

# 《数 字 图 象 处 理》

## 综合作业 3

综合报告

班级：自 45

姓名：林子坤

学号：2014011541

## 目录

<b>一、项目简介</b> .....	<b>4</b>
<b>二、程序属性</b> .....	<b>4</b>
<b>三、需求分析与设计思路</b> .....	<b>4</b>
<b>四、算法原理与设计<sup>[1]</sup></b> .....	<b>5</b>
<b>4.1 步骤一：去除沙尘颜色</b> .....	<b>5</b>
4.1.1 方法：三通道明暗处理（McCann Retinex 算法 <sup>[2]</sup> ）.....	5
4.1.1.1 算法原理 <sup>[3]</sup> .....	5
4.1.1.2 算法设计 <sup>[4]</sup> .....	6
4.1.2 处理结果 .....	7
<b>4.2 步骤二：消除雾气影响</b> .....	<b>7</b>
4.2.1 方法 1：RGB 直方图均衡处理 .....	7
4.2.1.1 算法原理 .....	7
4.2.1.2 算法设计 .....	7
4.2.1.3 处理结果 .....	8
4.2.2 方法 2：HSI 直方图均衡处理 .....	8
4.2.2.1 算法原理 .....	8
4.2.2.2 算法设计 .....	9
4.2.2.3 处理结果 .....	9
4.2.3 方法 3：同态滤波（使用 Retinex 图像增强算法原理） .....	10
4.2.3.1 算法原理 .....	10
4.2.3.2 算法设计 .....	12
4.2.3.3 处理结果 .....	12
4.2.4 方法 4：对比度调节 .....	13
4.2.4.1 算法原理 .....	13
4.2.4.2 算法设计 .....	13
4.2.4.3 处理结果 .....	13
4.2.5 方法 5：引导滤波 .....	14
4.2.5.1 算法原理 <sup>[5]</sup> .....	14
4.2.5.2 算法设计 <sup>[6]</sup> .....	14

4.2.5.3 处理结果 .....	15
4.2.6 方法 6：暗原色先验滤波 .....	16
4.2.6.1 算法原理 <sup>[8]</sup> .....	16
4.2.6.2 算法设计 .....	16
4.2.6.3 处理结果 .....	17
<b>五、程序结构.....</b>	<b>17</b>
<b>六、输出结果与讨论 .....</b>	<b>18</b>
<b>6.1 原图对比测试 .....</b>	<b>18</b>
<b>6.2 算法鲁棒性测试 .....</b>	<b>18</b>
6.2.1 使用相似风景图测试（4 张） .....	18
6.2.2 使用二次元图片测试（2 张） .....	21
6.2.3 使用清华大霾图片测试（3 张） .....	23
<b>七、心得与体会 .....</b>	<b>26</b>
<b>参考文献.....</b>	<b>26</b>

# 综合作业3 图像去雾

## 综合报告

林子坤

(自动化系 自45班 2014011541)

### 一、项目简介

本次项目的功能是使用适当的图像增强算法，实现图像的增强，尽可能消除雾气的影响。在本次项目中，本人自行编写了前4种算法的代码，并通过查阅论文和参考已有代码的方式理解并实现了引导滤波算法、暗原色算法的去雾功能。在项目的最后，使用了原始图片、4张相似图片和5张其他图片（共计10张图片）进行鲁棒性测试，获得的效果图片附于文末以便参照与对比。

### 二、程序属性

本程序使用 Matlab(R2016a, maci64)编写，“.m”文件支持的运行环境是所有装有 Matlab 的计算机。

以下是一张较为理想的去雾图片与原始图像的对比效果图。



原始图片



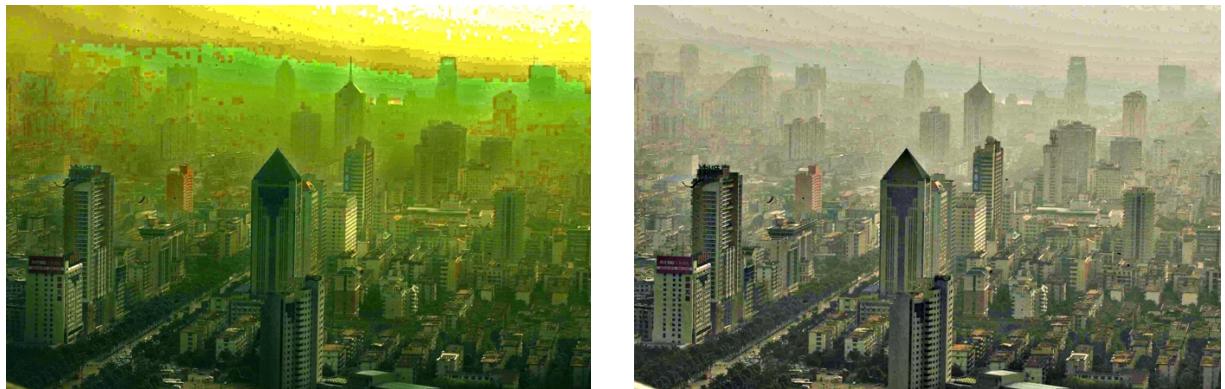
去雾后效果

### 三、需求分析与设计思路

“雾是帝都重，霾是故乡醇。”生活在北京的我们时常会体验到大雾、大霾、沙尘等恶劣的空气条件。如果在这样能见度极低的条件下进行风景照的拍摄，我们不免会吐槽：“这么黄！分明啥都看不清呀！”而在南方或是山区，潮湿的空气经常会带来水气十足的大雾，这也会造成拍摄照片时无法看清照片中的风景。实际上，这些照片还是可以“抢救”一下的一

一如果使用合适的图像增强算法，我们可以在一定程度上消除雾气的影响，让大雾缭绕的照片恢复本来的样子。

在进行去雾之前，同学们都发现了原始图像中的一个问题——原始图像可能是在较为严重的沙尘环境下拍摄的，因此除了“雾”以外还有“污”的影响。如果仅仅简单地进行图像增强去雾，势必会造成建筑物颜色增强的同时，黄色背景也被加强了。实验验证了我们的想法，以下为直接使用原始图片进行去雾算法所得到的结果，可谓是“黑云压城城欲摧，甲光向日金鳞开”。



为了解决这个问题，设计算法时，在进行去雾之前应该对图片进行预先处理，在查阅资料后，本人决定使用“三通道明暗处理”的方法。而在随后的图像去雾步骤中，现在有几种主流的算法，其中最为著名的便是 2009 年 CVPR（计算机视觉与模式识别会议）最佳论文“Single Image Haze Removal”。而其他一些常规做法例如 RGB 直方图均衡化、同态滤波等也时常能取得不错的效果，各个算法的原理、实现与比较将在正文中予以阐述。

## 四、算法原理与设计<sup>[1]</sup>

### 4.1 步骤一：去除沙尘颜色

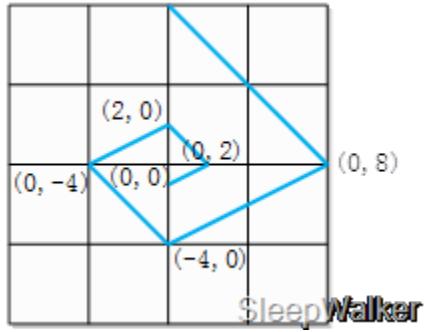
#### 4.1.1 方法：三通道明暗处理（McCann Retinex 算法<sup>[2]</sup>）

##### 4.1.1.1 算法原理<sup>[3]</sup>

McCann Retinex 算法是 McCann 和 Frankle 一起提出的一种 Retinex 算法，该算法是一种基于多重迭代策略的 Retinex 算法，单个点的像素值取决于一条特定路径的环绕的结果，经过多次迭代逼近理想值。通过比较螺旋式路径上的各像素点的灰度值来估计和去除图像的照度分量。

该算法估测图像经过以下几个步骤：

- 1、将原图像的三个彩色通道均变换到对数域；
- 2、初始化常数图像矩阵  $R_0(x, y)$ ，作为进行迭代运算的初始值；
- 3、计算路径，如下图所示，令  $S_1, S_2, \dots, S_m$  为路径上的点，从远到近排列；



4、对路径上的像素点按照如下公式运算：

$$\begin{aligned} R_c &= \frac{I_m + R_c}{2} = \frac{R_m + S_c - S_m + R_c}{2} = \frac{S_c - S_m}{2} + \frac{R_m + R_c}{2} \\ &= R_0 + \frac{r_c - r_m}{2} + \frac{r_c - r_{m-1}}{2} + \dots + \frac{r_c - r_1}{2} \end{aligned}$$

其中  $R_c$  表示中心位置最终的反射率估计， $R_0$  为常数值为转换后的图像中的最大值，在步骤 2 中被确定。最终公式中可以看到，最终照度分量被去除了，而中心位置的反射率由路径上各点的反射率之间的差值进行估计，并且从轨迹上可以看到，靠的越近的在最终估计的时候所占比重越大。

此迭代方案基于成对像素亮度值间的互动，这一像素对在图像中的坐标  $(x, y)$ 、 $(x_s, y_s)$ 。第一步处理的像素被预定义的距离  $shift$  分开，下一步将比较方向顺时针旋转 90 度，距离  $D$  减半，直至到达单位像素距离。每一步计算中迭代次数  $nIterator$  由用户决定。每次迭代有四个步操作：比例(ratio)、乘积(product)、重置(reset)、平均(average)。

在程序中，设  $OP$  为上一步迭代的乘积； $NP$  为当前迭代的乘积； $IP$  为中间乘积结果； $R$  为原始图像；符号 \* 表示重置操作。每步迭代使用如下公式估计点处的明度值：

$$NP(x, y) = \sqrt{OP(x, y) * IP(x, y)} = \sqrt{OP(x, y)[OP(xs, ys) \frac{R(x, y)}{R(xs, ys)}]^*}$$

#### 4.1.1.2 算法设计<sup>[4]</sup>

注：本部分算法设计参考了来自 Simon Fraser University 的 Florian Ciurea, Brian Funt 和 John McCann 的 Matlab 功能实现代码。

初始距离  $shift$  按照经验值设置为 2 的指数：

$$shift = 2^{fix[\log \min(rows, cols) - 1]}$$

在迭代次数范围内且距离  $shift$  不小于 1 时，分别进行列操作和行操作，分别按行、按列按照以上公式进行加减运算。且每进行一次运算后用  $-\frac{shift}{2}$  代替  $shift$  作为新的距离。

行列操作的具体过程如下：

(1) 列操作：开始行变量  $s\_row$  设置为 0，开始列变量  $s\_col$  设置为  $shift$ 。用  $shift$  列右侧的块减去左侧的块，并赋值给右侧块的相应位置。如果  $shift$  为负数，则用  $shift$  列左侧的块减去右侧的块，并赋值给左侧块的相应位置。

(2) 行操作：开始行变量  $s\_row$  设置为  $shift$ , 开始列变量  $s\_col$  设置为 0。用  $shift$  列下侧的块减去上侧的块，并赋值给下侧块的相应位置。如果  $shift$  为负数，则用  $shift$  列上侧的块减去下侧的块，并赋值给上侧块的相应位置。

#### 4.1.2 处理结果

经过 McCann Retinex 算法处理后，结果对比图如下图所示。可以看出，黄色的沙尘颜色基本被完全去除，但还有较为较强的“雾感”，需要更进一步地进行去除。



## 4.2 步骤二：消除雾气影响

### 4.2.1 方法 1：RGB 直方图均衡处理

#### 4.2.1.1 算法原理

直方图均衡化的算法原理是：假定图像的总点数为  $N$ ，共 256 级灰度，则理想情况变换后对应于每一灰度  $k$  累积点数为  $(k + 1)\frac{N}{256}$  ( $k = 0, \dots, 255$ )。对于原图的每一灰度  $i$  ( $i=0, \dots, 255$ )，使得其累计点数  $\sum_0^i n_i$  最接近于理想累积数  $(k + 1)\frac{N}{256}$  ( $k = 0, \dots, 255$ ) 的  $i-k$  关系即是变换函数。

#### 4.2.1.2 算法设计

本程序首先将原图像的三个彩色通道分离，并分别通过以下步骤实现直方图均衡化。

(1) 统计各灰度级像素总数目并记录于 PixelN 数组。由于 Matlab 中数组下标从 1 开始，因此存储时将每个像素的灰度值加 1 进行储存。

(2) 计算各灰度级分布密度并记录于 PixelP 数组。

$$\text{分布密度} = \text{灰度级像素总数目} / \text{总像素数}$$

其中：

$$\text{总像素数} = \text{图片高度} \times \text{图片宽度}$$

可得：

$$\text{PixelP}(i) = \text{PixelN}(i) / (\text{height} * \text{width})$$

(3) 根据以下方式递推，计算灰度级累计分布并记录于 PixelAccu 数组。

$$\text{PixelAccu}(1)=\text{PixelP}(1)$$

$$\text{PixelAccu}(i)=\text{PixelAccu}(i-1)+\text{PixelP}(i)$$

(4) 将累计分布转换为 0~255 的灰度级并记录于 PixelDisc 数组。由于灰度级具有离散性，因此对计算结果强制类型转换为 uint8 形式（即 0~255）。

(5) 将各灰度值映射存储于新图像中。

#### 4.2.1.3 处理结果

经过 RGB 直方图均衡处理后，结果对比图如下所示。



原始图片



处理后效果

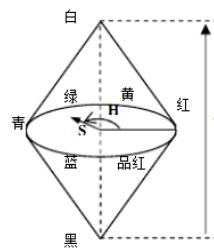
可以看出，对于近处的建筑物，雾气的影响基本上被完全去除，颜色基本上接近于日常生活中实际建筑物的颜色，且对于图片中原本难以辨别的部分（如“阳光 100”标牌）做到了较为完整的恢复。

然而，对于远处的建筑物，该方法获得的结果雾气的影响却依旧较为明显。个人分析，原因有二：1、远处建筑物在原始图片中受雾气影响过重，信息还原较为艰难；2、RGB 直方图均衡处理方法完全根据灰度值而定，对所处理的数据不经选择，因此对于一些噪声不仅不能消除，还可能会增强。

#### 4.2.2 方法 2：HSI 直方图均衡处理

##### 4.2.2.1 算法原理

HSI 模型中的 H、S、I 分别代表色调 (Hue)、饱和度 (Saturation)、强度 (Intensity)。HSI 模型是开发基于彩色描述的图像处理算法的理想工具，能够以对人来说自然且直观的方式进行彩色描述。



#### 4.2.2.1.1 RGB 转 HSI

公式如下：

$$H = \begin{cases} \theta & \text{if } (B \leq G) \\ 360 - \theta & \text{if } (B > G) \end{cases}$$
$$\theta = \cos^{-1} \left[ \frac{(R-G)+(R-B)}{2\sqrt{(R-G)^2 + (R-B)(G-B)}} \right]$$
$$S = 1 - \frac{3 \min(R, G, B)}{R+G+B}$$
$$I = \frac{1}{3}(R+G+B)$$

#### 4.2.2.1.2 HSI 直方图均衡处理

利用 4.2.1 中的直方图均衡处理方法对强度分量 I 进行直方图均衡处理。由于进行均衡处理后图像饱和度会下降，因此还需要对 S 分量进行适当增强，本程序中将 S 分量进行了 3 倍倍乘。

#### 4.2.2.1.3 HSI 转 RGB

公式如下：

$$0 \leq H \leq 120^\circ \quad 120^\circ \leq H \leq 240^\circ \quad 240^\circ \leq H < 360^\circ$$
$$R = I(1 + \frac{S \cos H}{\cos(60^\circ - H)}) \quad R = I(1 - S) \quad R = 1 - (G + B)$$
$$G = 1 - (R + B) \quad G = I(1 + \frac{S \cos H}{\cos(60^\circ - H)}) \quad G = I(1 - S)$$
$$B = I(1 - S) \quad B = 1 - (R + G) \quad B = I(1 + \frac{S \cos H}{\cos(60^\circ - H)})$$

#### 4.2.2.2 算法设计

对强度分量 I 进行直方图均衡处理方式与 4.2.1.2 中一致，此处不再赘述。

#### 4.2.2.3 处理结果

经过 HSI 直方图均衡处理后，结果对比图如下所示。

可以看出，对于近处的建筑物，雾气的影响基本上被完全去除，且对于图片中原本难以辨别的部分（如“阳光 100”标牌）做到了较为完整的恢复，效果与 RGB 直方图均衡处理相似。

然而，对于远处的建筑物，该方法获得的结果仍存在 RGB 直方图均衡处理的相似问题，且颜色失真较为明显。个人分析这是因为为了增加饱和度，人为将 S 分量进行了 3 倍倍乘，而实际上这个数字很可能是不准确的，而且也是因图而异的。



原始图片

处理后效果

### 4.2.3 方法 3：同态滤波（使用 Retinex 图像增强算法原理）

和上次大作业相似的是，本次大作业的任务依旧有类似“入射光”“反射光”照射问题，也有对过白的部分进行信息恢复的要求，因此本人自然地想到了上次的同态滤波方法。然而，直接套用上次的代码所得到的结果并不尽如人意，因此在查阅资料后，设计的方法基于 Retinex 图像增强算法原理，并取得了比较好的效果。

#### 4.2.3.1 算法原理

##### 4.2.3.1.1 简单同态滤波

注：本部分内容参考了冈萨雷斯《数字图像处理》第 4 章“4.9.6 同态滤波”一小节中的内容。

一幅图像  $f(x, y)$  可以表示为其照射分量  $i(x, y)$  和反射分量  $r(x, y)$  的乘积，即

$$f(x, y) = i(x, y) r(x, y)$$

其中，图像的照射分量通常由慢的空间变化来表征，而反射分量往往引起突变，特别是在不同物体的连接部分。这些特性导致图像取对数后的傅里叶变换的低频成分与照射想联系，而高频成分与反射相联系，这便是我们设计同态滤波器的基础。

我们定义

$$z(x, y) = \ln i(x, y) + \ln r(x, y)$$

则有

$$\Im\{z(x, y)\} = \Im\{\ln i(x, y)\} + \Im\{\ln r(x, y)\}$$

或

$$Z(u, v) = F_i(u, v) + F_r(u, v)$$

我们用同态滤波器  $H(u, v)$  对  $Z(u, v)$  进行滤波，故有

$$S(u, v) = H(u, v)Z(u, v) = H(u, v)F_i(u, v) + H(u, v)F_r(u, v)$$

在空间域中，滤波后的图像是

$$s(x, y) = \Im^{-1}\{H(u, v)F_i(u, v)\} + \Im^{-1}\{H(u, v)F_r(u, v)\}$$

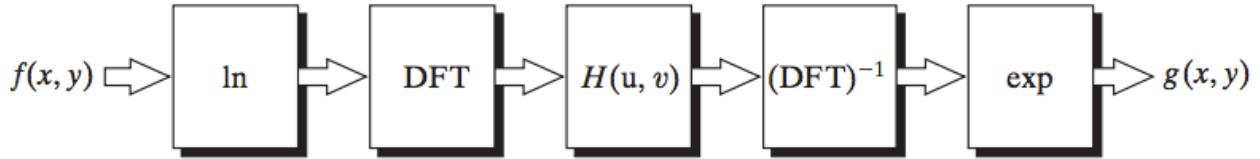
$$= i'(x, y) + r'(x, y)$$

最后，因为 $z(x, y)$ 是通过取输入图像的自然对数形成的，我们通过取滤波后的结果的指数这一反处理来形成输出图像。

$$g(x, y) = e^{s(x, y)} = e^{i'(x, y)} e^{r'(x, y)} = i_0(x, y) r_0(x, y)$$

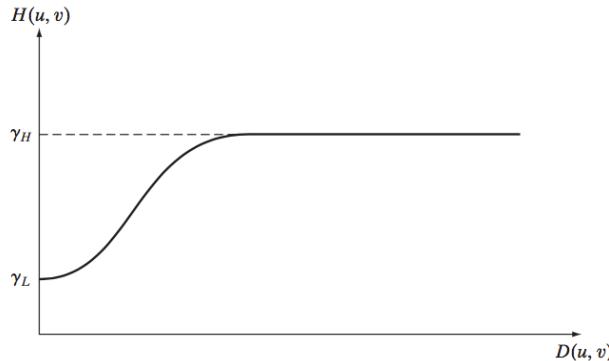
其中 $i_0(x, y)$ 和 $r_0(x, y)$ 是输出（处理后）图像的照射和反射分量。

综上所述，同态滤波方法的流程步骤如下图所示。



同态滤波方法的关键在于照射分量和反射分量的分离，其实现形式为 $Z(u, v) = F_i(u, v) + F_r(u, v)$ 。如式 $S(u, v) = H(u, v)Z(u, v) = H(u, v)F_i(u, v) + H(u, v)F_r(u, v)$ 指出的那样，同态滤波函数 $H(u, v)$ 可分别对这些分量进行操作。

同态滤波器需要制定一个滤波器函数，它可用不同的可控方法影响傅里叶变换的低频和高频分量。这个滤波器函数的剖面图如下图所示。



它可用形式稍微变换一下的高斯高通滤波器来得到。

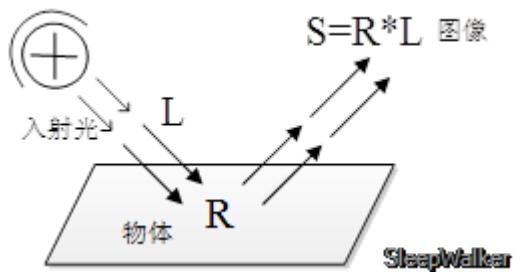
$$H(u, v) = (\gamma_H - \gamma_L) \left[ 1 - e^{-c[D^2(u, v)/D_0^2]} \right] + \gamma_L$$

#### 4.2.3.1.2 Retinex 图像增强算法<sup>[3]</sup>

Retinex 理论始于 Land 和 McCann 于 20 世纪 60 年代作出的一系列贡献，其基本思想是人感知到某点的颜色和亮度并不仅仅取决于该点进入人眼的绝对光线，还和其周围的颜色和亮度有关。Retinex 这个词是由视网膜(Retina)和大脑皮层(Cortex)两个词组合构成的。Land 之所以设计这个词，是为了表明他不清楚视觉系统的特性究竟取决于此两个生理结构中的哪一个，抑或是与两者都有关系。

Retinex 理论的基本内容是物体的颜色是由物体对长波（红）、中波（绿）和短波（蓝）光线的反射能力决定的，而不是由反射光强度的绝对值决定的；物体的色彩不受光照非均性

的影响，具有一致性，即 Retinex 理论是以色感一致性（颜色恒常性）为基础的。如下图所示，观察者所看到的物体的图像  $S$  是由物体表面对入射光  $L$  反射得到的，反射率  $R$  由物体本身决定，不受入射光  $L$  变化。



#### 4.2.3.2 算法设计

可以发现，Retinex 图像增强算法原理与之前的同态滤波原理极具相似性，但是在以下滤波过程的处理方式上存在差异。以下为算法设计。

(1) 将图像变换到对数域，即：

$$\log R(x, y) = \log \frac{S(x, y)}{L(x, y)} = \log S(x, y) - \log L(x, y)$$

(2) 将  $L(x, y)$  亮度图像估计为空间平滑的图像，这可以通过以下低通滤波过程得到：

$$L(x, y) = F(x, y) * S(x, y)$$

其中  $F(x, y) = \lambda e^{-\frac{(x^2+y^2)}{c^2}}$ ，相当于高斯滤波。

(3) 综合以上两式可得恢复  $R(x, y)$  的公式：

$$\log R(x, y) = \log S(x, y) - \log [F(x, y) * S(x, y)]$$

(4) 对  $R(x, y)$  进行直方图归一化。

#### 4.2.3.3 处理结果

经过 Retinex 图像增强算法处理后，结果对比图如下所示。



原始图片

处理后效果

可以看出，对于近处的建筑物，雾气的影响基本上被完全去除，颜色基本上接近于日常生活中实际建筑物的颜色，且对于图片中原本难以辨别的部分（如“阳光 100”标牌）做到了较为完整的恢复。对于远处的建筑物，该方法也能够去除相对更多的雾气影响，还原效果大大好于前两种算法。

但是，由于滤波算法的本身性质，在图片中会存在一些周期性噪声，这在该图上部体现得较为明显。

#### 4.2.4 方法 4：对比度调节

##### 4.2.4.1 算法原理

该方法直接使用 Matlab 中自带函数 imadjust，将一定灰度范围的像素点映射到另一输出灰度范围。

##### 4.2.4.2 算法设计

直接使用 Matlab 中自带函数 imadjust 函数

```
J = imadjust(I,[low_in; high_in],[low_out; high_out])
```

将图像 I 中的亮度值映射到 J 中的新值，即将 low\_in 至 high\_in 之间的值映射到 low\_out 至 high\_out 之间的值。low\_in 以下与 high\_in 以上的值被剪切掉了，也就是说，low\_in 以下的值映射到 low\_out，high\_in 以上的值映射到 high\_out。它们都可以使用空的矩阵[]，默认值是[0 1]。

在经过多次试验后，将各个值设置如下：

```
low_in = 0.3  
high_in = 0.95  
low_out = 0  
high_out = 0.8
```

##### 4.2.4.3 处理结果

经过对比度调节算法处理后，结果对比图如下所示。



原始图片



处理后效果

可以看出，由于对比度调节仅仅是进行简单灰度拉伸变换调整，因此效果较为平庸，对于雾气有一定程度的去除，但并不是特别明显。

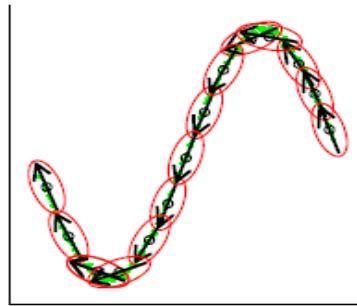
注：以上4种处理方法的实现以了解原理并自行编写代码为主，而以下2种处理方法的实现以查阅相关论文并参考已有代码为主，其中使用他人代码的部分将在正文与参考文献中加以注释，特此说明。

#### 4.2.5 方法5：引导滤波

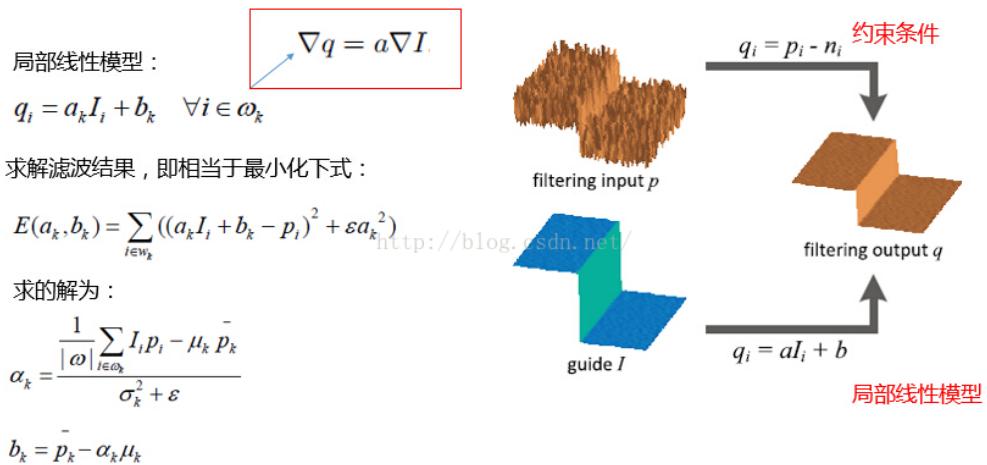
##### 4.2.5.1 算法原理<sup>[5]</sup>

引导滤波是由何恺明等人于2010年提出，其详细阐述在参考资料[5]所示的论文“Guided Image Filtering”中得以详细阐述。以下为该算法的简单原理。

该算法本质上具有  $O(n)$  复杂度，相对于双边滤波有更好的边缘保持特性，且不会出现梯度反转现象。在不同引导图像的引导下，可广泛应用于降噪、去雾、高动态范围压缩等。



该模型认为，某函数上一点与其邻近部分的点成线性关系，一个复杂的函数就可以用很多局部的线性函数来表示，当需要求该函数上某一点的值时，只需计算所有包含该点的线性函数的值并做平均即可。这便是所谓“局部线性模型”，呈现如下：



##### 4.2.5.2 算法设计<sup>[6]</sup>

下图为引导滤波算法的伪代码实现过程

**Input:** filtering input image  $p$ , guidance image  $I$ , radius  $r$ , regularization  $\epsilon$   
**Output:** filtering output  $q$ .

```

1: meanI = f_mean(I)
   meanp = f_mean(p)
   corrI = f_mean(I.*I)
   corrIp = f_mean(I.*p)
2: varI = corrI - meanI.*meanI
   covIp = corrIp - meanI.*meanp
3: a = covIp./(varI + ε)
   b = meanp - a.*meanI
4: meana = f_mean(a)
   meanb = f_mean(b)
5: q = meana.*I + meanb

```

其中,  $cov_{Ip}$  为求得解  $a_k$  的分子, 根据 “局部线性模型” 计算公式,  $I_p$  在窗口  $\omega_k$  中的和, 再除以窗口中像素的个数, 刚好就是盒式滤波, 因此我们可以将输入的引导图像  $I$  和滤波图像  $p$  相乘, 并对相乘后的图像做盒式滤波, 即得第一项的结果  $corr_{Ip}$ 。后面的  $\mu_k$  和  $\bar{p}_k$  分别为引导图像  $I$  和滤波图像  $p$  在窗口  $\omega_k$  中均值, 因此分别对  $I$  和  $p$  进行盒式滤波, 再将盒式滤波之后的结果相乘即可。实际上,  $a_k$  的分子就是  $I_p$  在窗口  $\omega_k$  中的协方差。

$var_l + \epsilon$  为求的解  $a_k$  的分母。根据 “局部线性模型” 计算公式,  $\sigma_k^2$  是引导图  $I$  在窗口  $\omega_k$  中的方差。

最终结果为  $\bar{a}_k I_i + \bar{b}_k$ , 即最终滤波后的输出。

程序编写过程参考了来自参考资料[7]的代码, 代码注释中标注的原理式编号均来自于参考资料[5] (何恺明论文 “Guided Image Filtering”) 的公式编号。

#### 4.2.5.3 处理结果

经过引导滤波算法处理后, 结果对比图如下所示。



可以看出, 引导滤波算法的处理取得了相当不错的效果。这体现在以下几个方面: 1、对于近处的建筑物, 雾气的影响基本上被完全去除, 颜色基本上接近于日常生活中实际建筑物的颜色, 且对于图片中原本难以辨别的部分 (如 “阳光 100” 标牌) 做到了较为完整的恢复。2、对于远处的建筑物, 该方法的还原效果也很强, 雾气的影响明显减弱了。3、图片过渡十分自然, 颜色与现实生活中的建筑物十分接近, 且周期性噪声的影响较少。可以认为是一种较为完美的去雾算法。

## 4.2.6 方法 6：暗原色先验滤波

### 4.2.6.1 算法原理<sup>[8]</sup>

暗原色先验来自对户外无雾图像数据库的统计规律，它基于经观察得到的这么一个关键事实——绝大多数的户外无雾图像的每个局部区域都存在某些至少一个颜色通道的强度值很低的像素。利用这个先验建立的去雾模型，可以直接估算雾的浓度并且复原得到高质量的去除雾干扰的图像。即，在不包括天空的绝大部分局部区域，总会存在一些我们称之为“dark pixels”的像素，至少有一个颜色通道具备很低的强度值。在被雾干扰的图像里，这些暗像素的强度值会被大气中的白光成分所充斥而变得较高。

因此，这些暗像素能够直接用来评估雾光的透射信息。结合一个已有的雾成像模型和插值法抠图修复，我们可以得到高质量的去雾图像和很好的深度图。

在图像处理中，McCarney 的大气散射模型被得以广泛应用，其模型原理式为：

$$I(x) = t(x)J(x) + (1-t(x))A$$

其中  $I(x)$  为输入图像（观测到的）光强度， $t(x)$  指光线透射率， $A$  是大气光成分， $J$  是无雾时景物的光线强度。图像去雾目标就是由已知的  $I(x)$  求得未知参数  $J, A, t(x)$ 。由于方程的个数少于未知量的个数，需增加约束条件来求解。暗原色先验即是其中的一种约束。

对户外无雾图像  $J$  进行分块，对每个像素块定义暗原色如下：

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

式中： $\Omega(x)$  是以  $x$  为中心的正方形邻域， $J_c$  为  $J$  三原色的一个通道， $J^{dark}(x)$  即为图像  $J$  在这个邻域的暗原色。观察统计表明  $J^{dark}$  趋于零。在带雾图像中，这些暗原色的值会升高。

### 4.2.6.2 算法设计

暗原色先验算法的核心便是求解上述模型。为了求解以上模型，首先对 McCarney 的大气散射模型两端在每个像素块求取最小值：

$$\min(\min_{y \in \Omega(x)} (\frac{I_c(y)}{A_c})) = t(x) \min_c (\min_{y \in \Omega(x)} (\frac{J_c(y)}{A_c})) + (1-t(x))$$

根据暗原色先验无雾图像的暗原色趋于零，故由上式可得：

$$t(x) = 1 - \min_c (\min_{y \in \Omega(x)} (\frac{I_c(y)}{A_c}))$$

而右式第二项即为带雾图像在  $x$  邻域的暗原色值，因此局部区域的  $t$  值可以求得，从而可以得到整幅图的透射率  $t(x)$ 。但当  $t(x)$  趋于零时，图像趋于包含噪音，因此常常需要设置一个透射率下限  $t_0$ 。

最后需要估计大气光，在一般的单一图像去雾中，通常整幅图最大强度的像素作为大气光的估测。

最后得到J的求解公式为：

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

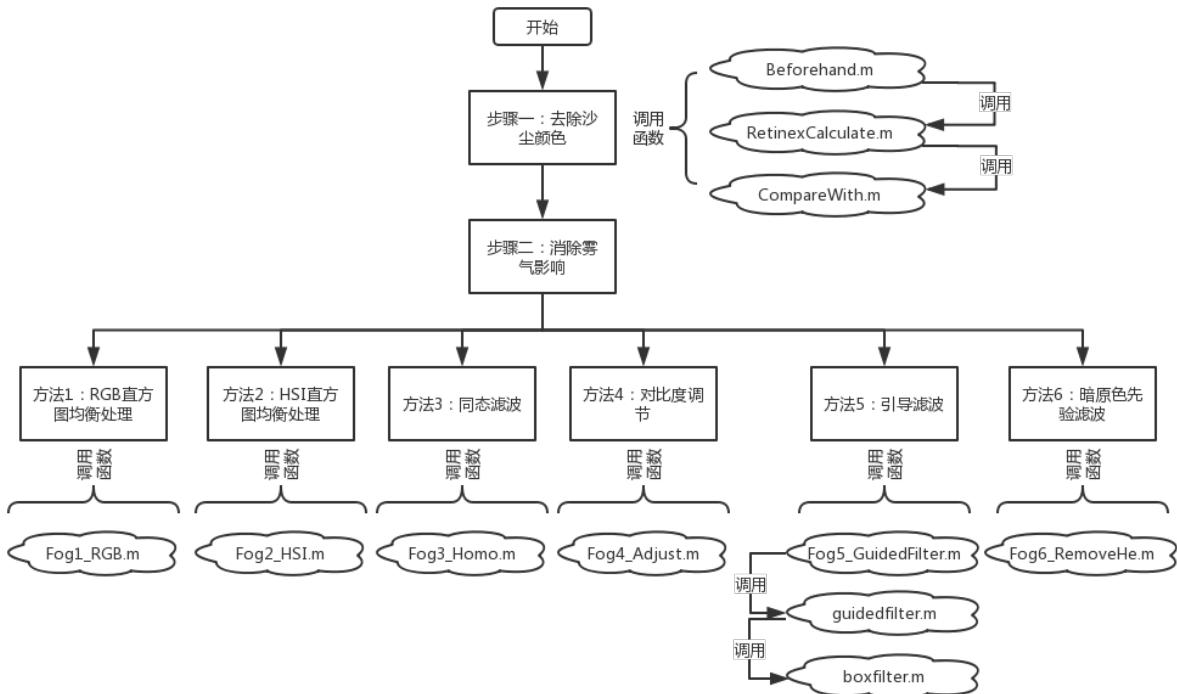
#### 4.2.6.3 处理结果

经过暗原色先验滤波算法处理后，结果对比图如下所示。



可以看出，暗原色先验滤波的处理取得的效果也较为平庸，对于近处的建筑物的处理结果并没有使用之前算法所得结果清晰，但对于远处建筑物的去雾效果也还是可以接受的。其原因可能正如在何恺明先生的论文中阐述的那样，“我们的实现也有其限制因素。当取景对象在较大范围内和天空接近并且没有阴影覆盖的时候，暗原色的猜想将不成立。”

## 五、程序结构



## 六、输出结果与讨论

### 6.1 原图对比测试



原始图片 1

RGB 直方图均衡处理	HSI 直方图均衡处理	同态滤波
对比度调节	引导滤波	暗原色先验滤波

通过对比可以看出，对于近处的建筑物，RGB 直方图均衡处理、HSI 直方图均衡处理、同态滤波、引导滤波算法均能够进行较好的增强和还原；对于远处的建筑物，同态滤波、引导滤波和暗原色先验滤波算法的效果较好。而在这些算法中，引导滤波所得到的效果是最为自然的，且没有较多周期性噪声。

### 6.2 算法鲁棒性测试

为了验证各个算法的鲁棒性，本人又使用了 4 张相似的风景图片和 5 张其他风格的图片进行鲁棒性测试，测试结果如下。

#### 6.2.1 使用相似风景图测试（4 张）

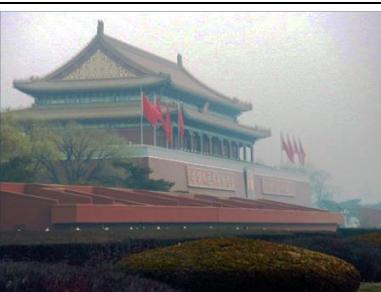
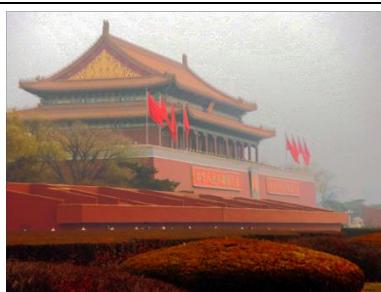
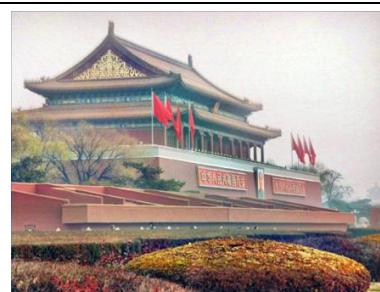
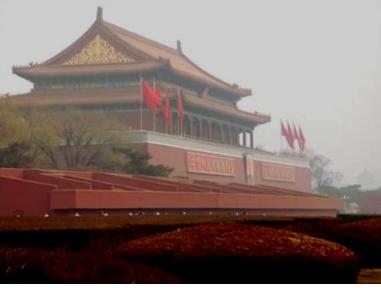
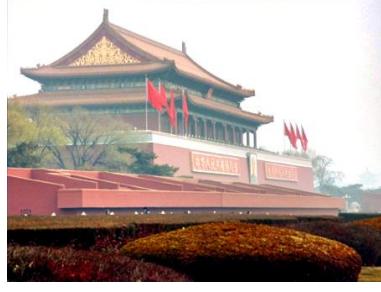
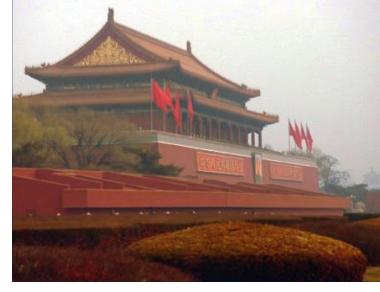


原始图片 2

RGB 直方图均衡处理	HSI 直方图均衡处理	同态滤波
		
对比度调节	引导滤波	暗原色先验滤波
		



原始图片 3

RGB 直方图均衡处理	HSI 直方图均衡处理	同态滤波
		
对比度调节	引导滤波	暗原色先验滤波
		



原始图片 4

RGB 直方图均衡处理	HSI 直方图均衡处理	同态滤波
对比度调节	引导滤波	暗原色先验滤波



原始图片 5

RGB 直方图均衡处理	HSI 直方图均衡处理	同态滤波
对比度调节	引导滤波	暗原色先验滤波

以下是通过主观视觉所得到的处理结果对比表：“😊”表示令人满意，“best”表示各种方法中效果最佳的方法。

	图 2	图 3	图 4	图 5
方法 1	😊	存在颜色失真	😊	😊
方法 2	😊	😊	😊	存在颜色失真
方法 3	😊 best	😊	😊 best	😊
方法 4	去雾效果不佳	去雾效果不佳	去雾效果不佳	去雾效果不佳
方法 5	图片上部变白	😊 best	图片上部变白	😊 best
方法 6	😊	😊	😊	😊

“没有对比就没有伤害”，通过对比可以看出：方法 3、5、6 具有较强的鲁棒性。其中，方法 3（基于 Retinex 图像增强算法原理的同态滤波算法）均能够获得很好的处理效果，但在使用时也要注意周期性噪声对图片的影响。方法 6（暗原色先验滤波算法）也均能够获得较好的效果。方法 5（引导滤波算法）在部分情况下能够获得比其他算法好得多的效果，但在一些情况下会出现因为过度增强导致图片上部变白的情况。

同时我们可以发现，RGB 直方图均衡算法、HSI 直方图均衡算法有可能会出现颜色失真的情况。一方面这是三通道分别均衡必然带来的调整不平衡，另一方面为了增加饱和度，人为将 S 分量进行了 3 倍倍乘，而实际上这个数字很可能是不准确的，而且也是因图而异的。

对于 4 张图片中均表现不佳的方法 4（对比度调节），本人分析的原因是：对比度调节的输入输出上下界是因图而异的，程序中设计的参数是最适合于原始图片 1 的，但并不一定适用于其他图片，说明对比度调节方法的鲁棒性较差。

### 6.2.2 使用二次元图片测试（2 张）

完全出于个人兴趣，本人对出自动画中的两张图片进行了增强测试。由于动画图片中的色彩相比现实生活更为绚烂，因此图像增强的效果应该与现实图片存在一些不同。以下是我的测试结果。



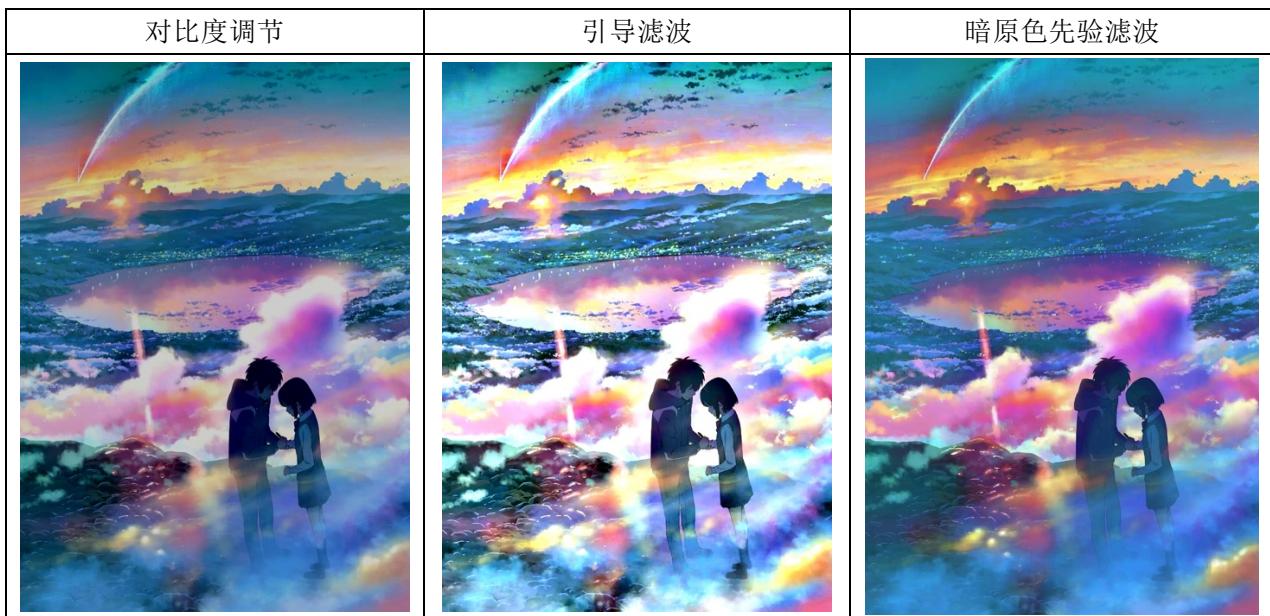
原始图片 6

RGB 直方图均衡处理	HSI 直方图均衡处理	同态滤波
对比度调节	引导滤波	暗原色先验滤波



原始图片 7

RGB 直方图均衡处理	HSI 直方图均衡处理	同态滤波



以下是通过主观视觉所得到的处理结果对比表：“😊”表示令人满意，“best”表示各种方法中效果最佳的方法。

	图 6	图 7
方法 1	😊	脸部变黑
方法 2	😊	脸部变黑
方法 3	😊 best	😊 best
方法 4	去雾效果不佳	去雾效果不佳
方法 5	图片上部变白	😊
方法 6	😊	去雾效果不佳

通过比较可以看出，由于动画图片中颜色较为丰富，且跨度较大，因此各种算法在动画图片中经常出现各种问题，例如在图 7 中出现的最常见的问题便是人物脸部变黑，这在其他正常的风景图中是不会出现的。可以发现鲁棒性最强的依旧是方法 3（基于 Retinex 图像增强算法原理的同态滤波算法）。

### 6.2.3 使用清华大霾图片测试（3 张）

两年前，来自清华大学自动化系 1 字班的摄影师宋天瑜学长拍摄了一系列夜晚的大雾霾照片。本以为这是一套很好的测试图，但经过尝试后本人发现：首先，这些照片均是拍摄在夜间的，背景颜色从原来的浅色调变为深色调，对于不少算法需要进行修正；其次，这些照片的雾霾实在是太严重了！使用人眼都无法看出原来没有雾霾时候的样子。以下是我的测试结果。



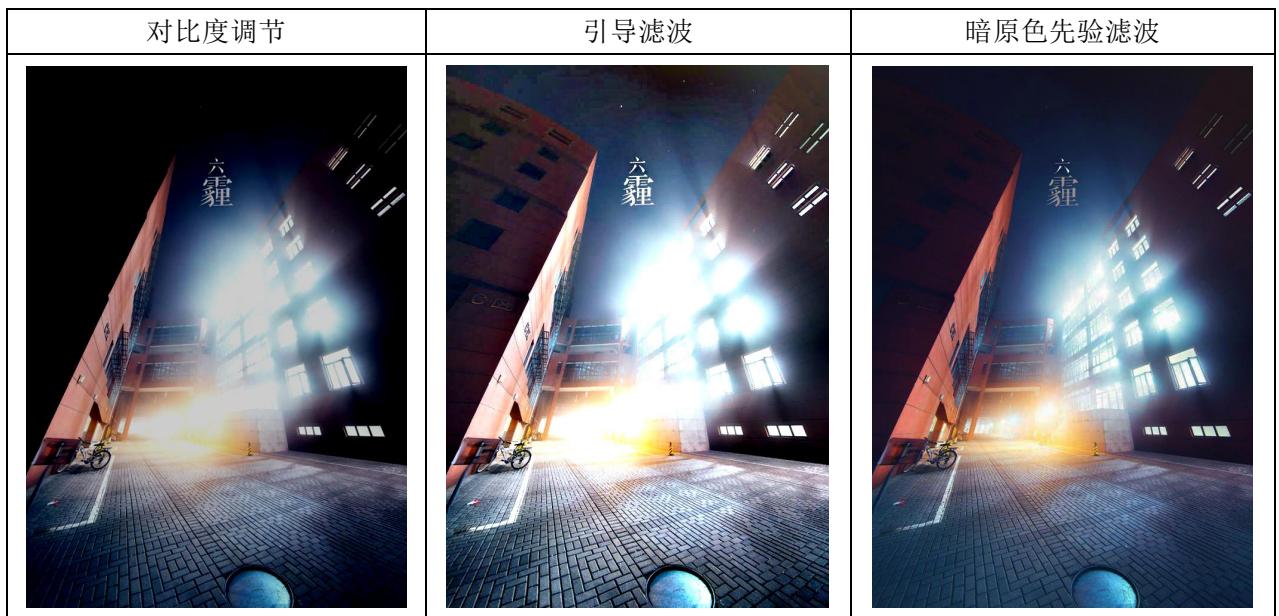
原始图片 8

RGB 直方图均衡处理	HSI 直方图均衡处理	同态滤波
对比度调节	引导滤波	暗原色先验滤波

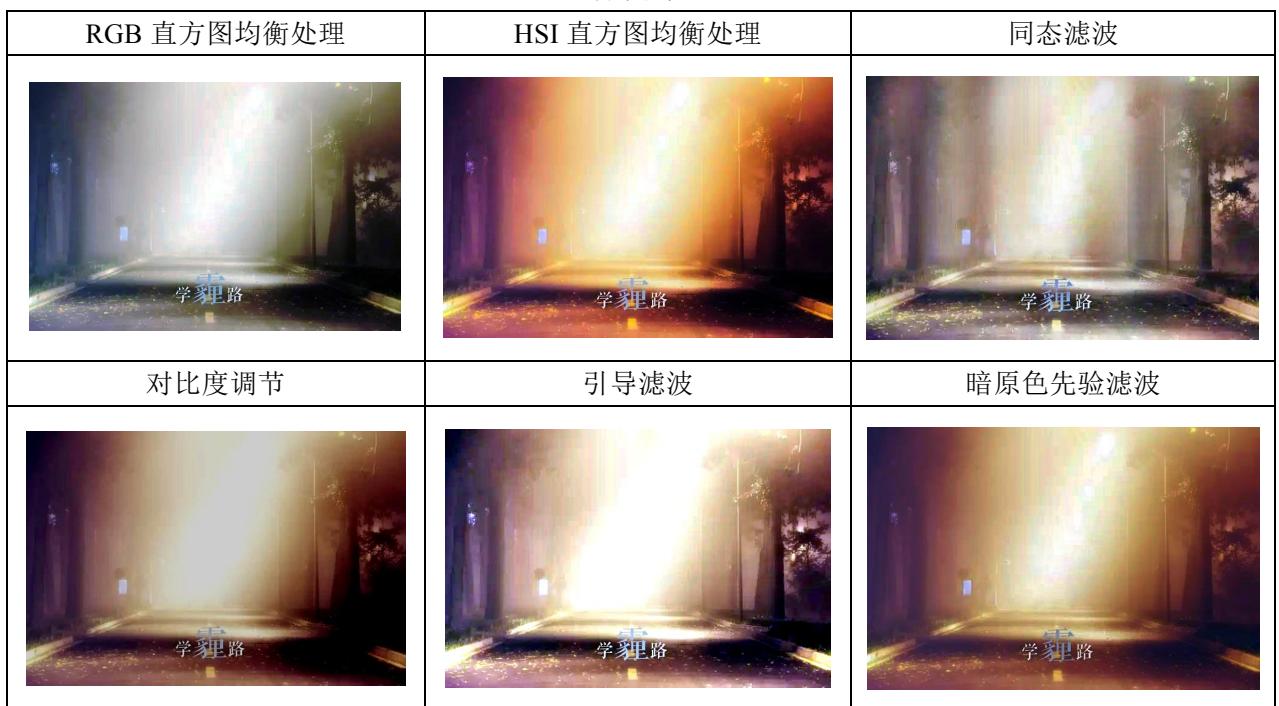


原始图片 9

RGB 直方图均衡处理	HSI 直方图均衡处理	同态滤波
对比度调节	引导滤波	暗原色先验滤波



原始图片 10



通过比较可以看出，在各种算法中，基于 Retinex 图像增强算法原理的同态滤波算法和引导滤波算法具有较强的鲁棒性，均能获得较可以接受的结果。而其余算法均有可能出现颜色失真、去雾效果不佳等问题。

## 七、心得与体会

这次作业实现过程也是一个不断“否定”的过程。在每次尝试一种方法后，我经常会认为这种方法所得到的结果“真不错”。然而通过一定的思考我又能想出新的解决方法，可能会获得更好的效果。将这些方法加以尝试后，我总能有一种获得新大陆的快感。在尝试各种方法的过程中，经常需要查阅各种文献或是前人代码，通过对文献和代码的阅读与理解，我对图像去雾的各种算法有了更加深刻的理解。

本次实验过程并不是一帆风顺的，期间不免遇到了各种各样的问题。而且，由于编程能力有限，在运行 Matlab 程序时还经常会出现各种办法。我的解决方案是将错误信息放到搜索引擎里搜索，并且输出相关参数来判断具体的错误是出在哪一个过程中，最终总能够排查出相应问题。

## 参考文献

- [1] Digital Image Processing\_3<sup>rd</sup>, Rafael C.Gonzalez
- [2] Frankle J A, Mccann J J. Method and apparatus for lightness imaging: US, US4384336[P]. 1983.
- [3] <http://www.cnblogs.com/sleepwalker/p/3676600.html>
- [4] <http://www.mamicode.com/info-detail-346279.html>
- [5] He K, Sun J, Tang X. Guided Image Filtering[M]// Computer Vision – ECCV 2010. Springer Berlin Heidelberg, 2010:1397-1409.
- [6] <http://blog.csdn.net/cyh706510441/article/details/49467191>
- [7] <http://download.csdn.net/detail/ckghostwj/7706037>
- [8] He K, Sun J, Tang X. Single Image Haze Removal Using Dark Channel Prior[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2010, 33(12):2341-2353.