Image Processing

Image Restoration (Part I)

Pattern Recognition and Image Processing Laboratory (Since 2012)

Introduction

Restoration attempts to reconstruct or recover an image that has been degraded by using a priori knowledge of the degradation phenomenon.

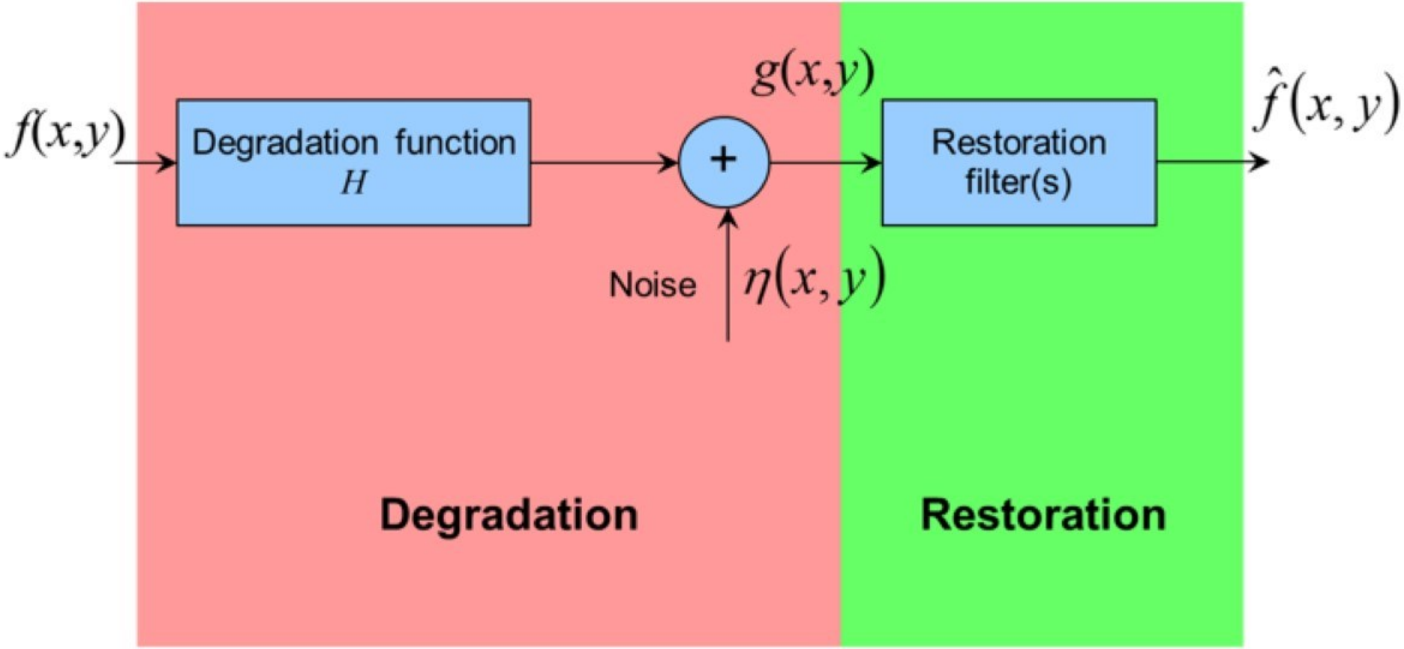
Introduction

… Thus, restoration techniques are oriented toward modeling the degradation and applying the inverse process in order to recover the original image.

Introduction

Degradation Image

A Model of the Image Degradation/ Restoration Process



The more we know about H and $\eta(x,y)$, the closer $\hat{f}(x,y)$ will be to $f(x,y)$.

A Model of the Image Degradation/ Restoration Process

In the **spatial domain**, the degraded image is given by

$$g(x,y)=h(x,y)*f(x,y)+\eta(x,y)$$

Frequency domain

$$G(u,v)=H(u,v)F(u,v)+N(u,v)$$

Noise Models

Two types of noise models:

- Noise in spatial domain
- Noise in frequency domain

Noise Models

Adding noise with function imnoise

```
>> g = imnoise(f, type, parameters)
>> ex5_01   % See demonstration
```

Noise Models

Generating spatial random noise with a specified distribution

```
>> ex5_01   % See demonstration
```

Noise Models

Periodic Noise

Periodic noise in an image arises typically from electrical and/or electromechanical interference during image acquisition.

```
>> ex5_01   % See demonstration
```

Restoration in the Presence of Noise Only–Spatial Filtering

Spatial noise filters

```
>> ex_snf   % See demonstration
```

Restoration in the Presence of Noise Only–Spatial Filtering

Spatial noise filters

Arithmetic mean: $$A(a_1, a_2, \ldots, a_n) = \frac{1}{n}\sum_{i=1}^{n} a_i$$

Geometric mean: $$G(a_1, a_2, \ldots, a_n) = \left(\prod_{i=1}^{n} a_i\right)^{1/n} = \sqrt[n]{a_1 a_2 \cdots a_n}$$

Contraharmonic mean: $$C(x_1, x_2, \ldots, x_n) = \frac{\left(\frac{x_1^2 + x_2^2 + \cdots + x_n^2}{n}\right)}{\left(\frac{x_1 + x_2 + \cdots + x_n}{n}\right)},$$

PSF: Point Spread Function, a degradation function in a spatial domain.

Command Window

New to MATLAB? See resources for Getting Started.

```
>>  help spfilt
 spfilt performs linear and nonlinear spatial filtering
   F = spfilt(G, TYPE, M, N, PARAMETER) performs spatail filtering
   of image G using a type filter of size M-by-N. Valid calls to
   spfilt are as follows:

   F = spfilt(G, 'amean', M, N)     Arithmetic mean filtering.
   F = spfilt(G, 'gmean', M, N)     Geometric mean filtering.
   F = spfilt(G, 'hmean', M, N)     Harmonic mean filtering.
   F = spfilt(G, 'chrmean', M, N)    Contraharmonic mean filtering of
                                    order Q = 1.5.
   F = spfilt(G, 'median', M, N)    Median filtering.
   F = spfilt(G, 'max', M, N)       Max filtering.
   F = spfilt(G, 'min', M, N)       Min filtering.
   F = spfilt(G, 'midpoint', M, N) Midpoint filtering.
   F = spfilt(G, 'atrimmed', M, N) Alpha-trimmed mean filtering.
                                    Parameter D must be a nonnegative
                                    Even integer; its default value
                                    is D = 2.

   The default values when only G and TYPE are input are M = N = 3,
   Q = 1.5, and D = 2.

fx >>
```

Harmonic and Contraharmonic Filters

Harmonic mean filter

$$\hat{f}(x,y) = \frac{mn}{\sum_{(s,t)\in S_{xy}} \frac{1}{g(s,t)}}$$

Works well for salt noise but fails for pepper noise

Contraharmonic mean filter

mn = size of moving window

$$\hat{f}(x,y) = \frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

Positive Q is suitable for eliminating pepper noise.
Negative Q is suitable for eliminating salt noise.

Q = the filter order

For $Q = 0$, the filter reduces to an arithmetic mean filter.
For $Q = -1$, the filter reduces to a harmonic mean filter.

2. Understanding alpha-trimmed mean filter

Now let us see, how to get alpha-trimmed mean value in practice. The basic idea here is to order elements, discard elements at the beginning and at the end of the got ordered set and then calculate average value using the rest. For instance, let us calculate alpha-trimmed mean for the case, depicted in **fig. 1**.

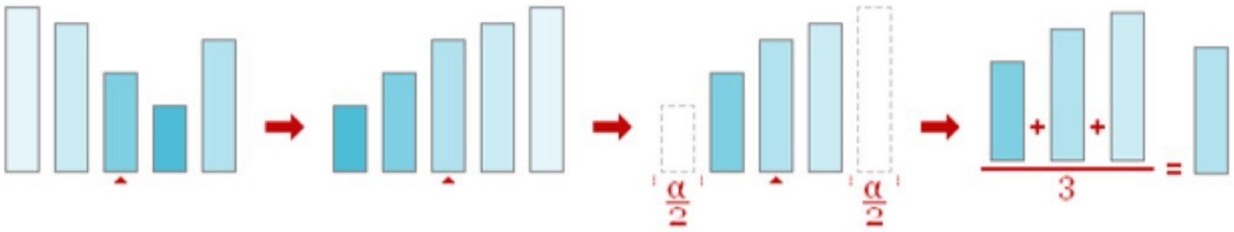


Fig. 1. Alpha-trimmed mean calculation.

Restoration in the Presence of Noise Only–Spatial Filtering

Adaptive spatial filters

```
Zmin = minimum intensity value in Sxy
Zmax = maximum intensity value in Sxy
Zmed = median of the intensity value in Sxy
Zxy = intensity value at coordinates (x, y)
Level A: If Zmin < Zmed < Zmax, go to level B
              Else increase the window size
         If window size <= Smax, repeat level A
              Else output Zmed
Level B: If Zmin < Zxy < Zmax, output Zxy
              Else output Zmed
```

Restoration in the Presence of Noise Only–Spatial Filtering

Adaptive spatial filters

```
>> ex_asf   % See demonstration
```

Periodic Noise Reduction by Frequency Domain Filtering

```
>> ex5_02   % See demonstration
```

The end of part I

```
% ex5_01.m

% Generating Spatial Random Noise
clear all
close all;

f = imread('lena.bmp');

M = 256;
N = 256;
a = 0.5;
b = 1;
R = a + sqrt(b*log(1-rand(M, N)));      % Rayleigh CDF
figure, imshow(uint8(abs(R)), []);      % Noise with a Specified
Distribution

pause;

clear all
close all;

C = [   0 64;
        0 128;
       32 32;
       64 0;
      128 0;
      -32 32]
[r, R, S] = imnoise3(512, 512, C);
figure, imshow(S, []);
figure, imshow(r, []);

pause;

clear all
close all;

C = [   0 32;
        0 64;
       16 16;
       32 0;
       64 0;
      -16 16]
[r, R, S] = imnoise3(512, 512, C);
figure, imshow(S, []);
figure, imshow(r, []);

pause;
```

```
clear all
close all;

C = [  6 32;
      -2 2];

[r, R, S] = imnoise3(512, 512, C);
figure, imshow(S, []);
figure, imshow(r, []);

pause;

clear all
close all;

C = [1 5];

[r, R, S] = imnoise3(512, 512, C);
figure, imshow(S, []);
figure, imshow(r, []);
```

```matlab
% ex_snf.m

% -----------------------------------------------------
% Demonstration of Spatial Noise Filters

f = imread('lena.bmp');
[M, N] = size(f);
R = imnoise2('salt & pepper', M, N, 0.1, 0); % corrupted by pepper noise.
c = find(R == 0);
gp = f;
gp(c) = 0;

figure(1); imshow(f);
figure(2); imshow(gp);

R = imnoise2('salt & pepper', M, N, 0, 0.1); % corrupted by salt noise.
c = find(R == 1);
gs = f;
gs(c) = 255;

figure(3); imshow(gs);

% Using a contraharmonic filter.

fp = spfilt(gp, 'chmean', 3, 3, 1.5);

figure(4); imshow(fp);

fs = spfilt(gs, 'chmean', 3, 3, -1.5);

figure(5); imshow(fs);

fpmax = spfilt(gp, 'max', 3, 3);
fsmin = spfilt(gs, 'min', 3, 3);

figure(6); imshow(fpmax);
figure(7); imshow(fsmin);
```

```matlab
% adpmedian.m

function f = adpmedian(g, Smax)
% ADPMEDIAN performs adaptive median filtering
% F = ADPMEDIAN(G, Smax) performs adaptive median filtering of
% image G. The median filter starts at size 3-by-3 and iterates up
% to size SMAX-by-SMAX. SMAX must be an odd integer greater than 1.

% SMAX must be an odd, positive integer greater than 1.
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

% Output zmed for remaining unprocessed pixels. Note that this zmed was
% computed using a window size Smax-by-Smax, which is the final value of k
% in the loop.
f(~alreadyProcessed) = zmed(~alreadyProcessed);
```

```
% ex_asf.m

% -------------------------------------------------------
% Demonstration of Adaptive Spatial Filters

f = imread('lena.bmp');
g = imnoise(f, 'salt & pepper', 0.20);  % corrupted by salt & pepper noise.
f1 = medfilt2(g, [3 3], 'symmetric');
f2 = adpmedian(g, 5);

figure(1); imshow(f);
figure(2); imshow(g);
figure(3); imshow(f1);
figure(4); imshow(f2);
```

```matlab
% ex5_02.m

clear all
close all

%f = imread('building.tif');
f = imread('lena.bmp');

%f = f(1:512,1:512);
figure(1), imshow(f);
sz = size(f);

C = [6 32];                        % assign locations of noise spectrum
[r, R, S] = imnoise3(sz(1), sz(2), C);
figure(2), imshow(r, []);          % show periodic noise in sptial domain
figure(3), imshow(S, []);          % show locations of noise spectrum

fn = (double(f) + (mat2gray(r).*255))./2;  % add noise into an image
figure(4), imshow(fn, []);

PQ = paddedsize(size(fn));
FN = fft2(fn, PQ(1), PQ(2));

FNs = fftshift(FN);
FNlog = log(1 + FNs);
figure(5), imshow(uint8(abs(FNlog)),[]);

[H D] = notchfilt('notch', sz(1), sz(2), 5, 8);
H1 = fftshift(H);
figure(6), imshow(uint8(H1.*255),[]);

g = dftfilt(fn, H);
figure(7), imshow(uint8(g),[]);
```

Image Processing

Workshop on Image Restoration (Part I)

Pattern Recognition and Image Processing Laboratory (Since 2012)

Workshop on Image Restoration (Part I)

1. ให้แสดงวิธีการคำนวณเพื่อกู้คืนภาพที่ถูกสัญญาณรบกวนดังรูปที่ 1.1 ด้วยอัลกอริธึมใน Slide#16 แล้วนำผลลัพธ์มาแสดงไว้ในรูปที่ 1.2

20	21	25	24	24
21	255	26	0	25
27	26	28	29	29
25	25	0	28	30
26	27	28	30	30

(1.1)

(1.2)

Workshop on Image Restoration (Part I)

2. ให้เขียน MATLAB Script เพื่อสร้างสัญญาณรบกวนแบบ Periodic Noise บน Frequency Domain อย่างน้อย 3 รูปแบบ

Workshop on Image Restoration (Part I)

3. ให้เขียน MATLAB Script เพื่อทำการ Add สัญญารบกวนจากข้อ 2 แต่ละรูปแบบลงในภาพ lena.bmp โดยกระทำบน Frequency Domain

Workshop on Image Restoration (Part I)

4. ให้เขียน MATLAB Script เพื่อกำจัดสัญญาณรบกวนในภาพ lena.bmp ในข้อ 2 โดยกระทำบน Frequency Domain