

Lecture 5: Perception I

Professor Katie Driggs-Campbell

January 30, 2024

ECE484: Principles of Safe Autonomy

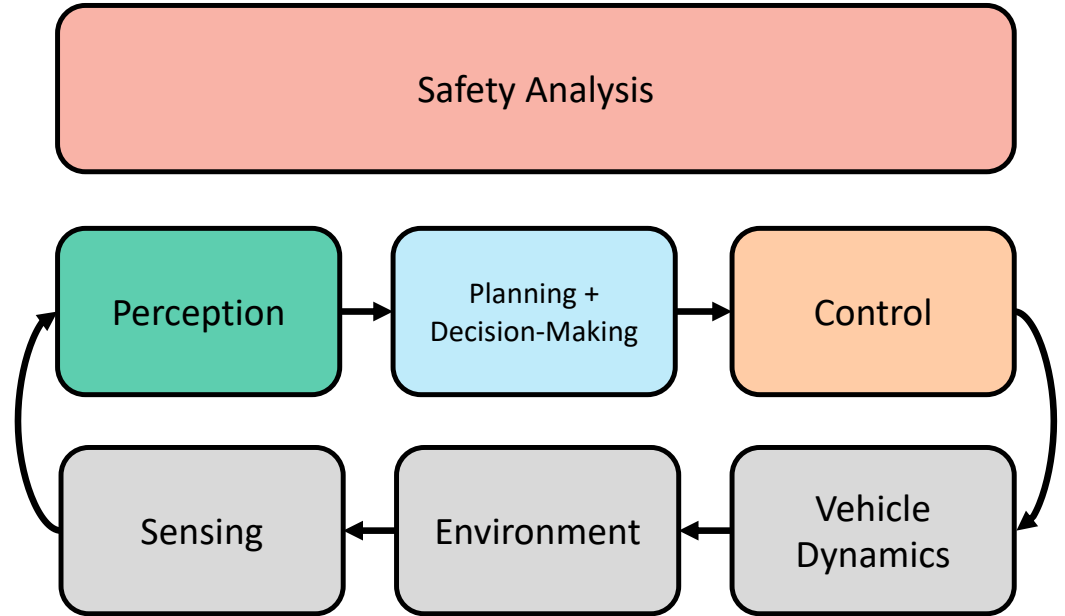


Administrivia

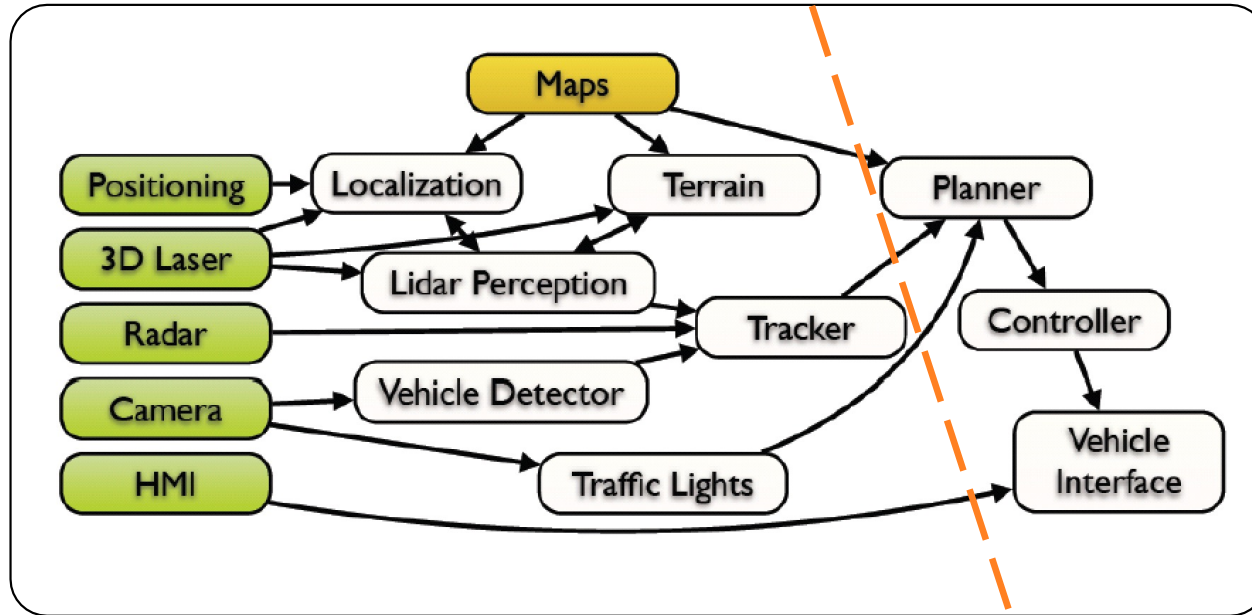
- MP1 released this week
- Upcoming due dates:
 - Form teams ASAP if you haven't already!
 - HW0 and MP0 due Friday 2/9
 - HW1 and MP1 due Friday 2/16
- commenting on pop quizzes



Autonomous GEM Vehicle



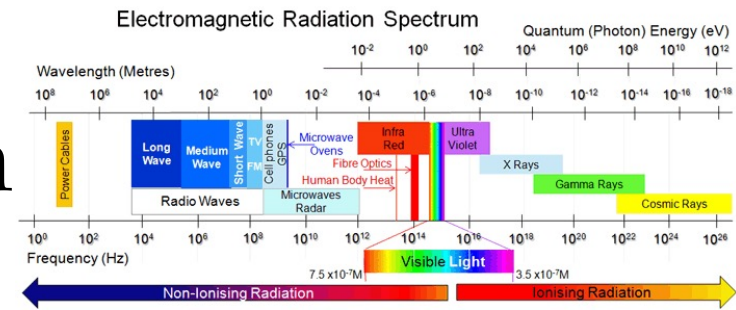
AV Perception Pipeline



This architecture from a slide from M. James of Toyota Research Institute, North America



The Challenge of Perception



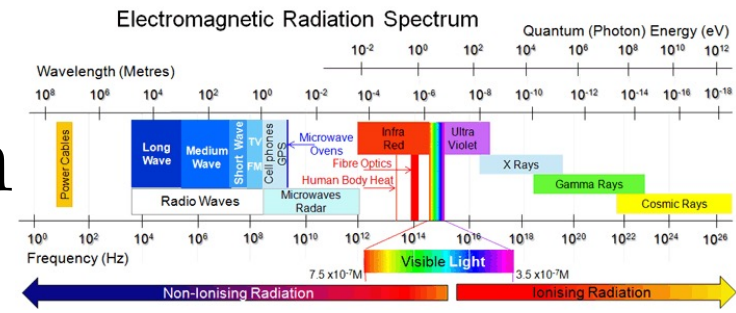
Sensor Goal: Process electromagnetic radiation from the environment to construct a *model* of the world, so that the constructed model is close to the real world and that the output is *actionable*

Challenging Questions:

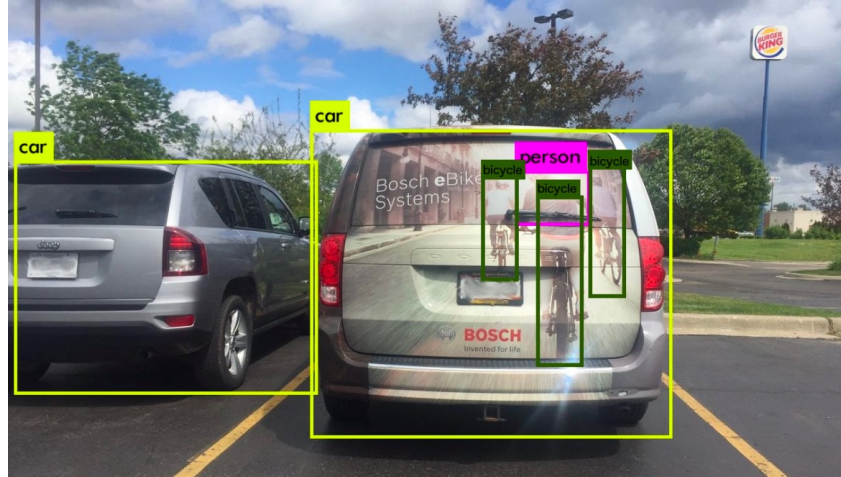
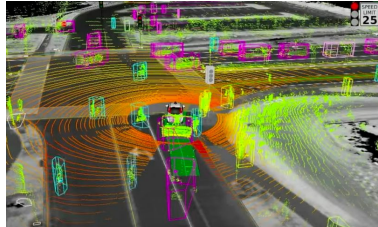
How does one define a car? How can you pass this information to a controller?



The Challenge of Perception



Sensor Goal: Process electromagnetic radiation from the environment to construct a *model* of the world, so that the constructed model is close to the real world and that the output is *actionable*



Today's Plan

- Basic image processing with filtering
- Edge detection



Motivation: Filtering for image de-noising



Modify the pixels in an image based on some function of a local neighborhood of the pixels

- **Scaling:** $\text{img}' = k * \text{img}$
- **Shifting right by s :** $\text{img}'[k] = \text{img}[k-s]$
 - $\text{img}'[0] \dots \text{img}'[s-1]$ is undefined

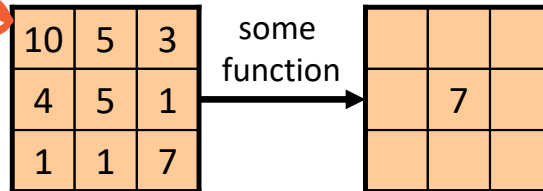


Motivation: Filtering for image de-noising



Modify the pixels in an image based on some function of a local neighborhood of the pixels

- Scaling: $\text{img}' = k * \text{img}$
- Shifting right by s : $\text{img}'[k] = \text{img}[k-s]$
 - $\text{img}'[0] \dots \text{img}'[s-1]$ is undefined
- Linear filtering: replace each pixel by a linear combination of neighbors

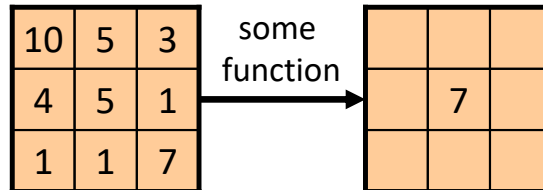


Motivation: Filtering for image de-noising



Modify the pixels in an image based on some function of a local neighborhood of the pixels

- Scaling: $\text{img}' = k * \text{img}$
- Shifting right by s : $\text{img}'[k] = \text{img}[k-s]$
 - $\text{img}'[0] \dots \text{img}'[s-1]$ is undefined
- Linear filtering: replace each pixel by a linear combination of neighbors



Defining convolution

Let f be the image and g be the ^{filter} kernel.

The output of convolving f with g is denoted $f * g$.

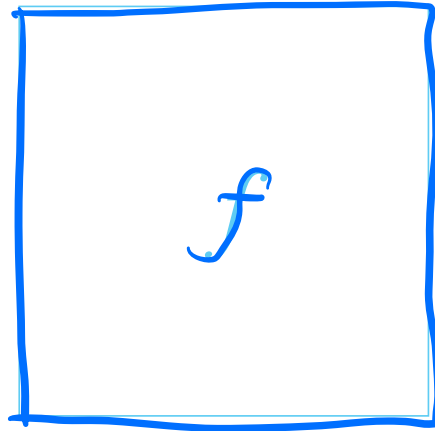
$$(f * g)[m, n] = \sum_{k, l} f[m-k, n-l] \cdot g[k, l]$$



Defining convolution

Let f be the image and g be the kernel.

The output of convolving f with g is denoted $f * g$.



Defining convolution

Let f be the image and g be the kernel.

The output of convolving f with g is denoted $f * g$.

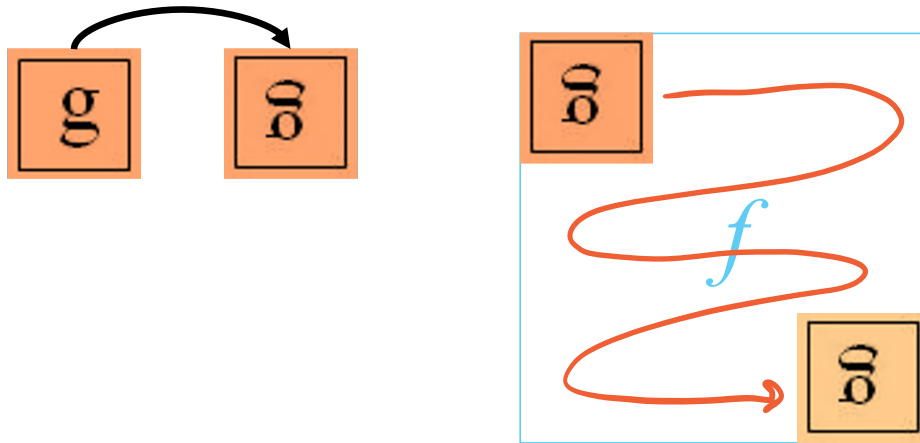


Image Convolution

g

1,1	1,2	1,3
2,1	2,2	2,3
3,1	3,2	3,3



$$h[m,n] = (f * g)[m,n] = \sum_{k,l} f[m-k, n-l] g[k,l]$$

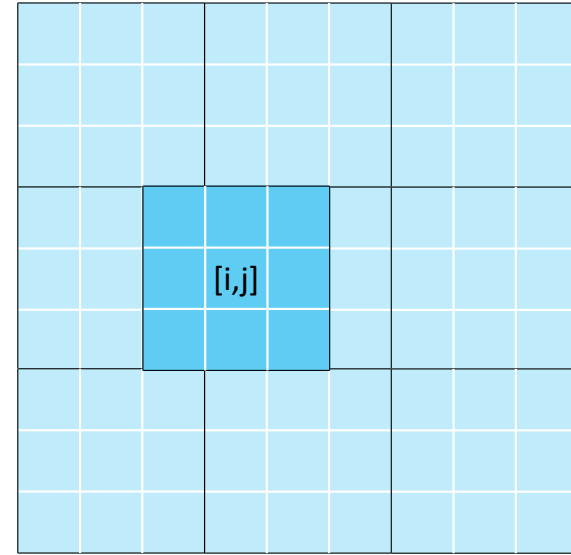
$$\begin{aligned} h[i,j] = & g[1,1] \cdot f[i-1, j-1] + g[1,2] \cdot f[i-1, j] + g[1,3] \cdot f[i-1, j+1] \\ & + g[2,1] \cdot f[i, j-1] + g[2,2] \cdot f[i, j] + g[2,3] \cdot f[i, j+1] \\ & + g[3,1] \cdot f[i+1, j-1] + g[3,2] \cdot f[i+1, j] + g[3,3] \cdot f[i+1, j+1] \end{aligned}$$



Example Filter

$$(f * g)[m,n] = \sum_{k,l} f[m-k, n-l] g[k,l]$$

1,1	1,2	1,3
2,1	2,2	2,3
3,1	3,2	3,3



$$\frac{1}{9} \begin{bmatrix} | & | & | \\ | & | & | \\ | & | & | \end{bmatrix}$$



Key properties

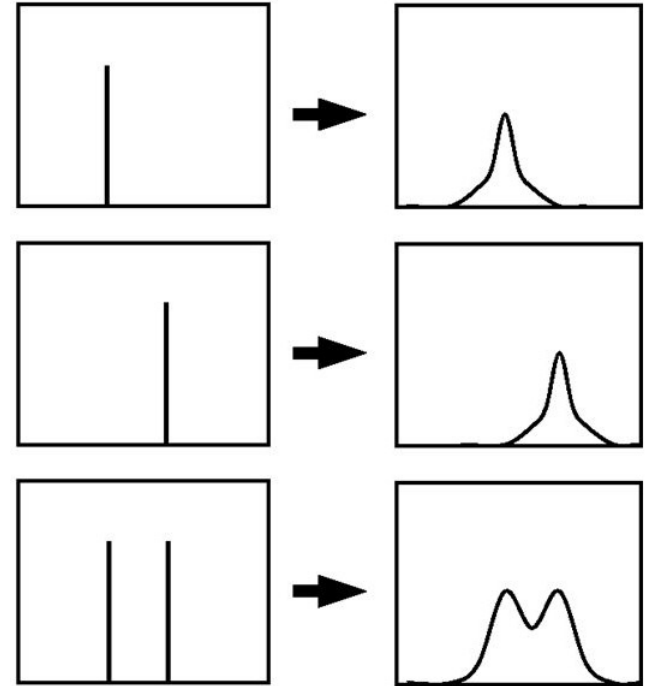
- **Shift invariance:** same behavior regardless of pixel location:

$$\text{filter}(\text{shift}(f)) = \text{shift}(\text{filter}(f))$$

- **Linearity:**

$$\text{filter}(f_1 + f_2) = \text{filter}(f_1) + \text{filter}(f_2)$$

- **Theoretical result:** any linear shift-invariant operator can be represented as a convolution

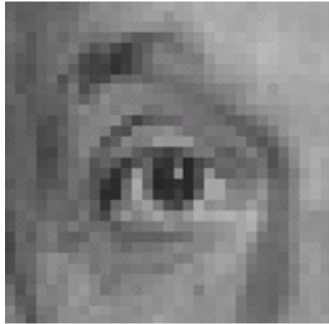


Properties in more detail

- Commutative: $a * b = b * a$
 - Conceptually no difference between filter and signal
- Associative: $a * (b * c) = (a * b) * c$
 - Often apply several filters one after another: $((a * b_1) * b_2) * b_3$
 - This is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$
- Distributes over addition: $a * (b + c) = (a * b) + (a * c)$
- Scalars factor out: $ka * b = a * kb = k(a * b)$
- Identity: unit impulse $e = [\dots, 0, 0, 1, 0, 0, \dots]$, $a * e = a$



Practice with linear filters (1)

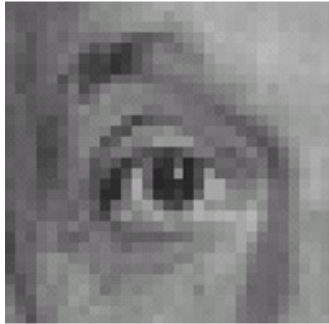


Original

0	0	0
0	1	0
0	0	0

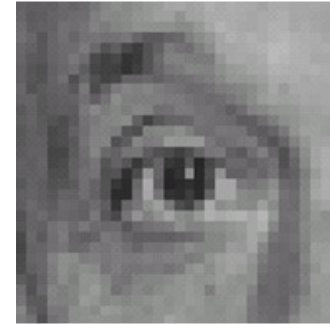


Practice with linear filters (1)



Original

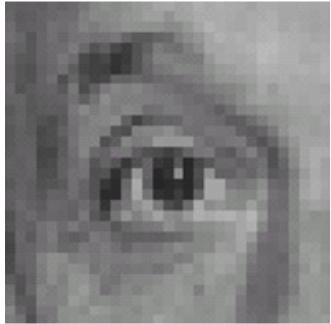
0	0	0
0	1	0
0	0	0



Filtered
(no change)



Practice with linear filters (2)

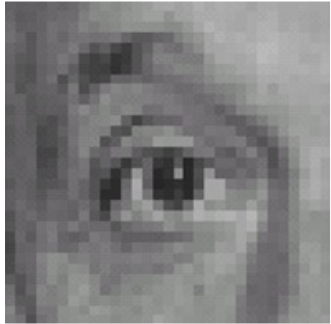


Original

0	0	0
0	0	1
0	0	0



Practice with linear filters (2)



Original

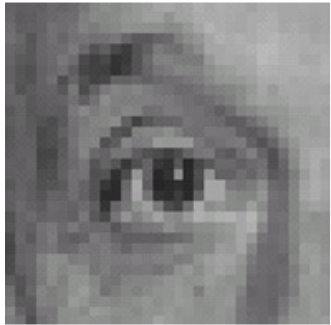
0	0	0
0	0	1
0	0	0



Shifted *left*
By 1 pixel



Practice with linear filters (3)

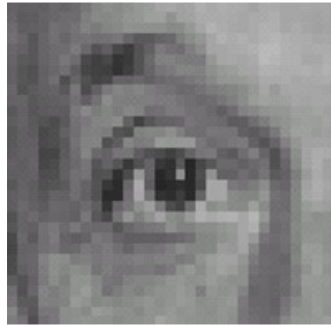


Original

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

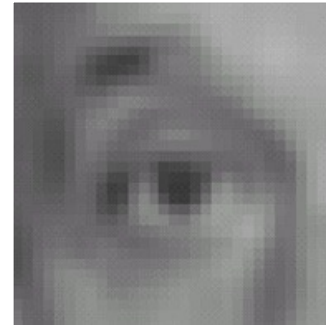


Practice with linear filters (3)



Original

$$\frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$



Blur (with a
box filter)



Practice with linear filters (4)



Original

0	0	0
0	2	0
0	0	0

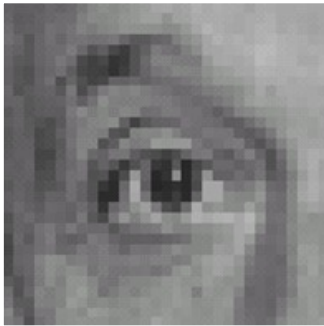
−

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

1	1	1
1	1	1
1	1	1

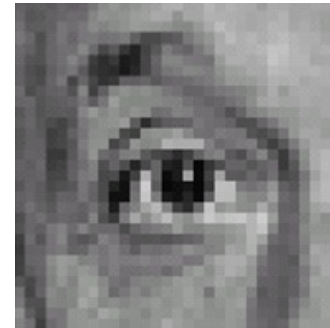


Practice with linear filters (4)



Original

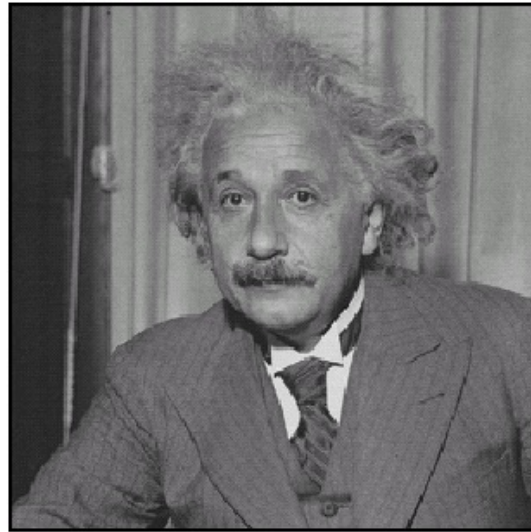
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



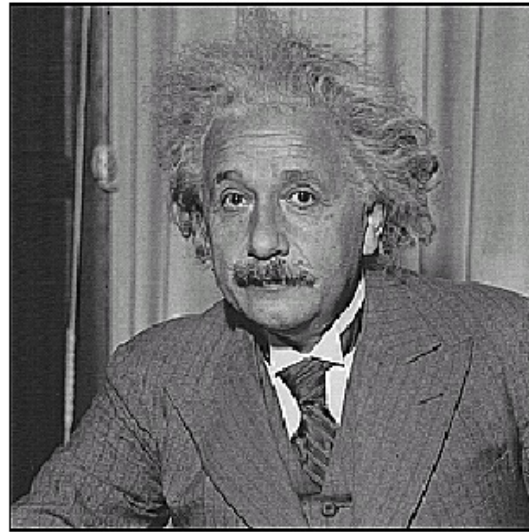
Sharpening filter: Accentuates differences with local average



Sharpening Filter



before



after



Sharpening

What does blurring take away?



Sharpening

What does blurring take away?



-



=



Sharpening

What does blurring take away?



-



=



Let's add it back:



+



Sharpening

What does blurring take away?



-



=



Let's add it back:



+

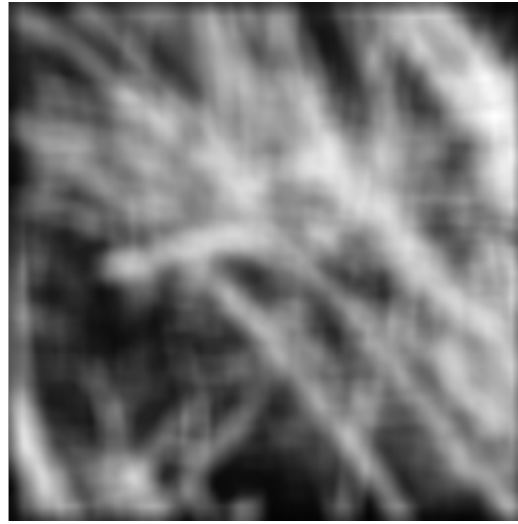


=

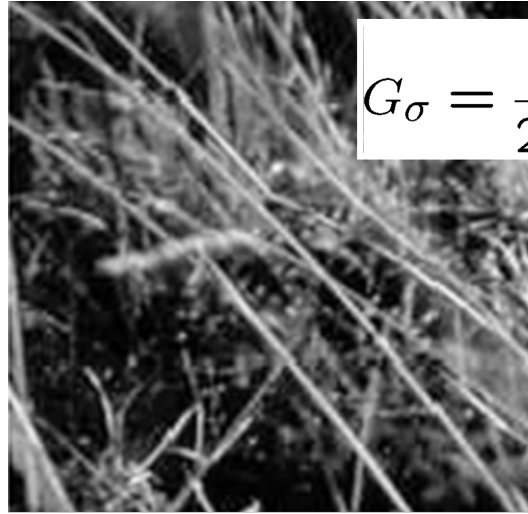


Soft Smoothing

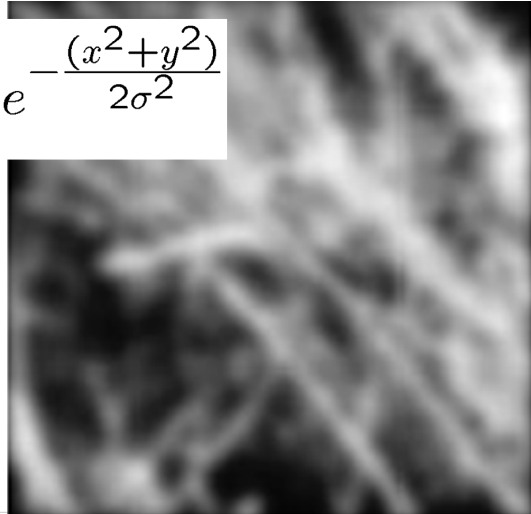
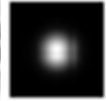
Box Filter



Gaussian Filter

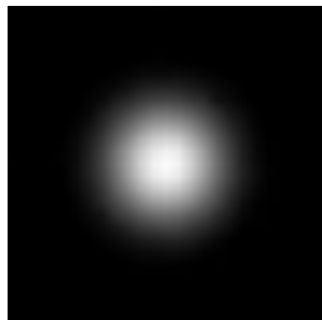
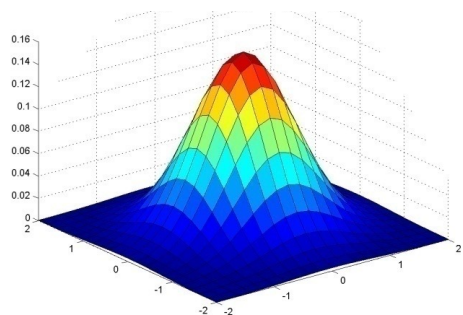


$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$



Gaussian Kernel

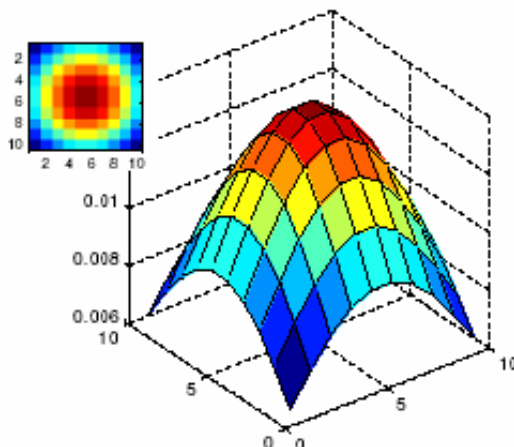
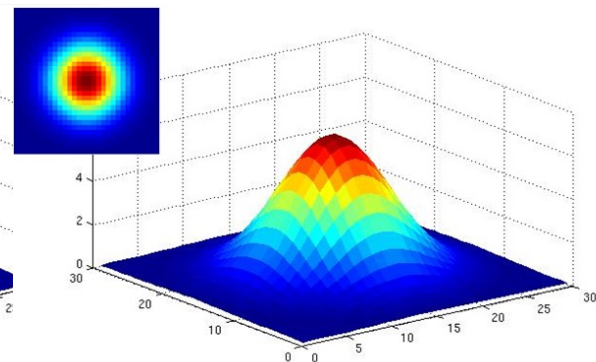
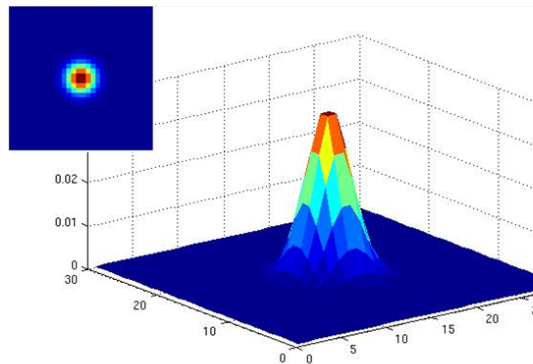
$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$



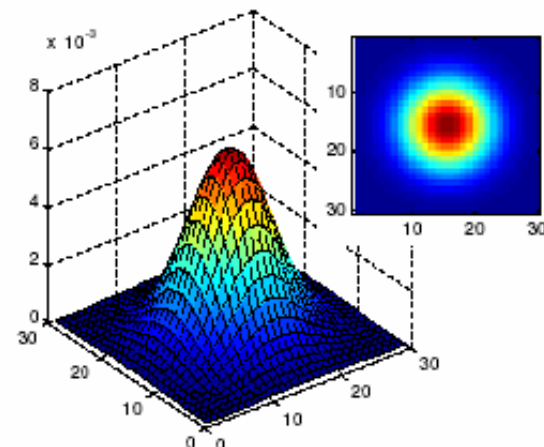
$g =$

0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, $\sigma = 1$



$\sigma = 5$ with 10x10 kernel



$\sigma = 5$ with 30x30 kernel



Edge detection



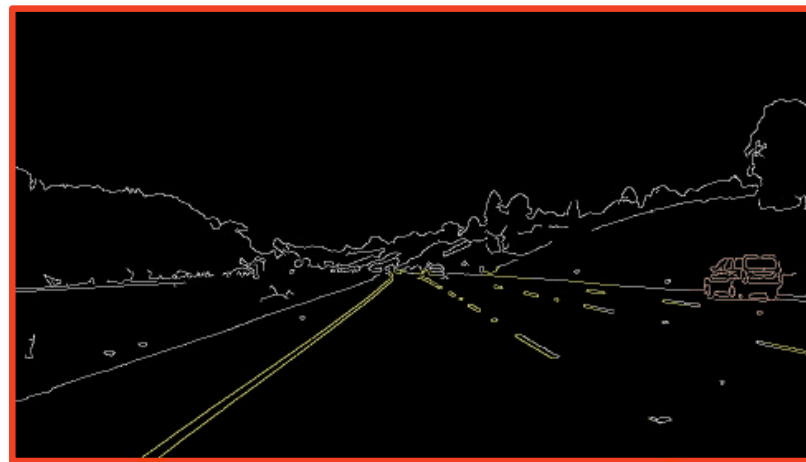
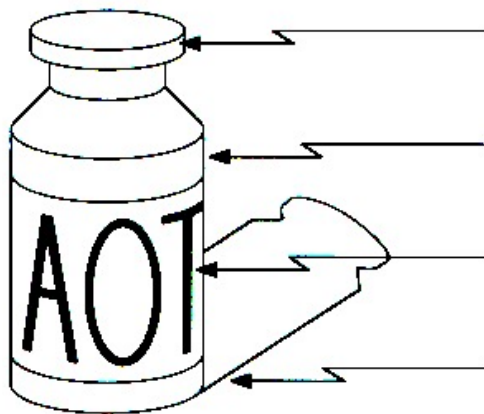
[Winter in Kraków photographed by Marcin Ryczek](#)



Edge detection

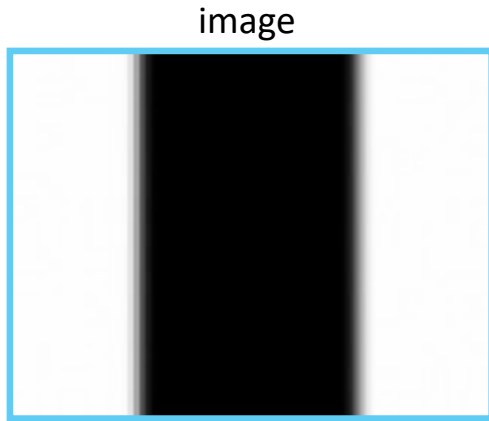
Goal: Identify sudden changes (discontinuities) in an image

→ Intuitively, edges carry most of the semantic and shape information from the image (e.g., lanes, traffic signs, cars)



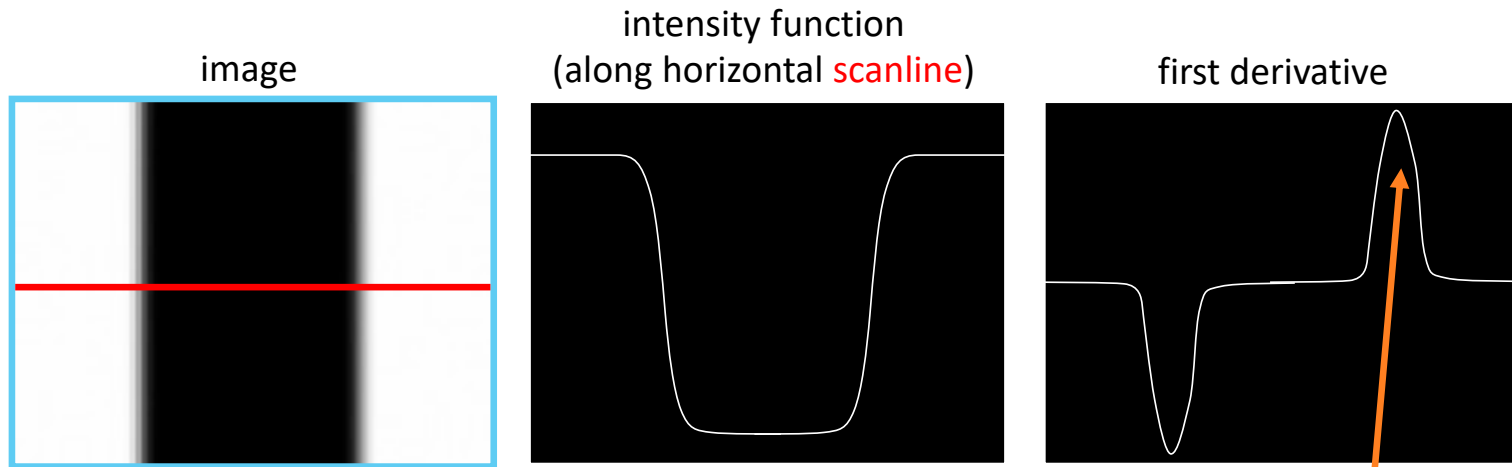
Edge detection

An edge is a place of rapid change in the image intensity function



Edge detection

An edge is a place of rapid change in the image intensity function



edges correspond to
extrema of derivative



Partial Derivatives

- For $f(x, y)$, the partial derivative with respect to x is:

$$\frac{\partial f(x, y)}{\partial x} = \lim_{\epsilon \rightarrow 0} \frac{f(x + \epsilon, y) - f(x, y)}{\epsilon}$$

- The finite difference equation is:

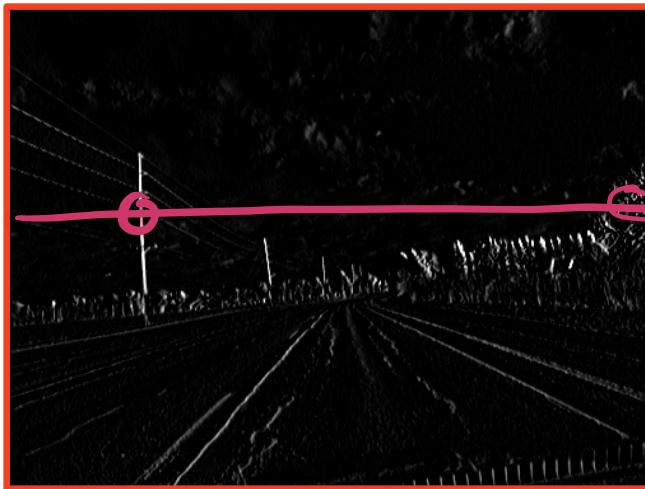
$$\frac{\partial f(x, y)}{\partial x} \approx \frac{f(x + 1, y) - f(x, y)}{1}$$



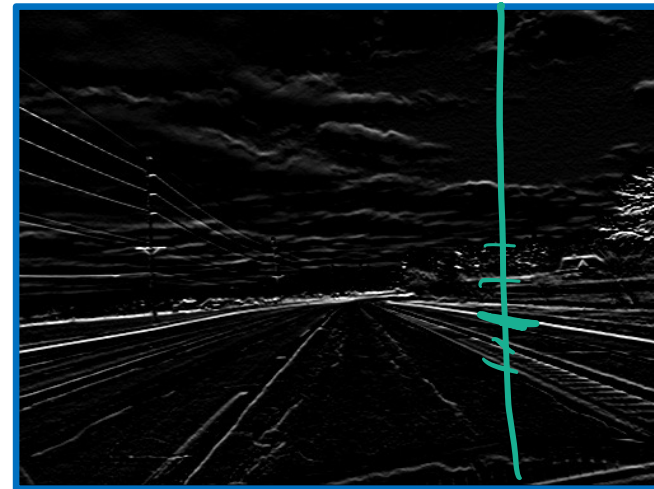
Partial derivatives of an image



$x \rightarrow$



$\partial f / \partial x$



$\partial f / \partial y$



Other approximations for finite difference filters

Prewitt: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

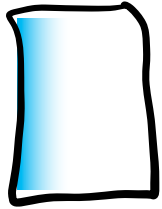
Sobel: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$



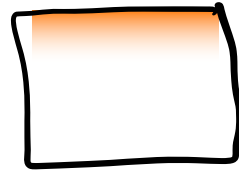
Image Gradient

$$\nabla f = \left[\frac{\partial f}{\partial x}, 0 \right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$

$$\nabla f = \left[0, \frac{\partial f}{\partial y} \right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$



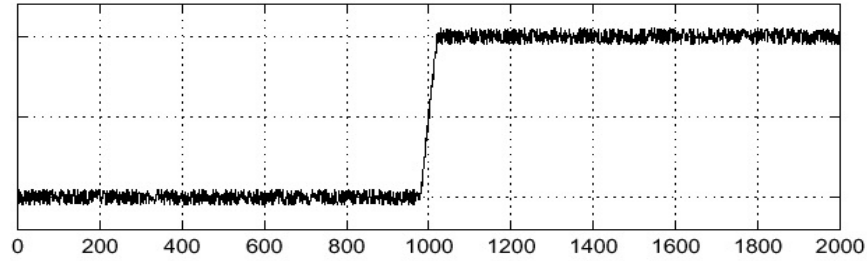
Gradient Direction: $\theta = \tan^{-1} \left(\frac{\partial f / \partial y}{\partial f / \partial x} \right)$

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

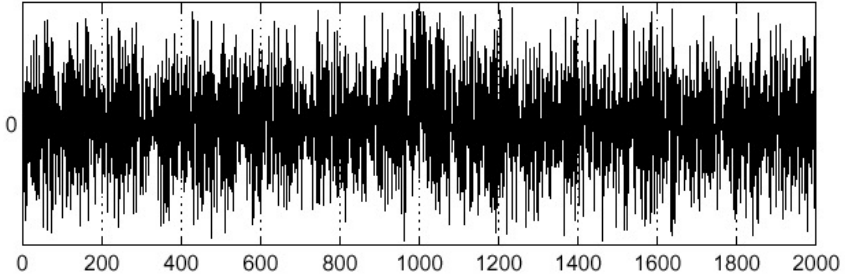


Impact of Noise

f

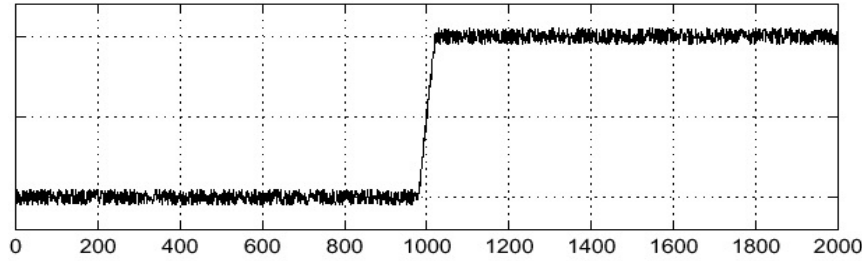


$\frac{d}{dx} f(x)$

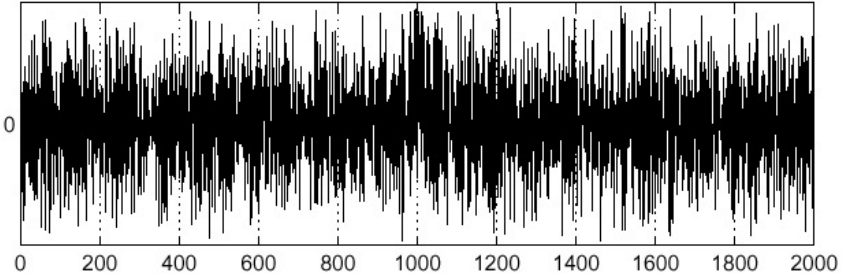


Impact of Noise

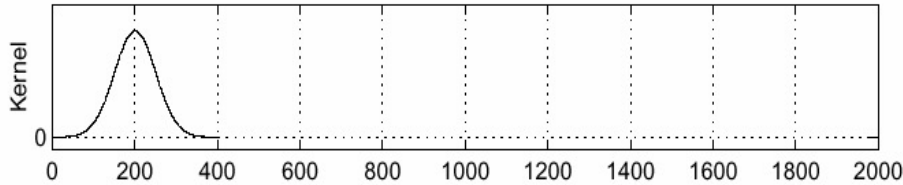
f



$\frac{d}{dx} f(x)$

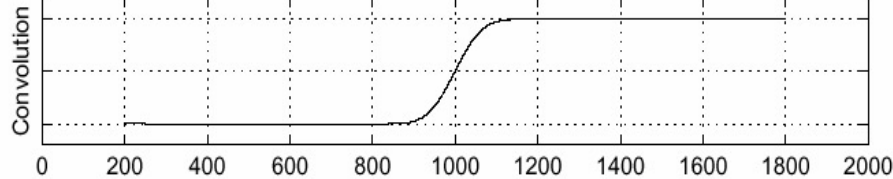


g

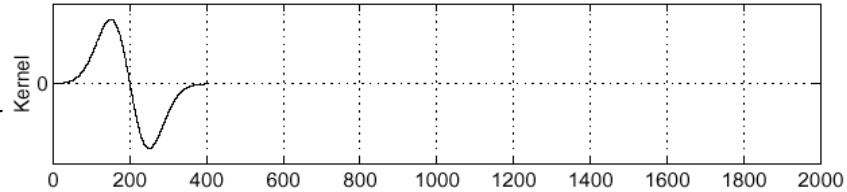


$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

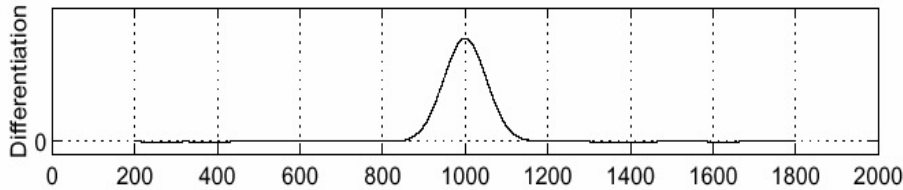
$f * g$



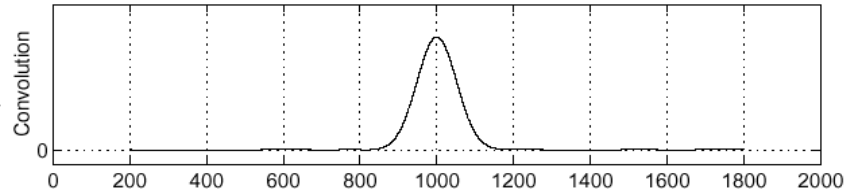
$\frac{d}{dx} g$



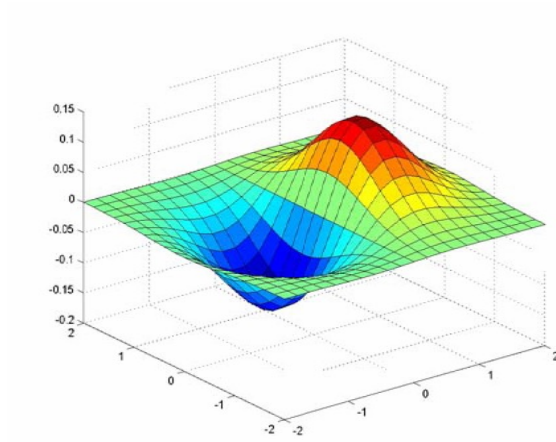
$\frac{d}{dx}(f * g)$



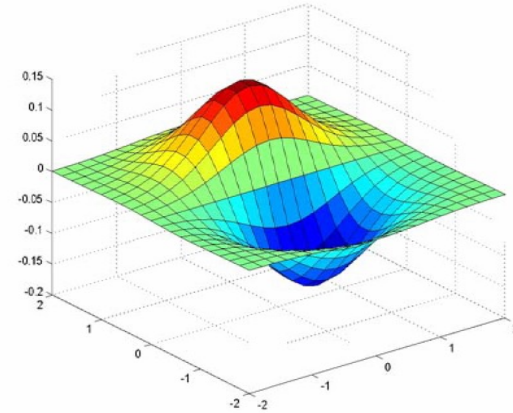
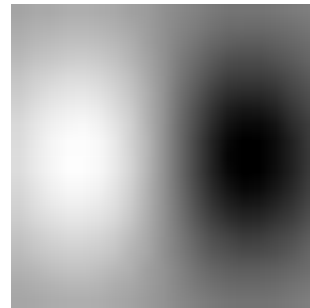
$f * \frac{d}{dx} g$



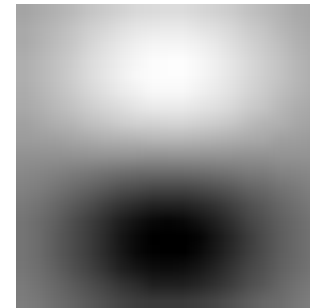
Derivative of Gaussian filters



x-direction



y-direction

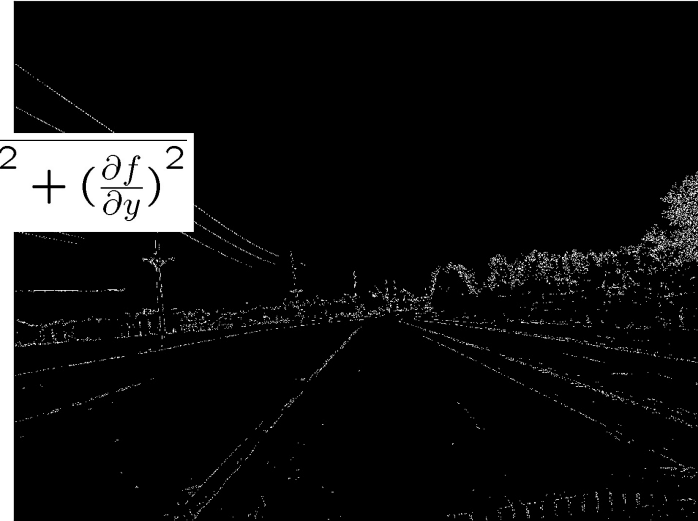


Building an edge detector

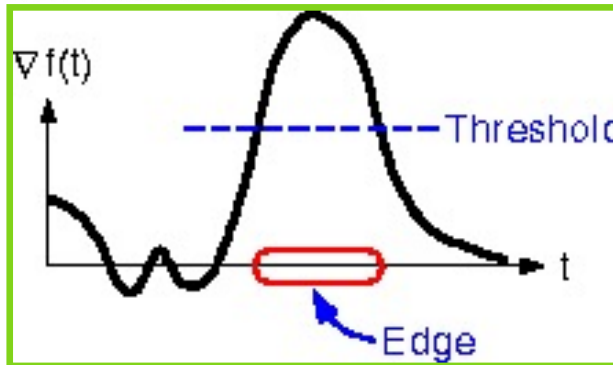
Original Image



Edge Image

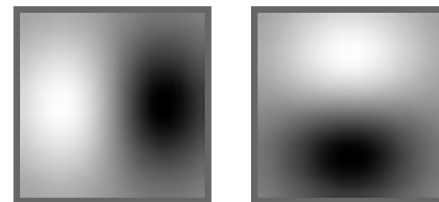
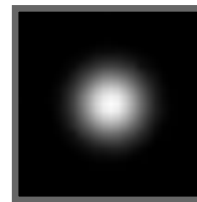


$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$



Review: smoothing vs. derivative filters

- Smoothing filters
 - Gaussian: remove “high-frequency” components; “low-pass” filter
 - What should the values sum to?
 - One: constant regions are not affected by the filter
- Derivative filters
 - Derivatives of Gaussian
 - What should the values sum to?
 - Zero: no response in constant regions

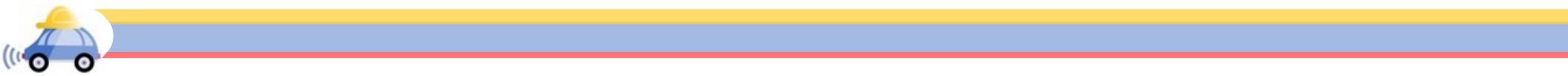


Summary

- Convolution as translation invariant linear operations on signals and images
 - Examples of filters for smoothing and sharpening images
 - Did not discuss: how to pick a filter for different types of noise?
 - Did not discuss: nice properties of Gaussian Filters (like separability)
- Explored edge detection to understand shapes and semantic meaning in an image
 - Use partial derivatives to determine changes in intensity
 - Did not discuss: how does thresholding impact the edges detected?
 - Did not discuss: advanced edge detectors (Canny Edge Detector)
- Next time: the basics of object recognition!



Extra Slides



Separability of the Gaussian filter

$$\begin{aligned} G_{\sigma}(x, y) &= \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right)\right) \left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{y^2}{2\sigma^2}\right)\right) \end{aligned}$$

The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

In this case, the two functions are the (identical) 1D Gaussian



Why is separability useful?

- Separability means that a 2D convolution can be reduced to two 1D convolutions (one along rows and one along columns)
- What is the complexity of filtering an $n \times n$ image with an $m \times m$ kernel?
 - $O(n^2 m^2)$
- What if the kernel is separable?
 - $O(n^2 m)$



Noise in Images

- Salt and pepper noise: contains random occurrences of black and white pixels
- Impulse noise: contains random occurrences of white pixels
- Gaussian noise: variations in intensity drawn from a Gaussian normal distribution



Original



Salt and pepper noise



Impulse noise



Gaussian noise

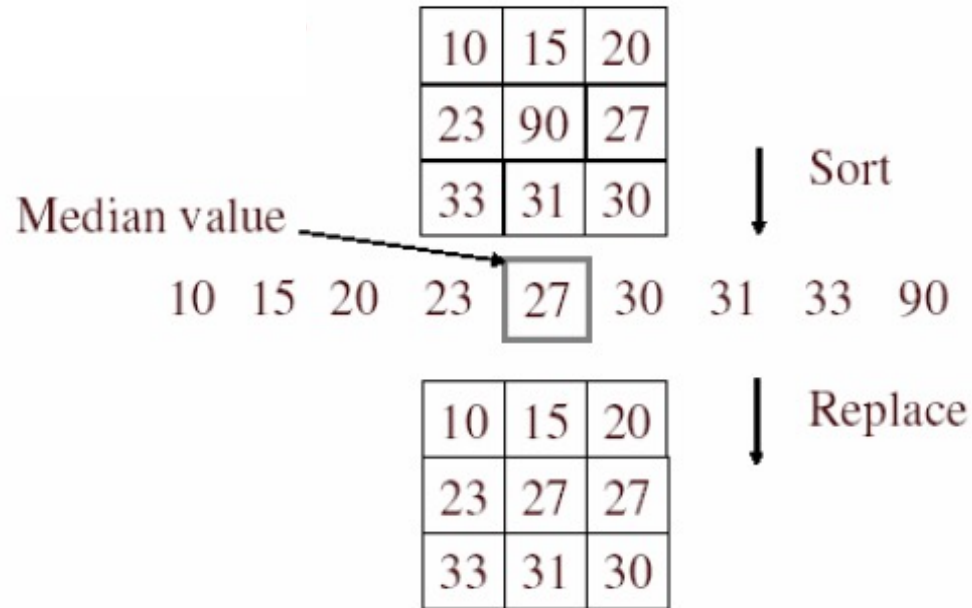


Reducing salt-and-pepper noise



Alternative idea: Median filtering

- A median filter operates over a window by selecting the median intensity in the window



Median filter

- Is median filtering linear?
- Let's try filtering

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 2 & 2 & 2 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \rightarrow 1$$

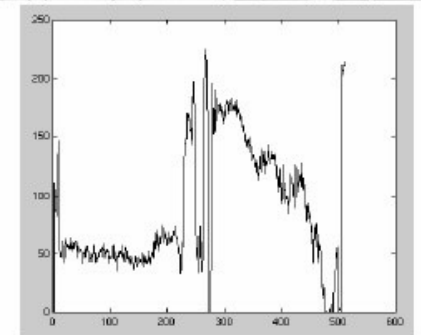
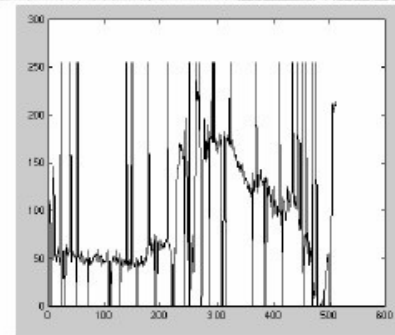
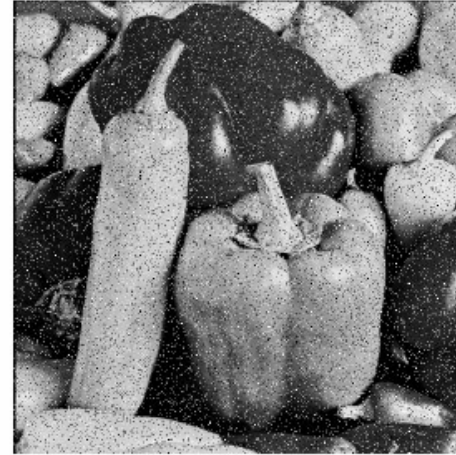
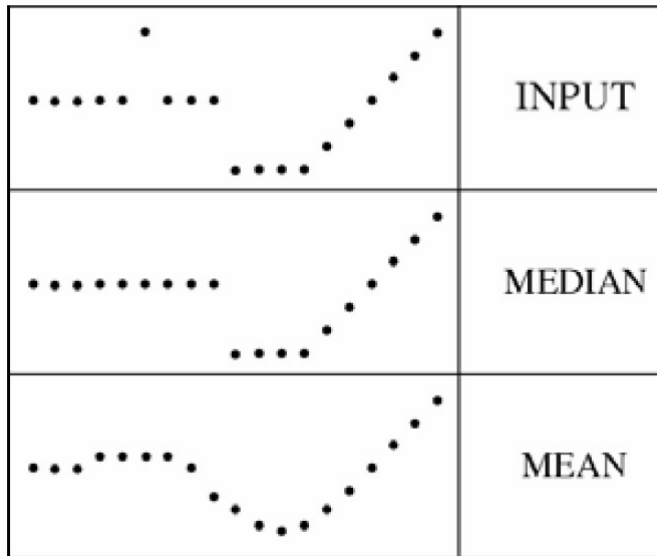
$$\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 2 & 2 & 2 \end{array} \xrightarrow{f} 2$$



Median filter

- What advantage does median filtering have over Gaussian filtering?
 - Robustness to outliers

filters have width 5 :



Gaussian vs. median filtering

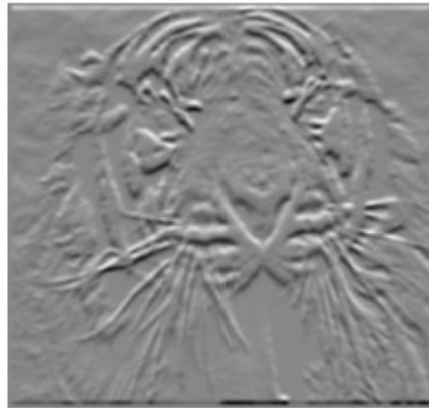
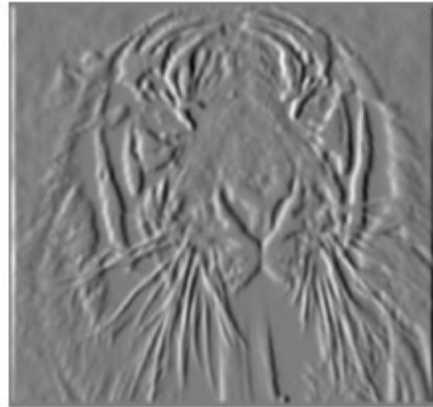


Partial derivatives of an image

$$\frac{\partial f(x, y)}{\partial x}$$



$$\frac{\partial f(x, y)}{\partial y}$$



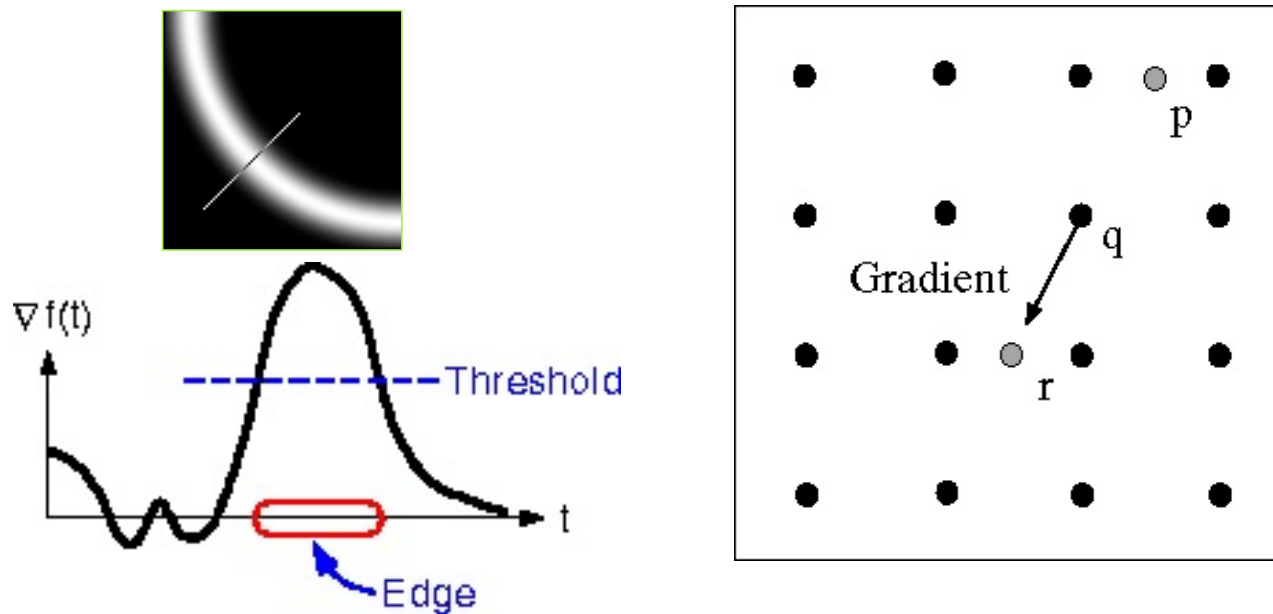
-1	1
----	---

-1	or	1
1		-1

Which shows changes with respect to x?



Non-maximum suppression



- For each location q above threshold, check that the gradient magnitude is higher than at neighbors p and r along the direction of the gradient
 - May need to interpolate to get the magnitudes at p and r

