实验 10-12

学号：201958508239 姓名：**刘鑫** 班级： 计198-2

时间： 2021.6.16 地点： 计算中心406 指导老师：\_**\_徐金东\_**\_\_

一、实验目的

1. 对一幅图像依次添加高斯噪声、椒盐噪声，分别运用算术均值滤波、几何均值滤波、中值滤波、修正的阿尔法均值滤波进行图像恢复，显示并比较分析结果。

2. 设计自适应中值滤波器，对图像添加椒盐噪声，比较中值滤波和自适应中值滤波结果，并分析其原因。

3. 彩色图像在RGB和HSI空间下分别进行平滑（一种即可）。

4. 彩色图像在RGB和HSI空间下分别进行锐化（一种即可）。

二、实验内容

**1．编写spfilt.m 和 myImgInpainting.m文件**

%-------------spfilt.m---------------

function f = spfilt(g, type, m, n, parameter)

if nargin == 2

m = 3; n = 3; Q = 1.5; d = 2;

elseif nargin == 5

Q = parameter; d = parameter;

elseif nargin == 4

Q = 1.5; d = 2;

else

error('Wrong number of inputs.');

end

% Do the filtering.

switch type

case 'amean'

w = fspecial('average', [m n]);

f = imfilter(g, w, 'replicate');

case 'gmean'

f = gmean(g, m, n);

case 'hmean'

f = harmean(g, m, n);

case 'chmean'

f = charmean(g, m, n, Q);

case 'median'

f = medfilt2(g, [m n], 'symmetric');

case 'max'

f = ordfilt2(g, m\*n, ones(m, n), 'symmetric');

case 'min'

f = ordfilt2(g, 1, ones(m, n), 'symmetric');

case 'midpoint'

f1 = ordfilt2(g, 1, ones(m, n), 'symmetric');

f2 = ordfilt2(g, m\*n, ones(m, n), 'symmetric');

f = imlincomb(0.5, f1, 0.5, f2);

case 'atrimmed'

if (d <= 0) | (d/2 ~= round(d/2))

error('d must be a positive, even integer.')

end

f = alphatrim(g, m, n, d);

otherwise

error('Unknown filter type.')

end

end

%------------------------------------

function image = changeclass(class, varargin)

switch class

case 'uint8'

image = im2uint8(varargin{:});

case 'uint16'

image = im2uint16(varargin{:});

case 'double'

image = im2double(varargin{:});

otherwise

error('Unsupported IPT data class.');

end

end

%--------------------------------%

function f = gmean(g, m, n)

% Implements a geometric mean filter.

inclass = class(g);

g = im2double(g);

% Disable log(0) warning.

warning off;

f = exp(imfilter(log(g), ones(m, n), 'replicate')).^(1 / m / n);

warning on;

f = changeclass(inclass, f);

end

%--------------------------------%

function f = harmean(g, m, n)

% Implements a harmonic mean filter.

inclass = class(g);

g = im2double(g);

f = m \* n ./ imfilter(1./(g + eps),ones(m, n), 'replicate');

f = changeclass(inclass, f);

end

%--------------------------------------%

function f = charmean(g, m, n, q)

% Implements a contraharmonic mean filter.

inclass = class(g);

g = im2double(g);

f = imfilter(g.^(q+1), ones(m, n), 'replicate');

f = f ./ (imfilter(g.^q, ones(m, n), 'replicate') + eps);

f = changeclass(inclass, f);

end

%---------------------------------%

function f = alphatrim(g, m, n, d)

% Implements an alpha-trimmed mean filter.

inclass = class(g);

g = im2double(g);

f = imfilter(g, ones(m, n), 'symmetric');

for k = 1:d/2

f = imsubtract(f, ordfilt2(g, k, ones(m, n), 'symmetric'));

end

for k = (m\*n - (d/2) + 1):m\*n

f = imsubtract(f, ordfilt2(g, k, ones(m, n), 'symmetric'));

end

f = f / (m\*n - d);

f = changeclass(inclass, f);

end

%------myImgInpainting.m------

function myImgInpainting(filename)

g = imread(filename);

if length(size(g)) > 2

g = rgb2gray(g);

end

g1 = imnoise(g, 'gaussian');

g2 = imnoise(g, 'salt & pepper');

f11 = spfilt(g1, 'amean');

f12 = spfilt(g1, 'gmean');

f13 = spfilt(g1, 'median');

f14 = spfilt(g1, 'atrimmed');

f21 = spfilt(g2, 'amean');

f22 = spfilt(g2, 'gmean');

f23 = spfilt(g2, 'median');

f24 = spfilt(g2, 'atrimmed');

subplot(2, 3, 1);

imshow(g, []);

title('Raw Image');

subplot(2, 3, 2);

imshow(g1, []);

title('Gaussian noise Image');

subplot(2, 3, 3);

imshow(f11, []);

title('Amean Image');

subplot(2, 3, 4);

imshow(f12, []);

title('Gmean Image');

subplot(2, 3, 5);

imshow(f13, []);

title('Median Image');

subplot(2, 3, 6);

imshow(f14, []);

title('Atrimmed Image');

figure

subplot(2, 3, 1);

imshow(g, []);

title('Raw Image');

subplot(2, 3, 2);

imshow(g2, []);

title('Salt & pepper noise Image');

subplot(2, 3, 3);

imshow(f21, []);

title('Amean Image');

subplot(2, 3, 4);

imshow(f22, []);

title('Gmean Image');

subplot(2, 3, 5);

imshow(f23, []);

title('Median Image');

subplot(2, 3, 6);

imshow(f24, []);

title('Atrimmed Image');

end

**2.编写adpmedian.m和myAdpmedian.m实现中值滤波和自适应中值滤波**

%自适应中值滤波

function f = adpmedian (g, Smax)

if (Smax <= 1) || (Smax/2 == round(Smax/2)) || (Smax ~= round(Smax))

error ('SMAX must be an odd integer > 1.')

end

[M, N] = size(g);

%Initial setup.

f = g;

f(:) = 0;

alreadyProcessed = false (size(g));

% Begin filtering.

for k = 3:2:Smax

zmin = ordfilt2(g, 1, ones(k, k),'symmetric');

zmax = ordfilt2(g, k \* k, ones(k, k), 'symmetric');

zmed = medfilt2(g, [k k], 'symmetric');

processUsingLevelB = (zmed > zmin) & (zmax > zmed) &...

~alreadyProcessed;

zB = (g > zmin) & (zmax > g);

outputZxy = processUsingLevelB & zB;

outputZmed = processUsingLevelB & ~zB;

f (outputZxy) = g(outputZxy);

f (outputZmed) = zmed(outputZmed);

alreadyProcessed = alreadyProcessed | processUsingLevelB;

if all (alreadyProcessed (:))

break;

end

end

f (~alreadyProcessed) = zmed (~alreadyProcessed);

end

%-----载入图像并进行中值滤波和自适应中值滤波----------

function myAdpmedian(filePath)

g = imread(filePath);

if length(size(g)) > 2

g = rgb2gray(g);

end

f = imnoise(g, 'salt & pepper', .25);

g1 = medfilt2(f, [7 7], 'symmetric');

g2 = adpmedian(f, 7);

subplot(2, 2, 1);

imshow(g);

title('Raw Image');

subplot(2, 2, 2);

imshow(f);

title('salt & pepper');

subplot(2, 2, 3);

imshow(g1);

title('median2');

subplot(2, 2, 4);

imshow(g2);

title('adpmedian');

end

**3.编写RGB和HSI互相转换函数**

%------RGB转HSI-------

function hsi = rgb2hsi(rgb)

rgb = im2double(rgb);

r = rgb(:, :, 1);

g = rgb(:, :, 2);

b = rgb(:, :, 3);

% Implement the conversion equations.

num = 0.5\*((r - g) + (r - b));

den = sqrt((r - g).^2 + (r - b).\*(g - b));

theta = acos(num./(den + eps));

H = theta;

H(b > g) = 2\*pi - H(b > g);

H = H/(2\*pi);

num = min(min(r, g), b);

den = r + g + b;

den(den == 0) = eps;

S = 1 - 3.\* num./den;

H(S == 0) = 0;

I = (r + g + b)/3;

% Combine all three results into an hsi image.

hsi = cat(3, H, S, I);

end

function rgb = hsi2rgb(hsi)

%rgb=hsi2rgb(hsi)把一幅HSI图像转换为RGB图像

%其中HSI是double型

%提取HSI的各个分量

hsi = im2double(hsi);%把hsi转化为双精度浮点类型

H = hsi(:, :, 1) \* 2 \* pi;

S = hsi(:, :, 2);

I = hsi(:, :, 3);

%执行变换方程

R = zeros(size(hsi, 1), size(hsi, 2));

G = zeros(size(hsi, 1), size(hsi, 2));

B = zeros(size(hsi, 1), size(hsi, 2));

%RG区(0<=H<2\*pi/3)

idx = find((0 <= H) & (H < 2 \* pi / 3));%寻找0<=H<2\*pi/3

B(idx) = I(idx).\*(1 - S(idx));

R(idx) = I(idx).\*(1 + S(idx).\*cos(H(idx))./cos(pi / 3 - H(idx)));

G(idx) = 3 \* I(idx) - (R(idx) + B(idx));

%BG区(2\*pi/3<=H<4\*pi/3)

idx = find((2\*pi/3<=H) & (H<4\*pi/3));%寻找2\*pi/3<=H<4\*pi/3

R(idx) = I(idx).\*(1-S(idx));

G(idx) = I(idx).\*(1+S(idx).\*cos(H(idx)-2\*pi/3)./cos(pi-H(idx)));

B(idx) = 3\*I(idx)-(R(idx)+G(idx));

%BR区(4\*pi/3<=H<=2\*pi)

idx = find((4\*pi/3<=H) & (H<=2\*pi));%寻找4\*pi/3<=H<=2\*pi

G(idx) = I(idx).\*(1-S(idx));

B(idx) = I(idx).\*(1 + S(idx).\*cos(H(idx) - 4 \* pi/3)./cos(5 \* pi/3 - H(idx)));

R(idx) = 3 \* I(idx) - (G(idx) + B(idx));

%将3个分量联合成为一个RGB图像

rgb = cat(3, R, G, B);

rgb = max(min(rgb, 1), 0);

end

**4.编写基于RGB和HIS空间的锐化和平滑处理函数**

%-----------对彩色图像进行平滑滤波----------

function myColourAverageFilter(filePath)

% 读入图像并获转换为HSI格式

I = imread(filePath);

H = rgb2hsi(I);

h = H(:, :, 1);

s = H(:, :, 2);

i = H(:, :, 3);

% 平滑滤波模板

w = fspecial('average', 25);

% 直接在RGB空间进行平滑处理

I1 = imfilter(I, w, 'replicate');

% 分别对H S I 分量平滑，转回RGB 观察效果

h\_filtered = imfilter(h, w, 'replicate');

s\_filtered = imfilter(s, w, 'replicate');

i\_filtered = imfilter(i, w, 'replicate');

f0 = cat(3, h\_filtered, s\_filtered, i\_filtered);

f0 = hsi2rgb(f0);

f0 = min(f0, 1);

f1 = cat(3, h\_filtered, s, i);

f1 = hsi2rgb(f1);

f1 = min(f1, 1);

f2 = cat(3, h, s\_filtered, i);

f2 = hsi2rgb(f2);

f2 = min(f2, 1);

f3 = cat(3, h, s, i\_filtered);

f3 = hsi2rgb(f3);

f3 = min(f3, 1);

subplot(2, 3, 1);

imshow(I);

title('Raw Image');

subplot(2, 3, 2);

imshow(I1);

title('RGB Average filter');

subplot(2, 3, 3);

imshow(f0);

title('HSI AverageFilter');

subplot(2, 3, 4);

imshow(f1);

title('HSI H-AverageFilter');

subplot(2, 3, 5);

imshow(f2);

title('HSI S-AverageFilter');

subplot(2, 3, 6);

imshow(f3);

title('HSI I-AverageFilter');

end

%--------对彩色图进行图像锐化--------

function myColourSharpening(filePath)

% 读入图像并获转换为HSI格式

I = imread(filePath);

H = rgb2hsi(I);

h = H(:, :, 1);

s = H(:, :, 2);

i = H(:, :, 3);

% 拉普拉斯算子

lapmask = [1 1 1; 1 -8 1; 1 1 1];

% 直接在RGB空间进行平滑处理

I1 = imsubtract(I, imfilter(I, lapmask, 'replicate'));

% 分别对H S I 分量平滑，转回RGB 观察效果

h\_filtered = imsubtract(h, imfilter(h, lapmask, 'replicate'));

s\_filtered = imsubtract(s, imfilter(s, lapmask, 'replicate'));

i\_filtered = imsubtract(i, imfilter(i, lapmask, 'replicate'));

f0 = cat(3, h\_filtered, s\_filtered, i\_filtered);

f0 = hsi2rgb(f0);

f0 = min(f0, 1);

f1 = cat(3, h\_filtered, s, i);

f1 = hsi2rgb(f1);

f1 = min(f1, 1);

f2 = cat(3, h, s\_filtered, i);

f2 = hsi2rgb(f2);

f2 = min(f2, 1);

f3 = cat(3, h, s, i\_filtered);

f3 = hsi2rgb(f3);

f3 = min(f3, 1);

subplot(2, 3, 1);

imshow(I);

title('Raw Image');

subplot(2, 3, 2);

imshow(I1);

title('RGB LapSharpening');

subplot(2, 3, 3);

imshow(f0);

title('HSI LapSharpening');

subplot(2, 3, 4);

imshow(f1);

title('HSI H-LapSharpening');

subplot(2, 3, 5);

imshow(f2);

title('HSI S-LapSharpening');

subplot(2, 3, 6);

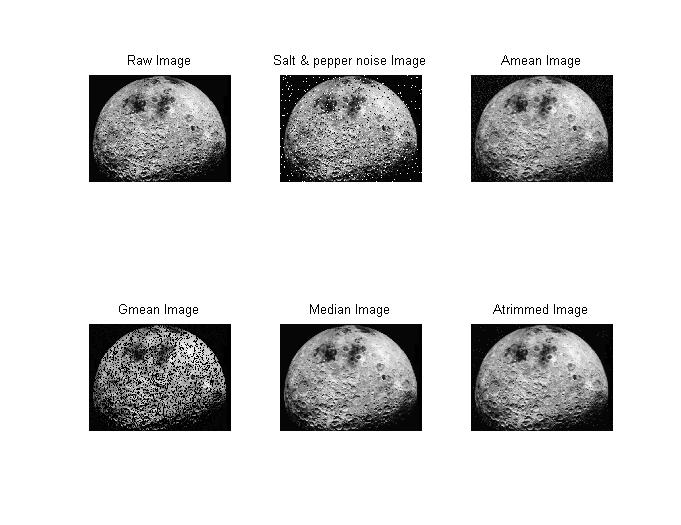
imshow(f3);

title('HSI I-LapSharpening');

end

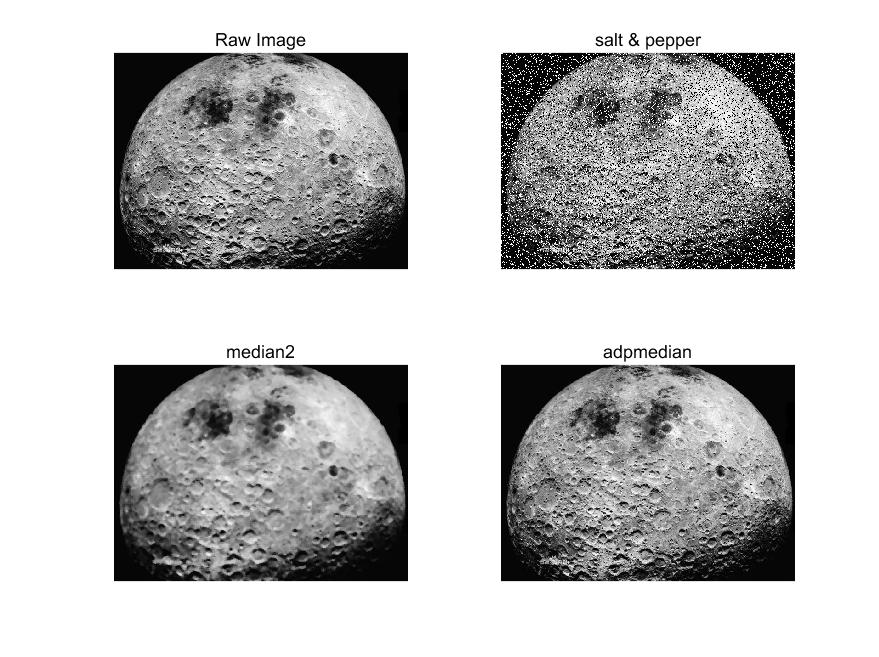
三、实验结果与分析

**实验10**



可以明显观察到修正的阿尔法均值滤波效果最佳，几何均值滤波效果最差，因为在混合噪声中，最值对几何均值滤波影响较大。

**实验11**



**实验12**



从运行结果来看，对HIS的H（色调）进行滤波是不合适的，往往会出现原图没有的一些颜色。

四、总结

在实验10中，学习了修正阿尔法均值滤波器，该方法类似于去掉最高分与去掉最低分评价一个选手水平的方法，即将滤波范围内的数据进行排序，从大到小的顺序去除d个数据，从小到大的顺序去除d个数据，将剩下的数据计算均值。这样的滤波器，很擅长去除椒盐噪声与其他类型噪声一起污染过的图片。

在实验11中，在噪声密度不是很大的情况下（根据经验，噪声的出现的概率小于0.2），使用中值滤波的效果不错。但是当噪声出现的概率比较高时，原来的中值滤波算法就不是很有效了。只有增大滤波器窗口尺寸，尽管会使图像变得模糊。使用自适应中值滤波器的目的就是，根据预设好的条件，动态地改变中值滤波器的窗口尺寸，以同时兼顾去噪声作用和保护细节的效果。

在实验12中，将以前对灰度图像的处理技术，迁移到了彩色图像中，对于RGB图像，MATLAB的图像处理函数是可以直接运算的，这是通过向量运算实现的，RGB图像有R,G,B三个灰度图，可以通过向量一起进行锐化、平滑等操作；而对于HIS图像则略有不同，需要自己先将RGB图像转换到HIS各式，然后对分量处理，值得注意的是，通常来说不能对HIS的三个分量都进行处理，比如在这里处理H（色调）分量后图像明显出现了一些之前没有的颜色，经常处理I（亮度）分量，然后将HIS格式转回RGB格式输出显示。