数字图象处理 PJ：传统方法实现图像去雾

陈锐林 21307130148

2024 年 5 月 14 日

1. 背景介绍：

如果天空中有雾霾或者烟雾，我们拍摄的图片会变得模糊不清、对比度降低；不利于对信息的保存和后续的处理。图像去雾即通过一些算法，对图像进行处理，恢复模糊图片的细节和色彩。

1.1. 去雾任务难点概述：

现阶段去雾任务仍有很多难点。

从对雾的物理模型来说，需要较复杂的模型才能完成工作；并且很多时候这是一个不适定问题。

雾天拍摄的图效果差可能由光照、阴影、色彩很多问题造成；求解也复杂。

现实场景复杂，如白天和黑夜对同一场景的拍摄也存在很大区别。

有的场景下无法达到实时性去雾或对高分辨率的图，始终需要较长时间。

真实数据收集难，筛选标准高；合成数据集无法完美模拟现实场景。

1.2. 总结过去、现有方法：

首先是基于大气散射模型的算法。$\boldsymbol{I}(\boldsymbol{x}) = \boldsymbol{D}(\boldsymbol{x}) + \boldsymbol{A}(\boldsymbol{x})$，其中 $\boldsymbol{I}, \boldsymbol{D}, \boldsymbol{A}$ 分别是观测到的图、经过衰减的无雾图、经过衰减的大气光。通过对衰减系数（透射率）的估算，即可恢复出无雾图；如暗通道先验 [6] 就是利用这一模型去估计透射率。在这篇文章之后，很多算法也都基于对透射率的修正和估算。

还有从图像增强的角度出发的算法。如对图像进行对比度拉伸、颜色变换、各种滤波；常见的方法有：直方图均衡化、Retinex 增强、小波变换等等。

以及基于深度学习的去雾算法也崭露头角。深度学习的算法也大多希望通过估计一些系数来恢复图片，比如对单张图片进行估计的 DehazeNet[4]，或者通过卷积采集多尺度特征的 MSCNN[8]，或者通过无监督学习学习参数的 DAD[9]。

总体上看，自从暗通道先验的做法提出之后，传统的方法大都聚焦在任何更好地估计透射率上；也很难做到巨大的突破。而基于深度学习的方法其本身的可解释性就比较差，再加上很多数据集也是人工合成的图片；可能会出现在合成数据集上所向披靡，但在真实世界中屡屡碰壁。所以这仍是一个很值得研究的话题。

2. 选择算法介绍：

本次实验，我实现了 3 种去雾算法；分别是 Baseline–暗通道先验去雾算法 [6]、基于图像增强的自动色彩均衡（ACE）算法 [5]、同样基于图像复原（大气散射模型）的全局去雾算 [3] 法。

在接下来这部分，我会详细介绍这些算法的实现原理；并且进行一定的对比与联系。

2.1. 暗通道先验去雾（Baseline）：

该算法基于大气散射模型：

$$\boldsymbol{I}(\boldsymbol{x}) = \boldsymbol{J}(\boldsymbol{x})t(\boldsymbol{x}) + \boldsymbol{A}(1 - t(\boldsymbol{x})) \tag{1}$$

其中 $\boldsymbol{I}$ 是有雾图像；$\boldsymbol{J}$ 是无雾图像；$\boldsymbol{A}$ 是大气光照值，t 是该点的透射率；公式的左侧就能理解为考虑投射的环境衰减，右侧是环境光照。

该方法还需要定义图像的暗通道；其实就是在 RGB 三个通道中找到最小的值。再稍微考虑到图像的相邻像素间存在紧密联系；考虑将某点 $\boldsymbol{x} = (a, b)$ 的暗通道值定义为以其为中心的正方形窗口某个三个通道中的最小值。定义如下：

$$J^{dark}(\boldsymbol{x}) = \min_{\boldsymbol{y}\in\Omega(\boldsymbol{x})}(\min_{c\in\{r,g,b\}} J^c(\boldsymbol{x})) \tag{2}$$

而该算法的重要的先验就是，在无雾图像 $\boldsymbol{J}(\boldsymbol{x})$ 中，其暗通道 $J^{dark} \to 0$。

回到对 (1) 的变形，单看某个通道 c；变形为：

$$\frac{\boldsymbol{I}^c(\boldsymbol{x})}{\boldsymbol{A}^c} = t(\boldsymbol{x})\frac{\boldsymbol{J}^c(\boldsymbol{x})}{\boldsymbol{A}^c} + (1 - t(\boldsymbol{x})) \tag{3}$$

再考虑窗口的影响，以及假定在较小的窗口内 t 是不变的，就有：

$$\min_{\boldsymbol{y}\in\Omega(\boldsymbol{x})}(\min_c \frac{\boldsymbol{I}^c(\boldsymbol{x})}{\boldsymbol{A}^c}) = \widetilde{t}(\boldsymbol{x}) \min_{\boldsymbol{y}\in\Omega(\boldsymbol{x})}(\min_c \frac{\boldsymbol{J}^c(\boldsymbol{x})}{\boldsymbol{A}^c}) + (1 - \widetilde{t}(\boldsymbol{x})) \tag{4}$$

而根据暗通道先验的条件，$\min_{\boldsymbol{y}\in\Omega(\boldsymbol{x})}(\min_c \frac{\boldsymbol{J}^c(\boldsymbol{x})}{\boldsymbol{A}^c}) \to 0$，于是只要得到 $\widetilde{t}(\boldsymbol{x})$ 就可以复原出图片。

显然，这里的 $\widetilde{t}(\boldsymbol{x})$ 就是：

$$\widetilde{t}(\boldsymbol{x}) = 1 - \omega \min_{\boldsymbol{y}\in\Omega(\boldsymbol{x})}(\min_c \frac{\boldsymbol{I}^c(\boldsymbol{x})}{\boldsymbol{A}^c}) \tag{5}$$

其中，ω 是为了保留一部分的雾成分；会更真实。

在最终去雾前，对 t 进行一定的修正。这里采用的是导向滤波（因为比起 soft matting 效率高很多）。导向滤波的输入是目标图 P（如我们这的 $t(\boldsymbol{x})$）和引导图 I（如我们的原图 $\boldsymbol{I}(\boldsymbol{x})$，输出一张大体上与 P 相似，但是有着 I 纹理的图；能对我们得到的 $t(\boldsymbol{x})$ 进行平滑和优化。具体计算如下：

$$
\begin{aligned}
&1:\ \text{mean}_I = f_{\text{mean}}(I)\\
&\quad\ \text{mean}_p = f_{\text{mean}}(p)\\
&\quad\ \text{corr}_I = f_{\text{mean}}(I.*I)\\
&\quad\ \text{corr}_{Ip} = f_{\text{mean}}(I.*p)\\
&2:\ \text{var}_I = \text{corr}_I - \text{mean}_I.*\text{mean}_I\\
&\quad\ \text{cov}_{Ip} = \text{corr}_{Ip} - \text{mean}_I.*\text{mean}_p\\
&3:\ a = \text{cov}_{Ip}./(\text{var}_I + \epsilon)\\
&\quad\ b = \text{mean}_p - a.*\text{mean}_I\\
&4:\ \text{mean}_a = f_{\text{mean}}(a)\\
&\quad\ \text{mean}_b = f_{\text{mean}}(b)\\
&5:\ q = \text{mean}_a.*I + \text{mean}_b
\end{aligned}
$$

这里的操作都只涉及元素乘（$.*$ 和 $./$），大部分功能由均值滤波实现。所以是在 $O(N)$ 的复杂度。

于是最终图像恢复为：

$$\boldsymbol{J}(\boldsymbol{x}) = \frac{\boldsymbol{I}(\boldsymbol{x}) - \boldsymbol{A}}{\max(t(\boldsymbol{x}), t_0)} + \boldsymbol{A} \tag{6}$$

为了避免因为 $t(\boldsymbol{x})$ 过小导致结果太大，过于割裂。

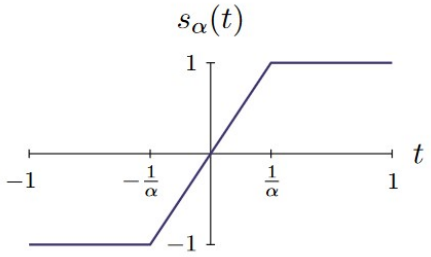
2.2. 自动色彩均衡（Automatic Color Enhancement ACE）：

这个算法从图像增强的角度出发进行去雾；并不对场景做物理建模，而是直接对色调进行调整。根据原文可以看到，是仿照人视网膜的一些机制做的调整方案。

算法第一步是对每个通道值进行色差的校正。类似地，RGB 三个通道分别处理，然后同样以点 $\boldsymbol{x} = (a, b)$ 为中心作一个区域 Ω；利用邻近的信息对 $\boldsymbol{x}$ 做恢复。具体计算如下：

$$\boldsymbol{R}(\boldsymbol{x}) = \sum_{\boldsymbol{y}\in\Omega-\{\boldsymbol{x}\}} \frac{s_\alpha(\boldsymbol{I}(\boldsymbol{x}) - \boldsymbol{I}(\boldsymbol{y}))}{||\boldsymbol{x} - \boldsymbol{y}||} \tag{7}$$

其中 $\boldsymbol{I}$ 是当前计算通道的值（就是计算了亮度差）；距离采用欧氏距离，用来表示远近的权重；s_α 是一个映射函数，表达为 $s_\alpha(t) = \min(1, \max(-1, \alpha t))$。这个函数控制我们的亮度范围，配合上之后的延展操作，可以做到压缩或者扩张范围的作用。其图示如下：



算法第二步是进行延申，这步主要是确保上面输出的 $\boldsymbol{R}(\boldsymbol{x})$ 的上确界和下确

界都达到 1 和 0。具体公式如下：

$$\boldsymbol{L}(\boldsymbol{x}) = \frac{\boldsymbol{R}(\boldsymbol{x}) - \min R}{\max R - \min R} \tag{8}$$

最后将图像重新乘回 8bit 的范围就可以了。

在这篇文献中，采用了卷积或者插值来降低复杂度。这个实现可能比较麻烦，而 [11] 提出了另一种快速的做法；其主要利用了两个假设：将图片做多次 ACE 和将图片进行尺寸缩放后做 ACE 并不会很大影响结果。

若基于以上假设，就能使用以下快速算法。主要思路在于缩小分辨率来加速计算。

Algorithm 1: FastACE($\boldsymbol{I}, \alpha, \beta$)

Input: 待去雾图像 $\boldsymbol{I}$，亮度函数中的 α，标识 Ω 范围的 β

Output: 去雾图

1 h,w = 图的长和宽;

2 将 $\boldsymbol{I}$;

3 resize 为 $(\frac{h+1}{2}, \frac{w+1}{2})$，储存在 $\boldsymbol{I'}$;

4 递归求解 HalfI = FastACE($\boldsymbol{I'}, \alpha, \beta$);

5 将 HalfI 和 $\boldsymbol{I'}$ 重新变回原来分辨率;

6 最后输出结果：HalfI + ACE($\boldsymbol{I}, \alpha, \beta$) - ACE($\boldsymbol{I'}, \alpha, \beta$);

这里其实就是通过降低分辨率先得到一个信息缺失的图；之后再通过原图进行一些信息的补充，即加上原图中各个信息的相关关系（因为前面的 $\boldsymbol{R}$ 就是考察了周围信息得到的）再减掉当前元素的周围信息。

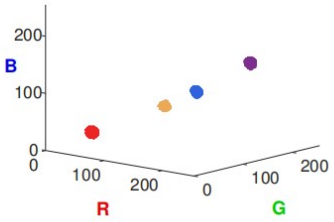
看起来，这并没有降低工作量，因为我们还在调用原来的 ACE 接口，然而原先的 Ω 通常是一个很大的数，现在只需要设定为很小就可以了。（比如取 300 和 3，就是意味指 90000 和 9 个点的区别）。

2.3.Non-Local Image Dehazing：

这个算法同样基于 (1) 中的大气散射模型和 Baseline 的暗通道先验；这里主要做的工作就是进一步完善 $t(\boldsymbol{x})$ 的计算。

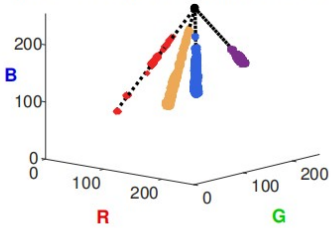
这个算法还有其余的先验条件。首先是一张图中不同的颜色数远小于总的像素数；所以可以把所有的像素根据某种规则分为一些簇类（这里取 500 个）。还可以判断，在清晰图像中归为一类的点，在有雾图中会形成一条线（haze-line），在这条线的两端就是原色 $\boldsymbol{J}$ 和大气光 $\boldsymbol{A}$。这点从下面这个图能很清晰地看到，并且在清晰图中距离近的（如紫色）在 haze-line 中一样离得近；而像红色部分就会分得更开。

(a) Haze-free image



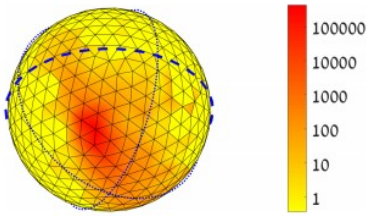
(b) Corresponding *clusters*

(c) Synthetic hazy image.



(d) Corresponding *haze-lines*

算法第一步就是找到 haze-line。取 $\boldsymbol{I}_A = \boldsymbol{I}(\boldsymbol{x}) - \boldsymbol{A}$ 为我们要寻找的；将其表示为 $[r(\boldsymbol{x}), \theta(\boldsymbol{x}), \phi(\boldsymbol{x})]$。这里 $r(\boldsymbol{x})$ 表示的距离，θ, ϕ 用来标识其位置，可以理解为在上面这种 RGB 三个坐标中，已知 $\boldsymbol{A}$，就还需要 3 个参数来找到当前的像素点。而利用公式 (1) 对 $\boldsymbol{I}_A$ 进行变形，就可以得到 $\boldsymbol{I}_A = t(\boldsymbol{x}) \cdot [\boldsymbol{J}(\boldsymbol{x}) - \boldsymbol{A}]$。考虑同一条线上的点，它们只有 t 是不同的，所以对于他们的 θ, ϕ 应该是一样的。那么我们如果我们作出一个球体来放置所有的像素点，每条 haze-line 上的点就处于同一个小方块，如下图。



于是我们可以认为对两个共线的点，$\boldsymbol{J}_1 - \boldsymbol{A} = \alpha(\boldsymbol{J}_2 - \boldsymbol{A})$，其中 α 是一个比例。

在具体实现上，我们在对点进行分类时，就是用我们的 θ, ϕ 去和上面这个球表面的所有点比距离；簇类的实现使用 KD-Tree 实现即可，会比 k-means 快不少。

算法的第二步就是初步估计初始的透射率 $t(\boldsymbol{x})$。根据前面的定义 $\boldsymbol{r}(x) = t(\boldsymbol{x}) \, ||\boldsymbol{J} - \boldsymbol{A}||$，于是取透射率为 1 时的值为 $r_{\max}$，那么 $t(\boldsymbol{x}) = \dfrac{r(\boldsymbol{x})}{r_{\max}}$。而针对每一簇 haze-line，我们要找到最大的 $\hat{r}_{\max}$，就只要遍历所有的点，找到 $\max\limits_{H}\{r_{\max}\}$ 即可。

算法第三步是对 $t(\boldsymbol{x})$ 做一些修正，原文的方法是利用类似于拉格朗日乘子去算；而我这直接复用了 Baseline 中的 Guided Filter 做修正。

算法第四步是完成图像复原的工作，和 Baseline 的做法一样，不再赘述。

3. 实验介绍：

本次实验对 4 个网络上的数据集：D-HAZY[2]/HazeRD[10]/O-HAZE[1]/RESIDE[7]，以及自己搜集的 5 张图片作了去雾的工作。

代码几乎都忠实地完成了算法的内容，除了个别地方修改（如将 NonLocal 的 regularization 改成 GuidedFilter、并在 Baseline 和 NonLocal 最后都加了一个轻微的色调校正函数）。

对于所有图片都将其尺寸改到 600×400（长，宽）；在一样的环境下运行了函数。最后记录了各个算法在每个数据集上的运行时间及各项指标，并且对实验结果进行了对比分析。

3.1. 实验启动：

项目内容已上传至 Github，通过命令：git clone https://github.com/lrcc275/DIP-PJ.git 可以直接克隆。或者访问网址 (point here) 下载。

项目文件夹目录如下图。大部分的文件含义已经注明。较重点的是 Report_21307130148_陈锐林.pdf 是本次项目的报告；在代码中，evaluate.py 是用来整个数据集跑的，需要手动修改里面的 haze_folder_path 和 gt_folder_path，以及下面 for 循环中 dehaze 方法的选择，因为数据集的图片格式不一定相同，需要相应更改后面的通配符内容。main.py 函数提供了一张图片的去雾（并展示），不过需要自己填上路径。

```
—DIP-PJ                              #Latex 源代码
    —Figures                         #报告相关图片
        —DIP1.jpg
        —DIP2.jpg
        —.....
    —main.tex                        #编译文件
—DIP-Project                         #python 源代码
    —D-HAZY-dehaze                   #数据集，下同
    —D-HAZY-haze
    —HazeRD-dehaze
    —HazeRD-haze
    —O-HAZY-dehaze
    —O-HAZY-haze
    —RESIDE-dehaze
    —RESIDE-haze
    —Self-select-pictures            #自己搜集的五张图片
    —venv
    —Baseline_DCP.py                 #Baseline 方法
    —eval.py                         #定义 MSE PSNR SSIM 的计算
    —evaluate.py                     #测试程序
    —ImageEnhancement_ACE.py         #ACE 方法
    —ImageRestoration_NonLocal.py    #NonLocal 方法
    —main.py                         #可以查看单张图片结果
    —NonLocal_Data.txt               #NonLocal 用到的数据
—Report_21307130148_陈锐林.pdf       #报告
```

3.1.1. 数据集介绍：

数据集名	选用部分	图片对数	合成雾/真实雾	场景
D-HAZY	NYU_GT	1449	合成	各种场景
HazeRD	all	15(75 张有雾选最模糊的)	合成	室外
O-HAZE	all	45	真实	室外
RESIDE	SOTS	542	合成	室外居多，少部分室内

此外还有自己收集的 5 张图片，后面会展示出来。

3.1.2. 实验设置介绍：

实验的基础设置就是将每张图片都转为 600×400 的大小，并且对每种算法、每个数据集进行测试；记录下各个指标即可。

在一些参数上，Baseline 保留设定，即 $r = 7$ 的滤波半径、0.001% 的 $\boldsymbol{A}$ 阈值、0.1 的 $t_{\min}$。ACE 中，α 设定为 3，$r = 4$ 的 Ω 范围。在 NonLocal 中，暗通道的参数和 Baseline 一致。

3.1.3. 指标介绍：

主要就是用到三个指标 $MSE, PSNR, SSIM$。

MSE 是去雾图和数据集提供的清晰图之间，每个像素的平均差值。考虑到我们是彩色图像，所以 MSE 计算为：$\frac{1}{H \times W \times C} \sum_{i,j,c} (DI(i,j,c) - GT(i,j,c))^2$，这里 DI, GT 就是经过算法 dehaze 后的以及原始图。

$PSNR$ 是峰值信噪比，其表达式为 $10 \log_{10}(\frac{(2^n - 1)^2}{MSE})$。$n$ 为每像素的比特数，这里取 8。

$SSIM$ 从亮度、对比度、结构三个方面衡量了图像间的相似性。我们先计算出两幅图像的均值、方差、协方差。分别记为：$\mu_x, \sigma_x, \mu_y, \sigma_y, \sigma_{xy}$。那么 $SSIM$ 就等于：

$$l(x,y)c(x,y)s(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3}$$

一般来说，三个常数分别取为 $(0.01 \times 255)^2, (0.03 \times 255)^2, (0.01 \times 255)^2/2$。

$CIEDE2000$ 是一个重新定义了色差的指标。首先它将图片转为 $L^*a^*b^*$ 空间(三个字母表示: 亮度和两个对立维度)。它的计算如下：

$$\Delta E = \sqrt{(\frac{\Delta L}{K_L S_L})^2 + (\frac{\Delta C}{K_C S_C})^2 + (\frac{\Delta H}{K_H S_H})^2 + R_T(\frac{\Delta C}{K_C S_C})(\frac{\Delta H}{K_H S_H})}$$

其中 $\Delta L, \Delta H, \Delta C$ 分别是亮度差、色相差、色度差；S_L, S_C, S_H 是三个由其他值计算出的权重信息；K_L, K_C, K_H 是人工设置的因子；R_T 是用于校正偏差的旋转因子。

3.2. 实现方法在数据集上的各种指标结果：

首先是每个算法在各个数据集上的结果。

Baseline	O-HAZE	RESIDE	HazeRD	D-HAZY
时间/s	82.358	815.400	26.024	2195.332
MSE/img	105.022	102.285	99.141	102.647
$PSNR$/img	27.924	28.047	28.179	28.025
$SSIM$/img	0.654	0.659	0.761	0.713
$CIEDE2000$/img	30.873	28.114	17.380	32.234

ACE	O-HAZE	RESIDE	HazeRD	D-HAZY
时间/s	58.399	526.039	18.377	1429.121
MSE/img	105.068	106.178	104.331	105.204
$PSNR$/img	27.924	27.875	27.950	27.912
$SSIM$/img	0.668	0.604	0.778	0.674
$CIEDE2000$/img	29.847	37.027	23.046	36.346

NonLocal	O-HAZE	RESIDE	HazeRD	D-HAZY
时间/s	62.654	550.781	22.630	1679.926
MSE/img	103.014	102.615	99.970	103.680
$PSNR$/img	28.007	28.032	28.136	27.981
$SSIM$/img	0.655	0.683	0.810	0.703
$CIEDE2000$/img	31.052	28.395	18.064	32.488

以及每个算法的平均指标的对比。

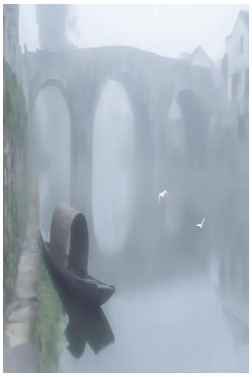
	Baseline	ACE	NonLocal
时间/s	1.521	0.991	1.129
MSE/img	102.577	105.452	103.357
$PSNR$/img	28.030	27.903	27.997
$SSIM$/img	0.698	0.656	0.697
$CIEDE2000$/img	31.007	36.286	31.270

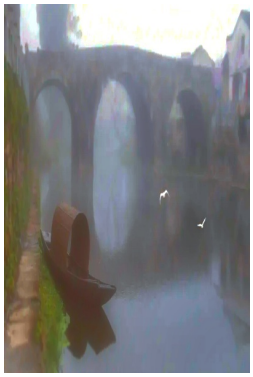
自己搜集的图像在下一部分展示。

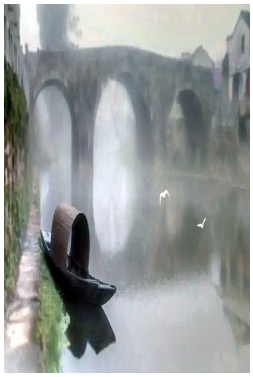
3.3. 样例介绍：

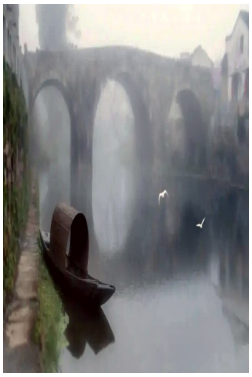
首先展示我收集的 5 张有雾图像的去雾效果。

p1 来自：【视觉影像】津门网江新华：雾景视界的美学观照.

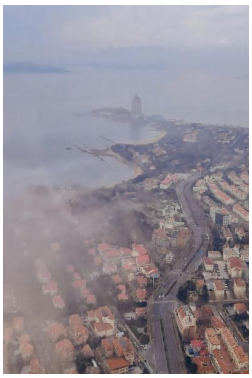

p1-原图

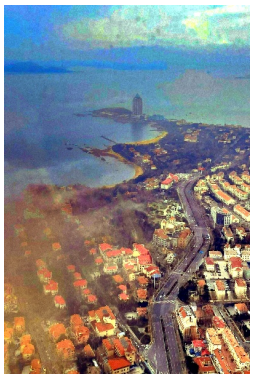

p1-Baseline


p1-ACE

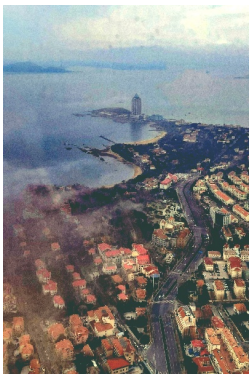

p1-NonLocal

p2 来自：手机摄影 | 这不是油画这是青岛今天的平流雾.

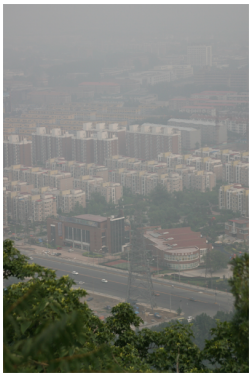

p2-原图

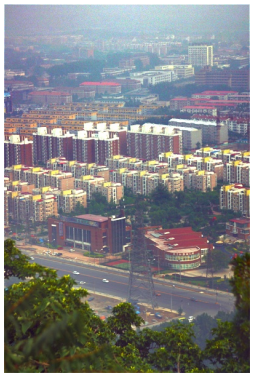

p2-Baseline

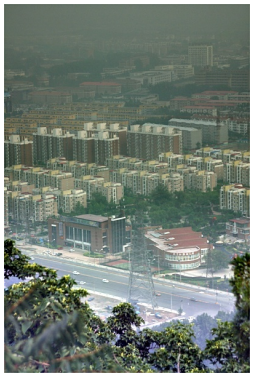

p2-ACE

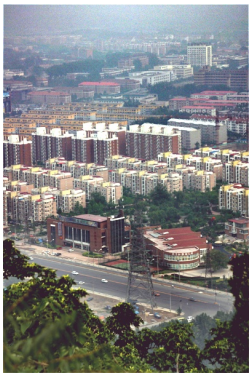

p2-NonLocal

p3 来自：[6]

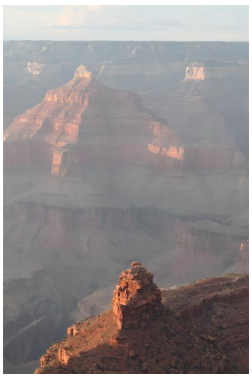
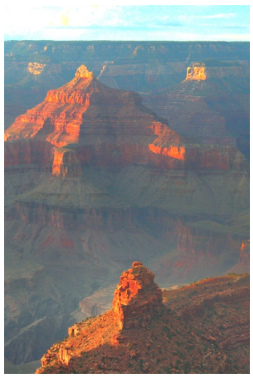

p3-原图


p3-Baseline


p3-ACE


p3-NonLocal

p4 来自：Adobe 展示模拟“时光机”与超强除雾新技术

p4-原图　　p4-Baseline　　p4-ACE　　p4-NonLocal

p5 来自：上海现绝美平流雾，市民游客夜游宛如仙境的外滩.



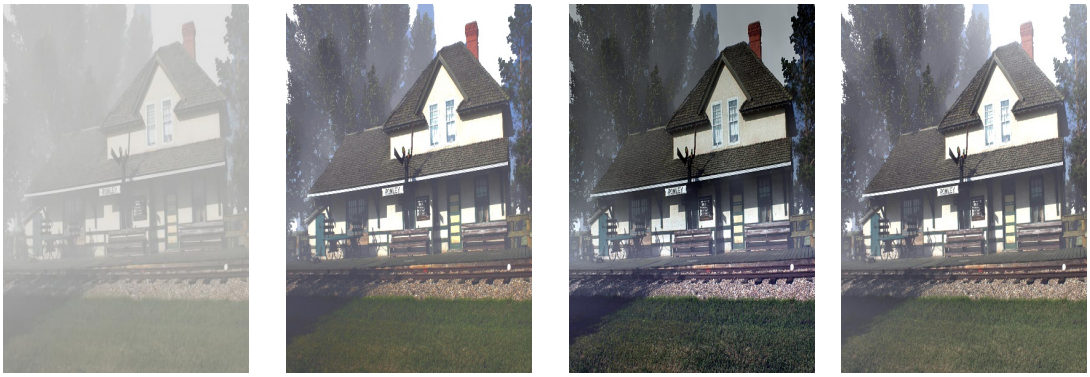

p5-原图　　p5-Baseline　　p5-ACE　　p5-NonLocal

以及从各位数据集取的一张结果。
从 O-HAZY 截取的：




原图　　Baseline　　ACE　　NonLocal

从 HazeRD 截取的：



原图　　Baseline　　ACE　　NonLocal

从 D-HAZY 截取的：

原图　　Baseline　　ACE　　NonLocal

从 RESIDE 截取的：

原图　　Baseline　　ACE　　NonLocal

需要注意的是因为 Baseline 和 NonLocal 方法直接生成会有点暗，所以这里是接了一个色彩的增强模块的（具体可以看代码），而 ACE 没有做（因为看着已经比较舒服了）。

4. 结论与感悟：

从本次实验的数据来看，Baseline 算法的速度是较慢的；相比其和 NonLocal 的算法，透射率的求解是不同的。在 Baseline 中，是用腐蚀操作，对一个单元要考察范围是 15×15 的元素，就是 $600 \times 400 \times 15\times$；在 NonLocal 中是用 kdtree 来实现查找，查找的数据集是作者提供的一个 1000 个坐标的文件，所以对一个单元，进行查找的平均复杂度是 $O(\log n)$，$n = 1000$，即使 opencv 可能做出了相关的优化，NonLocal 算法的开销还是小很多。同理，对 ACE 算法来说，首先它是递归求解的，真正求解是我们只要对周围的 $3 \times 3/4 \times 4$ 的元素求解色差即可；这开销也是小的。

从其他指标结果来看，MSE 和 $PSNR$ 是差不多的，大概就是一个像素点有 10 的差距。而在 $SSIM$ 和 $CIEDE2000$ 上能较明显地看到暗通道的思想的优势。所以如果要选择一种适中的算法，兼具速度优势和效果优势的 NonLocal 会更好。

但是也必须说，指标的计算和人眼的测量差距还是很大的。如果对比着清晰图和经过算法去雾后的图，像上面 O-HAZY 那种，就会感觉 Baseline 的算法也太艳了点，ACE 的还更接近现实，NonLocal 比较适中；但是从指标上来看差距并不强烈；当然这里的“艳”是因为加了一个色彩校正，但不加的话反而暗得吓人，感觉更糟糕。

从未来的改进来说，如果还采用暗通道优先的方式，透射率的计算还是需要再改进；比如 NonLocal 中是有 1000 个预先提取的数据做 KDtree 的搜索，可以再多些点。并且 NonLocal 这种方法非常依赖于这些数据，选择更优的数据集是可行之策；并且如何在提高数据量的同时不让复杂度太大也是值得思考的。而 ACE 算法一个可行的方向是修改我们的 s_α 函数的设计，可以采用其他符合要求的奇函数。

总的来说，在传统方法上，去雾工作仍然还有很大的改善空间。特别是在前几个数据集都鲜少有对夜间雾图的数据；因为夜间的情况也较复杂，可能非常雾和灯光（比如高层建筑物的光）是交杂在一起的，人眼能很好区分，但是计算机可能不行。

参考文献

[1] Codruta O. Ancuti, Cosmin Ancuti, Radu Timofte, and Christophe De Vleeschouwer. O-haze: a dehazing benchmark with real hazy and haze-free outdoor images. In *IEEE Conference on Computer Vision and Pattern Recognition, NTIRE Workshop*, NTIRE CVPR'18, 2018.

[2] Cosmin Ancuti, Codruta O. Ancuti, and Christophe De Vleeschouwer. D-hazy: A dataset to evaluate quantitatively dehazing algorithms. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 2226–2230, 2016.

[3] Dana Berman, Tali treibitz, and Shai Avidan. Non-local image dehazing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[4] Bolun Cai, Xiangmin Xu, Kui Jia, Chunmei Qing, and Dacheng Tao. Dehazenet: An end-to-end system for single image haze removal. *IEEE Transactions on Image Processing*, 25(11):5187–5198, 2016.

[5] Pascal Getreuer. Automatic Color Enhancement (ACE) and its Fast Implementation. *Image Processing On Line*, 2:266–277, 2012. `https://doi.org/10.5201/ipol.2012.g-ace`.

[6] Kaiming He, Jian Sun, and Xiaoou Tang. Single image haze removal using dark channel prior. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2011.

[7] Boyi Li, Wenqi Ren, Dengpan Fu, Dacheng Tao, Dan Feng, Wenjun Zeng, and Zhangyang Wang. Benchmarking single-image dehazing and beyond. *IEEE Transactions on Image Processing*, 28(1):492–505, 2019.

[8] Wenqi Ren, Jinshan Pan, Hua Zhang, Xiaochun Cao, and Ming Hsuan Yang. Single image dehazing via multi-scale convolutional neural networks with holistic edges. *International Journal of Computer Vision*, 128(1):240–259, 2020.

[9] Yuanjie Shao, Lerenhan Li, Wenqi Ren, Changxin Gao, and Nong Sang. Domain adaptation for image dehazing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[10] Yanfu Zhang, Li Ding, and Gaurav Sharma. Hazerd: An outdoor scene dataset and benchmark for single image dehazing. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3205–3209, 2017.

[11] 张方略, 徐欣宇, and 胡事民. 自动色彩均衡算法的速度优化. 武汉大学学报: 工学版, 46(6):6, 2013.