

Image processing for bioinformatics

Laboratory Deconvolution

1 Examples

1.1 Linear restoration: Wiener deconvolution (PSF known)

Code
<pre>1 %% deconvwnr Gaussian blur 2 I = im2double(imread('cameraman.tif')); 3 PSF = fspecial('gaussian', 10, 5); % Point-spread function 4 blurred = imfilter(I, PSF, 'conv', 'circular'); 5 wnr0 = deconvwnr(blurred, PSF, 0);</pre>
Image

Table 1: Gaussian blur

Code
<pre>1 %% deconvwnr Gaussian blur + noise 2 I = im2double(imread('cameraman.tif')); 3 4 PSF = fspecial('gaussian', 10, 5); 5 blurred = imfilter(I, PSF, 'conv', 'circular'); 6 7 noise_mean = 0; noise_var = 0.00001; 8 blurred_noisy = innoise(blurred, 'gaussian', noise_mean, noise_var); 9 10 wnr0 = deconvwnr(blurred_noisy, PSF, 0); 11 k = 0.001; 12 wnrK = deconvwnr(blurred_noisy, PSF, k);</pre>
Image

Table 2: Gaussian blur + noise

Code
<pre> 1 %% deconvwnr motion blur 2 I = im2double(imread('cameraman.tif')); 3 4 LEN = 25; 5 THETA = 25; 6 PSF = fspecial('motion', LEN, THETA); 7 blurred = imfilter(I, PSF, 'conv', 'circular'); 8 9 wnr0 = deconvwnr(blurred, PSF, 0); </pre>
Image

Table 3: Motion blur

Code
<pre> 1 %% deconvwnr motion blur + noise 2 rng default; 3 I = im2double(imread('cameraman.tif')); 4 5 LEN = 25; 6 THETA = 25; 7 PSF = fspecial('motion', LEN, THETA); 8 blurred = imfilter(I, PSF, 'conv', 'circular'); 9 10 noise_mean = 0; 11 noise_var = 0.0001; 12 blurred_noisy = imnoise(blurred, 'gaussian', noise_mean, noise_var); 13 14 wnr0 = deconvwnr(blurred_noisy, PSF, 0); 15 k = 0.005; 16 wnrK = deconvwnr(blurred_noisy, PSF, k); </pre>
Image

Table 4: Motion blur + noise

1.2 Linear restoration: Wiener deconvolution (PSF empirical)

Code
<pre> 1 %% deconvwnr Gaussian PSF empirical 2 I = im2double(rgb2gray(imread('saturn.png'))); 3 I = I(600:1000,1:600); 4 5 noise_mean = 0; 6 noise_var = 10; 7 8 PSF = fspecial('gaussian', 10, noise_var); 9 blurred = imfilter(I, PSF, 'conv', 'circular'); 10 11 PSF_emp = blurred(88:99,153:164); % White point 12 PSF_emp = PSF_emp / sum(PSF_emp(:)); 13 14 wnr2 = deconvwnr(blurred, PSF, 0); 15 wnr2_emp = deconvwnr(blurred, PSF_emp, 0.001); </pre>
Image
<p>The figure displays five sub-images illustrating the restoration process. The top row shows the 'Original Image' (a grayscale image of Saturn), the 'Blurred Image' (the original image convolved with a Gaussian PSF), the 'Restored Image PSF original' (the blurred image restored using the original PSF), and the 'Point-spread function (PSF) original' (a 10x10 Gaussian PSF). The bottom row shows the 'Restored Image PSF empirical from star' (the blurred image restored using an empirical PSF derived from a star in the blurred image) and the 'Point-spread function (PSF) empirical' (a 12x12 empirical PSF derived from a star in the blurred image).</p>

Table 5: PSF empirical, Gaussian blur

2 Assignment

1. Implement the direct inverse filter

$$\hat{F}(u, v) = \frac{G(u, v)}{H(u, v) + K}$$

where:

- \hat{F} and G are the Fourier Transform (FT) of the restored and degraded image respectively,
- H is the Point-spread function (PSF) in frequency domain
- and K is a value that depends on the amount of noise.

To compute H you can use the Matlab function *psf2otf*.

$$H = \text{psf2otf}(\text{PSF}, \text{size}(\text{Img}));$$

2. Implement Wiener deconvolution

$$\hat{F}(u, v) = \left[\frac{1}{H(u, v)} \cdot \frac{|H(u, v)|^2}{|H(u, v)|^2 + K} \right] G(u, v)$$

where:

- \hat{F} and G are the Fourier Transform (FT) of the restored and degraded image respectively,
- H is the Point-spread function (PSF) in frequency domain,
- $|H(u, v)|^2 = H^*(u, v)H(u, v)$, with $H^*(u, v)$ complex conjugate of $H(u, v)$
- and K is a value that depends on the amount of noise.



Table 6: Wiener deconvolution

3. Apply a deconvolution function (Matlab or implemented) to restore the following images [*plate0{1,2,3}.png*]:



Table 7: Plates of cars

3 Solutions

1. Implement the direct inverse filter
2. Implement Wiener deconvolution

Code - Direct inverse filter and Wiener deconvolution

```
1 %% Direct inverse filter and Wiener deconvolution
2 clc; clear; close all;
3 rng default;
4 I = im2double(imread('cameraman.tif'));
5
6 LEN = 25; THETA = 0; PSF = fspecial('motion', LEN, THETA);
7
8 blurred = imfilter(I, PSF, 'conv', 'circular');
9 noise_mean = 0;
10 noise_var = 0.00001;
11 blurred_noisy = imnoise(blurred, 'gaussian', noise_mean, noise_var);
12
13 [m,n] = size(blurred_noisy);
14 % Compute H so that it has the same size as I (FT)
15 H = psf2otf(PSF, [m,n]); % Convert point-spread function to optical transfer function (FT)
16 denom = abs(H);
17 % Make sure that denominator is not 0 anywhere.
18 denom = max(denom, sqrt(eps));
19 % Apply the filter G in the frequency domain.
20 K = 0.1;
21 wnr0 = ifft2(fft2(blurred_noisy)./(denom + K));
22
23 % Compute the Wiener restoration filter:%
24 %  $F(k,l) = H^*(k,l) / |H(k,l)|^2 + K$ 
25 % where K is empirically estimated .
26
27 K = 0.001;
28
29 % Compute the denominator of G in pieces.
30 denom = conj(H).*H + K;
31
32 % Make sure that denominator is not 0 anywhere.
33 denom = max(denom, sqrt(eps));
34
35 G = conj(H) ;
36 G = G ./ denom;
37
38 % Apply the filter G in the frequency domain.
39 wnr1 = ifft2(G .* fft2(blurred_noisy));
40
41 PSFzeros = zeros(size(PSF)+4); PSFzeros(3:end-2,3:end-2) = PSF;
42
43 nr = 2; nc = 4;
44 subplot(nr, nc, 1); imshow(I); title('Original Image');
45 subplot(nr, nc, 2); imshow(PSFzeros, [min(PSFzeros(:)), max(PSFzeros(:))]); title('Point-spread
    function (PSF)');
46 subplot(nr, nc, 3); imshow(blurred_noisy); title('Blurred and noisy Image');
47 subplot(nr, nc, 4); imshow(wnr1); title('Restored Image (Wiener filter)');
48 subplot(nr, nc, 6); imshow(H, [min(H(:)), max(H(:))]); title('psf2otf optical transfer function');
49 subplot(nr, nc, 8); imshow(wnr0); title('Restored Image (IMG/H)');
```

3. Apply a deconvolution function (Matlab or implemented) to restore the following images [*plate0{1,2,3}.png*]:

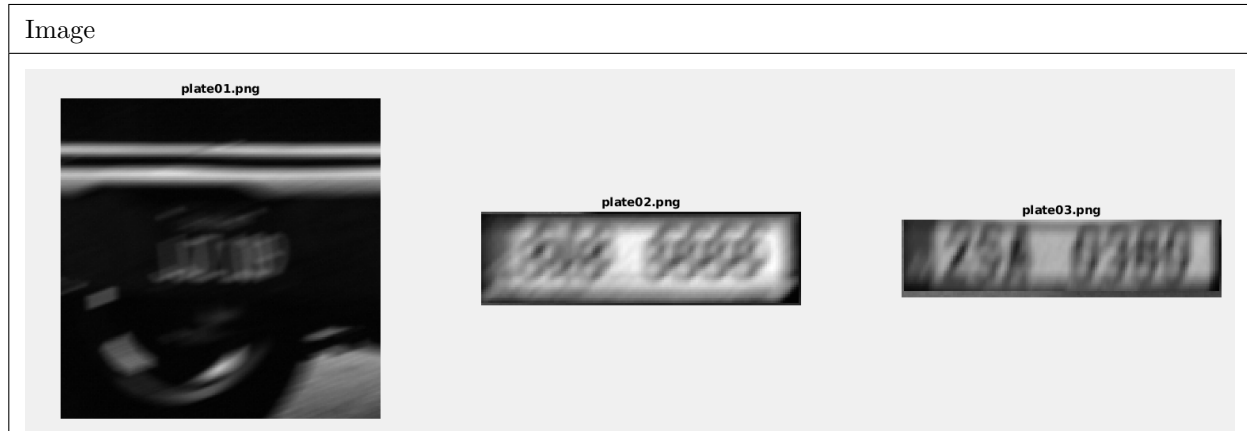


Table 8: Plates of cars

Code - Brute-force search LEN and THETA of motion kernel

```

1 %% Brute-force search LEN and THETA
2 clc; clear; close all;
3 I = im2double(imread('plate03.png'));
4 k = 0.02;
5 figure;
6 nr = 11;
7 nc = 11;
8 cont = 0;
9 for i=9:19 % LEN
10     for j = 40:5:90 % THETA
11         cont = cont + 1;
12         PSF = fspecial('motion', i, j); % LEN and THETA
13         wnrK = deconvwnr(I, PSF, k);
14         subplot(nr, nc, cont); imshow(wnrK); title([num2str(i), ' ', num2str(j)])
15     end
16 end

```

Code - Plates image configuration

```

1 %% Plates image configuration
2 clc; clear; close all;
3 rng default;
4 I_1 = im2double((imread('plate01.png')));
5 LEN = 20;
6 THETA = 20;
7 PSF = fspecial('motion', LEN, THETA);
8 wnr0_1 = deconvwnr(I_1, PSF, 0);
9 k_1 = 0.005;
10 wnrK_1 = deconvwnr(I_1, PSF, k_1);
11 PSFzeros_1 = zeros(size(PSF)+4); PSFzeros_1(3:end-2,3:end-2) = PSF;
12
13 I_2 = im2double((imread('plate02.png')));
14 LEN = 14;
15 THETA = 40;
16 PSF = fspecial('motion', LEN, THETA);
17 wnr0_2 = deconvwnr(I_2, PSF, 0);
18 k_2 = 0.01;
19 wnrK_2 = deconvwnr(I_2, PSF, k_2);
20 PSFzeros_2 = zeros(size(PSF)+4); PSFzeros_2(3:end-2,3:end-2) = PSF;
21
22 I_3 = im2double((imread('plate03.png')));
23 LEN = 14;
24 THETA = 70;
25 PSF = fspecial('motion', LEN, THETA);
26 wnr0_3 = deconvwnr(I_3, PSF, 0);
27 k_3 = 0.02;
28 wnrK_3 = deconvwnr(I_3, PSF, k_3);
29 PSFzeros_3 = zeros(size(PSF)+4); PSFzeros_3(3:end-2,3:end-2) = PSF;
30
31
32 nr = 3;
33 nc = 4;
34 subplot(nr, nc, 1); imshow(I_1); title('Original Image');
35 subplot(nr, nc, 2); imshow(PSFzeros_1, [min(PSFzeros_1(:)), max(PSFzeros_1(:))]);
36 title('Point-spread function (PSF)');
37 subplot(nr, nc, 3); imshow(wnr0_1); title('Restored Image k=0');
38 subplot(nr, nc, 4); imshow(wnrK_1); title(['Restored Image k=', num2str(k_1)])
39
40 subplot(nr, nc, 1+4); imshow(I_2); title('Original Image');
41 subplot(nr, nc, 2+4); imshow(PSFzeros_2, [min(PSFzeros_2(:)), max(PSFzeros_2(:))]);
42 title('Point-spread function (PSF)');
43 subplot(nr, nc, 3+4); imshow(wnr0_2); title('Restored Image k=0');
44 subplot(nr, nc, 4+4); imshow(wnrK_2); title(['Restored Image k=', num2str(k_2)])
45
46 subplot(nr, nc, 1+8); imshow(I_3); title('Original Image');
47 subplot(nr, nc, 2+8); imshow(PSFzeros_3, [min(PSFzeros_3(:)), max(PSFzeros_3(:))]);
48 title('Point-spread function (PSF)');
49 subplot(nr, nc, 3+8); imshow(wnr0_3); title('Restored Image k=0');
50 subplot(nr, nc, 4+8); imshow(wnrK_3); title(['Restored Image k=', num2str(k_3)])

```