实验一

深智能硕 52　2025213972　马逸宁

1.运行环境

语言：python

依赖：torch、numpy、PIL、os 等

2.实验步骤

1) 计算输入带雾图像的暗通道

$$J^{dark}(\mathbf{x}) = \min_{c\in\{r,g,b\}} (\min_{\mathbf{y}\in\Omega(\mathbf{x})} (J^c(\mathbf{y}))),$$

根据公式，我们先用最小值滤波对图像进行处理，再求得三通道最小值。通过对比发现交换顺序得到的结果一样。

```
def dark_channel(image, radius=15):
    min_rgb, _ = torch.min(image, dim=1, keepdim=True)  # shape: (B, 1, H, W)
    kernel_size = 2 * radius + 1
    padding = radius
    dark_channel = -F.max_pool2d(-min_rgb,
                                 kernel_size=kernel_size,
                                 stride=1,
                                 padding=padding)
#     dark_channel = -F.max_pool2d(-image,
#                                  kernel_size=kernel_size,
#                                  stride=1,
#                                  padding=padding)
#     dark_channel, _ = torch.min(dark_channel, dim=1, keepdim=True)

    return dark_channel
```

2) 估计大气亮度

记录暗通道图中灰度最大的前 0.1%的像素所在的位置，把带雾图像在这些位置中最大的灰度值当做大气亮度。

```
##2. 估计大气亮度
def estimate_atmospheric_light(dark_channel, haze_img):
    H, W = dark_channel.shape
    if haze_img.shape != (3, H, W):
        raise ValueError(f"haze_img形状应为 (3, {H}, {W})，但得到 {haze_img.shape}")

    threshold = np.percentile(dark_channel, 99.9)
    mask = dark_channel >= threshold
    # 获取每个通道的掩码区域，在三个通道的所有前0.1%像素中找到最大值
    top_pixels_rgb = [haze_img[i, mask] for i in range(3)]
    atmospheric_light = [np.max(top_pixels_rgb[i]) for i in range(3)]

    return atmospheric_light
```

3) 计算传导图像

根据暗通道和估计的大气亮度计算传导图，考虑到透视情况，将 ω 设置为 0.95。

$$\tilde{t}(\mathbf{x}) = 1 - \omega \min_{c}(\min_{\mathbf{y}\in\Omega(\mathbf{x})} (\frac{I^c(\mathbf{y})}{A^c})).$$

```python
##3、计算传导图像
def compute_transmission_simple(dark_channel, atmospheric_light, omega=0.95):
    if not np.isscalar(atmospheric_light):
        A = np.max(atmospheric_light)
    else:
        A = atmospheric_light
    # 计算传导图像
    transmission = 1 - omega * (dark_channel / A)
    transmission = np.clip(transmission, 0.1, 1.0)

    return transmission
```

4) 估计原始场景图

根据计算的传导图像恢复原始的场景的图像。

$$\mathrm{I(x)} = \mathrm{J(x)t(x)} + \mathrm{A}\big(1 - \mathrm{t(x)}\big),$$

$$\mathrm{J(x)} = \frac{\big[\mathrm{I(x)} - \mathrm{A}\big(1 - \mathrm{t(x)}\big)\big]}{\mathrm{t(x)}} = \frac{\mathrm{I(x)} - \mathrm{A}}{\max\left(\mathrm{t(x)}, t_0\right)} + \mathrm{A},$$

这里，I(x)为观测到的带雾图像，J(x)为原始场景图，t(x)为传导图像，A 为大气亮度。但当透射率 t(x)接近于零时，$J(x)t(x)$可能非常接近于零，此时直接恢复的场景容易产生噪声。因此将透射率 t(x)限制在一个下界为 t_0的范围。

```python
##4. 恢复图像
def recover_scene_image(haze_img, transmission, atmospheric_light, t0=0.1):
    C, H, W = haze_img.shape

    if np.isscalar(atmospheric_light):
        A = np.array([atmospheric_light, atmospheric_light, atmospheric_light])
    else:
        A = np.array(atmospheric_light)
        if len(A) != 3:
            raise ValueError(f"atmospheric_light应为长度为3的数组，但得到 {len(A)}")

    A = A.reshape(3, 1, 1)

    if transmission.shape != (H, W):
        raise ValueError(f"transmission形状应为 ({H}, {W})，但得到 {transmission.shape}")

    # 对传导图像应用下界阈值
    t_clipped = np.maximum(transmission, t0)
    # 恢复原始图像: J = (I - A) / t + A
    scene_img = (haze_img - A) / t_clipped + A
    scene_img = np.clip(scene_img, 0, 255)

    return scene_img
```

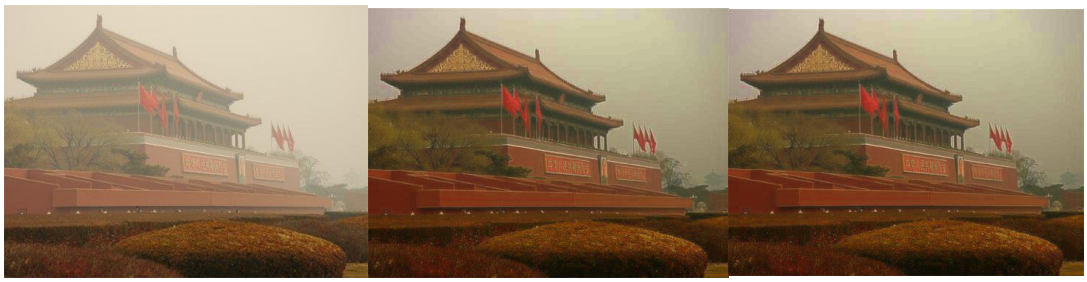
5) 优化传导图像

```python
##5. 优化传导图像
def laplacian_filter(image):
    """应用拉普拉斯算子进行滤波"""
    kernel = np.array([[0, 1, 0], [1, -4, 1], [0, 1, 0]], dtype=np.float32)
    return cv2.filter2D(image, -1, kernel)


def soft_matting_with_laplacian(image, transmission, max_iter=30, epsilon=1e-3):
    """使用拉普拉斯算子改进Soft Matting算法"""
    height, width = image.shape[:2]

    # 初始透射率估计
    transmission = transmission / np.max(transmission)

    # 反向传播迭代
    for _ in range(max_iter):
        # 计算透射率的拉普拉斯算子
        laplacian_transmission = laplacian_filter(transmission)

        # 根据拉普拉斯算子调整透射率
        transmission_new = transmission - epsilon * laplacian_transmission

        # 保证透射率在合理范围内
        transmission_new = np.clip(transmission_new, 0, 1)

        # 如果变化小于阈值，停止迭代
        if np.max(np.abs(transmission_new - transmission)) < epsilon:
            break
        transmission = transmission_new

    return transmission
```

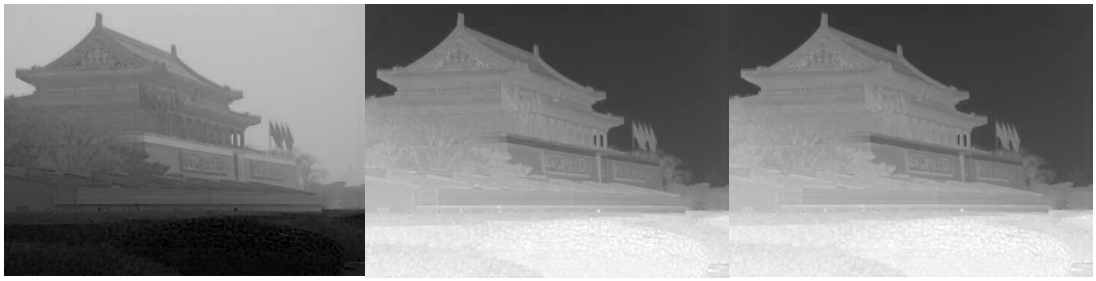
3.实验成果

在最小值滤波大小为 3×3 情况下实验结果。

其中 dark_channel 为图像暗通道信息，transmission-map 为引导图像，softmatting 为优化引导图像，origin 为带雾图像，recover 为普通引导图像恢复结果，recover with softmatting 为优化引导图像恢复结果。

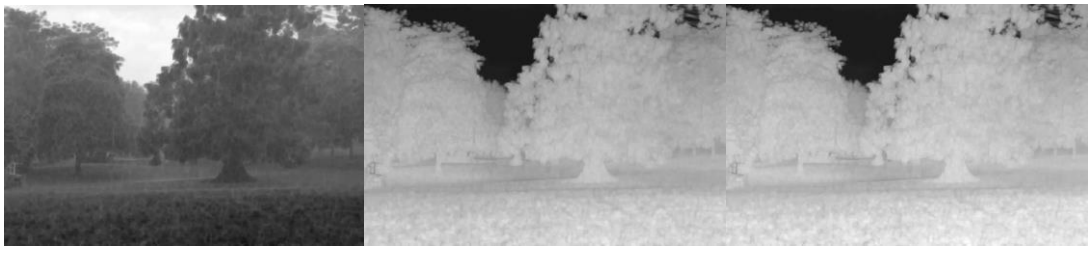


Origin　　recover　　recover with softmatting



Dark_channel　　transmission-map　　softmatting

Origin　　recover　　recover with softmatting



Dark_channel　　transmission-map　　softmatting

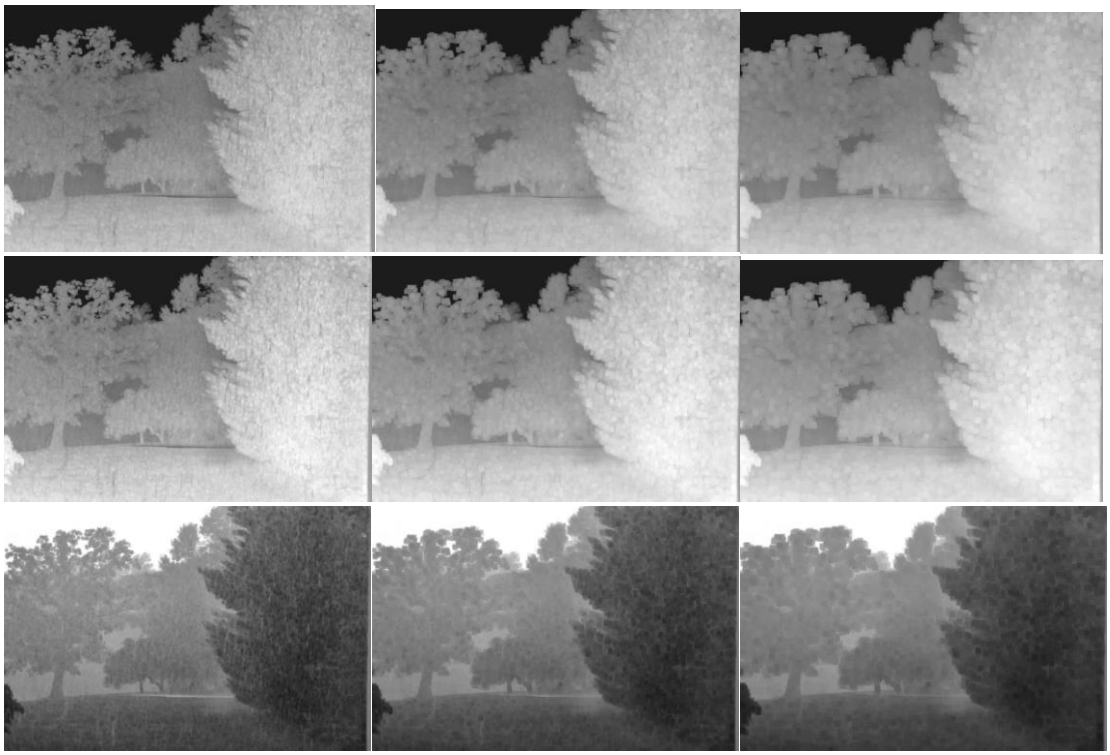
Origin recover recover with softmatting

Dark_channel transmission-map softmatting

Origin recover recover with softmatting

Dark_channel transmission-map softmatting

进一步对比试验，在不同的最小值滤波半径下对比实验结果，得到传导图像，soft-matting 传导图像，暗通道灰度图和 soft-matting 恢复结果如下图所示。



从上至下依次为传导图像，soft-matting 传导图像，暗通道灰度图和 soft-matting 恢复结果，从左至右分别为 radius 选取为 1、2、3，即最小池化卷积核大小为 3×3、4×4、5×5 情况下的结果。

随着半径增大，恢复效果的颗粒感明显下降，并且边界恢复效果明显下降。

4.实验总结或心得感悟

本方法利用暗通道先验，用于单幅图像的去雾。通过对带雾图像、真实图像、引导图像和大气光强的关系建模，并通过求得暗通道先验来反求引导图像，进一步通过拉普拉斯滤波优化了引导图像的求解。