Introduction to Computer Vision

Lecture 2 - Classic Vision I

Prof. He Wang

Recap: Overview of Computer Vision

- Compared to human vision, computer vision deals with the following tasks:
  - visual data acquisition (similar to human eyes but comes with many more choices)
  - signal processing and feature extraction (mostly low-level)
  - analyze local structures and then 3D reconstruct the original scene (mid-level)
  - understanding (mostly high-level)
  - generation
  - vision-language tasks
  - and further enabling embodied agents to take actions.

The Early History of Computer Vision

The Birth of Artificial Intelligence

Alan Turing and Turing test

1950, Turing wrote the article ***"Computing machinery and intelligence"***, in which he described what would become known as the **"Turing Test"**.

The Dartmouth Conference

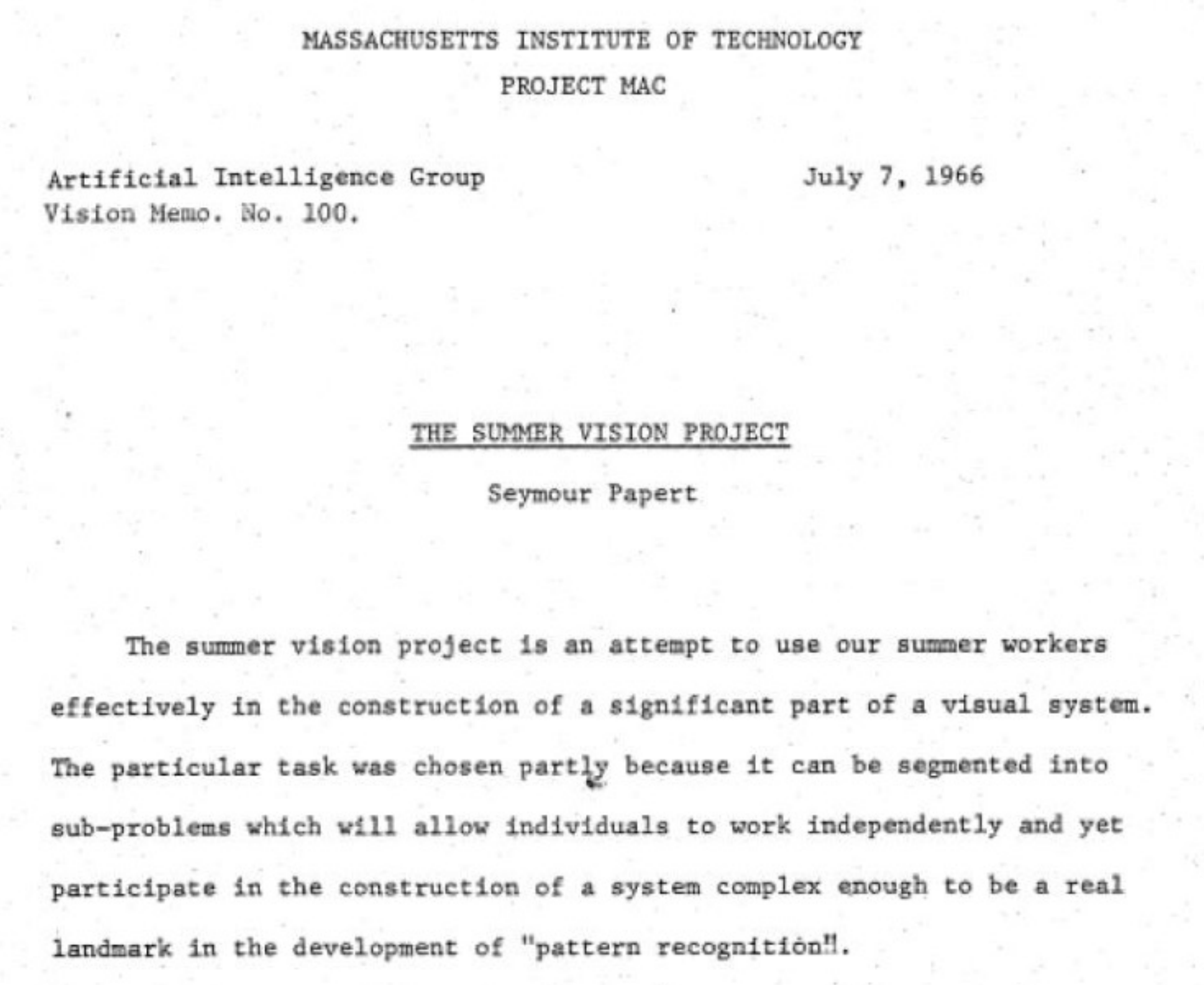
August 1956. *From left to right: Oliver Selfridge, Nathaniel Rochester, Ray Solomonoff, Marvin Minsky, Trenchard More, John McCarthy, Claude Shannon.*

Early in 1960s: CV as a Summer Project

- A visual perception component of an ambitious agenda to mimic human intelligence.

- AI pioneers believed that solving the “visual input” problem would be easier than solving higher-level reasoning and planning.

- Marvin Minsky at MIT asked his undergrad Gerald Jay Sussman to “spend the summer linking a camera to a computer and getting the computer to describe what it saw”. *However, we know this is not that easy.*



MASSACHUSETTS INSTITUTE OF TECHNOLOGY
PROJECT MAC

Artificial Intelligence Group July 7, 1966
Vision Memo. No. 100.

THE SUMMER VISION PROJECT

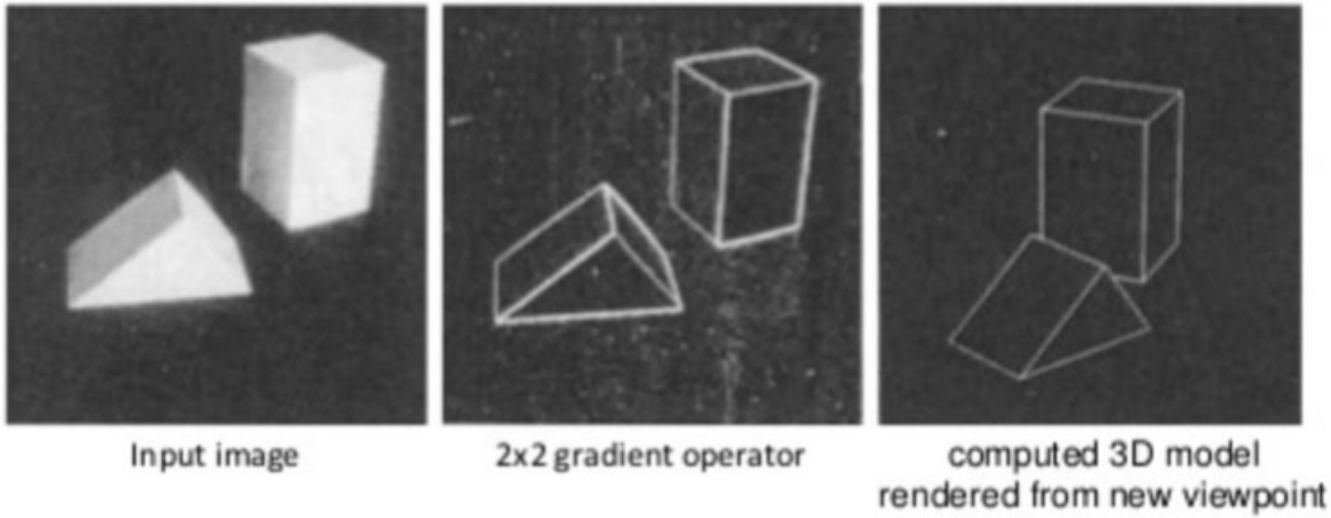
Seymour Papert

The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system. The particular task was chosen partly because it can be segmented into sub-problems which will allow individuals to work independently and yet participate in the construction of a system complex enough to be a real landmark in the development of "pattern recognition".

Early in 1960s: Interpretation of Synthetic Objects

Larry Roerts
1963, 1st thesis of Computer Vision

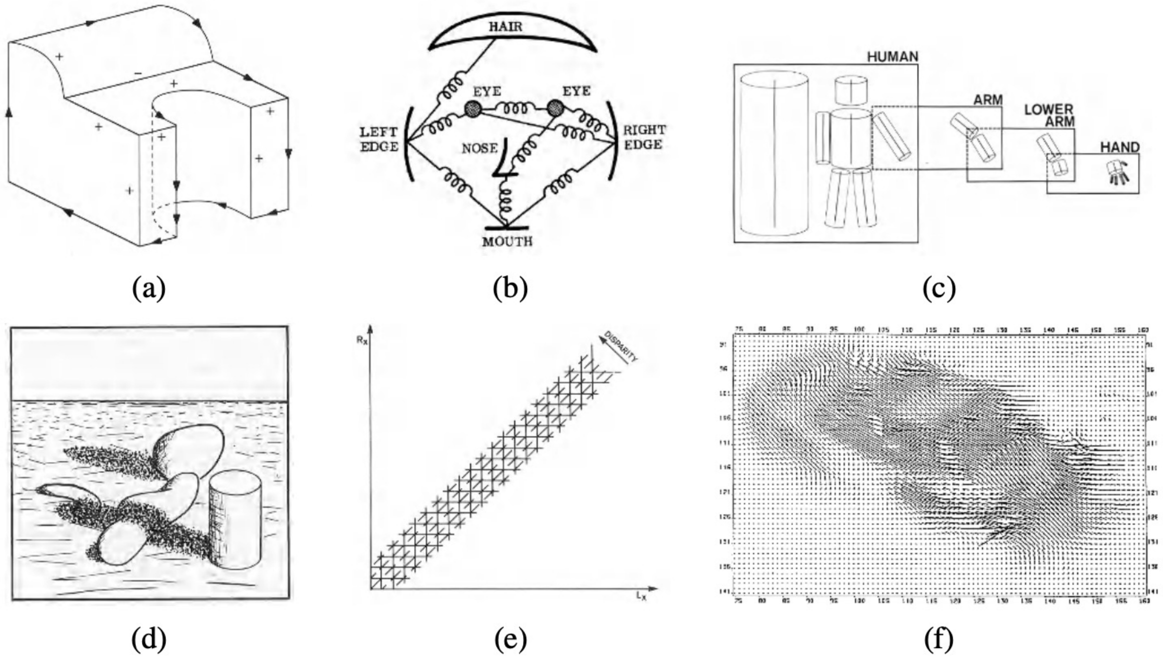
Ph.D. thesis "Machine Perception of Three-Dimensional Solids"



Borrowed from Stanford CS231N Lecture 01.

1970s/1980s: reconstruction as the first step

- What distinguished computer vision from the already existing field of digital image processing:
  - the desire to recover the three-dimensional structure of the world from images
  - And use this as a stepping stone to- wards full scene understanding



(a) (b) (c)

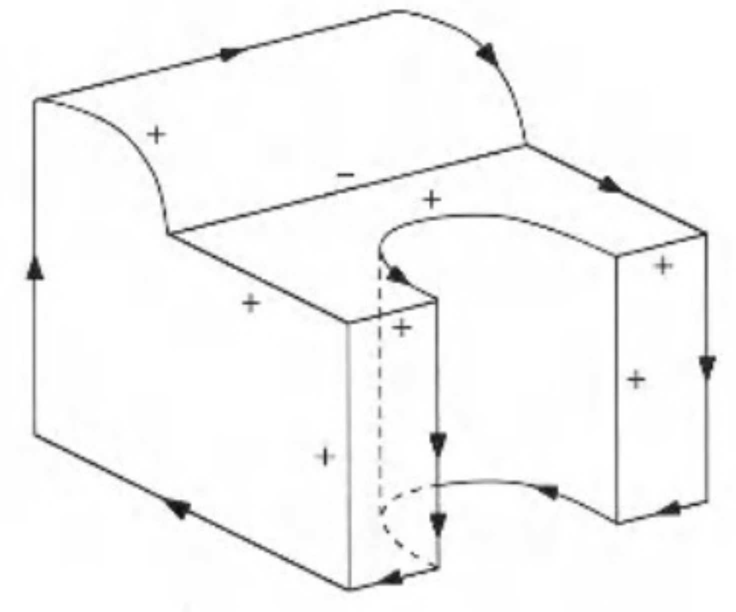
(d) (e) (f)

Figure 1.7 Some early (1970s) examples of computer vision algorithms: (a) line labeling (Nalwa 1993) © 1993 Addison-Wesley, (b) pictorial structures (Fischler and Elschlager 1973) © 1973 IEEE, (c) articulated body model (Marr 1982) © 1982 David Marr, (d) intrinsic images (Barrow and Tenenbaum 1981) © 1973 IEEE, (e) stereo correspondence (Marr 1982) © 1982 David Marr, (f) optical flow (Nagel and Enkelmann 1986) © 1986 IEEE.

Szeliski, Richard. Computer vision: algorithms and applications. Springer Science & Business Media, 2010.

Basic Ideas

- Extracting edges and then inferring the 3D structure of an object or a “blocks world” from the topological structure of the 2D lines



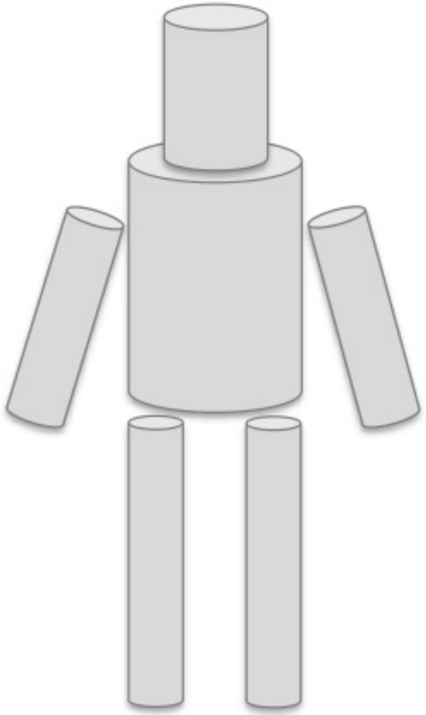
Line labeling (Nalwa 1993)

Szeliski, Richard. Computer vision: algorithms and applications. Springer Science & Business Media, 2010.

Three-dimensional Modeling of Non-polyhedral Objects

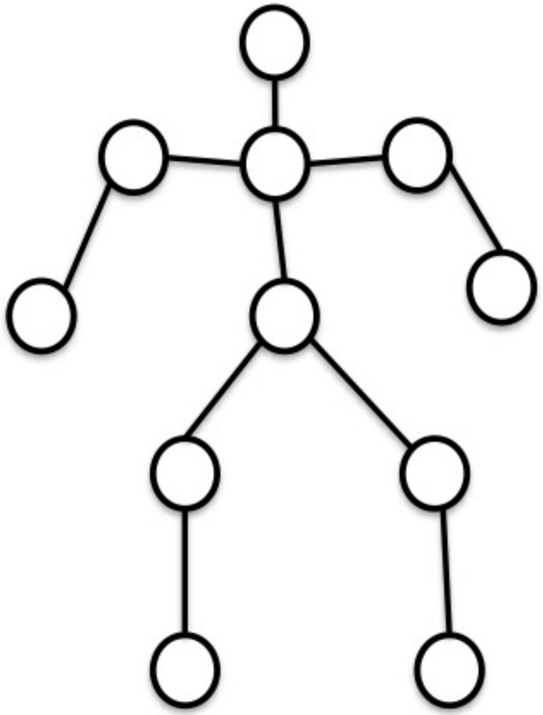
- Generalized Cylinder

Brooks & Binford, 1979



- Pictorial Structure

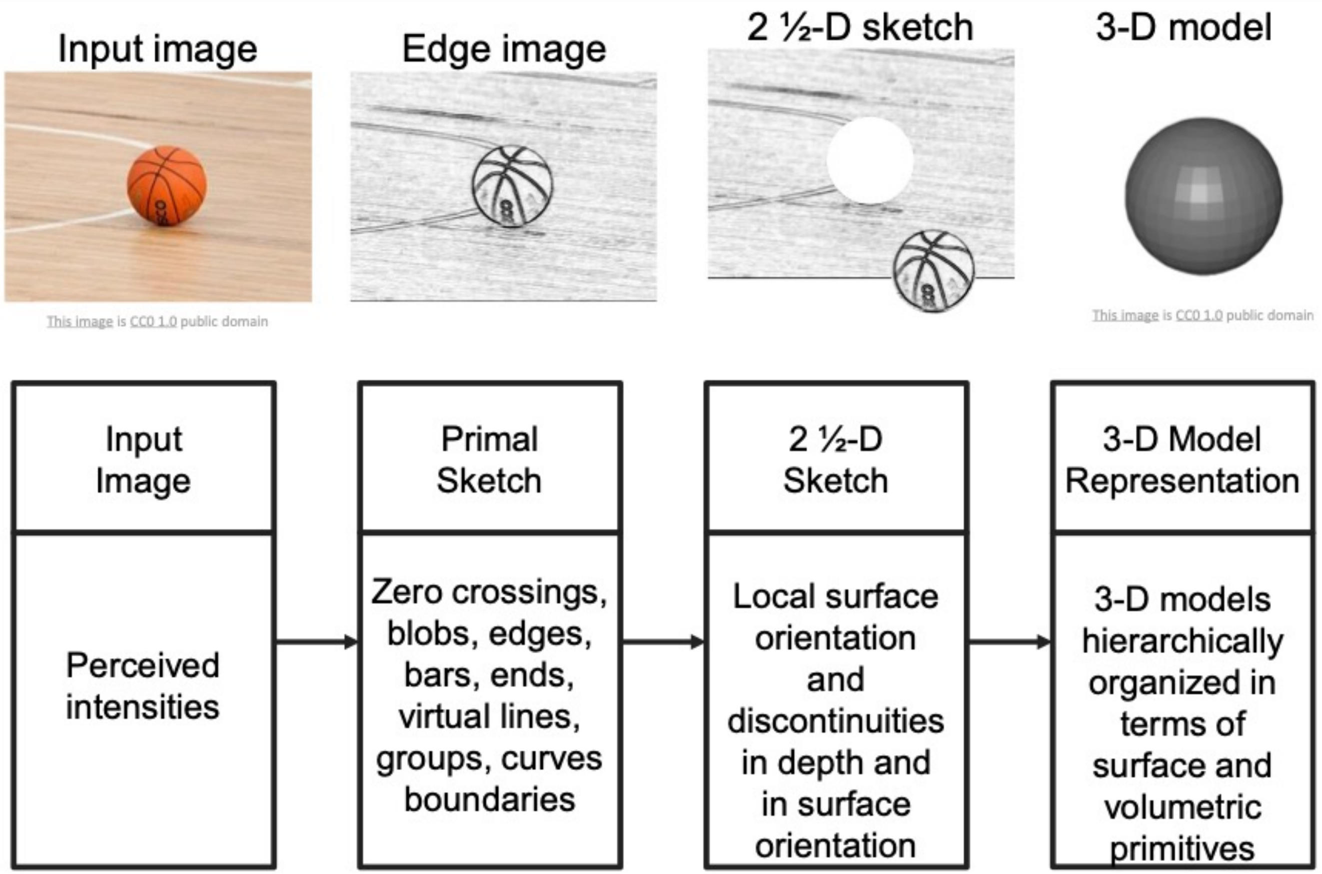
Fischler and Elschlager, 1973



Borrowed from Stanford CS231N Lecture 01.

David Marr’s 2.5-D Sketch

- 2.5-D Sketch:
  - A surface based representation that bridges 2D and 3D
  - Depth-from-X: computed from a 2-D image-based representation (primal sketch) via extracting information about
    - surface orientation
    - depth from a variety of sources, such as shading, stereo, and motion.



This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

Input Image

Perceived intensities

Primal Sketch

Zero crossings, blobs, edges, bars, ends, virtual lines, groups, curves boundaries

2 ½-D Sketch

Local surface orientation and discontinuities in depth and in surface orientation

3-D Model Representation

3-D models hierarchically organized in terms of surface and volumetric primitives

Stages of Visual Representation, David Marr, 1970s

Palmer, Stephen E. Vision science: Photons to phenomenology. MIT press, 1999.

3D Reconstruction

Structure from Motion
(Tomasi and Kanade 1992)

Dense stereo matching
(Boykov, Veksler, and Zabih 2001)

Multi-view reconstruction
(Seitz and Dyer 1999)

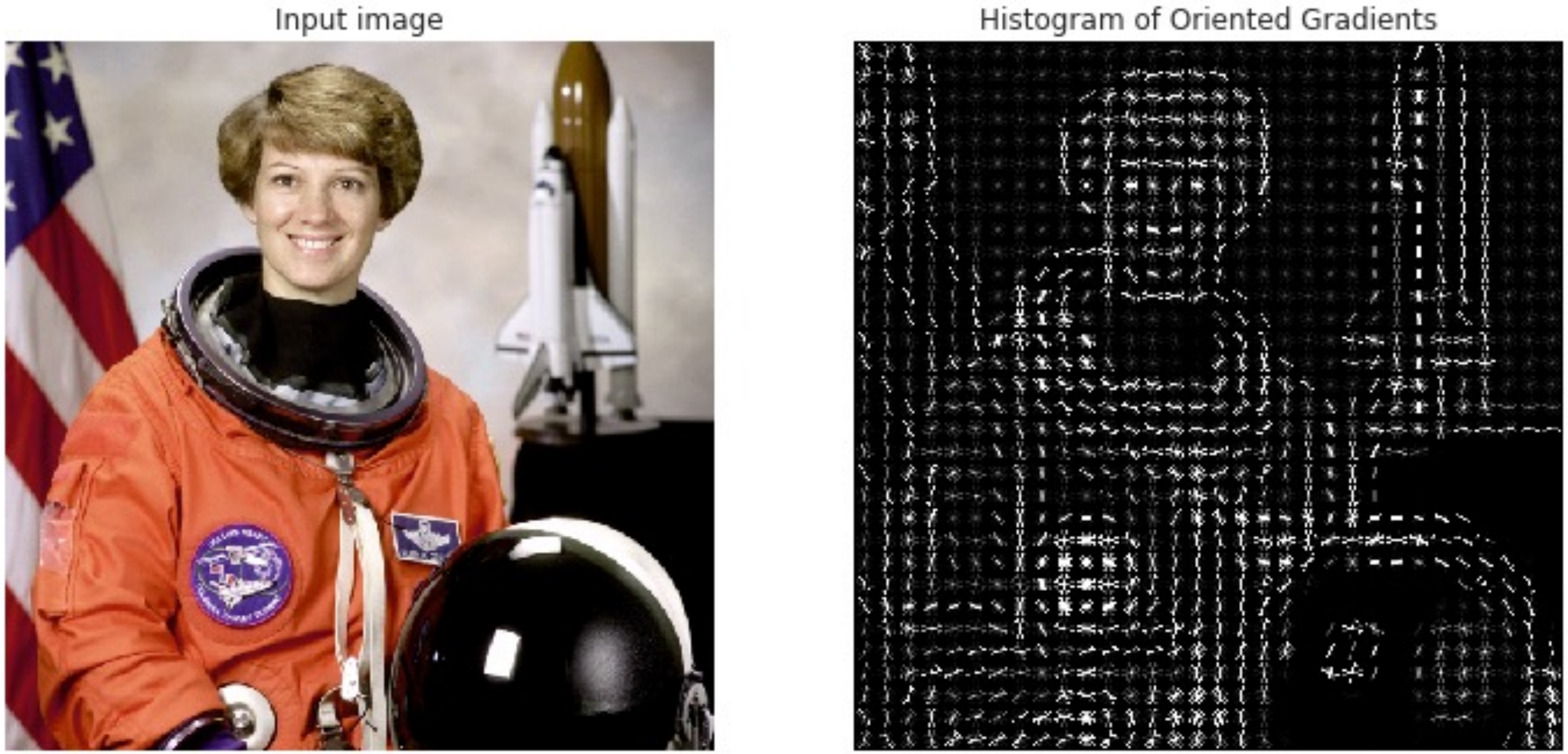
Szeliski, Richard. Computer vision: algorithms and applications. Springer Science & Business Media, 2010.

Recognition and Segmentation

D. Lowe. IJCV, 1992

Normalized Cut (Shi & Malik, 1997)

Descriptors



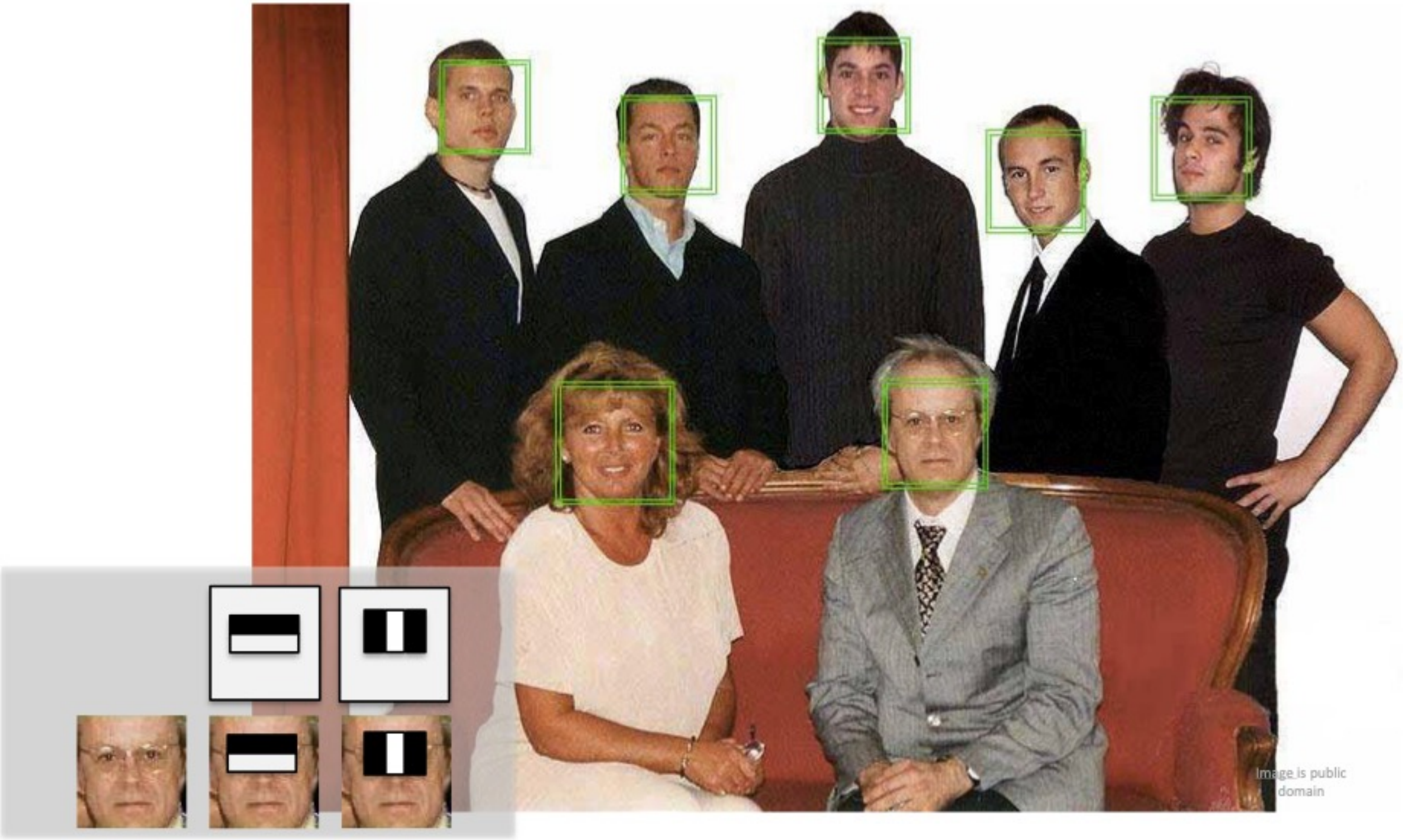
Input image

Histogram of Oriented Gradients

Histogram of Gradients (HoG) Dalal & Triggs, 2005

Credit: https://iq.opengenus.org/object-detection-with-histogram-of-oriented-gradients-hog/

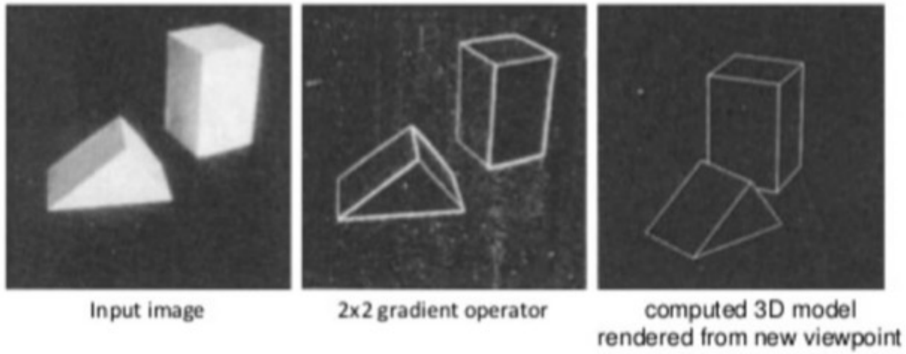
Detection



Face Detection, Viola & Jones, 2001

CV from the Classic Era to the Deep Learning Era

- Previous works don’t leverage learning.
- However, many techniques and concepts proposed by them are still foundations for modern computer vision.
- Current trend:
  - From non-learning based method to **learning-based method**
  - Rely on **big data**
  - Requires more **computation resources**.


Input image
2x2 gradient operator
computed 3D model rendered from new viewpoint

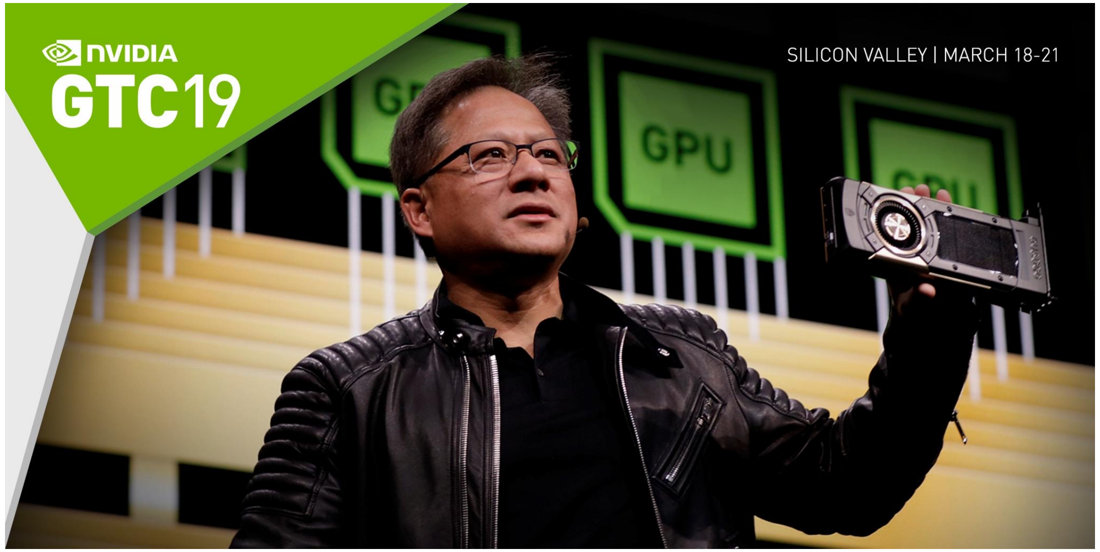
Algorithm: Deep Learning

2018 Turing Awards: Geoffrey Hinton, Yann LeCun, and Yoshua Bengio

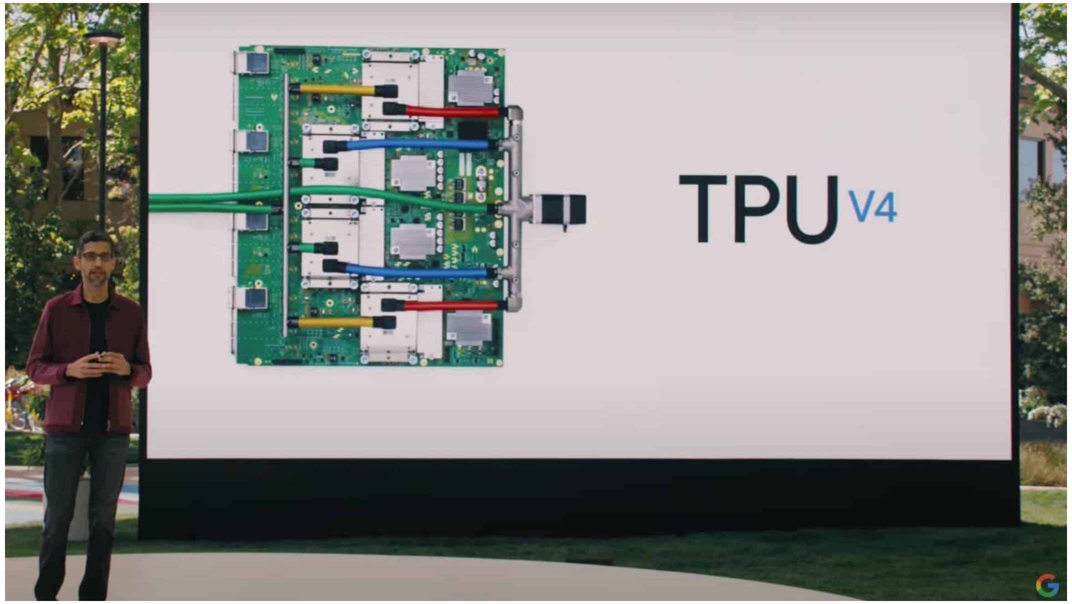
Data: ImageNet and Its Benchmark

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li & L. Fei-Fei. *CVPR*, 2009.

Computational Resources: GPU



NVIDIA and its GPU



Google and its TPU

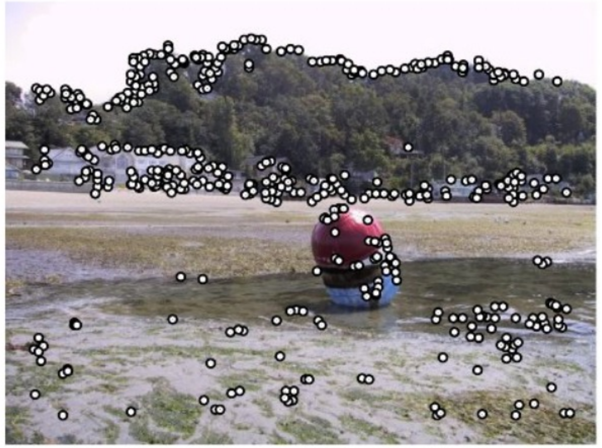
Today's Topic

- **Low-level vision**
  - Image processing
  - Edge/corner detection
  - Feature extraction
- Mid-level vision
  - Grouping
  - Inferring scene geometry (3D reconstruction)
  - Inferring camera and object motion
- High-level vision
  - Object recognition
  - Scene understanding
  - Activity understanding

Outline of Today's Lecture

- Images as functions

- Classic (non-learning) methods
    - Edge detectors
    - Corner detectors
    - Line fitting



Adaptive non-maximal suppression (ANMS) (Brown, Szeliski, and Winder 2005)

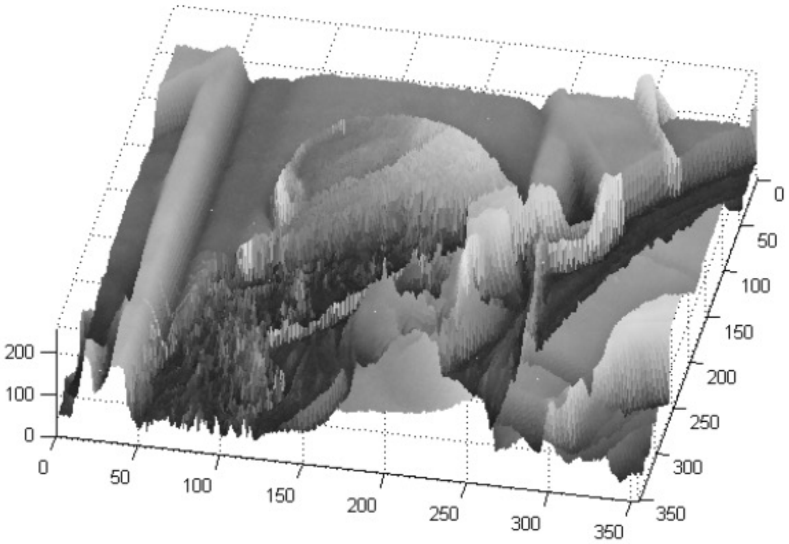
https://medium.com/@realderektan/self-driving-car-project-part-1-lane-lines-detector-6d960e2b023

Images as Functions

Images as Functions

- **An Image** as a function f from R^2 to R^M:
  - $f(x, y)$ gives the **intensity** at position (x, y)
  - Defined over a rectangle, with a finite range:

$$f\colon \underbrace{[a,b] \times [c,d]}_{\text{Domain support}} \rightarrow \underbrace{[0,255]}_{\text{range}}$$



Slide borrowed from Stanford CS131

Images as Functions

- **An Image** as a function f from R^2 to R^M:
  - $f(x, y)$ gives the **intensity** at position (x, y)
  - Defined over a rectangle, with a finite range:

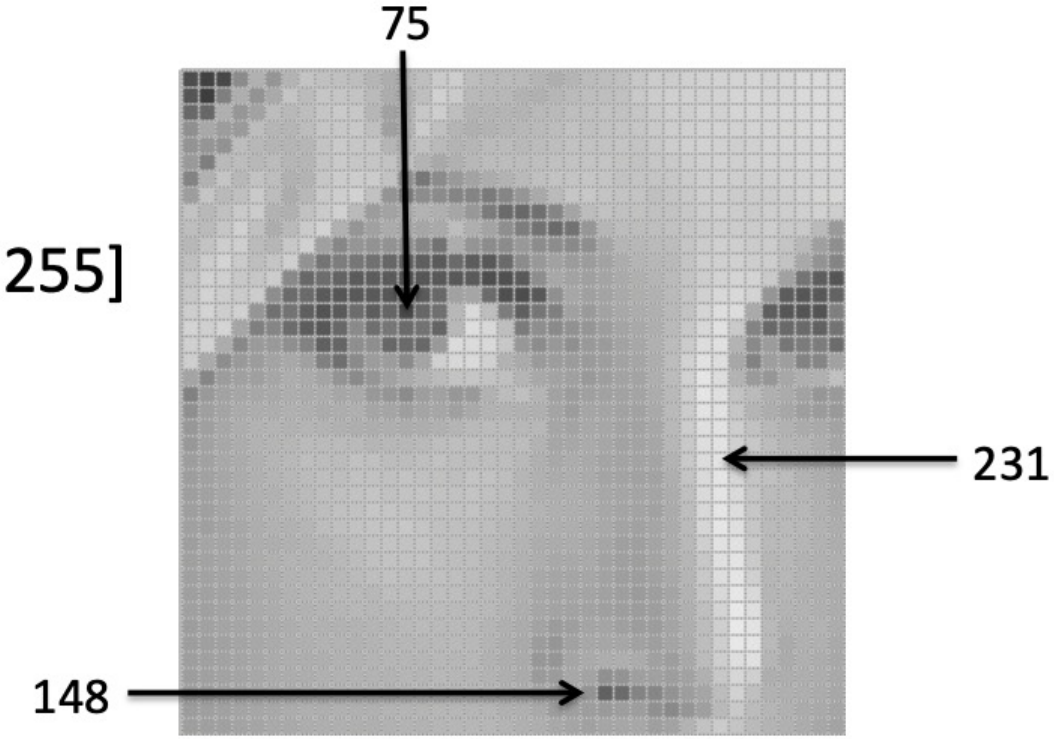
$$f: \underbrace{[a,b] \text{ x } [c,d\,]}_{\text{Domain support}} \rightarrow \underbrace{[0,255]}_{\text{range}}$$

- A color image: $f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$

Slide borrowed from Stanford CS131

Images as Functions

- An image contains discrete number of pixels
  - A simple example
  - Pixel value:
    - "grayscale"
    (or "intensity"): [0,255]
    - "color"
      - RGB: [R, G, B]

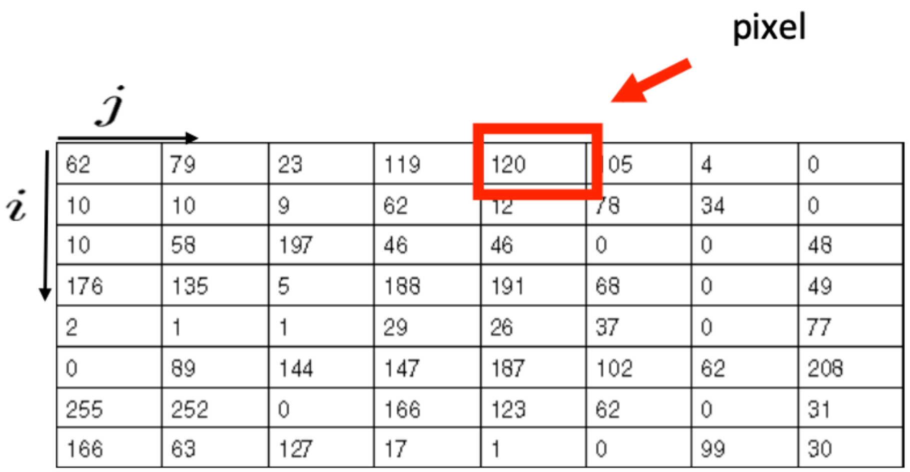


Slide borrowed from Stanford CS131

Images as Functions

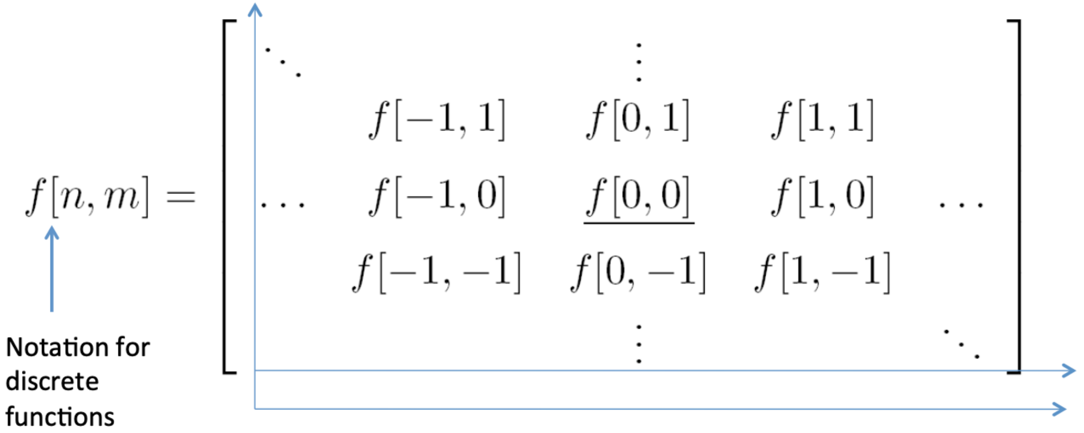
- Images are usually **digital (discrete)**:
  - **Sample** the 2D space on a regular grid

- Represented as a matrix of integer values



62	79	23	119	120	05	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

Cartesian coordinates



$$f[n,m] = \begin{bmatrix} \ddots & & \vdots & & \\ & f[-1,1] & f[0,1] & f[1,1] & \\ \dots & f[-1,0] & \underline{f[0,0]} & f[1,0] & \dots \\ & f[-1,-1] & f[0,-1] & f[1,-1] & \\ & & \vdots & & \ddots \end{bmatrix}$$

Notation for discrete functions

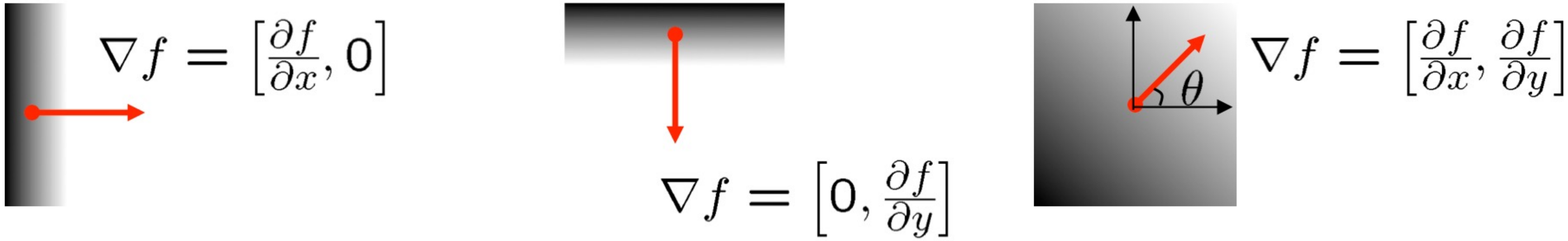
Slide borrowed from Stanford CS131

Image Gradient

- Image as a function:
- Image gradient:

$$f = f(x, y)$$

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



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

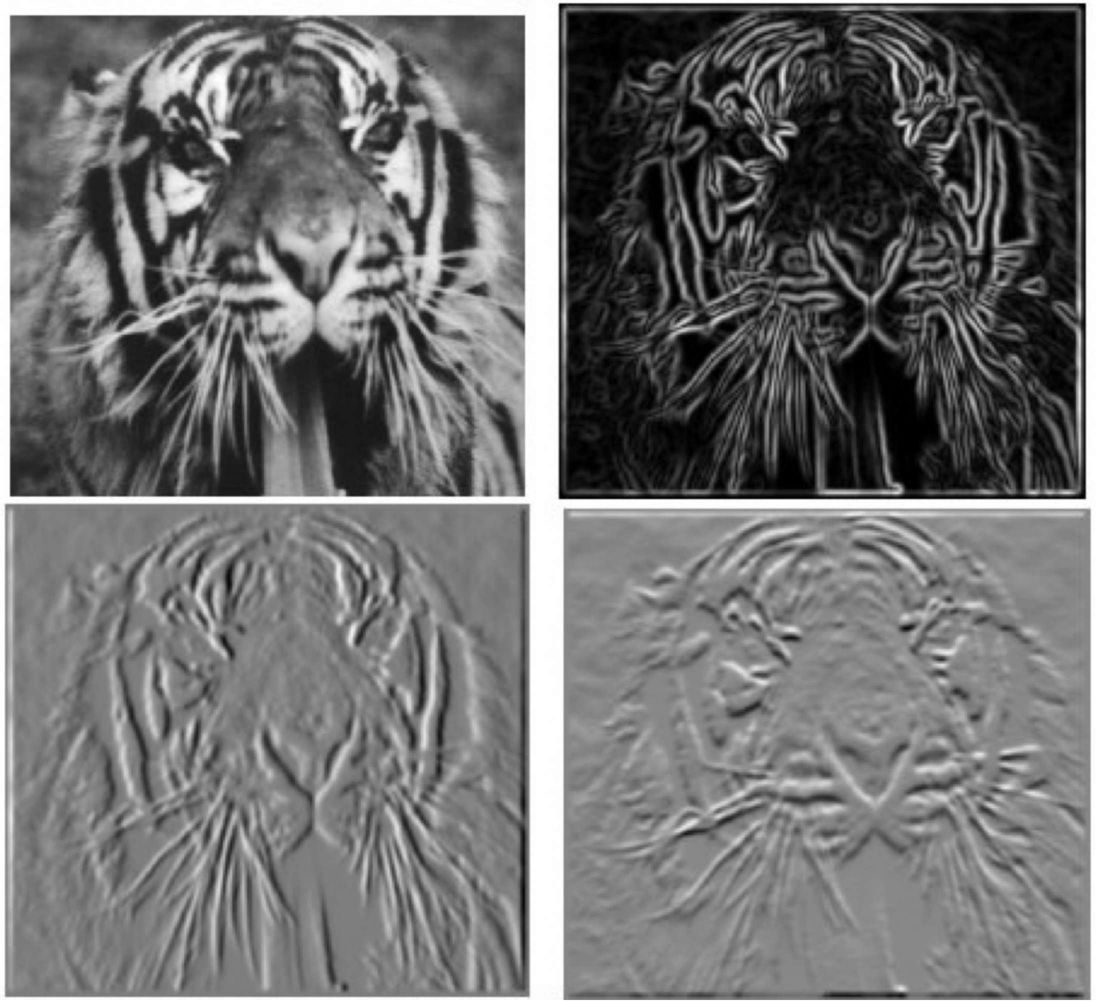
$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$

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

- In practice, use finite difference to replace gradient.
  - $\frac{\partial f}{\partial x}|_{x=x_0} \approx \frac{f(x_0+1, y_0) - f(x_0-1, y_0)}{2}$
- The image gradient points in the direction of the most rapid change in intensity.

Image borrowed from Stanford CS131

Visualizing Image Gradient



Original Image

Gradient magnitude

x-direction

y-direction

Gradient magnitude: $||\nabla f|| = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2}$

Source: Feifei Li

Filters

- **Filering:**
  - – Form a new image whose pixels are a combination original pixel values

Goals:

-Extract useful information from the images

- Features (edges, corners, blobs…)

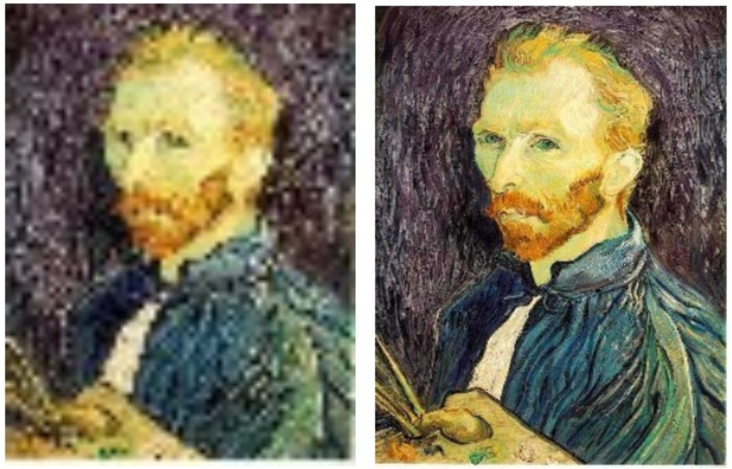
- Modify or enhance image properties:

- super-resolution; in-painting; de-noising

De-noising

Salt and pepper noise

Super-resolution



Slide borrowed from Stanford CS131

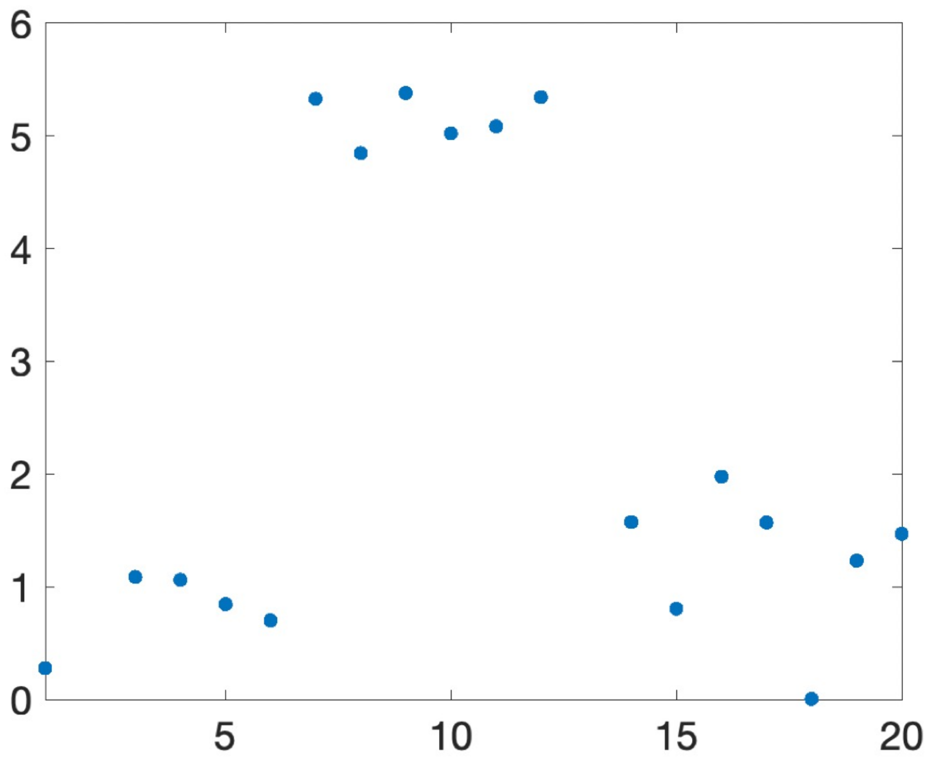
1D Discrete-Space Systems (Filters)

$$f[n] \rightarrow \boxed{\text{System } \mathcal{G}} \rightarrow h[n]$$

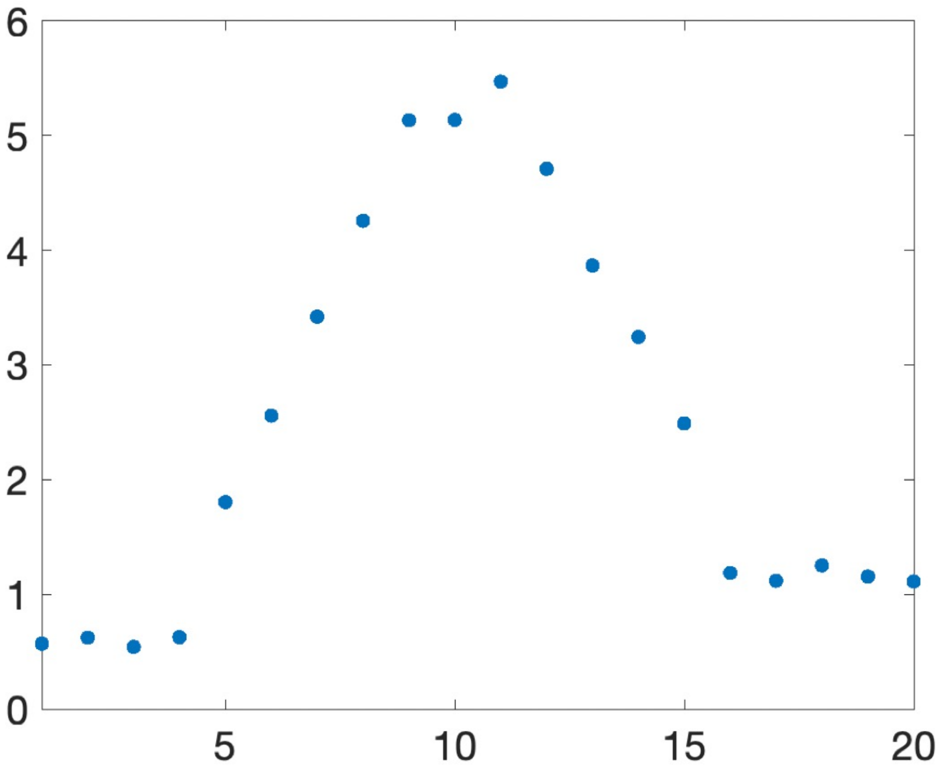
$$h = \mathcal{G}(f), h[n] = \mathcal{G}(f)[n]$$

1D Filter Example: Moving Average

Original data $f[n]$

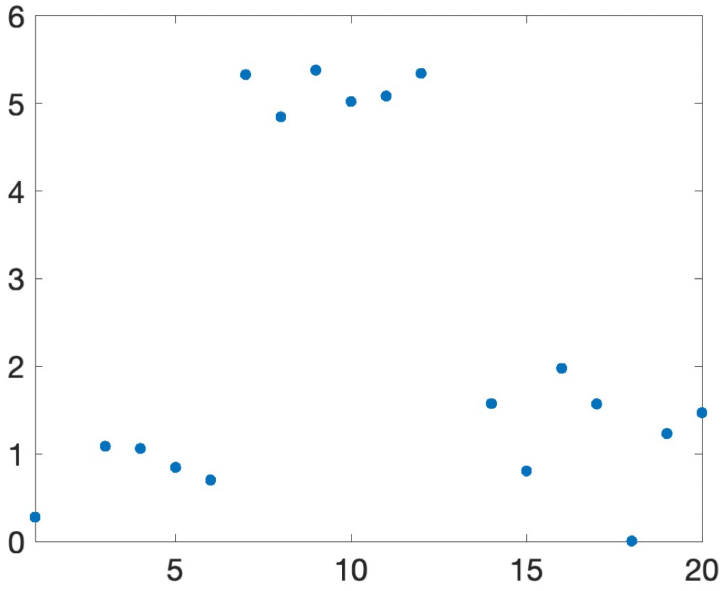


After moving average with window size = 5, $h[n]$

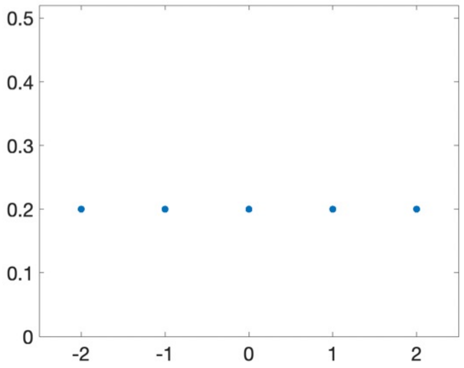


1D Filter Example: Moving Average

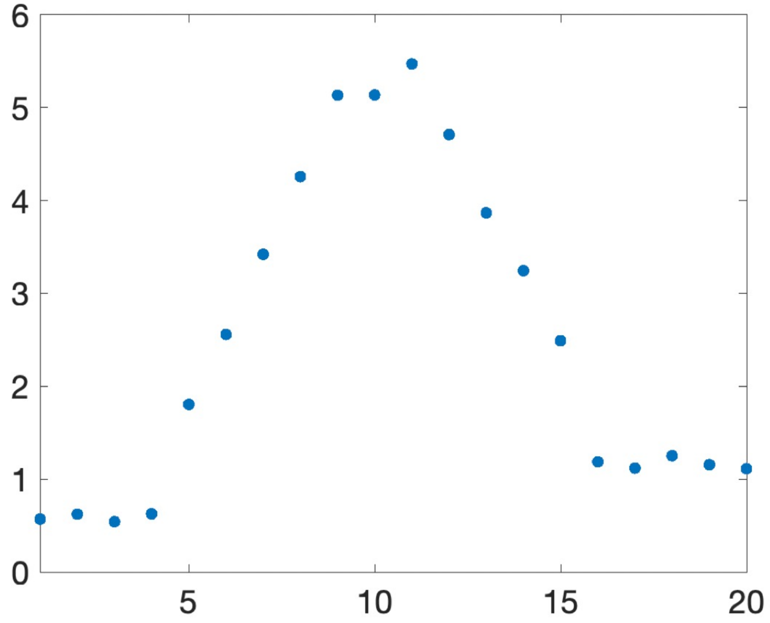
Original data $f[n]$



Weight function (equal weight)



After moving average, $h[n]$

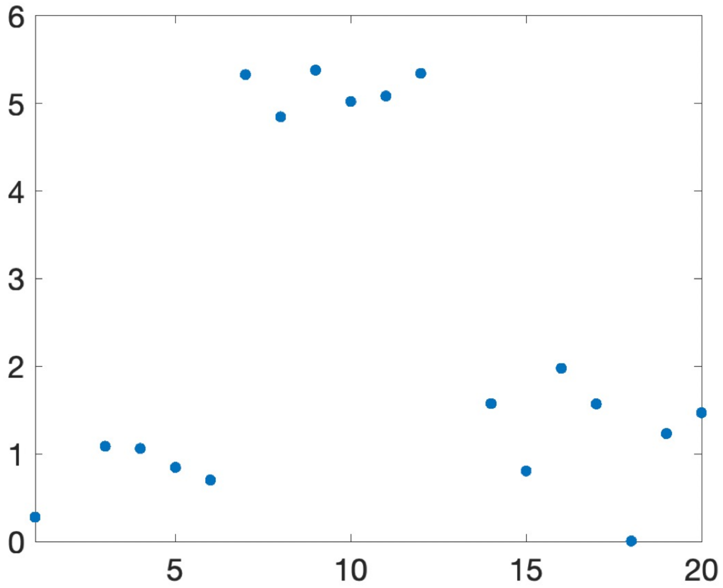


Let’s use the language of image or signal processing!

1D Discrete Convolution ($*$)

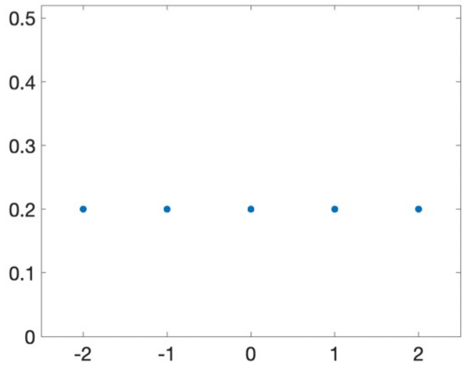
We can express this moving averaging using convolution!

Original data $f[n]$



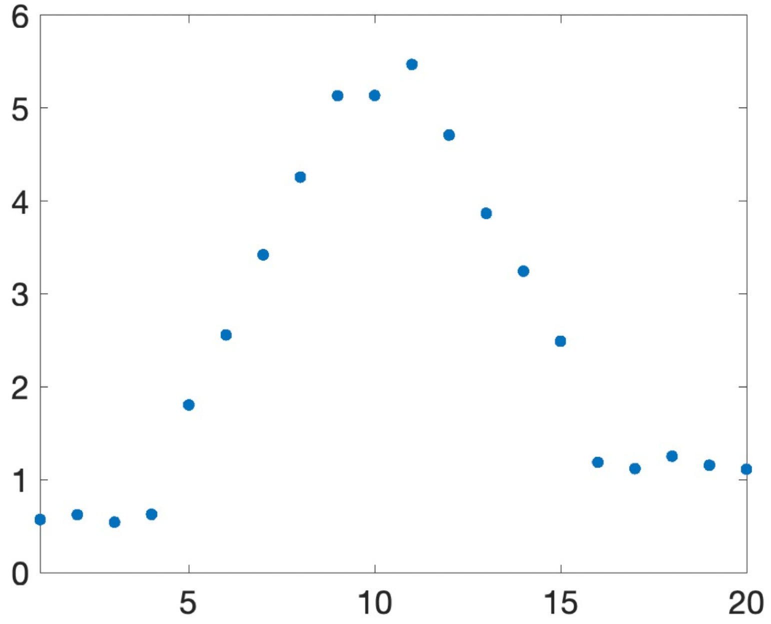
$*$

Filter $g[n]$



$=$

After moving average, $h[n]$



$$h[n] = (f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] g[n-m]$$

Quick Facts of Convolution

	Discrete signal	Continuous signal
Convolution	$(f*g)[n]=\sum_{m=-\infty}^{\infty} f[m]g[n-m]$	$(f*g)(x) = \int_{t=-\infty}^{\infty} f(t)g(x-t)dt$
Fourier Transform	$\mathcal{F}(f)[n] = \sum_{m=0}^{M-1} f[m]\exp(-\frac{i2\pi}{M}mnt)$	$\mathcal{F}(f) = \int_{t=-\infty}^{\infty} f(t)\exp(-i2\pi\omega t)dt$

- Derivative Theorem

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

- Convolution Theorem

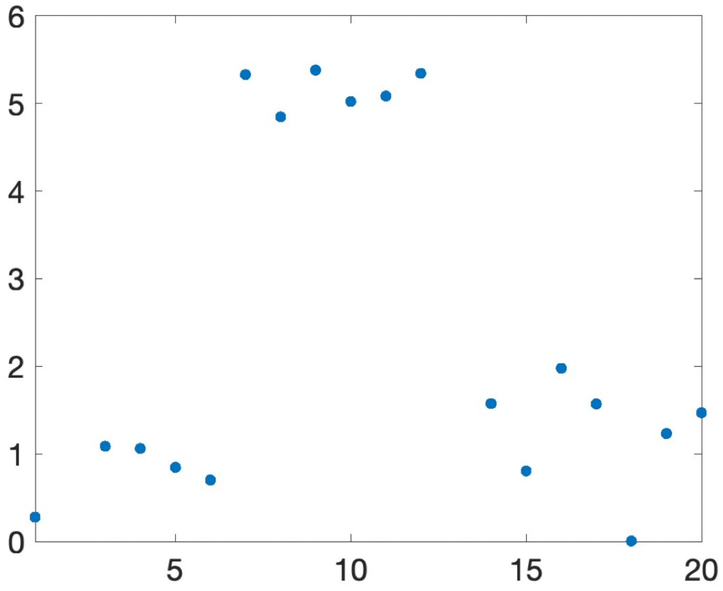
$$\mathcal{F}(f*g) = \mathcal{F}(f)\mathcal{F}(g)$$

$$\therefore h = \mathcal{F}^{-1}(\mathcal{F}(f)\mathcal{F}(g))$$

Discrete Convolution: *

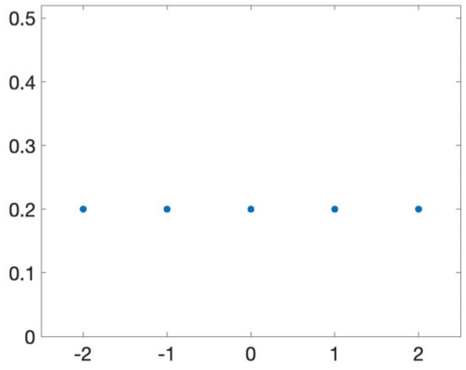
Our filter is indeed a rectangular function. What is its Fourier transform?

Original data $f[n]$



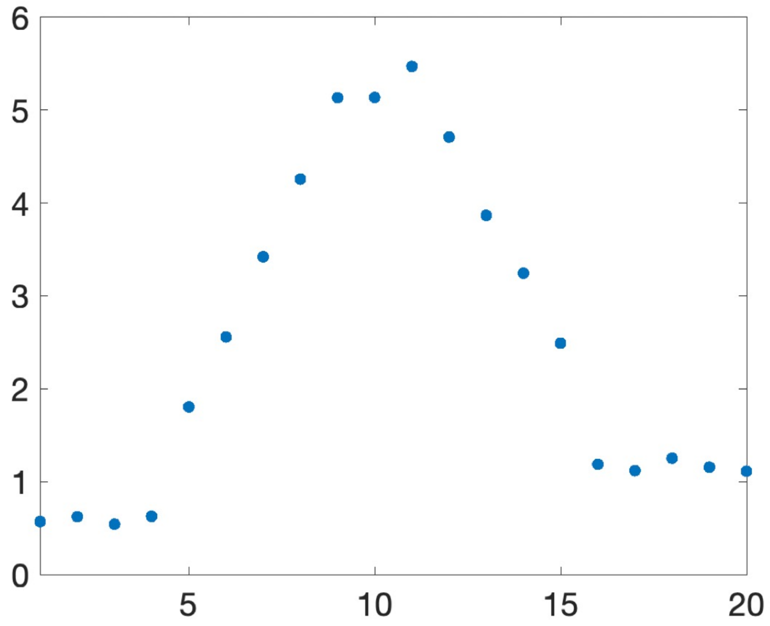
*

Filter $g[n]$



=

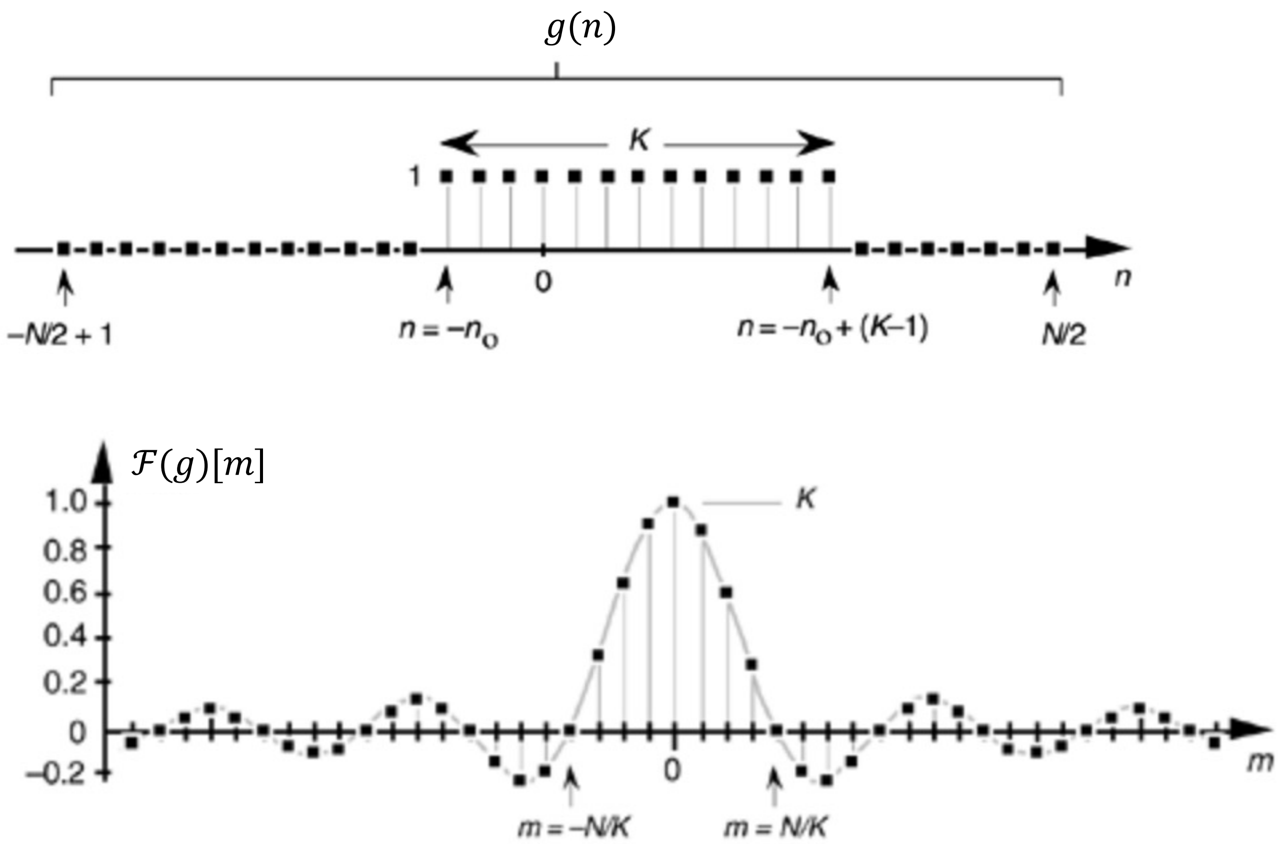
After moving average, $h[n]$



$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] g[n-m]$$

Rectangular Function and its Fourier Transform

Figure 3-24. Rectangular function of width K samples defined over N samples where K < N.

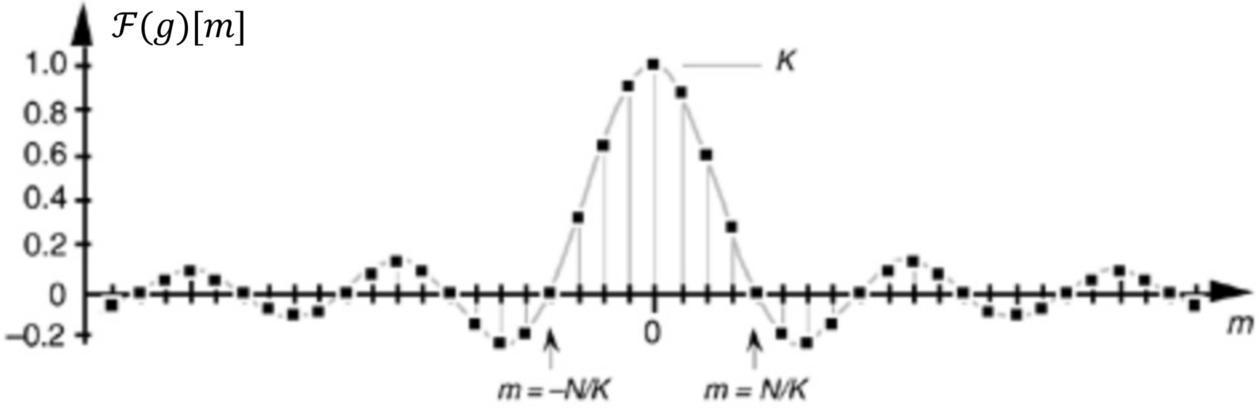


$\mathcal{F}(g)[m]$ mainly concentrates around 0

$\Rightarrow$ g is a low-pass filter.

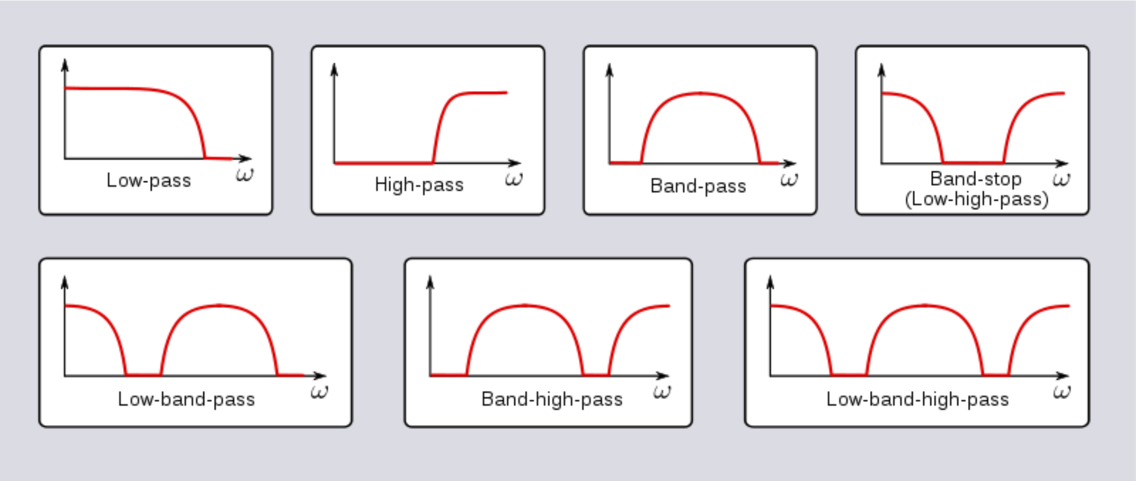
For more information about this discrete Fourier transform, please see https://flylib.com/books/en/2.729.1/the_dft_of_rectangular_functions.html

From a Low-Pass Filter Perspective



$\mathcal{F}(g)[m]$ mainly concentrates around 0

$\Rightarrow$ g is a low-pass filter.



According to Convolution theorem, $\mathcal{F}(f*g)=\mathcal{F}(f)\mathcal{F}(g)$

$\mathcal{F}(f)\mathcal{F}(g)$ means the high frequency part of $\mathcal{F}(f)$ turns to 0 in $\mathcal{F}(f*g)$.

When you removes high frequency parts in the signal, the signal becomes smooth. That’s how Fourier transform explains the smoothing effect by moving average.

https://en.wikipedia.org/wiki/Filter_(signal_processing)

Linear System ↔ Linear Filters ↔ Convolution

$$f[n] \rightarrow \boxed{\text{System } \mathcal{G}} \rightarrow h[n]$$

$$h = \mathcal{G}(f), h[n] = \mathcal{G}(f)[n]$$

- Linear filtering $\mathcal{G}$:
  - $h[n]$ is a linear combination of values from $f[n]$
  - The weight of this linear combination is the same at each point n
- Then $\mathcal{G}$ is a linear system (function) iff $\mathcal{G}$ satisfies

$$\mathcal{G}(\alpha f_1 + \beta f_2) = \alpha\mathcal{G}(f_1) + \beta\mathcal{G}(f_2)$$

- It can be proved that linear filters can also be expressed using convolutions.

2D Discrete-Space Systems (Filters)

$$f[n,m] \rightarrow \boxed{\text{System } \mathcal{G}} \rightarrow h[n,m]$$

$$h = \mathcal{G}(f), h[n,m] = \mathcal{G}(f)[n,m]$$

2D Discrete Filter Example: Moving Average

- 2D DS moving average over a 3 × 3 window of neighborhood

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

$$h[m,n] = \frac{1}{9} \sum_{k=n-1}^{n+1} \sum_{l=m-1}^{m+1} f[k,l]$$

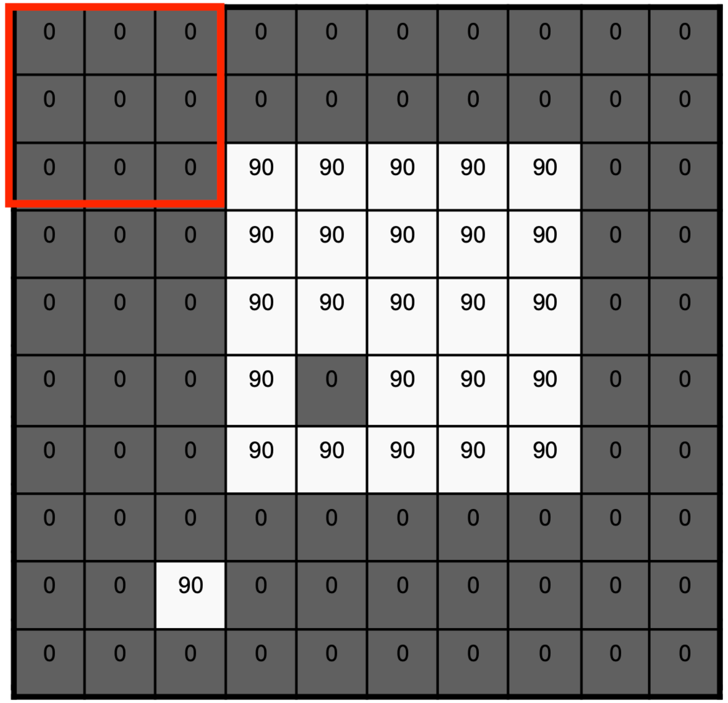
$$= \frac{1}{9} \sum_{k=-1}^{1} \sum_{l=-1}^{1} f[n-k, m-l]$$

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

Image borrowed from Stanford CS131

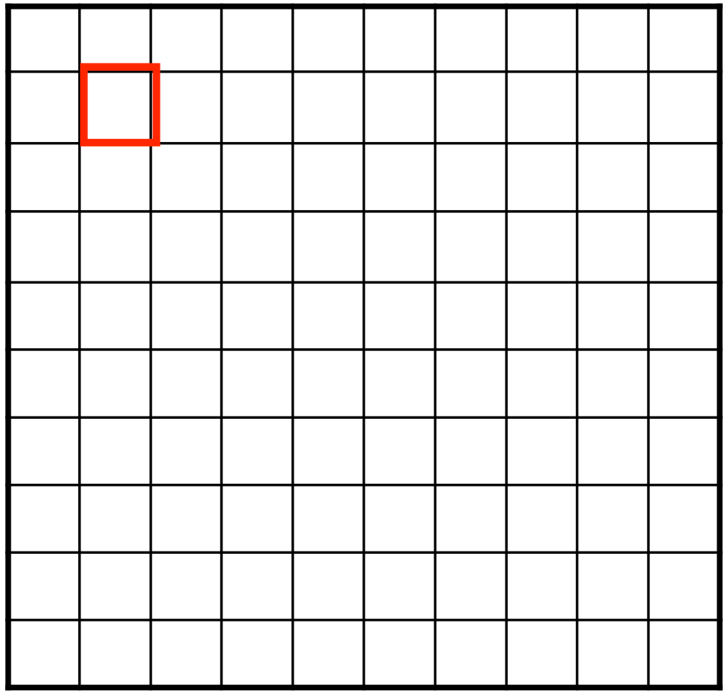
2D Discrete Filter Example: Moving Average

$f[m,n]$



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[m,n]$

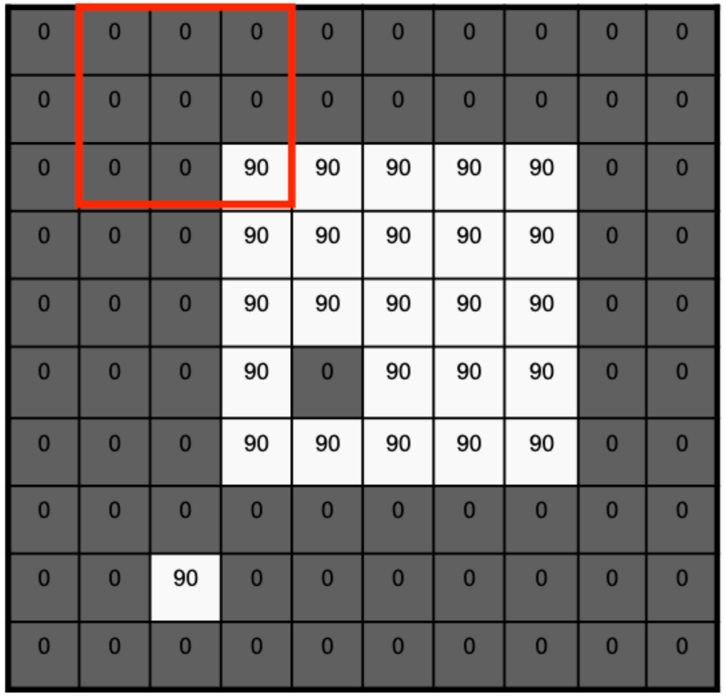


$$(f*g)[m,n]=\sum_{k,l} f[k,l]\,g[m-k,n-l]$$

Image borrowed from Stanford CS131

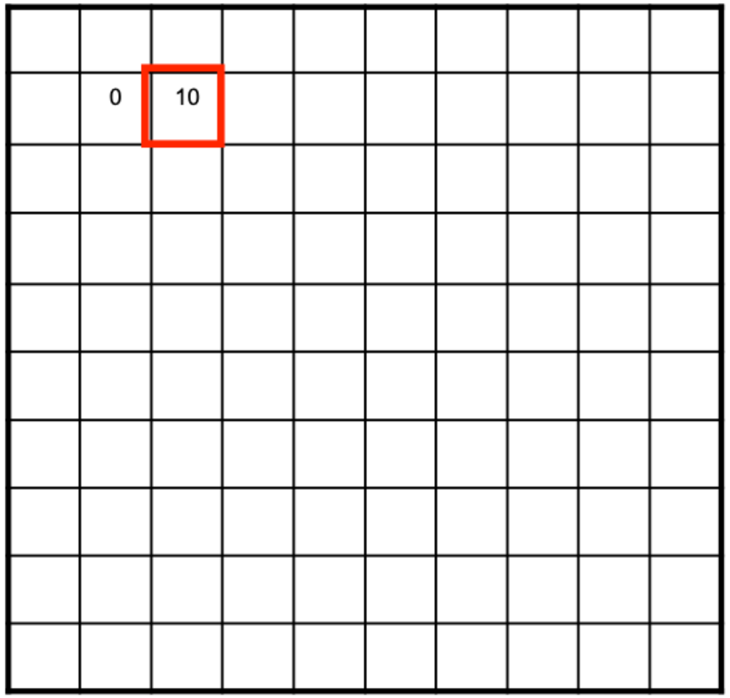
2D Discrete Filter Example: Moving Average

$f[m,n]$



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[m,n]$

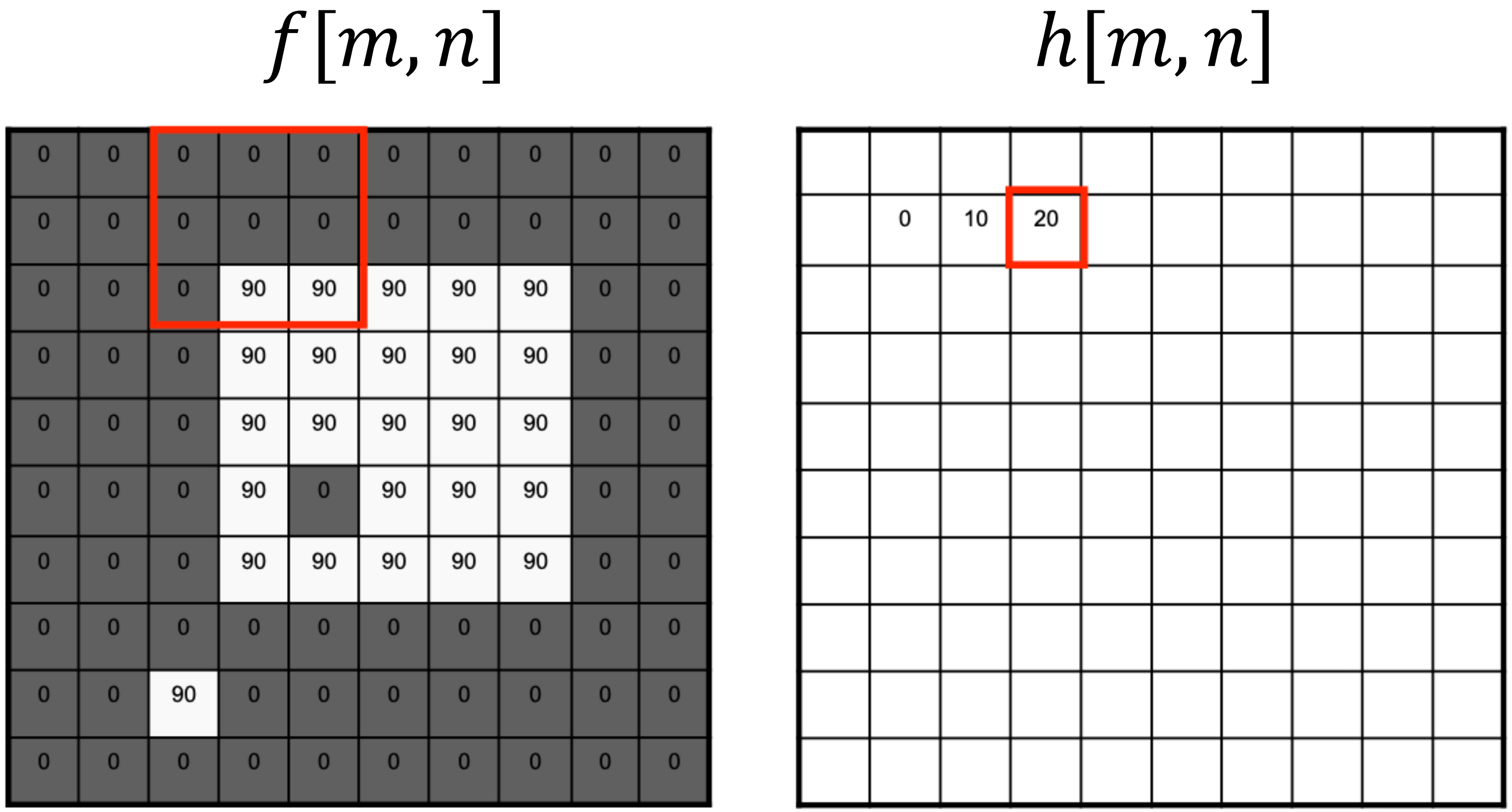


	0	10							

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

Image borrowed from Stanford CS131

2D Discrete Filter Example: Moving Average



$f[m,n]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

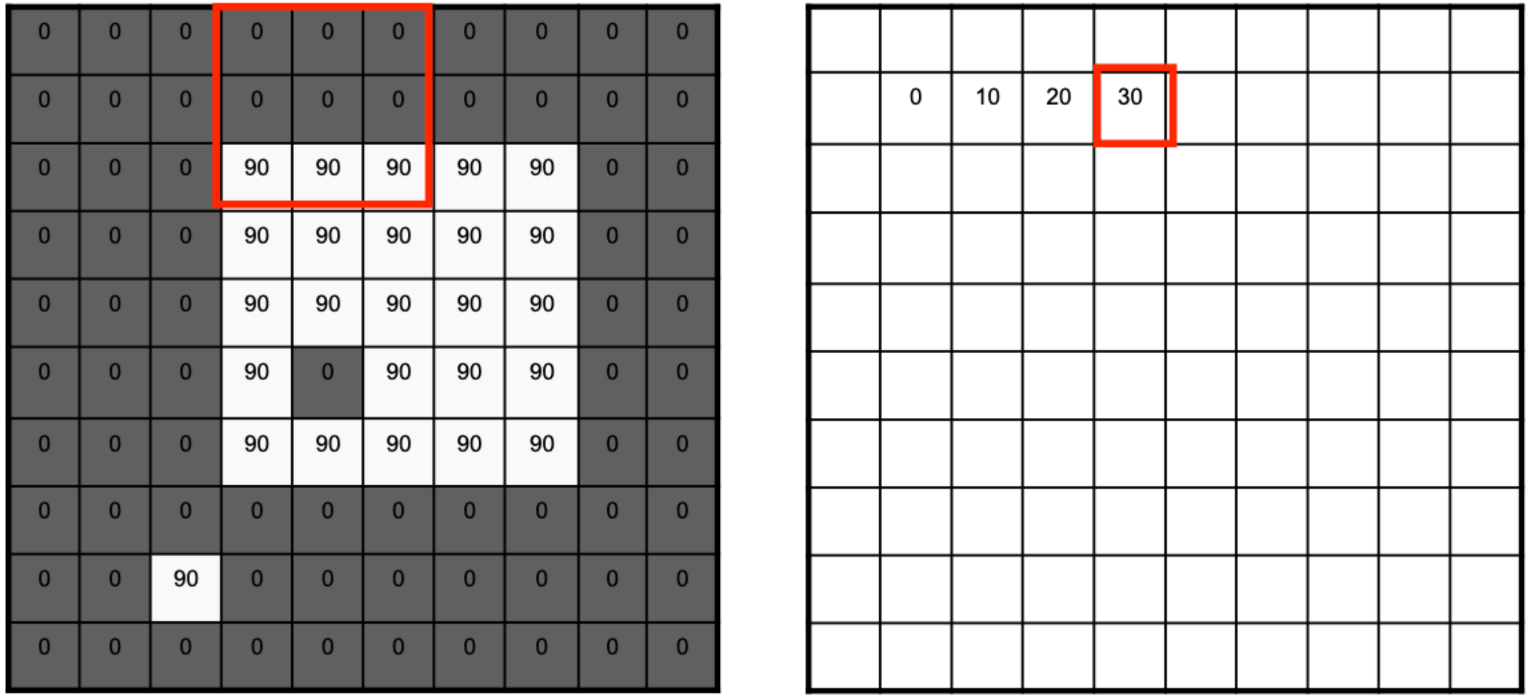
$h[m,n]$

	0	10	20						

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

Image borrowed from Stanford CS131

2D Discrete Filter Example: Moving Average



$f[m,n]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[m,n]$

	0	10	20	30					

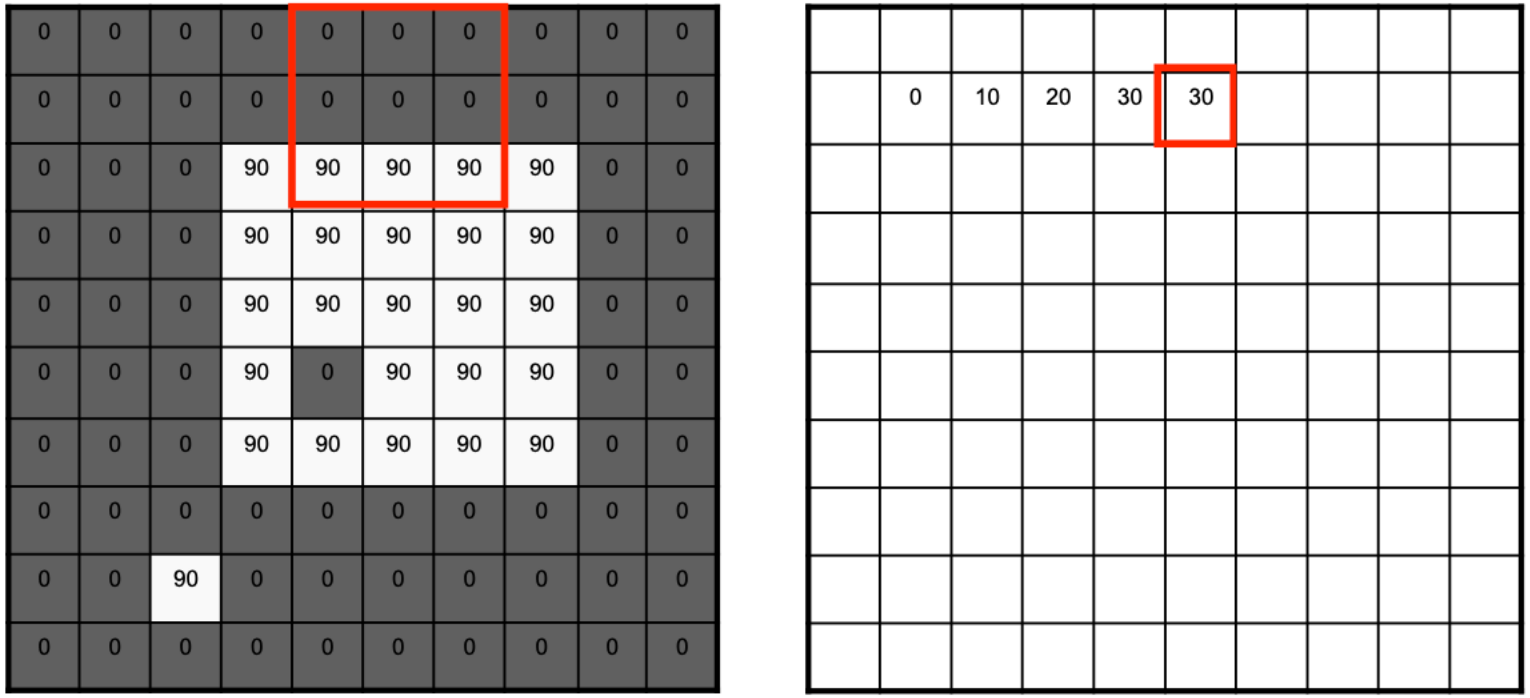
$$(f * g)[m,n] = \sum_{k,l} f[k,l]\, g[m-k, n-l]$$

Image borrowed from Stanford CS131

2D Discrete Filter Example: Moving Average

$f[m,n]$

$h[m,n]$



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

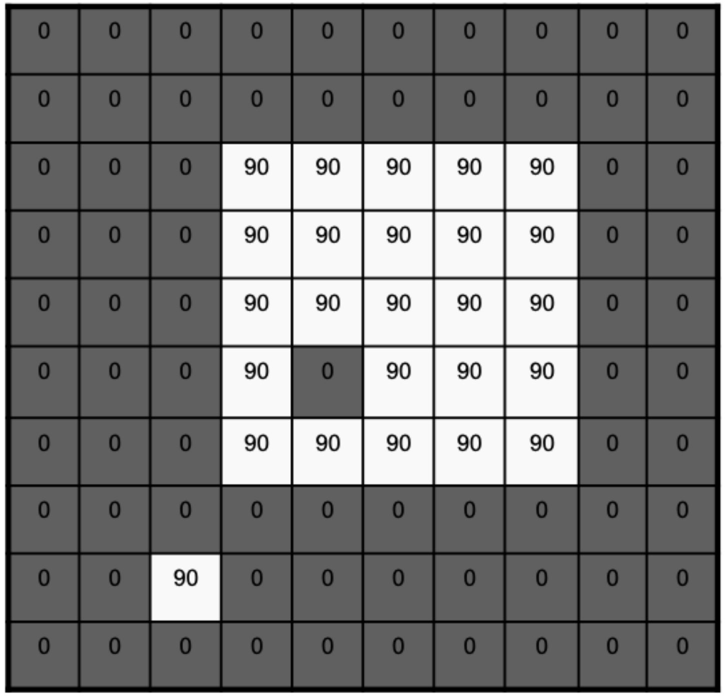
	0	10	20	30	30				

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

Image borrowed from Stanford CS131

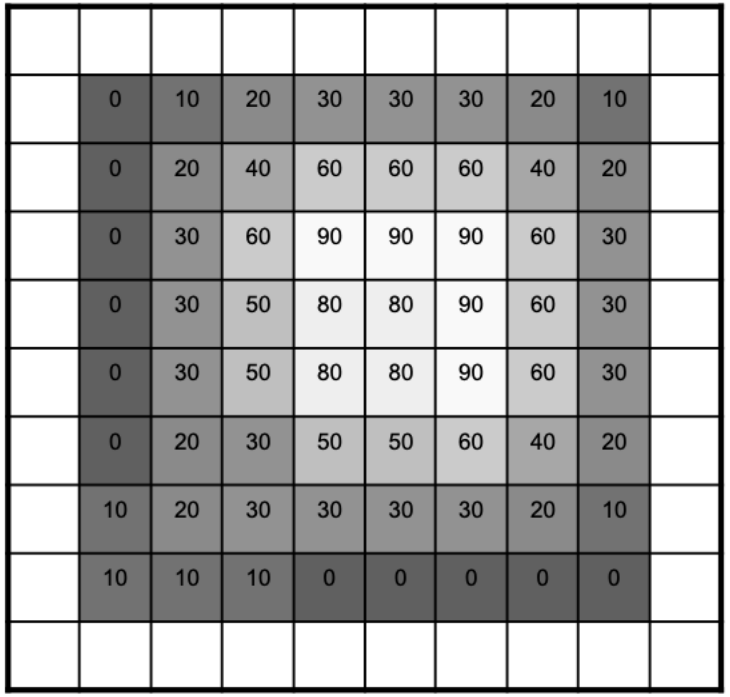
2D Discrete Filter Example: Moving Average

$f[m,n]$



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[m,n]$



	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

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

Source: S. Seitz

Image borrowed from Stanford CS131

Summary of Moving Average

- Replaces each pixel with an average of its neighborhood.
- Achieve smoothing effect (remove sharp features)

$g[\cdot\,,\cdot\,]$

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

Slide borrowed from Stanford CS131

Non-linear Filtering Example: Binarization via Thresholding

Define a threshold τ, *e.g.*, $\tau = 100$.

$$h[m,n] = \begin{cases} 1, & f[n,m] > \tau \\ 0, & \text{otherwise.} \end{cases}$$

Image borrowed from Stanford CS131

Is thresholding a linear system?

- f1[n,m] + f2[n,m] > T
- f1[n,m] < T
- f2[n,m]<T

No!

Edge Detection

Start with A Task: Lane Detection

How to detect the lane?

https://medium.com/@realderektan/self-driving-car-project-part-1-lane-lines-detector-6d960e2b023

Start with Detecting Edges

- Edge detector

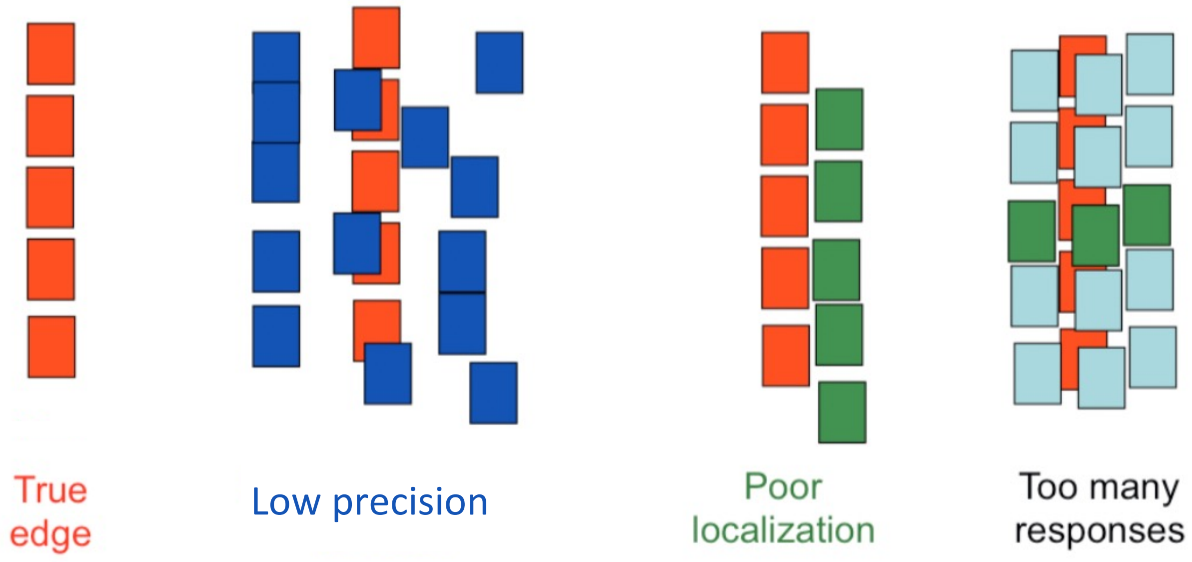
https://towardsdatascience.com/edge-detection-in-python-a3c263a13e03

What is an Edge?

- An edge is defined as a region in the image where there is a “significant” change in the pixel intensity values (or having high contrast) along one direction in the image, and almost no changes in the pixel intensity values (or low contrast) along its orthogonal direction.

https://towardsdatascience.com/edge-detection-in-python-a3c263a13e03

Criteria for Optimal Edge Detection



$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

TP: true positives, FP: false positives
TN: true negatives, FN:false negatives

- High precision: make sure all detected edges are true edges (via minimizing FP).
- High recall: make sure all edges can be detected (via minimizing FN).
- Good localization: minimize the distance between the detected edge and the ground truth edge
- Single response constraint: minimize redundant responses

Image borrowed from Stanford CS231A Lecture 10

What Causes An Edge?

- Depth discontinuity
- Surface orientation discontinuity
- Surface color discontinuity
- Illumination discontinuity

Image borrowed from Stanford CS231A Lecture 10

Characterizing Edges

- An edge is defined as a region in the image where there is a "significant" change in the pixel intensity values along one direction in the image, and almost no changes in the pixel intensity values along its orthogonal direction.

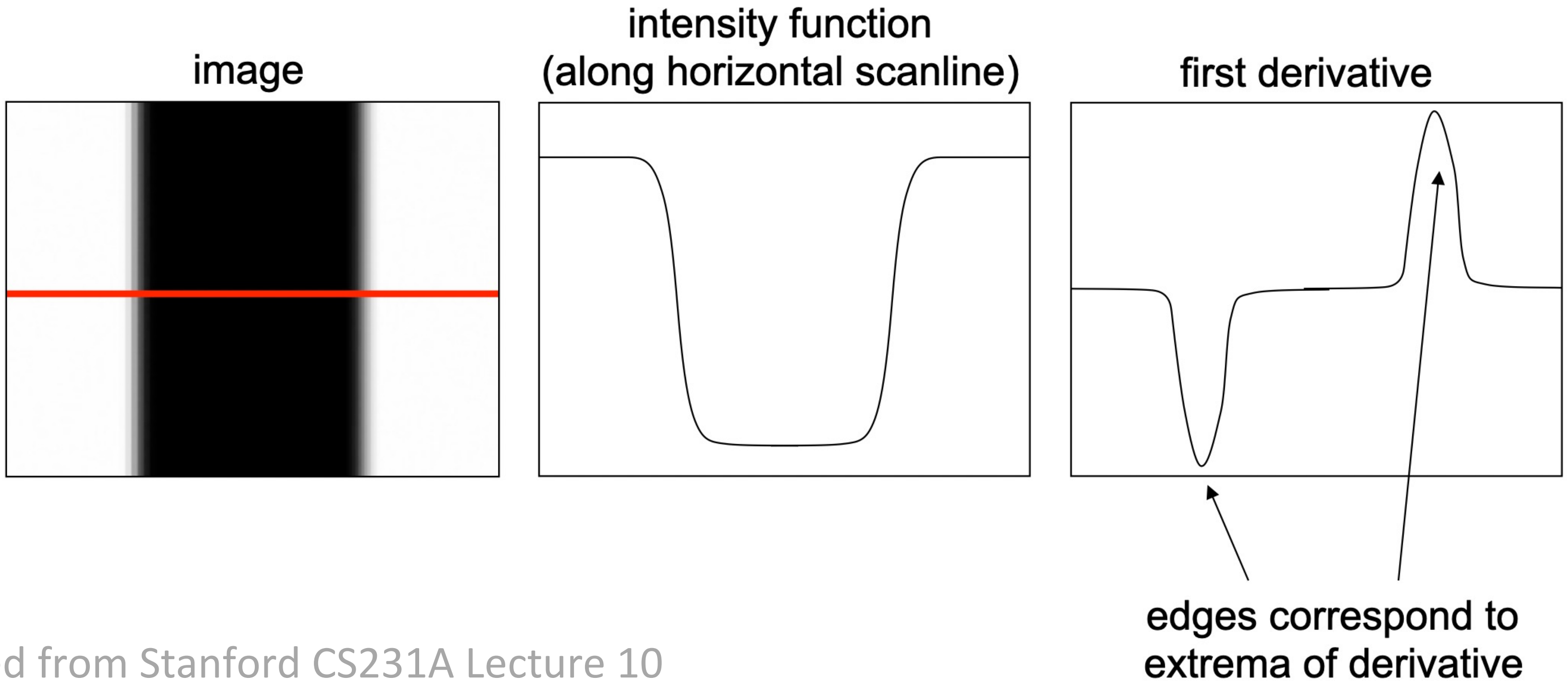
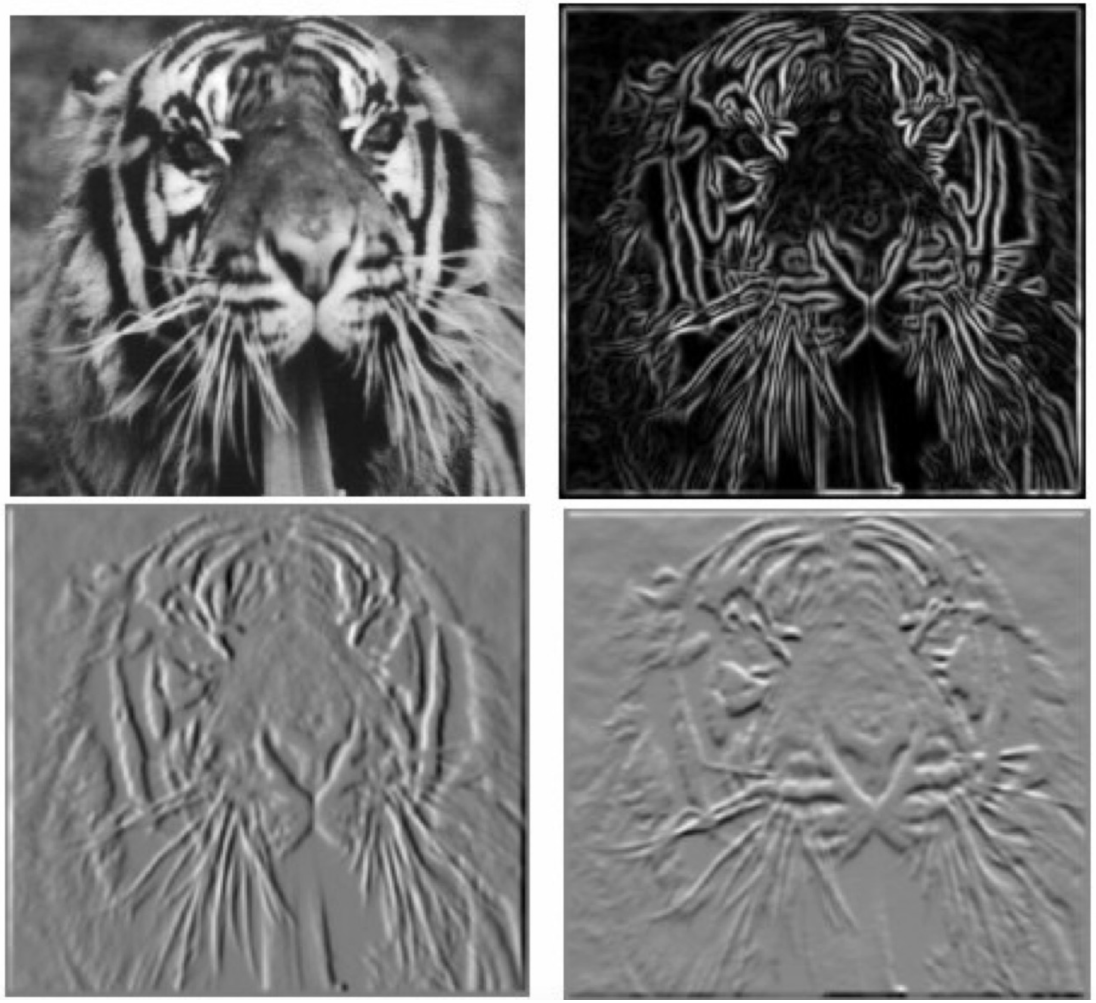


Image borrowed from Stanford CS231A Lecture 10

Visualizing Image Gradient



Original Image

Gradient magnitude

x-direction

y-direction

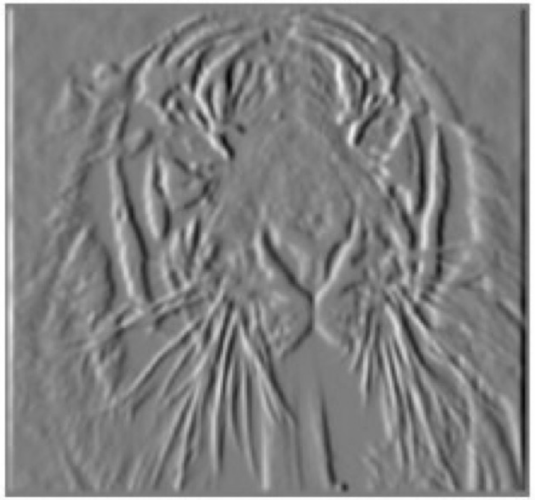
Gradient magnitude: $$||\nabla f|| = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2}$$

Source: Feifei Li

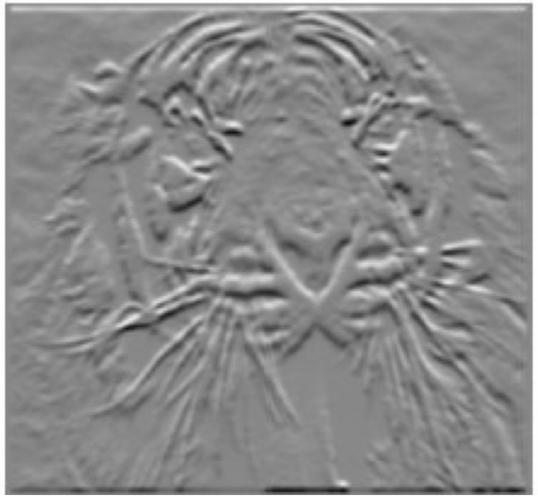
Problem

Original
Image

Gradient
magnitude



x-direction

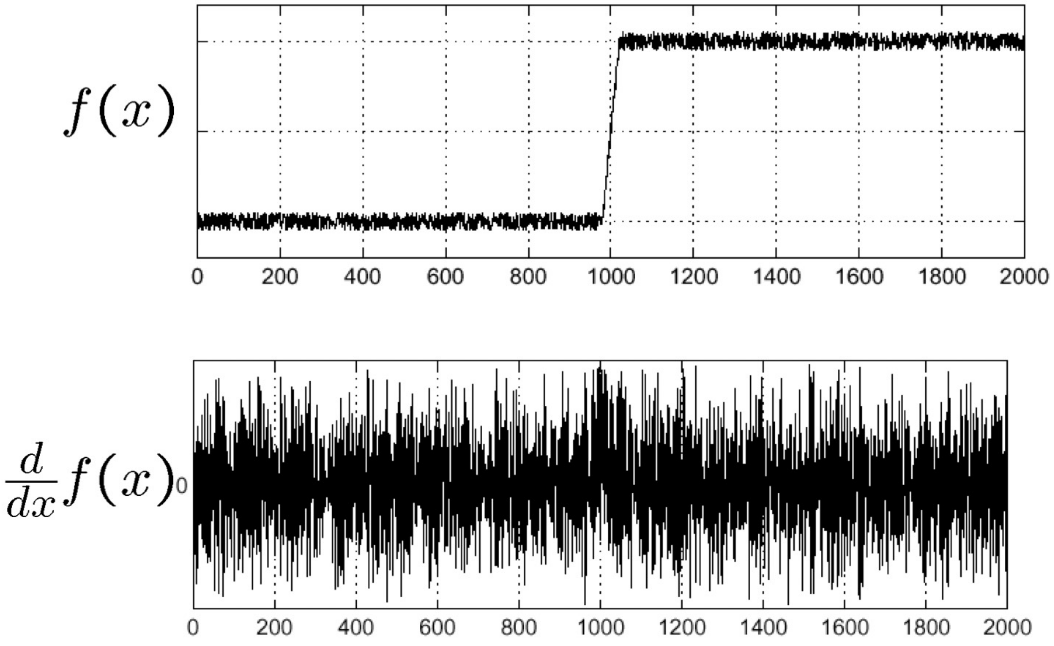


y-direction

- Gradient is non-zero everywhere. Where is the edges?

Effects of Noises

- Consider one row in the image

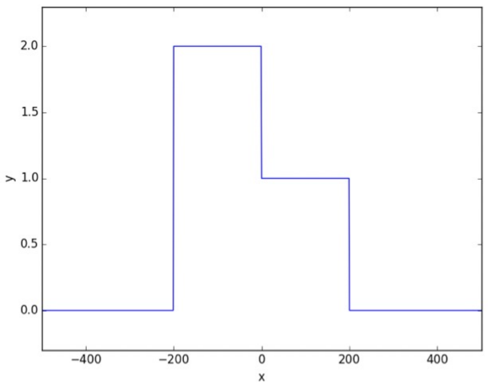


- Image gradients are too sensitive to noise.
- Gradients of the true edge is overwhelmed by noises.
- We need smoothing!

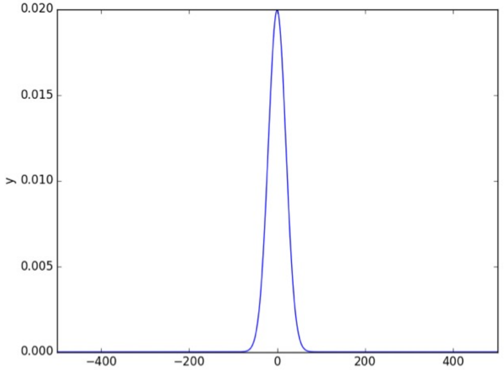
Source: Steven Seitz

Smoothing by Gaussian Filter

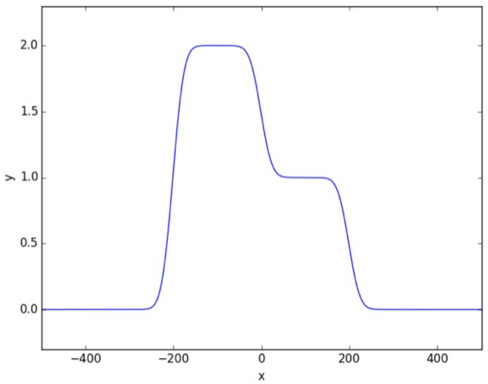
f = function of measurement



g = weighting function



f convolved with g (written f∗g)

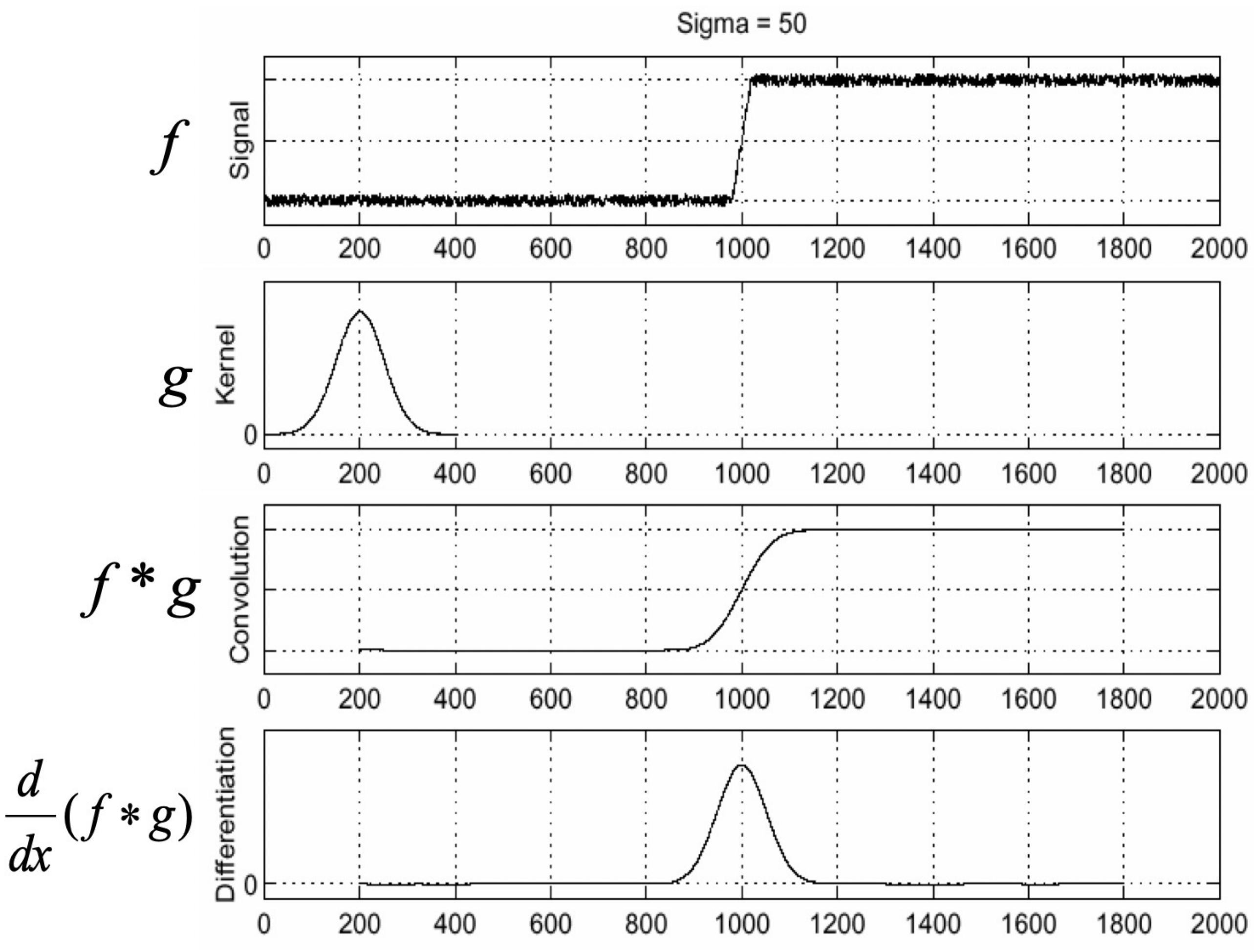


- Gaussian transforms to another Gaussian,
- low-pass filter!

$$g = \frac{1}{\sqrt{2\pi\sigma^2}} \exp - \frac{x^2}{2\sigma^2} \qquad \mathcal{F}(g) = \exp(-\frac{\sigma^2\omega^2}{2})$$

- The bigger σ is, the sharper $\mathcal{F}(g)$ is. When $\sigma \to +\infty$, filter all high-frequence parts and then the signal becomes a constant.
- The smaller σ is, the boarder $\mathcal{F}(g)$ is. When $\sigma \to 0$, $\mathcal{F}(g) = 1$, no filtering at all.

Smoothing by a Low-Pass Filter



Source: Steven Seitz

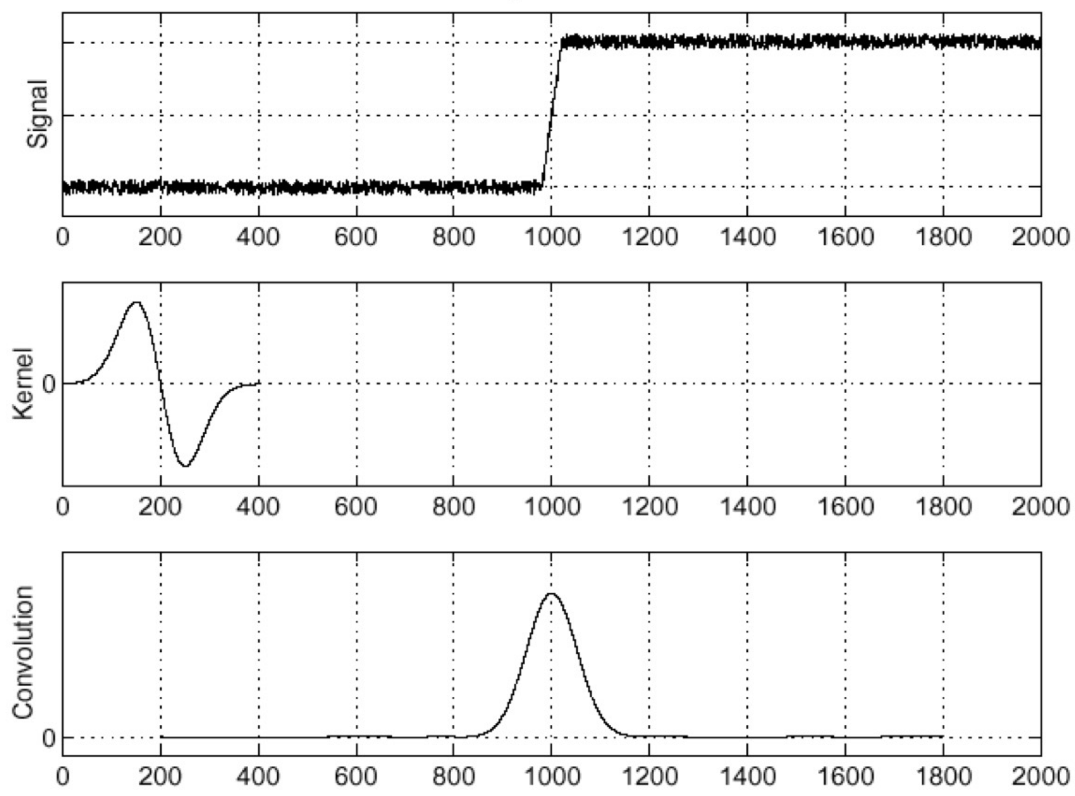
Derivative Theorem of Convolution

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

$$f$$

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

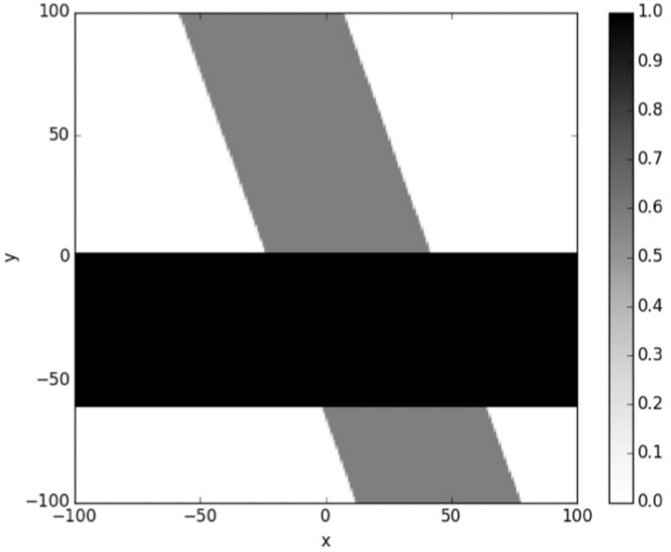
$$f * \frac{d}{dx} g$$



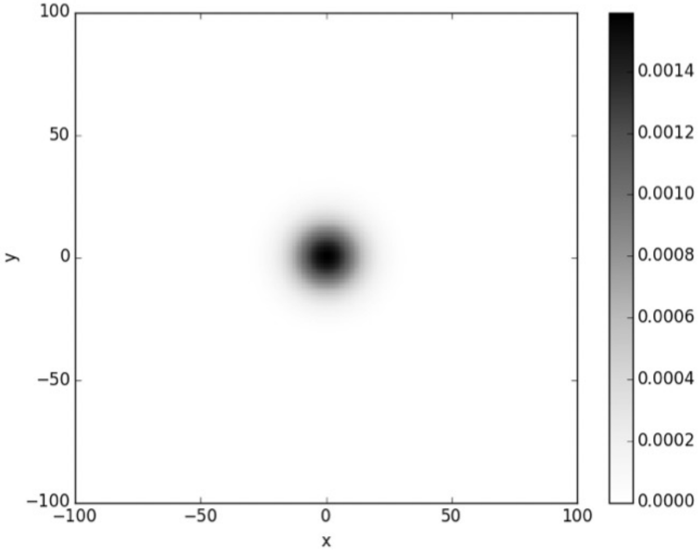
- Saves us one operation.

Two-Dimensional Convolution

f

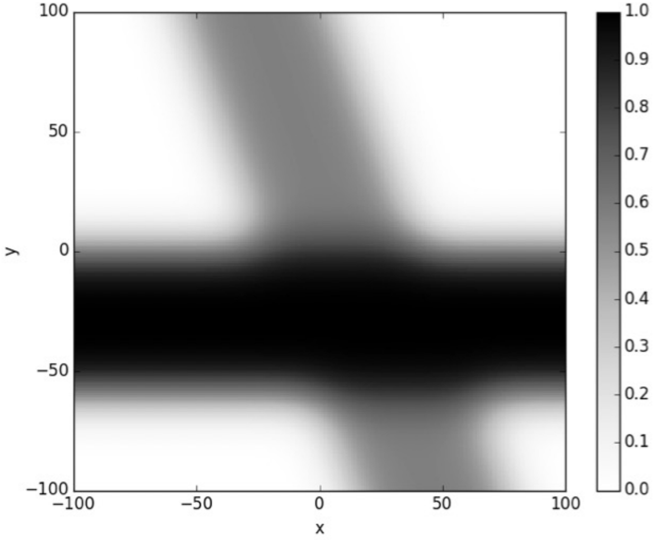


$*$



$$g = \frac{1}{2\pi\sigma^2} \exp - \frac{x^2 + y^2}{2\sigma^2}$$

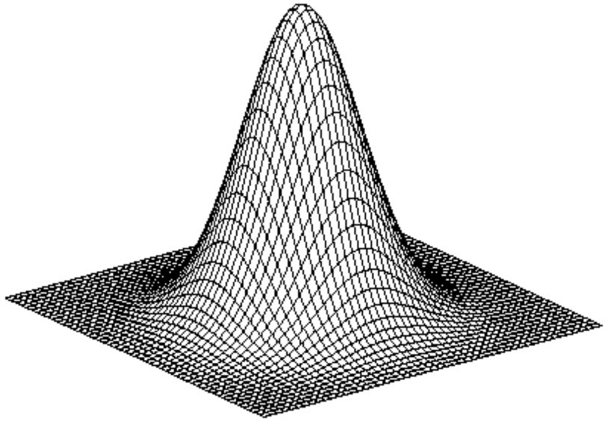
$=$



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

Source: Feifei Li

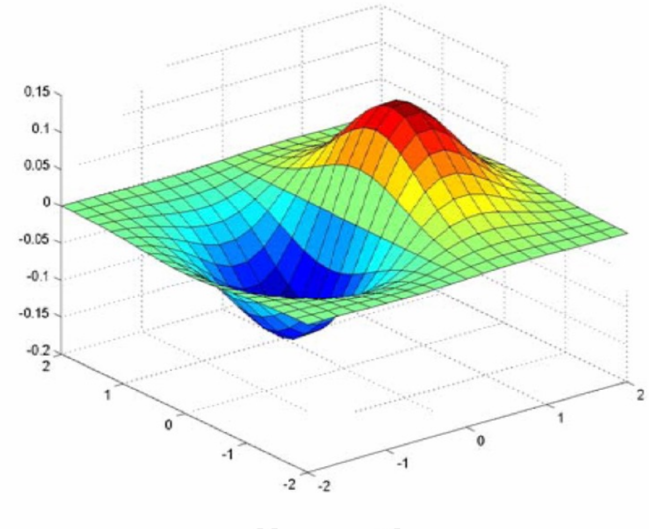
Derivative of 2D Gaussian Filter



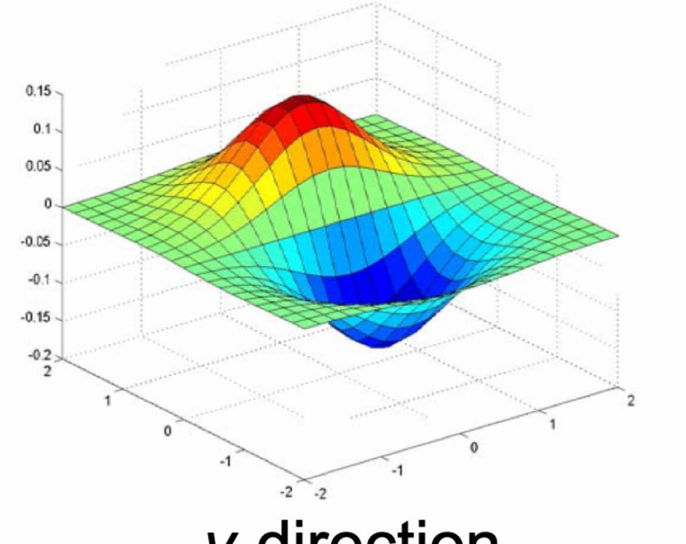
2D-gaussian

* [1 -1]

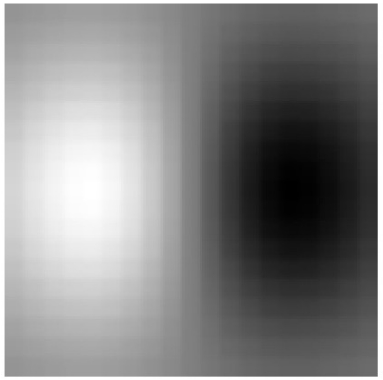
=

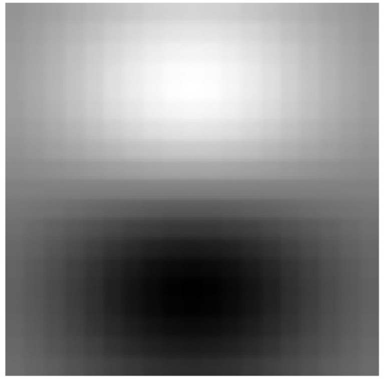


x-direction



y-direction





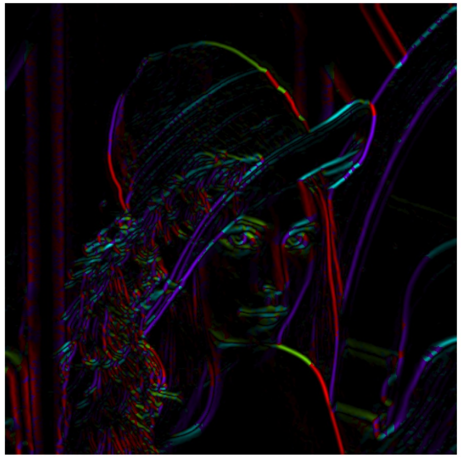
Source: Feifei Li

Compute Gradient


x-derivative of Gaussian

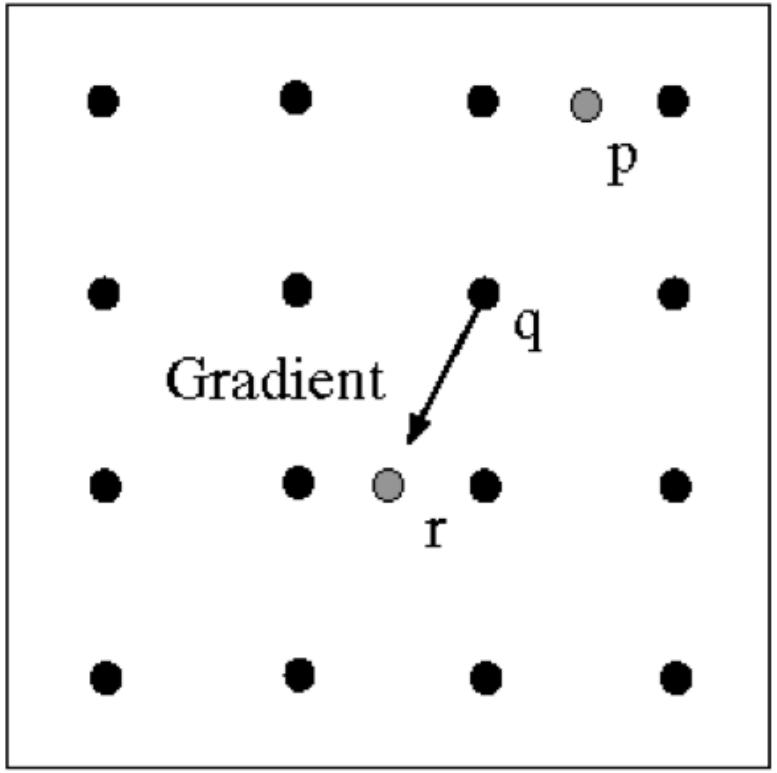

y-derivative of Gaussian


Gradient magnitude


Thresholding and Gradient orientation

Source: Feifei Li

Non-Maximal Suppression (NMS)

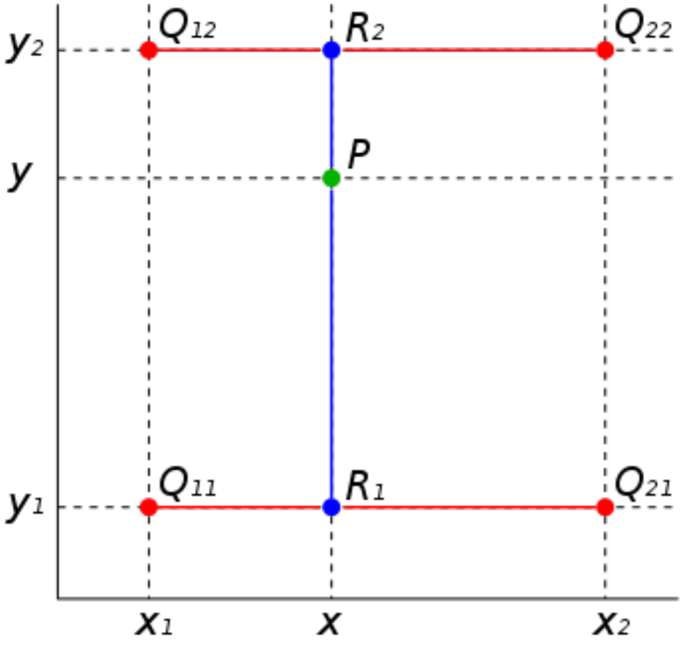


- For each point q on grids, compute the gradient $g(q)$.
- Move along the gradient to get two neighbors: $r = q + g(q), p = q - g(q)$
- Perform bilinear interpolation to get $g(p)$ and $g(r)$.
- If the magnitude of g(q) is larger than $g(p)$ and $g(r)$, q is a maximum that should be kept.

Source: J. Hayes

Bilinear Interpolation

For $P(x, y)$, given its four surrounding grid points $f(Q_{11}), f(Q_{12}), f(Q_{21})$ and $f(Q_{22})$, how to obtain $f(P)$ via *bilinear interpolation*?



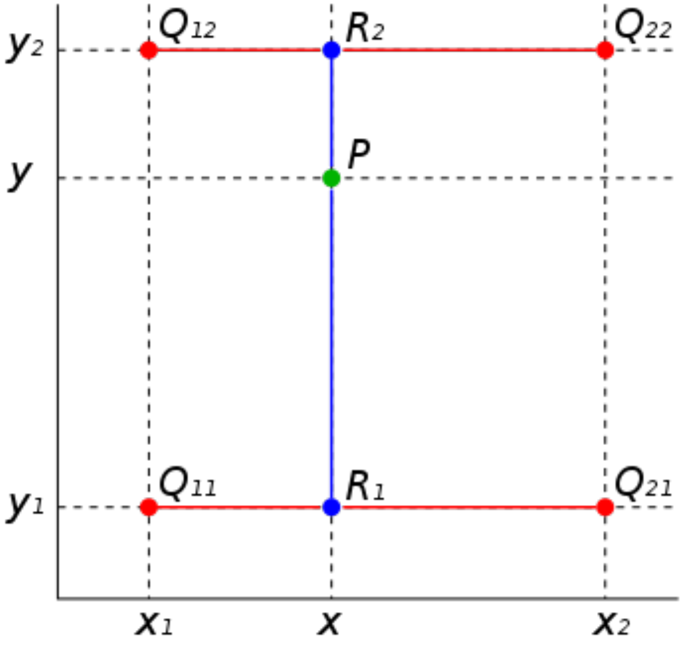
First, linear interpolate to obtain $f(R_1)$ and $f(R_2)$

R_1: $$f(x, y_1) = \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}),$$

R_2: $$f(x, y_2) = \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}).$$

Bilinear Interpolation

For $P(x, y)$, given its four surrounding grid points $f(Q_{11}), f(Q_{12}), f(Q_{21})$ and $f(Q_{22})$, how to obtain $f(P)$ via *bilinear interpolation*?



First, linear interpolate to obtain $f(R_1)$ and $f(R_2)$

R_1: $$f(x, y_1) = \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}),$$

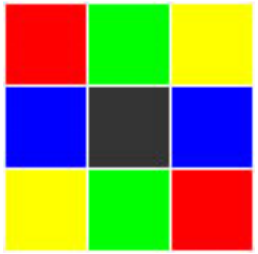
R_2: $$f(x, y_2) = \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}).$$

Then, linear interpolate between $f(R_1)$ and $f(R_2)$ to obtain $f(P)$:

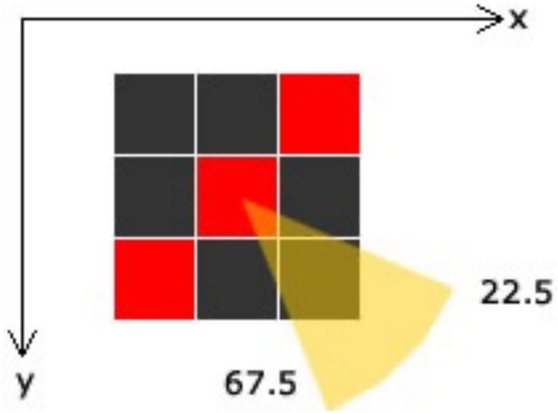
P: $$\begin{aligned} f(x, y) &= \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2) \\ &= \frac{y_2 - y}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \right) + \frac{y - y_1}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \right) \end{aligned}$$

A Simplified Version of NMS

The orientation of each pixel is put into one of the four bins.



Example: gradient orientation from 22.5 to 67.5 degrees



To check if the central red pixel belongs to an edge, you need to check if the gradient is maximum at this point. You do this by comparing its magnitude with the top left pixel and the bottom right pixel.

Before and After NMS

Thin multi-pixel wide “ridges” down to single pixel width

Source: J. Hayes

Hysteresis Thresholding

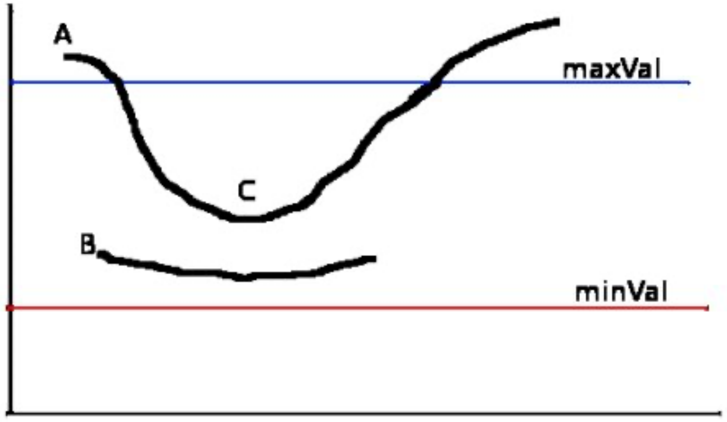
- Use a high threshold (maxVal) to start edge curves and a low threshold (minVal) to continue them.
  - Pixels with gradient magnitudes $>$ maxVal should be reserved
  - Pixels with gradient magnitudes $<$ minVal should be removed.
  - How to decide maxVal and minVal? Examples:
    - maxVal = 0.3 $\times$ average magnitude of the pixels that pass NMS
    - minVal = 0.1 $\times$ average magnitude of the pixels that pass NMS

Source: J. Hayes

Edge Linking

- Drop-outs?

- Now using the direction information and the lower threshold, we'll "grow" these edges.
  - If the current pixel is not an edge, check the next one.
  - If it is an edge, check the two pixels in the direction of the edge (ie, perpendicular to the gradient direction). If either of them (or both)



Edge Linking

- Drop-outs?

- Now using the direction information and the lower threshold, we'll "grow" these edges.
  - If the current pixel is not an edge, check the next one.
  - If it is an edge, check the two pixels in the direction of the edge (ie, perpendicular to the gradient direction). If either of them (or both)
    - have the direction in the same bin as the central pixel
    - gradient magnitude is greater than minVal
    - they are the maximum compared to their neighbors (NMS for these pixels), then you can mark these pixels as an edge pixel

Edge Linking

- Drop-outs?
- Now using the direction information and the lower threshold, we'll "grow" these edges.
  - If the current pixel is not an edge, check the next one.
  - If it is an edge, check the two pixels in the direction of the edge (ie, perpendicular to the gradient direction). If either of them (or both)
    - have the direction in the same bin as the central pixel
    - gradient magnitude is greater than minVal
    - they are the maximum compared to their neighbors (NMS for these pixels), then you can mark these pixels as an edge pixel
  - Loop until there are no changes in the image Once the image stops changing, you've got your canny edges! That's it! You're done!

Canny Edge Detector

- The most widely used edge detector in computer vision
- Canny shows that the first derivative of the Gaussian closely approximates the operator that optimizes the product of signal-to-noise ratio and localization.

JJ. Canny, A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

Tradeoff between Smoothing and Localization

original

Canny with $\sigma = 1$

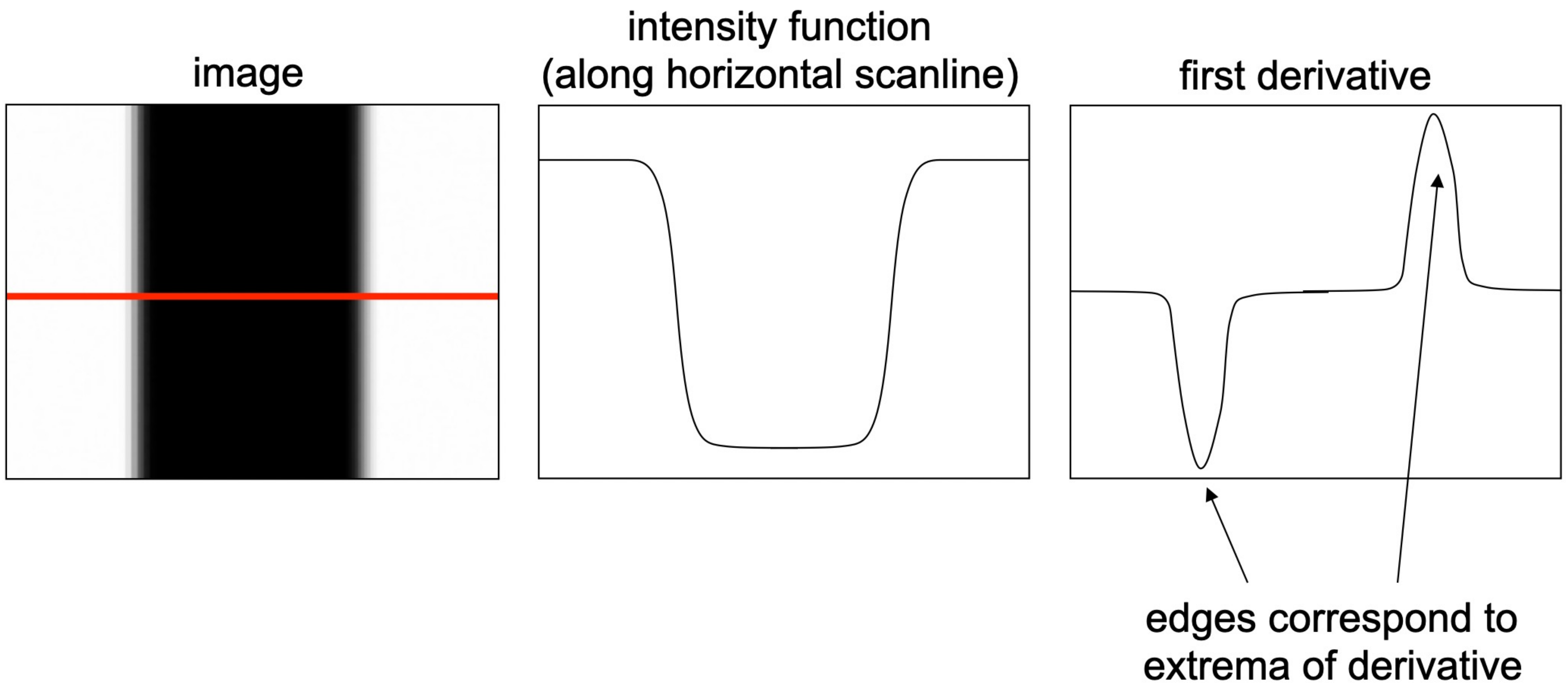
Canny with $\sigma = 2$

- Note a larger σ corresponds to stronger smoothing.
- Smoothed derivative reduces noises but blurs edges.
- Find edges at different scales.

Image credit: J. Hayes

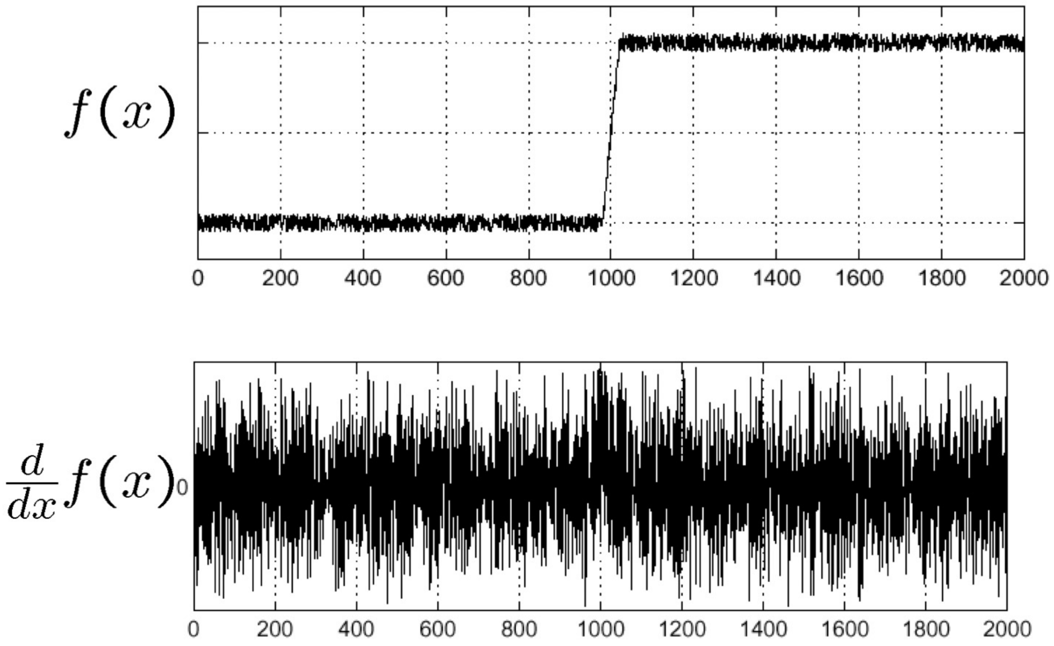
Summary of Edge Detection

- What is an edge?



Summary of Edge Detection

- Edge: where pixel intensity changes drastically
- Compute image gradient to find edge, however noises can be overwhelming and fail the detection



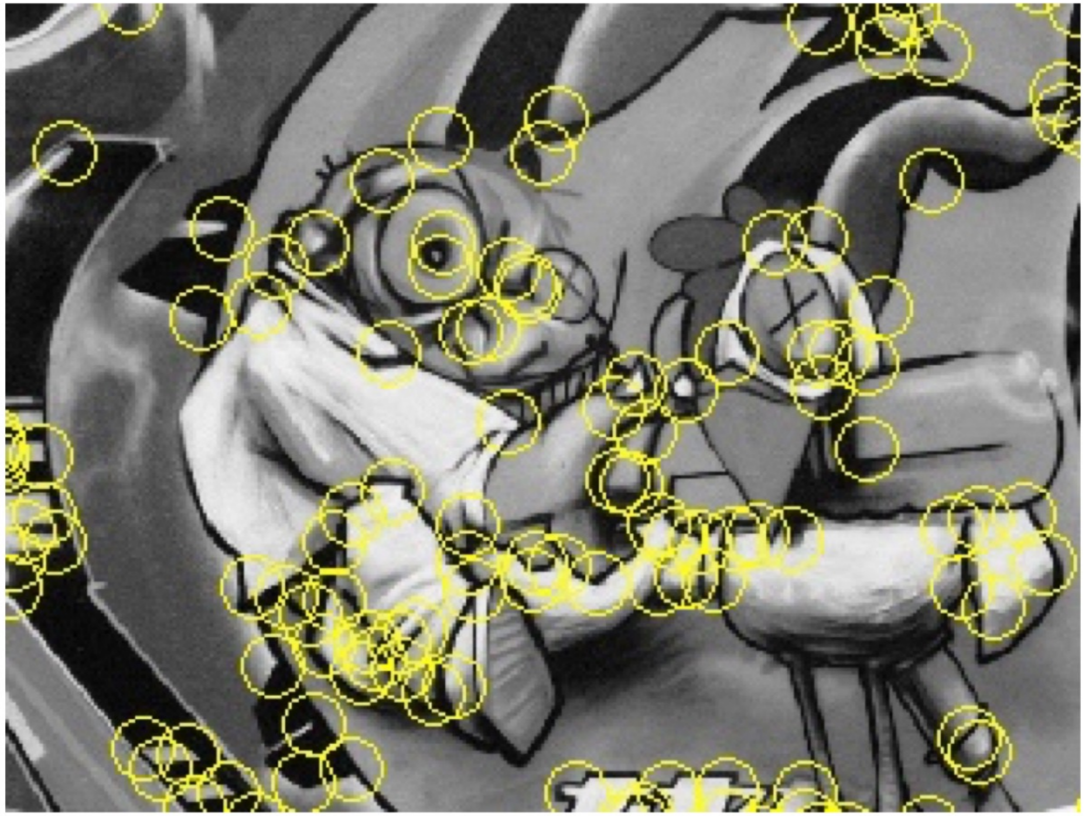
Summary of Canny Edge Detection

- Edge: where pixel intensity changes drastically
- Jointly detecting edge and smoothing by convolving with the derivative of a Gaussian filter
- Non-maximal suppression
- Thresholding and linking (hysteresis):

Keypoint Detection

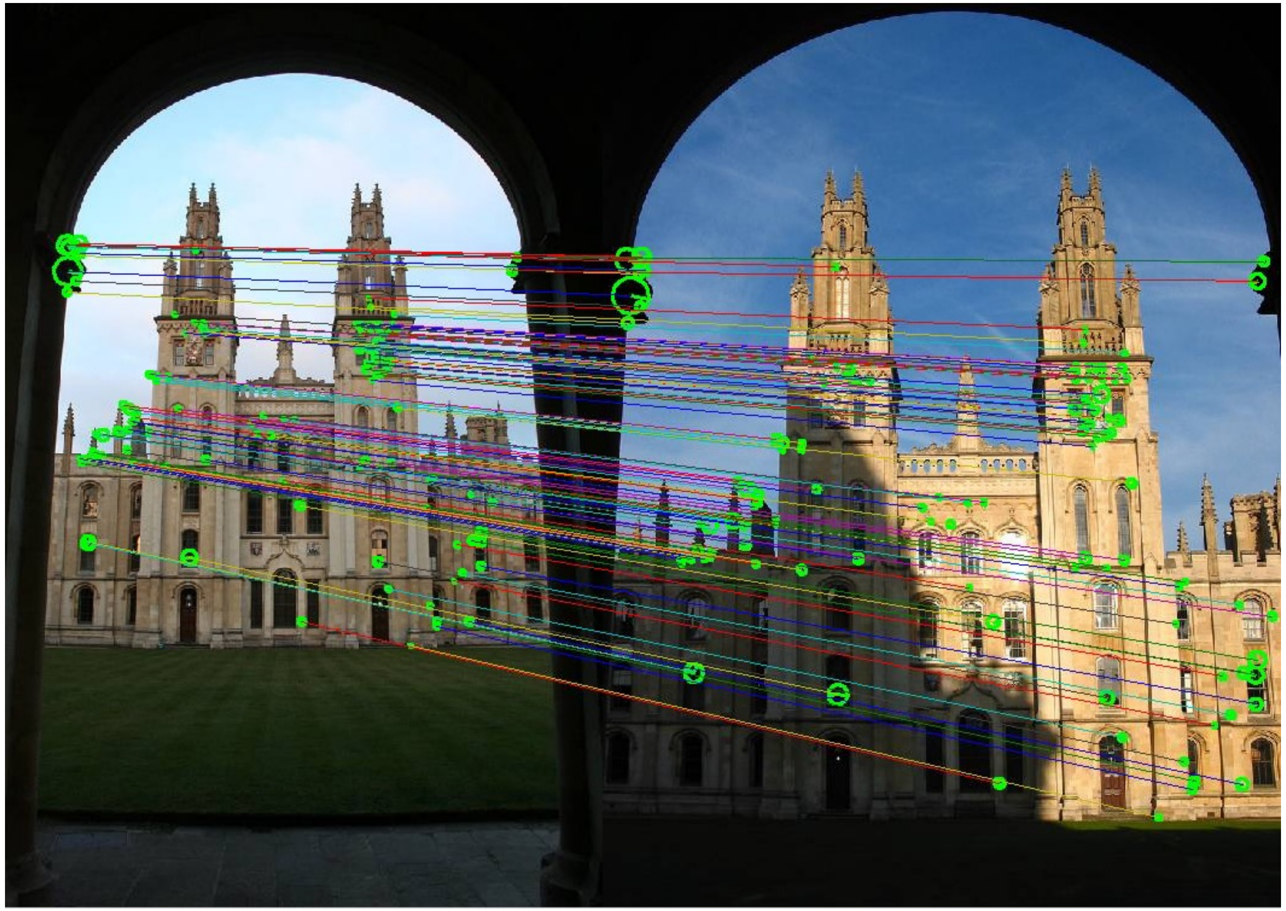
Some slides are borrowed from Stanford CS131.

Keypoint Localization



- In addition to edges, keypoints are also important to detect.

Applications: Image Matching



Separately detect keypoints and then find matching.

Slide borrowed from Stanford CS131

What Points are Keypoints?

- Saliency: interesting points

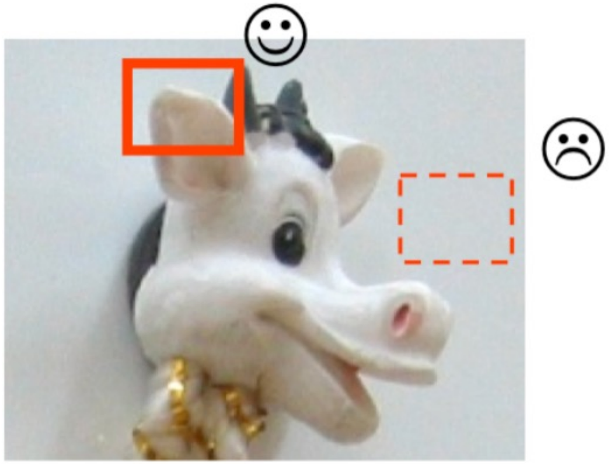
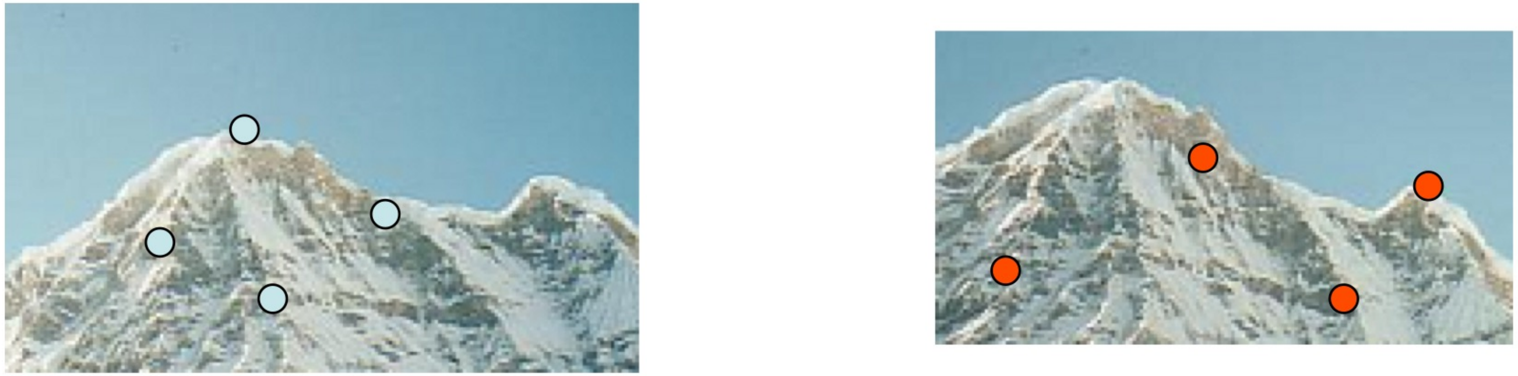


Image borrowed from Stanford CS131

More Requirements

- Saliency: interesting points
- Repeatability: detect the same point independently in both images



No chance to match!

Image borrowed from Stanford CS131

More Requirements

- Repeatability: detect the same point independently in both images
- Saliency: interesting points
- Accurate localization

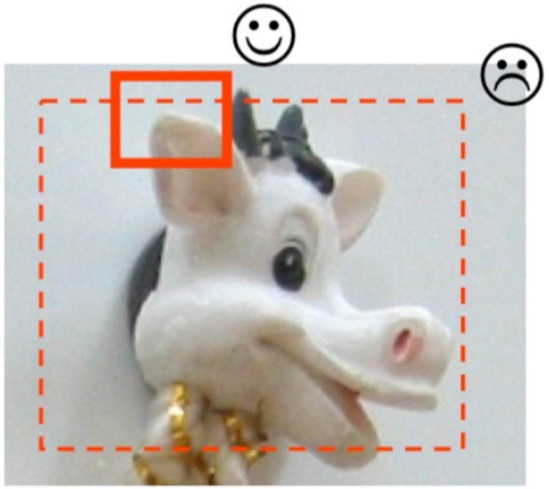
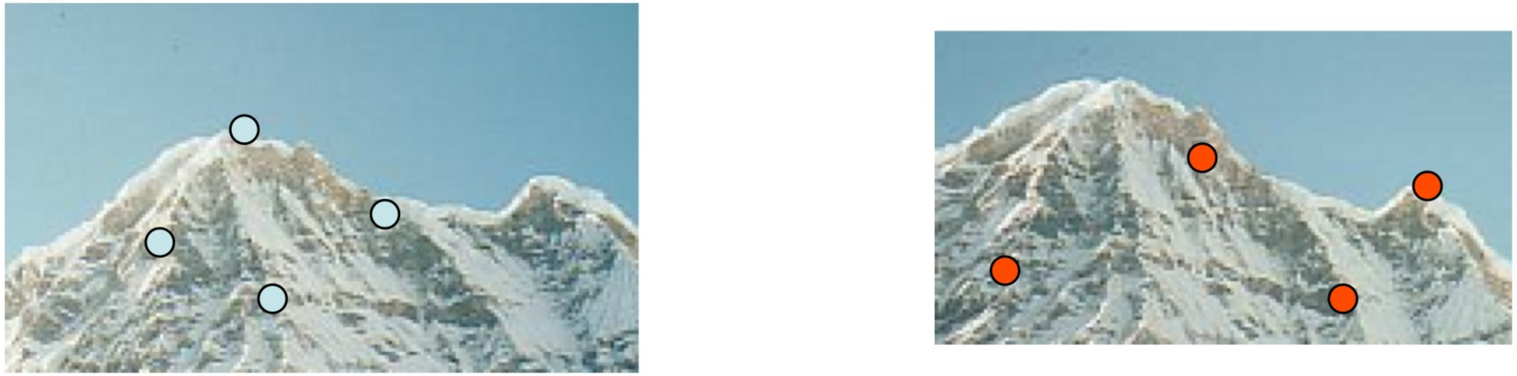


Image borrowed from Stanford CS131

More Requirements

- Repeatability: detect the same point independently in both images
- Saliency: interesting points
- Accurate localization
- Quantity: sufficient number

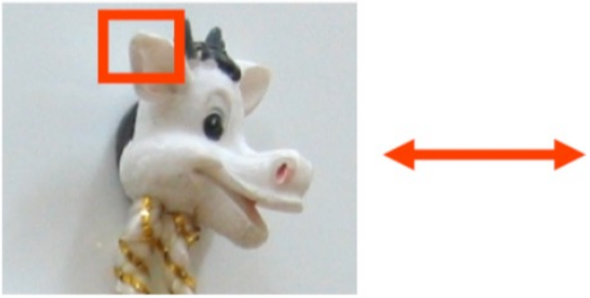


No chance to match!

Image borrowed from Stanford CS131

Repeatability and Invariance

- For a keypoint detector to be repeatable, it has to be invariant to:
  - Illumination
  - Image scale
  - Viewpoint



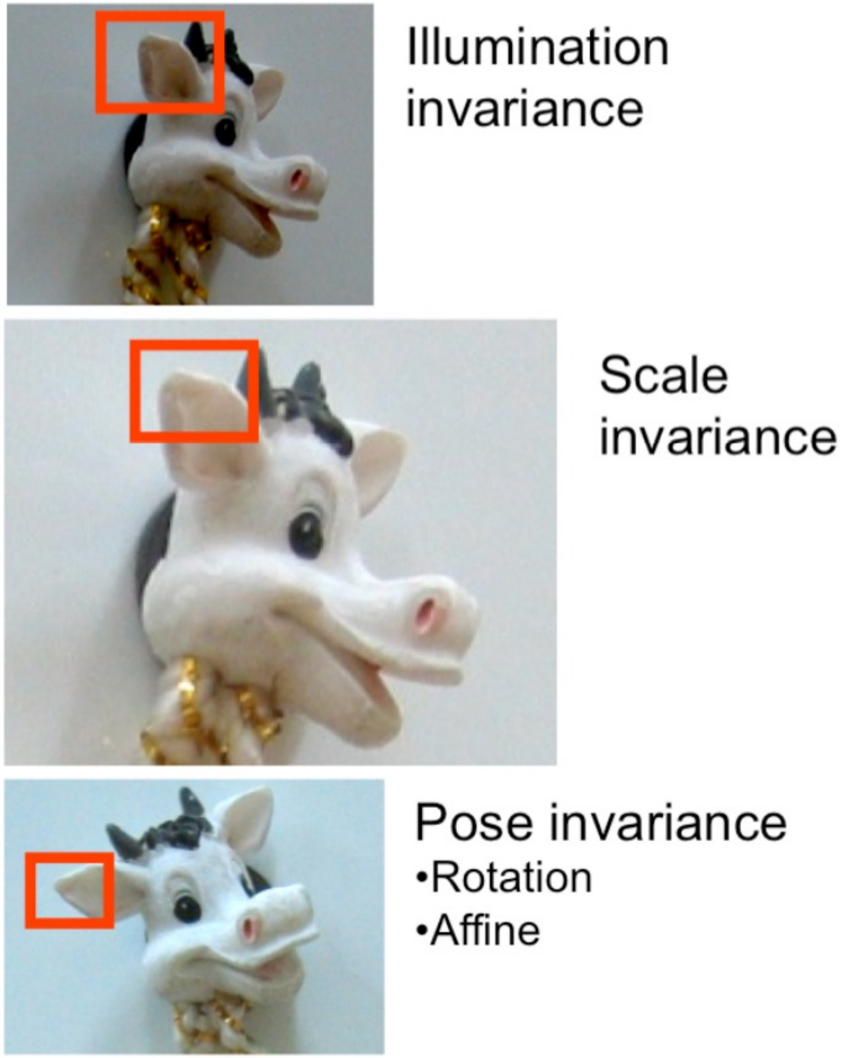
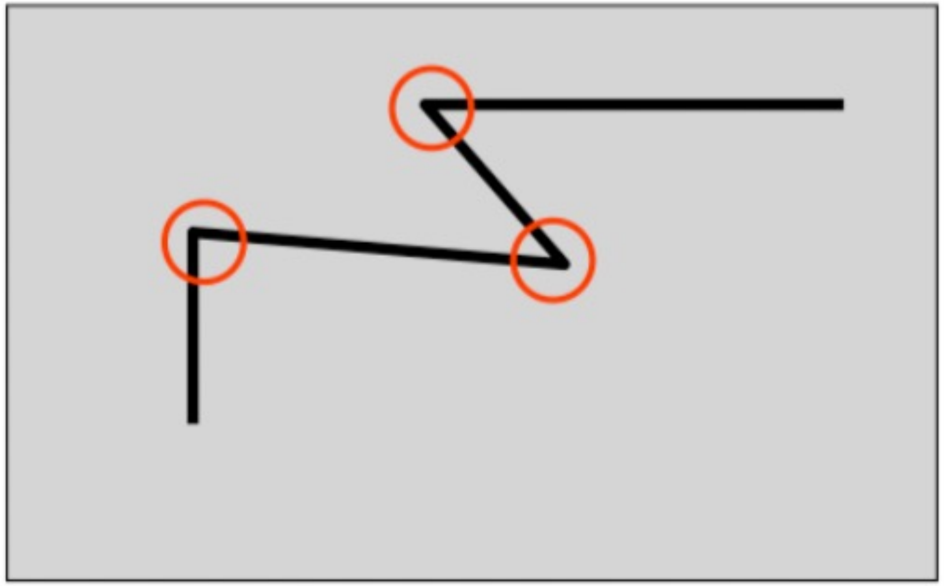


Image borrowed from Stanford CS131

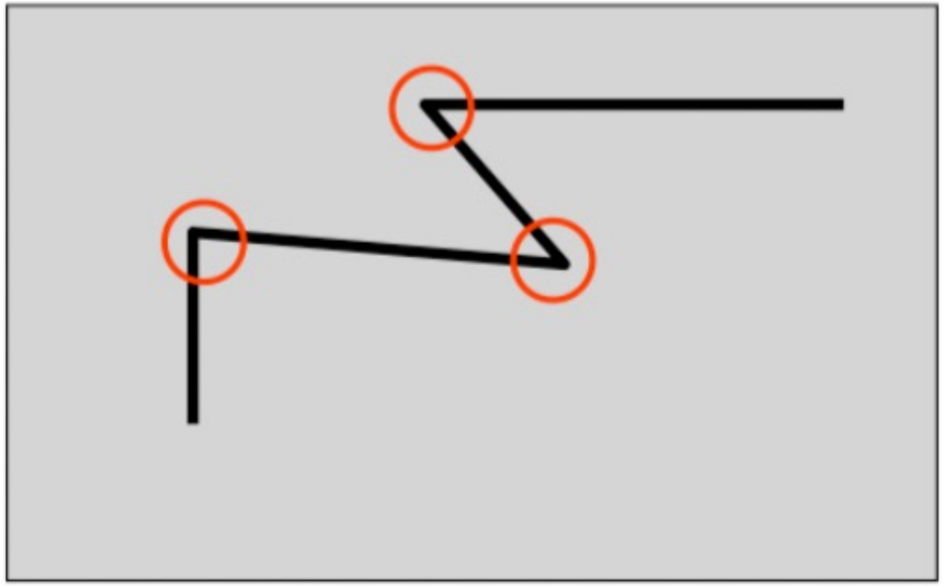
Corners as Keypoints



- Corners are such kind of keypoints, because they are
  - Salient;
  - Repeatable (one corner would still be a corner from another viewpoint);
  - Sufficient (usually an image comes with a lot of corners);
  - Easy to localize.

Image borrowed from Stanford CS131

The Properties of a Corner

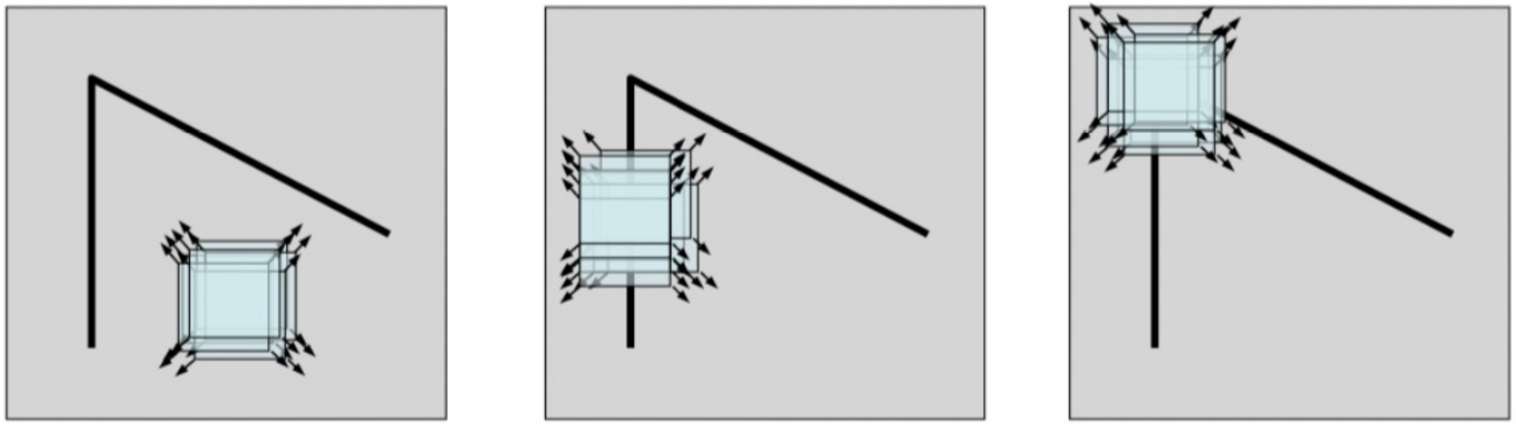


- The key property of a corner: In the region around a corner, image gradient has two or more dominant directions

Image borrowed from Stanford CS131

The Basic Idea of Harris Corner

- Move a window and explore intensity changes within the window



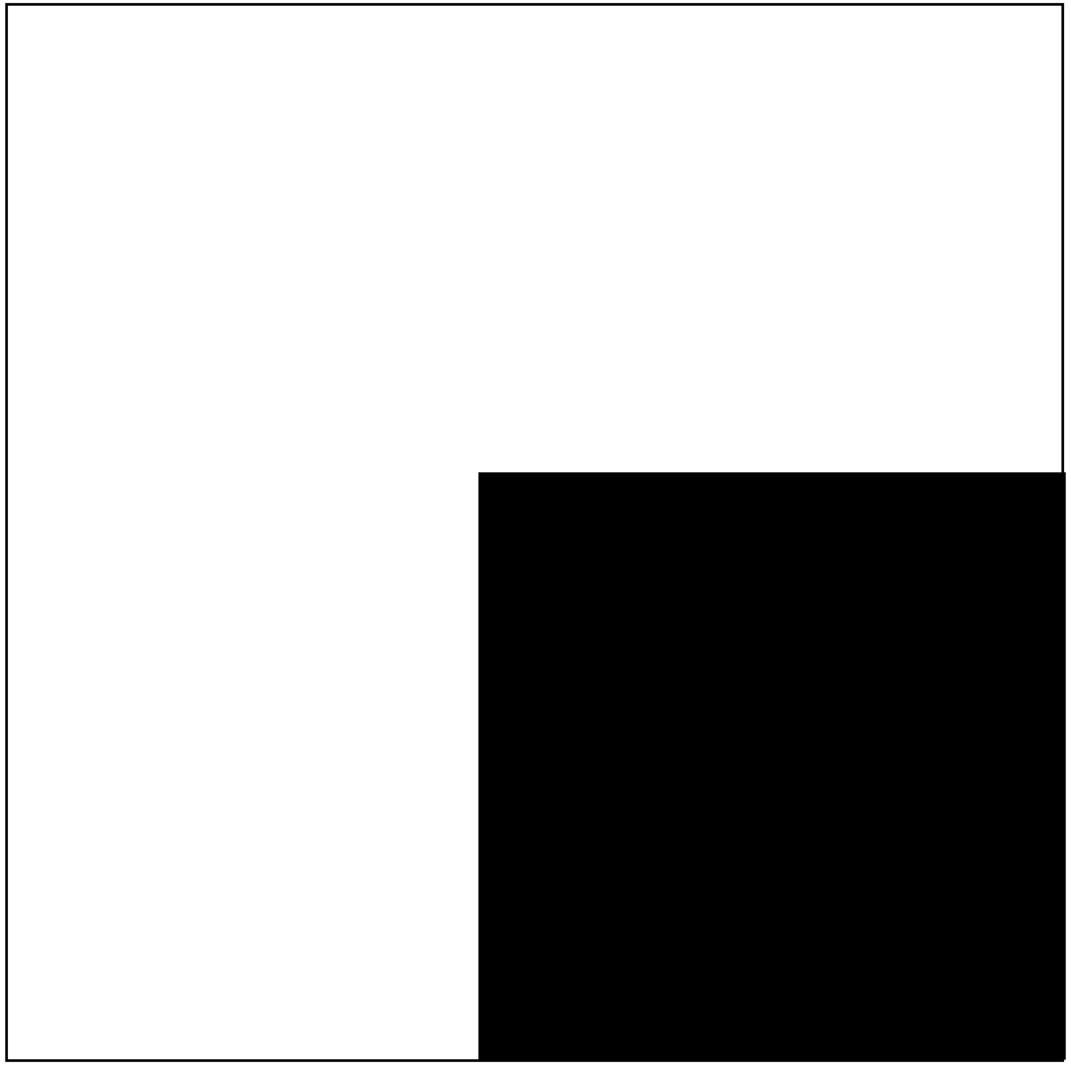
Flat region: no change in all directions

Edge: no change along the edge direction

Corner: significant change in all directions

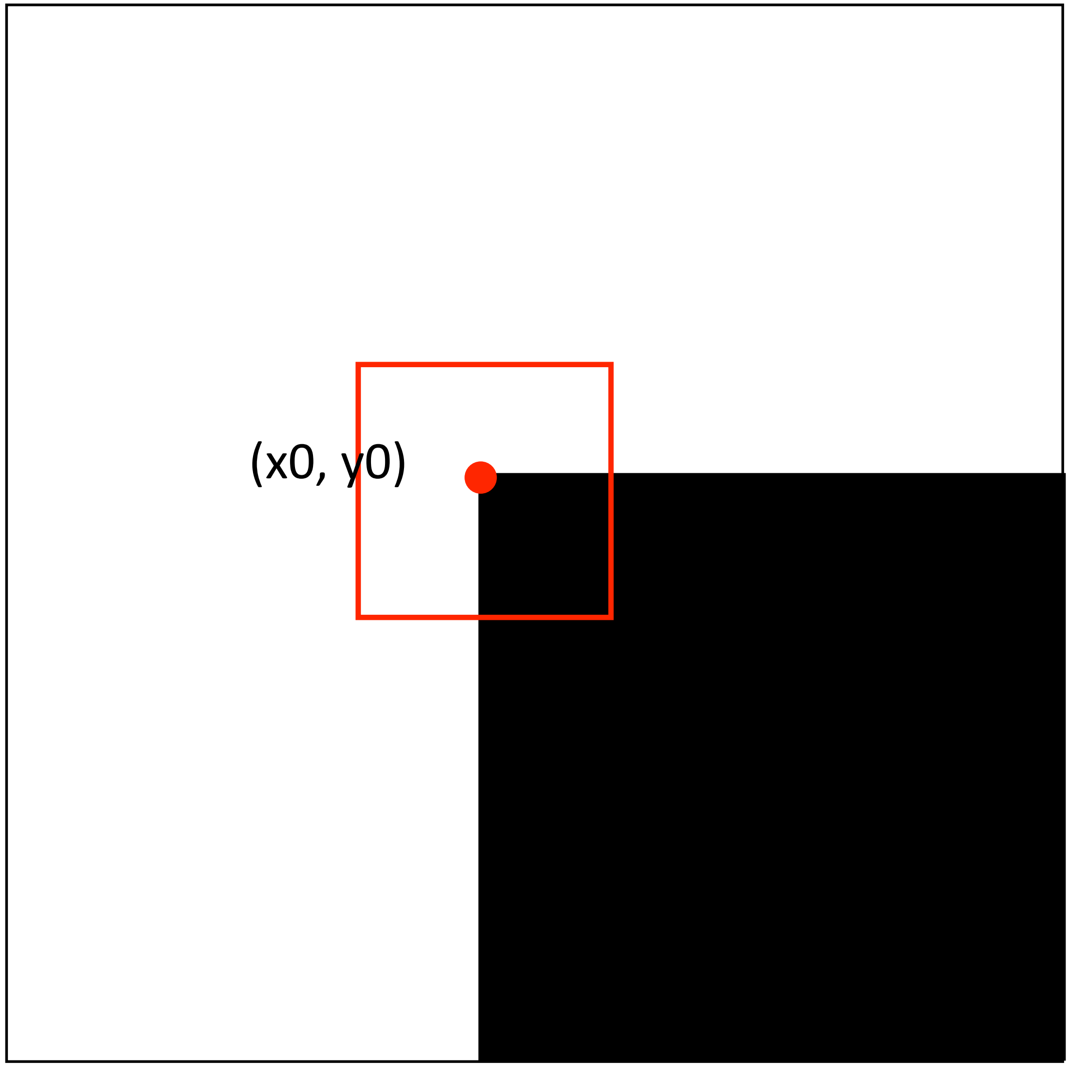
Image borrowed from Stanford CS131

The Basic Idea of Harris Corner



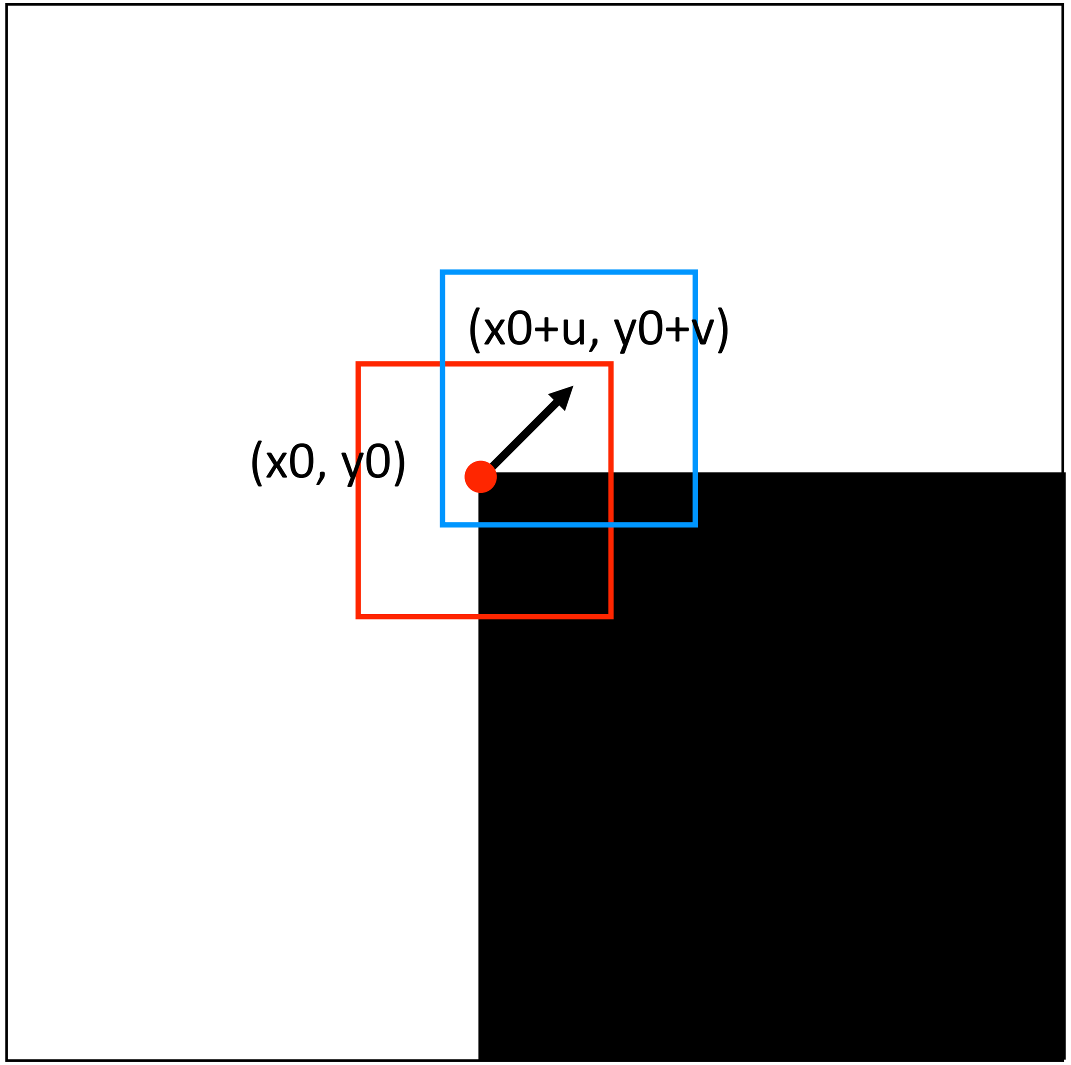
Original image

The Basic Idea of Harris Corner



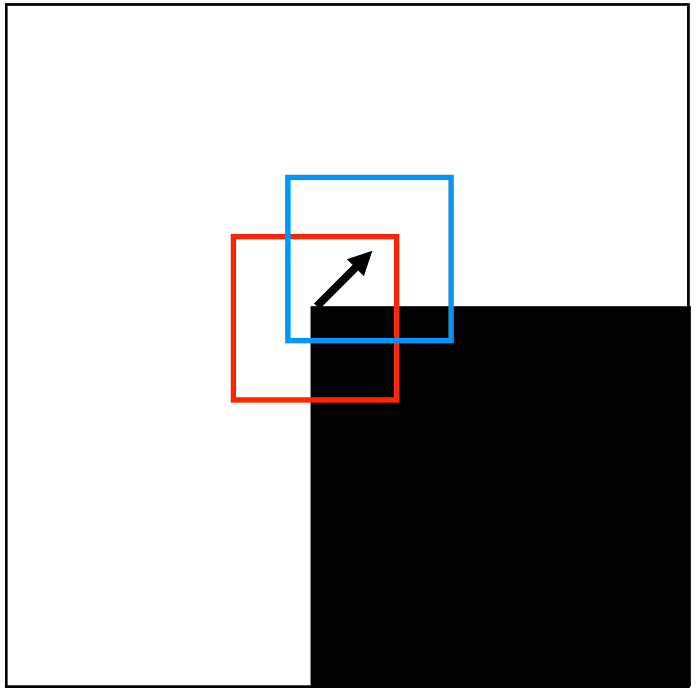
Local neighborhood of a corner point (x0, y0)

The Basic Idea of Harris Corner



Move along direction (u, v)

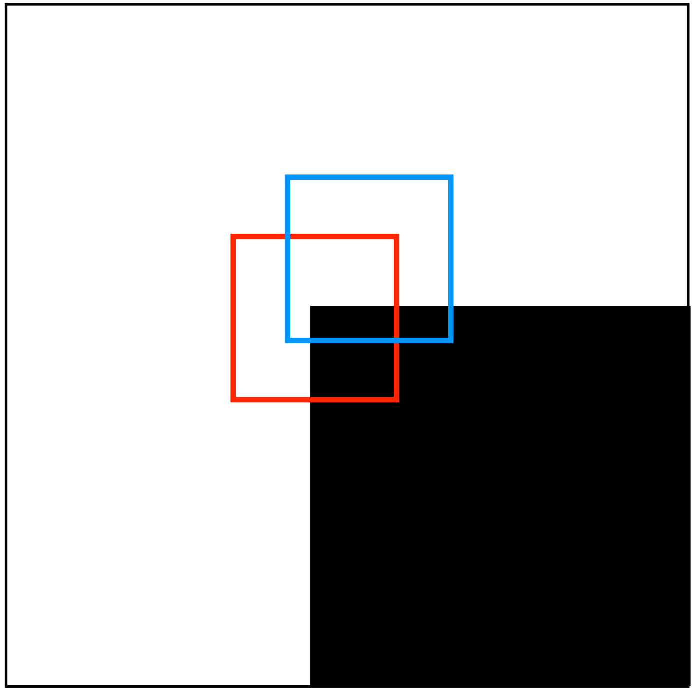
The Basic Idea of Harris Corner



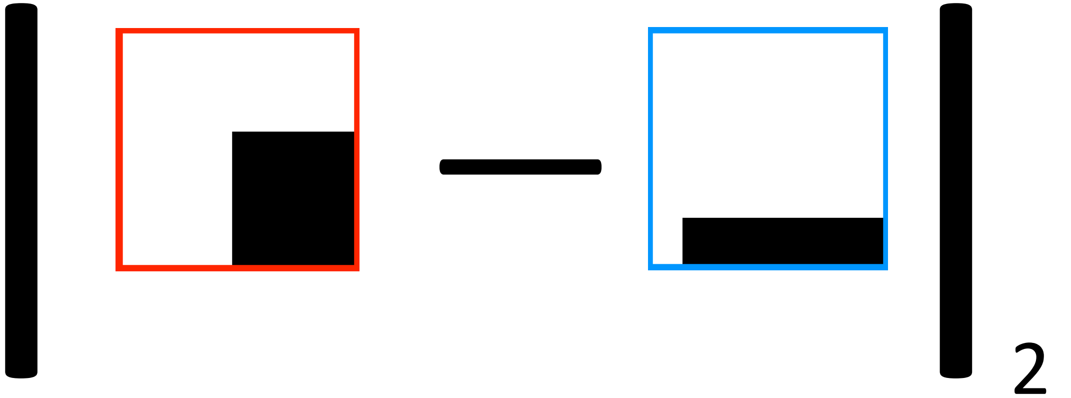
Local neighborhood of a corner point (x0, y0)

Local neighborhood of point (x0+u, y0+v)

The Basic Idea of Harris Corner



Change along direction (u, v) =

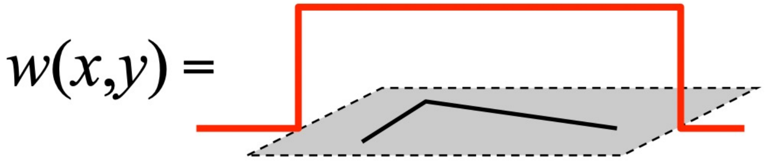


$$= \sum_{(x,y)\in N} [I(x+u, y+v) - I(x,y)]^2$$

Where N is the neighborhood of (x0, y0)

Notation

Rectangle Window Function



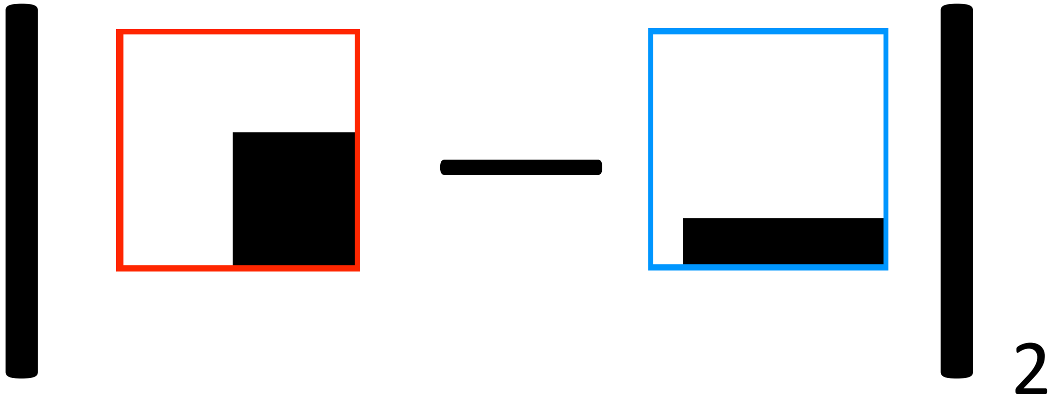
1 in window, 0 outside

$$w'(x,y) = w(x - x_0, y - y_0)$$

Square intensity difference

$$D(x,y) = [I(x+u,y+v) - I(x,y)$$

The Basic Idea of Harris Corner



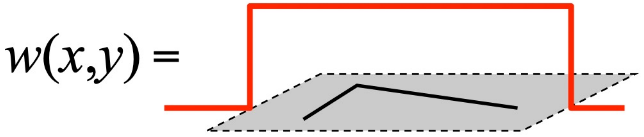
$$= \sum_{(x,y)\in N} [I(x+u,y+v) - I(x,y)]^2$$

$$= \sum_{x,y} w'(x,y)[I(x+u,y+v) - I(x,y)]^2$$

$$= \sum_{x,y} w'(x,y)D(x,y)$$

$$= w' * D$$

Rectangle Window Function



1 in window, 0 outside

$$w'(x,y) = w(x - x_0, y - y_0)$$

Square intensity difference

$$D(x,y) = [I(x+u,y+v) - I(x,y)]^2$$

Harris Detector

First-order Taylor expansion: $I[x+u, y+v] - I[x, y] \approx I_x u + I_y v$

$$\therefore D(x, y) = (I[x+u, y+v] - I[x, y])^2 \approx (I_x u + I_y v)^2 = [u, v] \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$\therefore E_{(x_0, y_0)}(u, v) = w' * D = [u, v]\ w' * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

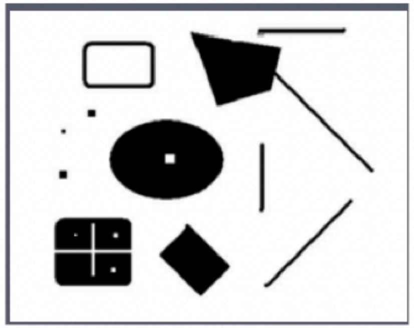
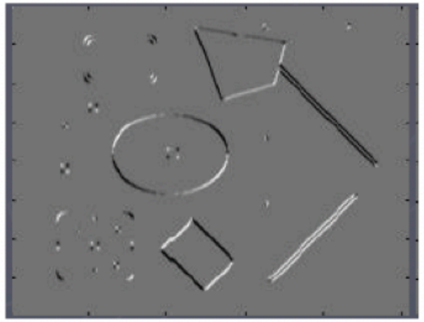


Image I

I_x

I_y



$I_x I_y$

Image borrowed from Stanford CS131

Harris Detector

If we are checking the corner at (x_0, y_0), then the change along direction (u0, v0) is:

$$E_{(x_0,y_0)}(u,v) \approx [u,v]\ M(x_0,y_0) \ \begin{bmatrix} u \\ v \end{bmatrix}$$

where $$M(x,y) = w' * \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} = \begin{bmatrix} w' * I_x^2 & w' * (I_xI_y) \\ w' * (I_xI_y) & w' * I_y^2 \end{bmatrix}$$

Image borrowed from Stanford CS131

Harris Detector

$$M(x, y) = w' * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} w' * I_x^2 & w' * (I_x I_y) \\ w' * (I_x I_y) & w' * I_y^2 \end{bmatrix}$$

- M is a symmetric matrix.
- M is a positive semi-definite matrix. (since all its principle minors ≥ 0.)
- Simple case: M is diagonal at (x_0, y_0) : $M(x_0, y_0) = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$ $(\lambda_1 \geq 0,\ \ \lambda_2 \geq 0)$

$$\therefore E_{(x_0, y_0)}(u, v) \approx [u, v]\ M(x_0, y_0) \begin{bmatrix} u \\ v \end{bmatrix} = \lambda_1 u^2 + \lambda_2 v^2$$

  - This corresponds to an axis-aligned corner.
  - If either $\lambda \approx 0$, this is not a corner.

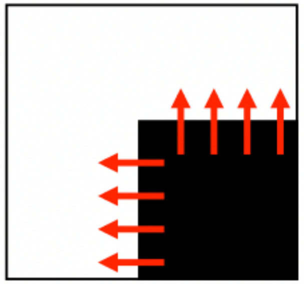


Image borrowed from Stanford CS131

Harris Detector

- General case:

  since M is a symmetric matrix, perform eigendecomposition:

$$M(x,y) = w' * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R \quad (\lambda_1 \geq 0,\ \ \lambda_2 \geq 0)$$

R is an orthogonal matrix, λs are the eigenvalues of M!

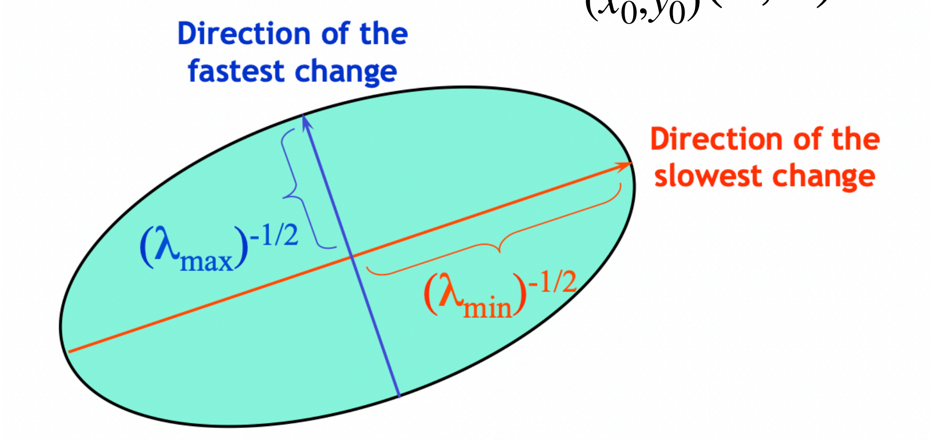
Harris Detector

- General case: since M is a symmetric matrix, perform eigen-decomposition:

$$M(x,y) = w' * \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R \qquad (\lambda_1 \geq 0,\ \ \lambda_2 \geq 0)$$

$$\therefore E_{(x_0,y_0)}(u,v) \approx \lambda_1 u_R^2 + \lambda_2 v_R^2$$

where $\begin{bmatrix} u_R \\ v_R \end{bmatrix} = R \begin{bmatrix} u \\ v \end{bmatrix}$

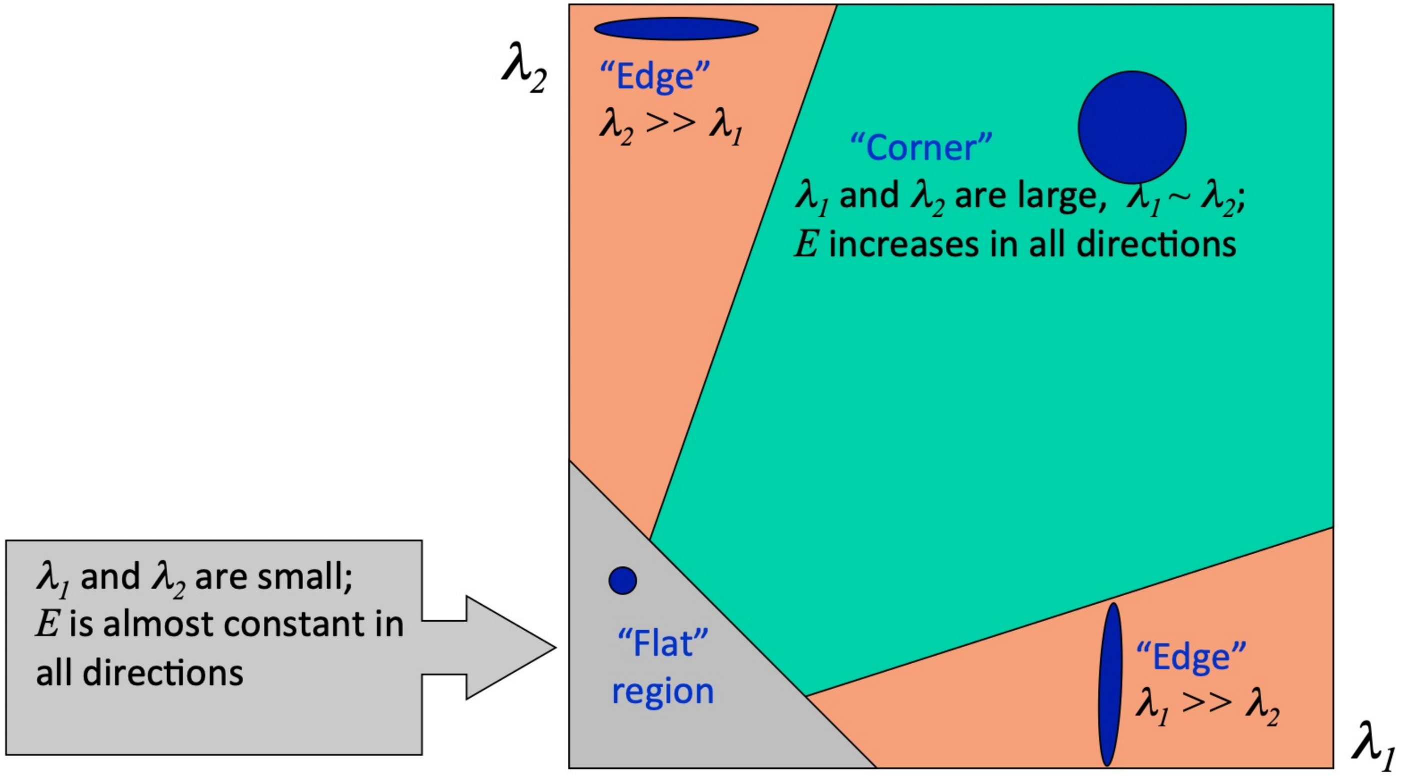


The energy landscape is a paraboloid!

Image borrowed from Stanford CS131

Eigenvalues

- Classification of the type of the image point according to the eigenvalues of M.



"Edge"
$\lambda_2 >> \lambda_1$

"Corner"
λ_1 and λ_2 are large, $\lambda_1 \sim \lambda_2$;
E increases in all directions

λ_1 and λ_2 are small;
E is almost constant in all directions

"Flat" region

"Edge"
$\lambda_1 >> \lambda_2$

Two conditions must be satisfied:

$$\lambda_1, \lambda_2 > b$$

$$\frac{1}{k} < \frac{\lambda_1}{\lambda_2} < k$$

Image borrowed from Stanford CS131

Corner Response Function θ

- Fast approximation:

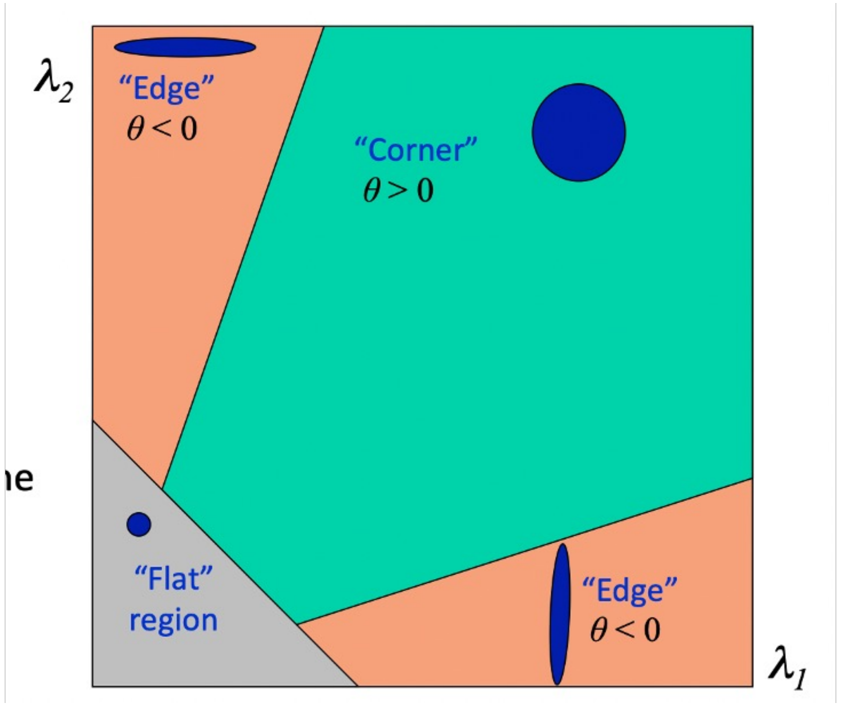
$$\theta = \frac{1}{2}\underline{(\lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2})) + \frac{1}{2}\underline{(\lambda_1\lambda_2} - 2t) \quad \alpha in[0.04,0.06]$$

$$\frac{1}{k} < \frac{\lambda_1}{\lambda_2} < k \qquad\qquad \lambda_1, \lambda_2 > b$$

$$= \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 - t$$

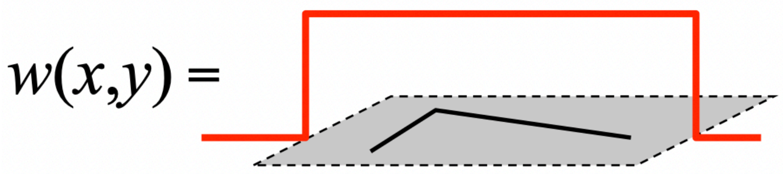
$$= \underline{det(M)} - \underline{\alpha Trace(M)}^2 - t$$

Orthogonal transformation won’t change the determinant and trace of a matrix



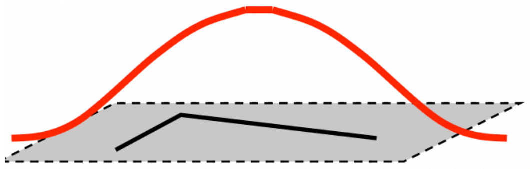
Choices of Window Functions

Rectangle window



1 in window, 0 outside

or



Gaussian

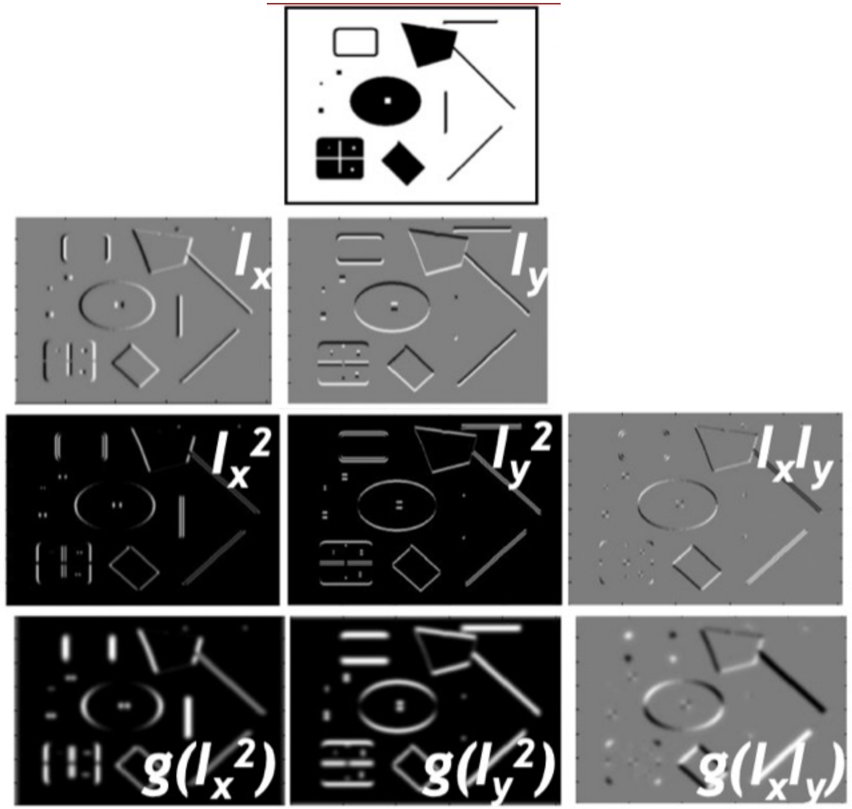
$$M(x,y) = w * \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}$$

$$M(x,y) = g(\sigma) * \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}$$

Not rotation-invariant.

Rotation-invariant.

Summary of Harris Detector



1. Image derivatives

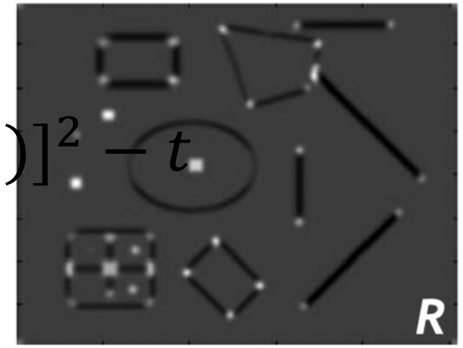
2. Square of derivatives

3. Rectangle window or Gaussian filter

4. Corner response function

$$\theta = g(I_x^2)g(I_y^2) - [g(I_xI_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2 - t$$

5. Non-maximum suppression



Step-by-Step Harris Detector

- Input: two images

Image borrowed from Stanford CS131

Step-by-Step Harris Detector

- Compute corner response θ

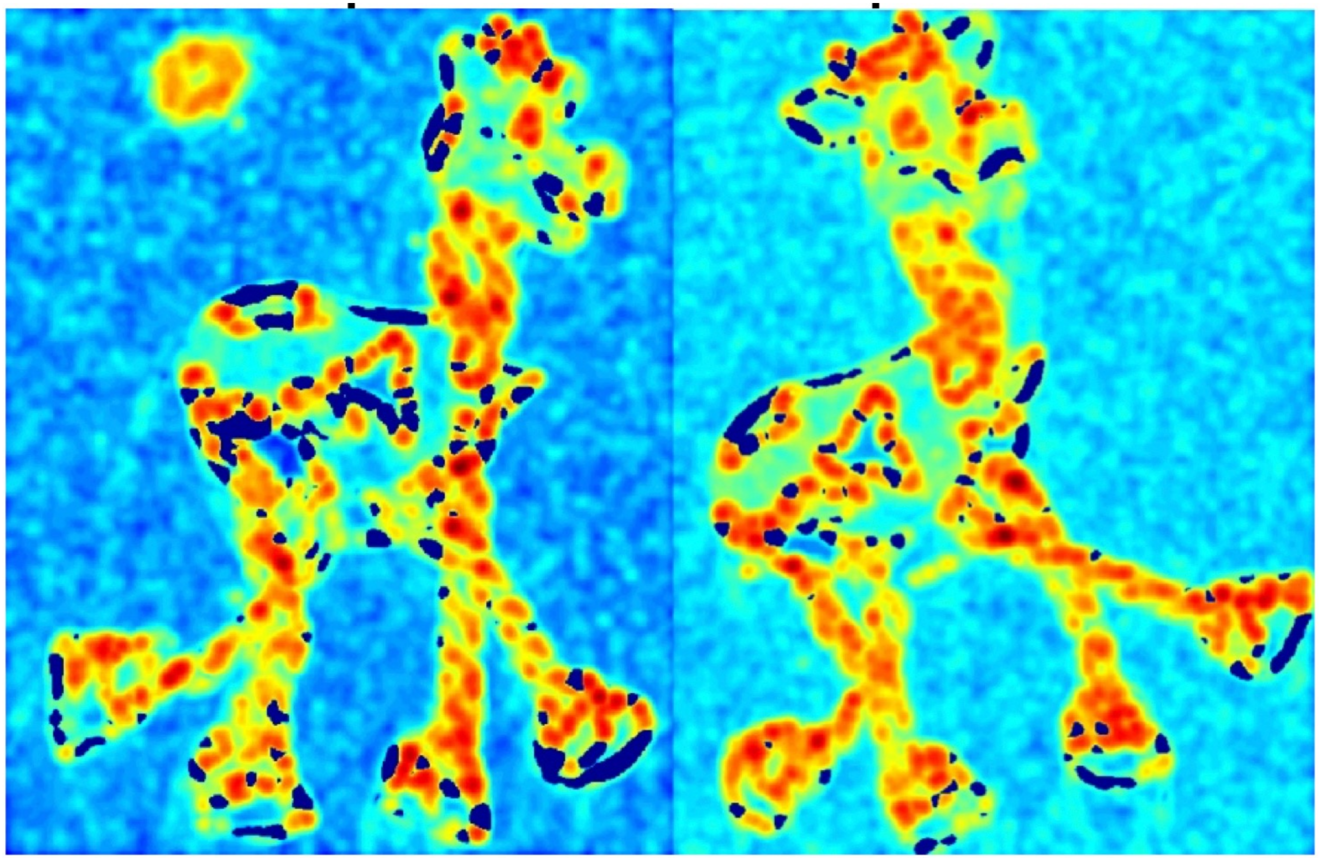


Image borrowed from Stanford CS131

Step-by-Step Harris Detector

- Thresholding and perform non-maximal suppression

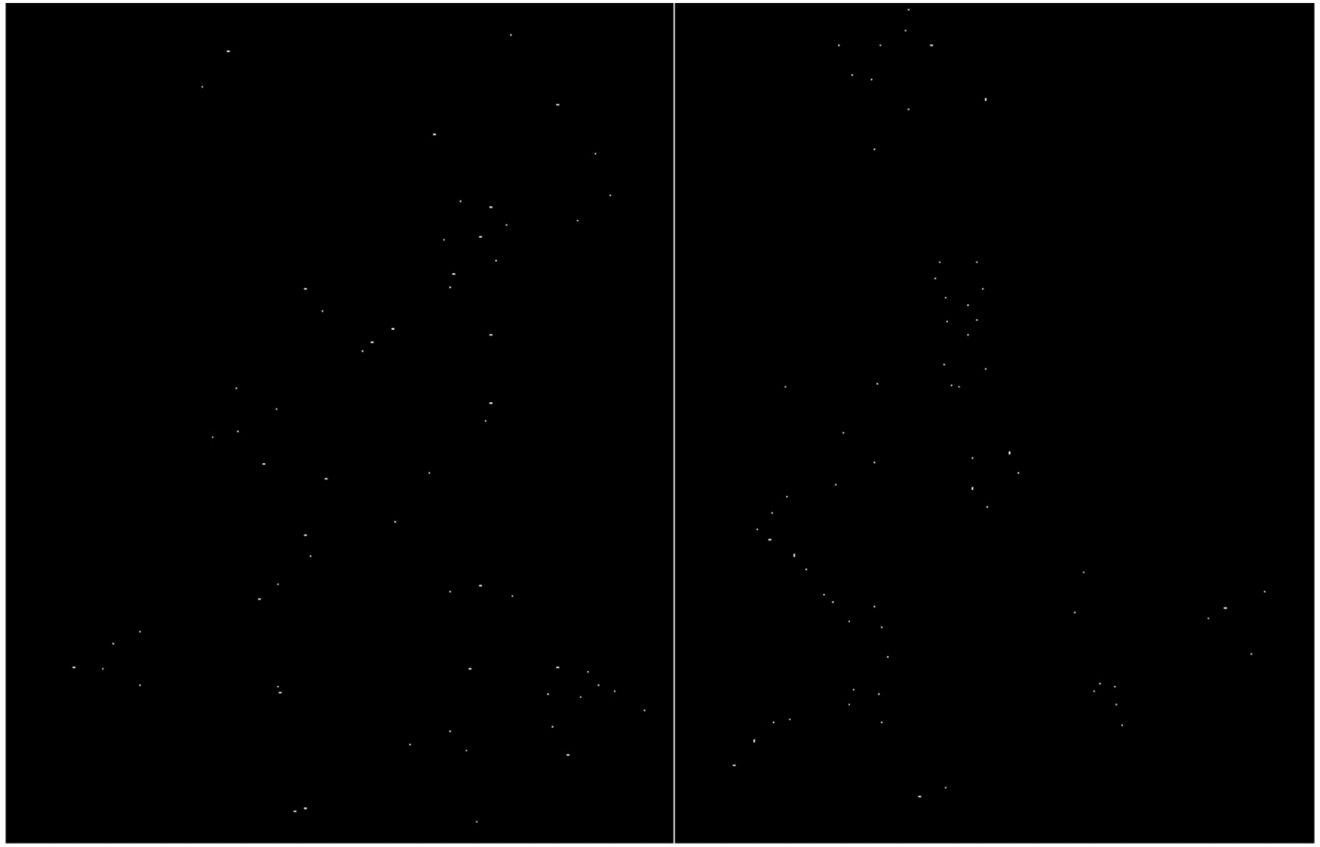


Image borrowed from Stanford CS131

Step-by-Step Harris Detector

- Results

Image borrowed from Stanford CS131

Properties of Harris Detector

- Corner response is equivariant with both translation and image rotation.

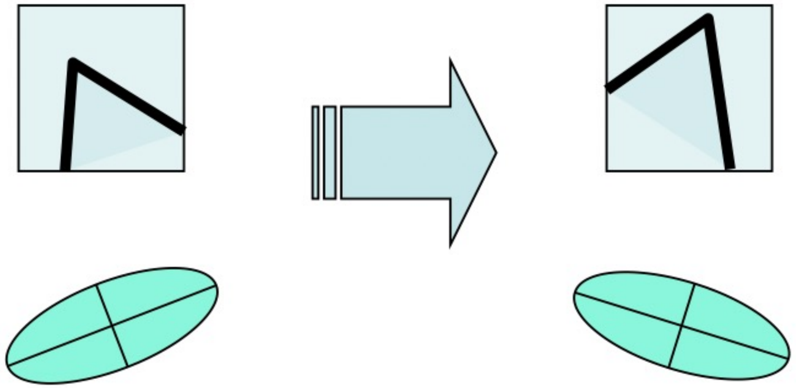
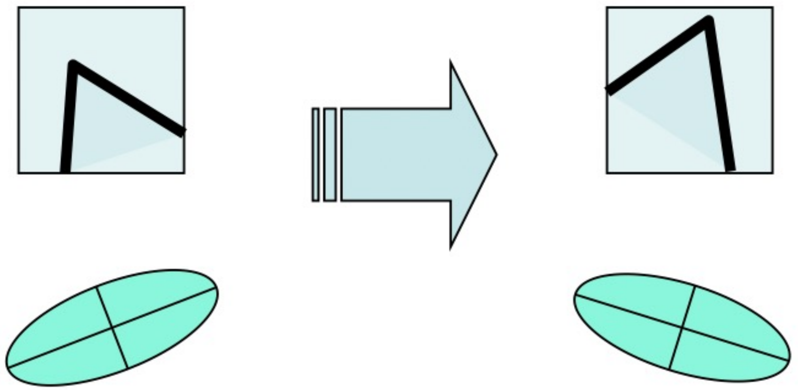


Image borrowed from Stanford CS131

Properties of Harris Detector

- Corner response is equivariant with both translation and image rotation.
- Not invariant to scale.



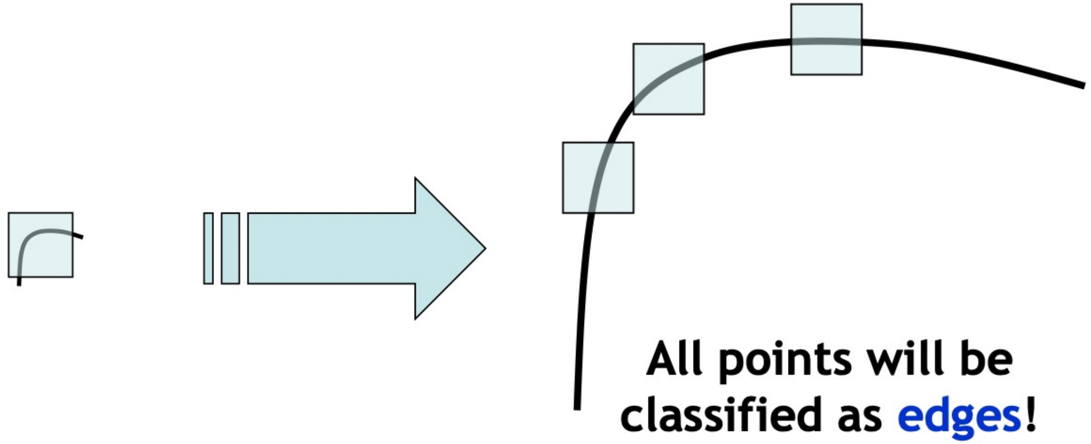
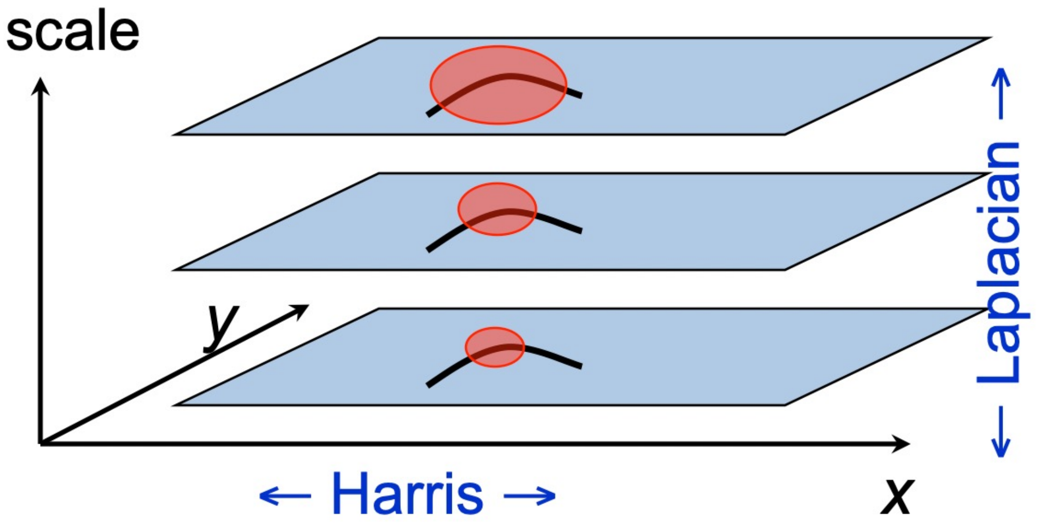


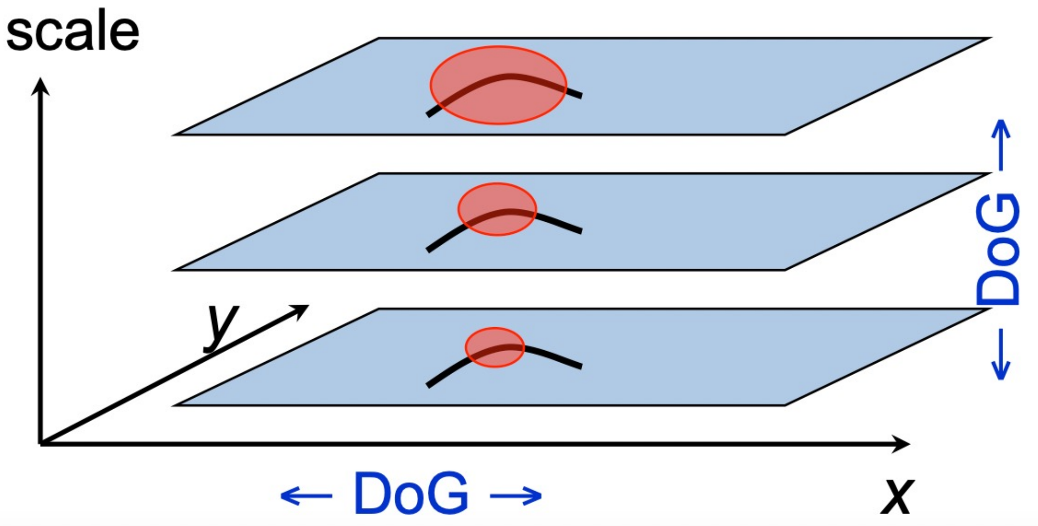
Image borrowed from Stanford CS131

Scale Invariant Detectors

- **Harris-Laplacian**[1]
  Find local maximum of:
  - Harris corner detector in space (image coordinates)
  - Laplacian in scale



- **SIFT (Lowe)**[2]
  Find local maximum of:
  - Difference of Gaussians in space and scale



Slides borrowed from Stanford CS131

Next week: Lecture 3,
Classic Vision Methods II