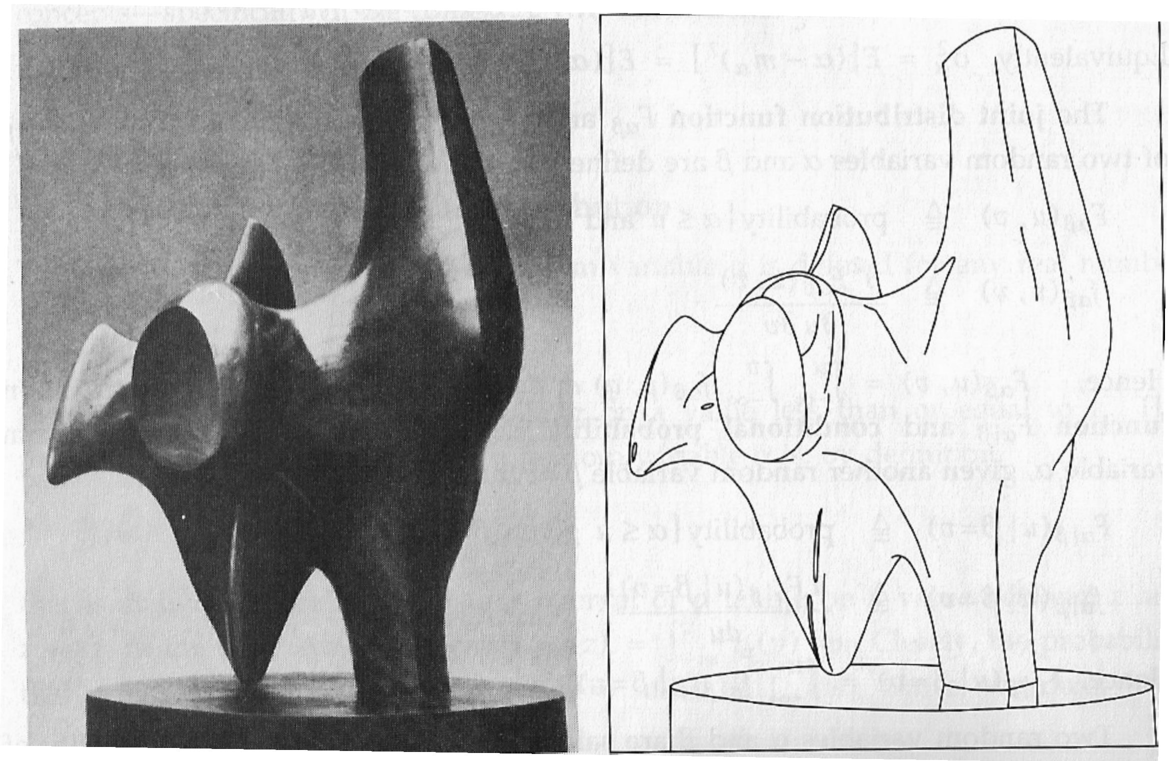
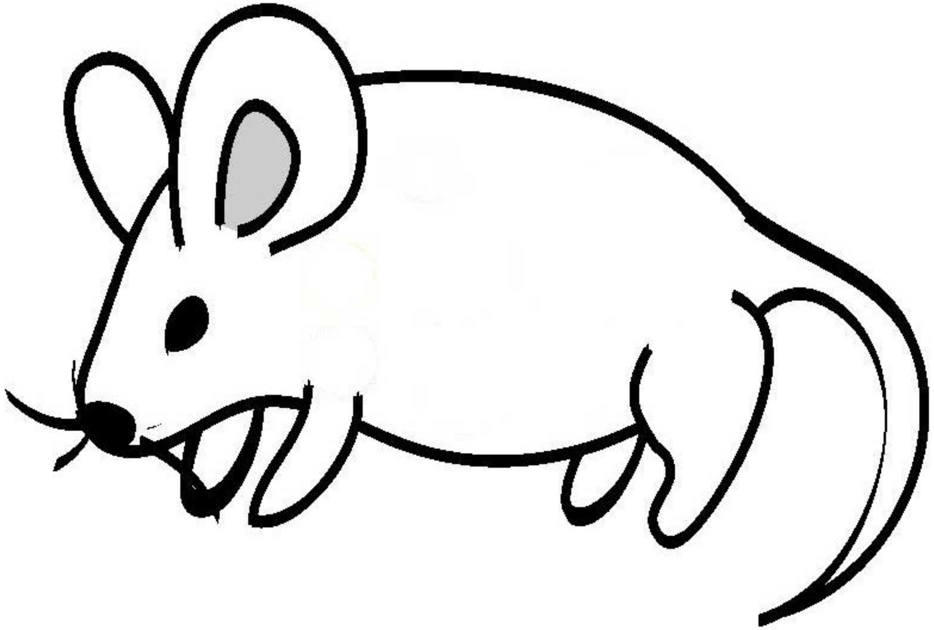
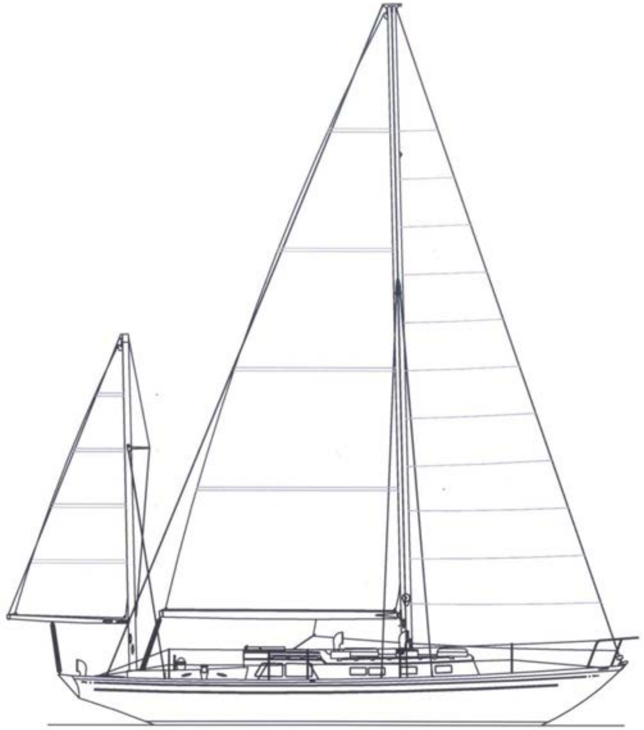
Edge Detection



- What's an edge
- Image gradients
- Edge operators

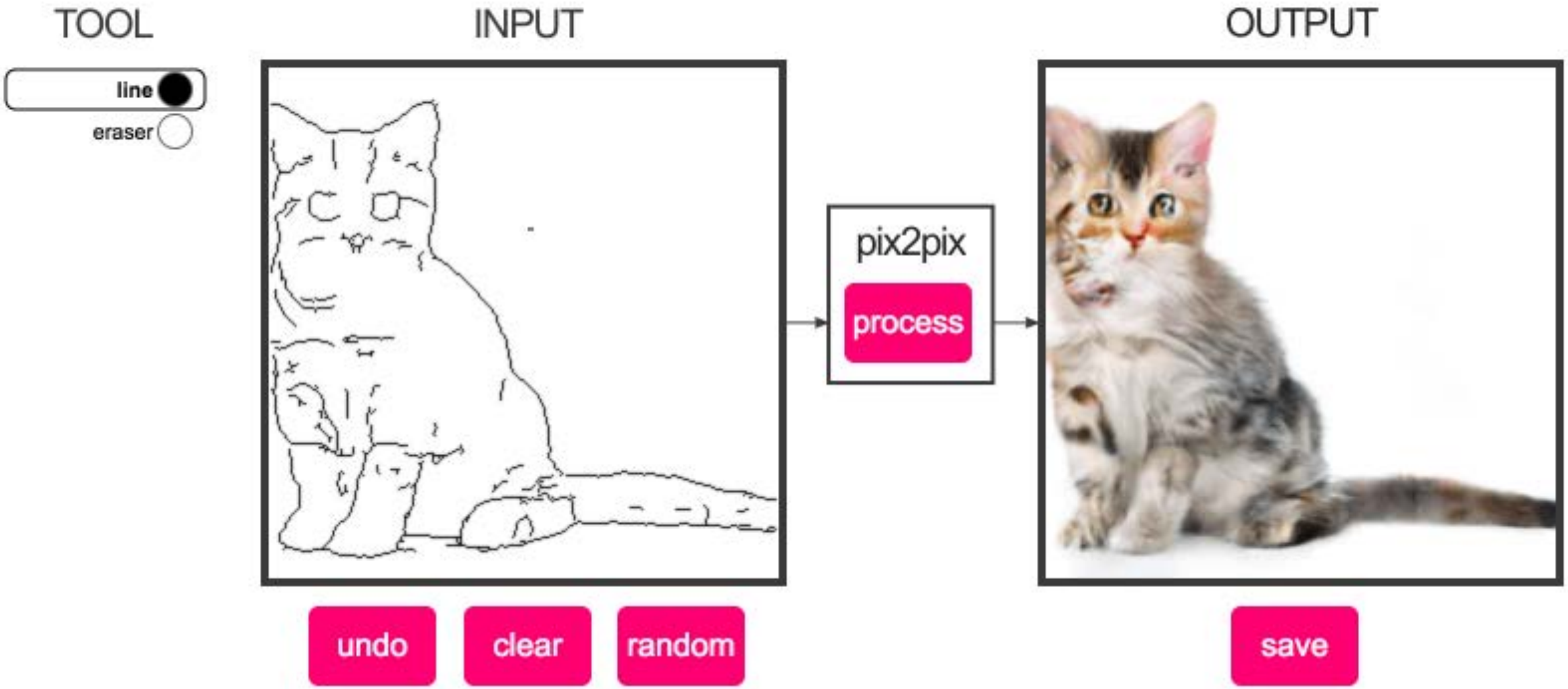
Line Drawings





- Edges seem fundamental to human perception.
- They form a compressed version of the image.

From Edges To Cats



Deep-Learning based generative model.

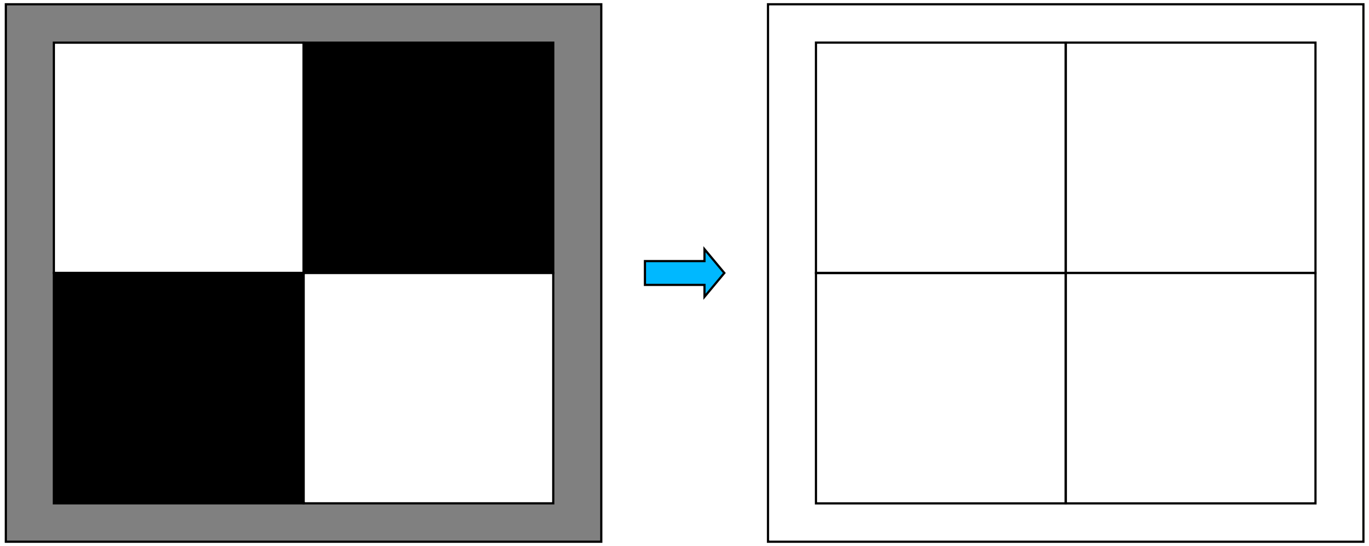
Demo https://affinelayer.com/pixsrv/

Maps and Overlays

Corridor

Corridor

Edges and Regions



Edges:

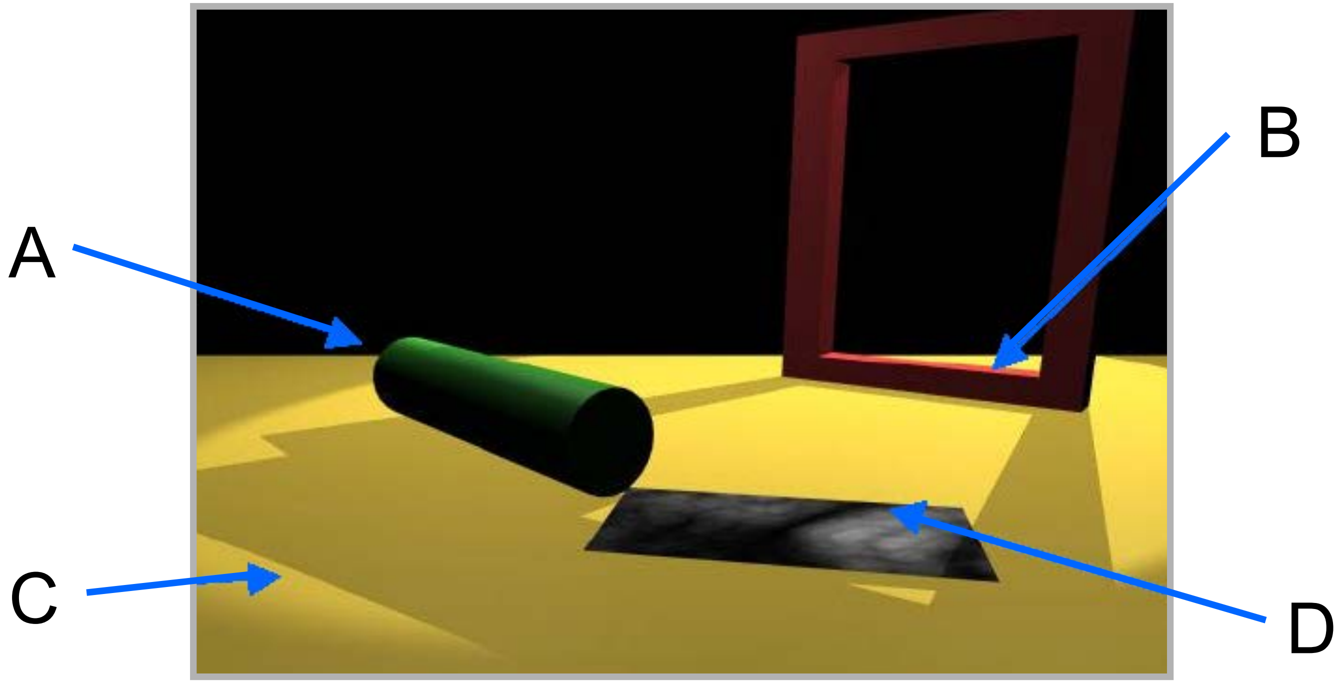
- Boundary between bland image regions.

Regions:

- Homogenous areas between edges.

→ Edge/Region Duality.

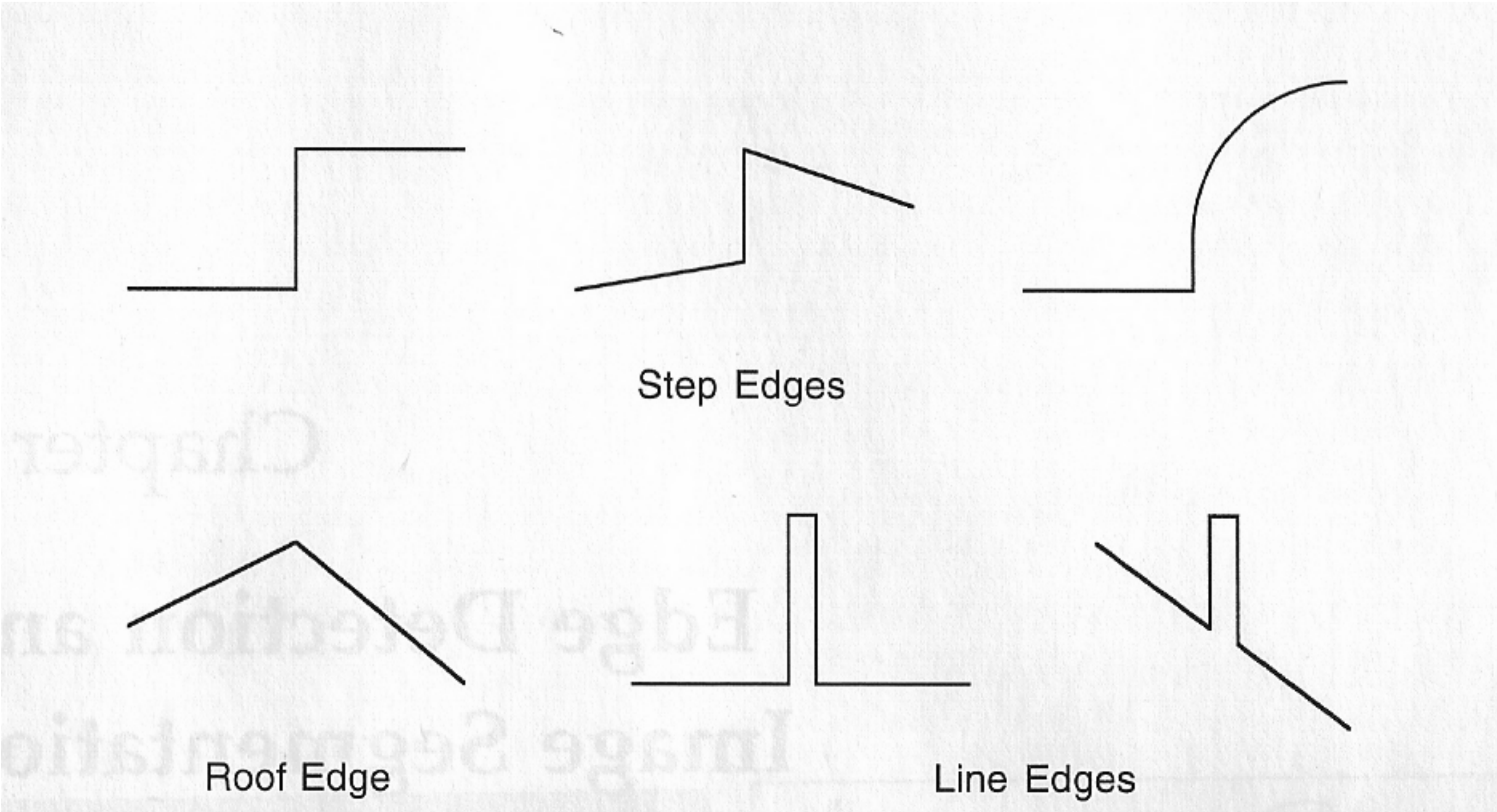
Discontinuities



- A. Depth discontinuity: Abrupt depth change in the world
- B. Surface normal discontinuity: Change in surface orientation
- C. Illumination discontinuity: Shadows, lighting changes
- D. Reflectance discontinuity: Surface properties, markings

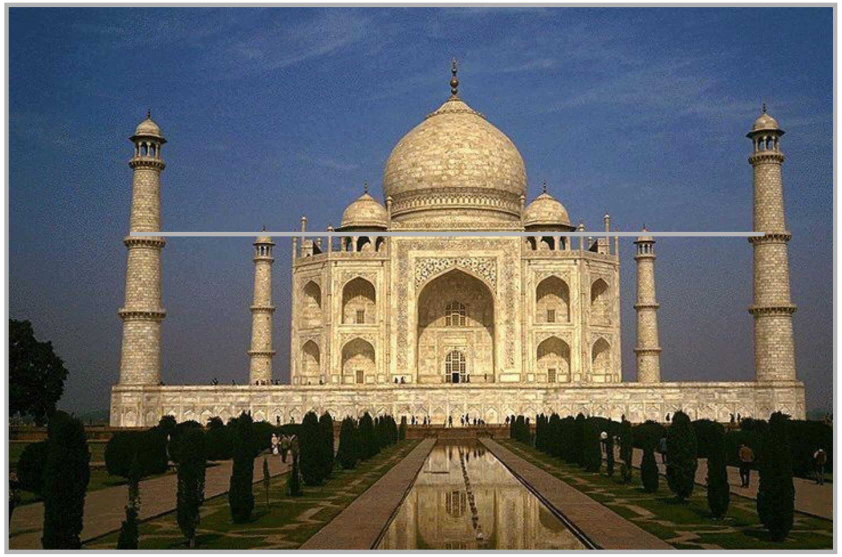
→ Sharply different Gray levels on both sides

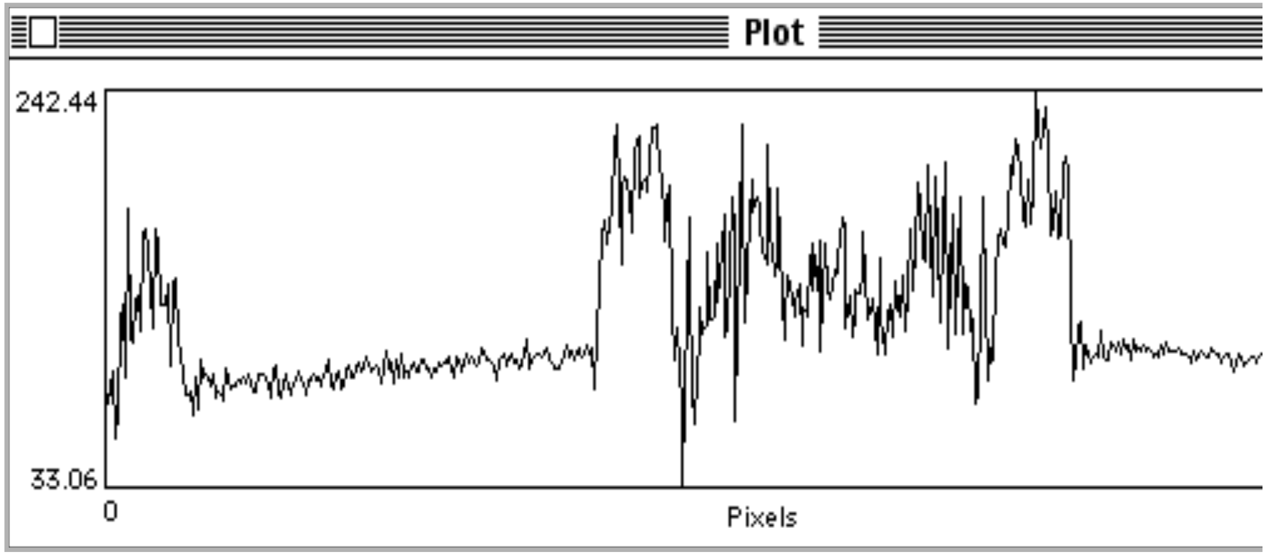
EDGE PROFILES

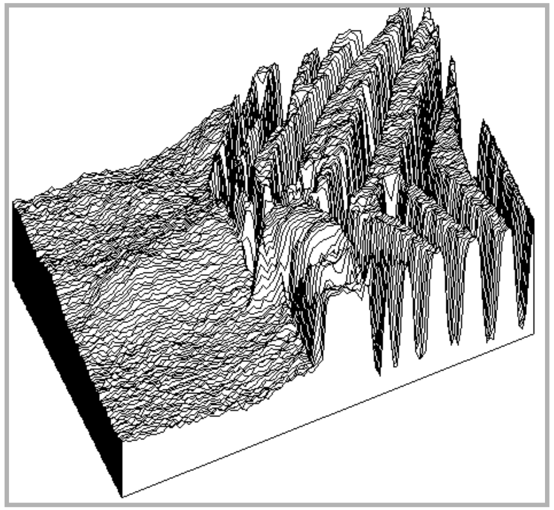


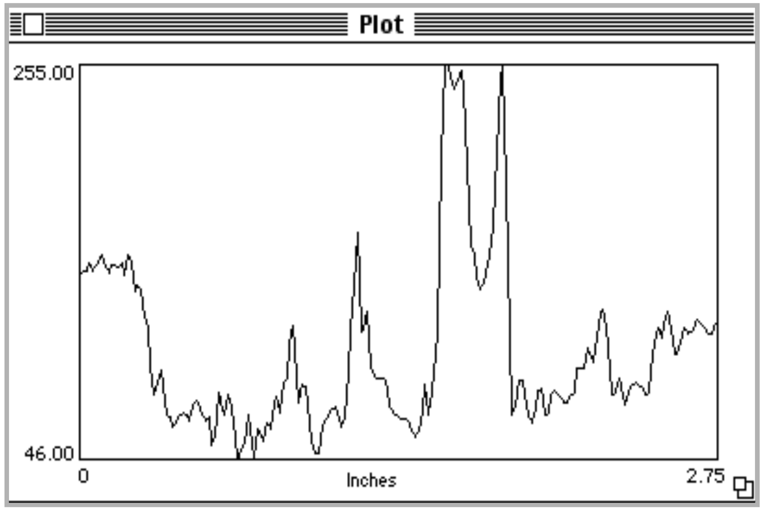
Edges are where a change occurs

REALITY

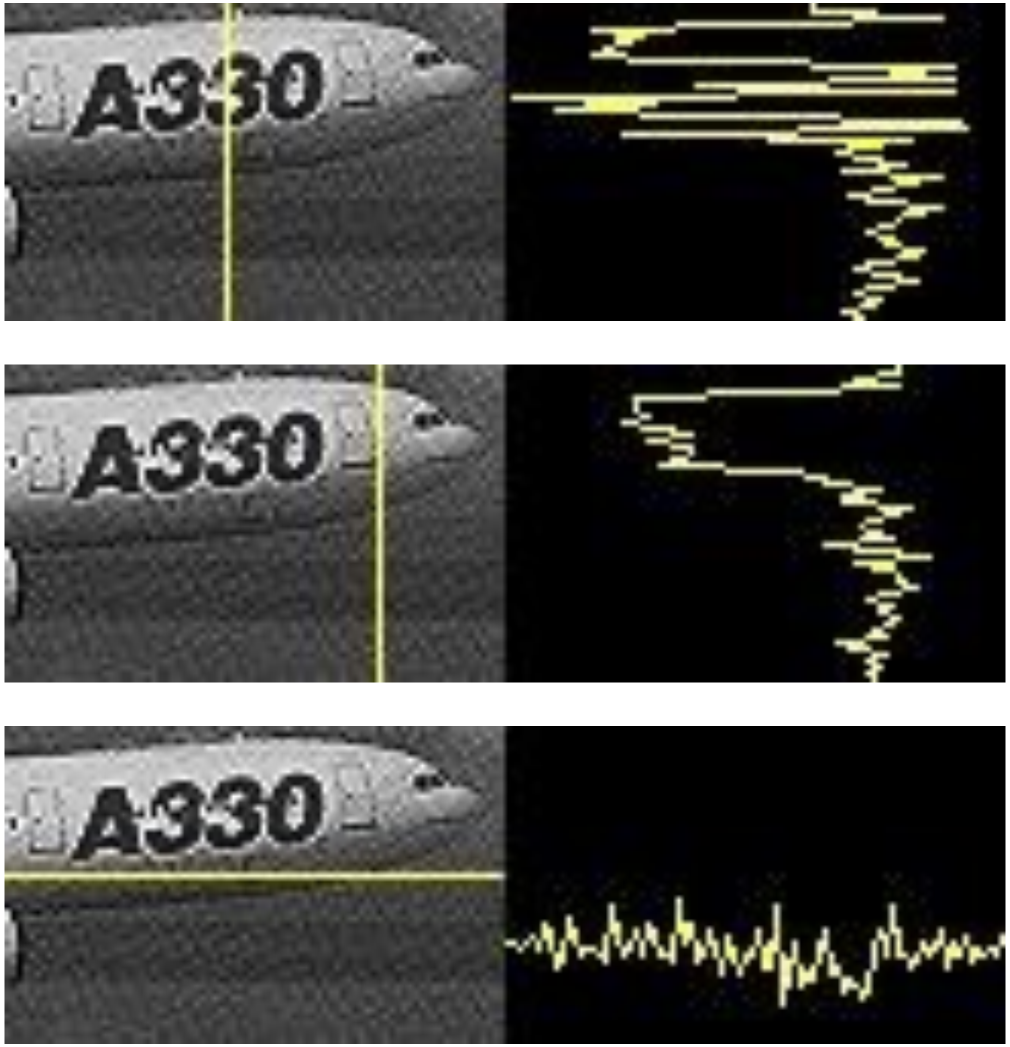








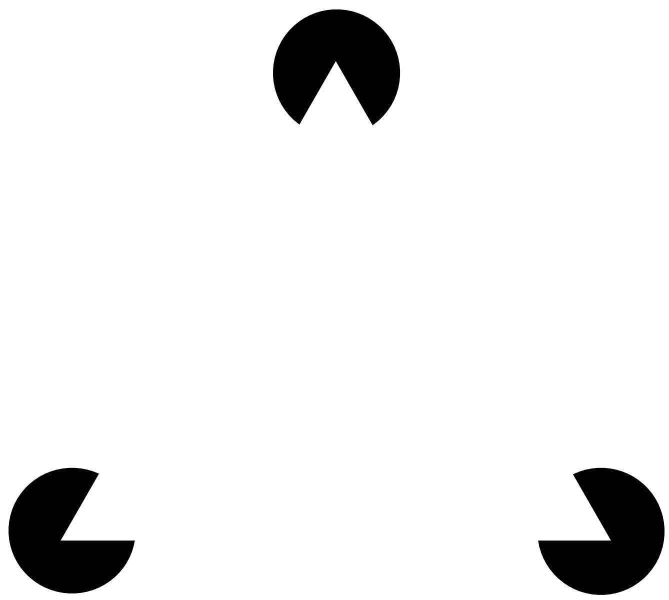
More Reality



Very noisy signals

→ Prior knowledge is required!!

Optional: Illusory Contours

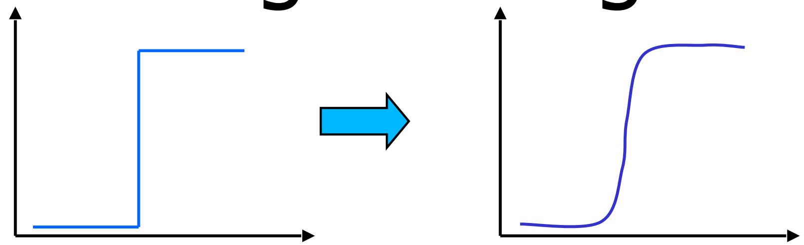


- No closed contour, but we still perceived an edge.
- This will not be further discussed in this class.

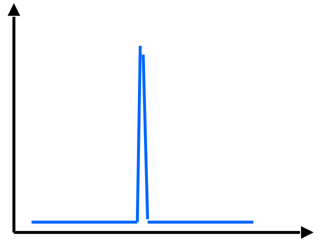
Ideal Step Edge

Rapid change in image => High local gradient

f(x) = step edge

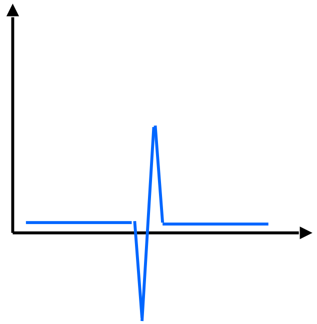


1st Derivative f’(x)



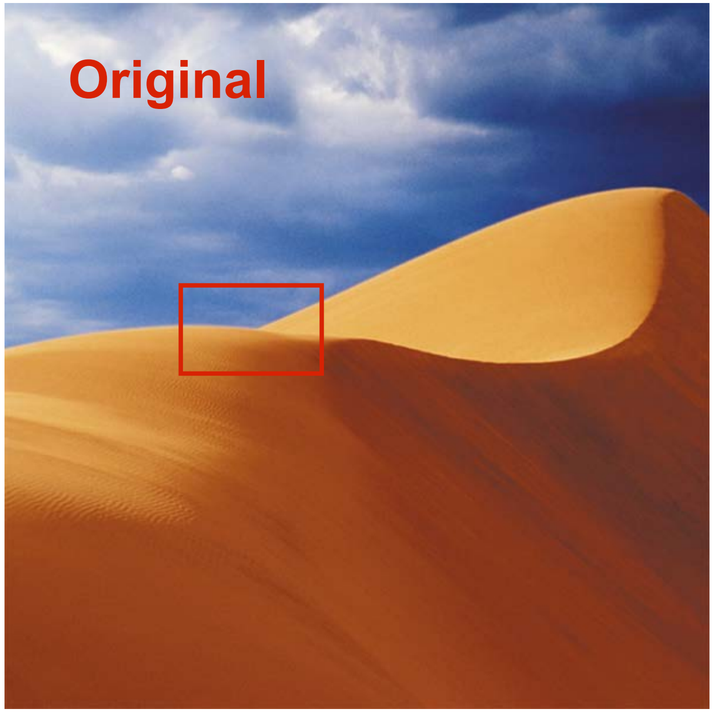
maximum

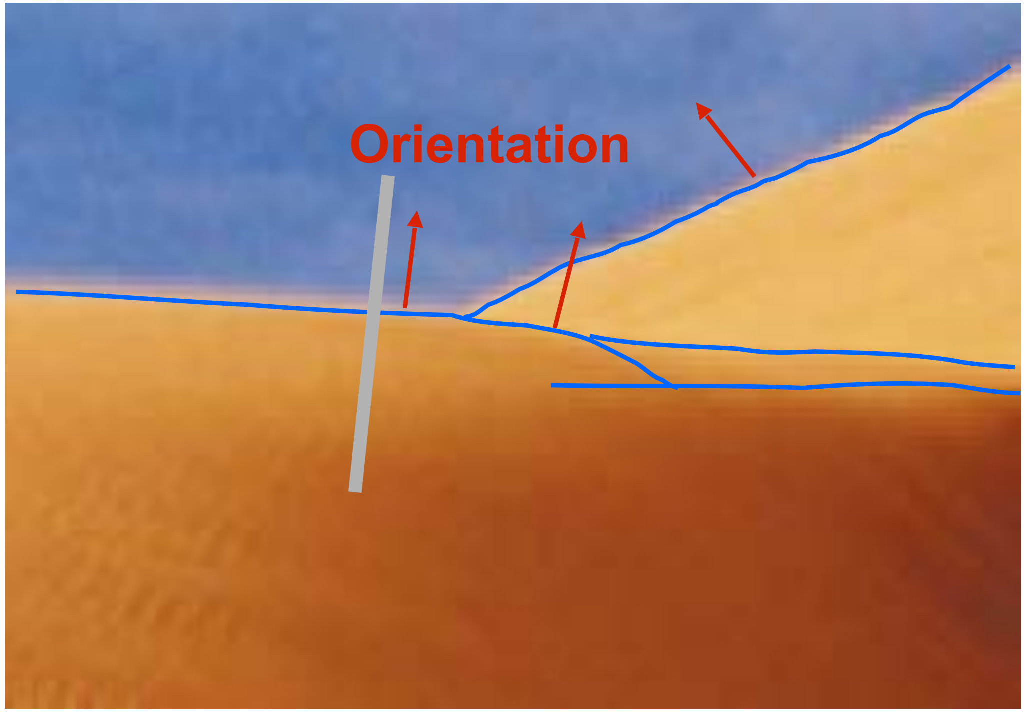
2nd Derivative f’’(x)

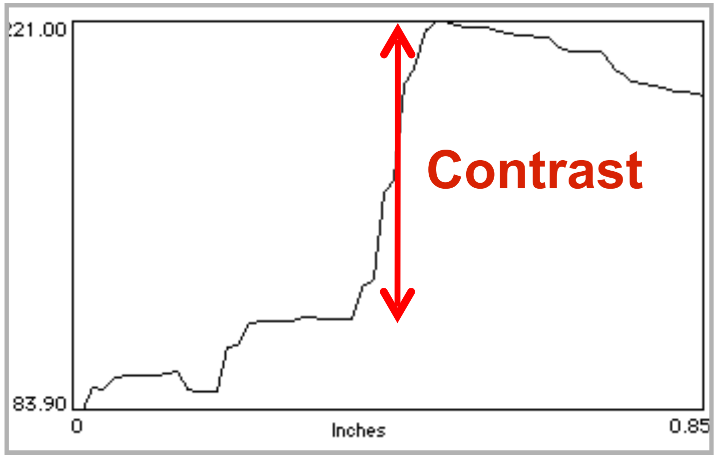


zero crossing

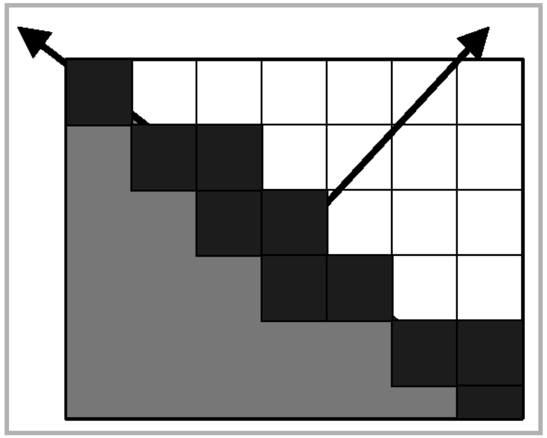
Edge Properties





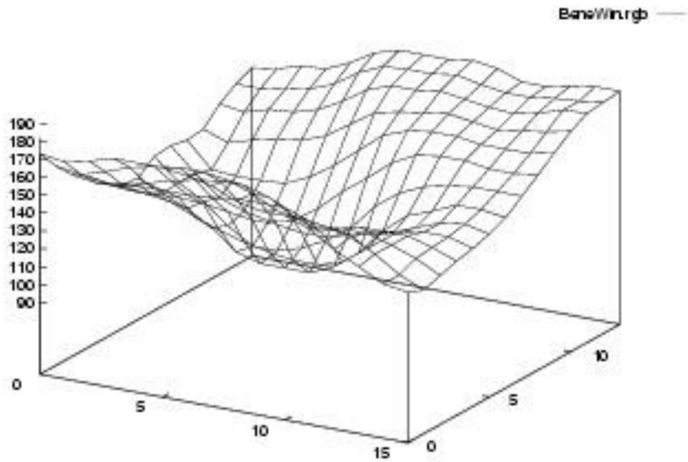


Edge Descriptors

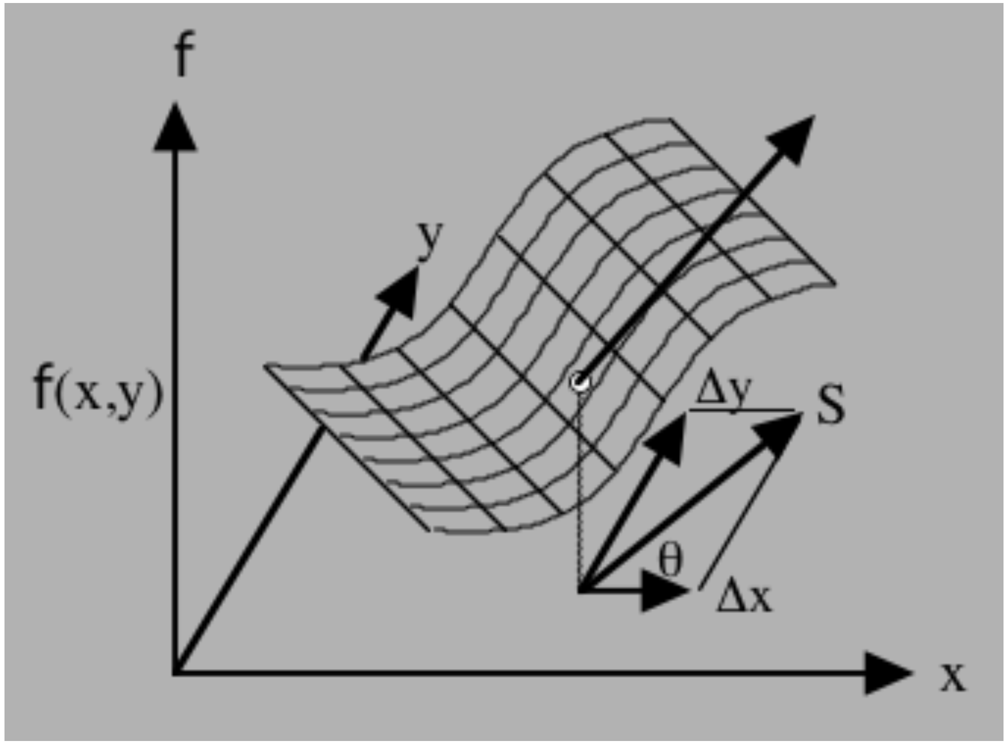


- Edge Normal:
  - Unit vector in the direction of maximum intensity change
- Edge Direction:
  - Unit vector perpendicular to the edge normal
- Edge position or center
  - Image location at which edge is located
- Edge Strength
  - Speed of intensity variation across the edge.

Images as 3-D Surfaces



Geometric Interpretation



Since I(x,y) is not a continuous function:

1. Locally fit a smooth surface.
2. Compute its derivatives.

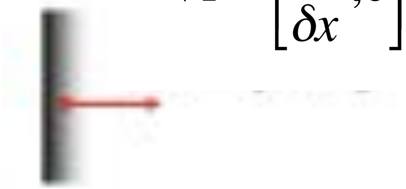
Image Gradient

The gradient of an image

$$\nabla I = \left[\frac{\delta I}{\delta x}, \frac{\delta I}{\delta y}\right]$$

points in the direction of most rapid change in intensity.

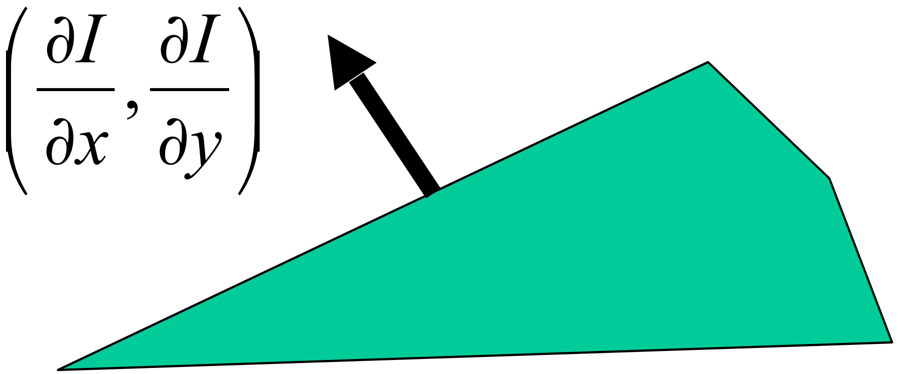
$$\nabla I = \left[\frac{\delta I}{\delta x}, 0\right]$$



$$\nabla I = \left[0, \frac{\delta I}{\delta y}\right]$$

$$\nabla I = \left[\frac{\delta I}{\delta x}, \frac{\delta I}{\delta y}\right]$$

Magnitude And Orientation



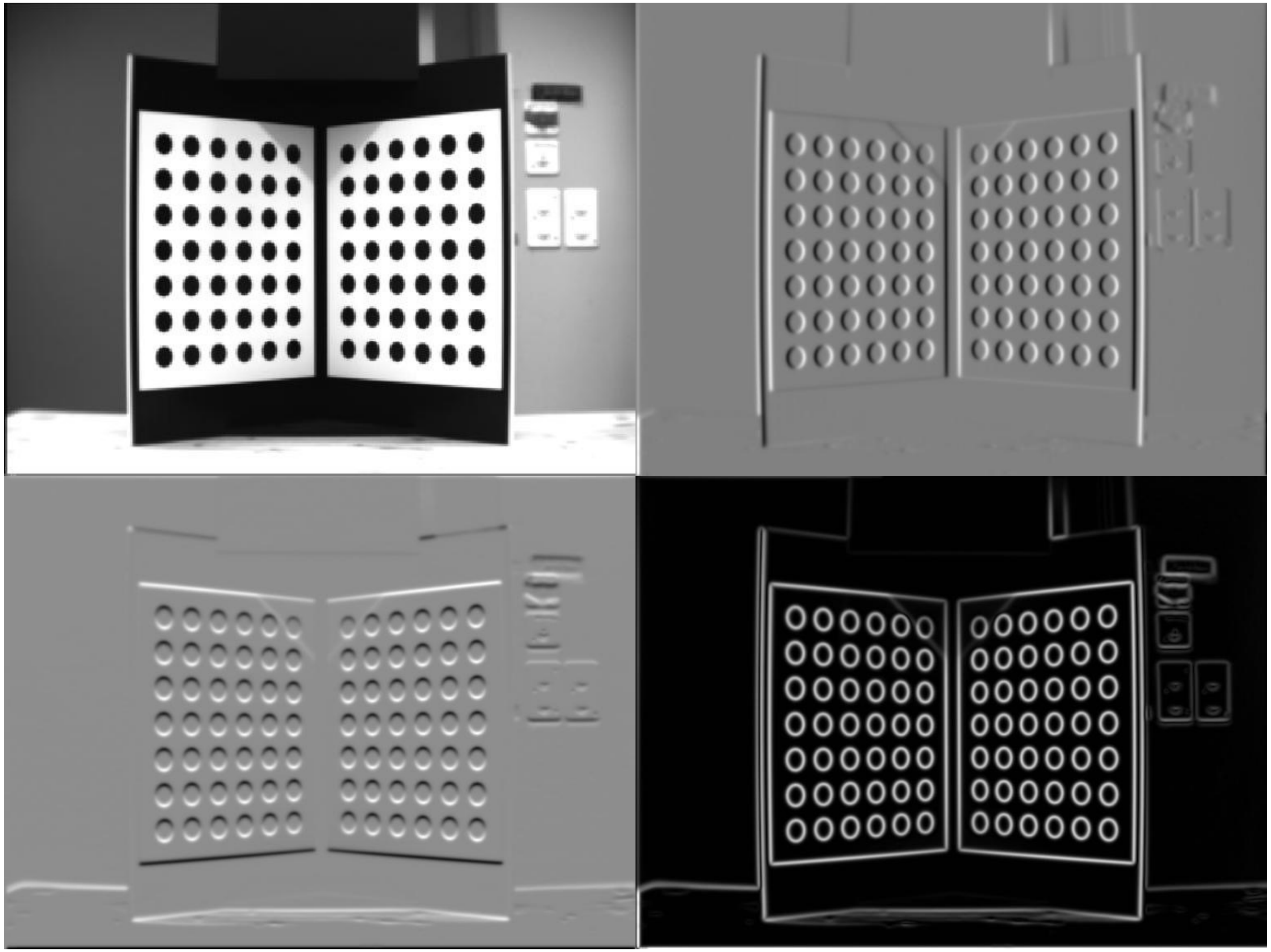
$$\text{Measure of contrast: } G = \sqrt{\frac{\partial I}{\partial x}^2 + \frac{\partial I}{\partial y}^2}$$

$$\text{Edge orientation: } \theta = \arctan\left(\frac{\partial I}{\partial y}, \frac{\partial I}{\partial x}\right)$$

Gradient Images

I

$\frac{\partial I}{\partial x}$



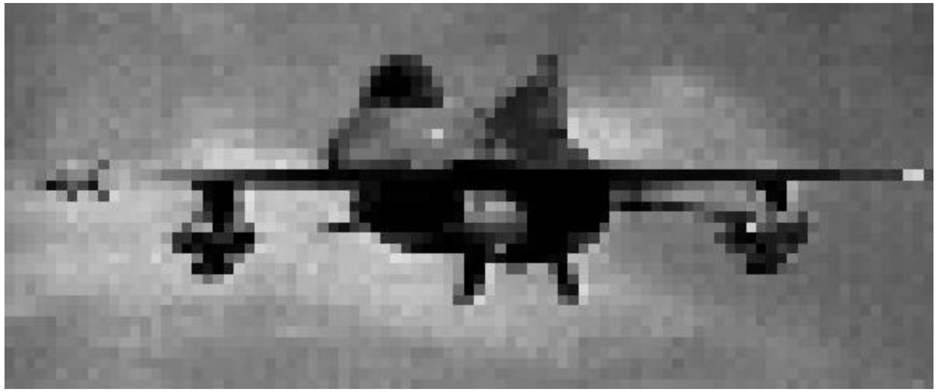
$\frac{\partial I}{\partial y}$

$\sqrt{\frac{\partial I}{\partial x}^2 + \frac{\partial I}{\partial y}^2}$

The gradient magnitude is unaffected by orientation ….

Real Images





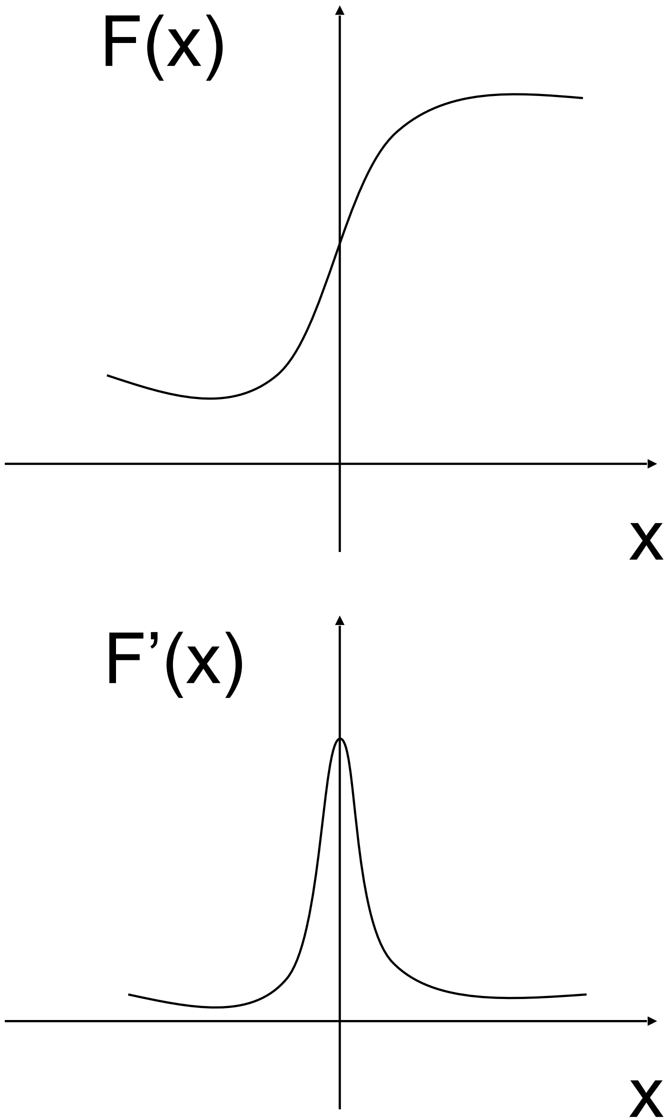

… but not directly usable in most real-world images.

Edge Operators

- Difference Operators
- Convolution Operators
- Trained Detectors
- Deep Nets

EPFL

Gradient Methods

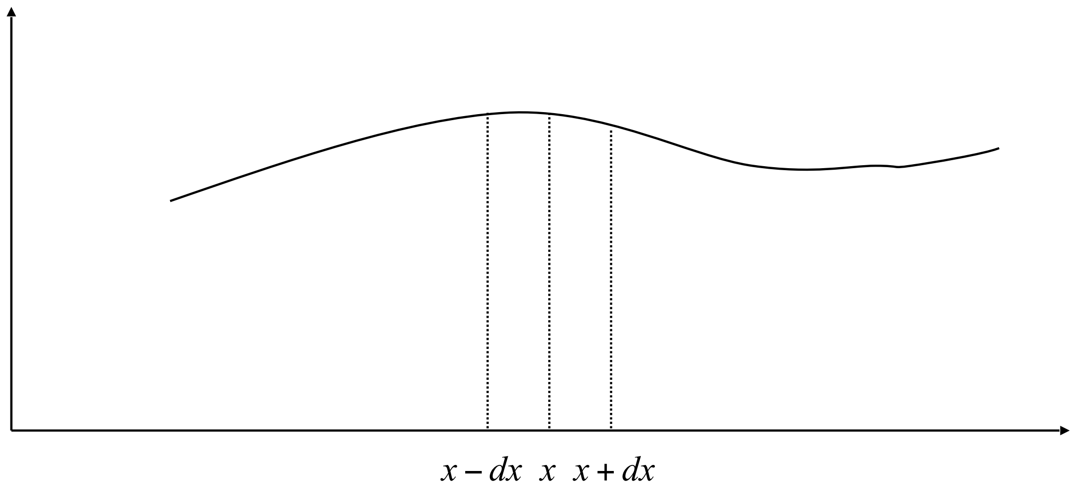


Edge = Sharp variation

Large first derivative

1D Finite Differences

In one dimension:

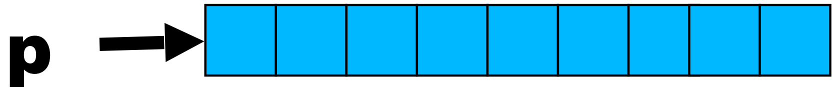


$$\frac{df}{dx} \approx \frac{f(x+dx)-f(x)}{dx} \approx \frac{f(x+dx)-f(x-dx)}{2dx}$$

$$\frac{d^2 f}{dx^2} \approx \frac{f(x+dx)-2f(x)+f(x-dx)}{dx^2}$$

Coding 1D Finite Differences

Line stored as an array:

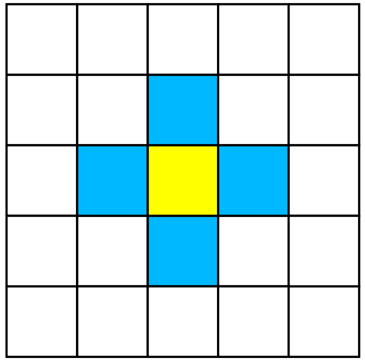


- for i in range(n-1):
  q[i]=(p[i+1]-p[i])

- for i in range(1,n-1):
  q[i]=(p[i+1]-p[i-1])/2

- q=(p[2:]-p[:-2])/2

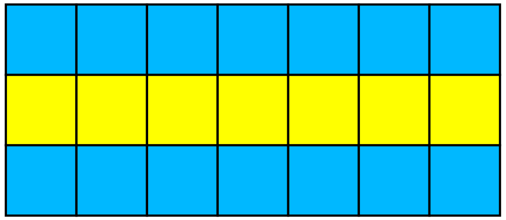
2D Finite Differences



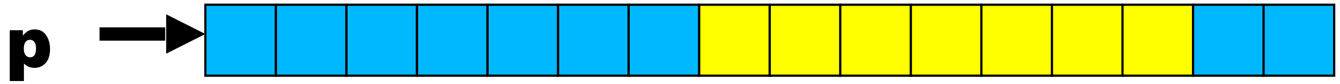
$$\frac{\partial f}{\partial x} \approx \frac{f(x+dx,y)-f(x,y)}{dx} \approx \frac{f(x+dx,y)-f(x-dx,y)}{2dx}$$

$$\frac{\partial f}{\partial y} \approx \frac{f(x,y+dy)-f(x,y)}{dy} \approx \frac{f(x,y+dy)-f(x,y-dy)}{2dy}$$

Coding 2D Finite Differences



Python



C

Image stored as a 2D array:

- dx = p[1:,:]-p[:-1,:]
  dy = p[:,1:]-p[:,:-1]

- dx = (p[2:,:]-p[:-2,:])/2
  dy = (p[:,2:]-p[:,:-2])/2

Image stored in raster format:

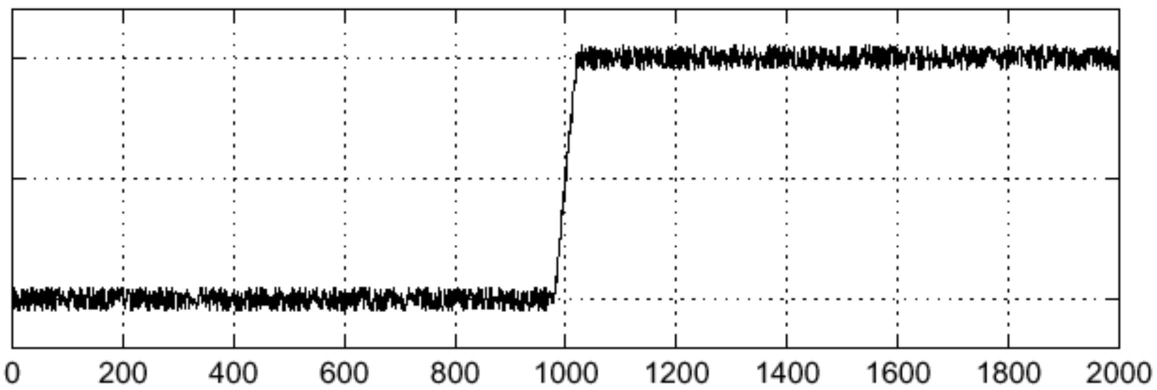
```
{
  int i;
  for(i=0;i<xdim;i++){
    dx[i] = p[i+1]     -p[i];
    dy[i] = p[i+xdim]-p[i];
  }
}
```

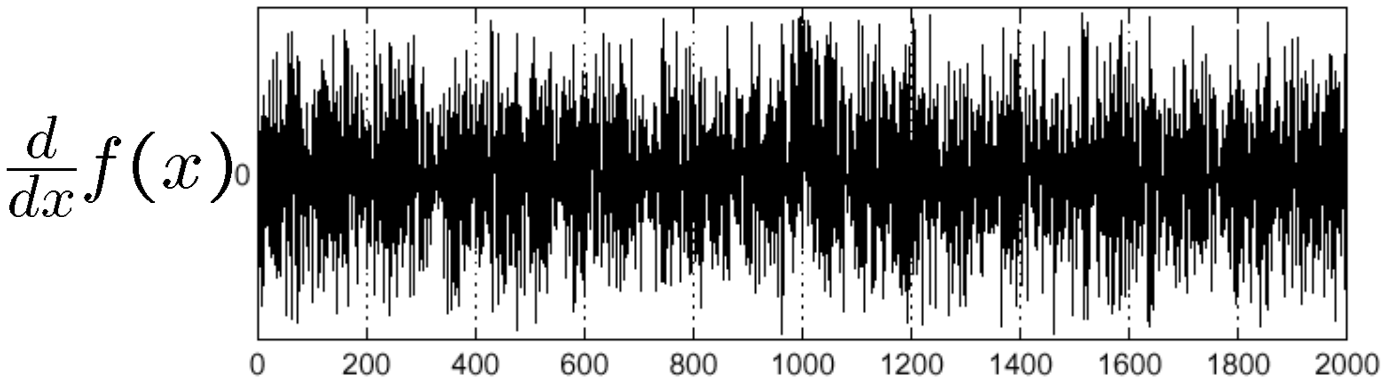
- Only 1D array accesses
- No multiplications

—> Can be exploited to increase speed.

Noise in 1D

Consider a single row or column of the image:



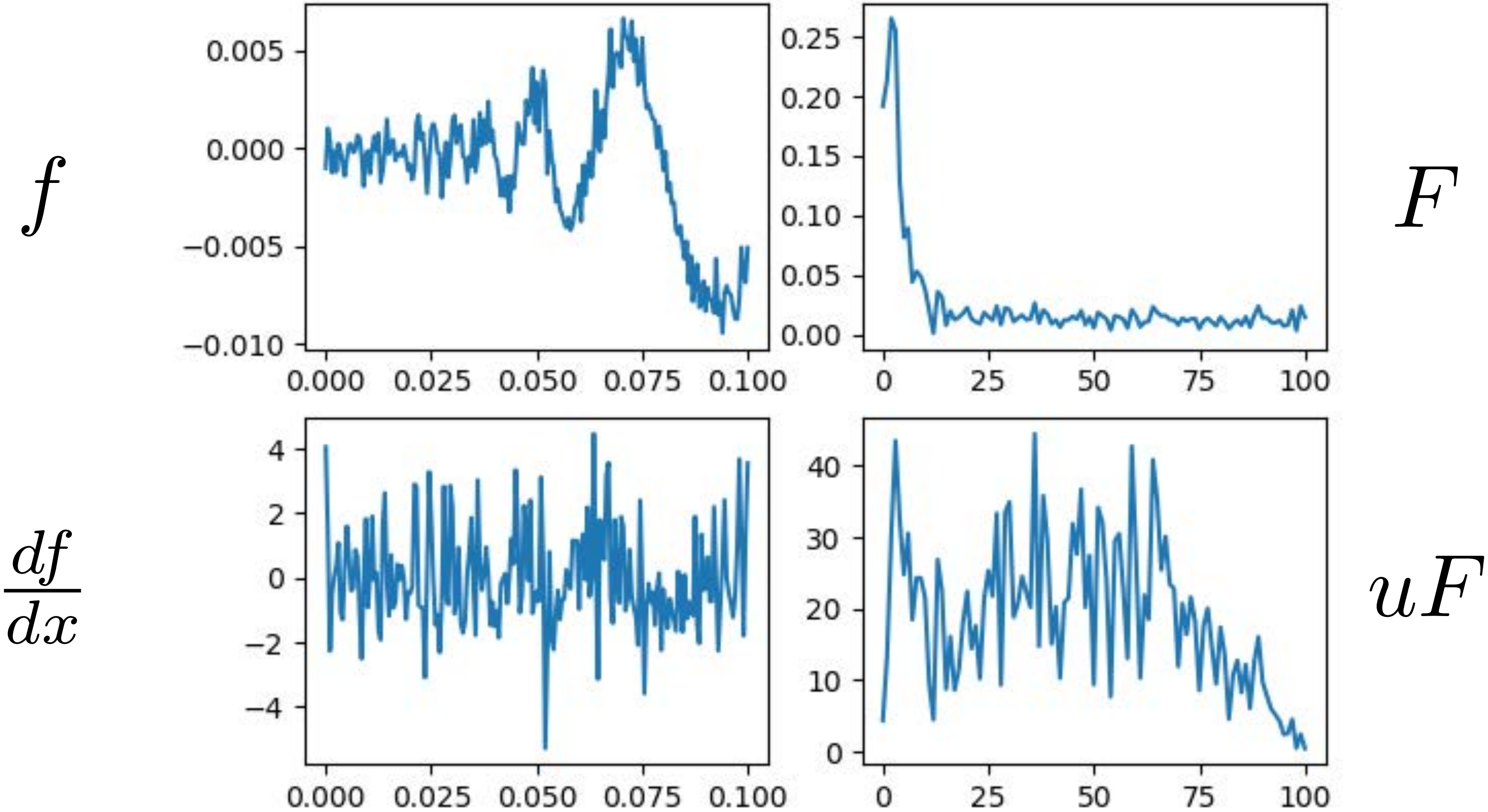


Fourier Interpretation

Function	Fourier Transform
$\frac{df}{dx}(x)$	$uF(u)$
$\frac{\delta f}{\delta x}(x,y)$	$uF(u,v)$
$\frac{\delta f}{\delta y}(x,y)$	$vF(u,v)$

→ Differentiating emphasizes high frequencies and therefore noise!

$f(x) = x^2 \sin(1/x)$



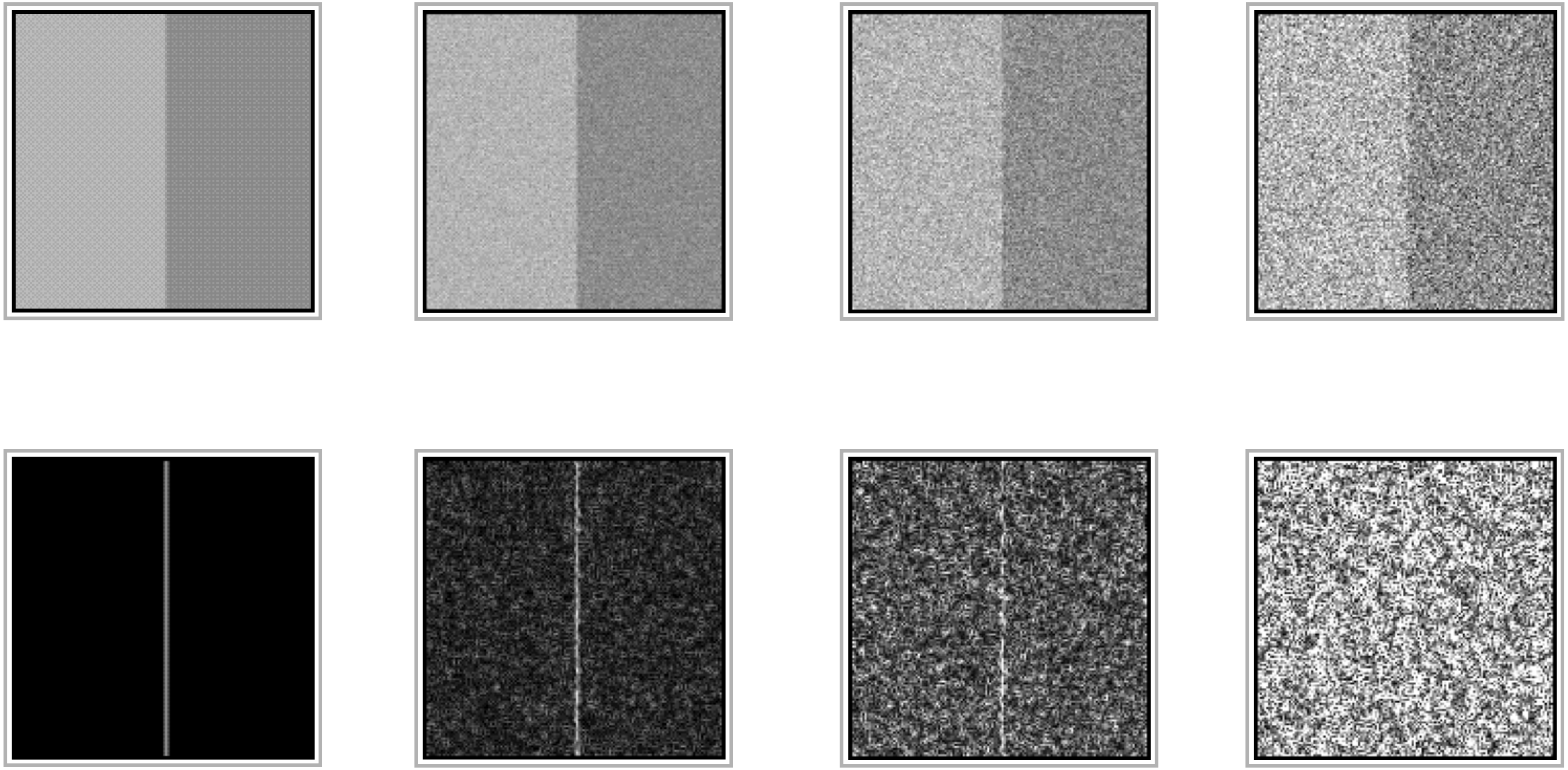
Original function + Noise

Fourier transform

Noise in 2D

Ideal step edge

Step edge + noise



Increasing noise level →

As the amount of noise increases, the derivatives stop being meaningful.

Removing Noise

Problem:

- High frequencies and differentiation do not mix well.

Solution:

- Suppress high frequencies by
  - using the Discrete Fourier Transform.

Discrete Fourier Transform

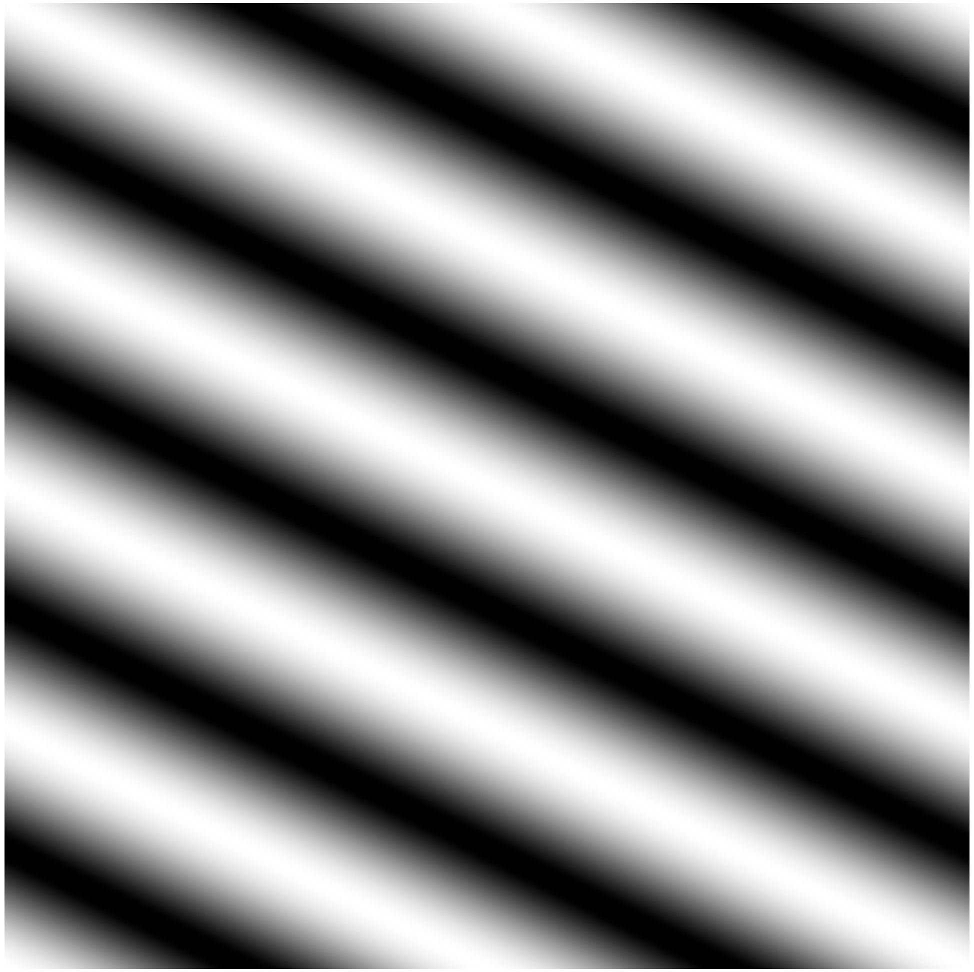
$$F(\mu,\nu) = \frac{1}{\sqrt{M*N}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-2i\pi(\mu x/M + \nu y/N)}$$

$$f(x,y) = \frac{1}{\sqrt{M*N}} \sum_{\mu=0}^{M-1} \sum_{\nu=0}^{N-1} F(\mu,\nu) e^{+2i\pi(\mu x/M + \nu y/N)}$$

The DFT is the discrete equivalent of the 2D Fourier transform:

- The 2D function f is written as a sum of sinusoids.
- The DFT of f convolved with g is the product of their DFTs.

Fourier Basis Element



Real part of

$$e^{+2i\pi(ux+vy)}$$

where

- $\sqrt{u^2 + v^2}$ represents the frequency,
- $\text{atan}(v, u)$ represents the orientation.

Fourier Basis Element

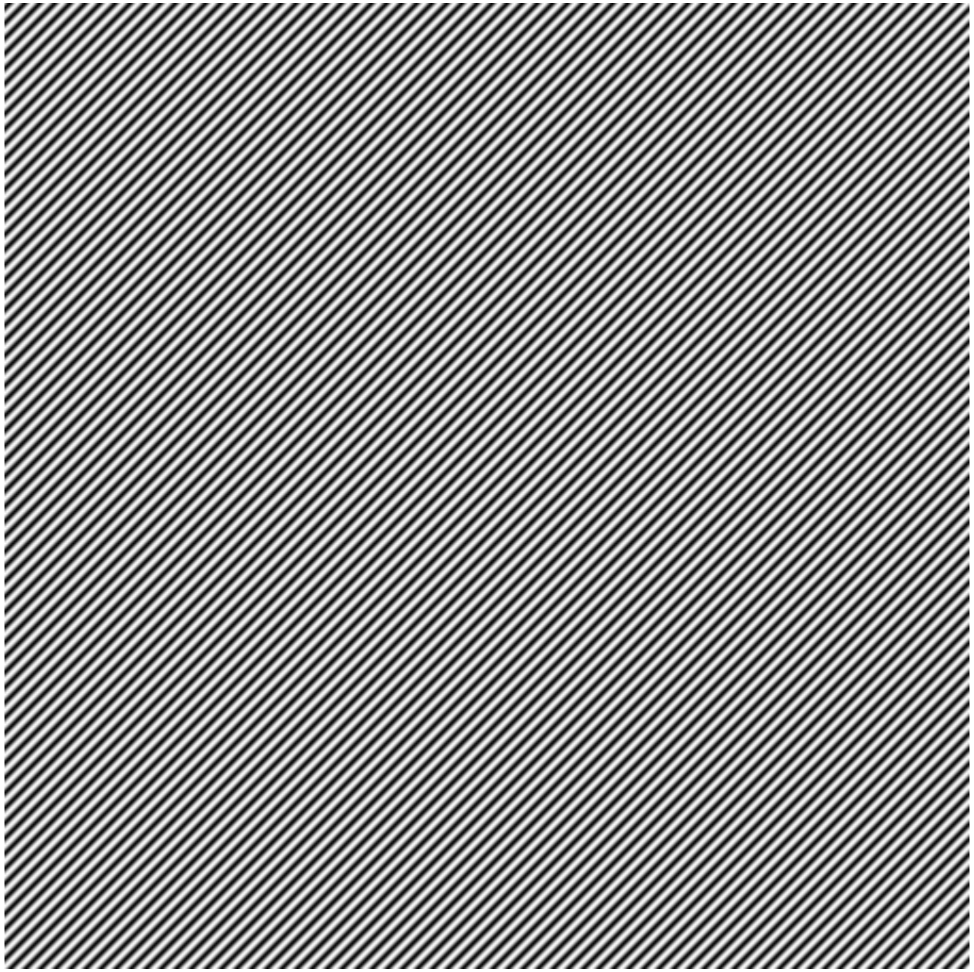
Real part of

$$e^{+2i\pi(ux+vy)}$$

where

- $\sqrt{u^2 + v^2}$ is larger than before.

Fourier Basis Element



Real part of

$$e^{+2i\pi(ux+vy)}$$

where

- $\sqrt{u^2 + v^2}$ is larger still.

Truncated Inverse DFT

$$F(\mu, \nu) = \frac{1}{\sqrt{M * N}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-2i\pi(\mu x/M + \nu y/N)}$$

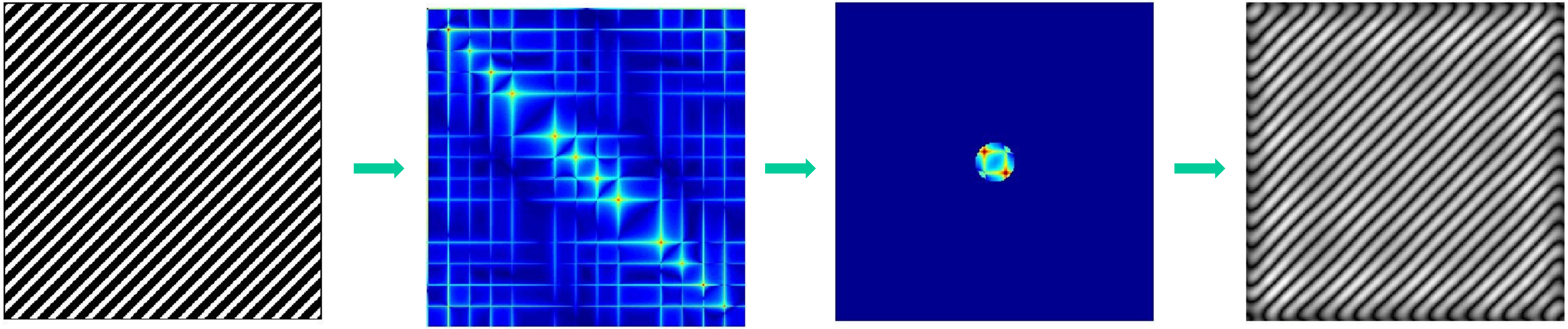
$$f(x, y) = \frac{1}{\sqrt{M * N}} \sum_{\mu=0}^{M-1} \sum_{\nu=0}^{N-1} F(\mu, \nu) e^{+2i\pi(\mu x/M + \nu y/N)}$$

$$f(x, y) = \frac{1}{\sqrt{M * N}} \sum_{\mu^2+\nu^2<T} F(\mu, \nu) e^{+2i\pi(\mu x/M + \nu y/N)}$$

T is a hand-specified threshold.

- The sinusoids corresponding to $\mu^2 + \nu^2 \geq T$ depict high frequencies.
- Removing them amounts to removing high-frequencies.

Smoothing by Truncating the IDFT



Rotated stripes:

- Dominant diagonal structures
- Discretization produces additional harmonics

—> Removing higher frequencies and reconstructing yields a smoothed image.

Removing Noise

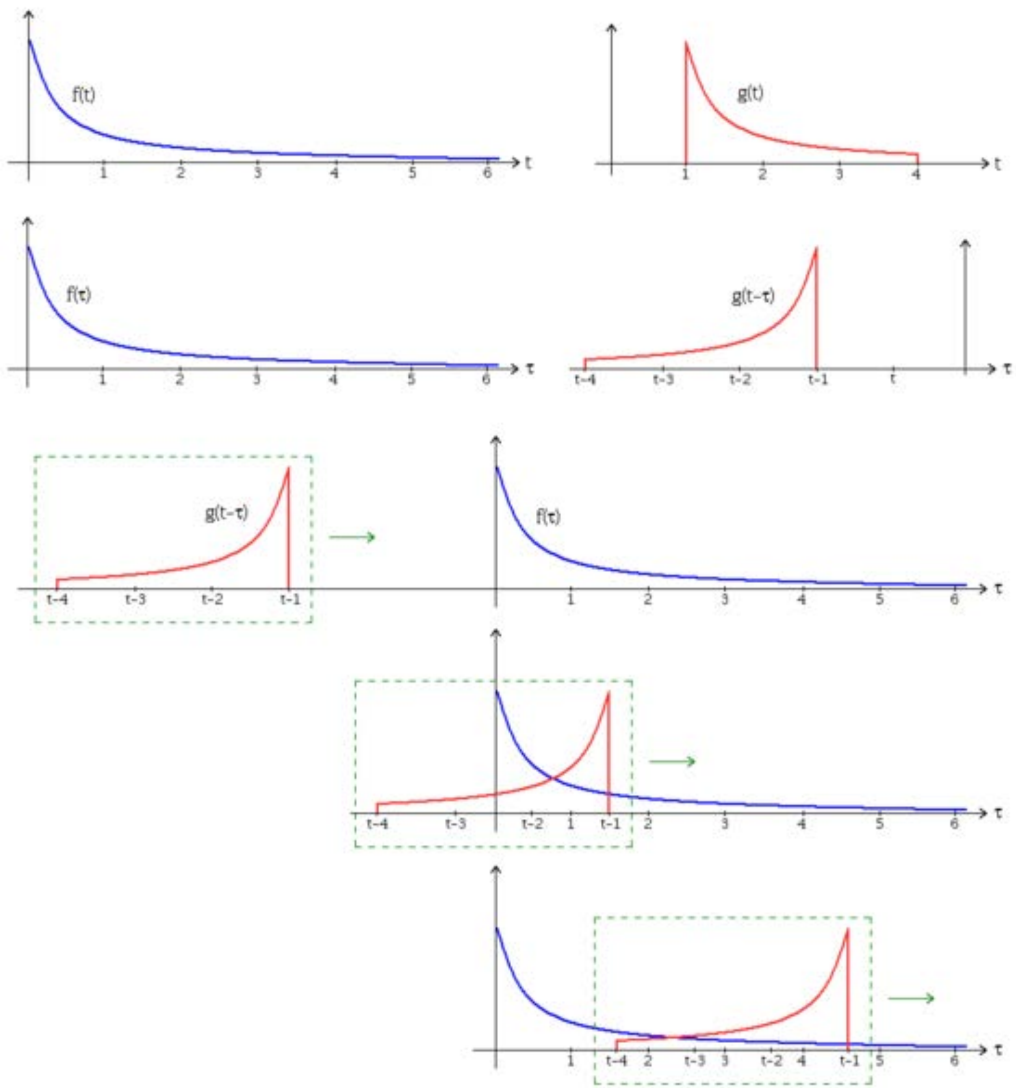
Problem:

- High frequencies and differentiation do not mix well.

Solution:

- Suppress high frequencies by
  - using the Discrete Fourier Transform,
  - convolving with a low-pass filter.

1D Convolution



$$g * f(t) = \int_{\tau} g(t-\tau) f(\tau) d\tau$$

EPFL

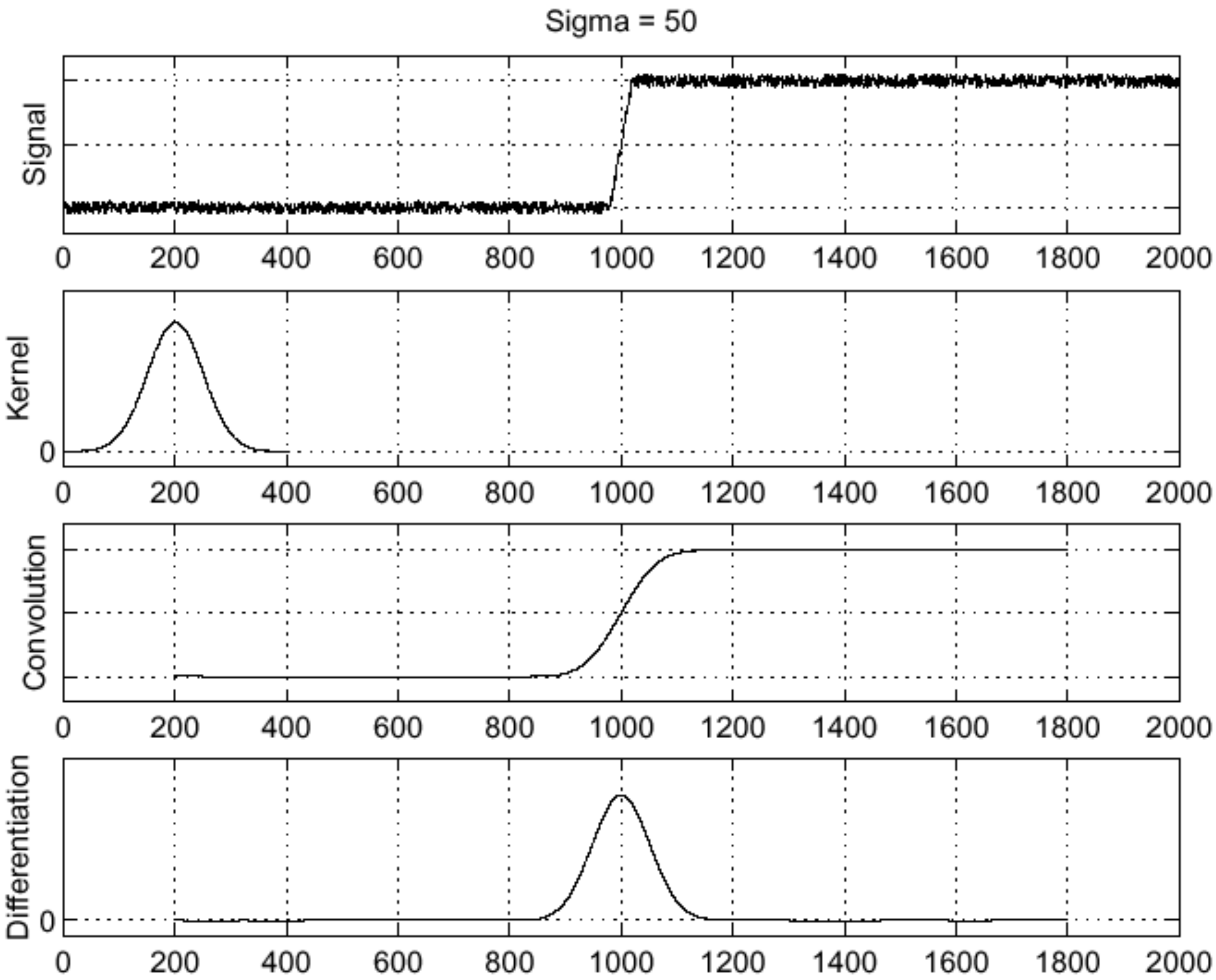
Smooth Before Differentiating

f

g

$g * f$

$\frac{\partial}{\partial x}(g * f)$

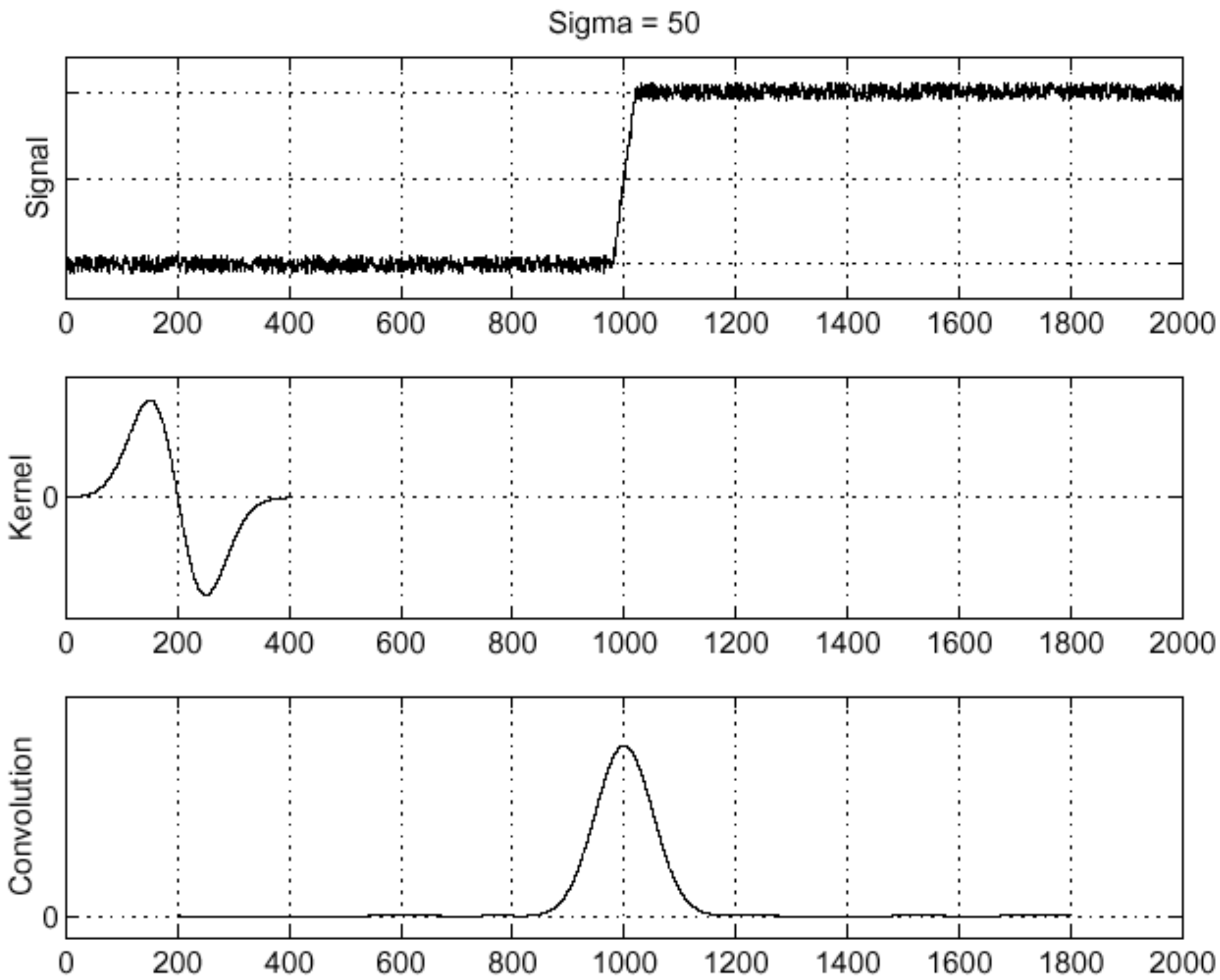


Simultaneously Smooth and Differentiate

f

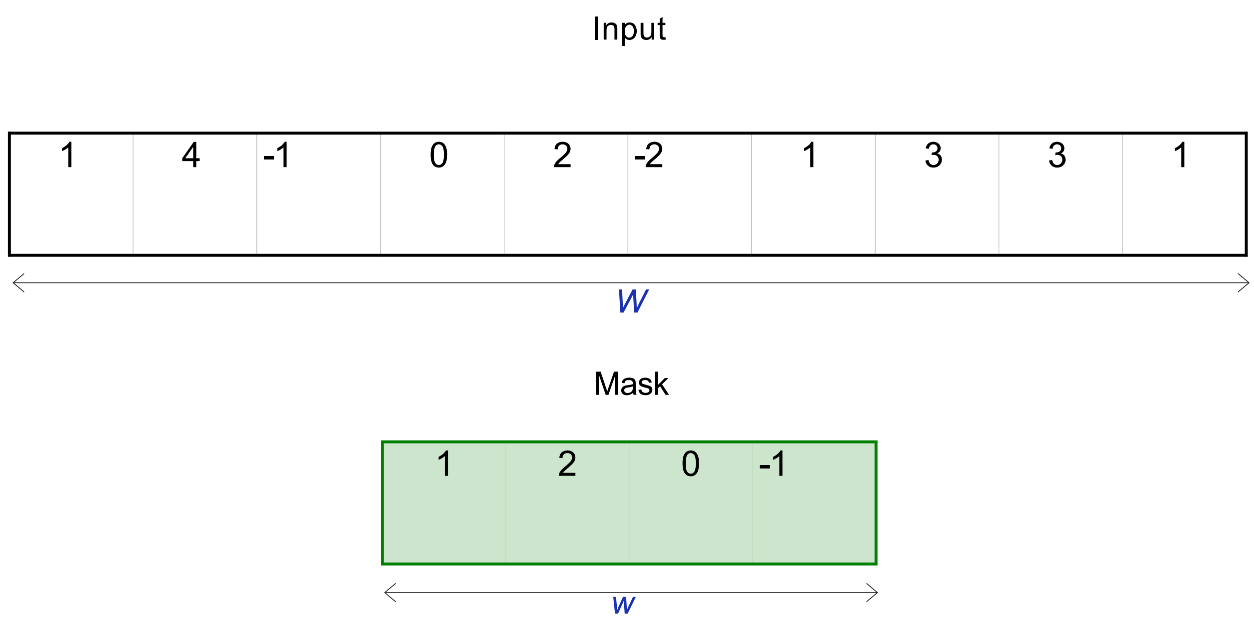
$\frac{\partial g}{\partial x}$

$$\frac{\partial}{\partial x}(g * f) = \frac{\partial g}{\partial x} * f$$

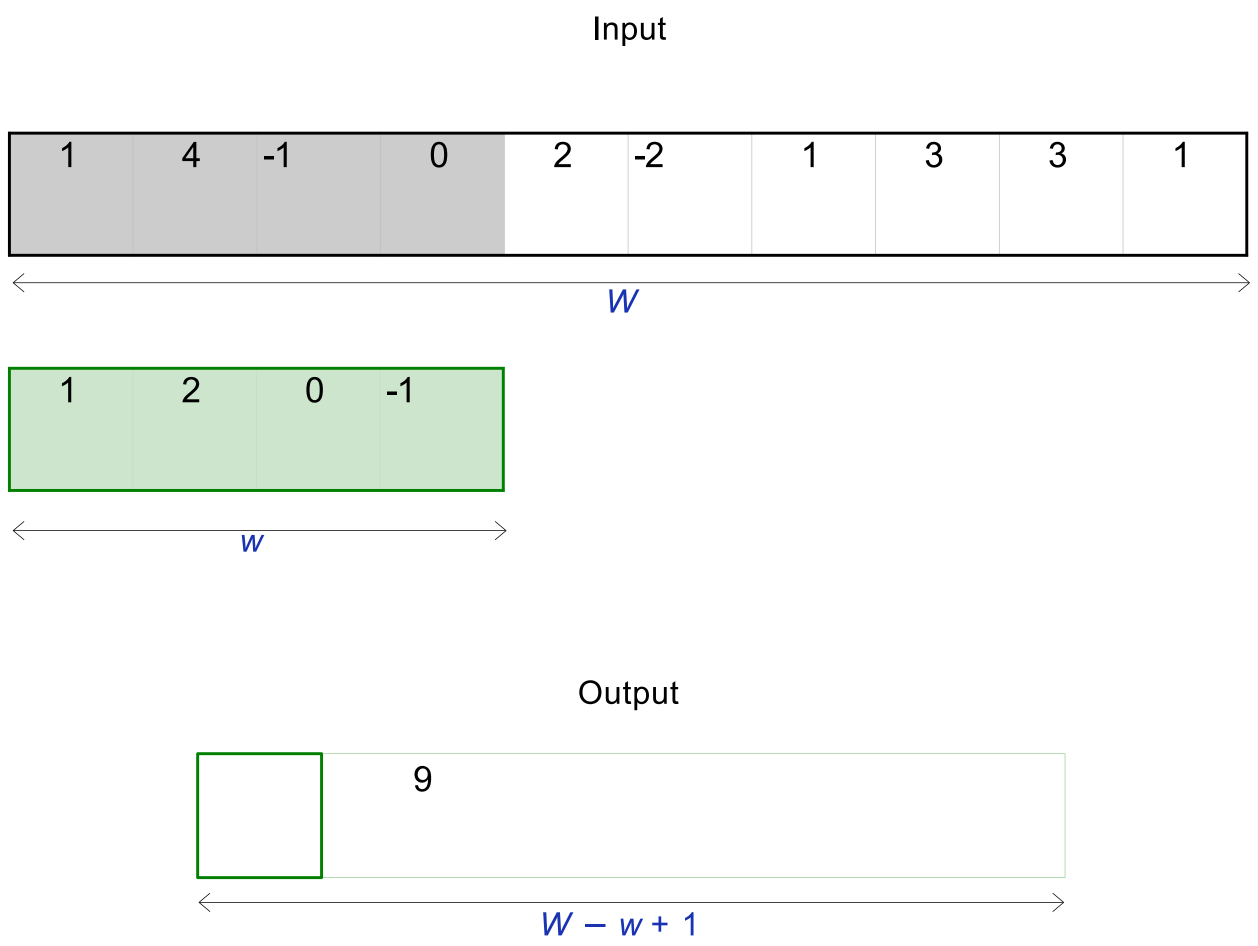


--> Faster because dg/dx can be precomputed.

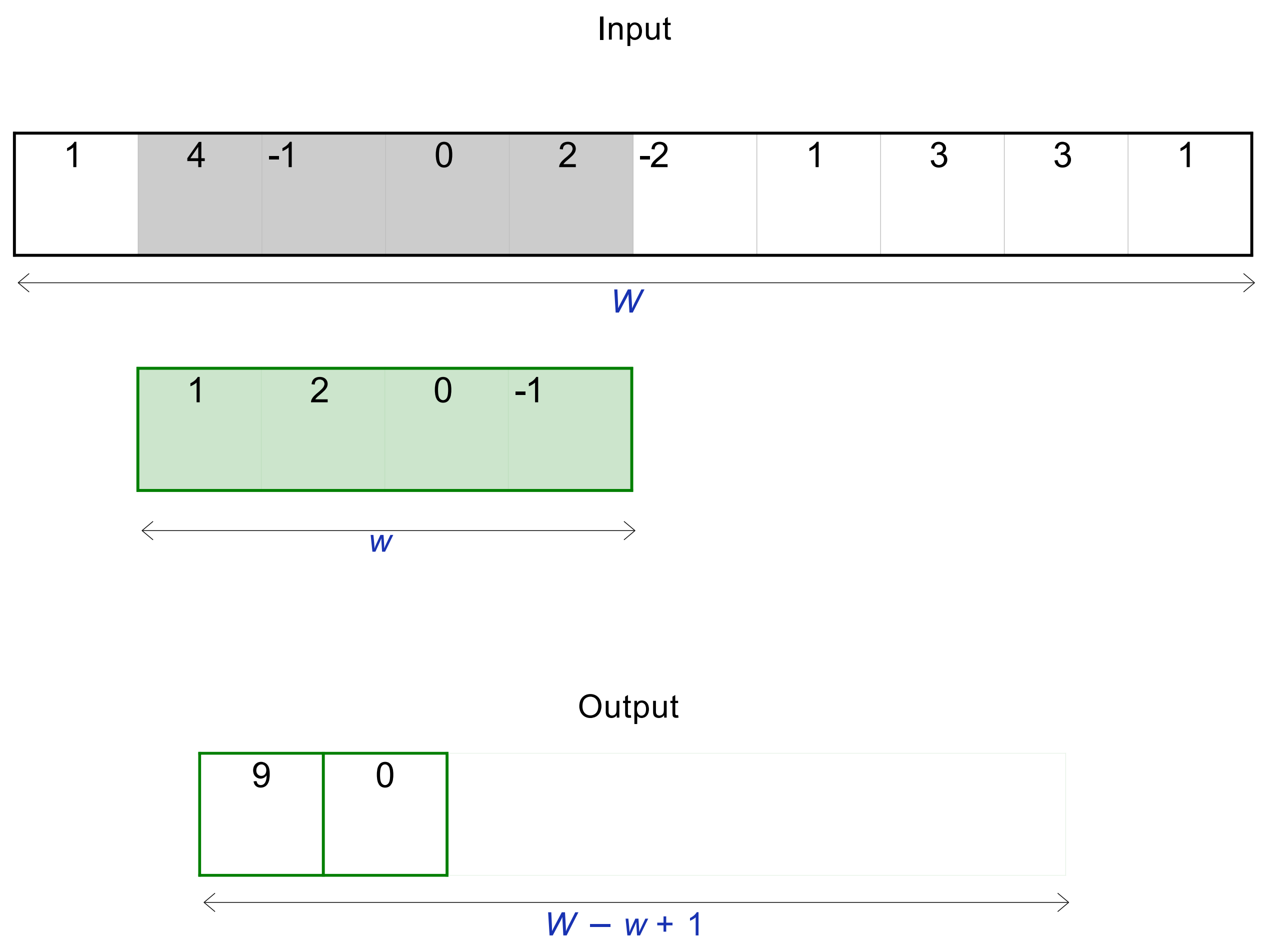
Discrete 1D Convolution



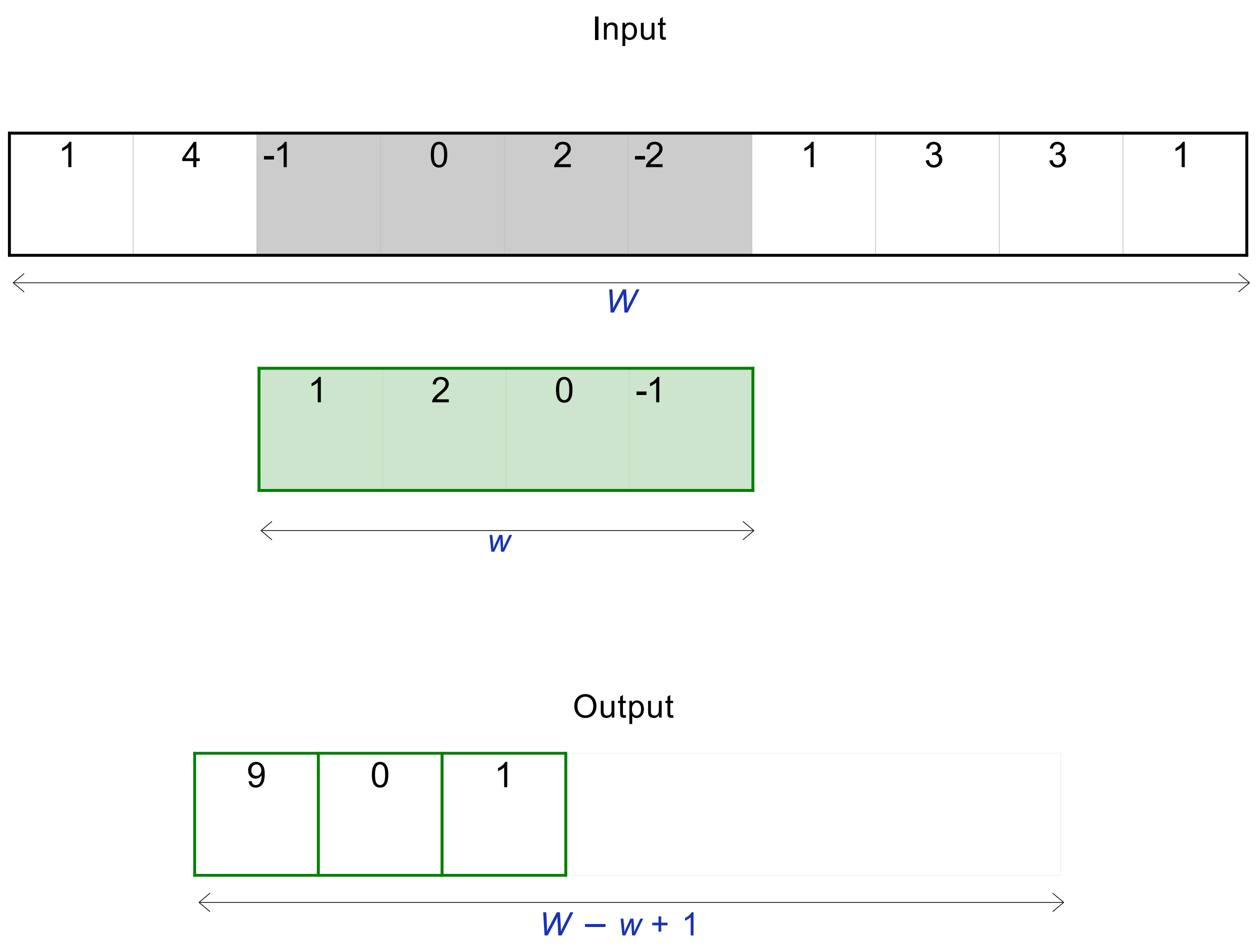
Discrete 1D Convolution



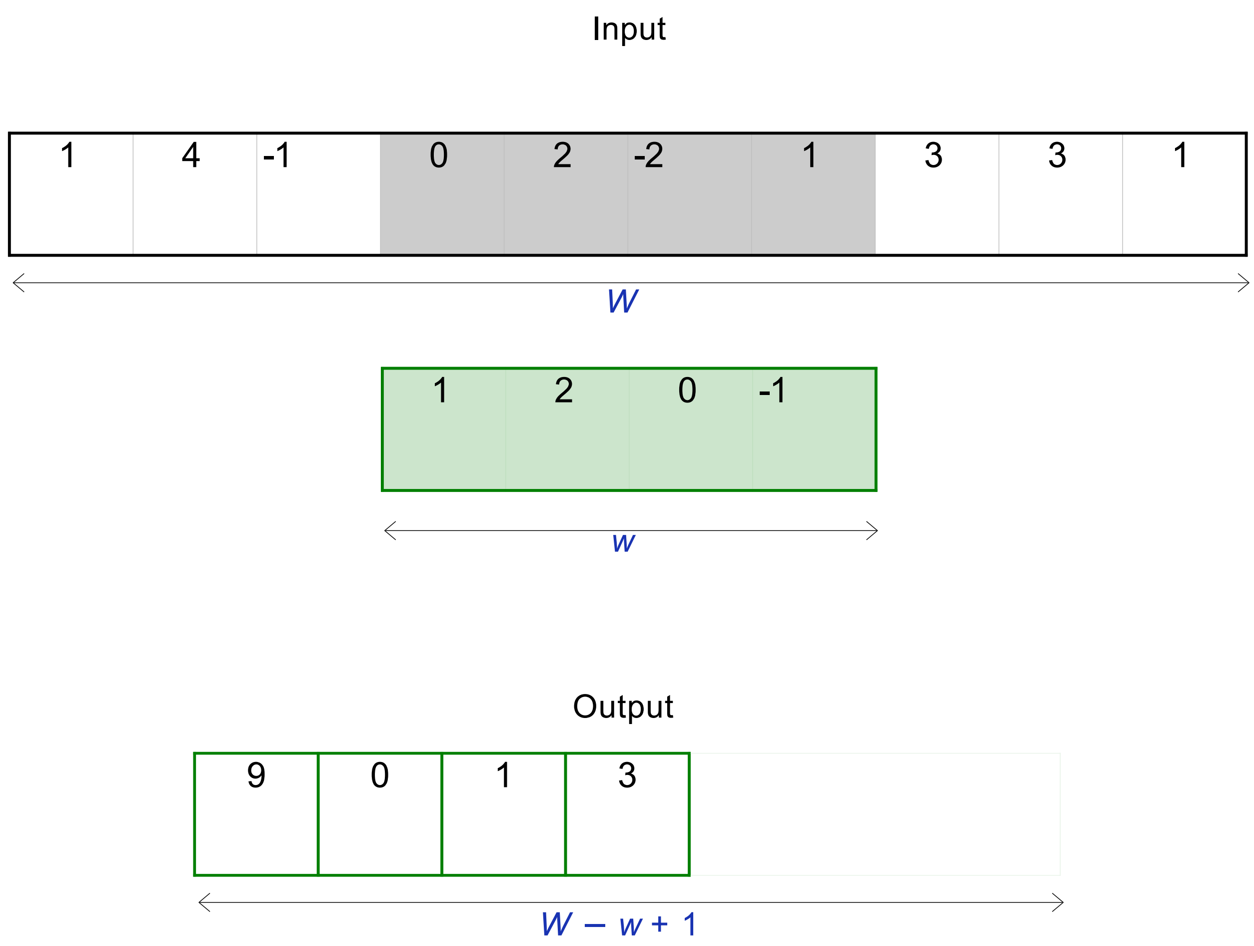
Discrete 1D Convolution



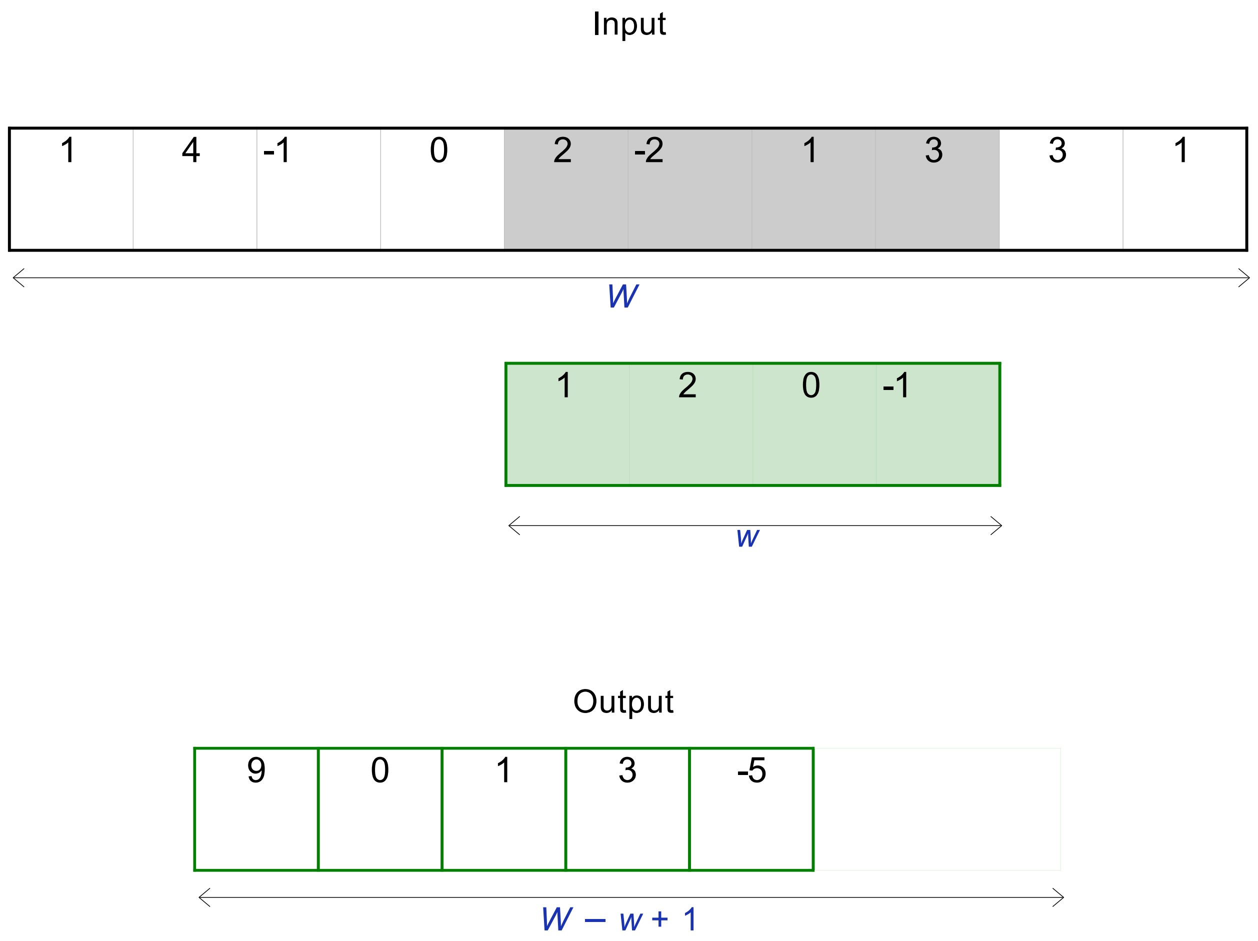
Discrete 1D Convolution



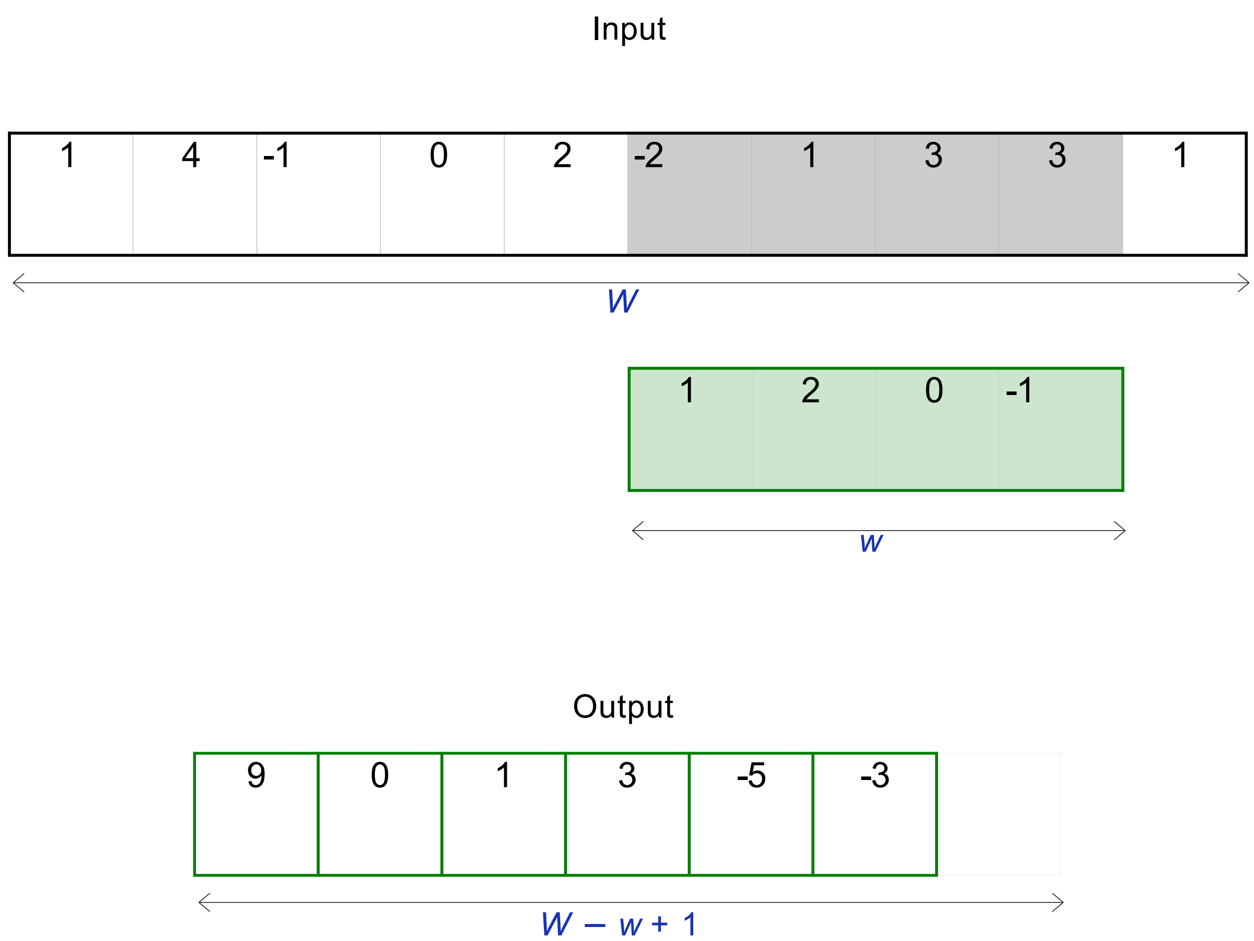
Discrete 1D Convolution



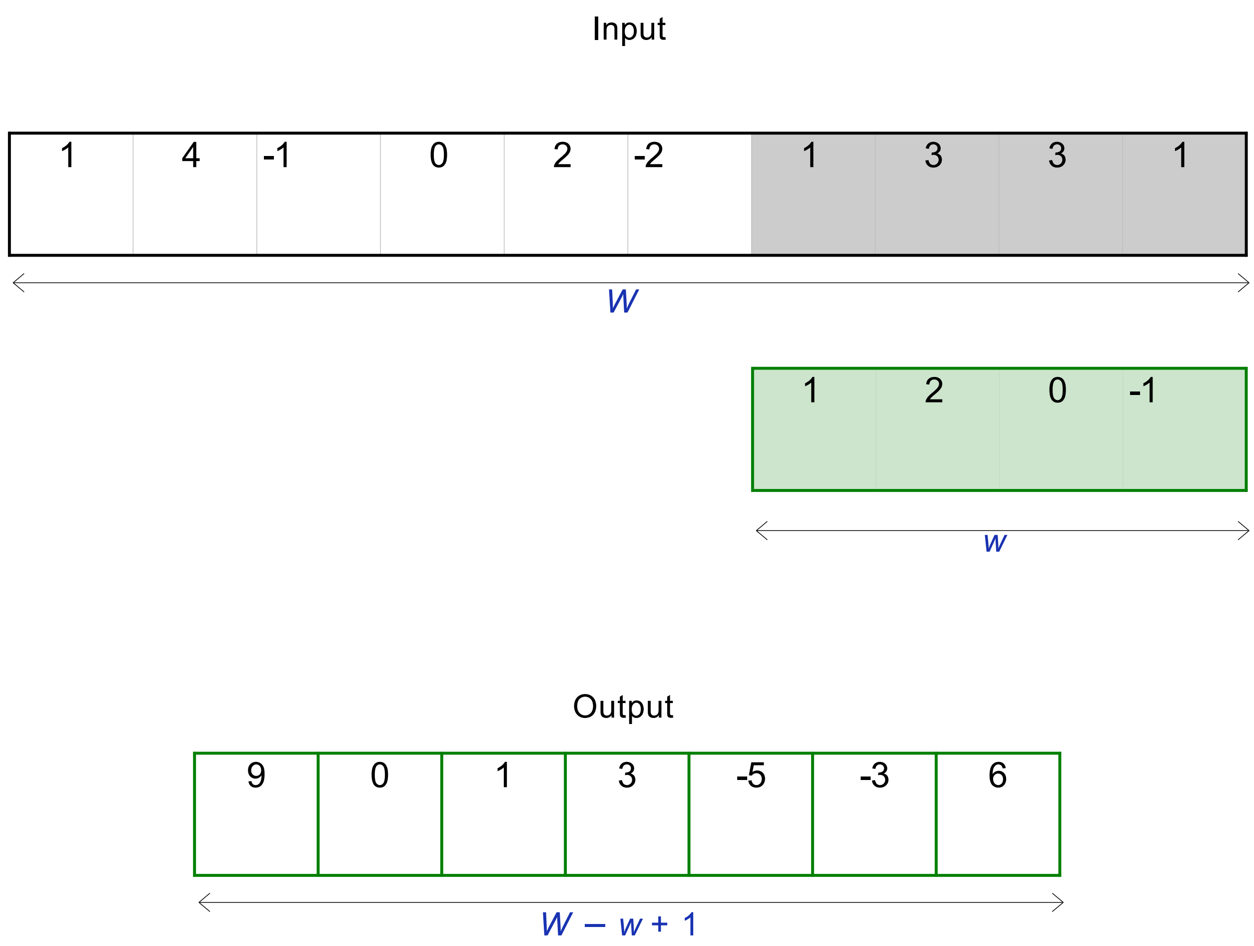
1D Convolution



Discrete 1D Convolution



Discrete 1D Convolution



Discrete 1D Convolution

Input

f

1	4	-1	0	2	-2	1	3	3	1

W

Mask

m

1	2	0	-1

w

Output

m*f

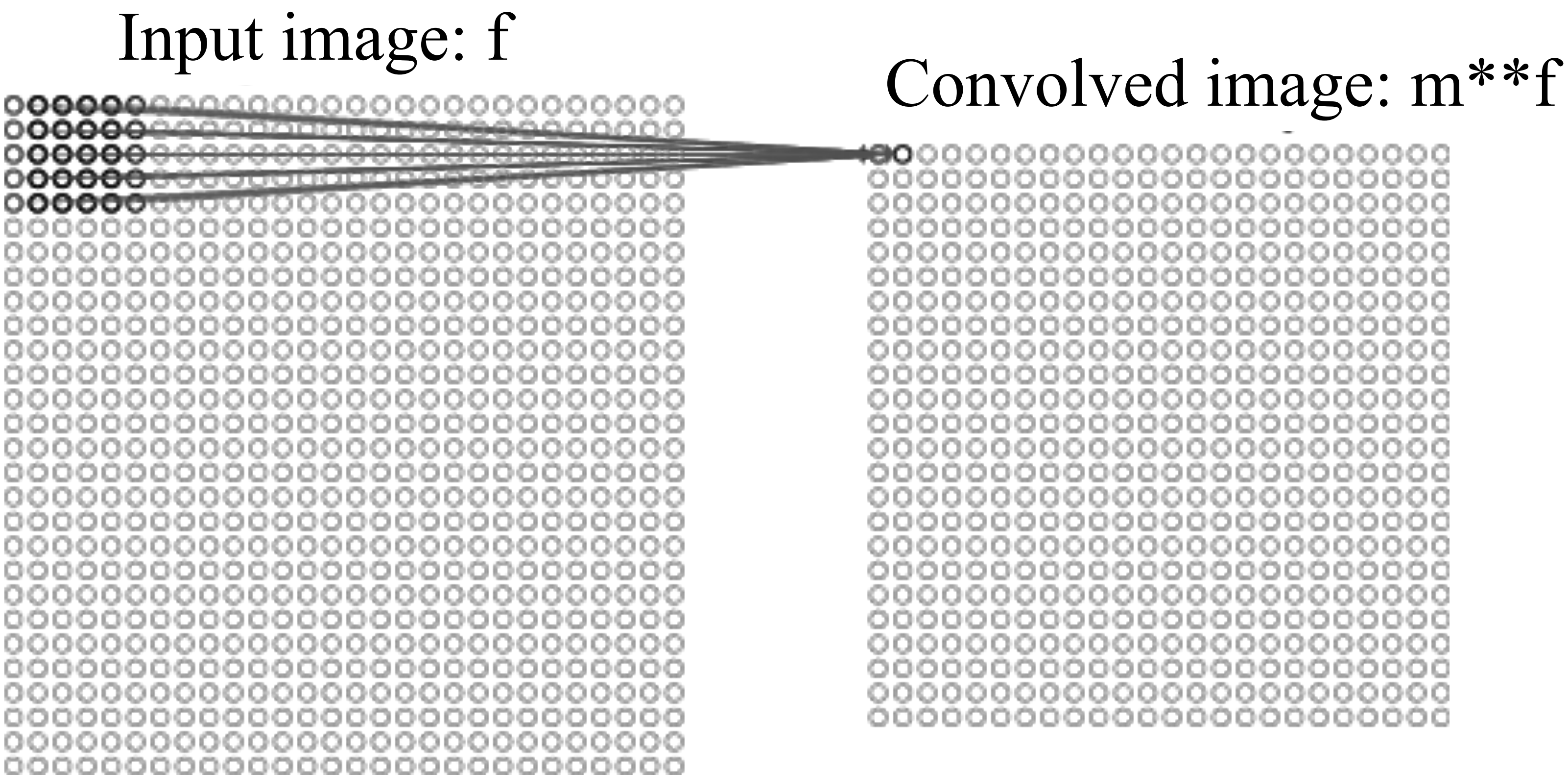
9	0	1	3	-5	-3	6

$W - w + 1$

$$m * f(x) = \sum_{i=0}^{w} m(i) f(x - i)$$

EPFL

Discrete 2D Convolution



Convolution mask m, also known as a *kernel*.

$$\begin{bmatrix} m_{11} & \dots & m_{1w} \\ \dots & \dots & \dots \\ m_{w1} & \dots & m_{ww} \end{bmatrix}$$

$$m**f(x,y) = \sum_{i=0}^{w} \sum_{j=0}^{w} m(i,j) f(x-i, y-j)$$

Convolution In C

Naive C implementation:

```
static double g[][]={{-1.0,-2.0,-1.0},{0.0,0.0,0.0},{1.0,2.0,1.0}};
{
        for(i=i0;i<N;i++)
            for(j=j0;j<N;j++){
              q[i][j]=0;
              for(a=a0;a<W;a++)
                for(b=b0;b<W,b++)
                  q[i][j]+=g[a][b]*p[i-a][j-b];
    }
}
```

Computational complexity:

- Lots of memory access

→ Slow, but can be sped up when the filters are separable.

N^2W^2 multiplications for a $N \times N$ image and a $W \times W$ mask.

Differentiation As Convolution

$$[-1,1] \rightarrow \frac{\partial f}{\partial x} \approx \frac{f(x+dx,y)-f(x,y)}{dx}$$

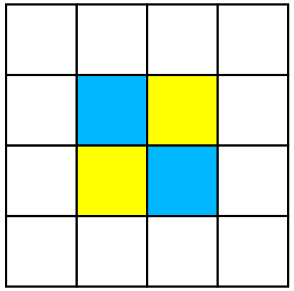
$$[-0.5,0,0.5] \rightarrow \frac{\partial f}{\partial x} \approx \frac{f(x+dx,y)-f(x-dx,y)}{2dx}$$

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix} \rightarrow \frac{\partial f}{\partial y} \approx \frac{f(x,y+dy)-f(x,y)}{dy}$$

$$\begin{bmatrix} -0.5 \\ 0 \\ 0.5 \end{bmatrix} \rightarrow \frac{\partial f}{\partial y} \approx \frac{f(x,y+dy)-f(x,y-dy)}{2dy}$$

→ Use wider masks to add some smoothing

2x2 Masks:
Roberts Operator

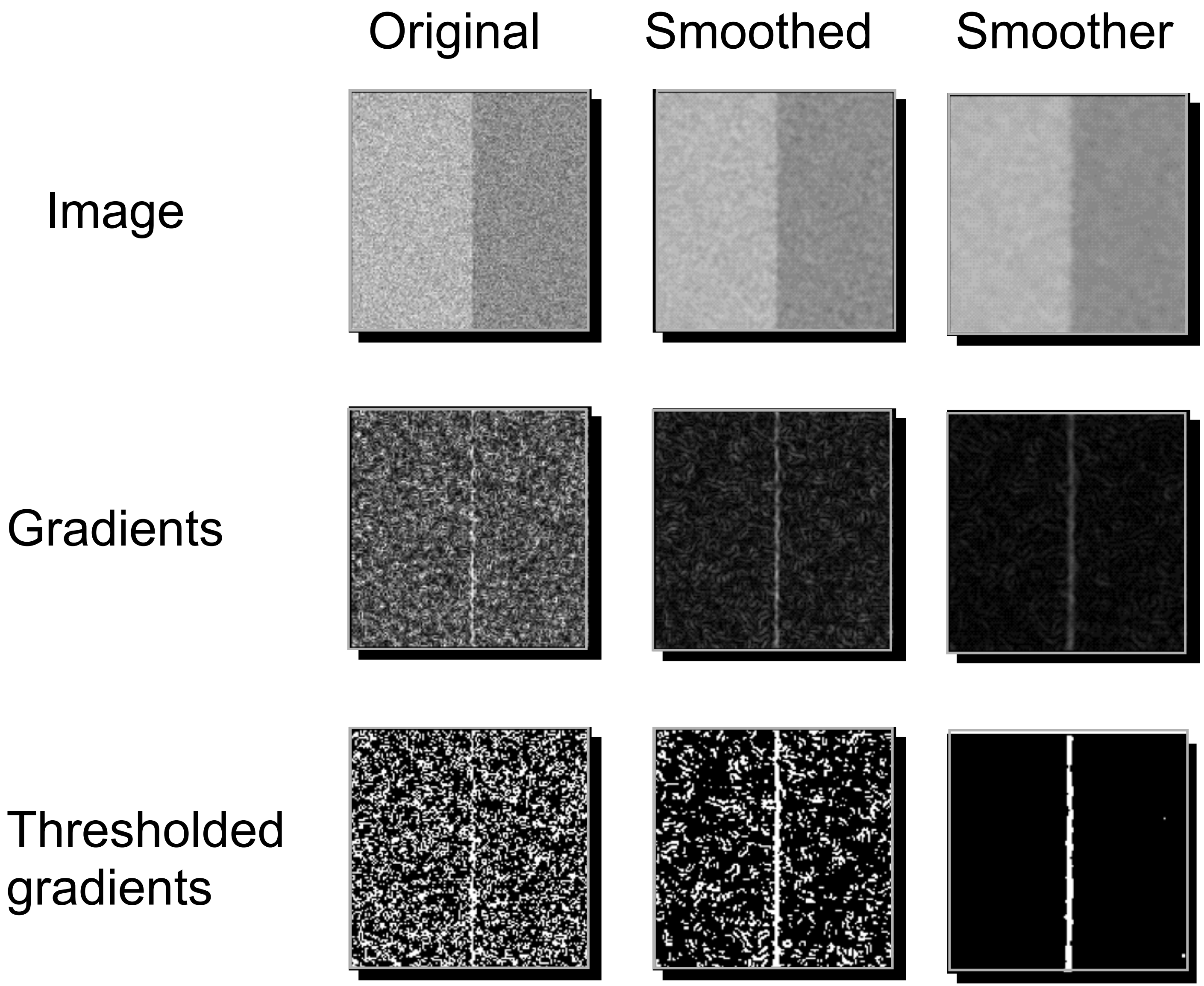


$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \text{and} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

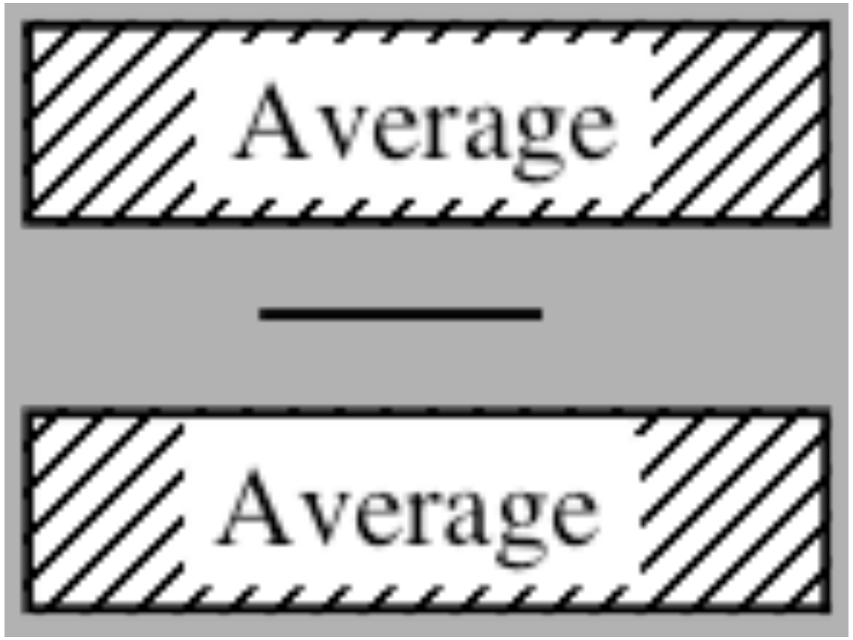
$$G = \sqrt{\left[I(x+1,y+1) - I(x,y) \right]^2 + \left[I(x+1,y-1) - I(x-1,y+1) \right]^2}$$

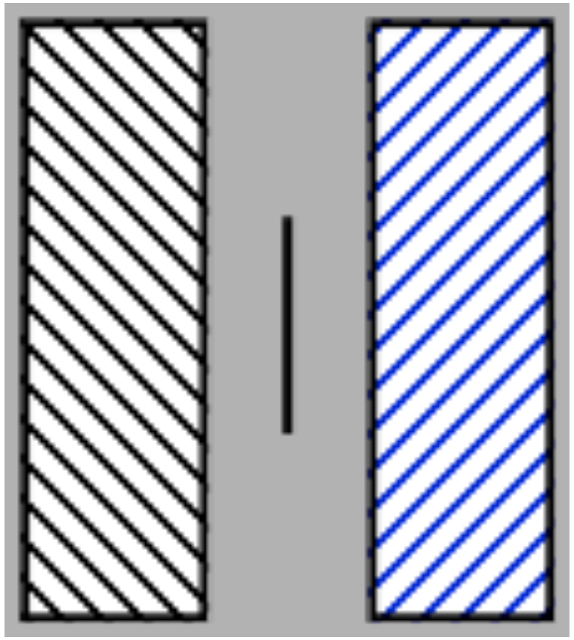
→ Equivalent to fitting plane to patch.

Benefits of Smoothing



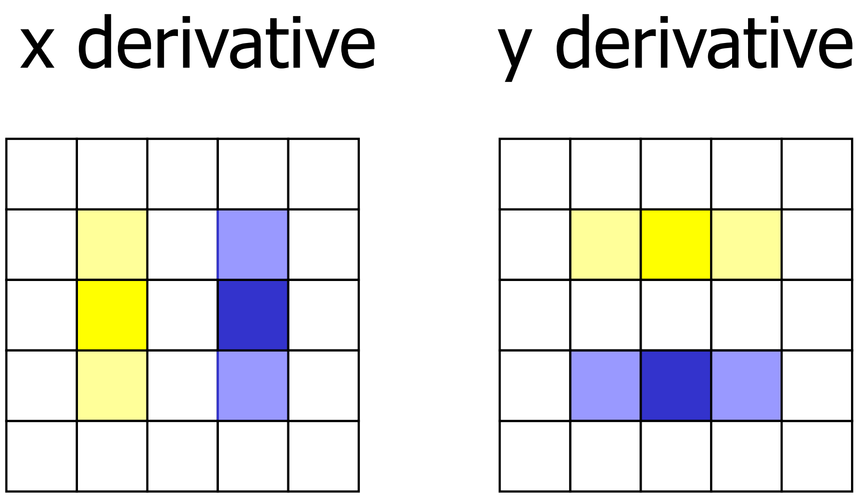
Smoothing and Differentiating





Compute the difference of averages on either side of the central pixel.

3X3 Masks



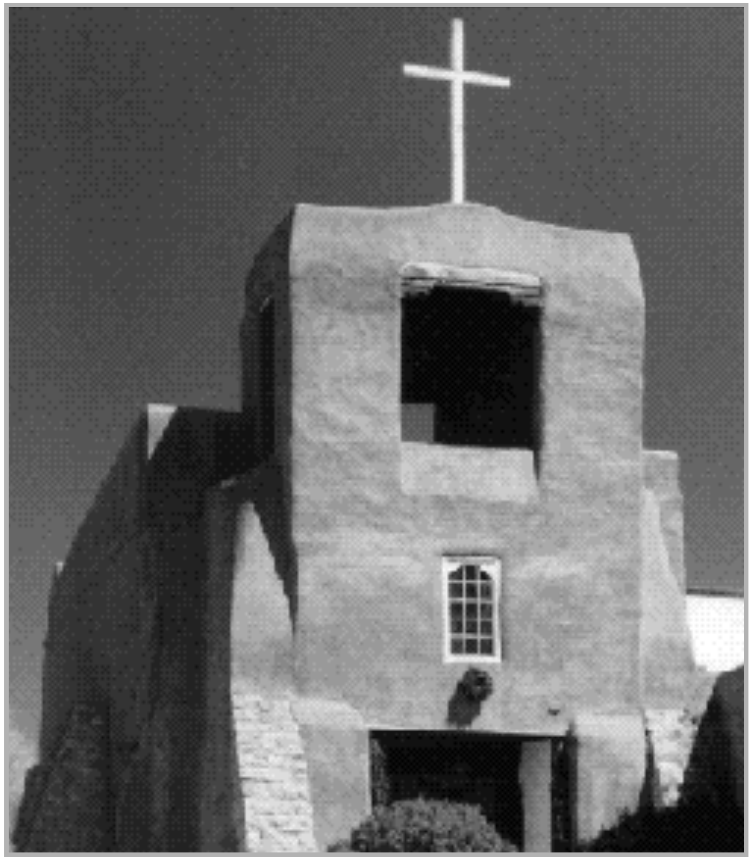
$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \text{and} \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Prewitt operator

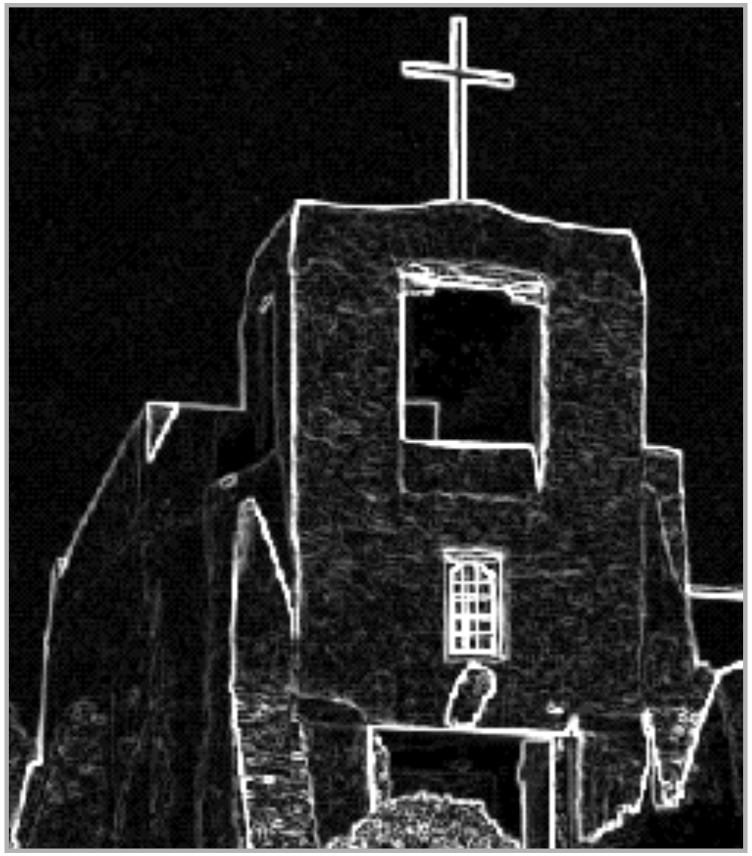
$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{and} \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Sobel operator

Prewitt Example

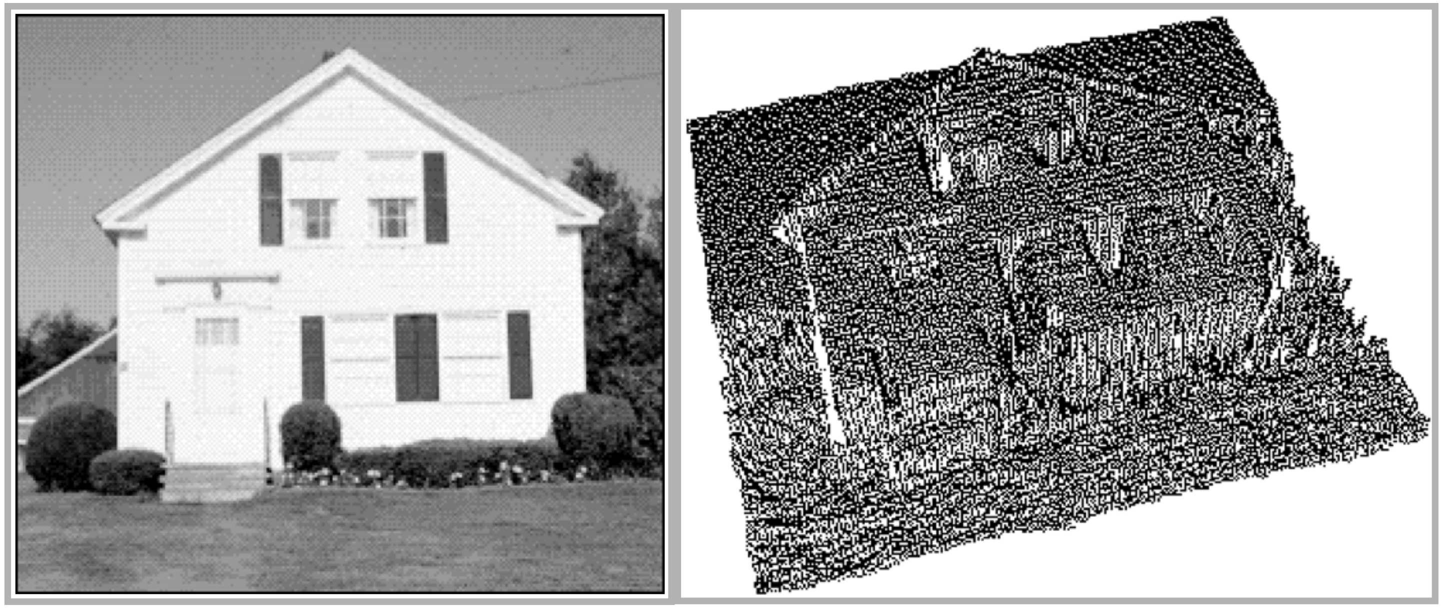


Santa Fe Mission

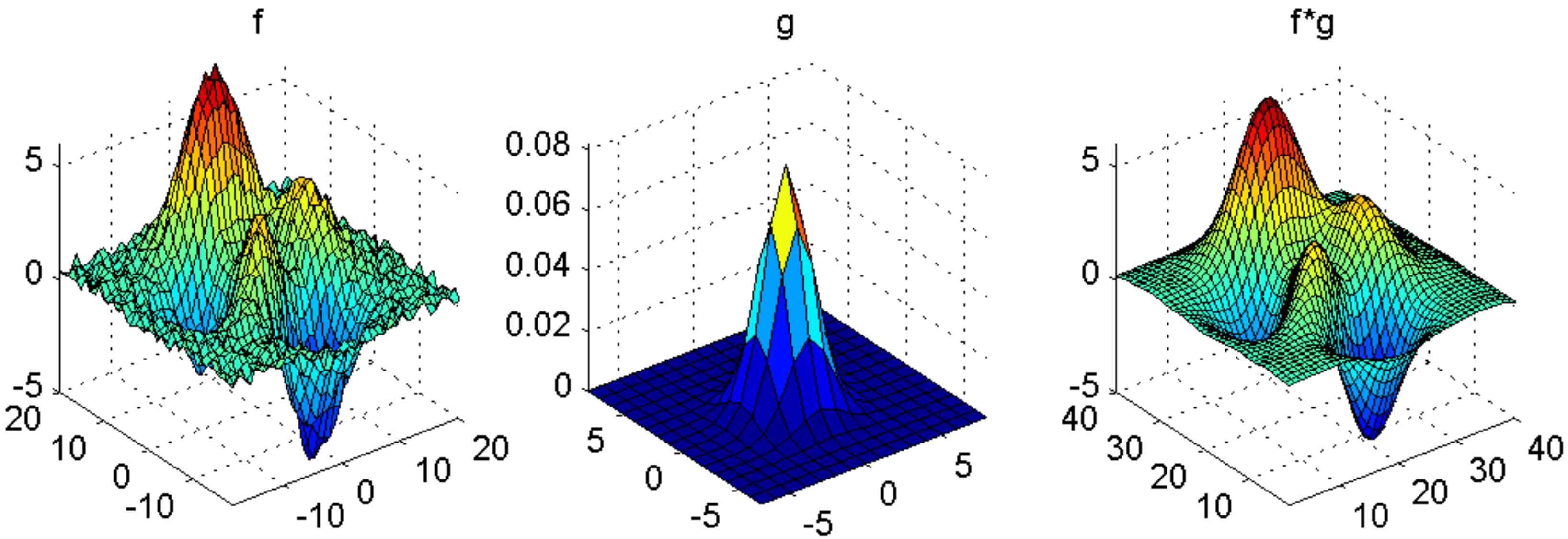


Gradient Image

Sobel Example

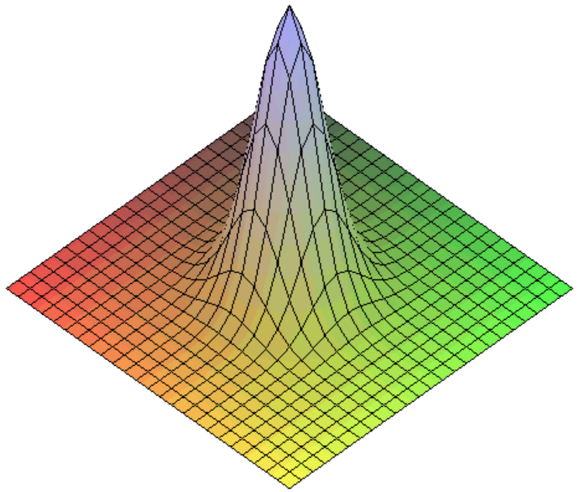


Gaussian Smoothing

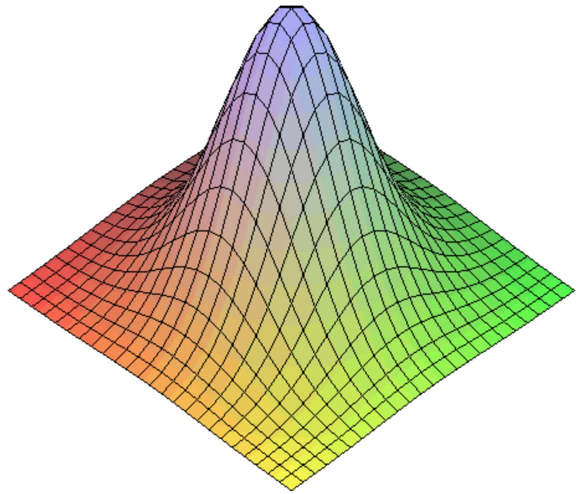


- More principled way to eliminate high frequency noise.
- Is fast because the kernel is
  - small,
  - separable.

Gaussian Functions



$\sigma = 1$

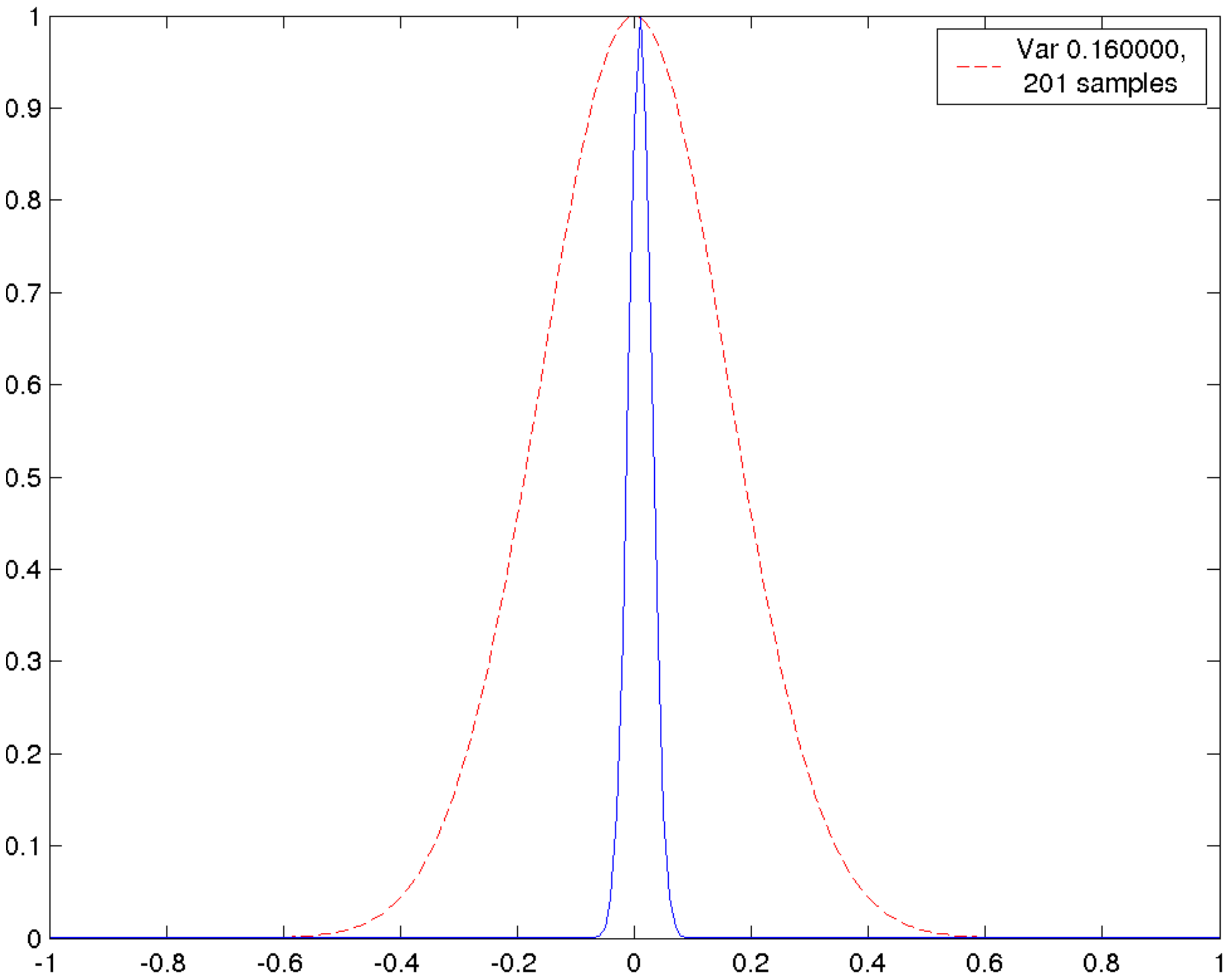


$\sigma = 2$

$$g_2(x, y) = \frac{1}{2\pi\sigma^2} \exp(-(x^2 + y^2)/2\sigma^2)$$

- The integral is always 1.0
- The larger σ, the broader the Gaussian is.
- As σ approaches 0, the Gaussian approximates a Dirac function.

DFT of a Gaussian



- The DFT of a Gaussian is a Gaussian.
- It has finite support.
- Its width is inversely proportional to that of the original Gaussian.

Gaussians as Low-Pass Filters

- The Fourier transform of a convolution is the product of their Fourier transforms: $\mathscr{F}(g * f) = \mathscr{F}(g)\mathscr{F}(f)$.
- If g is a Gaussian, so is $\mathscr{F}(g)$.
- Furthermore if g is broad, the support of $\mathscr{F}(g)$ is small.
- So is the support of $\mathscr{F}(g * f)$.
- There are no more high-frequencies in $g * f$.

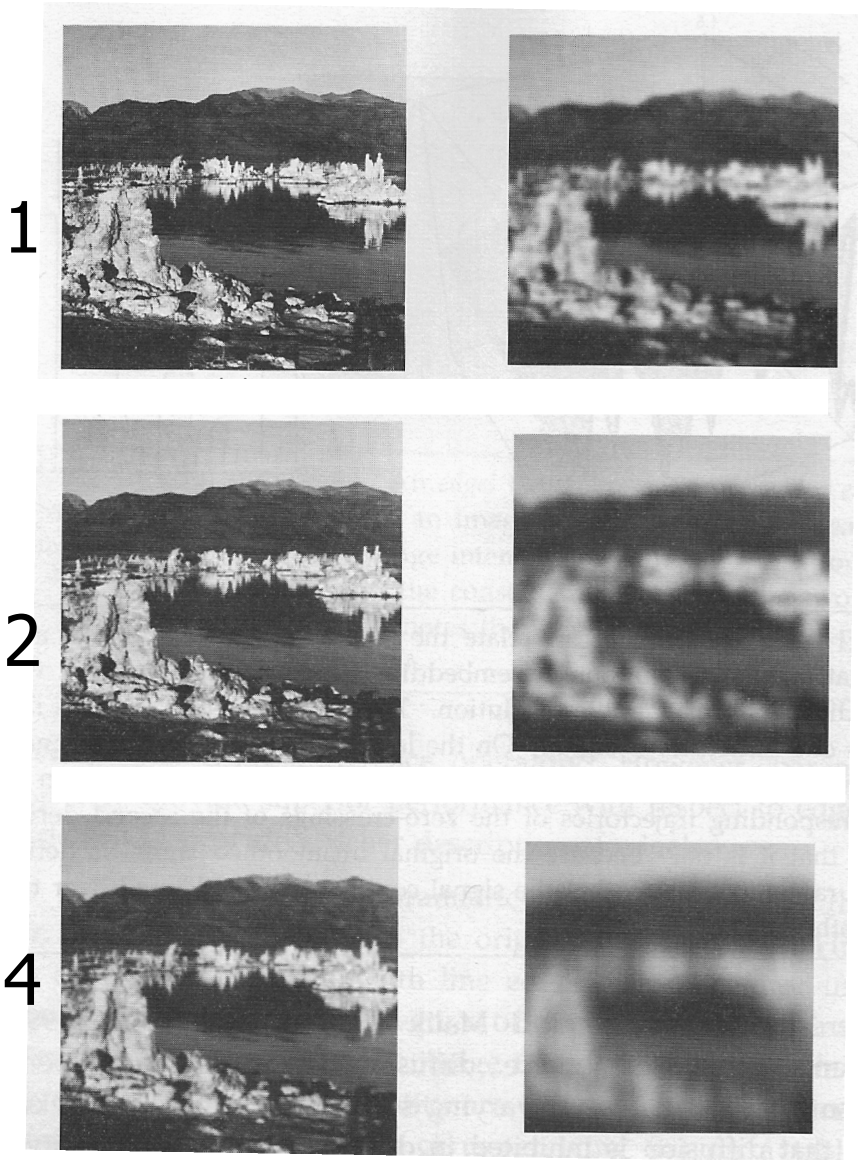
—> Convolving with a Gaussian suppresses the high frequencies.

Gaussian Smoothed Images

σ =1

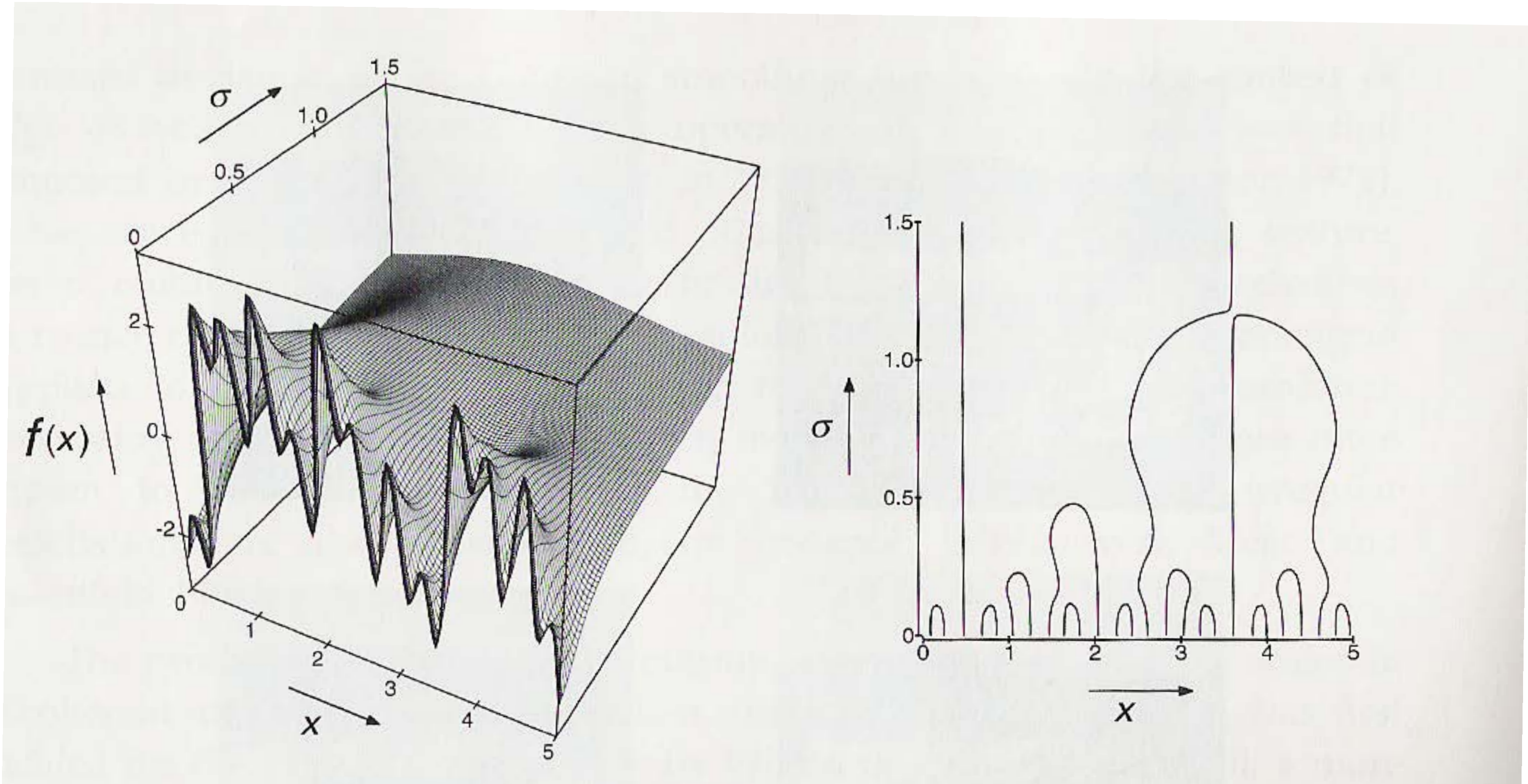
σ =2

σ =4



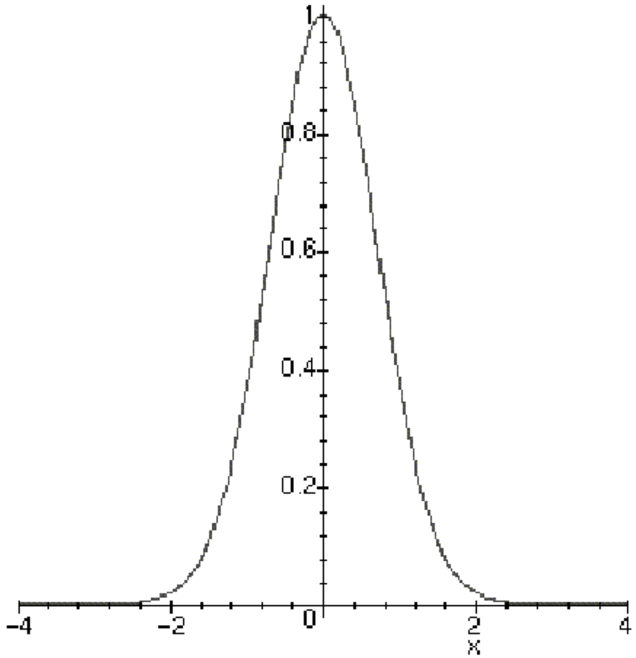
Original image Blurred image

Scale Space



Increasing scale (σ) removes high frequencies (details) but never adds artifacts.

Separability



$$g_1(x) = \frac{1}{\sqrt{\pi}\sigma} \exp(-x^2/\sigma^2)$$

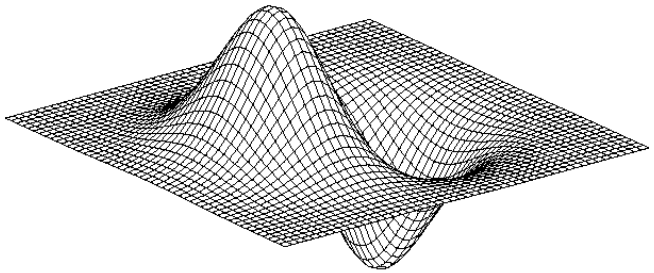
$$g_2(x,y) = g_1(x)g_1(y)$$

$$\int_u \int_v g_2(u,v) f(x-u, y-v)\,du\,dv = \int_u g_1(u) \left(\int_v g_1(v) f(x-u, y-v)\,dv \right) du$$

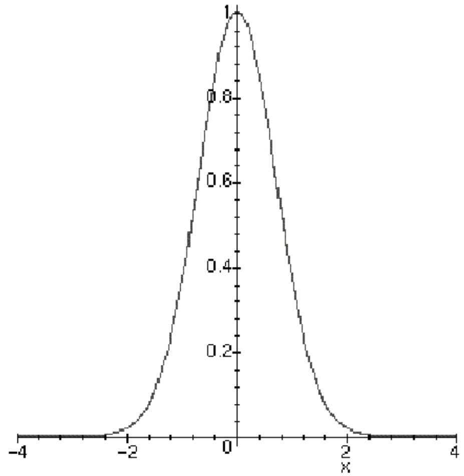
$$= \int_v g_1(v) \left(\int_u g_1(u) f(x-u, y-v)\,du \right) dv$$

—> 2D convolutions are never required. Smoothing can be achieved by successive 1D convolutions, which is faster.

Continuous Gaussian Derivatives



=



x

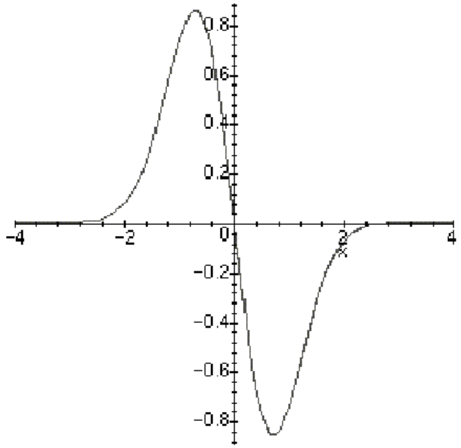
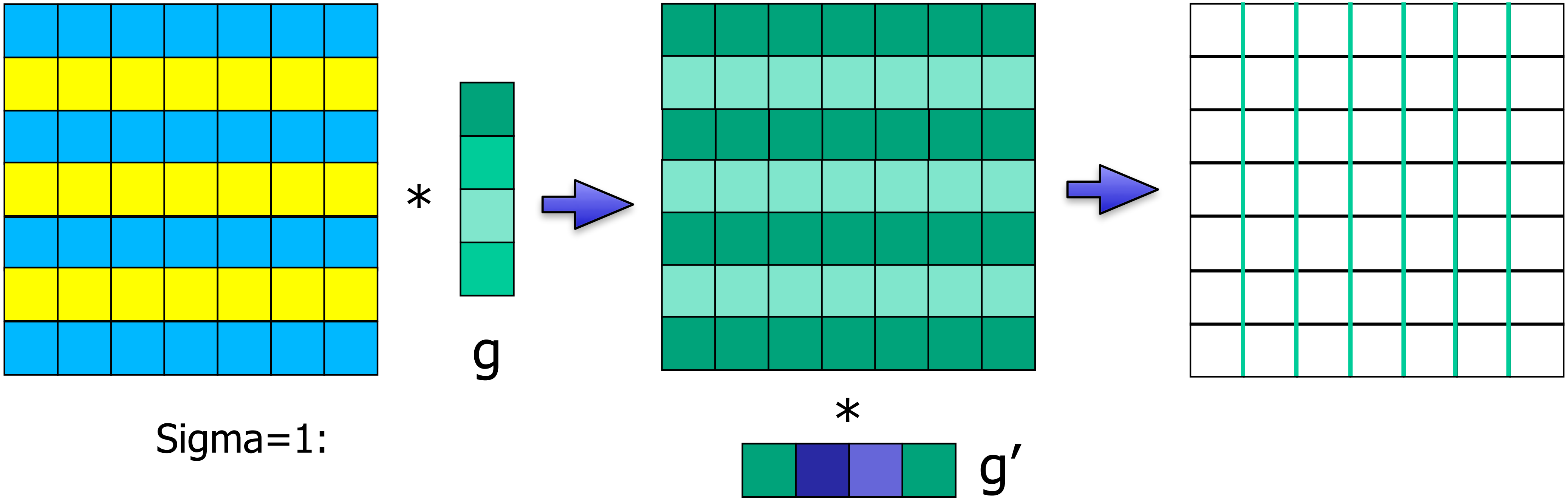


Image derivatives computed by convolving with the derivative of a Gaussian:

$$\frac{\partial}{\partial x}\int_u\int_v g_2(u,v)f(x-u,y-v)dudv = \int_u g_1^{'}(u)\left(\int_v g_1(v)f(x-u,y-v)dv\right)du$$

$$\frac{\partial}{\partial y}\int\int g_2(u,v)f(x-u,y-v)dudv = \int_v g_1^{'}(v)\left(\int_u g_1(u)f(x-u,y-v)du\right)dv$$

Discrete Gaussian Derivatives



Sigma=1:

g : 0.000070 0.010332 0.207532 0.564131 0.207532 0.010332 0.000070

g′: 0.000418 0.041330 0.415065 0.000000 -0.415065 -0.041330 -0.000418

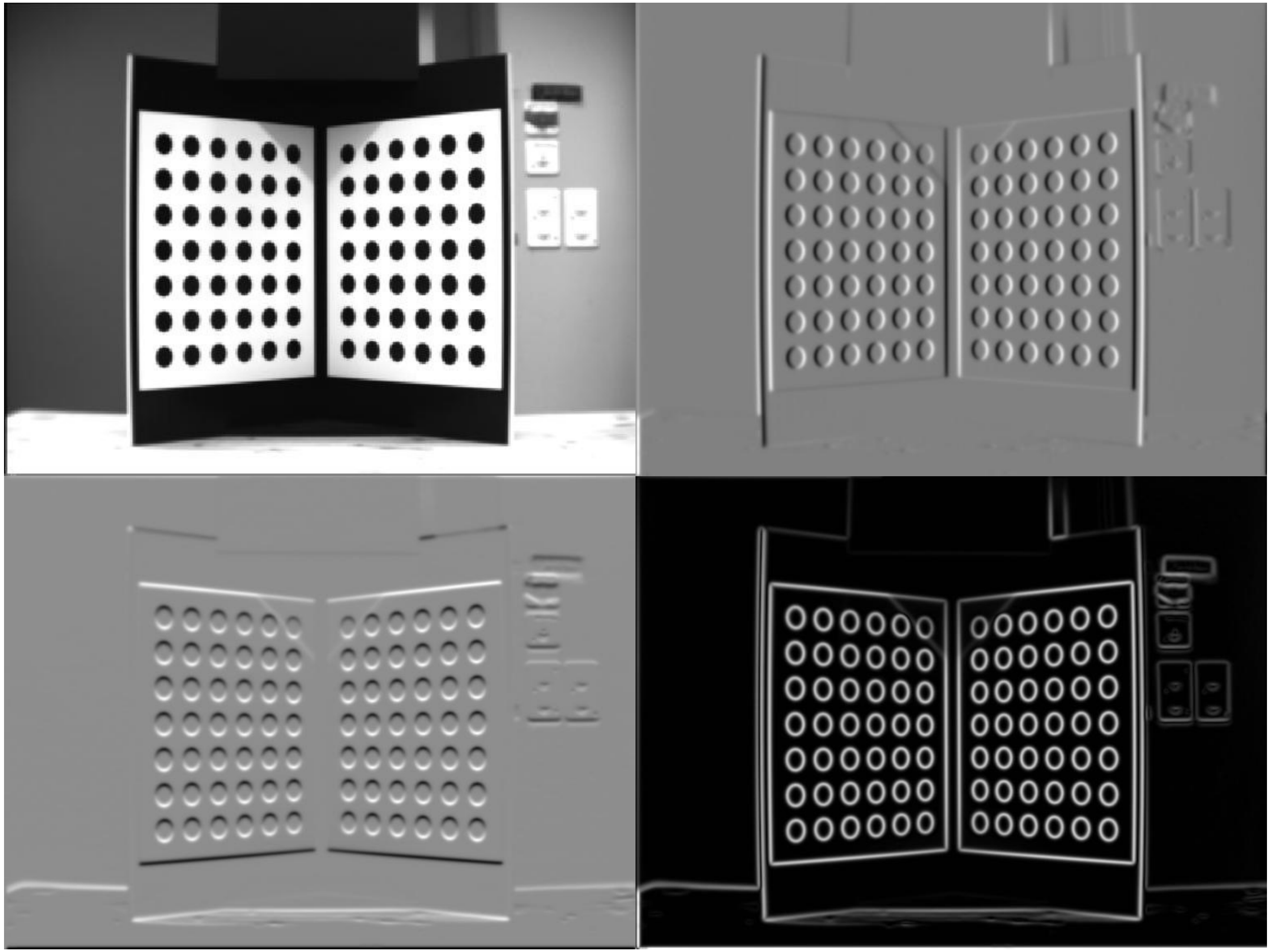
Sigma=2:

g : 0.005167 0.029735 0.103784 0.219712 0.282115 0.219712 0.103784 0.029735 0.005167

g′: 0.010334 0.044602 0.103784 0.109856 0.000000 -0.109856 -0.103784 -0.044602 -0.010334

—> Only requires 1D convolutions with relatively small masks.

Derivative Images



I

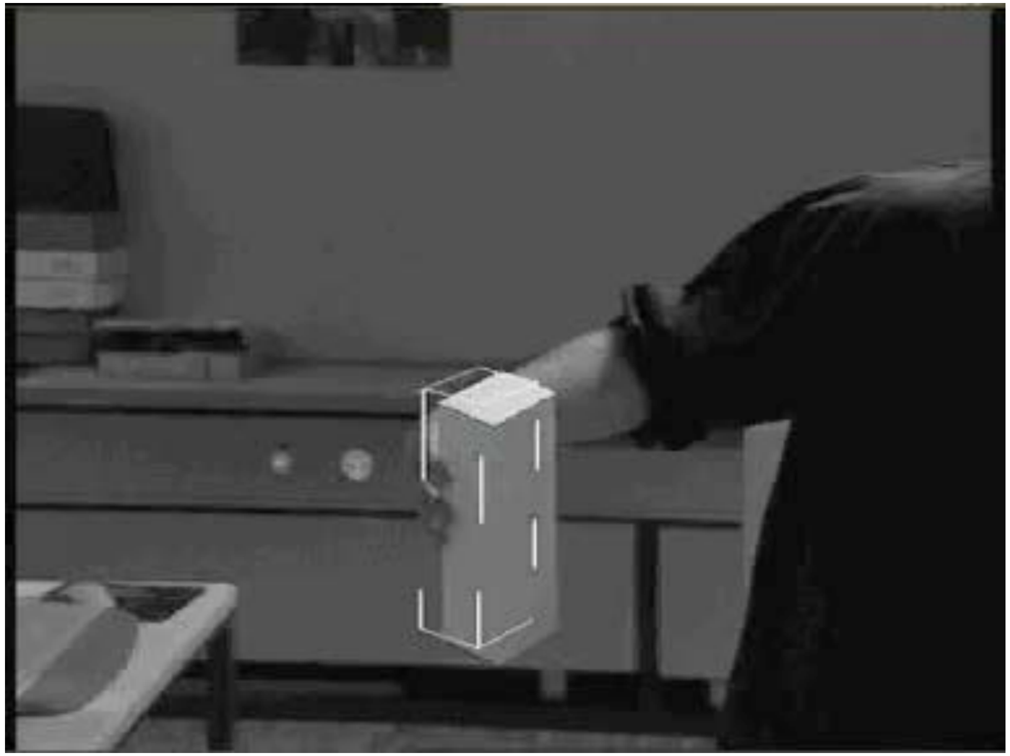
$\frac{\partial I}{\partial x}$

$\frac{\partial I}{\partial y}$

$\sqrt{\frac{\partial I}{\partial x}^2 + \frac{\partial I}{\partial y}^2}$

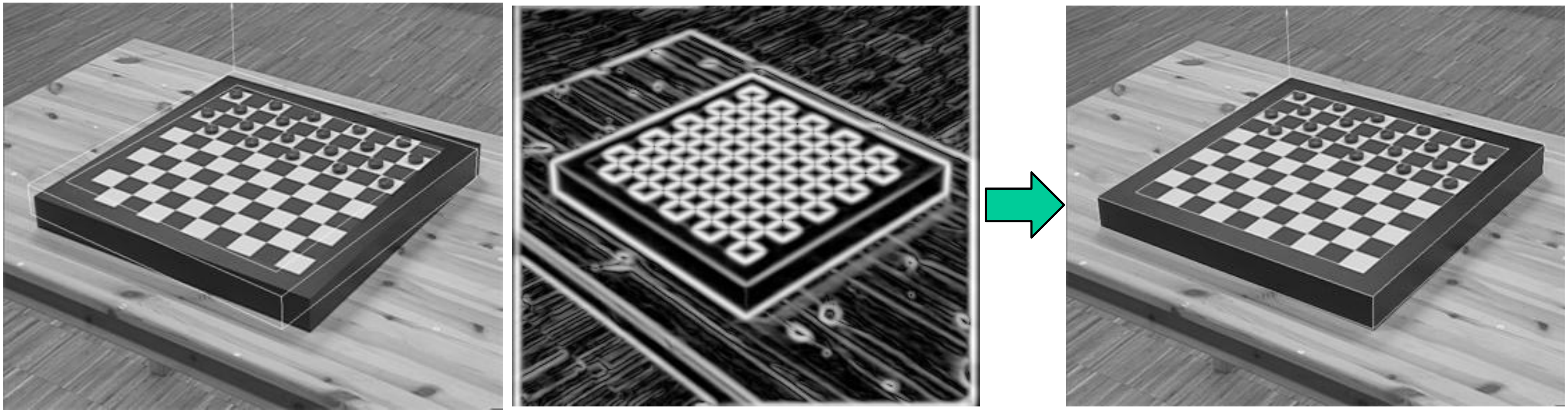
EPFL

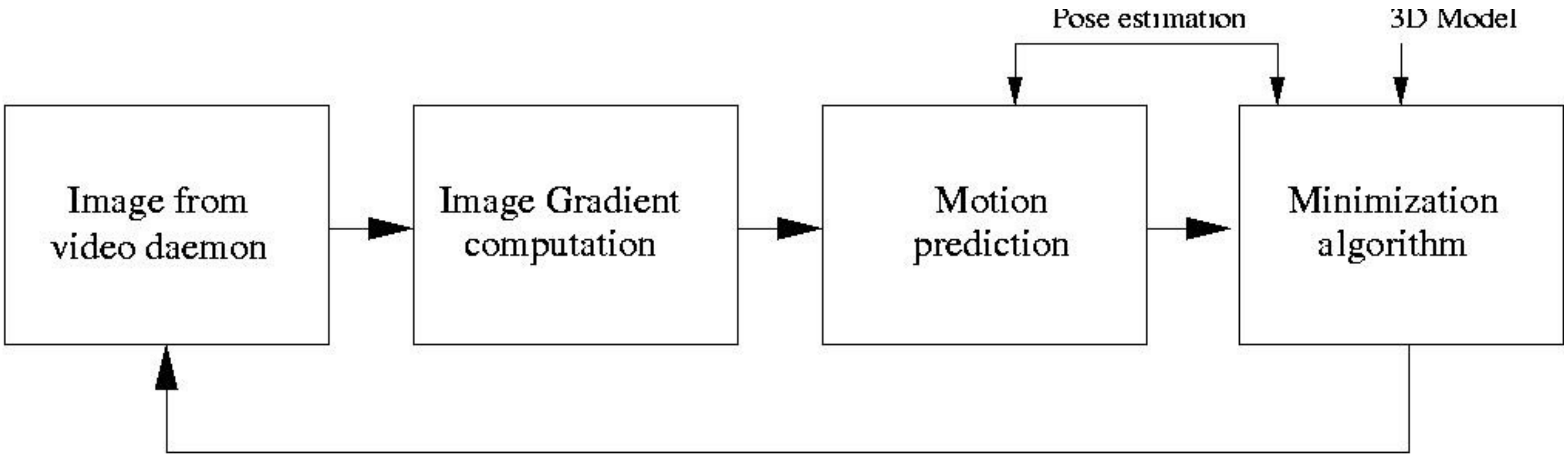
Gradient-Based Tracking



Maximize edge-strength along projection of the 3—D wireframe.

Gradient Maximization





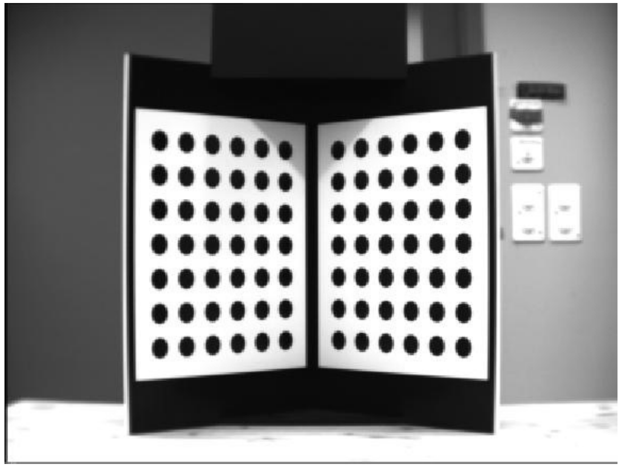
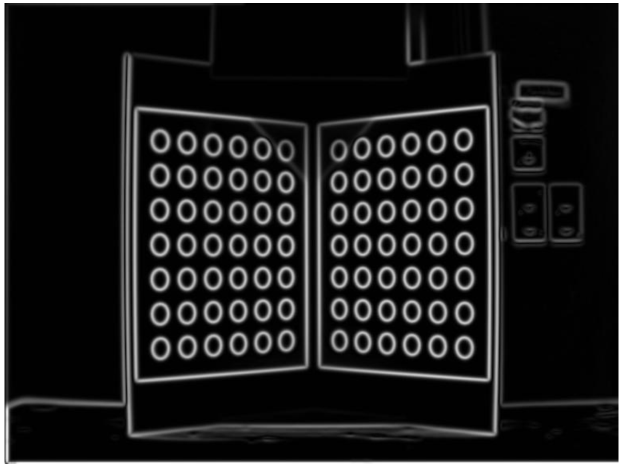
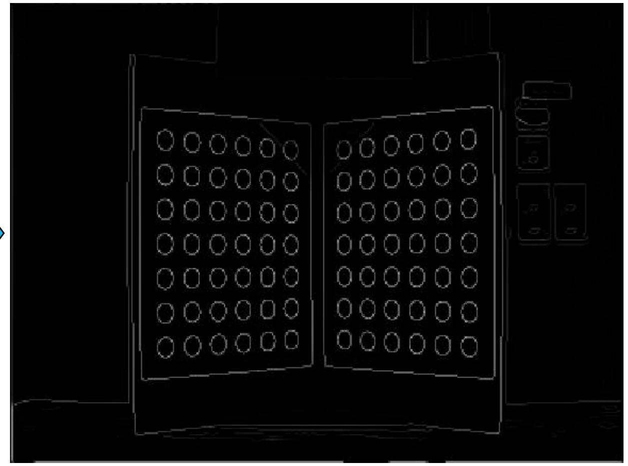
Real-Time Tracking

Canny Edge Detector

I

$$\sqrt{\frac{\partial I}{\partial x}^2 + \frac{\partial I}{\partial y}^2}$$

Thinned gradient images


Canny Edge Detector

Convolution

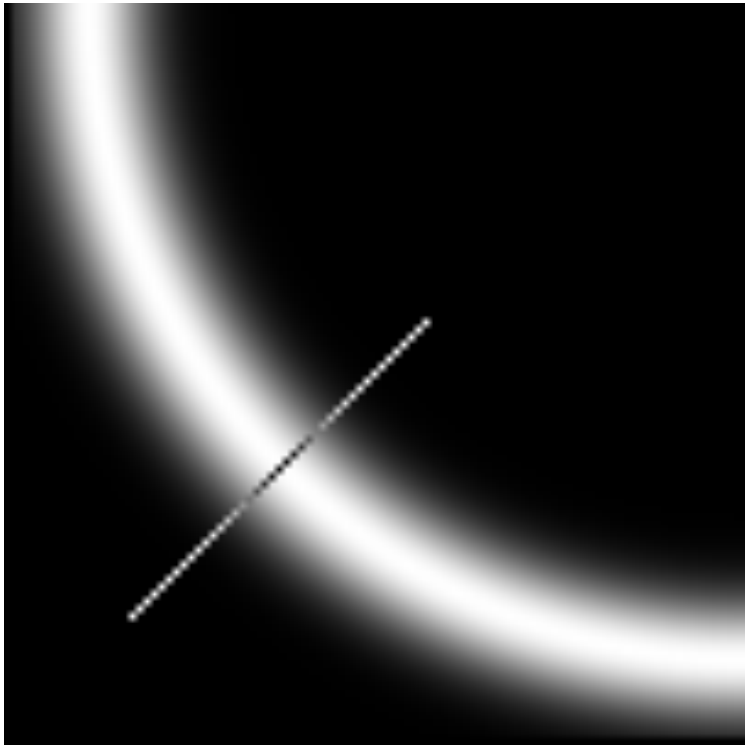
- Gradient strength
- Gradient direction

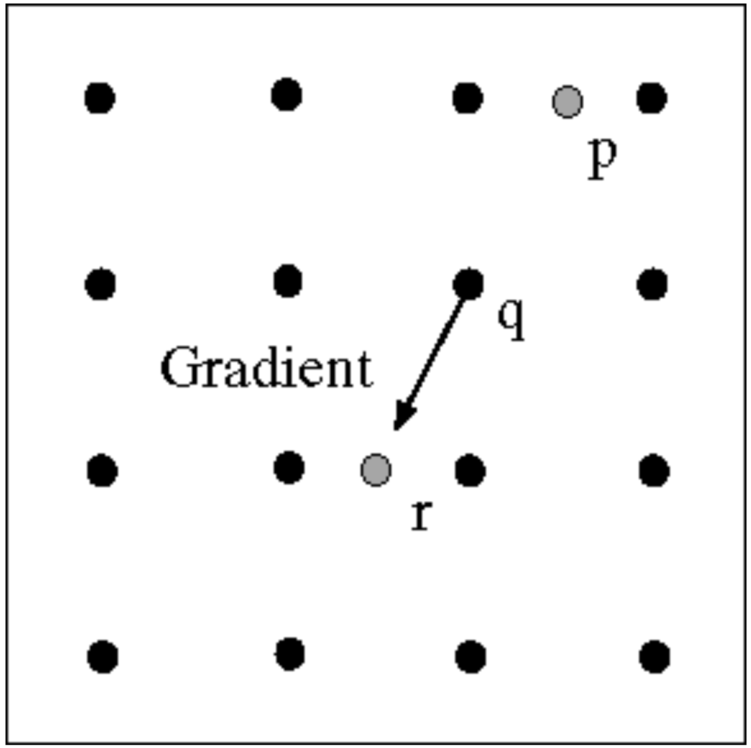
Thresholding

Non Maxima Suppression

Hysteresis Thresholding

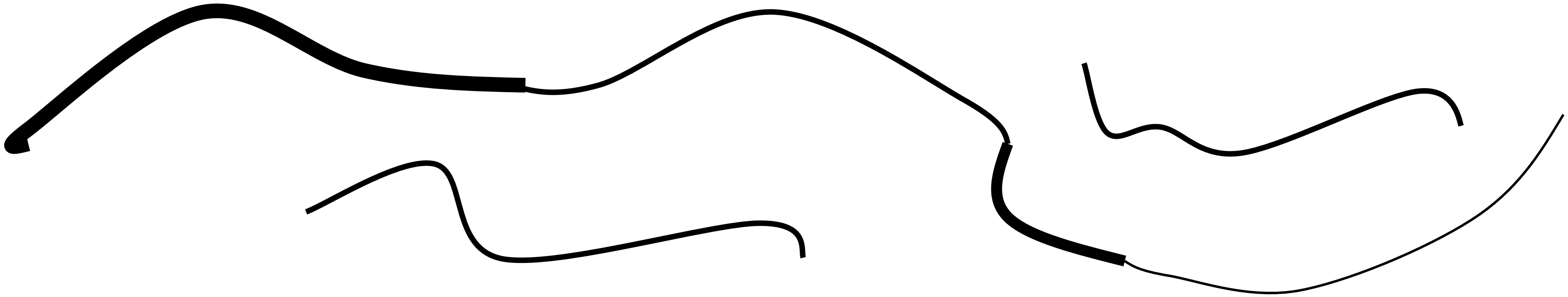
Non-Maxima Suppression





Check if pixel is local maximum along gradient direction, which requires checking interpolated pixels p and r.

Hysteresis Thresholding



- Algorithm takes two thresholds: high & low
  - A pixel with edge strength above high threshold is an edge.
  - Any pixel with edge strength below low threshold is not.
  - Any pixel above the low threshold and next to an edge is an edge.

- Iteratively label edges
  - Edges grow out from ‘strong edges’
  - Iterate until no change in image.

Canny Results

σ=1, T2=255, T1=1

'Y' or 'T' junction problem with Canny operator

Canny Results

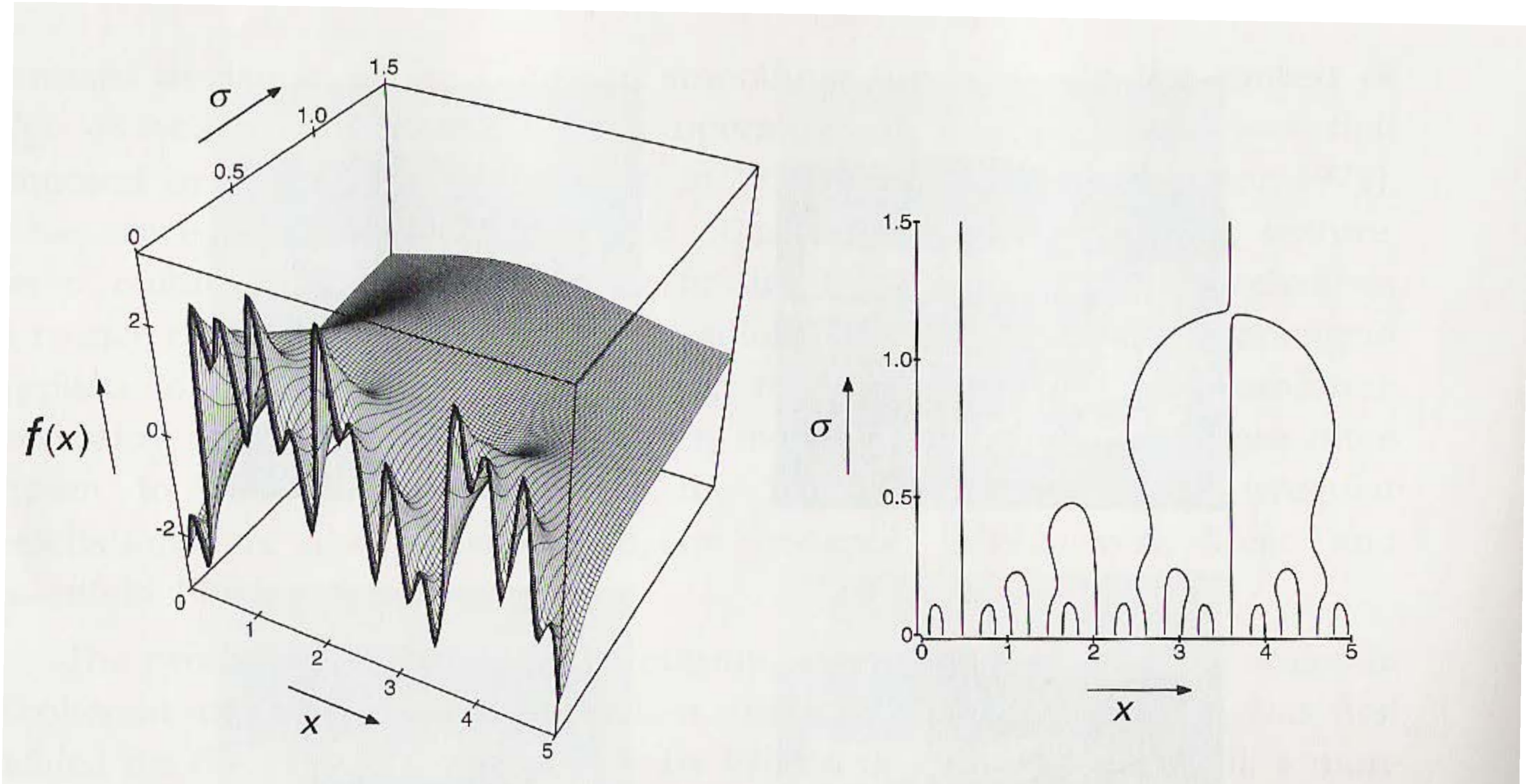
σ=1, T2=255, T1=220

σ=1, T2=128, T1=1

σ=2, T2=128, T1=1

Heath et al., PAMI'97

Scale Space Revisited



Increasing scale (σ) removes details but never adds new ones:

- Edge position may shift.
- Two edges may merge.
- An edge may **not** split into two.

Multiple Scales

$\sigma = 1$ $\sigma = 2$ $\sigma = 4$

→Choosing the right scale is a difficult semantic problem.

EPFL

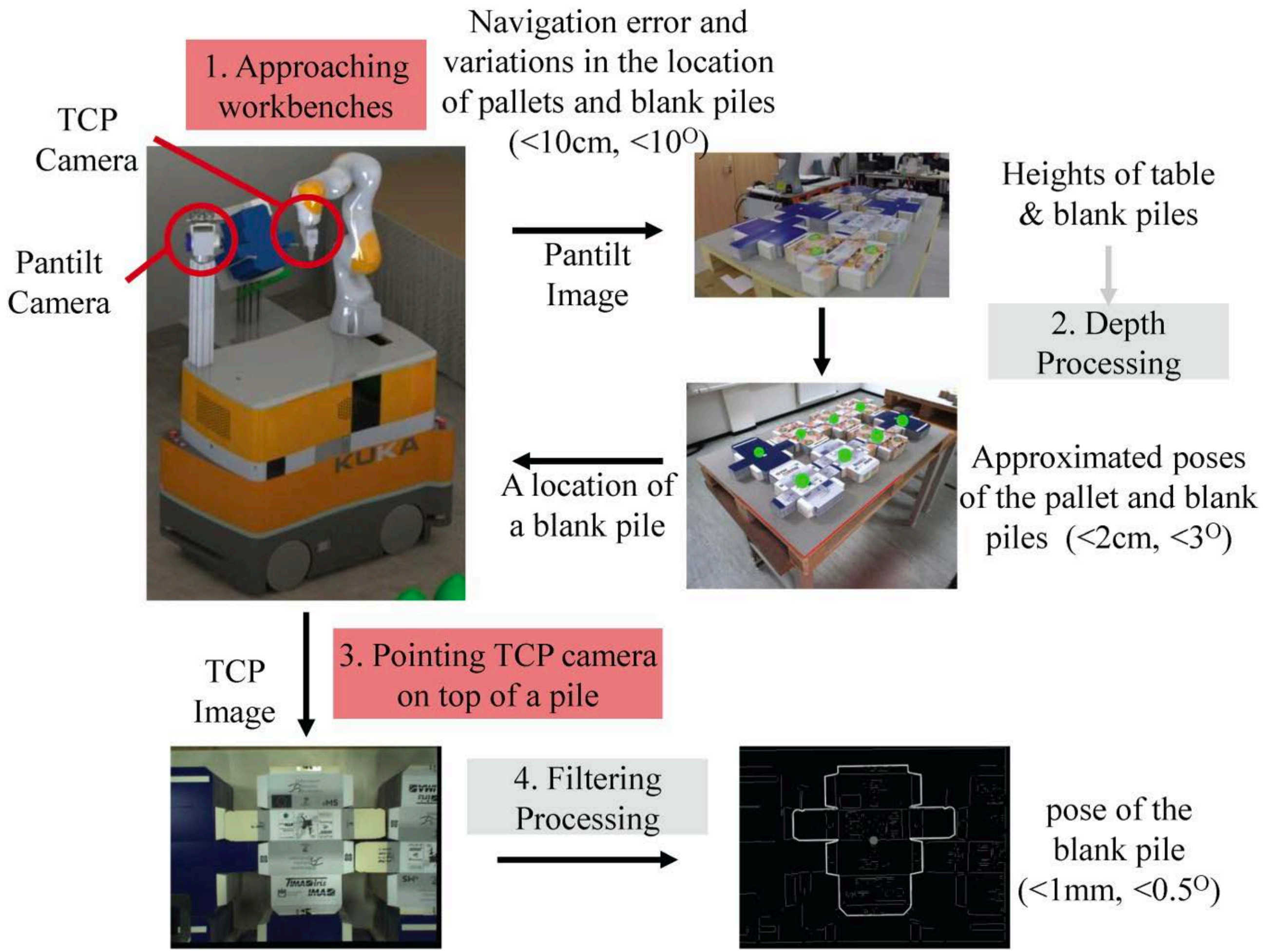
Scale vs Threshold

Fine scale
High threshold

Coarse scale
High threshold

Coarse scale
Low threshold

Industrial Application



In industrial environments where the Canny parameters can be properly adjusted:

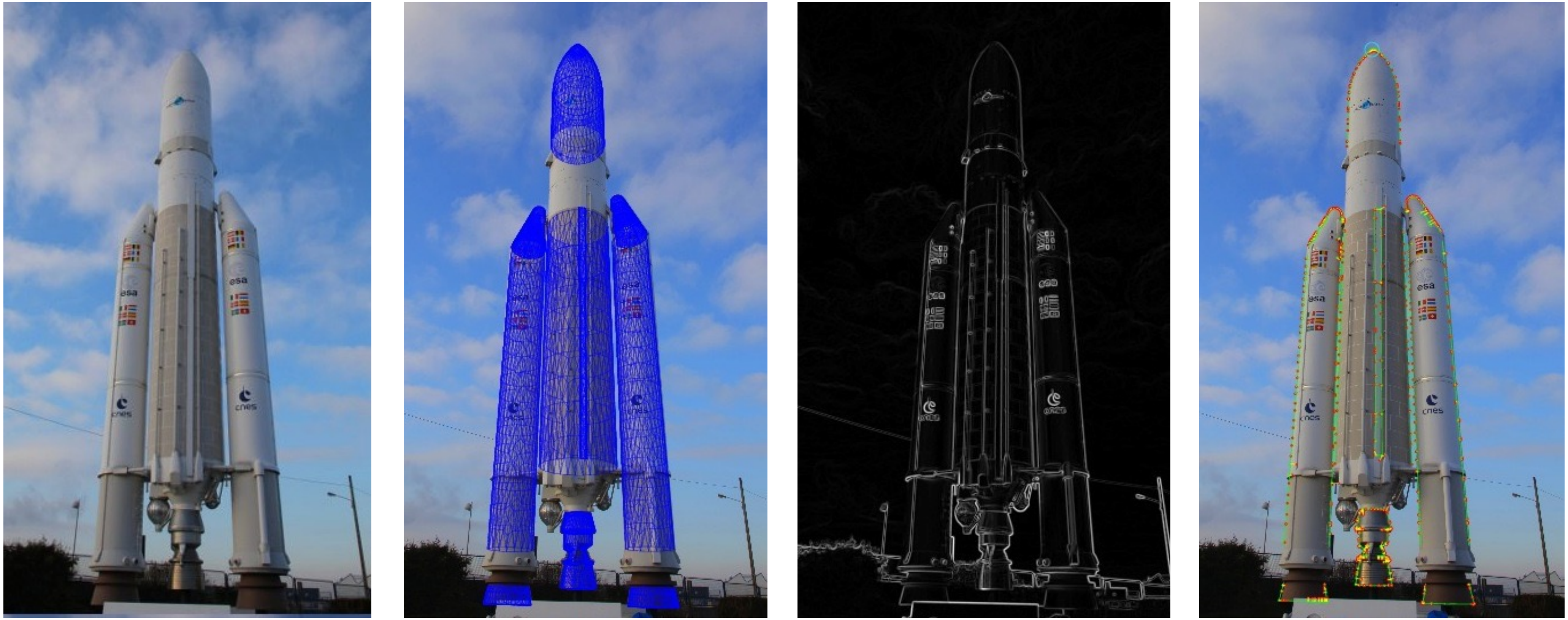
- It is fast.
- Does not require training data.

Lim et al. , Advanced Robotics 2019.

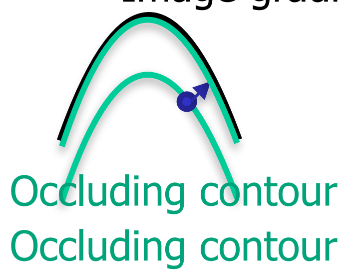
Visual Servoing

—> A useful tool in our toolbox.

Tracking a Rocket

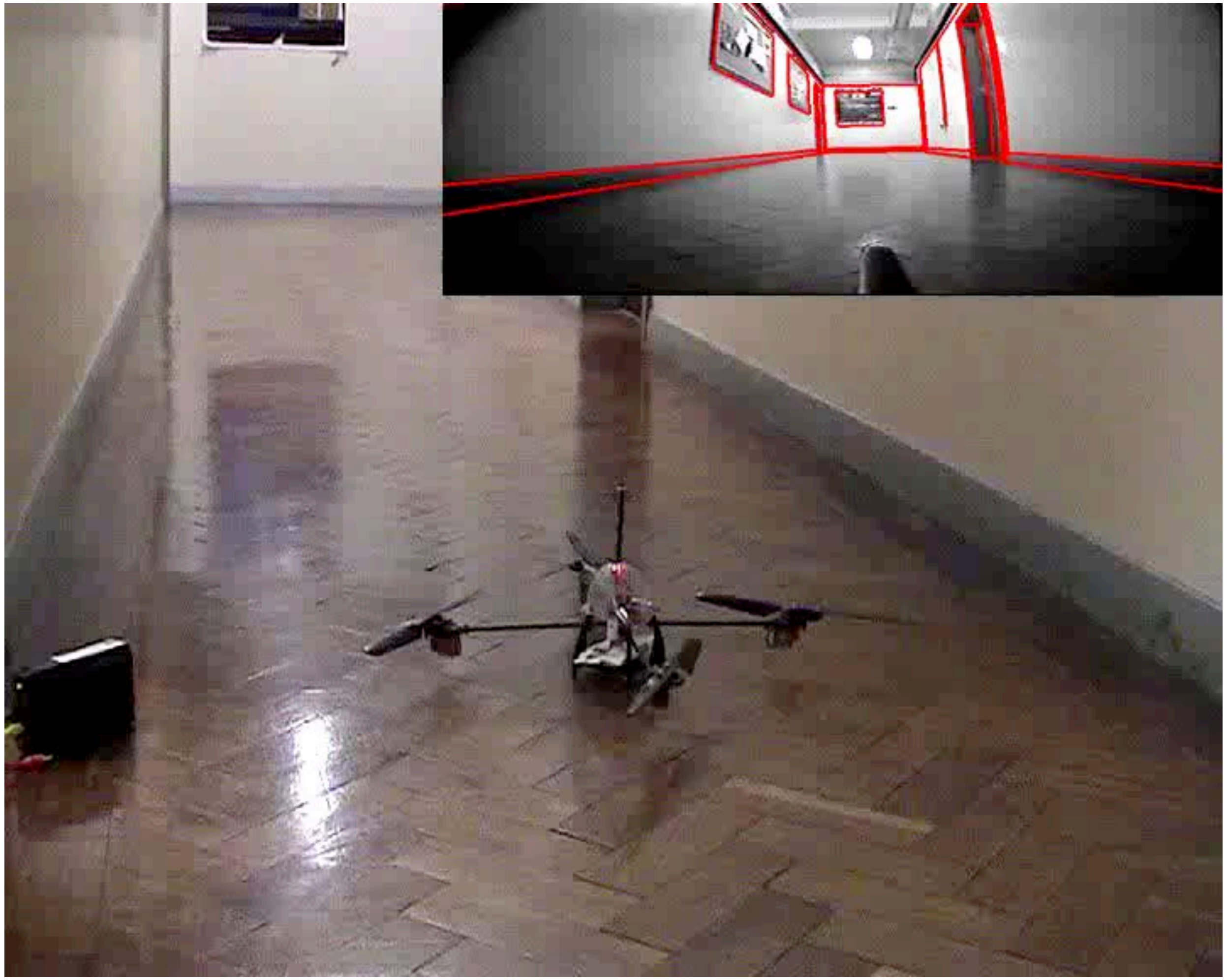


Given an initial pose estimate:

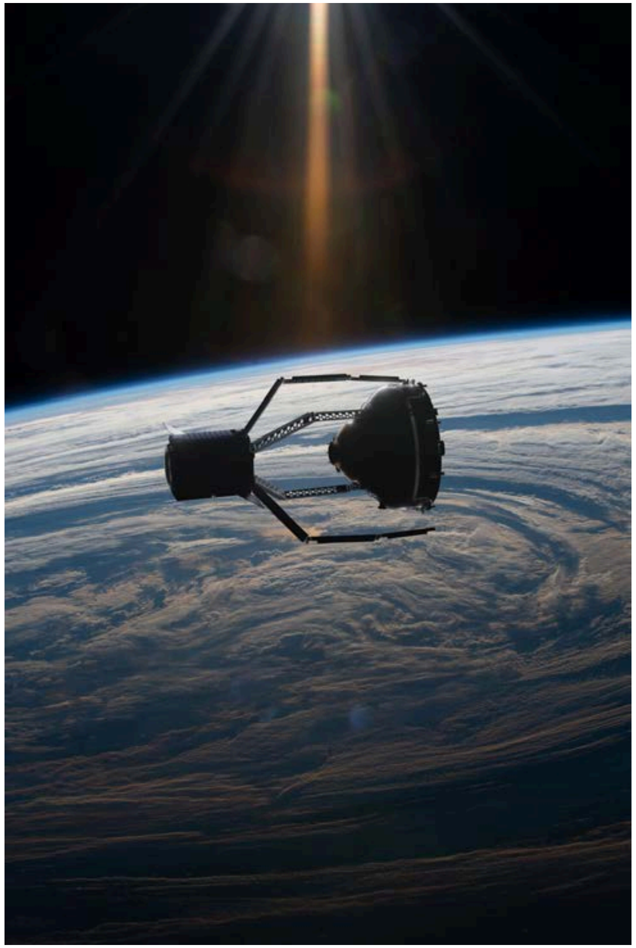


- Find the occluding contours.
- Find closest edge points in the normal direction.
- Re-estimate pose to minimize sum of square distances.
- Iterate until convergence.

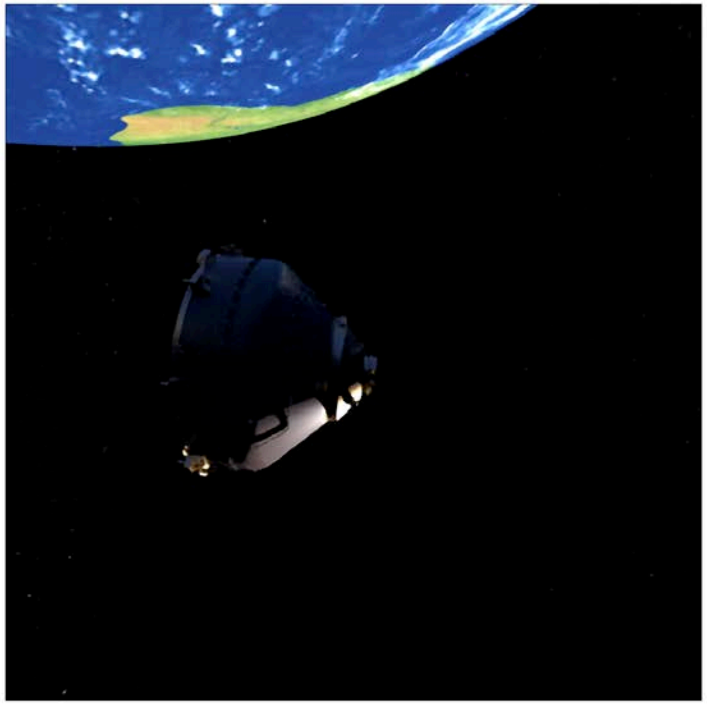
Visual Servoing

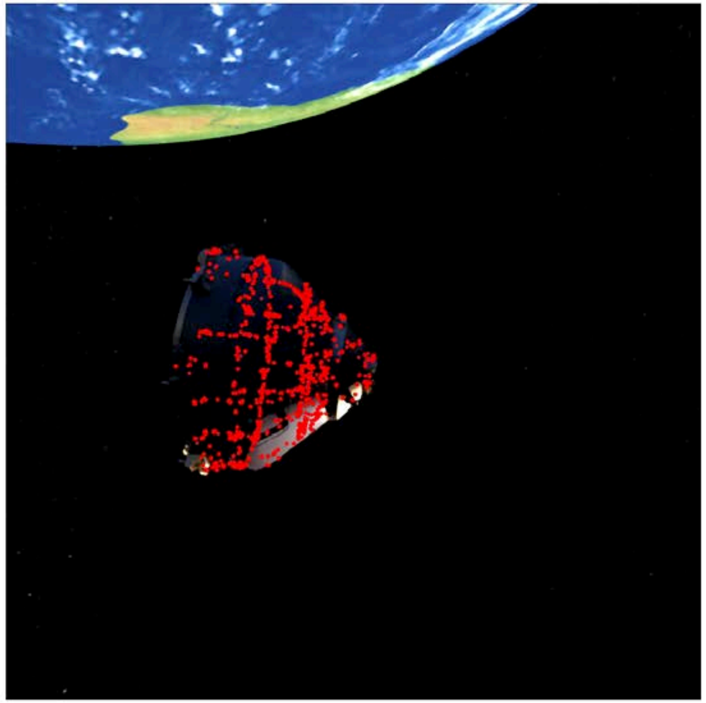


Space Cleaning



Capturing and deorbiting a dead satellite.





- A more sophisticated version of this old algorithm will blast off in 2026!
- ESA does not yet trust neural nets for such a mission.

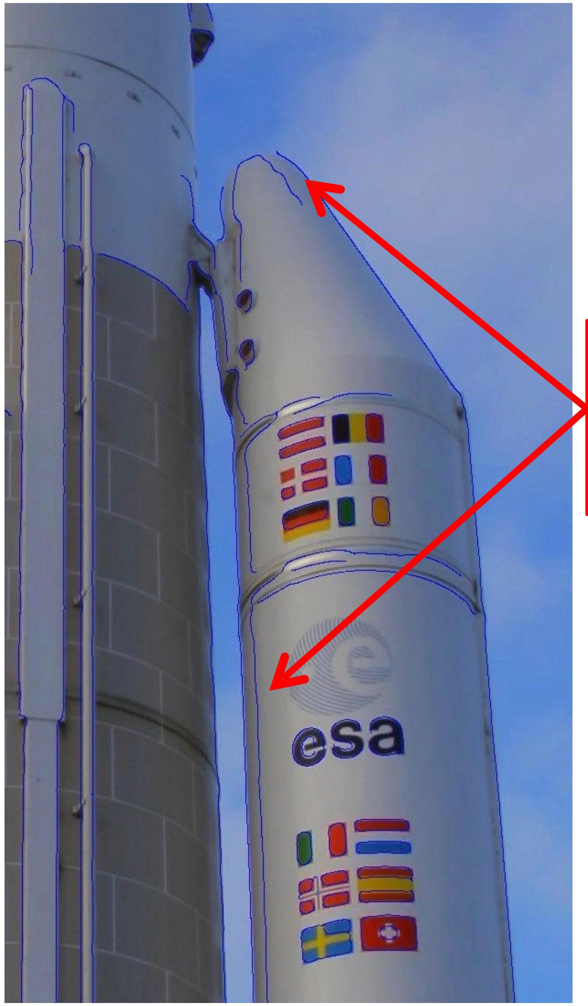
https://clearspace.today/

Limitations of the Canny Algorithm

There is no ideal value of σ!

EPFL

Steep Smooth Shading



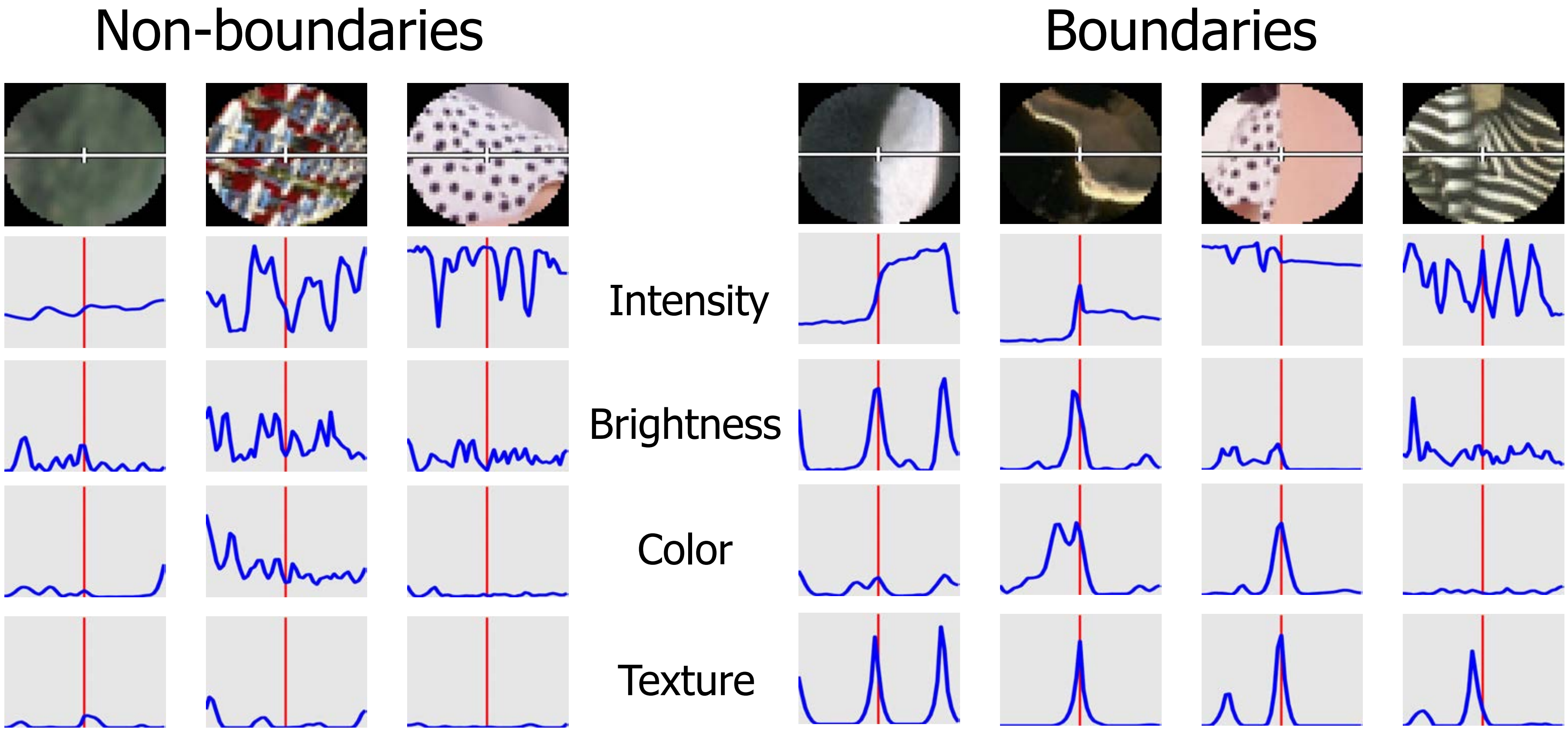
- Rapidly varying gray levels.
- Large gradients.

→ Shading can produce spurious edges.

Texture Boundaries

- Not all image contours are characterized by strong contrast.
- Sometimes, textural changes are just as significant.

Different Boundary Types

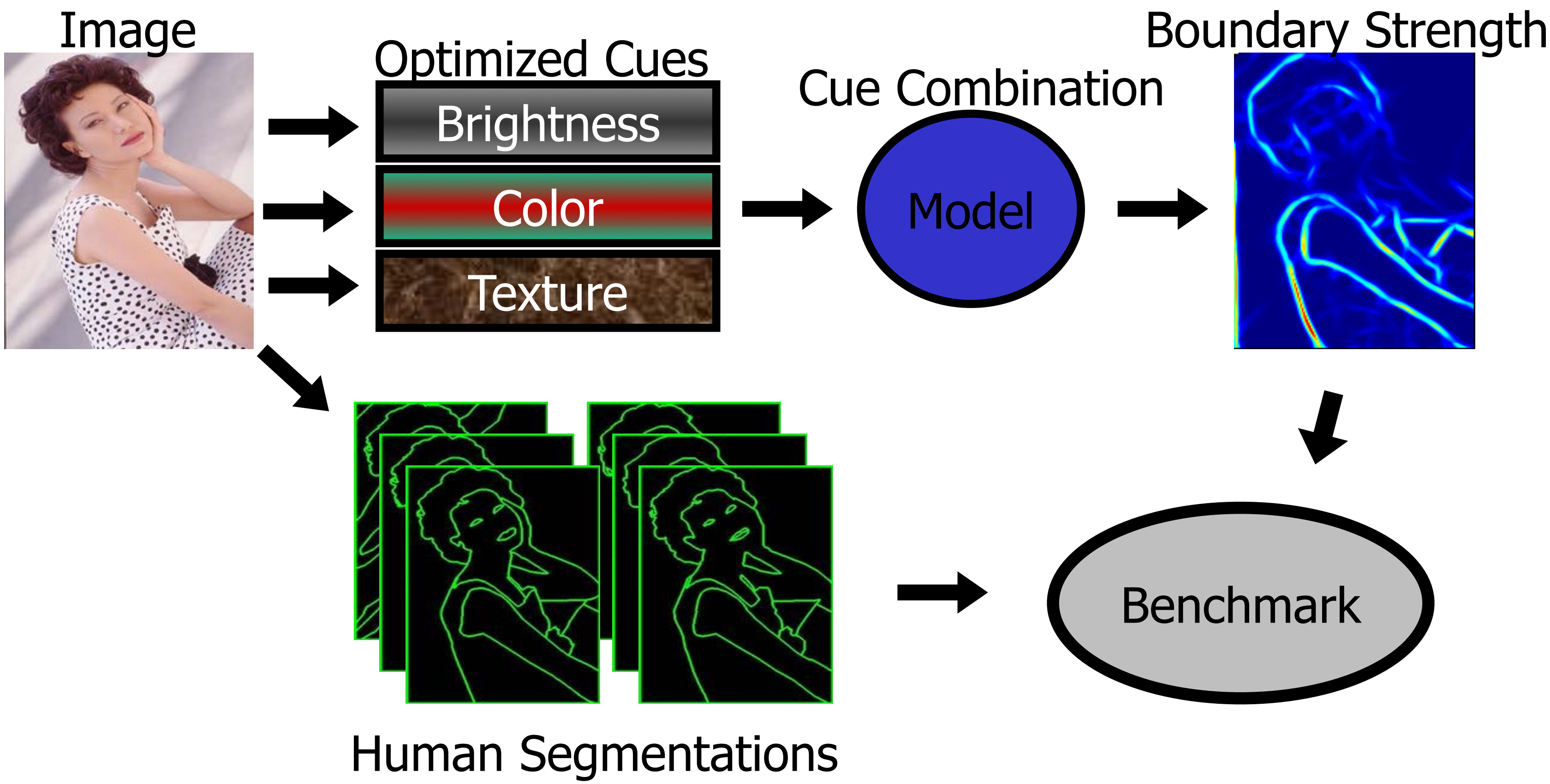


Martin et al. , PAMI'04

Training Database

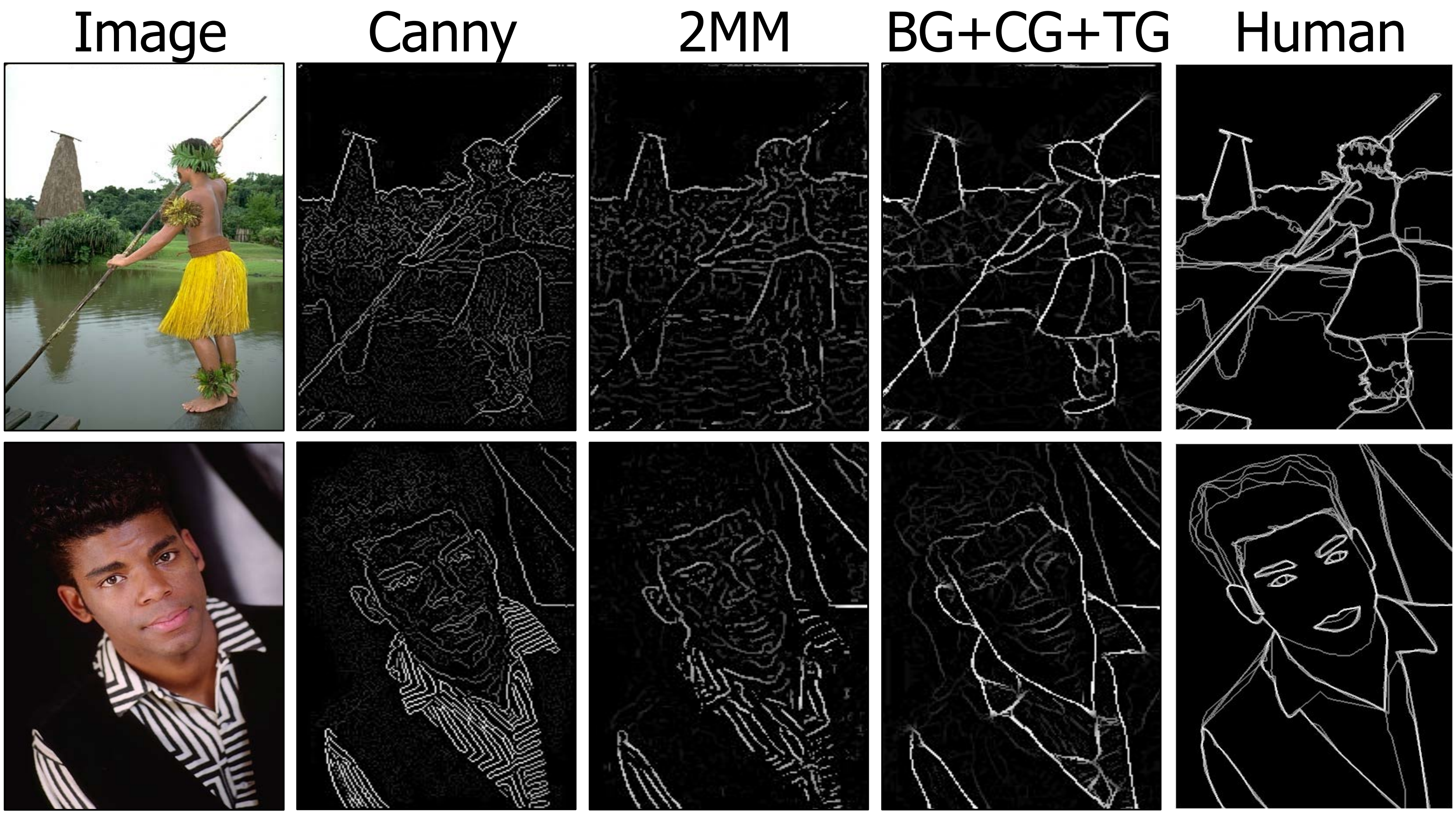
1000 images with 5 to 10 segmentations each.

Machine Learning



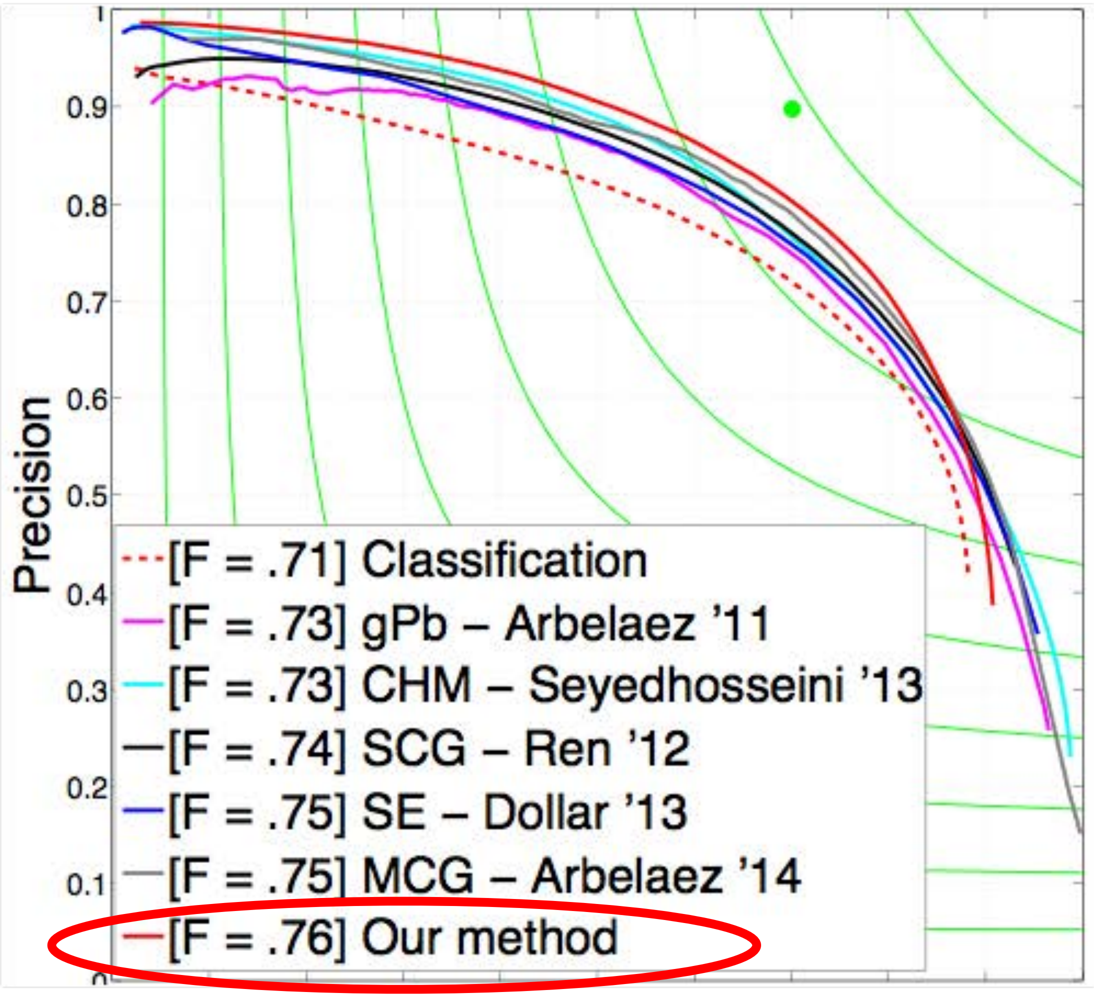
Learn the probability of being a boundary pixel on the basis of a set of features.

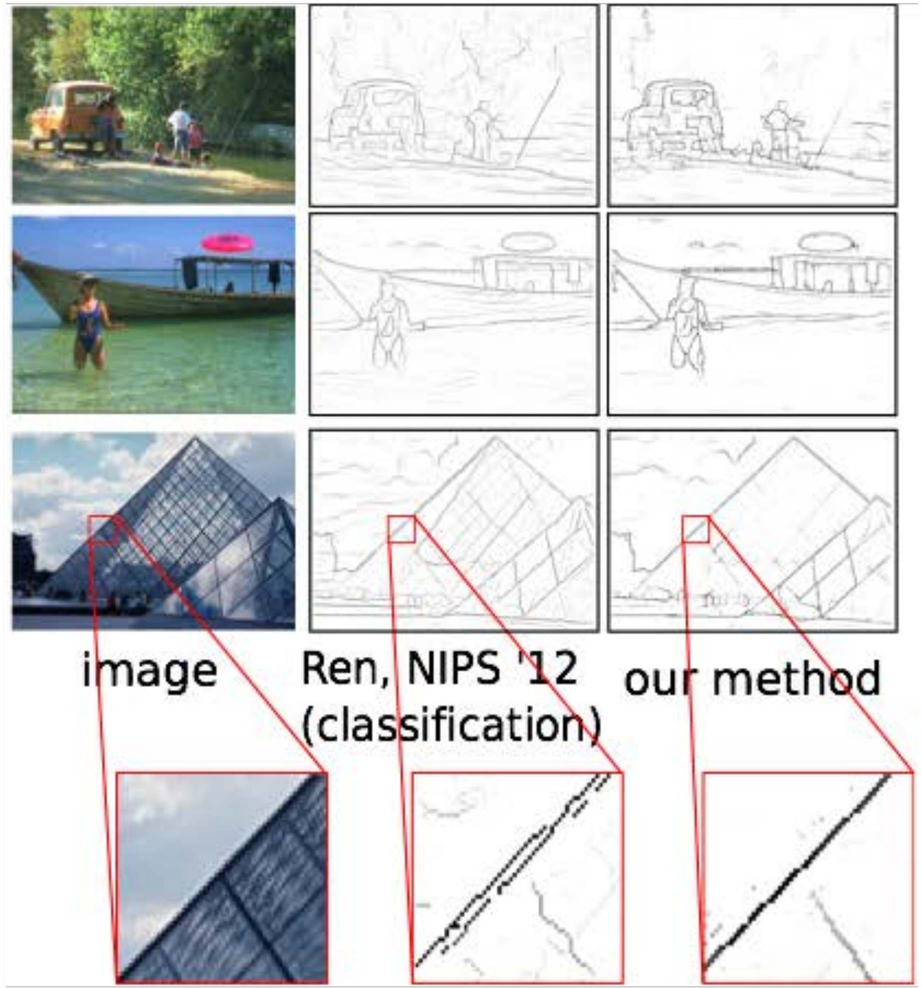
Comparative Results



Martin et al. , PAMI'04

Classification vs Regression

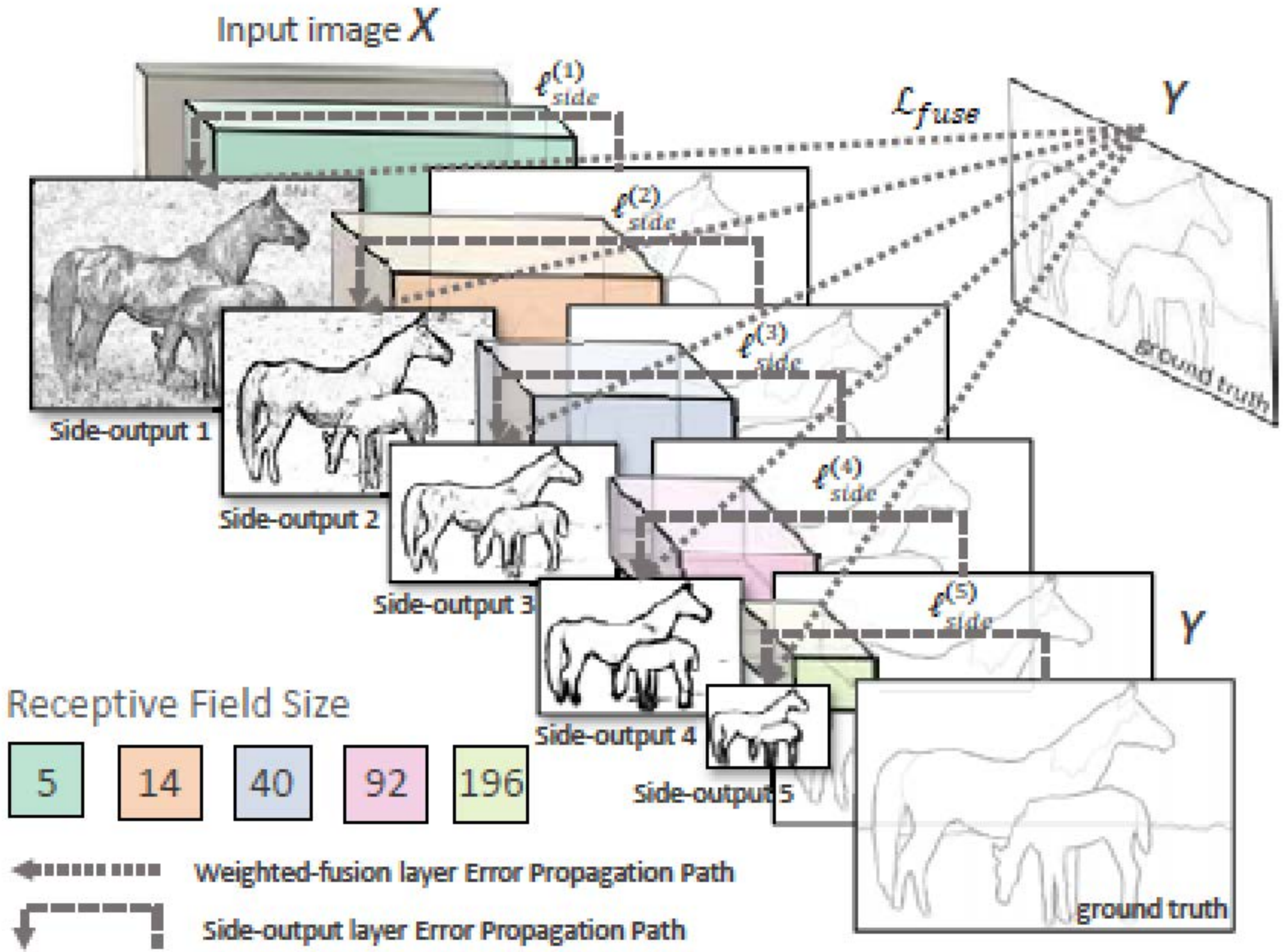




Yes!

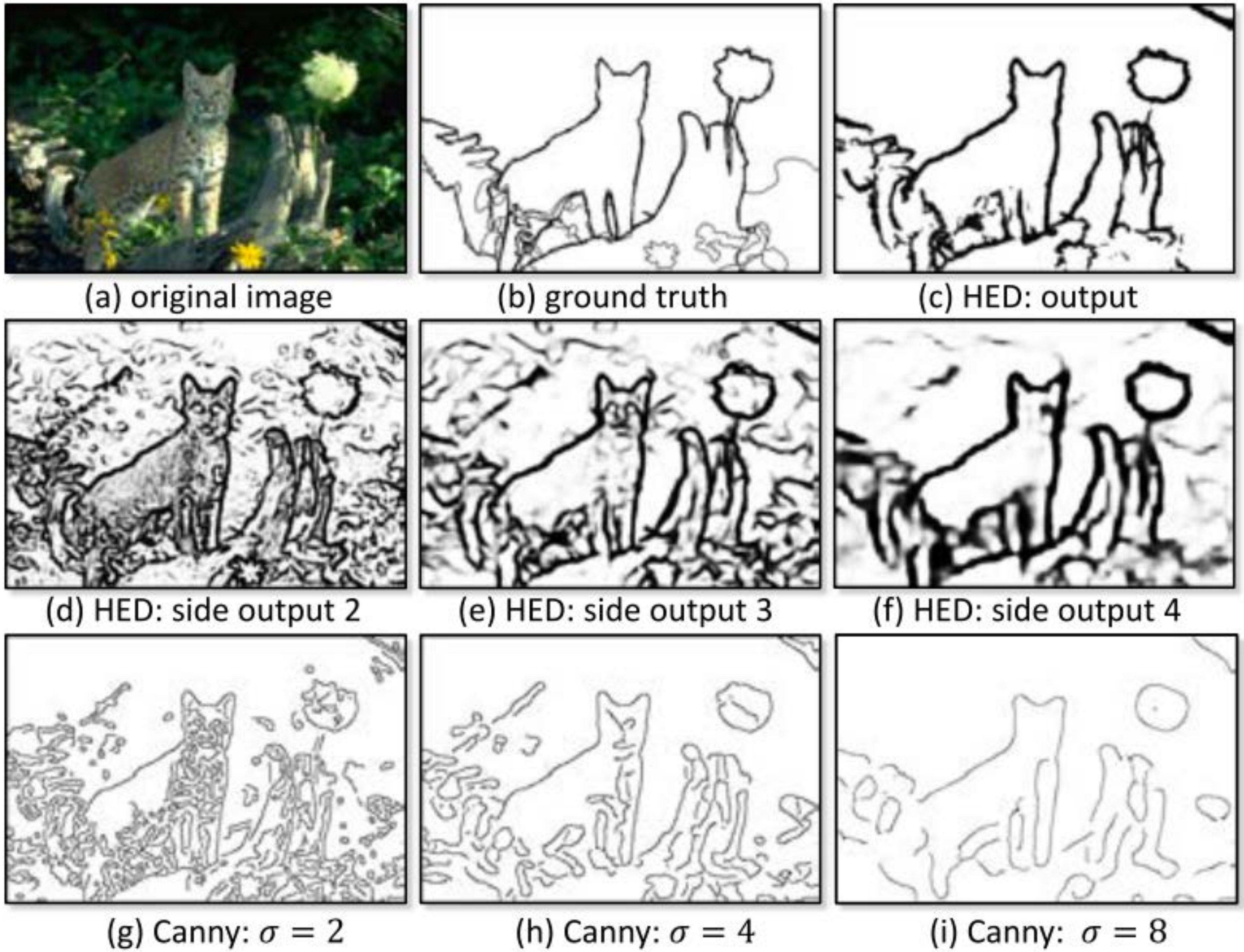
Sironi, et al. PAMI'15

Deep Learning

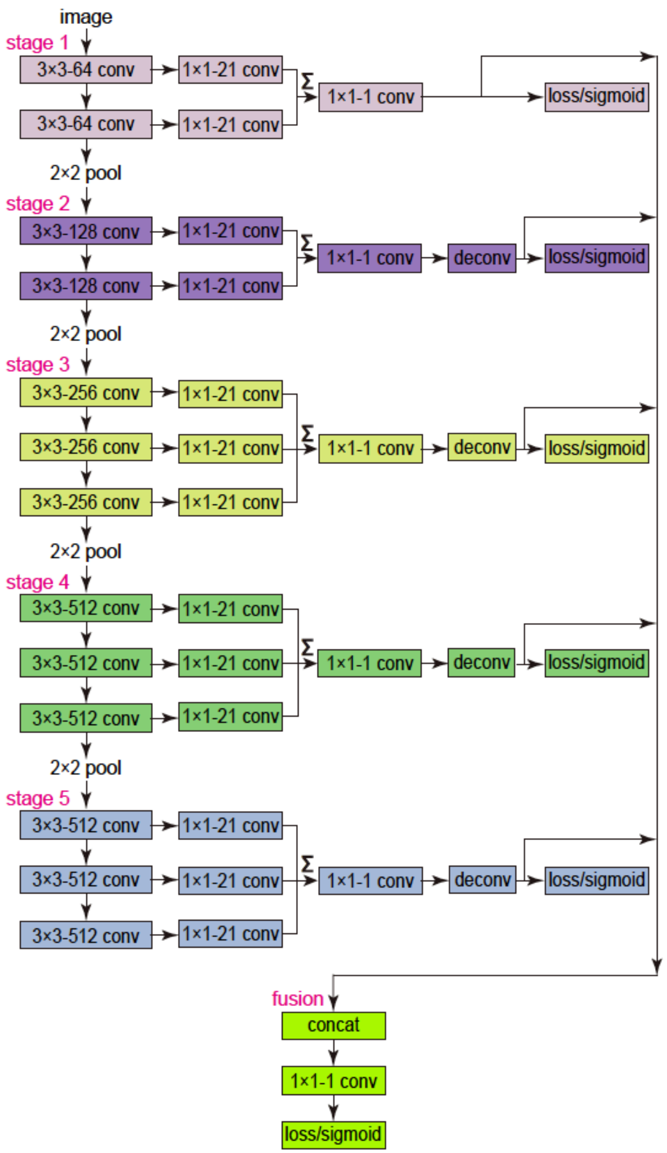


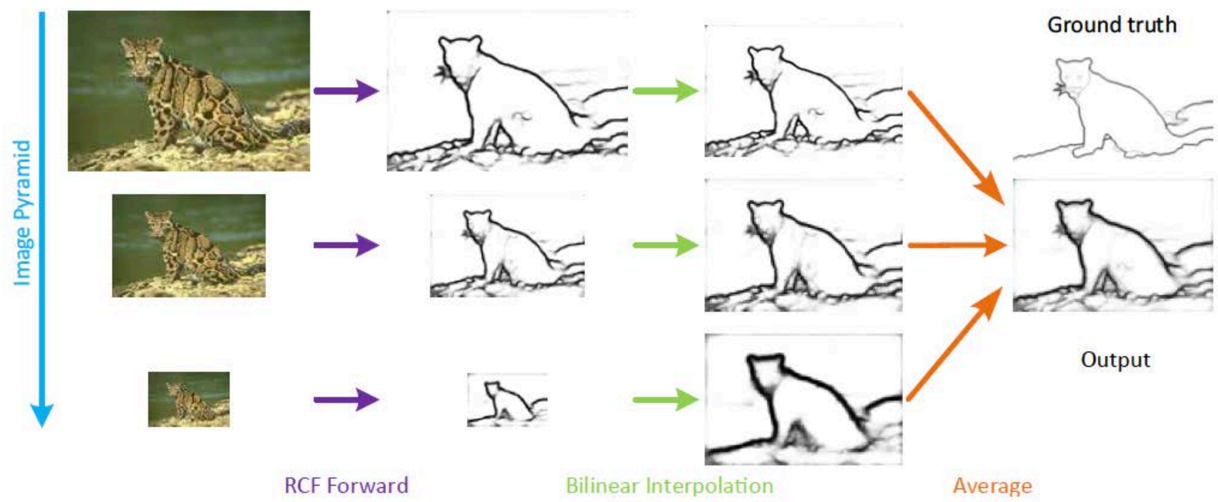
Xie and Tu, ICCV'15

Deep Learning Vs Canny



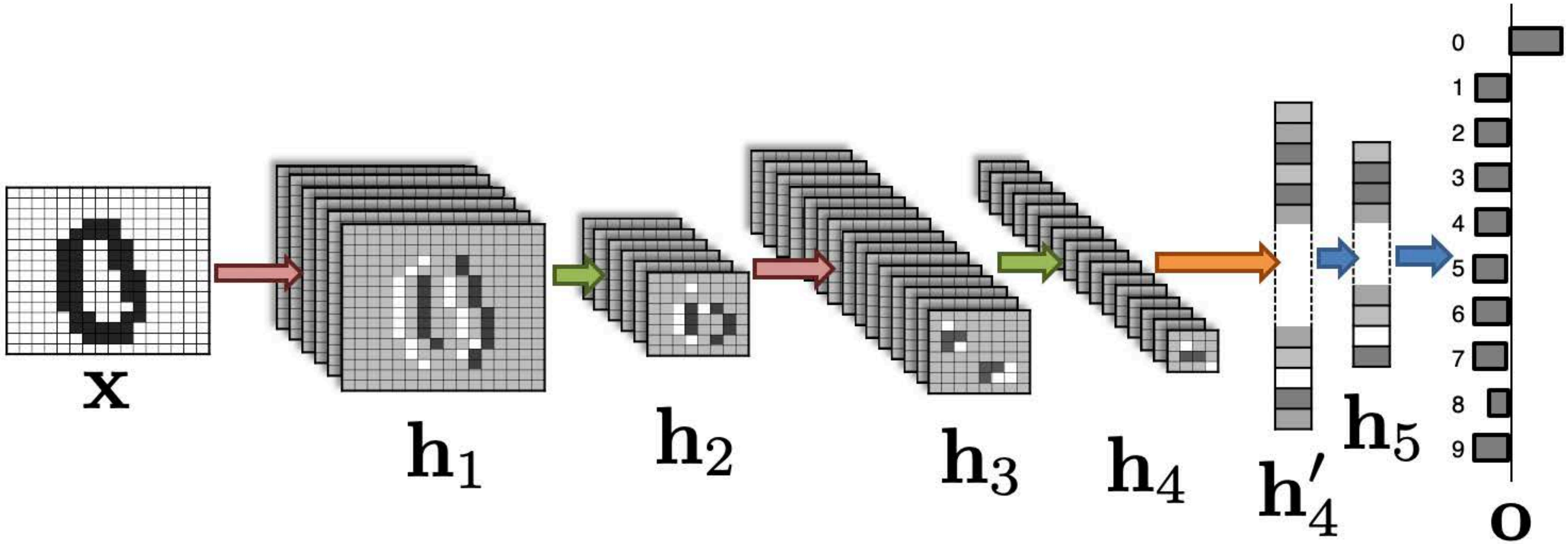
Deeper Learning





Liu et al. , CVPR’17

Convolutional Neural Network

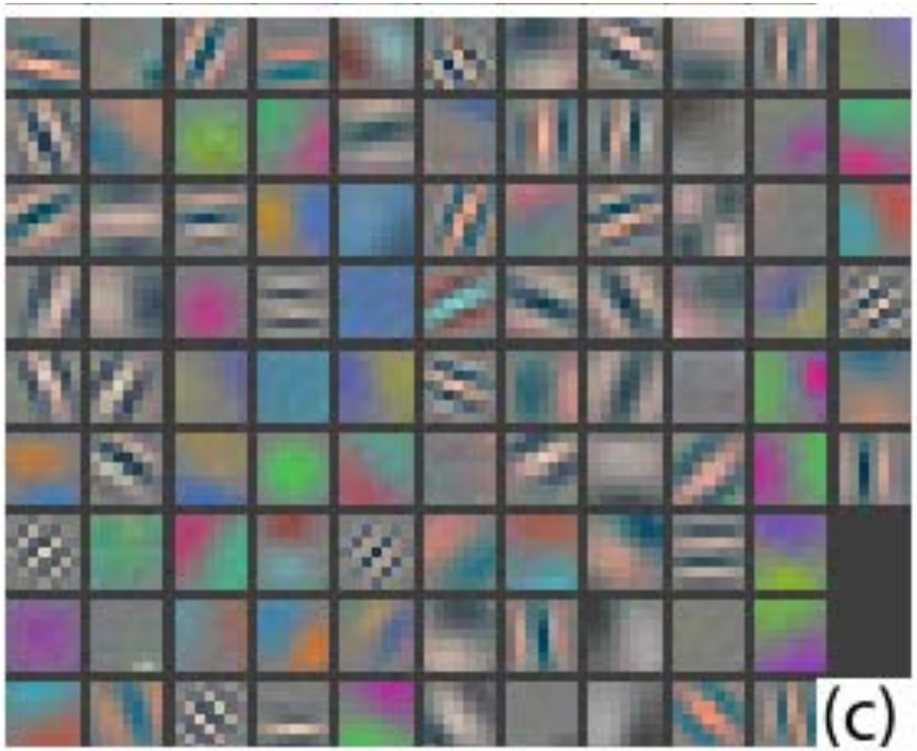


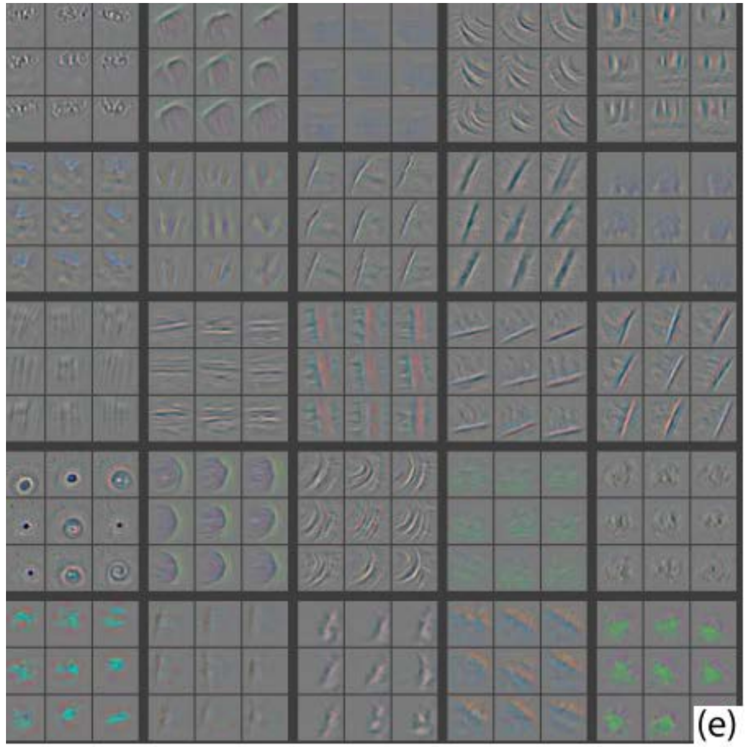
- Succession of convolutional and pooling layers.
- Fully connected layers at the end.

—> Will be discussed in more detail in the next lecture.

EPFL

A Partial Explanation?

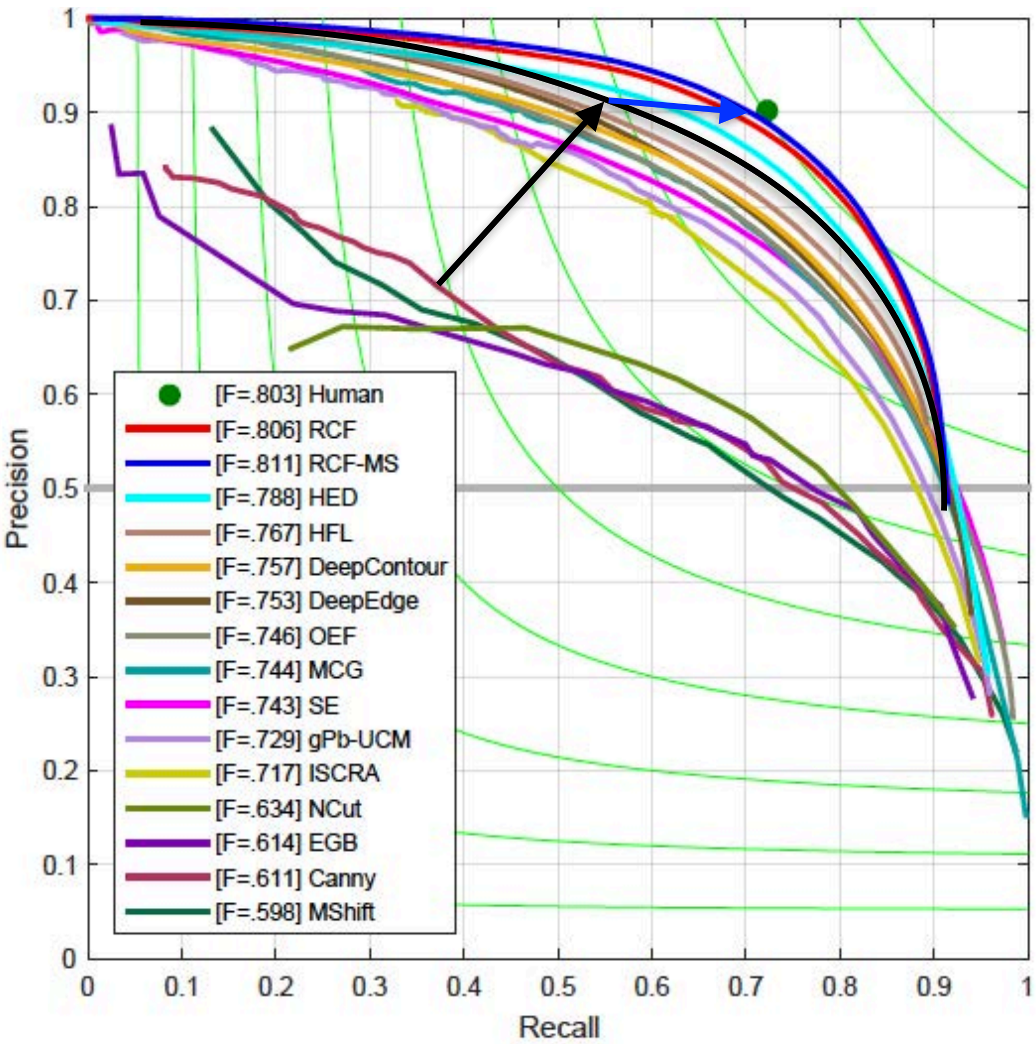




First and second layer features of a Convolutional Neural Net:

- They can be understood as performing multiscale filtering.
- The weights and thresholds are chosen by the optimization procedure.

50 Years Of Edge Detection



- Convolution operators respond to steep smooth shading.
- Parametric matchers tend to reject non ideal edges.
- Arbitrary thresholds and scale sizes are required.
- Learning-based methods need exhaustive databases.

- There still is work to go from contours to objects.

Canny, PAMI'86 —> Sironi et al. PAMI'15 Sironi et al. PAMI'15 —> Liu et al. , CVPR'17

Let us talk about deep networks.