数字图像处理第三次大作业实验报告

自 42 张博文 2014011455

目录

1.程序框架

我编写的去雾程序在 img_dfog.m 文件中。我一共使用了三种方法来实现去雾：带分割的直方图均衡化、暗通道去雾和 Retinex 去雾算法。

在 M 文件中 img_dfog 函数为主函数，其余函数的名字和功能分别如下表

函数名	函数调用方式		函数功能
UseHistogram	newimg=UseHistogram(oldimg)		使用直方图均衡化的方式去雾
	子函数	newimg =SquareHistogram(oldimg,m,n)	将 oldimg 均匀分成 m×n 个子图，再对每个子图进行直方图均衡化
		newimg=Histogram(oldimg)	对 oldimg 进行直方图均衡化
UseDarkCha	newimg=UseDarkCha(oldimg)		使用暗通道去雾
	子函数	newimg =minfilt2(oldimg,[m n])	对 oldimg 进行窗口大小为 m×n 的最小值滤波
		tx =guidefilter(oldimg,tx,m,eps)	根据 oldimg，对 tx 进行导向滤波
UseRetinex	newimg=UseRetinex(oldimg)		使用 Retinex 理论去雾
	子函数	newimg=Retinex(oldimg,cigma)	对 RGB 图 oldimg 的三个通道分别进行基于参数为 cigma 的高斯低通滤波的同态滤波
		newimg=Homofilter(oldimg,F)	对灰度图 oldimg 在频域与 F 相乘，进行同态滤波

函数 img_dereflect 是整个程序的入口，程序读入图片后会按照以上三种方式分别进行去雾处理，最终一同呈现。

2.去雾原理与流程

2.1 直方图均衡化

直方图反映了图像各灰度级的像素出现的概率，而直方图均衡化是变换图像的灰度直方图，使各种灰度大致有相同的点数。使用了直方图均衡化之后会从视觉上感觉图像变清晰了，因此也可以实现去雾的功能。

直方图均衡化的原理是首先求得累积直方图

$$P(i)=\sum_{0}^{i}P_i$$

其中P_i是图像各灰度出现的频率

之后将所有灰度值为 i 的像素点的灰度值改为 j，i 和 j 满足关系

$$j = P(i) \times \max(i)/N$$

其中 N 为像素总数。

由于直方图均衡化只能对一个通道的灰度级进行，所以在实际应用中，应该分别对图像的 RGB 三个通道使用直方图均衡化（或者对 HSV 的 V 通道进行，但处理题目图片的效果不是太好），最终效果如下

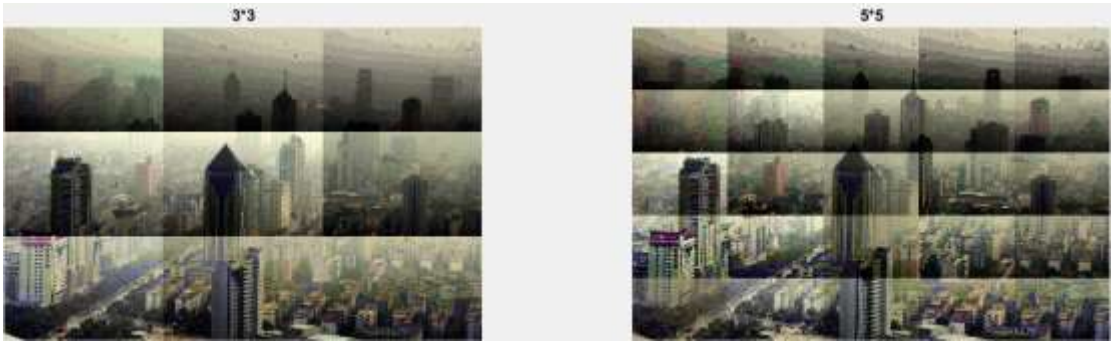
可以发现，虽然图片下方确实变的清晰了，但是图片上方很多位置仍然存在着较多的雾。这是因为直方图均衡化是对全局图像进行的，要想增强局部的对比度，应当将图像分割成子图，对每个子图分别进行直方图均衡化。

因此我将直方图均衡化的方法改进为：
①对图像进行均匀分割
②对每个子图分别进行直方图均衡化
③去除子图之间的块状效应

2.1.1 子图直方图均衡化

基本的直方图均衡化函数在 Histogram 函数中；而将图像分割成子图、再对每个子图分别进行直方图均衡化的功能我写在了 SquareHistogram 函数中。具体流程为：根据输入的横向分割数(xnum)，纵向分割数(ynum)，将图像的宽度和高度均匀分割，遍历每一块子图，对其调用 Histogram 函数进行分割。

分别进行 3×3 分割和 5×5 分割再均衡化的结果如下



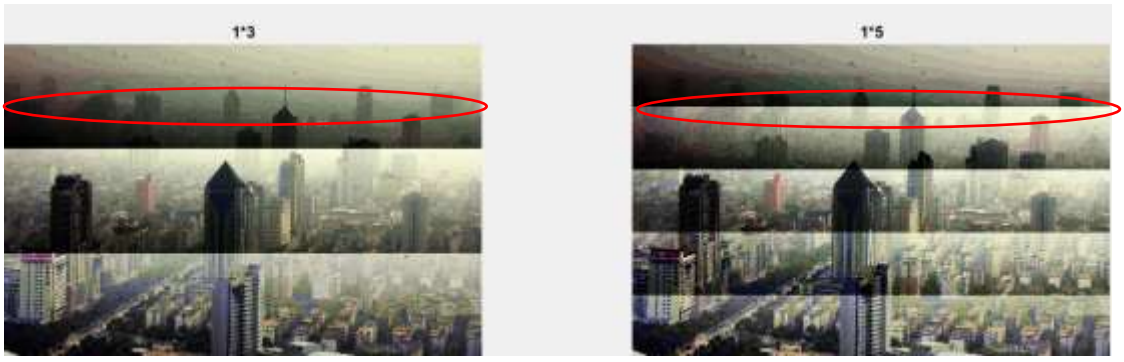
可以看到随着分割数的增大，局部的图像确实变的更加清晰，比如 5×5 的该部分和全局均衡化的对比如下。左图为 5×5 分割，右图为全局直方图均衡化。

2.1.2 消除块状效应

虽然将图像分割为子图后图像变的更清晰，但是子图间的块状效应也更加明显，为了消除块状效应，我采取了以下方法。

分割要分成纵向和横向分割分别进行，这里以纵向分割为例说明。

首先将图像按照 1×3 和 1×5 分别分割，结果如下



之后，以 1×5 图像为底，将子图分界线区域的像素用 1×3 图像中相应位置的像素替换，并且加入和渐变，最终效果图如下

同理，再横向分割时则是先将图像分割为 3×1 和 5×1，之后再按照上述方法进行去除块状效应，最后将横向分割得到的图像、纵向分割得到的图像、全局直方图均衡化得到的图像按权重叠加，则得到最终的结果图：


子图直方图均衡化　　全局直方图均衡化

可以看到子图直方图均衡化得到的结果还是要比全局直方图均衡化得到的结果要清晰的。

2.2 暗通道算法

在计算机视觉和计算机图形学中，下式描述雾图的方程被广泛使用

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

其中J(x)是原始无雾的图像，I(x)是现有的有雾图像，t(x)是图像各个位置的透射率，A是全球大气光成分，三个通道均为常数。

由一些理论推导（见参考文献[1]）最终可得

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

其中t_0为阈值，使t(x)不至于太小引起除法出现错误，在本程序中t_0取0.1。此外当前的未知量是A和t(x)，这时需要使用到暗通道的概念。

暗通道的定义如下：

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

J^c是彩色图像的每个通道，Ω(x)是以像素x为中心的一个窗口。这样得到的就是暗通道图像。

在得到了暗通道图像后，找到具有前0.1%最大的像素值的点坐标，再在对应的原图中查询对应的所有位置中的最大值，作为该通道的A值。

而对于t(x)，可通下式获得

$$\tilde{t}(x) = 1 - \omega \min_{y \in \Omega(x)} \left[\min_c \frac{I^c(y)}{A^c} \right]$$

其中ω为为了使图像看上去更真实而保留一定雾的因子，在本程序中取0.95

在获得了t(x)后可以使用原图像对t(x)进行导向滤波，以使得物体边缘过渡更自然，最终在获得了各分量后可以就可以对原图进行去雾。

2.2.1 求取各参数值

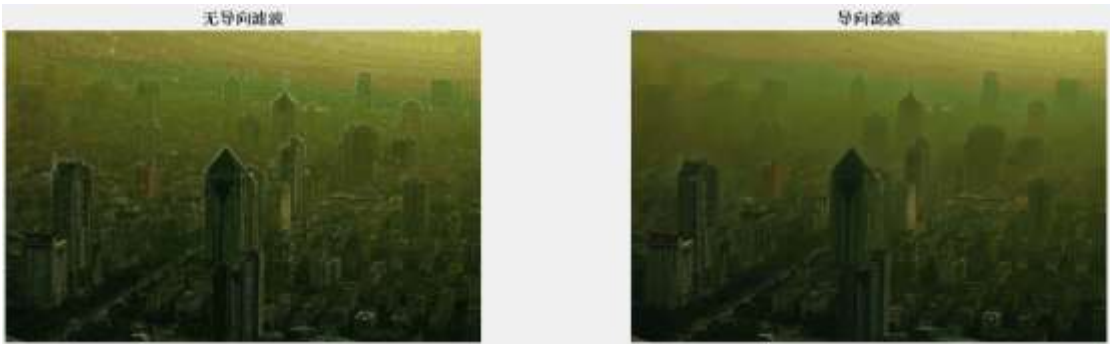
首先是求取暗通道图 Jdark(x)。

求取暗通道图像分两步。首先求出图像像素点处 RGB 通道中的最小值，之后对图像的各个像素点进行最小值滤波。在本程序中窗口大小为 15×15。

基于窗口的最小值滤波可以使用 ordfilt2 函数，但当窗口较大时运行速度会较慢；所以本程序使用了 matlab 官方提供的脚本 minfilt2.m 实现最小值滤波的功能。

在求得了暗通道图 Jdark(x)后，可以依次求出 A 和 t(x)的值。其中 A 的值是通过统计 Jdark(x)中像素值较大的位置，再在原图这些位置中找到最大的值。t(x)则是用原图三个通道分别除以三个通道的 A 值，再在每个像素点取三个通道中的最小值，之后仍然通过一个窗口大小为 15×15 的最小值滤波。

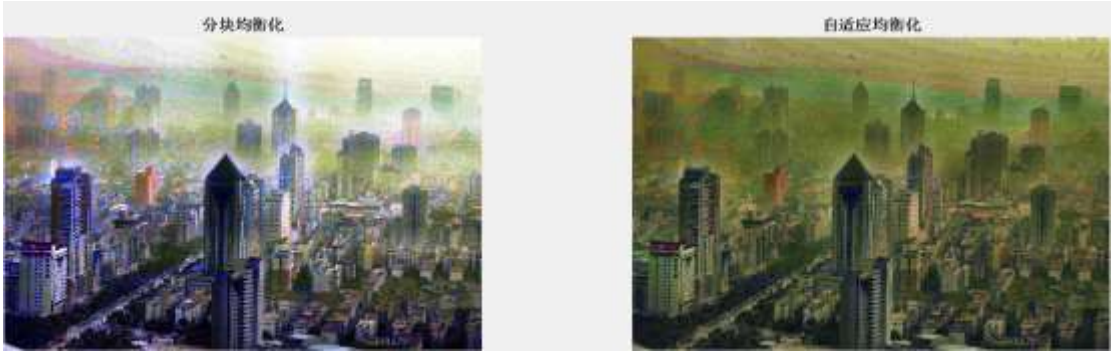
有了 A 和 t(x)的值，就可以对原图像进行恢复了，但是根据参考资料中提到，此种情况是假定在窗体内 t(x)的值是固定的，所以最终得到的结果在物体边缘处仍然会有雾，为了解决这个办法可以用官方的脚本 guidedfilter.m 进行导向滤波。最终结果如下：



2.2.3 调整色调

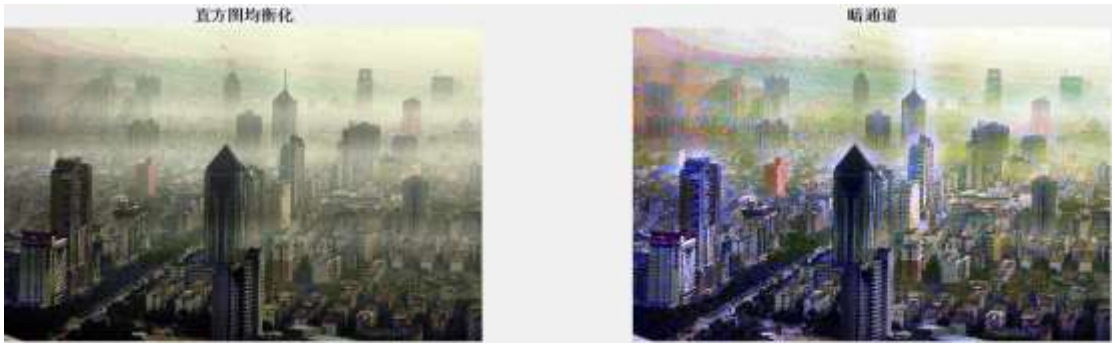
由于某种原因，最终得到的图像整体会发黄（其实是整体亮度较低），为了解决这个问题最简单的方式就是对 RGB 三个通道分别进行直方图均衡化来调整图片整体色调。

我使用了 2.1 中我编写的分块均衡化和 matlab 自带的自适应均衡化 adapthisteq 函数，分别处理结果如下。



可以发现分块均衡化的结果要优于自适应均衡化得到的结果。当然，如果图像的亮度在按通道处理的过程中没有受到太多的影响，则可以省略最后调整色调的这一过程。

另外，只有直方图均衡化和先进行暗通道滤波，再进行直方图均衡化的对比如下：



2.3 Retinex 理论

Retinex 的原理和同态滤波的原理有些类似，即认为观测得到的图像 S，是由入射光性质 L 和物体反射性质 R 相乘得到的，即

$$S(x,y)=R(x,y)\times L(x,y)$$

因此去除了入射光 R 即可以得到本来图像的样子。为了去除入射光 R，需要先对图像取对数，即

$$\log\big(S(x,y)\big)=\log\big(R(x,y)\big)+\log(L(x,y))$$

而入射光动态变化较小，所以对取了对数的图像在频域中进行高斯低通滤波，就会得到入射光的部分即雾的对数部分，之后用原图取对数减去雾的对数，即得图像本身的对数部分。即

$$\log\big(L'(x,y)\big)=ifft2(fft2(\log(S(x,y)))\times H(u,v))$$

$$\log\big(R(x,y)\big)=\log\big(S(x,y)\big)-\log(L'(x,y))$$

最后将得到的$\log\big(R(x,y)\big)$作 exp 运算，再将其动态范围拉伸（这里使用了自适应直方图均衡化函数 adapthisteq），即得到最终的结果。

2.3.1 高斯低通函数的确定

在使用 Retinex 算法时，需要用高斯函数进行滤波，根据资料指出，能够满足需求的高斯函数应有如下形式

$$H(x,y)=Ce^{-\frac{x^2+y^2}{2\sigma^2}}$$

其中 C 是归一化系数，即使得的 H(x,y)在二维平面上的积分结果为 1。实际计算时只需在构建出右边的结果后将结果除以各个位置数值之和。

另外，为了满足不同尺度的要求，在程序中 σ 分别取了 128，256，512，最终对三个结果进行平均。

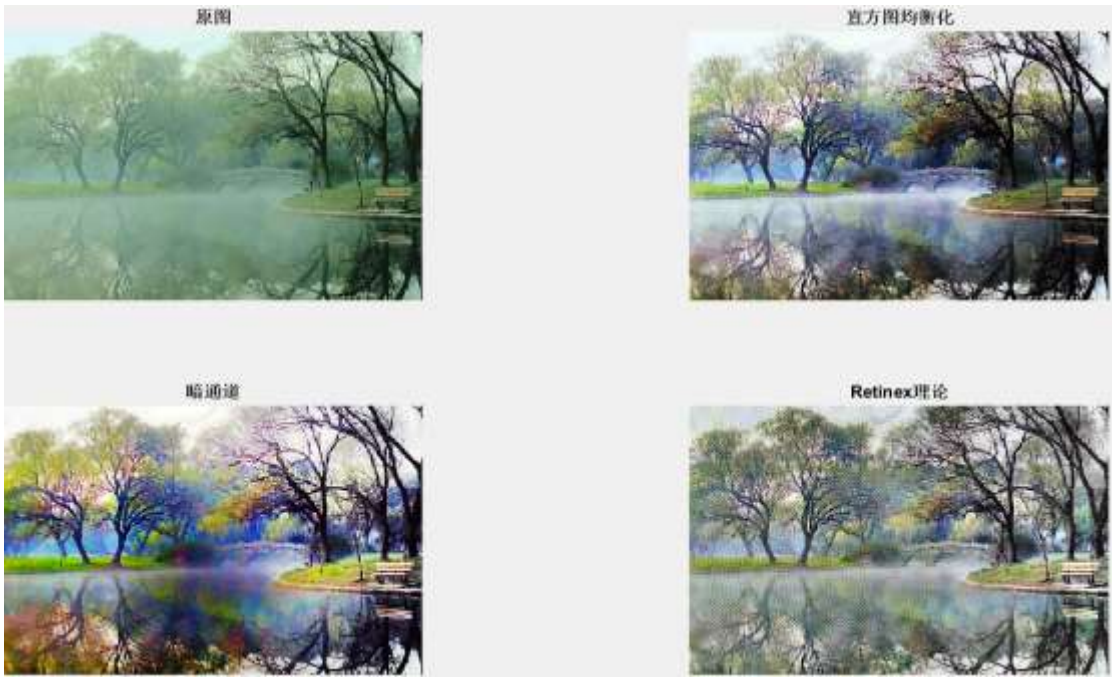
最终得到的结果和只用自适应直方图均衡化得到的结果如下

3.程序结果与性能

3.1 效果图与鲁棒性

最终得到的效果图如下

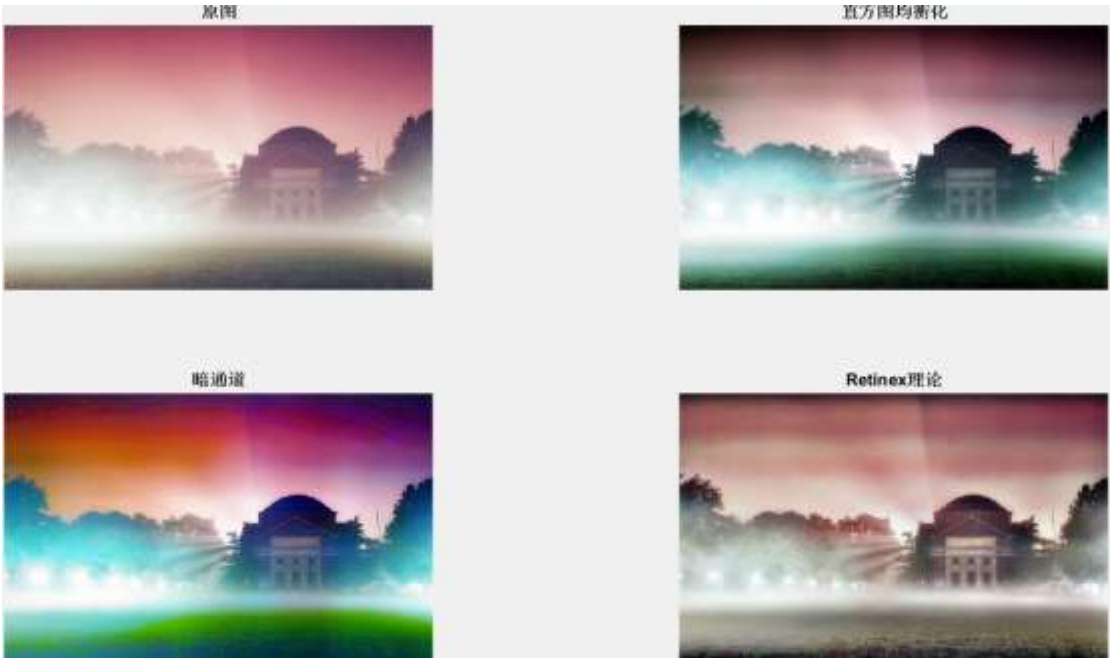
为验证程序鲁棒性，我添加了八张带雾的图片，最终去雾效果如下：


原图
直方图均衡化
暗通道
Retinex理论

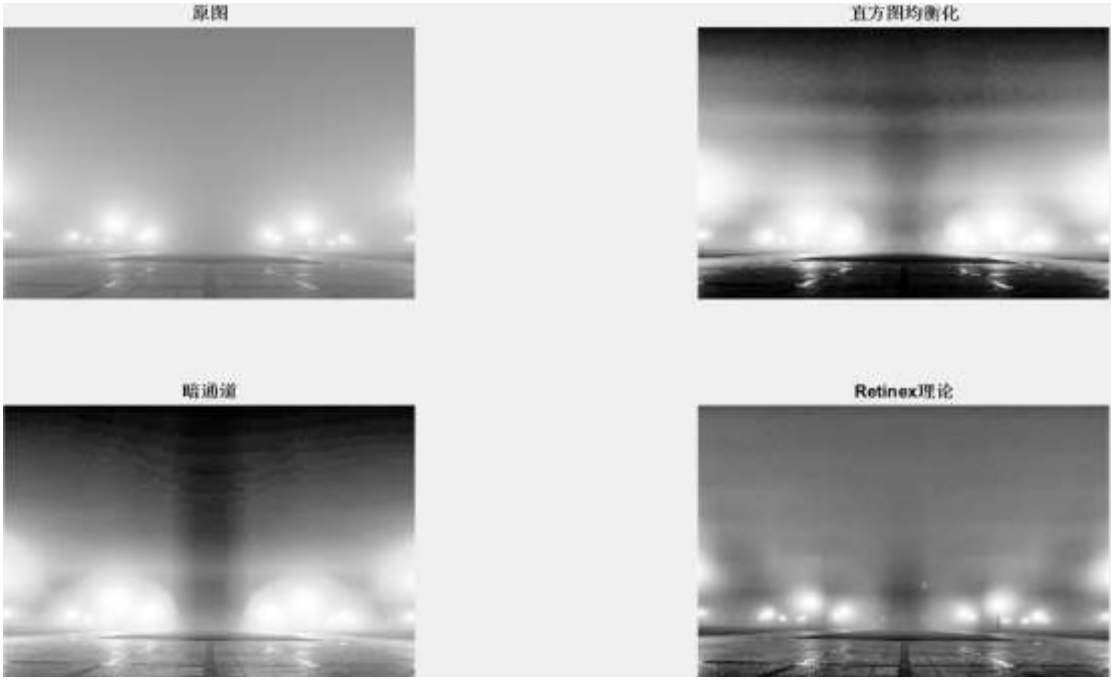

原图
直方图均衡化
暗通道
Retinex理论


原图
直方图均衡化
暗通道
Retinex理论


原图
直方图均衡化
暗通道
Retinex理论

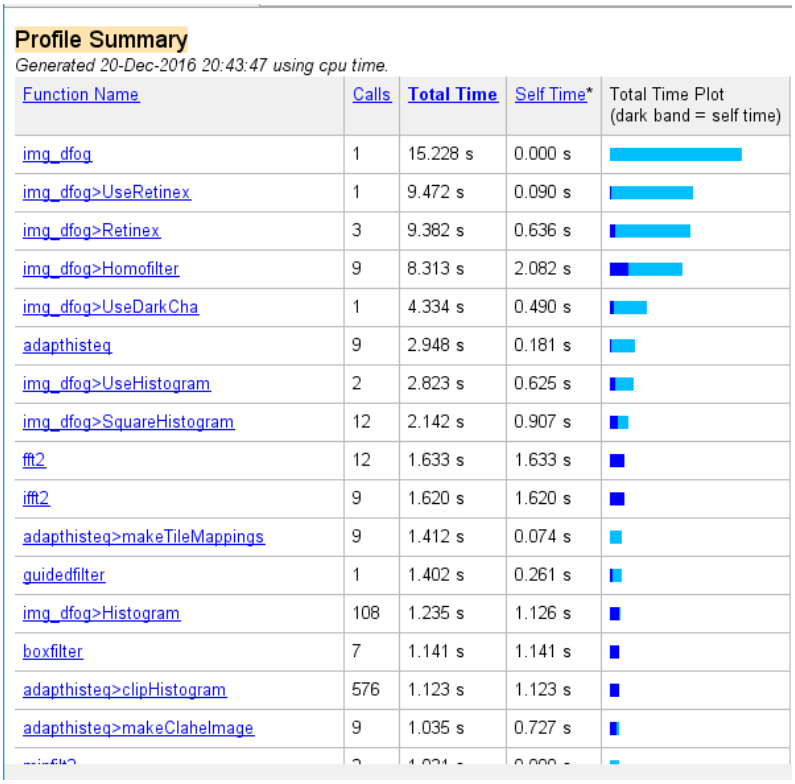

原图
直方图均衡化
暗通道
Retinex理论


原图
直方图均衡化
暗通道
Retinex理论



3.2 算法复杂度

对作业中所给的 2443×1559 的图片运行程序耗时如下：



Profile Summary

Generated 20-Dec-2016 20:43:47 using cpu time.

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
img_dfog	1	15.228 s	0.000 s	
img_dfog>UseRetinex	1	9.472 s	0.090 s	
img_dfog>Retinex	3	9.382 s	0.636 s	
img_dfog>Homofilter	9	8.313 s	2.082 s	
img_dfog>UseDarkCha	1	4.334 s	0.490 s	
adapthisteq	9	2.948 s	0.181 s	
img_dfog>UseHistogram	2	2.823 s	0.625 s	
img_dfog>SquareHistogram	12	2.142 s	0.907 s	
fft2	12	1.633 s	1.633 s	
ifft2	9	1.620 s	1.620 s	
adapthisteq>makeTileMappings	9	1.412 s	0.074 s	
guidedfilter	1	1.402 s	0.261 s	
img_dfog>Histogram	108	1.235 s	1.126 s	
boxfilter	7	1.141 s	1.141 s	
adapthisteq>clipHistogram	576	1.123 s	1.123 s	
adapthisteq>makeClaheImage	9	1.035 s	0.727 s	

3.3 算法比较

根据上述测试集中图片可以发现，使用暗通道的方法去雾效果从视觉角度来说是最好的，大多数情况会比另外两种方法要清晰，当然在整个环境都比较暗的时候效果可能会差一些。但是由于原图发黄的问题在暗通道处理后又使用了直方图均衡化的方式处理，导致色彩和原图可能会产生较大差异。

基于 Retinex 原理的去雾方法在一些情况下效果非常好，但在一些情况下似乎去雾并不是很明显，但是此方法对原图的色彩保持是三种算法中最好的。

将图像分割再进行直方图均衡化的算法则比较稳定，在各个测试图片中都有比较稳定的去雾效果。

参考文献

[1]CSDN 博客 “暗通道优先的图像去雾算法“
http://blog.csdn.net/baimafujinji/article/details/27206237
[2] CSDN 博客“Retinex 算法详解”
http://blog.csdn.net/carson2005/article/details/9502053