Image processing for bioinformatics

Laboratory
Deconvolution

1 Examples

1.1 Linear restoration: Wiener deconvolution (PSF known)

Code

```
%% deconvwnr Gaussian blur
I = im2double(imread(’cameraman.tif’));
PSF = fspecial(’gaussian’, 10, 5); % Point−spread function
blurred = imfilter(I, PSF, ’conv’, ’circular’);
wnr0 = deconvwnr(blurred, PSF, 0);
```

Image

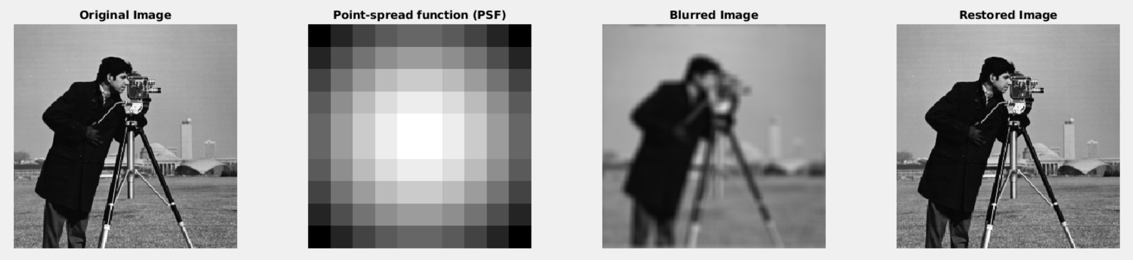


Table 1: Gaussian blur

Code

```
%% deconvwnr Gaussian blur + noise
I = im2double(imread(’cameraman.tif’));

PSF = fspecial(’gaussian’, 10, 5);
blurred = imfilter(I, PSF, ’conv’, ’circular’);

noise_mean = 0; noise_var = 0.00001;
blurred_noisy = imnoise(blurred, ’gaussian’, noise_mean, noise_var);

wnr0 = deconvwnr(blurred_noisy, PSF, 0);
k = 0.001;
wnrK = deconvwnr(blurred_noisy, PSF, k);
```

Image

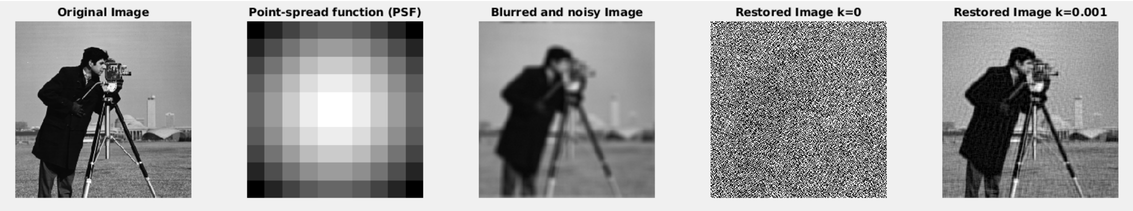


Table 2: Gaussian blur + noise

Code

```
%% deconvwnr motion blur
I = im2double(imread('cameraman.tif'));

LEN = 25;
THETA = 25;
PSF = fspecial('motion', LEN, THETA);
blurred = imfilter(I, PSF, 'conv', 'circular');

wnr0 = deconvwnr(blurred, PSF, 0);
```

Image

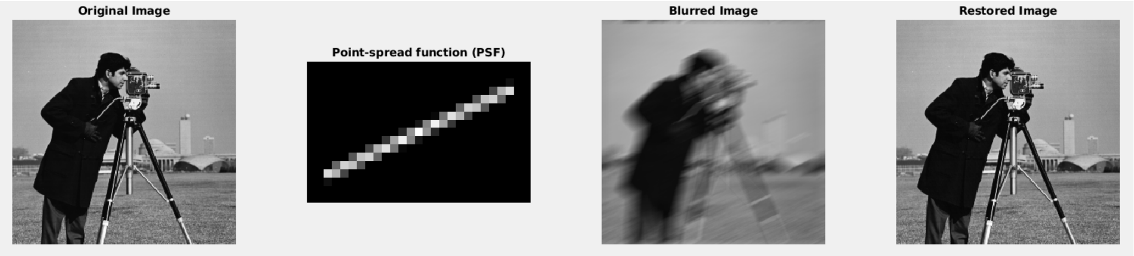


Table 3: Motion blur

Code

```
%% deconvwnr motion blur + noise
rng default;
I = im2double(imread('cameraman.tif'));

LEN = 25;
THETA = 25;
PSF = fspecial('motion', LEN, THETA);
blurred = imfilter(I, PSF, 'conv', 'circular');

noise_mean = 0;
noise_var = 0.0001;
blurred_noisy = imnoise(blurred, 'gaussian', noise_mean, noise_var);

wnr0 = deconvwnr(blurred_noisy, PSF, 0);
k = 0.005;
wnrK = deconvwnr(blurred_noisy, PSF, k);
```

Image

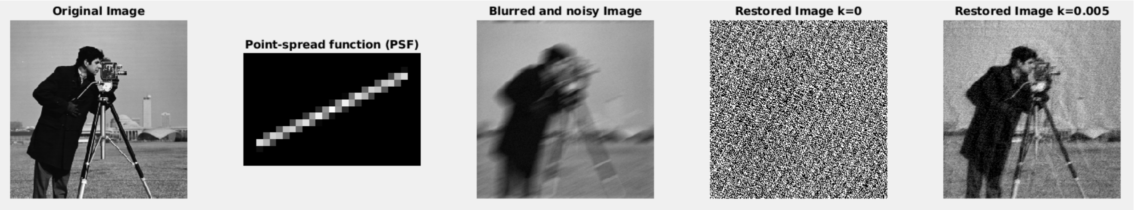


Table 4: Motion blur + noise

1.2 Linear restoration: Wiener deconvolution (PSF empirical)

Code

```
%% deconvwnr Gaussian PSF empirical
I = im2double(rgb2gray(imread(’saturn.png’)));
I = I(600:1000,1:600);

noise_mean = 0;
noise_var = 10;

PSF = fspecial(’gaussian’, 10, noise_var);
blurred = imfilter(I, PSF, ’conv’, ’circular’);

PSF_emp = blurred(88:99,153:164); % White point
PSF_emp = PSF_emp / sum(PSF_emp(:));

wnr2 = deconvwnr(blurred, PSF, 0);
wnr2_emp = deconvwnr(blurred, PSF_emp, 0.001 );
```

Image

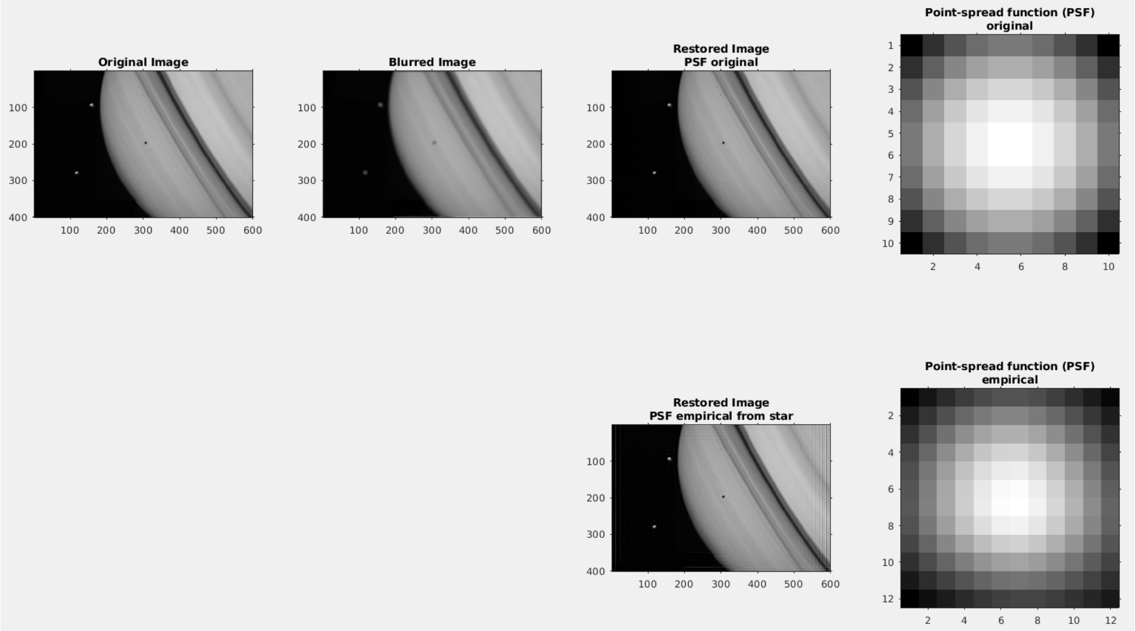


Table 5: PSF empirical, Gaussian blur

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 = psf2otf(PSF, size(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.

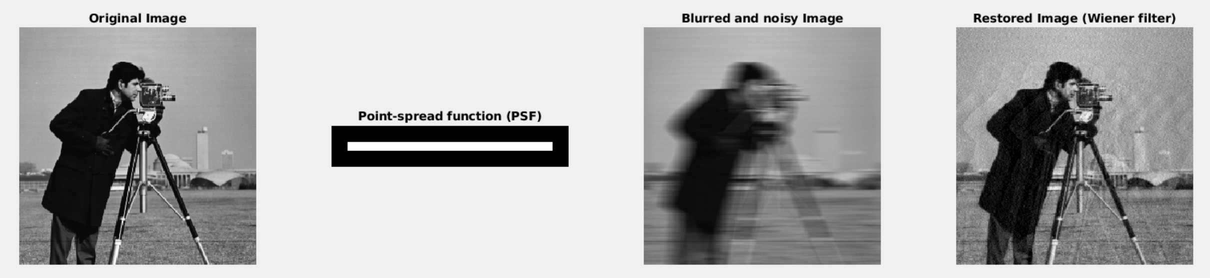
Image
 Original Image Point-spread function (PSF) Blurred and noisy Image Restored Image (Wiener filter)

Table 6: Wiener deconvolution

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

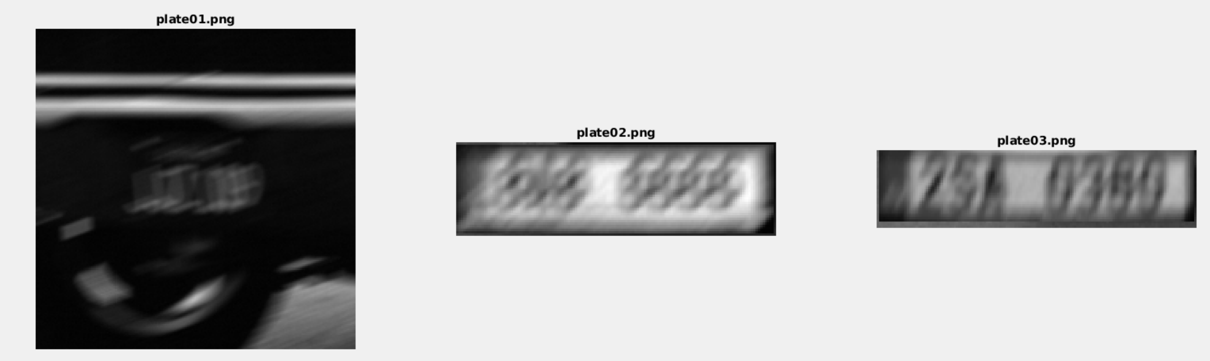
Image


Table 7: Plates of cars

3 Solutions

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

Code - Direct inverse filter and Wiener deconvolution

```matlab
%% Direct inverse filter and Wiener deconvolution
clc; clear; close all;
rng default;
I = im2double(imread('cameraman.tif'));

LEN = 25; THETA = 0; PSF = fspecial('motion', LEN, THETA);

blurred = imfilter(I, PSF, 'conv', 'circular');
noise_mean = 0;
noise_var = 0.00001;
blurred_noisy = imnoise(blurred, 'gaussian', noise_mean, noise_var);

[m,n] = size(blurred_noisy);
% Compute H so that it has the same size as I (FT)
H = psf2otf(PSF, [m,n]); % Convert point-spread function to optical transfer function (FT)
denom = abs(H);
% Make sure that denominator is not 0 anywhere.
denom = max(denom, sqrt(eps));
% Apply the filter G in the frequency domain.
K = 0.1;
wnr0 = ifft2(fft2(blurred_noisy)./(denom + K));

% Compute the Wiener restoration filter:%
% F(k,l) = H*(k,l) / |H(k,l)|^2 + K
% where K is empiricaly estimated .

K = 0.001;

% Compute the denominator of G in pieces.
denom = conj(H).*H + K;

% Make sure that denominator is not 0 anywhere.
denom = max(denom, sqrt(eps));

G = conj(H) ;
G = G ./ denom;

% Apply the filter G in the frequency domain.
wnr1 = ifft2(G .* fft2(blurred_noisy));

PSFzeros = zeros(size(PSF)+4); PSFzeros(3:end-2,3:end-2) = PSF;

nr = 2; nc = 4;
subplot(nr, nc, 1); imshow(I); title('Original Image');
subplot(nr, nc, 2); imshow(PSFzeros, [min(PSFzeros(:)), max(PSFzeros(:))]); title('Point-spread function (PSF)');
subplot(nr, nc, 3); imshow(blurred_noisy); title('Blurred and noisy Image');
subplot(nr, nc, 4); imshow(wnr1); title('Restored Image (Wiener filter)');
subplot(nr, nc, 6); imshow(H, [min(H(:)), max(H(:))]); title('psf2otf optical transfer function');
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*]:

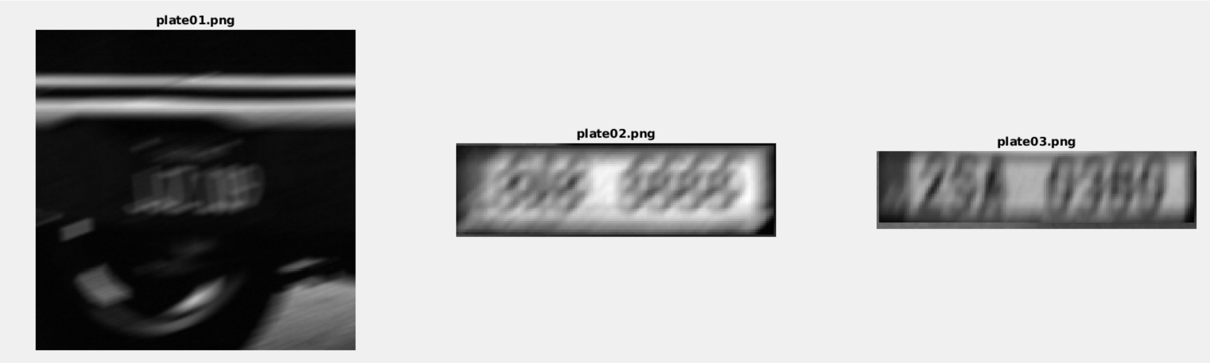
Image


Table 8: Plates of cars

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

```matlab
%% Brute−force search LEN and THETA
clc; clear; close all;
I = im2double((imread('plate03.png')));
k = 0.02;
figure;
nr = 11;
nc = 11;
cont = 0;
  for i=9:19 % LEN
    for j = 40:5:90 % THETA
      cont = cont + 1;
      PSF = fspecial('motion', i, j); % LEN and THETA
      wnrK = deconvwnr(I, PSF, k);
      subplot(nr, nc, cont); imshow(wnrK); title([num2str(i),' ',num2str(j)])
    end
  end
```

Code - Plates image configuration

```
%% Plates image configuration
clc; clear; close all;
rng default;
I_1 = im2double((imread('plate01.png')));
LEN = 20;
THETA = 20;
PSF = fspecial('motion', LEN, THETA);
wnr0_1 = deconvwnr(I_1, PSF, 0);
k_1 = 0.005;
wnrK_1 = deconvwnr(I_1, PSF, k_1);
PSFzeros_1 = zeros(size(PSF)+4); PSFzeros_1(3:end-2,3:end-2) = PSF;

I_2 = im2double((imread('plate02.png')));
LEN = 14;
THETA = 40;
PSF = fspecial('motion', LEN, THETA);
wnr0_2 = deconvwnr(I_2, PSF, 0);
k_2 = 0.01;
wnrK_2 = deconvwnr(I_2, PSF, k_2);
PSFzeros_2 = zeros(size(PSF)+4); PSFzeros_2(3:end-2,3:end-2) = PSF;

I_3 = im2double((imread('plate03.png')));
LEN = 14;
THETA = 70;
PSF = fspecial('motion', LEN, THETA);
wnr0_3 = deconvwnr(I_3, PSF, 0);
k_3 = 0.02;
wnrK_3 = deconvwnr(I_3, PSF, k_3);
PSFzeros_3 = zeros(size(PSF)+4); PSFzeros_3(3:end-2,3:end-2) = PSF;


nr = 3;
nc = 4;
subplot(nr, nc, 1); imshow(I_1); title('Original Image');
subplot(nr, nc, 2); imshow(PSFzeros_1, [min(PSFzeros_1(:)), max(PSFzeros_1(:))]);
title('Point-spread function (PSF)');
subplot(nr, nc, 3); imshow(wnr0_1); title('Restored Image k=0');
subplot(nr, nc, 4); imshow(wnrK_1); title(['Restored Image k=',num2str(k_1)])

subplot(nr, nc, 1+4); imshow(I_2); title('Original Image');
subplot(nr, nc, 2+4); imshow(PSFzeros_2, [min(PSFzeros_2(:)), max(PSFzeros_2(:))]);
title('Point-spread function (PSF)');
subplot(nr, nc, 3+4); imshow(wnr0_2); title('Restored Image k=0');
subplot(nr, nc, 4+4); imshow(wnrK_2); title(['Restored Image k=',num2str(k_2)])

subplot(nr, nc, 1+8); imshow(I_3); title('Original Image');
subplot(nr, nc, 2+8); imshow(PSFzeros_3, [min(PSFzeros_3(:)), max(PSFzeros_3(:))]);
title('Point-spread function (PSF)');
subplot(nr, nc, 3+8); imshow(wnr0_3); title('Restored Image k=0');
subplot(nr, nc, 4+8); imshow(wnrK_3); title(['Restored Image k=',num2str(k_3)])
```