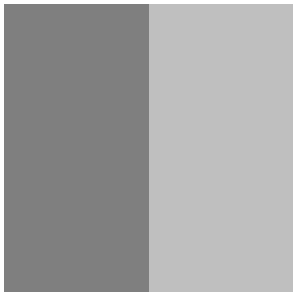
Edge detection

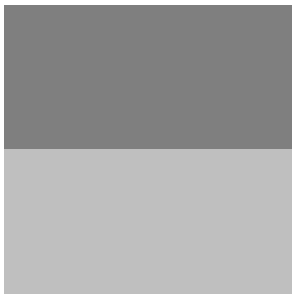
Edges

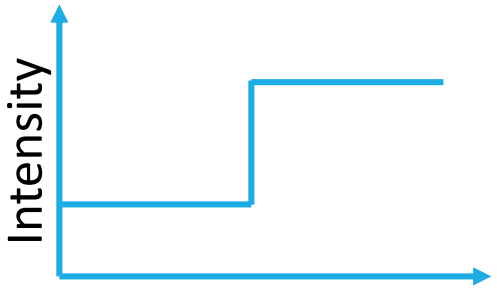
- Edges
  - Define the shape of objects in a scene
  - A set of connected pixels that lie on the boundary between two regions
  - Correspond to large variations in the values of the pixels
- Edge detection
  - One of the most used operations in image segmentation
    - e.g., shape analysis and recognition
  - Mid-level image processing
  - Local operation to determine intensity changes
  - Estimation of paths on the image dividing areas with different intensity values (segmentation)

Examples of edges

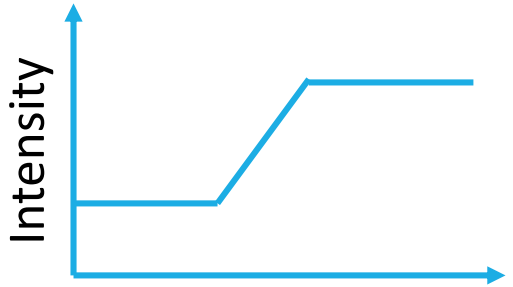
Step edge: computer generated; 1-pixel wide edge



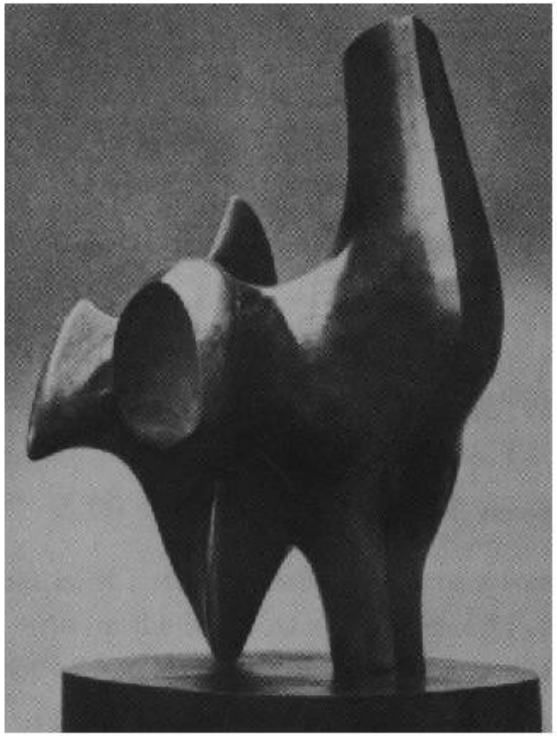


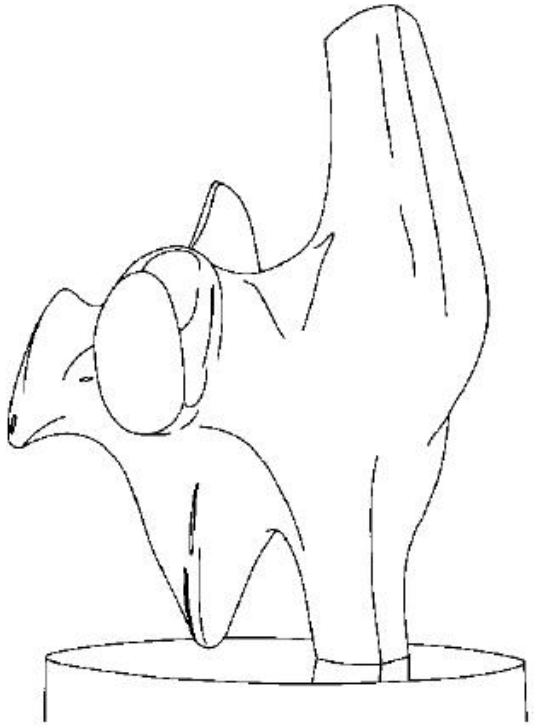


Ramp edge: a ‘thick’ edge



Edge detection





How can you tell which pixel is on an edge?

Edge detection

Strategy

- Find points of large variations in an image.

Requirement

- A good edge detector must understand the difference between
  - Variations caused by image noise.
  - Variations caused by textures in the objects.
  - Variations caused by edges of objects.

Results of edge detection

- Often presented as a subset of image pixels representing its edges-shown
  - As a 2-level image or
  - By superimposing the edges with a different color of the original image.

Edge detection

Differential models

- Gradient
- Laplacian

Template methods

- Roberts
- Prewit
- Sobel

Optimization methods

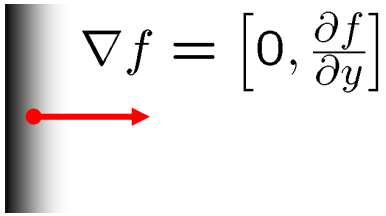
- Based on modification of edges, noise, quality measure of edge detection
- Marr-Hildreth
- Canny

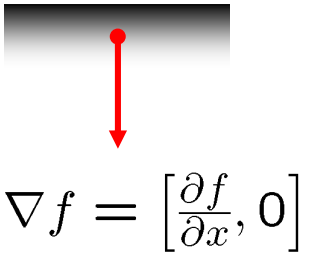
Gradient

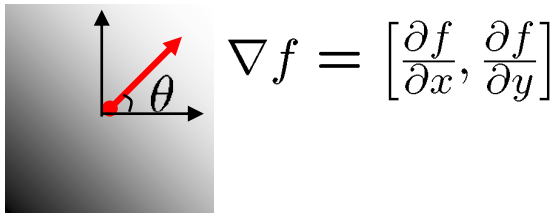
The gradient of an image *f* at point (*x*,*y*):

$$\nabla f(x,y) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix}$$

The gradient points in the direction of the greatest rate of change of *f* at location (*x*,*y*).







Gradient

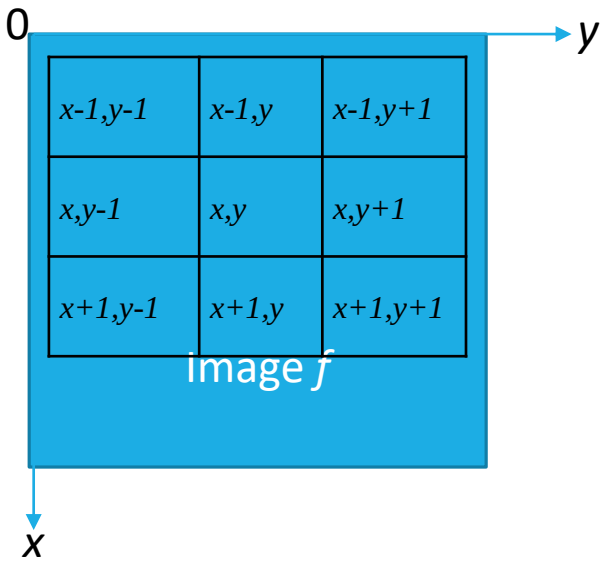
The approximation of the derivatives of *f*:

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

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

Thus, the gradient transforms to:

$$\nabla f(x,y) = \begin{bmatrix} f(x+1,y) - f(x,y) \\ f(x,y+1) - f(x,y) \end{bmatrix}$$



Gradient using pixel difference

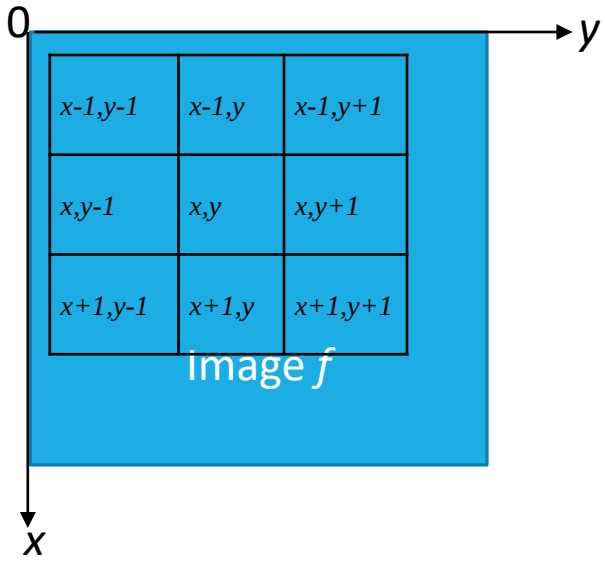
$$\nabla f(x,y) = \begin{bmatrix} f(x+1,y) - f(x,y) \\ \\ f(x,y+1) - f(x,y) \end{bmatrix} = \begin{bmatrix} Mask_x \circ f(x,y) \\ \\ Mask_y \circ f(x,y) \end{bmatrix}$$

convolution

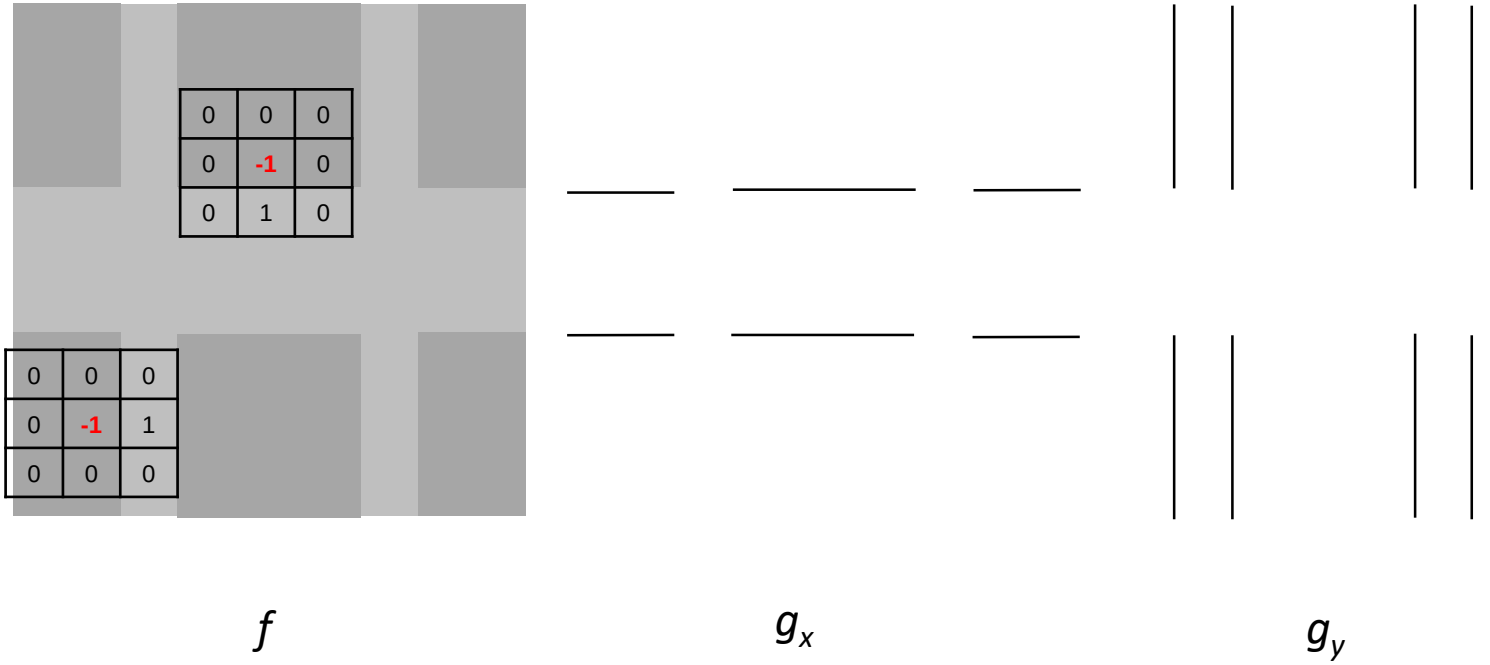
Where

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

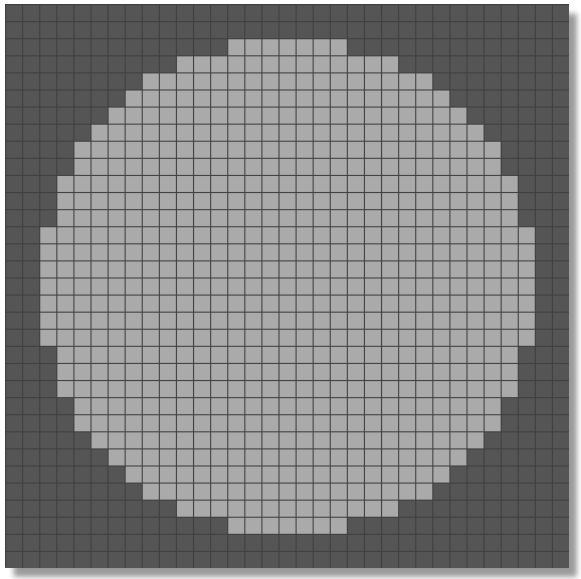
$$Mask_y = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$



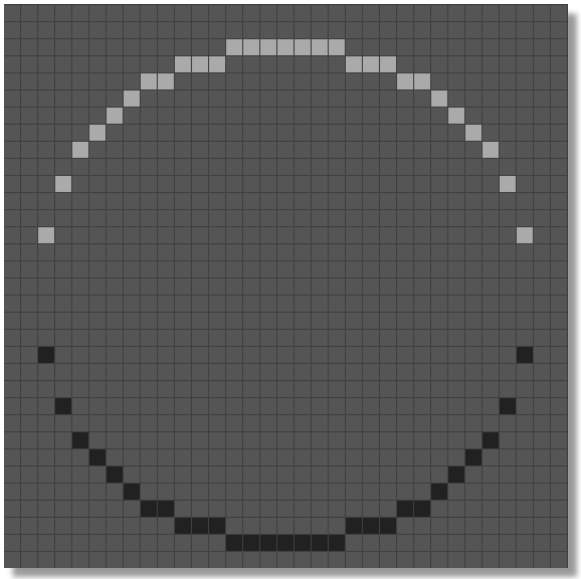
Vertical and horizontal edges



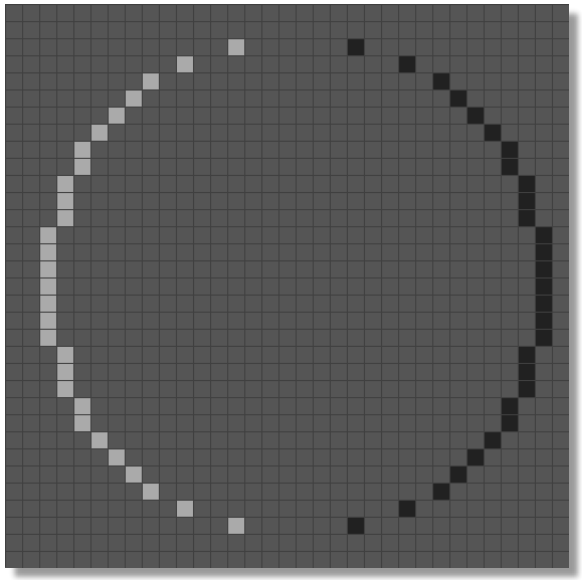
Vertical and horizontal edges



original: f



g_x

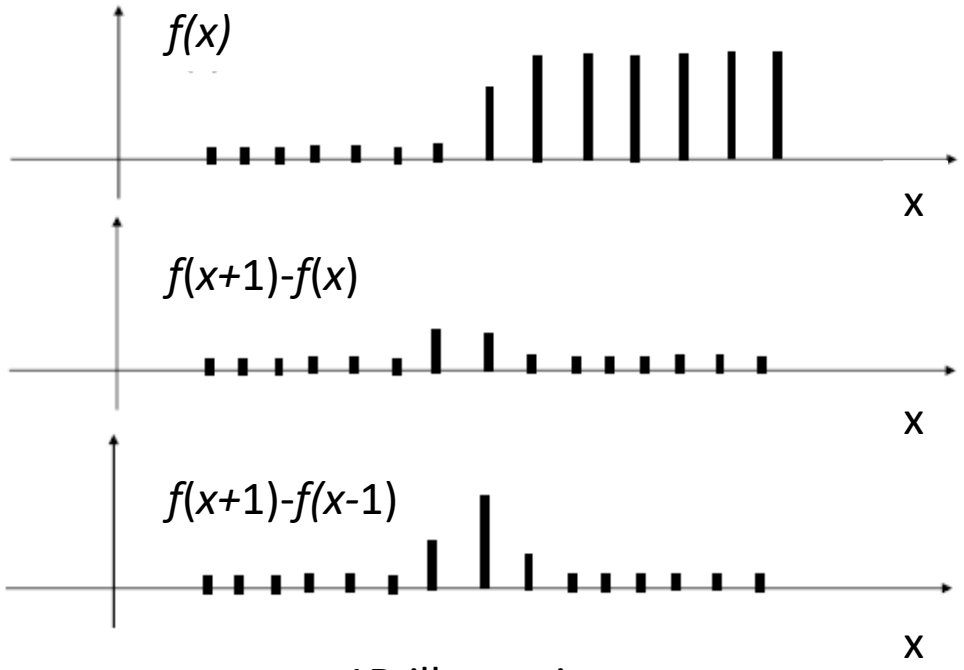


g_y

Gradient using symmetric differences

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

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



1D illustration

Gradient template methods

Roberts

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

Prewit

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

Sobel

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

Comparison: templates

Roberts Prewit Sobel

* post-processing used after edge detection

Edge strength and direction

The **magnitude** (length) of vector (∇f):

$$M(x,y) = mag(\nabla f) = \sqrt{(g_x)^2 + (g_y)^2} \approx |g_x| + |g_y|$$

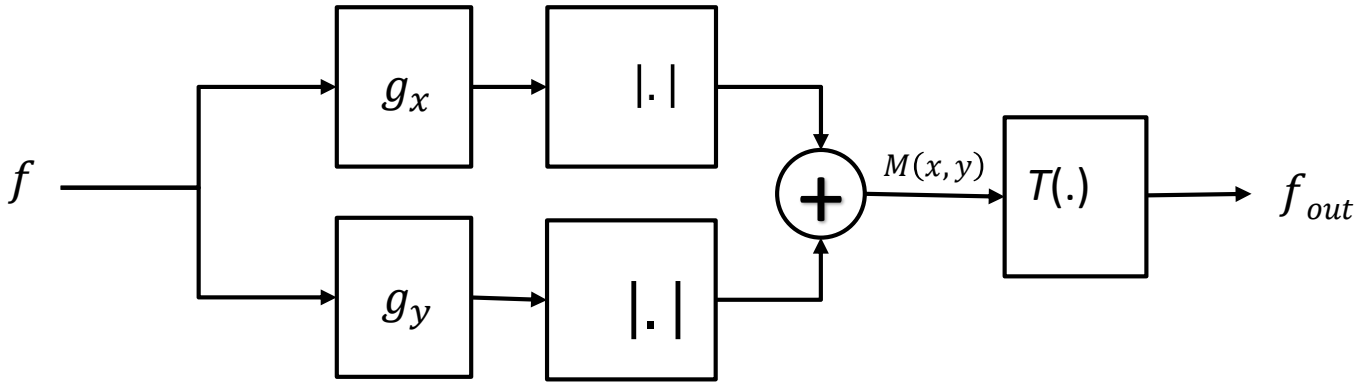
- identifies the edge strength.

The **direction** of vector (∇f):

$$\theta(x,y) = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

- The edge is perpendicular to the direction of the gradient vector at (*x,y*).

Gradient edge detection: flow chart



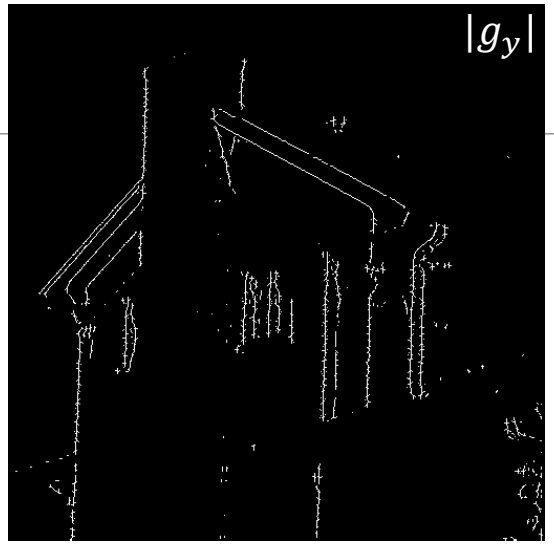
Choice of *T*(.) has important effect on the edge detection output

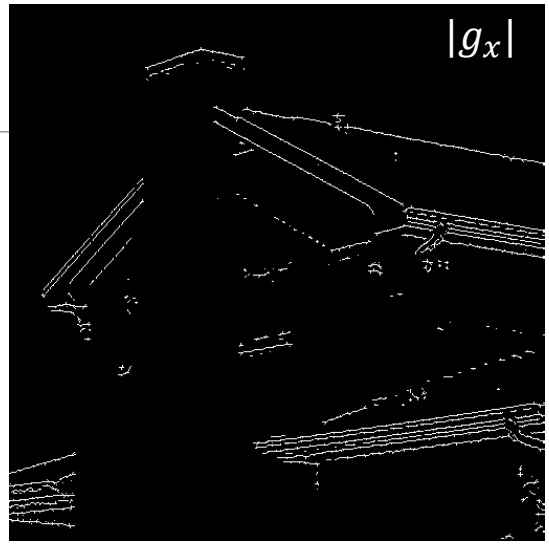
- *T*(.)→thresholding to reduce the effect of noise
- *T*(.)→detection of local maximum (to find the max values in an edge)
- *T*(.)→other post-processing methods, e.g. to remove noise detected as contour.

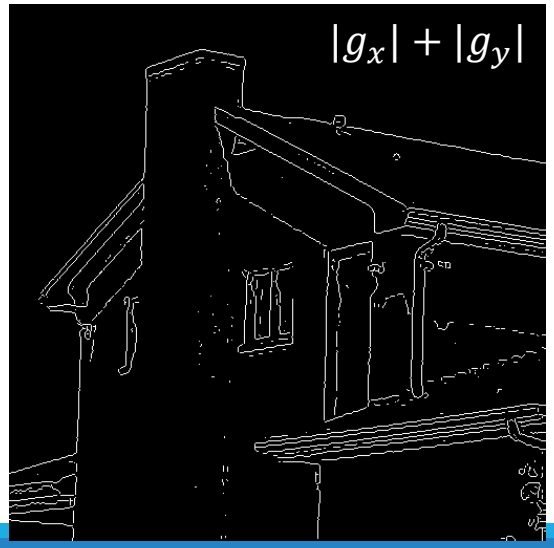
Example

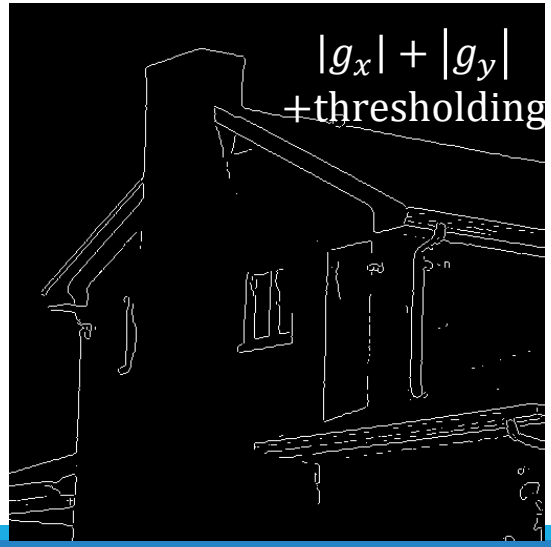
Ideal

* post-processing used after edge detection









|gx|

Reality!!

|gx|

|gx|+|gy|

|gx|+|gy|+Thresholding

Laplacian

The Laplacian is a 2D isotropic measure of the 2nd derivatives of an image. It highlights the regions of rapid intensity changes.

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

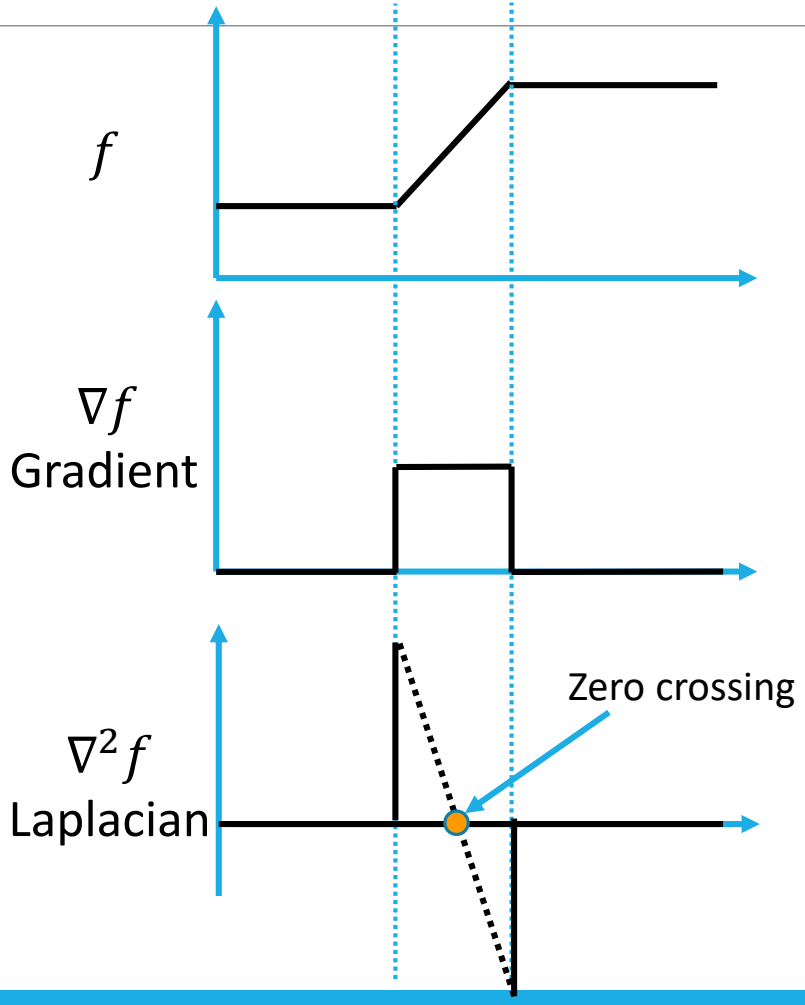
The approximation of the Laplacian

$$\begin{aligned}&\nabla^2 f(x,y)\\ &\approx f(x+1,y) + f(x,y+1) + f(x-1,y) + f(x,y-1) - 4f(x,y)\\ &\qquad = Mask_L \circ f(x,y)\end{aligned}$$

$$where \quad Mask_L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Gradient vs. Laplacian

image f



Gradient vs. Laplacian

1. 1st derivatives produce thicker edges.
2. 2nd derivatives have a stronger response to fine details such as thick lines, isolated points and noise.
3. 2nd derivatives produce a double edge response at ramp and step transitions in intensity.
4. The sign of the 2nd derivatives can be used to determine where a transition into an edge is from light to dark or dark to light.

Laplacian masks

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

the Laplacian is very sensitive to noise.

Marr-Hildreth edge detection

2nd derivative is very sensitive to noise, the image needs to be smoothed by a low pass filter first.

- Usually, Gaussian filter is used.

Marr-Hildreth edge detection

1. filter the image f with a Gaussian lowpass filter.
2. compute the Laplacian.
3. find the zero crossings to get f_{out}.

Marr-Hildreth edge detection

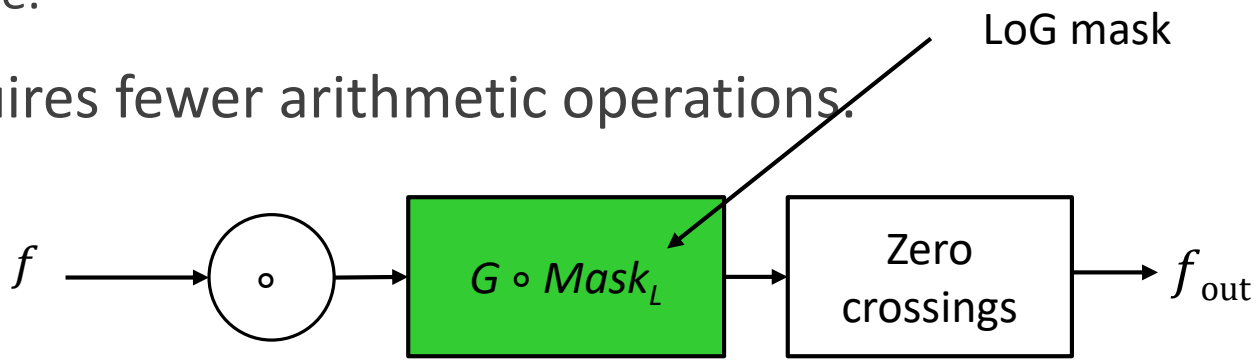
Since the convolution operation is associative

$$(f \circ G) \circ Mask_L = f \circ (G \circ Mask_L)$$

we can

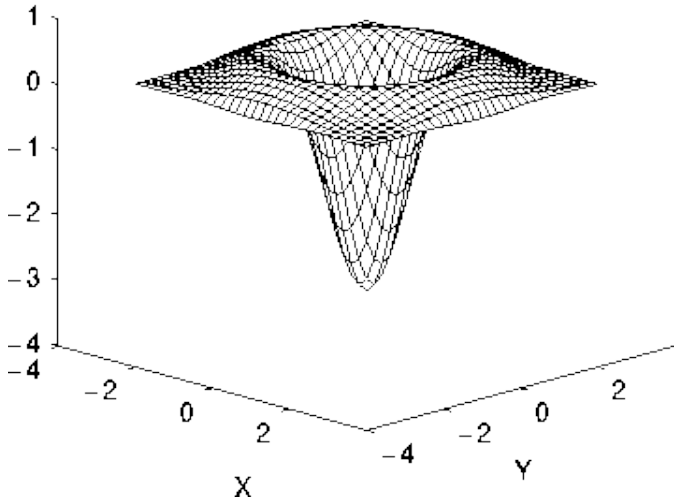
- first convolve the Gaussian mask with the Laplacian mask to generate LoG mask
- then convolve this hybrid mask (Laplacian of Gaussian mask) with the image.

It requires fewer arithmetic operations.



Laplacian of Gaussian (LoG) mask

The continuous 2D LoG function look Mexican Hut.



0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

The LoG mask with σ =1.4 and n=9

n is the smallest odd integer $\geq 6\sigma$

Laplacian of Gaussian (LoG) mask

The LoG filter can be approximated by *DoG* (**D**ifference **o**f **G**aussians)

- DoG: difference of two differently sized Gaussians

$$DoG(x,y) = f_{G1} - f_{G2}$$

where $\sigma_1 > \sigma_2$

Another even faster and cruder approximation

- DoB (**D**ifference **o**f **B**oxes): difference between two differently sized box filters

Laplacian of Gaussian (LoG) mask

At a reasonable sharp edge between two uniform regions with different intensities, the LoG response will be

- =0 at a long distance from edge.
- >0 to the darker side of the edge
- <0 just to lighter of the edge
- =0 at the some in between, on the edge itself.

image *f*

The output of the LoG filter passed through zero at edges can be used to detect those edges: zero crossings.

Canny edge detector

The superior edge detector in general but more complex.

It is designed based on three basic objectives:

1. Low error rate
   - All edges should be found. The edges detected must be as close as possible to the true edges.
2. Edge points should be well localized
   - The distance between a point marked as an edge by the detector and the centre of the true edge should be minimum.
3. Single edge point response
   - Detector should only return one point for each true edge point.

Canny edge detector

The steps are:

1. **Gaussian filter** to smooth the image
2. The image **gradient (magnitude and angle) calculation**
3. **Non-maximum suppression:** edge thinning
4. **Hysteresis thresholding**
5. **Double thresholding**: strong and weak
   - A weak edge is not considered a real contour unless it is connected to a strong contour.
   - Less sensitive to noise
   - Better in detecting the real weak contours
6. **Connectivity analysis**
   - Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

Step 5 and 6 are also called ‘Hysteresis Thresholding’.

Effect of σ (Gaussian mask size)

original

Canny with $\sigma = 1$

Canny with $\sigma = 2$

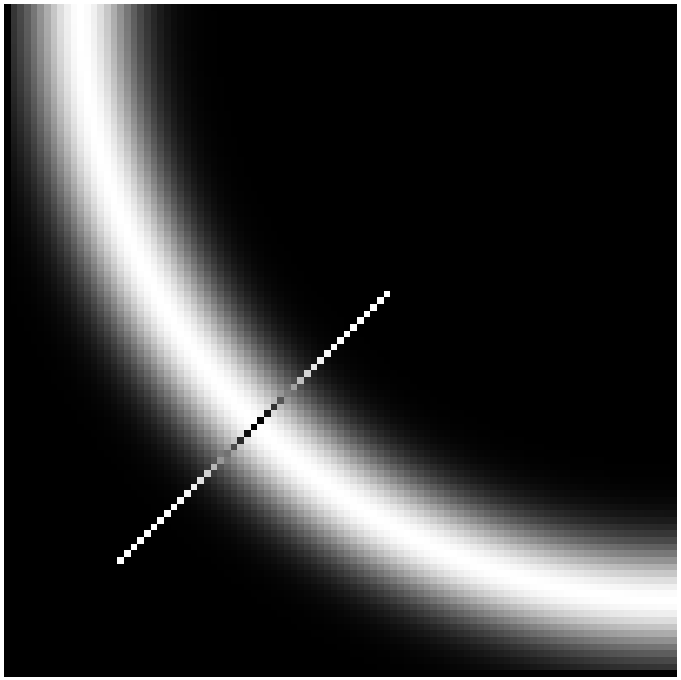
The choice of σ depends on desired behavior:

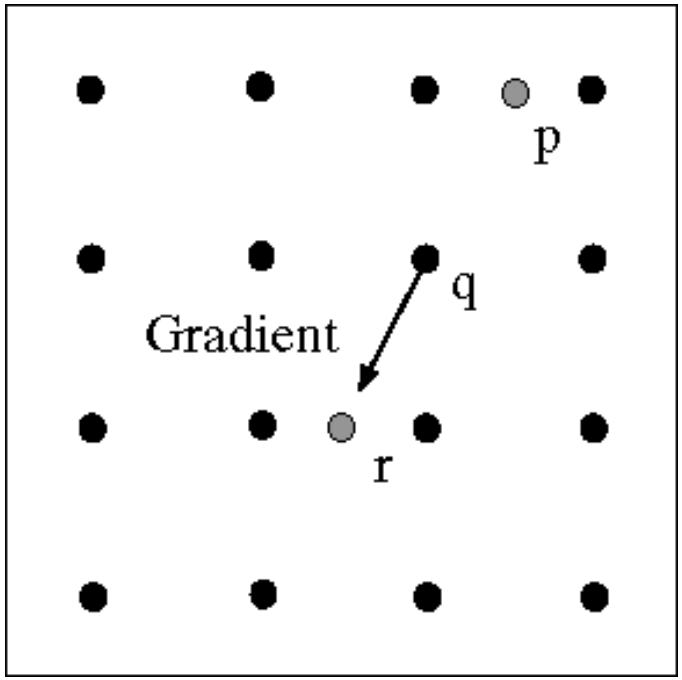
- large σ detects large scale edges.
- small σ detects fine features.

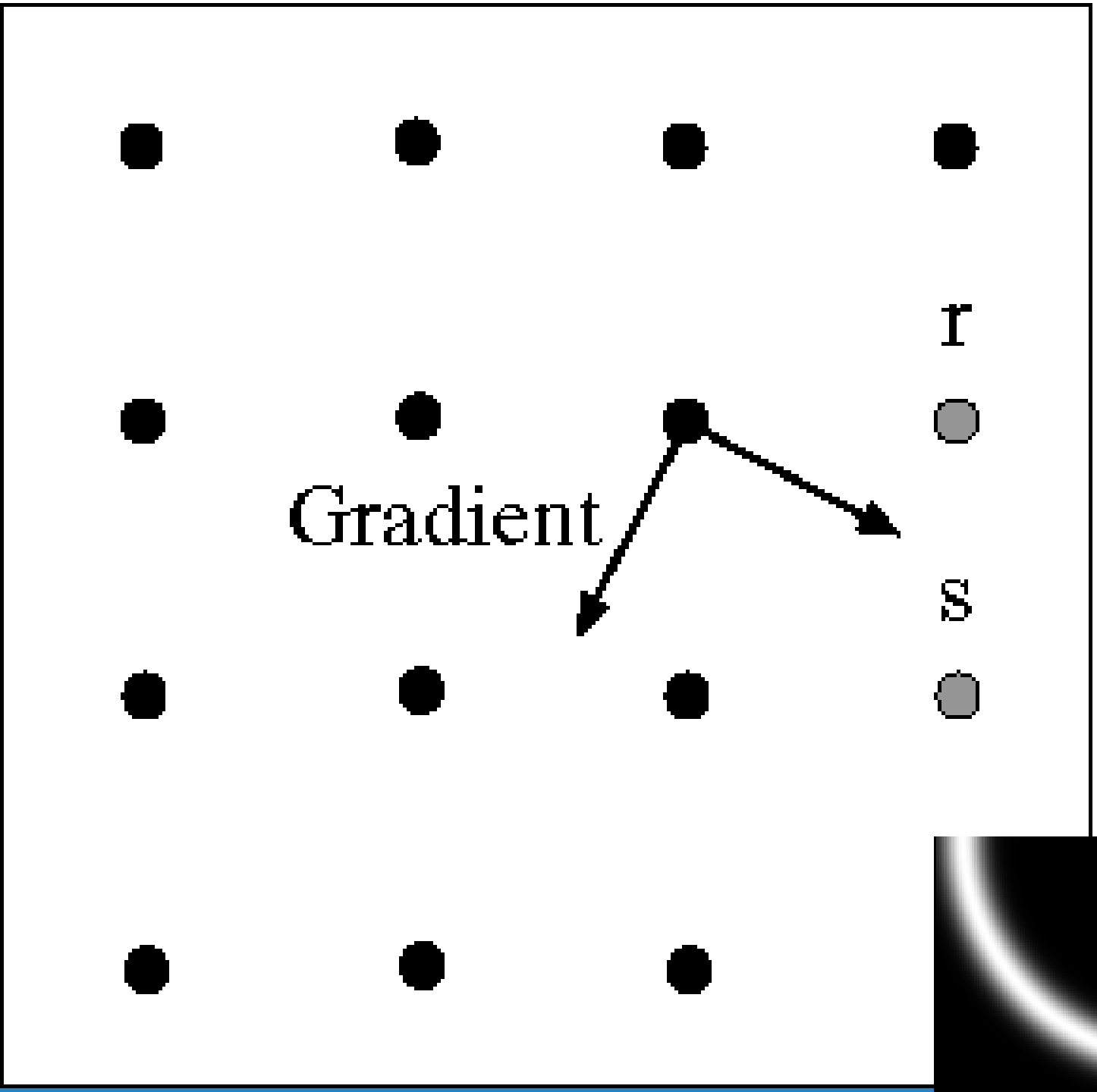
Non-maximum suppression

Check if the pixel is local maximum along gradient direction.

- requires checking interpolated pixels p and r.

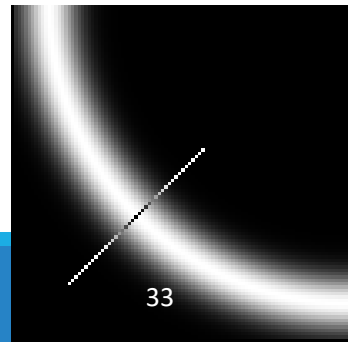






Predicting the next edge point

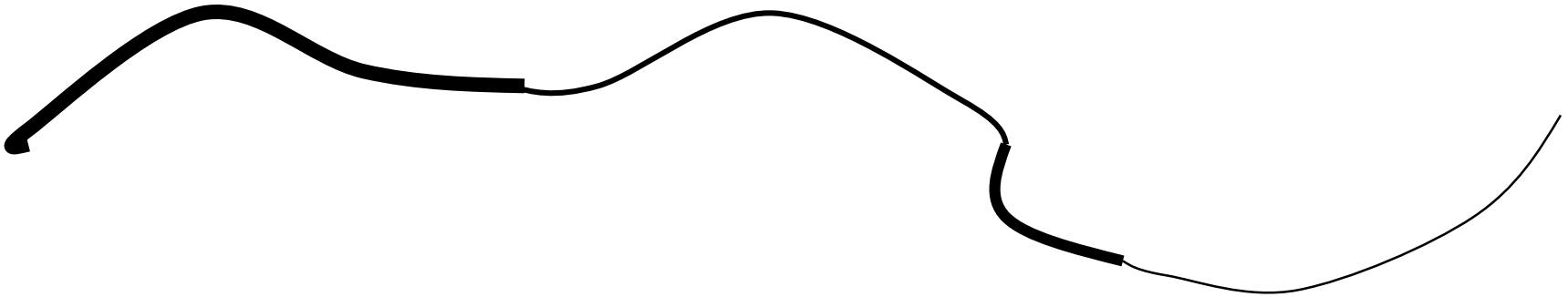
Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).



Hysteresis thresholding

Check if the maximum value of a gradient value is sufficiently large.

- drop-outs? use **hysteresis thresholding**
  - use a high threshold to start edge curves and a low threshold to continue them.



Original

Original

Gaussian smoothing

Smoothed

Gradient

magnitute

Non-maximum suppression

after non-maximum suppression

Hysteresis thresholding

after Hysteresis thresolding

Q&A