传统暗光增强算法总结

> 实际上是一些图像增强的算法，并不完全针对于暗光增强的应用情景。
> 参考资料：
>
> - 知乎讲解
> - CSDN文章
> - CSDN文章2
> - 博客园文章
> - 高斯滤波

直方图均衡化

如果一个图像的灰度值分布比较集中，那么它体现在视觉上就是对比度较差，较难分辨。直方图均衡化试图通过一个变换重新分配图像的灰度值，使得图像的各个像素点灰度值尽量铺满`0-255`的范围，从而提高视觉质量。

> 灰度值越高，图像点越暗；灰度值越低，图像点越亮。

它的基本**工作原理/步骤**可概括如下：

1. 计算原图直方图：

首先，计算输入图像的灰度直方图，记录每个灰度级（例如 0 到 255）的像素数量。这个直方图展示了图像中不同灰度级的分布情况。

2. 计算直方图的CDF（累计概率密度函数）

CDF的公式是：

$$CDF(k) = \sum_{i=0}^{k} p(i)$$

其中，$p(i)$ 是灰度级 i的概率密度，表示灰度级 i 在图像中的出现频率。

3. CDF归一化

为了确保新的图像灰度值范围与原图相同，通常需要对 CDF 进行归一化，使得 CDF 的值在 [0, 255] 范围内（对于 8 位图像）。归一化的公式为：

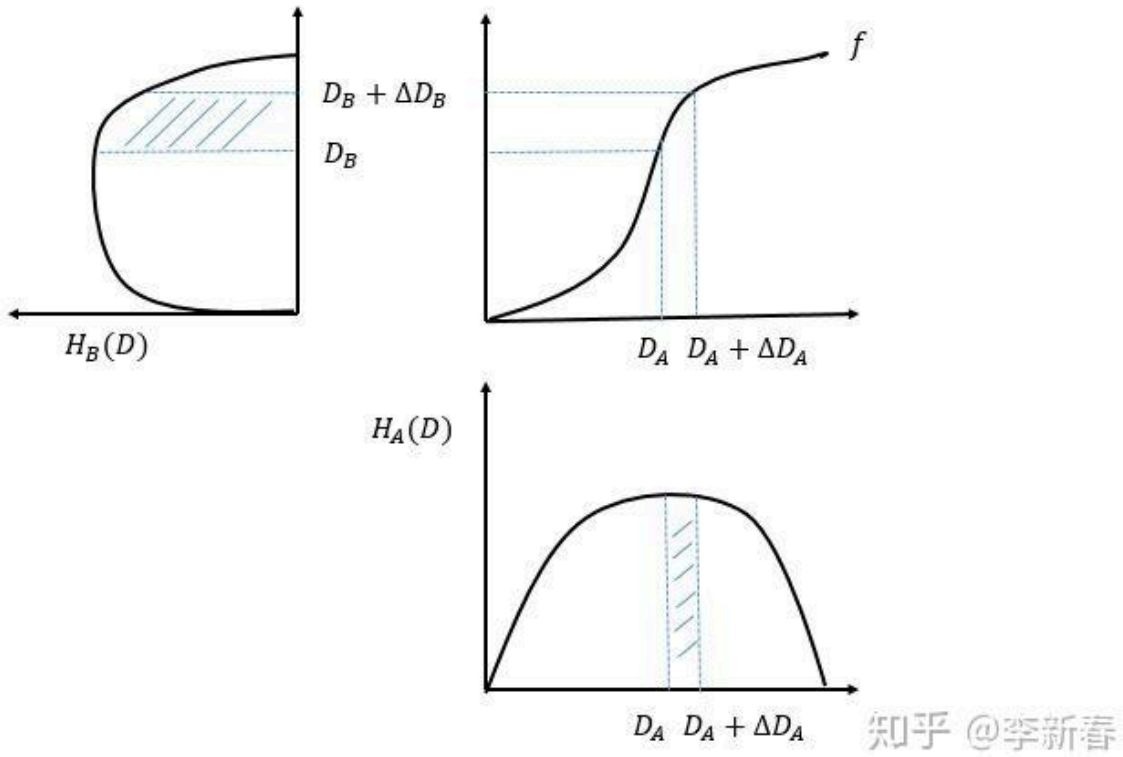
$$CDF_{\text{normalized}}(k) = \frac{CDF(k) - CDF_{\min}}{N - 1} \times (L - 1)$$

其中，$CDF_{\min}$ 是 CDF 中的最小值，N 是图像中总的像素数，L 是灰度级数（对于 8 位图像，L=256）。

4. 映射

根据归一化后的 CDF，将每个原始图像中的灰度值映射到新的灰度值。每个像素的灰度值会根据其对应的 CDF 值重新分配。

> - **均匀分布目标**：理想情况下，直方图均衡化的目的是让每个灰度级的像素数量大致相等，这样可以避免某些灰度级被过多使用，而其他灰度级则被稀疏使用。通过这种方式，图像的对比度和细节可以得到最大化。
> - **图像均匀化**：在灰度级数量固定（例如 256）时，理想的均匀分布帮助消除图像中灰度的集中的问题，尤其是在图像对比度低的情况下。均匀分布的目标可以显著提升图像的视觉效果，增加图像的细节感。



下图为经典`HE`的推导过程

按照Figure 5的图示来说明：图中右下方是A图像的灰度直方图分布(便于画图，这里画作连续分布)，图中右上方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：

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

其中$H_A(D)$可以理解为一种概率分布，D是灰度值，每一个灰度值对应具有该灰度值的点的个数。理想的灰度值分布是尽量均匀的，所以得到了图中说的$H_B(D)$。

5. 代码实现

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

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

# mapping
img_eq = hists_cdf[img]
```

6. 变体/改进

自适应直方图均衡化（AHE）

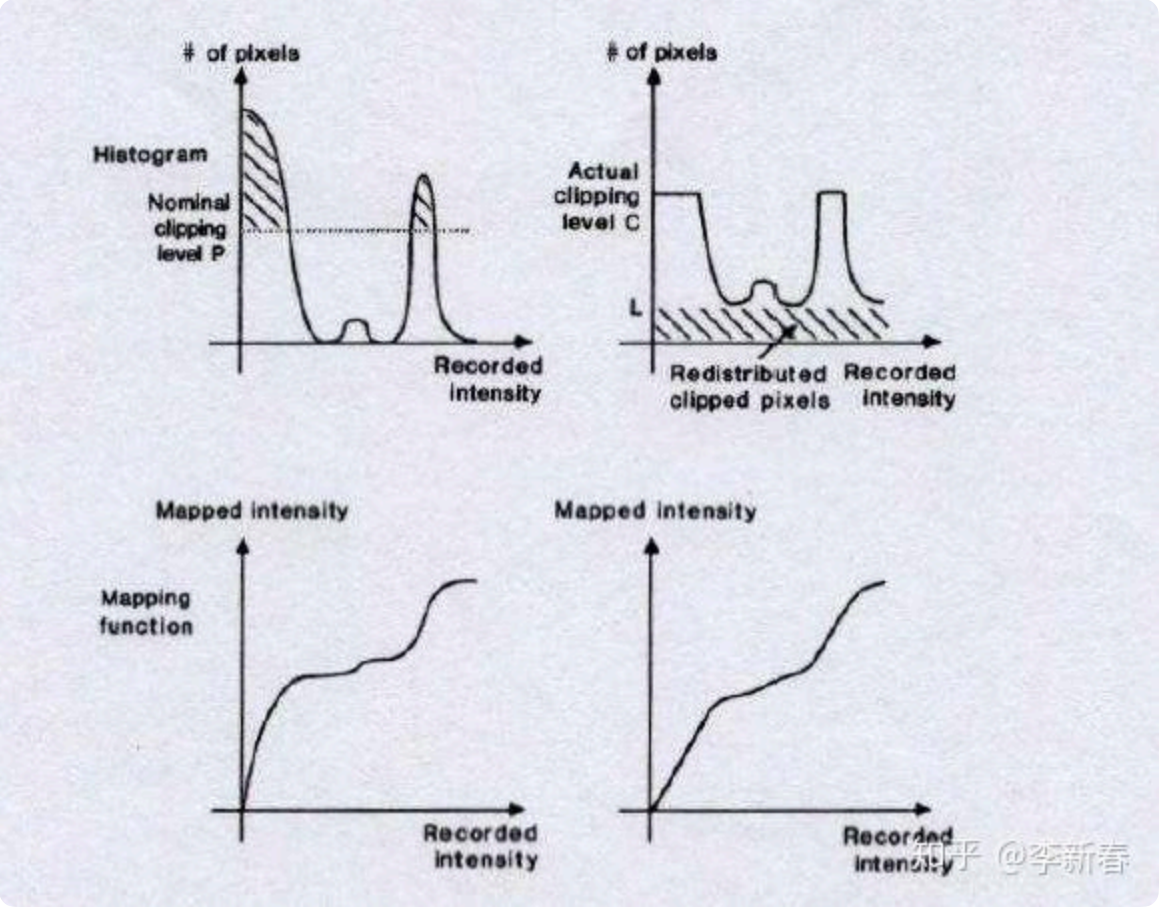
在前面介绍的直方图均衡化中，是直接对全局图像进行均衡化，是Global Histogram Equalization，而没有考虑到局部图像区域(Local Region)，自适应过程就是在均衡化的过程中只利用局部区域窗口内的直方图分布来构建映射函数 f。也就是使用滑动窗口。
**优点：

- 能够考虑图像的局部性，不丢弃局部细微特征；
  缺点：
- 传统滑动方式计算复杂度太大；
- 图像出现块状不连续现象，以及会出现CDF曲线“阶跃”的情况，导致图像被过度增强，噪声放大。

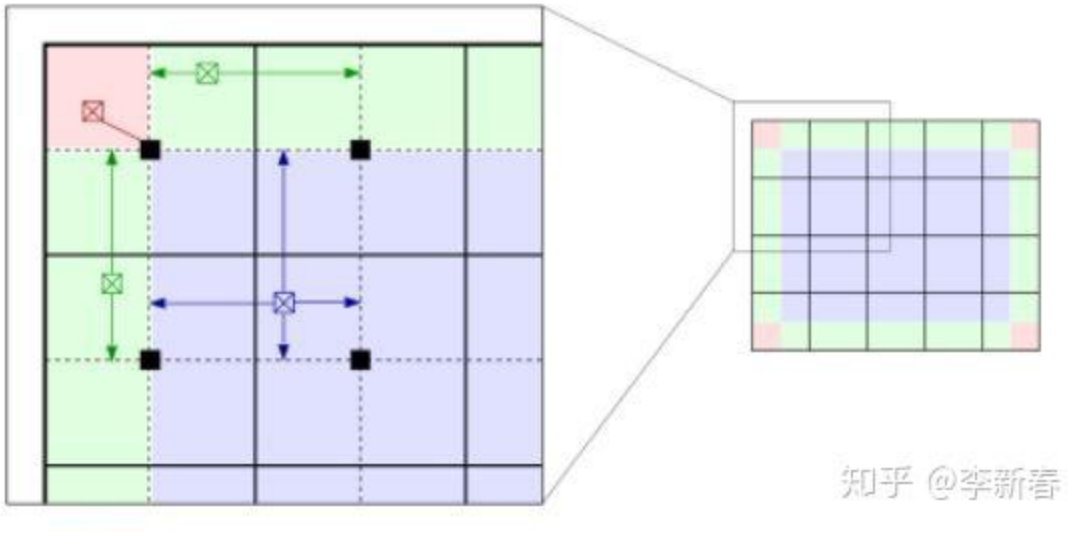
限制对比度自适应直方图均衡化（CLAHE）

有两种改进方法：

- *Clip Histogram*



  也就是设置一个阈值，所有灰度值大于这一阈值的都裁剪掉，最后平均加到所有点的灰度值上。
- *CLAHE 插值*
  任何一个像素点都可以用四个最邻近的窗口的CDF值加权（**二维线性插值**）来获得。



自适应局部区域伸展(Local Region Stretch)直方图均衡化

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。

> 示例代码请参照本仓库：lxcnju/histogram_equalization；
> 从仓库中的示例图片来看，可能用普通的HE的处理效果就不错。CLAHE可能会抑制原本灰度值较高的点，从而导致原本的亮区被抑制。

Gamma 校正

伽马校正（Gamma Correction）是一种图像处理技术，用于调整图像的亮度。它解决了显示系统（如显示器或相机）中输入与输出亮度之间的非线性关系。

基本概念：

图像的像素强度 I 与其校正后的强度 I' 之间的关系可以通过伽马函数表示：

$$I' = I^{\gamma}$$

其中，γ（伽马值）是一个常数，决定了校正的程度。通常来说：

- 如果 γ = 1，则不进行校正（即输入和输出相同）。
- 如果γ > 1，图像变暗（更偏向于阴影部分）。
- 如果γ < 1，图像变亮（更偏向于高光部分）。

伽马校正的目的：

1. **显示系统**：大多数显示器和其他显示设备对于输入信号的亮度响应不是线性的。伽马校正通过补偿这种非线性行为，使输出图像看起来更自然。
2. **人眼视觉**：人眼对暗区的差异更为敏感，而对亮区的差异较不敏感。伽马校正有助于调整图像的亮度，以更好地符合人眼的视觉感知，使暗区的细节更清晰可见。

实际应用：

伽马校正广泛应用于图像增强（包括低光图像处理）、摄影和视频编码等领域，目的是使图像看起来更自然，或者纠正图像捕捉或显示过程中引入的亮度问题。

Retinex图像增强算法

> 参考材料

Retinex理论

Retinex模型的理论基础是**三色理论**和**颜色恒常性**。

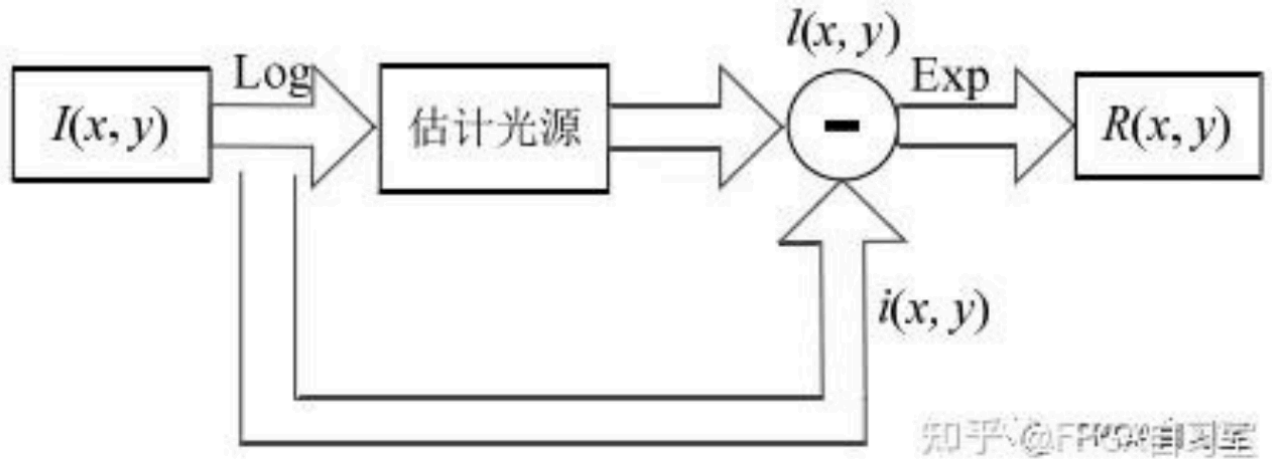
- 颜色恒常性：物体的颜色是由物体对长波（红色）、中波（绿色）、短波（蓝色）光线的反射能力来决定的，而不是由反射光强度的绝对值来决定的，物体的色彩不受光照非均匀性的影响，具有一致性。
  Retinex 理论认为图像$I(x,y)$是由**照度图像**与**反射图像**组成。前者指的是物体的入射分量的信息，用 $L(x,y)$ 表示；后者指的是物体的反射部分，用 $R(x,y)$ 表示。公式为

  $$I(x,y) = R(x,y) \cdot L(x,y)$$

  同时，由于对数形式与人类在感受亮度的过程属性最相近（灰度值越高的区域，人类对于灰度值的感知越不敏锐），因此将上述过程转换到对数域进行处理，这样做也将复杂的乘法转换为加法：

  $$i(x,y) = r(x,y) + l(x,y)$$

而Retinex算法的步骤可以概括如下：**高斯滤波**、**对数与反对数Map**以及**图像拉伸算法**。



几种Retinex算法

不同的Retinex算法的基本步骤都是一样的：通过对原始图像进行 *高斯滤波* 来获取照度图像，并尽量准确的获取照度图像，最后将照度图像从原始图像中分离出来，从而获得反射图像。

SSR算法

SSR (Singal Scale Retinex)，即单尺度视网膜算法，是最基础的一个Retinex算法。具体步骤总结如下：

- 输入原始图像 $I(x,y)$ 和滤波的半径范围 σ;
- 计算原始图像$I(x,y)$ 高斯滤波后的结果，得到 $L(x,y)$;

> 这里的意思是，可以用原始图像高斯滤波后的结果看成最后的光照结果。也就是
>
> $$L_i(x,y) = I_i(x,y) * G(x,y,\sigma)$$
>
> 注意，这里的运算是**卷积运算**。关于高斯滤波的更详细介绍可以参考这篇文章。

> 补充，对于这个滤波函数（又称**中心环绕函数**），公式也可以如下表示：
>
> $$F(x,y)=\lambda e^{\frac{-(x^2+y^2)}{c^2}}$$
>
> c 表示为高斯环绕尺度，λ 是一个尺度，它的取值必须满足以下条件：
>
> $$\iint F(x,y)dxdy=1$$

- 按照公式计算，得到 $log[R(x,y)]$；将过程整合在一起就是如下过程：

$$r_{SSR_i}(x,y,\sigma)=\log(I_i(x,y))-\log(I_i(x,y)*G(x,y,\sigma))$$

- 其中i表示第i个通道。

- 将得到的结果量化为 $[0,255]$ 范围的像素值，然后输出结果图像。量化公式如下：

$$R(x,y)=(Value-Min)/(Max-Min)*255$$

> - 需要注意的是，这一量化公式的最大/最小以及原值，指的都是还未变换回去的值，也就是**对数变换后直接用量化公式得出最终的灰度值，而不是再用指数变换还原**！
> - 还有MSR算法，可以参考上面说到的CSDN文章了解一下，并结合其他资料学习，此处略过。

MSRCR 算法（多尺度-带色彩恢复Retinex算法，Multi-Scale Retinex with Color Restoration）

公式如下：

$$R_{MSRC_i}(x,y,\sigma)=\beta\log\left(\alpha\frac{I_t(x,y)}{\sum_{l=1}^{3}I_l(x,y)}\right)R_{MSR_l}(x,y,\sigma)$$

分解来看，可以看以下的解读：

在前面的增强过程中，图像可能会因为增加了噪声，而使得图像的局部细节色彩失真，不能显现出物体的真正颜色，整体视觉效果变差。
针对这一点不足，MSRCR在MSR的基础上，加入了色彩恢复因子C来调节由于图像局部区域对比度增强而导致颜色失真的缺陷。
改进算法如下所示：

$$R_{MSRCR_i}(x,y) = C_i(x,y)R_{MSR_i}(x,y) \text{ .....(8)}$$
$$C_i(x,y) = f[I_i^{'}(x,y)] = f[\frac{I_i(x,y)}{\sum_{j=1}^{N} I_j(x,y)}] \text{ .....(9)}$$
$$f[I_i^{'}(x,y)] = \beta \log[\alpha I_i^{'}(x,y)] = \beta\{\log[\alpha I_i^{'}(x,y)] - \log[\sum_{j=1}^{N} I_j(x,y)]\} \text{ .....(10)}$$

其中参数说明如下：

- Ii(x, y)表示第i个通道的图像
- Ci表示第i个通道的彩色回复因子，用来调节3个通道颜色的比例；
- f(·)表示颜色空间的映射函数；
- β是增益常数；
- α是受控制的非线性强度；

MSRCR算法利用彩色恢复因子C，调节原始图像中3个颜色通道之间的比例关系，从而把相对较暗区域的信息凸显出来，达到了消除图像色彩失真的缺陷。
处理后的图像局部对比度提高，亮度与真实场景相似，在人们视觉感知下，图像显得更加逼真。
但是MSRCR算法处理图像后，像素值一般会出现**负值**。所以从对数域$r(x,y)$转换为实数域$R(x,y)$后，需要通过改变**增益**$Gain$，**偏差**$Offset$ 对图像进行修正。使用公式可以表示为：

$$R_{MSRCR_i}(x,y)' = G \cdot R_{MSRCR_i}(x,y) + O$$

> 另外也有McCann算法，时间有限未能整理，可自行点击上方链接学习。

图像拉伸算法：

Retinex 算法压缩了原始图像的动态范围，所以进行完 Retinex 算法处理之后，需要对处理之后的图像进行**灰度拉伸**，也就是扩展图像灰度级动态范围的处理。目前广泛采用的图像拉伸方法主要包括**线性拉伸**、**直方图均衡化**（HE）及**限制对比度自适应直方图均衡化**（CLAHE）。

> 注意到，线性拉伸的类似思想其实在SSR算法的最后一步**量化**已经体现，其公式也正是把输入变量变成$I(x,y)$带入公式，所以在这种情况下不需要额外去做其他处理。
> 另外两种前面也有介绍了。