

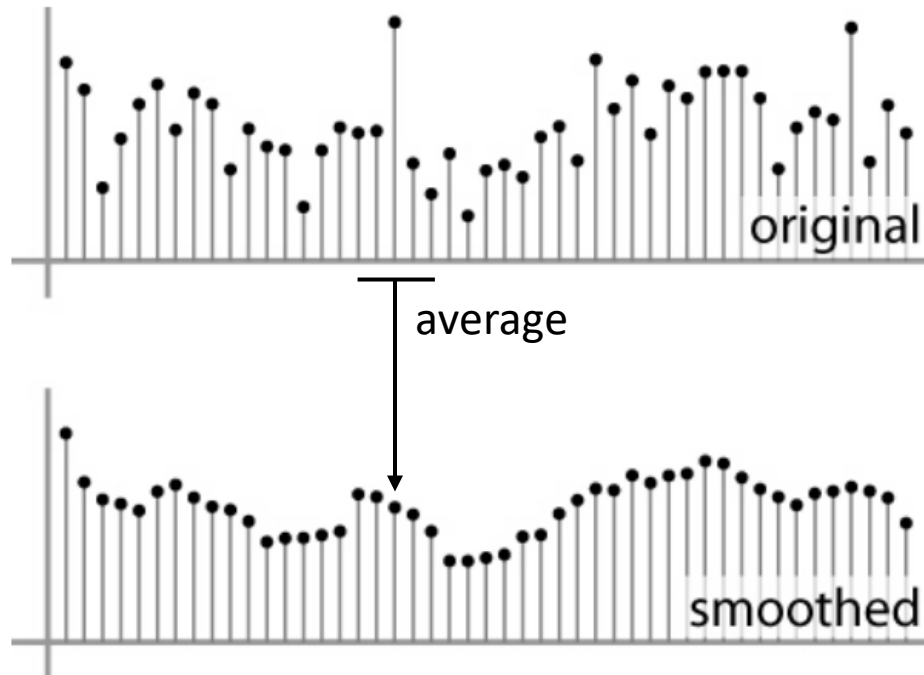
Local Filtering and Edge Detection

Alessandro Giusti

Local filtering

Simple filtering in 1D: moving average

Replace every value with the average of the pixels in its neighborhood



$$b_{\text{smooth}}[i] = \frac{1}{2r + 1} \sum_{j=i-r}^{i+r} b[j]$$

Convolution

A convolution is a **weighted moving average**

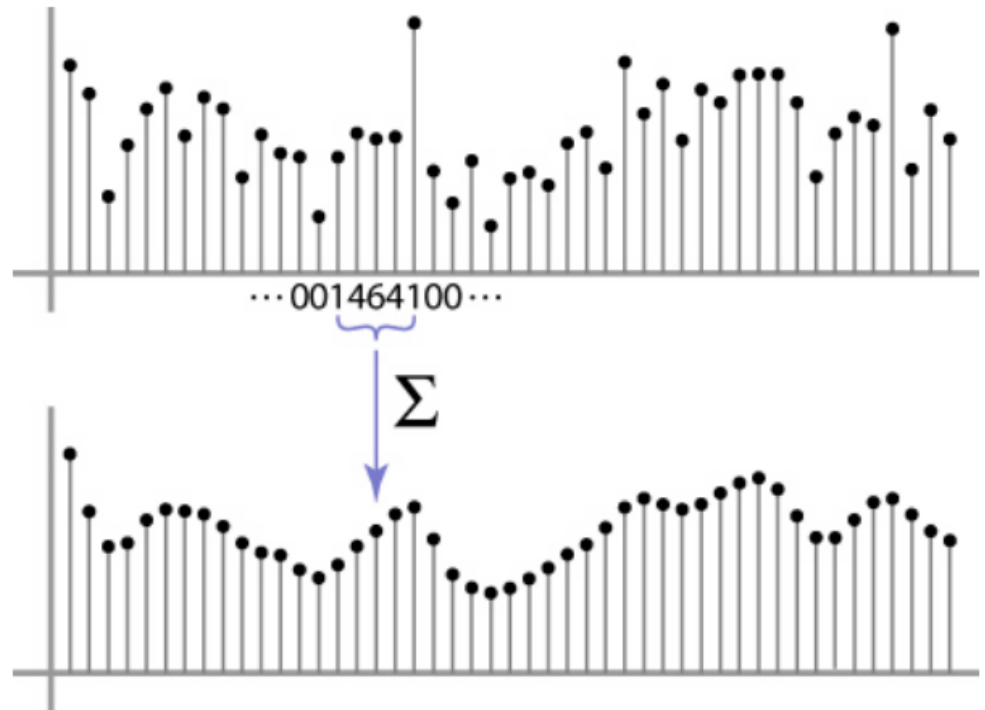
$$(a \star b)[i] = \sum_j a[j]b[i - j]$$

The sequence of weights $a[j]$ is called filter or convolution kernel.

1D convolution example

Using a bell-shaped kernel

$$a = [1, 4, 6, 4, 1] / 16$$

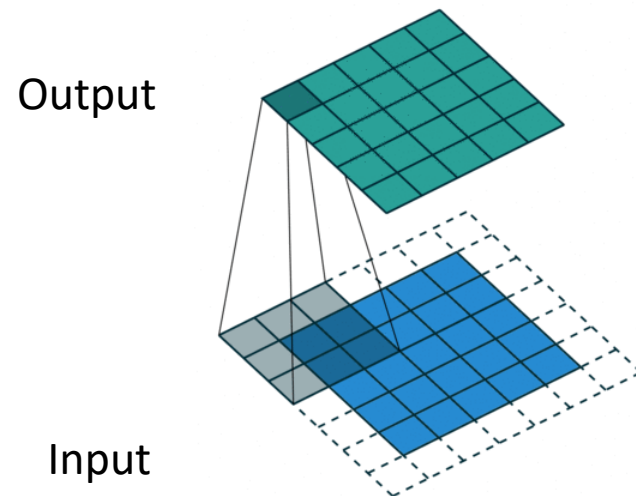


2D convolution

Same as 1D, with one more index

$$(a \star b)[i, j] = \sum_{i', j'} a[i', j'] b[i - i', j - j']$$

Now the filter is a rectangle you slide around over a grid of numbers



Example

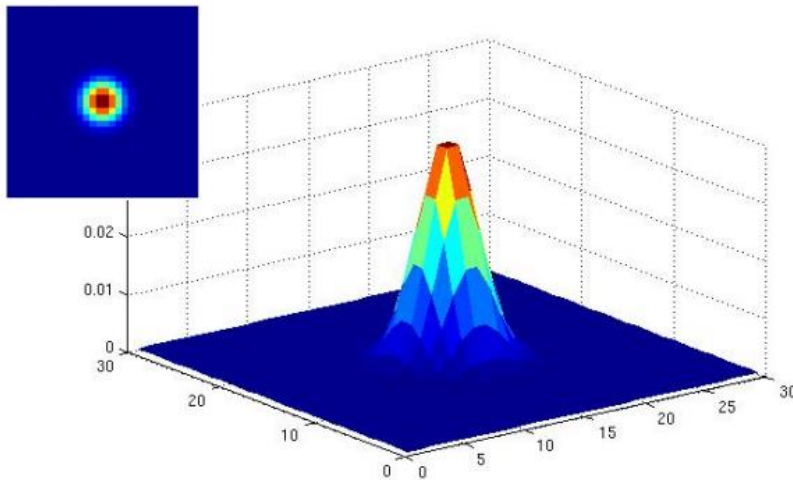
[illegible][illegible]

Example

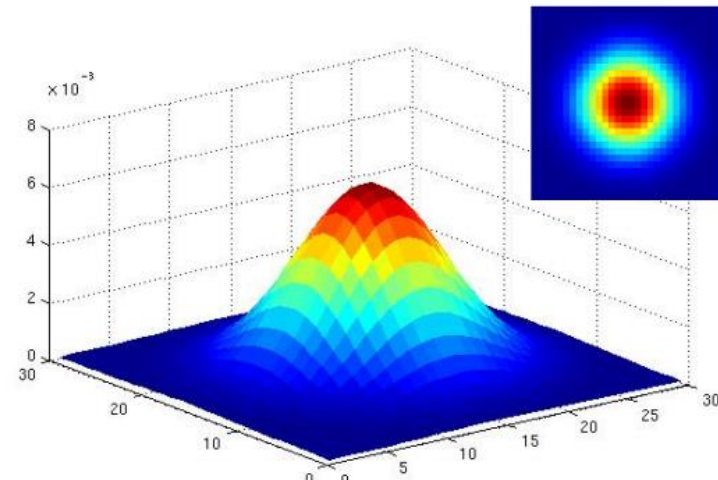
[illegible][illegible]

Smoothing with gaussian filters

- Variance (sigma) determines the amount of smoothing



$\sigma = 2$ with
 30×30 kernel



$\sigma = 5$ with
 30×30 kernel

Example



Original

0	0	0
0	1	0
0	0	0

Kernel

Example



Original

0	0	0
0	1	0
0	0	0

Kernel



same as original

Example



Original

$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1

Kernel

Example

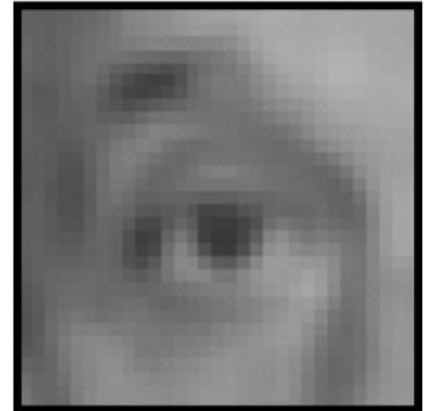


Original

$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1

Kernel



smoothed

Example



Original

0	0	0
0	2	0
0	0	0

—

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

Kernel

Example



Original

0	0	0
0	2	0
0	0	0

-

$\frac{1}{9}$

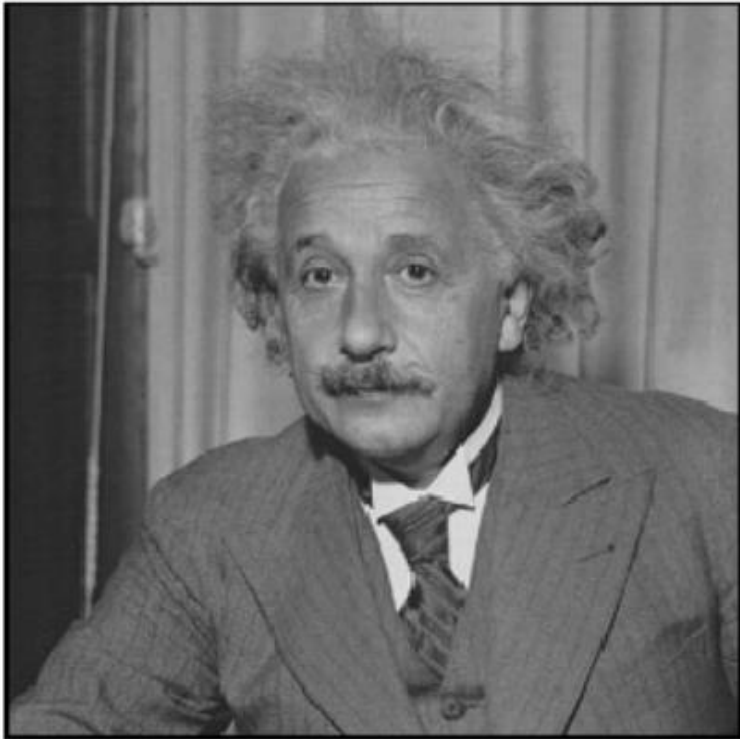
1	1	1
1	1	1
1	1	1

Kernel

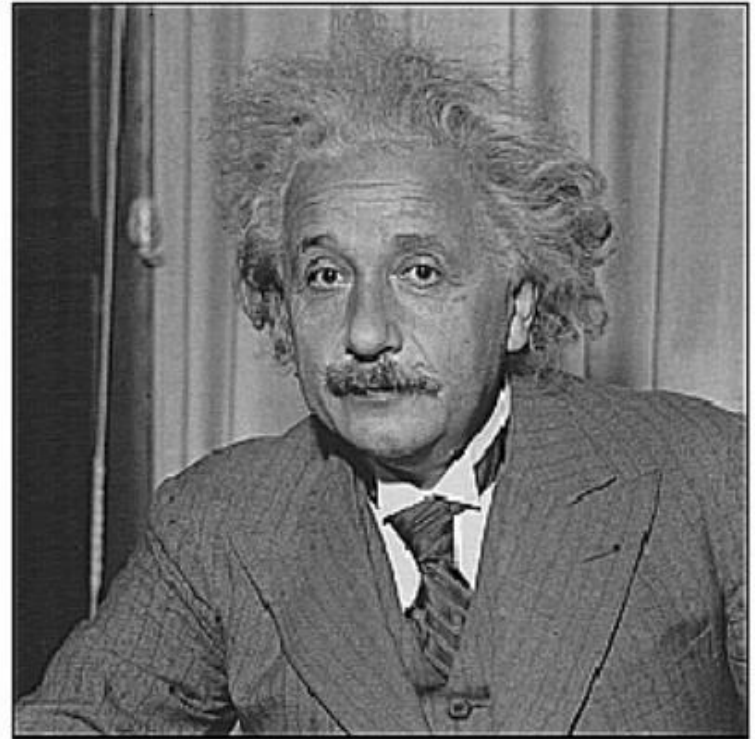


sharpened

Sharpening



before



after

Why does sharpening work?



−



=



+



=



Edge detection

Edges are important features when we look at images!

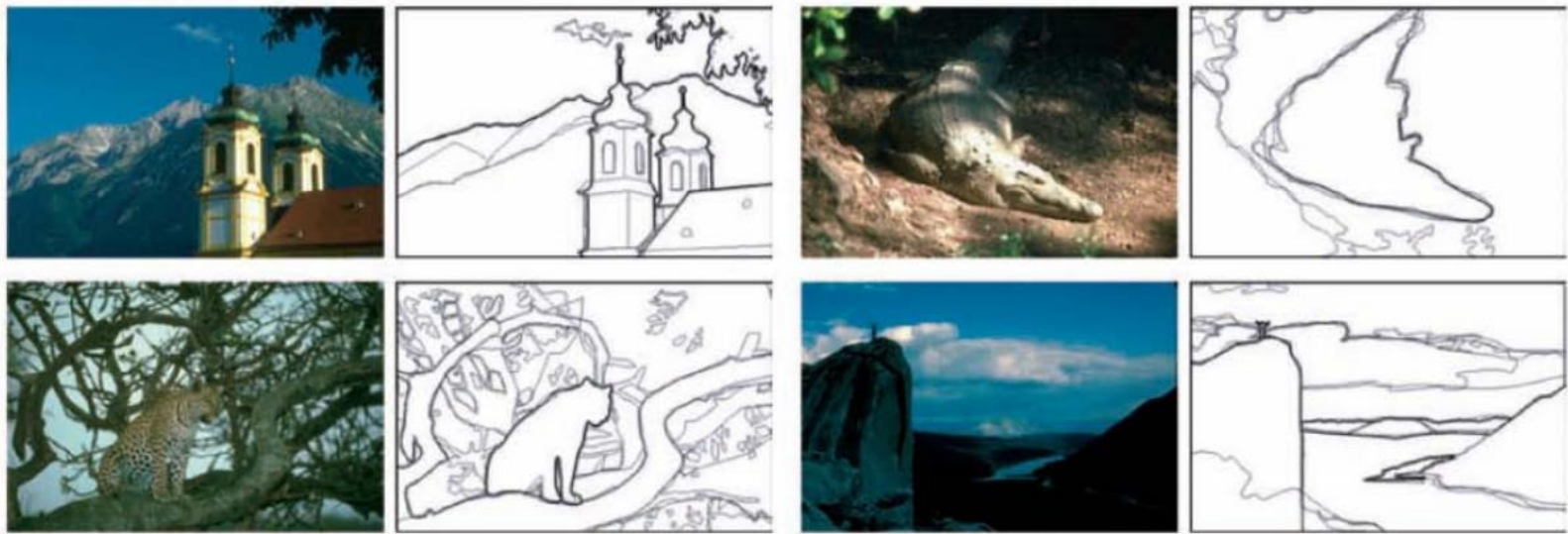
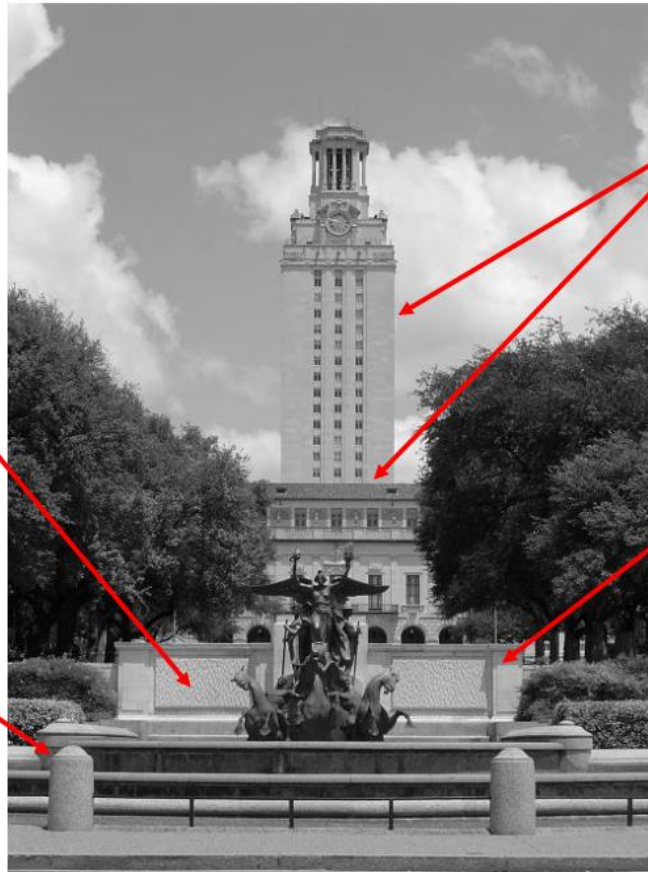


Figure 4.31 Human boundary detection (Martin, Fowlkes, and Malik 2004) © 2004 IEEE. The darkness of the edges corresponds to how many human subjects marked an object boundary at that location.

Causes of edges

Reflectance change:
appearance
information, texture

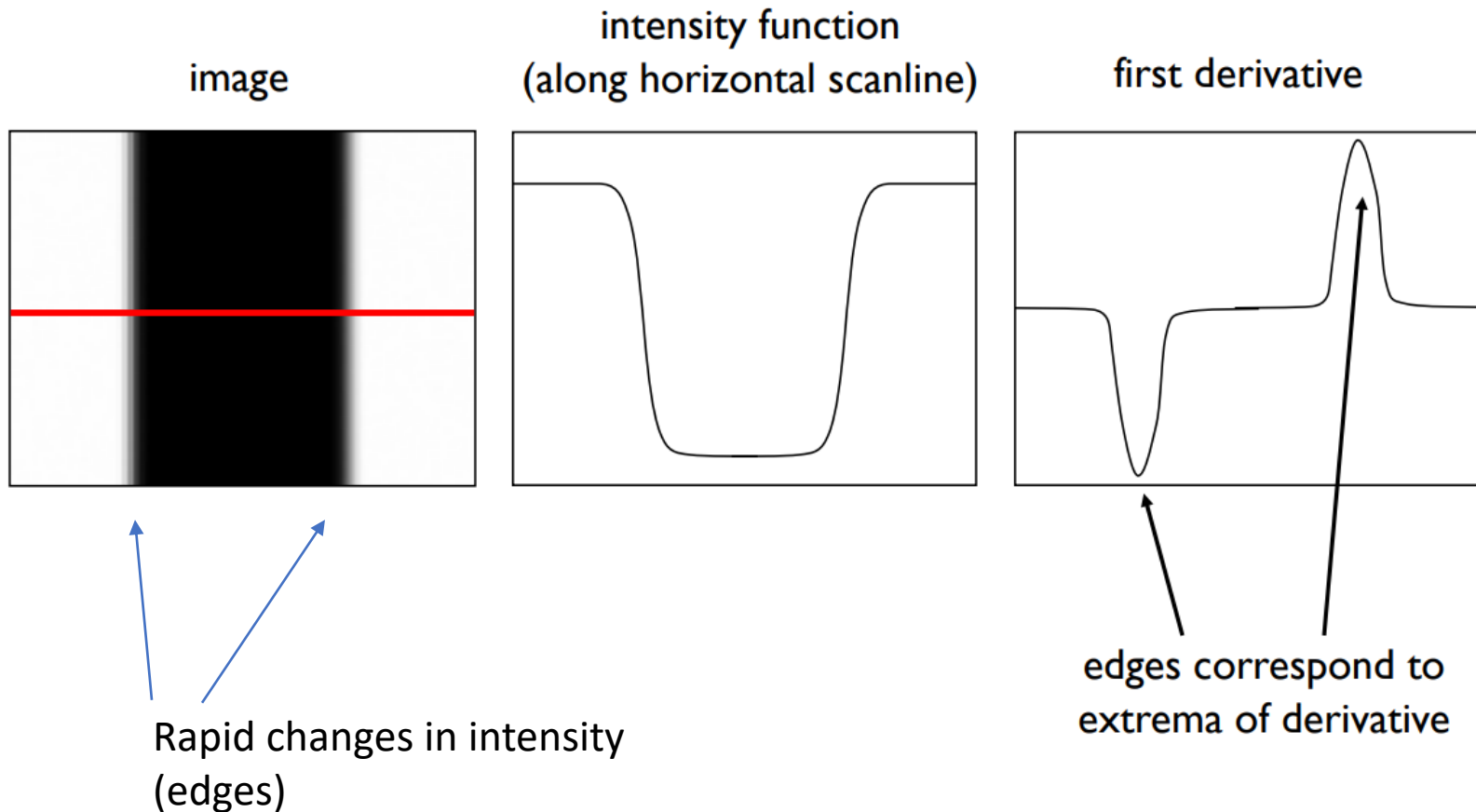
Change in surface
orientation: shape



Depth discontinuity:
object boundary

Cast shadows

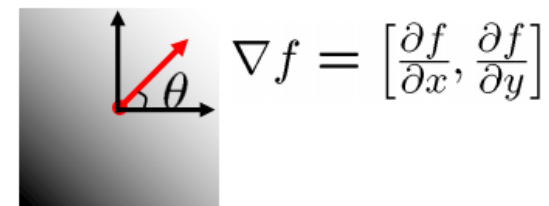
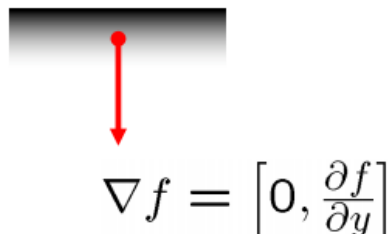
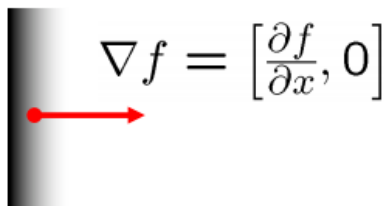
What is an edge?



In 2D...

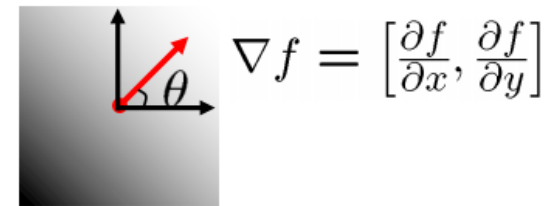
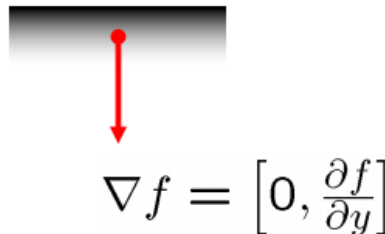
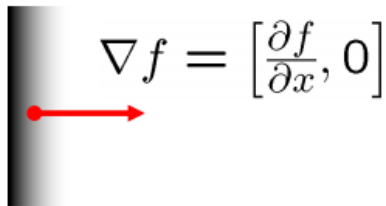
- In 2D, the derivative corresponds to the **gradient**
- The gradient is a vector defined for every point in the image
- It points to the direction of most rapid increase of the intensity

The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$



Gradient direction and strength

The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$



- Gradient direction (orthogonal to the edge):

$$\theta = \tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$$

- Gradient magnitude (strength of the edge):

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2}$$

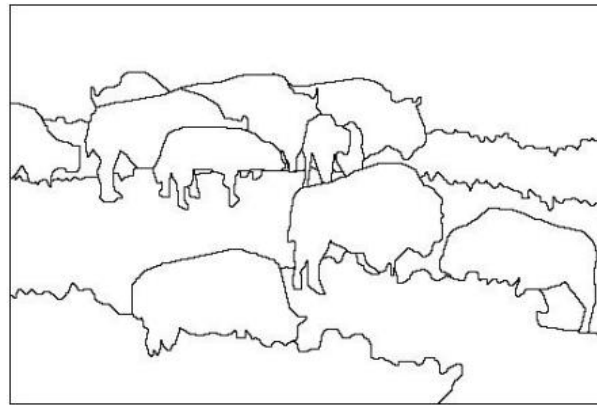
Keep in mind:

- Not all important edges have strong gradients
- Not all strong gradients are important edges

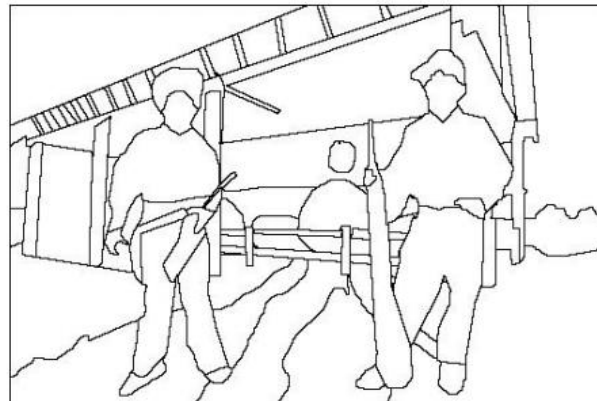
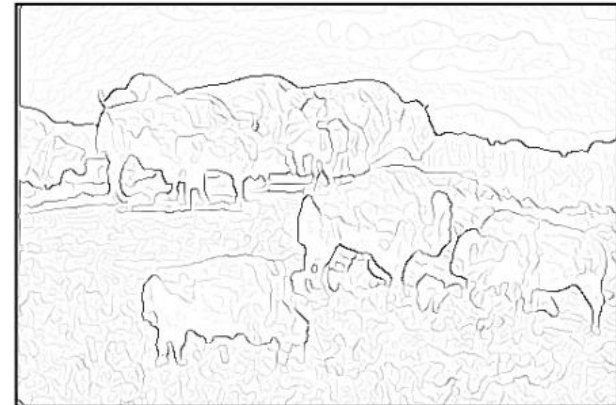
image



human segmentation



gradient magnitude



Computing the gradient on an image

- How do we approximate $\frac{\partial f}{\partial x}$?
- How do we approximate $\frac{\partial f}{\partial y}$?

Computing the gradient on an image

- How do we approximate $\frac{\partial f}{\partial x}$? convolve with

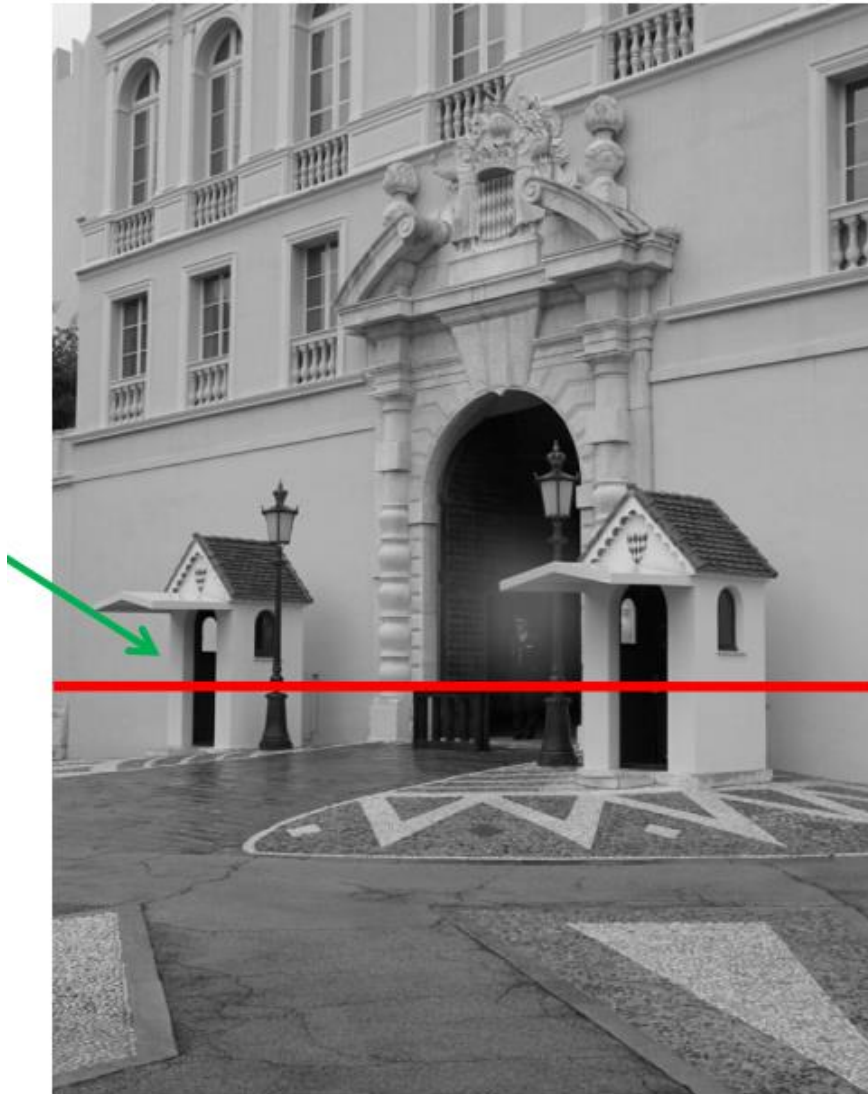
-1	1
----	---

- How do we approximate $\frac{\partial f}{\partial y}$? convolve with

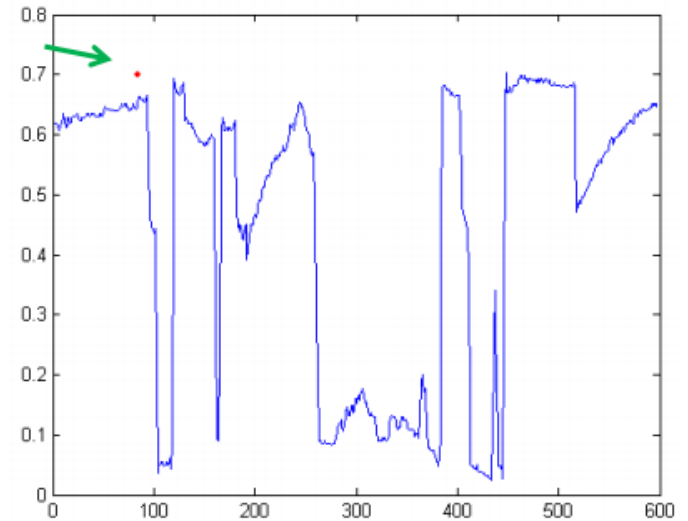
-1
1

Finite differencing!

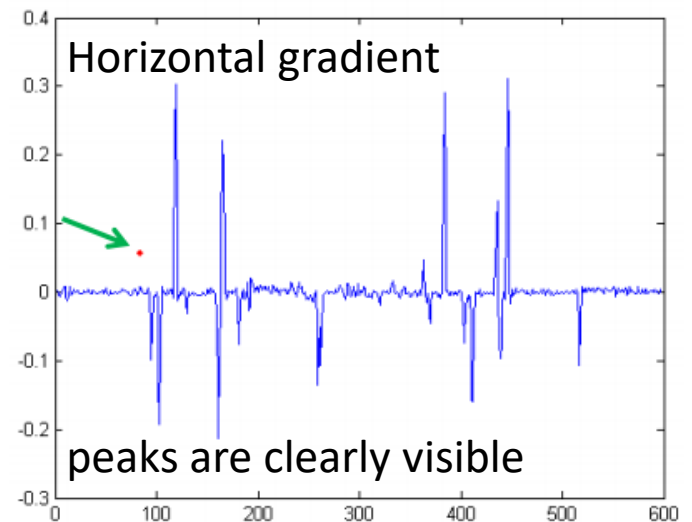
Example of finite differencing



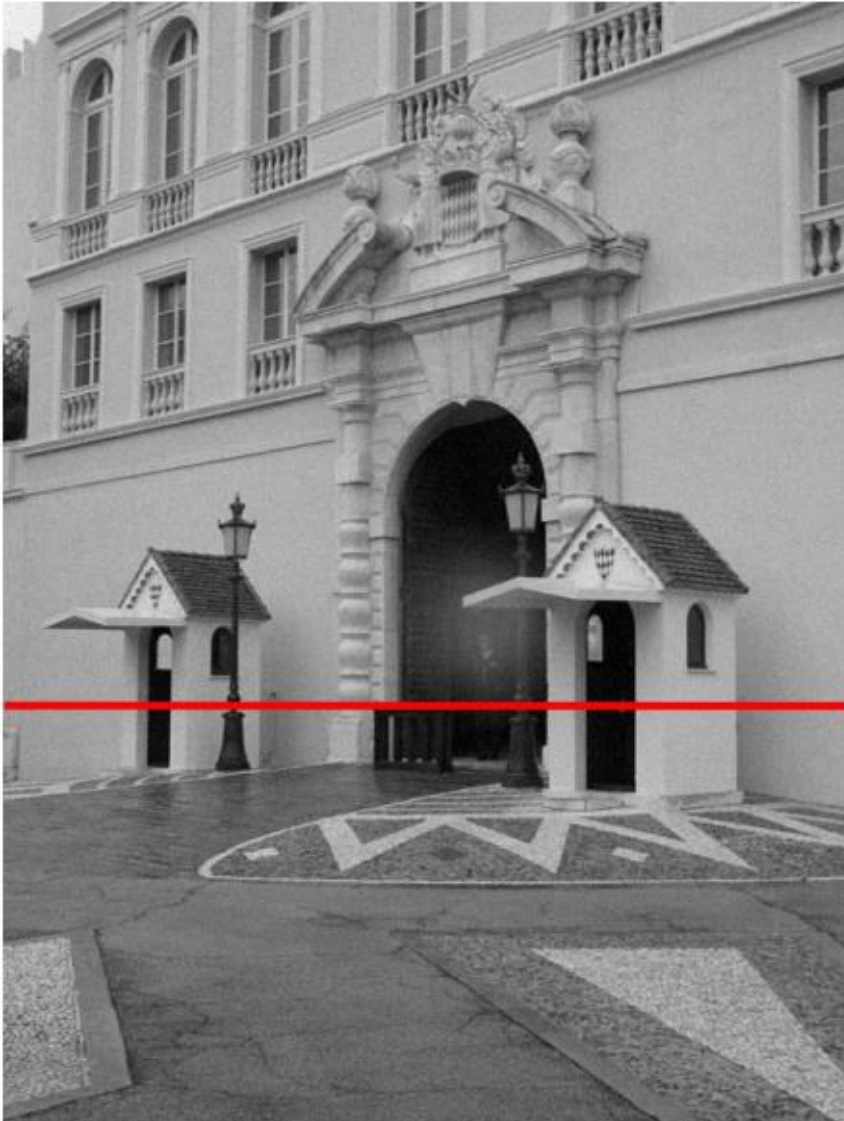
Intensity along profile



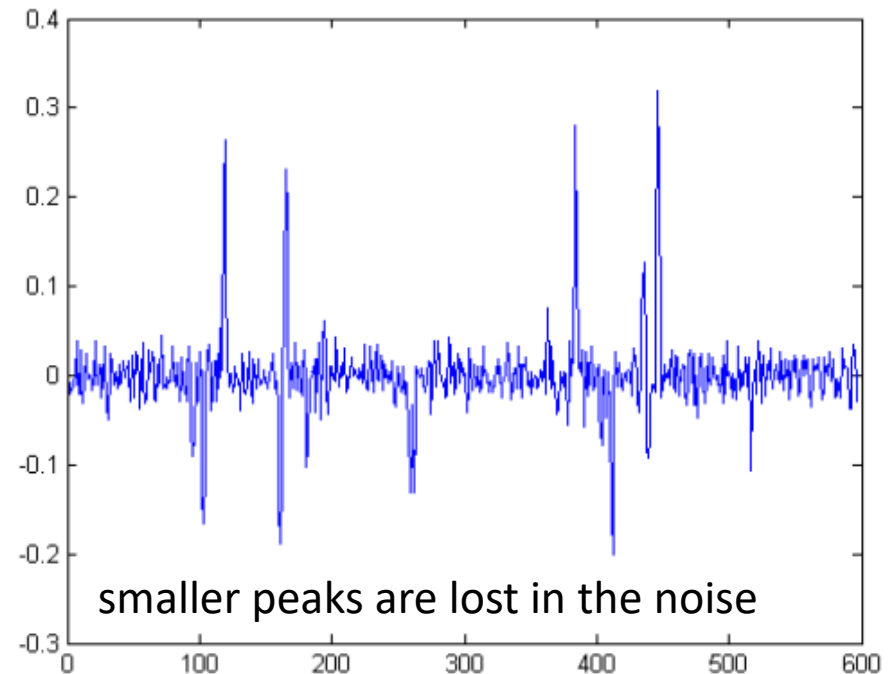
Horizontal gradient



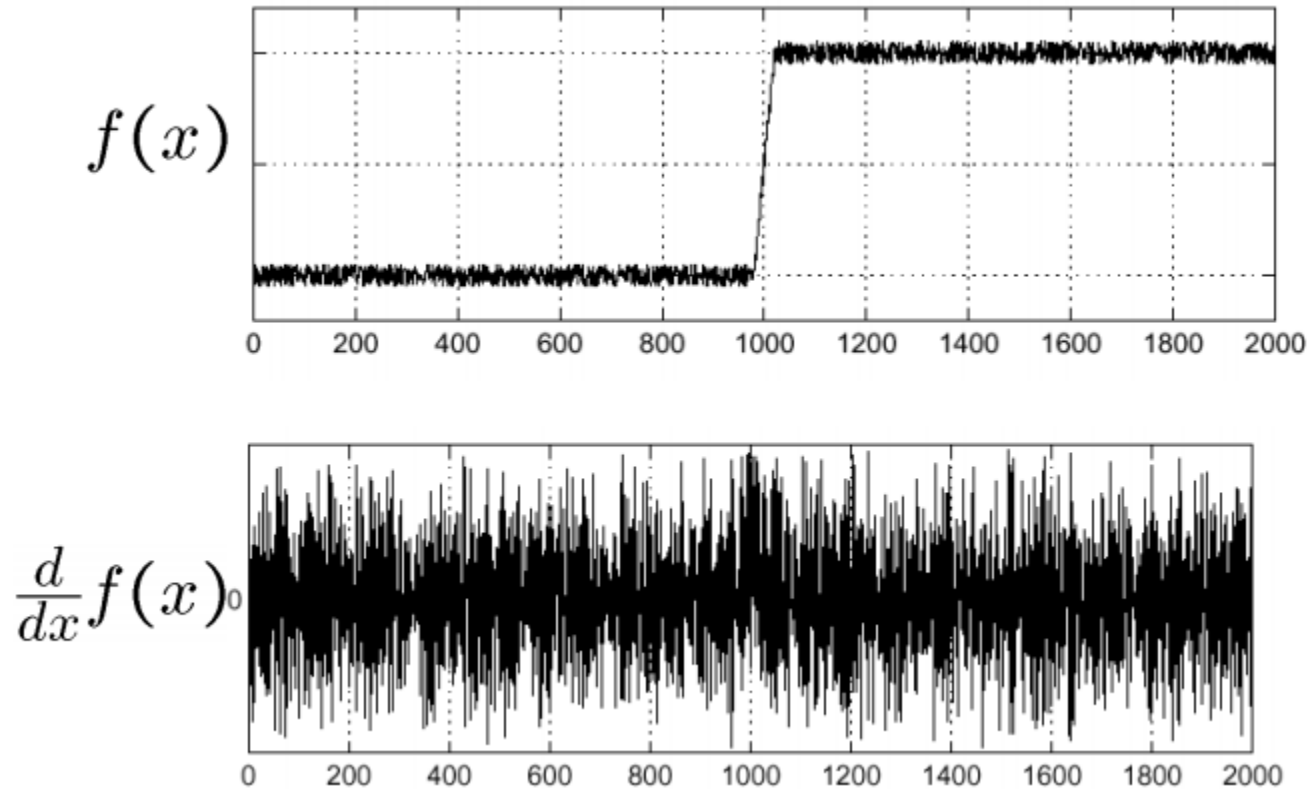
If we add a tiny amount of noise to the image...



Horizontal gradient

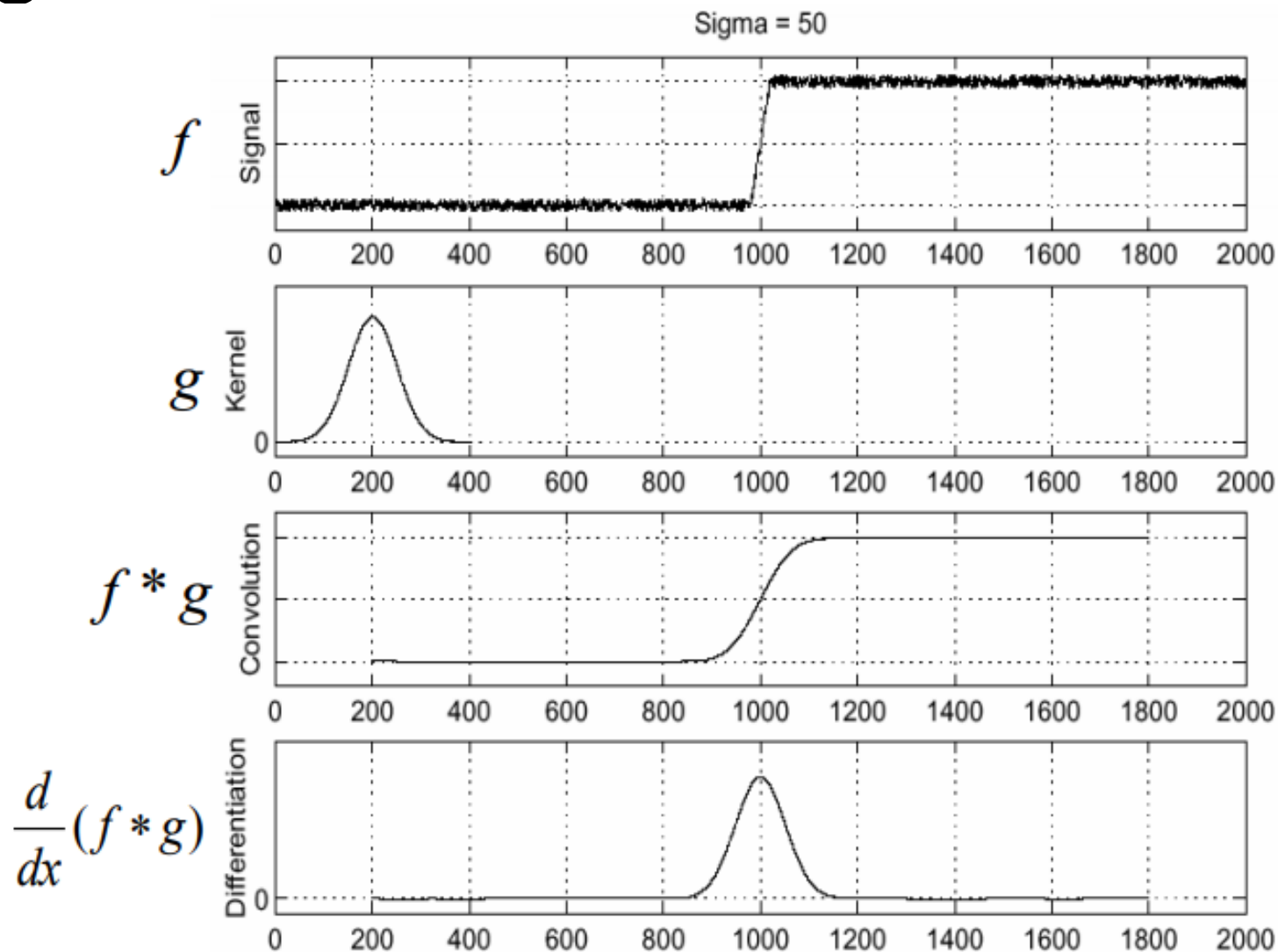


Another 1D example

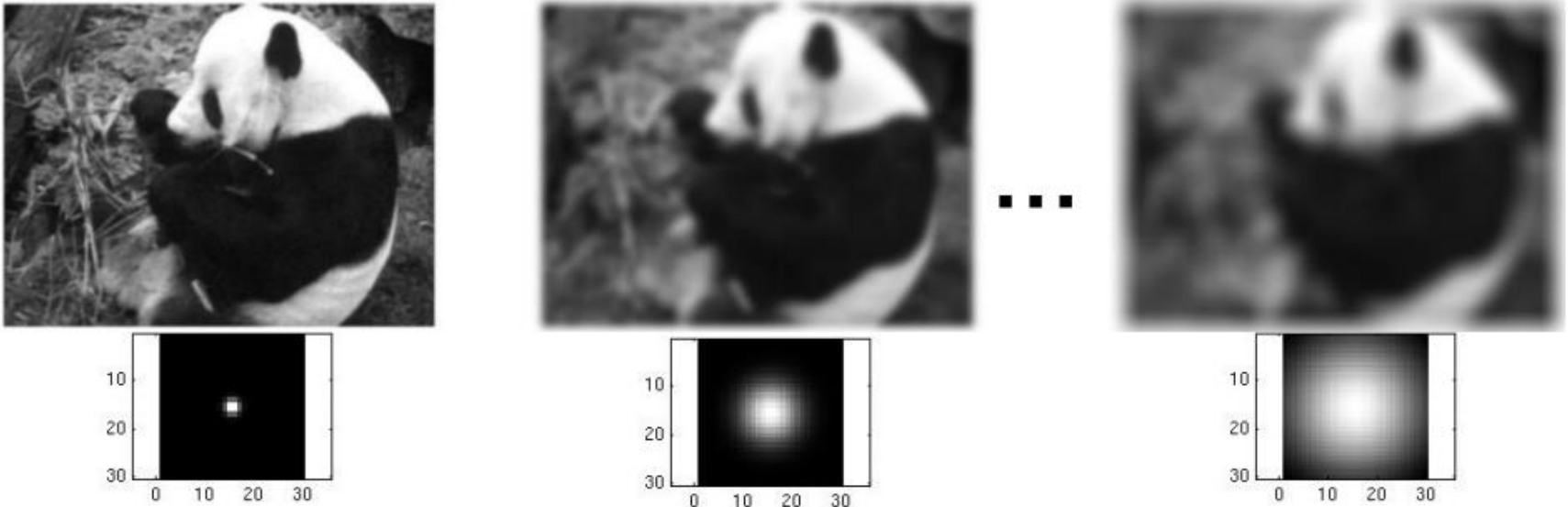


Where is the edge?!?

Solution: smooth, then compute gradient

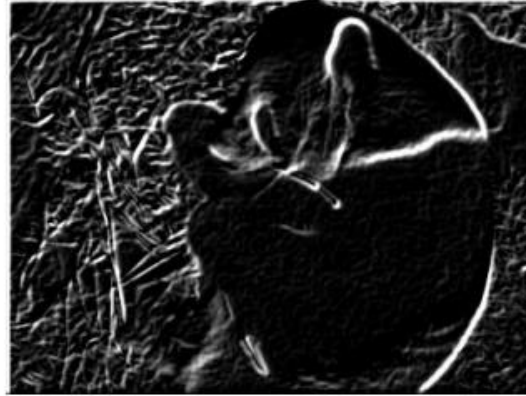


Different amounts of smoothing...

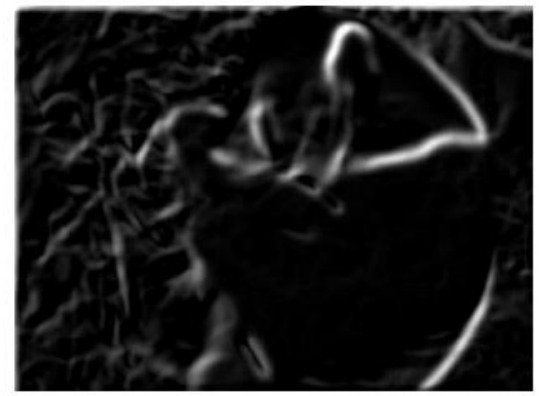


Amount of smoothing depends on the value of sigma
(width of the gaussian)

.... show different structures in the gradients



$\sigma = 1$ pixel



$\sigma = 3$ pixels

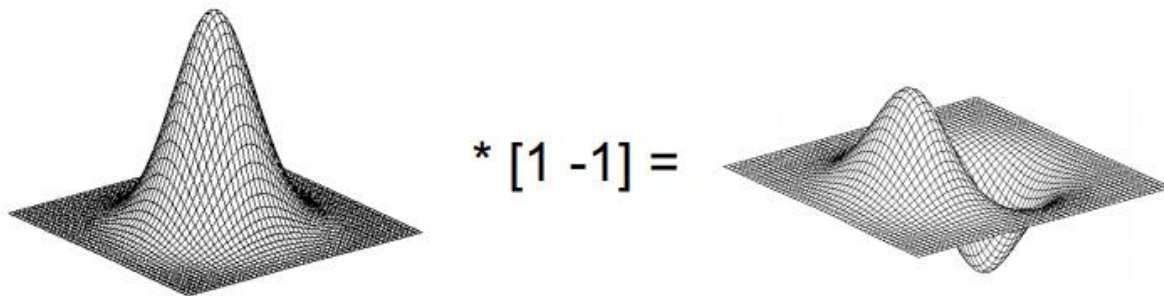
Larger values of sigma: larger scale edges detected

Smaller values of sigma: finer details detected

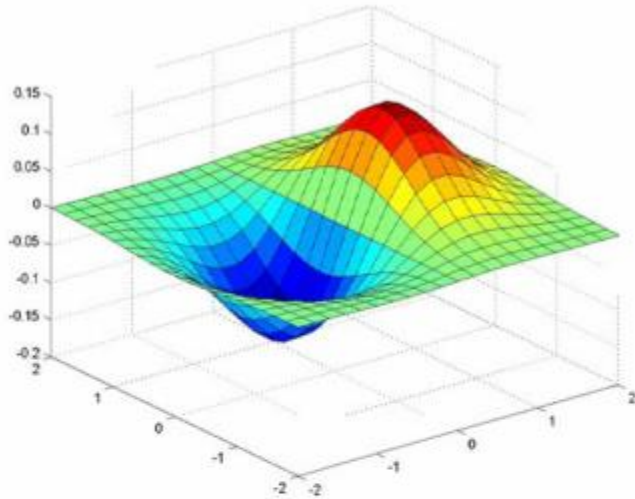
In practice

To compute gradients, convolve with a derivative-of-gaussian filter.

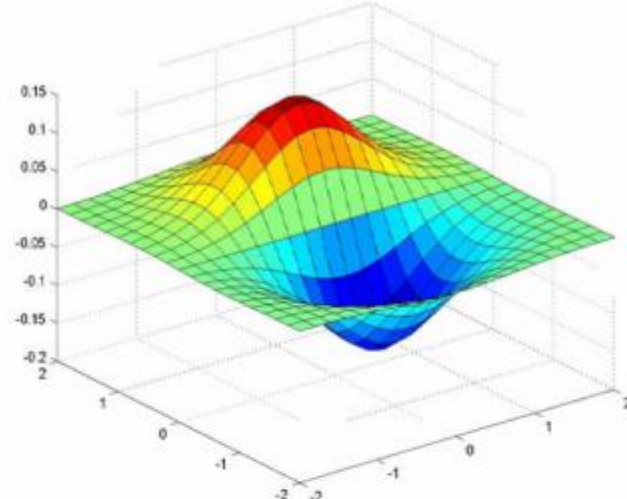
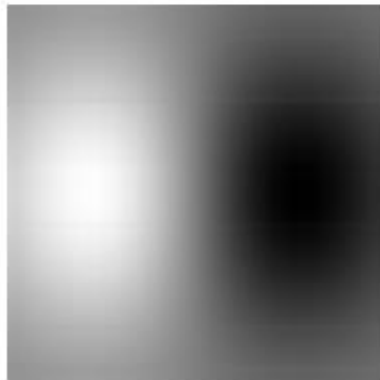
This is equivalent to smoothing with a gaussian and then taking the derivative.



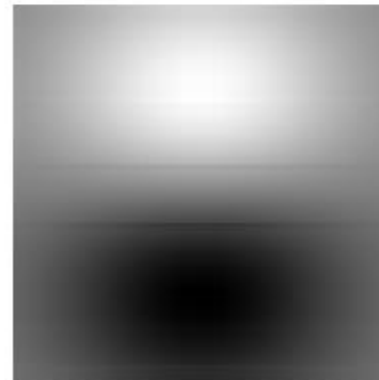
Derivative-of-gaussian filters



x-direction



y-direction



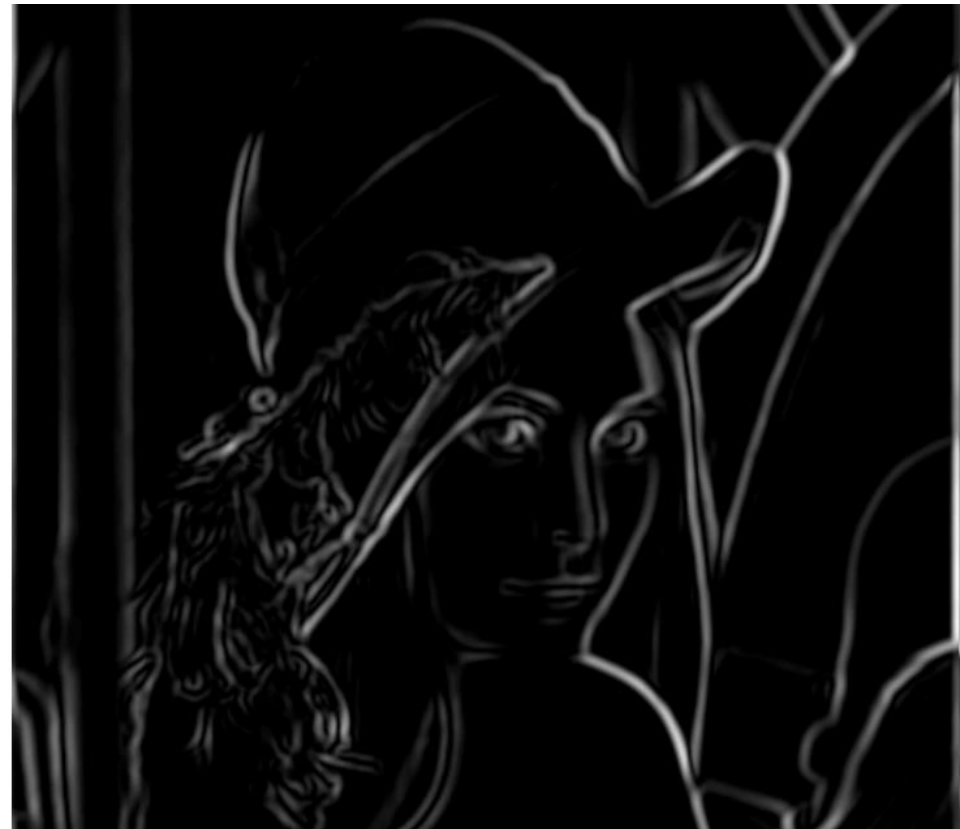
Notes

- Gaussian smoothing filter
 - removes “high-frequency” components
 - values sum to one
- Derivative filters
 - contain some negative values
 - values sum to 0
 - yield large responses at points with high contrast (e.g. edges)

Edge detection summary

$$\frac{\partial f}{\partial x} \quad \frac{\partial f}{\partial y}$$

- Approximate gradients along x and y by convolving with derivative-of-gaussian filters
- Compute gradient magnitude as $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$



Canny edge detection algorithm

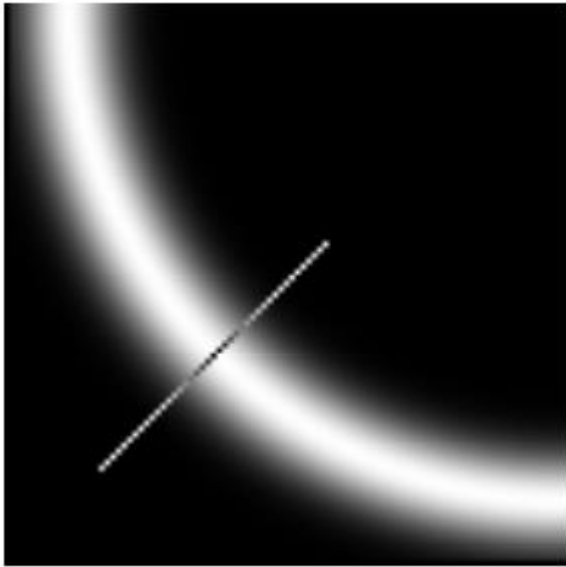
$$\frac{\partial f}{\partial x} \quad \frac{\partial f}{\partial y}$$

- Approximate gradients along x and y by convolving with derivative-of-gaussian filters
- Compute gradient magnitude as $\|\nabla f\| = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2}$
- Make edges 1-pixel-wide (thinning):
non-maxima suppression along perpendicular direction to edge, i.e. direction of gradient
- Only keep strong edges
hysteresis thresholding

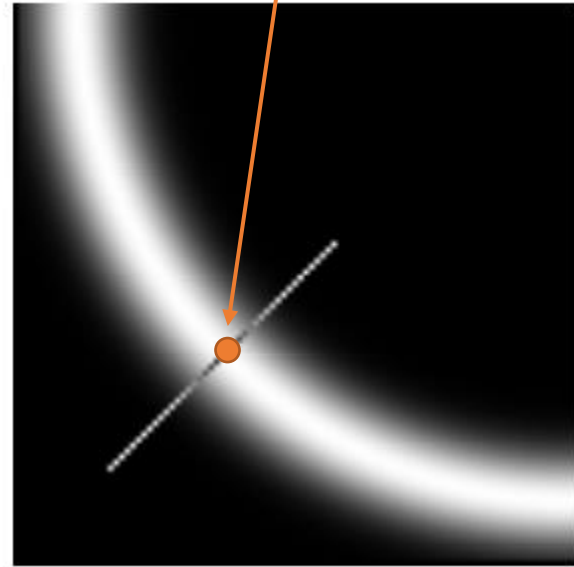
Non-maxima suppression

Intuitively:

of all the possible edge pixels along the line...



keep only **this one**

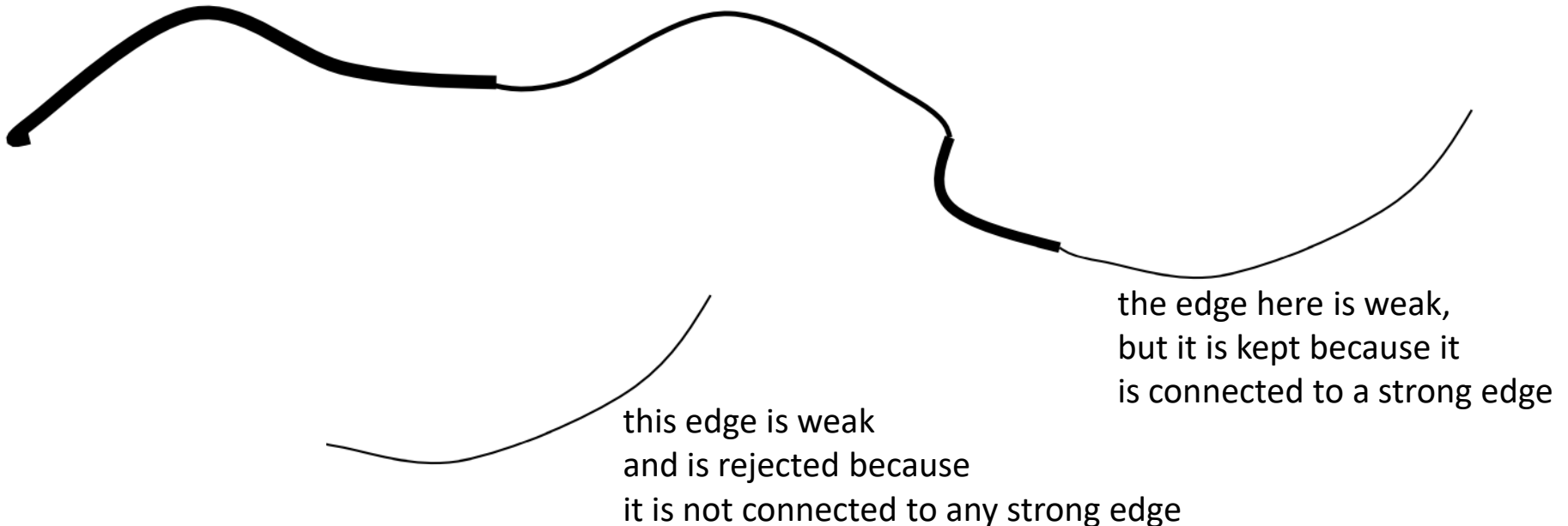


... then repeat along the whole edge. Results in thinning the edge to a 1-pixel width

Hysteresis thresholding

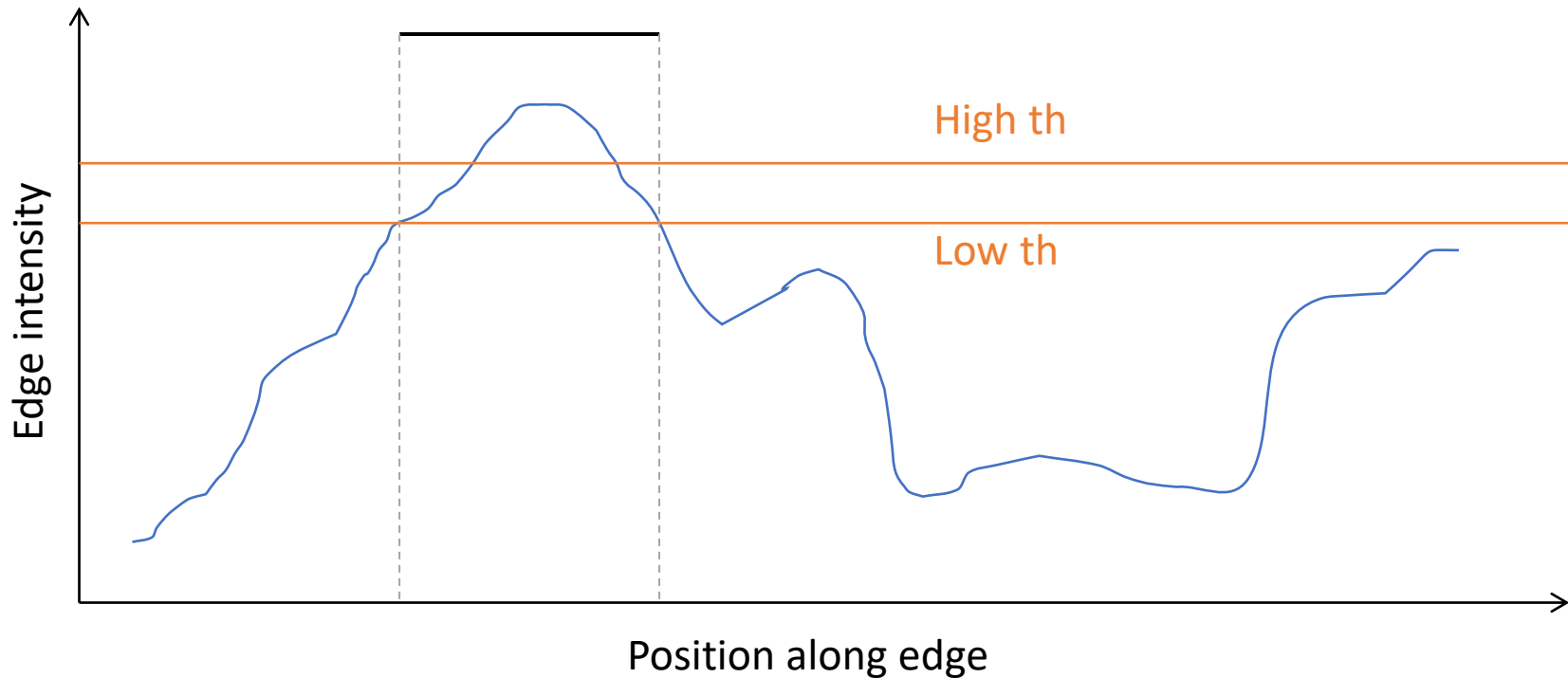
Intuitively:

Use a high threshold to start curves and a low threshold to continue them



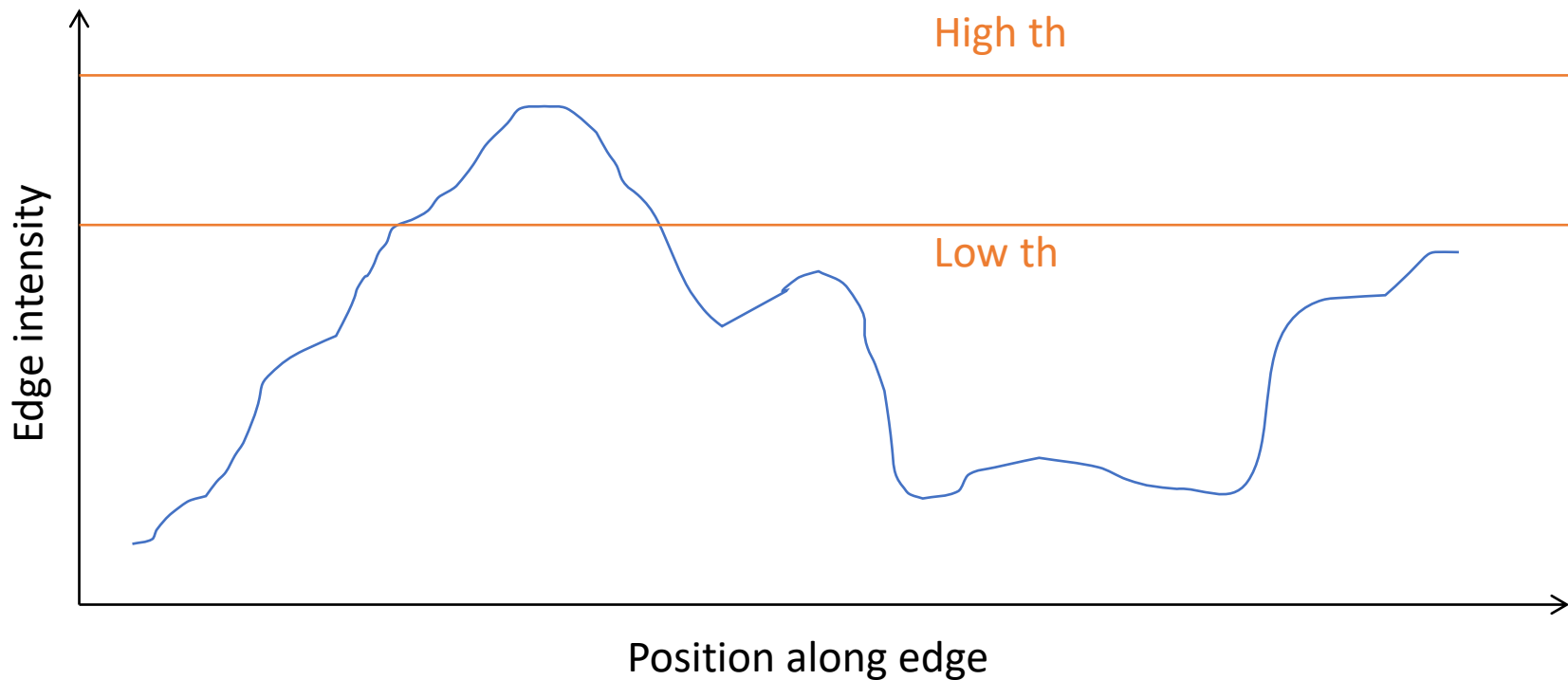
Hysteresis thresholding

Quiz: which parts of the edge would survive?



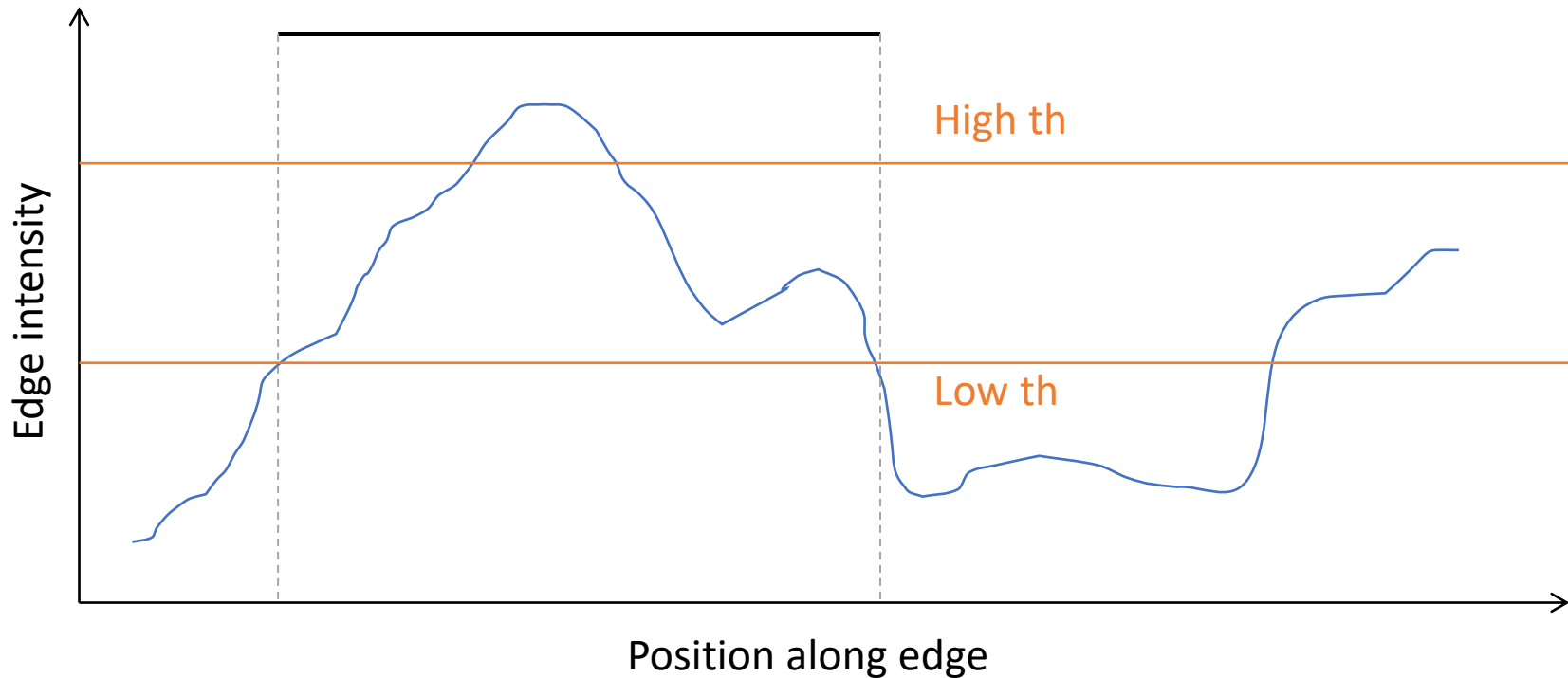
Hysteresis thresholding

Quiz: which parts of the edge would survive? None!



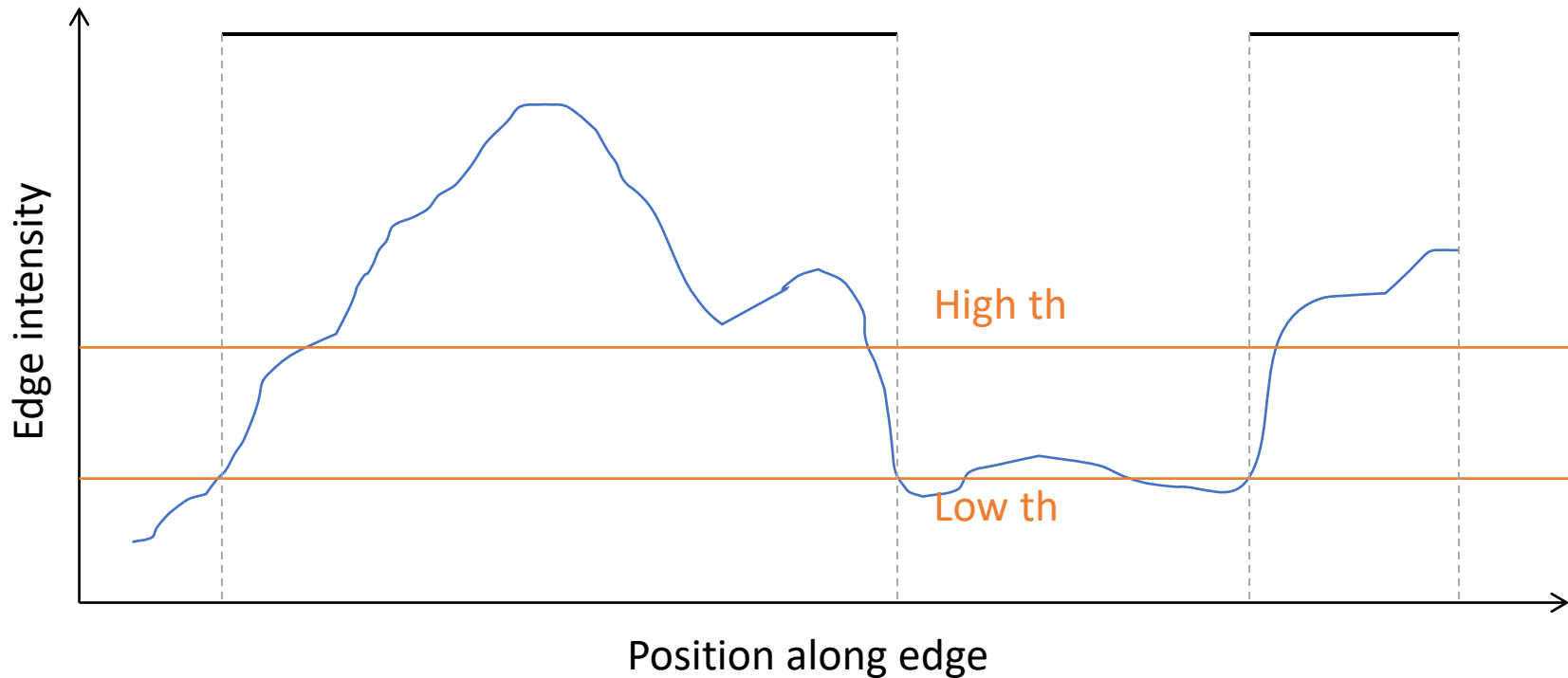
Hysteresis thresholding

Quiz: which parts of the edge would survive?



Hysteresis thresholding

Quiz: which parts of the edge would survive?





original image



high threshold
(strong edges)



low threshold
(weak edges)



hysteresis threshold



high threshold
(strong edges)



low threshold
(weak edges)



hysteresis threshold



original image



high threshold
(strong edges)



low threshold
(weak edges)



hysteresis threshold

Effect of gaussian sigma



original



Canny with $\sigma = 1$



Canny with $\sigma = 2$

canny

```
skimage.feature.canny(image, sigma=1.0, low_threshold=None, high_threshold=None,  
mask=None, use_quantiles=False) \[source\]
```

Edge filter an image using the Canny algorithm.

Parameters:

image : 2D array

Grayscale input image to detect edges on; can be of any dtype.

sigma : float

Standard deviation of the Gaussian filter.

low_threshold : float

Lower bound for hysteresis thresholding (linking edges). If None, low_threshold is set to 10% of dtype's max.

high_threshold : float

Upper bound for hysteresis thresholding (linking edges). If None, high_threshold is set to 20% of dtype's max.

mask : array, dtype=bool, optional

Mask to limit the application of Canny to a certain area.

use_quantiles : bool, optional

If True then treat low_threshold and high_threshold as quantiles of the edge magnitude image, rather than absolute edge magnitude values. If True then the thresholds must be in the range [0, 1].

Returns:

output : 2D array (image)

The binary edge map.