

实现功能

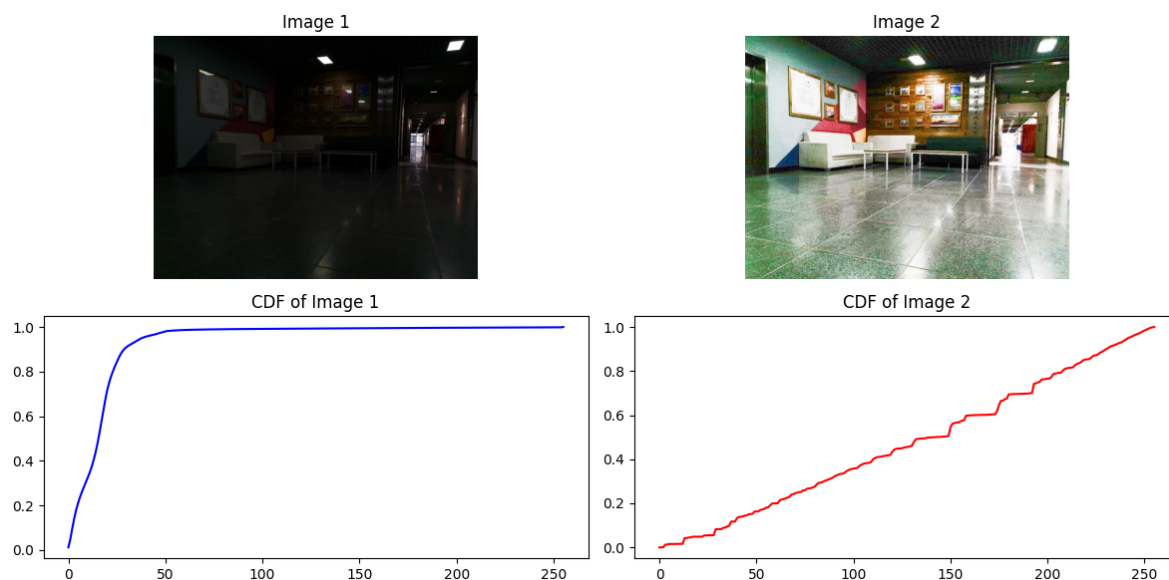
- 手动实现直方图均衡化
- 手动实现CLAHE

直方图均衡化

代码

```
def histogram_equalization(img):  
    # 将图像从 BGR 转换为 HLS 色彩空间  
    hls_img = cv2.cvtColor(img, cv2.COLOR_BGR2HLS)  
  
    # 分离 H、L、S 三个通道  
    h_channel, l_channel, s_channel = cv2.split(hls_img)  
  
    # 保存原始 L 通道数据用于绘制 CDF  
    original_l = l_channel.copy()  
  
    # 对 L 通道进行直方图均衡化  
    h, w = l_channel.shape  
  
    # 手动计算原始图像的直方图和 CDF  
    hist_original = np.zeros(256)  
    for i in range(h):  
        for j in range(w):  
            hist_original[l_channel[i, j]] += 1  
    cdf_original = hist_original.cumsum()  
    cdf_original_normalized = cdf_original / cdf_original.max() # 归一化到 [0,1]  
  
    # 计算均衡化映射表  
    hist = hist_original  
    cdf = cdf_original  
    cdf_normalized = cdf * 255 / cdf[-1]  
    equalization_map = cdf_normalized.astype('uint8')  
  
    # 应用映射表进行像素值映射  
    equalized_l = np.zeros_like(l_channel)  
    for i in range(h):  
        for j in range(w):  
            equalized_l[i, j] = equalization_map[l_channel[i, j]]  
  
    # 合并 H、均衡化后的 L 和 S 通道  
    equalized_hls = cv2.merge([h_channel, equalized_l, s_channel])  
  
    # 将图像从 HLS 转回 BGR 色彩空间  
    equalized_img = cv2.cvtColor(equalized_hls, cv2.COLOR_HLS2BGR)  
  
    return equalized_img
```

效果



左为原图，右为直方图均衡化提亮后的图片。

CLAHE

代码

CLAHE算法的主要步骤为：

- 图像分块
- 按照预定义的阈值裁剪直方图
- 计算变换函数
- 插值来过渡块与块之间的边界

```
def clahe_algorithm(img, clip_limit=2.0, tile_grid_size=(8, 8)):  
    # 将图像从 BGR 转换为 LAB 色彩空间  
    lab_img = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)  
  
    # 分离 L、A、B 三个通道  
    l_channel, a_channel, b_channel = cv2.split(lab_img)  
  
    # 保存原始 L 通道用于绘制 CDF  
    original_l = l_channel.copy()  
  
    # 获取图像尺寸  
    h, w = l_channel.shape  
  
    # 计算每个 tile 的大小  
    tile_h = h // tile_grid_size[1]  
    tile_w = w // tile_grid_size[0]  
  
    # 初始化空的数组用于存储均衡化的结果  
    equalized_l = np.zeros_like(l_channel, dtype=np.float32)  
  
    # 对每个 tile 进行处理  
    for i in range(tile_grid_size[1]):
```

```

for j in range(tile_grid_size[0]):
    # 计算 tile 的位置
    y_start = i * tile_h
    y_end = (i + 1) * tile_h if i != tile_grid_size[1] - 1 else h
    x_start = j * tile_w
    x_end = (j + 1) * tile_w if j != tile_grid_size[0] - 1 else w

    # 提取 tile
    tile = l_channel[y_start:y_end, x_start:x_end]

    # 计算直方图
    hist = np.bincount(tile.flatten(), minlength=256).astype(np.float32)

    # 计算剪辑阈值
    clip_limit_value = clip_limit * np.mean(hist)

    # 对直方图进行剪辑
    excess = hist - clip_limit_value
    excess[excess < 0] = 0
    hist = hist.clip(max=clip_limit_value)
    # 重新分配被剪辑的像素
    redistribute = excess.sum() / 256
    hist += redistribute

    # 计算剪辑后的 CDF
    cdf = hist.cumsum()
    cdf = (cdf - cdf.min()) * 255 / (cdf.max() - cdf.min())
    cdf = cdf.astype('uint8')

    # 映射像素值
    equalized_tile = cdf[tile]

    # 将处理后的 tile 放回对应位置
    equalized_l[y_start:y_end, x_start:x_end] = equalized_tile

# 进行双线性插值, 平滑 tiles 之间的过渡
# 创建网格坐标
grid_x = np.linspace(0, w, tile_grid_size[0]+1)
grid_y = np.linspace(0, h, tile_grid_size[1]+1)
grid_x = grid_x.astype(int)
grid_y = grid_y.astype(int)

# 初始化空的数组用于存储插值结果
final_l = np.zeros_like(l_channel, dtype=np.uint8)

for i in range(tile_grid_size[1]):
    for j in range(tile_grid_size[0]):
        # 当前 tile 的四个角点
        x1, x2 = grid_x[j], grid_x[j+1]
        y1, y2 = grid_y[i], grid_y[i+1]

        # 获取当前 tile 及相邻 tiles 的中心值用于插值
        tile = equalized_l[y1:y2, x1:x2]

        # 插值
        final_l[y1:y2, x1:x2] = tile

```

```

# 将 L 通道转换为 uint8 类型
final_l = final_l.astype('uint8')

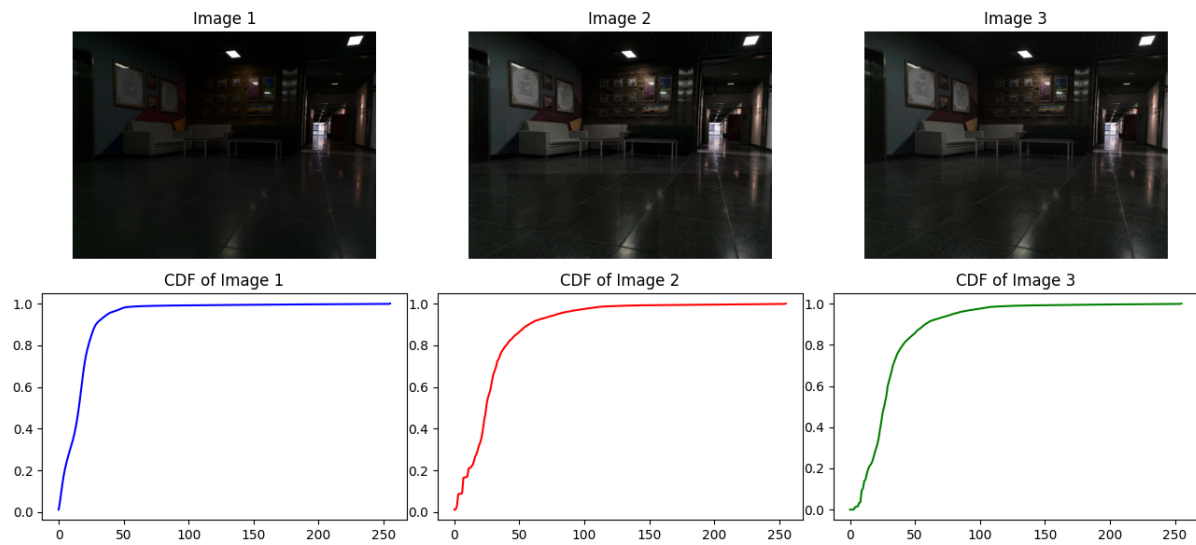
# 合并均衡化后的 L 通道和原始的 A、B 通道
equalized_lab = cv2.merge([final_l, a_channel, b_channel])

# 将图像从 LAB 转回 BGR 色彩空间
equalized_img = cv2.cvtColor(equalized_lab, cv2.COLOR_LAB2BGR)

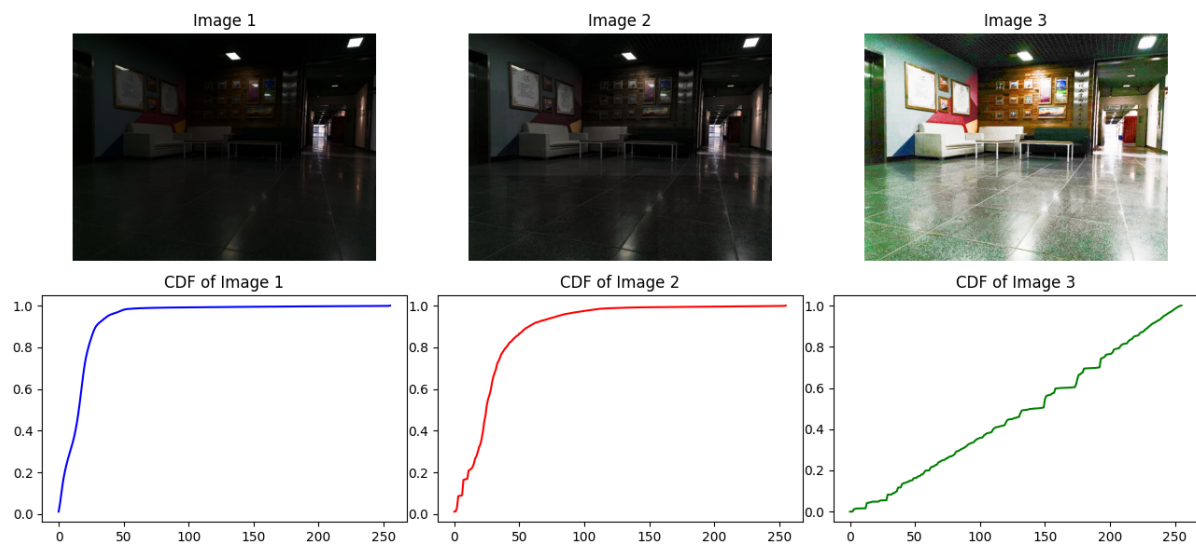
return equalized_img

```

结果



左中右分别为原图、手工实现的CLAHE和Opencv官方的CLAHE。



这是与HE的对比。可以看出在暗部细节以及噪点控制上更加优秀。

UI界面

图像处理网页演示工具

使用方式，在浏览器中打开<http://127.0.0.1:8088/>即可


此网页演示提供以下图像处理工具:

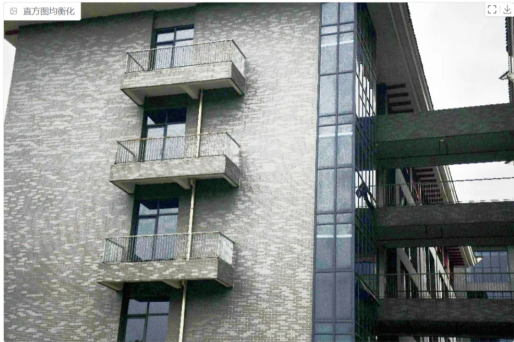
- 作业1: 色相/饱和度/亮度调整工具
- 作业2: 图像缩放工具
- 作业3: DCGAN图像生成工具
- 作业4: XXX工具
- 作业5: XXX工具

作业1: 色相/饱和度/亮度调整工具 作业2: 图像缩放工具 作业3: DCGAN图像生成工具 作业4: 图像去噪工具 作业5: 图像提亮工具


作业五: 图像提亮工具

📁 输入图像





📁 CLAHE



运行

通过 API 使用 🍷 · 使用 Gradio 构建 🍷

左边输入待提亮图像，右边会使用两种算法对其进行提亮。