Image Processing 1

Vinay P. Namboodiri

January 27, 2019

1. Point Processing
   - Intensity Transformation
   - Histogram Equalization
   - Neighborhood operations
2. Linear Filtering
   - Separable Filtering
   - Band-pass and steerable filters
   - Summed area table
3. Non-Linear Filtering
   - Median filtering
   - Bilateral Filtering
   - Non-local means
4. Fourier transform

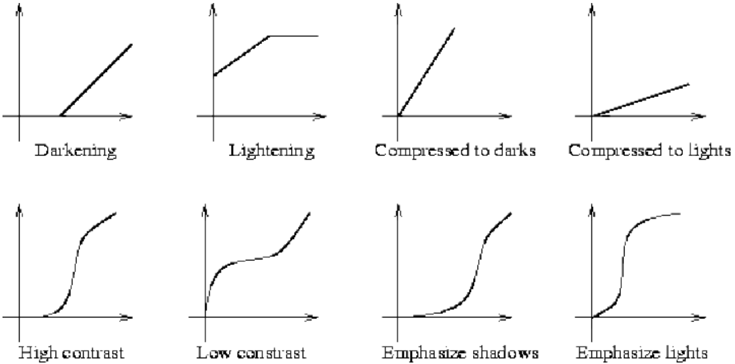
Input

Point Operations on Images

Point mapping operator defined by

$$s = M(r) \tag{1}$$

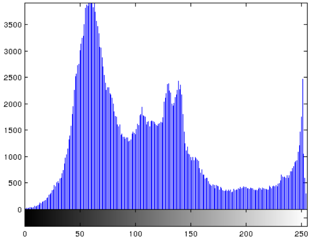
where s is destination pixel intensity value and r is source pixel intensity value



Histogram

It is a discrete probability distribution of the image intensity values

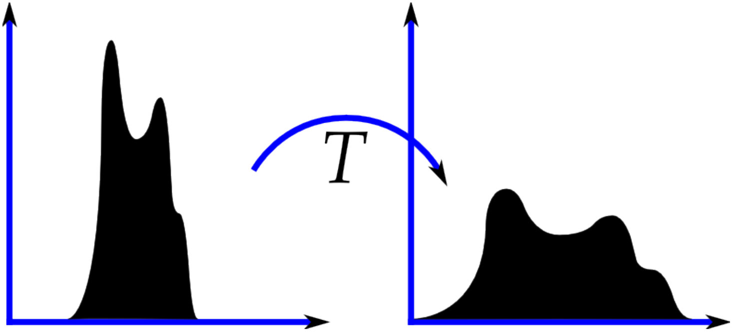
(a) Image



(b) Histogram

Histogram Equalization

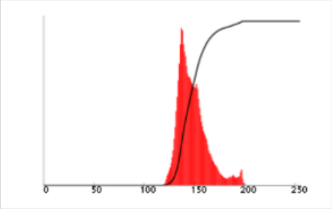
One method to enhance images is to equalize the histogram



More generally, histogram can be modified by histogram specification

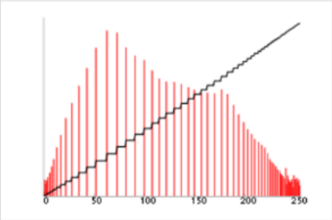
Example of Histogram Equalization

An unequalized image



Corresponding histogram (red) and cumulative histogram (black)

The same image after histogram equalization

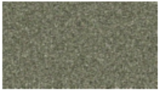


Corresponding histogram (red) and cumulative histogram (black)

Shuffling the pixels

What happens if we shuffle all the pixels in an image randomly?

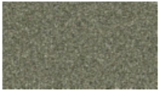


Shuffling the pixels

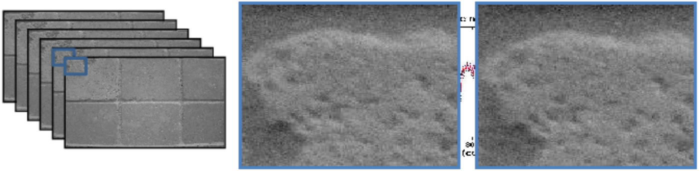
What happens if we shuffle all the pixels in an image randomly?



- Different image with same histogram
- Need more local operators

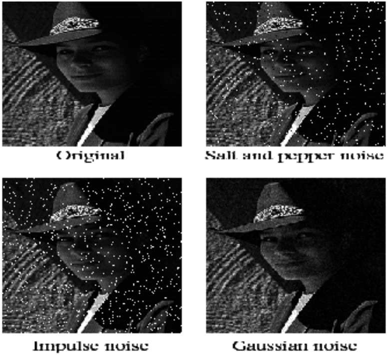
Motivation: Noise reduction



- We can measure **noise** in multiple images of the same static scene.
- How could we reduce the noise, i.e., give an estimate of the true intensities?

Types of Noise

- **Salt and pepper noise**: random occurrences of black and white pixels
- **Impulse noise:** random occurrences of white pixels
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution

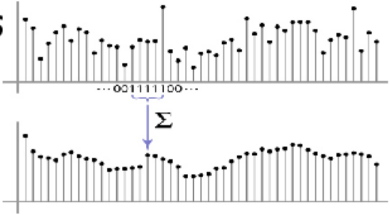


First attempt at a solution

- Let's replace each pixel with an average of all the values in its neighborhood
- Assumptions:
  - Expect pixels to be like their neighbors
  - Expect noise processes to be independent from pixel to pixel

Weighted moving average

- Can add weights to our moving average
- *Weights*



Local Linear Operator

Output pixel is a weighted sum of input pixel values

$$g(i,j) = \sum_{k,l} f(i+k, j+l)h(k,l). \tag{2}$$

Entries in kernel $h(k,l)$ are called the filter coefficients. Operator is termed *correlation* operator.

$$g = f \otimes h. \tag{3}$$

Local Linear Operator

Output pixel is a weighted sum of input pixel values

$$g(i,j) = \sum_{k,l} f(i+k, j+l)h(k,l). \tag{2}$$

Entries in kernel $h(k,l)$ are called the filter coefficients. Operator is termed *correlation* operator.

$$g = f \otimes h. \tag{3}$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

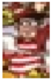
f(x,y)

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

g(x,y)

Correlation for template matching

Scene



Template

auman

Convolution Operator

Variant of correlation operator

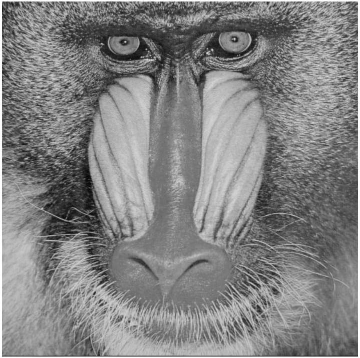
$$g(i,j) = \sum_{k,l} f(i-k, j-l)h(k,l). \tag{4}$$

$$g = f * h \tag{5}$$

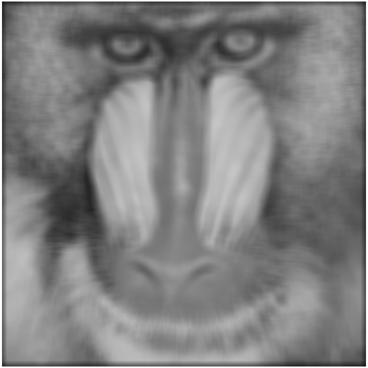
- Convolution of a kernel function h with an impulse signal δ results in the same kernel function whereas correlation reflects the kernel.
- Correlation and Convolution are identical when the filter is symmetric.
- Convolution is associative
- Correlation and Convolution are linear shift invariant operators f*(g*h) = (f*g)*h

Examples

Example of box filtering



(c)



(d)

Gaussian Filter

- What if we want nearest neighboring pixels to have the most influence on the output?

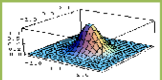
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	[illegible]	[illegible]	[illegible]	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x, y]$

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

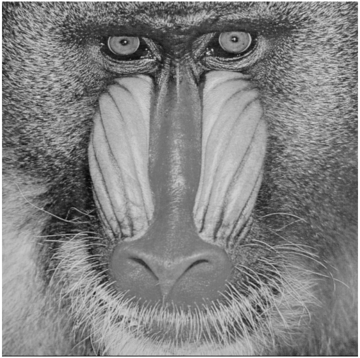
$H[u, v]$

This kernel is an approximation of a Gaussian function: $\frac{1}{2\pi\sigma^2}e^{-\frac{u^2+v^2}{\sigma^2}}$



Examples

Example of Gaussian filtering



(e)

(f)

Separable Filtering

- In some cases, the convolution operator can be speeded by separating the kernel into separate vertical and horizontal kernels

$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \end{bmatrix} \circ \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 \end{bmatrix}$$

Examples of Separable Filtering

$$\frac{1}{K^2}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline 1 & 1 & \cdots & 1 \\ \hline \vdots & \vdots & 1 & \vdots \\ \hline 1 & 1 & \cdots & 1 \\ \hline \end{array}$$

$$\frac{1}{K}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline \end{array}$$

(a) box, $K = 5$

$$\frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

$$\frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array}$$

(b) bilinear

$$\frac{1}{256}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 6 & 24 & 36 & 24 & 6 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array}$$

$$\frac{1}{16}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array}$$

(c) “Gaussian”

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

$$\frac{1}{2}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline \end{array}$$

(d) Sobel

$$\frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline -2 & 4 & -2 \\ \hline 1 & -2 & 1 \\ \hline \end{array}$$

$$\frac{1}{2}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline \end{array}$$

(e) corner

Band-pass filter

Gaussian kernel given by

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{6}$$

- It is a low pass filter
- Band-pass filters are obtained by taking derivative of Gaussian filter The second derivative of an image is the Laplacian operator given by

$$\nabla^2 f = \frac{\partial^f}{\partial x^2} + \frac{\partial^f}{\partial y^2} \tag{7}$$

Bandpass filter is obtained by Laplacian of Gaussian filter given by

$$\nabla^2 g(x, y; \sigma) = \left(\frac{x^2 + y^2}{\sigma^4} - \frac{2}{\sigma^2} \right) G(x, y; \sigma) \tag{8}$$

Steerable Filter

- Directional derivative is obtained by taking dot product between the derivative operator and a unit direction $\hat{\mathbf{u}} = (cos\theta, sin\theta)$

$$\hat{\mathbf{u}}\dot{\nabla}(G * f) = \nabla_{\hat{\mathbf{u}}}(G * f) = (\nabla_{\hat{\mathbf{u}}} G) * f \tag{9}$$

(a) (b) (c)

Figure 3.15 Second-order steerable filter (Freeman 1992) © 1992 IEEE: (a) original image of Einstein; (b) orientation map computed from the second-order oriented energy; (c) original image with oriented structures enhanced.

Summed area table

Summed area table can be precomputed. It is relevant when there are repeated convolutions with different box filters.

$$s(i,j) = \sum_{k=0}^{i} \sum_{l=0}^{j} f(k,l) \tag{10}$$

Can be efficiently computed using a recursive algorithm

$$s(i,j) = s(i-1,j) + s(i,j-1) - s(i-1,j-1) + f(i,j) \tag{11}$$

$s(i,j)$ is called an integral image

Median Filter

1	2	1	2	4
2	1	3	5	8
1	3	7	6	9
3	4	8	6	7
4	5	7	8	9

(a) median = 4

1	2	1	2	4
2	1	3	5	8
1	3	7	6	9
3	4	8	6	7
4	5	7	8	9

(b) α-mean= 4.6

- Extension of Median filter is to compute weighted median.
- Each pixel is used a number of times depending on its weight from the centre. ,j,k,l)

Obtained by minimizing the following objective function

$$\sum_{k,l} w(k,l)|f(i+k,j+l) - g(i,j)| \tag{12}$$

Bilateral Filtering

- It is so called because it combines locality in spatial domain and intensity domain

$$g(i,h) = \frac{\sum_{k,l} f(k,l)w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)} \tag{13}$$

where the weighting coefficient $w(i,j,k,l)$ is given by

$$w(i,j,k,l) = \exp\left(-\frac{(i-k)^2+(j-l)^2}{2\sigma_d^2} - \frac{||f(i,j)-f(k,l)||^2}{2\sigma_r^2}\right) \tag{14}$$

Example of Bilateral Filtering

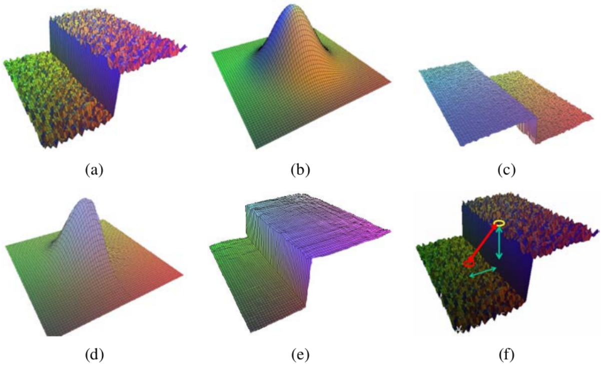


Figure 3.20 Bilateral filtering (Durand and Dorsey 2002) © 2002 ACM: (a) noisy step edge input; (b) domain filter (Gaussian); (c) range filter (similarity to center pixel value); (d) bilateral filter; (e) filtered step edge output; (f) 3D distance between pixels.

Non-local means

NL-means applies, to each pixel location, an adaptive averaging kernel that is computed from patch distances

$$D(i,j) = \frac{1}{n^2}||H_i - H_j||^2 \tag{15}$$

where $D(i,j)$ is the distance value, n is the number of pixels in a patch; H_i and H_j are patches in the image
Denoised image f is given by

$$f_i = \sum_j K_{i,j} f_j \tag{16}$$

where the weights K are computed as

$$\hat{K_{i,j}} = e^{\frac{D_{i,j}}{2\tau^2}} \text{ and } K_{i,j} = \frac{\hat{K}_{i,j}}{\sum_j' \hat{K_{i,j'}}} \tag{17}$$

Non-local means example

Figure 5. Denoising experience on a natural image. From left to right and from top to bottom: noisy image (standard deviation 20), Gauss filtering, anisotropic filtering, Total variation, Neighborhood filtering and NL-means algorithm. The removed details must be compared with the method noise experience, Figure 4.

Use of Filters

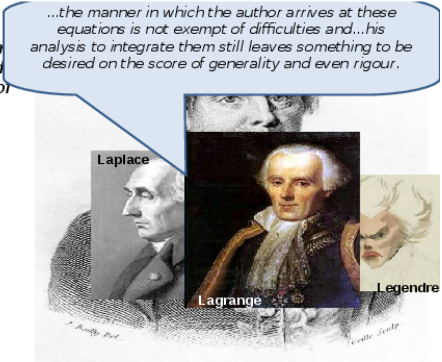
- We use filters for restoration purposes such as noise removal,
- for obtaining various features for higher level tasks such as recognition,
- and for tasks such as template matching

Jean Baptiste Joseph Fourier (1768-1830)

had crazy idea (1807):

> ***Any*** *univariate function ca be rewritten as a weighted sum of sines and cosines o different frequencies.*

- Don't believe it?
  - Neither did Lagrange, Laplace, Poisson and other big wigs
  - Not translated into English until 1878!
- But it's (mostly) true!
  - called Fourier Series
  - there are some subtle restrictions

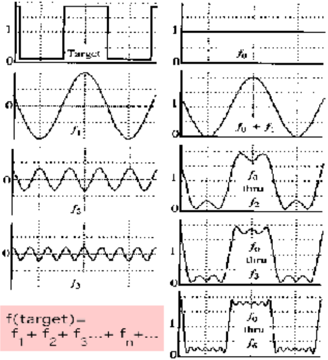


Example

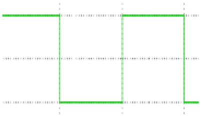
Our building block:

$$A\sin(\omega x + \phi)$$

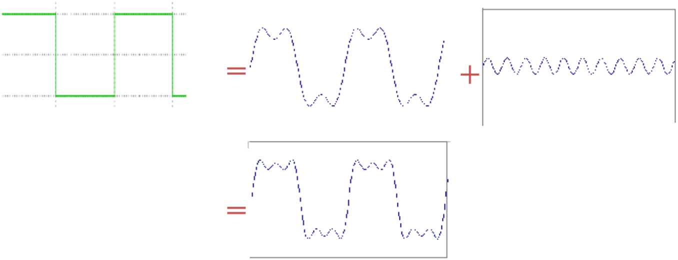
Add enough of them to get any signal *g(x)* you want!



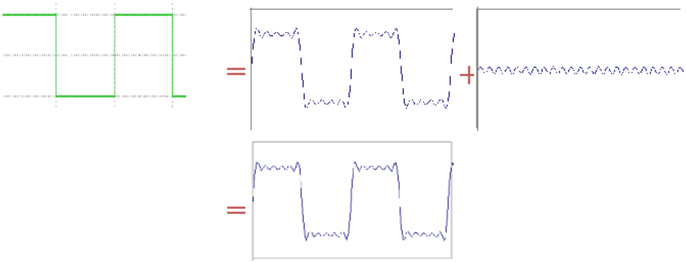
Example



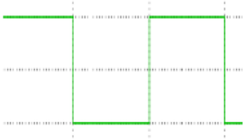
Example



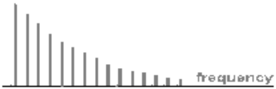
Example



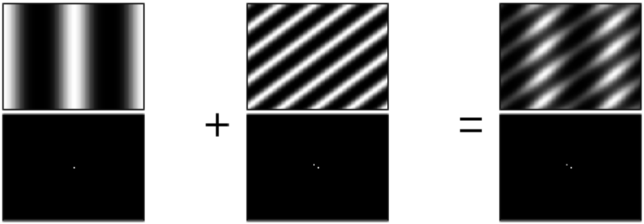
Example



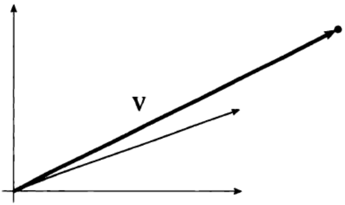
$$= \boxed{A\sum_{k=1}^{\infty}\frac{1}{k}\sin(2\pi kt)}$$



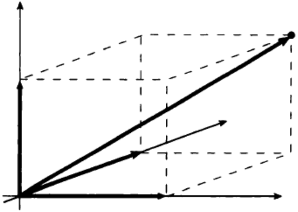
Example in image



Projection of points in Space

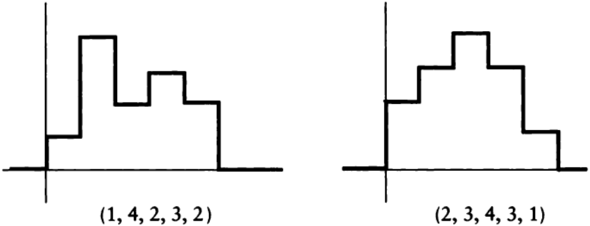


(a)



(b)

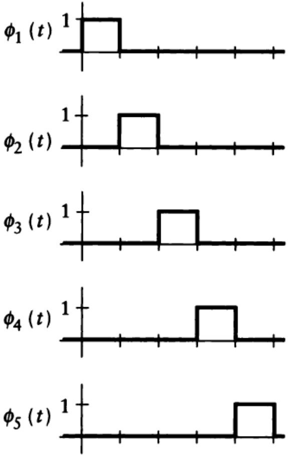
Projection of Functions



How to form the function in terms of basis functions?

$$f(t) = c_1\phi_1(t) + c_2\phi_2(t) + ... + c_n\phi_n(t) \tag{18}$$

Projection of Functions



Fourier transform

In the same way we obtain Fourier transform as projection of a signal $h(x)$ on to a sinusoidal basis function

In the continuous domain it is given by

$$H(\omega) = \int_{-\infty}^{\infty} h(x) e^{(-j\omega x)} dx \tag{19}$$

Illustration of Fourier transform

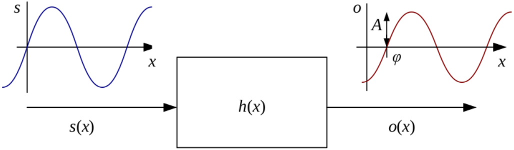


Figure 3.24 The Fourier Transform as the response of a filter $h(x)$ to an input sinusoid $s(x) = e^{j\omega x}$ yielding an output sinusoid $o(x) = h(x) * s(x) = Ae^{j\omega x + \phi}$.

The Fourier transform is a tabulation of the magnitude and phase response at each frequency

$$H(\omega) = F\{h(x)\} = Ae^{j\phi} \tag{20}$$

where A is the magnitude and ϕ is the phase response
The magnitude encodes how much signal there is at a particular frequency.

Discrete Fourier transform

Now, in the discrete domain we have samples only at discrete intervals, i.e. $h(x) = h[1], h[2], ..., h[n]$
Therefore, for discrete signals, we have

$$H\omega = \int_0^{(N-1)T} h(x)e^{-j\omega x}dx \tag{21}$$

$$H\omega = h[0]e^{-j0} + h[1]e^{-j\omega T} + ... + f[k]e^{-j\omega kT} \tag{22}$$

And therefore the Fourier transform in discrete domain is given by

$$H(\omega) = \sum_0^{N-1} h(x)e^{-j\omega kT} \tag{23}$$

Convolution property of Fourier Transform

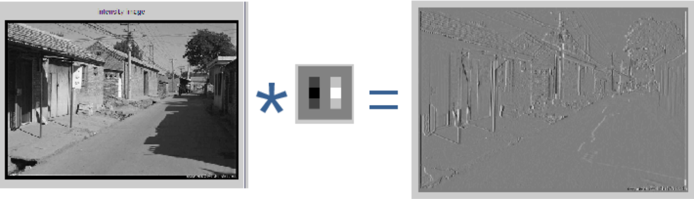
- The Fourier transform of the convolution of two functions is the product of their Fourier transforms

$$F[g * h] = F[g]F[h] \tag{24}$$

- Convolution in spatial domain is equivalent to multiplication in frequency domain

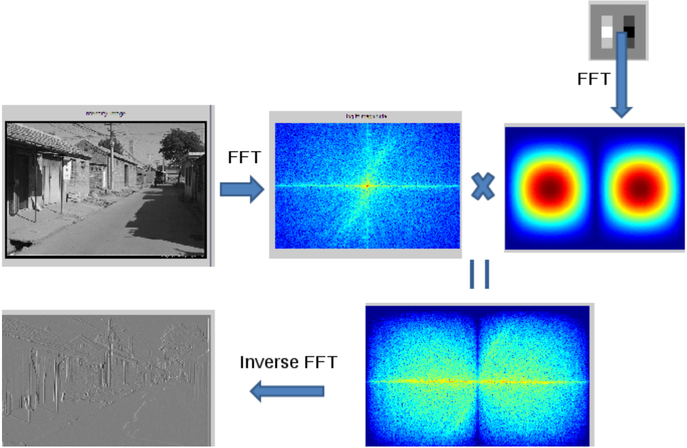
$$g * h = F^{-1}\left[F[g]F[h]\right] \tag{25}$$

Filtering in spatial domain



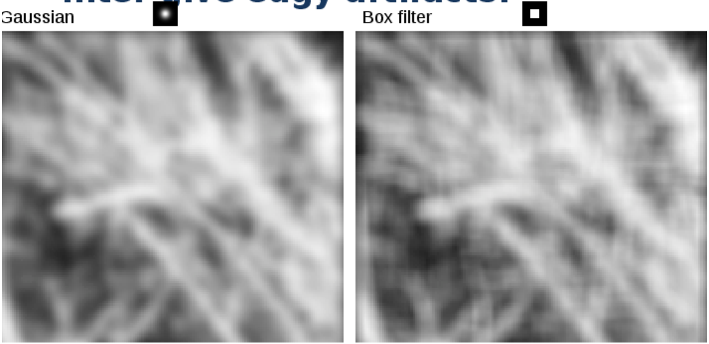
1	0	-1
2	0	-2
1	0	-1

Filtering in frequency domain



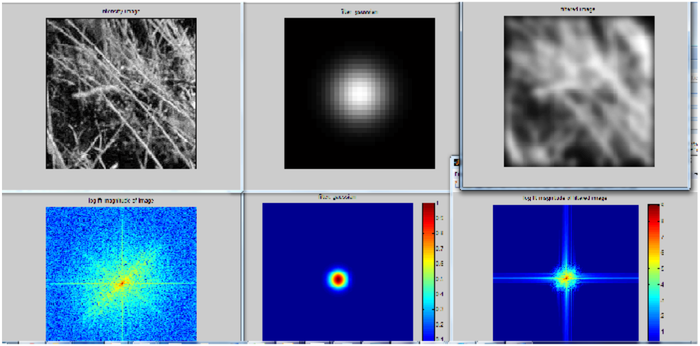
Filtering

Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?



Gaussian Filter

Gaussian



Box Filter

Box Filter