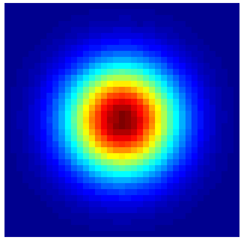
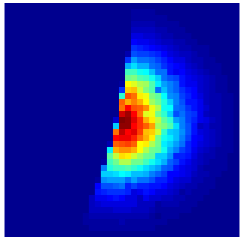
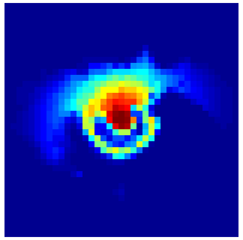
Spatial Operations

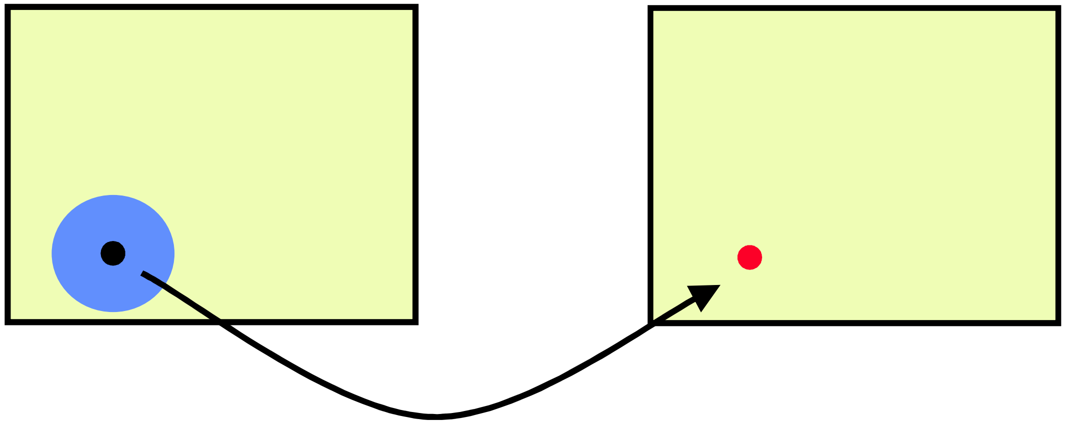




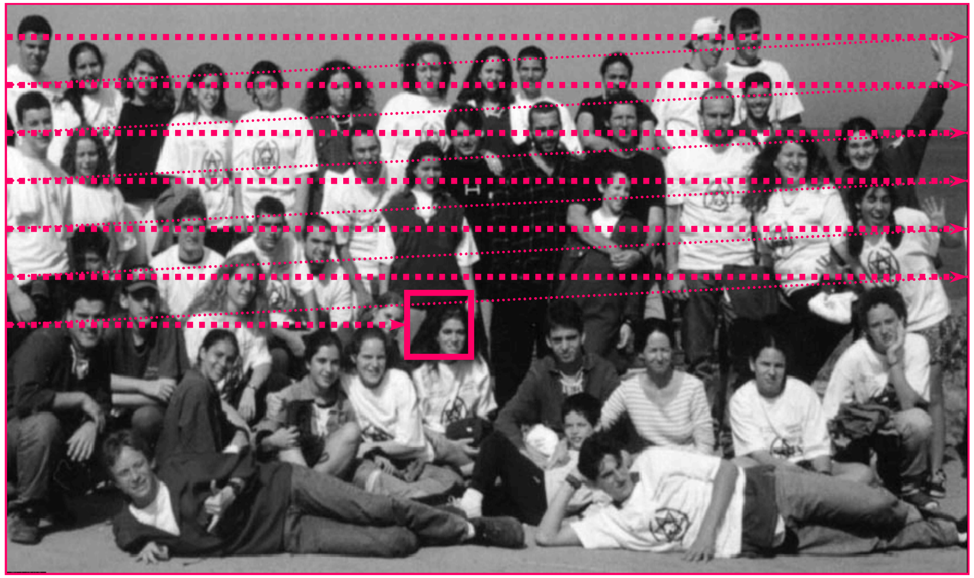


Spatial Operations

$$f'(x, y) = M\left(\left\{f(i, j) \mid (i, j) \in N(x, y)\right\}\right)$$



Very simple Examples: Min/Max filters



30	10	20
10	25	250
20	25	30

↓

10, 10, 20, 20, 25, 25, 30, 30, 250

min max

- Min filter: $f'(x,y) = \min(\{f(m,n)\})_{(m,n)\in N(x,y)}$
- Max filter $f'(x,y) = \max(\{f(m,n)\})_{(m,n)\in N(x,y)}$

Original Image

Salt & Pepper Noise

$$f_n(x, y) = \begin{cases} f(x, y) & \text{with probability } p \\ 255 & \text{with probability } (1-p)/2 \\ 0 & \text{with probability } (1-p)/2 \end{cases}$$

Min(f_n)

2x2 neighborhood

Max(f_n)

2x2 neighborhood

$MaxMin(f_n)$ $MinMax(f_n)$

Noisy Image F_n

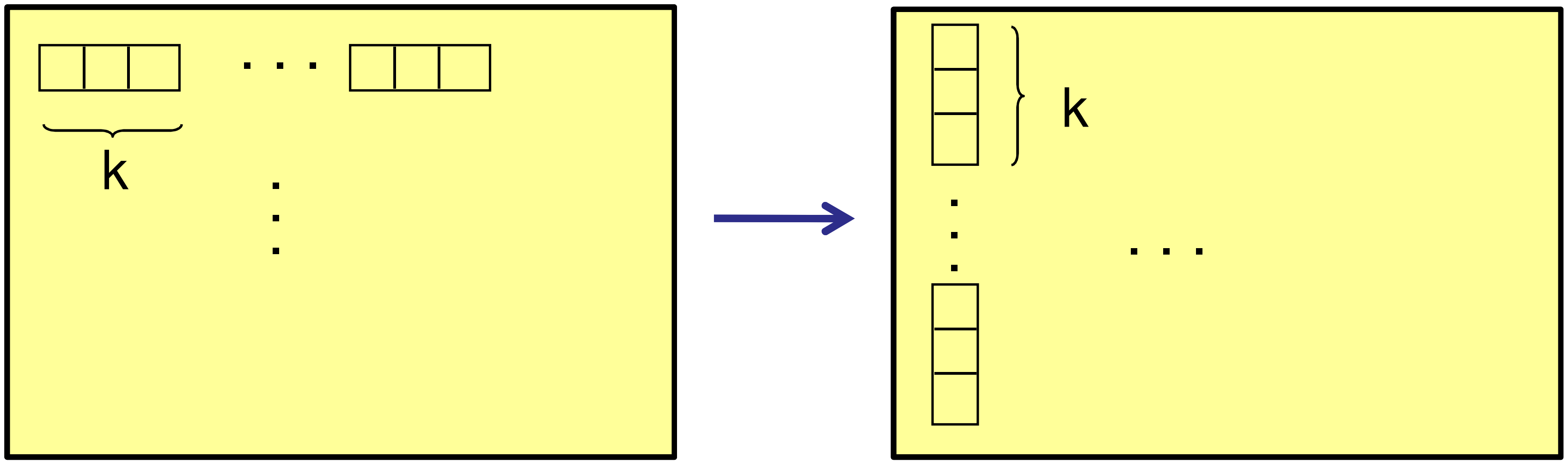
$MaxMin(MinMax(f_n))$

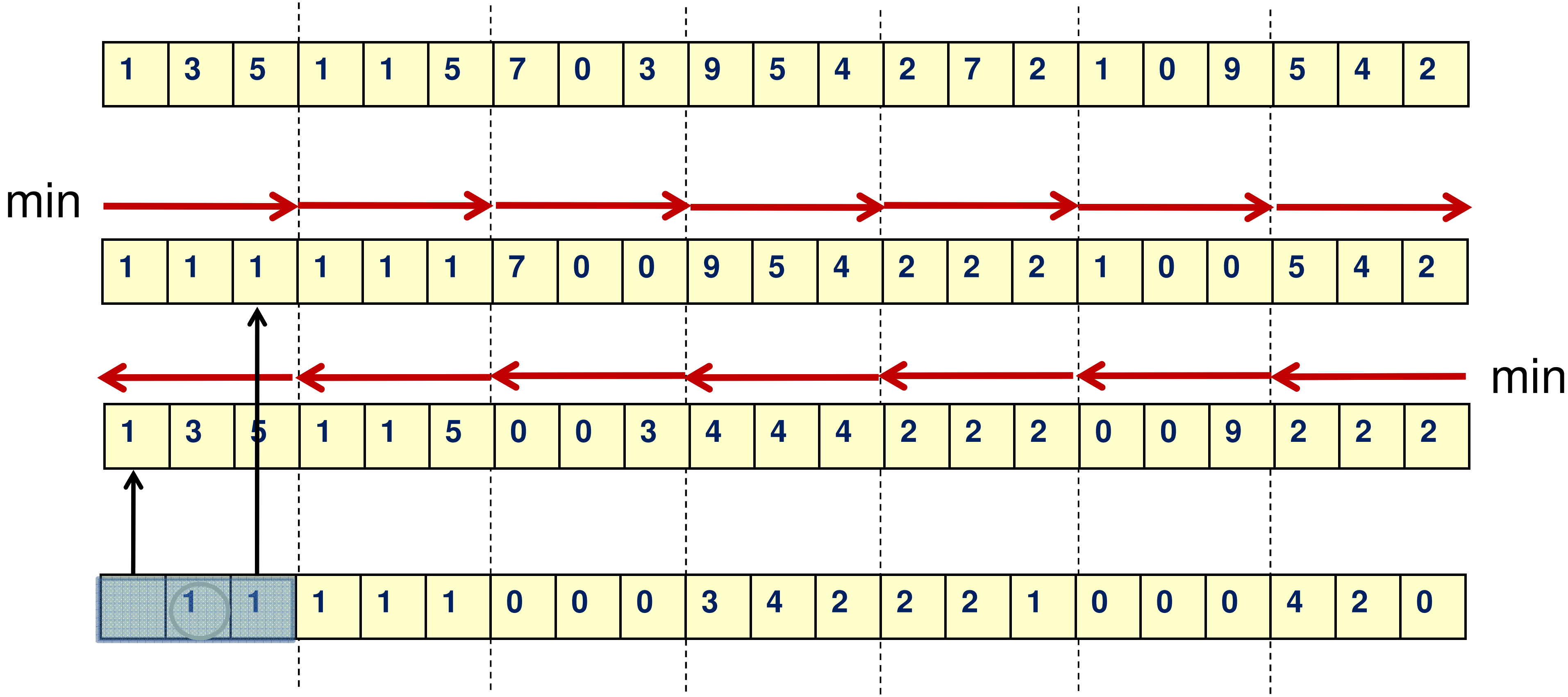
Complexity Min/Max Filters

- **Naïve**: Calculating the min/max filter for nxn image and for kxk neighborhood size requires k^2n^2 operations.
- **Separabilty**: Min/max filters are separable, thus calculations can be applied with $2kn^2$ operations:

$$Min(f,k,k)=min(min(f,1,k),k,1)$$

- Linear time algorithm is available.





- Linear time min/max filter requires 3n comparisons at each axis (Werman & Gil 1992)

The Median Filter

30	10	20
10	25	250
20	25	30

$$f'(x,y) = med(\{f(m,n)\})_{(m,n)\in N(x,y)}$$

10, 10, 20, 20, 25, 25, 30, 30, 250

median

- The median minimizes the sum of absolute differences (SAD) of {f(m,n)}:

$$\mathrm{med}(\{\mathrm{I}(\mathrm{m},\mathrm{n})\}) = \min_{\mathrm{u}} \sum_{(\mathrm{m},\mathrm{n})\in \mathrm{N}} |\mathrm{I}(\mathrm{m},\mathrm{n}) - \mathrm{u}|$$

- Is median filter separable?
- What about complexity?

Noisy Image

median (f)

3x3 neighborhood

Degraded Image

Source: Freeman and Durand

3x3 median filter

Source: Freeman and Durand

5x5 median filter

Source: Freeman and Durand

5x5 median filt

Source: Freeman and Durand

The Average Filter

$$f'(x,y) = \text{mean}\left(\{f(m,n)\}\right)_{(m,n)\in N(x,y)} = \frac{1}{|N|}\sum_{(m,n)\in N(x,y)} I(m,n)$$

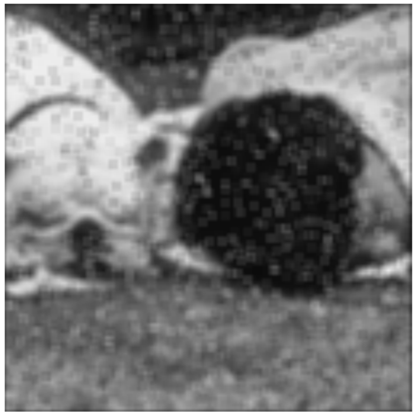
- The average minimizes the sum of squared differences (SSD) of {f(m,n)}:

$$\text{m}ean\left(\{I(m,n)\}\right) = \min_{u} \sum_{(m,n)\in N} \left(I(m,n) - u\right)^2$$

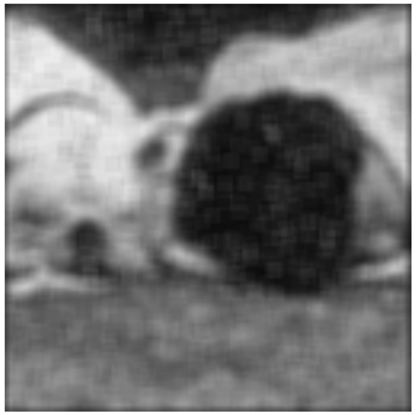
- Is average filter separable?
- What about complexity?

Average filter for Noise Reduction

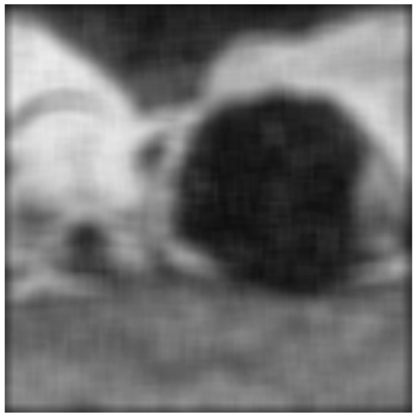
Noisy image



3x3 average



5x5 average



7x7 average

median

The Convolution

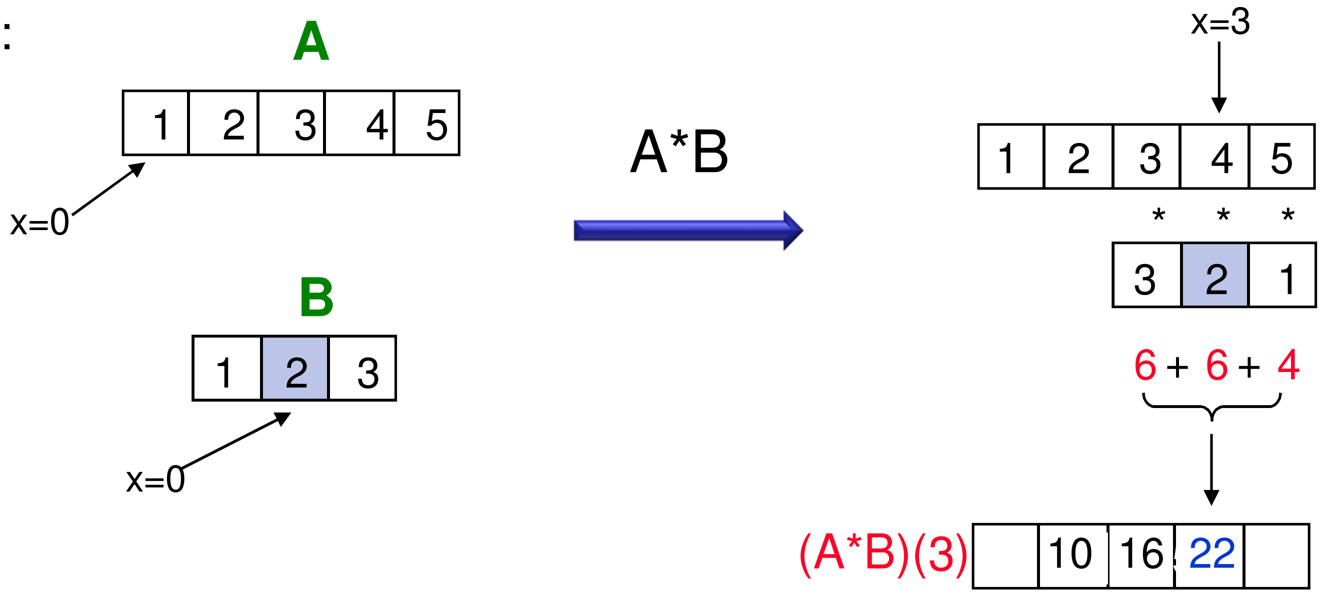
- The average filter is a particular example of a more general operation: **Image Convolution.**
- Let **A, B** be images. B is typically smaller than A.
- B is typically called the **mask** or the **kernel**.
- The convolution for 1D signal

$$(A * B)(x) = \sum_i A(i)B(x-i)$$

1D Convolution

$$(A * B)(x) = \sum_i A(i)B(x-i)$$

Example:



What happens near the edges?

- Option 1: Zero padding

0 0 0 | 1 | 2 | 3 | 4 | 5 | 0 0 0   * | 1 | 2 | 3 |

4	10	16	22	22

- Option 2: Wrap around

3 4 5 | 1 | 2 | 3 | 4 | 5 | 1 2 3 4 5

19	10	16	22	23

- Option 3: Reflection

3 2 1 | 1 | 2 | 3 | 4 | 5 | 5 4 3 2

7	10	16	22	27

What is the length of the result?

- Option 1: "same" (size A)

  0 0 0 | 1 | 2 | 3 | 4 | 5 | 0 0 0 * | 1 | 2 | 3 |

  0 0 1 | 4 | 10 | 16 | 22 | 22 | 15 0 0

- Option 2: "full" (size A + size B + 1)

  0 0 0 | 1 | 2 | 3 | 4 | 5 | 0 0 0

  0 0 | 1 | 4 | 10 | 16 | 22 | 22 | 15 | 0 0

- Option 3: "valid" (size A – size B +1)

  0 0 0 | 1 | 2 | 3 | 4 | 5 | 0 0 0

  0 0 1 4 | 10 | 16 | 22 | 22 15 0 0

Examples

Example 1:

A * **B** = **A*B**

A:

1	2	3	4	5

x=0

B:

0	1	0

x=0

A*B:

1	2	3	4	5

Example 2:

A * **B** = **A*B**

A:

1	2	3	4	5

B:

0	2	0

A*B:

2	4	5	8	10

Example 3:

A * **B** = **A*B**

A:

1	2	3	4	5

B:

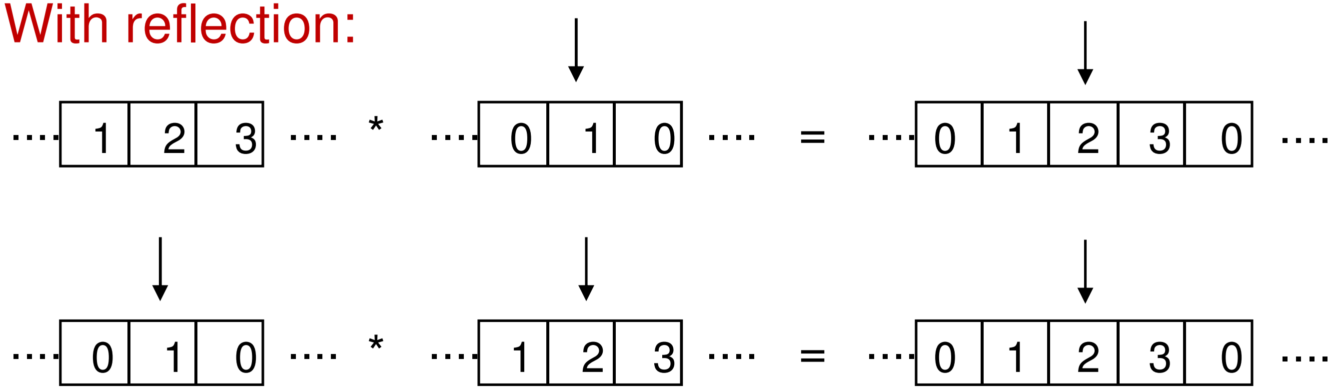
1/3	1/3	1/3

A*B:

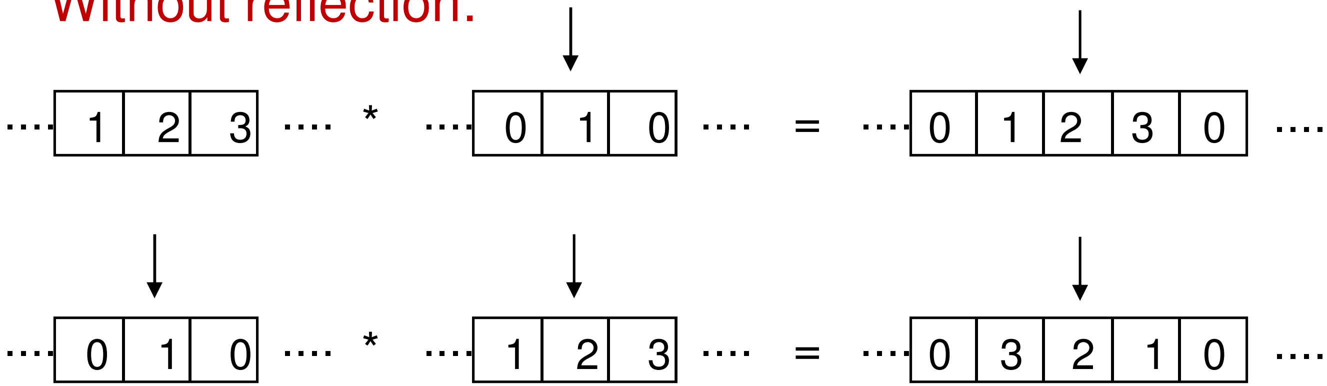
1	2	3	4	3

- Why should we flip the mask before the convolution?

With reflection:



Without reflection:



- Reflection is needed so that convolution is commutative:

A*B=B*A

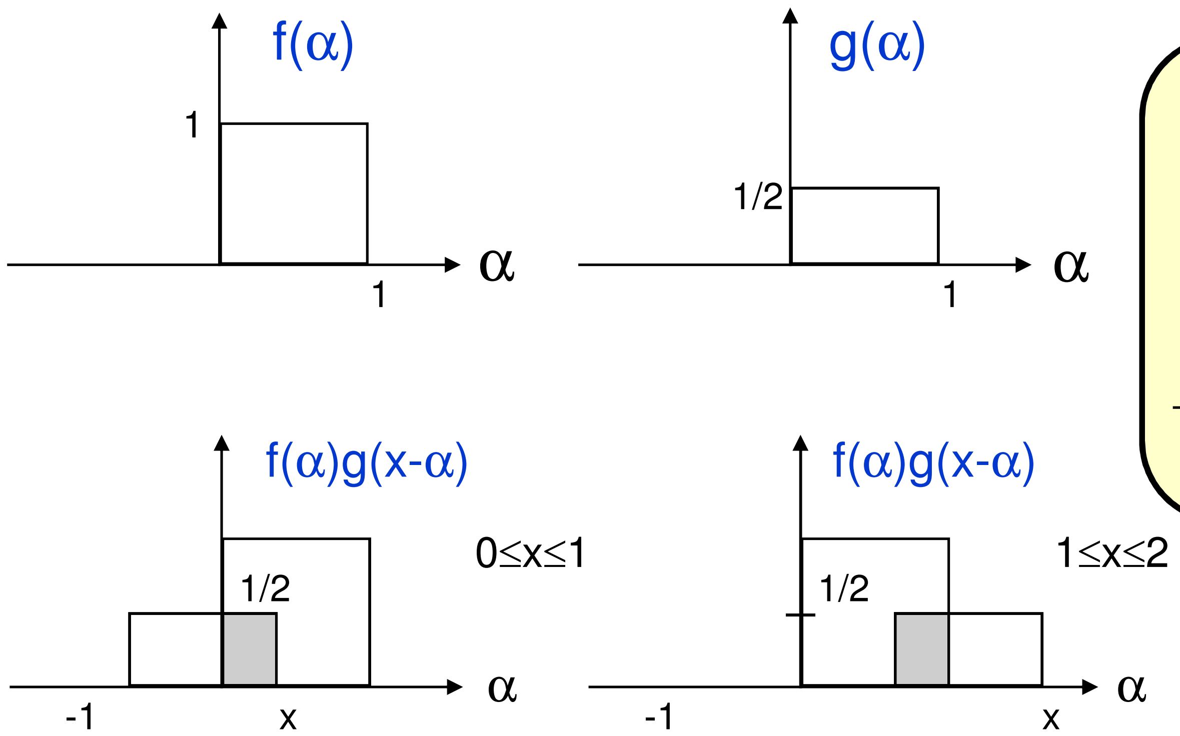
Correlation

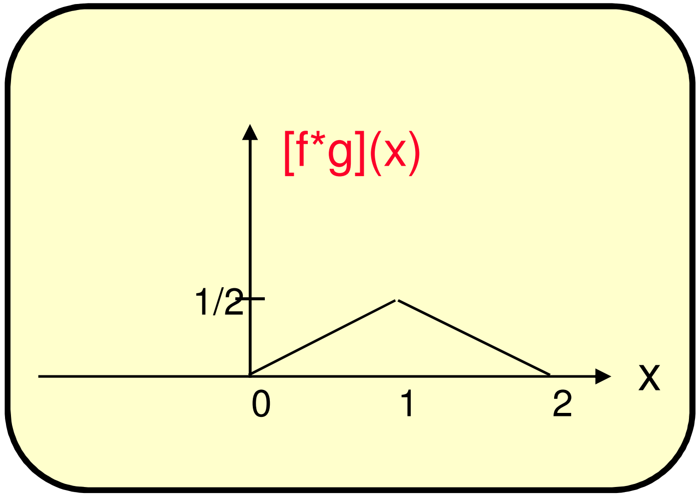
$$(A \circ B)(x) = \sum_i A(i)B(i-x)$$

$$A \circ B \neq B \circ A$$

Convolution: 1D continuous case

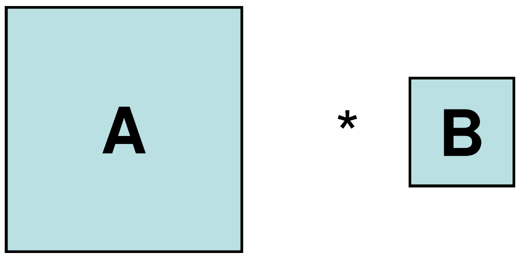
$$(f * g)(x) = \int_{-\infty}^{\infty} f(\alpha)\, g(x-\alpha)\, d\alpha$$

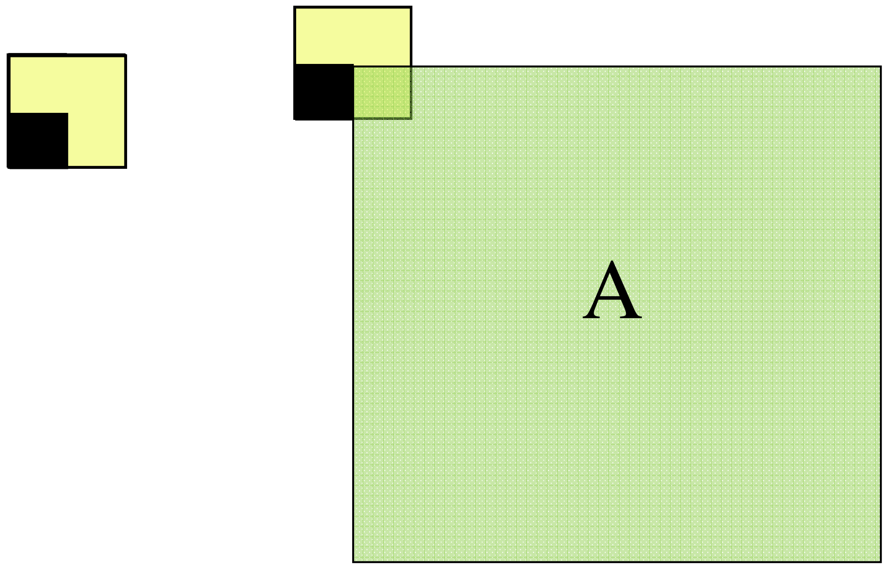


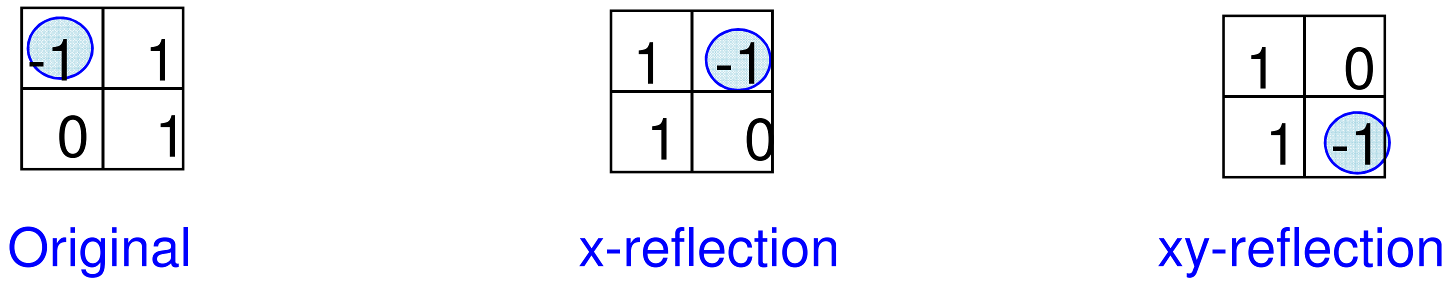


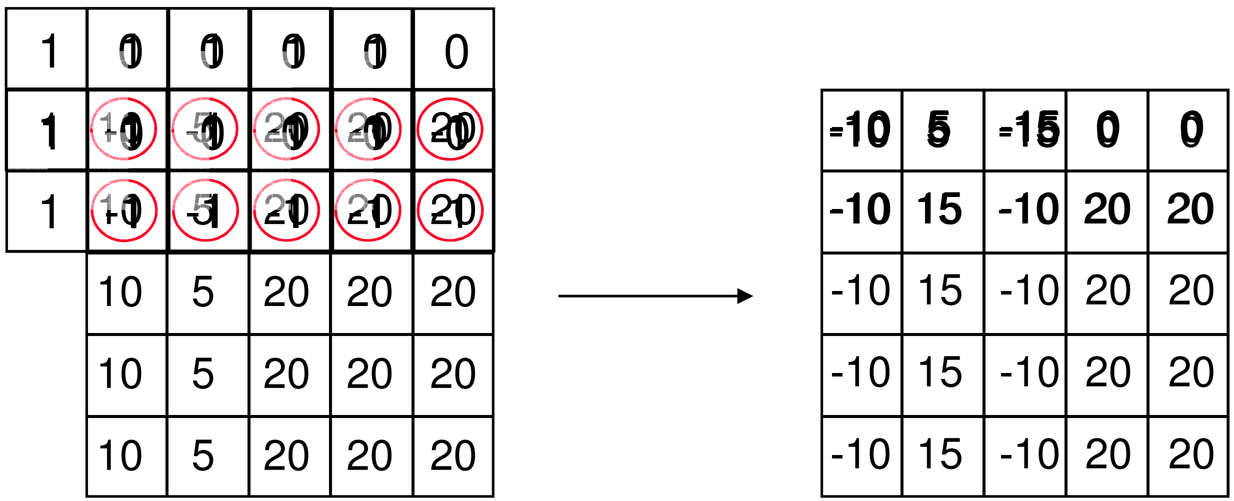
2D convolution

$$(A*B)(x,y) = \sum_{i,j} A(i,j)\, B(x-i,y-j)$$









(zero padding)

2D Convolution in Matlab

C = CONV2(A, B)
performs the 2-D convolution of matrices A and B.
If [ma,na] = size(A) and [mb,nb] = size(B), then size(C) = [ma+mb-1,na+nb-1].

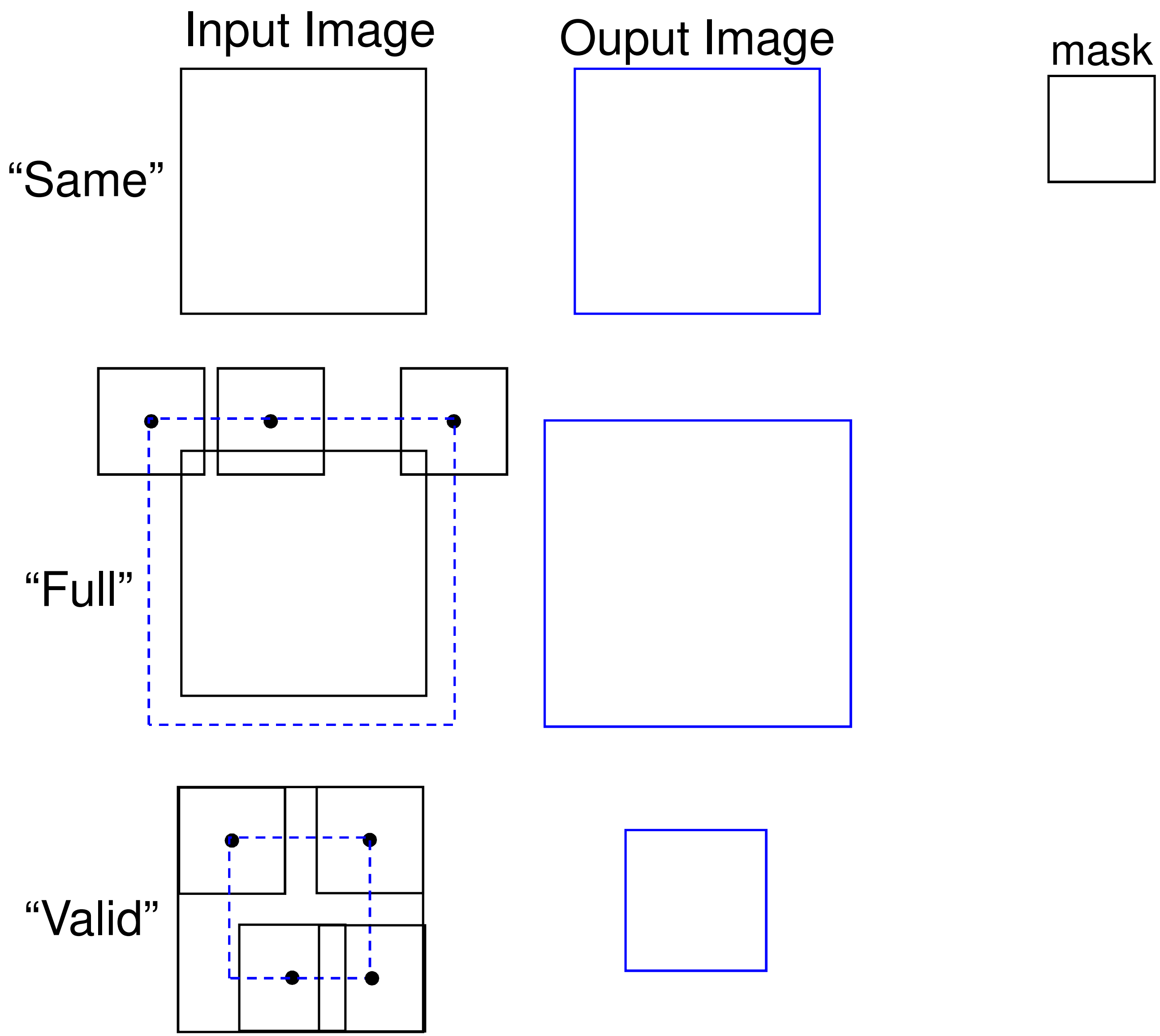
C = CONV2(HCOL, HROW, A)
convolves A separable with HCOL in the column direction and HROW in the row direction. HCOL and HROW should both be vectors.

C = CONV2(... ,'shape')
returns a subsection of the 2-D convolution with size specified by 'shape':

'full' - (default) returns the full 2-D convolution,
'same' - returns the central part of the convolution that is the same size as A.
'valid' - returns only those parts of the convolution that are computed without the zero-padded edges, size(C) = [ma-mb+1,na-nb+1] when size(A) > size(B).

CONV2 is fastest when size(A) > size(B).

2D Convolution in Matlab – Output size



Grayscale Convolution – Examples
The Delta Kernel

$$\delta(x - x_0) = \begin{cases} 1 & \text{if } x = x_0 \\ 0 & \text{otherwise} \end{cases}$$

$\delta(x) =$

0	1	0

$$A(x) * \delta(x) = A(x)$$

$$\delta(x - x_0, y - y_0) = \begin{cases} 1 & \text{if } x = x_0 \ \& \ y = y_0 \\ 0 & \text{otherwise} \end{cases}$$

$\delta(x, y) =$

0	0	0
0	1	0
0	0	0

$$A(x,y) * \delta(x,y) = A(x,y)$$

- Due to shift-invariance:

$$A(x,y)*\delta(x-x_0,y-y_0) = A(x-x_0,y-y_0)$$

δ(x,y)

0	0	0
0	1	0
0	0	0

δ(x-1,y-1)

0	0	0
0	0	0
0	0	1

A

1	2	3
4	5	6
7	8	9

*

δ(x-1,y-1)

0	0	0
0	0	0
0	0	1

=

A(x-1,y-1)

0	0	0
0	1	2
0	4	5

(Zero padding)

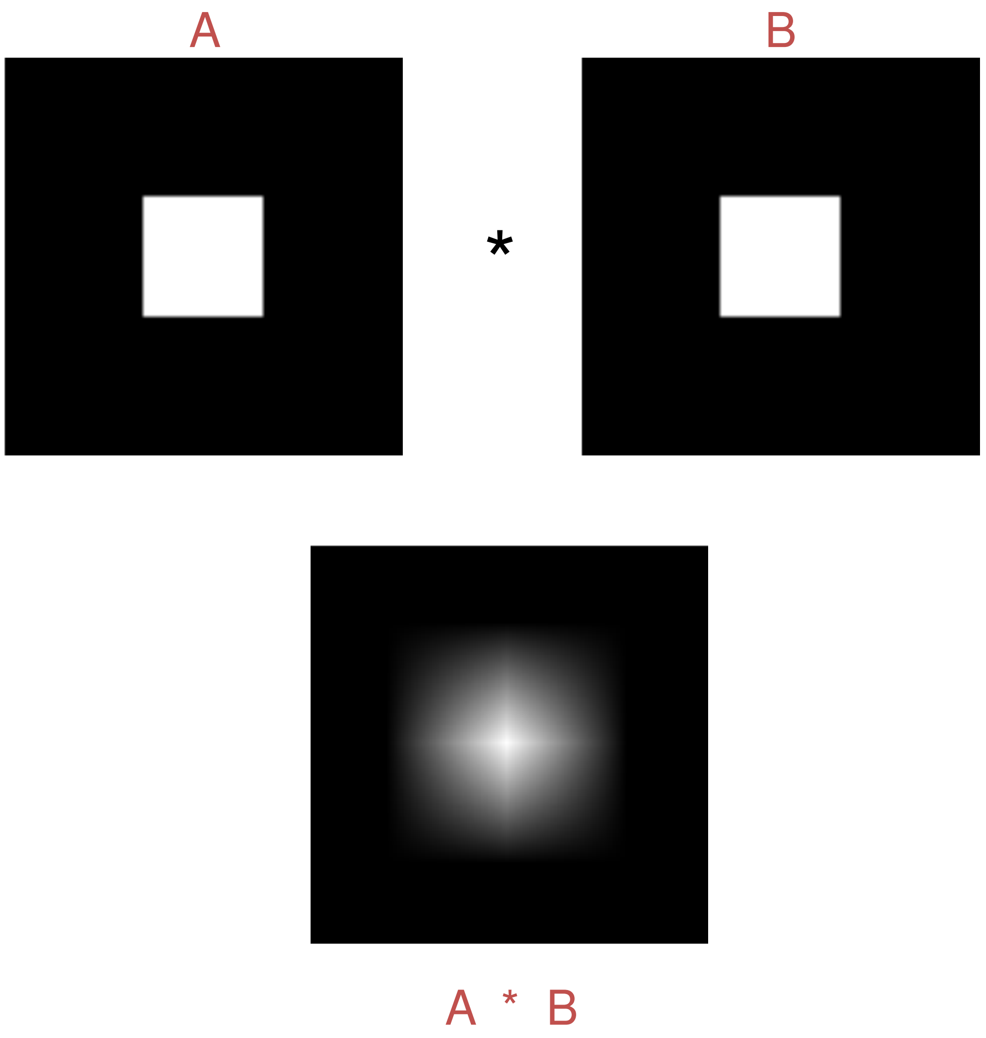
=

A(x-1,y-1)

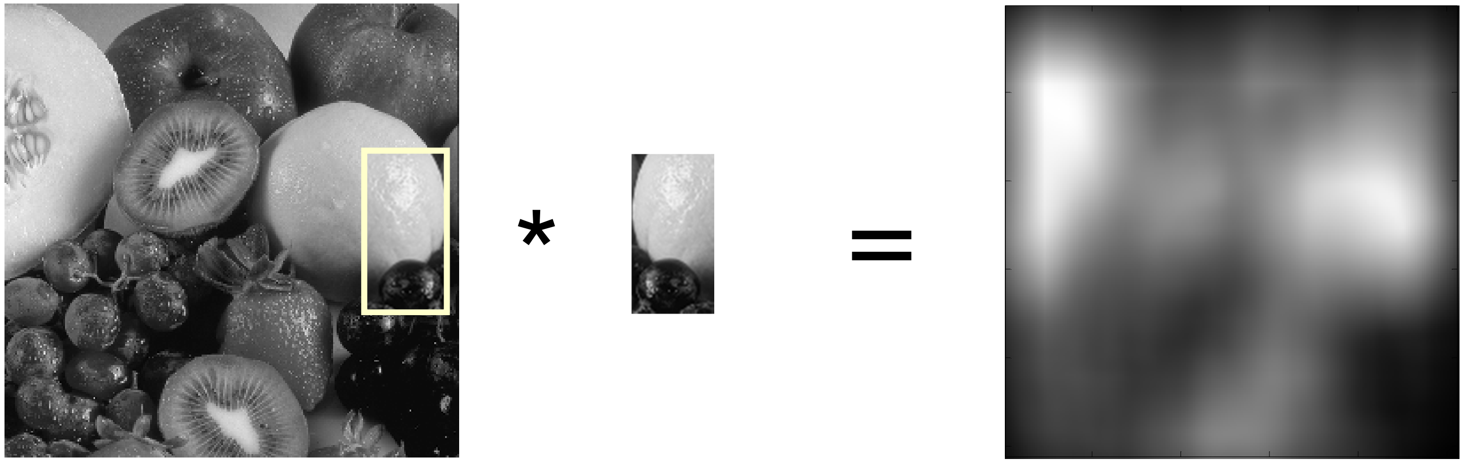
9	7	8
3	1	2
6	4	5

(Wrap around)

Grayscale Convolution - Example

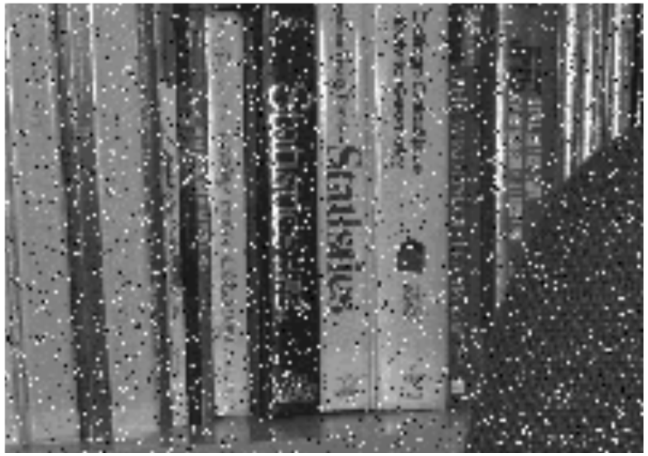


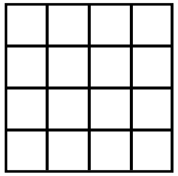
Grayscale Convolution - Examples



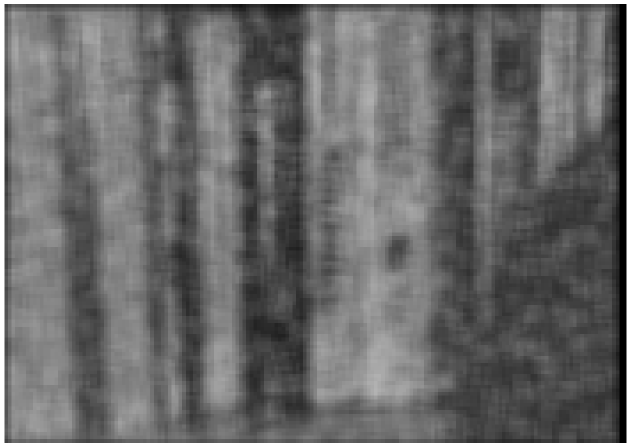
Convolution Examples - Oriented Filters

Salt & Pepper noise

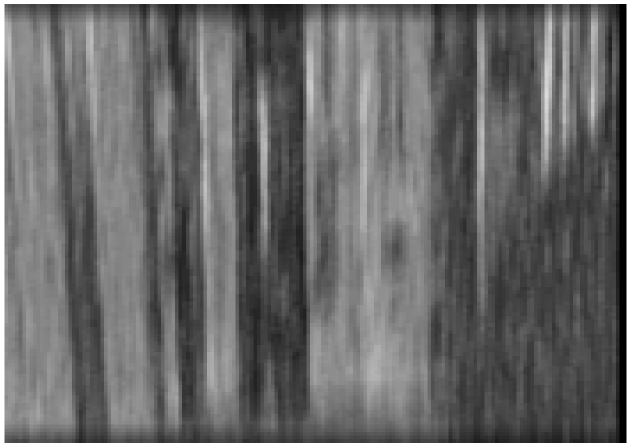


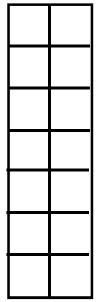


4x4 Average



7x2 Average





Convolution Properties

- Commutative:

  $A*B = B*A$

- Associative:

  $(A*B)*C = A*(B*C)$

- Linear:

  $A*(\alpha B+\beta C)= \alpha A*B+ \beta A*C$

- Shift-Invariant

  $A*B(x-x_0,y-y_o)= (A*B)\ (x-x_0,y-y_o)$

Convolution Complexity

- Assume **A** is nxn and **B** is kxk then

  A*B takes $O(n^2k^2)$ operations.

  (applying with FFT takes $O(N^2 \log n)$)

- (A*B)*C = A*(B*C)
  - If B and C are kxk then

    (A*B)*C takes $O(2n^2k^2)$ operations

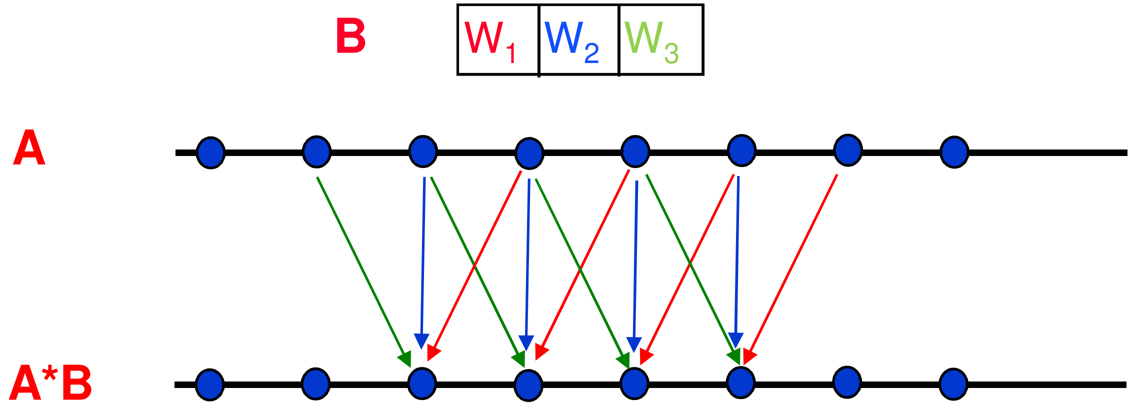
    while A*(B*C) takes $O(k^4+4n^2k^2)$ operations.
- Separability
  - In some cases it is possible to decompose B (kxk) into B=C*D where C is 1xk and D is kx1.

    In such a case A*B takes $O(n^2k^2)$

    while (A*C)*D takes $O(2n^2k)$.

$$\begin{bmatrix} 1 & -1 \end{bmatrix} * \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

The Image average



$$sum(A*B(x)) = W_1 sum(A(x)) + W_2 sum(A(x)) + W_3 sum(A(x))$$

$$= (W_1 + W_2 + W_3)\ sum(A(x))$$

If $W_1+W_2+W_3 = 1$ then $Av(A)=Av(A*B)$

To maintain the average - sum of elements of B must equal 1.

In General: sum(A*B) = sum(A) x sum(B)

Blurring Kernels (low pass)

- Averaging kernels:

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

3 X 3

1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25

5 X 5

- Gaussian kernels (soft blurring):

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

1/6 x

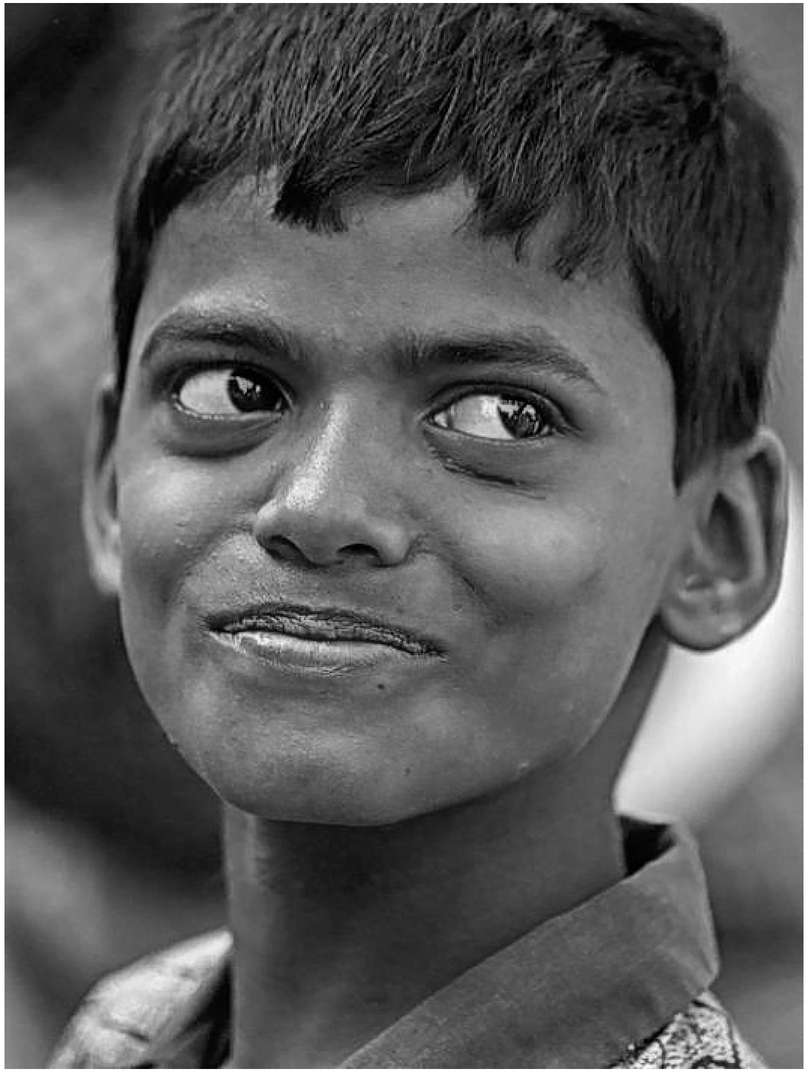
0	1	0
1	2	1
0	1	0

1/81 x

1	2	3	2	1
2	4	6	4	2
3	6	9	6	3
2	4	6	4	2
1	2	3	2	1

- Both are separable kernels.

Original image

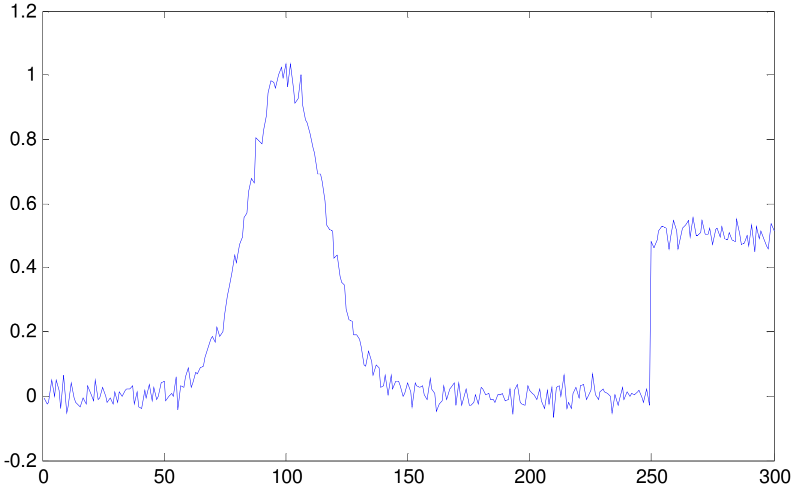


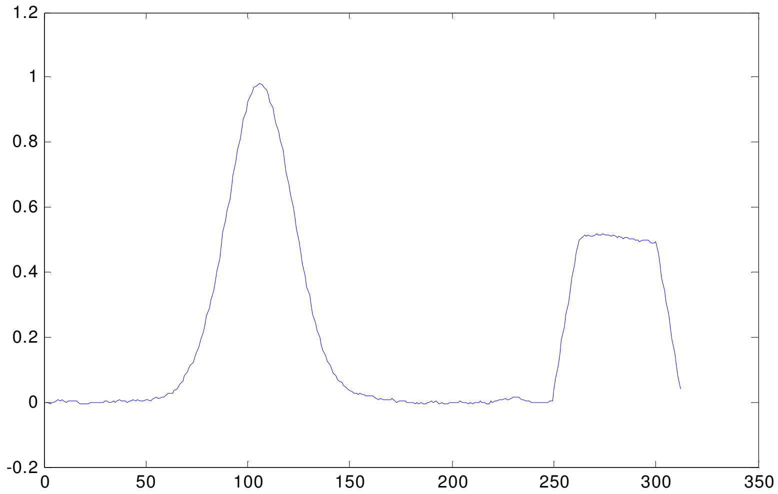
Gaussian blur with $\sigma=5$

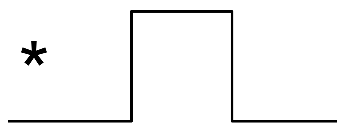
Gaussian blur with σ=9

Image De-noising by Filtering

- Zero mean additive noise can be attenuated by smoothing the image.
- Trade off: Edges and high frequencies are smoothed as well.







Noisy Images

Original

Gaussian noise

salt & pepper

Salt & Pepper Noise:

Median filter
3x3 window

Gaussian blur
std=1.5

Gaussian Noise:

Median filter
5x5 window

Gaussian blur
std=3

Edge Enhancement by Filtering

A sharpening filter is applied in order to enhance edges and fine details (high frequency) in an image:

Assume: A is a sharp image.

G is a Gaussian mask.

$B = A * G$ is a blurred image.

Sharpen B: $B_{sharp} = A - B$

Sharpened B $= B + B_{sharp}$

Problem: A and G are unknown.

Edge Enhancement by Filtering

A sharpening filter is applied in order to enhance edges and fine details (high frequency) in an image:

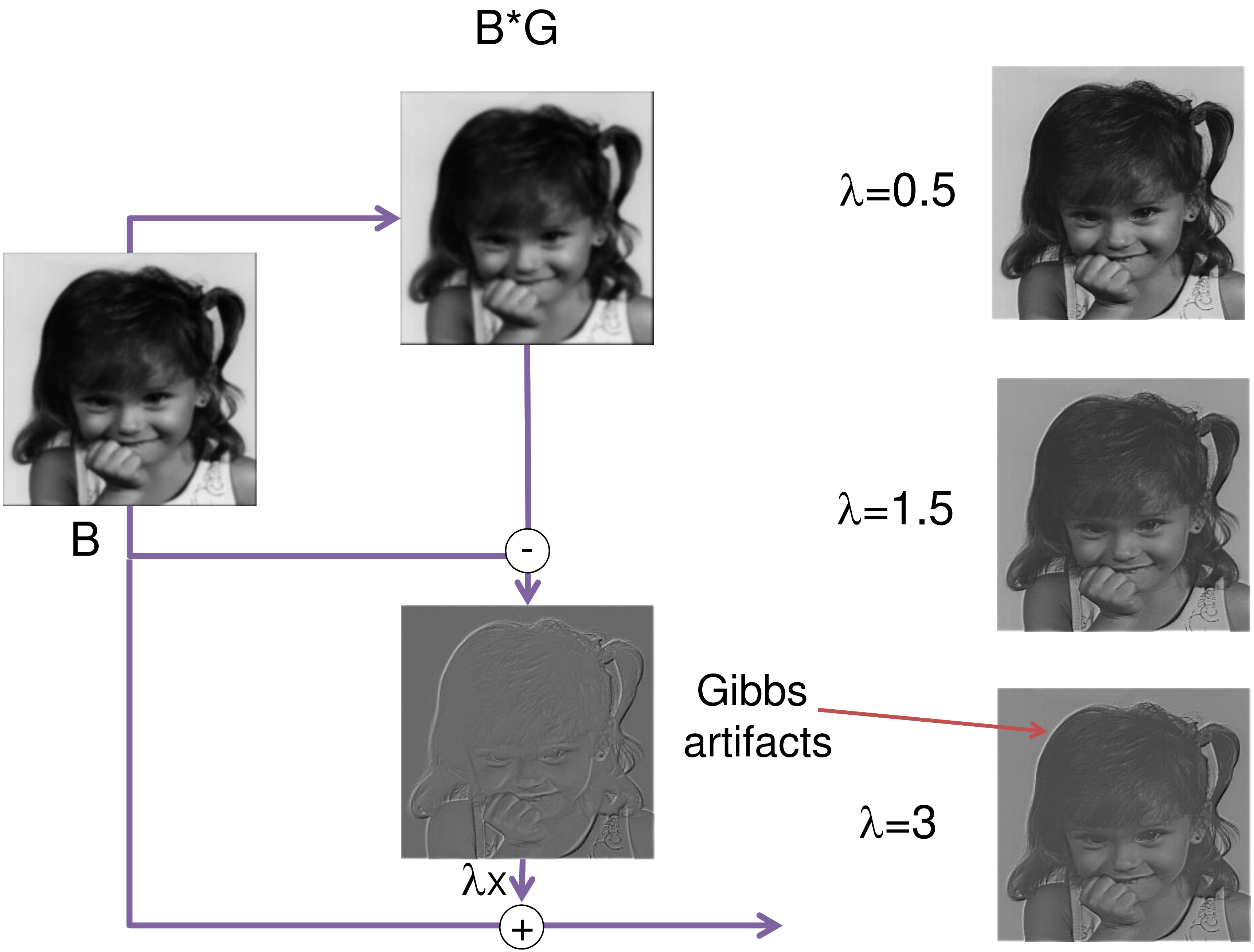
- Assume B is an image to be enhanced.
- Define: $B_{Blur}=B*G$ is a blurred image, where G is a blurring mask.
- $B_{sharp}=B-B_{Blur}=B*(\delta - G)$ contains fine details of image B.
- $B+\lambda B_{sharp} = B*(\delta+\lambda(\delta-G))=B*S(\lambda)$ amplifies fine details image.
- The parameter λ controls the amount of amplification.

G =

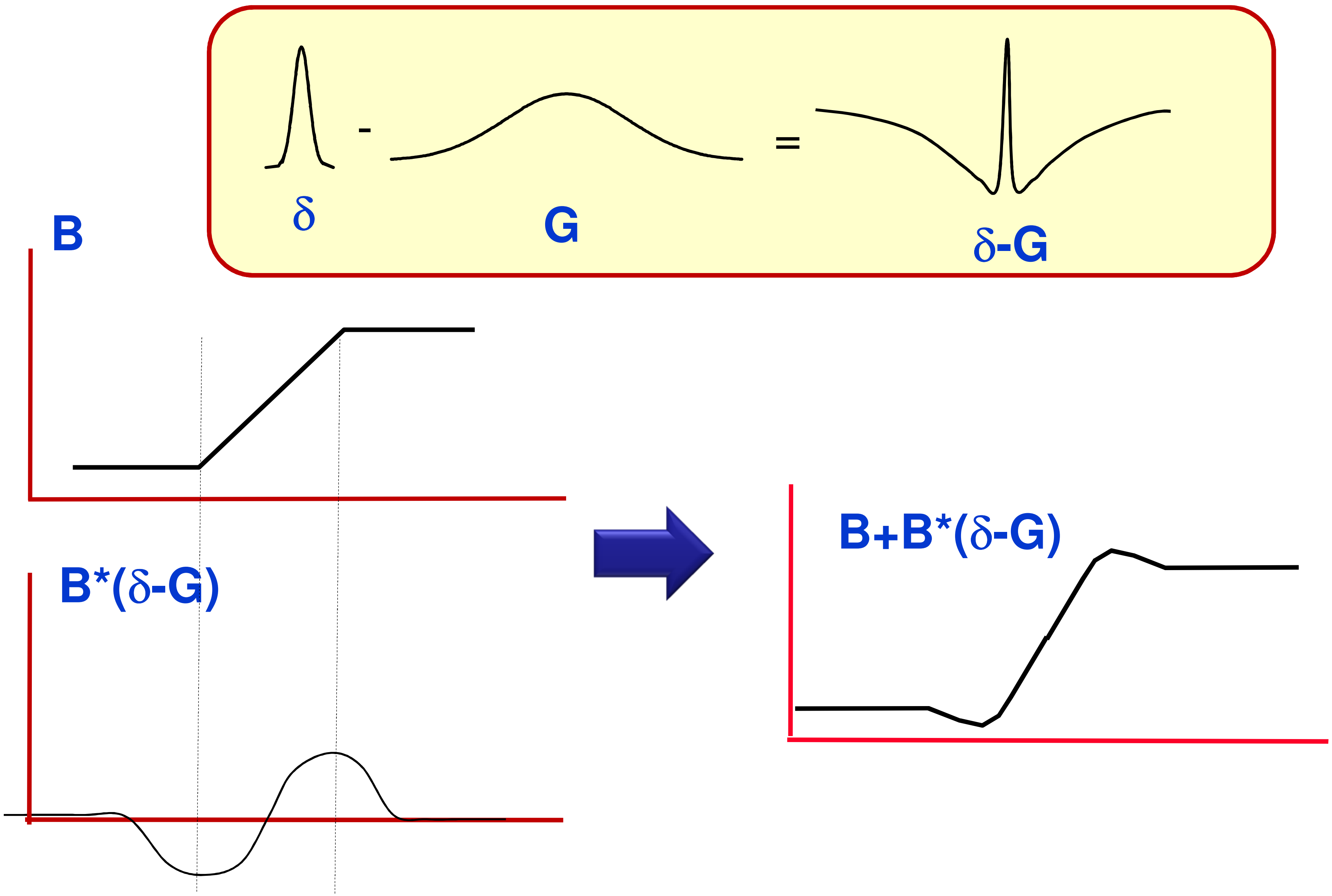
0	1/6	0
1/6	2/6	1/6
0	1/6	0

S(1) =

0	-1/6	0
-1/6	10/6	-1/6
0	-1/6	0


B*G
B
-
λx
+
Gibbs artifacts
λ=0.5
λ=1.5
λ=3

Gibbs Artifacts



Sharpening - Example

Original

Blur

$\lambda = 2$

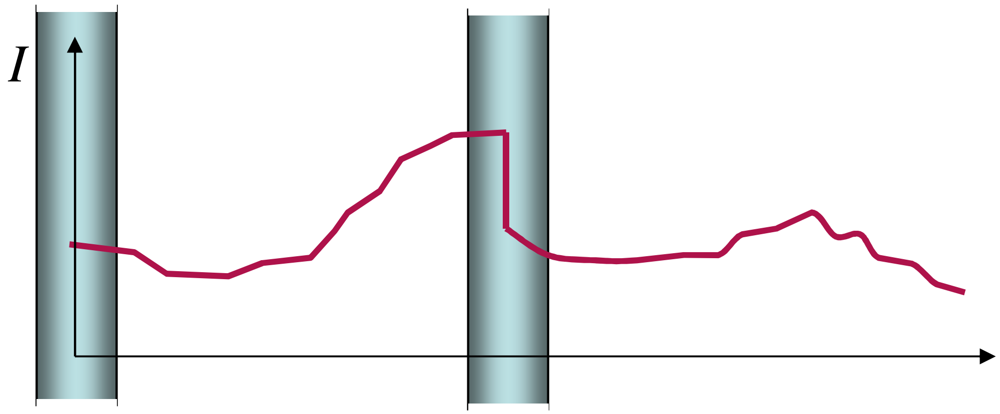
$\lambda = 8$

$\lambda = 16$

Adaptive Filtering

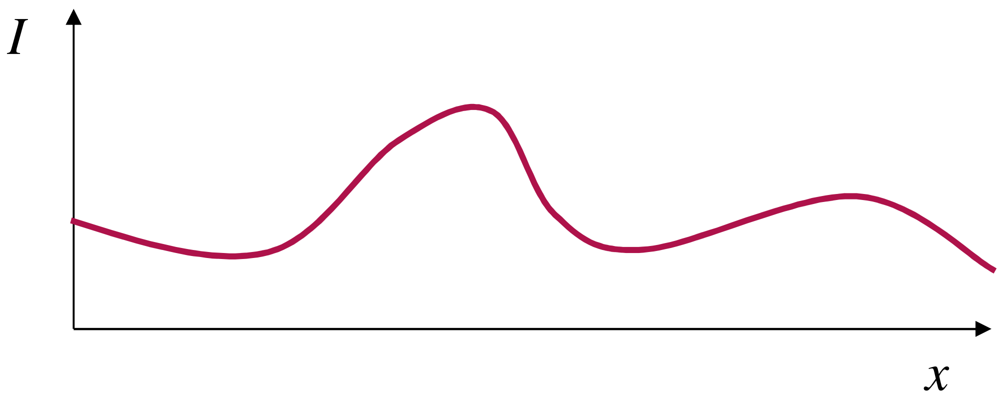
- The convolution is a *non-adaptive* filtering in the sense that the convolution mask is space invariant.
- *Adaptive* filtering refers to image operations that adapt their performance based on the input signal.
- Example for adaptive-filtering: The Bilateral Filter

Gaussian Filter



$$\hat{f}_p = \sum_{j \in \Omega} W_s(p-j) f_j$$

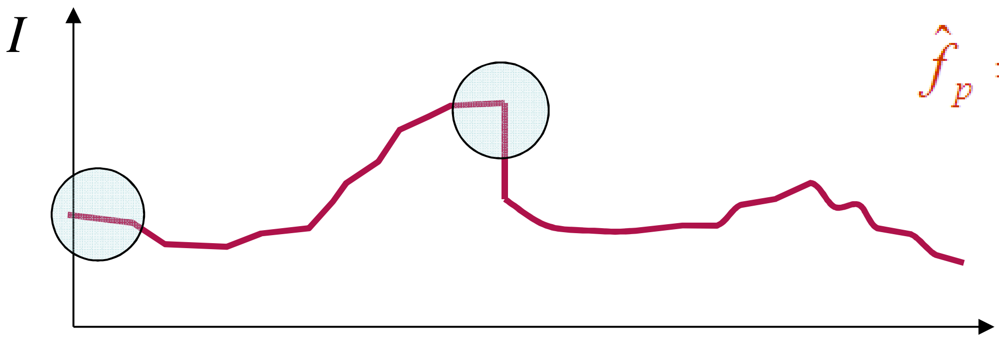
$$W_s(t) = e^{-\frac{t^2}{2\sigma_s^2}}$$



Smooth edges

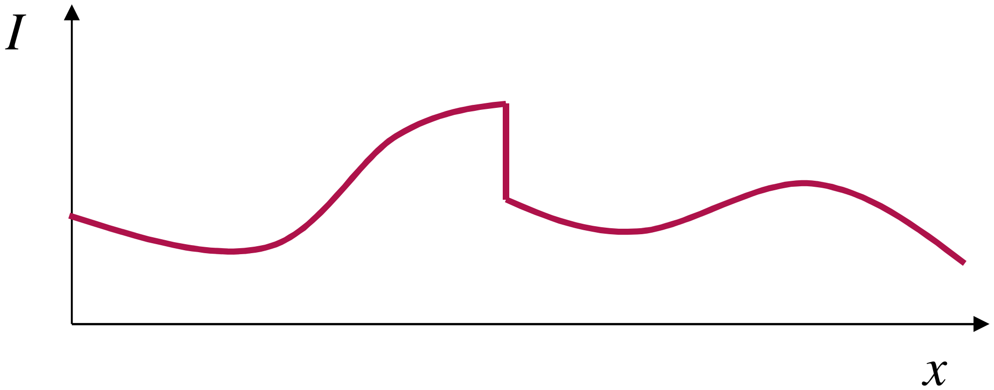
slide from Darya Frolova and Denis Simakov

Bilateral Filter



$$\hat{f}_p = \frac{\sum_{j\in\Omega} W_s(p-j)\,W_p(f_p - f_j)\,f_j}{\sum_{j\in\Omega} W_s(p-j)\,W_p(f_p - f_j)}$$

$$W_p(t) = e^{-\frac{t^2}{2\sigma_p^2}}$$



Preserves discontinuities

slide from Darya Frolova and Denis Simakov

Gaussian Filter

In convolution filtering neighboring pixels are weighted according to their spatial distance:

$$\hat{I}(x) = \sum_{j \in N_p} W_s(x-j)\, I(j)$$

$$W_s(x-j) = e^{-\left(\frac{x-j}{\sigma_{sp}}\right)^2}$$

Bilateral Filter

In bilateral filtering the weights are determined according to spatial and photometric distances:

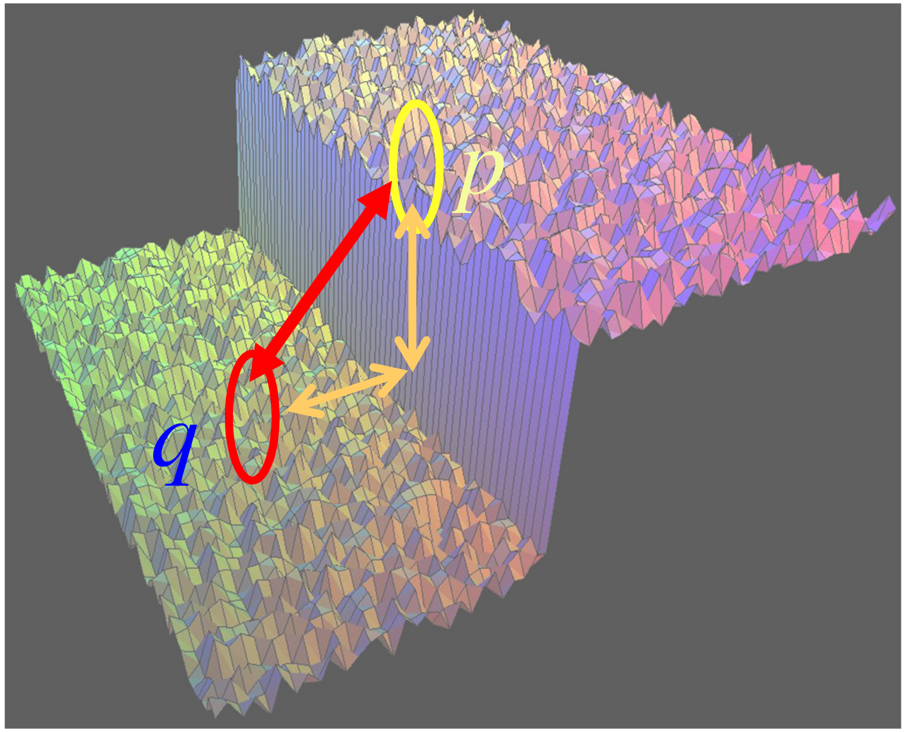
$$\hat{I}(x) = \frac{\sum_{j \in N_p} W_s(x-j) W_p(I(x)-I(j)) I(j)}{\sum_{j \in N_p} W_s(x-j) W_p(I(x)-I(j))}$$

$$W_s(x-j) = e^{-\left(\frac{x-j}{\sigma_{sp}}\right)^2} \qquad W_p(I(x)-I(j)) = e^{-\left(\frac{I(x)-I(j)}{\sigma_{ph}}\right)^2}$$

Typical bilateral weighting functions:

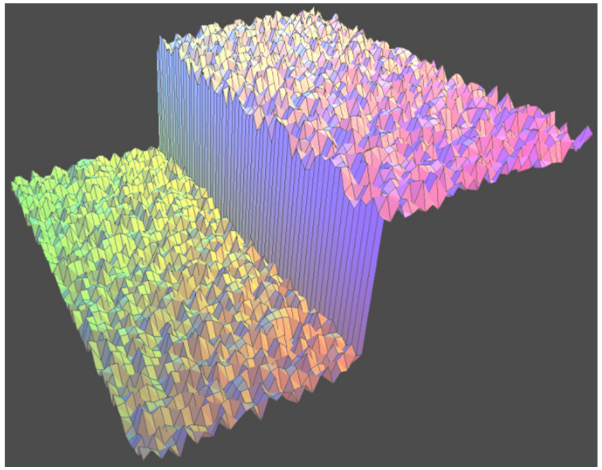
$$W_s(p-q) = e^{-\left(\frac{p-q}{2\sigma_s}\right)^2}$$

$$W_p(f_p - f_q) = e^{-\left(\frac{f_p - f_q}{2\sigma_p}\right)^2}$$

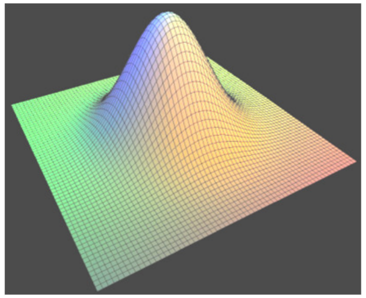


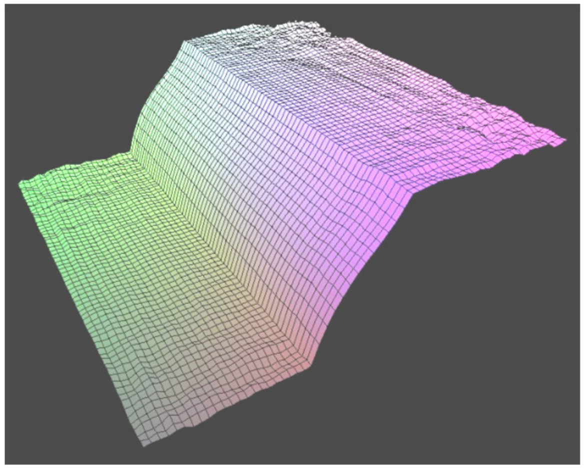
Slide from F. Durand

Gaussian Filtering:



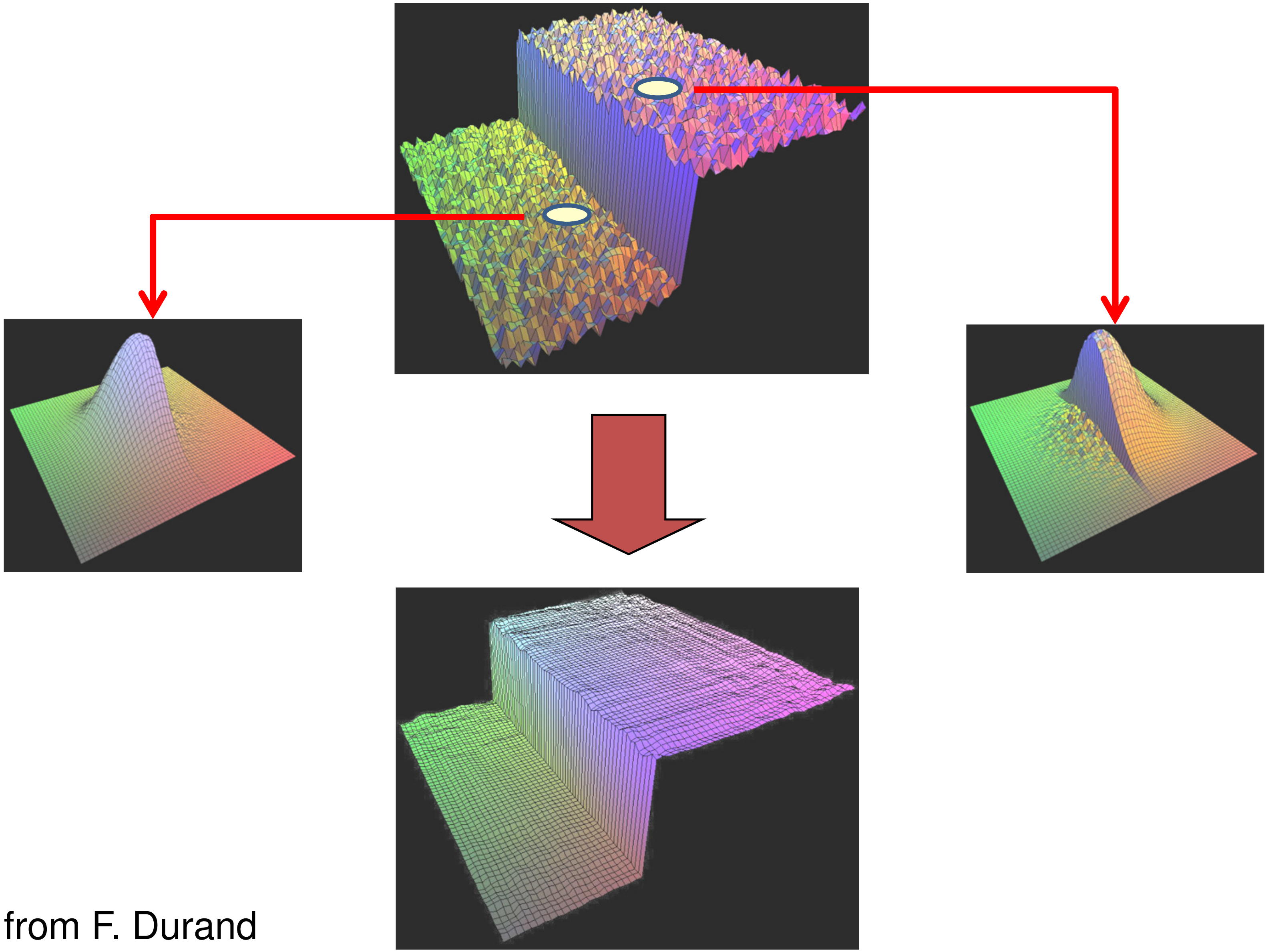
*





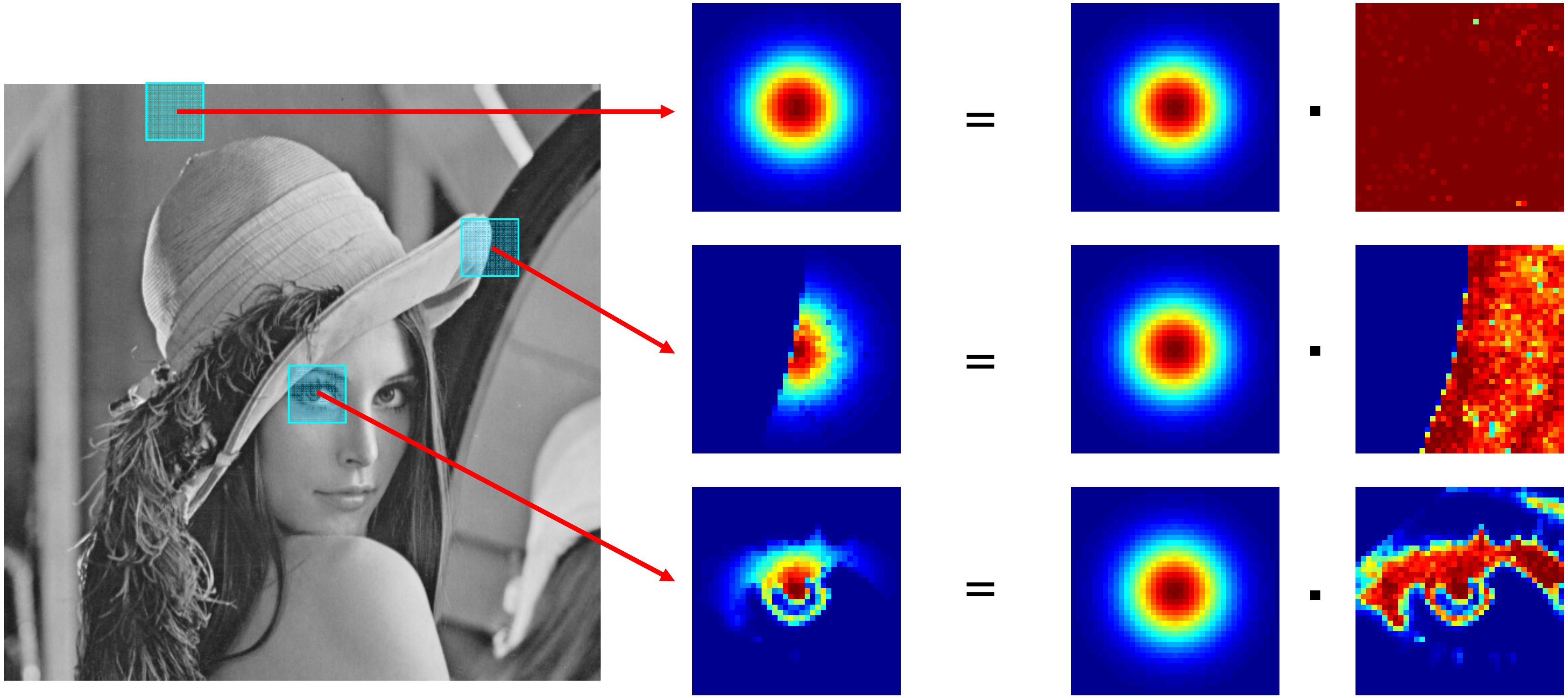
Slide from F. Durand

Bilateral Filtering:



Slide from F. Durand

Bilateral weights:



from P. Milinfar.

Gaussian Smoothing:

Bilateral (edge-preserving) Smoothing:


Noisy Image
SONY
WHISKEY
BECHEROVKA


Gaussian Smoothing


Bilateral Smoothing
BECHEROVKA

Noisy Lena

Median Lena

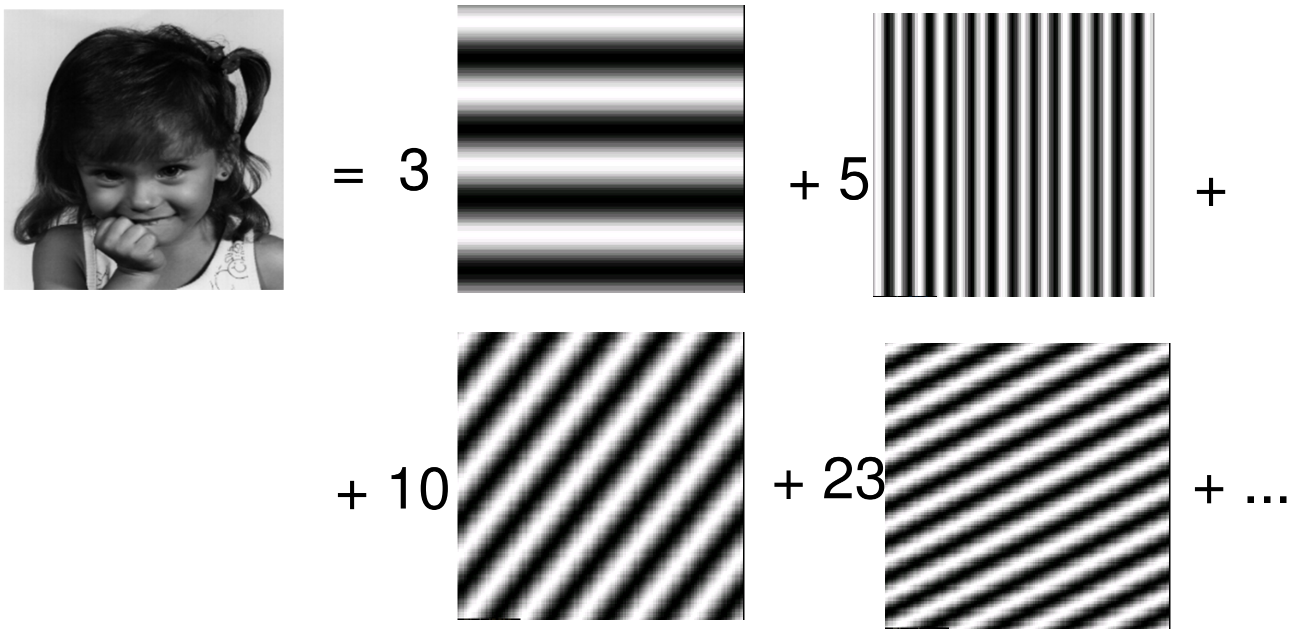
Modern Denoising approach (DUDE)

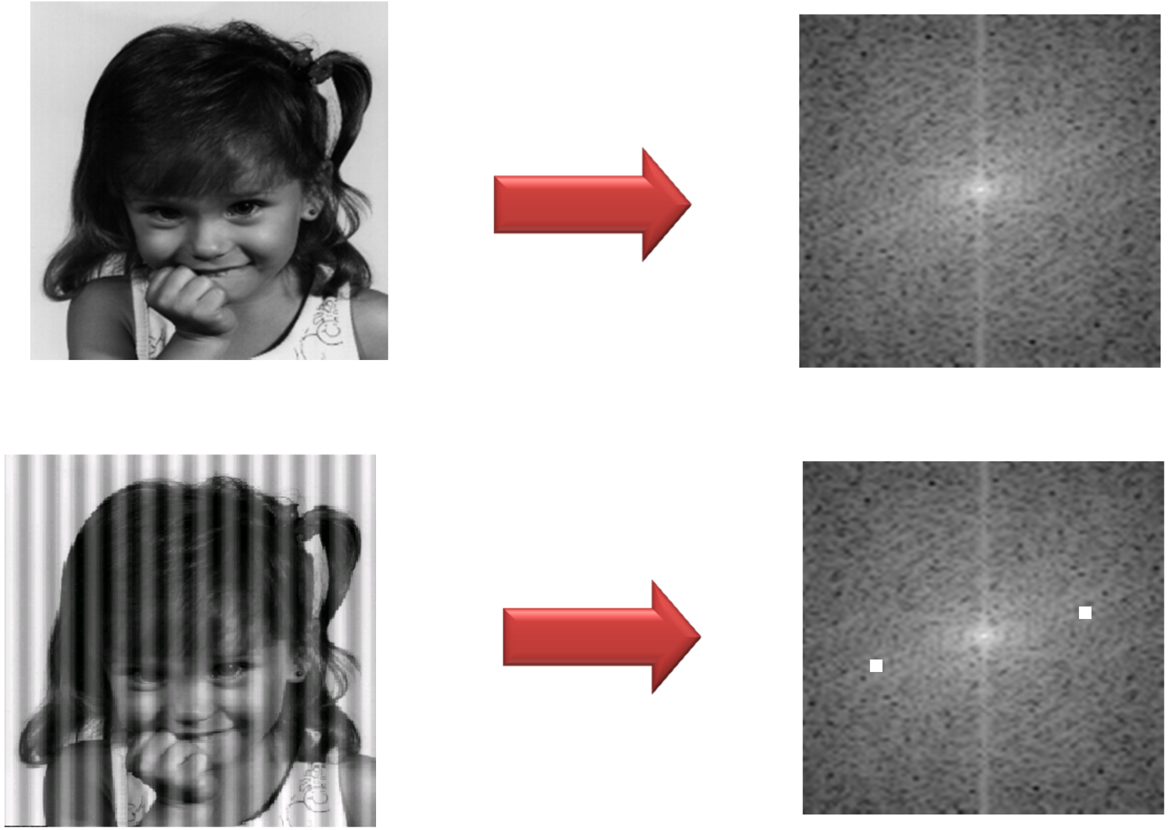
Dude Lena

How can we enhance such an image?

Solution: Image Representation

$$\begin{bmatrix}2&1&3\\5&8&7\\0&3&5\end{bmatrix} = 2\begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix} + 1\begin{bmatrix}0&1&0\\0&0&0\\0&0&0\end{bmatrix} + 3\begin{bmatrix}0&0&1\\0&0&0\\0&0&0\end{bmatrix} + 5\begin{bmatrix}0&0&0\\1&0&0\\0&0&0\end{bmatrix} + \ldots$$


= 3
+ 5
+
+ 10
+ 23
+ ...



- Global phenomena becomes local
- Spatial correction in possible in the new representation
- Stay tuned....

T H E E N D