基于暗通道先验算法的图像去雾

42204341-张悦鑫

1 引言

随着工业的发展，环境污染越来越严重，雾霾等空气污染现象时不时发生在人们生活中。雾霾不仅影响了人们的日常出行，也影响了人门获取室外图像的清晰度。在有雾的天气采集图像时，自然光线会受到大气中悬浮颗粒的散射，导致最终采集到的图像观感不清晰，亮度、对比度和颜色等特征都有所锐减，无法达到人们日常生活和科研时的图像质量需求。因此，如何实现高质量的图像去雾有非常重要的现实意义。图1为有雾图像成因。

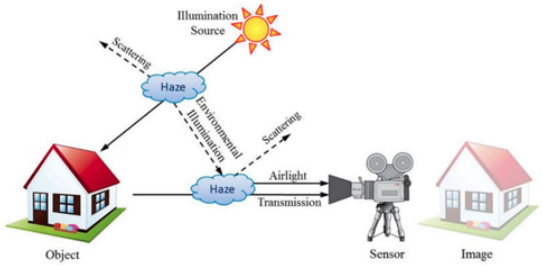

Figure 1: 有雾图像成因

2 算法原理

2.1 图像退化模型

在基于图像复原的去雾算法研究中，图像退化模型为：

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

式中：x代表图像中的单个像素；I (x)代表有雾图像；J(x)代表清晰无雾图像；A为大气光值；t(x)为透射率或者介质传输图。这是一种被广泛应用的物理模型，它解释了有雾图像形成的物理本质，为图像去雾领域的研究学者提供了一个很好的方向。图像去雾的目标就是通过一定的手段获得大气光值A和透射率t(x)，然后代入图像退化模型即可得到无雾图像J(x)：

$$J(x) = [I(x) - A]/t(x) + A \tag{2}$$

2.2 暗通道先验理论

基于统计大量清晰图像得到的暗通道先验理论是指大部分不含天空的优质图像的所有像素在RGB这3个通道中最少存在一个颜色通道灰度值相当低以至趋近于0,也就是在一定的微小区域里最小辐射强度值极低。对一幅图像J，其暗通道的定义为：

$$J^{\text{dark}}(x) = \min_{c \in \{R,G,B\}} \left(\min_{y \in \Omega(x)} J^c(y) \right), \tag{3}$$

其中，J^c(y)为J(y)的某一RGB颜色通道，$\Omega(x)$表示以像素点x为中心的方形区域。

暗通道先验理论指出：

$$J^{\text{dark}}(x) \to 0 \tag{4}$$

实际生活中造成暗原色中低通道值主要有三个因素：(a)汽车、建筑物和城市中玻璃窗户的阴影，或者是树叶、树与岩石等自然景观的投影；(b)色彩鲜艳的物体或表面，在RGB的三个通道中有些通道的值很低（比如绿色的草地、树、植物，红色或黄色的花朵、叶子，或者蓝色的水面）；(c)颜色较暗的物体或者表面，例如灰暗色的树干和石头。总之，自然景物中到处都是阴影或者彩色，这些景物的图像的暗原色总是很灰暗的。

有雾图像与无雾图像的暗通道图像对比如图2所示。

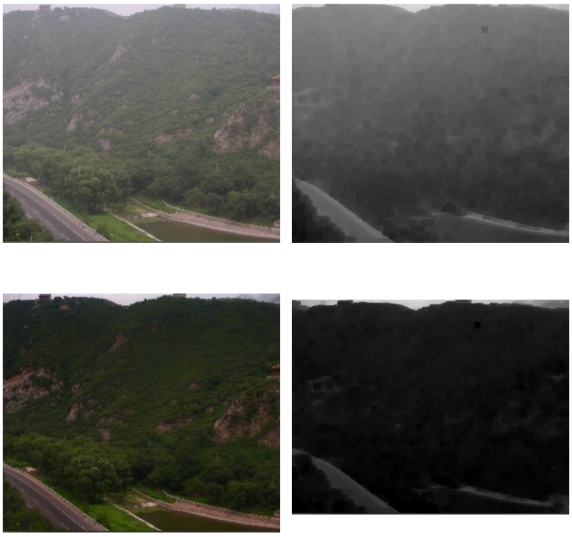

Figure 2: 有雾与无雾图像暗通道对比

2.3 算法步骤

首先，认定大气光A是一个常数，对等式(1)两边执行最小化运算，可以得到公式(5):

$$\min_{y\in\Omega(x)}\left(I^c(y)\right)=\min_{y\in\Omega(x)}\left(J^c(y)\right)t(x)+A^c(1-t(x)) \tag{5}$$

其中，c为颜色通道，$c\in\{R,G,B\}$。
对公式(5)两边同除A^c，可以得到公式(6):

$$\min_{y\in\Omega(x)}\left(\frac{I^c(y)}{A^c}\right)=\min_{y\in\Omega(x)}\left(\frac{J^c(y)}{A^c}\right)t(x)+(1-t(x)) \tag{6}$$

3 个颜色通道分别对公式(6)采取最小运算，可以得到公式(7):

$$\min_{c}\left(\min_{y\in\Omega(x)}\left(\frac{I^c(y)}{A^c}\right)\right)=\min_{c}\left(\min_{y\in\Omega(x)}\left(\frac{J^c(y)}{A^c}\right)\right)t(x)+(1-t(x)) \tag{7}$$

根据暗通道先验算法可知，J^{dark}的值非常小，趋近于0，根据公式(4)可以得到公式(8):

$$J^{\text{dark}}(x)=\min_{c}\left(\min_{y\in\Omega(x)}\left(J^c(y)\right)\right)=0 \tag{8}$$

A^c是一个常数，所以通过公式(6)和公式(8)可以得出公式(9):

$$\min_{c}\left(\min_{y\in\Omega(x)}\left(\frac{J^c(y)}{A^c}\right)\right)=0 \tag{9}$$

把公式(9)带入公式(7)，可以得到粗略的透射率 t ，如公式(10)所示:

$$t(x)=1-\min_{c}\left(\min_{y\in\Omega(x)}\left(\frac{I^c(y)}{A^c}\right)\right) \tag{10}$$

在现实中，即使是非常晴朗的天气，大气的透射率也不会完全为1，我们在看远处的物体时模糊依然存在。过度的去模糊会导致图像的不自然，所以为了还原贴近自然的图像，我们需要保留一定的模糊度，所以我们可以在式(10) 的模糊率前加一个系数，如公式(11)所示：

$$t(x) = 1 - \omega \min_{c} \left(\min_{y \in \Omega(x)} \left(\frac{I^c(y)}{A^c} \right) \right) \tag{11}$$

上述推论中都是假设全球达气光A值时已知的，在实际中，我们可以借助于暗通道图来从有雾图像中获取该值。具体步骤如下：

（1）从暗通道图中按照亮度的大小取前0.1%的像素。

（2）在这些位置中，在原始有雾图像I中寻找对应的具有最高亮度的点的值，作为A值。

到这一步，我们就可以进行无雾图像的恢复了。由式(1)可知：J(x)=[I(x)-A]/t(x)+A 。现在I,A,t都已经求得了，因此，完全可以进行J的计算。当投射图t的值很小时，会导致J的值偏大，从而使图像整体向白场过度，因此一般可设置一阈值t_0，当t值小于t_0时，令t=t_0，本文中所有效果图均以t_0=0.1为标准计算。因此，最终的恢复公式如公式(12)：

$$\mathbf{J}(x) = \frac{\mathbf{I}(x) - \mathbf{A}}{\max(t(x), t_0)} + \mathbf{A} \tag{12}$$

整体流程图如图3所示：

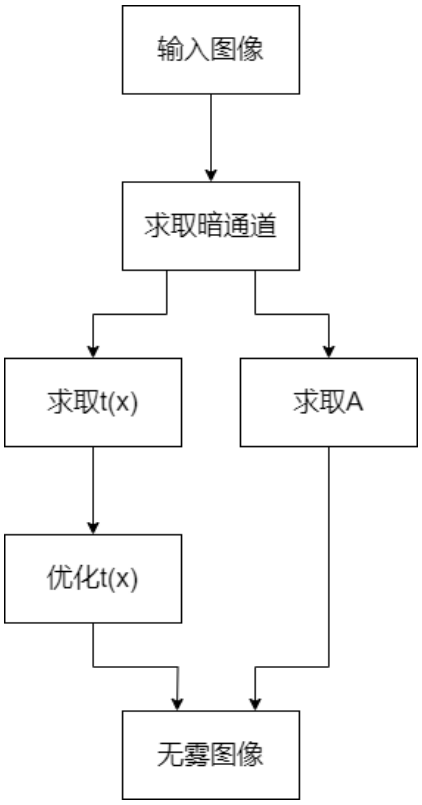

Figure 3: 算法流程图

3 程序设计

图4为有雾图像，将通过过以下程序对其进行去雾。

3.1 获取图像暗通道

Figure 4: 原始有雾图像

```
% Jdark = min(min(Jc))
function Jdark = get_dark_channel(image, win_size)

[m, n, ¬] = size(image);

Ir = image( : , : , 1);
Ig = image( : , : , 2);
Ib = image( : , : , 3);

% Select window definition initialization
Irr = 1./zeros(m + (win_size-1), n + (win_size-1));
Igg = 1./zeros(m + (win_size-1), n + (win_size-1));
Ibb = 1./zeros(m + (win_size-1), n + (win_size-1));

radius_size = floor(win_size / 2);
Irr(radius_size : (m + radius_size-1), radius_size :(n + radius_size-1)) = Ir;
Igg(radius_size : (m + radius_size-1), radius_size :(n + radius_size-1)) = Ig;
Ibb(radius_size : (m + radius_size-1), radius_size :(n + radius_size-1)) = Ib;

Jdark = zeros(m, n);

for i = 1 : 1 : m
    for j = 1 : 1 : n
        Rmin = min(min( Irr(i : i + (win_size-1), j : j + (win_size-1)) ));
        Gmin = min(min( Igg(i : i + (win_size-1), j : j + (win_size-1)) ));
        Bmin = min(min( Ibb(i : i + (win_size-1), j : j + (win_size-1)) ));

        Jdark(i,j) = min(min(Rmin,Gmin),Bmin);
    end
end

% Display image dark channel
figure('name', 'dark channel');
imshow(Jdark);
```

第一步：准备工作
1.接收输入图像和窗口大小参数;
2.获取图像的尺寸信息(高度m和宽度n);
3.将图像分解为RGB三个通道。
第二步：边界扩展
1.创建三个扩展后的矩阵(Irr, Igg, Ibb)，尺寸比原图大;
2.扩展尺寸= 原尺寸+ (窗口大小-1);
3.使用无穷大值填充扩展区域;
4.将原始图像数据放在扩展矩阵的中心位置。
第三步：计算暗通道
1.创建结果矩阵Jdark;
2.对图像的每个像素点(i,j):
(1)在该点周围取一个窗口大小的区域;
(2)分别计算该窗口内RGB三个通道的最小值;
(3)从这三个最小值中再取一个最小值;

(4)将最终的最小值存入结果矩阵对应位置。

第四步：显示结果

1.用figure创建新窗口;

2.显示计算得到的暗通道图像.

程序3.1执行完获得的暗通道图像如图5所示。

Figure 5: 暗通道图像

3.2 大气光强度求解

```
% Solve for atmospheric light intensity A
function Atomsphere = get_atomsphere(image, dark_channel)

[m, n, ¬] = size(image);

pixels_num = m * n; % Sum of pixels

select_pixel_num = floor(pixels_num * 0.0001); % Select 0.1% of pixels

max_pix = [0, 0];

for i = 1 : 1 : select_pixel_num
    MaxVaule = max(max(dark_channel));
    [x, y] = find(dark_channel == MaxVaule);
    dark_channel(dark_channel == MaxVaule) = 0; % Clear the maximum value and find ...
        other suitable values.
    max_pix = vertcat(max_pix, [x, y]);

    num = length(max_pix);
    if num > select_pixel_num
        break;
    end
end

% Take select_pixel_num appropriate values, and remove the first initialization ...
    value and subsequent values that exceed the number.
Max_Pix = max_pix(2 : select_pixel_num + 1, : );

Rsum = 0;  Jr = image(:,:,1);
Gsum = 0;  Jg = image(:,:,2);
Bsum = 0;  Jb = image(:,:,3);

for i = 1 : 1 : select_pixel_num
    Rsum = Rsum + Jr(Max_Pix(i, 1), Max_Pix(i, 2));
    Gsum = Gsum + Jg(Max_Pix(i, 1), Max_Pix(i, 2));
    Bsum = Bsum + Jb(Max_Pix(i, 1), Max_Pix(i, 2));
end

sum = [Rsum/select_pixel_num, Gsum/select_pixel_num, Bsum/select_pixel_num];

Atomsphere = repmat(reshape(sum, [1, 1, 3]), m, n);
```

第一步：准备工作

1.接收输入图像和暗通道图像;
2.获取图像的尺寸信息（高度和宽度）;
3.计算图像的总像素数;
4.选择需要计算的前0.1% 的像素数。
第二步：寻找暗通道的亮度最大值
1.初始化存储最大值像素位置的数组;
2.循环进行以下操作，直到选出所需数量的像素:
(1)找到暗通道图像中的最大值;
(2)获取最大值所在的坐标(x, y);
(3)将找到的最大值清零，继续寻找下一个最大值;
(4)将坐标保存到数组中;
3.选出数组中前0.1% 的坐标点。
第三步：计算大气光强度A
1.初始化R、G、B 三个通道的总和;
2.遍历前0.1% 的像素坐标点:
(1)从图像中提取对应像素的RGB值;
(2)分别累加RGB通道的值;
3.计算RGB三个通道的均值;
4.将均值作为大气光强度A，扩展成与原图大小相同的矩阵。
第四步：结果
1.返回与原图尺寸相同的大气光强度矩阵。

3.3 透视率图获取

```
function q = guidedfilter(I, p, r, eps)
%   GUIDEDFILTER   O(1) time implementation of guided filter.
%
%   - guidance image: I (should be a gray-scale/single channel image)
%   - filtering input image: p (should be a gray-scale/single channel image)
%   - local window radius: r
%   - regularization parameter: eps

[hei, wid] = size(I);
N = boxfilter(ones(hei, wid), r); % the size of each local patch; N=(2r+1)^2 except ...
    for boundary pixels.

mean_I = boxfilter(I, r) ./ N;
mean_p = boxfilter(p, r) ./ N;
mean_Ip = boxfilter(I.*p, r) ./ N;
cov_Ip = mean_Ip - mean_I .* mean_p; % this is the covariance of (I, p) in each ...
    local patch.

mean_II = boxfilter(I.*I, r) ./ N;
var_I = mean_II - mean_I .* mean_I;

a = cov_Ip ./ (var_I + eps);
b = mean_p - a .* mean_I;

mean_a = boxfilter(a, r) ./ N;
mean_b = boxfilter(b, r) ./ N;

q = mean_a .* I + mean_b;
end
```

第一步：准备工作
1. 接收引导图像I和待滤波图像p;
2. 输入参数包括局部窗口半径r和正则化参数ϵ;
3. 获取图像尺寸（高度和宽度）。
第二步：计算局部统计量
1. 使用boxfilter计算每个局部窗口的像素总数N，窗口大小为$(2r+1)^2$;
2. 计算局部均值:

(1)mean_I：引导图像I 的局部均值；
(2)mean_p：输入图像p 的局部均值；
3. 计算协方差和方差：
(1)协方差$\text{cov}_{Ip} = \text{mean}_{Ip} - \text{mean}_I \cdot \text{mean}_p$；
(2)方差$\text{var}_I = \text{mean}_{II} - \text{mean}_I^2$。
第三步：计算线性系数
1. 计算线性系数a 和偏置b：
(1)$a = \frac{\text{cov}_{Ip}}{\text{var}_I + \epsilon}$；
(2)$b = \text{mean}_p - a \cdot \text{mean}_I$；
2. 使用boxfilter计算a 和b 的局部均值：
(1)mean_a：系数a 的局部均值；
(2)mean_b：系数b 的局部均值。
第四步：生成滤波输出
1. 结合线性模型，生成输出图像q：
(1)$q = \text{mean}_a \cdot I + \text{mean}_b$。
第五步：结果
1. 函数返回滤波后的输出图像q，其保留了边缘信息，同时去除了噪声。
优化后的透视率图如图6所示。

Figure 6: 优化后的透视率图

3.4 主函数

```
win_size = 15;
W = 0.95;
t0 = 0.1;
r = win_size*4;
eps = 10^-6;

image = double(imread('forest.jpg'))/255;
figure('name', 'forest.jpg'); imshow(image);

Jdark = get_dark_channel(image, win_size);

Atom  = get_atomsphere(image, Jdark);

t = 1 - W * get_dark_channel(image ./ Atom, win_size);

trans_est = guidedfilter(double(rgb2gray(image)), t, r, eps);
figure('name', 't'); imshow(trans_est);

max_trans_est = repmat(max(trans_est, 0.1), [1, 1, 3]);

% Solving for clear images
% J = (I-A)/max(t,t(0)) + A
result = ( (image - Atom)./max_trans_est ) + Atom;

figure('name', 'forest_recover.jpg'); imshow(result);
```

第一步：参数设置
1. 设置窗口大小win_size = 15，用于暗通道计算；
2. 设置常量参数：
(1)$W = 0.95$：权重参数，用于估算透射率；
(2)$t_0 = 0.1$：最低透射率值；
(3)$r = \text{win\_size} \times 4$：引导滤波的窗口半径；
(4)$\epsilon = 10^{-6}$：正则化参数，用于引导滤波。
3. 读取图像forest.jpg并归一化到[0, 1]。
第二步：暗通道和大气光估计
1. 调用函数get_dark_channel，计算输入图像的暗通道图像Jdark；
2. 调用函数get_atomsphere，估算大气光值Atom。
第三步：透射率估计
1. 初步估算透射率t：
(1)使用公式$t = 1 - W \cdot \text{get_dark_channel}(I/A)$，计算初步的透射率图；
2. 对初步估算的透射率图t 进行引导滤波：
(1)转换图像为灰度图；
(2)使用guidedfilter对透射率进行细化，得到优化后的透射率trans_est。
第四步：图像恢复
1. 设置最低透射率值$t_0 = 0.1$；
2. 对优化后的透射率trans_est进行扩展，以使维度与原图相同；
3. 根据去雾公式恢复清晰图像：
(1)$J = \frac{I-A}{\max(t,t_0)} + A$；
(2)使用上面的公式计算恢复后的图像result。
第五步：结果展示
1. 显示原始图像；
2. 显示优化后的透射率图像；
3. 显示去雾后的清晰图像result。
去雾成功的图像如图7所示。

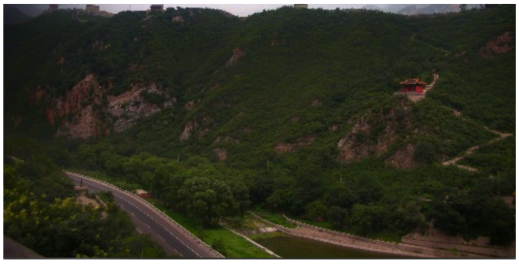

Figure 7: 去雾图像

4 UI设计

使用matlab自带的App Designer进行UI设计，通过拖拉组件形成界面，然后在组件里面写回调函数。初步设计如图8所示，成果如图9所示。

Figure 8: 初步设计界面

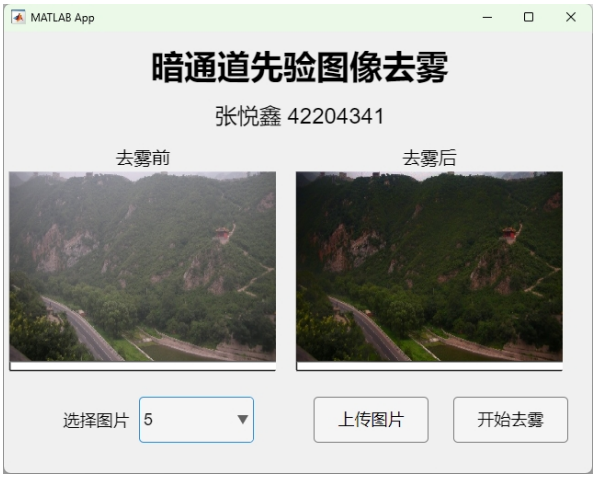

Figure 9: 成果显示

References

[1] He, Kaiming, Jian Sun, and Xiaoou Tang. "Single image haze removal using dark channel prior." IEEE transactions on pattern analysis and machine intelligence 33.12 (2010): 2341-2353.