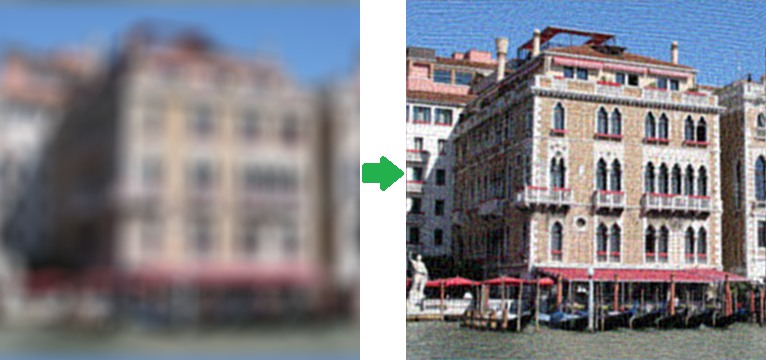
# Restoration of defocused and blurred images



Introduction

Let's demonstrate it using a small example for a one-dimensional case - let's suppose we have a row of pixels with the following values:  
x1 | x2 | x3 | x4... - Source image

After blurring the value of each pixel is added to the value of the left one: x'i = xi + xi-1. Normally, it is also required to divide it by 2, but we will drop it out for simplicity. As a result we have a blurred image with the following pixel values:  
x1 + x0 | x2 + x1 | x3 + x2 | x4 + x3... - Blurred image

Now we will try to restore it, we will consequentially subtract values according to the following scheme - the first pixel from the second one, the result of the second pixel from the third one, the result of the third pixel from the fourth one and so on, and we will get the following:  
x1 + x0 | x2 - x0 | x3 + x0 | x4 - x0... - Restored image

Blurring process model

*f(x, y)* - source image (non-blurred)  
*h(x, y)* - blurring function  
*n(x, y)* - additive noise  
*g(x, y)* - blurring result image

We will form the blurring process model in the following way:  
*g(x, y) = h(x, y) \* f(x, y) + n(x, y)*(1)

The task of restoration of a blurred image consists in finding the best approximation *f'(x, y)* to the source image

Blurring functions

|  |  |  |
| --- | --- | --- |
|  | |  |
|  |  |

Inverse filter

Here we are tempted to divide this equation by *H(u, v)* and get the following evaluation *F^(u, v)* of the source image:

|  |  |
| --- | --- |
|  |  |

So, we take the source image,  


convert it into a grayscale one, using Matlab, and get the spectrum:

% Load image

I = imread('image\_src.png');

figure(1);

imshow(I);

title('Source image');

% Convert image into grayscale

I = rgb2gray(I);

% Compute Fourier Transform and center it

fftRes = fftshift(fft2(I));

% Show result

figure(2);

imshow(mat2gray(log(1+abs(fftRes)))); title('FFT - amplitude spectrum (log scale)');

figure(3); imshow(mat2gray(angle(fftRes))); title('FFT - phase smectrum');

% Load image

I = im2double(imread('image\_src.png'));

figure(1); imshow(I); title('Source image');

% Blur image

Blurred = imfilter(I, PSF,'circular','conv' );

figure(2); imshow(Blurred); title('Blurred image');

% Add noise

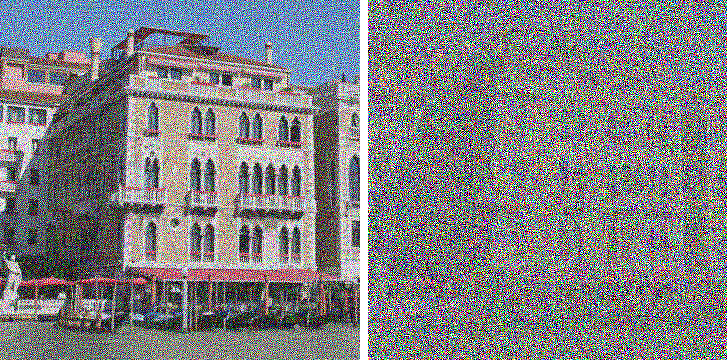
noise\_mean = 0;

noise\_var = 0.0;

Blurred = imnoise(Blurred, 'gaussian', noise\_mean, noise\_var);

% Deconvolution

figure(3); imshow(deconvwnr(Blurred, PSF, 0)); title('Result');



|  |  |
| --- | --- |
| **noise\_var = 0.0000001** | **noise\_var = 0.000005** |

Existing approaches to deconvolution

|  |  |
| --- | --- |
|  |  |

This result was found by Wiener

The next method is "Constrained Least Squares Filtering", other names: "Tikhonov filtration", "Tikhonov regularization".

|  |  |
| --- | --- |
|  |  |

Another interesting approach was offered by Richardson (1972 year) and Lucy independently (1974 year), so this approach is called as method Lucy-Richardson

|  |  |
| --- | --- |
|  |  |



% Load image

I = im2double(imread('image\_src.png'));

figure(1); imshow(I); title('Source image');

% Blur image

PSF = fspecial('disk', 15);

Blurred = imfilter(I, PSF,'circular','conv' );

% Add noise

noise\_mean = 0;

noise\_var = 0.00001;

Blurred = imnoise(Blurred, 'gaussian', noise\_mean, noise\_var);

figure(2); imshow(Blurred); title('Blurred image');

estimated\_nsr = noise\_var / var(Blurred(:));

% Restore image

figure(3), imshow(deconvwnr(Blurred, PSF, estimated\_nsr)), title('Wiener');

figure(4); imshow(deconvreg(Blurred, PSF)); title('Regul');

figure(5); imshow(deconvblind(Blurred, PSF, 100));

title('Blind');

figure(6); imshow(deconvlucy(Blurred, PSF, 100));

title('Lucy');

Results:

|  |  |
| --- | --- |
|  |  |
| **Wiener filter** | **Tikhonov regularization** |

|  |  |
| --- | --- |
|  |  |
| **Lucy-Richardson filter** | **Blind deconvolution** |