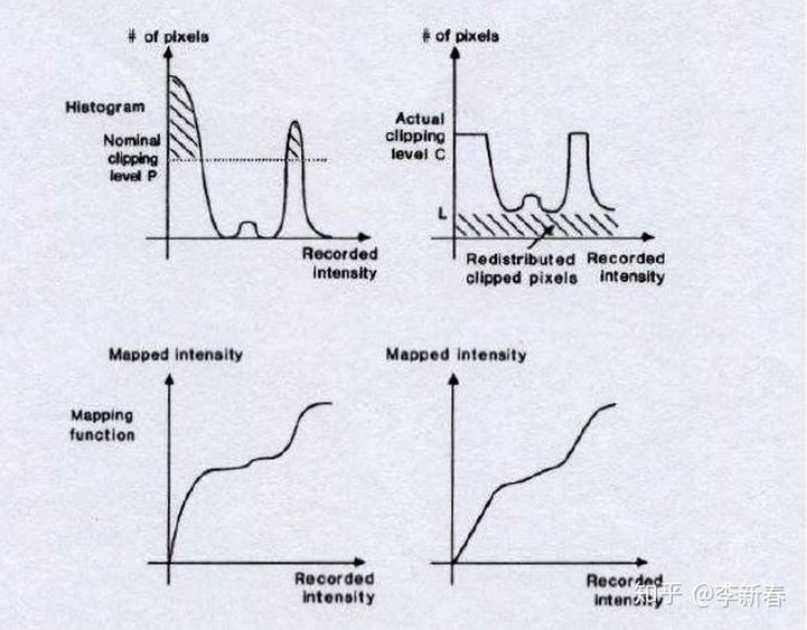



Figure 10 : Clip Histogram

第二，提出了一种插值的方法，加速直方图均衡化。主要思想用Figure 11(图片来自：Adaptive histogram equalization)解释。首先，将图像分块，每块计算一个直方图CDF，这在Figure 11中的表示是黑色实线边框的小块，这里简称为窗口。其次，对于图像的每一个像素点，找到其邻近的四个窗口(边界先不讨论)，就如图中蓝色像素点，分别计算四个窗口直方图CDF对蓝色像素点的映射值，记作 $f_{ul}(D), f_{ur}(D), f_{dl}(D), f_{dr}(D)$ ，然后进行双线性插值得到最终该像素点的映射值，双线性插值(BiLinear)公式为：

$$f(D) = (1-\Delta y)((1-\Delta x)f_{ul}(D) + \Delta x f_{bl}(D)) + \Delta y((1-\Delta x)f_{ur}(D) + \Delta x f_{br}(D))$$

$\Delta x, \Delta y$ 是蓝色像素点相对于左上角窗口中心，即左上角黑色像素点的距离与窗口大小的比值。

上面没有考虑边界情况，对于图中红色像素点，只使用其最近的窗口的CDF进行映射；对于图中绿色像素点，采用邻近的两个窗口的CDF映射值进行线性插值。

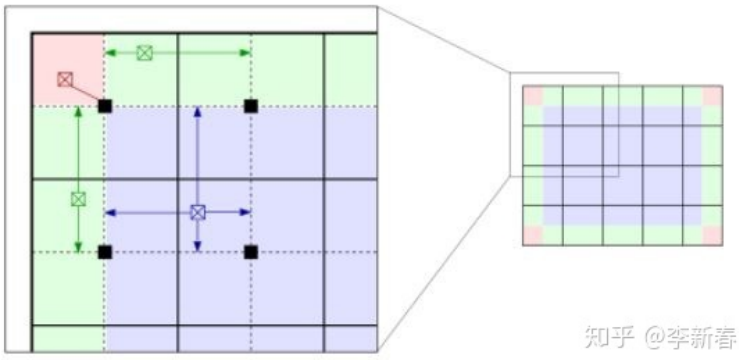



Figure 11 : CLAHE插值

下面给出使用CLAHE得到的car.jpg的增强效果，及其直方图分布，见Figure 12 & 13。对比Figure 3，图中右上角的对比度明显增强，这是局部直方图均衡化的作用；对比 Figure 8，避免了块状不连续的缺陷。

CLAHE涉及到窗口大小、影响区域大小、直方图阈值三个部分的参数，可以通过调参更好地去控制对比度增强的结果。


Figure 12 : CLAHE 对比度增强的car

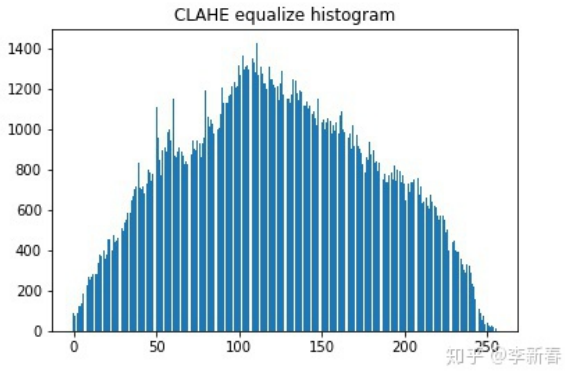



Figure 13 : CLAHE car 直方图

## 5、自适应局部区域伸展(Local Region Stretch)直方图均衡化原理和实现

上面提到的AHE和CLAHE都是基于块状区域进行直方图均衡化的，但是能不能根据灰度级 区域 近似的区域进行均衡化呢？比如对图像中灰度级[min, max]范围里面的所有像素点进行均衡化，使得像素点的直方图尽量在[min, max]上均匀分布。在论文*Adaptive Contrast Enhancement Using Local Region Stretching*中，根据亮度(Brightness)对图像进行分割成几个区域，然后分别做直方图均衡化。

下面根据论文的主要思想，引入下面的方法：统计图像直方图，按照灰度级划分为三个灰度区间，使得三个区间的像素点数量近似相等，这样就分别在[0, level1), [level1, level2), [level2, 255]三个灰度区间做直方图均衡化，最后合并。

以car.jpg为例，level1 = 115, level2 = 140，在三个区间上做直方图均衡化的结果为Figure 14。

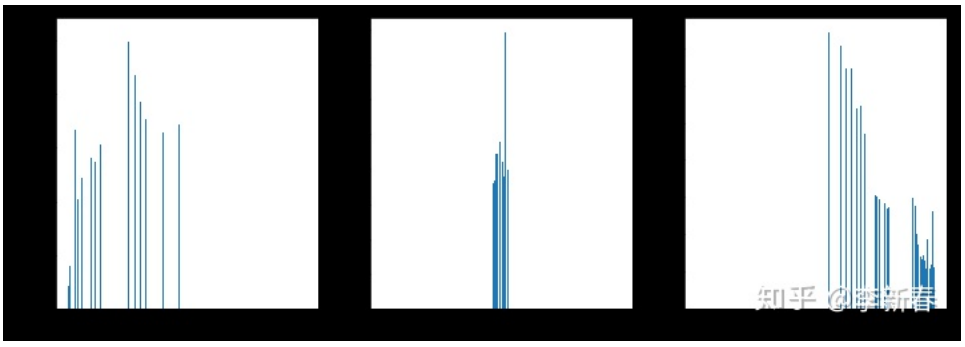



Figure 14 : 灰度区间分别直方图均衡化

最后得到的增强后的car.jpg为Figure 15：


Figure 15 : 分灰度区间直方图均衡化后结果

## 6、总结和参考文献

本文主要是对几种直方图均衡化的方法做了介绍，一些自适应的直方图均衡化方法在实际中应用很有效。根据实际情况进行选择。

代码在我的Github: lxcnju/histogram_equalization

## 参考文献：

1. Adaptive histogram equalization

2. Histogram equalization

3. CS6640 - Project 2

4. Image enhancement techniques for cockpit displays

5. Adaptive Histogram Equalization and Its Variations

6. Adaptive Contrast Enhancement Using Local Region Stretching

编辑于 2018-11-19

图像处理 图像 算法

**文章被以下专栏收录**

**程序员的伪文艺**
机器学习与数据挖掘

**图像处理**

**你需要的投稿信件的模板**
你需要的投稿信件的模板

知乎 首发于 程序员的伪文艺

推荐阅读

直方图均衡化

赵澍

说

这
相
重
讲
学
是

幻

**27 条评论** 切换为时间排序

写下你的评论...

薛丰铧 2019-11-08

CLAHE里的双线性插值的公式写错了，△x和△y应该相互调换位置。

而且AHE里，step size 不一定非要是1/2 window size， 小一点均衡效果会更好，比如说对于64*64的window， step size 可以选为4,2或者1。

不过楼主其他部分写得还是很有用的，我数组图像处理的project就是完全照着你的方法写的！

4

知乎用户 2020-02-09

HB(D)应该是1/L吧，因为你这里算的是连续直方图，还没有离散化，所以与像素点数无关，因为从[0, L]的灰度（连续）的概率一样且积分为1，所以概率密度应该是1/L。

2

李思奇 2019-06-18

这么好的文章为啥没人赞

2

GoGoing 2019-05-15

简直不能太赞了，感谢分享

2

SP LEE 08-11

level / (m * n)是什么?

1

alasuo 05-05

请问一下局部直方图均衡化和自适应直方图均衡化有什么区别呀?

1

ml13Z 04-23

不错写的挺好

1

观云 2020-12-19

终于在这里找到了AHE的马赛克程序，不过输入灰度图，出来的是RGB的，请教下您应该改哪

▲ 赞同 334 ▼

Omtrix 2020-11-12
直方图均衡化的python代码的变量能注释一下吗?
1

知乎用户 04-10
理科生的美，严谨通透
赞

奇异 03-24
想问一下我复现GitHub模型的时候用main.py测试出来的图片是蓝色的呀
赞

大家好 03-17
如果能先讲一下H的意义就更好啦
赞

虚御安玄 2020-11-26
感谢分享!
赞

0704 2020-10-20
有一个问题：在解释clahe的双线性插值时，周围4个点的值是通过周围四个窗口的cdf对所求点通过映射所得. 那有没有一种情况，所求点原本的灰度值，在周围四个cdf中都找不到，那还咋求映射值?
赞

ljh 2020-10-09
楼主，你的CLAHE python代码中没有考虑边界条件吧，r和c都是从0开始，不可能出现负数的
赞

知乎用户 2020-09-22
感谢
赞

0704 2020-07-12
在对原区间(DA, DA +△DA)和目标区别（DB,DB + △DB）积分时，你说这可以理解对应区间的像素总和是相等的. 对此，我有点疑问，x轴代表灰度级，y轴像素个数，积分出来的是面积（这个面积应该是没有实际意义，如果是速度和时间，面积代表是路程）. 原区间和目标区间的像素数确实相等，但这跟他们积分出来的面积又有什么关系？ 就是这一点有些疑惑!
赞

球神不如球己 2020-04-16
大神，你好，感谢你的分享。但有个小小的问题，想请教一下。我先构造了一个图片数组来测试，总共300个像素，其中100个灰度为50,100个灰度为51,100个灰度为52。按照文中提供的程序思路得到的结果是50->85 ,51->170, 52->255。但是我用opencv自带的cv2.equalizeHist()进行直方图均衡化处理，得到的结果是50->0, 51->128, 52->255。因为我也没找到这个自带函数更多的文档，请教下为什么会有这样的差异呢。
赞

未央 2020-02-19
感谢!！！
赞

▲ 赞同 334 ▼

知乎 首发于 程序员的伪文艺