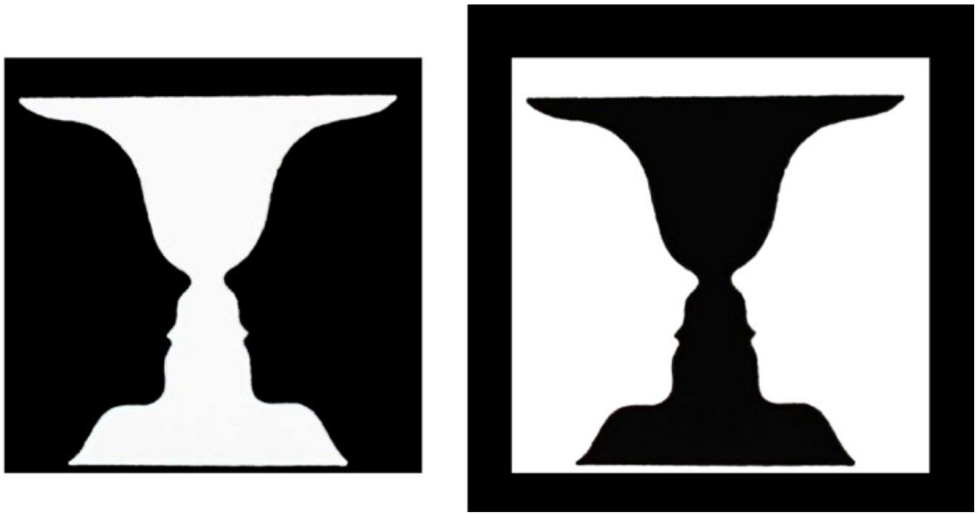
Edge Detection

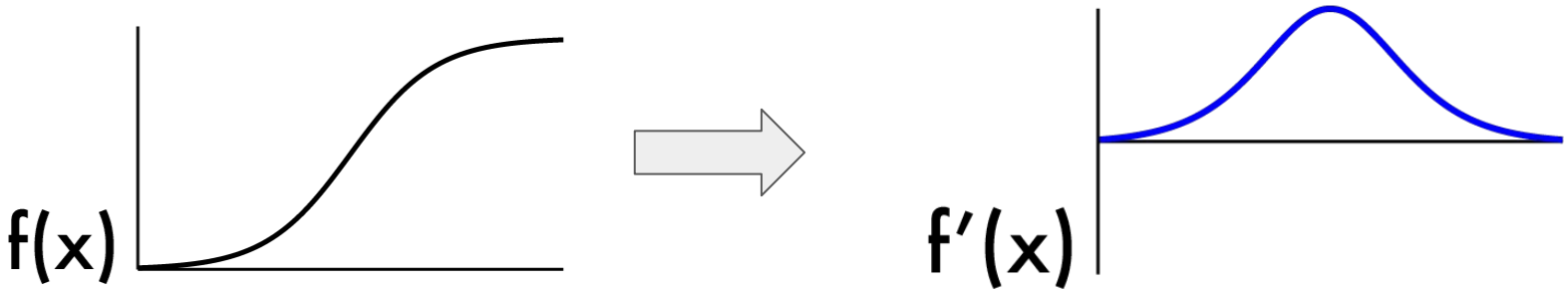
Outline

- Edges
- Image Gradient
- Sobel Operator
- LoG
- Canny Edge Detector

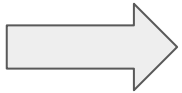


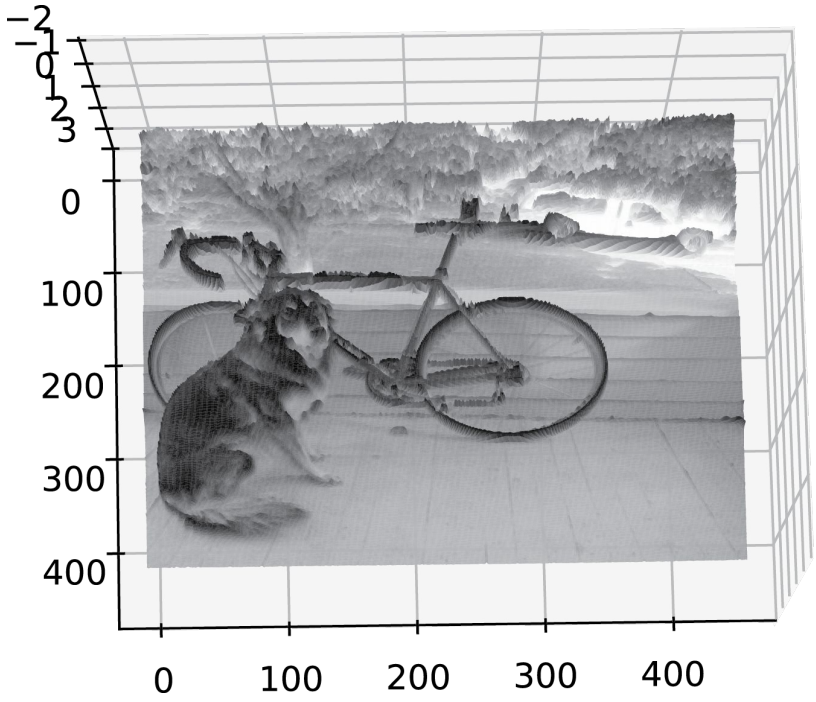
Derivatives of Continuity

let’s say “CHANGEs”



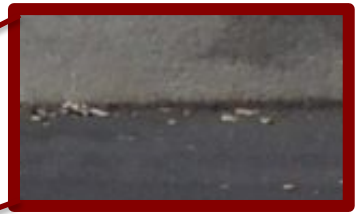
Images as Functions





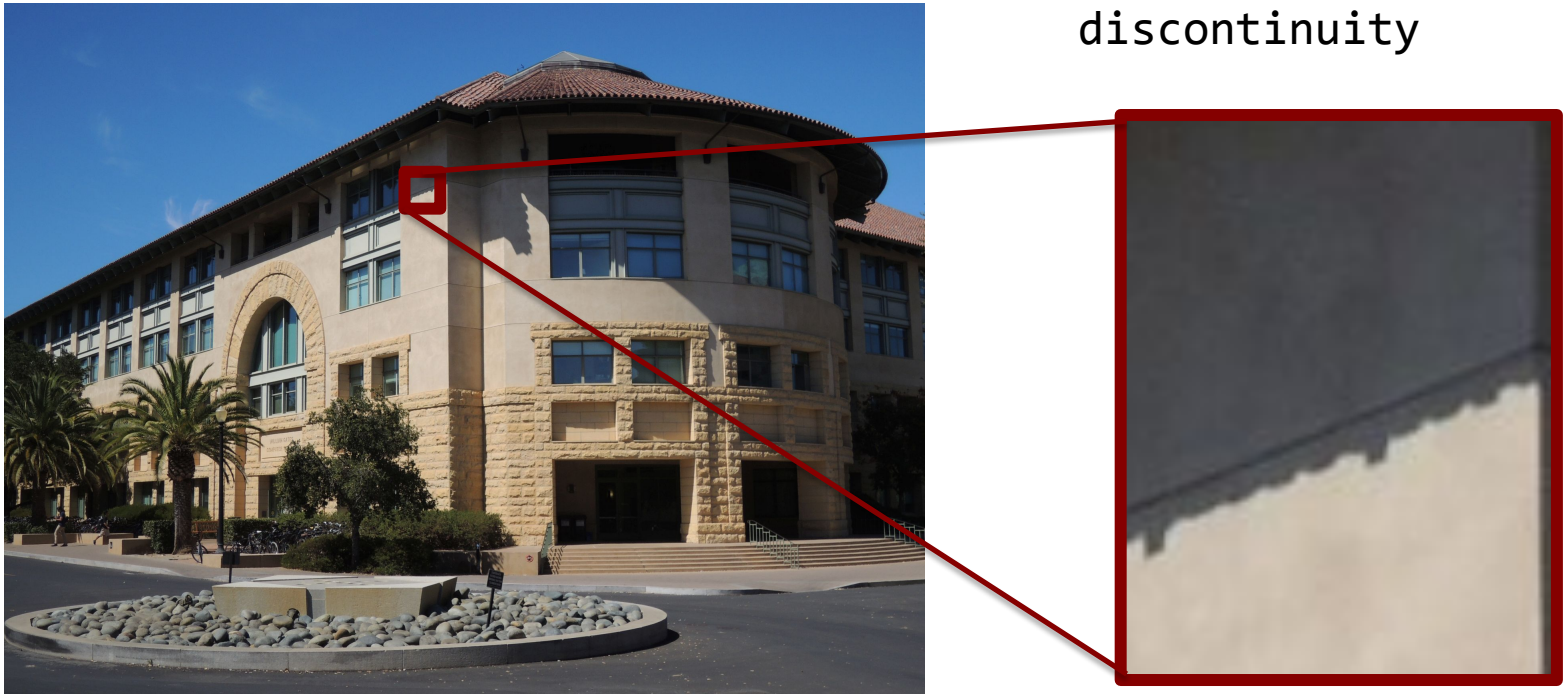
What are edges (discontinuity) ?

Surface normal discontinuity



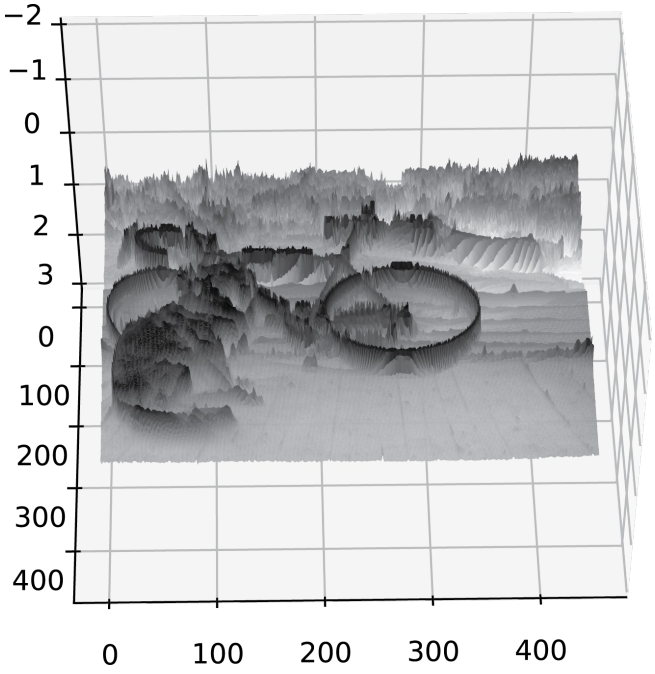
Cont’d

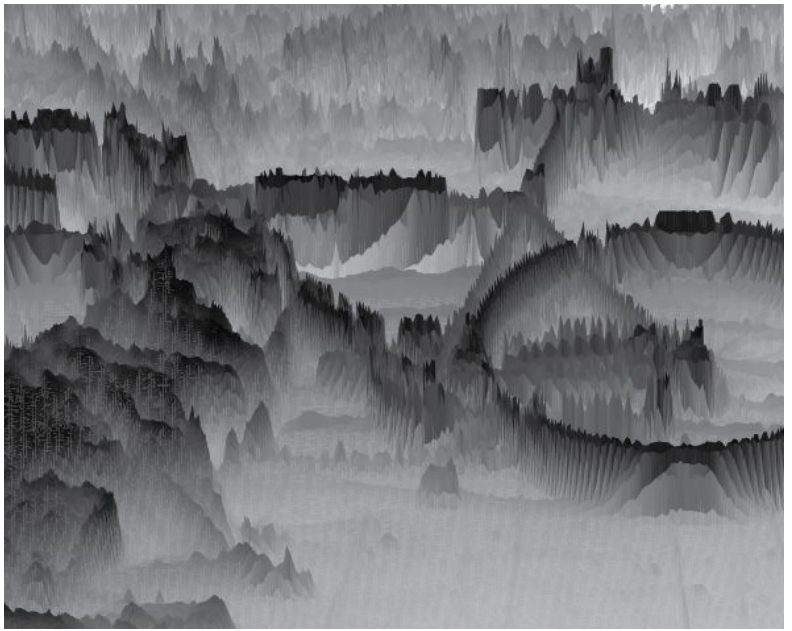
Cont’d



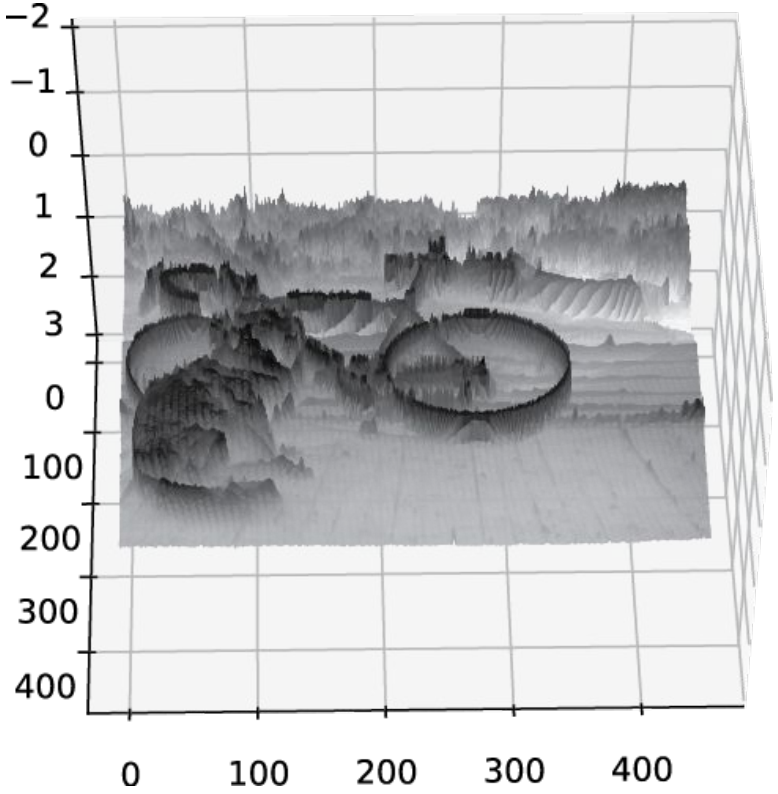
What are edges ?

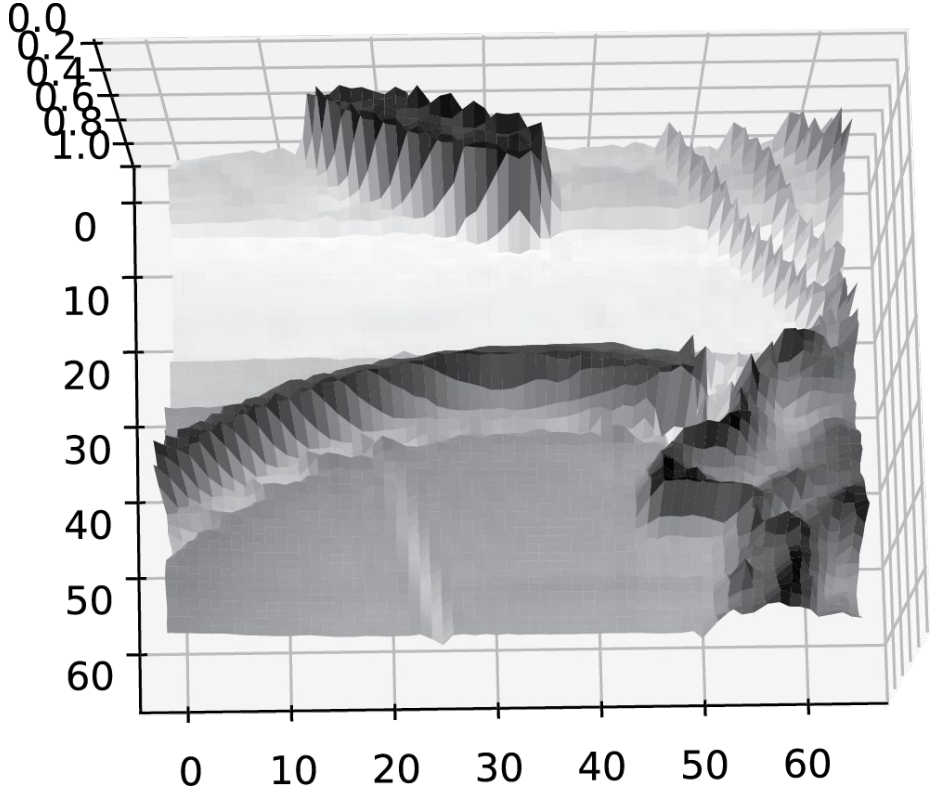
- Edges are rapid changes in this function



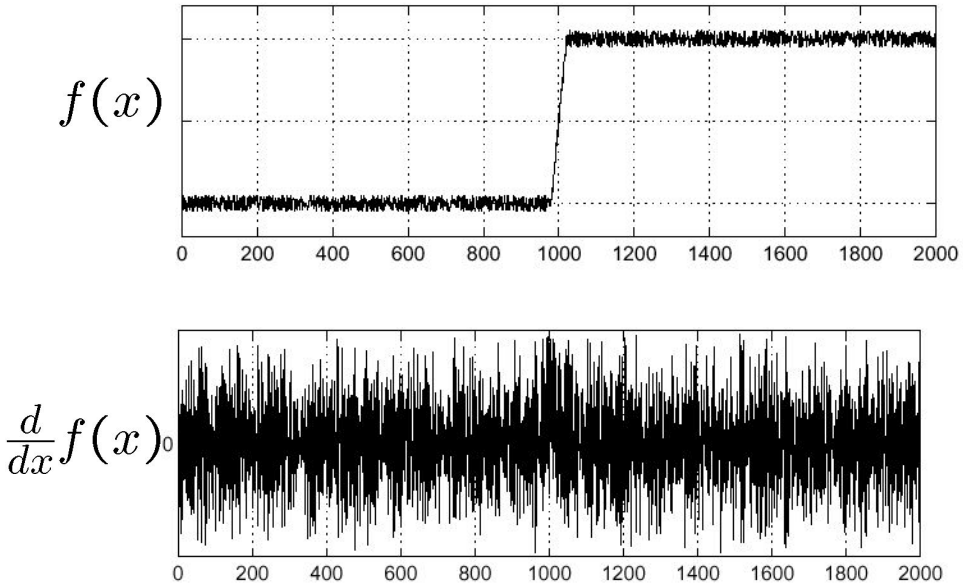


Images are noisy!





Effects of noise



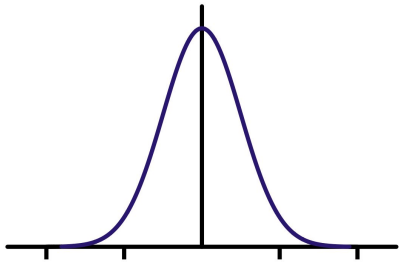
Where is the edge?

Smoothing with different filters

- Mean smoothing

$$\begin{bmatrix}1\\1\\1\end{bmatrix} \qquad \begin{bmatrix}1 & 1 & 1\end{bmatrix}$$

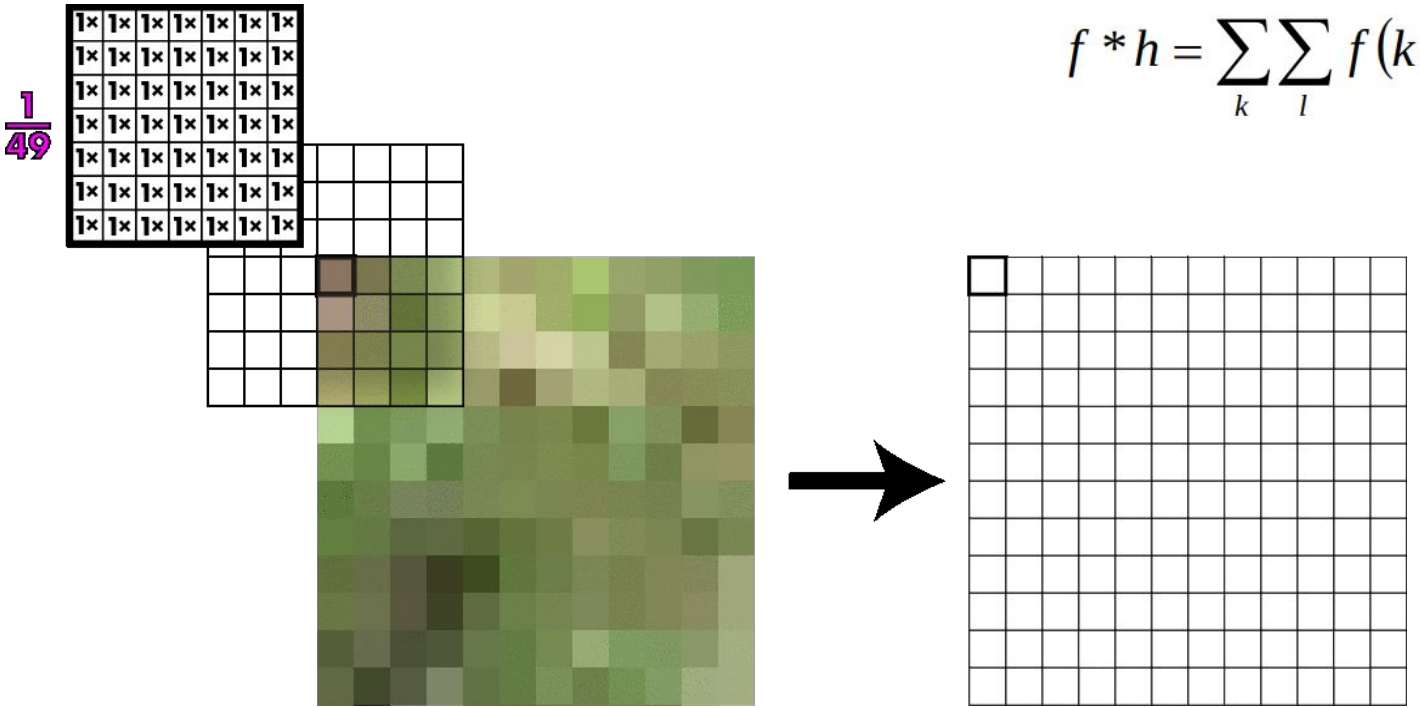
- Gaussian (smoothing * derivative)



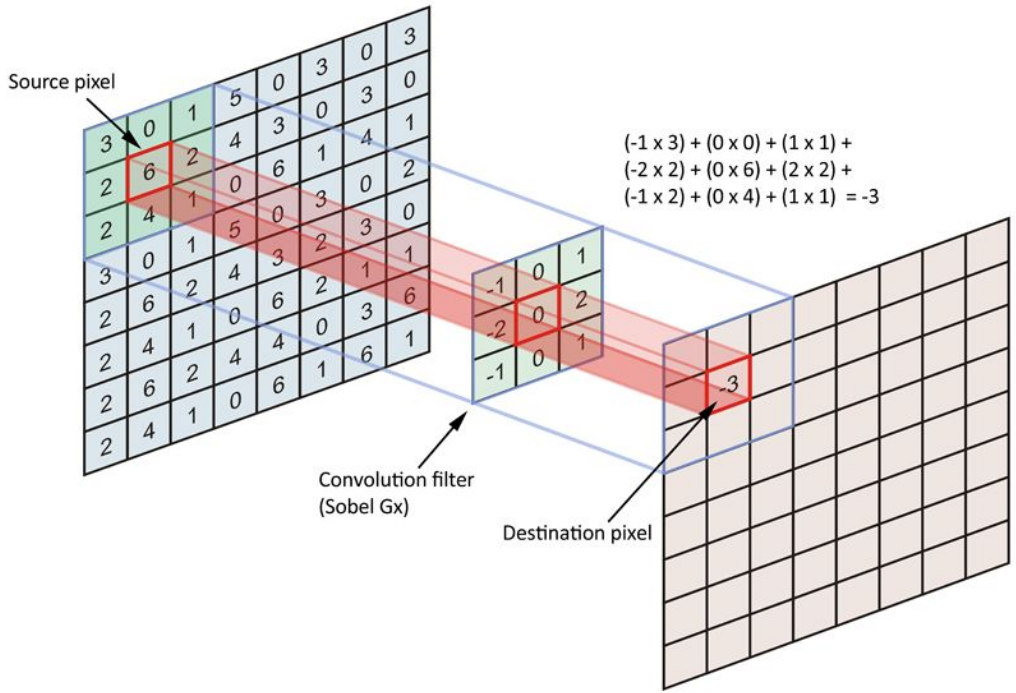
$$\begin{bmatrix}1\\2\\1\end{bmatrix} \qquad \begin{bmatrix}1 & 2 & 1\end{bmatrix}$$

Slide credit: Steve Seitz

Kernel slides across image

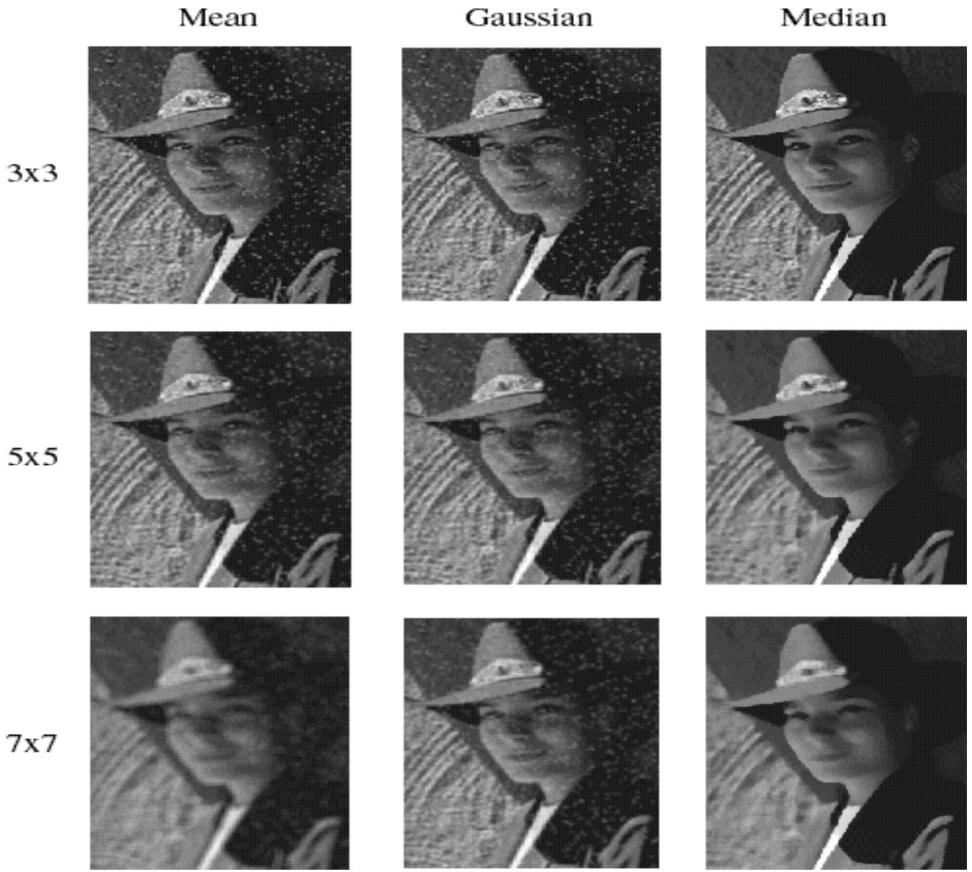


$$f * h = \sum_k \sum_l f(k,l)h(-k,-l)$$

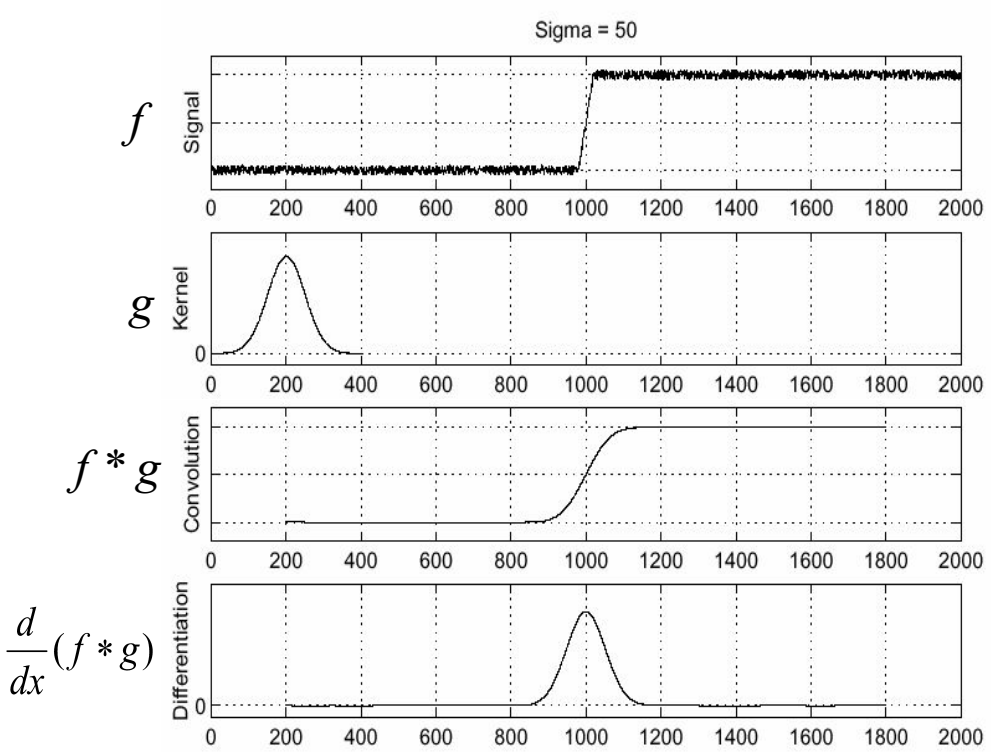


https://setosa.io/ev/image-kernels/

Smoothing with different filters



Solution: smooth first



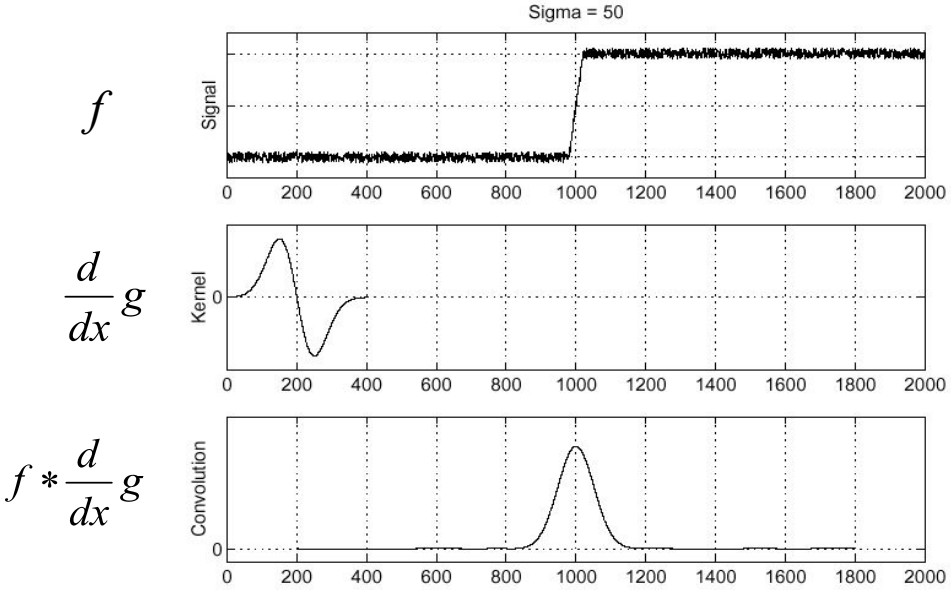
- To find edges, look for peaks in

Derivative theorem of convolution

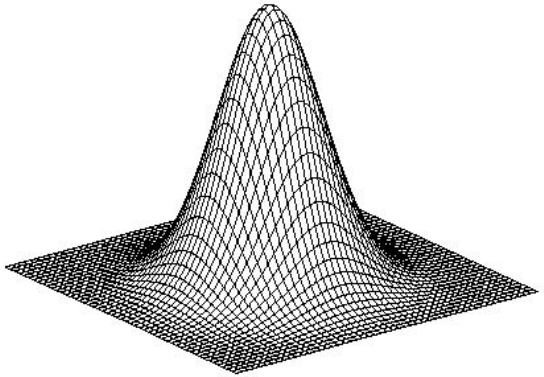
- This theorem gives us a very useful property:

$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

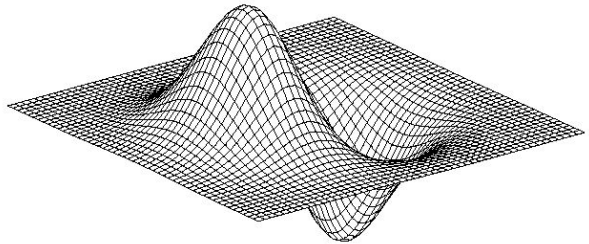
- This saves us one operation:



Derivative of Gaussian filter



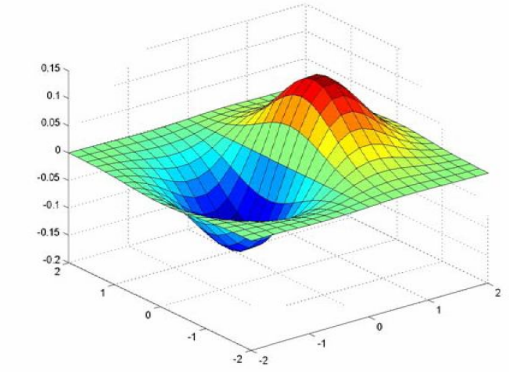
* [1 0 -1] =



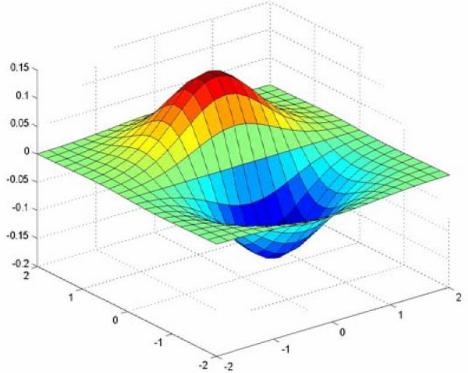
2D-gaussian

x - derivative

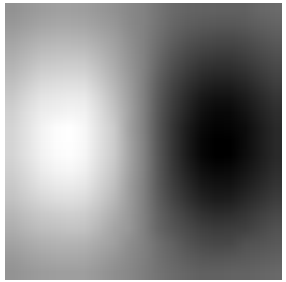
Derivative of Gaussian filter

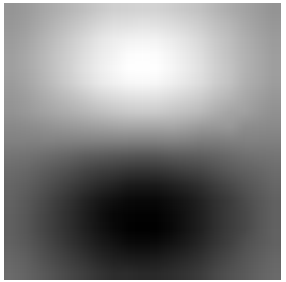


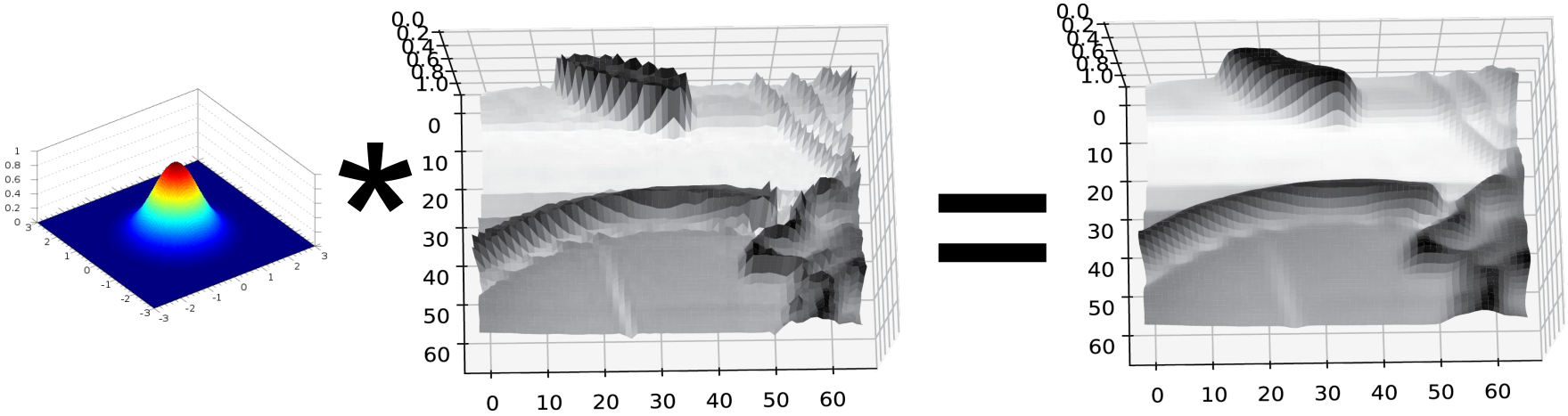
x-direction



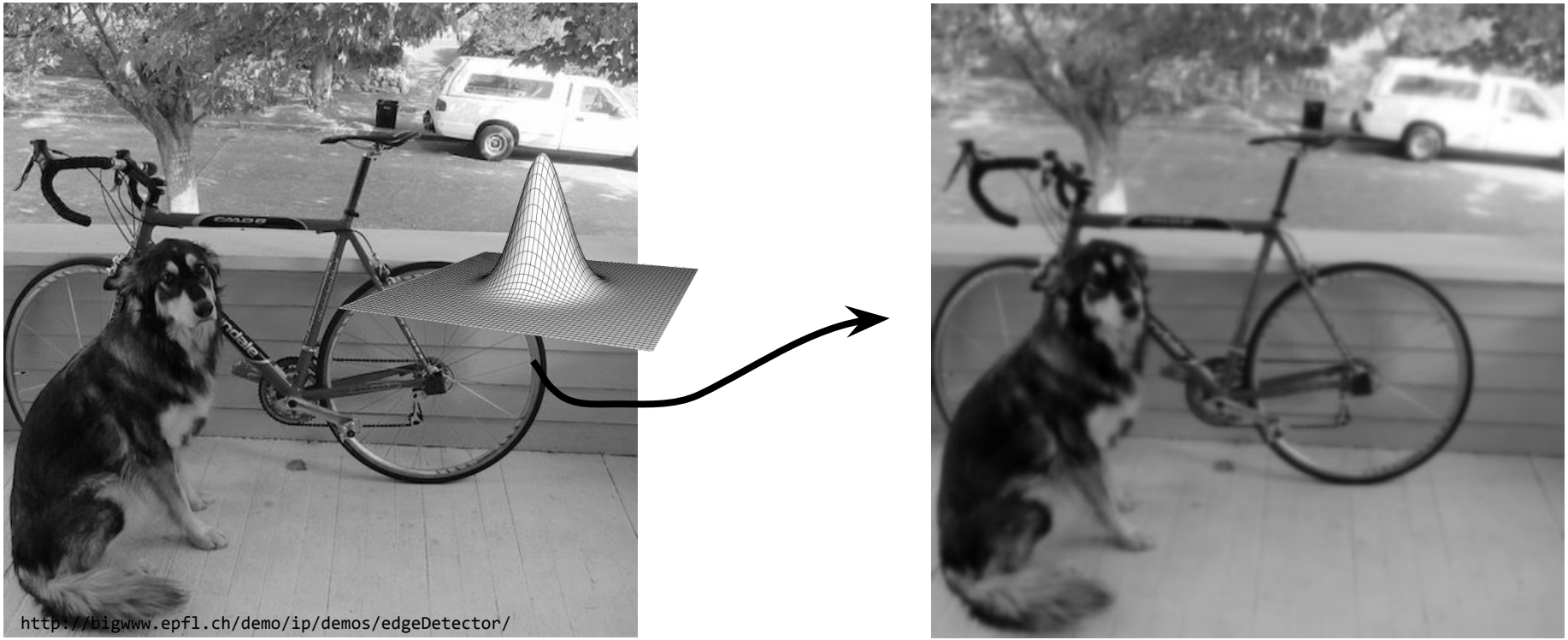
y-direction



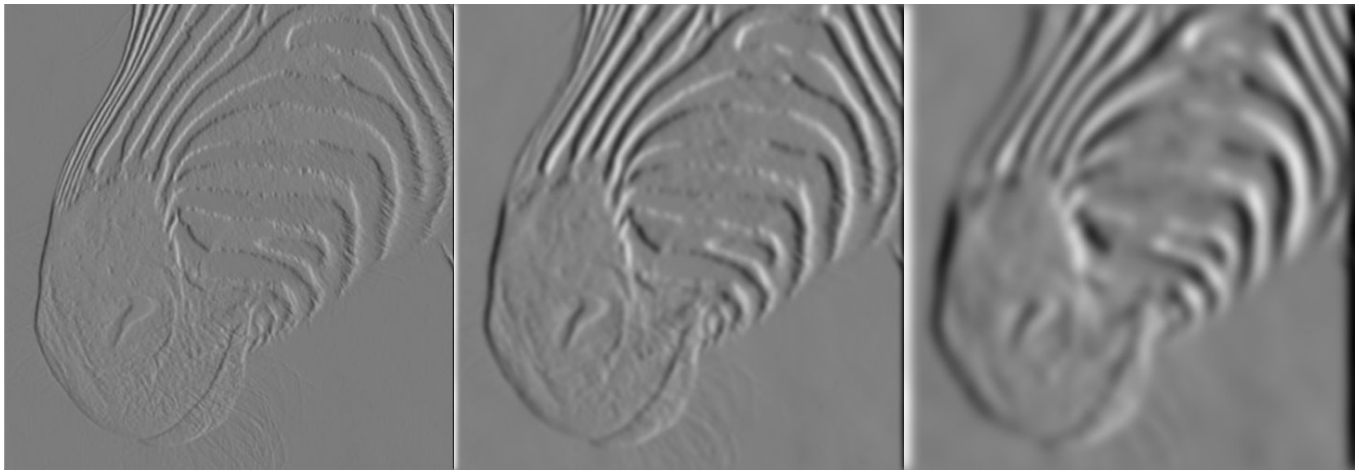




```
img_blur = cv2.GaussianBlur(img,(3,3), SigmaX=0, SigmaY=0)
```


http://bigwww.epfl.ch/demo/ip/demos/edgeDetector/

Tradeoff between smoothing at different scales



1 pixel 3 pixels 7 pixels

- Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”.

Sobel Operator

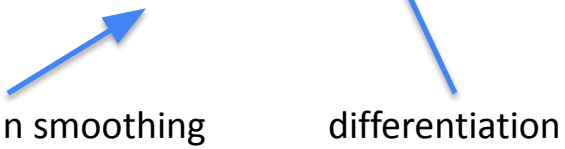
- Uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives
- One for horizontal changes, and one for vertical

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \qquad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Sobel Operation

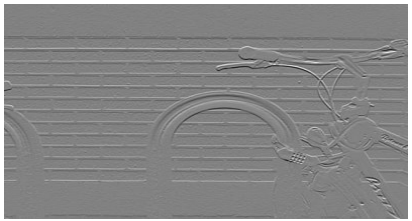
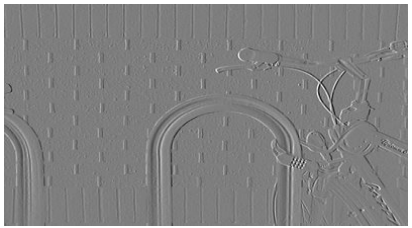
- Smoothing + differentiation

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} +1 & 0 & -1 \end{bmatrix}$$



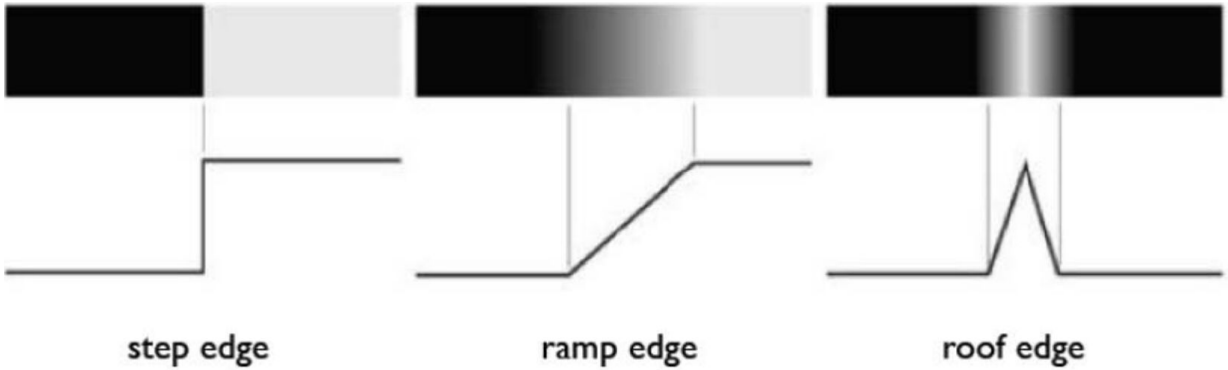
Sobel Operation

```
# Sobel Edge Detection
sobelx = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=5)
# Sobel Edge Detection on the X axis
sobely = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=5)
# Sobel Edge Detection on the Y axis
sobelxy = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=1, ksize=5)
# Combined X and Y Sobel Edge Detection
```


- Magnitude
- Direction

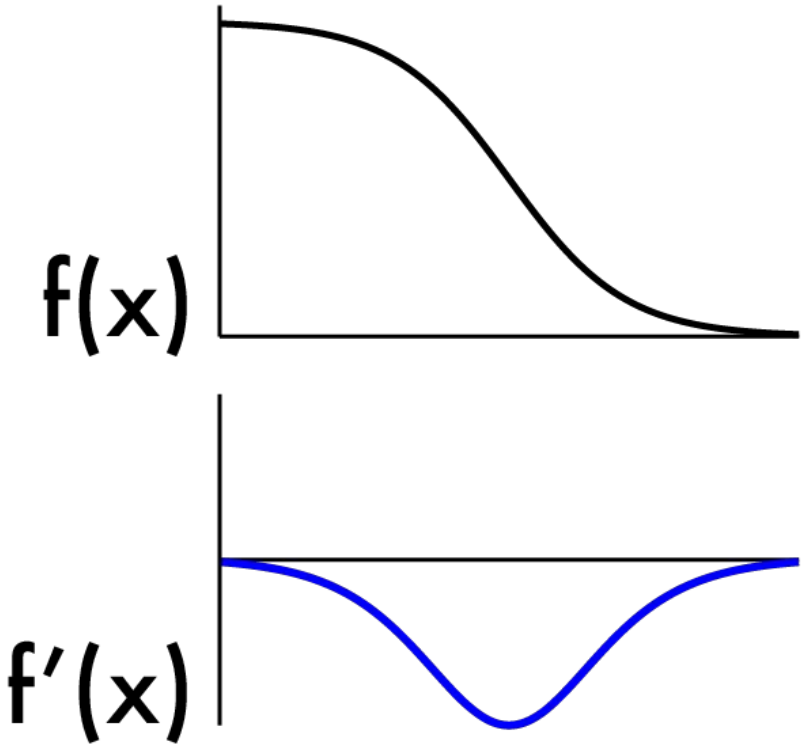
Sobel Filter Problems



- Poor Localization (Trigger response in multiple adjacent pixels)
- Thresholding value favors certain directions over others
  - Can miss oblique edges more than horizontal or vertical edges
  - False negatives

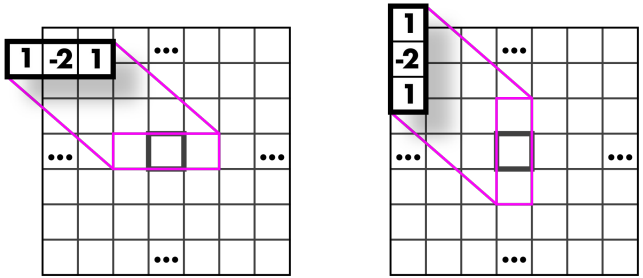
Finding edges

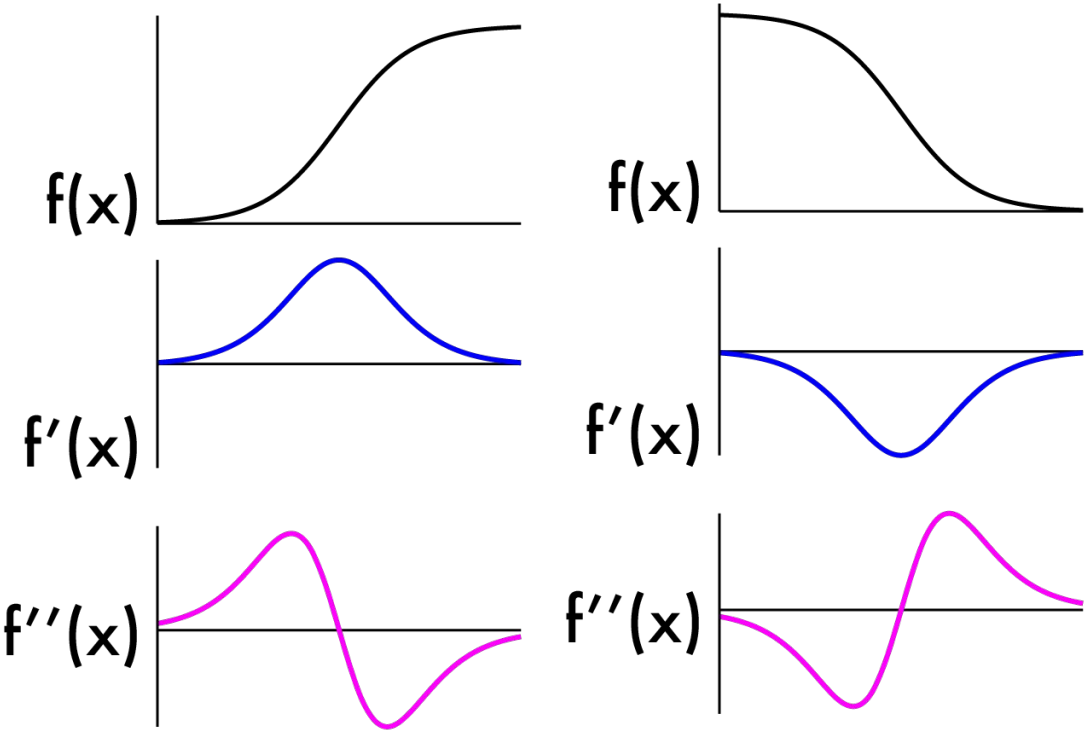
- Could take derivative
- Find high responses
- Sobel filters!
- But…
- Edges go both ways
- Want to find extrema



2nd derivative!

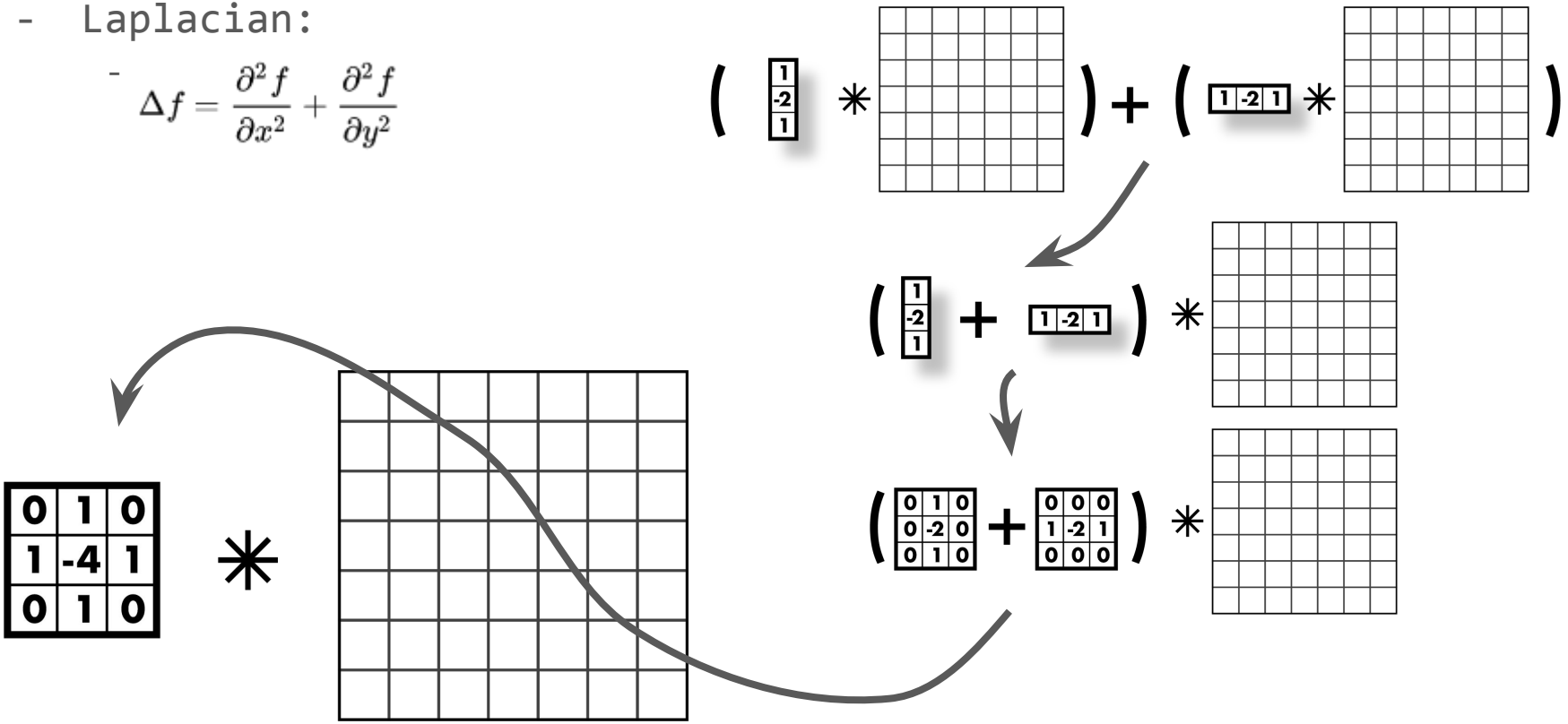
- Laplacians
- Crosses zero at extrema

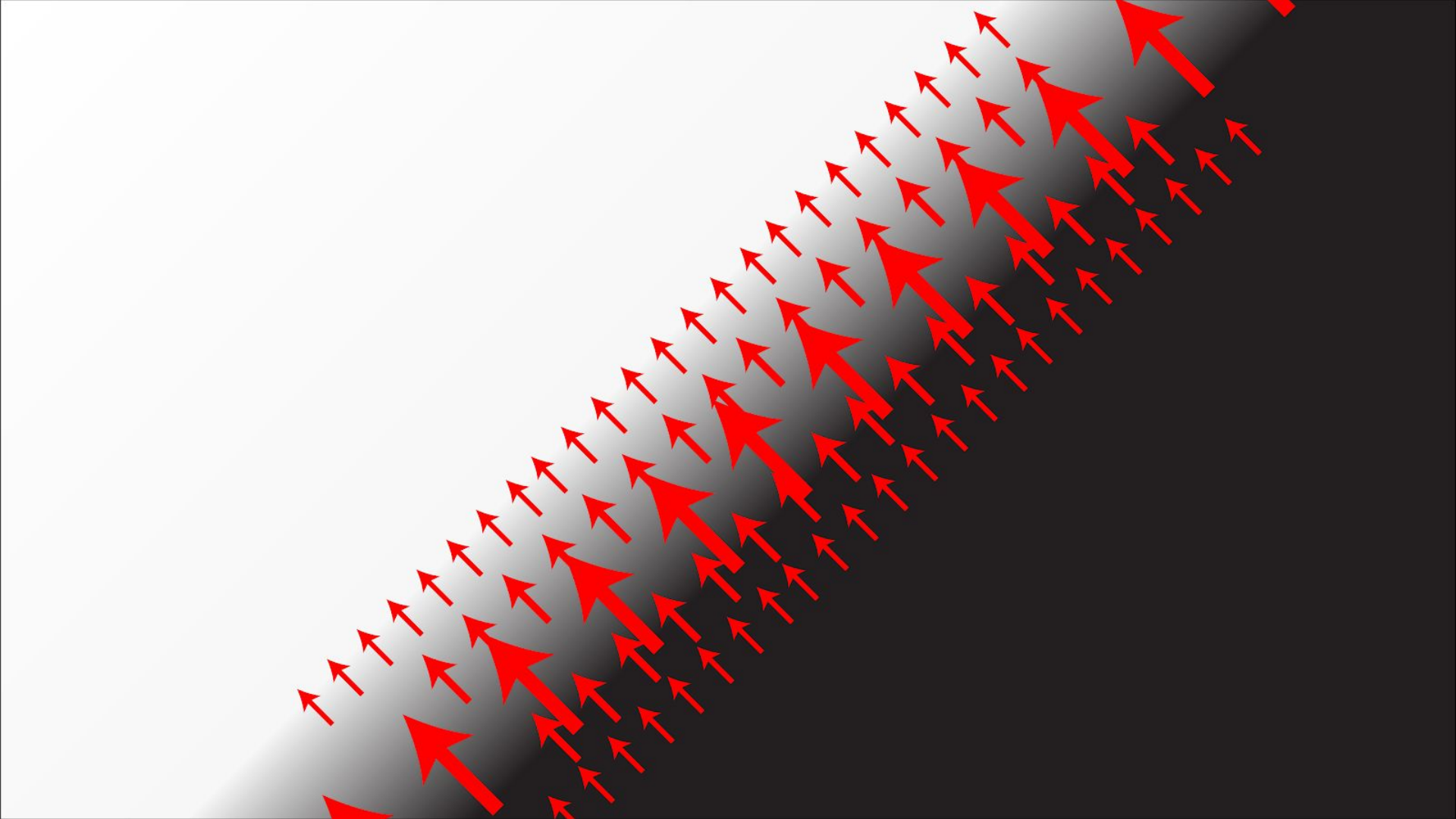


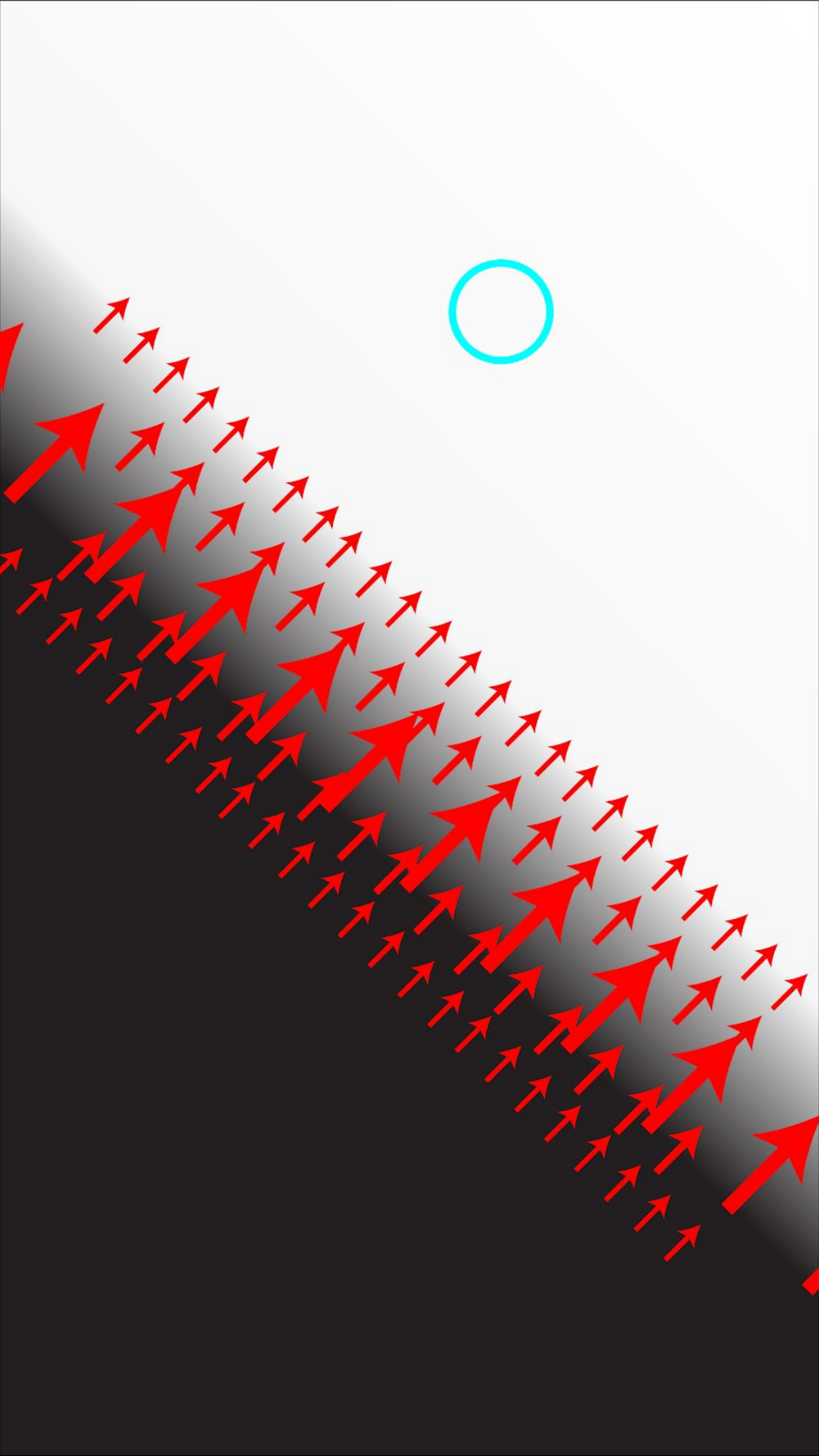


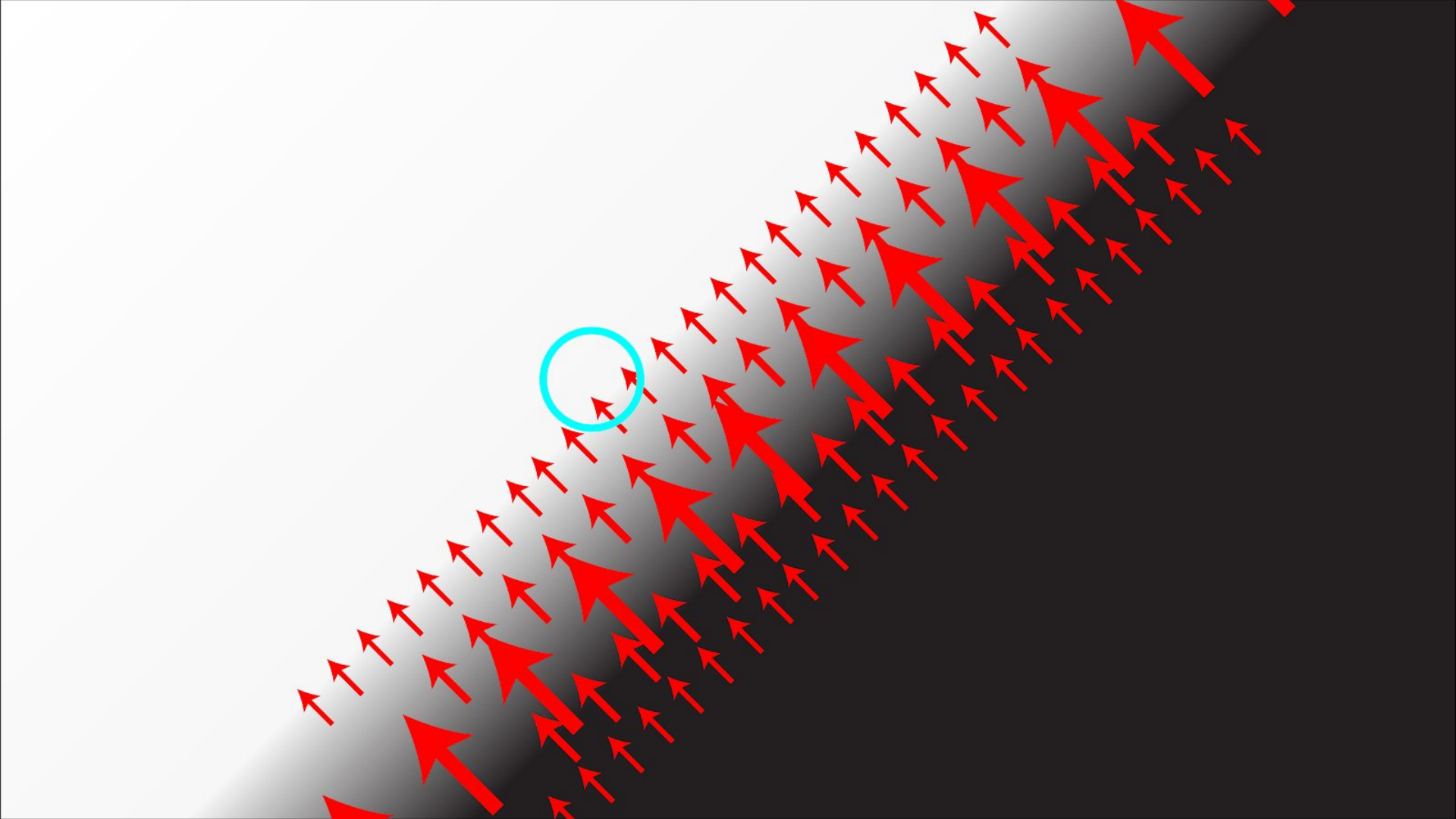
Laplacians

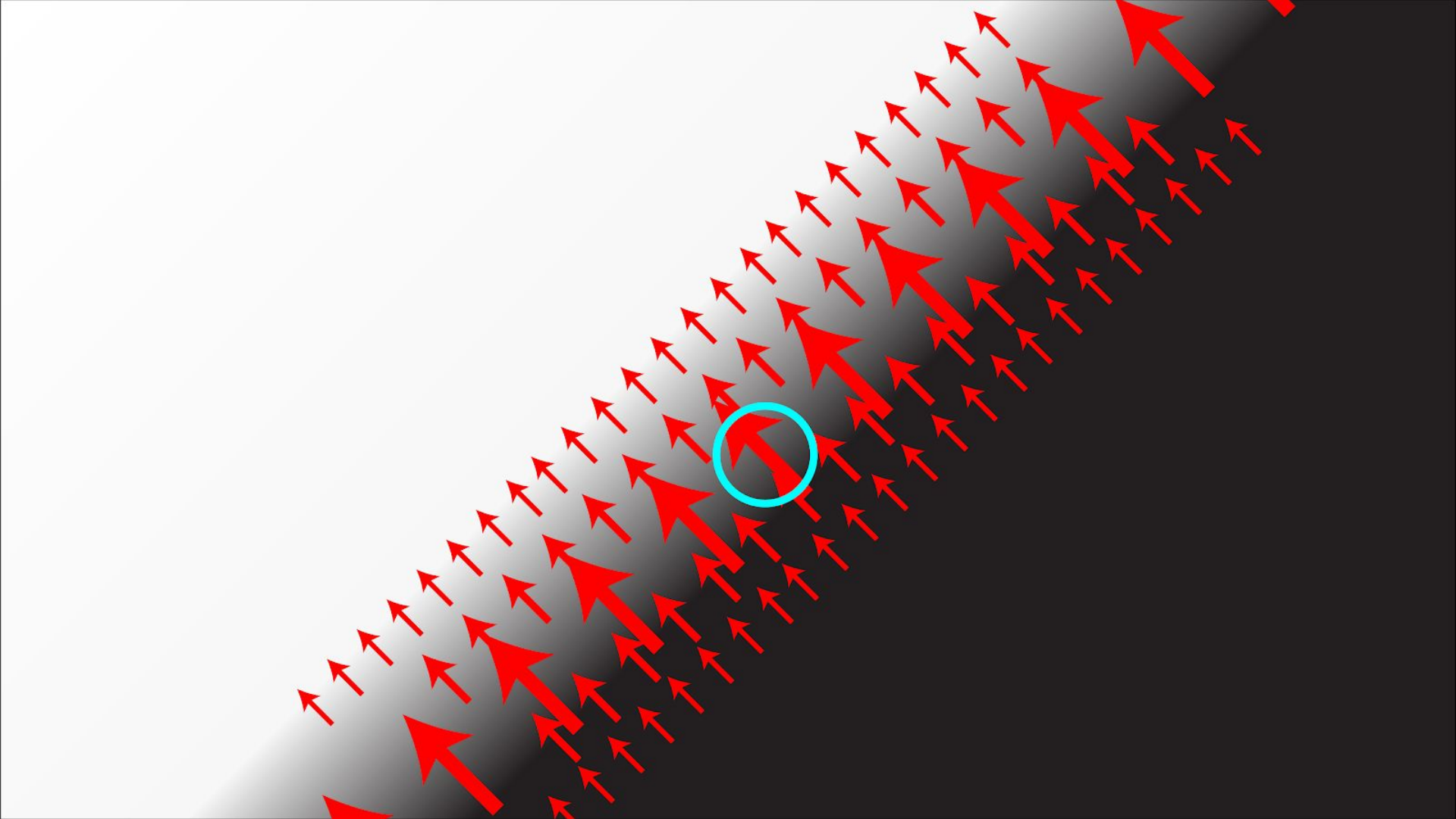
- Laplacian:
  - $\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

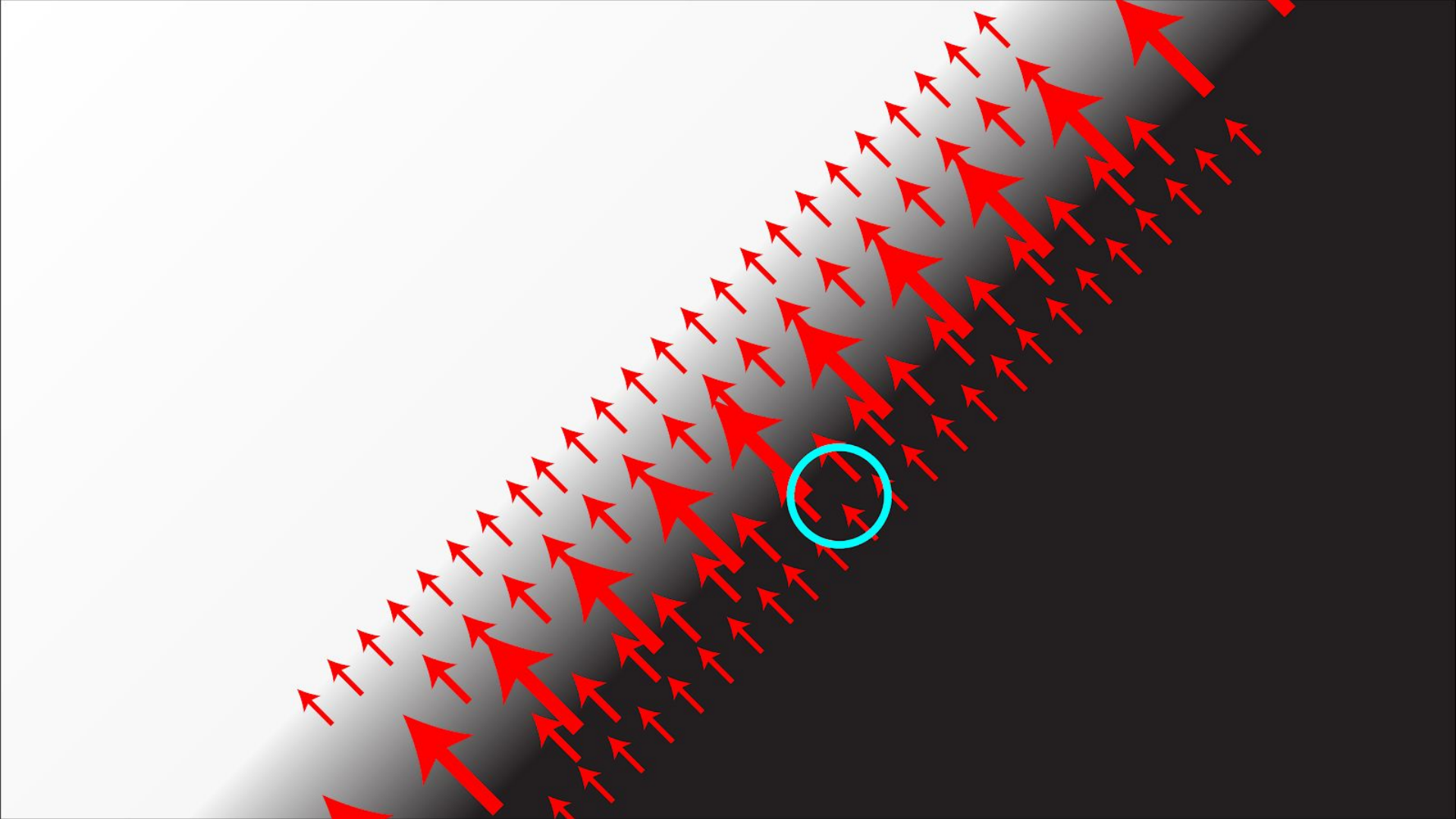




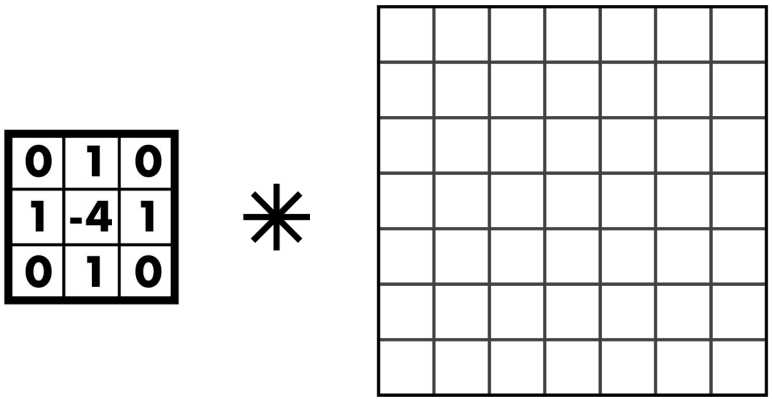


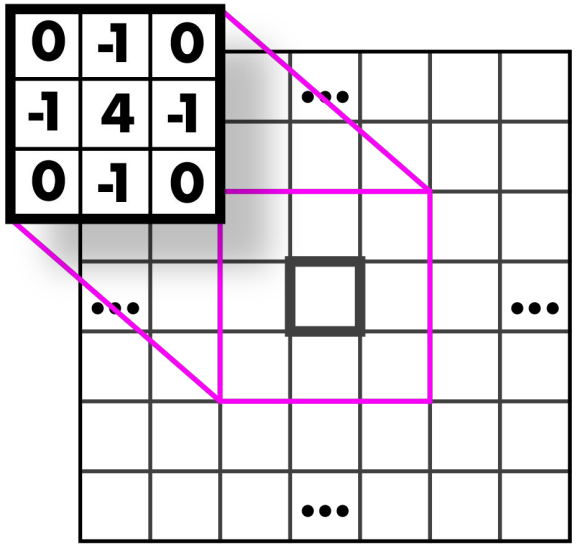






- Negative Laplacian, -4 in middle
- Positive Laplacian
- Can get good approx. with
  5x5 - 9x9 kernels





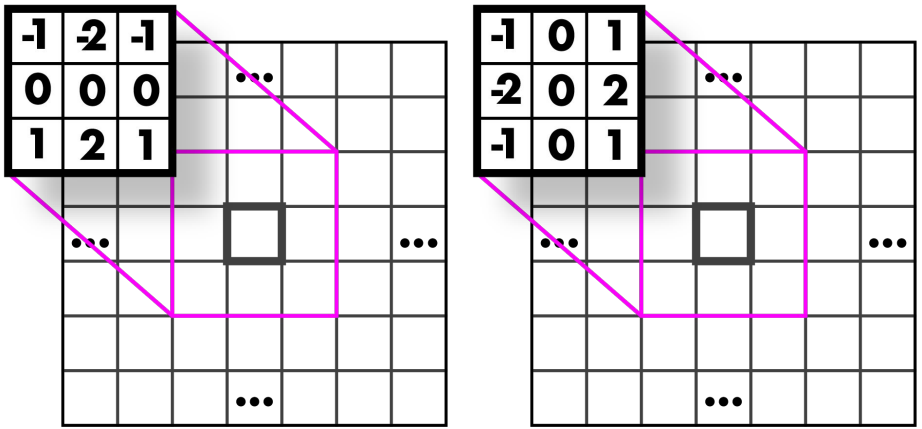
Canny Edge Detection

Algorithm:

- Smooth image (only want “real” edges, not noise)
- Calculate gradient direction and magnitude
- Non-maximum suppression perpendicular to edge
- Threshold into strong, weak, no edge
- Connect together components

Gradient magnitude and direction

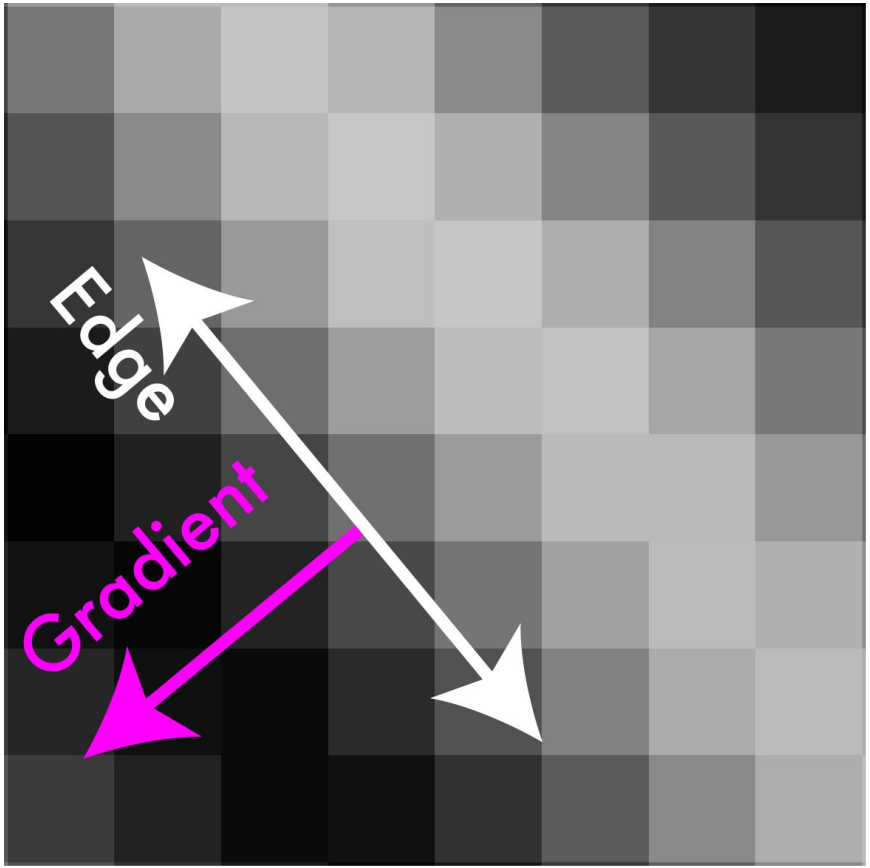
- Sobel filter

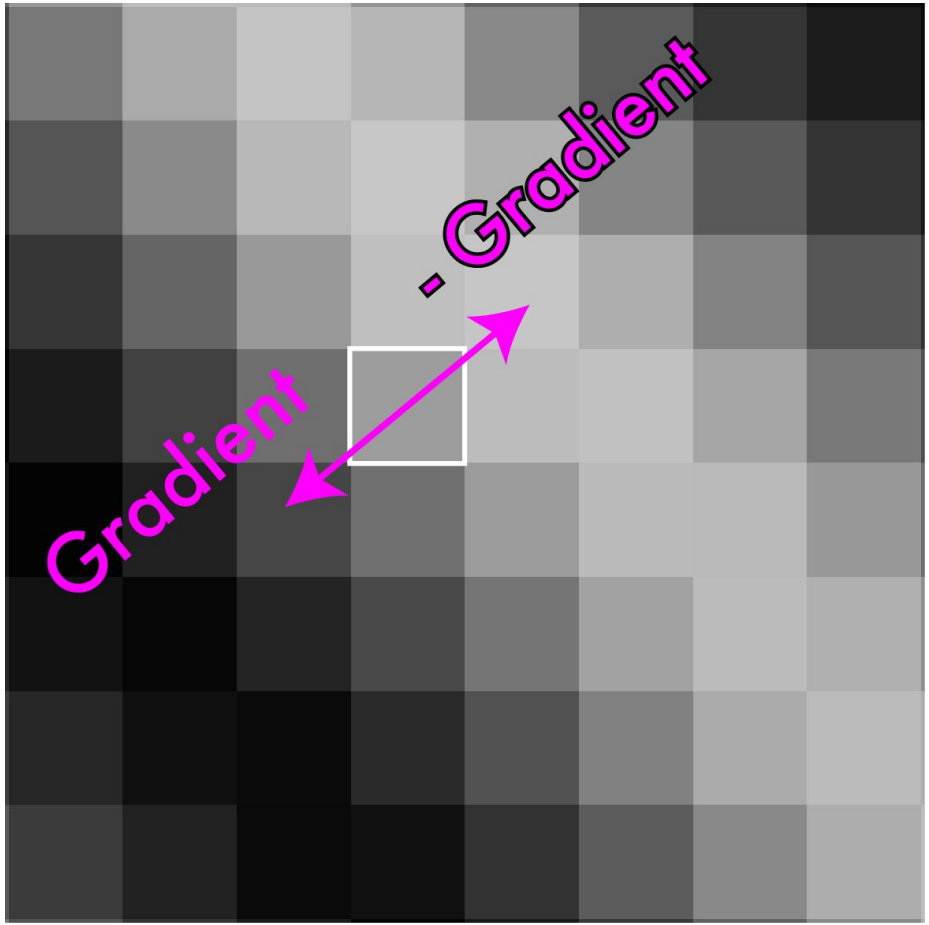


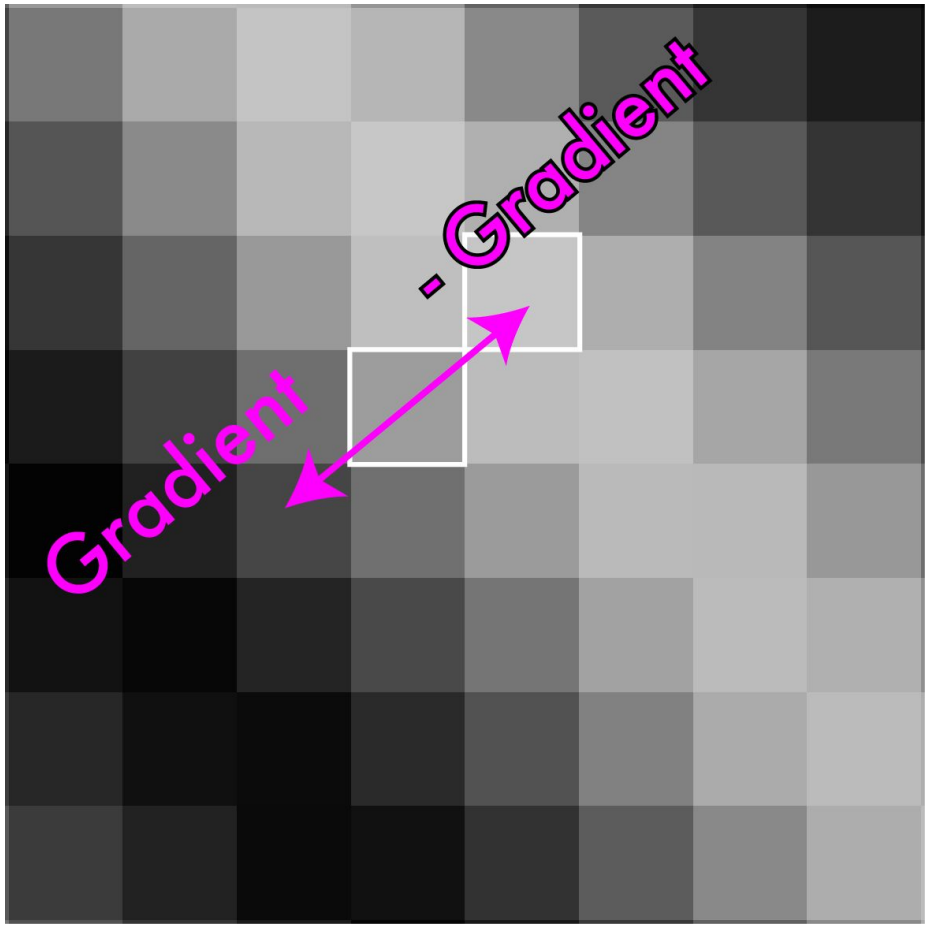
Non-maximum suppression

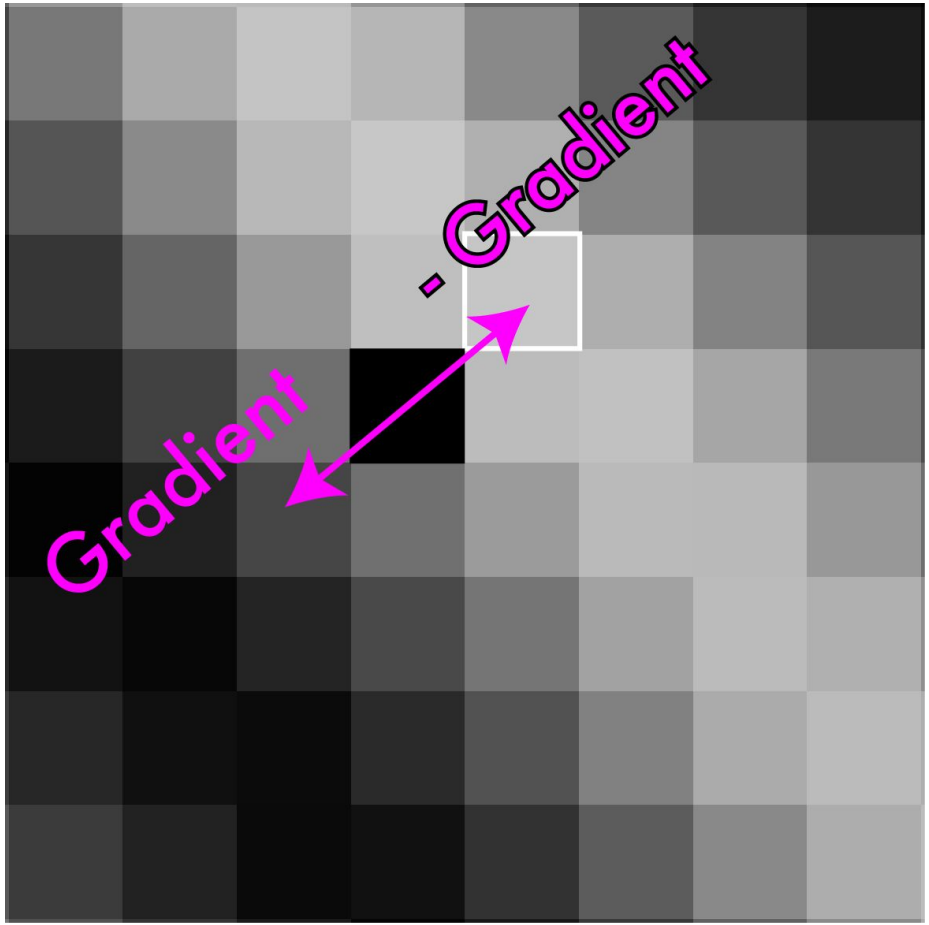
- Want single pixel edges, not thick blurry lines
- Need to check nearby pixels
- See if response is highest

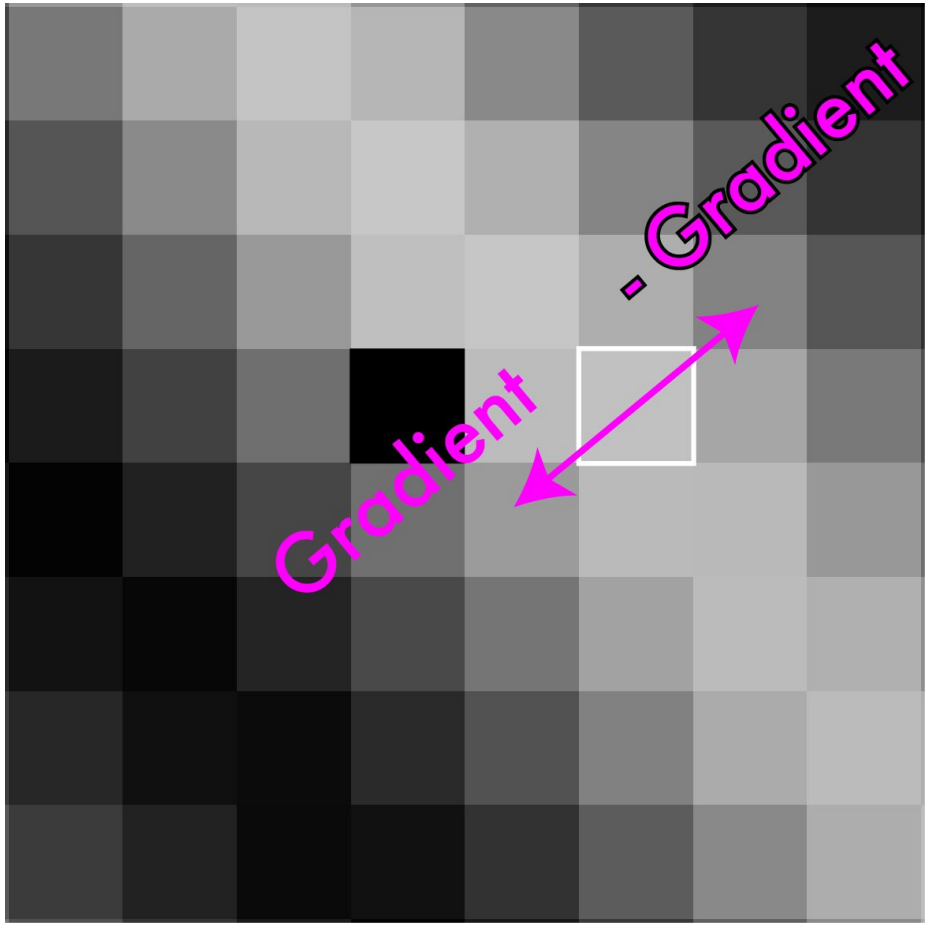
Non-maximum suppression

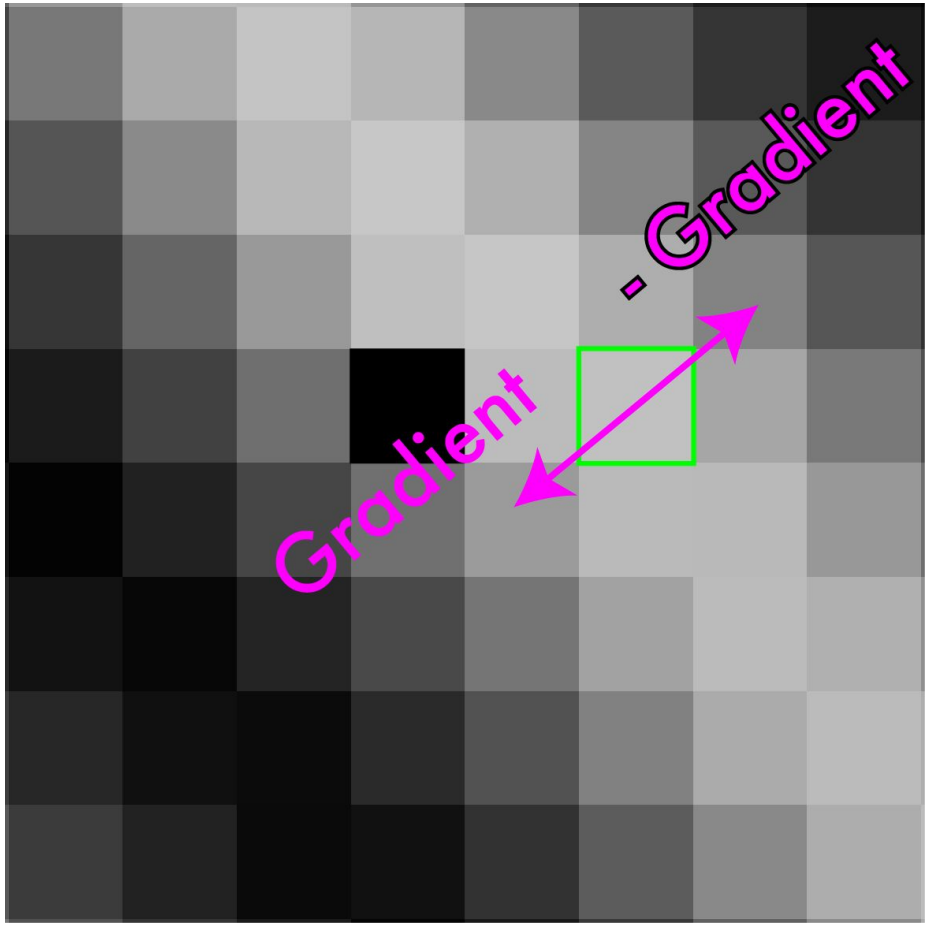


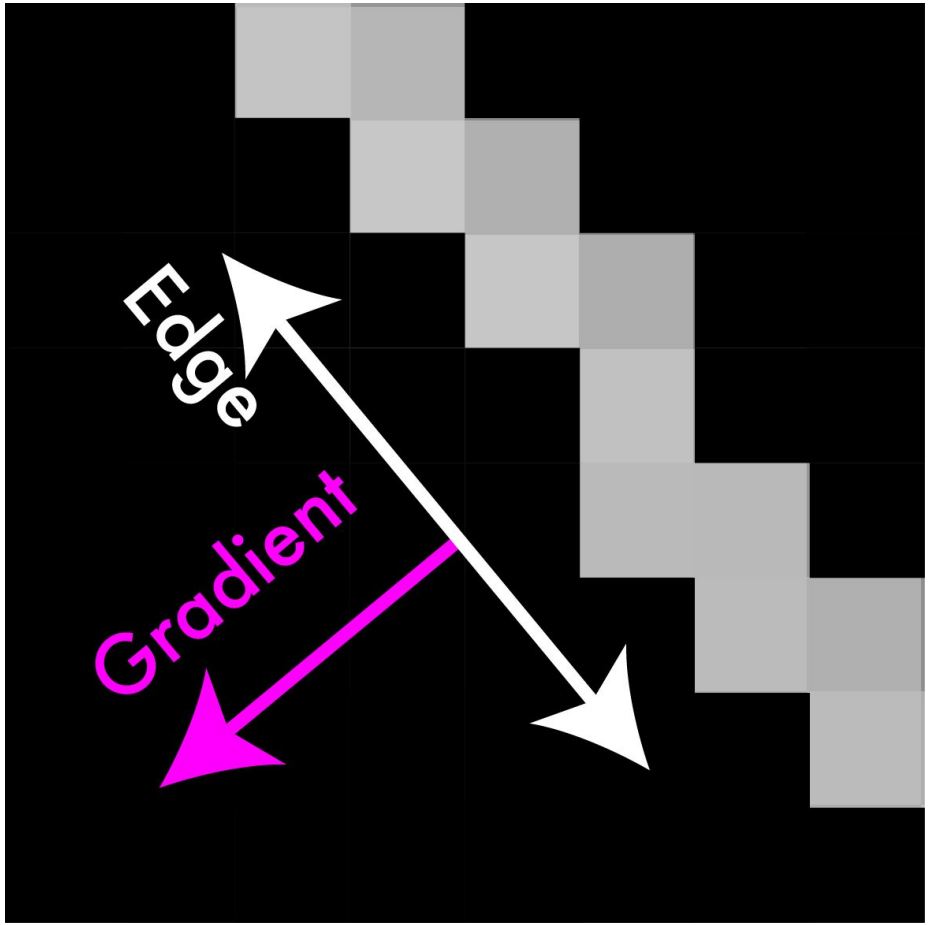

- Gradient
Gradient


- Gradient
Gradient

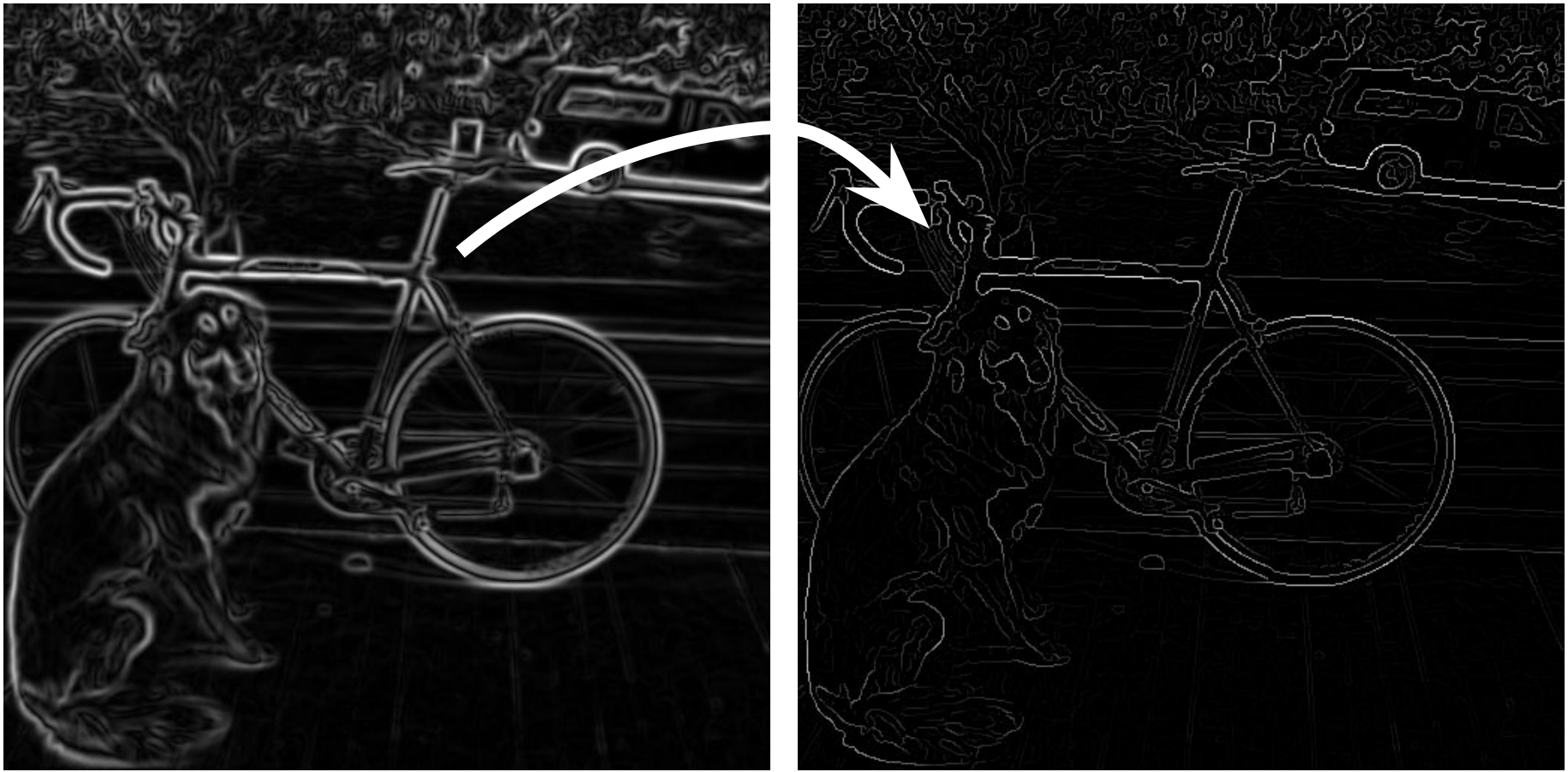

- Gradient
Gradient


- Gradient
Gradient


- Gradient
Gradient


Edge
Gradient

Non-maximum suppression



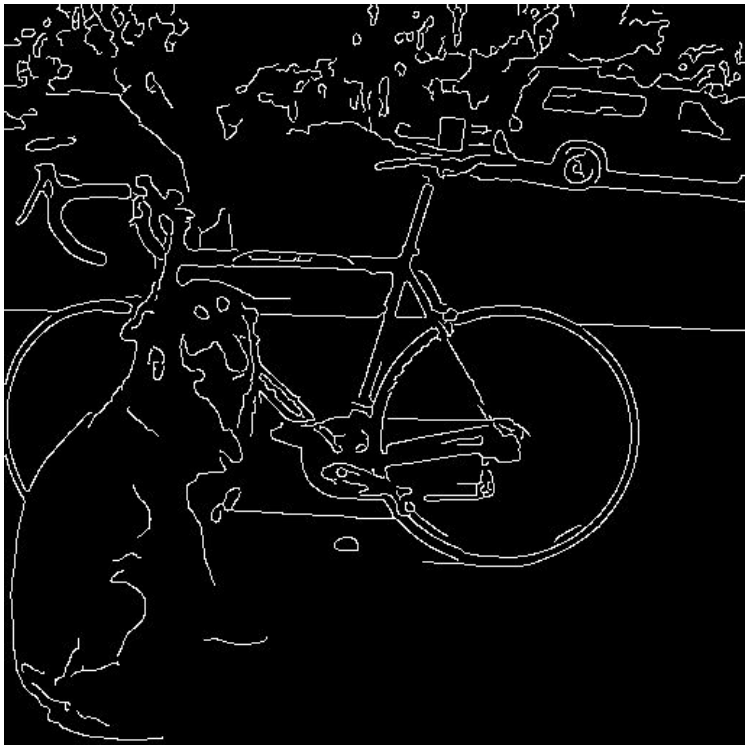
Threshold edges

- Still some noise
- Only want strong edges
- 2 thresholds, 3 cases
  - R > T: strong edge
  - R < T but R > t: weak edge
  - R < t: no edge
- Why two thresholds?

- Strong edges are edges!
- Weak edges are edges

  iff they connect to strong
- Look in some neighborhood

  (usually 8 closest)



Canny Edge Detection

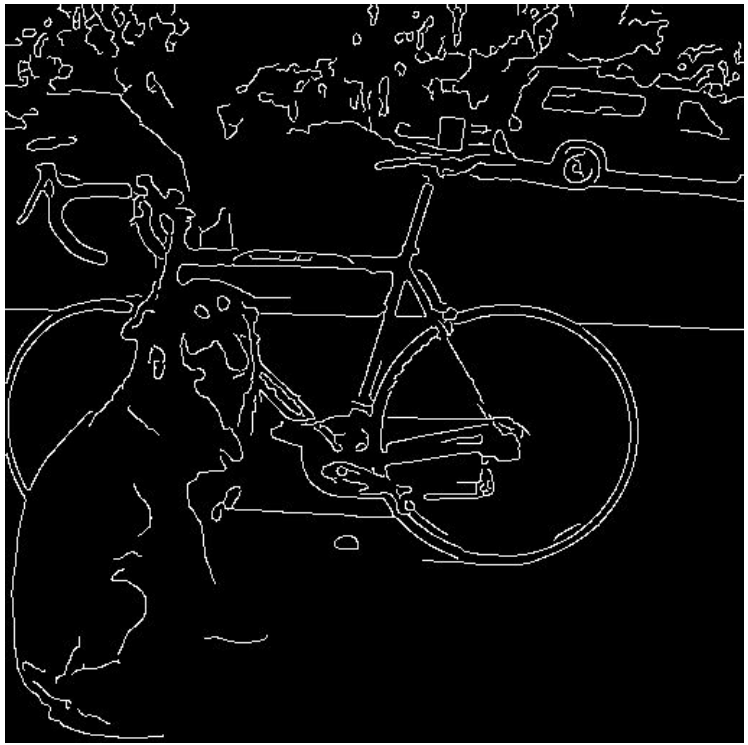
Algorithm:

- Smooth image (only want “real” edges, not noise)
- Calculate gradient direction and magnitude
- Non-maximum suppression perpendicular to edge
- Threshold into strong, weak, no edge
- Connect together components

```
# Canny Edge Detection

edges = cv2.Canny(image = img_blur, threshold1 = 100, threshold2 = 200)
```

Canny Edge Detection



http://bigwww.epfl.ch/demo/ip/demos/edgeDetector/

Q & A

Thank You !