AI70601/CSE61101

3D Vision and Machine Perception

Prof. Kyungdon Joo

3D Vision & Robotics Lab.

AI Graduate School (AIGS) & Computer Science and Engineering (CSE)

Contents

- Other types of cameras

- Image Processing Basic

- Image Gradients

Other types of cameras

Imaging sensors

360 cameras

- capture all of the surrounding area around the camera without blind spots

360° video example

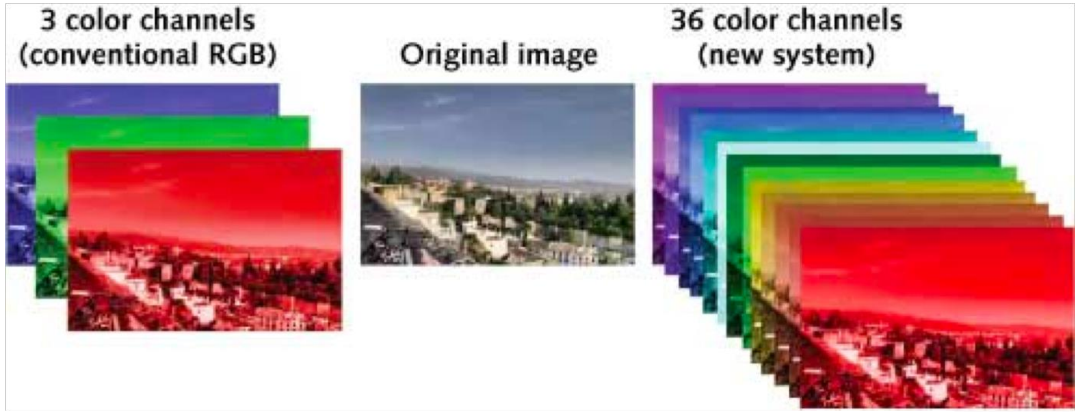
Samsung Gear 360 LG 360 Cam

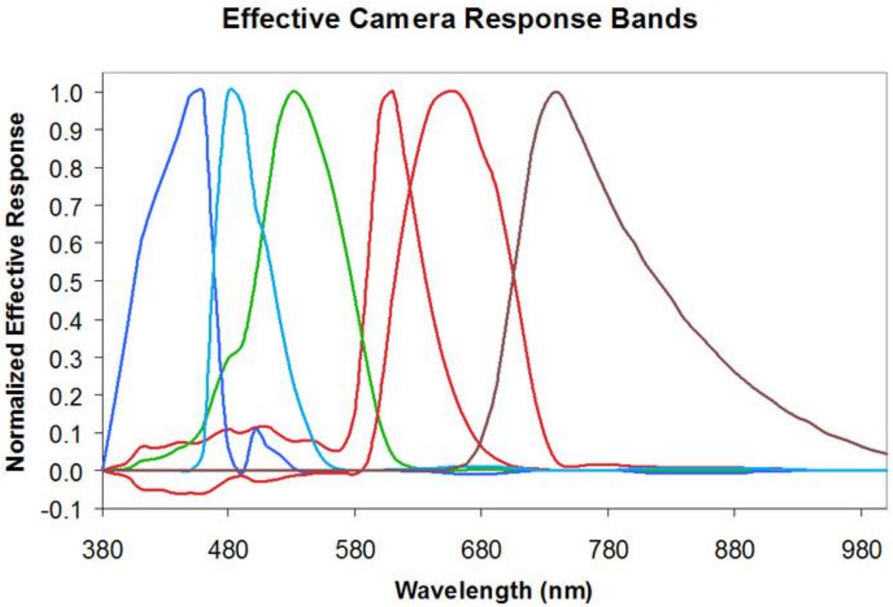
Portable 360° cameras

The left video from https://www.youtube.com/watch?v=mj2OQvDuy_g
Slide by K. J. Yoon, ME491: Visual Intelligence

Multi-spectral cameras

- captures image data within specific wavelength ranges across the electromagnetic spectrum.

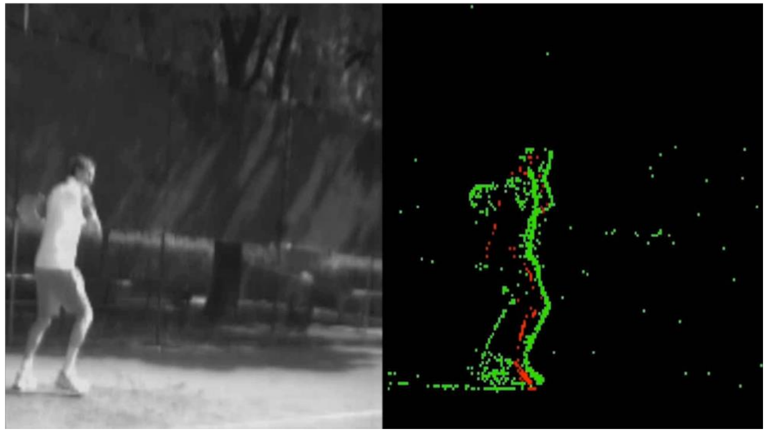


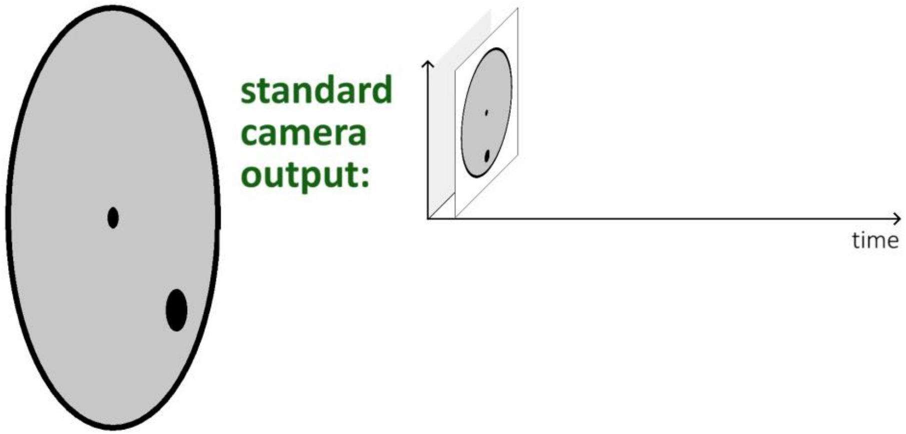


Event camera

- Asynchronous events (pixel-level relative brightness changes caused by movement)
  - Low latency
  - Micro-second temporal resolution
  - High dynamic range (low-light to very bright)

Limits of standard camera

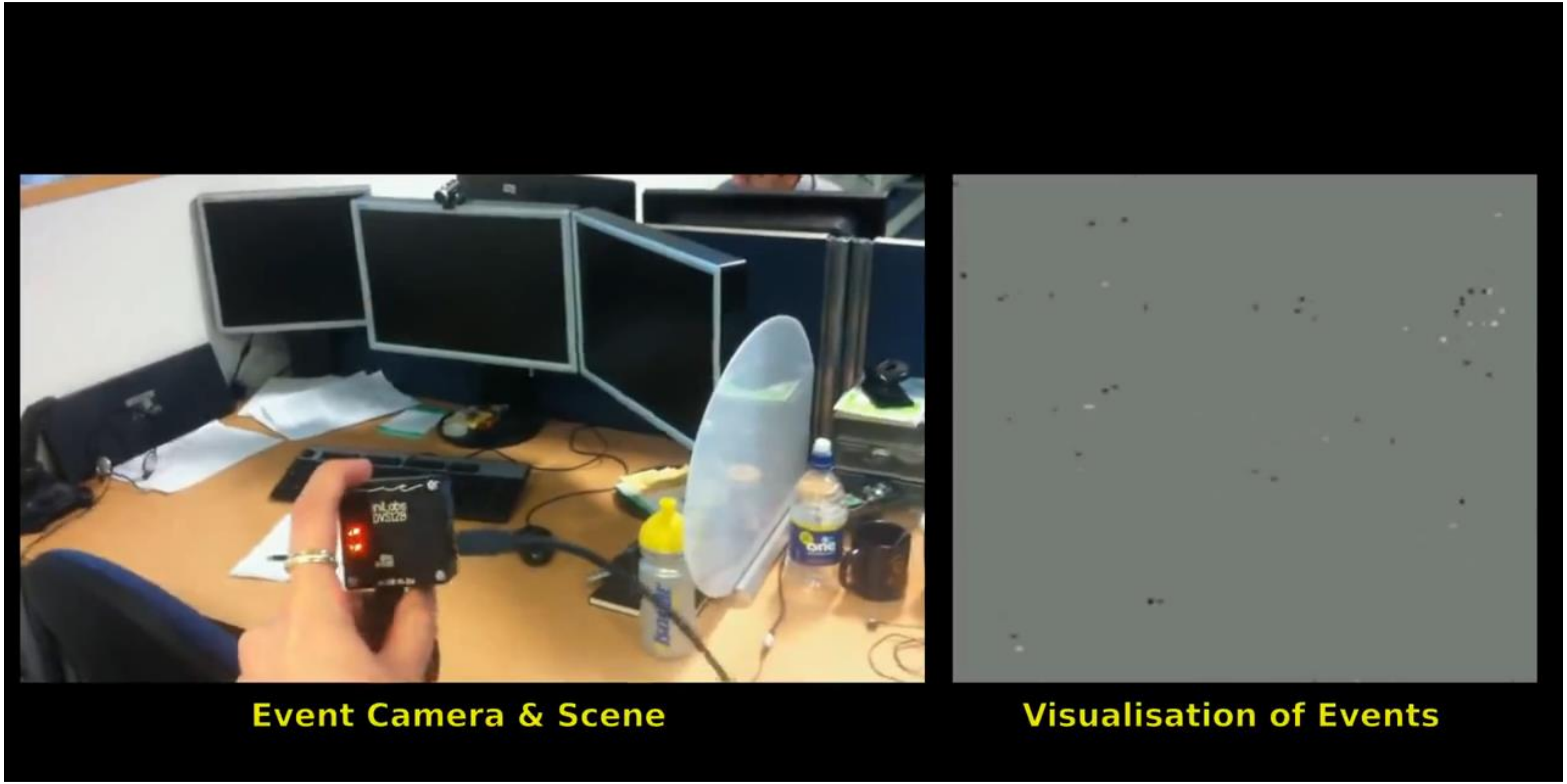




http://rpg.ifi.uzh.ch/research_dvs.html

Event camera-based vision applications

- Image Reconstruction from events

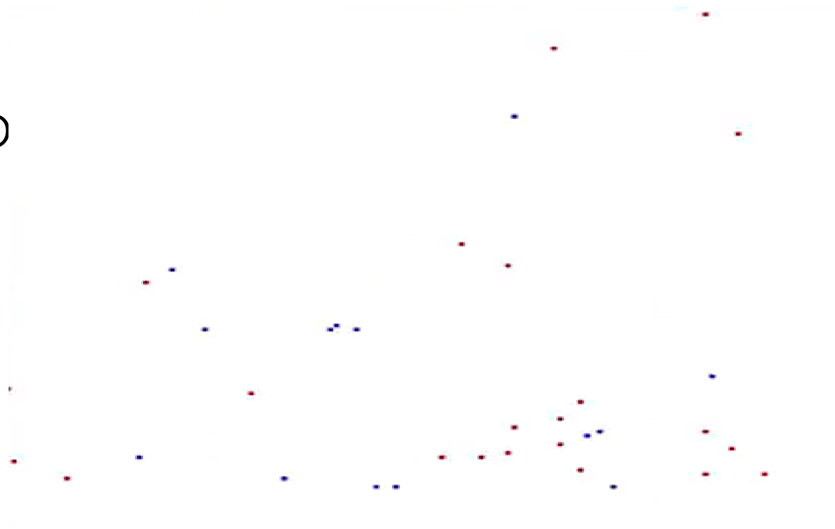


Kim et al., Simultaneous Mosaicing and Tracking with an Event Camera, BMVC'14

Event camera-based vision applications

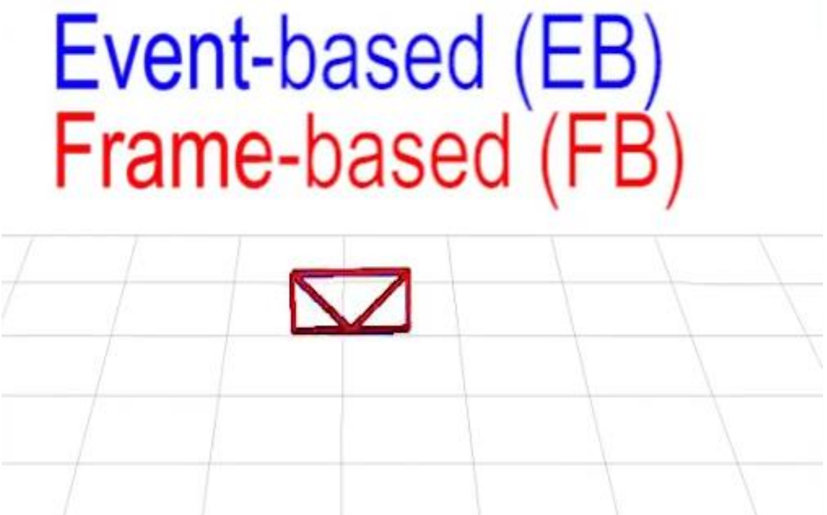
- 6DoF Tracking
  from Photometric Map

Event camera



Standard camera

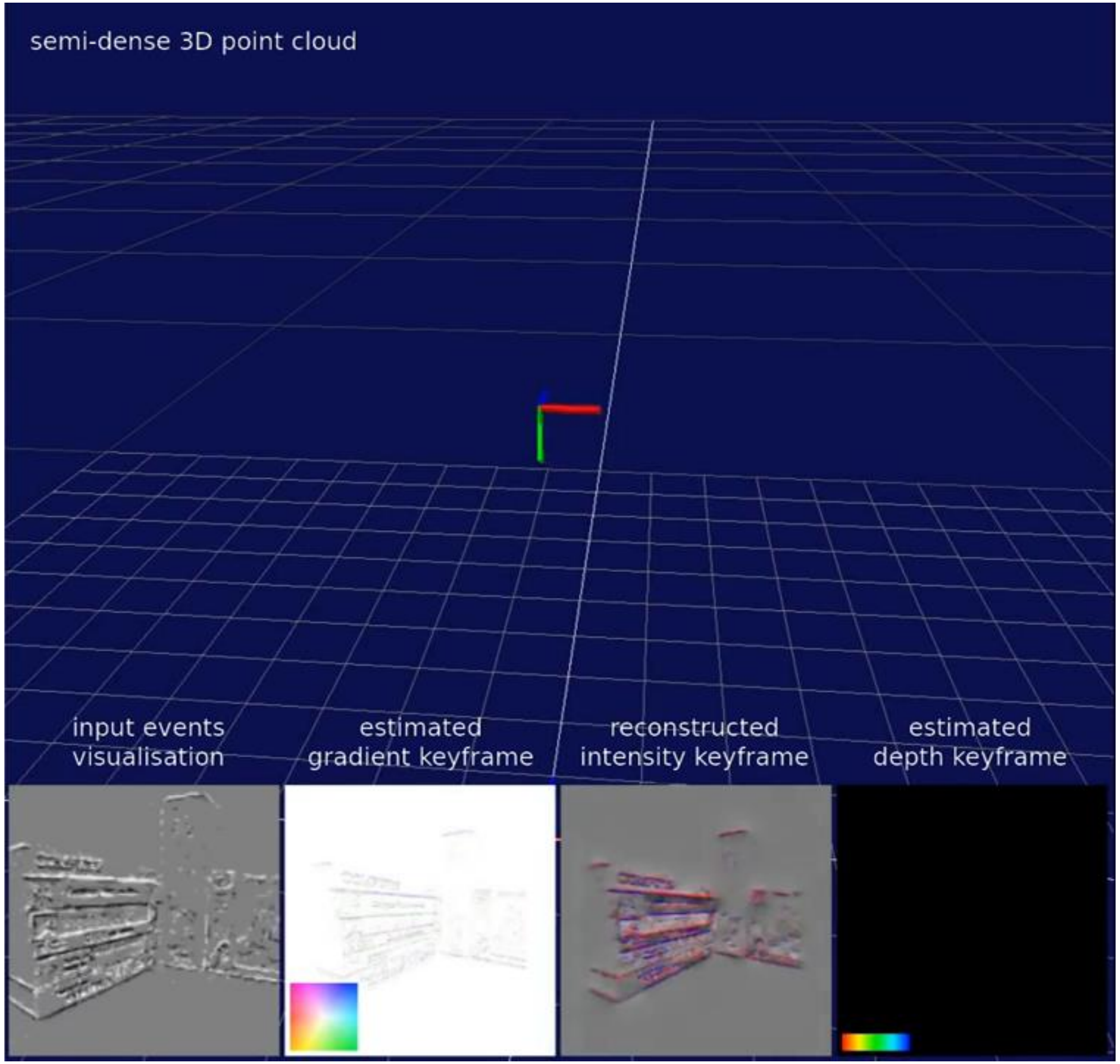
Motion estimation



Gallego et al., Event-based 6-DOF Camera Tracking from Photometric Depth Maps, T-PAMI'18. [PDF](PDF) Video

Event camera-based vision applications

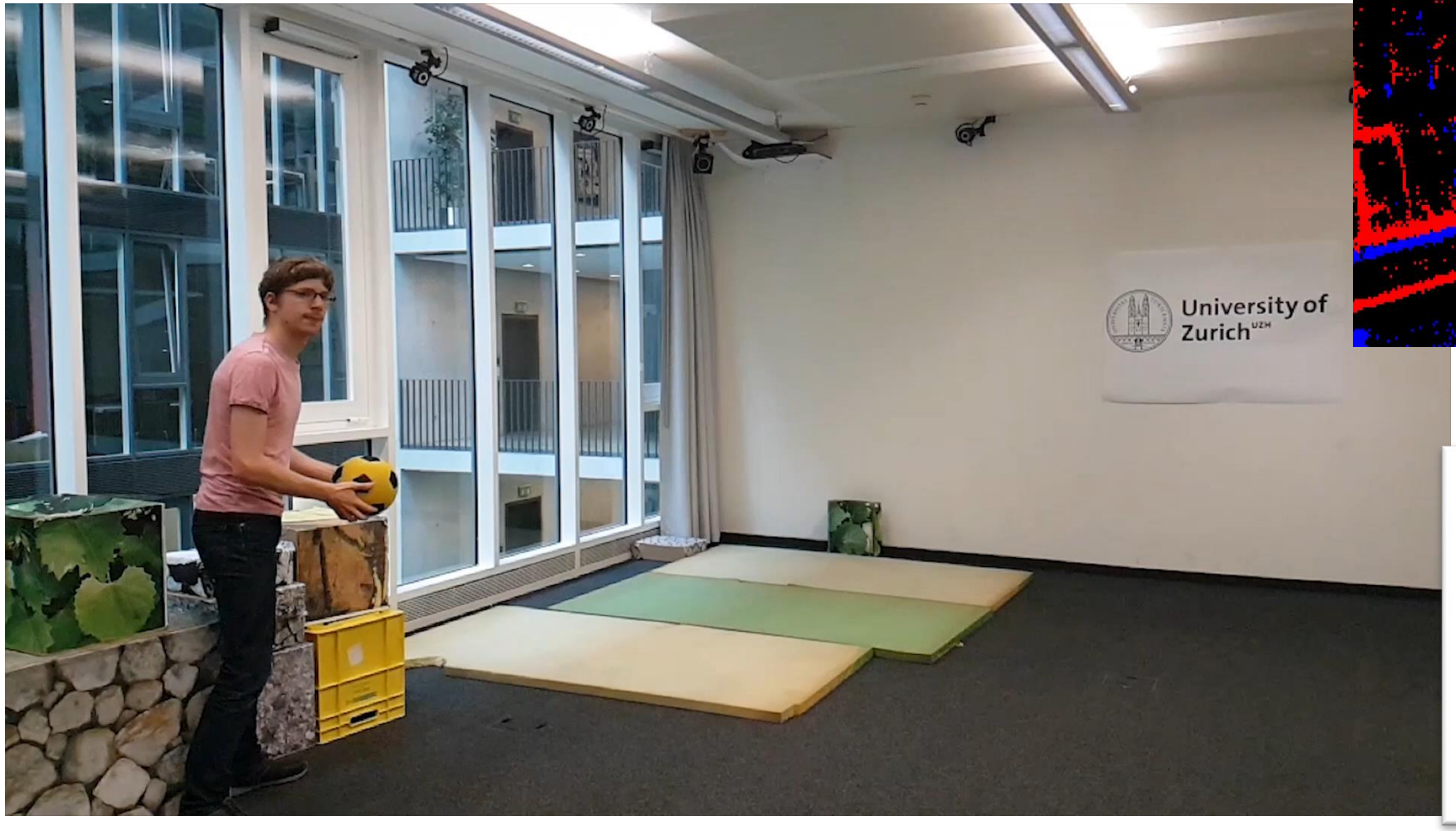
- SLAM

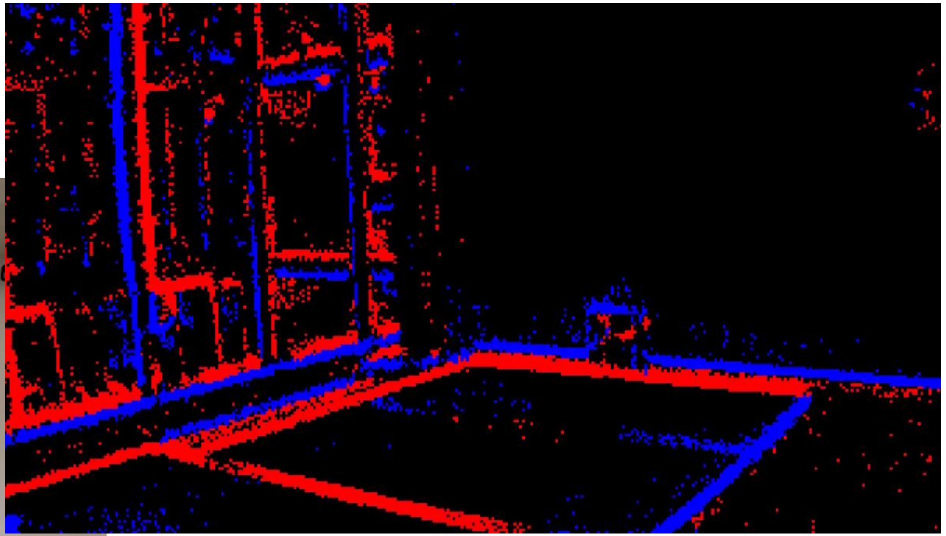


Kim et al., Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera, ECCV'16

Event camera-based vision applications

- Dynamic Obstacle Detection & Avoidance
  - Works with relative speeds of up to 10 m/s
  - Perception latency: 3.5 ms





Falanga et al., Dynamic Obstacle Avoidance for Quadrotors with Event Cameras, Science Robotics, 2020.
Falanga et al. How Fast is too fast? The role of perception latency in high speed sense and avoid, RAL'19.

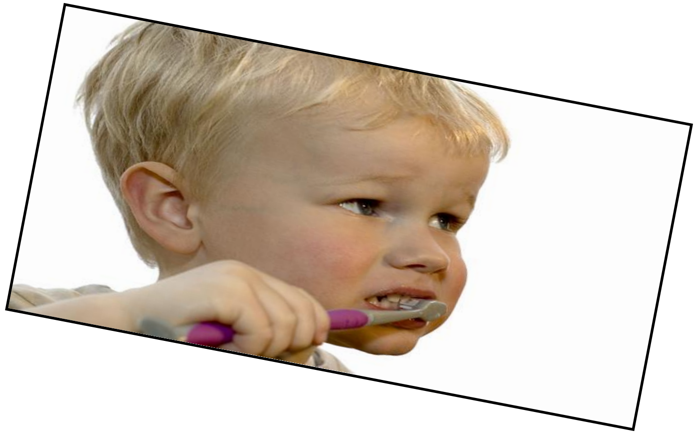
Image Processing Basic

What types of image transformations can we do?

Filtering

changes pixel *values*

Warping



changes pixel *locations*

What types of image transformations can we do?

F

Filtering

$$G(\boldsymbol{x}) = h\{F(\boldsymbol{x})\}$$

G

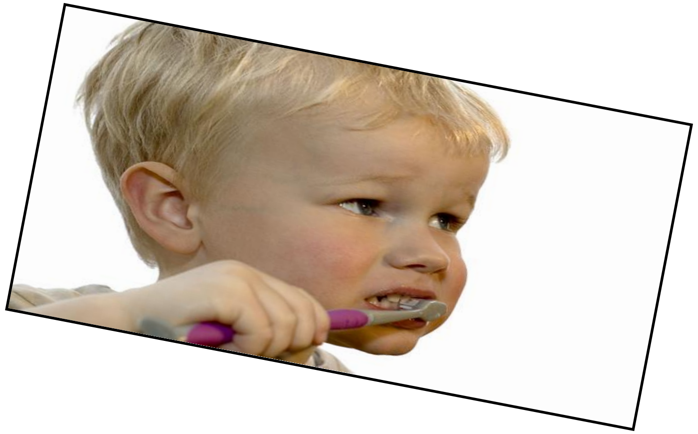
changes *range* of image function

F

Warping

$$G(\boldsymbol{x}) = F(h\{\boldsymbol{x}\})$$

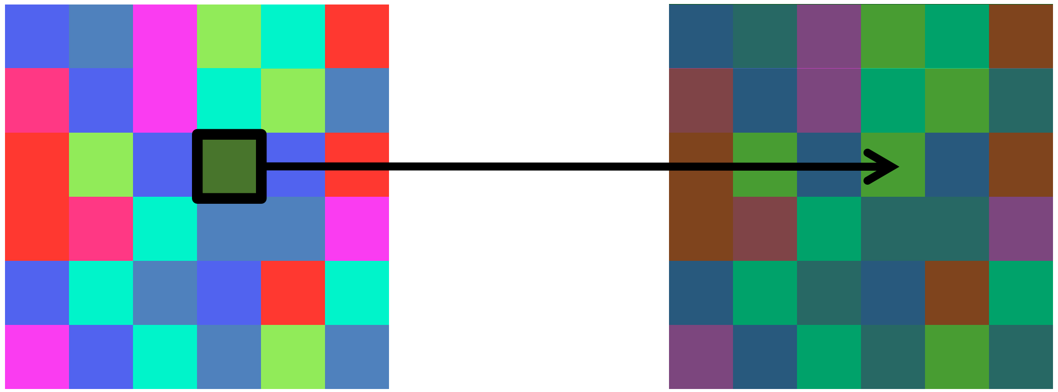
G



changes *domain* of image function

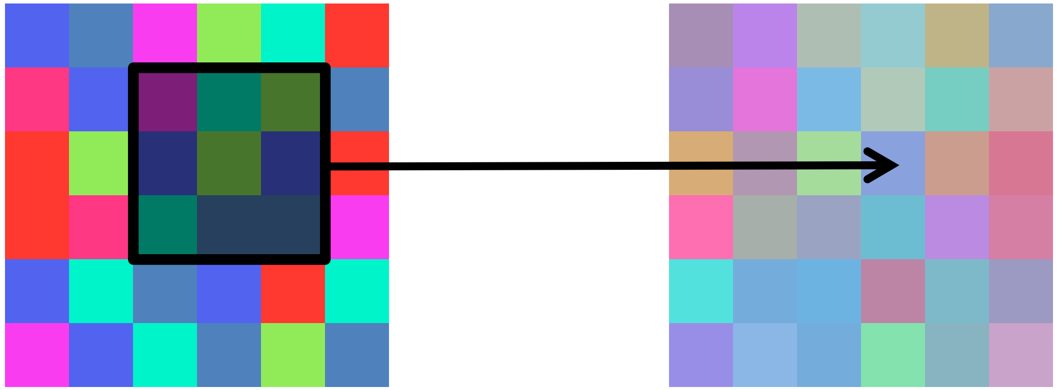
What types of image filtering can we do?

Point operation



point processing

Neighborhood operation



“filtering”

Point processing

Examples of point processing

original

darken

lower contrast

non-linear lower contrast

invert

lighten

raise contrast

non-linear raise contrast

Examples of point processing

- How would you implement these?

original

darken

lower contrast

non-linear lower contrast

invert

lighten

raise contrast

non-linear raise contrast

Examples of point processing

- How would you implement these?

original

$$x$$

darken

$$x - 128$$

lower contrast

$$\frac{x}{2}$$

non-linear lower contrast

$$\left(\frac{x}{255}\right)^{1/3} \times 255$$

invert

$$255 - x$$

lighten

$$x + 128$$

raise contrast

$$x \times 2$$

non-linear raise contrast

$$\left(\frac{x}{255}\right)^{2} \times 255$$

Linear shift-invariant image filtering

Linear shift-invariant image filtering

- Replace each pixel by a *linear* combination of its neighbors (and possibly itself).

- The combination is determined by the filter's *kernel*.

- The same kernel is *shifted* to all pixel locations so that all pixels use the same linear combination of their neighbors => *shift-invariant.*

Convolution for 2D discrete signals

- Definition of filtering as convolution:

notice the flip

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j) I(x - i, y - j)$$

filtered image

filter

input image

Convolution vs correlation

- Definition of discrete 2D convolution:

notice the flip

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j) I(x - i, y - j)$$

- Definition of discrete 2D correlation:

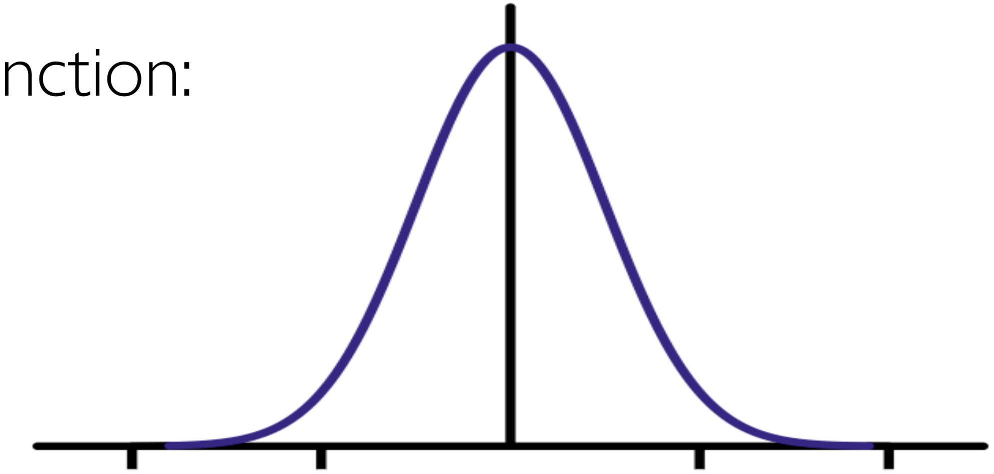
notice the lack of a flip

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j) I(x + i, y + j)$$

- Most of the time won't matter, because our kernels will be symmetric.

The Gaussian filter

- named (like many other things) after Carl Friedrich Gauss
- kernel values sampled from the 2D Gaussian function:



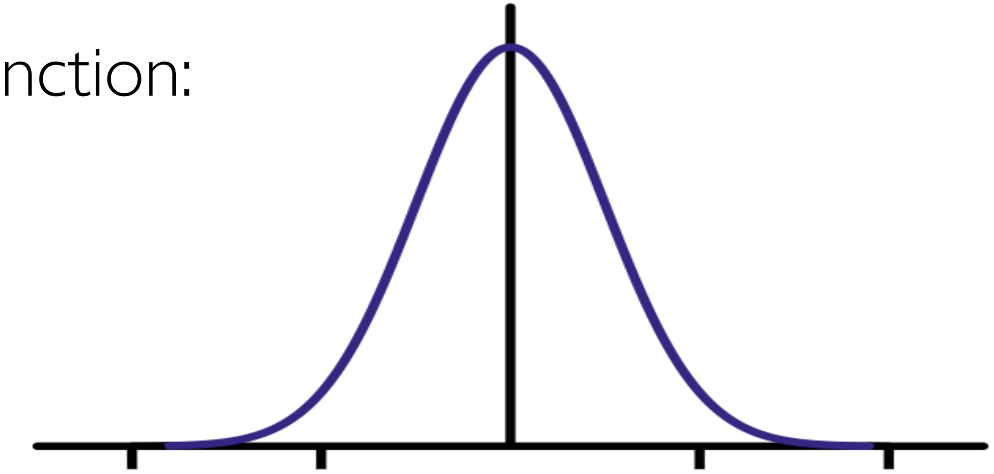
$$f(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}}$$

- weight falls off with distance from center pixel
- theoretically infinite, in practice truncated to some maximum distance
- Any heuristics for selecting where to truncate?

The Gaussian filter

- named (like many other things) after Carl Friedrich Gauss
- kernel values sampled from the 2D Gaussian function:

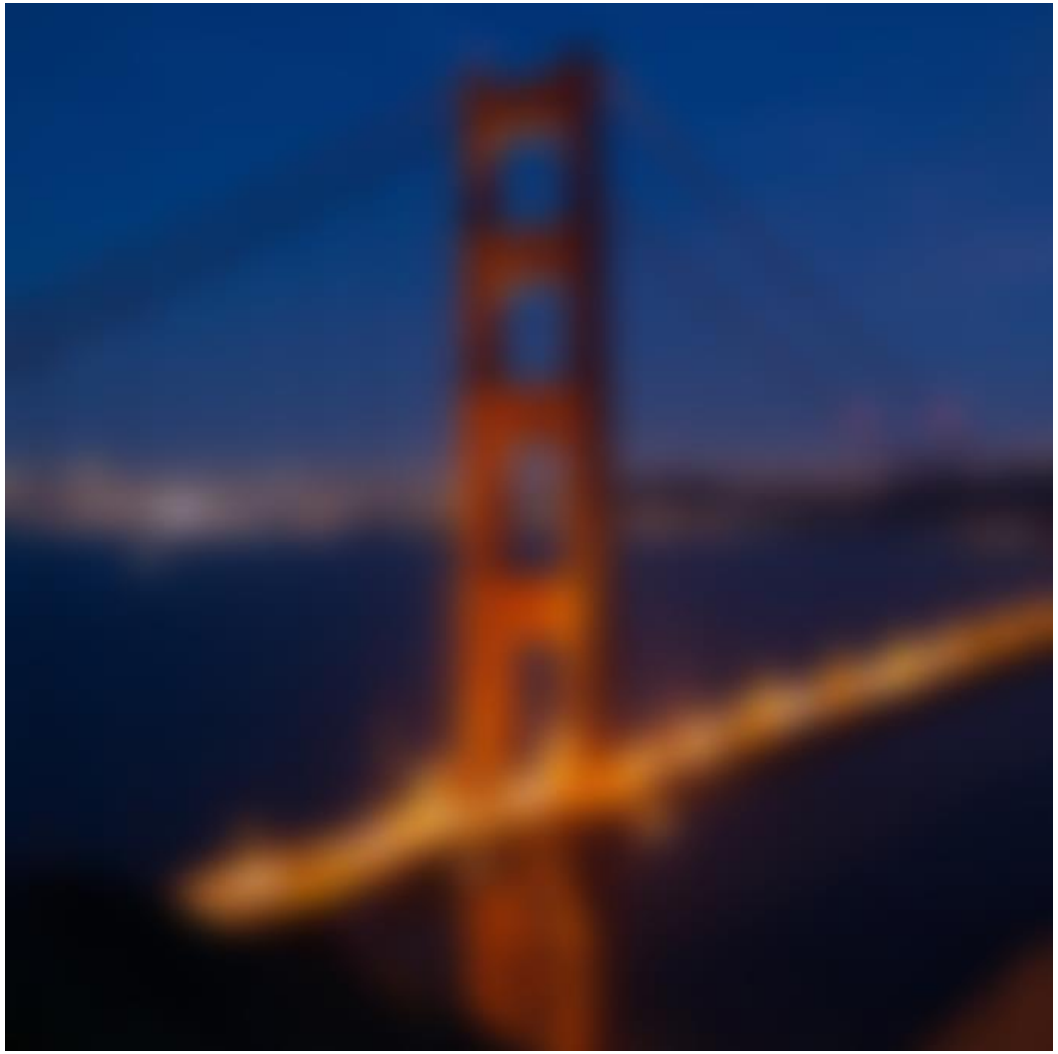
$$f(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}}$$



- weight falls off with distance from center pixel
- theoretically infinite, in practice truncated to some maximum distance
- Any heuristics for selecting where to truncate?
  - usually at 2σ or-3σ

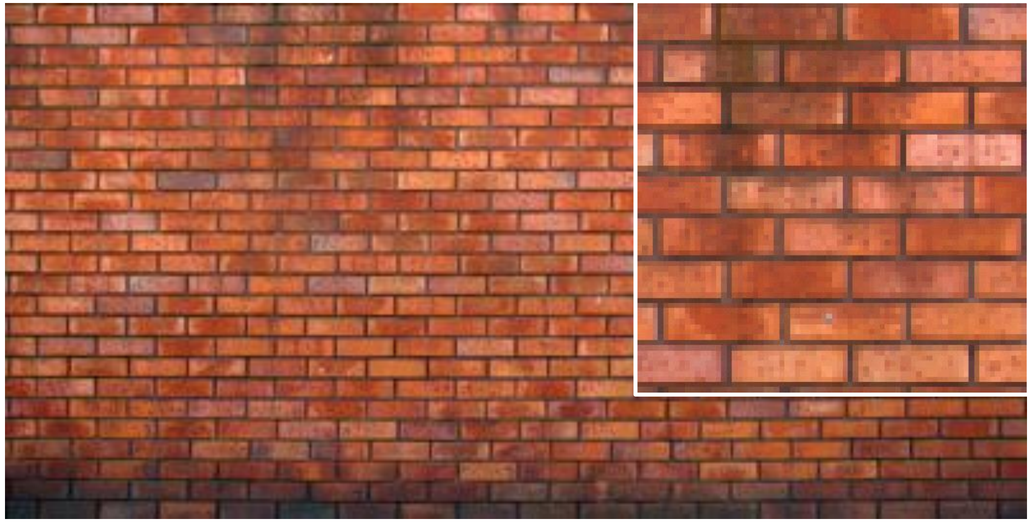
kernel $\frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$

Gaussian filtering example

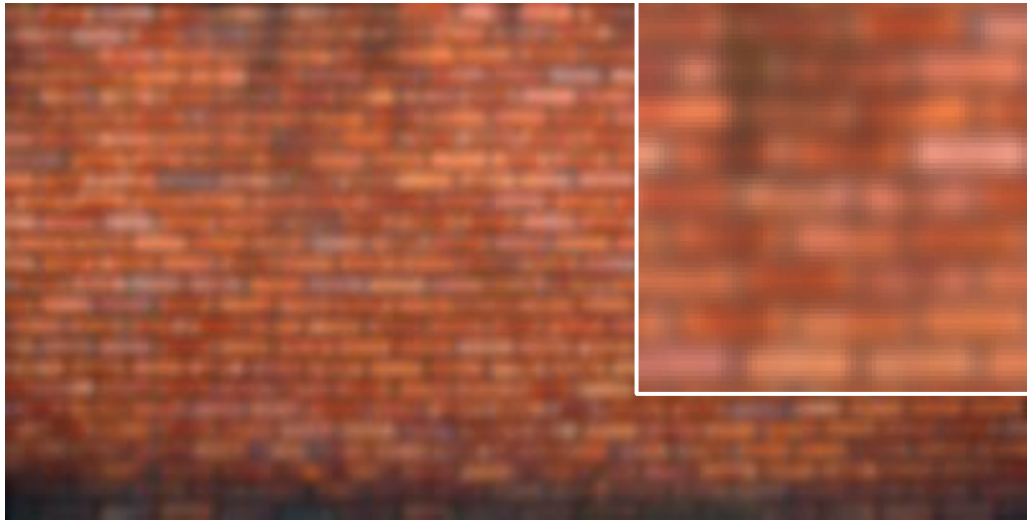


Gaussian vs box filtering

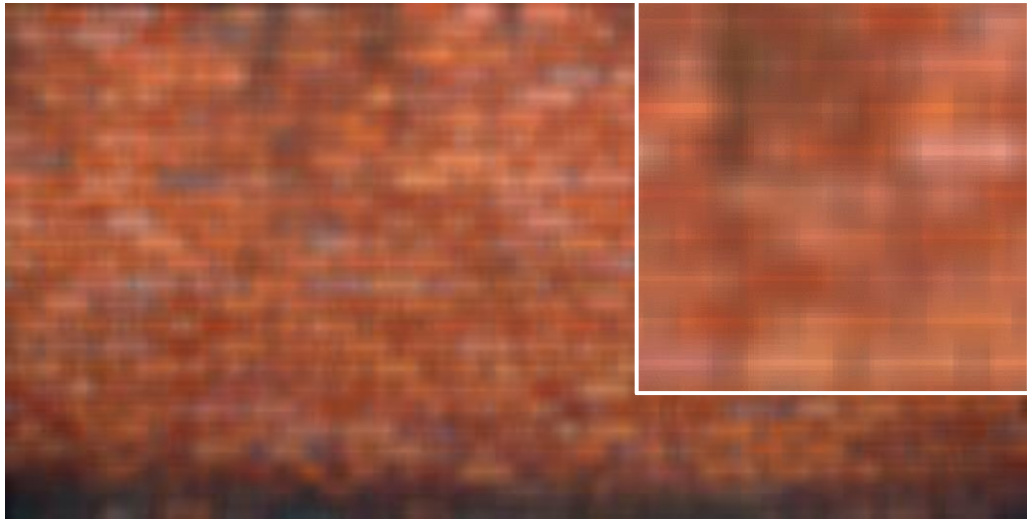
- Which blur do you think better?



original



7x7 Gaussian



7x7 box

Other filters

input

filter

0	0	0
0	1	0
0	0	0

output

unchanged

Other filters

input

filter

0	0	0
0	1	0
0	0	0

output

unchanged

input

filter

0	0	0
0	0	1
0	0	0

output

?

Other filters

input

filter

0	0	0
0	1	0
0	0	0

output

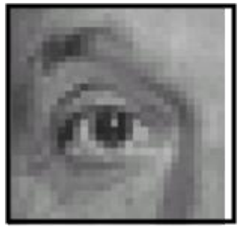
unchanged

input

filter

0	0	0
0	0	1
0	0	0

output



shift to left
by one

Other filters

input

filter

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

output

?

Other filters

- do nothing for flat areas
- stress intensity peaks

input

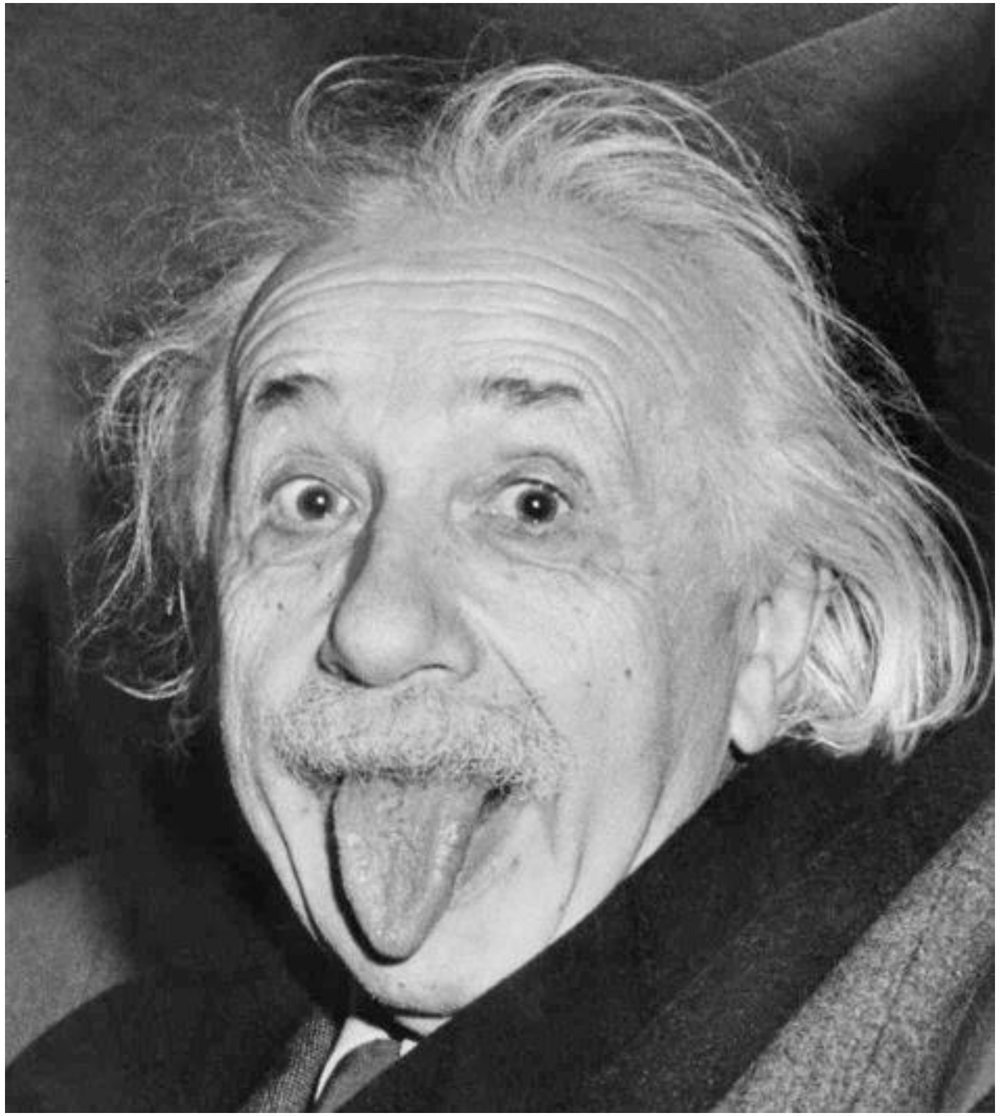
filter

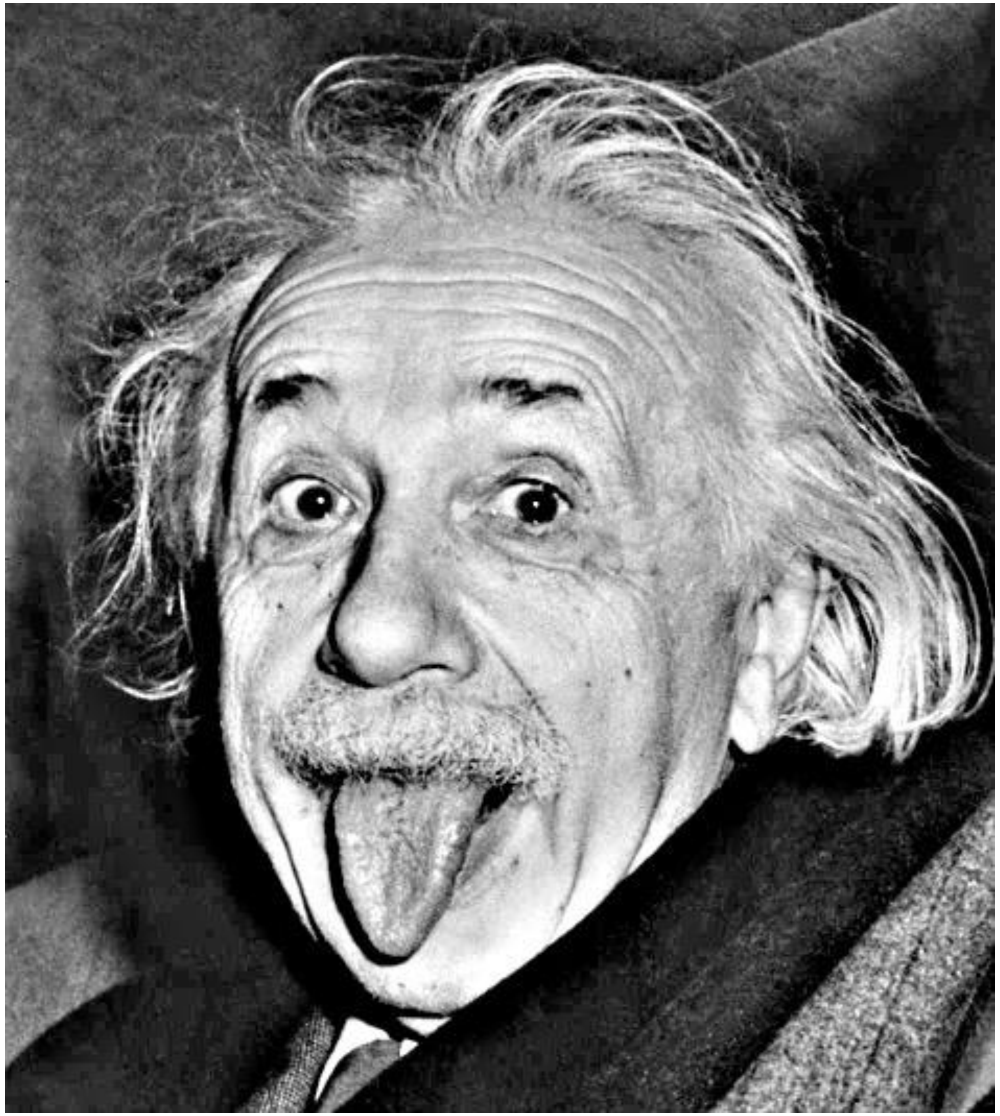
$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \frac{1}{9} \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

output

sharpening

Sharpening examples





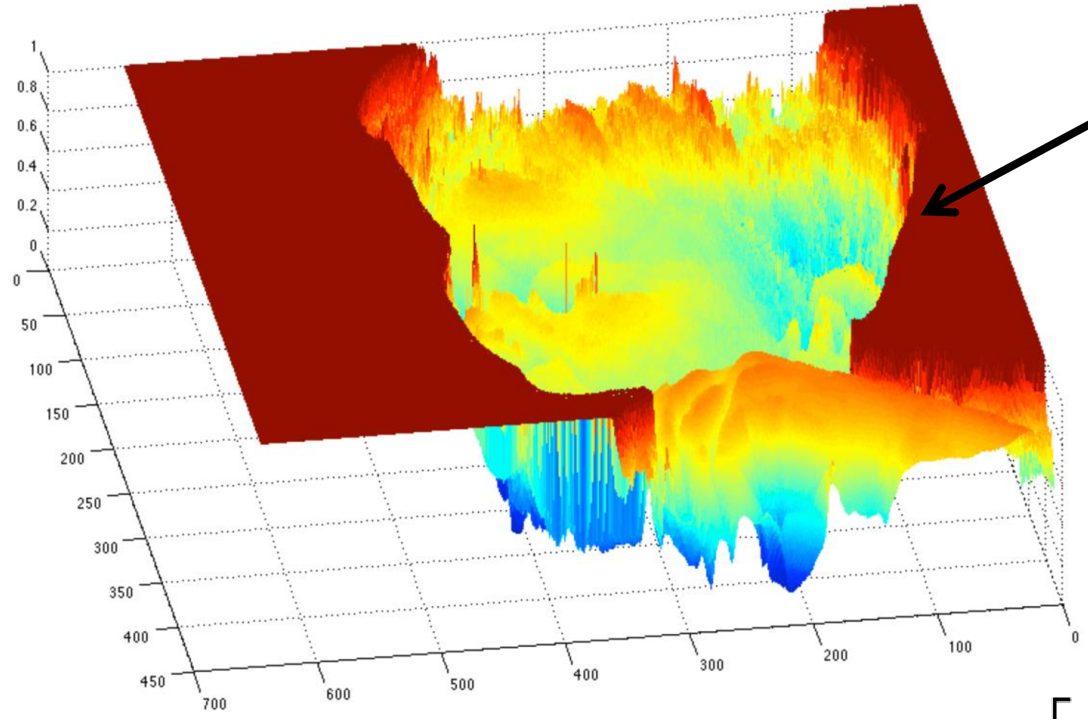
Sharpening examples

Image Gradients

What are image edges?

grayscale image

$f(\boldsymbol{x})$



Very sharp discontinuities in intensity.

domain $\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

Detecting edges

- How would you detect edges in an image (i.e., discontinuities in a function)?

Detecting edges

- How would you detect edges in an image (i.e., discontinuities in a function)?
  - You take derivatives: derivatives are large at discontinuities.
- How do you differentiate a discrete image (or any other discrete signal)?

Detecting edges

- How would you detect edges in an image (i.e., discontinuities in a function)?
  - You take derivatives: derivatives are large at discontinuities.

- How do you differentiate a discrete image (or any other discrete signal)?
  - You use finite differences.

Finite differences

- High-school reminder: definition of a derivative using forward difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Finite differences

- High-school reminder: definition of a derivative using forward difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- Alternative: use central difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+0.5h) - f(x-0.5h)}{h}$$

- For discrete signals: Remove limit and set h = 2

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

- What convolution kernel does this correspond to?

Finite differences

- High-school reminder: definition of a derivative using forward difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- Alternative: use central difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+0.5h) - f(x-0.5h)}{h}$$

- For discrete signals: Remove limit and set h = 2

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

- What convolution kernel does this correspond to?

-1	0	1	?

1	0	-1	?

Finite differences

- High-school reminder: definition of a derivative using forward difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- Alternative: use central difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+0.5h) - f(x-0.5h)}{h}$$

- For discrete signals: Remove limit and set h = 2

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

- What convolution kernel does this correspond to?

1D derivative filter

1	0	-1

The Sobel filter

- What filter is this?

$$\underbrace{\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix}}_{\text{Sobel filter}} = \begin{bmatrix}1\\ 2\\ 1\end{bmatrix} * \underbrace{\begin{bmatrix}1 & 0 & -1\end{bmatrix}}_{\text{1D derivative filter}}$$

The Sobel filter

- In a 2D image, does this filter responses along horizontal or vertical lines?

$$\underbrace{\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}}_{\text{Sobel filter}} = \underbrace{\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}}_{\text{Blurring}} * \underbrace{\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}}_{\text{1D derivative filter}}$$

The Sobel filter

- Horizontal Sobel filter:

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

- What does the vertical Sobel filter look like?

The Sobel filter

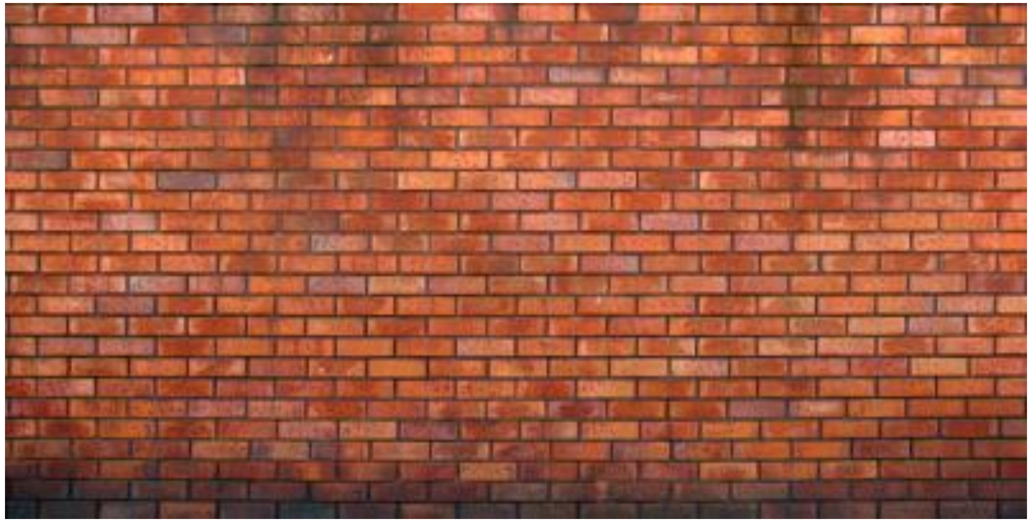
- Horizontal Sobel filter (x-axis):

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

- Vertical Sobel filter (y-axis):

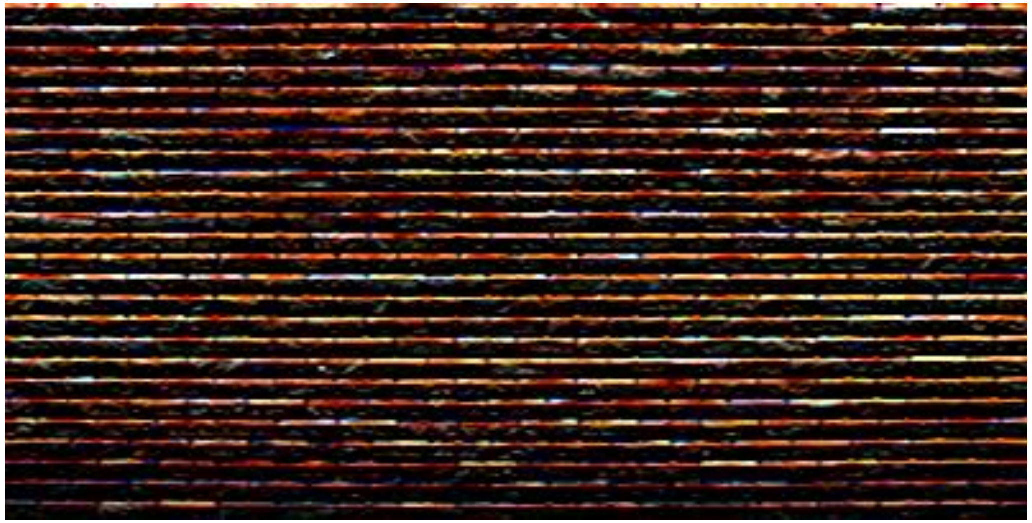
$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Sobel filter example



original

horizontal Sobel filter



vertical Sobel filter

Computing image gradients

- Select your favorite derivative filters.

$$\boldsymbol{S}_x = \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} \qquad \boldsymbol{S}_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

Computing image gradients

- Select your favorite derivative filters.

$$\boldsymbol{S}_x = \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline\end{array} \qquad \boldsymbol{S}_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline\end{array}$$

- Convolve with the image to compute derivatives.

$$\frac{\partial \boldsymbol{f}}{\partial x} = \boldsymbol{S}_x \otimes \boldsymbol{f} \qquad \frac{\partial \boldsymbol{f}}{\partial y} = \boldsymbol{S}_y \otimes \boldsymbol{f}$$

Computing image gradients

- Select your favorite derivative filters.

$$\boldsymbol{S}_x = \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline\end{array} \qquad \boldsymbol{S}_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline\end{array}$$

- Convolve with the image to compute derivatives.

$$\frac{\partial \boldsymbol{f}}{\partial x} = \boldsymbol{S}_x \otimes \boldsymbol{f} \qquad \frac{\partial \boldsymbol{f}}{\partial y} = \boldsymbol{S}_y \otimes \boldsymbol{f}$$

- Form the image gradient and compute its direction and amplitude.

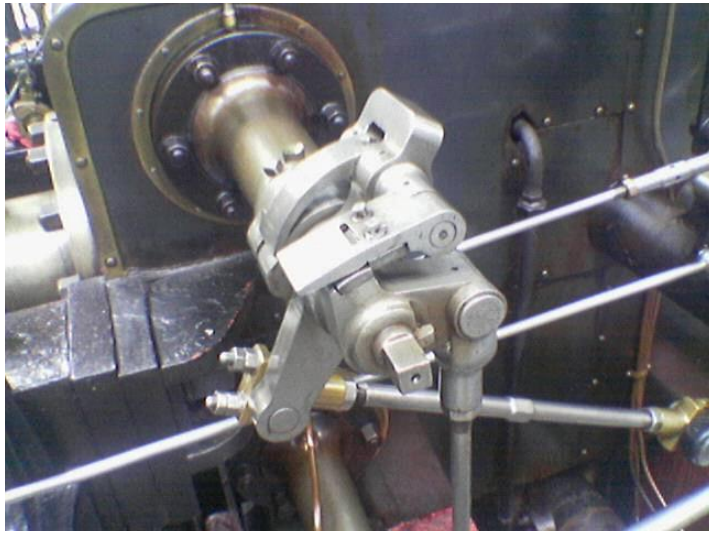
$$\nabla \boldsymbol{f} = \left[\frac{\partial \boldsymbol{f}}{\partial x}, \frac{\partial \boldsymbol{f}}{\partial y}\right] \qquad \theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right) \qquad ||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

gradient direction amplitude

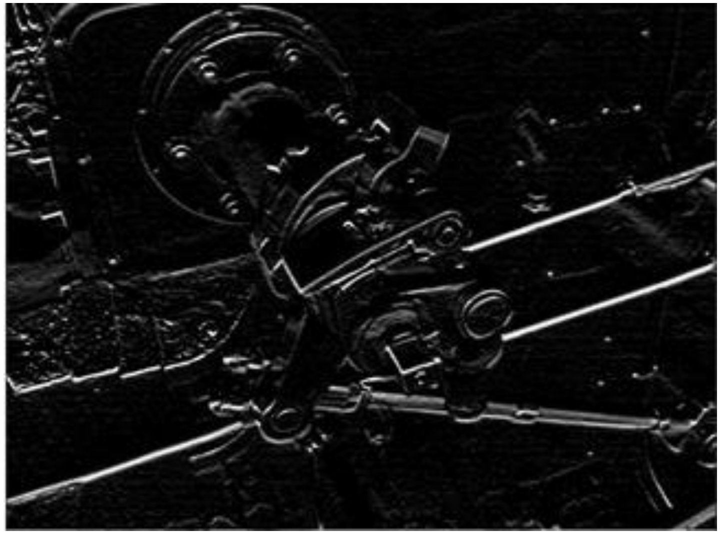
Image gradient example

- How does the gradient direction relate to these edges?

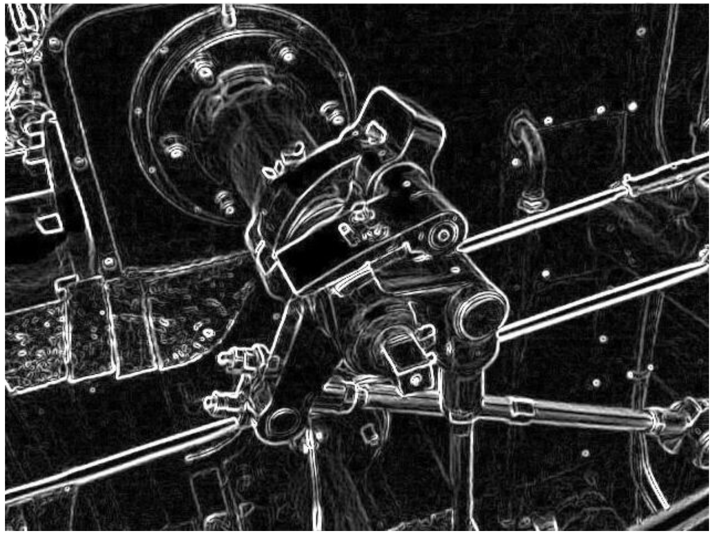
original



vertical
derivative



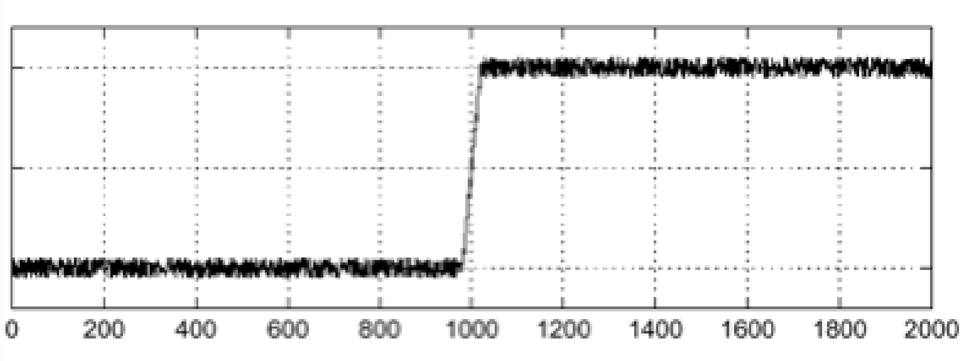
gradient
amplitude



horizontal
derivative

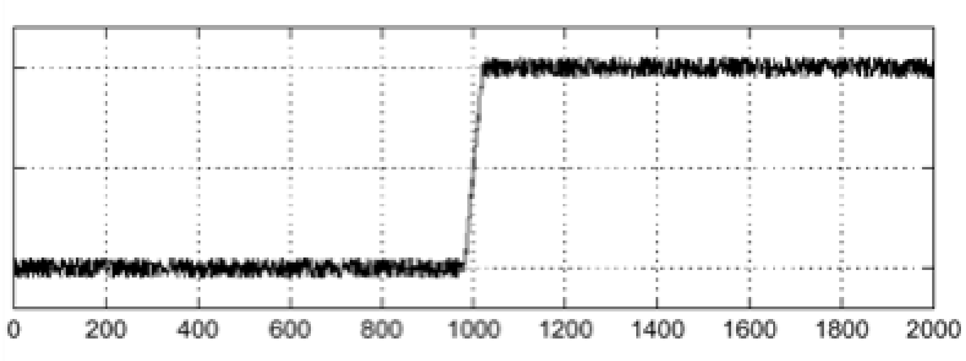
How do you find the edge of this signal?

intensity plot



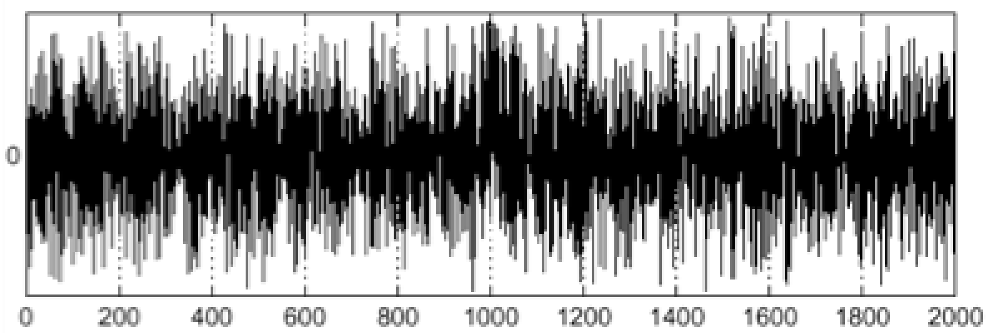
How do you find the edge of this signal?

intensity plot



- Using a derivative filter:

derivative plot



- What's the problem here?

Differentiation is very sensitive to noise

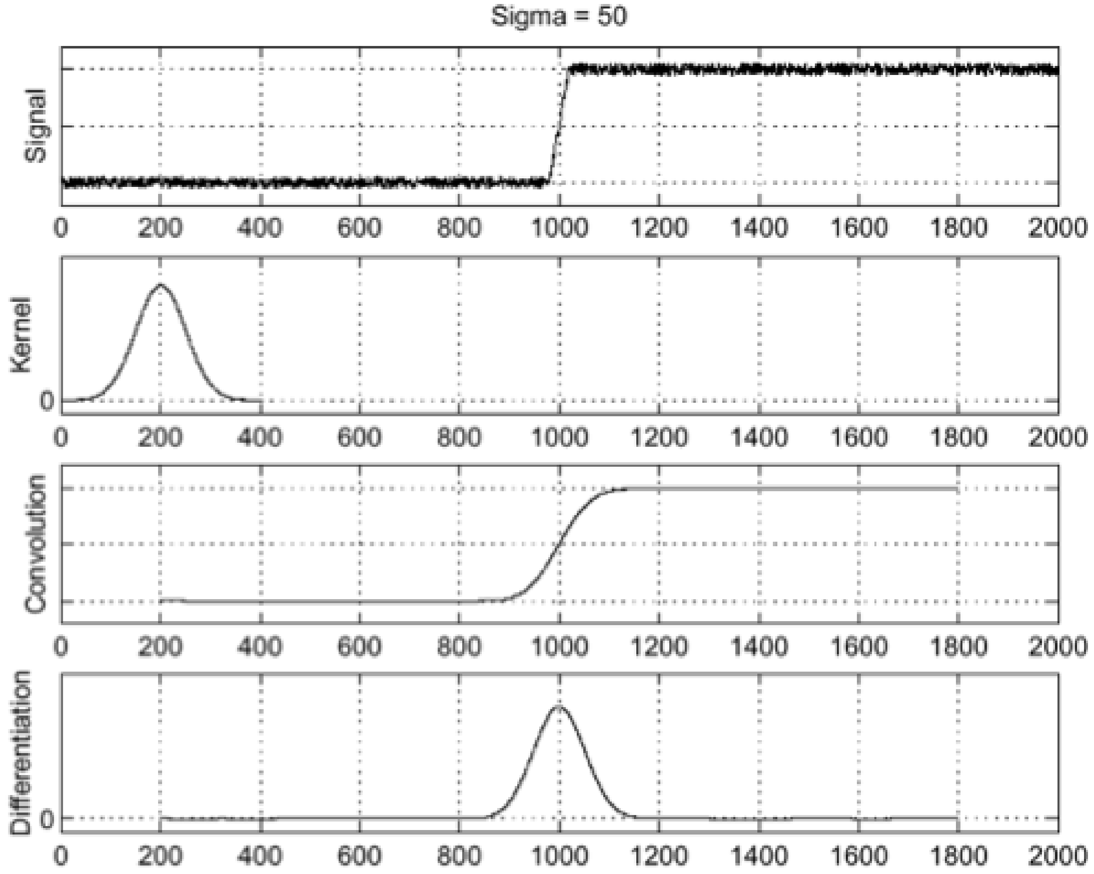
- When using derivative filters, it is critical to blur first!

input

Gaussian

blurred

derivative of
blurred



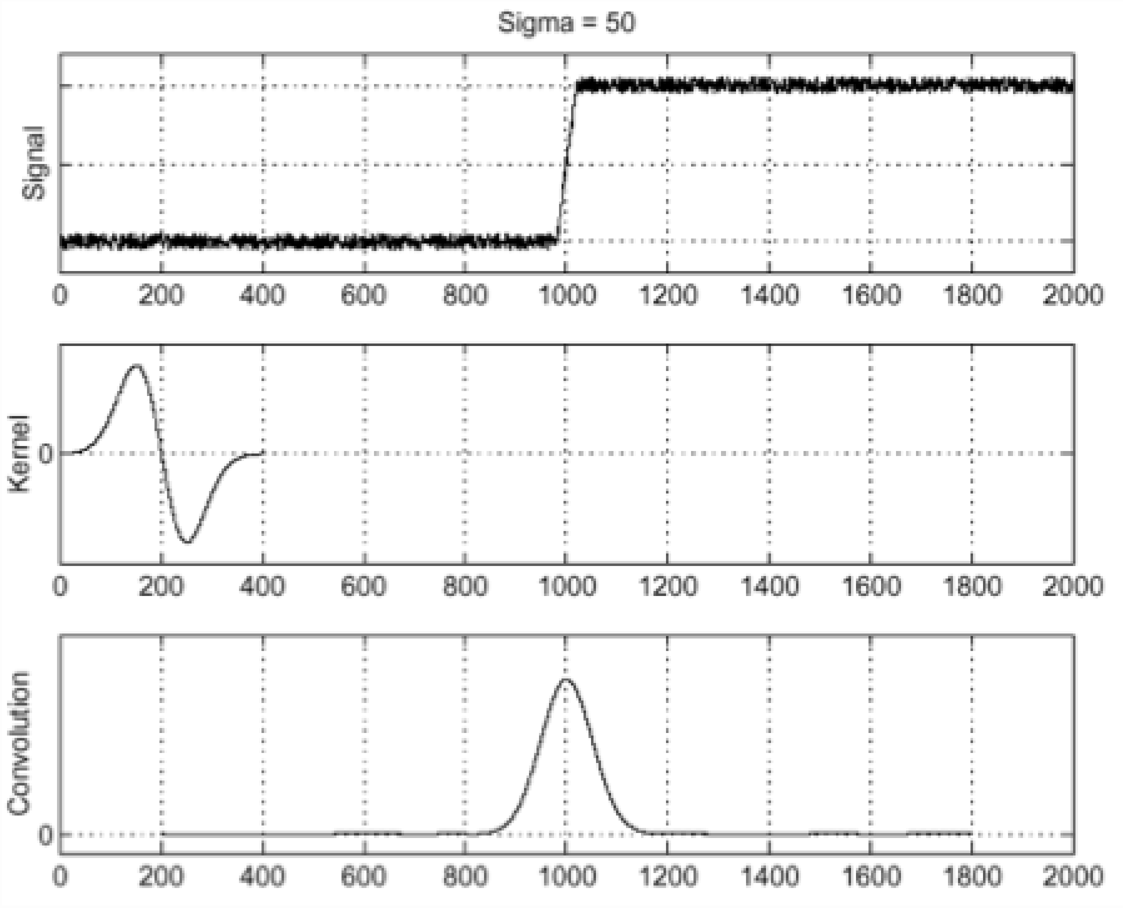
Derivative of Gaussian (DoG) filter

- Derivative theorem of convolution: $\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$
- How many operations did we save?

input

derivative of Gaussian

output (same as before)



Laplace filter

- Basically a second derivative filter.
  - We can use finite differences to derive it, as with first derivative filter.

first-order finite difference

$$f'(x) = \lim_{h \to 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h} \longrightarrow$$

1D derivative filter

1	0	-1

second-order finite difference

$$f''(x) = \lim_{h \to 0} \frac{f(x + h) - 2f(x) + f(x - h)}{h^2} \longrightarrow$$

Laplace filter

?

Laplace filter

- Basically a second derivative filter.
    - We can use finite differences to derive it, as with first derivative filter.

first-order finite difference

$$f'(x) = \lim_{h \to 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h} \longrightarrow$$

1D derivative filter

1	0	-1

second-order finite difference

$$f''(x) = \lim_{h \to 0} \frac{f(x + h) - 2f(x) + f(x - h)}{h^2} \longrightarrow$$

Laplace filter

1	-2	1

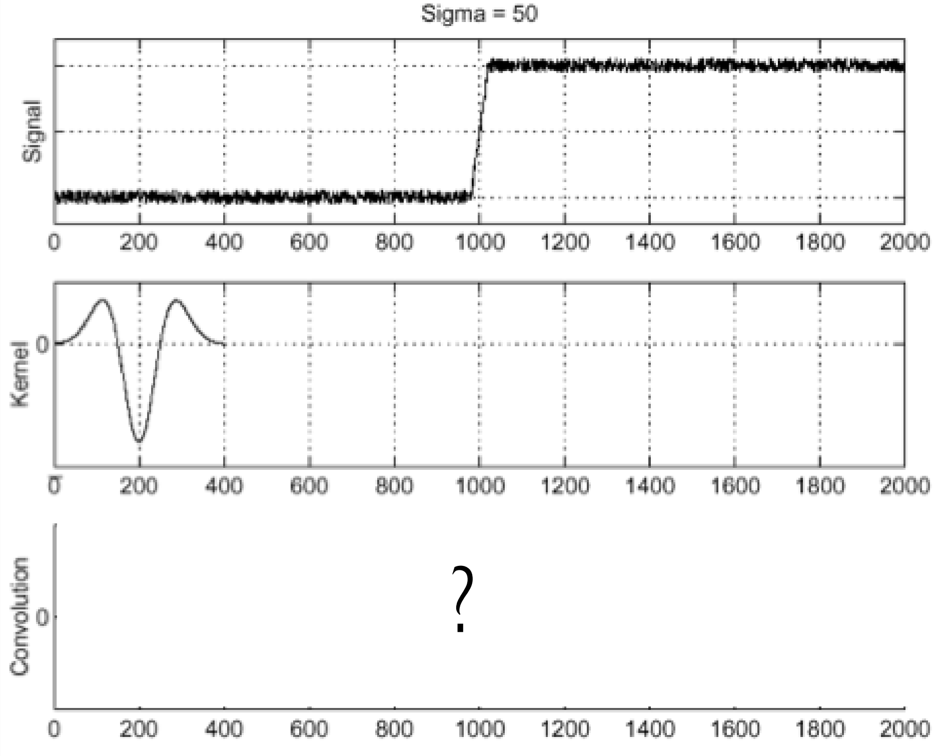
Laplacian of Gaussian (LoG) filter

- As with derivative, we can combine Laplace filtering with Gaussian filtering

input

Laplacian of
Gaussian

output



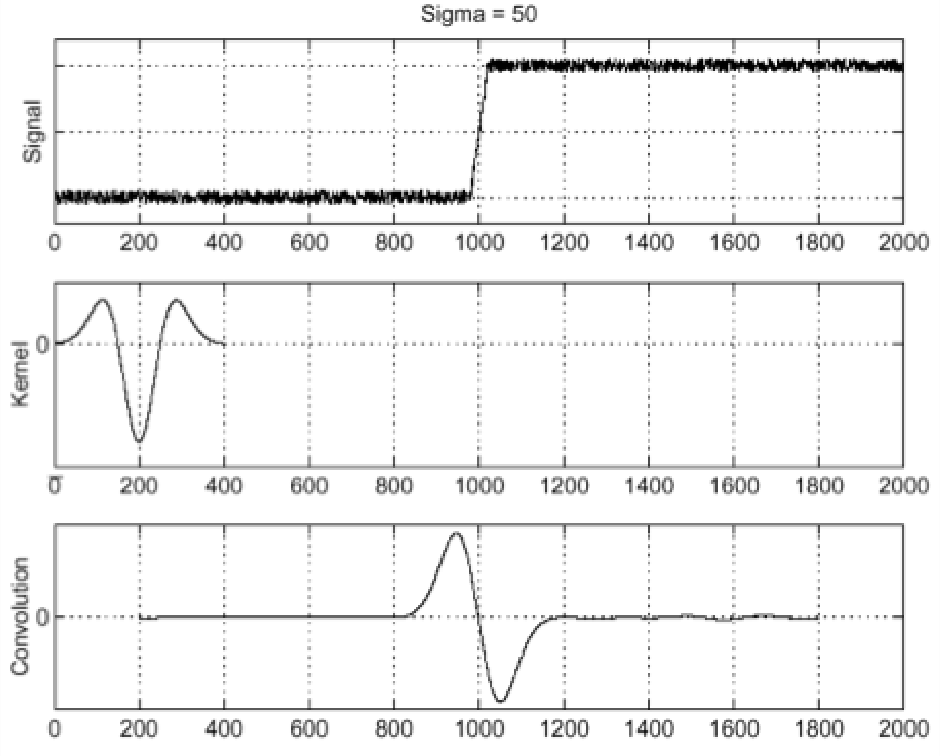
Laplacian of Gaussian (LoG) filter

- As with derivative, we can combine Laplace filtering with Gaussian filtering

input

Laplacian of Gaussian

output



“zero crossings” at edges

Laplace and LoG filtering examples

Laplacian of Gaussian filtering

Laplace filtering

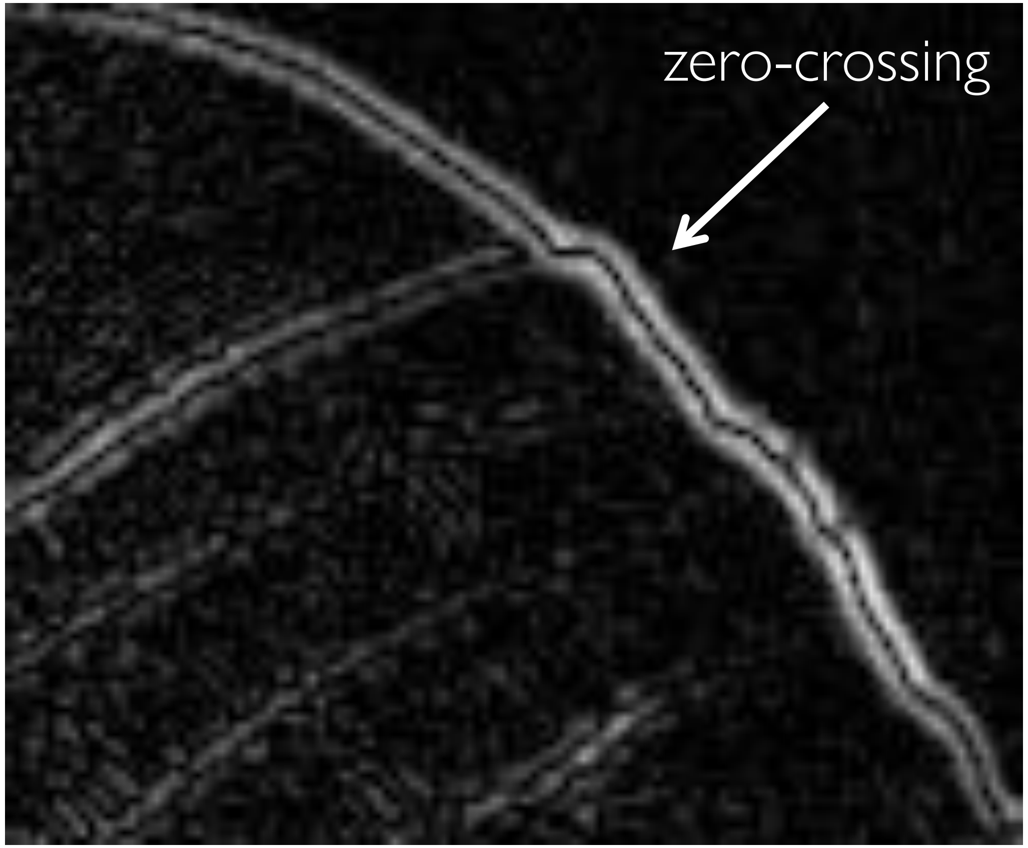
Laplacian of Gaussian vs Derivative of Gaussian

Laplacian of Gaussian filtering

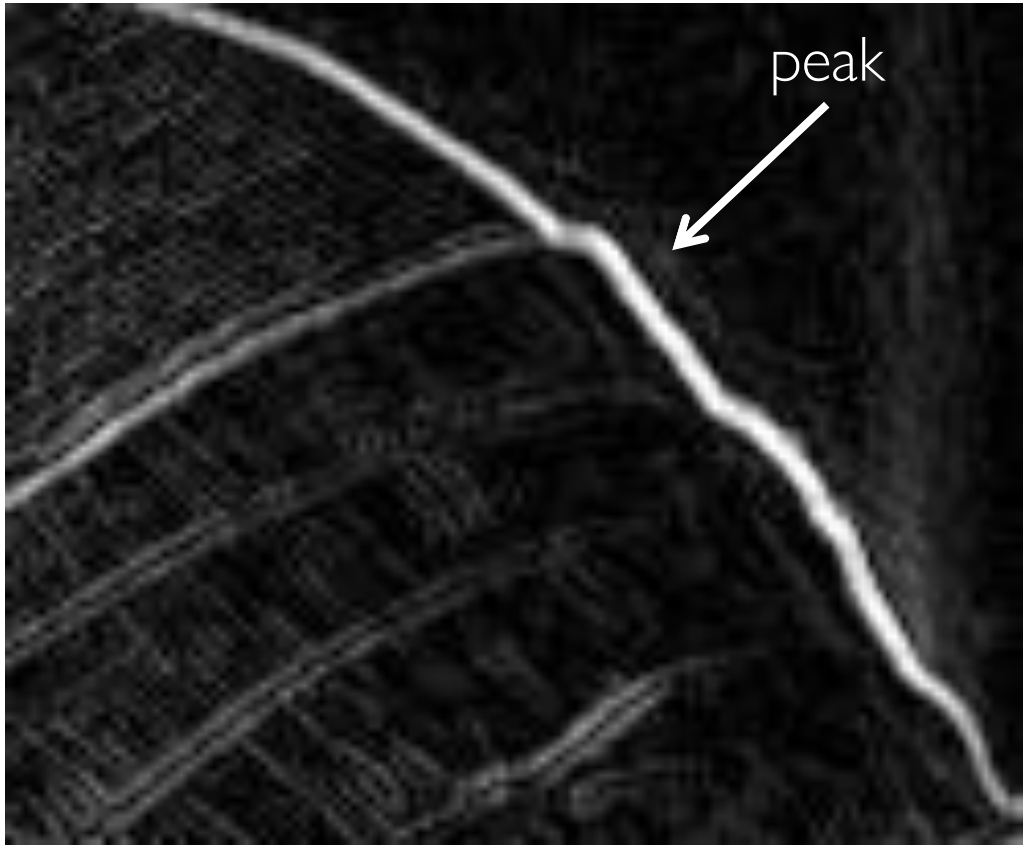
Derivative of Gaussian filtering

Laplacian of Gaussian vs Derivative of Gaussian

- Zero crossings are more accurate at localizing edges (but not very convenient).

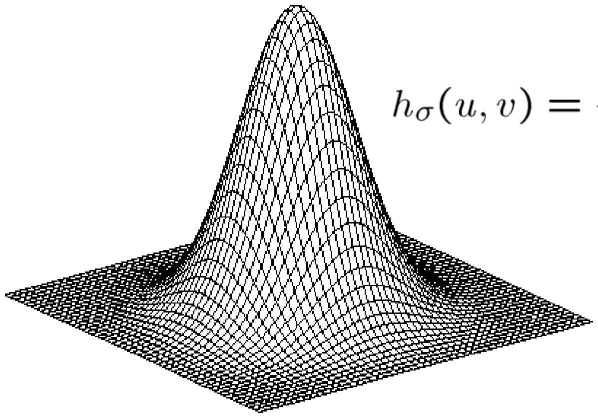


Laplacian of Gaussian filtering



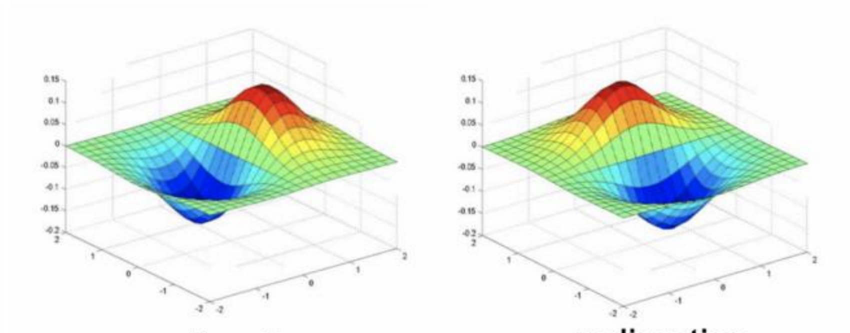
Derivative of Gaussian filtering

2D Gaussian filters

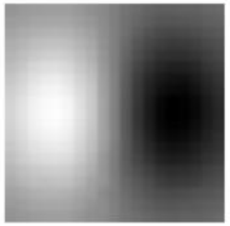


$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

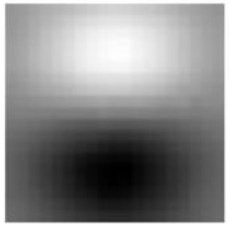
Gaussian



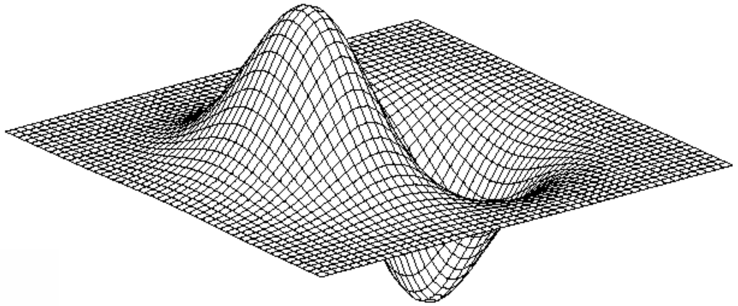
x-direction



y-direction

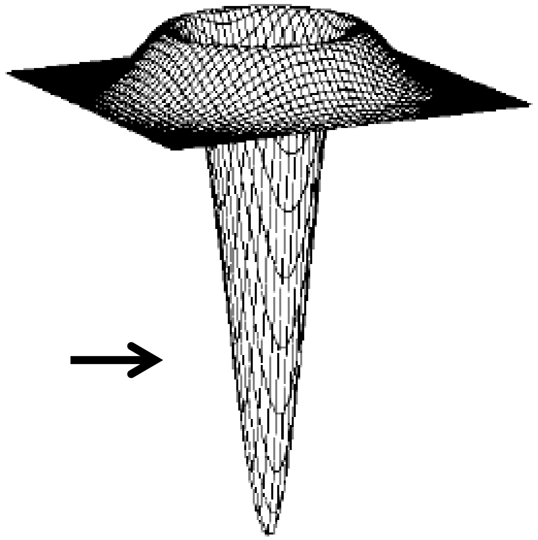


$$\frac{\partial}{\partial x} h_\sigma(u,v)$$



Derivative of Gaussian

$$\nabla^2 h_\sigma(u,v)$$



Laplacian of Gaussian