**简单的同态滤波和Retinex的图像去雾算法**

**摘要：**在雾天条件下通过摄像头，监控器等设备得到的图像，会变得质量较差，对比度较低。由于雾的影响，一方面会影响户外出行人们对远处事物的观察，造成交通事故等，另一方面严重影响了监控设备，摄影机等在户外使用的设备，造成交通状况紊乱。对此我融合了同态滤波和Retinex算法，开发一套简化版本的图像去雾算法。同态滤波过程中，用DFT代替FFT，增强了同态滤波的灵活性，再利用Retinex算法对较暗区域的图像处理有明显效果,整体降低了图像的模糊度，但是因为同态滤波的的参数需要根据图像的不同来调节高斯⾼通滤波器的参数，不能做到⾃适应。

**关键词：**同态滤波；Retinex

**一、引言**

针对图像去雾算法现在已经有很多成熟的算法，比较有代表性的就是直方图均衡化方法，但是通过直方图会放大噪声，所以得到图像的效果并不理想，但是相对于同态滤波来说，均衡化更加灵活。同态滤波和Retinex得到的图像收到参数影响较大，且Retinex算法容易产生光晕。实现效果最好的是何恺明博士等人提出的基于暗通道的图像去雾算法，即使没有融合深度学习，依然获得了理想的结果。

**二、方案选择与设计**

**Project的方案选择和思路：**

如果仅仅对图像进行Retinex增强，很容易使得图像出现光圈，而直方图均衡化和同态滤波提高图像对比度的作用，使原图像光照趋于平滑，所以我可以选择其中较优的一种是算法先对图像进行处理，然后再进行Retinex增强，这就避免了Retinex会产生光晕的现象。

第一步：调用自己编写的离散傅里叶函数进行离散傅里叶变换，将运算从空间域转换成时域。

第二步：调⽤⾃⼰实现的高斯高通滤波器并作⽤在经过离散傅里叶变换后的图像上。

第三步：将第二步得到的结果进行离散傅里叶逆变换，得到同态滤波的结果。

第四步：因为得到的是彩色图像，所以需要把他分解成R,G,B三个维度在进行数据类型转换。

第五步：将得到的图像进行Frankle-McCann的Retinex增强。得到最终图像。

**三、实验结果**

**Project设计结果：**

**1、同态滤波先对处理光照不均匀的图像**

我先进行了同态滤波和直方图均衡化处理的对比，运行了文件夹中的temp.m这个文档，其中Homomorphicfliter函数调用了dft和Idft，GaussianHighPassFilter函数实现了同态滤波，而直方图均衡化直接调用了Matlab自带的函数，通过调整同态滤波的参数rH,rL,c,D0得到的图像明显优于直方图均衡化，但是直方图均衡化是可以自适应的，即不需要手动设置参数，所以如果能够提出一种同态滤波的自适应算法，可能会让效果更具有普遍性。

|  |  |
| --- | --- |
| 直方图均衡化后得到的图像 | 同态滤波后得到的图像 |
| zft | tt |

图1 直方图均衡化后与同态滤波后得到的图像对比

## 上图是两种算法得到的图像，明显可以看到，通过同态滤波得到的图像更加平滑，也没有很刺眼的噪声。图像比直方图均衡化后的更加清晰，更适宜用来进行Retinex增强。

**2、同态滤波后进行Retinex增强**

从同态滤波得到的图像之后，将其分成R，G，B三个通道之后进行数据类型转换，进行Frankle-McCann Retinex增强算法，这里可以直接运行main.m文件，不需要提前使用temp.m，在实验中我发现，迭代次数越小，图像的对比度越小，动态压缩范围越小。迭代次数越大，图像的对比度越强烈。从直方图上看，迭代次数小时，直方图会聚集在一起，随着迭代次数等增加，直方图逐渐分布到全部动态范围内。 当迭代次数特别大时，图像会和原始图像很接近。所以理论上如果设置足够大的迭代次数可以复原图像，这里仅设置迭代次数为10，依然取得了很好的效果。

|  |  |
| --- | --- |
| 原始图像 | 同态滤波和Retinex增强后的图像 |
| a | last |

图2 原始图像与同态滤波和Retinex增强后得到的图像对比

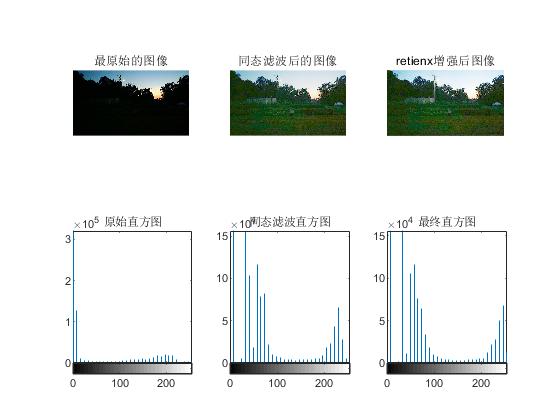
这类算法不仅有效的实现了去雾，而且对于去阴影也有很好的作用，下图时我在调试对光照不均匀的情况下增强图片对比度的实验，可以明显看到，经过两次处理后的直方图分布无比均衡，而且阴影部分更加清晰。

图3 光照不均匀的情况下增强图片对比度的实验结果

对于有的图像经过去雾算法处理以后，得到的直方图结果获取并不优于原有的直方图，但是可以看到他的去雾效果是十分明显的。

**四、结论总结**

这个project用到了离散傅里叶变换，离散傅里叶逆变换，高斯高通滤波，直方图均衡化，同态滤波和Retinex算法，一般来说，单纯的使用同态滤波的效果是要优于直方图均衡化的，但是因为要调整高斯高通滤波器的参数，所以对图像的适应能力较差。而同态滤波后的图像因为具有对比度提高和光照修正的效果，所以用同态滤波吧后的图像去进行Retinex增强的效果是优于单纯的进行Retinex增强。从学习这两种算法的过程中，我认识到同态滤波和Retinex增强算法是很相似的，他们实际上都是通过衰减低频，增高高频来达到了图像增强的目的。最后我对同态滤波算法提出一种改进的方式，但是没有实现。就是先使用同态滤波对图像进行一个高低分频，再将得到的低频分量进⾏全局的直⽅图均衡化处理；最后将⾼频分量跟低频分量进⾏线性融合。这样可以减少直方图均衡化增大的噪声和同态滤波带来的低灰度比例。

**参考文献**

[1]张新明,沈兰荪.基于小波的同态滤波器用于图像对比度增强[J].电子报,2001(04):531-533.

[2]汪秦峰,陈莉.基于同态滤波和Retinex的图像去雾算法[J].火控雷达技术,2016(45):45-46.

**附 录**

%%%%%%%%%%%%%%%% dft.m%%%%%%%%%%%%%%%%%%%%%%%%

%%离散傅里叶函数编写

function F\_ij = dft(f\_ij)

if ndims(f\_ij) == 3

f\_ij = rgb2gray(f\_ij);

end

% 这里将整数切换为双精度

f\_ij = double(f\_ij);

% 这里获取了图像的大小

[nrows, ncols] = size(f\_ij);

% 列式FT的矢量化实现

vy = (0:nrows-1)' \* (0:nrows-1);

M\_vy = exp(-1i\*2\*pi\*vy/nrows);

F\_xv = M\_vy \* f\_ij;

% 行式FT的矢量化实现

ux = (0:ncols-1)' \* (0:ncols-1);

M\_ux = exp(-1i\*2\*pi\*ux/ncols);

F\_ij = F\_xv \* M\_ux;

end

%%%%%%%%%%%%%%%%%%Idft.m%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%离散傅里叶逆变换函数编写

function f\_xy = Idft(F\_ij)

if ndims(F\_ij) == 3

F\_ij = rgb2gray(F\_ij);

end

[nrows, ncols] = size(F\_ij);

vy = (0:nrows-1)' \* (0:nrows-1);

M\_vy = exp(1i\*2\*pi\*vy/nrows);

f\_uy = M\_vy \* F\_ij;

ux = (0:ncols-1)' \* (0:ncols-1);

M\_ux = exp(1i\*2\*pi\*ux/ncols);

f\_xy = real(f\_uy \* M\_ux / (ncols\*nrows));

end

%%%%%%%%%%%%%%%GaussianHighPassFilter.m %%%%%%%%%%%%%%%%%%

%%% 高斯高通滤波器的实现

function H = GaussianHighPassFilter(I, RH, RL, C, D0)

if ndims(I) == 3

I = rgb2gray(I);

end

[M, N] = size(I);%%获取图像大小

% 记录图像的中点

n1 = floor(M/2);

n2 = floor(N/2);

H = zeros(n1, n2);

%计算高斯高通滤波器H（i，j）

for i=1:M

for j=1:N

D(i,j)=((i-n1).^2+(j-n2).^2);

H(i,j)=(RH-RL).\*(exp(C\*(-D(i,j)./(D0^2))))+RL;%高斯同态滤波

end

end

end

%%%%%%%%%%%%%%%%Homomorphicfilter.m%%%%%%%%%%%%%%%%%%%%

function I\_homo = Homomorphicfilter(I, rH, rL, c, d0)

%%if ndims(I) == 3

%%I = rgb2gray(I);%%获取图像

%%end

% I=double(rgb2gray(I));

I = double(I);

[M,N,channals]=size(I);

for ch=1:channals

I1=log(I(:,:,ch)+1);%取对数

FI=dft(I1);%傅里叶变换

H = GaussianHighPassFilter(I, rH, rL,c, d0);

I2=Idft(H.\*FI);%傅里叶逆变换

I\_homo(:,:,ch)=real(exp(I2));

end

minV=min(min(min(I\_homo)));

maxV=max(max(max(I\_homo)));

for ch=1:channals

for i=1:M

for j=1:N

I\_homo(i,j,ch)=255\* (I\_homo(i,j,ch)-minV)./(maxV-minV);

end

end

end

end

%%%%%%%%%%%%%%%%%%%retinex\_frankle\_mccann.m%%%%%%%%%%%%%%

function Retinex = retinex\_frankle\_mccann(L, nIterations)

global RR IP OP NP Maximum

RR = L;

Maximum = max(L(:));

[nrows, ncols] = size(L);

shift = 2^(fix(log2(min(nrows, ncols)))-1);

OP = double(Maximum)\*ones(nrows, ncols);

while (abs(shift) >= 1)

for i = 1:nIterations

CompareWith(0, shift);

CompareWith(shift, 0);

end

shift = -shift/2;

end

Retinex = NP;

%%%%%%%%%%%%%%%%%%%%CompareWith.m%%%%%%%%%%%%%%%%%%%%%%%%

function CompareWith(row, col)

global RR IP OP NP Maximum

IP = OP;

if (row + col > 0)

IP((row+1):end, (col+1):end) = double(OP(1:(end-row), 1:(end-col))) + ...

double(RR((row+1):end, (col+1):end) )- double(RR(1:(end-row), 1:(end-col)));

else

IP(1:(end+row), 1:(end+col)) = double(OP((1-row):end, (1-col):end)) + ...

double(RR(1:(end+row),1:(end+col))) - double(RR((1-row):end, (1-col):end));

end

IP(IP > Maximum) = Maximum;

NP = (IP + OP)/2;

OP = NP;

%%%%%%%%%%%%%%%%%%%%main.m %%%%%%%%%%%%%%%%%%%%%%%%%%%

clc,close all,clear all;

img=imread('image.jpg');

rH = 2; rL = 0.1; c = 0.5; D0 = 3;

ImgOriginal = Homomorphicfilter(img, rH, rL, c, D0);

figure(1)

subplot(1,2,2),imshow(uint8(ImgOriginal));title('同态滤波增强后');

[m,n,z] = size(ImgOriginal);

ImgOut = zeros(m,n,z);

nIterations=100;

for i = 1:z

ImChannel = log(double(ImgOriginal(:,:,i))+eps);

ImgOut(:,:,i)=retinex\_frankle\_mccann(ImChannel,nIterations);

ImgOut(:,:,i)=exp(ImgOut(:,:,i));

a=min(min(ImgOut(:,:,i)));

b=max(max(ImgOut(:,:,i)));

ImgOut(:,:,i)=((ImgOut(:,:,i)-a)/(b-a))\*255;

end

ImgOut=uint8(ImgOut);

figure(1);

subplot(2,3,1);imshow(img);title('最原始的图像');

subplot(2,3,2);imshow(uint8(ImgOriginal));title('同态滤波后的图像');

subplot(2,3,3);imshow(ImgOut);title('retienx增强后图像');

subplot(2,3,4);imhist(img,32);title('原始直方图')

subplot(2,3,5);imhist(uint8(ImgOriginal),32);title('同态滤波直方图')

subplot(2,3,6);imhist(ImgOut,32);title('最终直方图')

figure(2);

imshow(uint8(ImgOriginal));

figure(3);

imshow(ImgOut);

%%%%%%%%%%%%%%%%%%%%%%%%temp.m%%%%%%%%%%%%%%%%%%%%%%%%%

I = imread('a.jpg');

I\_gray = rgb2gray(I);

[I\_eq, I\_eq\_pdf] = histeq(I);

rH = 2; rL = 0.6; c = 0.5; D0 = 2;

I\_homo = Homomorphicfilter(I, rH, rL, c, D0);

%imwrite(I\_homo,'temp/test.jpg');

figure(1);

subplot(2,3,1);imshow(I\_gray);title('Original grayscale image');

subplot(2,3,2);imshow(I\_eq);title('直方均衡化的图像');

subplot(2,3,3);imshow(uint8(I\_homo));title('同态滤波图像');

subplot(2,3,4);imhist(I\_gray,32);title('原始直方图')

subplot(2,3,5);imhist(I\_eq,32);title('均衡化直方图')

subplot(2,3,6);imhist(uint8(I\_homo),32);title('同态滤波直方图')

figure(2), imshow(I\_eq);

figure(3), imshow(uint8(I\_homo));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**注：本文的核心算法是同态滤波以及Retienx增强融合的main函数。**