Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

Introduction to Computer Vision:

Local Operations: Convolution

Marcus Hudritsch (hsm4)

Image Operators: Local Operators

Point Operators

- Binarisation
- Gray level reduction
- Contrast & Brightness manipulations
- Histogram Equalization
- Arithmetic Operations
- Logic Operations

Global Operators

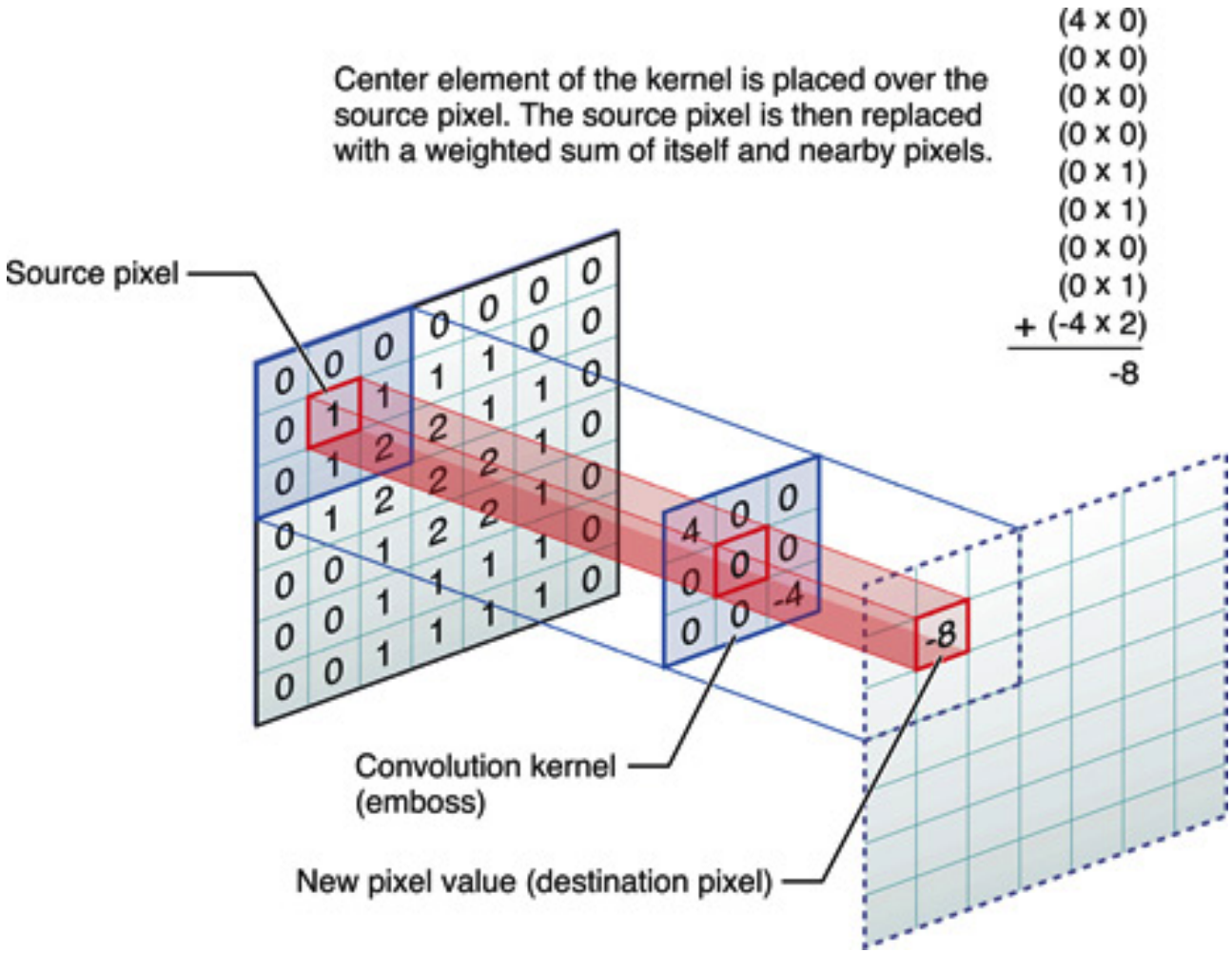
- Discrete Fourier Transform
- Wavelet Transform
- Hough Transform
- Principal Component Transform

Local Operators

- Filters
  - Low Pass Filter
    - Gauss & Box Filter
  - High Pass Filter
    - Sobel Filter
    - Laplace Filter
- Morphological Operators
  - Erosion & Dilation
  - Opening & Closing
- Rank Order Operators
  - Min. & Max. Filter
  - Median Filter

Local Operators: Filter with Convolution

- Local Operators build the sum of a **filter mask** values multiplied by the **underlying pixel** values into the **destination pixel**.
- The filter mask is normally **symmetric** and of **odd pixel size** (3x3, 5x5 etc.).



Local Operators: Filter with Convolution

- The mathematical operation behind is called **convolution**.
- In German it is called **Konvolution** or **Faltung**.
- The filter mask is also called **convolution kernel**.
- An **image *g*** convolved with a **convolution kernel *h*** with a **kernel radius *k*** (size=2·k+1) is therefore:

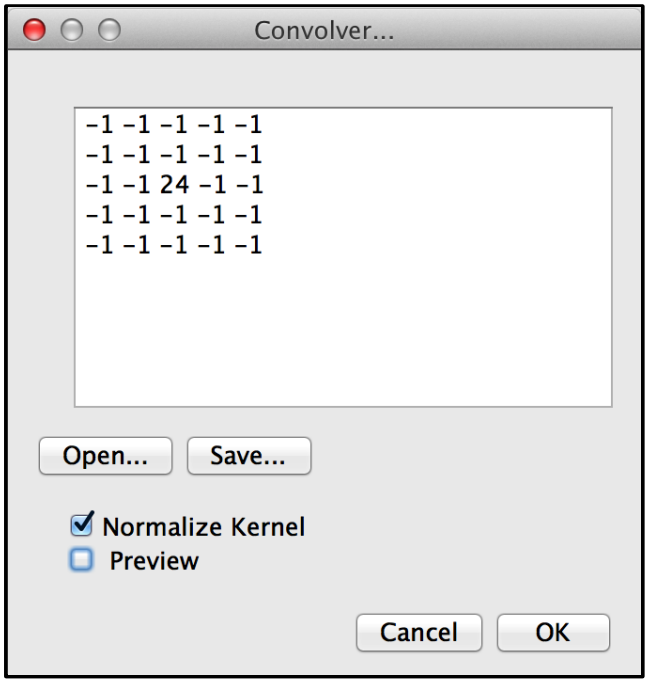
$$f(x,y) = g * h = \sum_{u=-k}^{k} \sum_{v=-k}^{k} g(x+u, y+v) \cdot h(u,v)$$

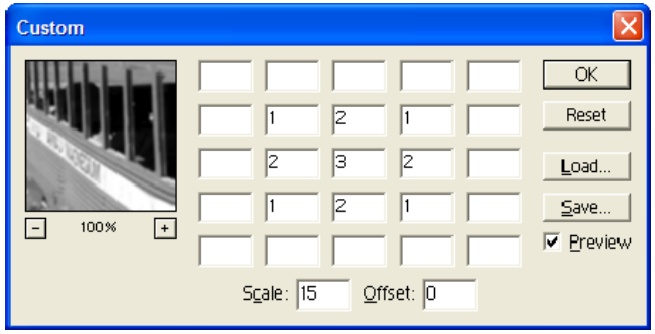
- For a 3x3 kernel mask this is:

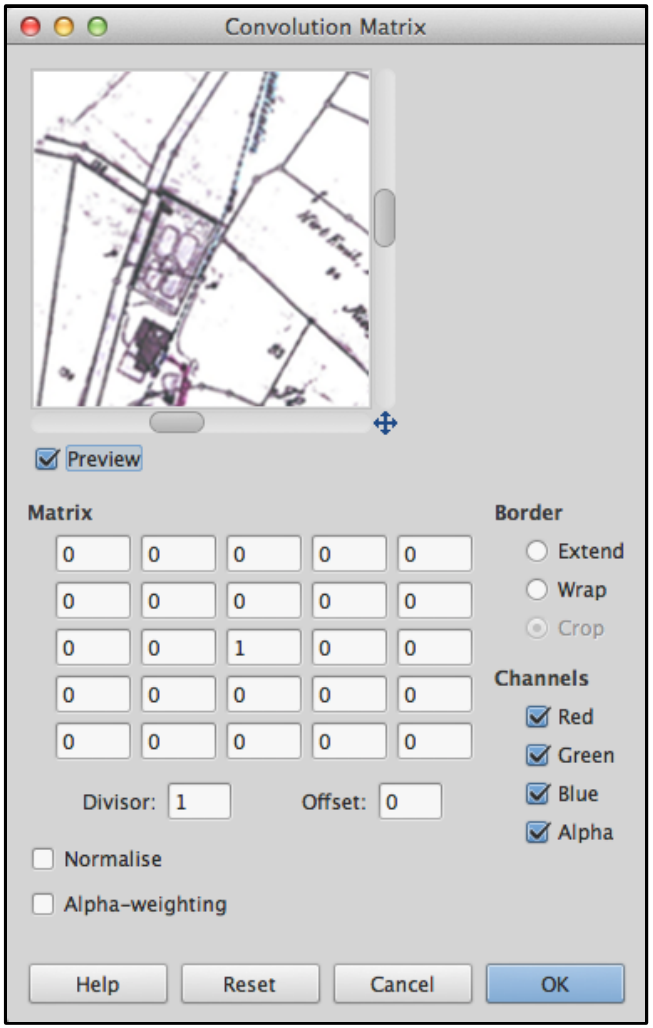
```
f(X,Y) = g(X-1,Y-1)·h(-1,-1) + g(X,Y-1)·h(0,-1) + g(X+1,Y-1)·h(1,-1) +
         g(X-1,Y  )·h(-1, 0) + g(X,Y  )·h(0, 0) + g(X+1,Y  )·h(1, 0) +
         g(X-1,Y+1)·h(-1,+1) + g(X,Y+1)·h(0,+1) + g(X+1,Y+1)·h(1,+1)
```

Local Operators: Filter with Convolution

- Most image processing tools offer a dialog for a generic filter mask definition:

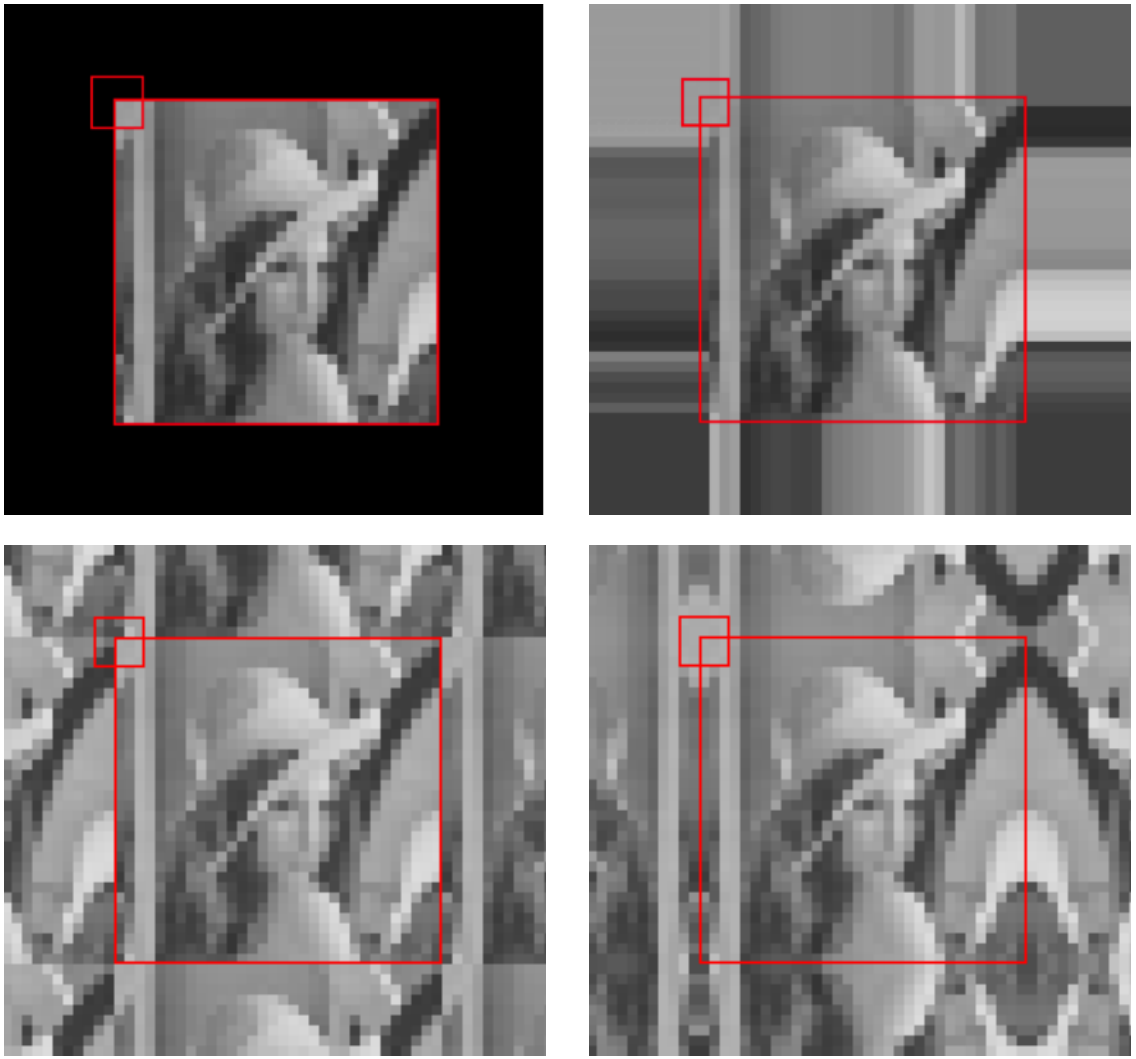






Local Operators: Filter with Convolution

- For a destination image with the same size we need pixels across the border.
- There are several solution for the **border problem**:



Local Operators: Filter with Convolution

- The mathematical definition of a linear **convolution** is slightly different:

$$f(x,y) = g * h = \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} g(x+u, y+v) \cdot h(-u, -v)$$

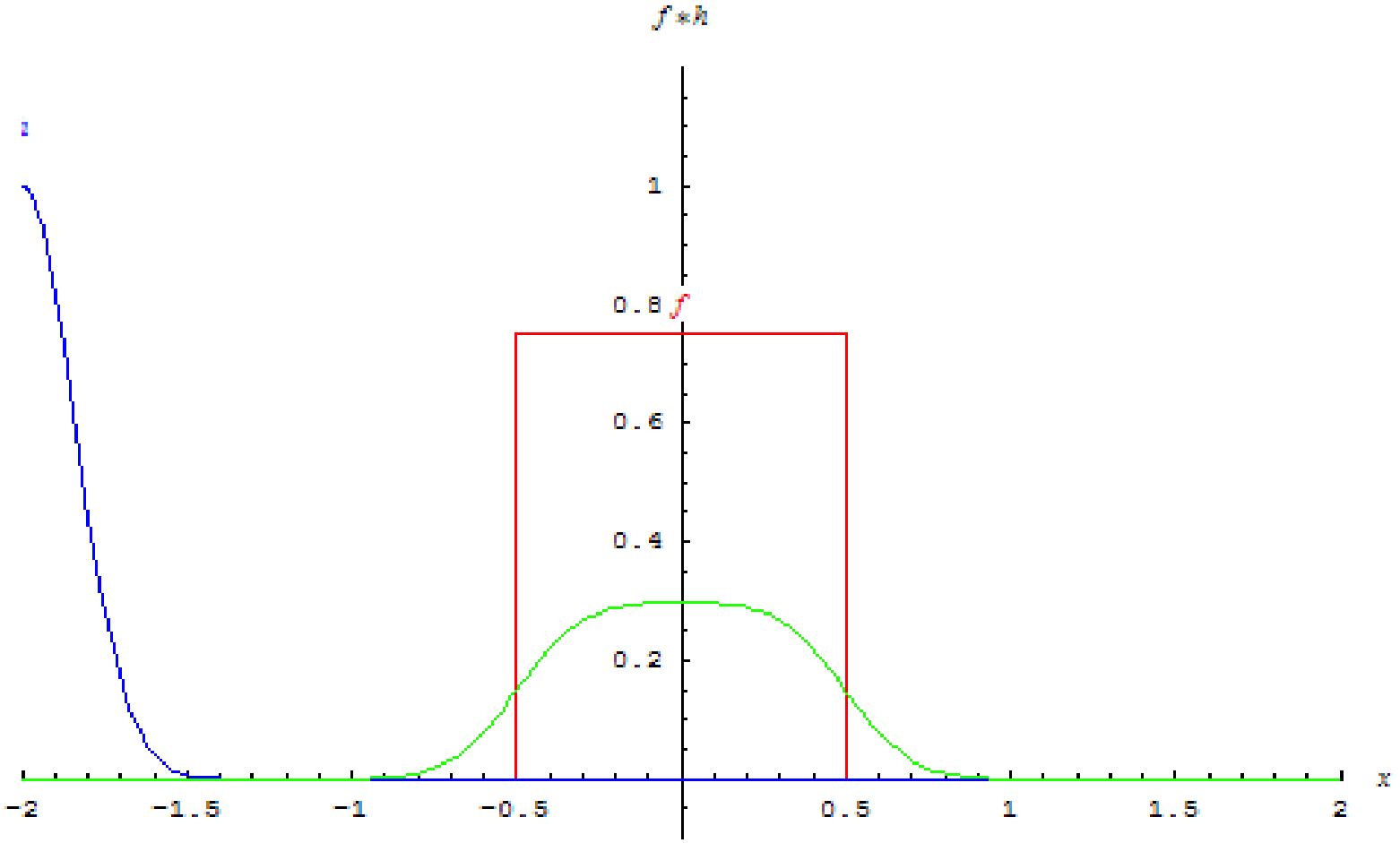
compaired to what we implemented:

$$f(x,y) = g * h = \sum_{u=-k}^{k} \sum_{v=-k}^{k} g(x+u, y+v) \cdot h(u, v)$$

- The different range (+/-∞) can be ignored because h(u,v) is zero outside.
- The **negative signes mirror the mask**.
  - Without this mirroring our convolution is in fact a **correlation**.

Local Operators: Filter with Convolution

- A convolution an operation between two functions that return a third function.
- One of the functions is slided over the other function.
- The convolution is the multiplied sum of the overlapping regions:



Local Operators: Convolution Properties

- A convolution is **commutative**: $g * h = h * g$

  We therefor can exchange the image with the convolution mask.

Local Operators: Convolution Properties

- A convolution is **commutative**: $g * h = h * g$
- A convolution is **associative**: $g * h * i = g * (h * i) = (g * h) * i$

  You can also therefore split a filtermask in two separate filters and apply them separately and more efficiently:

$$g * h_{xy} = g * (h_x * h_y) = (g * h_x) * h_y$$

$$h_{xy} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = h_x * h_y = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Local Operators: Convolution Properties

- A convolution is **commutative**: $g * h = h * g$
- A convolution is **associative**: $g * h * i = g * (h * i) = (g * h) * i$
- A convolution is **distributive**: $g * h + i * h = (g + i) * h$

This property is only true because the convolution is a **linear operation (filter)**:

We first convolve and then add two images:

```
[1 2 1] * [...0 0 1 0 0 0...] + [1 2 1] * [...0 0 0 1 0 0...] =
          [...0 1 2 1 0 0...] + [...0 0 1 2 1 0...] =
                    [...0 1 3 3 1 0...]
```

Or we add first and then convolve:

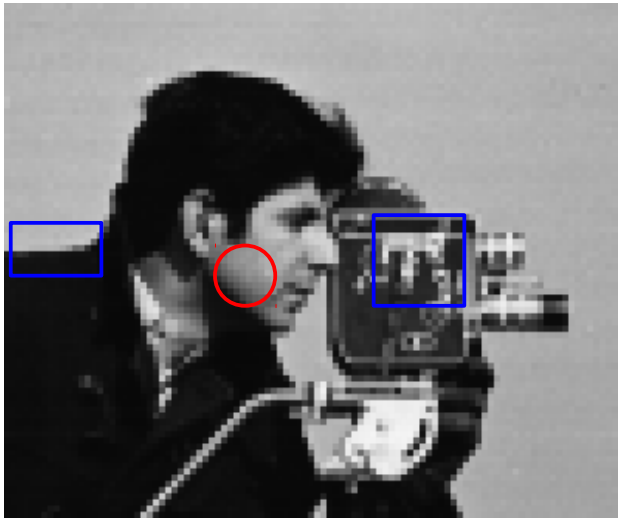
```
[1 2 1] * ([...0 0 1 0 0 0...] + [...0 0 0 1 0 0...]) =
                [1 2 1] *  [...0 0 1 1 0 0...] =
                      [...0 1 3 3 1 0...]
```

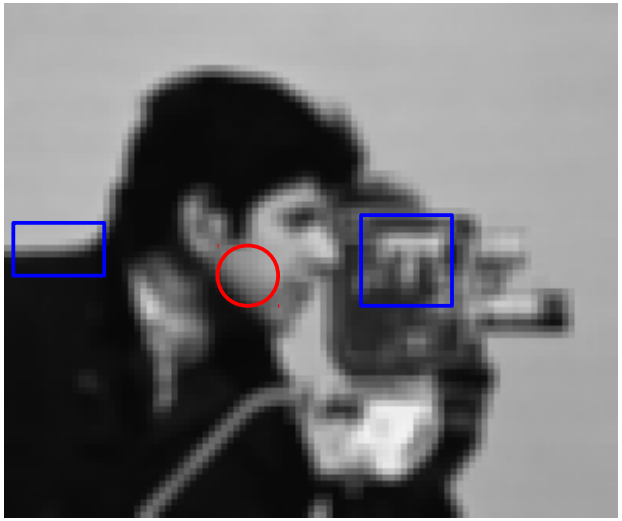
Local Operators: Low Pass Filter: Box Filter

- A **low pass filter blocks the high frequencies** in an image.
- **High frequencies** are in **fine details** and in **sharp edges**.
- **Low frequencies** are in **soft changes**.
- Low pass filter make image softer, fine details get vanished & noise is reduced.

- The **Box filter** simply averages the area under the filter mask:
- The smallest box filter is a 3x3 mask:

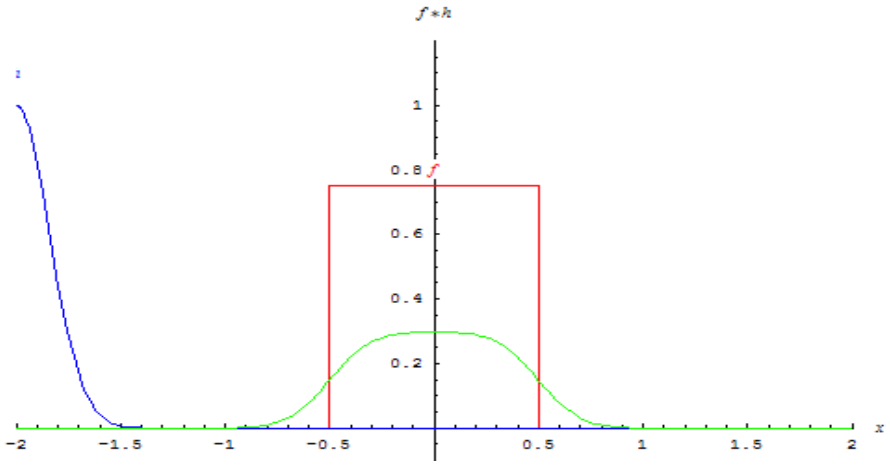
$$B = \frac{1}{9} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$





Local Operators: Low Pass Filter: Gaussian Filter

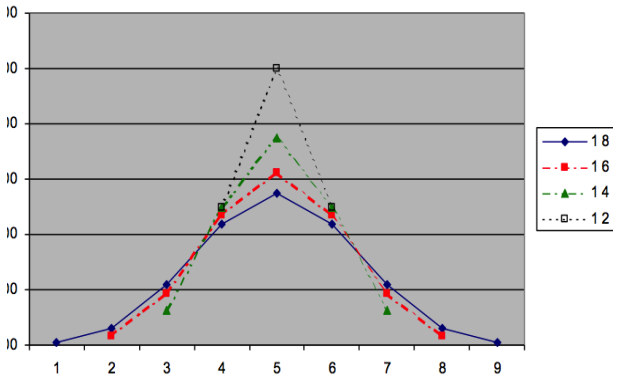
- A better low pass filter is the **gaussian filter**.
- It **approximates the gaussian curve** in 1D or Gaussian bell in 2D.
- We will see why it is better in the chapter of the Fourier transform.



Local Operators: Low Pass Filter: Gaussian Filter

- A better low pass filter is the **gaussian filter**.
- It **approximates the gaussian curve** in 1D or Gaussian bell in 2D.
- We will see why it is better in the chapter of the Fourier transform.
- A **square gaussian mask of odd size** can be built with an odd sequence of the **Pascal's triangle**.
- The Pascal's triangle approximate the gaussian curve:

```
Weight
1/  1                    1
1/  2                  1   1
1/  4                1   2   1
1/  8              1   3   3   1
1/ 16            1   4   6   4   1
1/ 32          1   5  10  10   5   1
1/ 64        1   6  15  20  15  6   1
1/128      1   7  21  35  35  21  7   1
1/256    1   8  28  56  70  56  28  8   1
```



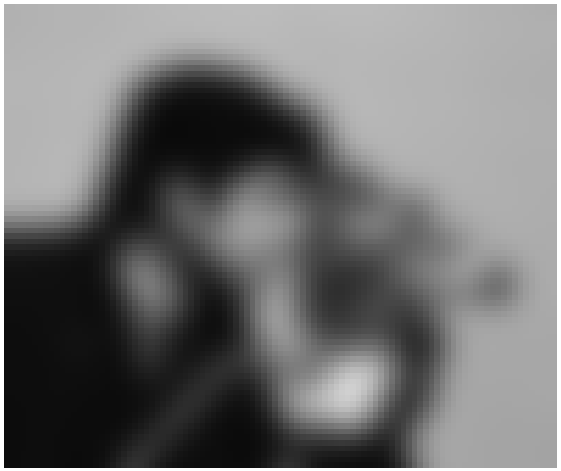
$$G = \frac{1}{16} \cdot \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} * \frac{1}{16} \cdot \begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix} = \frac{1}{256} \cdot \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Local Operators: Low Pass Filter

- Comparison of a 7x7 box filter vs. a 7x7 gaussian filter:

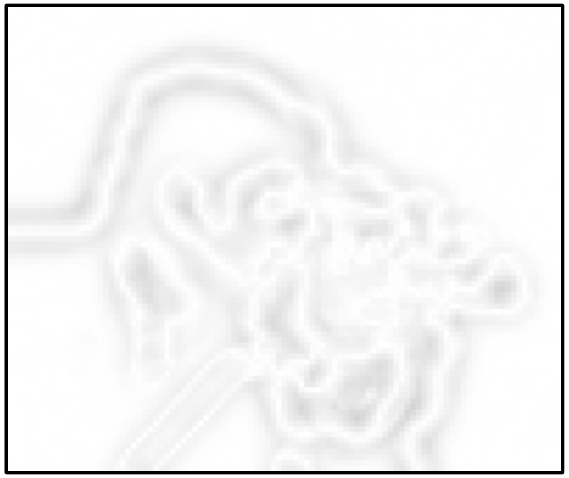
Original

box filtered



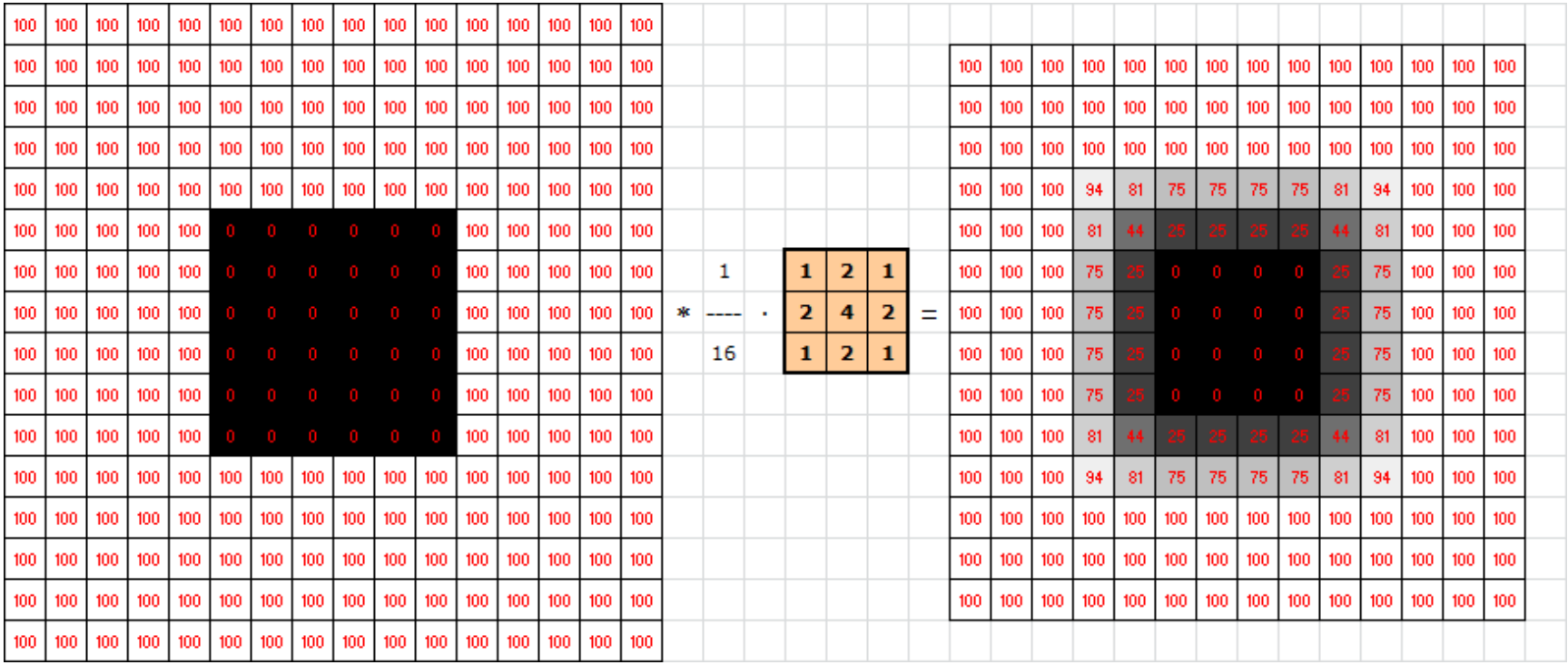
gaussian filtered

negative absolute difference



Local Operators: Exercise Low Pass Filter in Excel

- You can easily implement the convolution math in a cell of MS Excel or OO Calc.
- Make small sqaure cells
- Make separate value for the filter mask and filter weight.
- Implement the convolution formula and copy it to all destination cells.
- Visualize the cell differences with conditional formating.



Local Operators: High Pass Filter

- A **high pass filter blocks the low frequencies** in an image.
- High pass filter
  - increase the edges & noise
  - make monotonous areas to black (zero)

Local Operators: High Pass Filter: 1st Derivative

- **High pass filters** create the **1st and 2nd derivative** of the image signal.
- The are therefore also called **gradient filters**.
- The 1st derivatives of a discrete image signal in x- and y-directions are:

$$G'(x) = \frac{dG}{dx}(x) = G(x+dx) - G(x) = -1 \cdot G(x) + 1 \cdot G(x+1)$$

$$G'(y) = \frac{dG}{dy}(y) = G(y+dy) - G(y) = -1 \cdot G(y) + 1 \cdot G(y+1)$$

Local Operators: High Pass Filter: 1st Derivative

- **High pass filters** create the **1st and 2nd derivative** of the image signal.
- The are therefore also called **gradient filters**.
- The 1st derivatives of a discrete image signal in x- and y-directions are:

$$G'(x) = \frac{dG}{dx}(x) = G(x+dx) - G(x) = \boxed{-1} \cdot G(x) + \boxed{+1} \cdot G(x+1)$$

$$G'(y) = \frac{dG}{dy}(y) = G(y+dy) - G(y) = \boxed{-1} \cdot G(y) + \boxed{+1} \cdot G(y+1)$$

- With the weights we can build the filter masks for the convolution:

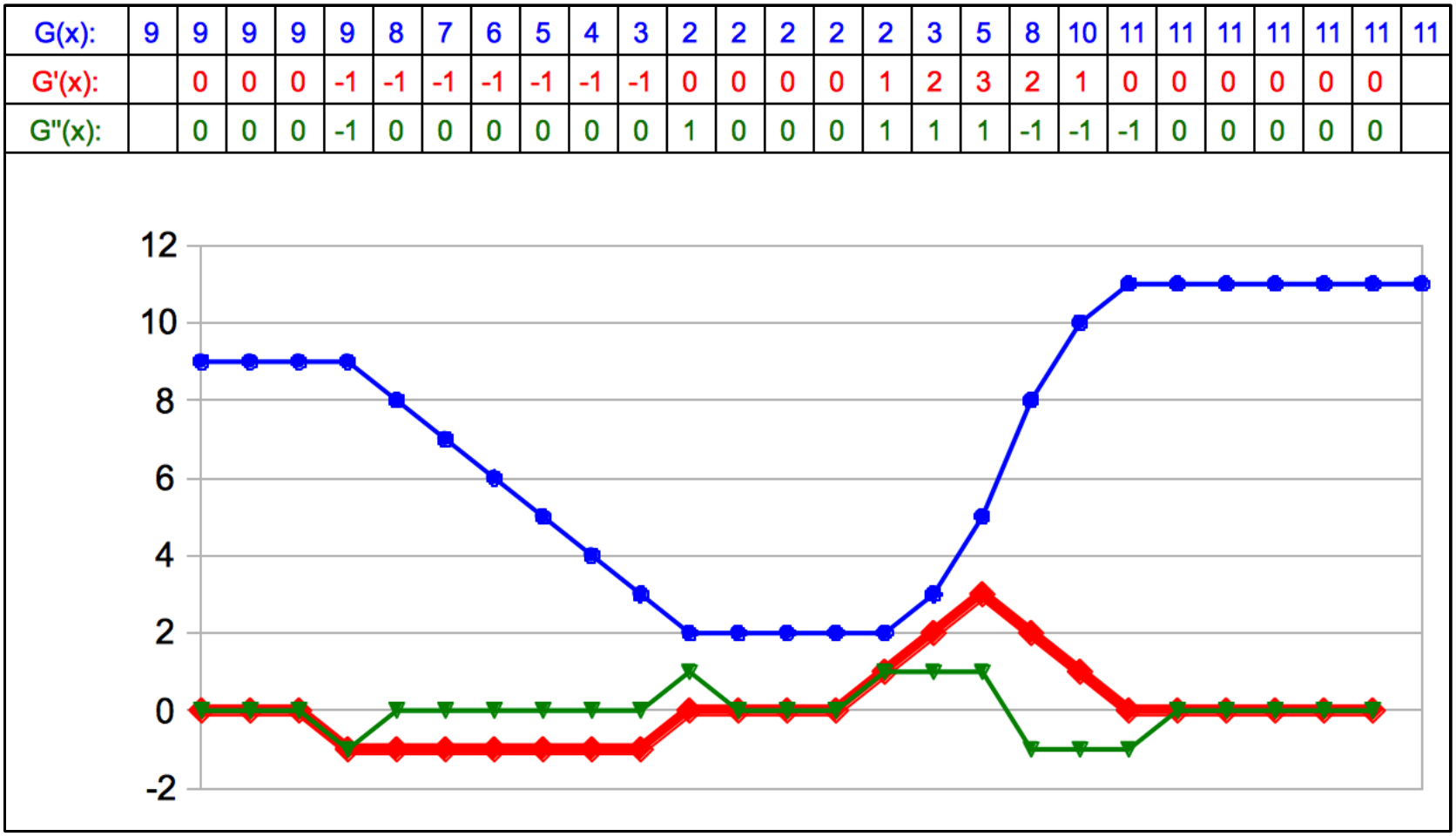
$$D_x = \begin{bmatrix} 0 & 0 & 0 \\ \boxed{-1} & \boxed{1} & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad D_y = \begin{bmatrix} 0 & \boxed{-1} & 0 \\ 0 & \boxed{1} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Local Operators: High Pass Filter: 1st Derivative

- The 1st derivatives of a discrete 1D-signal is:

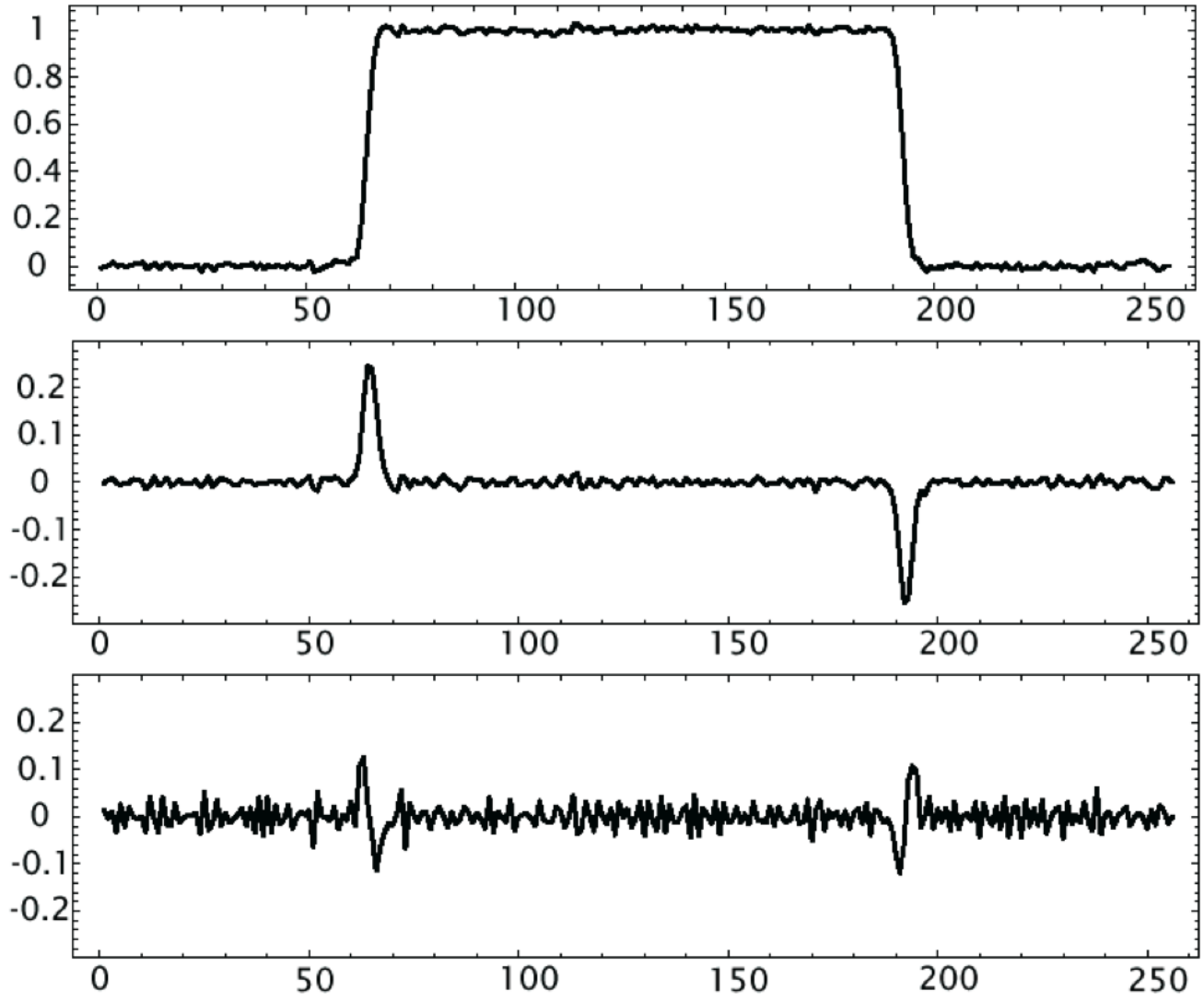
$$G'(x)=\frac{dG}{dx}(x)=G(x+dx)-G(x)=-1\cdot G(x)+1\cdot G(x+1)$$

G(x):	9	9	9	9	9	8	7	6	5	4	3	2	2	2	2	2	3	5	8	10	11	11	11	11	11	11	11
G'(x):		0	0	0	-1	-1	-1	-1	-1	-1	-1	0	0	0	0	1	2	3	2	1	0	0	0	0	0	0	
G"(x):		0	0	0	-1	0	0	0	0	0	0	1	0	0	0	1	1	1	-1	-1	-1	0	0	0	0	0	



Local Operators: High Pass Filter: 1st Derivative

- The noise is increasing with every derivative:



Local Operators: High Pass Filter: 1st Derivative

- We can get the **combined gradient** (Betrag) from G'(x) and G'(y) by:

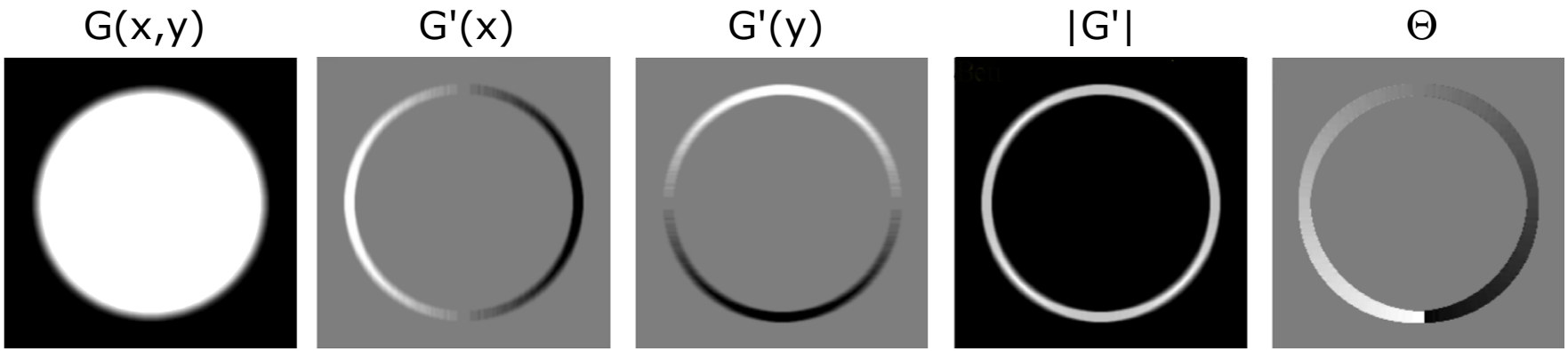
$$|G'| = \sqrt{G'_x{}^2 + G'_y{}^2}$$

or by the sum of the absolute values of the gradient components:

$$|G'| = |G'_x| + |G'_y|$$

- The **gradient direction** we get from:

$$\Theta = atan\left(\frac{G'_y}{G'_x}\right)$$



Local Operators: High Pass Filter: 1st Derivative

- Applying the filter masks D_x und D_y we get:

$$D_x = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad D_y = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

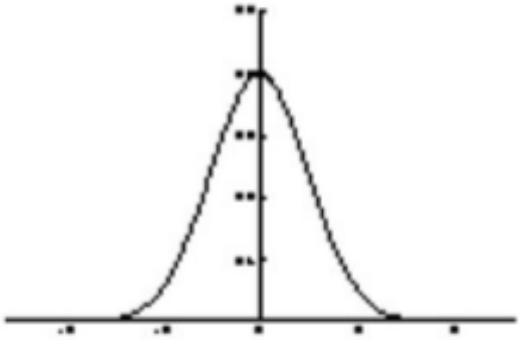


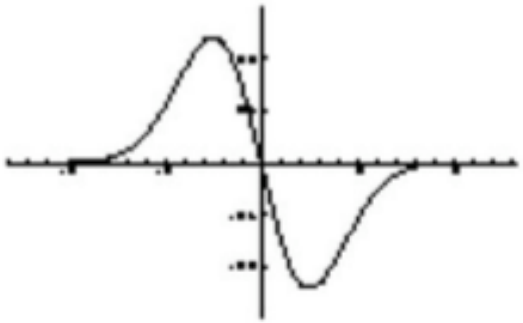
- We loose the negative values!

Local Operators: High Pass Filter: Sobel Filter

- The **1st derivative** of the **gaussian curve is**:

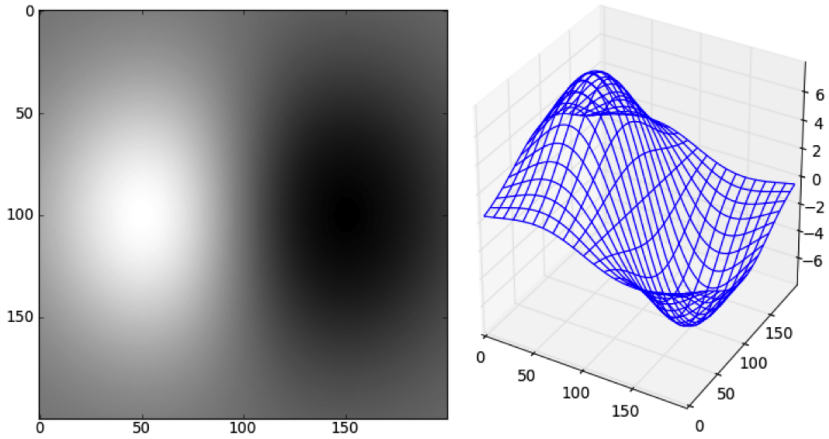
$$G(x) = e^{-\frac{x^2}{2\sigma^2}} \qquad G'(x) = \frac{-1}{\sigma^2} x \cdot e^{-\frac{x^2}{2\sigma^2}}$$

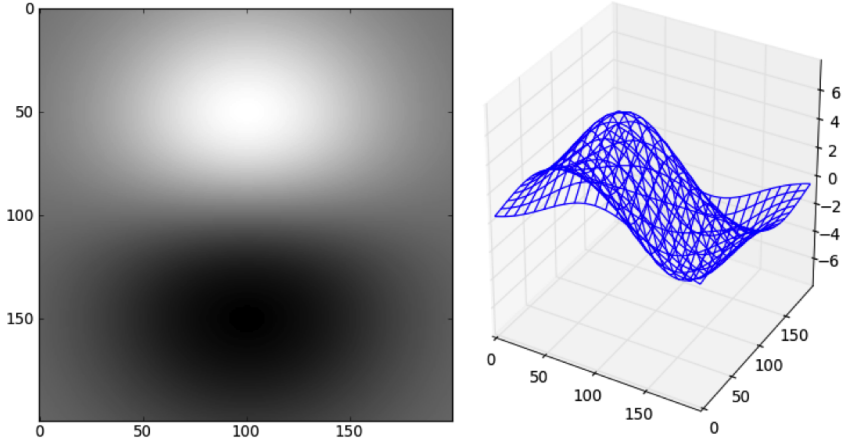




Local Operators: High Pass Filter: Sobel Filter

- The **Sobel filter** is the smallest discrete approximation to the **1st derivative** of the **gaussian bell.**
- It avoids additional noise.





$$S_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Local Operators: High Pass Filter: Sobel Filter

- The Sobel filter is the smallest discrete approximation to the 1st derivative of the gaussian bell:

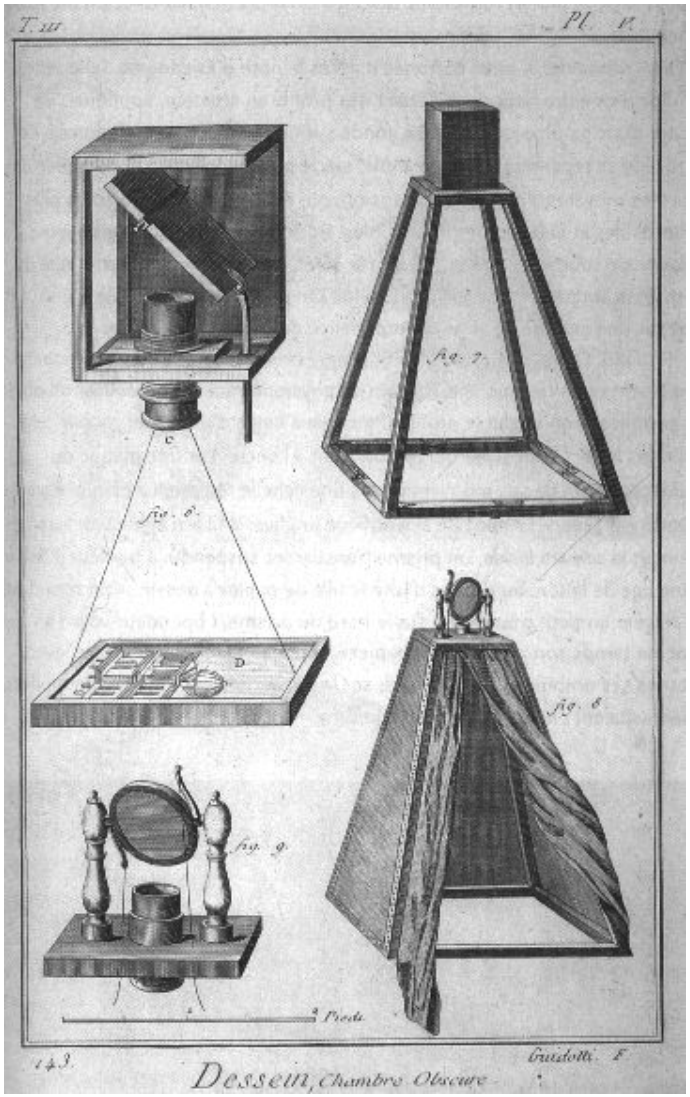
$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

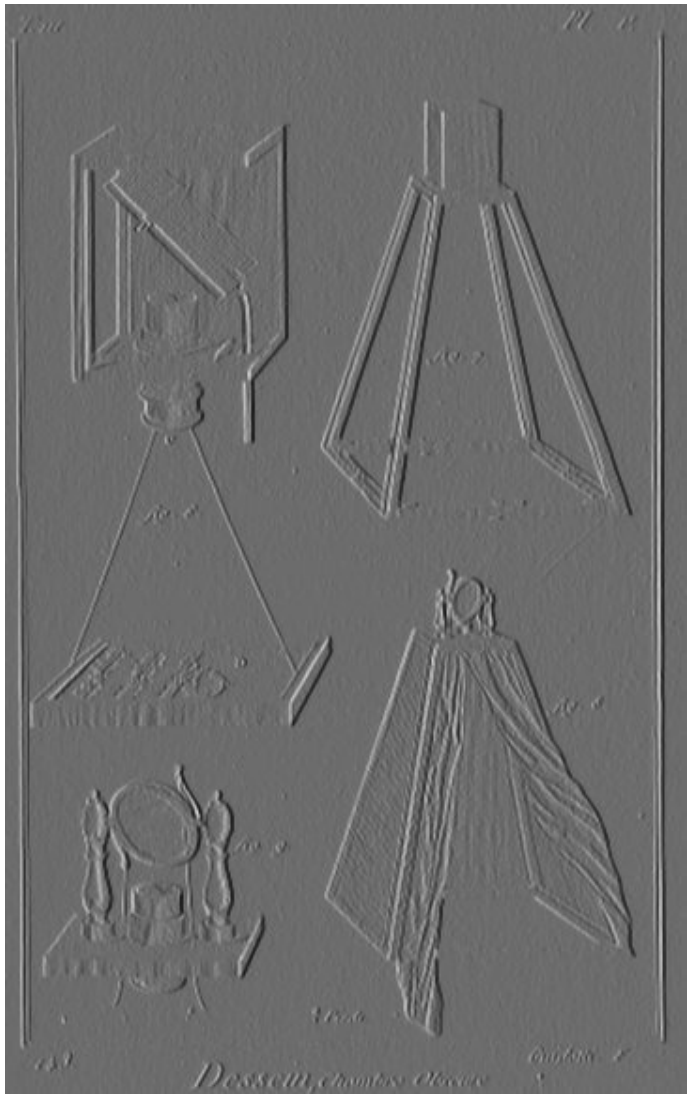


- We loose the negative values!

Local Operators: High Pass Filter: Sobel Filter

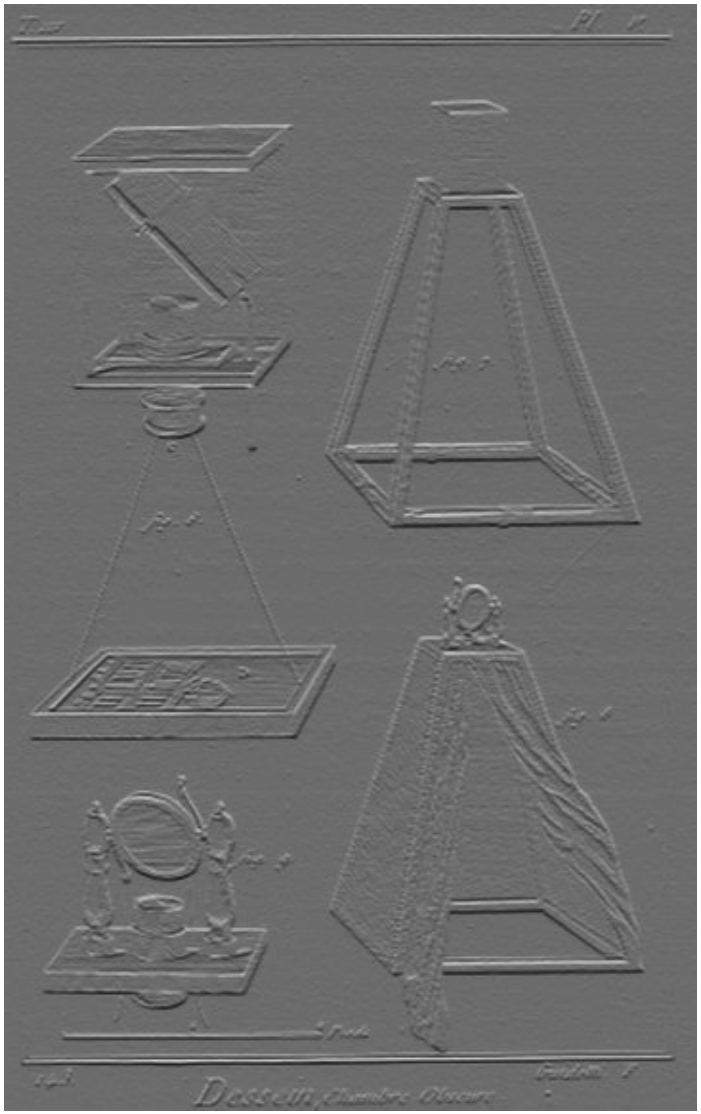
- To visualize the negative edges we can increase the luminosity by 128:

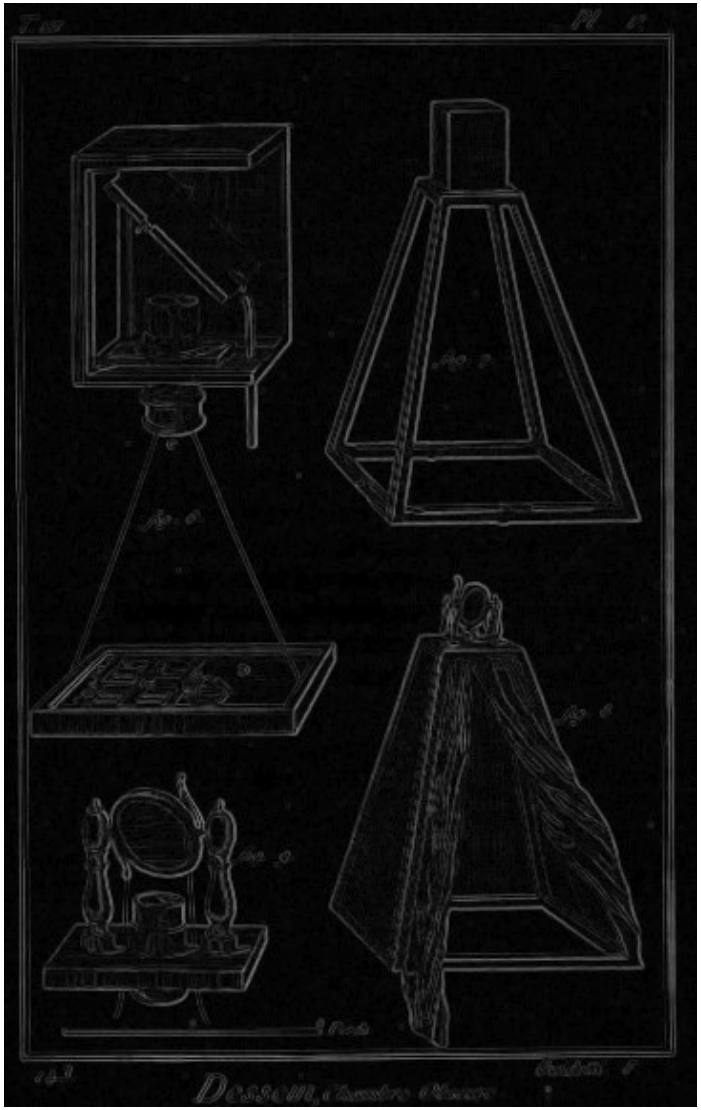




Local Operators: High Pass Filter: Sobel Filter

- We can combine all edges with: $|G'| = \sqrt{G'_x{}^2 + G'_y{}^2} = |G'_x| + |G'_y|$





Local Operators: High Pass Filter: Laplace Filter

- With the **Laplace** filter mask we can create the **2nd derivative**.
- The 2nd derivate is the derivative of the 1st derivative:

$$G'(x) = \frac{dG}{dx}(x) = G(x + dx) - G(x)$$

$$G''(x) = \frac{d^2G}{d^2x}(x) = \frac{dG}{dx}(x + dx) - \frac{dG}{dx}(x)$$

Local Operators: High Pass Filter: Laplace Filter

- With the **Laplace** filter mask we can create the **2nd derivative**.
- The 2nd derivate is the derivative of the 1st derivative:

$$G'(x) = \frac{dG}{dx}(x) = G(x+dx) - G(x)$$

$$G''(x) = \frac{d^2G}{d^2x}(x) = \frac{dG}{dx}(x+dx) - \frac{dG}{dx}(x)$$

$$G''(x) = (G(x+1) - G(x)) - (G(x) - G(x-1))$$

$$G''(x) = 1 \cdot G(x-1) - 2 \cdot G(x) + 1 \cdot G(x+1)$$

$$G''(y) = 1 \cdot G(y-1) - 2 \cdot G(y) + 1 \cdot G(y+1)$$

Local Operators: High Pass Filter: Laplace Filter

- From these weights we can build the 2nd derivative filter mask.
- It's called **Laplace Filter**:

$$G''(x) = 1 \cdot G(x-1) - 2 \cdot G(x) + 1 \cdot G(x+1)$$

$$G''(y) = 1 \cdot G(y-1) - 2 \cdot G(y) + 1 \cdot G(y+1)$$

$$L = L_x + L_y = \begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Local Operators: High Pass Filter: Laplace Filter

- From these weights we can build the 2nd derivative filter mask.
- It's called **Laplace Filter**:

$$G''(x) = 1 \cdot G(x-1) - 2 \cdot G(x) + 1 \cdot G(x+1)$$

$$G''(y) = 1 \cdot G(y-1) - 2 \cdot G(y) + 1 \cdot G(y+1)$$

$$L = L_x + L_y = \begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

- A more rotation invariant version sums up also the diagonal derivatives:

$$L = L_x + L_y + L_{d1} + L_{d2} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 1 \\ 0 & -2 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Local Operators: High Pass Filter: Laplace Filter

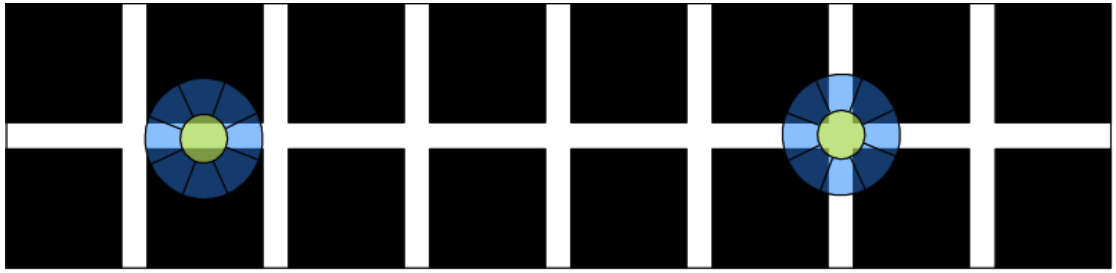
- With the **2nd derivative** we get on all edges **lower but positive values**.
- The **noise is increased**.

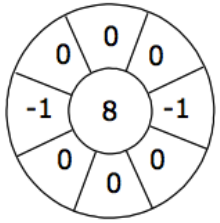
$$L=\begin{bmatrix}0 & 1 & 0\\ 1 & -4 & 1\\ 0 & 1 & 0\end{bmatrix}$$

$$L=\begin{bmatrix}1 & 1 & 1\\ 1 & -8 & 1\\ 1 & 1 & 1\end{bmatrix}$$

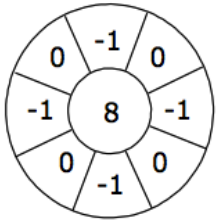
High Pass Filter: Laplace Filter

- You may remember what happens in the receptive fields (ganglion cells) of the retina.
- The weights in the receptive fields correspond to a Laplace filter:





Summe 6



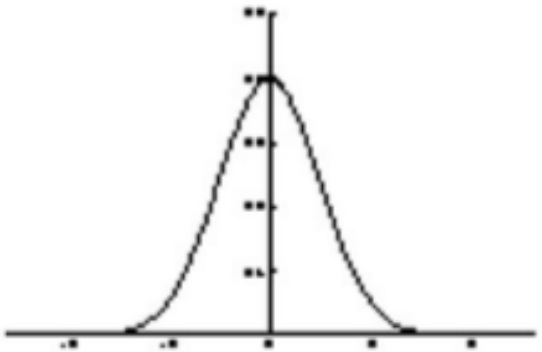
Summe 4

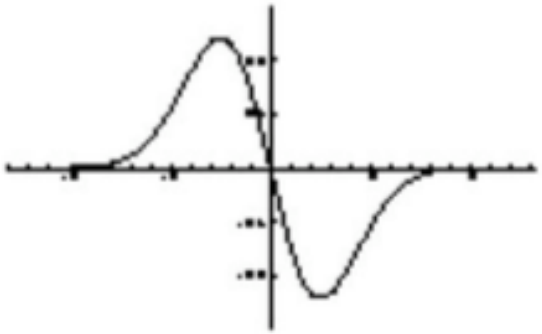
$$L = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

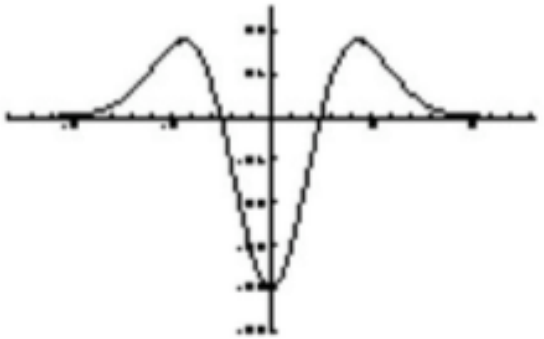
High Pass Filter: Laplace of Gaussian

- The best **2nd derivative** is again derived from the gaussian curve.
- The filter is called **Laplace of a Gaussian:**

$$G(x) = e^{-\frac{x^2}{2\sigma^2}} \qquad G'(x) = \frac{-1}{\sigma^2}\, x \cdot e^{-\frac{x^2}{2\sigma^2}} \qquad G''(x) = \frac{1}{\sigma^2}\left(\frac{x^2}{\sigma^2} - 1\right) \cdot e^{-\frac{x^2}{2\sigma^2}}$$





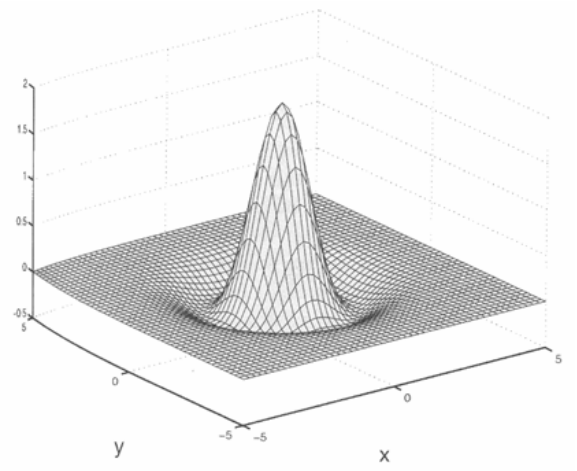


$$\log_{3x3} = \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix} \qquad \log_{5x5} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 1 & 2 & -16 & 2 & 1 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

High Pass Filter: Laplace of Gaussian

- For thicker edges Laplace filters can habe even bigger size.
- Because of its shape the Laplace filter has also the name **Mexican Hat**.

$$\log_{13x13} = -\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & -1 & -2 & -2 & -2 & -1 & -1 & 0 & 0 & 0 \\
0 & 0 & -2 & -2 & -3 & -3 & -4 & -3 & -3 & -2 & -2 & 0 & 0 \\
0 & -1 & -2 & -3 & -3 & -3 & -2 & -3 & -3 & -3 & -2 & -1 & 0 \\
0 & -1 & -3 & -3 & -1 & 4 & 6 & 4 & -1 & -3 & -3 & -1 & 0 \\
-1 & -2 & -3 & -3 & 4 & 14 & 19 & 14 & 4 & -3 & -3 & -2 & -1 \\
-1 & -2 & -4 & -2 & 6 & 19 & 24 & 19 & 6 & -2 & -4 & -2 & -1 \\
-1 & -2 & -3 & -3 & 4 & 14 & 19 & 14 & 4 & -3 & -3 & -2 & -1 \\
0 & -1 & -3 & -3 & -1 & 4 & 6 & 4 & -1 & -3 & -3 & -1 & 0 \\
0 & -1 & -2 & -3 & -3 & -3 & -2 & -3 & -3 & -3 & -2 & -1 & 0 \\
0 & 0 & -2 & -2 & -3 & -3 & -4 & -3 & -3 & -2 & -2 & 0 & 0 \\
0 & 0 & 0 & -1 & -1 & -2 & -2 & -2 & -1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$



Local Operators: Point Spread Function (PSF)

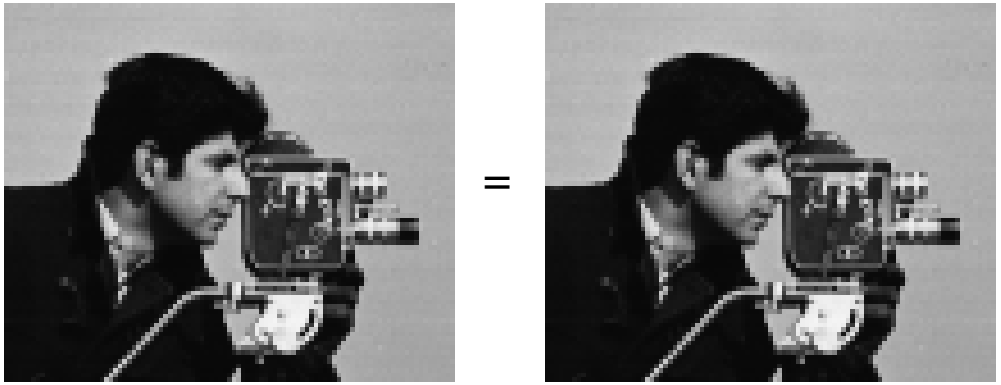
- A **PSF** is a special filter mask with a 1 in the center. All the rest is zero.
- Such a function is also called **Dirac** function:

$$PSF = \delta(i, j) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

- A convolution with a PSF results in a identical image:

$$g * \delta = \delta * g = g$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \quad = \quad$$

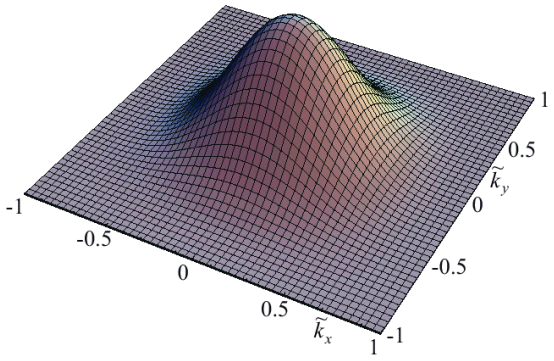


Local Operators: Point Spread Function (PSF)

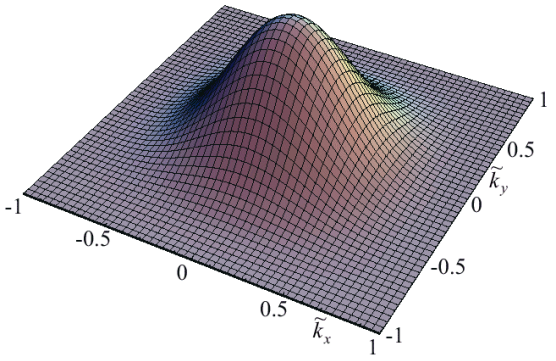
- A convolution with a PSF with a convolution mask return the convolution mask:

$$h * \delta = \delta * h = h$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} *$$



=

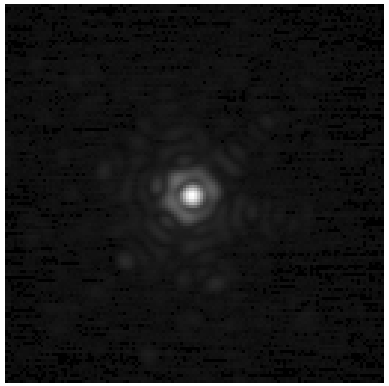


Local Operators: Point Spread Function (PSF)

- This can be interesting when the convolution mask is unknown.
- An optic works like a PSF: It spreads an infinitesimal light impulse on the image sensor to its PSF:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} *$$

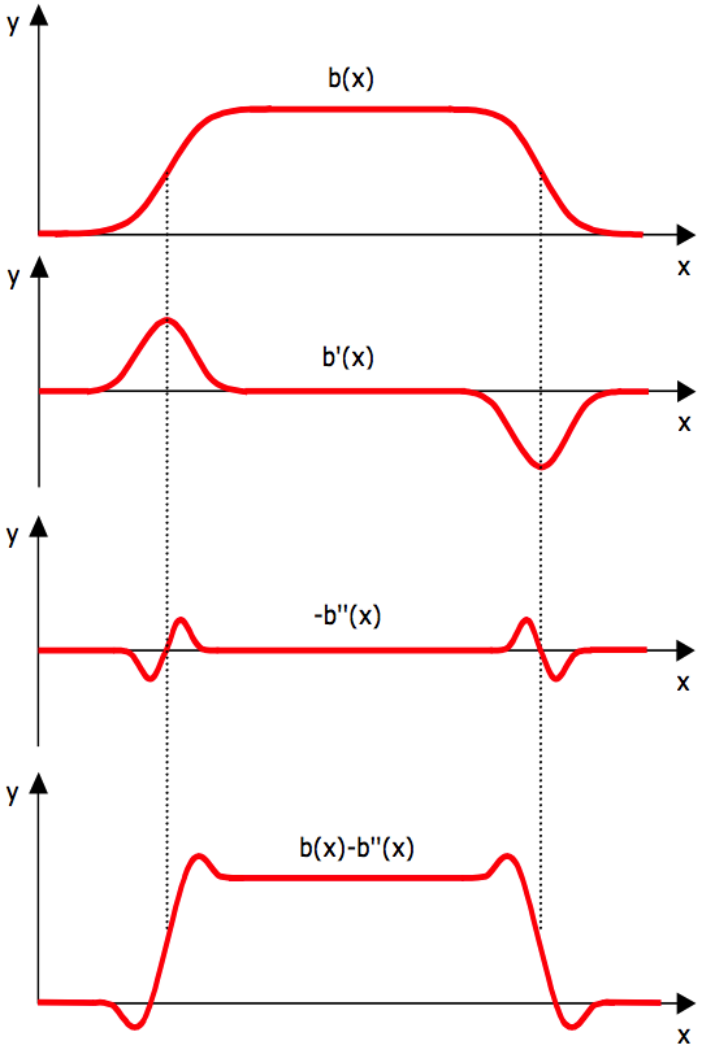
$=$



- If we can determine the PSF of a lens, we can sharpen an image with **inverse filtering** far more.

Applications: Sharpening with Laplace

- We can sharp an image by subtracting the laplace filtered image:



$$G_{SharpLoG} = b(x, y) - b(x, y) * L_{\log}$$

Applications: Sharpening with Laplace

- We can build a sharping filter by subtracting the laplace filter from an impulse filter:

$$G_{Sharp} = g * i - g * h = g * (i - h)$$

- Slight sharpening

$$G_{Sharp1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix} = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

- Medium sharpening

$$G_{Sharp2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

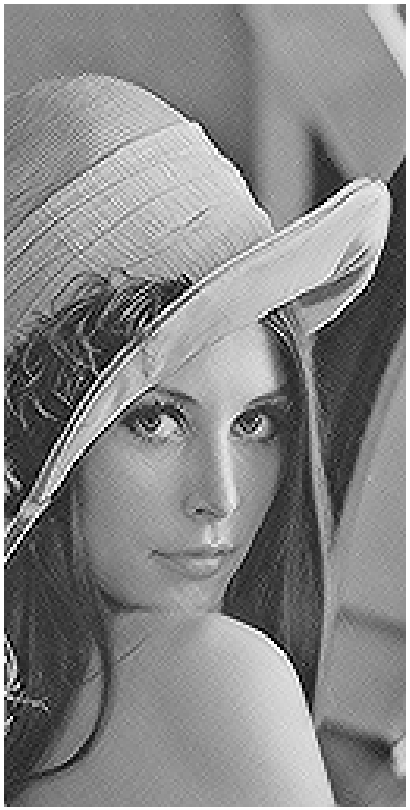
- Strong sharpening

$$G_{Sharp3} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Applications: Sharpening with Laplace

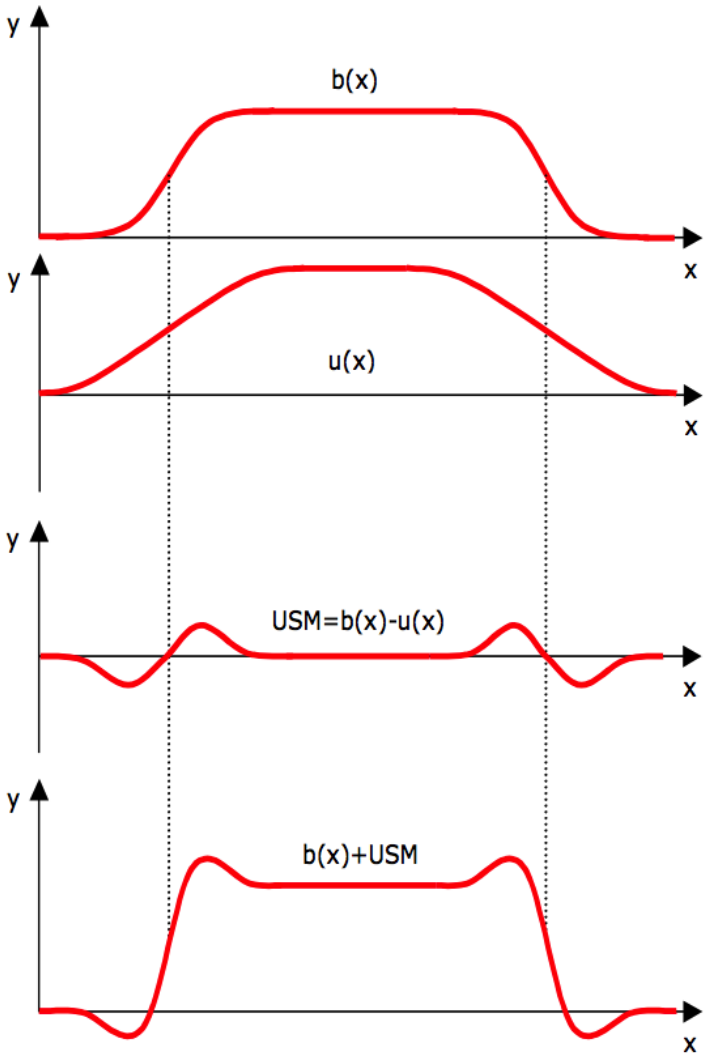
- The noise is increasing of course too!

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Applicatons: Sharpening with USM

- An alternative was is to use **Unsharp Masking**:



$$G_{CrispUSM} = b(x,y) + (b(x,y) - u(x,y))$$

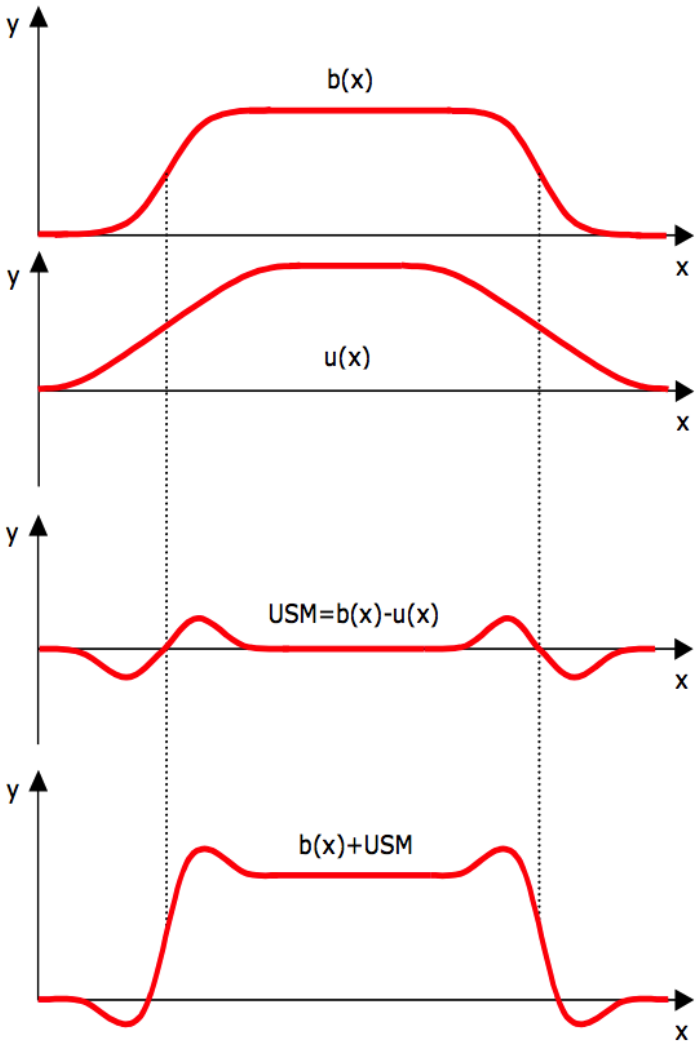
Applicatons: Sharpening with USM

Example of **Unsharp Masking**: Original image:

Image Source: luminous-landscape.com

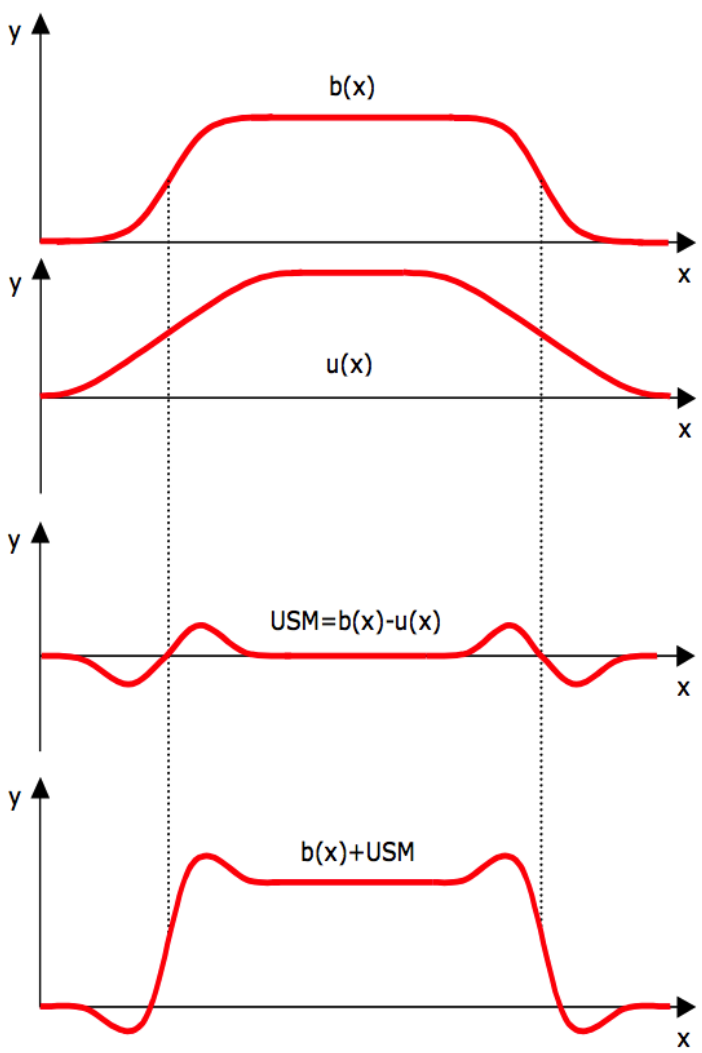
Applicatons: Sharpening with USM

Example of **Unsharp Masking**: Original image:



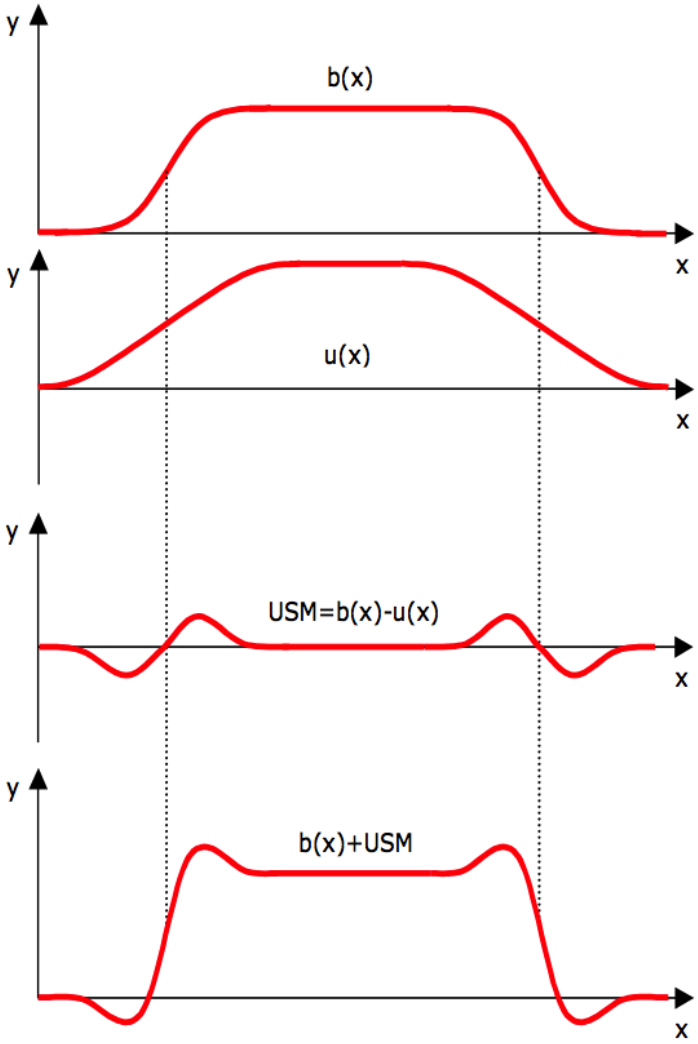
Applicatons: Sharpening with USM

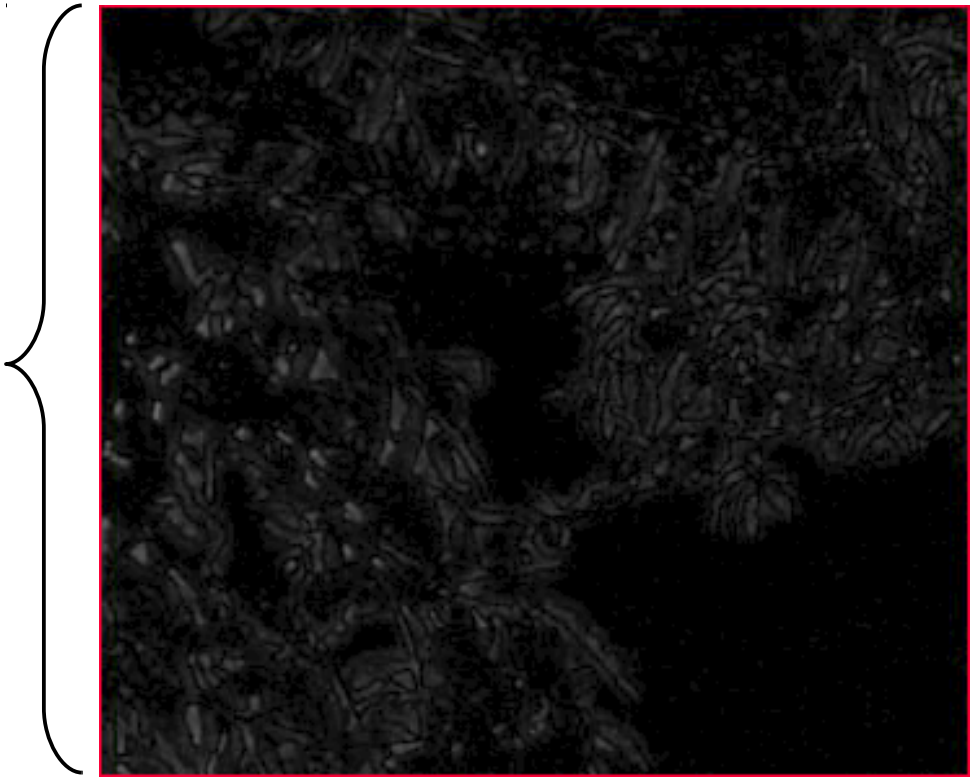
Example of **Unsharp Masking**: Create blured image:



Applicatons: Sharpening with USM

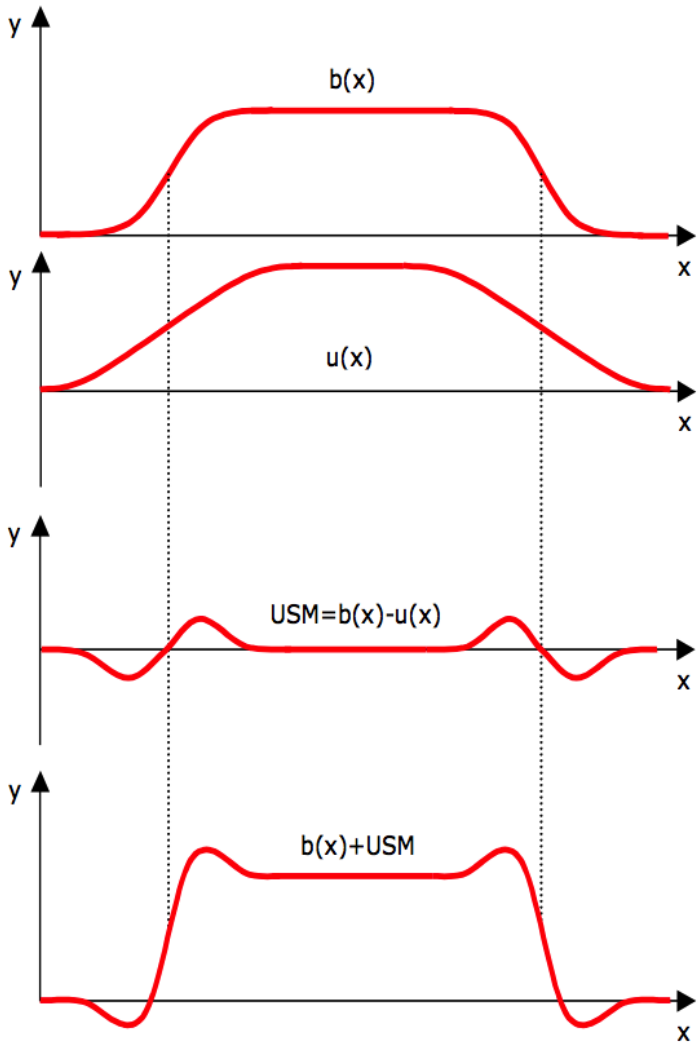
Example of **Unsharp Masking**: Create absolute difference image (USM):





Applicatons: Sharpening with USM

Example of **Unsharp Masking**: Add USM to Original image:



Applicatons: Sharpening with USM

Example of **Unsharp Masking**: Sharpened image:

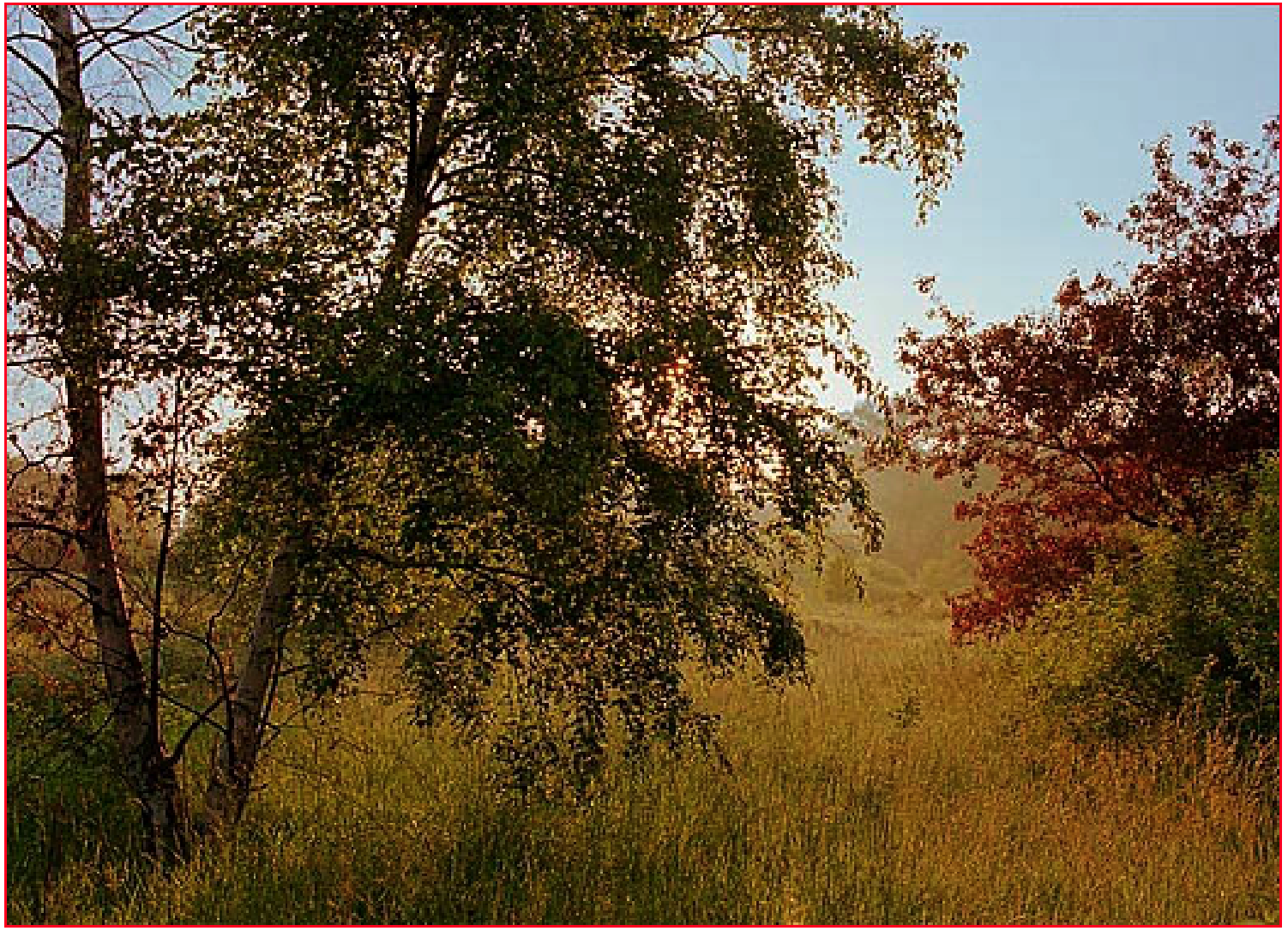


Image Source: luminous-landscape.com