Correlation and Convolution

They replace the value of an image pixel with a combination of its neighbors

Basic operations in images
Shift Invariant
Linear

Thanks to David Jacobs for the use of some slides

Consider 1D images

I = [5 4 2 3 7 4 6 5 3 6]

so I(1)=5, I(2)=4, and so on.

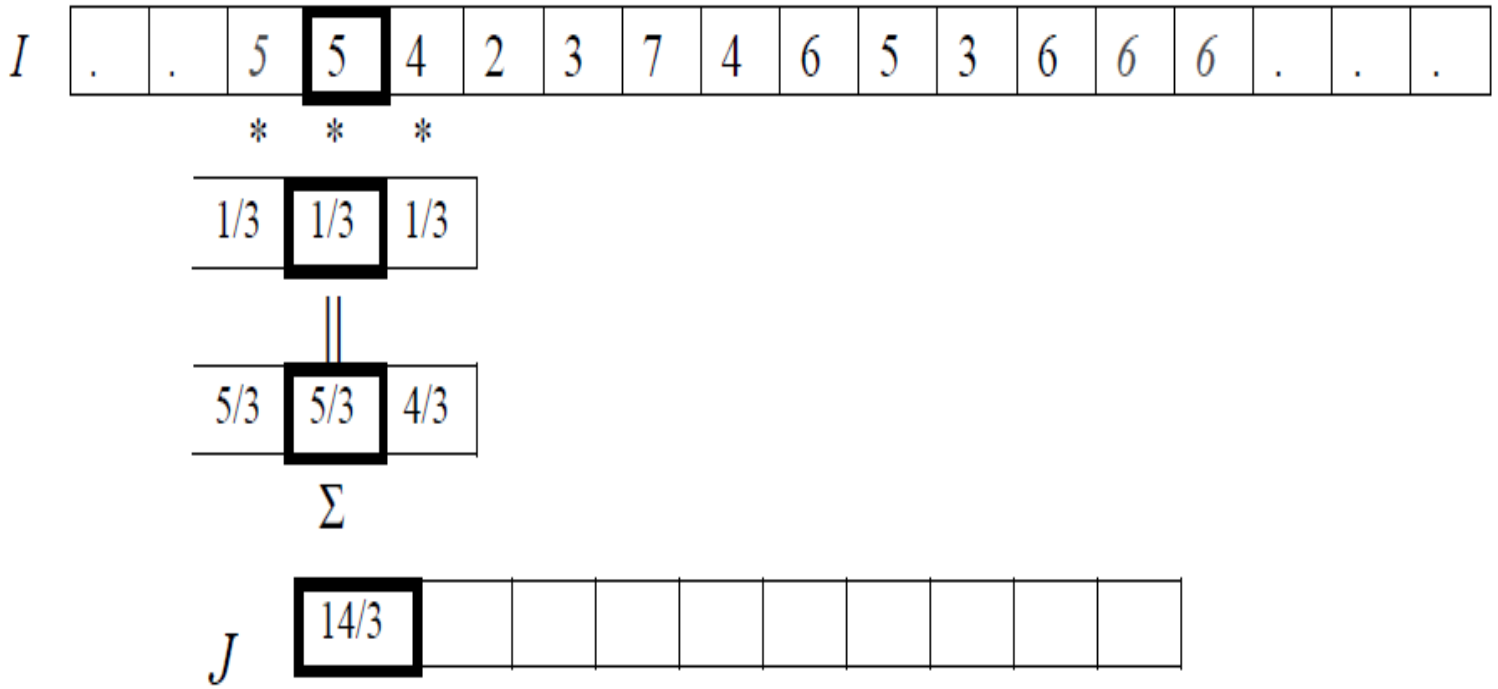
- Consider a simple averaging operation, in which we replace every pixel in a 1D image by the average of that pixel and its two neighbors, and we produce a new image J.
  So, J(3) = (I(2)+I(3)+I(4))/3 =(4+2+3)/3=3.
- How to deal with boundaries? What is the left neighbor of 5?

(a) [.. 0 0 0 **5 4 2 3 7 4 6 5 3 6** 0 0 ..] padded with zeros

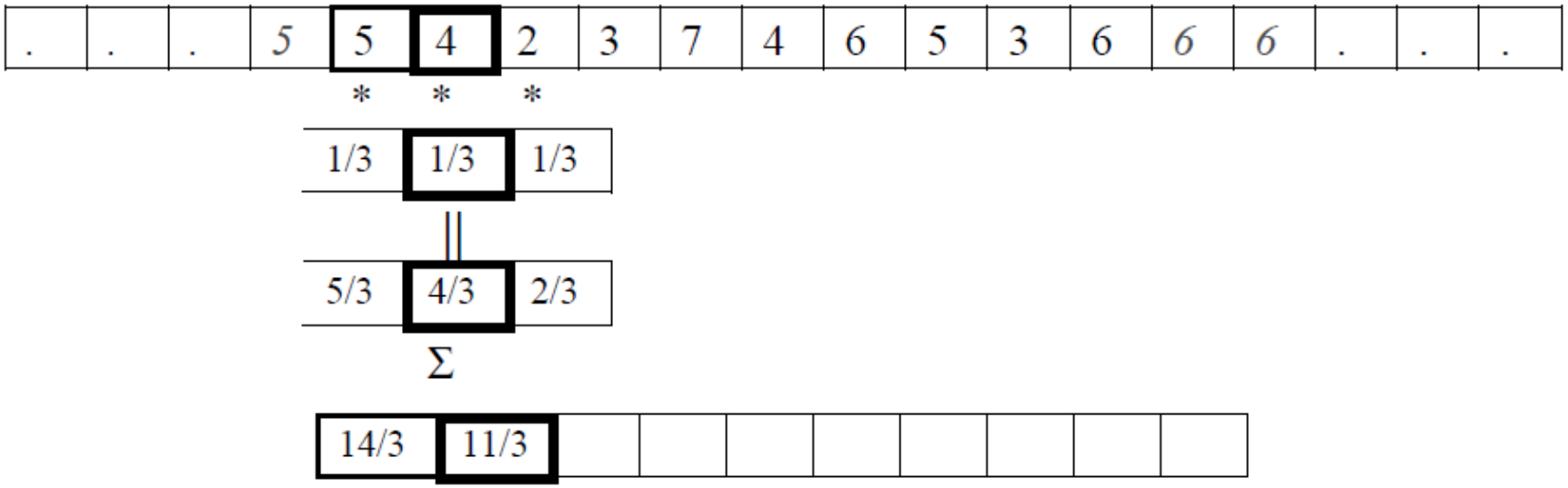
(b) [.. 5 5 5 **5 4 2 3 7 4 6 5 3 6** 6 6 ..] padded with 1st and last value

(c) [.. 5 3 6 **5 4 2 3 7 4 6 5 3 6** 5 4 ..] image repeated cyclic

Correlation: a sliding operation in a window



Continue with next pixel



Filters

- The numbers we multiply with, i.e. (1/3,1/3,1/3) form a filter. This filter is 3 pixels wide.
- We could build a filter for averaging that includes a pixel, its neighbors and their neighbors. This filter would be (1/5,1/5,1/5,1/5,1/5) and would be 5 pixels wide. These are “box” filters.
- We place the filter on the image, multiply the contents of every cell with the pixel below and add up the results.
- For this filter, the new image would be

J(1) = (I(-1)/5 + I(0)/5 + I(1)/5 + I(2)/5 + I(3)/5) = 1+1+1+4/5 + 2/5 = 4 1/5.

Mathematical Definition

- Suppose F is a correlation filter. It will be convenient notationally to suppose that F has an odd number of elements, so we can suppose that as it shifts, its center is right on top of an element of I. So we say that F has 2N+1 elements, and that these are indexed from -N to N, so that the center element of F is F(0).

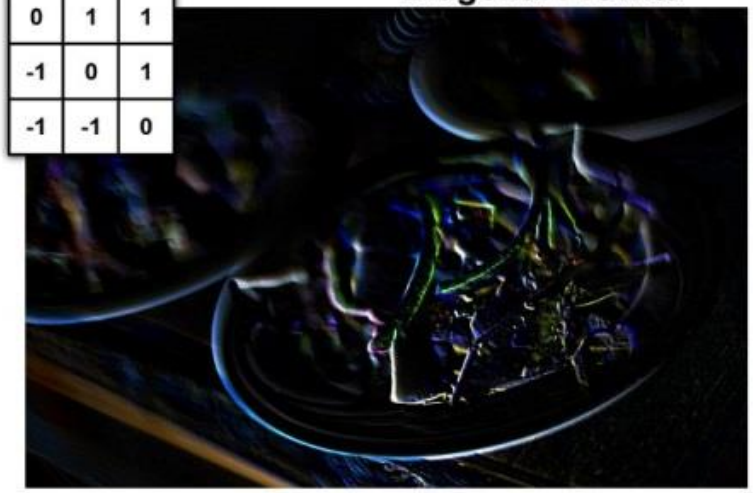
$$F \circ I(x) = \sum_{i=-N}^{N} F(i) I(x+i)$$

$$F(i) = \begin{cases} 1/3 & \text{for} \quad i = -1,0,1 \\ 0 & \text{for} \quad i \neq -1,0,1 \end{cases}$$

Examples in 2D

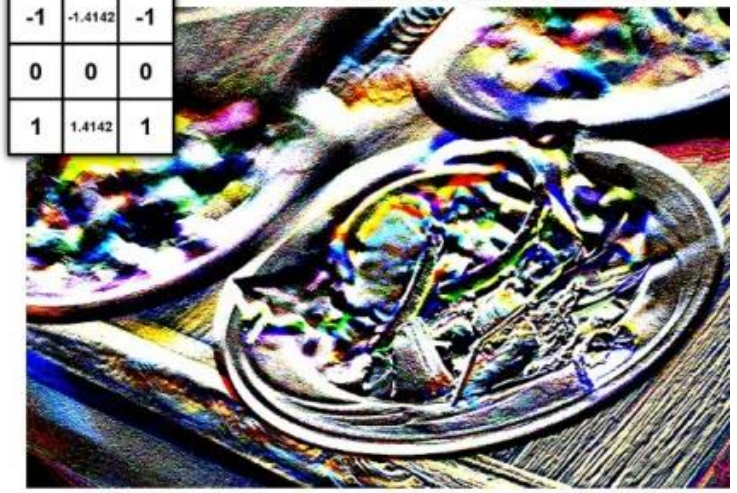
Diagonal Prewitt

0	1	1
-1	0	1
-1	-1	0



Horizontal Frei-Chen

-1	-1.4142	-1
0	0	0
1	1.4142	1

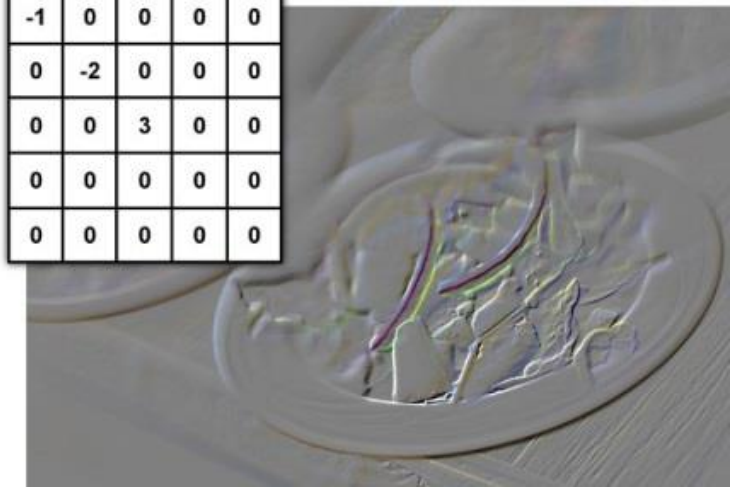


Sharpen

0	-1	0
-1	5	-1
0	-1	0

Emboss

-1	0	0	0	0
0	-2	0	0	0
0	0	3	0	0
0	0	0	0	0
0	0	0	0	0



Original

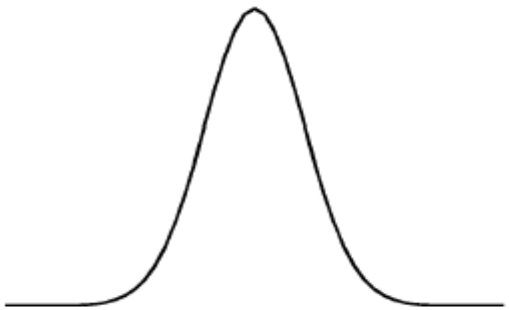
Filtered

Making filters from continuous functions

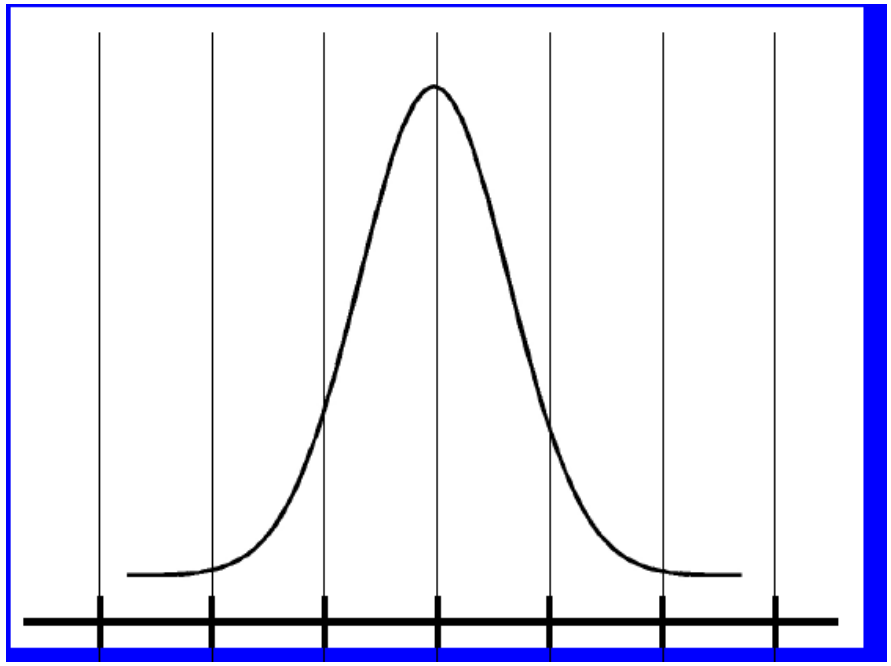
A one-dimensional Gaussian is:

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-(x-\mu)^2 \Big/ 2\sigma^2 \right)$$

This is also known as a Normal distribution. Here μ is the mean value, and σ is the variance. Here's a plot of a Gaussian:



- We would need to evaluate the Gaussian G at points (…,-3,-2,-1,0,1,2,3,..).
- Luckily G(x) tends to zero pretty quickly as x gets larger.



Taking derivatives with filters

Let's consider an example. Suppose image intensity grows quadratically with position, so that $I(x) = x^2$. Then if we look at the intensities at positions 1, 2, 3, … they will look like:

1	4	9	16	25	36	.	.	.

If we filter this with a filter of the form (-1/2 0 1/2) we will get:

1 1/2	4	6	8	10	12	.	.	.

We know that the derivative of $I(x)$, $dI/dx = 2x$. And sure enough, we have, for example, that $J(2) = 4$, $J(3) = 6$, …Notice that $J(1)$ is not equal to 2 because of the way we handle the boundary, by setting $I(0) = 1$ instead of $I(0) = 0$. So at the boundary, our image doesn't really reflect $I(x) = x^2$.

We could just as easily have used a filter like (0 -1 1), which computes the expression: $J(i) = (I(i+1)-I(i))/1$, or a filter (-1 1 0), which computes $J(i) = (I(i) – I(i-1))/1$. These are all reasonable approximations to a derivative. One advantage of the filter (-1/2 0 1/2) is that it treats the neighbors of a pixel symmetrically.

Matching with correlation

- Matching the filter with part of the image – their difference depends on 3 terms: the higher the correlation the better the match

$$\sum_{i=-N}^{N}\left(F(i)-I(x+i)\right)^2 = \sum_{i=-N}^{N}\left(F^2(i)+I^2(x+i)-2F(i)I(x+i)\right)$$

$$= \sum_{i=-N}^{N}\left(F^2(i)\right)+\sum_{i=-N}^{N}\left(I^2(x+i)\right)-2\sum_{i=-N}^{N}\left(F(i)I(x+i)\right)$$

Weaknesses of correlation

- Suppose we correlate the filter:

3	7	5

with the image:

3	2	4	1	3	8	4	0	3	8	0	7	7	7	1	2

We get the following result:

40	43	39	34	64	85	52	27	61	65	59	84	105	75	38	27

- Notice that we get a high result (85) when we center the filter on the sixth pixel, because (3,7,5) matches very well with (3,8,4). But, the highest correlation occurs at the 13th pixel, where the filter is matched to (7,7,7). Even though this is not as similar to (3,7,5), its magnitude is greater.

Matching

One way to overcome this is by just using the sum of square differences between the signals, as given above. This produces the result:

25	26	26	41	29	2	59	54	34	26	78	13	20	32	61	38

We can see that using Euclidean distance, (3,7,5) best matches the part of the image with pixel values (3,8,4). The next best match has values (0,7,7), but this is significantly worse.

Another option is to perform *normalized correlation* by computing::

$$\frac{\sum_{i=-N}^{N}\left(F(i)I(x+i)\right)}{\sqrt{\sum_{i=-N}^{N}\left(I(x+i)\right)^2}\sqrt{\sum_{i=-N}^{N}\left(F(i)\right)^2}}.$$

Normalized correlation

When we perform normalized correlation between (3,7,5) and the image I we get image J:

.946	.877	.934	.73	.81	.989	.64	.59	.78	.835	.61	.931	.95	.83	.57	.988

- Now, the region of the image that best matches the filter is (3,8,4). However, we also get a very good match between (3,7,5) and end of the image, which, when we replicate the last pixel at the boundary, is (1,2,2). These are actually quite similar, up to a scale factor. Keep in mind that normalized correlation would say that (3,6,6) and (1,2,2) match perfectly. Normalized correlation is particularly useful when some effect, such as changes in lighting or the camera response, might scale the intensities in the image by an unknown factor.

Correlation as an inner product

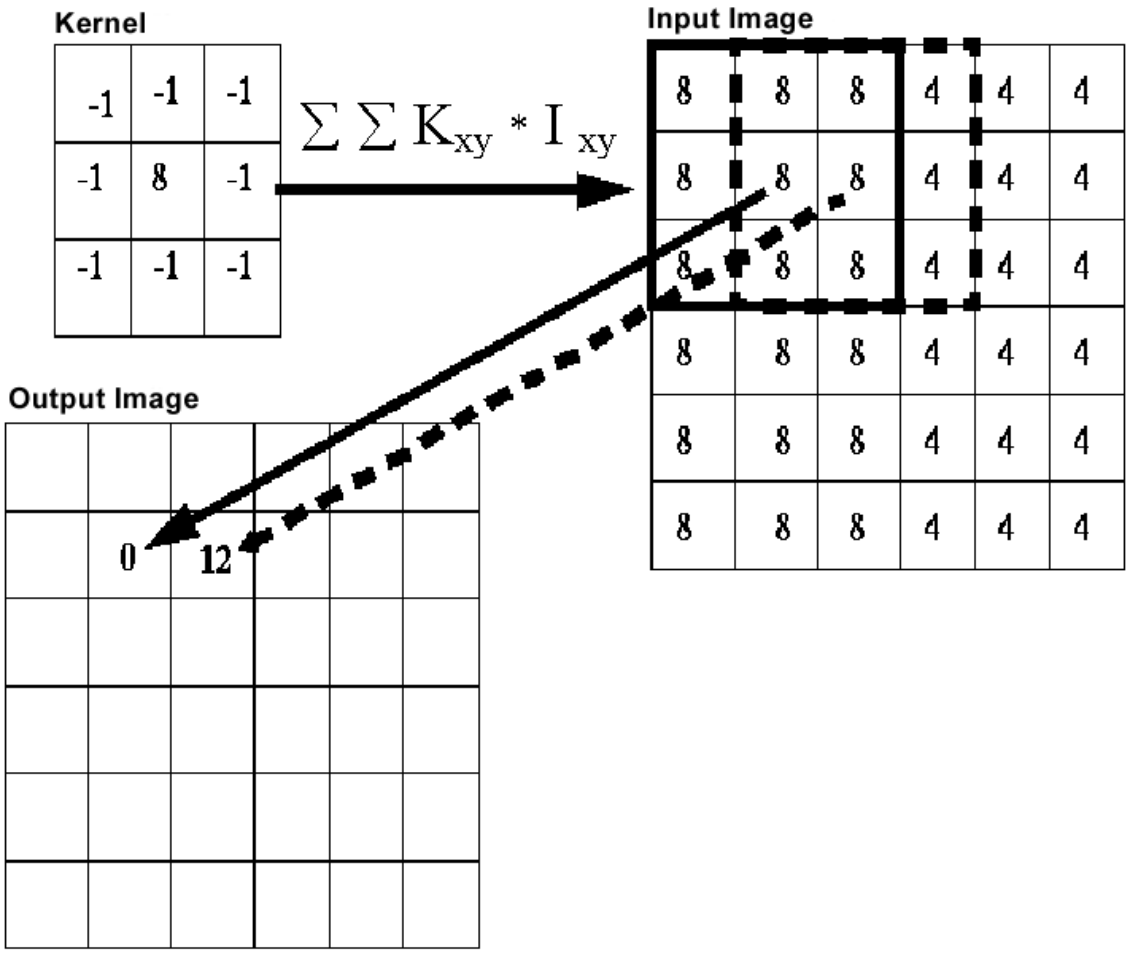
consider our filter as a vector, $F = (F(-N), F(-N+1) \ldots F(N))$. We are comparing this to a portion of an image, which we can also think of as a vector: $I_x = (I(x-N), I(x-N+1), \ldots I(x+N))$. Then, when we compute the correlation between F and I at the position x, we are computing $\langle F, I_x \rangle$. F and I_x are vectors in a (2N+1)-dimensional space, but still, everything we've learned about inner products applies. In particular,

$$\langle F, I_x \rangle = \|F\| \|I_x\| \cos\theta$$

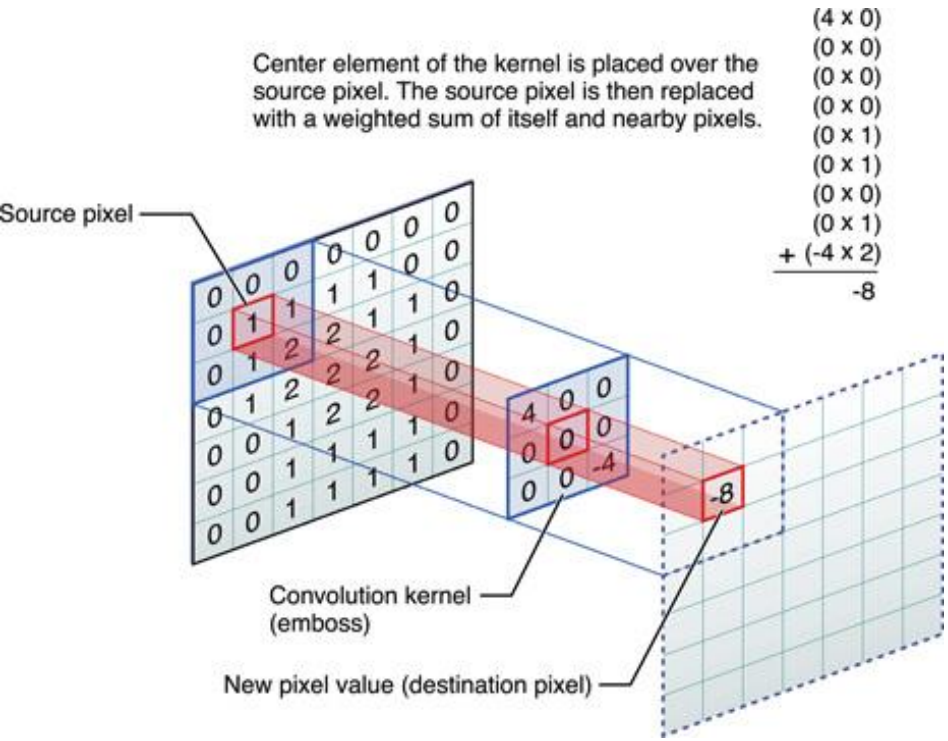
2D correlation

$$F \circ I(x,y) = \sum_{j=-N}^{N} \sum_{i=-N}^{N} F(i,j) I(x+i, y+j)$$

Examples


Kernel
-1 -1 -1
-1 8 -1
-1 -1 -1
$\sum \sum K_{xy} * I_{xy}$
Input Image
8 8 8 4 4 4
8 8 8 4 4 4
8 8 8 4 4 4
8 8 8 4 4 4
8 8 8 4 4 4
8 8 8 4 4 4
Output Image
0
12

Examples



22	15	1	3	60
42	5	38	39	7
28	9	4	66	79
0	82	45	12	17
99	14	72	51	3

×

0	0	0	0	0
0	0	0	1	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

=

	1	3	60	
	38	39	7	
	4	66	79	

Example

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

F

Below we show an image, *I,* and the result of applying the box filter above to *I* to produce the resulting image, *J.*

8	3	4	5
7	6	4	5
4	5	7	8
6	5	5	6

I

8	*8*	*3*	*4*	*5*	*5*
8	8	3	4	5	*5*
7	7	6	4	5	*5*
4	4	5	7	8	*8*
6	6	5	5	6	*6*
6	*6*	*5*	*5*	*6*	*6*

I with padded boundaries

6.44	5.22	4.33	4.67
5.78	5.33	5.22	5.67
5.56	5.44	5.67	6
5.22	5.33	5.78	6.33

J = F*o*I

Separable filters

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

Gaussian filtering has the same separability. For a 2D Gaussian, we use a function that is rotationally symmetric, where the height of the filter falls off as a Gaussian function of the distance from the center. This has the form:

$$G_0(x,y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\left(x^2 + y^2\right) \Big/ 2\sigma^2 \right)$$

This can be separated because it can be expressed as:

$$G_0(x,y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\left(x^2 + y^2\right) \Big/ 2\sigma^2 \right) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-x^2 \Big/ 2\sigma^2 \right) \exp\left(-y^2 \Big/ 2\sigma^2 \right)$$

It is the product of two components, one of which only depends on x, while the other only depends on y. The first component can be expressed with a filter that has a single row, and the second can be expressed with a filter that has a single column.

Convolution

Convolution is just like correlation, except that we flip over the filter before correlating. For example, convolution of a 1D image with the filter (3,7,5) is exactly the same as correlation with the filter (5,7,3). We can write the formula for this as:

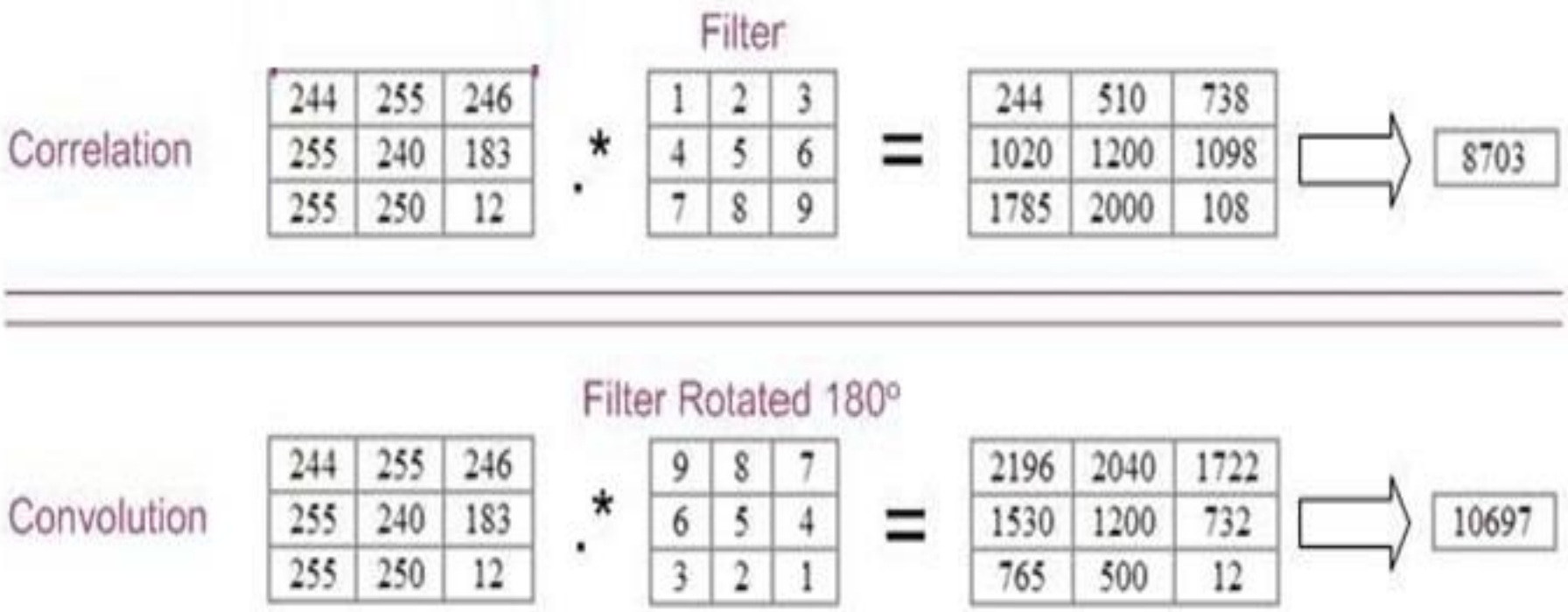
$$F * I(x) = \sum_{i=-N}^{N} F(i) I(x-i)$$

In the case of 2D convolution we flip the filter both horizontally and vertically. This can be written as:

$$F * I(x,y) = \sum_{j=-N}^{N} \sum_{i=-N}^{N} F(i,j) I(x-i, y-j)$$

Notice that correlation and convolution are identical when the filter is symmetric.

Example


Filter
Correlation
244 255 246
255 240 183
255 250 12
.*
1 2 3
4 5 6
7 8 9
=
244 510 738
1020 1200 1098
1785 2000 108
8703
Filter Rotated 180°
Convolution
244 255 246
255 240 183
255 250 12
.*
9 8 7
6 5 4
3 2 1
=
2196 2040 1722
1530 1200 732
765 500 12
10697

Correlation - Convolution

- Convolution is associative (F*G)*H=F*(G*H)

This is very convenient in filtering. If D is a derivative filter and G a smoothing filter then if I is the image: D*(G*I)= (D*G)*I

Correlation is not associative – it is mostly used in matching, where we do not need to combine different filters.

Dealing with Noise

- (other presentation)

Noise

Image processing is useful for noise reduction...

Original

Salt and pepper noise

Impulse noise

Gaussian noise

Common types of noise:

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

Additive noise

We often assume the noise is additive

- I = S + N. Noise doesn't depend on signal.
- We'll consider:

$$I_i = s_i + n_i \text{ with } E(n_i) = 0$$
$$s_i \text{ deterministic.}$$
$$n_i, n_j \text{ independent for } n_i \neq n_j$$
$$n_i, n_j \text{ identically distributed}$$

Practical noise reduction

- How can we “smooth” away noise in a single image?

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	100	130	110	120	110	0	0
0	0	0	110	90	100	90	100	0	0
0	0	0	130	100	90	130	110	0	0
0	0	0	120	100	130	110	120	0	0
0	0	0	90	110	80	120	100	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Mean filtering

$F[x, y]$

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

$G[x, y]$

Mean filtering

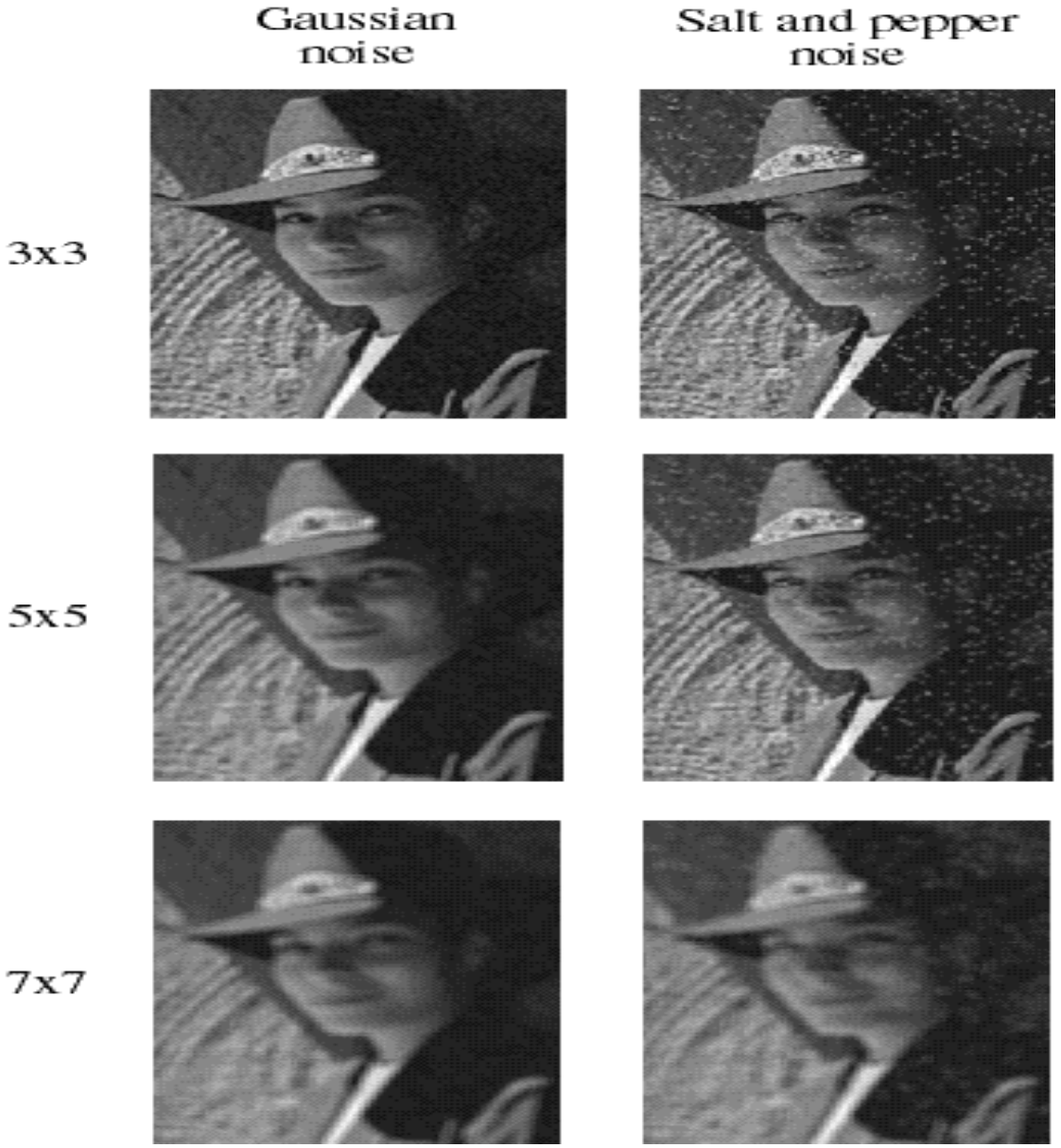
$F[x, y]$

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

$G[x, y]$

	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	

Effect of mean filters



Cross-correlation filtering

- Let's write this down as an equation. Assume the averaging window is (2k+1)x(2k+1):

$$G[i,j] = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]$$

- We can generalize this idea by allowing different weights for different neighboring pixels:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] F[i+u, j+v]$$

- This is called a **cross-correlation** operation and written:

$$G = H \otimes F$$

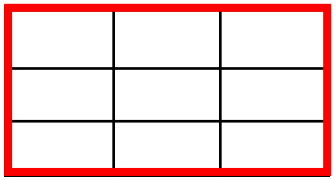
- H is called the "filter," "kernel," or "mask."
- The above allows negative filter indices. When you implement need to use: H[u+k,v+k] instead of H[u,v]

Mean kernel

- What's the kernel for a 3x3 mean filter?

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

$F[x, y]$



$H[u, v]$

Gaussian Filtering

- A Gaussian kernel gives less weight to pixels further from the center of the window

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

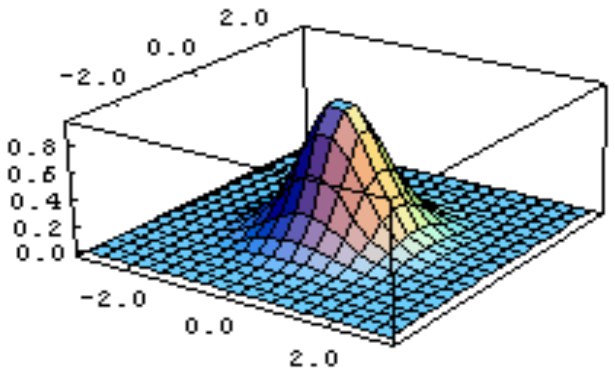
$$F[x, y]$$

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

$$H[u, v]$$

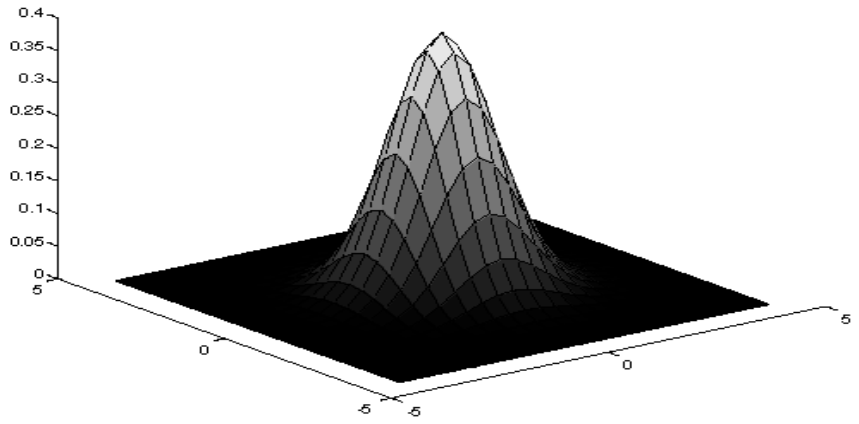
$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2}{}}$$

This kernel is an approximation of a Gaussian function:



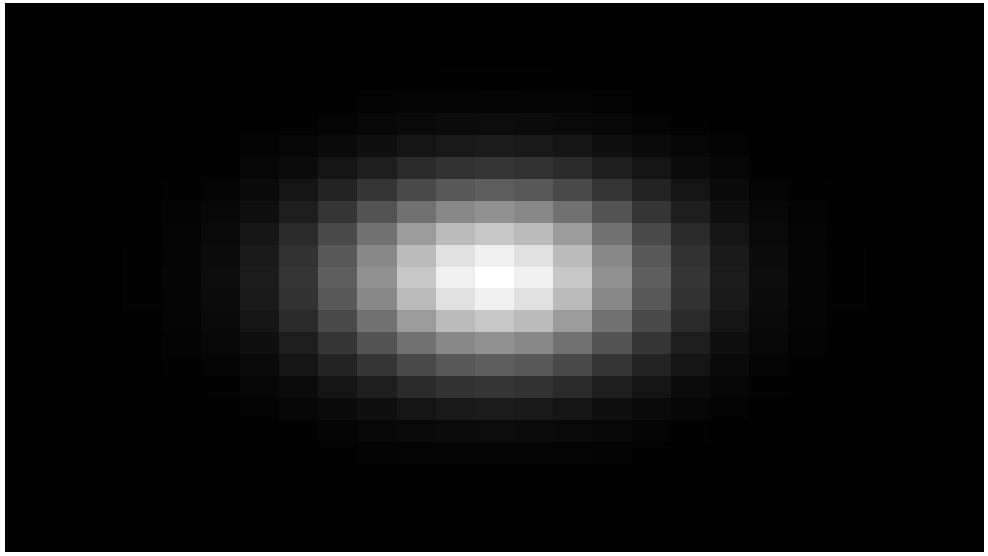
Gaussian Averaging

- Rotationally symmetric.
- Weights nearby pixels more than distant ones.
  - This makes sense as probabilistic inference.



A Gaussian gives a good model of a fuzzy blob

An Isotropic Gaussian

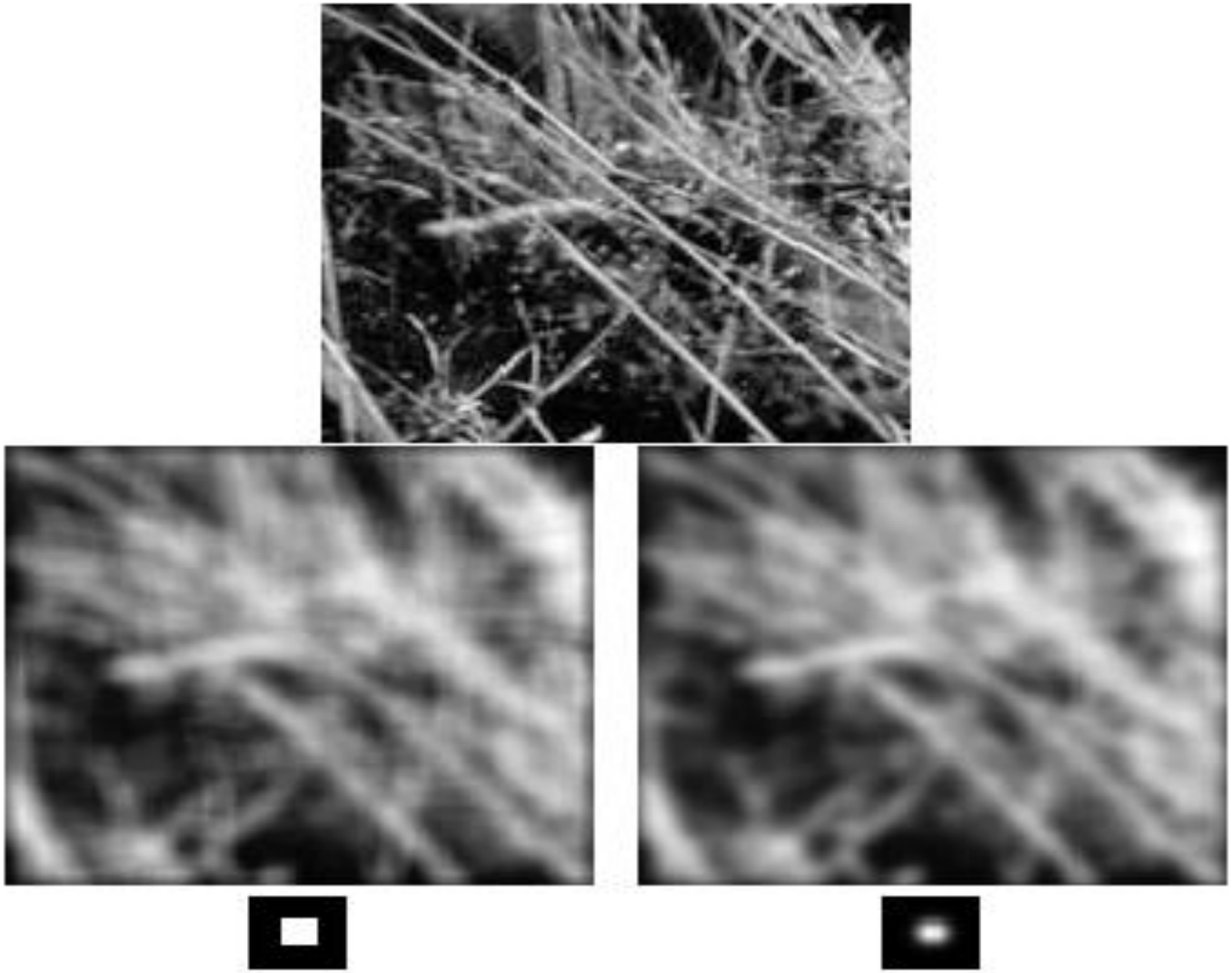


The picture shows a smoothing kernel proportional to

$$\exp\left(-\left(\frac{x^2+y^2}{2\sigma^2}\right)\right)$$

(which is a reasonable model of a circularly symmetric fuzzy blob)

Mean vs. Gaussian filtering



Efficient Implementation

Both, the BOX filter and the Gaussian filter are separable:

- First convolve each row with a 1D filter
- Then convolve each column with a 1D filter.

Convolution

- A **convolution** operation is a cross-correlation where the filter is flipped both horizontally and vertically before being applied to the image:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

- It is written: $G = H \star F$

- Suppose H is a Gaussian or mean kernel. How does convolution differ from cross-correlation?

Linear Shift-Invariance

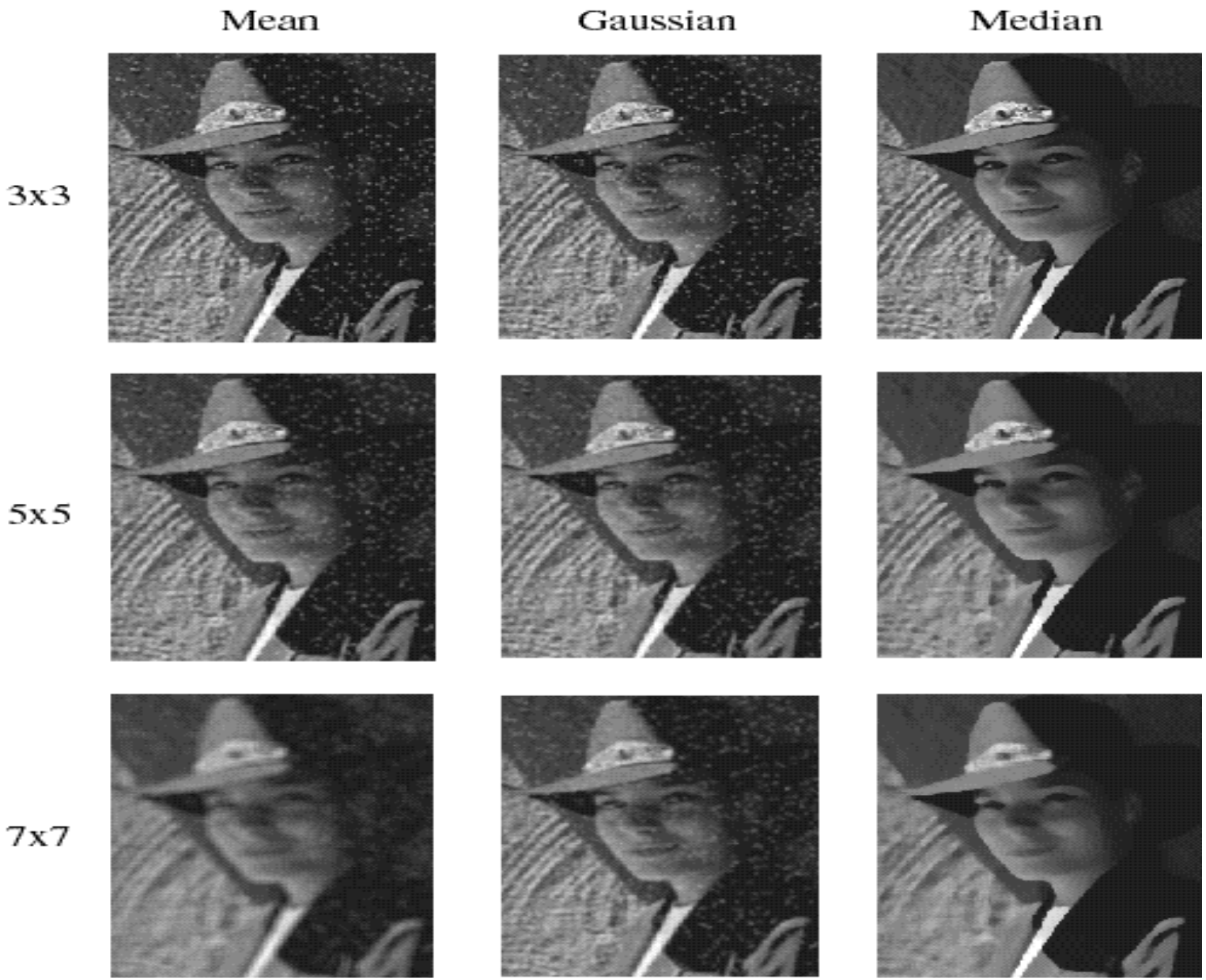
- A tranform T{} is
- Linear if:
- $T(a\, g(x,y)+b\, h(x,y)) = a\, T\{g(x,y)\} + b\, T(h(x,y))$

- Shift invariant if:
- Given $T(i(x,y)) = o(x,y)$
- $T\{i(x-x0, y- y0)\} = o(x-x0, y-y0)$

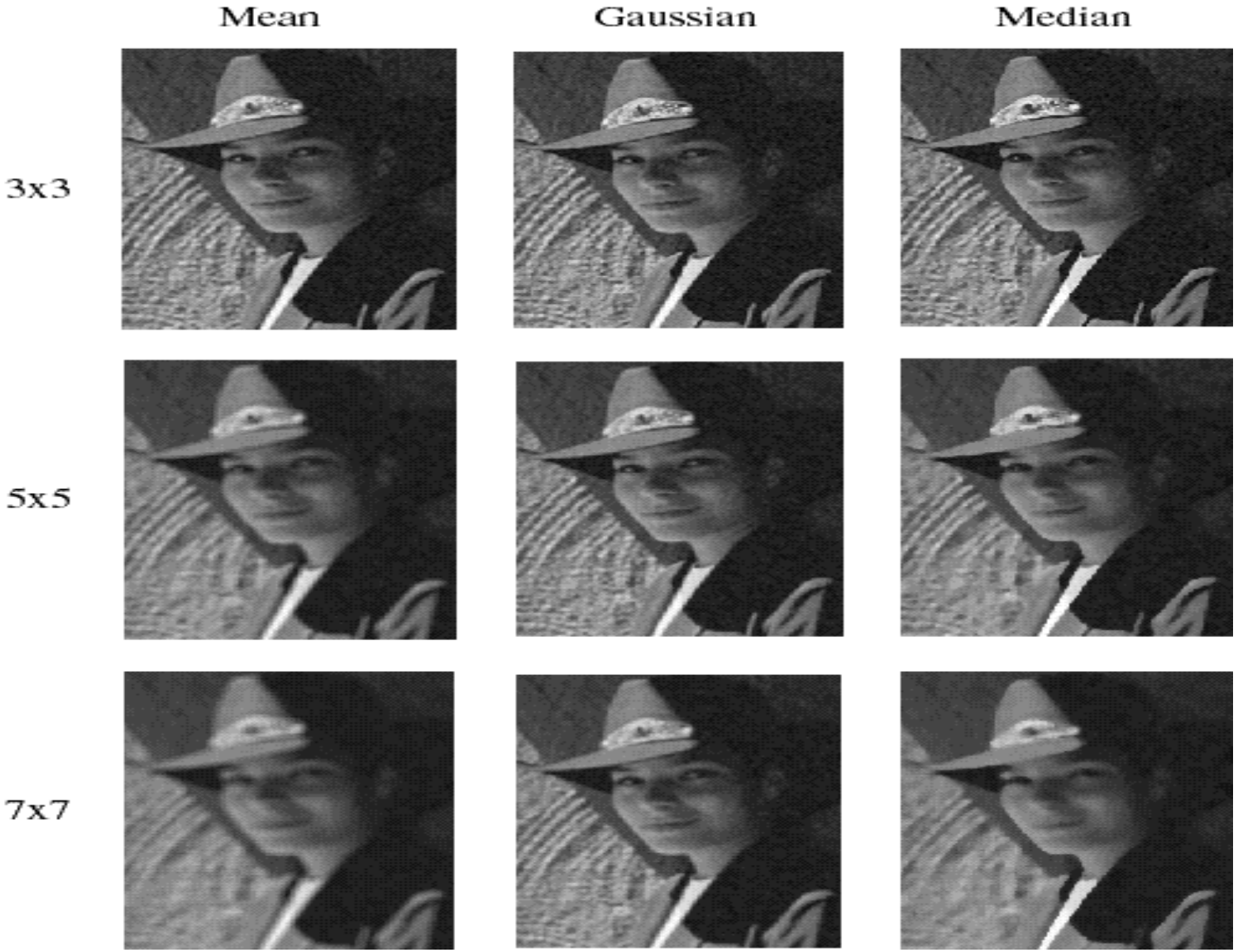
Median filters

- A **Median Filter** operates over a window by selecting the median intensity in the window.
- What advantage does a median filter have over a mean filter?
- Is a median filter a kind of convolution?
- Median filter is non linear

Comparison: salt and pepper noise



Comparison: Gaussian noise



Fourier Series

Analytic geometry gives a coordinate system for describing geometric objects.

$(x,y) = x(1,0) + y(0,1)$

(1,0) and (0,1) form an orthonormal basis for the plane. That means they are orthogonal – their inner product $\langle (1,0), (0,1) \rangle = 0$, and of unit magnitude.

Fourier Series gives a coordinate system for functions.

Why is an orthonormal basis a good representation?

- **Projection.** To find the x coordinate, we take <(x,y), (1,0)>.
- **Pythagorean theorem**. $\|(x,y)\|^2 = x^2 + y^2$.

The following functions provide an orthonormal basis for functions:

$$\frac{1}{\sqrt{2\pi}}, \frac{\cos(kx)}{\sqrt{\pi}}, \frac{\sin(kx)}{\sqrt{\pi}} \quad for\ k = 1,2,3,\ldots$$

What is the inner product for functions

- With discrete vectors, to compute the inner product we multiply together the values of matching coordinates, and then add up the results. We do the same thing with continuous functions; we multiply them together, and integrate (add) the result.

$$\langle f, g \rangle = \int_0^{2\pi} f(x) g(x) dx$$

$$f(x) = a_0 \frac{1}{\sqrt{2\pi}} + \sum_{k=1}^{\infty} \left(b_k \frac{\cos(kx)}{\sqrt{\pi}} + a_k \frac{\sin(kx)}{\sqrt{\pi}} \right)$$

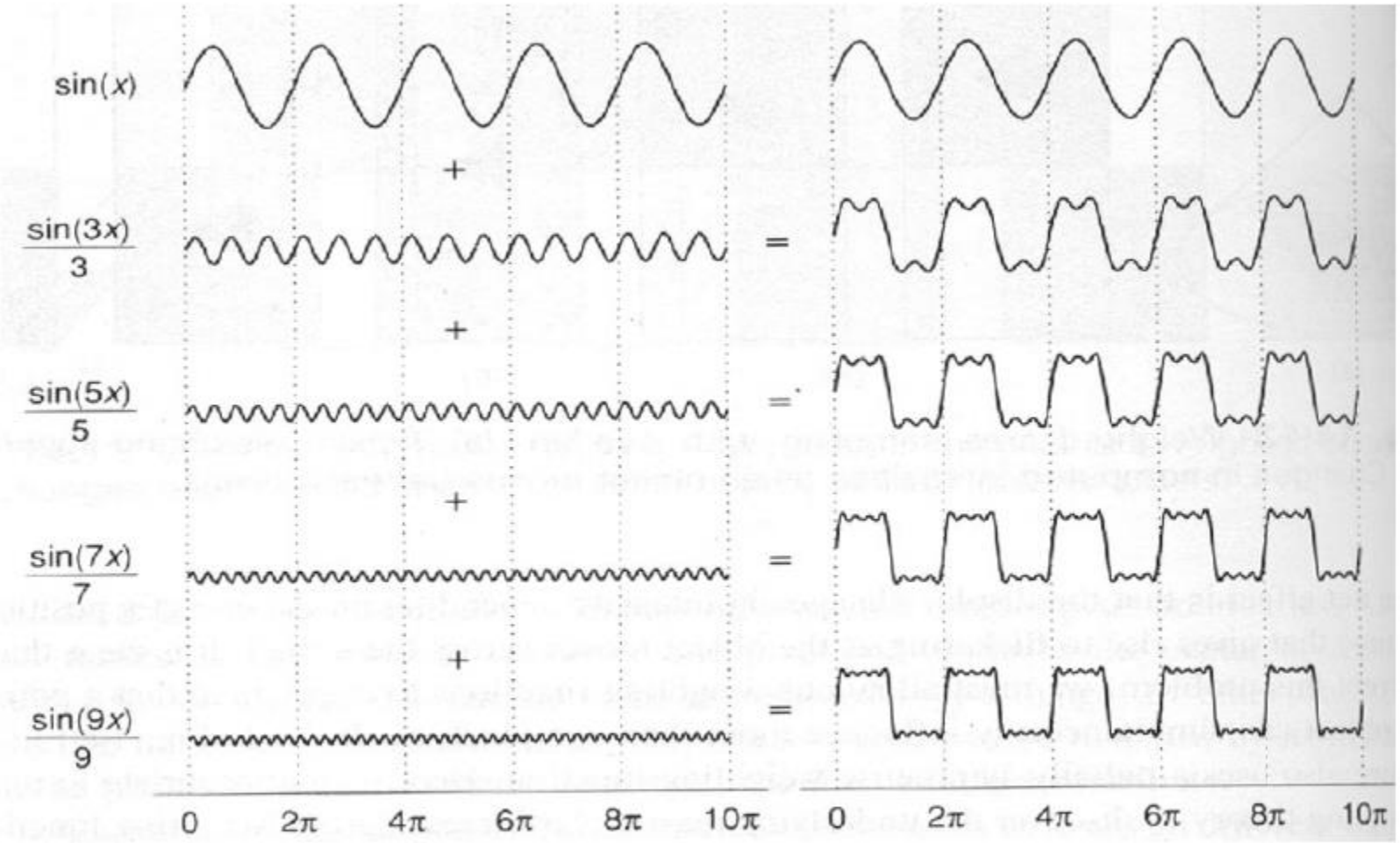


Image from *Computer Graphics: Principles and Practice*
by Foley, van Dam, Feiner, and Hughes

Fourier

$$f(x) = a_0 \frac{1}{\sqrt{2\pi}} + \sum_{k=1}^{\infty} \left(b_k \frac{\cos(kx)}{\sqrt{\pi}} + a_k \frac{\sin(kx)}{\sqrt{\pi}} \right)$$

- Values (a0, b1, a1, b2, a2, …) are the coordinates of the function in this new coordinate system provided by the Fourier Series.

What does this mean?

This result means that any function can be broken down into: the sum of sine waves of different amplitudes and phases.

We say that the part of the function that is due to longer frequency sine waves is the low frequency part of the function. The part that is due to high frequency sine waves is the high frequency component of the function.

If a function is very bumpy, with small rapid changes in its value, these rapid changes will be due to the high frequency component of the function. If we can eliminate these high-frequency components, we can make the function smoother.

Solving for the coordinates

$$a_0 = \left\langle f, \frac{1}{\sqrt{2\pi}} \right\rangle = \int_0^{2\pi} \frac{f(x)}{\sqrt{2\pi}} dx \qquad b_k = \left\langle f, \frac{\cos(kx)}{\sqrt{\pi}} \right\rangle = \int_0^{2\pi} \frac{f(x)\cos(kx)}{\sqrt{\pi}} dx$$

$$a_k = \left\langle f, \frac{\sin(kx)}{\sqrt{\pi}} \right\rangle = \int_0^{2\pi} \frac{f(x)\sin(kx)}{\sqrt{\pi}} dx$$

The analog to the Pythagorean theorem is called Parsevaal's theorem. This is:

$$\int_0^{2\pi} f^2(x)dx = a_0^2 + \sum_1^{\infty} \left(a_i^2 + b_i^2\right)$$

Convolution Theorem

- Let F, G, H be the Fourier series that represents the functions f, g, and h. That is, F,G, and H are infinite vectors. The convolution theorem states that convolution in the spatial domain is equivalent to component-wise multiplication in the transform domain

$$f * g = h \quad \boxtimes \quad FG = H.$$

Consequences of the Theorem

This is a remarkable theorem. As an example of its significance, suppose we convolve a filter, *F*, with an image *I*, and that *F* just consists of a single sine wave, *F=sin(x)*. Then the Fourier Series representation of *F* is:

$$a_1 = \sqrt{\pi}, \quad \text{all other components} = 0.$$

This means that no matter what *I* is, if we define *J=F*I*, then the components of the Fourier series of *J* will all be zero except for a_1, because all other components of *F* are 0, and when we multiply these by the corresponding components of *I* we always get 0.

Why averaging is not a good way to smooth

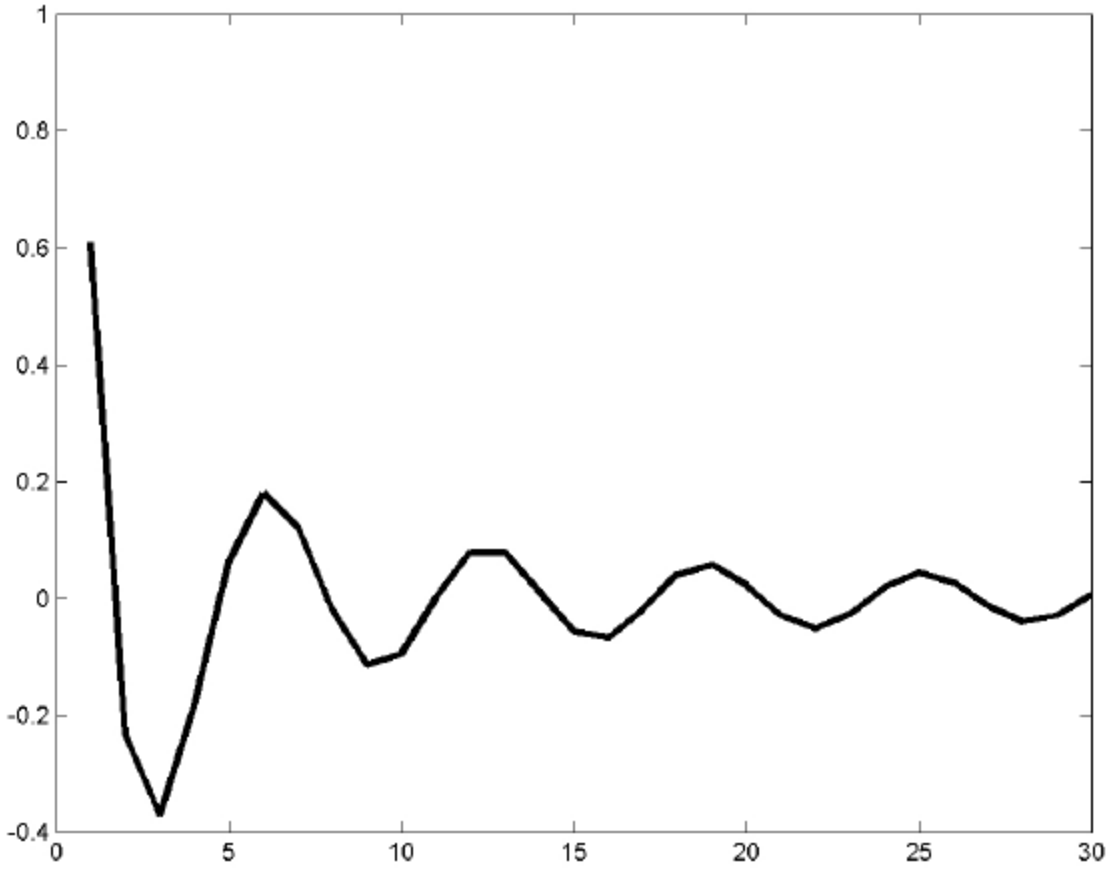
- Let f(x) be an averaging filter, which is constant in the range –T to T. Then, we can compute its Fourier Series as:

$$a_0 = \int_{-T}^{T} \frac{1}{\sqrt{2\pi}} dx = \frac{2T}{\sqrt{2\pi}}$$

$$a_k = \int_{-T}^{T} \frac{\sin(kx)}{\sqrt{\pi}} dx = \frac{2\cos(kT)}{k\sqrt{\pi}}$$

$$b_k = \int_{-T}^{T} \frac{\cos(kx)}{\sqrt{\pi}} dx = \frac{-2\sin(kT)}{k\sqrt{\pi}}$$

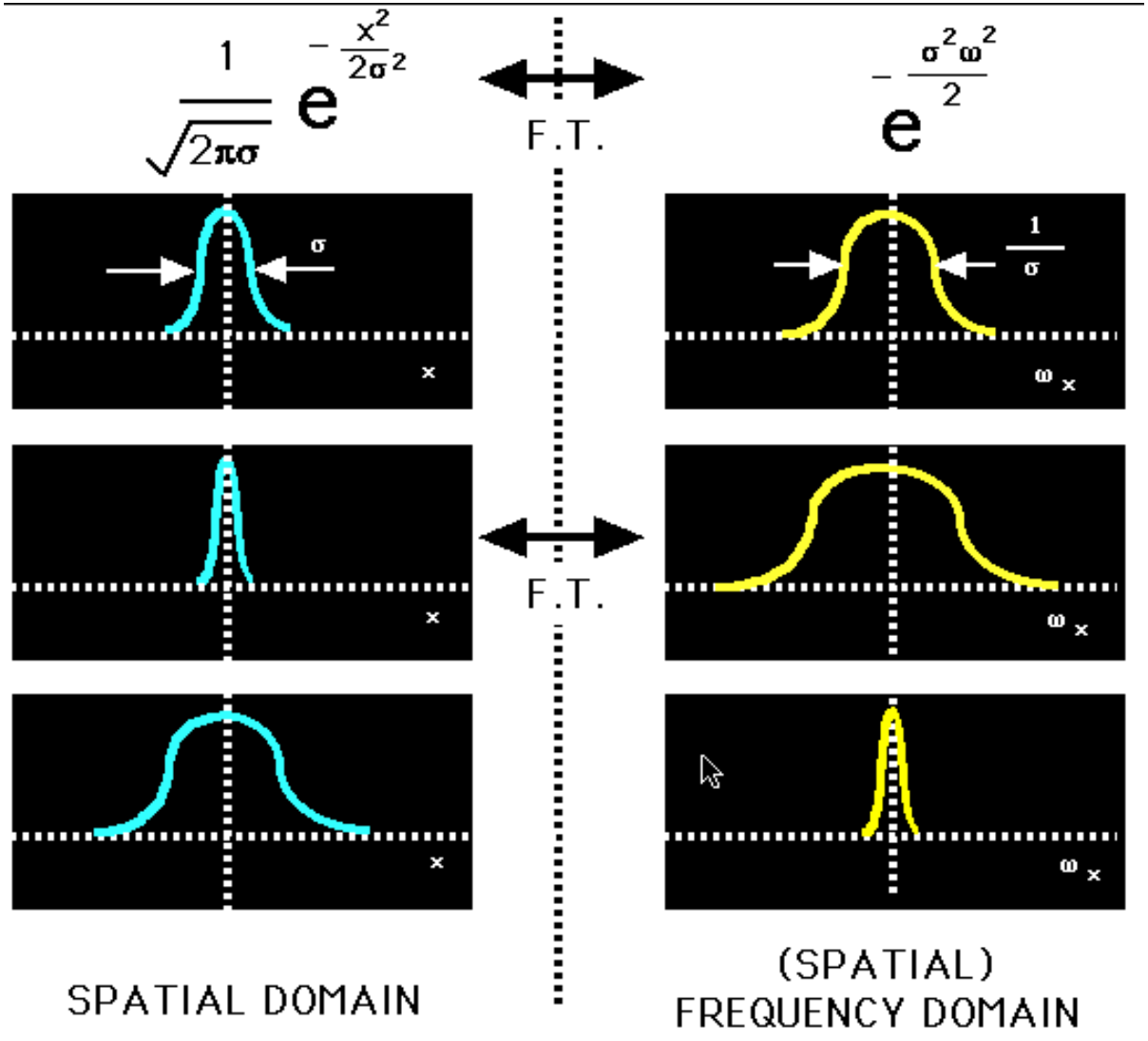
This is called the sinc function. If we graph it, we see that it looks like:



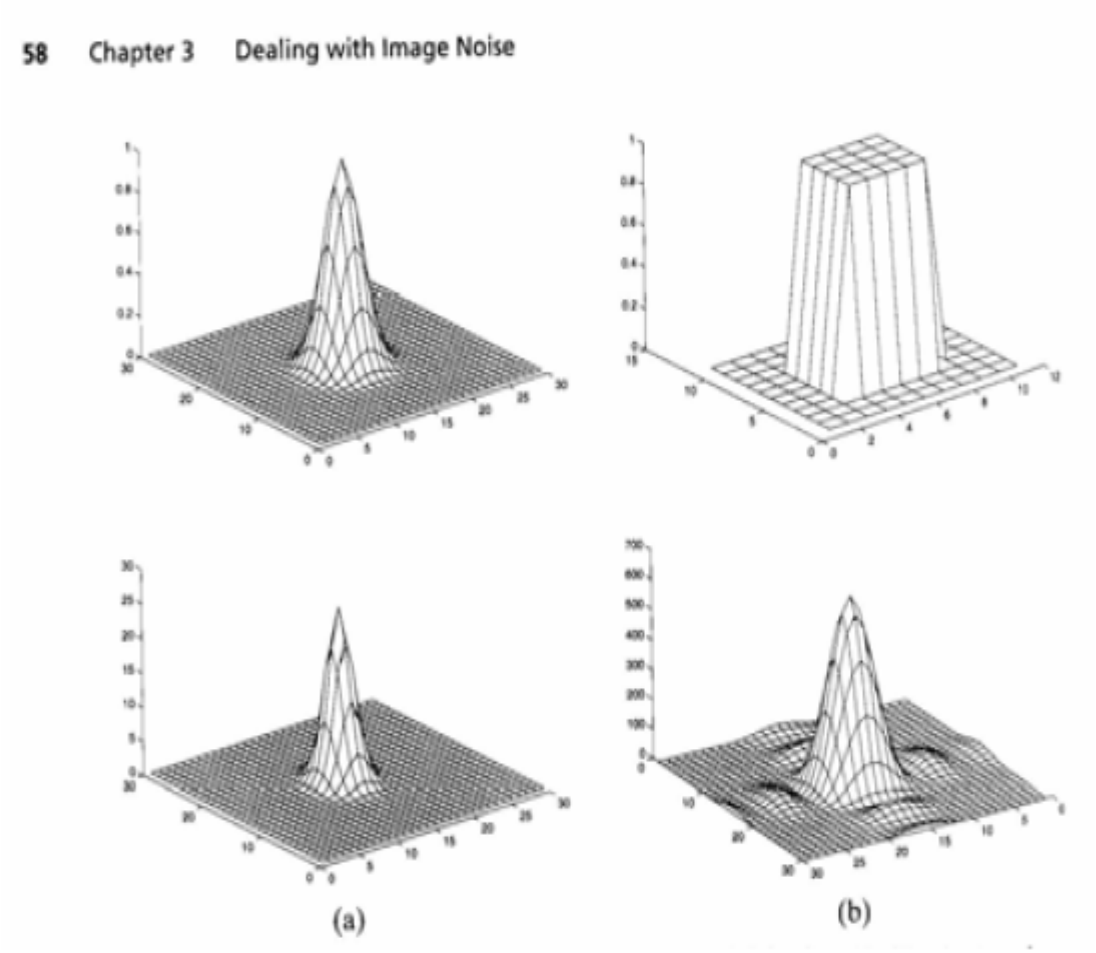
This does not decay as fast as a Gaussian. Also it has oscillations that produce odd effects. As a result of these oscillations, some high frequency components of an image will be wiped out by averaging, while some similar high frequency components remain.

The Fourier Series of a Gaussian

the Fourier series representation of a Gaussian is also a Gaussian. If g(t) is a Gaussian, the broader g is, the narrower G is (and vice versa). This means that smoothing with a Gaussian reduces high frequency components of a signal.


$\frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2}{2\sigma^2}}$
F.T.
$e^{-\frac{\sigma^2\omega^2}{2}}$
σ
x
$\frac{1}{\sigma}$
ω_x
F.T.
SPATIAL DOMAIN
(SPATIAL)
FREQUENCY DOMAIN

- Transform of box filter is sinc.
- Transform of Gaussian is Gaussian.



(Trucco and Verri)

Aliasing and Sampling Theorem

- Sampling means that we know the value of a function, f, at discrete intervals, f(nπ/T) for n = 0, +-1, +-T. That is, we have 2T + 1 uniform samples of f(x), assuming f is a periodic function

- Suppose f is band limited to T, i.e.

$$f(x) = a_0 \frac{1}{\sqrt{2\pi}} + \sum_{k=1}^{T} \left(b_k \frac{\cos(kx)}{\sqrt{\pi}} + a_k \frac{\sin(kx)}{\sqrt{\pi}} \right)$$

- We know the value of f(x) at 2T + 1 positions. If we plug these values into the above equation, we get 2T + 1 equations. The ai and bi give us 2T + 1 unknowns. These are linear equations. So we get a unique solution. This means that we can reconstruct f from its samples. However, if we have fewer samples, reconstruction is not possible. If there is a higher frequency component, then this can play havoc with our reconstruction of all low frequency components, and be interpreted as odd low frequency components.