Computer Vision

Lecture 5: Edge Detection

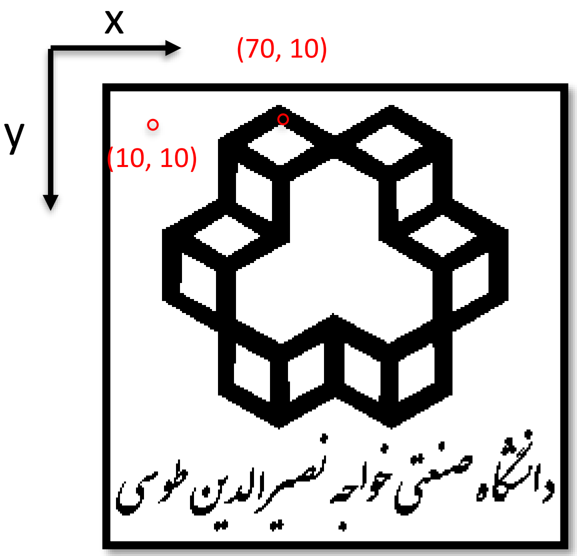
Dr. Esmaeil Najafi

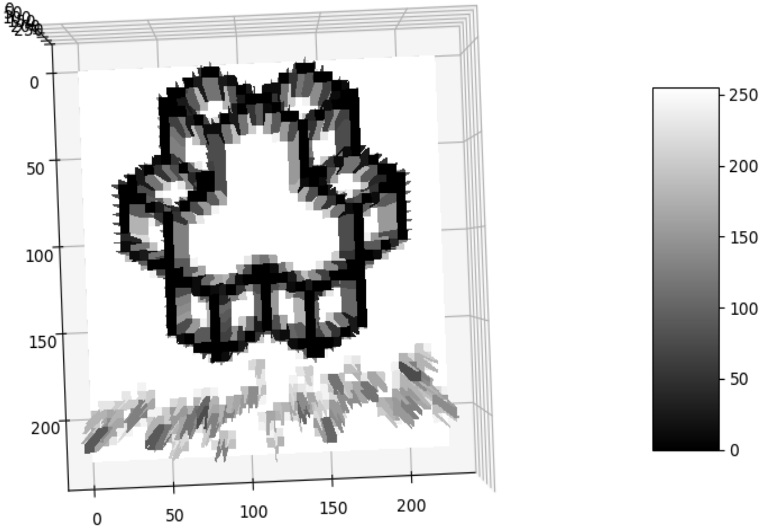
MSc. Javad Khoramdel

دانشگاه
خواجه نصیرالدین طوسی
K. N. Toosi University
of Technology

Image is a function

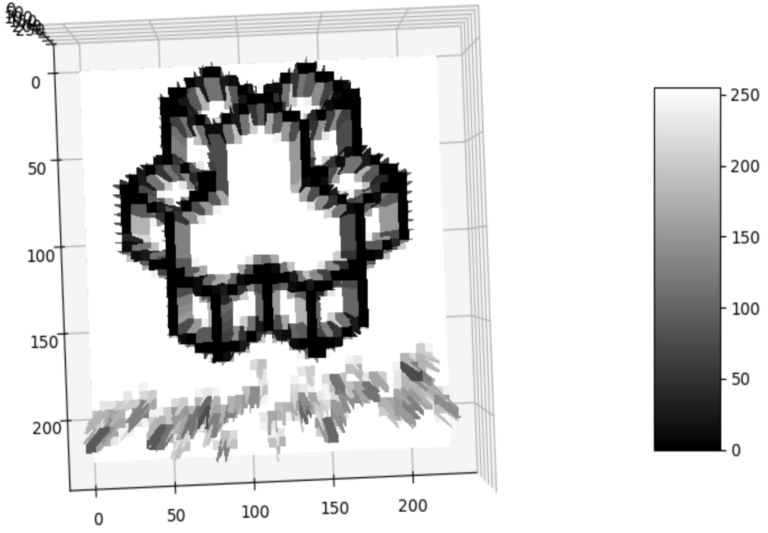
- Image is a function from spatial location to density.
  - Image(10, 10) = 255
  - Image (70, 10) = 0

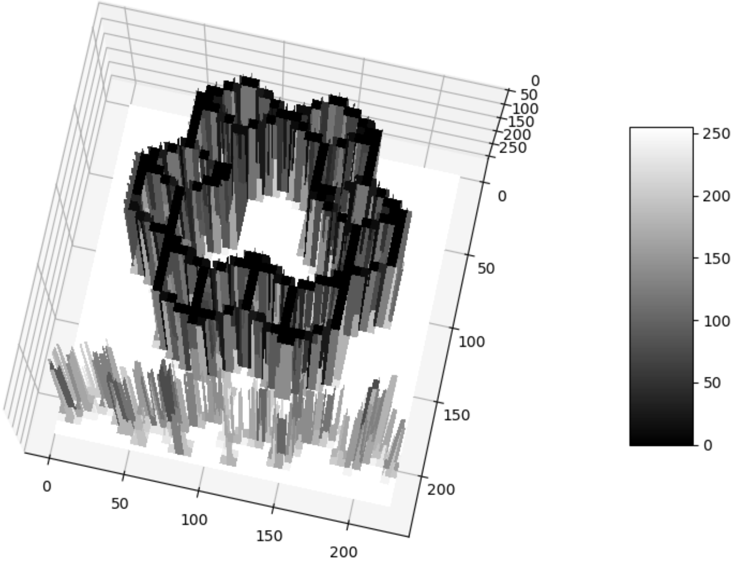




What’s an edge?

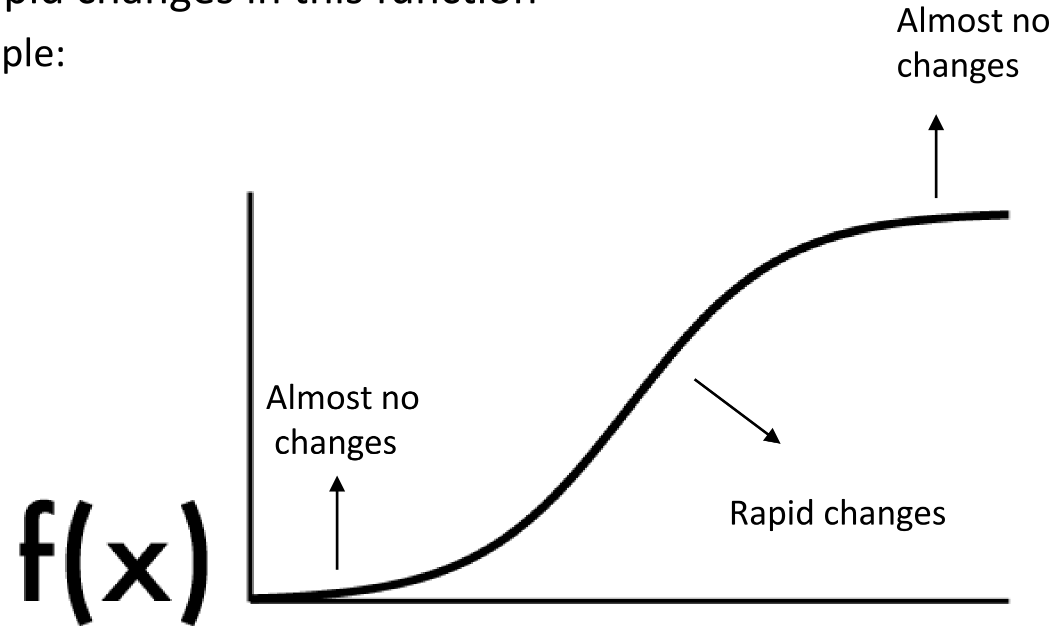
- Edges are rapid changes in this function





What's an edge?

- Edges are rapid changes in this function
  - 1D example:



Finding edges

- We can take derivative to spot the edges.
- Edges = high response

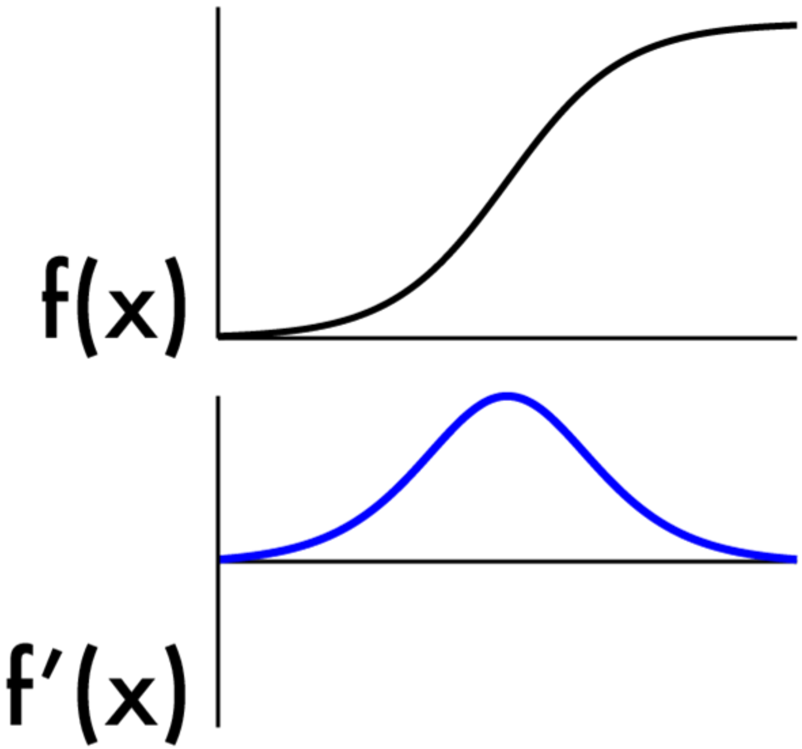


Image derivatives

- Recall:
  - $f'(a) = \lim_{h \to 0} \left(\frac{f(a+h) - f(a)}{h} \right)$
- We don't have an "actual" Function, must estimate
- Possibility: set h = 1
- What will that look like?

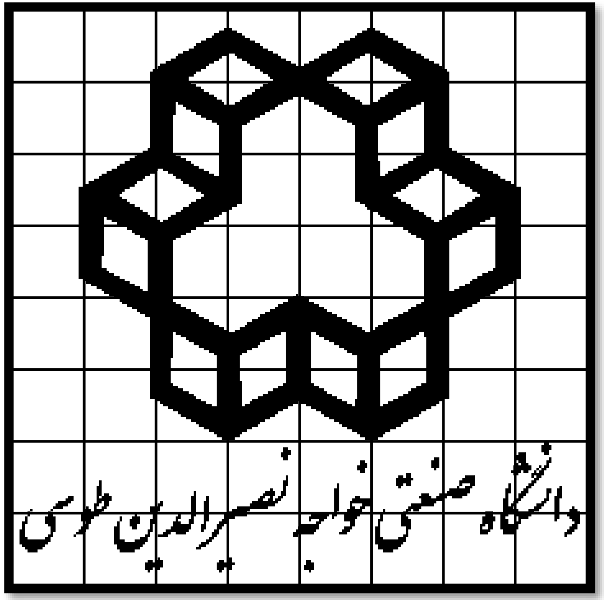
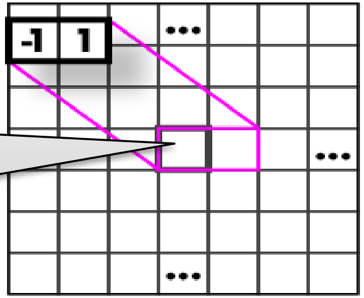


Image derivatives

- Recall:
  - $f'(a) = \lim_{h \to 0} \left(\frac{f(a+h) - f(a)}{h} \right)$
- We don’t have an “actual” Function, must estimate
- Possibility: set h = 1
- What will that look like?



We want to estimate the derivative at this location, but it seems the focus of this operation is not exactly at this location.

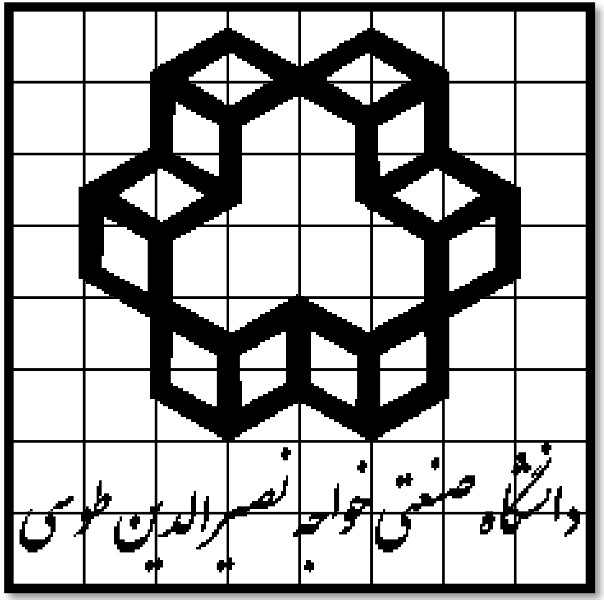
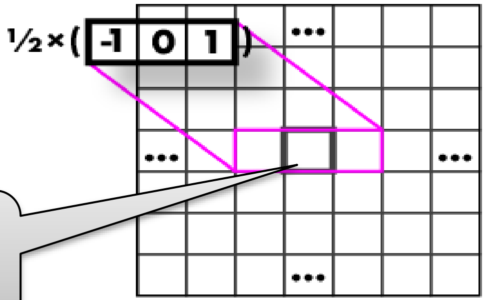
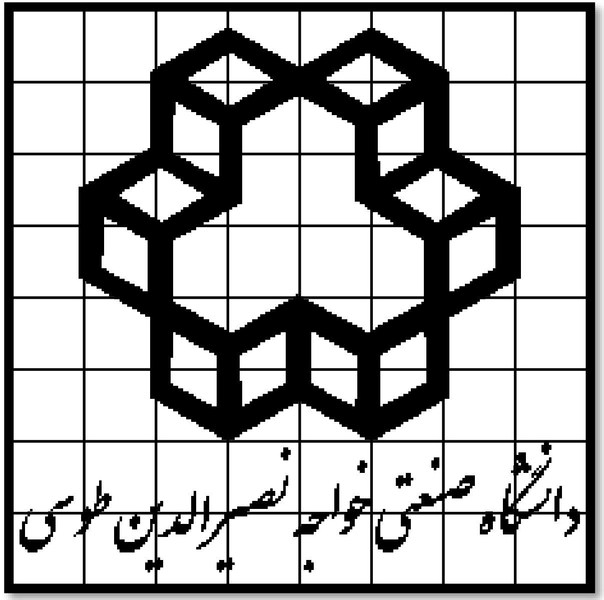


Image derivatives

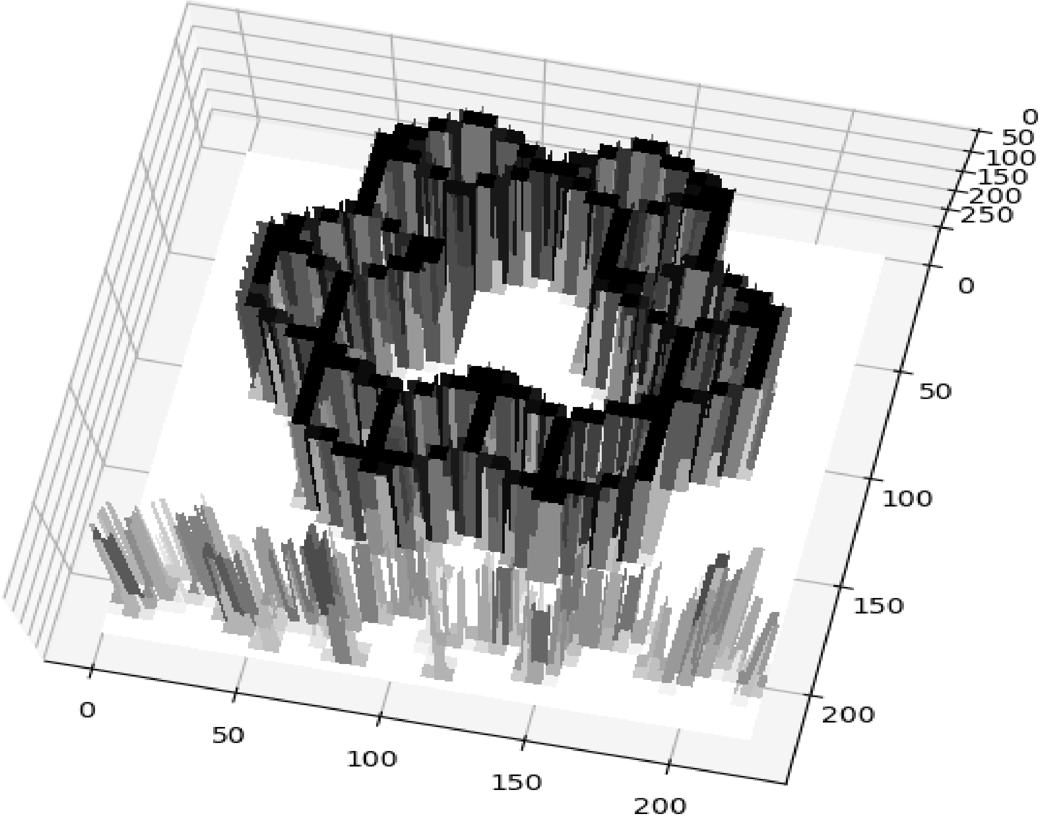
- Recall:
  - $f'(a) = \lim_{h \to 0}\left(\frac{f(a+h)-f(a)}{h}\right)$
- We don't have an "actual" Function, must estimate
- set h = 2
- What will that look like?





The focus is just where we want it to be.

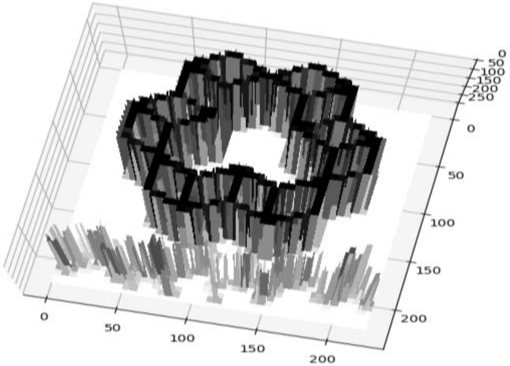
Images are noisy



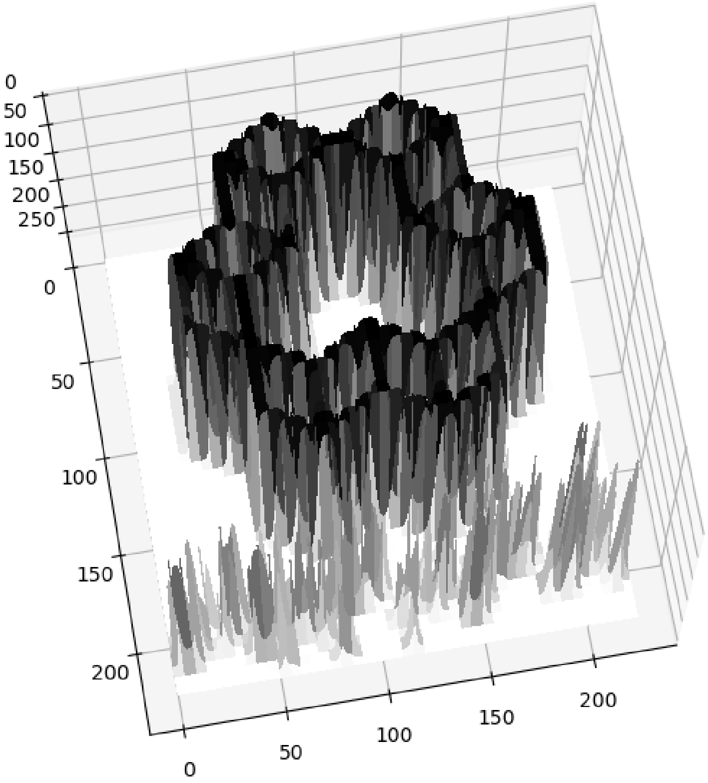
But we already know how to smooth!

1	2	1
2	4	2
1	2	1

*



=



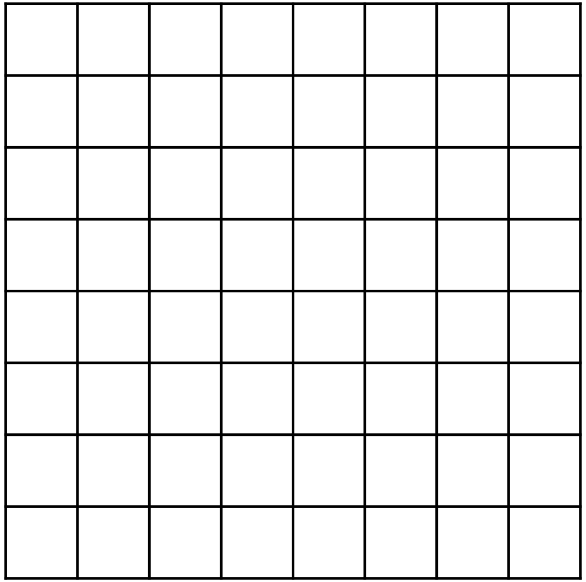
Gaussian filter

Raw image

Filtered image

Smooth first, then derivative

$$\frac{1}{2}\begin{bmatrix}-1 & 0 & 1\end{bmatrix} * \left(\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} * \text{[Raw image]} \right)$$



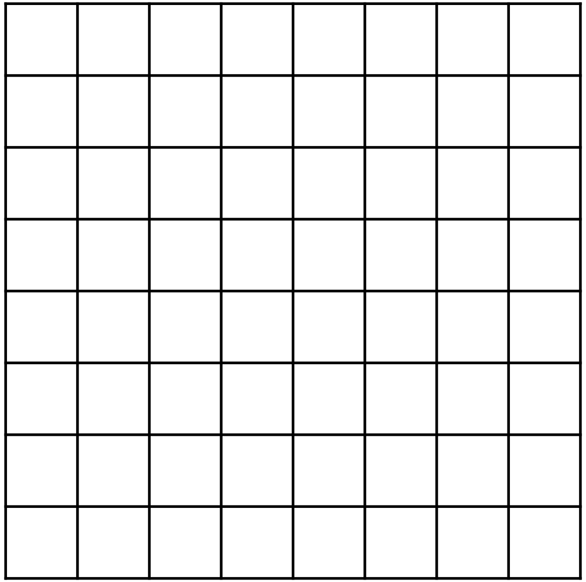
Derivative estimator filter (in x direction)

Gaussian filter

Raw image

Smooth first, then derivative

$$\left(\frac{1}{2} \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \right) *$$



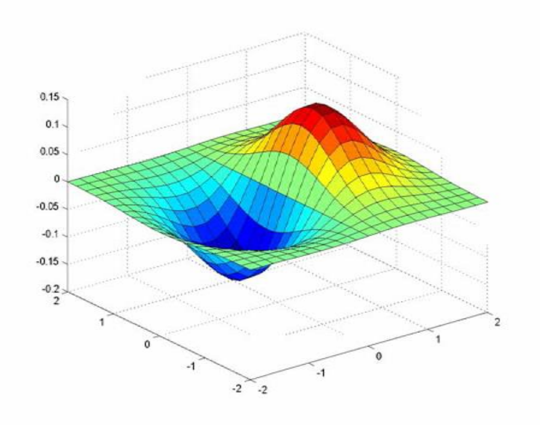
Derivative estimator filter (in x direction)

Gaussian filter

Raw image

Finding edges

$$\left(\frac{1}{2} \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \right) = \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array}$$



Derivative estimator filter (in x direction)

Gaussian filter

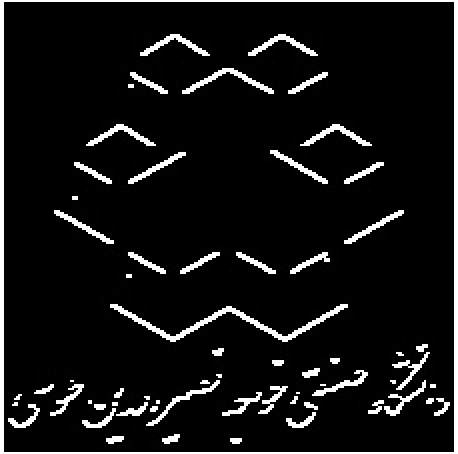
SobelX filter

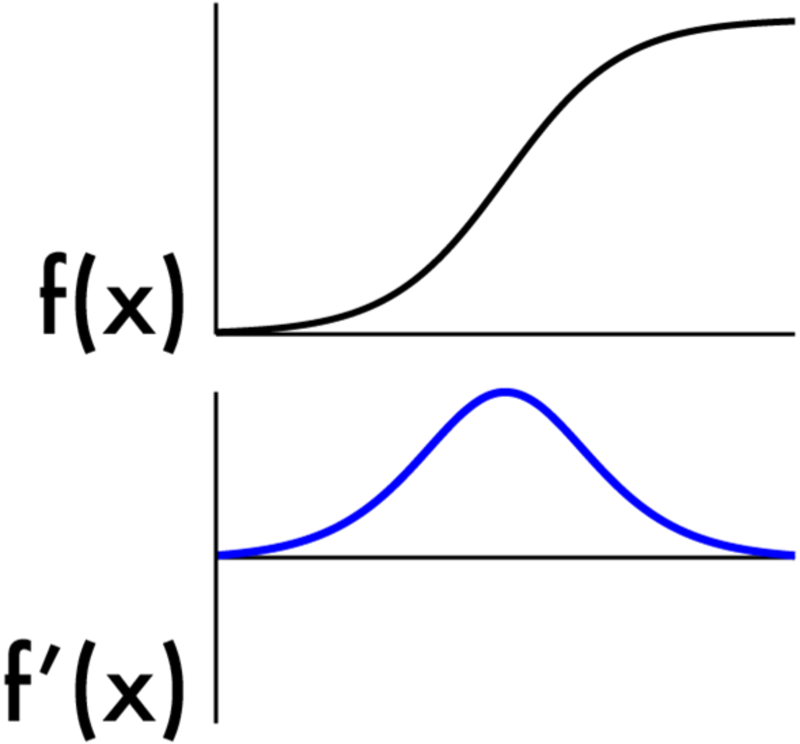
Finding edges

- We can take derivative with Sobel filters!
- But …

Filtered with SobelX

Filtered with SobelY

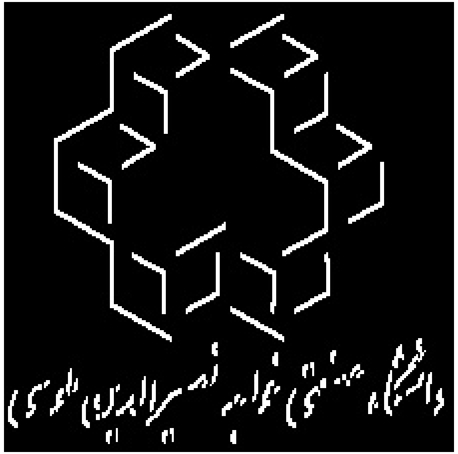




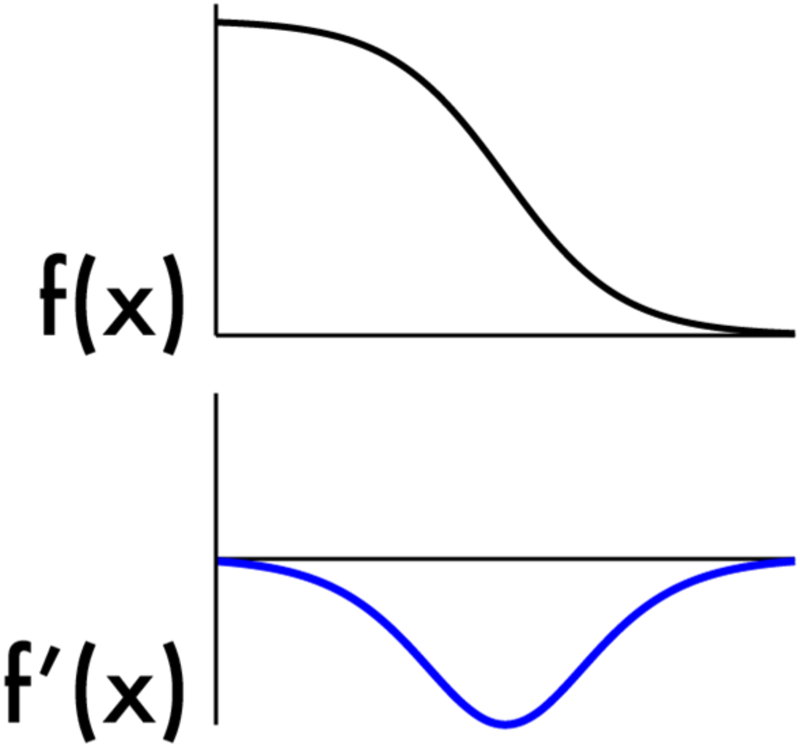
Finding edges

- We can take derivative with Sobel filters!
- But edges go both ways.

Filtered with
negative SobelX

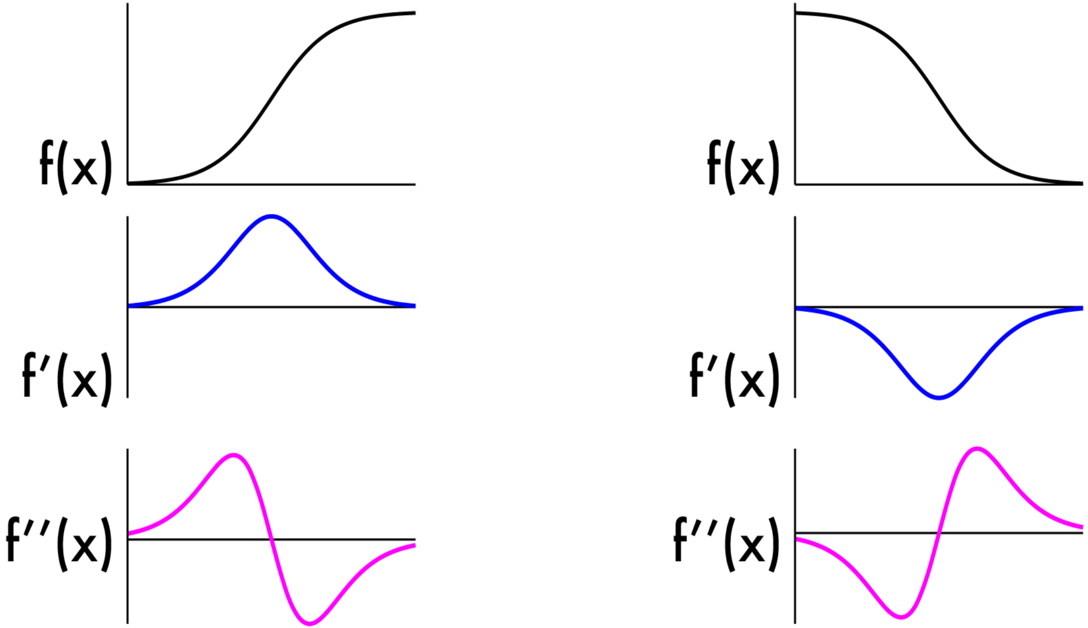


Filtered with
negative SobelY



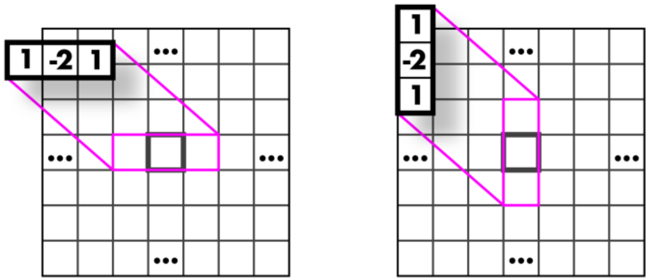
We want to find extrema

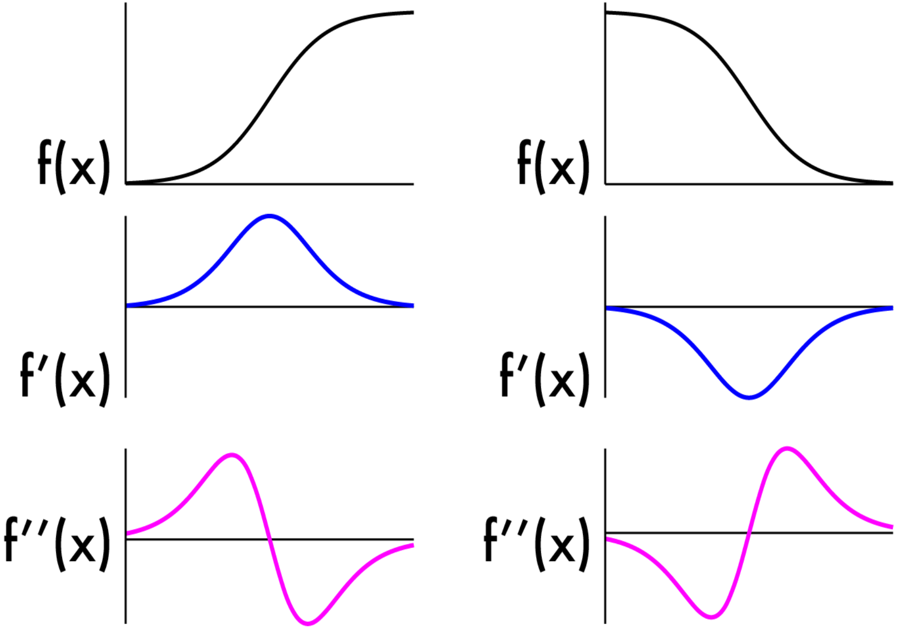
- 2nd derivative!
  - Crosses zero at extrema



Laplacian (2nd derivative)!

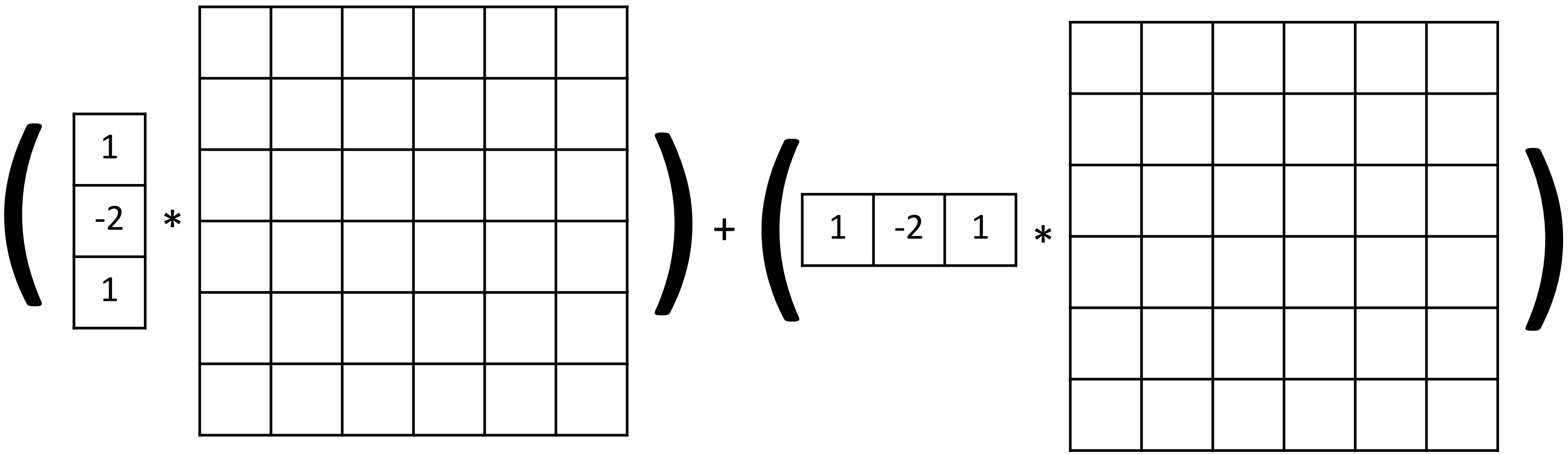
- Crosses zero at extrema
- Recall:
  - $f''(a) = \lim_{h \to 0} \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$
- Laplacian:
  - $\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$
- Again, have to estimate f''(x):





Laplacians

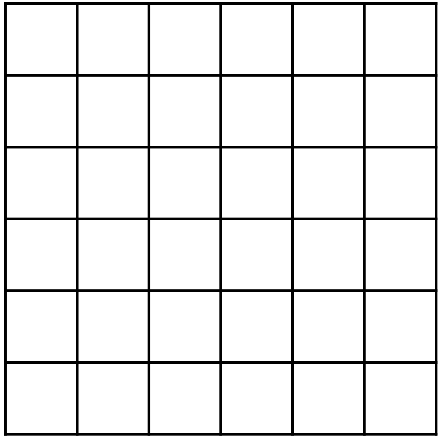
- Laplacian:
  - $\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$



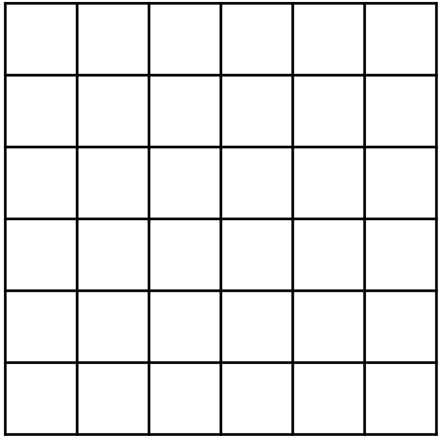
Laplacians

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

$$\underbrace{\left(\begin{bmatrix}1\\-2\\1\end{bmatrix} + \begin{bmatrix}1 & -2 & 1\end{bmatrix} \right)}_{\begin{bmatrix}0&1&0\\0&-2&0\\0&1&0\end{bmatrix} + \begin{bmatrix}0&0&0\\1&-2&1\\0&0&0\end{bmatrix}} *$$



$$= \begin{bmatrix}0&1&0\\1&-4&1\\0&1&0\end{bmatrix} *$$

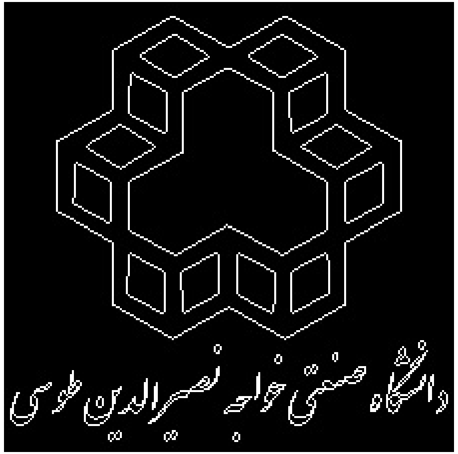


Laplacians

- Instead of using negative Laplacian (filter with a negative value in its middle), we can also use a positive Laplacian filter.

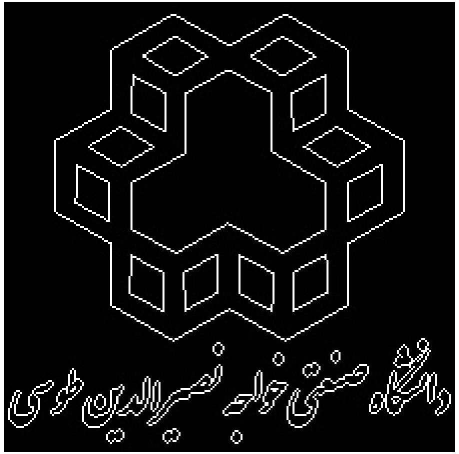
Negative Laplacian

0	1	0
1	-4	1
0	1	0



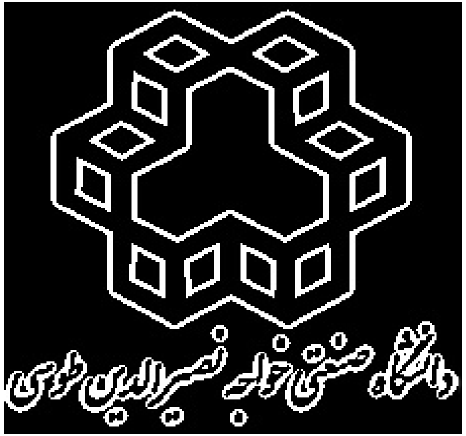
Positive Laplacian

0	-1	0
-1	4	-1
0	-1	0

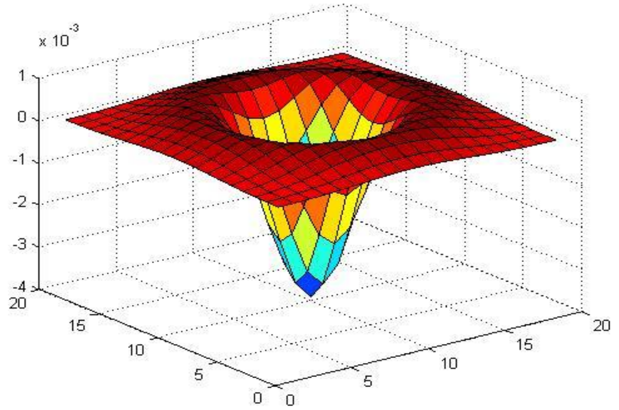


Laplacians also sensitive to noise

- Again, use gaussian smoothing
- We can just use one kernel since convs commute
- In other words, we can apply a gaussian filter to a Laplacian filter then use the result for finding the edges.
- This filter is called Laplacian of Gaussian (LoG)

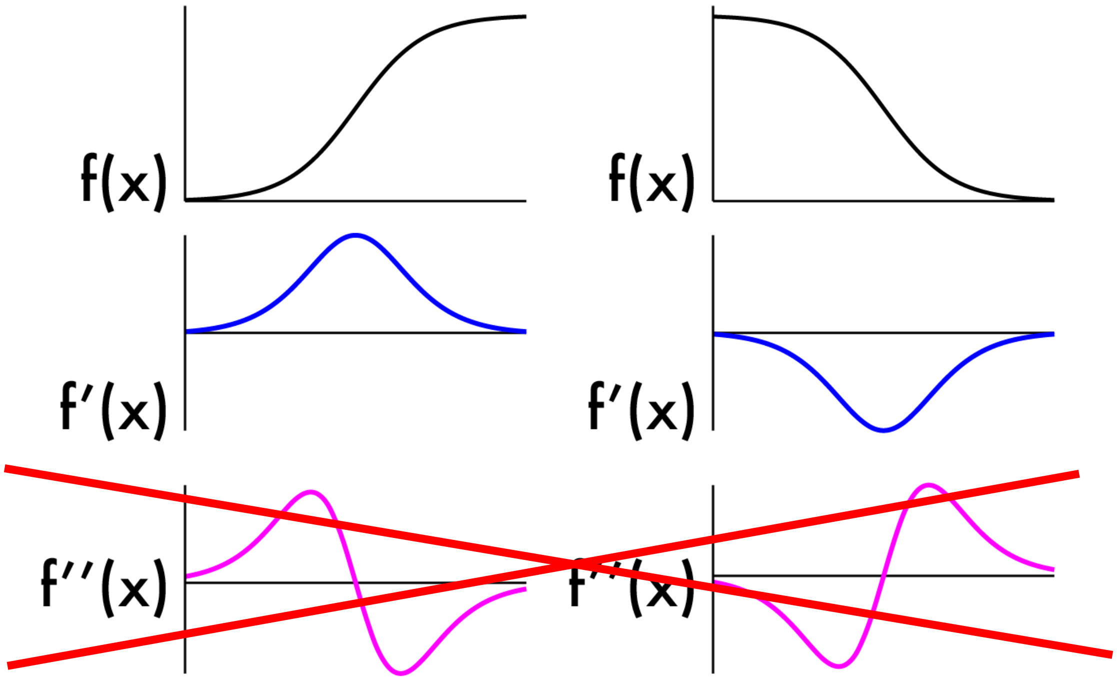


Filtered with LoG

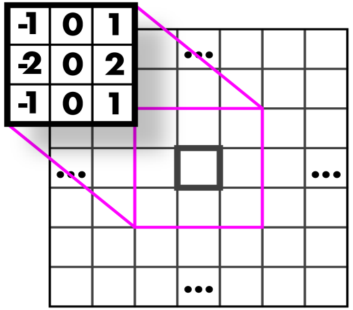


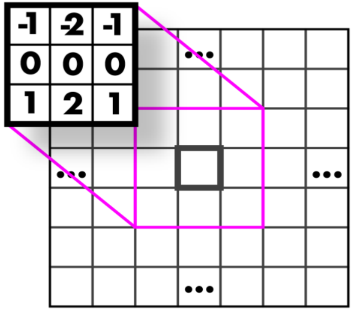
Another approach: gradient magnitude

- Don’t need 2nd derivatives because they are sensitive to noise.
- Just use magnitude of gradient
- But how?



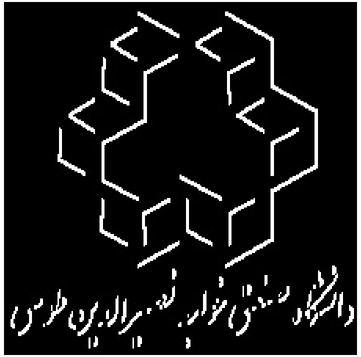
Another approach: gradient magnitude





By using x and y components of the gradient, we can find the gradient magnitude

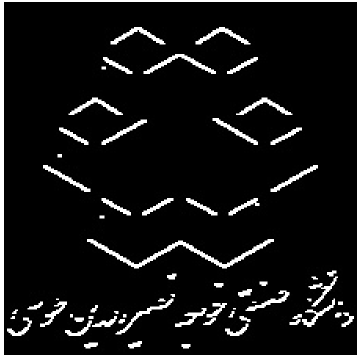
$\sqrt{}$



2

SobelX

+



2

SobelY

=

Gradient Magnitude

We are not done yet!

- Some edges are thicker than expected.
- There are some noisy points.
- What we should do now?
  - Canny edge detection!

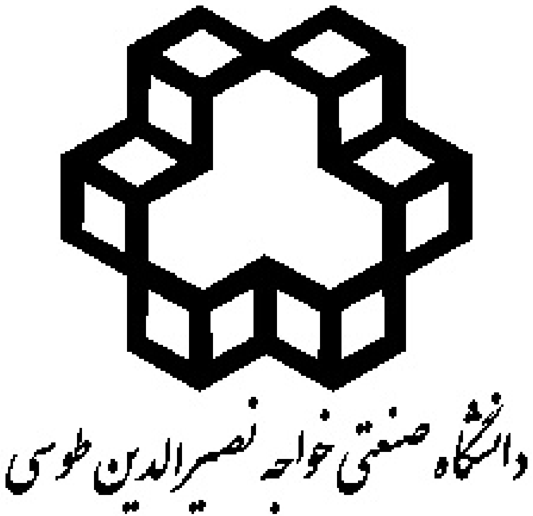
Canny edge detection

- Your first image processing pipeline!
  - Old-school computer vision is all about pipelines

- 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

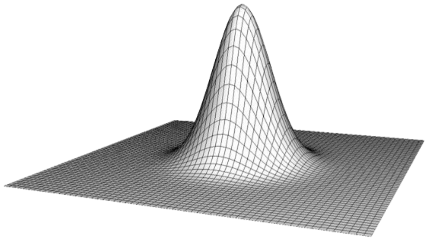
Smooth image

- You know how to do this, gaussians!



Raw image

*



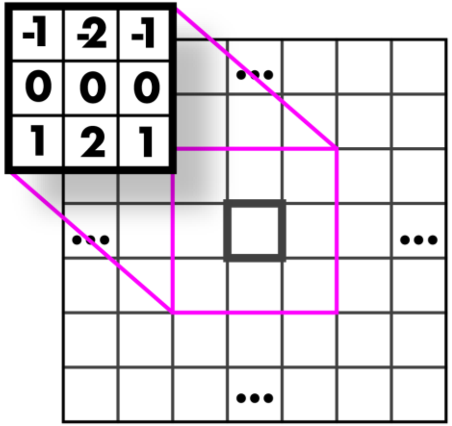
Gaussian filter

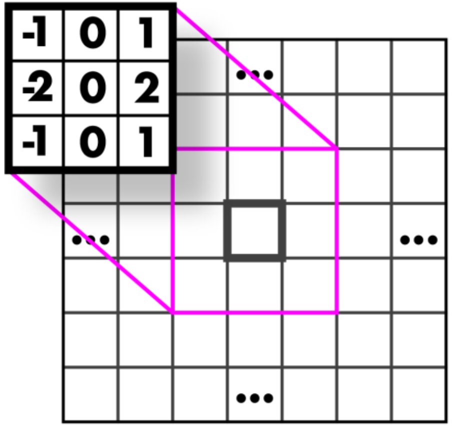
=

Blurred image

Gradient magnitude and direction

- Sobel filter
  - $Magnitude = \sqrt{SobelX^2 + SobelY^2}$
  - $Angle = Arctan2(SobelY, SobelX)$

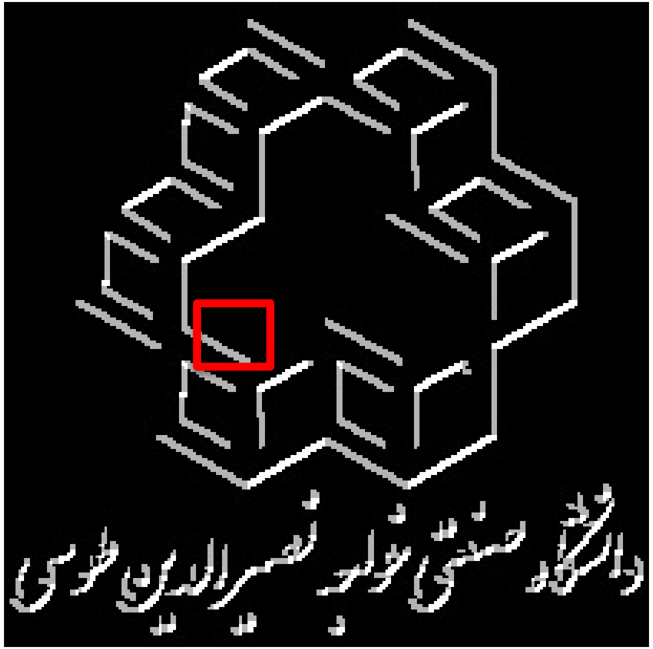




Gradient Magnitude

Non-maximum suppression

- We want single pixel edges, not thick blurry lines.
- We need to check nearby pixels and eliminate the additional pixels.

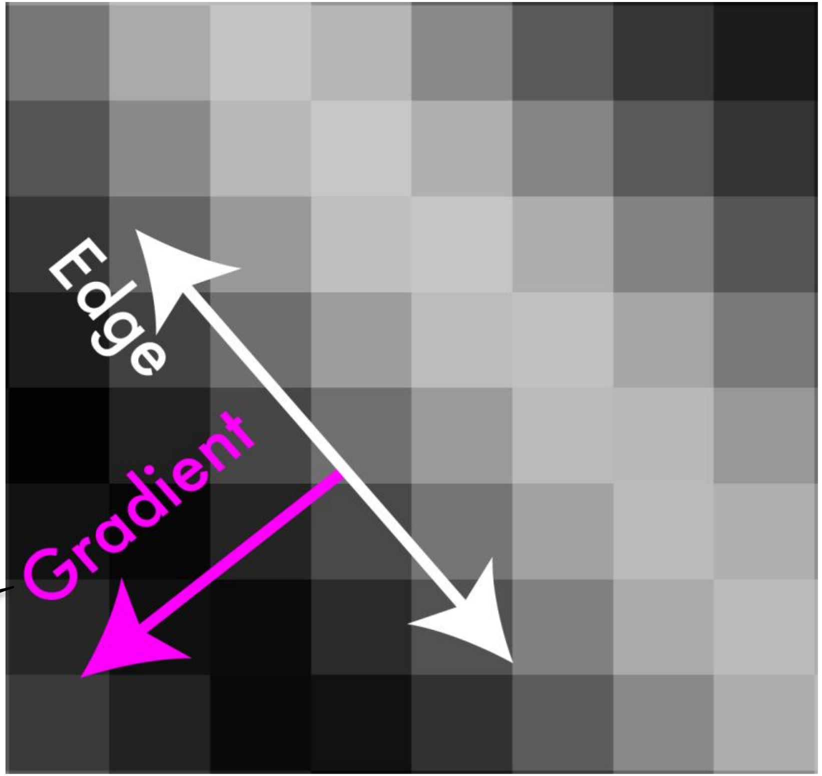


Gradient Magnitude

Non-maximum suppression

- For a given pixel, we compare its density with its neighbors in the direction of the gradient and the negative direction of the gradient.

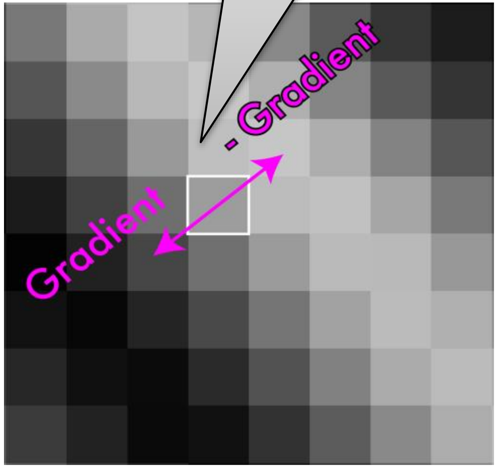
- If a pixel is brighter than its neighbors, we keep it; otherwise we eliminate it (i.e., we replace it with zero)

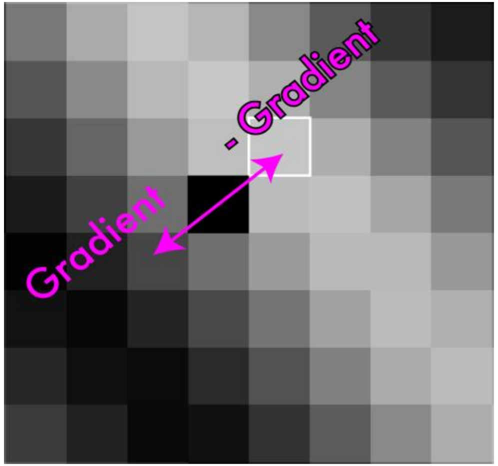


Gradient is in the direction of the highest changes, because of that, it is perpendicular to the edge.

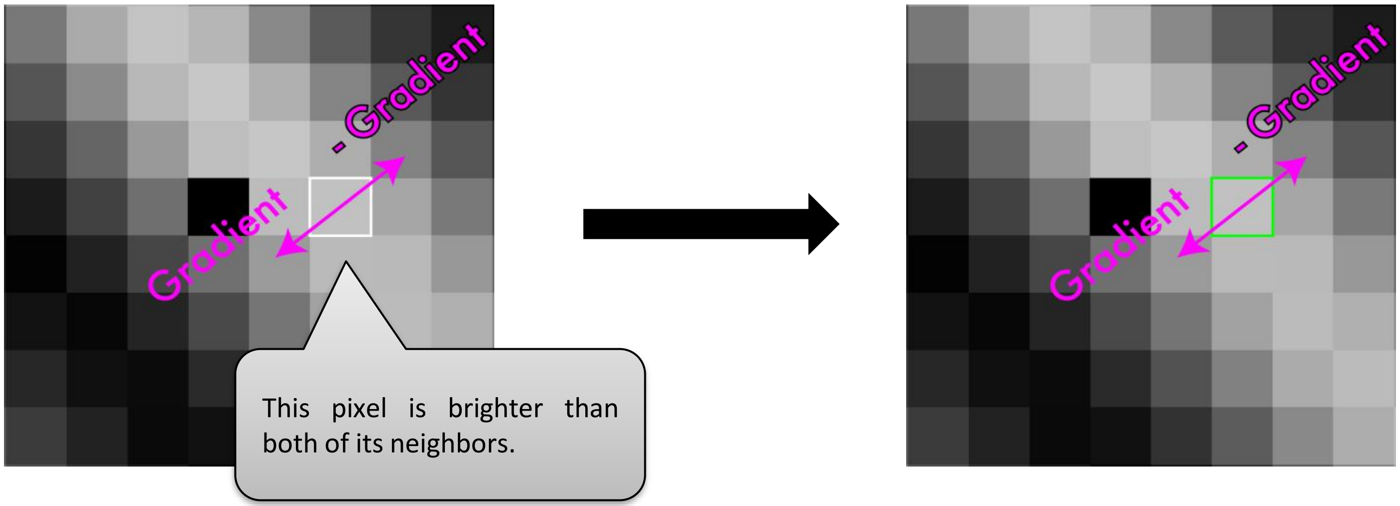
Non-maximum suppression

This pixel is not brighter than its neighbor in the negative direction of the gradient. Thus, we eliminate it.

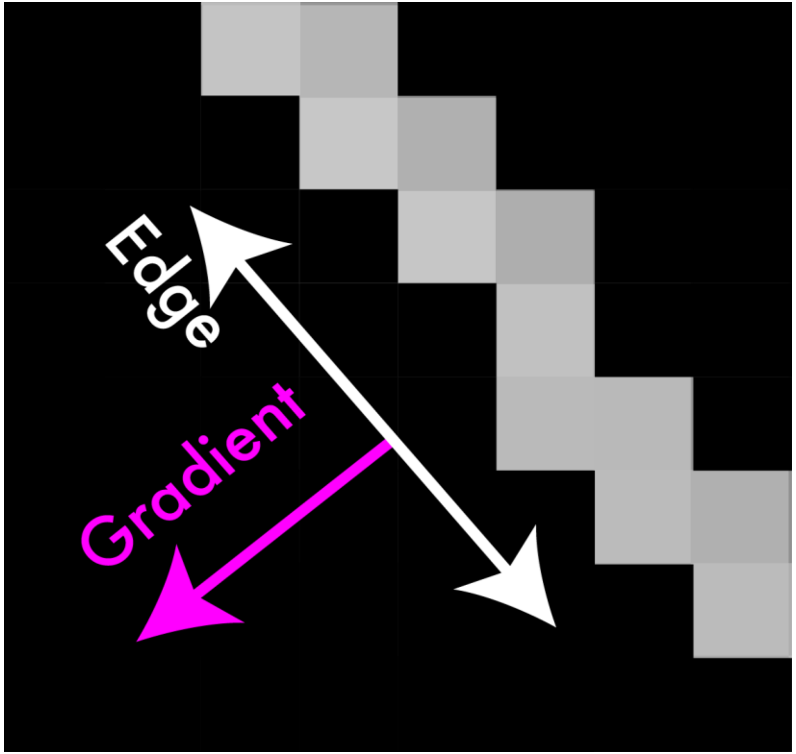




Non-maximum suppression



Non-maximum suppression

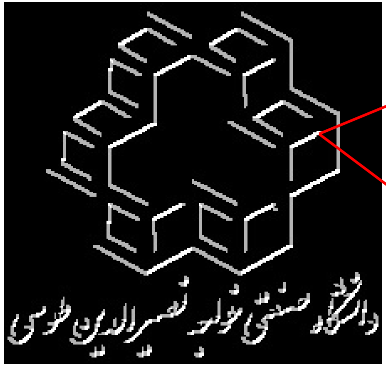


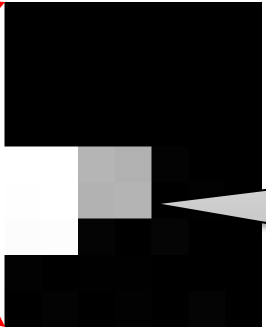
Threshold the edges

- Still there are some noise.
- We use 2 thresholds and classify each edge candidate based on these situations:
  - Pixel value > High threshold
    - ✓ strong edge
  - Pixel value < High threshold, but Pixel value > Low threshold
    - ✓ weak edge
  - Pixel value < Low Threshold
    - ✓ no edge
- Why two thresholds?

Connect 'em up!

- Strong edges are edges!
- Due to the noise, some edges which we expect to be strong edges may be affected and converted to weak edges.
  - That's why we use two thresholds!
- Weak edges are edges if and only if they connect to strong edges.
- We usually look at 8 closest neighbors of a weak edge point
- If there is a strong edge point in the neighborhood, we keep it, otherwise we eliminate it!





If we use a single threshold we may lose this point. But by using two thresholds and checking the neighborhood, we will see that this is a weak edge point which is connected to a strong edge neighbor.

Canny edge detection