Computer Vision 1

HC3a

Local Features

Edges, Lines, Corners

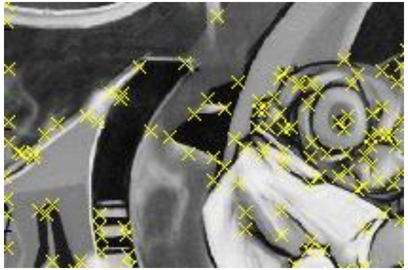
Dr. Martin Oswald, Dr. Dimitris Tzionas, Dr. Arun Mukundan,
(m.r.oswald, d.tzionas, a.mukundan)@uva.nl

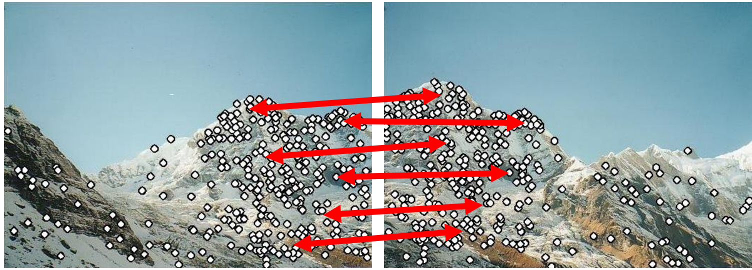
University of Amsterdam

computer vision lab

Outline

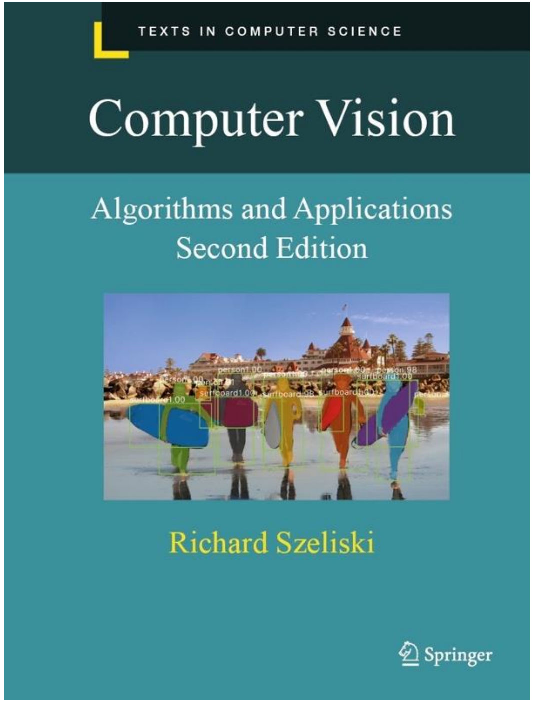
- Edge Detection
  - Derivatives of Image
  - Canny Edge Detector
- Line Fitting
  - Least Squares
  - RANSAC
  - Hough Transform
- Corners
  - Harris Corners
  - SIFT
  - Applications





Textbook

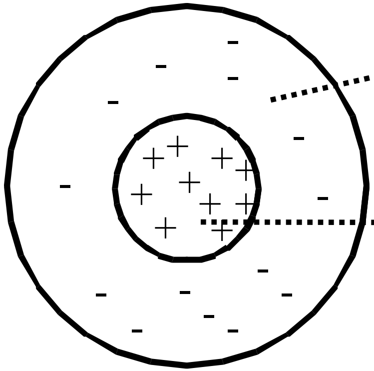
- Sec. 7.2
- Sec. 7.4
- Sec. 7.1



Outline

- Edge Detection
  - Derivatives of Image, Derivatives of Gaussian
  - Canny Edge Detector
- Line Fitting
  - Least Squares
  - RANSAC
  - Hough Transform
- Corners
  - Harris Corners
  - SIFT
  - Applications

Ganglion Cell – Week 1 Refresher



On-Center
Ganglion Cell

Inhibitory
(–) region

Excitatory
(+) region

When illuminated …

→ … **reduces** cell's output
(think: Contribute negatively)

→ … **increases** cell's output
(think: Contribute positively)

Total output:
Compute **Net Signal**

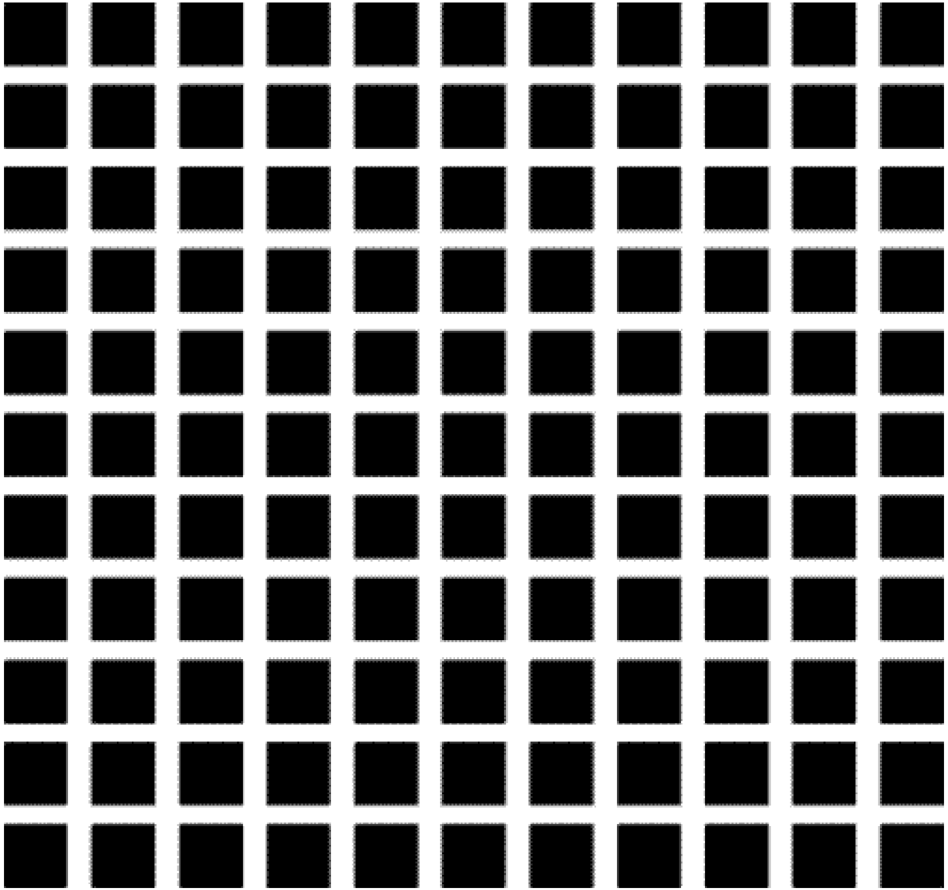
Ganglion Cells – Illusion – Hermann Grid

White grid on
black background

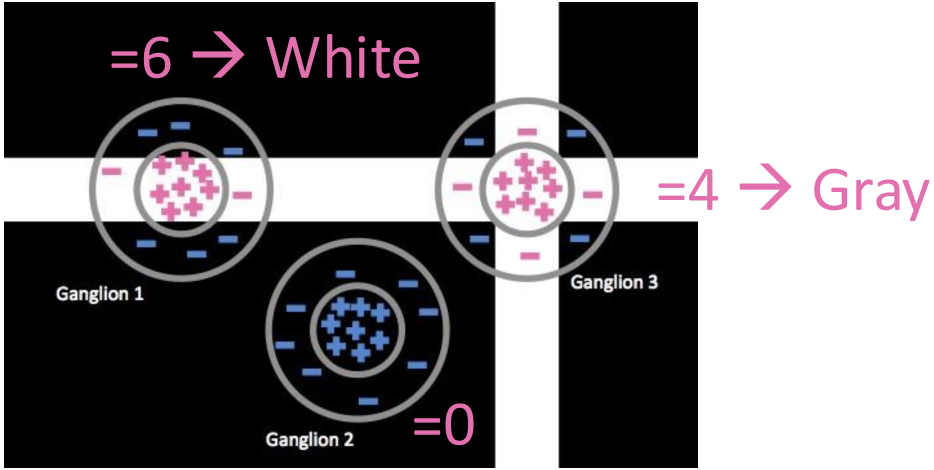
Illusion

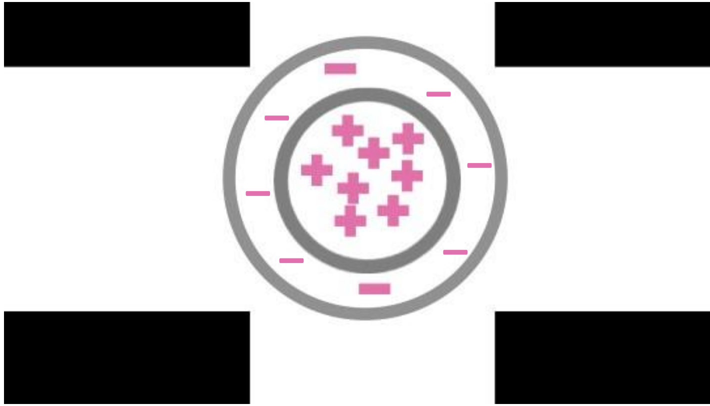
Faint grey spots @
intersections

in peripheral areas!



Ganglion Cells – Illusion – Hermann Grid





Illusion – Grey spots @ peripheral intersections

Pink → Inputs **stimulated** by light
Blue → Inputs that **not stimulated**

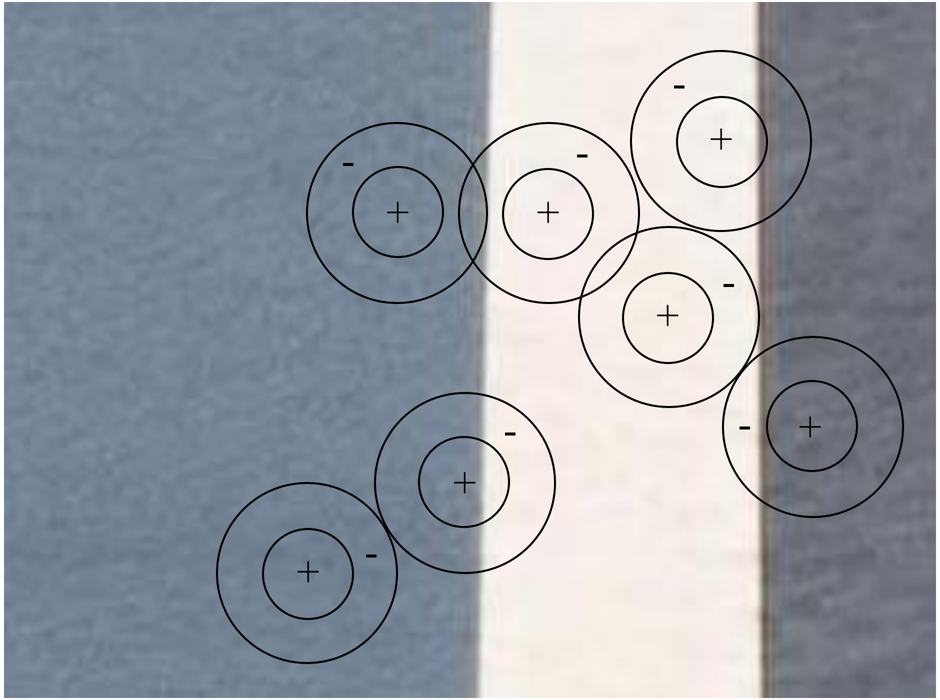
Compute **net** signal between activated + and –

Illusion disappears for intersection we directly look at!

Fovea → Receptive fields smaller
→ Cells @ intersection no longer surrounded by inhibitory components of grid

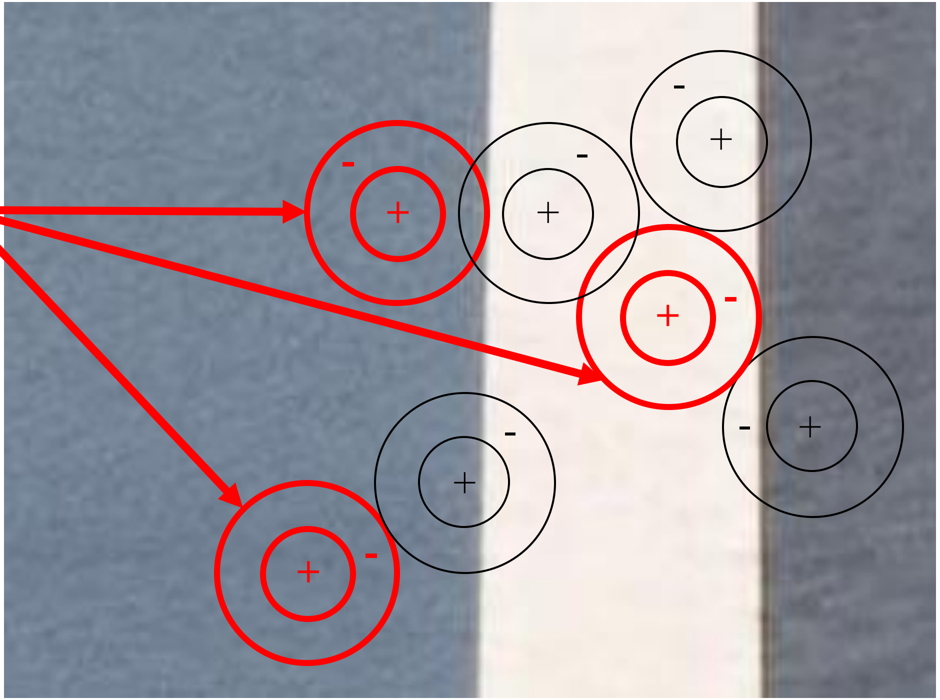
https://www.rosslab.neurobio.pitt.edu/making-sense-of-the-hermann-grid-illusion

Ganglion Cells – Responses

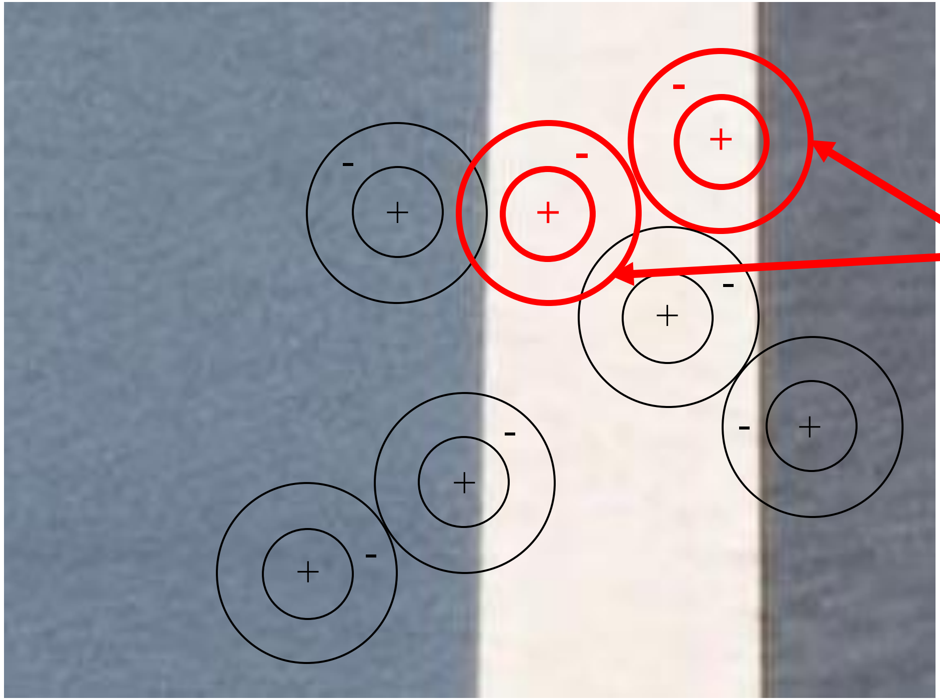


Ganglion Cells – Responses

Illuminated: Uniform
Response: Low

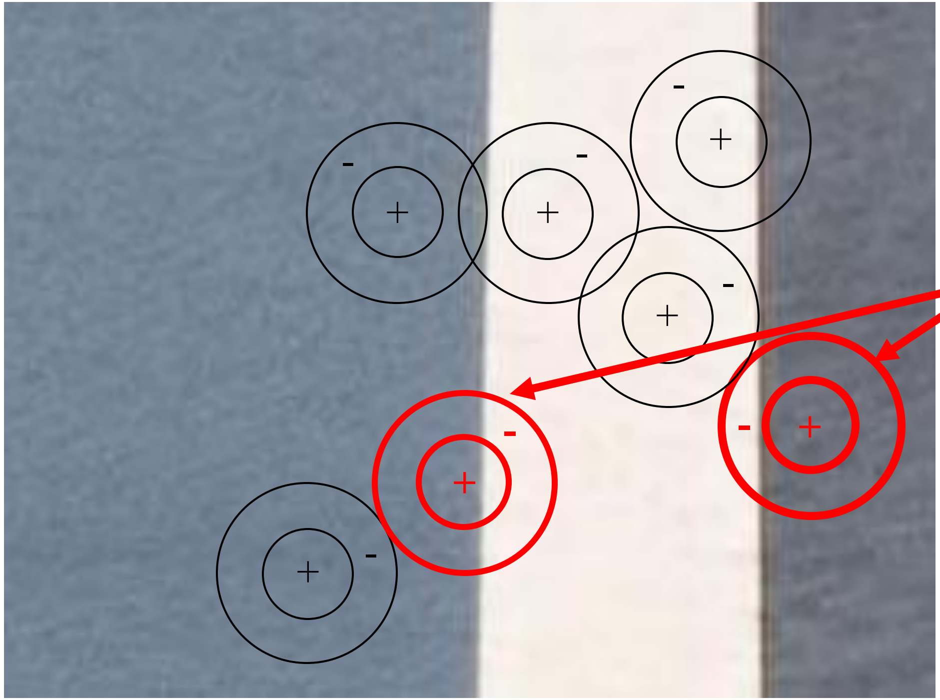


Ganglion Cells – Responses



+ Illuminated: Full
– Illuminated: Partial
Response: High

Ganglion Cells – Responses



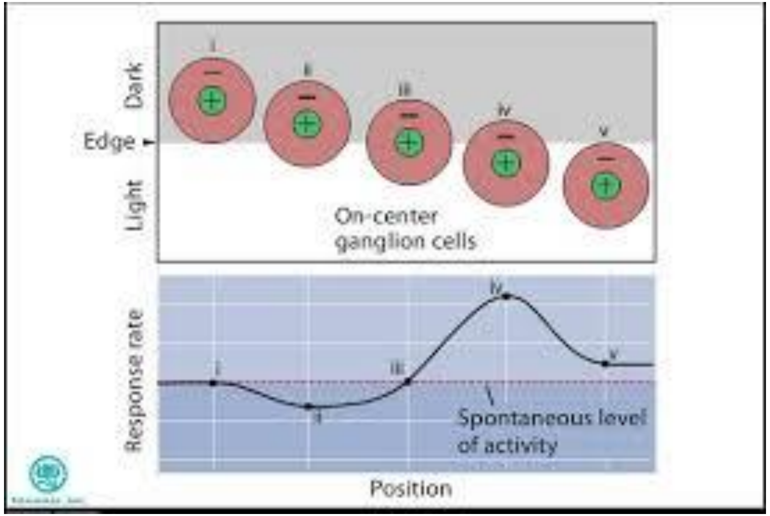
+ Illuminated: ~None

– Illuminated: Partial

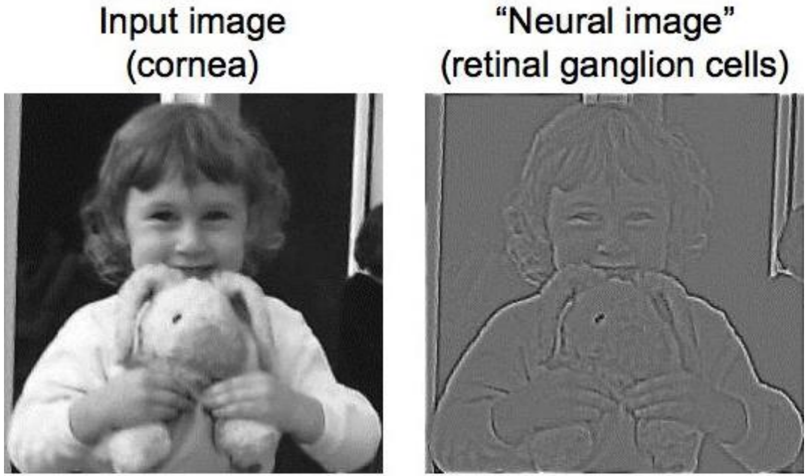
Response: Low

(maybe negative)

Ganglion Cell – Responses

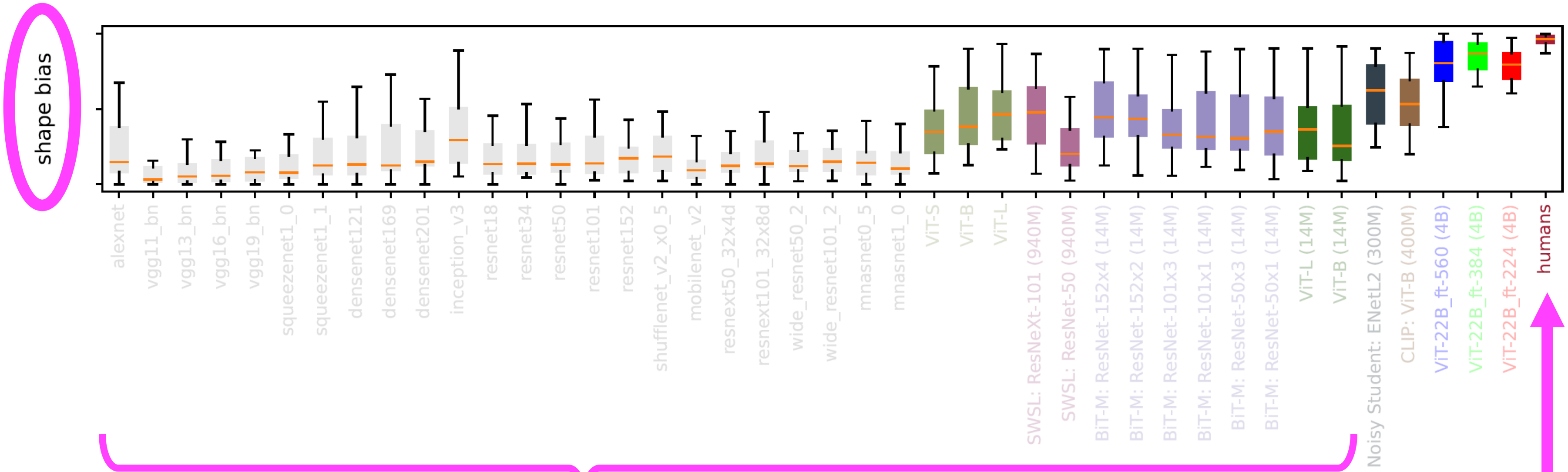


Ganglion Cells – Key Take-Away



Ganglion cells respond to **edges**

Human Perception – Bias towards Shape



Most **Computer Vision** models:

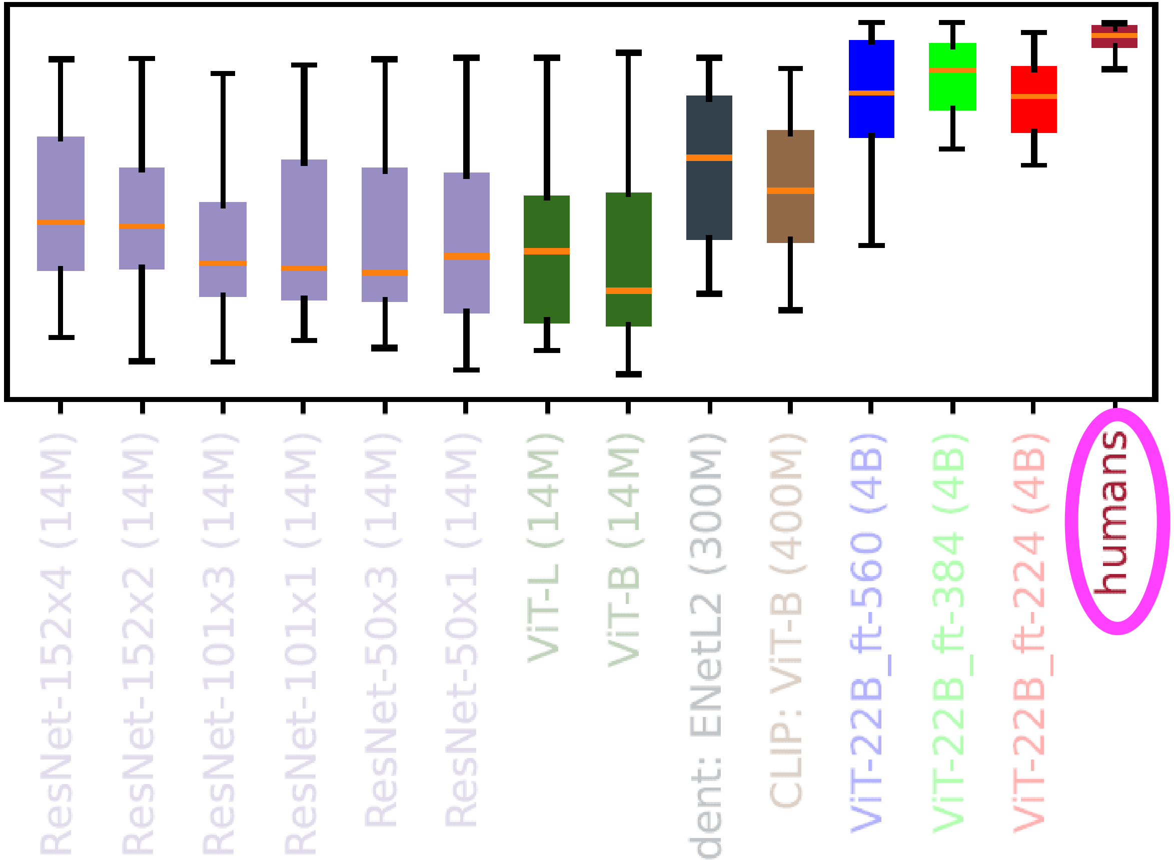
- High bias towards **Texture**
- Low bias towards **Shape**

Human perception is biased towards **Shape**

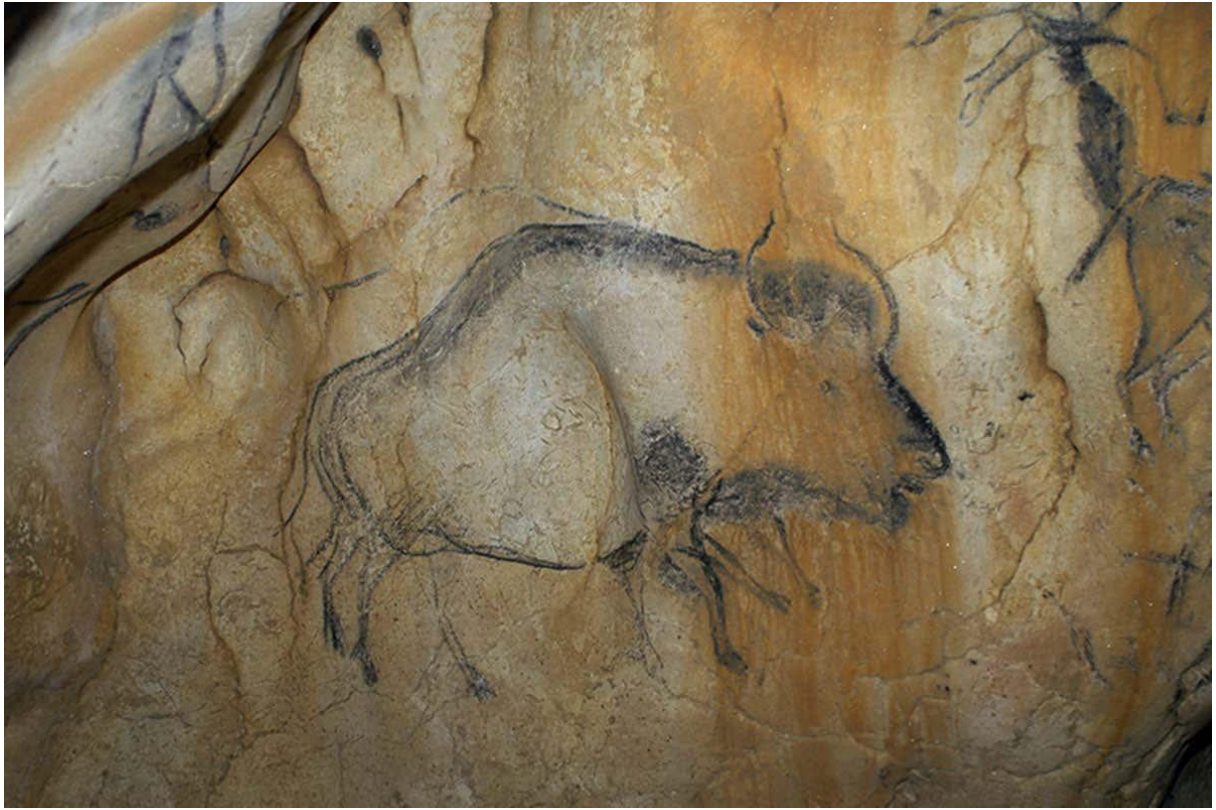
Dehghani et al, '*Scaling Vision Transformers to 22 Billion Parameters*', ICML 2023

Human Perception – Bias towards Shape

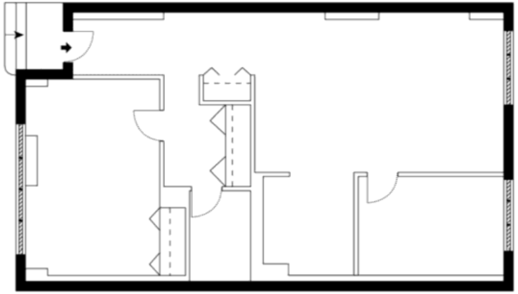
Big 'gap' between
- **Humans**
- **Computer Vision**

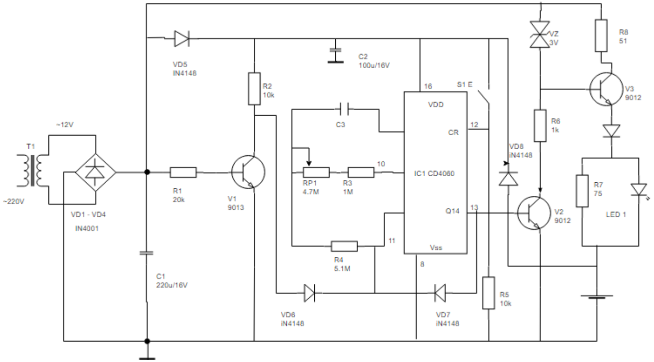


Early Edge Drawings



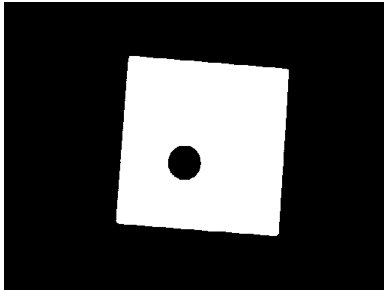
Typical Edge Drawings



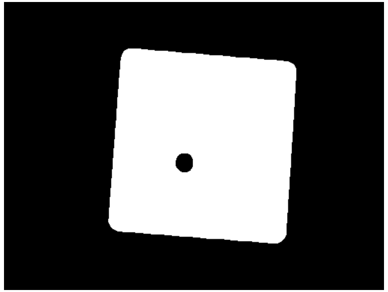


Edge Detection – Naive Way

1. **Dilate** input image
2. **Subtract** the input image *from* the dilated one
3. **Edges** arise!

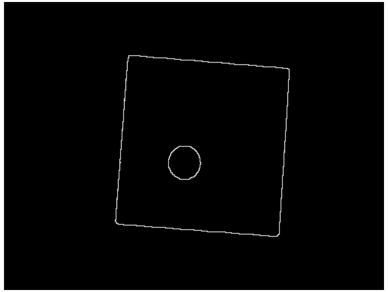


Input Image



Dilated Image

(exaggerated for visualization purposes)

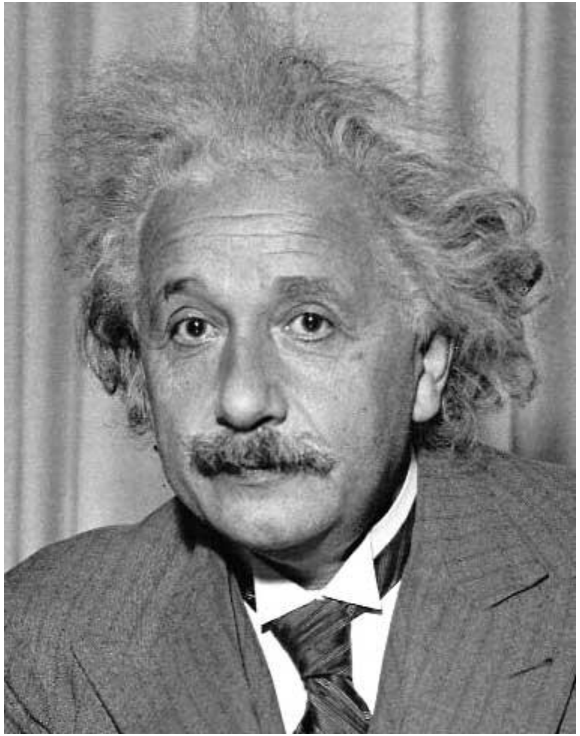


Edges

image ‘arithmetics’:

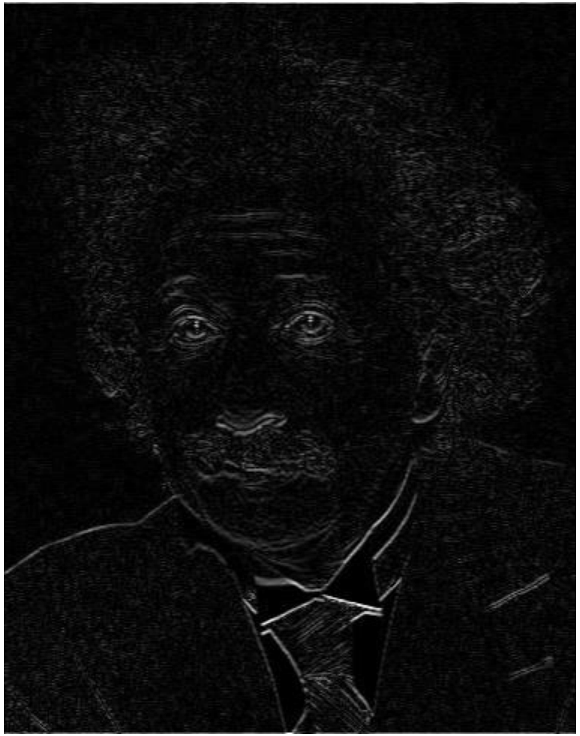
`Edges = dilated - input_img`

Derivative Filters



1	2	1
0	0	0
-1	-2	-1

Sobel



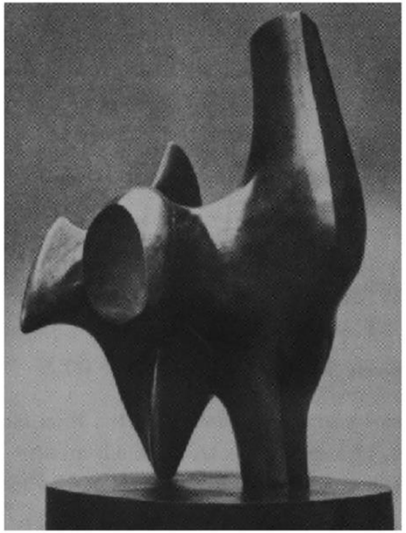
Horizontal Edges

(absolute value)

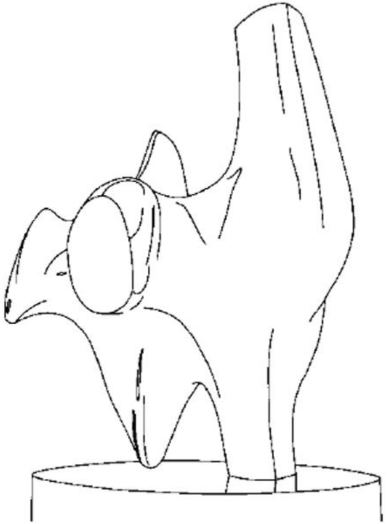
Outline

- Edge Detection
  - Derivatives of Image, Derivatives of Gaussian
  - Canny Edge Detector
- Line Fitting
  - Least Squares
  - RANSAC
  - Hough Transform
- Corners
  - Harris Corners
  - SIFT
  - Applications

Edge Detection



Input



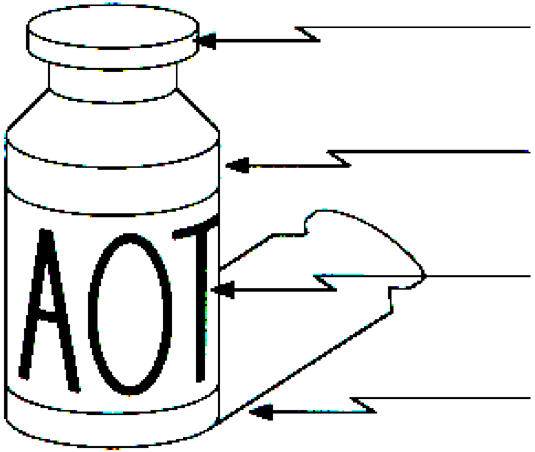
Output

Convert **2D image** → **Set of curves**

- Extracts salient features of the scene
- More compact than pixels

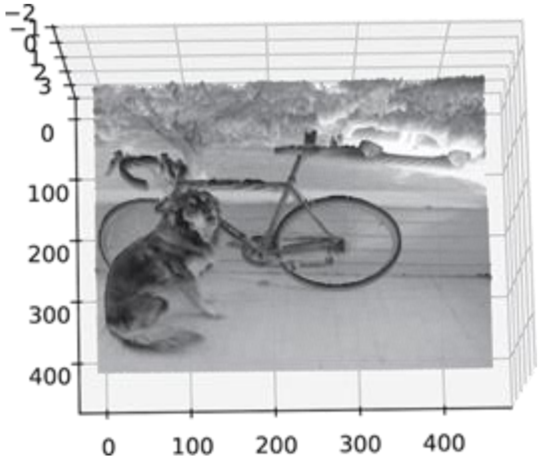
Edges – Origin

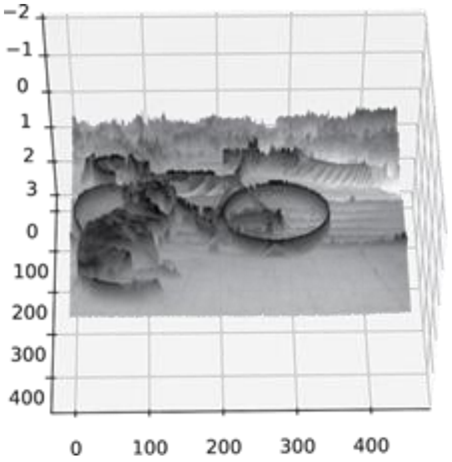
Caused by various factors:

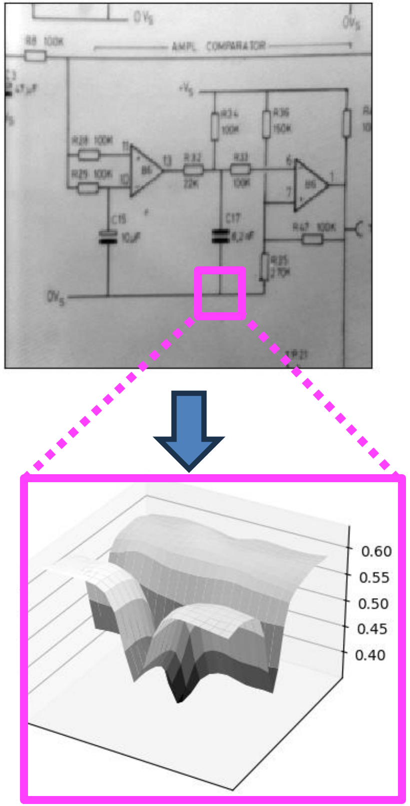


What is an Edge?

- Image is a function
- Think of the gray tones as HEIGHTS
- **Edges** are **rapid changes** – look like **steep cliffs**

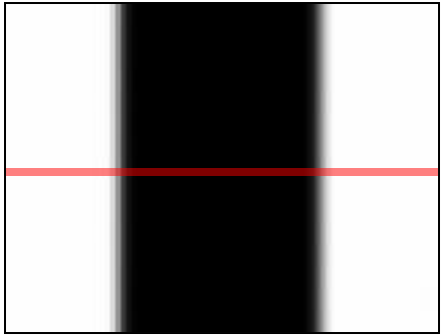




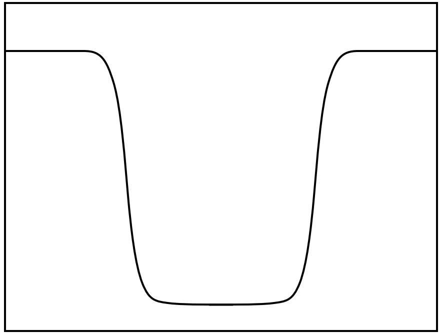


Edge Detection – 1D Function

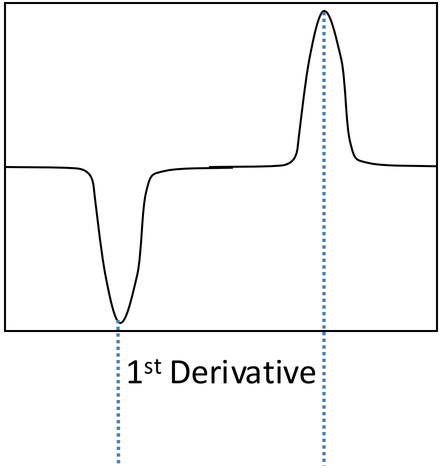
Edge → A place of **rapid change** in image intensity function



Image



Intensity Function
(along horizontal scanline)



1st Derivative

Edges correspond to
Extrema of Derivative

Slide by:
L. Lazebnik

Image Derivatives

How can we differentiate a *digital* image F[x,y]?

- **Option 1 ☹ :**
  - First, reconstruct a continuous image, f
  - Then, compute the derivative
- **Option 2 ☺ :**
  - Take **discrete derivative** (finite difference):

$$\frac{\partial f}{\partial x}[x, y] \approx F[x+1, y] - F[x, y]$$

How would you implement this as a **linear filter**?

$\frac{\partial f}{\partial x}$:

H_x

$\frac{\partial f}{\partial y}$:

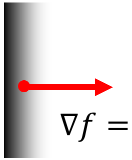
H_y

Slide by
S. Seitz

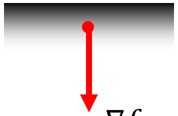
Image Gradient

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

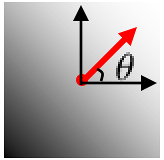
Gradient → Points in direction of most rapid increase in image intensity



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



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



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

Edge strength → **Gradient magnitude**:

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

Gradient direction:

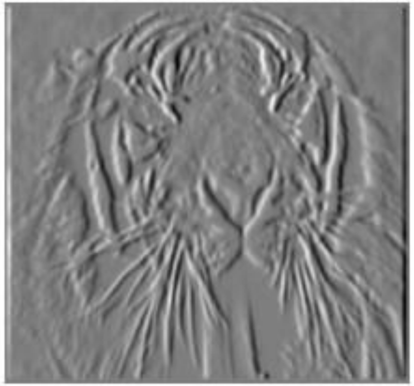
$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$

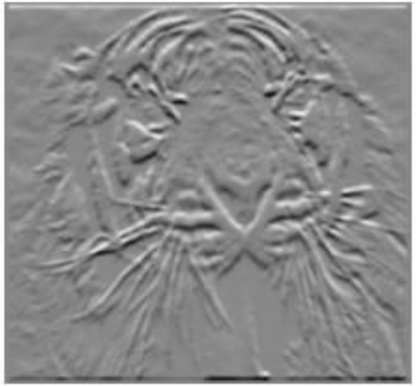
Image Gradients

Image

Gradient Magnitude

X-direction Gradients

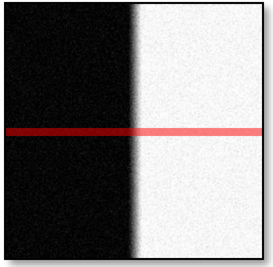




Y-direction Gradients

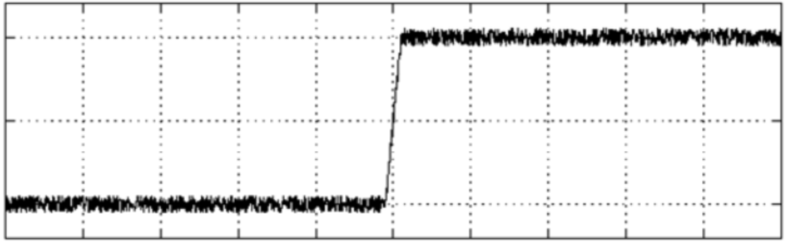
Slide by:
L. Lazebnik

Effect of Noise



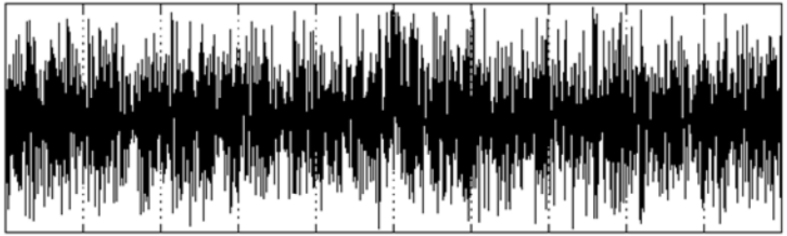
Noisy image

$f(x)$



Where is the edge?

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



Derivation amplifies noise
(like a high-pass filter)

Solution – Pre-smooth Image

Noisy image

f

Smoothing w. Gaussian kernel

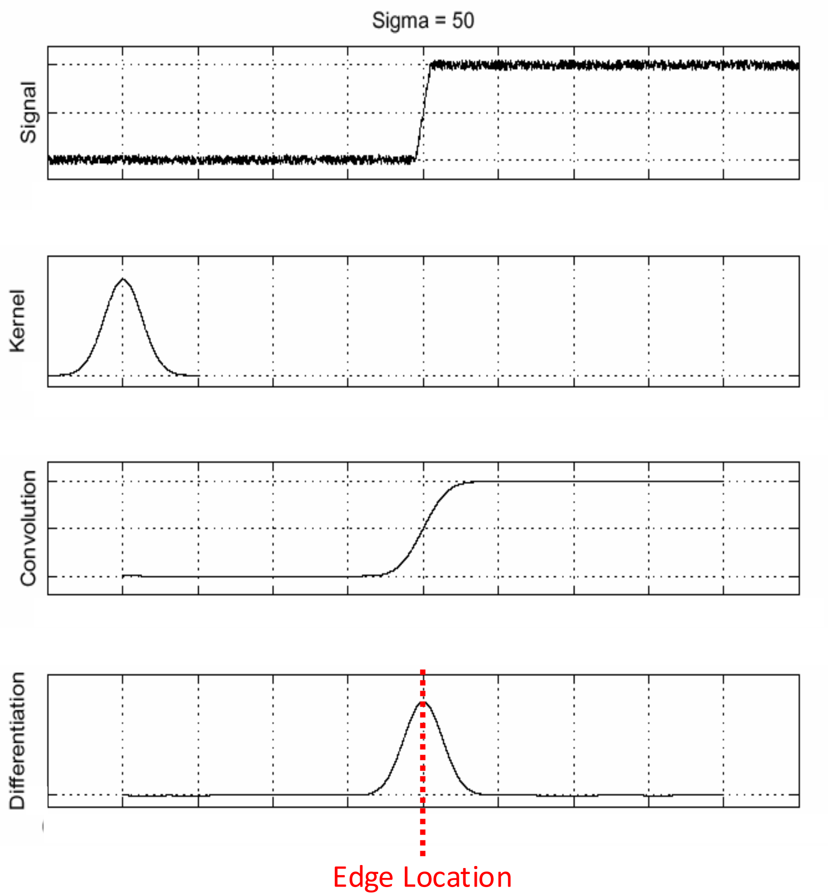
h

Convolve to smoothen

$f * h$

Differentiate & find extrema

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



To find **edges**, find **peaks** in $\frac{d}{dx}(f * h)$

But these are 3 steps 😢

And we must differentiate every time 😢

Adapted slide of S. Seitz

Pre-smooth & Pre-Differentiate

Noisy image

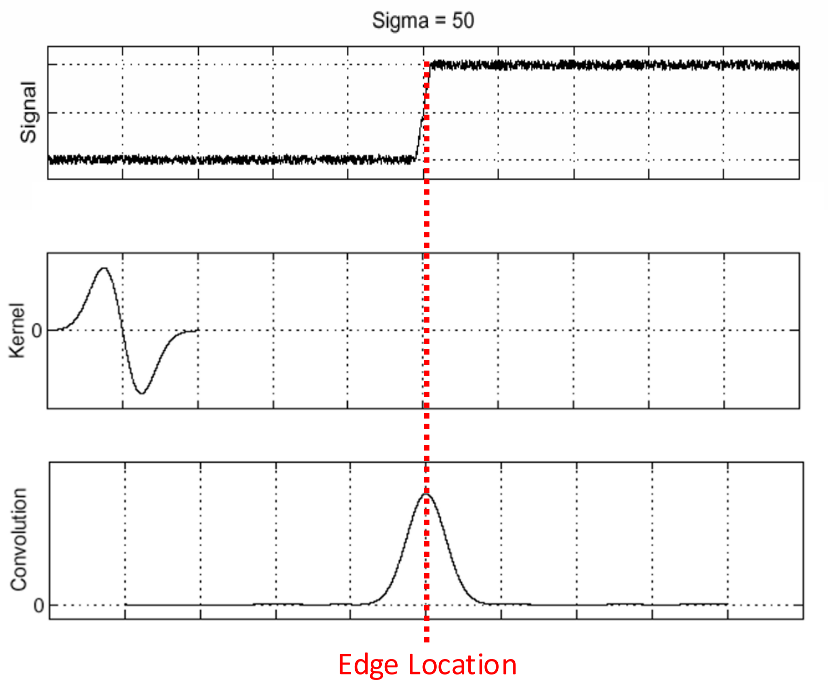
Differentiate kernel **offline** (only once)

Online convolution h pre-diff kernel

f

$\frac{dh}{dx}$

$f * \frac{dh}{dx}$



Convolution is Associative! $\frac{d}{dx}(f * h) = f * \frac{dh}{dx}$

To find **edges**, find **peaks** in $f * \frac{dh}{dx}$

1 step less online!
Remaining step is cheaper!

Differentiating Gaussian kernel h:

- Is exact – Compute first analytically, and later discretize
- Do it offline, only once

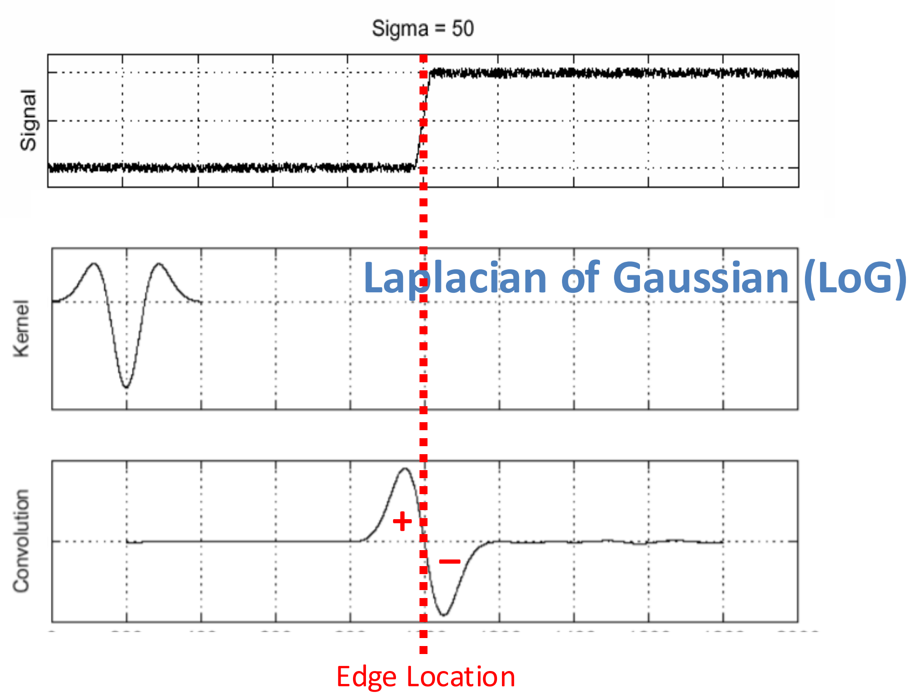
Extra Robustness

Edge as **zero-crossing**
of 2nd order derivative using
Laplacian of Gaussian (LoG)

f

$\frac{d^2}{dx^2}h$

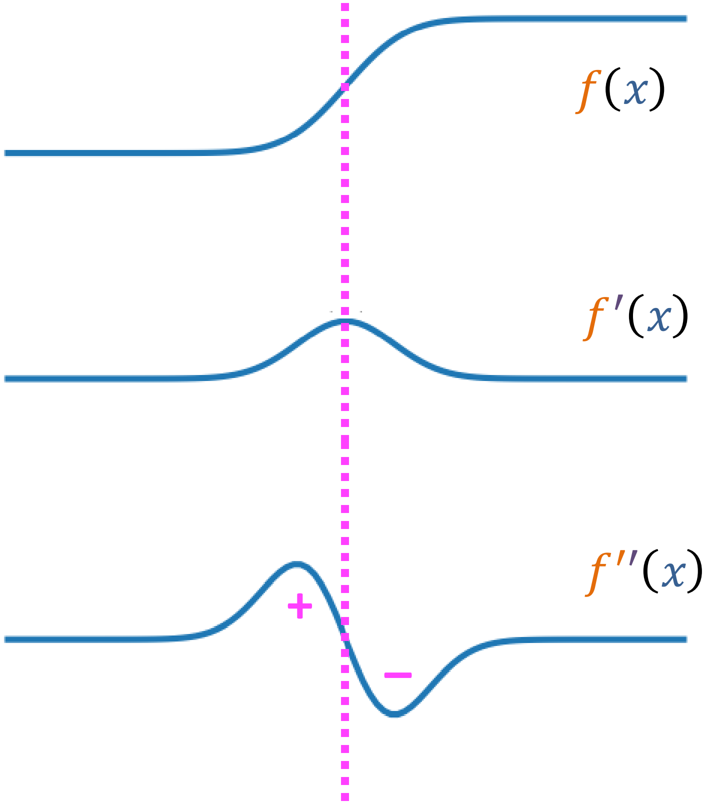
$f * \left(\frac{d^2}{dx^2}h\right)$



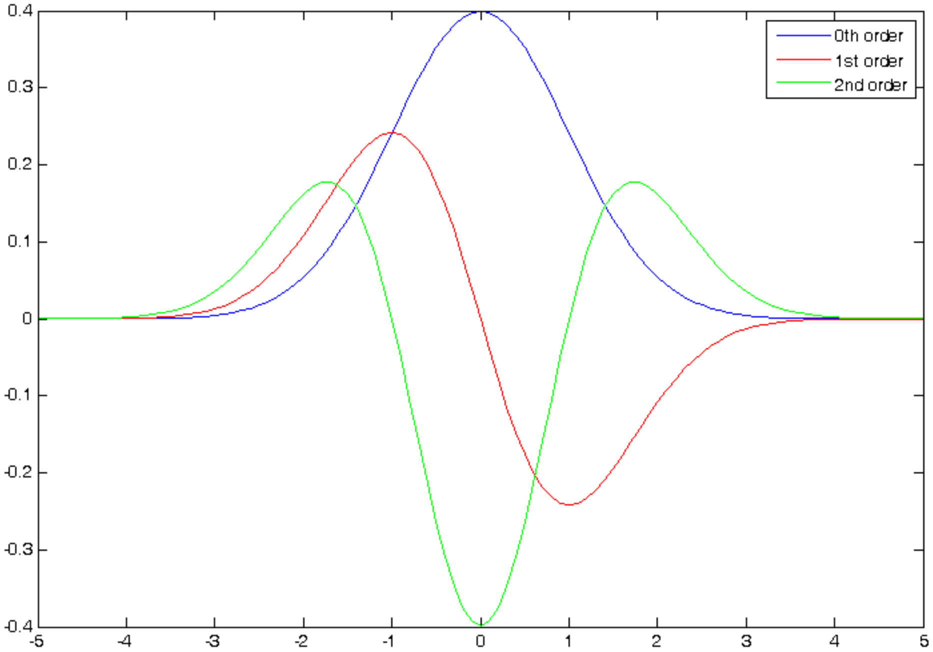
Extra Robustness

Extra robustness → Add a second condition!

- Ideal case
  - ***1st order*** derivative: local extremum $|f'(x)| \gg 0$
  - ***2nd order*** derivative is 0 $f''(x) \approx 0$
- Discrete signal in practice:
  - ***1st order*** derivative: local extremum
  - ***2nd order*** ***crosses 0*** → consider left/right samples!



1D Gaussian & Derivatives



$$h_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

$$h'_\sigma(x) = \frac{d}{dx} h_\sigma(x) = -\frac{x}{\sigma^2} h_\sigma(x)$$

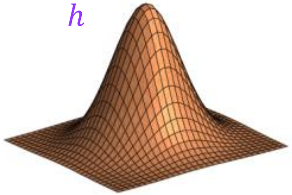
$$h''_\sigma(x) = \frac{d^2}{dx^2} h_\sigma(x) = \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right) h_\sigma(x)$$

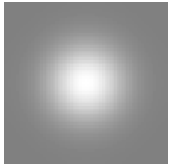
2D Gaussian & Derivatives

Height along ‘z’ re-encoded as bright/dark value

Gaussian kernel

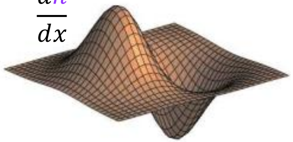
h



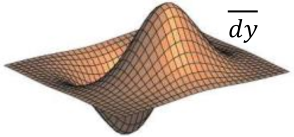


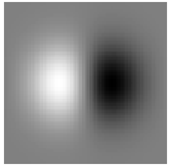
1st Order Derivatives

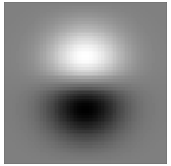
$\frac{dh}{dx}$



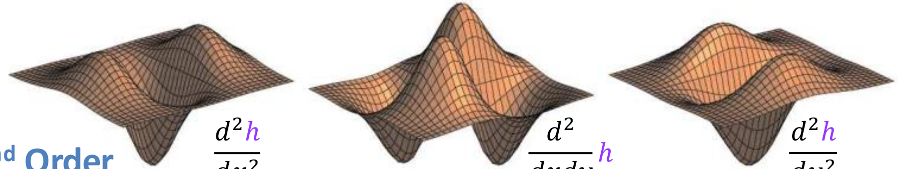
$\frac{dh}{dy}$



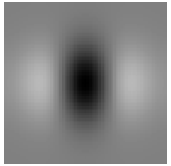


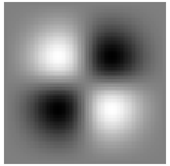


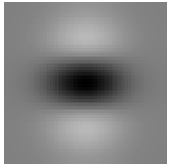
2nd Order Derivatives



$\frac{d^2h}{dx^2}$ $\frac{d^2}{dxdy}h$ $\frac{d^2h}{dy^2}$





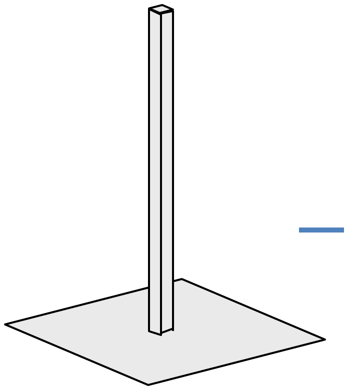


Laplacian

$$\nabla^2 h = \frac{d^2h}{dx^2} + \frac{d^2h}{dy^2}$$

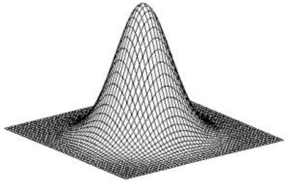
Difference of Gaussians (DoG)

- **Laplacian of Gaussian (LoG)** can be **approximated** as **Difference of Gaussians (DoG)**
- **Spoiler**: Used later today for **SIFT** keypoints



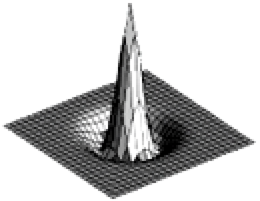
A narrow Gaussian
(or an impulse function)

−

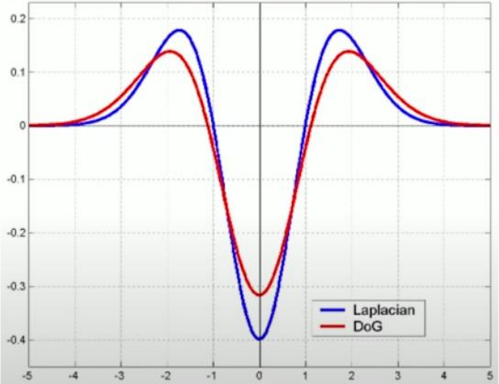


A wide Gaussian

≈

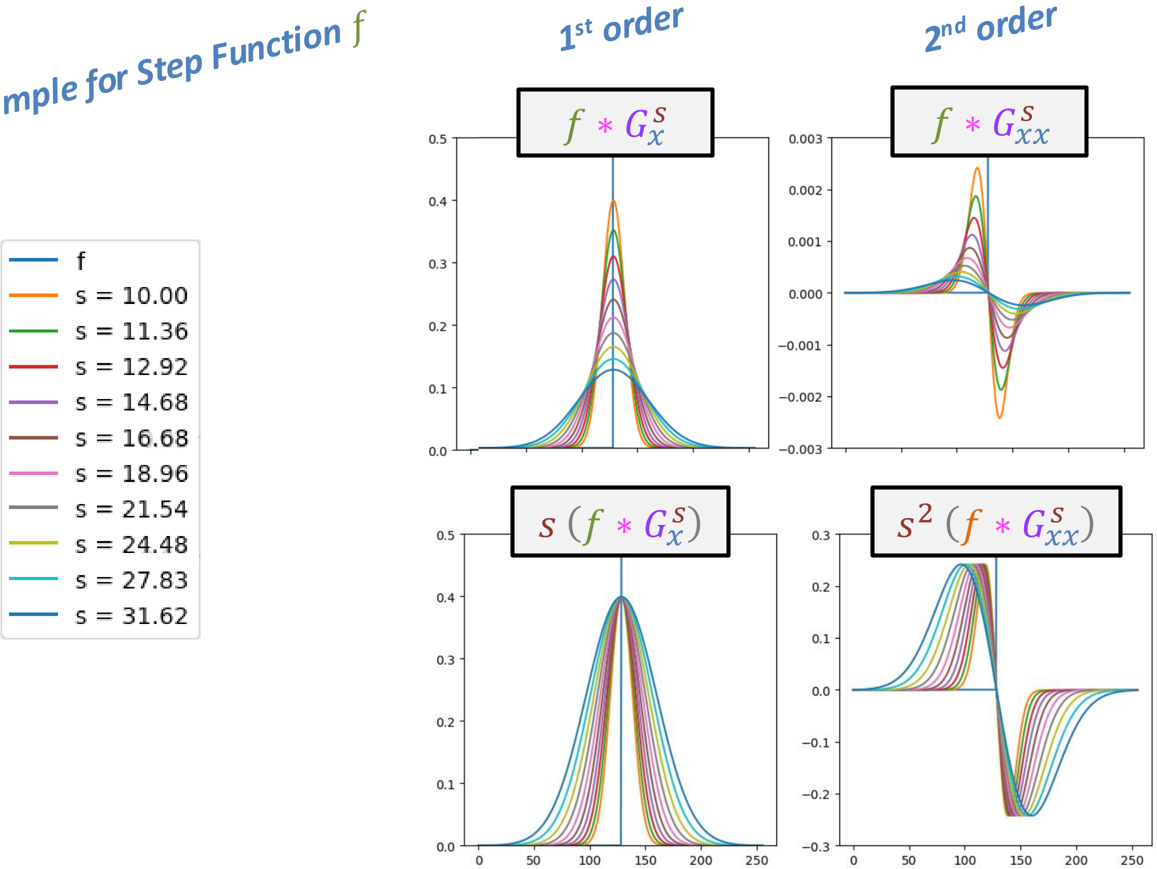


Laplacian of Gaussian (LoG)



Gaussian Derivatives & Scale

Example for Step Function f



Amplitude decreases as scale increases

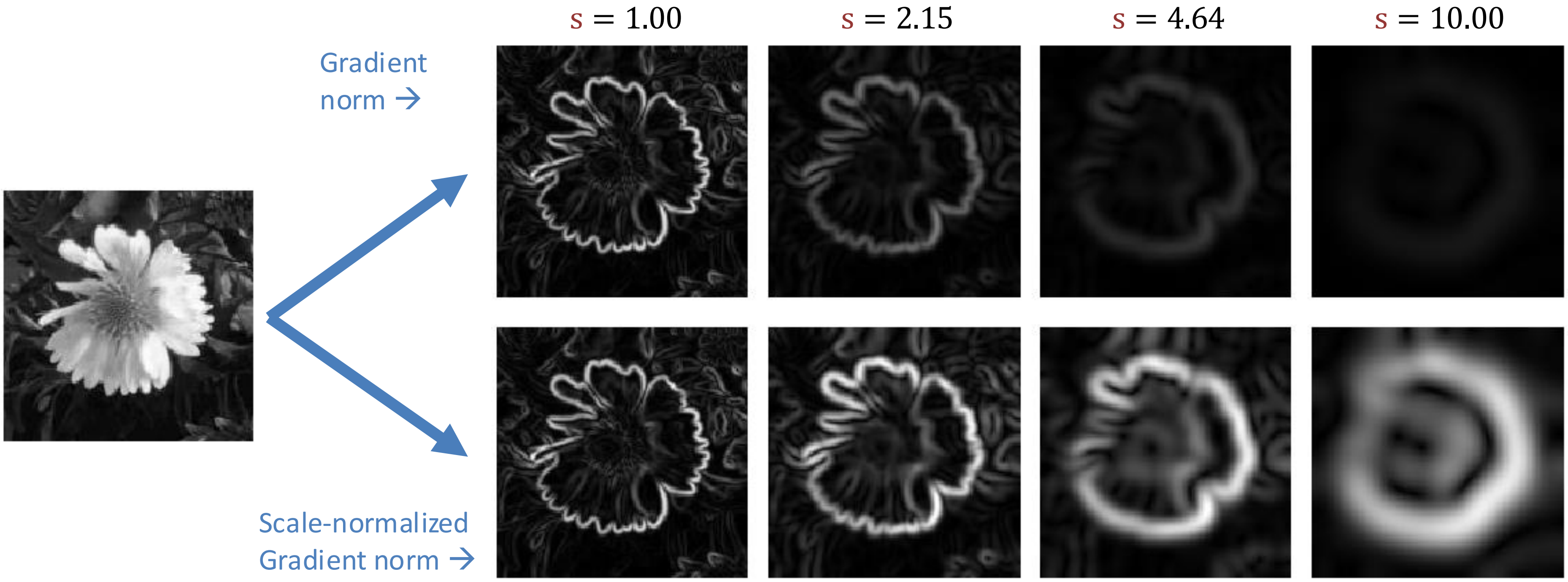
Comparing derivatives across scales → problem!

Scale-normalized Derivatives

k-th order

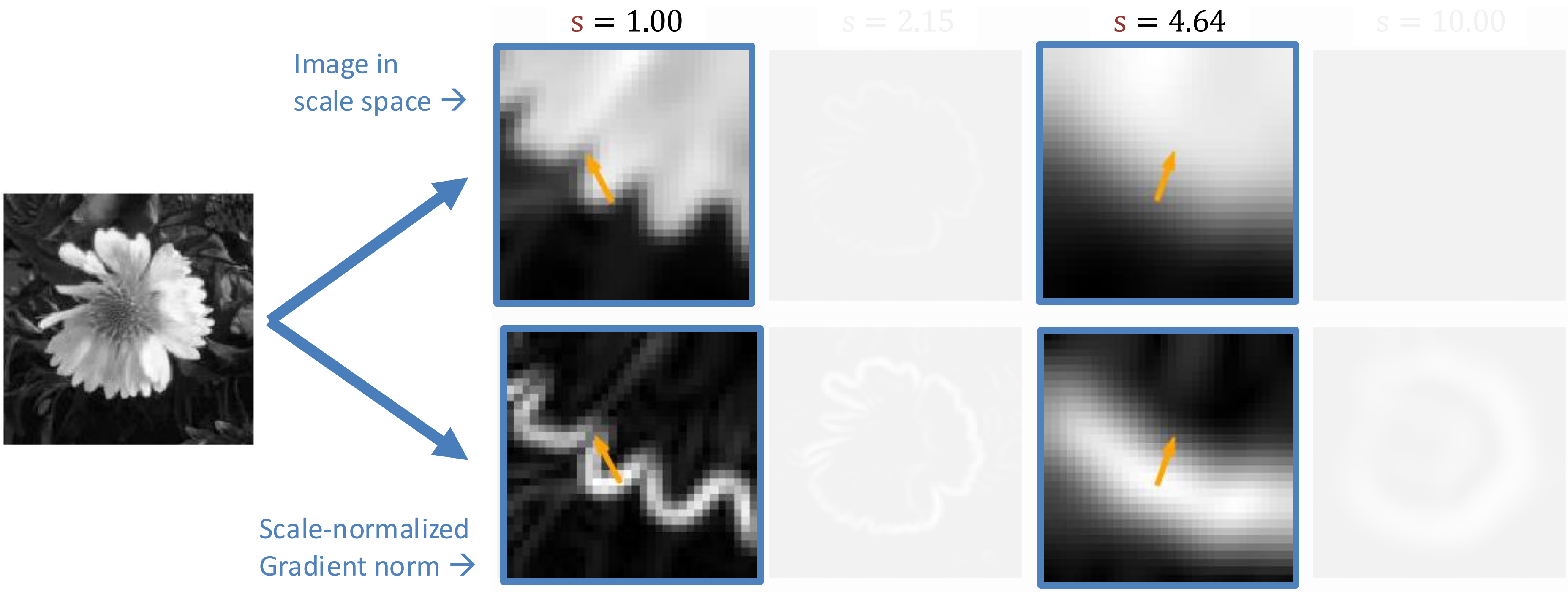
$$\boxed{s^k\left(f^0 * \partial^k_{\dots} G^s\right)}$$

Gaussian Derivatives & Scale



$$\boxed{s^k\left(f^0 * \partial^k_{\ldots} G^s\right)}$$

Gaussian Derivatives & Scale

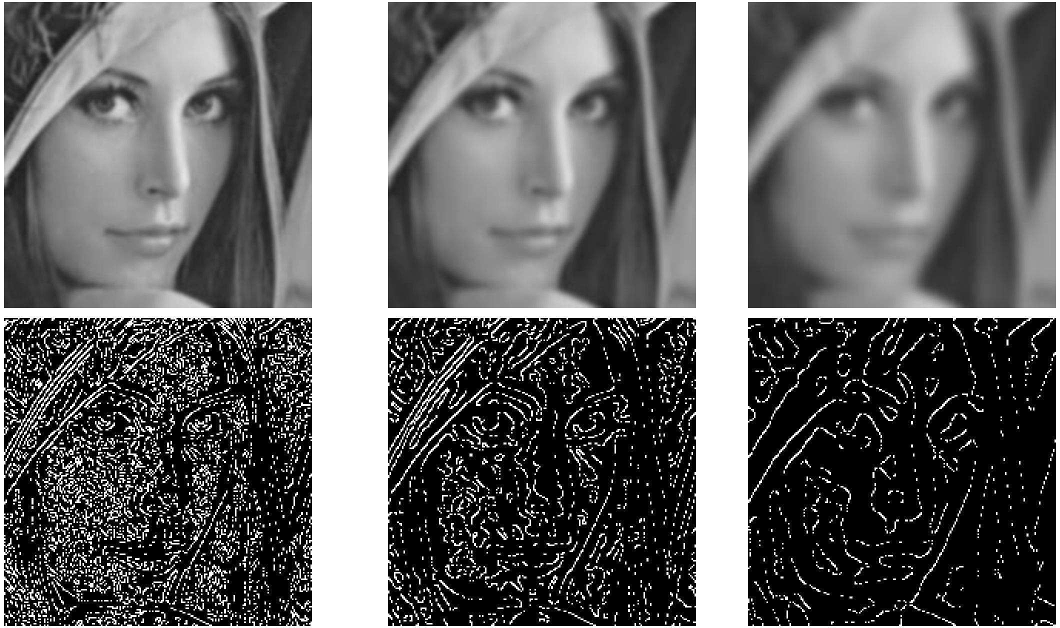


$\boxed{s^k\left(f^0 * \partial^k_{\dots} G^s\right)}$

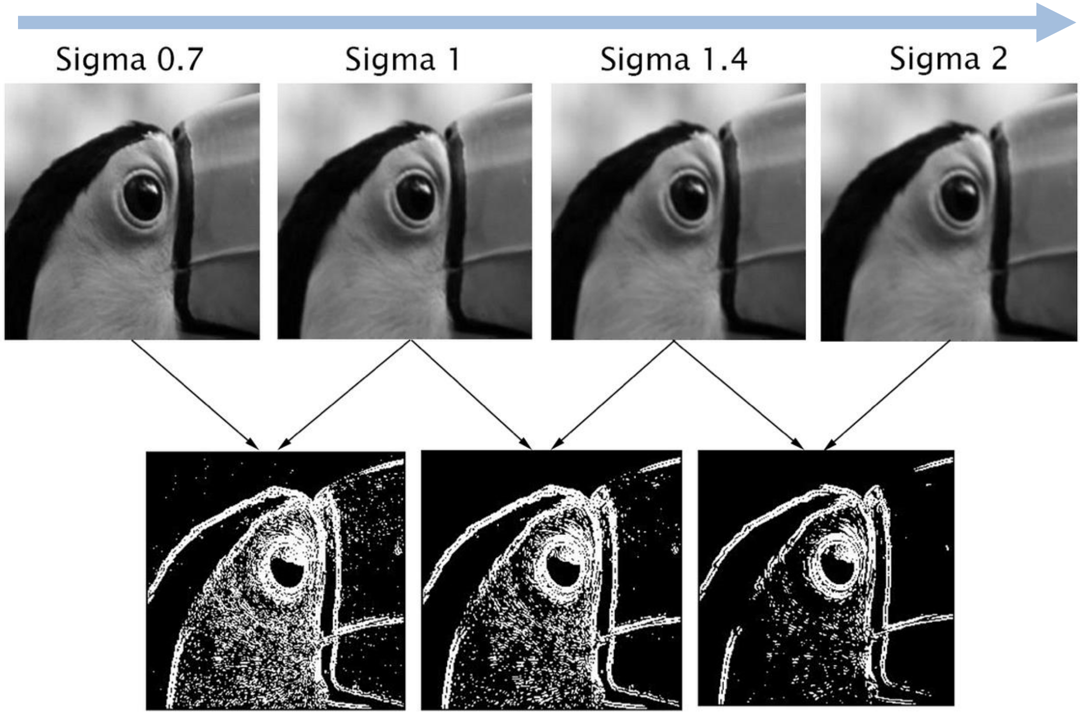
Gradient at each point → depends on **scale** of Gaussian conv

Change of Sigma

Small Sigma → Large Sigma



Change of Sigma – DoG



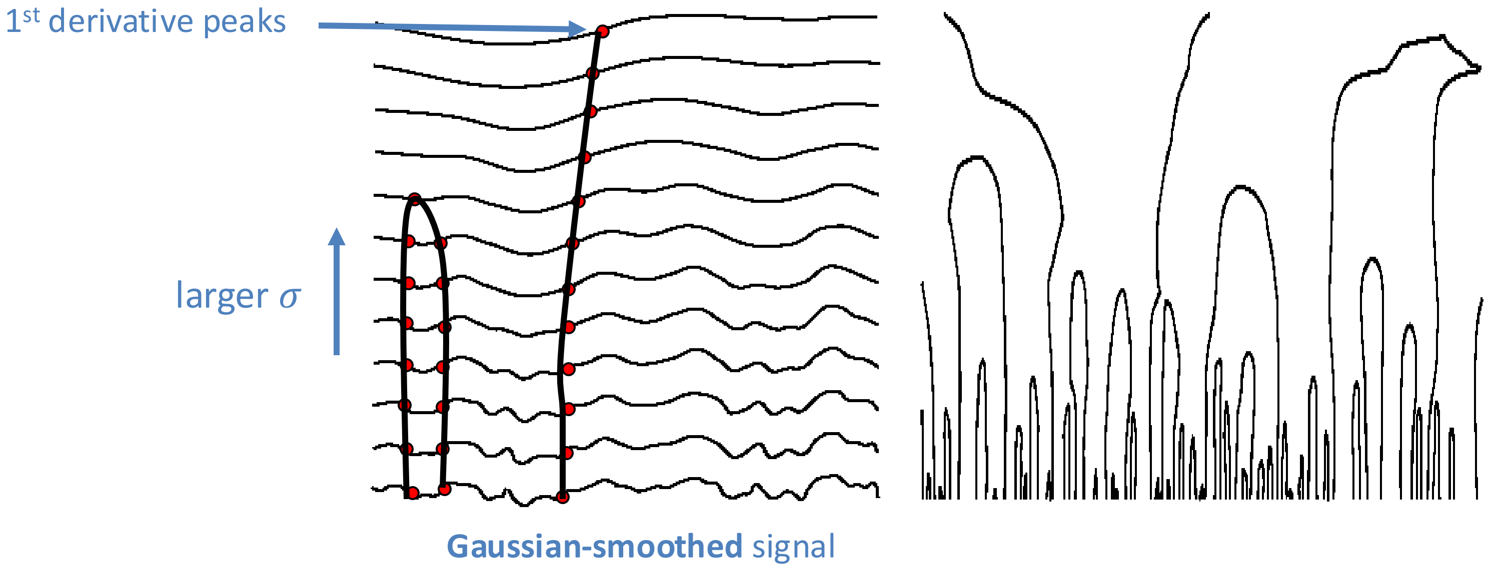
Difference of Gaussians (DoG)

Blurring with bigger-sigma Gaussian suppresses high-frequency info

Subtracting preserves info in the range of frequencies preserved in blurred imgs

DoG acts as a spatial band-pass filter

Edges & Scale [Witkin 83]



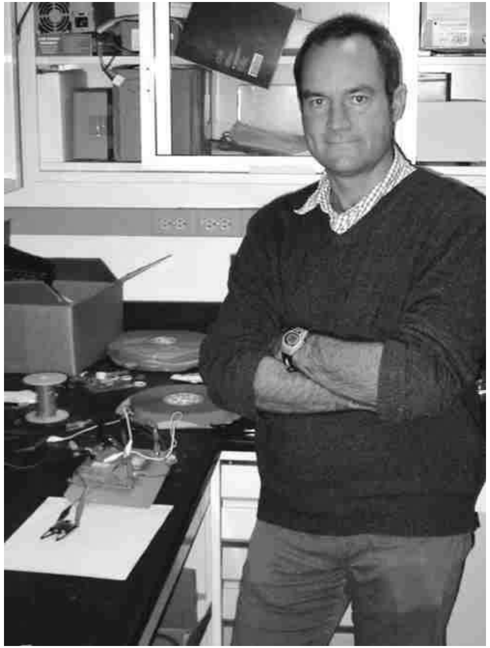
- When the scale σ increases:
  - Edge position may shift
  - Two edges may merge
  - An edge may ***not*** split into two

Slide by:
N. Snavely

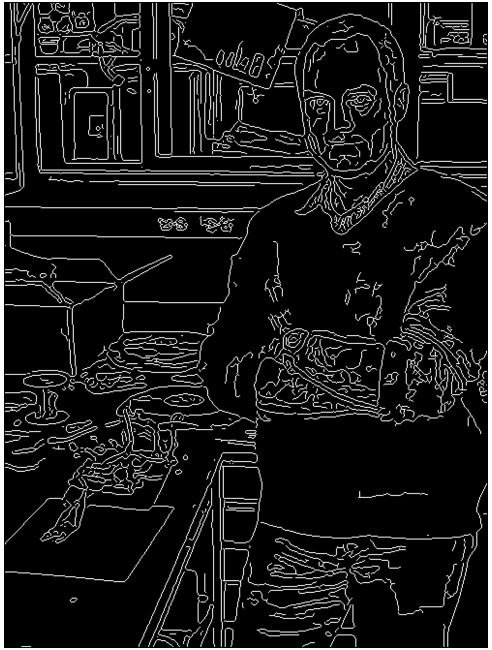
Outline

- Edge Detection
  - Derivatives of Image, Derivatives of Gaussian
  - Canny Edge Detector
- Line Fitting
  - Least Squares
  - RANSAC
  - Hough Transform
- Corners
  - Harris Corners
  - SIFT
  - Applications

Canny Edge Detector



John Canny



One of the most successful edge detection methods

John F. Canny, "A computational approach to edge detection," IEEE Trans. Pattern Analysis and Machine Intelligence, 1986

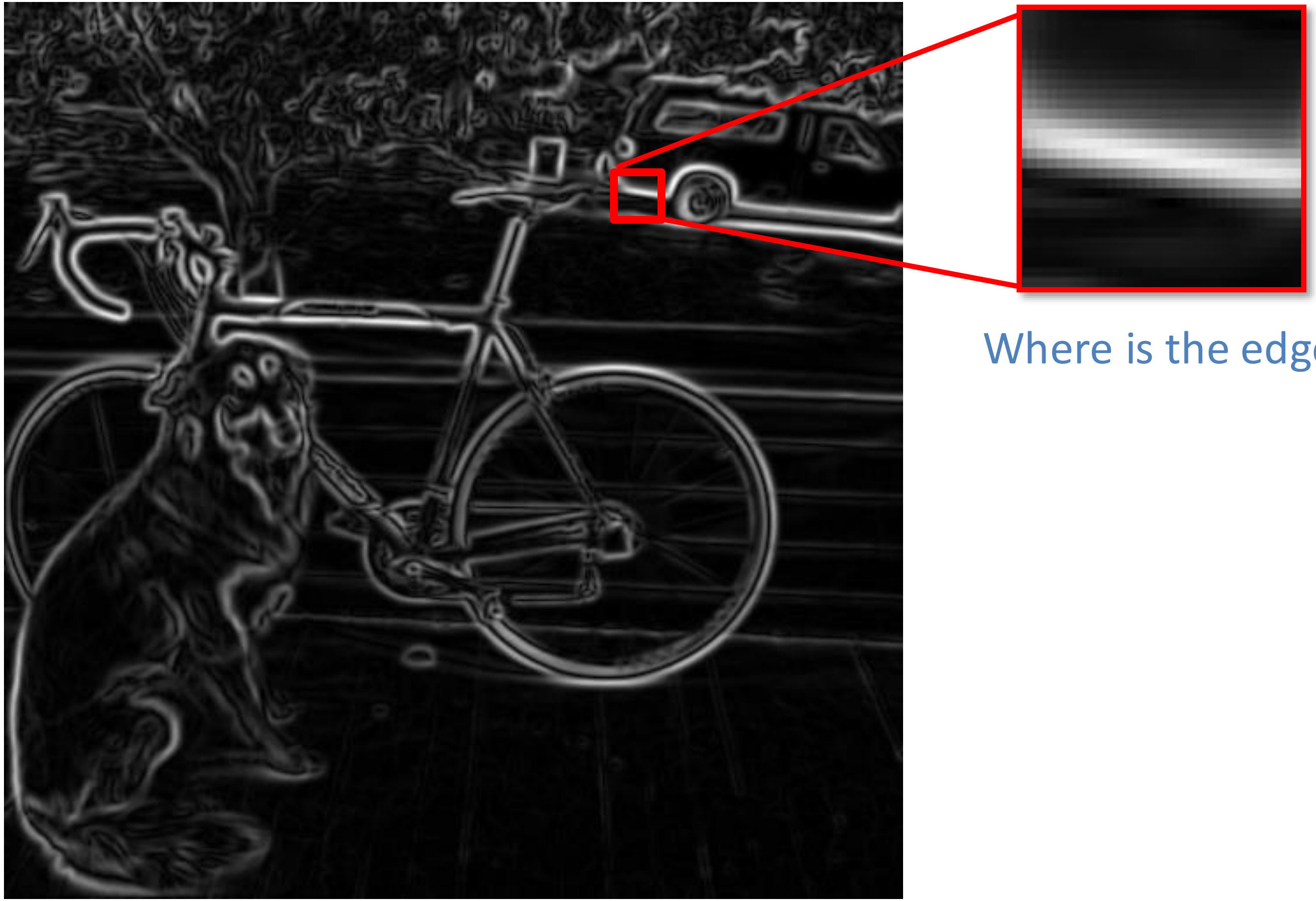
Canny Edge Detector – Input Image

Image credit:
Joseph Redmon

Canny Edge Detector – Gradient Magnitude

After first
pre-smoothing for
cleaner gradients

(Conv with DoG)



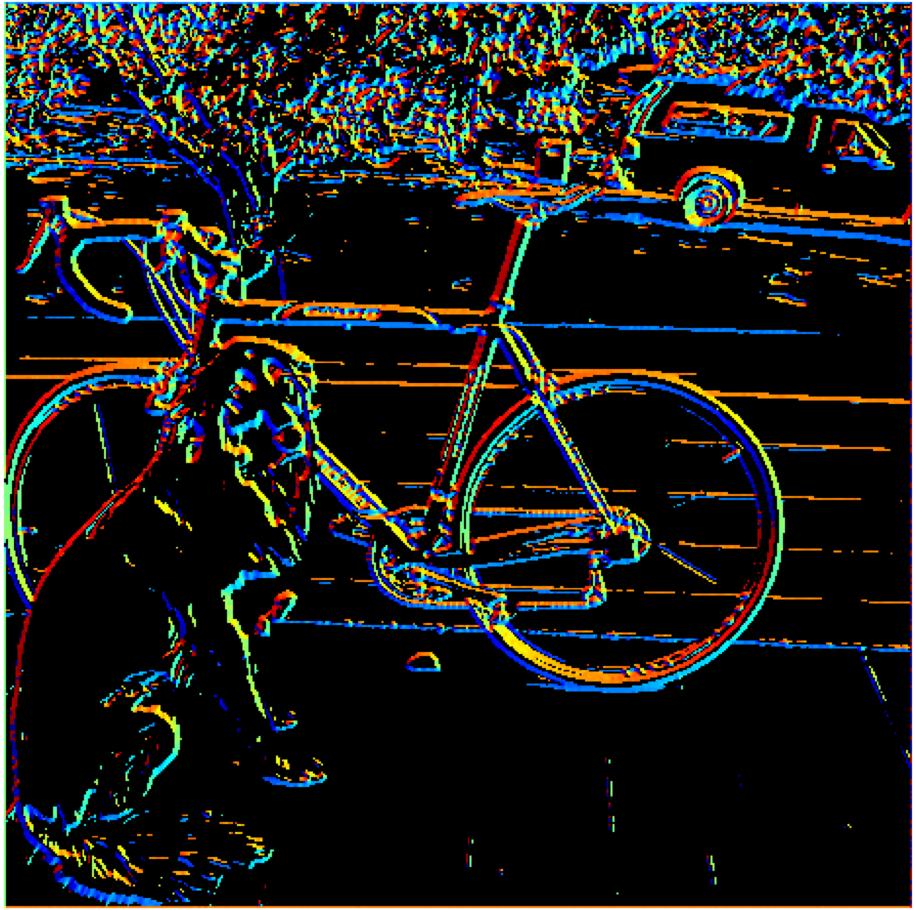
Where is the edge?

Slide by:
N. Snavely

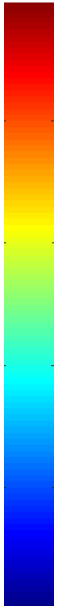
Canny Edge Detector – Gradient Orientation

Get orientation

(threshold at minimum gradient magnitude)



$\theta = \text{atan2}(gy, gx)$

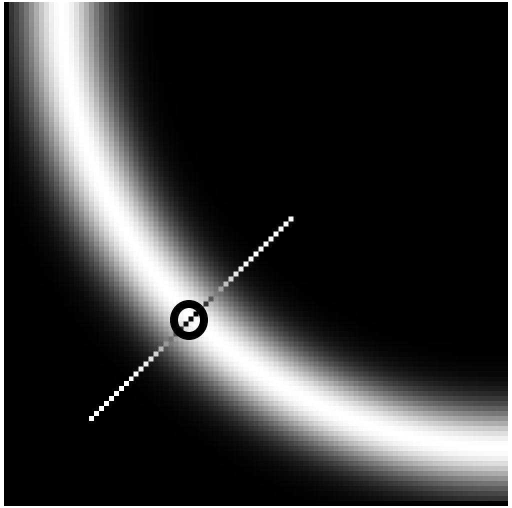


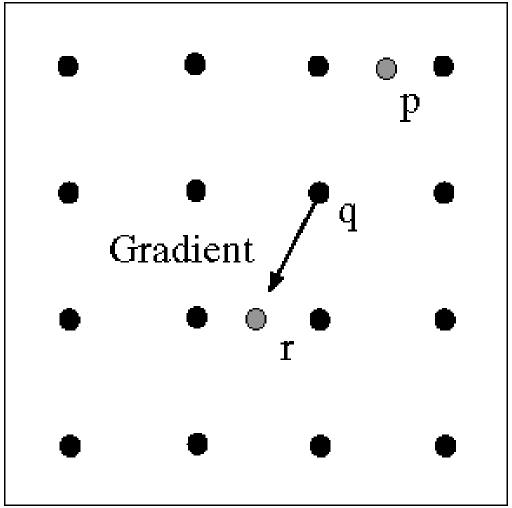
Slide by: N. Snavely

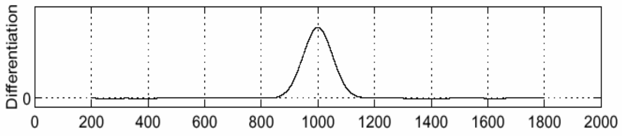
Canny Edge Detector – Non-Max Suppression

Check if pixel is **local maximum** along gradient direction

Requires **interpolating** pixels p and r







Slide by:
N. Snavely

Canny Edge Detector – Non-Max Suppression

Before

Non-max Supr.

After

Non-max Supr.

Canny Edge Detector – Non-Max Suppression

- Still, some noise
- Only want **strong** 'edgels' (edge pixels with value **R**)
- 2 thresholds **T** and **t**, 3 cases:
  - R>T → strong edge
  - t<R<T → weak edge
  - R<t → no edge
- Why two thresholds?

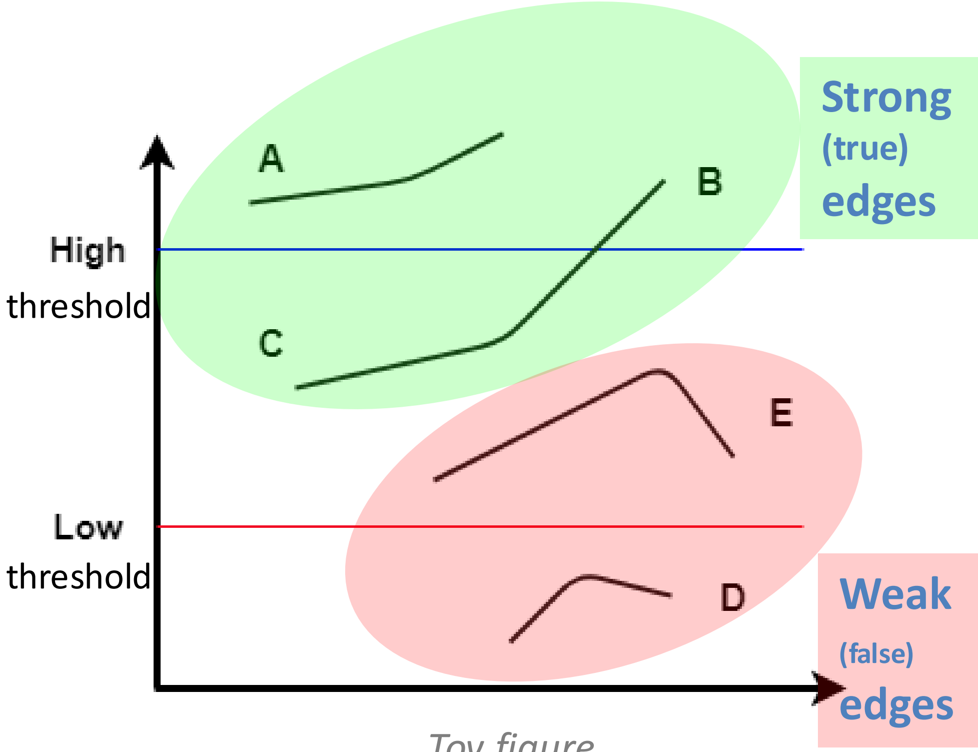
After

Non-max Supr.

Slide by
N. Snavely

Canny Edge Detector – Thresh. & Link Edges

- Need to threshold (hysteresis) resulting edges and link them
- Strong edgels → edges **(A,B)**
- Weak edgels → edges only if connected to strong ones **(C)**
- Look in neighborhood
  (usually 8 closest)



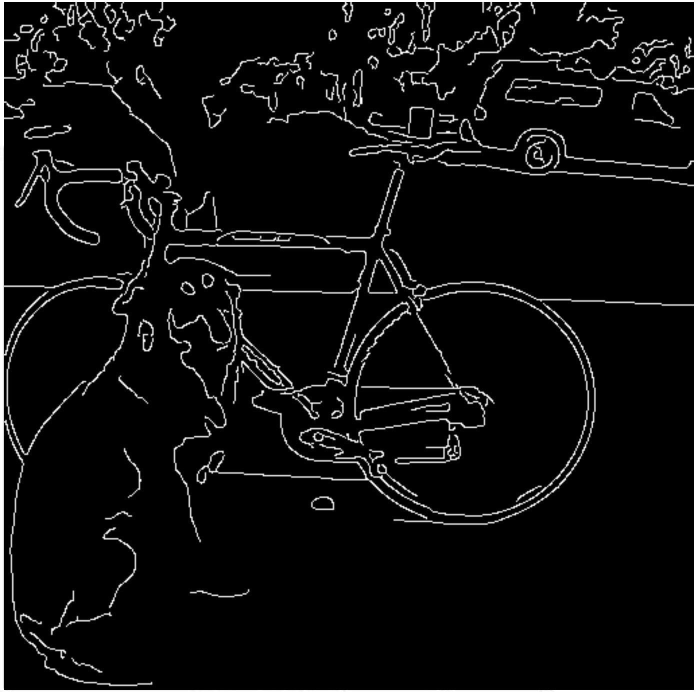
Toy figure
(lines represent pixel values)

The AI learner, Canny edge detection, 2019

Canny Edge Detector

Non-max Suppression

through edge following



Final Output

after thresholding edges

Canny Edge Detector – Full Example

Input Image

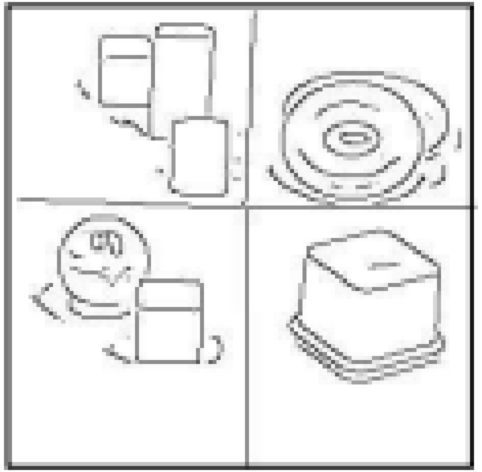
Gradient Magnitude

Non-max Suppression

Final Output

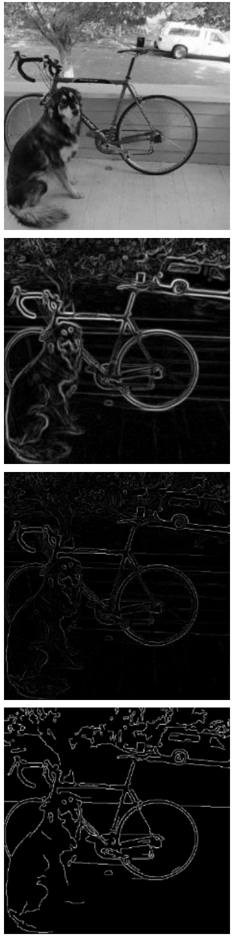
after hysteresis (thresholding) and linking edges

Canny Edge Detector



Also applicable on color images

Canny Edge Detector – Summary



MATLAB: `edge(image,'canny')`

OpenCV: `cv.canny(I, low_thr, high_thr)`

1. **Filter** image with Derivative of Gaussian
2. Compute gradient magnitude & orientation
3. Non-maximum suppression
4. Linking and thresholding (hysteresis):
   - Define two thresholds:
   - High threshold → Use to start edge curves
   - Low threshold → Use to continue them

Slide by: D. Lowe, L. Fei-Fei, J. Redmon

Canny Edge Detector

Original Image

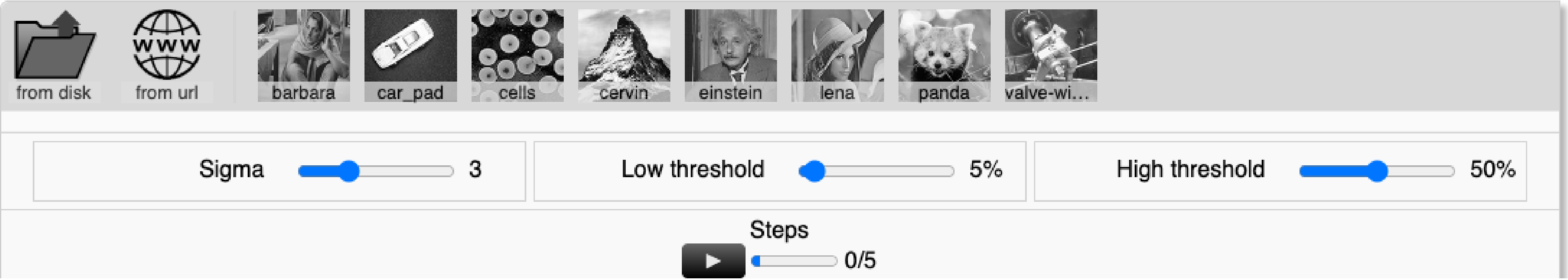
Canny with $\sigma = 1$

Canny with $\sigma = 2$

- Choice of σ depends on desired behavior:
  - Large σ detects 'large-scale' edges
  - Small σ detects fine edges

Slide by:
S. Seitz

Canny Edge Detector – Demo

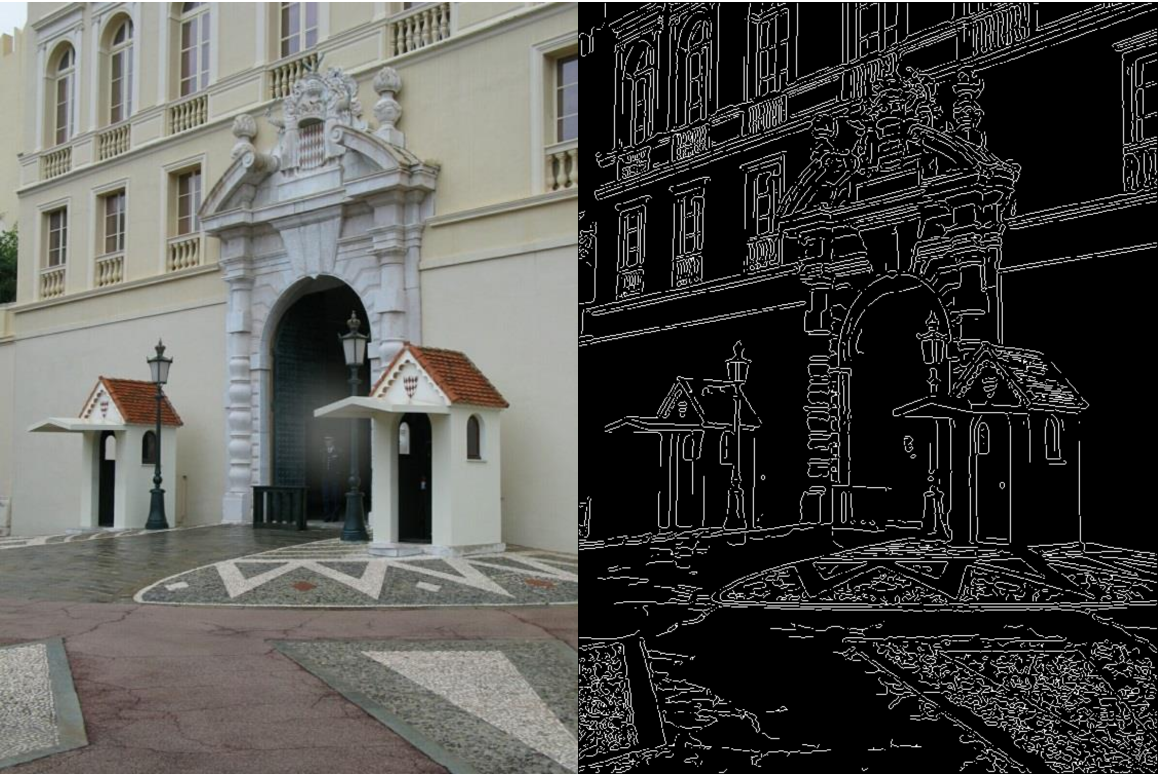


https://bigwww.epfl.ch/demo/ip/demos/edgeDetector

Outline

- Edge Detection
  - Derivatives of Image, Derivatives of Gaussian
  - Canny Edge Detector
- Line Fitting
  - Least Squares
  - RANSAC
  - Hough Transform
- Corners
  - Harris Corners
  - SIFT
  - Applications

Image Structure



Binary Edges Not Enough

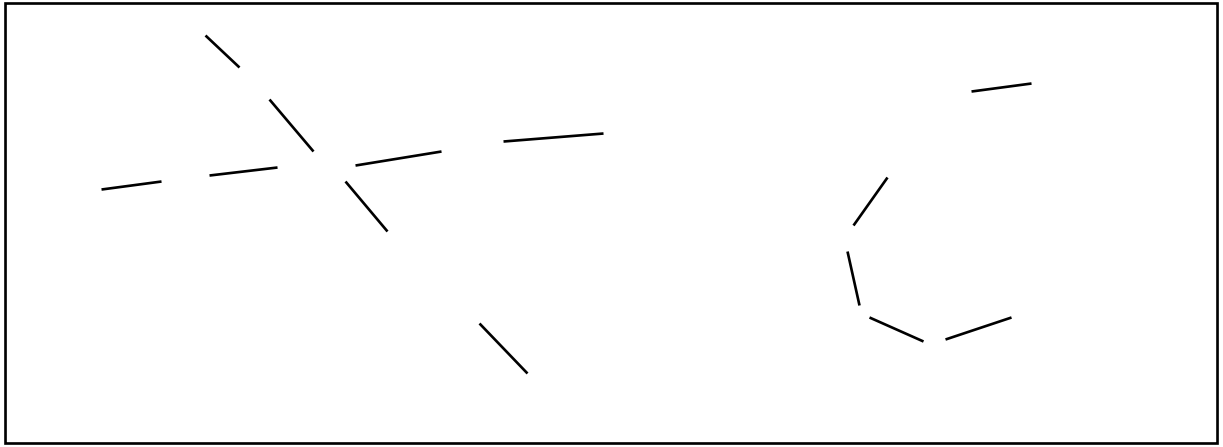
- Fragmented edges
- False positives
- Noisy edge orientations
- Undefined Membership which points belong to which line ?

Image Structure

Example of Vanishing Lines

Line Fitting – Goal

- From fragmented edges
- To straight lines or curves

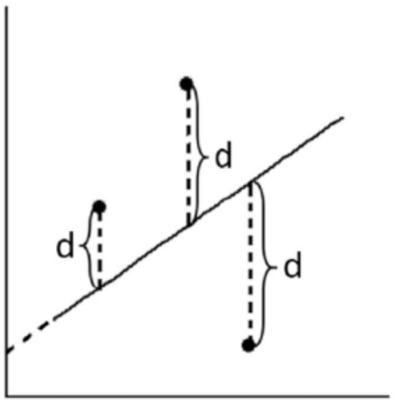


Outline

- Edge Detection
  - Derivatives of Image, Derivatives of Gaussian
  - Canny Edge Detector
- Line Fitting
  - Least Squares
  - RANSAC
  - Hough Transform
- Corners
  - Harris Corners
  - SIFT
  - Applications

Line Fitting – Ordinary Least Squares

- **Input:** Set of points belonging to a straight line
- **Output:** Find the line equation
- **Nugget:** Find the line parameters that minimize error in a least-square sense

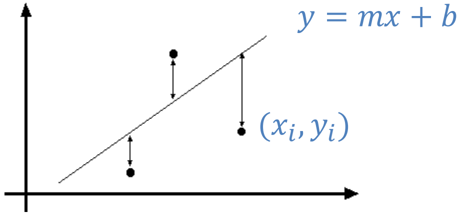


Least Squares

Line Fitting – Ordinary Least Squares

Input: Data points $(x_1, y_1), \dots, (x_n, y_n)$

Line model $y_i = mx_i + b$



Output: Find $p = (m, b)$ that **minimizes** $e = \sum_{i=1}^{n} (y_i - mx_i - b)^2$

Minimize Error:

$$e = \sum_{i=1}^{n} \left(y_i - \begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} \right)^2$$

$$= \left\| \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} \right\|^2$$

$$= \|\mathbf{y} - \mathbf{A}p\|^2$$

$$= \mathbf{y}^T\mathbf{y} - 2(\mathbf{A}p)^T\mathbf{y} + (\mathbf{A}p)^T(\mathbf{A}p)$$

$$\frac{de}{dp} = 2\mathbf{A}^T\mathbf{A}p - 2\mathbf{A}^T\mathbf{y} = 0$$

Pseudoinverse of A

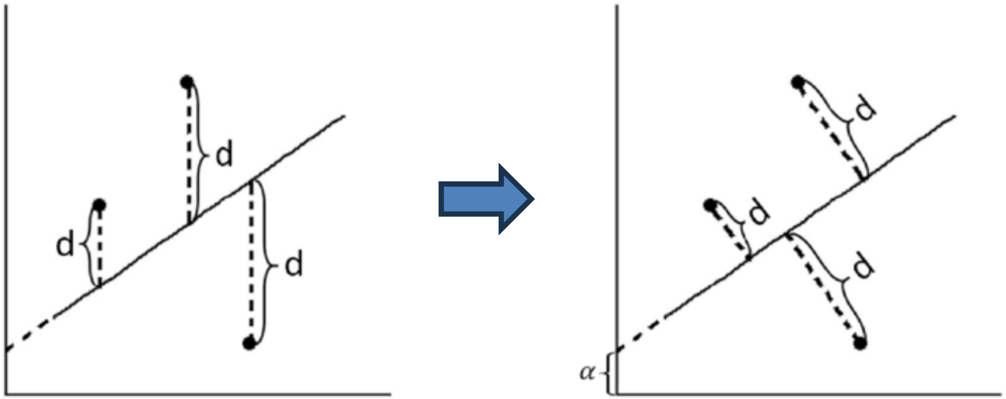
$$\therefore p = \boxed{(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T}\mathbf{y}$$

Matlab: `p = A \ y`

Slide by S. Lazebnik

Ordinary Least Squares – Problems

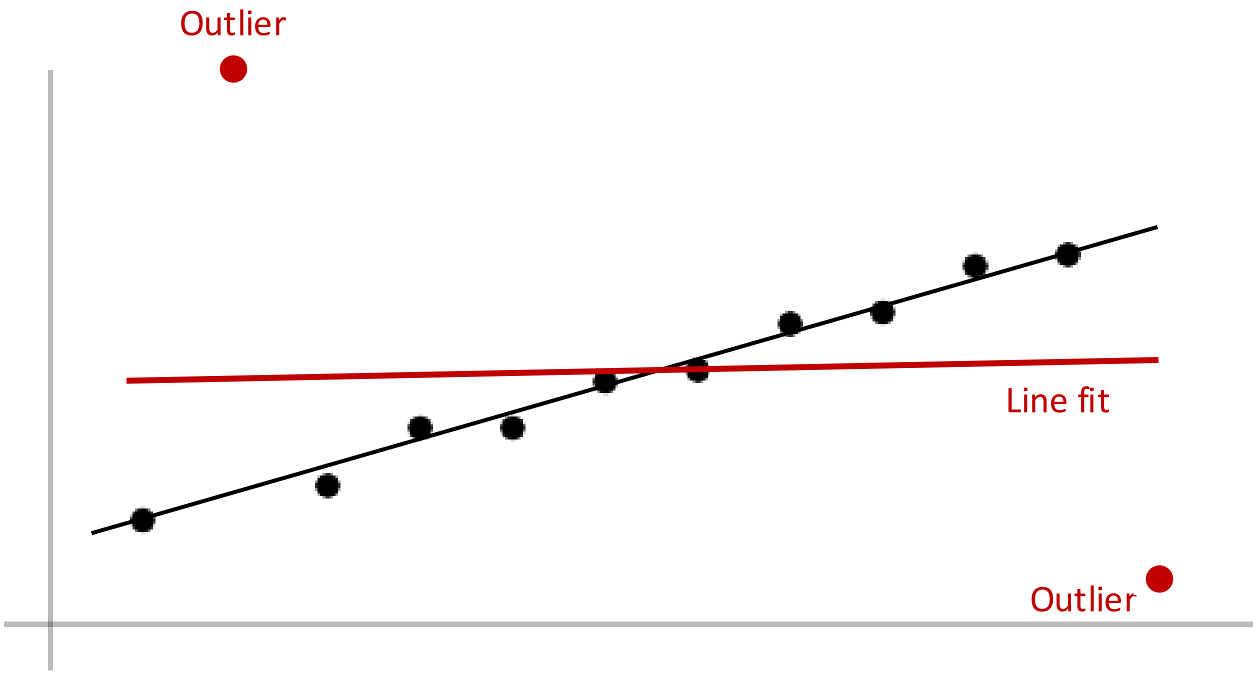
- **Problem:** Fitting performance affected by slope
  Fails completely for vertical lines
- **Solution:** 'Total least squares' using a perpendicular distance



Least Squares | Total Least Squares

Slide by
S. Lazebnik

Least Squares Fitting

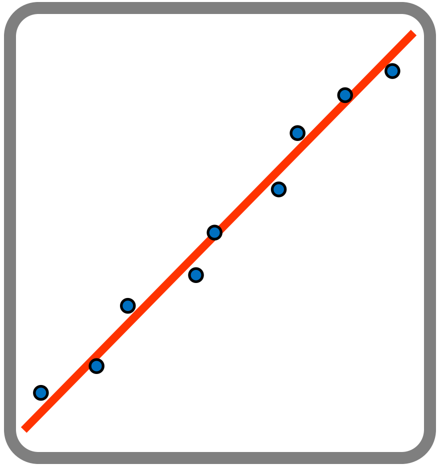


Heavily sensitive to outliers 😢

Outline

- Edge Detection
  - Derivatives of Image, Derivatives of Gaussian
  - Canny Edge Detector
- Line Fitting
  - Least Squares
  - RANSAC
  - Hough Transform
- Corners
  - Harris Corners
  - SIFT
  - Applications

RANSAC – Motivation

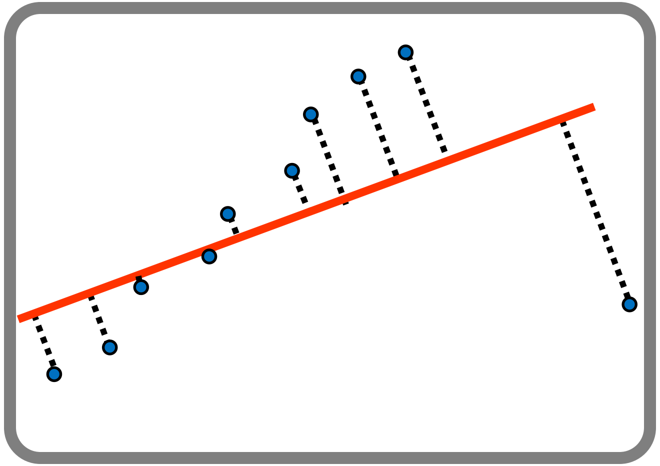


Only **inliers**

2D point → n-dim data point

2D line → parametric model

How do we fit a model to data points?



Effects due to a *single* **outlier** (**noisy point**)

Detecting outliers is crucial!

Model Fitting – 2D Line

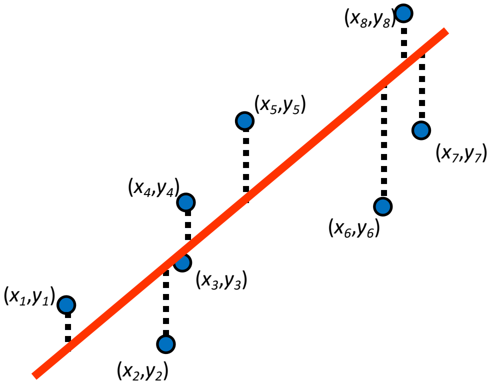
Line fitting

- **Input:** Set of n 2D points (x_i, y_i)
- **Output:** Line $\Theta = (a, b)$ *minimizing* **fitting error**

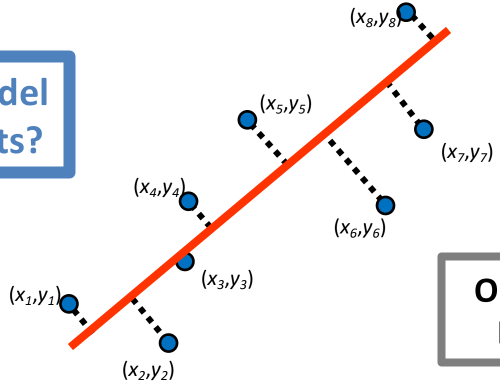
Line equation: $y = ax + b$

Fitting cost: YOU choose it ☺

Vertical Distance



How 'far' is my model from my data points?



Orthogonal Distance

Fitting Cost – Norm

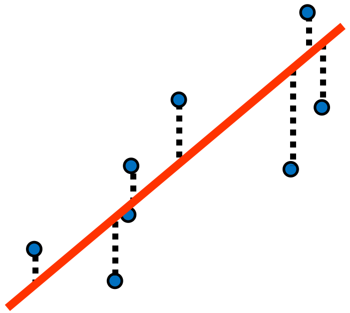
Let:

$$\mathbf{x} \in \mathbb{R}^n$$

$$\mathbf{x} = (x_1, \ldots, x_n)$$

L_p norm:

$$\boxed{\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}}$$



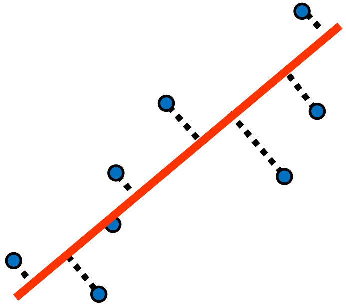
$$\mathbf{x} = (x_1, x_2, x_3) = (-3, 0, 4)$$

$\|x\|_1 = ?$

$$\|\mathbf{x}\|_1 = |-3| + |0| + |4| = 7$$

$\|x\|_2 = ?$

$$\|\mathbf{x}\|_2 = \sqrt{(-3)^2 + 0^2 + 4^2} = \sqrt{25} = 5$$



Fitting Cost – Norm

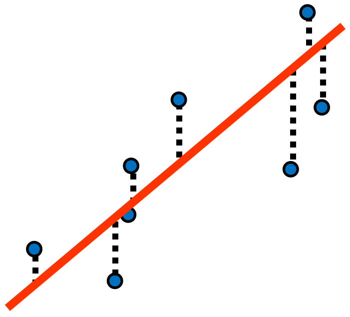
Let:

$$\mathbf{x} \in \mathbb{R}^n$$

$$\mathbf{x} = (x_1, \ldots, x_n)$$

L_p norm:

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$$



For $n = 1$:
(1D case)

Without proof here

$$(L_2): \quad \arg\min_u \sum_{i=1}^{n} |u - x_i|^2$$

Solution → Average

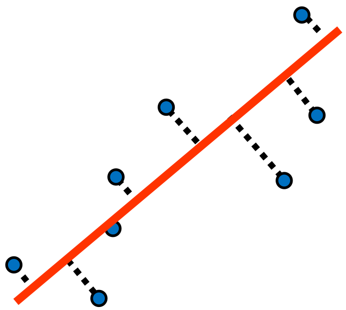
$$u = \frac{1}{n} \sum_{i=1}^{n} x_i$$

$$(L_1): \quad \arg\min_u \sum_{i=1}^{n} |u - x_i|$$

Solution → Median

if $a < b < c < d$,
the **median** of the **list** $\{a, b, c, d\}$
is the mean of b and c; i.e., it is $(b + c)/2$.

http://en.wikipedia.org/wiki/Median



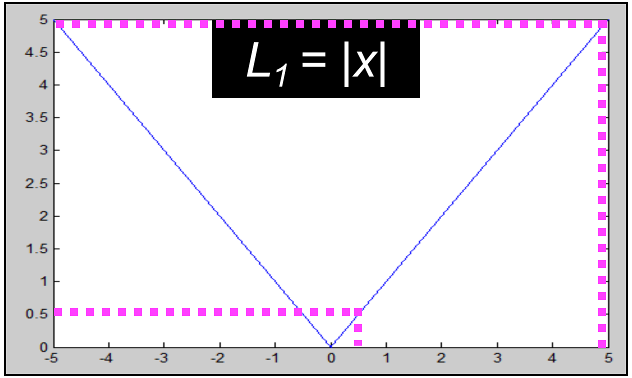
Slide by Jean-Charles Bazin

Fitting Cost – Norm

Y axis: How much 'penalty' u pay

x=5 y=5

x=0.5 y=0.5



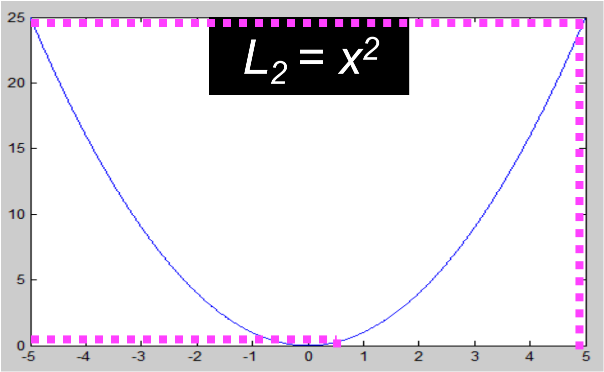
Do you need …

… heavy penalty on big errors?

Choose L_2 → →

… noticeable penalty on tiny errors?

← ← Choose L_1

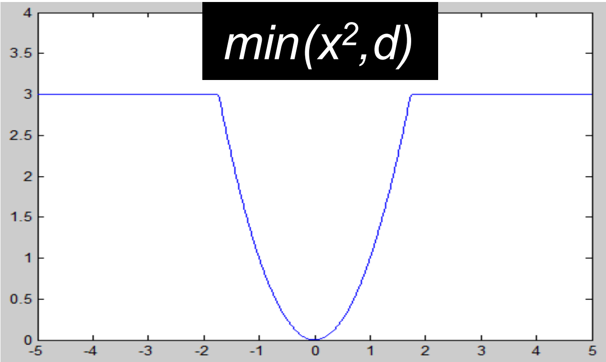


x=5 y=25

x=0.5 y=0.025

X axis: For an error u do

d →



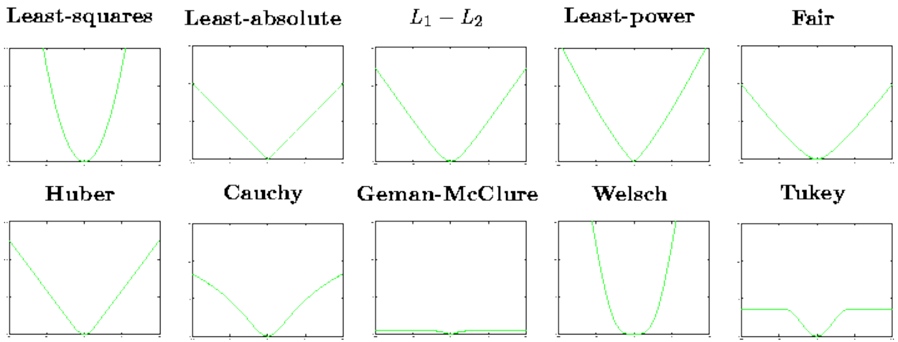
'Robustifier' → →
Influence of outliers 'saturates'

Which one to choose?

Design/play with your cost function

Cost Functions

Go beyond just norms → aka ‘***cost functions***’



type	$\rho(x)$
L_2	$x^2/2$
L_1	$\lvert x \rvert$
$L_1 - L_2$	$2(\sqrt{1 + x^2/2} - 1)$
L_p	$\frac{\lvert x \rvert^\nu}{\nu}$
“Fair”	$c^2[\frac{\lvert x \rvert}{c} - \log(1 + \frac{\lvert x \rvert}{c})]$
Huber $\begin{cases} \text{if } \lvert x \rvert \le k \\ \text{if } \lvert x \rvert \ge k \end{cases}$	$\begin{cases} x^2/2 \\ k(\lvert x \rvert - k/2) \end{cases}$
Cauchy	$\frac{c^2}{2} \log(1 + (x/c)^2)$
Geman-McClure	$\frac{x^2/2}{1 + x^2}$
Welsch	$\frac{c^2}{2}[1 - \exp(-(x/c)^2)]$
Tukey $\begin{cases} \text{if } \lvert x \rvert \le c \\ \text{if } \lvert x \rvert > c \end{cases}$	$\begin{cases} \frac{c^2}{6}(1 - [1 - (x/c)^2]^3) \\ (c^2/6) \end{cases}$

Rules of thumb (*but look things up on your own*):

- **L2, L1, Huber:** These are the most popular, depending on the applications
- **Huber:** Best of both worlds. Treats small errors like L2 (tolerates them) and large errors like L1 (robustness).
- **Dimitris:** For my problems – I use a lot the Geman-McClure one. Chosen after experimentation.

YOU *choose*/*define* it ☺ → *Play with parameters & judge for your own task/data* ⚠

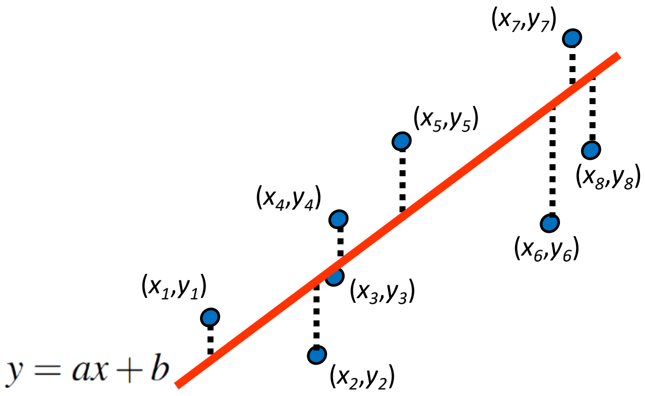
Model Fitting – Under Noise

Data points: (x_i, y_i)

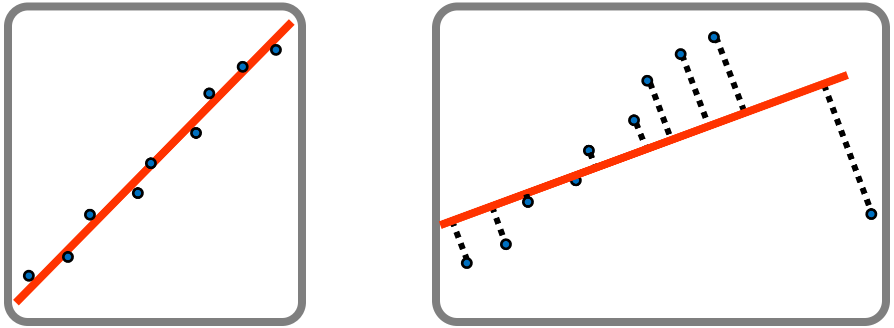
Line model: $y = ax + b$

Parameters: $\Theta = (a, b)$

L_1 cost: $\min_{\Theta=(a,b)} \sum_{i=1}^{n} |y_i - (ax_i + b)|$



But remember…



Only inliers

Effects due to single outlier

Detecting outliers is very important!

Solution: ***RANSAC***

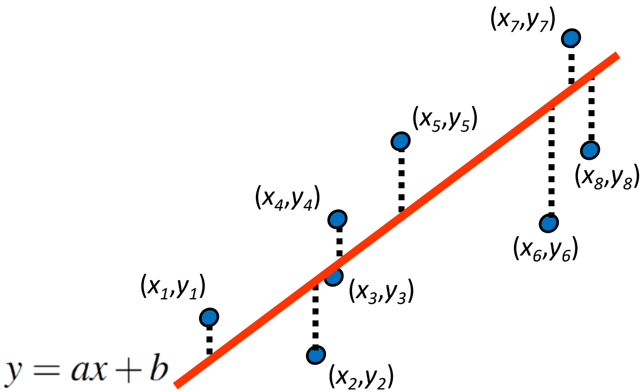
Model Fitting – Under Noise

Data points: (x_i, y_i)

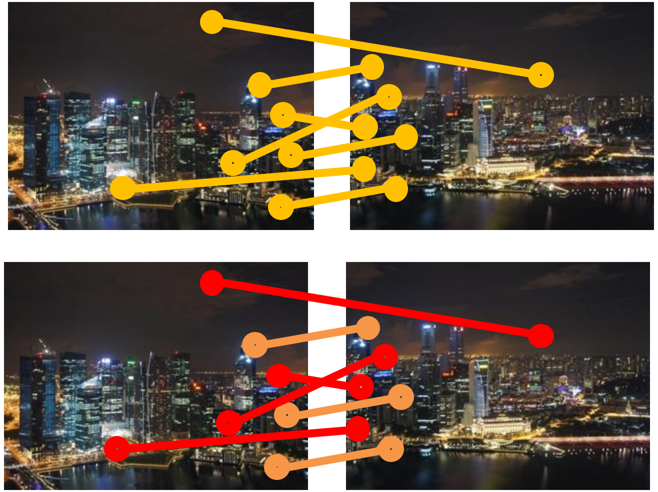
Line model: $y = ax + b$

Parameters: $\Theta = (a, b)$

L_1 cost: $\min_{\Theta=(a,b)} \sum_{i=1}^{n} |y_i - (ax_i + b)|$



Motivation

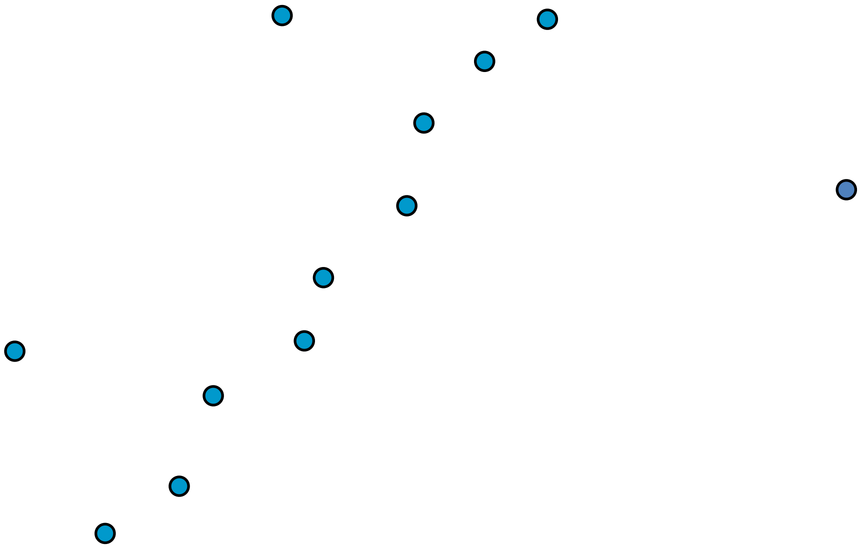


Detecting outliers is very important!

Solution: RANSAC

Also for Vision
(coming soon)

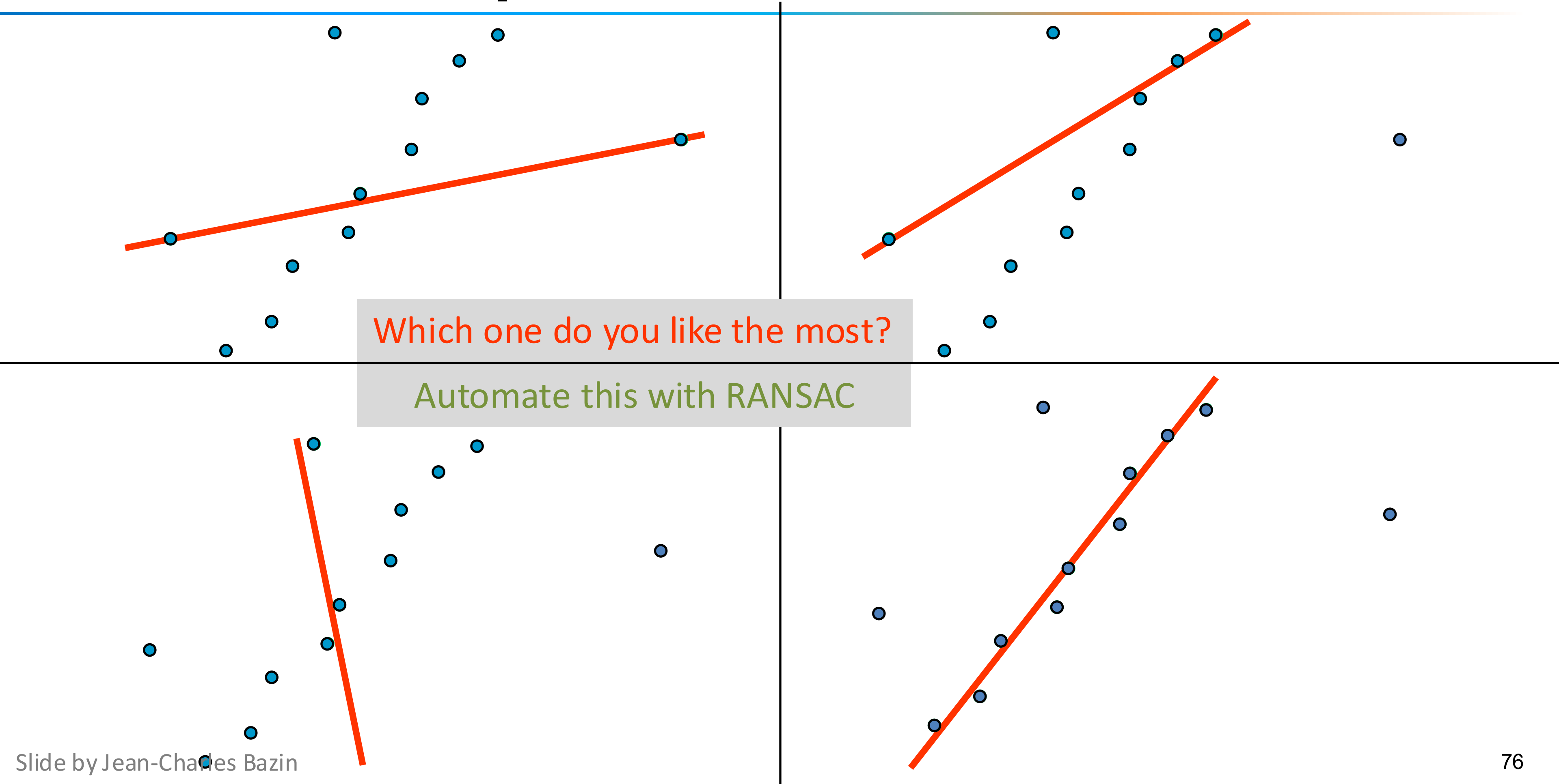
RANSAC – Example – Lines



Input: Set of points

Output: Estimate line equation & determine inliers/outliers

RANSAC – Example – Lines



Which one do you like the most?

Automate this with RANSAC

Slide by Jean-Charles Bazin

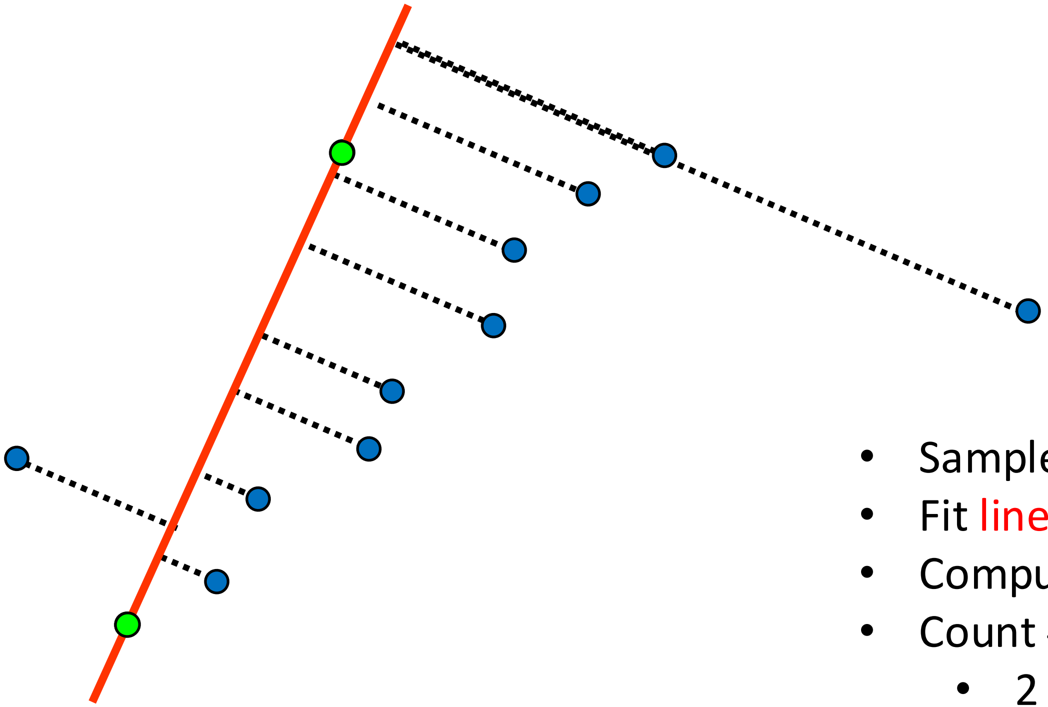
RANSAC – Example – Lines

RANSAC - RANdom SAmple Consensus
by Fischler and Bolles, 1981

Very ***easy*** method and ***good*** results

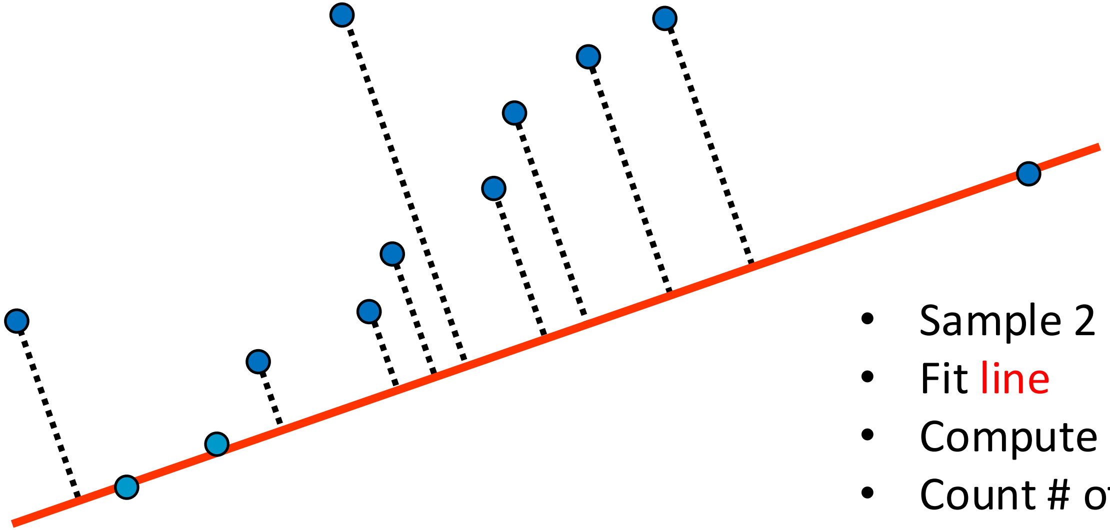
Aims to **maximize** the **number #** of **inliers** ('consensus set maximization') through **random sampling**

RANSAC – Example – Lines



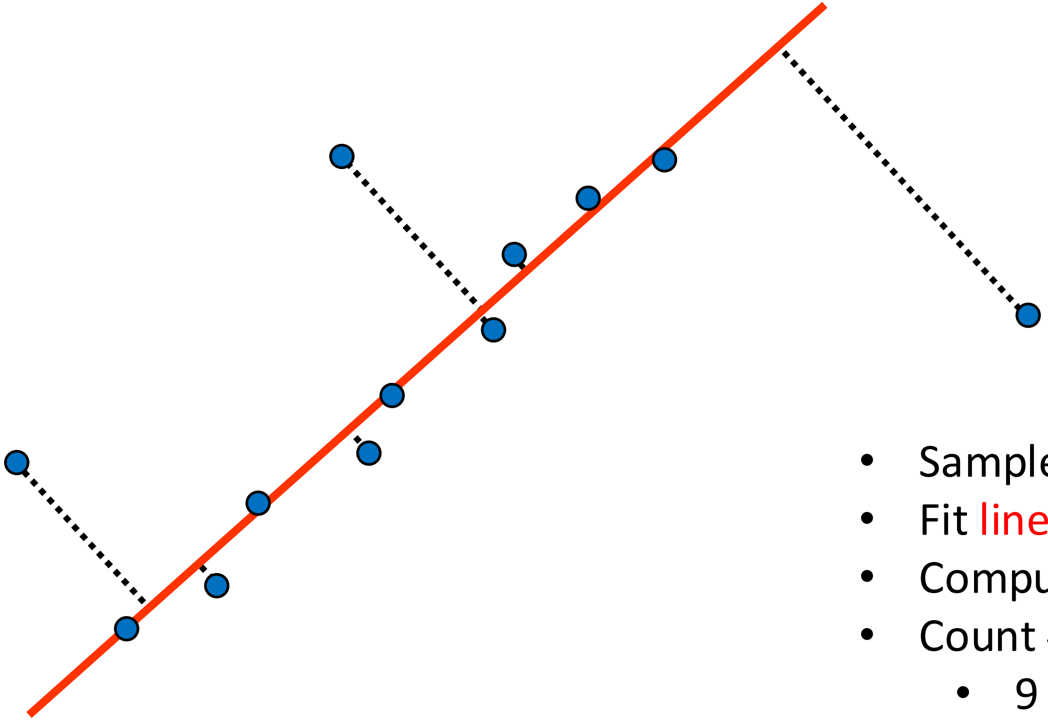
- Sample 2 points randomly
- Fit line
- Compute distances
- Count # of inliers
  - 2 out of 13

RANSAC – Example – Lines



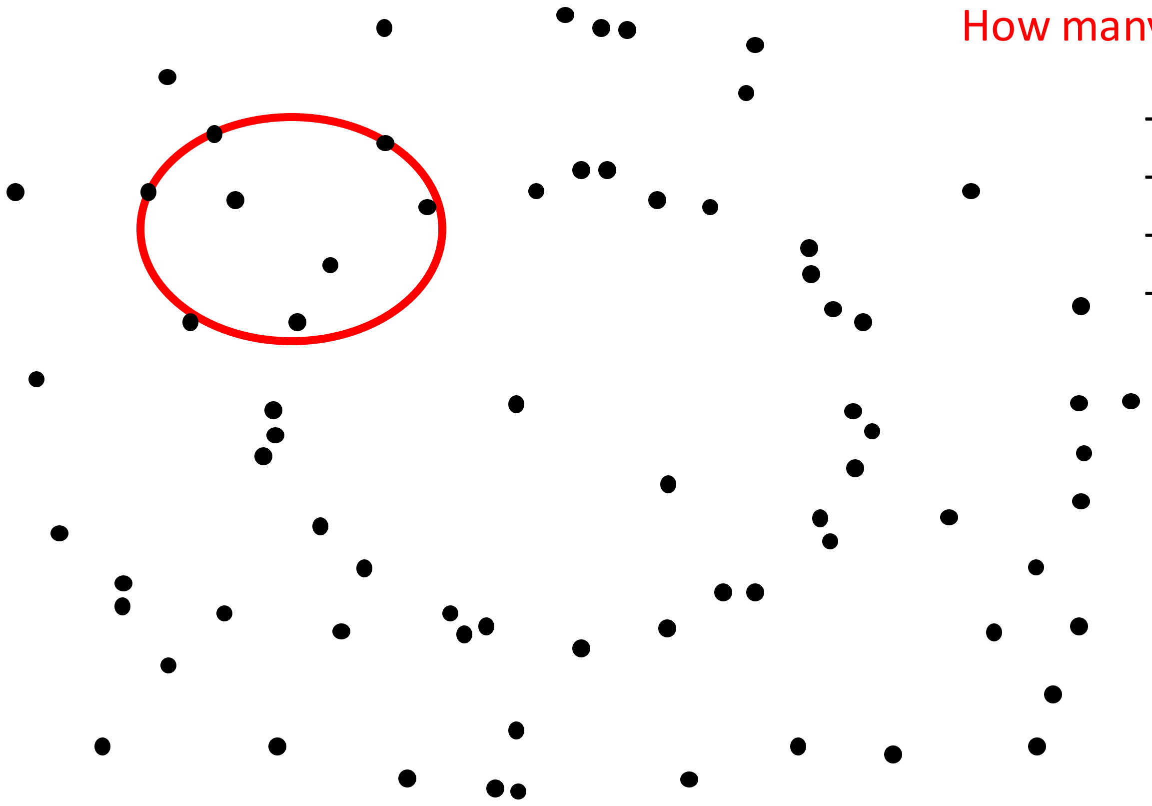
- Sample 2 points randomly
- Fit line
- Compute distances
- Count # of inliers
  - 3 out of 13

RANSAC – Example – Lines



- Sample 2 points randomly
- Fit line
- Compute distances
- Count # of inliers
  - 9 out of 13

RANSAC – Example – Circles



How many points for a circle?

-Sample 3 points randomly

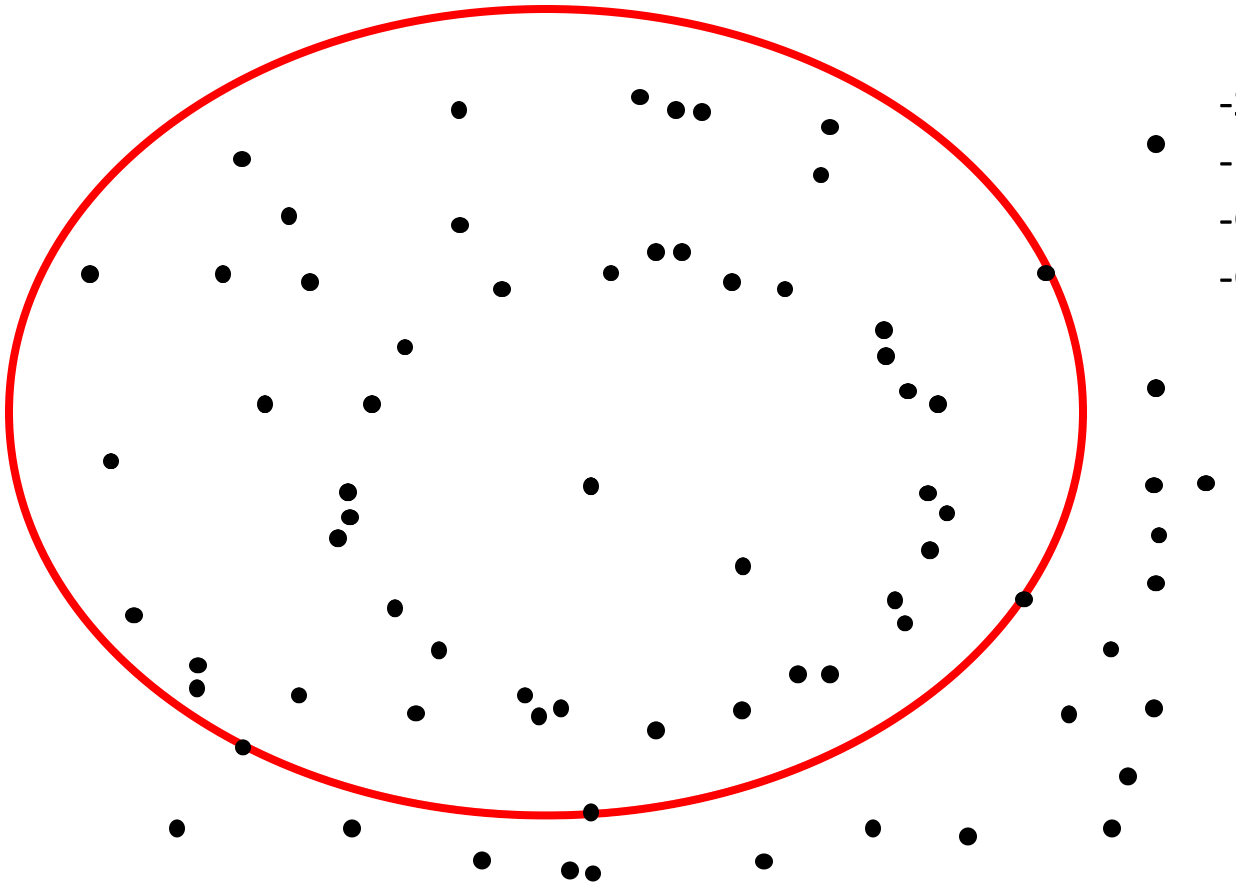
-Fit circle

-Compute distances

-Count # of inliers

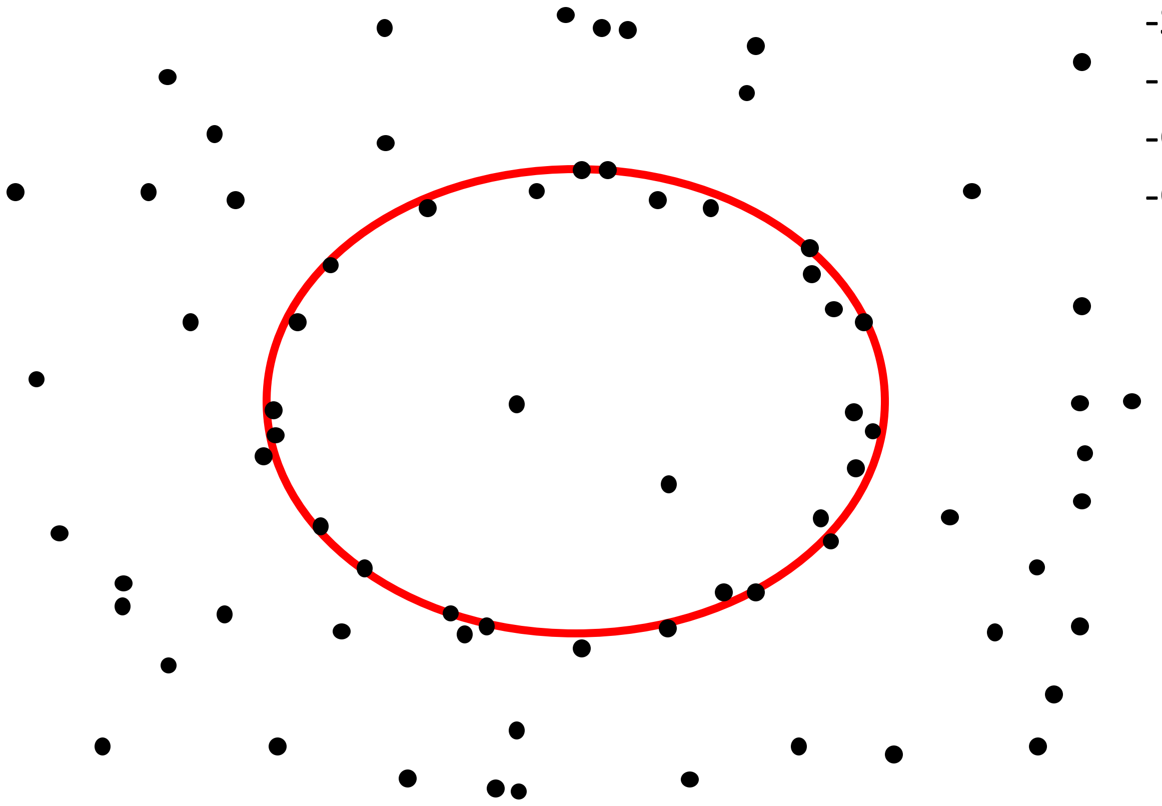
-5

RANSAC – Example – Circles



- -Sample 3 points randomly
- -Fit circle
- -Compute distances
- -Count # of inliers
  - -4

RANSAC – Example – Circles



-Sample 3 points randomly

-Fit circle

-Compute distances

-Count # of inliers

-23 ✓

RANSAC – Quiz

Count # of **Inlier Candidates**

Estimate **Parameters** using Sampled Data

Maximum # of Inlier Candidates until Now?

Calculate **Error** of Data w.r.t. the Estimation

Assemble Pseudo-Algorithm?

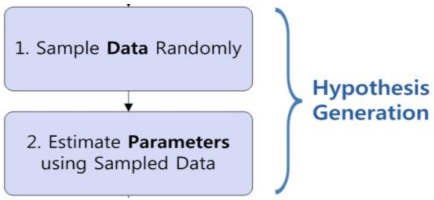
Enough # of Iteration?

Keep the Parameters as the Final Estimation

Sample **Data** Randomly

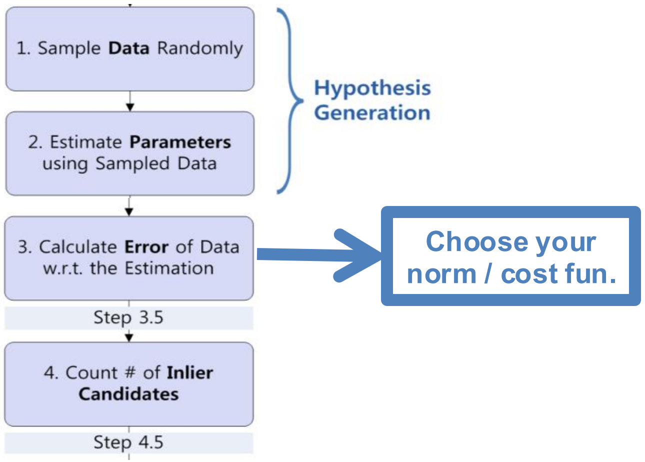
Return the Final Estimation

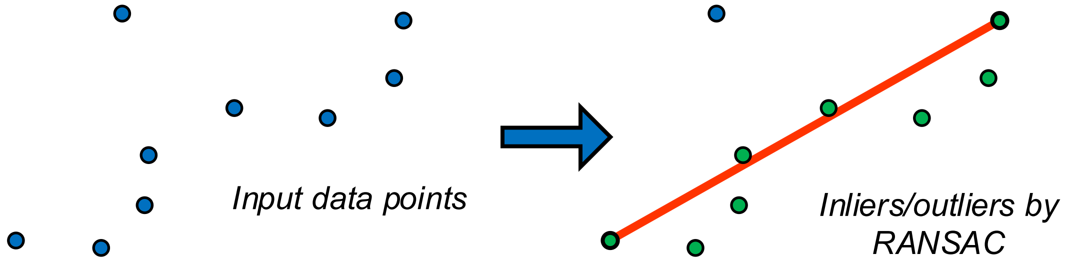
RANSAC – Overview



From "Performance Evaluation of RANSAC Family", by Choi et al., BMVC, 2009

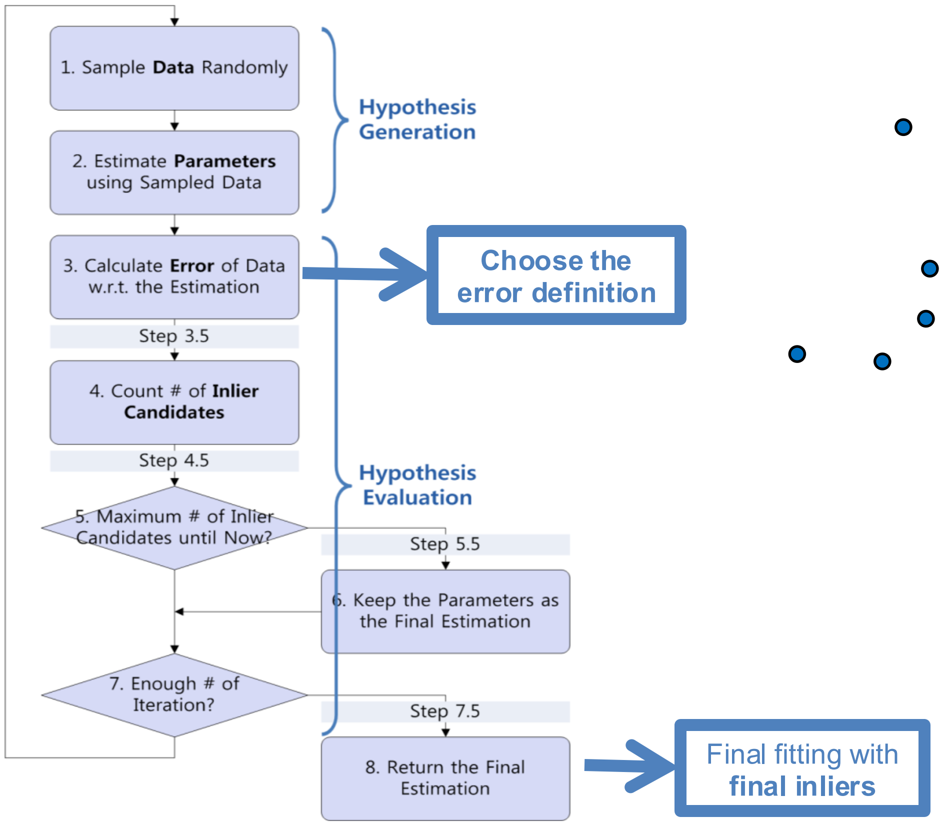
RANSAC – Overview

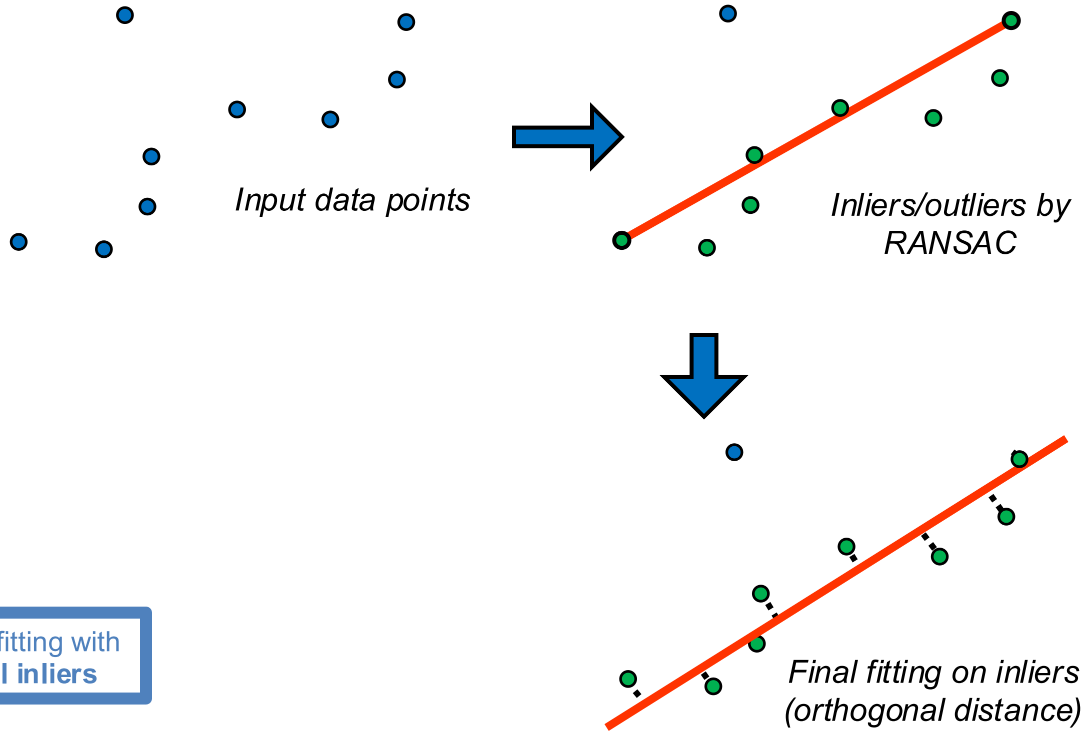




From "Performance Evaluation of RANSAC Family", by Choi et al., BMVC, 2009

RANSAC – Overview

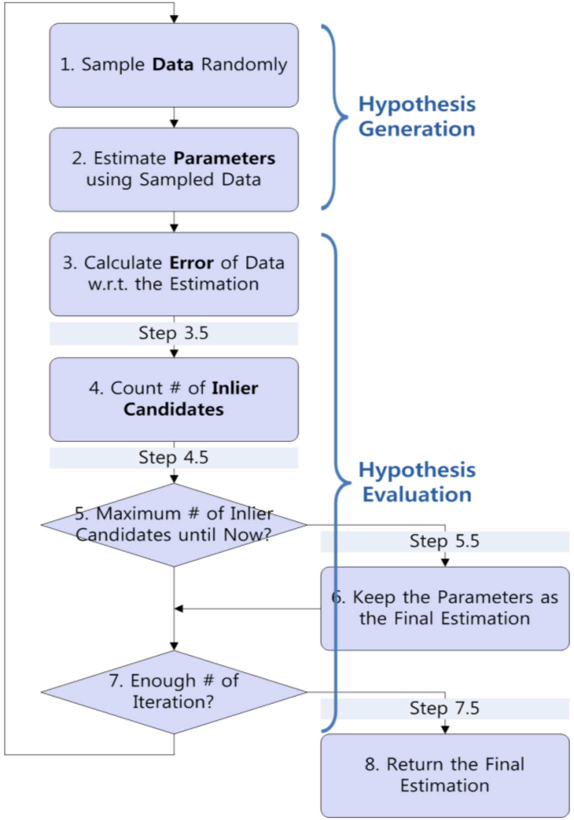


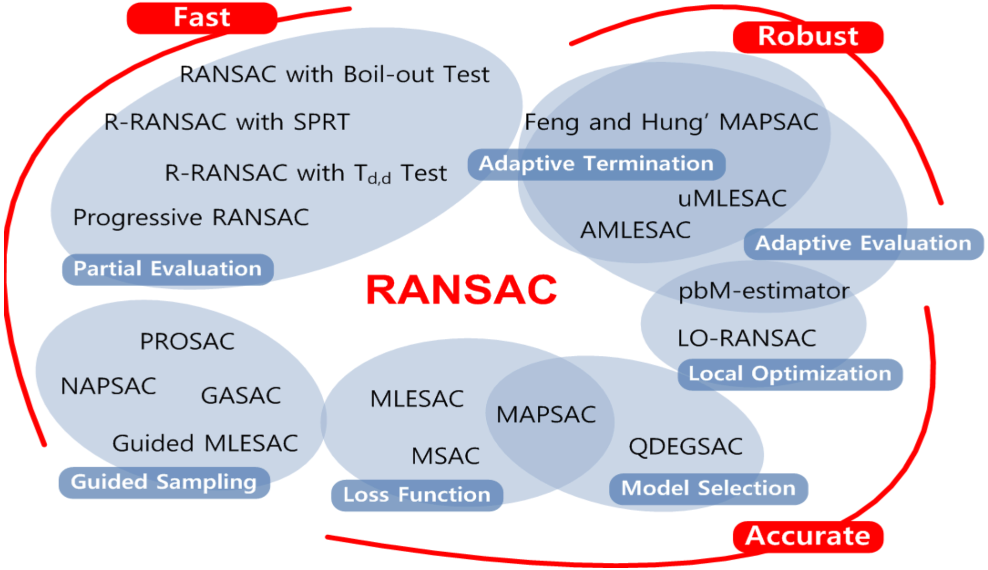


From "Performance Evaluation of RANSAC Family", by Choi et al., BMVC, 2009

RANSAC – Family

Outside the course scope





From "Performance Evaluation of RANSAC Family", by Choi et al., BMVC, 2009

RANSAC – Family – References

Outside the course scope

- Martin A. Fischler und Robert C. Bolles, “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Au Cartography”, Comm of the ACM, 1981
- D. Capel. “An effective bail-out test for RANSAC consensus scoring”, BMVC, 2005
- S. Choi and J.-H. Kim. “Robust regression to varying data distribution and its application to landmark-based localization”, IEEE SMC, 2008 [uMLESAC]
- O. Chum and J Matas. “Randomized RANSAC with $T_{d,d}$ test”, BMVC, 2002
- O. Chum and J. Matas. “Matching with PROSAC - Progressive Sample Consensus”, CVPR, 2005
- O. Chum and J. Matas. “Optimal randomized RANSAC”, TPAMI, 2008.
- O. Chum, J. Matas, and S. Obdrzalek. “Enhancing RANSAC by generalized model optimization”, ACCV, 2004.
- C.L. Feng and Y.S. Hung. “A robust method for estimating the fundamental matrix”. In Digital Image Computing: Techniques and Applications, 2003
- J.-M. Frahm and M. Pollefeys. “RANSAC for (quasi-)degenerate data (QDEGSAC)”, CVPR, 2006.
- A. Konouchine, V. Gaganov, and V. Veznevets. “AMLESAC: A new maximum likelihood robust estimator”, Int. Conf. on Computer Graphics and Vision (GrapiCon), 2005.
- J. Matas and O. Chum. “Randomized RANSAC with sequential probability ratio test”, ICCV, 2005
- D.R. Myatt, P.H.S Torr, S.J. Nasuto, J.M. Bishop, R. Craddock. “NAPSAC: High noise, high dimensional robust estimation - it’s in the bag”, BMVC, 2002
- D. Nister. “Preemptive RANSAC for live structure and motion estimation”, ICCV, 2003
- R. Raguram, J.-M. Frahm, and M. Pollefeys. “A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus”, ECCV, 2008
- V. Rodehorst and O. Hellwich. “Genetic Algorithm SAmple Consensus (GASAC)- a parallel strategy for robust parameter estimation”, CVPR Workshop, 2006
- R. Subbarao and P. Meer. “Beyond RANSAC: User independent robust regression”, CVPR Worksop, 2006
- B. J. Tordoff and D. W. Murray. “Guided-MLESAC: Faster image transform estimation by using matching priors”, TPAMI, 2005
- P.H.S. Torr and A. Zisserman. “MLESAC: A new robust estimator with application to estimating image geometry”, CVIU, 2000
- And many others

RANSAC – Parameters

$$p = 1 - (1 - w^n)^k$$

$$k = \frac{\log(1-p)}{\log(1-w^n)}$$

p → Reliability of RANSAC result

k → Number of iterations

w → Probability that a point is an inlier

n → Minimal number of samples for model (line, circle, projective transf., etc)

We stitch 2 images (spoiler: later today)
Find the number of RANSAC iterations for
- 95% reliability, and
- 20% inlier probability
- for SIFT corresp. (spoiler: n=4 corr.)

$$k = \frac{\log(1-p)}{\log(1-w^n)} = \frac{\log(1-0{,}95)}{\log(1-0{,}20^4)} = 1870{,}8344 \cong 1871$$

$p = 0{,}95$ ~ RANSAC reliability
$w = 0{,}20$ ~ inlier probability
$n = 4$ ~ we need 4 corresp. as minimum to stitch images with the projective transform.

https://en.wikipedia.org/wiki/Random_sample_consensus

RANSAC – Parameters

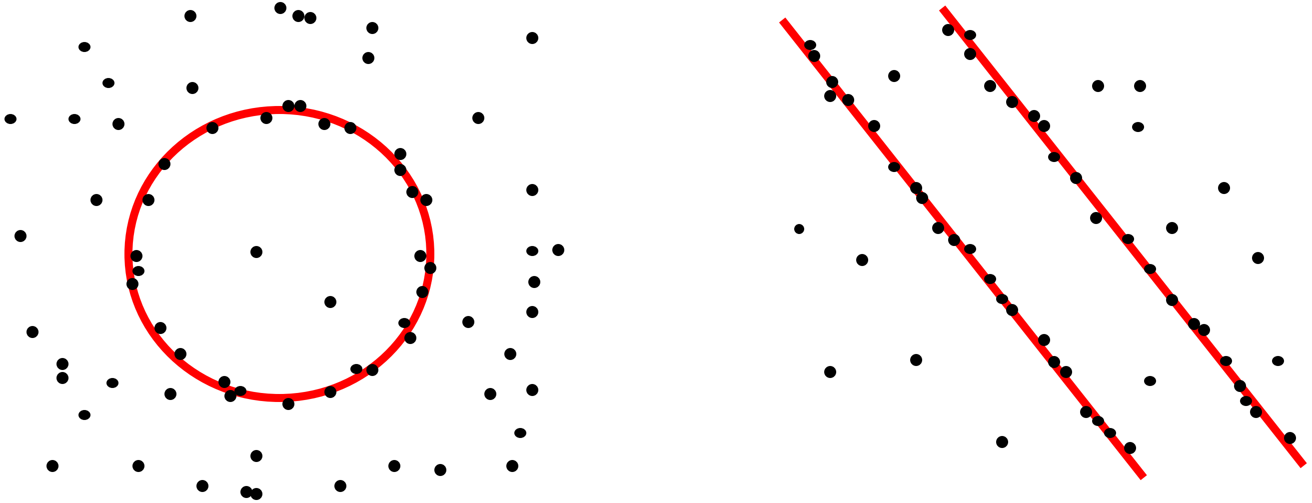
Sample size	proportion of outliers						
	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

*The **number of RANSAC iterations** required to ensure …*
*[input] … with a **probability 0.99**, and a given number of **minimal points** and a **proportion of outliers**,*
[output] that at least one sample has no outliers.

From book: [Multiple View Geometry, Hartley and Zisserman]

RANSAC

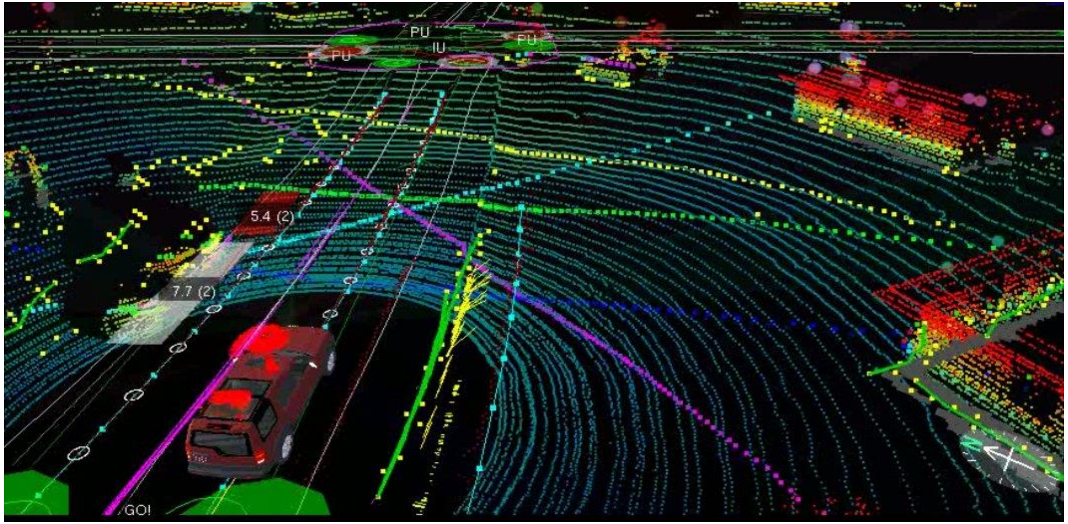
- Think about
  - Advantages & disadvantages of RANSAC
  - How to apply for detecting ellipses, planes, parallel lines, etc?



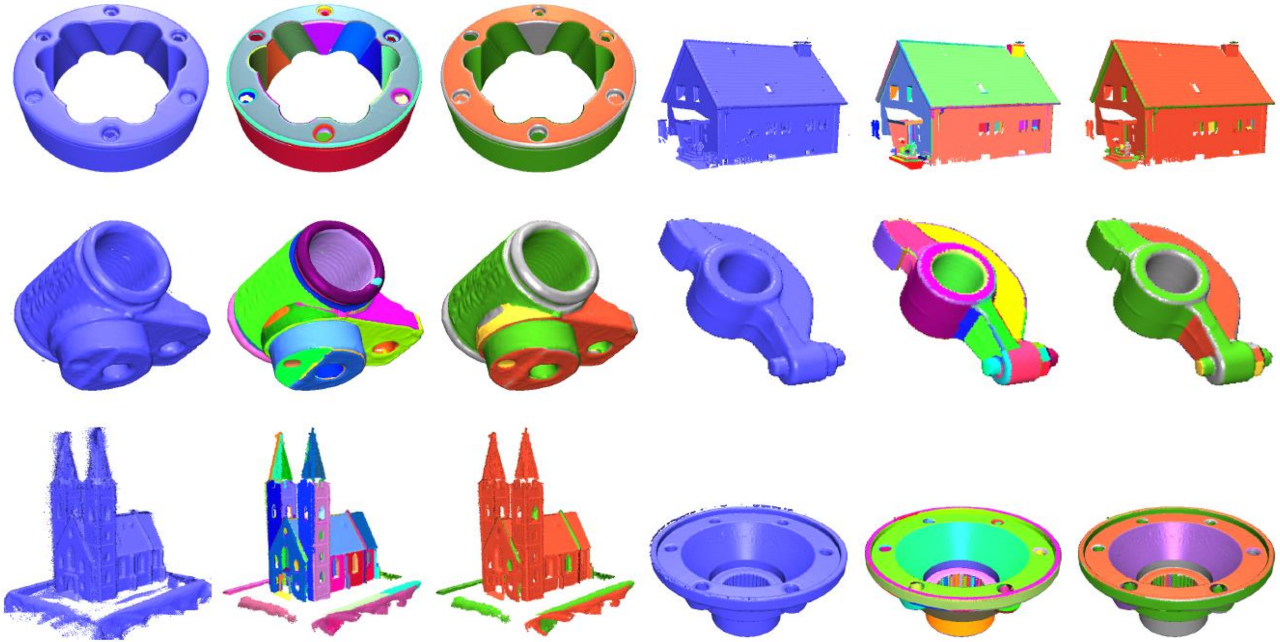
RANSAC – Applications

3D Ground plane extraction on LIDAR 3D point cloud

How many points?



RANSAC – Applications



Schnabel et al., “Efficient RANSAC for Point-Cloud Shape Detection”, Computer Graphics Forum, 2007

RANSAC – Pros & Cons

Advantages

- Robust estimation of model parameters
- Applicable for large number of parameters
- High degree of accuracy
- Even with outliers
- Easy to implement

Decent Trade-Offs!

Disadvantages

- No upper bound on runtime (so cap w. threshold)
- Accuracy may be suboptimal for low iter. Threshold
- Suboptimal when # of inliers <50% (so, check variants)
- Requires problem-specific thresholds
- Runtime grows quickly with
  - % of outliers and
  - number of parameters
- Not good for getting **multiple line fits**

Solution?

https://en.wikipedia.org/wiki/Random_sample_consensus

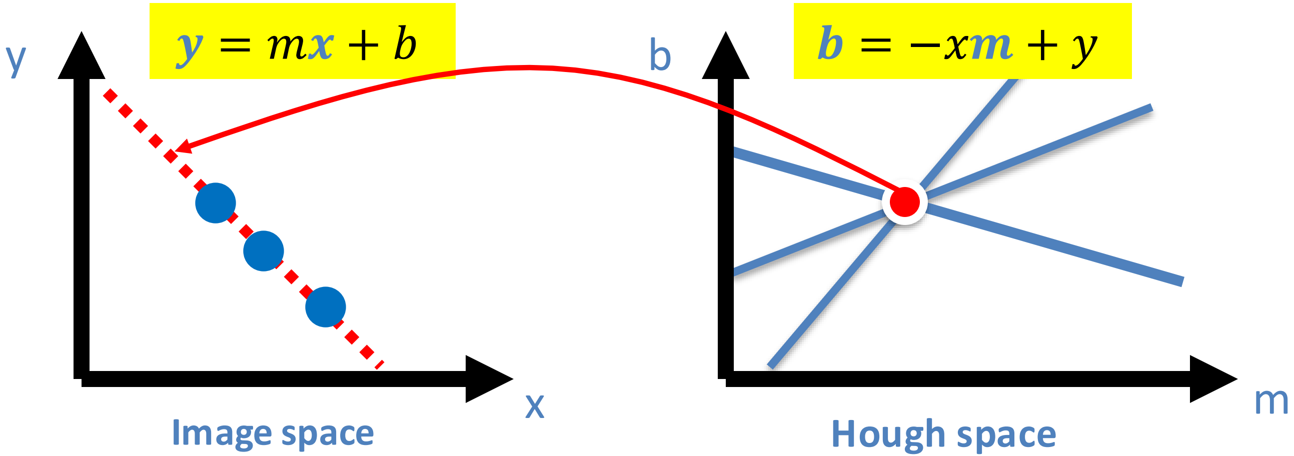
Outline

- Edge Detection
  - Derivatives of Image, Derivatives of Gaussian
  - Canny Edge Detector
- Line Fitting
  - Least Squares
  - RANSAC
  - Hough Transform
- Corners
  - Harris Corners
  - SIFT
  - Applications

Hough Transform – Fitting Multiple Lines

- Given a binary **edge** image
- Find the **lines** (or curves) that best explain the data points in parameter space
- Parameter space → Called '**Hough space**'

Toy figure
(values not accurate)



P.V.C. Hough, '*Machine Analysis of Bubble Chamber Pictures*', 1959

Point ↔	**Line**
Line ↔	**Point**

Slide from
S. Savarese

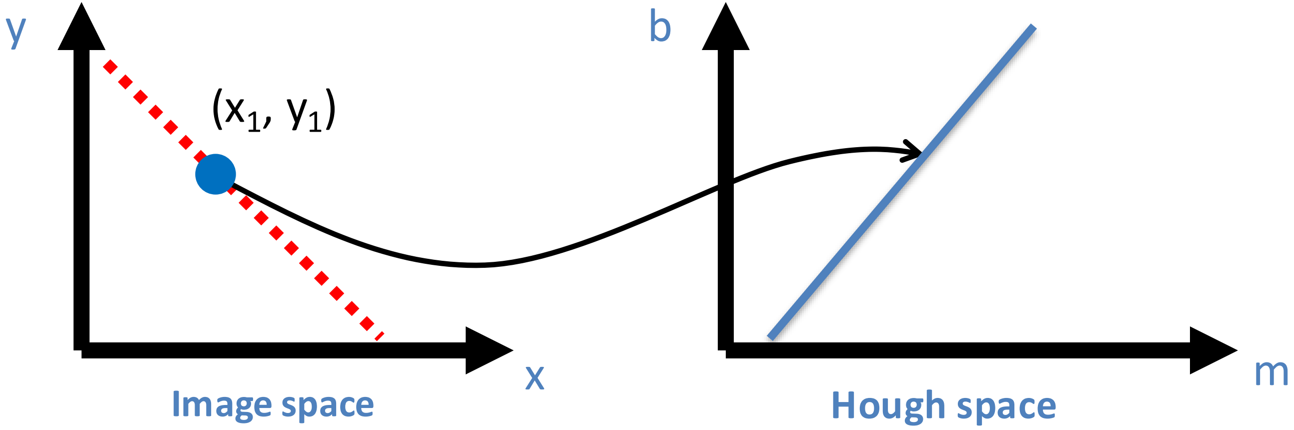
Map Point (x_1, y_1) → Line in Hough Space

$$y_1 = mx_1 + b$$

$$b = -x_1 m + y_1$$

Rewrite equation in terms of m and b

Toy figure
(values not accurate)



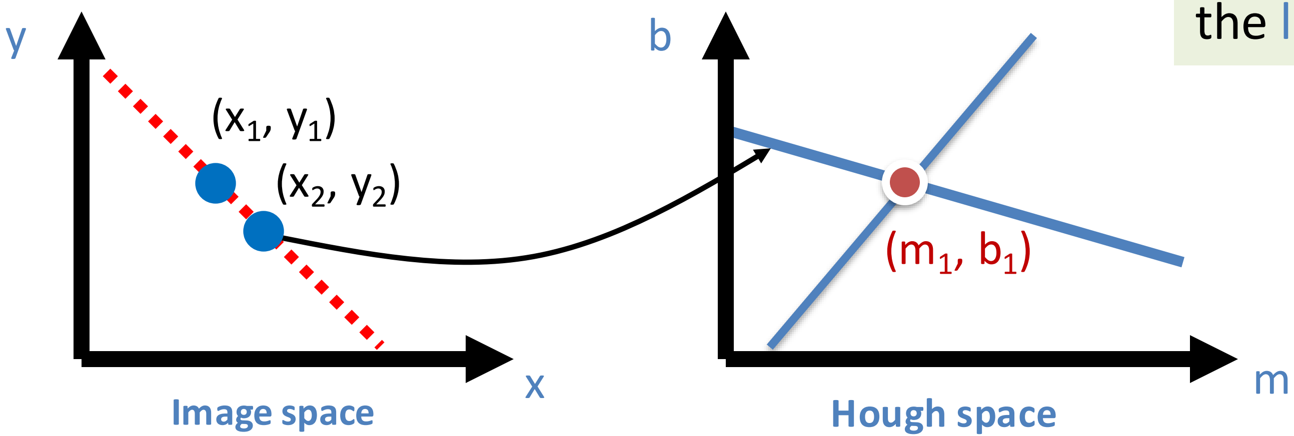
Map Point (x_2,y_2) → Line in Hough Space

$$y_2 = mx_2 + b$$

$$b = -x_2 m + y_2$$

Intersection point satisfies (x_1,y_1) & (x_2,y_2) & by extension the line

Toy figure
(values not accurate)



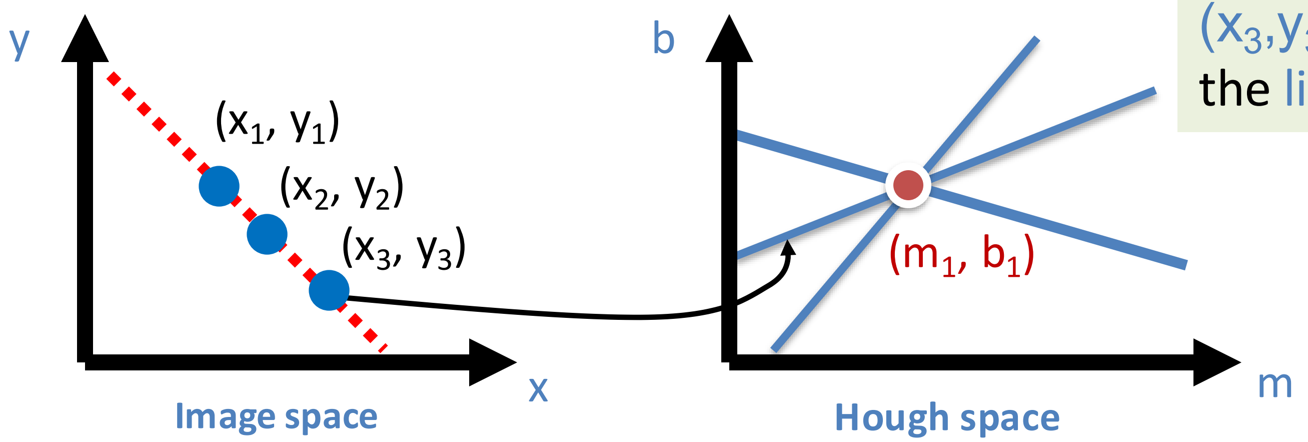
Map **Point** (x_3,y_3) → **Line** in **Hough Space**

$$y_3 = mx_3 + b$$

$$b = -x_3 m + y_3$$

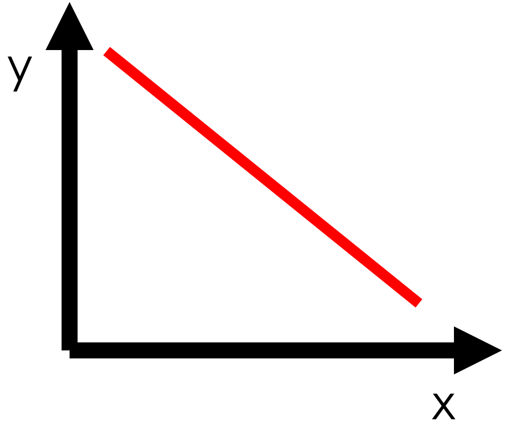
Intersection point (m_1,b_1) satisfies points (x_1,y_1), (x_2,y_2) (x_3,y_3) & thus the line

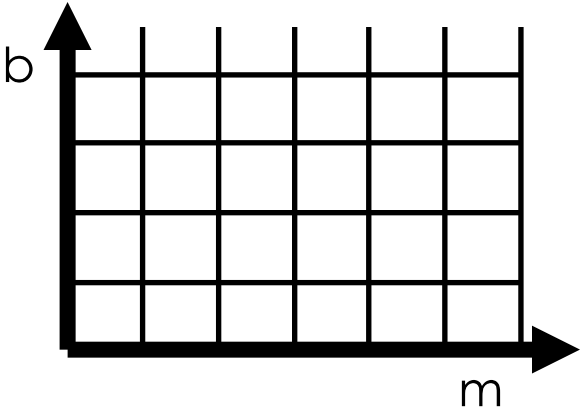
Toy figure
(values not accurate)



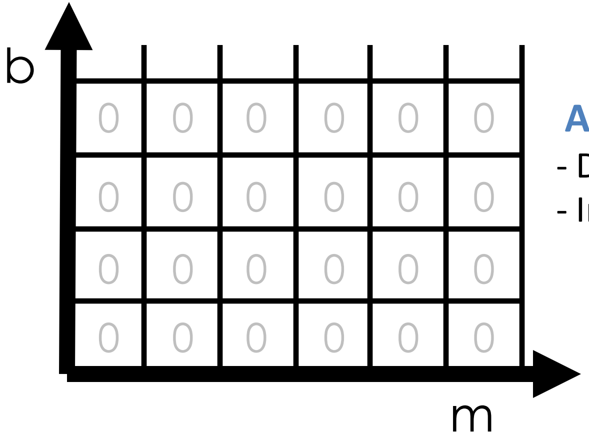
Hough Transform – Accumulator

Toy figure
(values not accurate)





- For each edgel → - Draw a line in Hough space
  - Add +1 in Accumulator cells

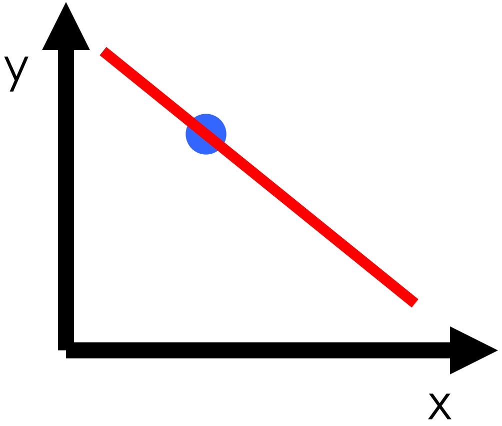


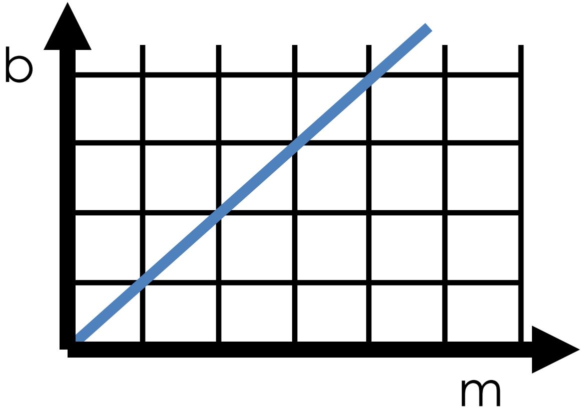
Accumulator
- Discrete grid
- Initialize w 0s

Slide from
S. Savarese

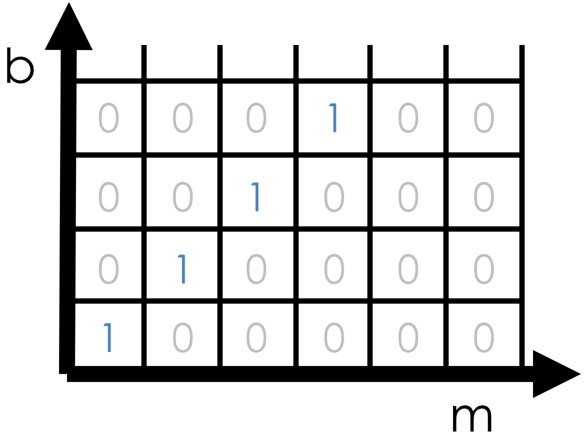
Hough Transform – Accumulator

Toy figure
(values not accurate)





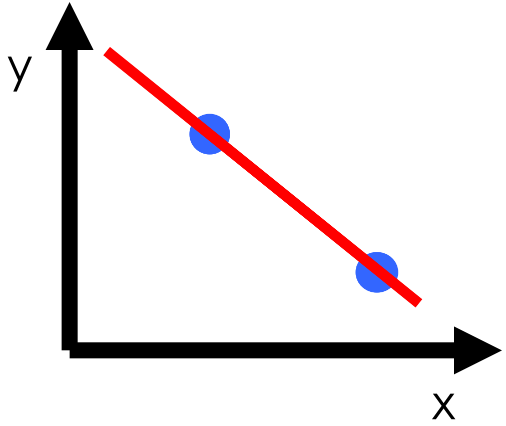
- For each edgel → - Draw a line in Hough space
  - Add +1 in Accumulator cells

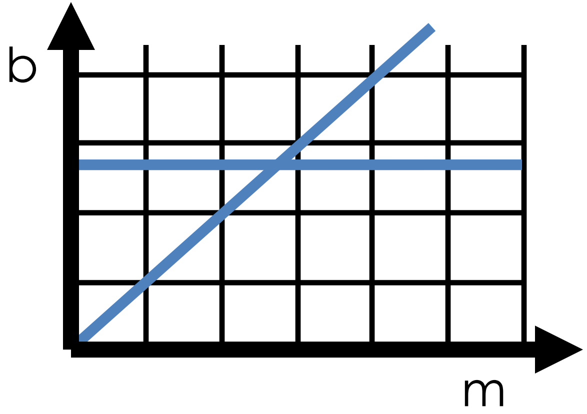


Slide from
S. Savarese

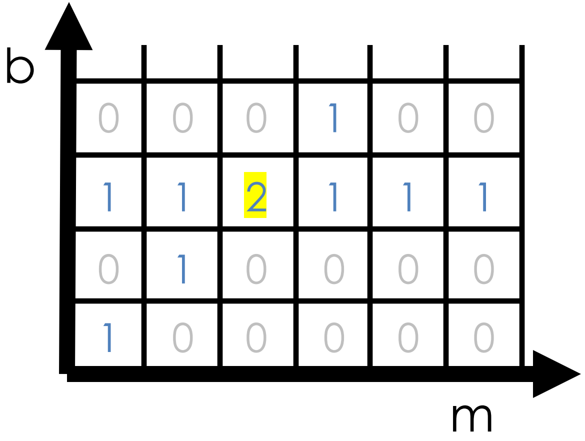
Hough Transform – Accumulator

Toy figure
(values not accurate)





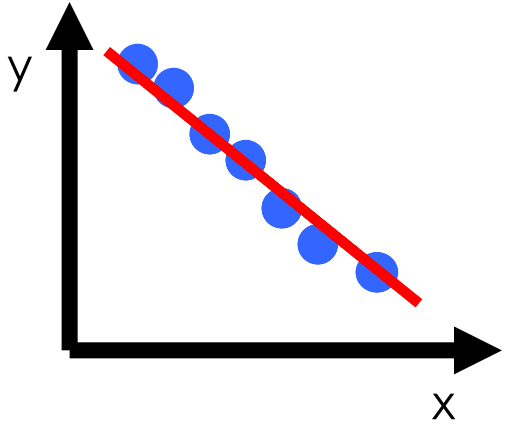
- For each edgel → - Draw a line in Hough space
  - Add +1 in Accumulator cells

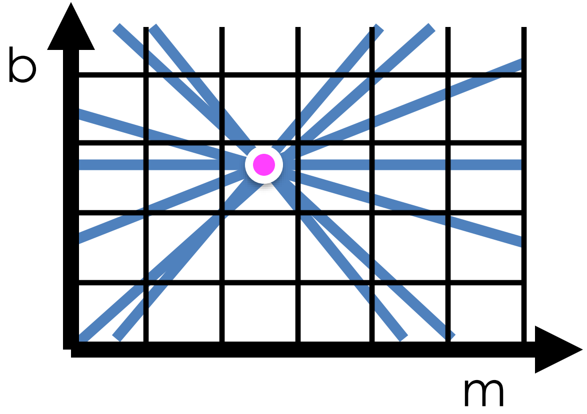


Slide from
S. Savarese

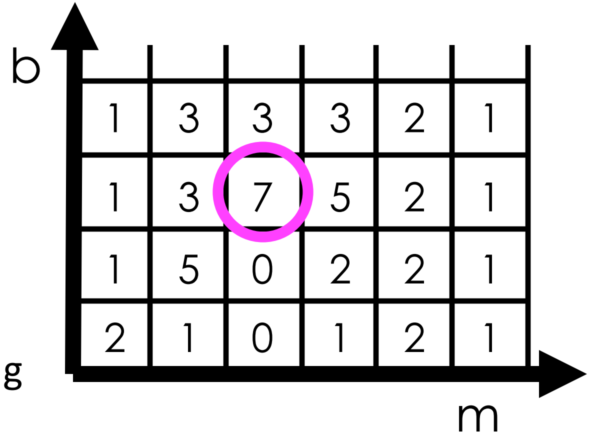
Hough Transform – Accumulator

Toy figure
(values not accurate)





- For each edgel → - Draw a line in Hough space
  - Add +1 in Accumulator cells
- Process all pixels → Accumulate values (‘voting’)
- Find local maxima → Can be many points
  due to many lines in image
- Map each maximum Point in Hough space → Line in Img



Slide from
S. Savarese

Hough Transform

- Detect multiple lines in image (from multiple local maxima)
- Easily extended for circles & ellipses
- Computationally efficient

- Problem: Parameters (m, b) are unbounded (slope m can be infinite)
  Need an infinitely large Accumulator (memory hungry)
- Solution? See next

Slide from
S. Savarese

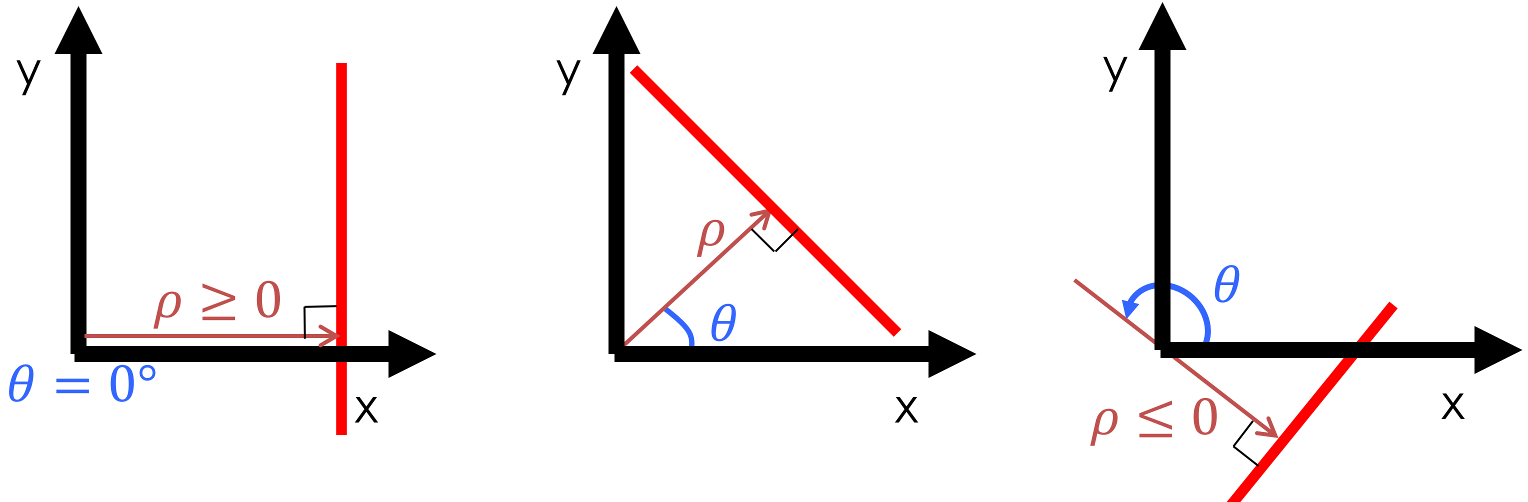
Hough Transform – Polar Form

Parameter space → **Polar** Representation

$$x\cos\theta + y\sin\theta = \rho$$

All parameters finite

$$\begin{cases} 0 \leq \theta < 180^{o} \\ \rho \text{ can be negative/positive} \end{cases}$$



Slide from
S. Savarese

Map Point (x_1, y_1) → Sinusoid in Hough Space

$$x_1 \cos\theta + y_1 \sin\theta = \rho$$

$$\begin{aligned} \rho &= x_1 \cos\boldsymbol{\theta} + y_1 \sin\boldsymbol{\theta} \\ &= \alpha_1 \sin(\boldsymbol{\theta} + \beta_1) \end{aligned}$$

Toy figure
(values not accurate)

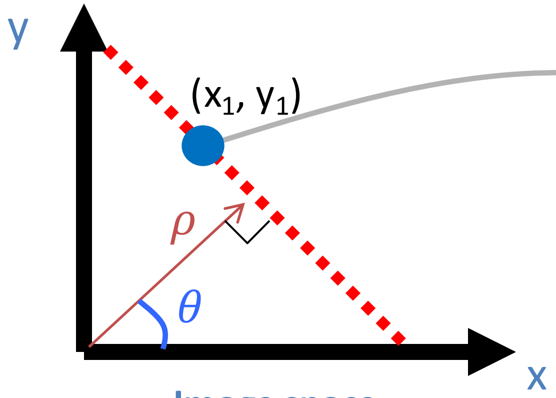
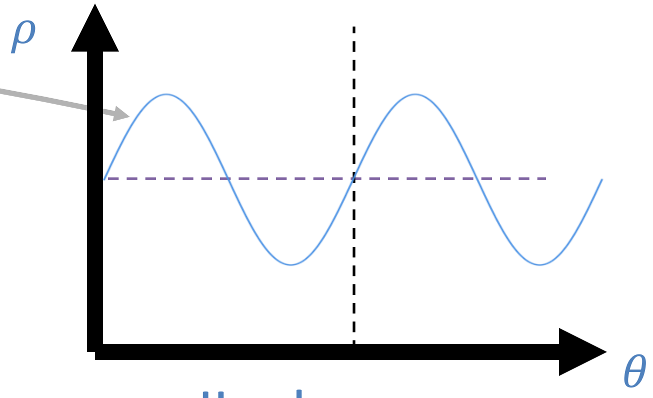


Image space



$0 \le \theta < 180^o$

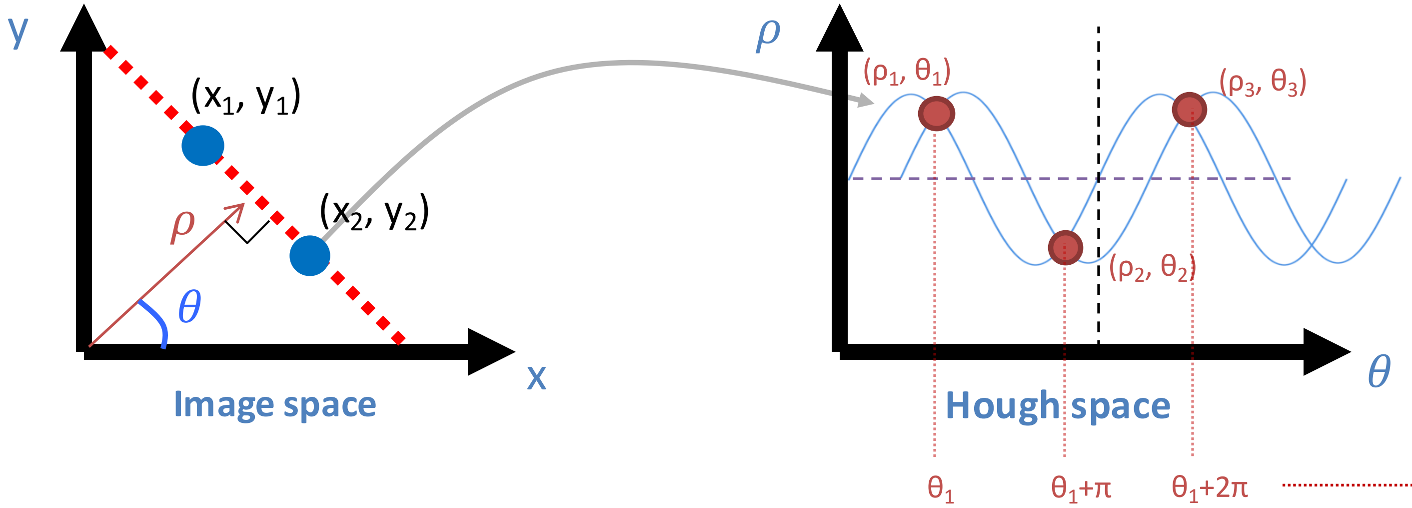
Hough space

Map Point (x_2, y_2) → Sinusoid in Hough Space

$$x_2 \cos\theta + y_2 \sin\theta = \rho$$

$$\begin{aligned}\rho &= x_2 \cos\theta + y_2 \sin\theta \\ &= \alpha_2 \sin(\theta + \beta_2)\end{aligned}$$

Toy figure
(values not accurate)



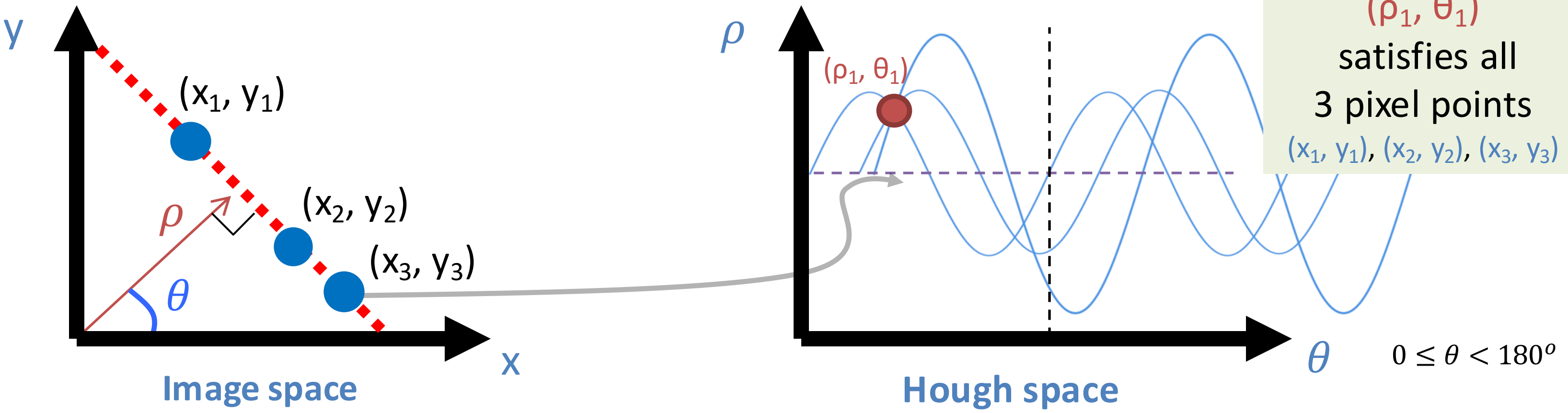
$0 \leq \theta < 180^o$

Map Point (x_3, y_3) → Sinusoid in Hough Space

$$x_3 \cos\theta + y_3 \sin\theta = \rho$$

$$\begin{aligned}\rho &= x_3 \cos\theta + y_3 \sin\theta \\ &= \alpha_3 \sin(\theta + \beta_3)\end{aligned}$$

Toy figure
(values not accurate)

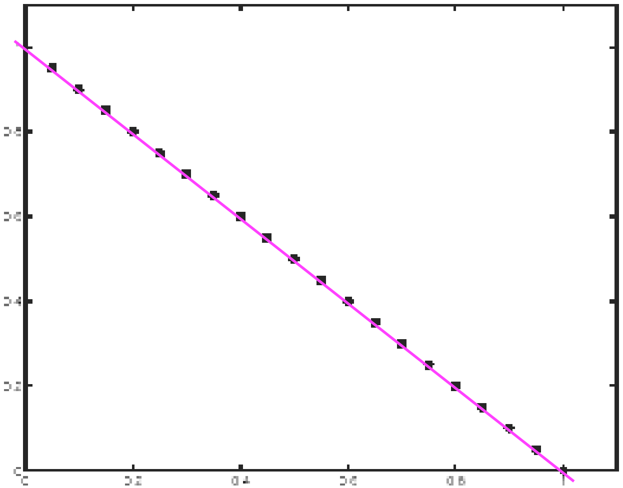


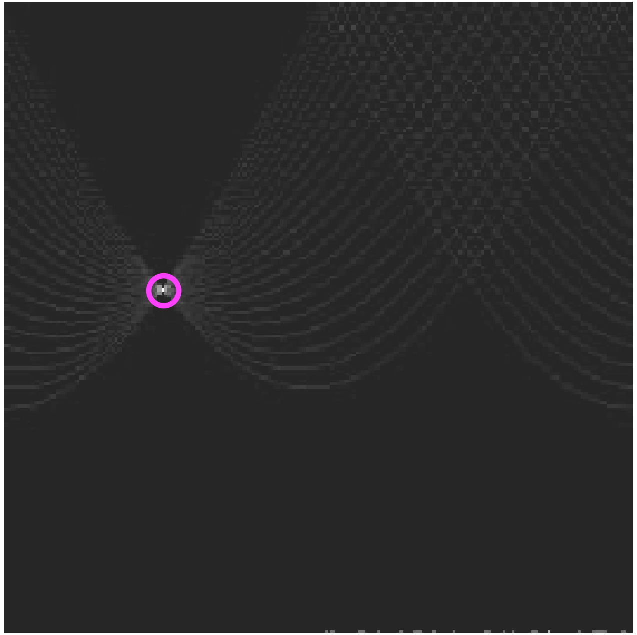
(ρ_1, θ_1) satisfies all 3 pixel points (x_1, y_1), (x_2, y_2), (x_3, y_3)

Hough Transform – Example

Clean data points → 'Clear' peak in Hough space

Data Points





Votes in **Hough** Space

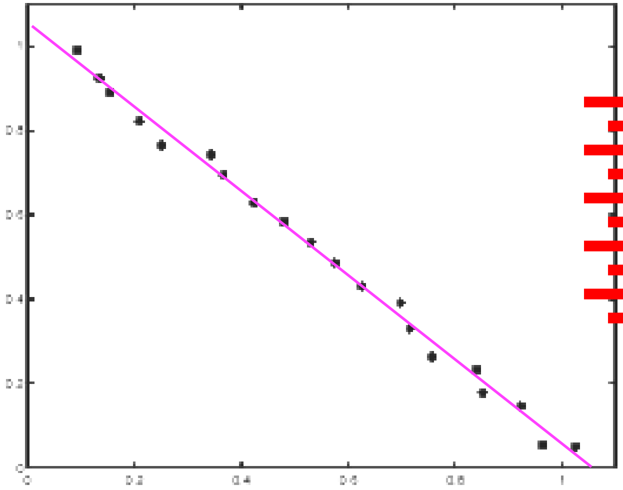
(regular grid)

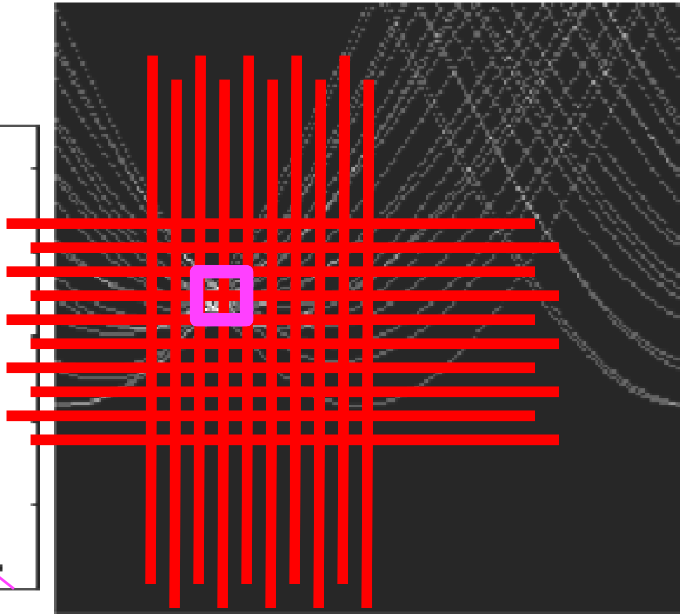
Slide from
S. Savarese

Hough Transform – Example

Noisy data points → ‘Fuzzy’ peak in Hough space → Adjust grid size (sparser)

Data Points





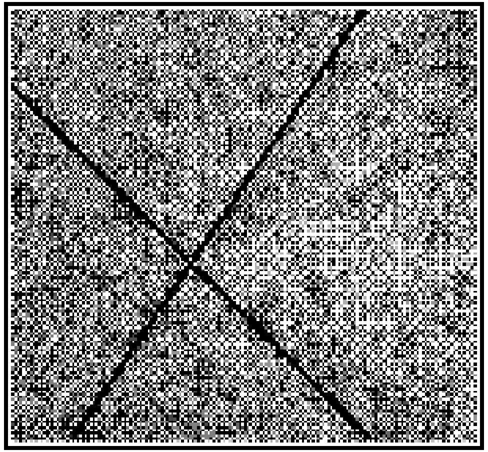
Votes in **Hough** Space
(regular grid)

Cell size – **Trade Offs**:

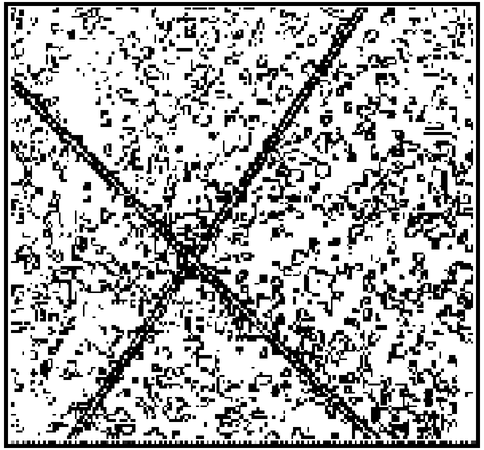
- **Too big**: Diff. lines merged
- **Too small**: Lines missed due to noise

Slide from
S. Savarese

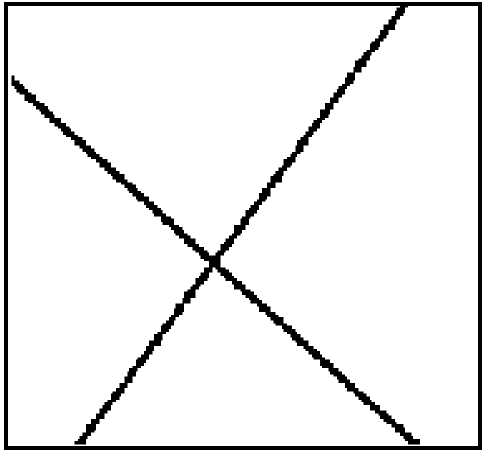
Hough Transform – Example



Image



Canny Edge Detection



Line Detection w Hough Transform

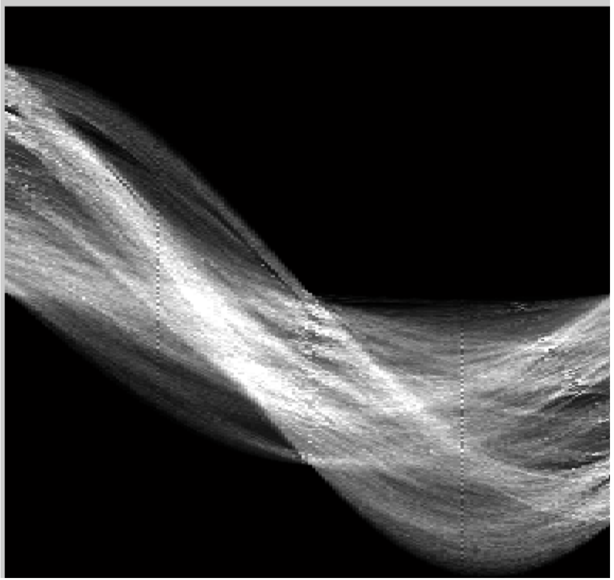
(Step 1/3) Image → Canny Edges

Image

Canny Edge Detection

(Step 2/3) Canny Edges → Hough Votes

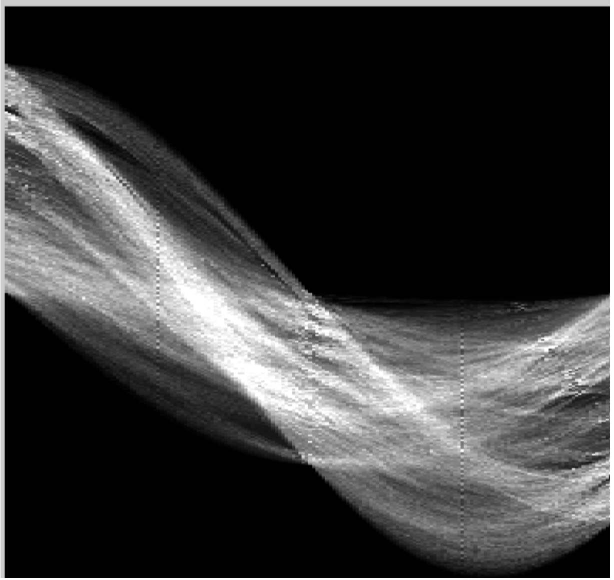
Canny Edge
Detection



Accumulator
for Hough space

(Step 3/3) Hough Votes → Lines

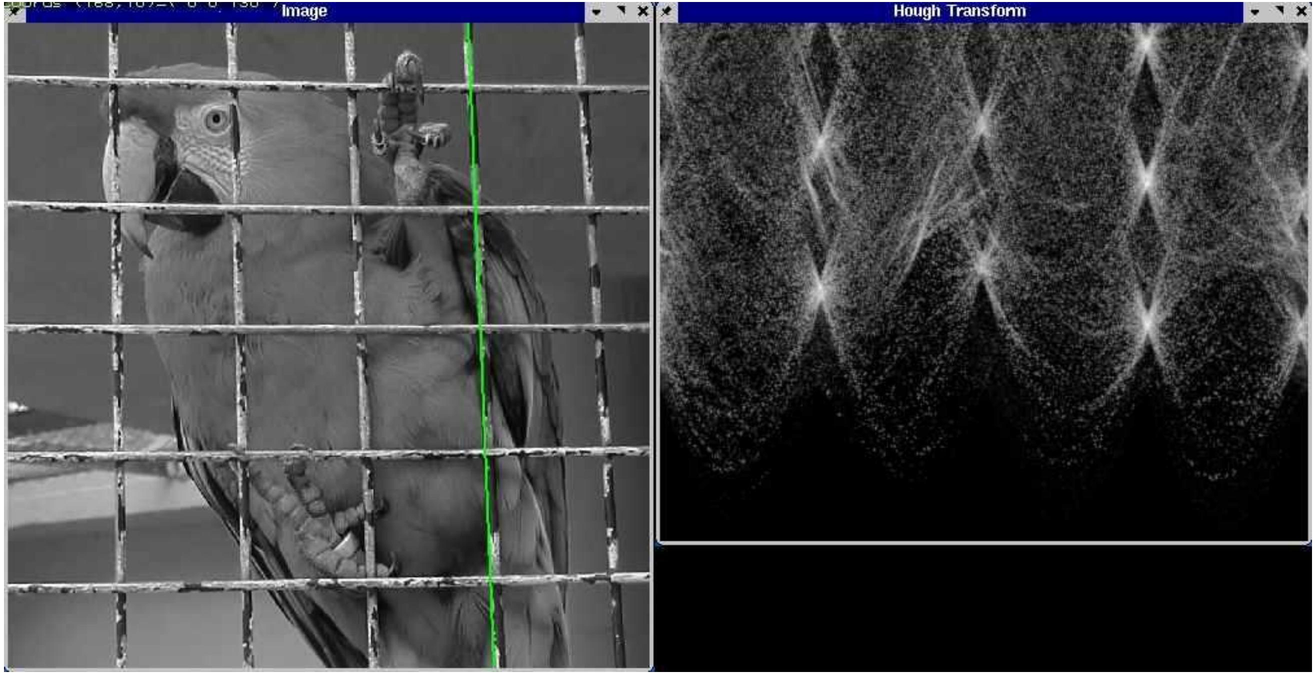
Line Detections



Accumulator
for Hough space

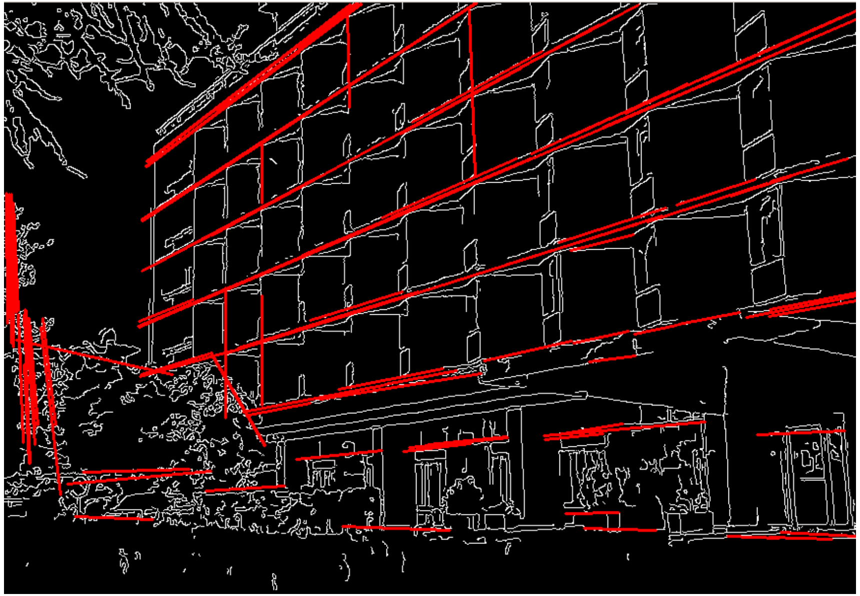
- Find peaks in Hough space
- Convert these to lines in pixel space

Hough Transform – Example



Legacy image – link broken: ~~http://ostatic.com/files/images/ss_hough.jpg~~

Hough Transform – Example



Probabilistic Hough Transform

(OpenCV)

Hough Transform – Pros & Cons

Pros ☺

- All points processed independently → Can cope with occlusion
- Some noise robustness (outliers unlikely to contribute consistently to single bin)
- Can detect multiple instances of a model in a single pass

Cons ☹

- Search-time complexity increases exponentially with # of model parameters
- Non-target shapes can produce spurious peaks in parameter space
- Quantization – Difficult to pick a 'good' grid size

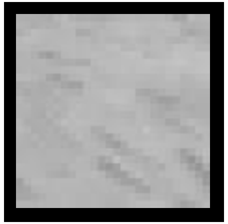
Hough Transform – Extension for Circles

https://learnopencv.com/hough-transform-with-opencv-c-python

Outline

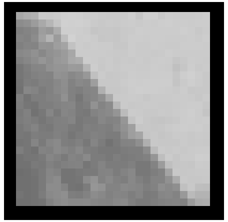
- Edge Detection
  - Derivatives of Image, Derivatives of Gaussian
  - Canny Edge Detector
- Line Fitting
  - Least Squares
  - RANSAC
  - Hough Transform
- Corners
  - Harris Corners
  - SIFT
  - Applications

Interest Points



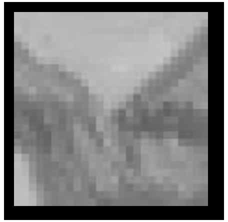
0D structure: ‘Flat’ areas

→ not useful for matching ☹



1D structure: Lines / Edges

→ edge, can be localized in 1D
subject to the aperture problem ☹



2D structure: **Corners / Interest Points**

→ can be localized in 2D
good for matching ☺

Interest Points have **2D** structure

Why Extract Features?

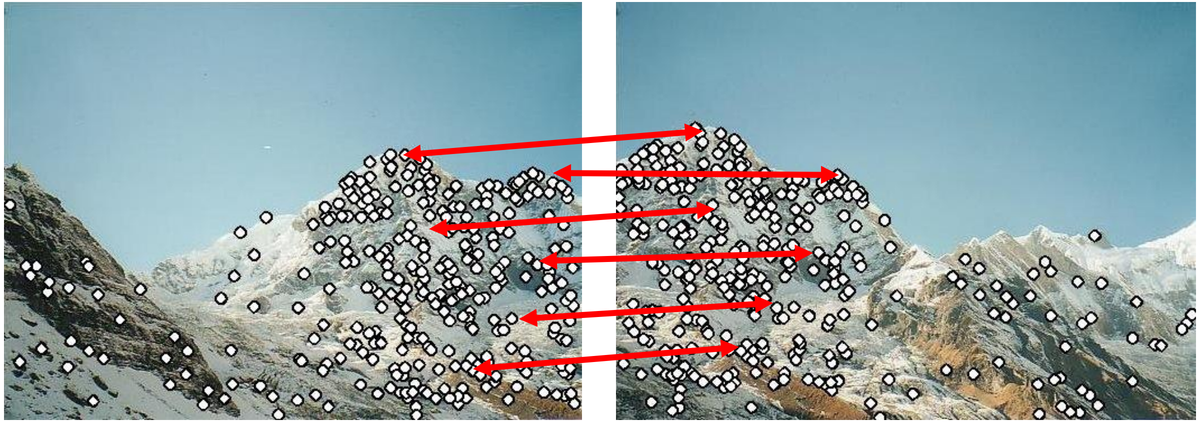
- Motivation → Panorama Stitching
  - We have 2 images
  - How to combine them?

Why Extract Features?

- Motivation → Panorama Stitching
  - We have 2 images
  - How to combine them?

Step 1 → Extract Features

Step 2 → Match Features



Why Extract Features?

- Motivation → Panorama Stitching
  - We have 2 images
  - How to combine them?

Step 1 → Extract Features

Step 2 → Match Features

Step 3 → Align Images

Feature Matching

Outline

- Edge Detection
  - Derivatives of Image, Derivatives of Gaussian
  - Canny Edge Detector
- Line Fitting
  - Least Squares
  - RANSAC
  - Hough Transform
- Corners
  - Harris Corners
  - SIFT
  - Applications

Harris Corner Detector – Mathematics

Change of intensity
for shift $[u, v]$

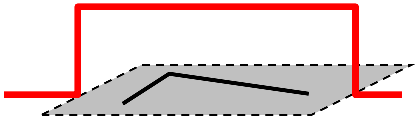
$$E(u,v) = \sum_{x,y} w(x,y)\ [I(x+u, y+v) - I(x,y)]^2$$

Window function

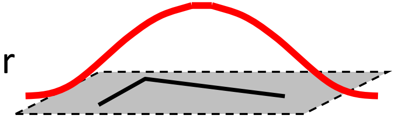
Shifted intensity

Intensity

Window function $w(x,y) =$



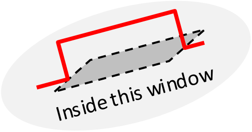
or



1 → in window
0 → outside

Gaussian

Harris Corner Detector – Mathematics



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

$$= \sum_{x,y} \left[I(x,y) - I_x u - I_y v + \varepsilon - I(x,y)\right]^2$$

$$\approx \sum_{x,y} \left[-I_x u - I_y v\right]^2$$

$$= \sum_{x,y} I_x^2 u^2 + I_x u I_y v + I_y v I_x u + I_y^2 v^2$$

$$= \sum_{x,y} [u \quad v] \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

Approximate Hessian
Encodes image 'curvature'
Product of 1st order derivatives instead of 2nd order derivatives

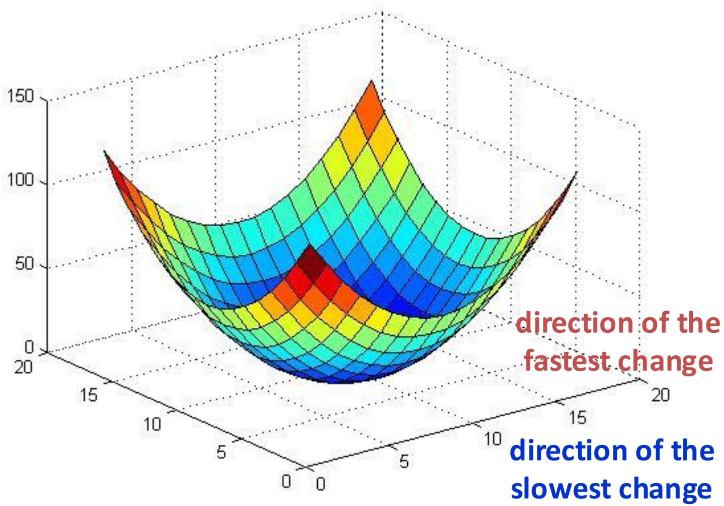
$$E(u,v) \cong [u, v]\, M \begin{bmatrix} u \\ v \end{bmatrix}$$

$$M = \sum_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}$$

M depends on img properties

Quadratic Form & Eigenvalues

$$E(u,v) \cong [u,v]\ M \begin{bmatrix} u \\ v \end{bmatrix}$$

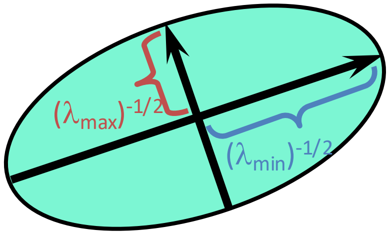


Measure of corner response:

$$R = \det M - k(\operatorname{trace} M)^2$$

$$\det M = \lambda_1 \lambda_2$$

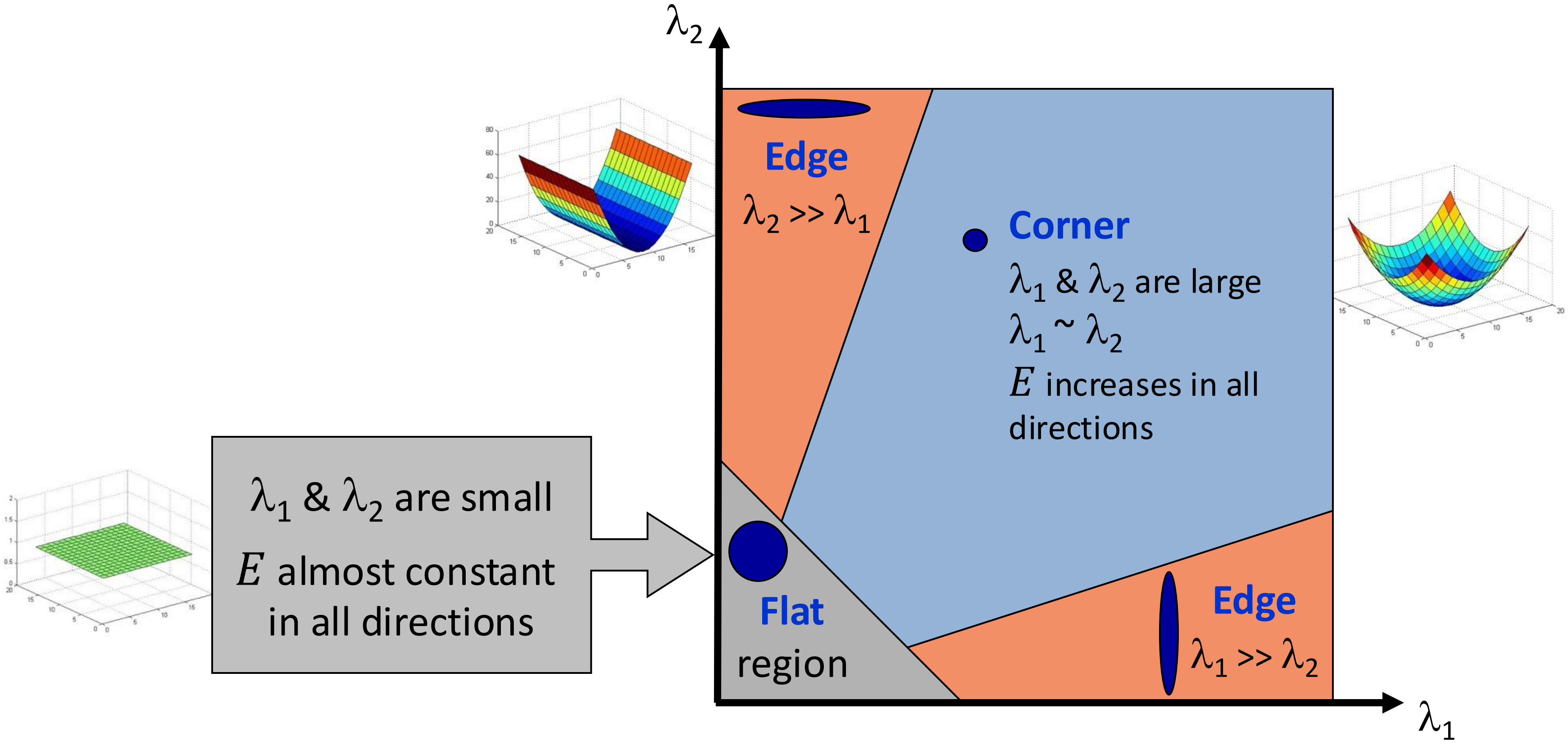
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$



(k → empirical constant)

(k = 0.04 – 0.06)

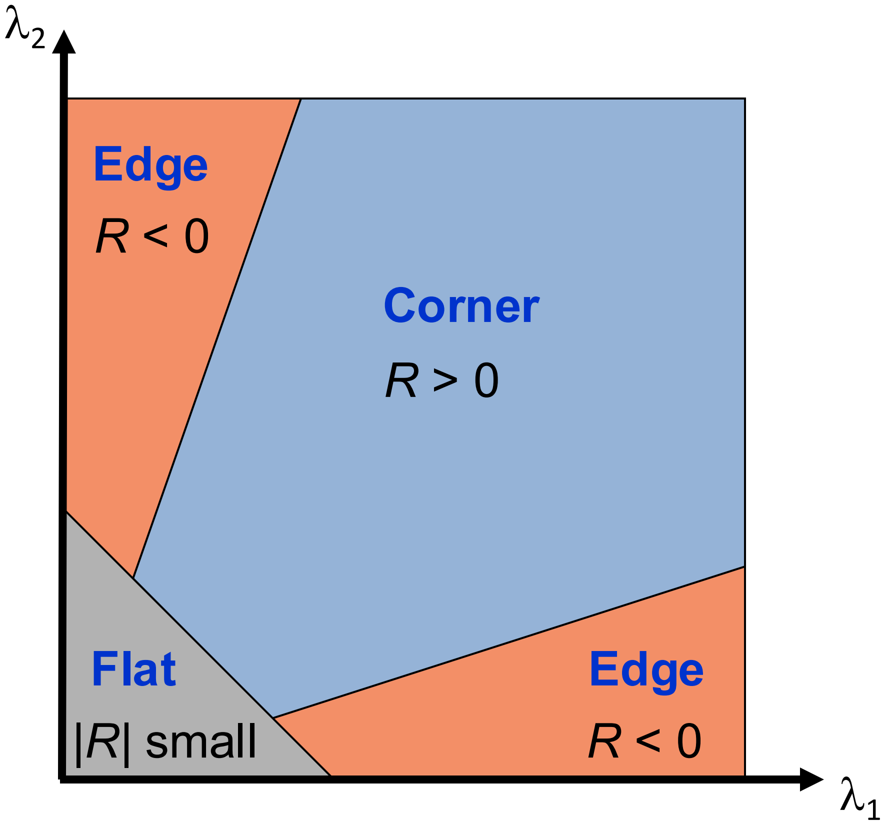
Eigenvalues of M – Classify Image Points



Harris Detector

$$R = \det M - k(\operatorname{trace} M)^2$$

- R depends only on eigenvalues of M
- R is large for a corner
- R is negative with large magnitude for an edge
- $|R|$ is small for a flat region



Harris Detector – Summary

1. Compute x and y derivatives of image

$$I_x = G_\sigma^x * I \quad I_y = G_\sigma^y * I$$

2. Compute products of derivatives at every pixel

$$I_{x2} = I_x.I_x \quad I_{y2} = I_y.I_y \quad I_{xy} = I_x.I_y$$

3. Compute the sums of the products of derivatives at each pixel

$$S_{x2} = G_{\sigma\prime} * I_{x2} \quad S_{y2} = G_{\sigma\prime} * I_{y2} \quad S_{xy} = G_{\sigma\prime} * I_{xy}$$

4. Define at each pixel (x, y) the matrix

$$H(x,y) = \begin{bmatrix} S_{x2}(x,y) & S_{xy}(x,y) \\ S_{xy}(x,y) & S_{y2}(x,y) \end{bmatrix}$$

5. Compute the response of the detector at each pixel

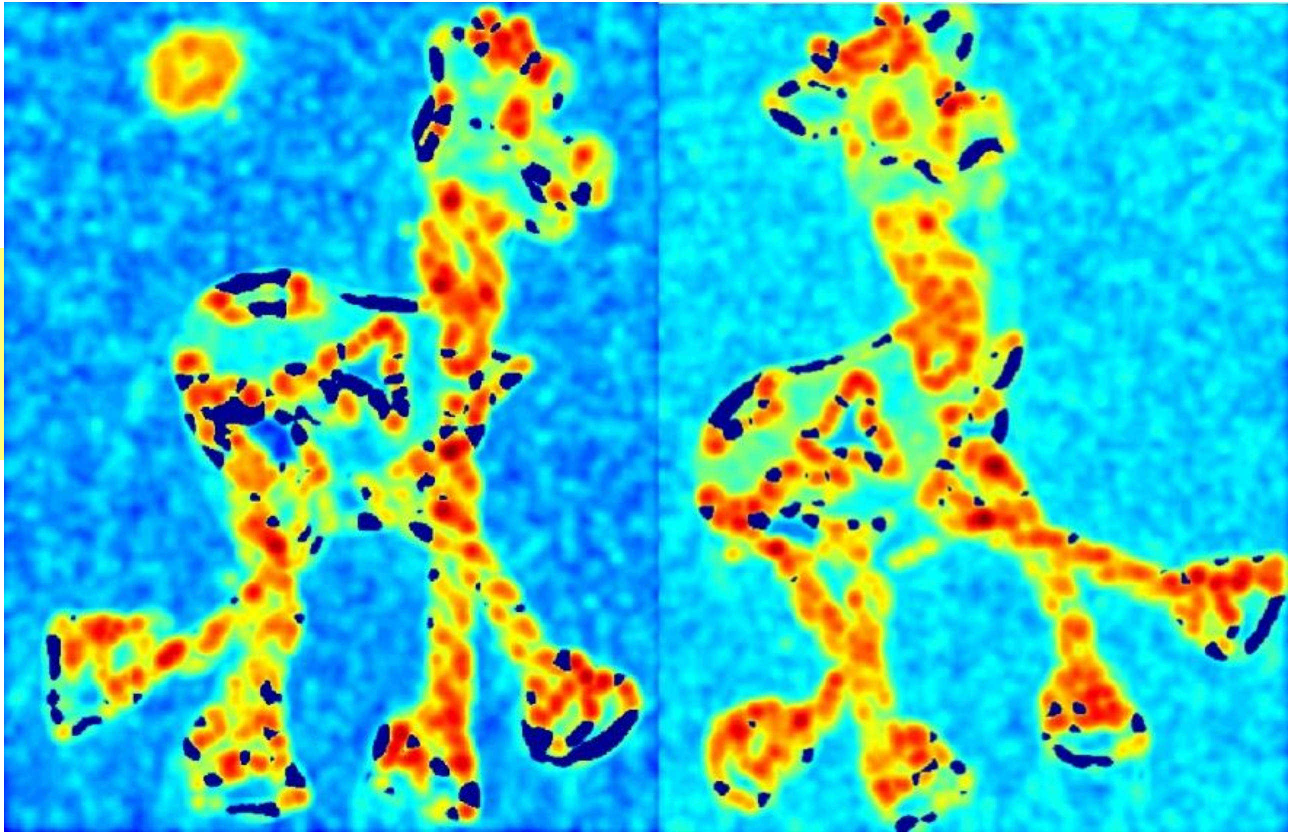
$$R = Det(H) - k(Trace(H))^2$$

6. Threshold on value of R. Compute nonmax suppression

Harris Detector – Workflow

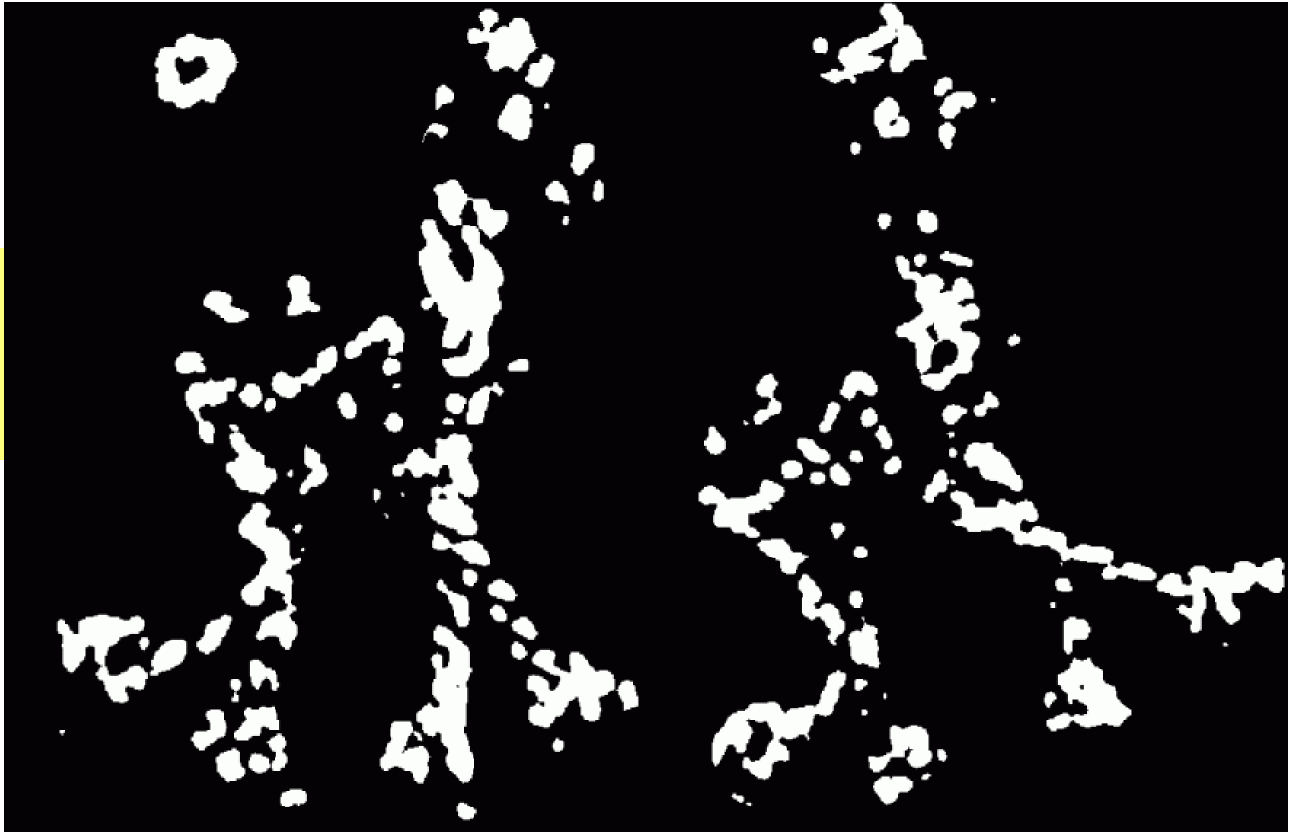
Harris Detector – Workflow

Compute
corner response
R



Harris Detector – Workflow

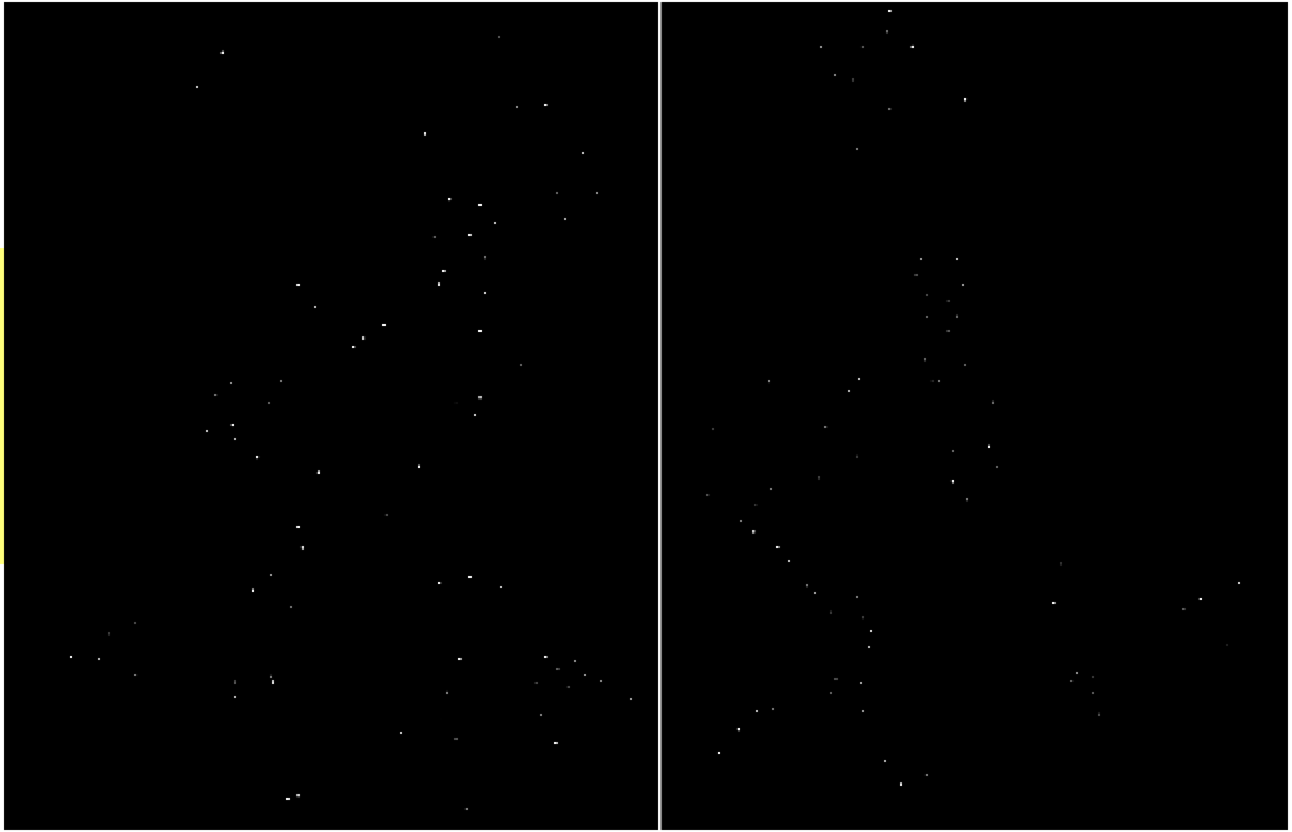
Points w. large corner response $R >$ threshold



Harris Detector – Workflow

Take only points at local maxima of R

(non-maxima suppression)



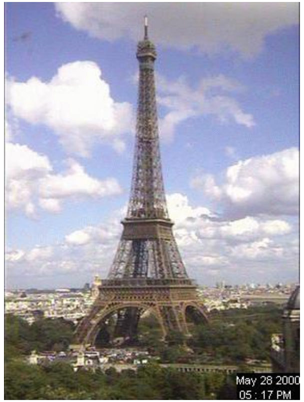
Harris Detector – Workflow

Outline

- Edge Detection
  - Derivatives of Image, Derivatives of Gaussian
  - Canny Edge Detector
- Line Fitting
  - Least Squares
  - RANSAC
  - Hough Transform
- Corners
  - Harris Corners
  - SIFT
  - Applications

Interest Points & Image Scale

Can we detect the same interest points despite changes in image scale?



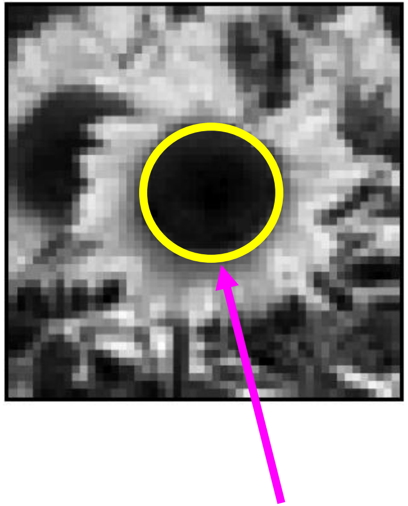
Blob Detection in 2D

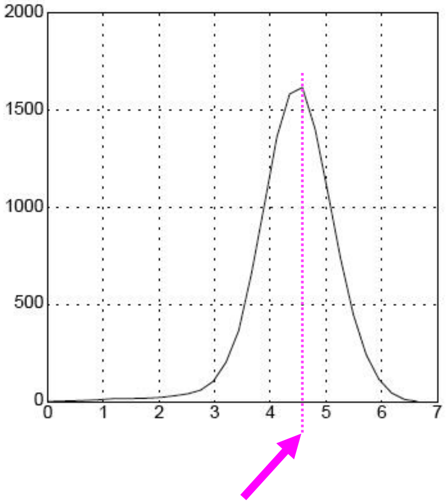
Characteristic Scale → The scale that produces peak of Laplacian response

Nugget:

- Convolve image with Gaussians of varying scale
- Detect max responses across scale space

A narrower or wider Gaussian → would have a weaker response for this flower





Characteristic Scale

Slide credit:
Lana Lazebnik

Example

¾ the
Original Size

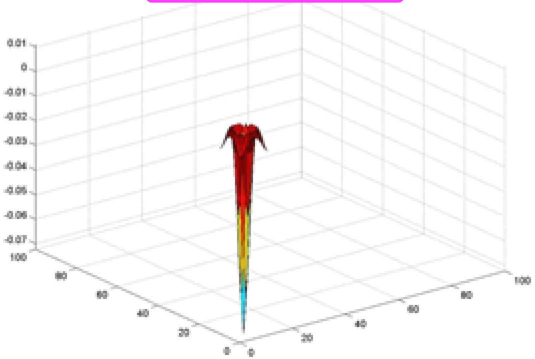
Original
Size

Slide by
Kristen Grauman

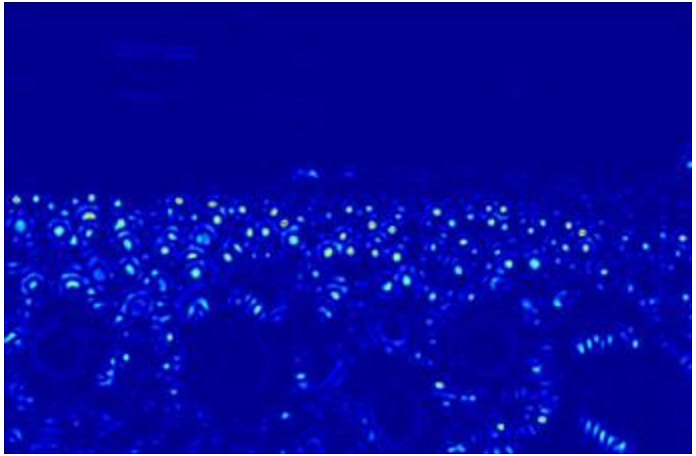
Example

Image at ¾ the original size →

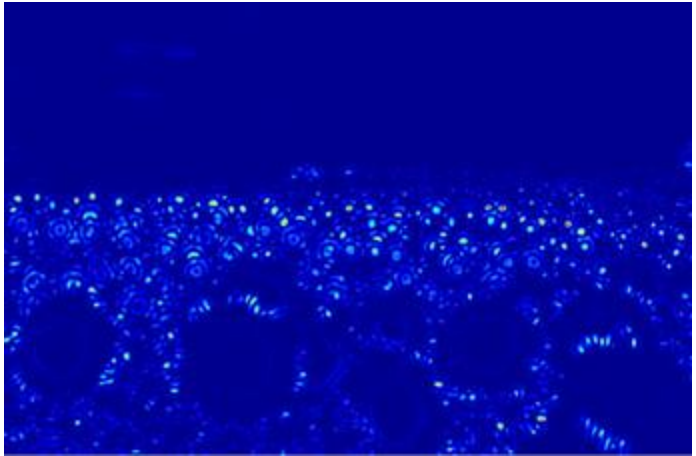
$\boxed{\sigma = 2.1}$



¾ the Original Size



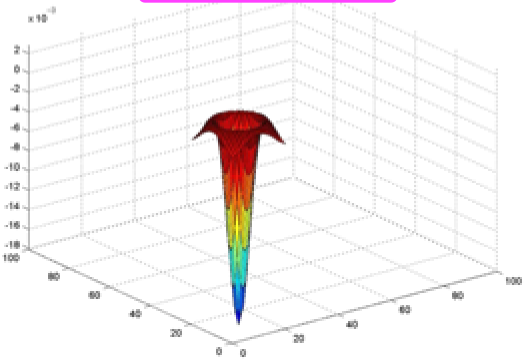
Original Size



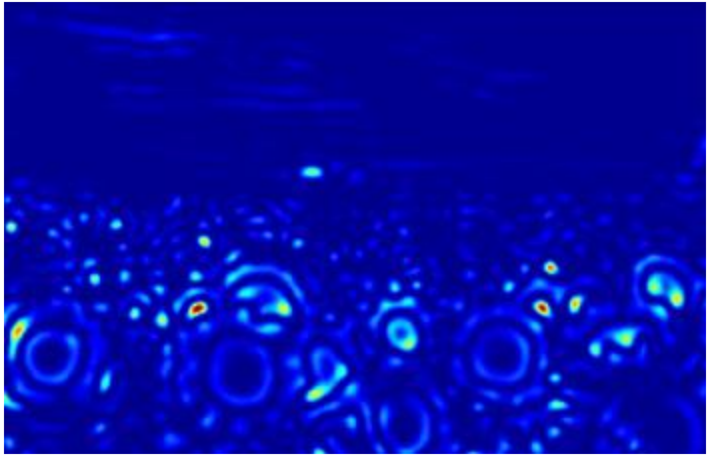
Slide by Kristen Grauman

Example

$\sigma = 4.2$



¾ the
Original Size

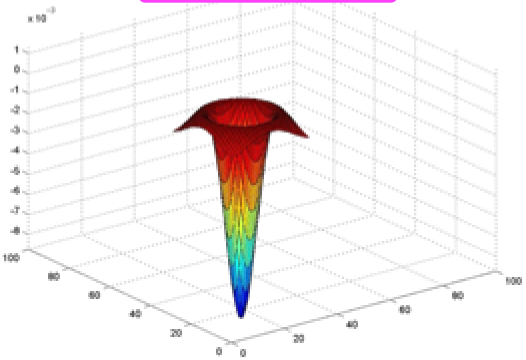


Original
Size

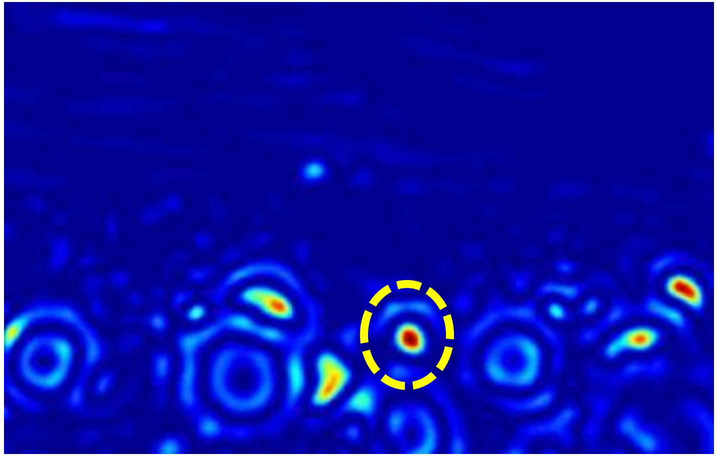
Slide by
Kristen Grauman

Example

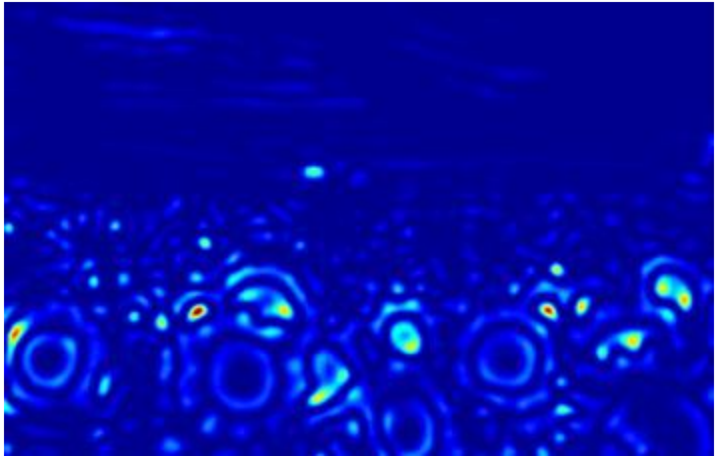
$\sigma = 6.0$



¾ the
Original Size



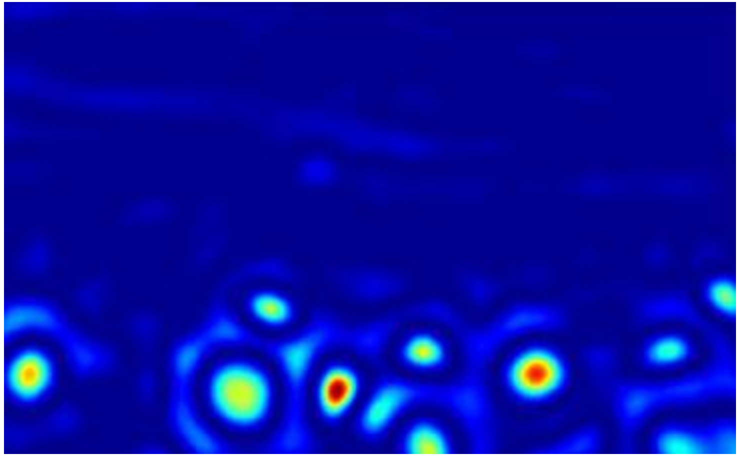
Original
Size



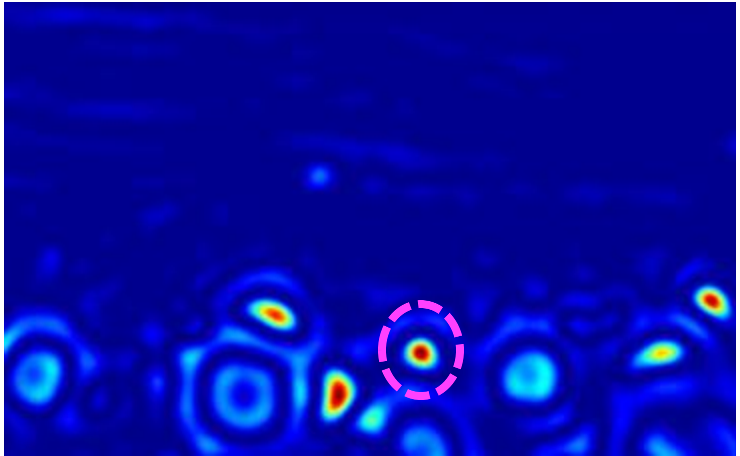
Slide by
Kristen Grauman

Example

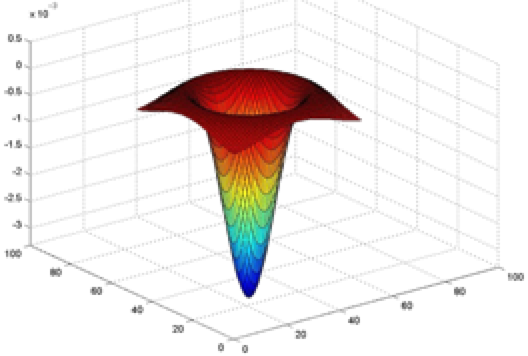
¾ the
Original Size



Original
Size



σ = 9.8

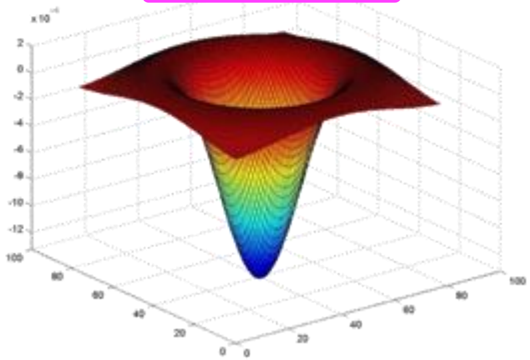


Slide by
Kristen Grauman

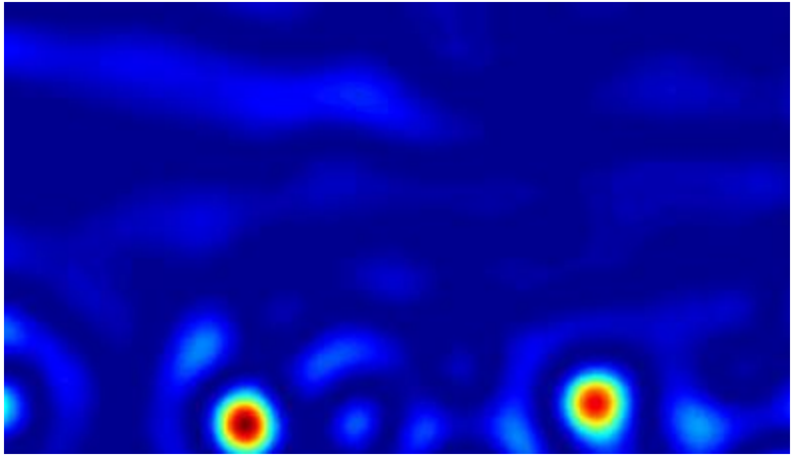
Example

A LoG of increasing sigma produces stronger activations for bigger-looking sunflowers (lying closer to the camera)

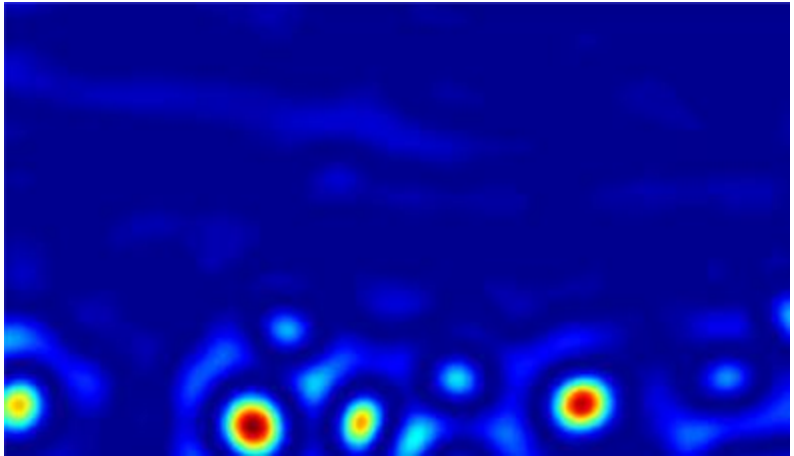
σ = 15.5



¾ the Original Size



Original Size

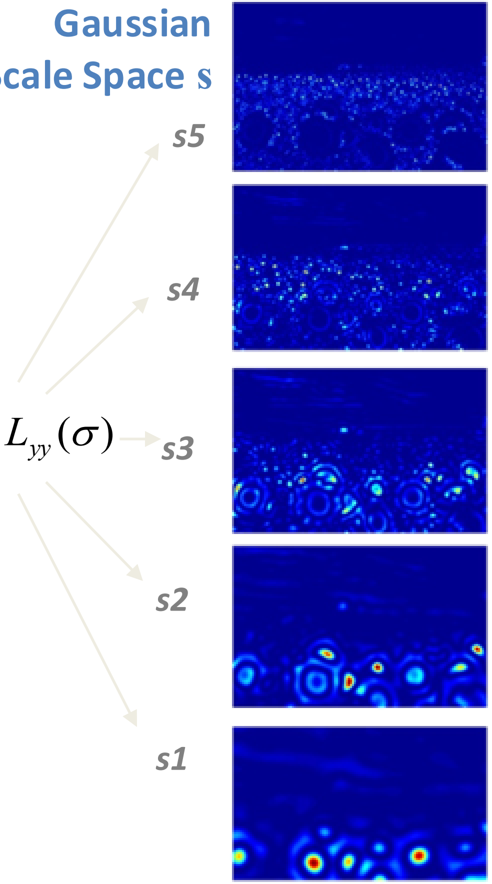


Slide by Kristen Grauman

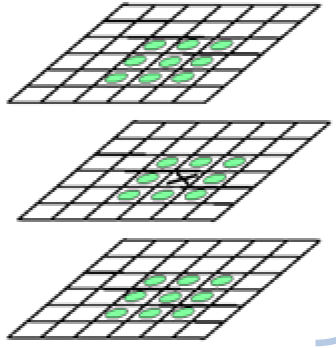
SIFT Keypoints

$$L_{xx}(\sigma)+L_{yy}(\sigma)$$

Gaussian Scale Space s



Difference of Gaussians (DoG)
Pyramid



Keypoints:
local **extrema** in
3x3x3 grid i.e. in both:

- Spatial space (x,y)
- Scale space (s)

Keypoints are **Scale Invariant**

Convolution with Gaussian is rotation invariant

Keypoints are **Rot. Invariant**

SIFT – Scale Invariant Feature Transform

Histogram computed in Gaussian-weighted neighborhood

Contributes to **Rot. Invariant** for keypoints

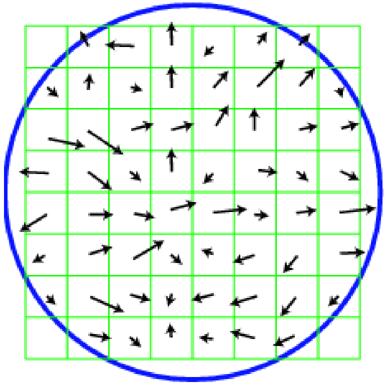
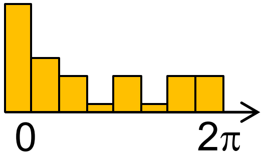
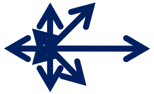


Image gradients



Gradient angle histogram

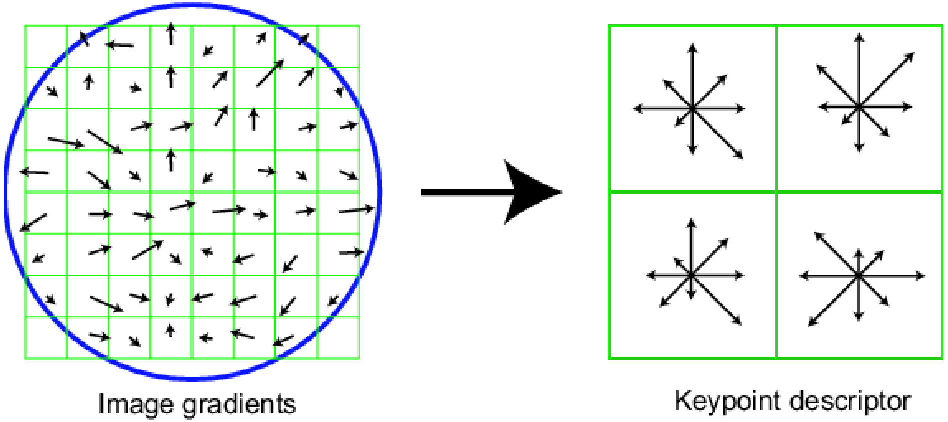


Basic idea:

- Take **16x16** window around detected keypoint (**8x8** example shown here)
- For each window pixel – Compute edge orientation (angle of the gradient - 90°)
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations

Adapted slide of David Lowe

SIFT – Descriptor



Full version of SIFT:

- Divide **16x16** window → **4x4** cell grid (**2x2** example shown here)
- For each cell, compute orientation histogram (8 orientation bins)
- 16 cells (4x4 grid) * 8 orientations (per cell) → **128 dimensional descriptor**

Adapted slide of David Lowe

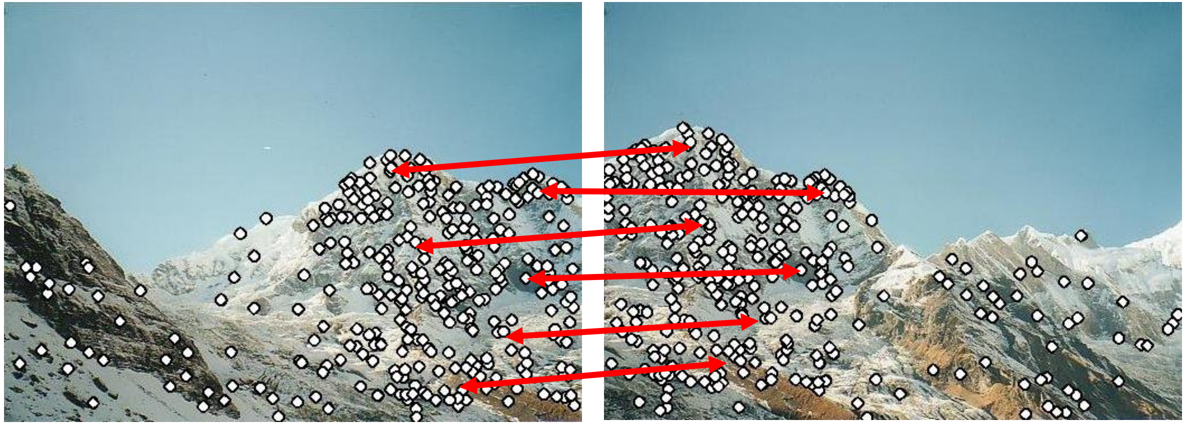
SIFT – Rotation Invariant (Robust)

For each patch:

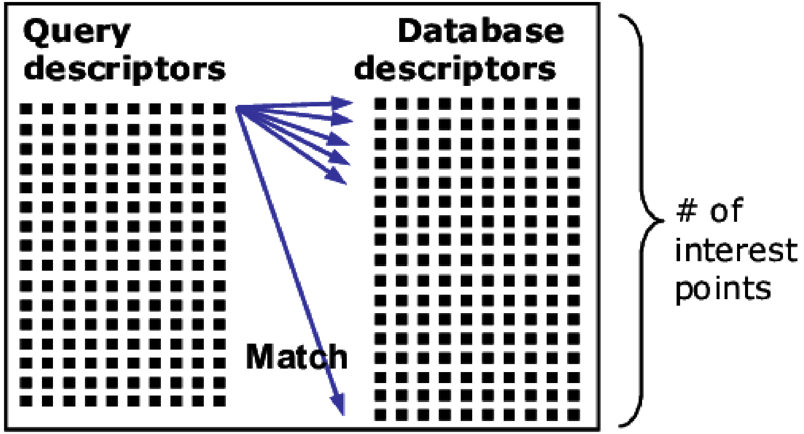
- Compute histogram of its gradient directions/orientations
- Find dominant orientation
- Descriptor represented relative to this orientation

Descriptor is Rotation Invariant!

SIFT – Features Matching – Brute-Force

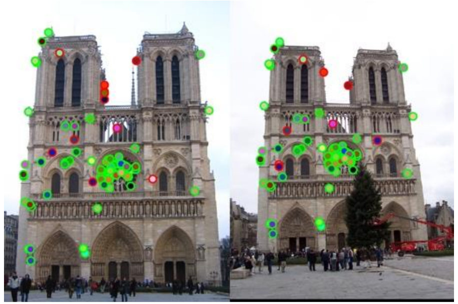


Feature Vectors →



Keypoints –VS– Features

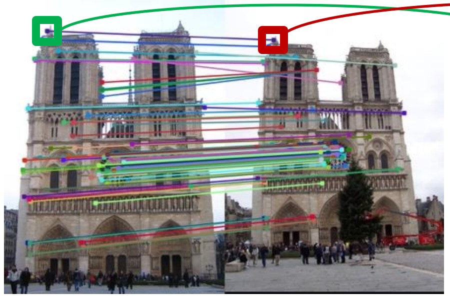
Keypoint Detector
(Finding 'interesting' points)



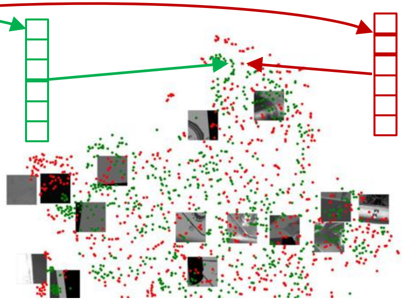
[S. Shah, Local Feature Matching]

Examples (namedrop): Harris corners, **Shi-Tomasi** corner detector, **SUSAN** corner detector, **FAST** feature detector, **Laplacian of Gaussian, Difference of Gaussian, Superpoint, …**

Feature Descriptor
(Describing a keypoint)



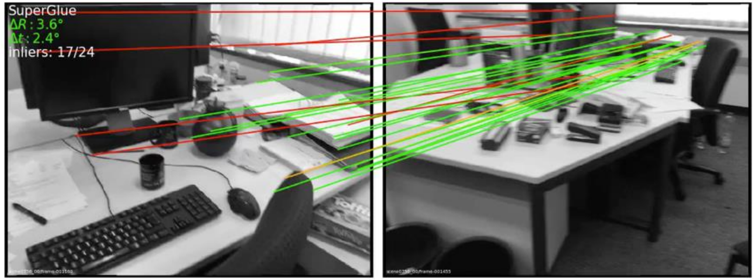
[S. Shah, Local Feature Matching]



[Ghimire et al., Applied Sciences, 2021]

Examples (namedrop): SIFT – Scale Invariant Feature Transform, **SURF** – Speeded-up Robust Features, **BRIEF** – Binary Robust Independent Elementary Features, **ORB** – Oriented FAST Rotated BRIEF, **Superpoint, …**

Goal: Feature Matching



[Sarlin et al., SuperGlue, CVPR 2020]

Adapted slide of
Martin Oswald

SIFT properties

- **Equivariant** (**'Invariant' is probably a better term**) to: ← (check all ⚠ icons)
  - Scale → Detect scale using DoG
  - Rotation → Dominant direction using histogram

- **Partially invariant** to
  - Illumination changes: Gradient magnitude is thrown away, only use direction
  - Camera viewpoint (affine transform)
  - Occlusion
  - Clutter

SIFT in OpenCV

1. **Detect keypoints** – OpenCV tuple:

$$(x, y, s, \theta, \bar{d})_i$$

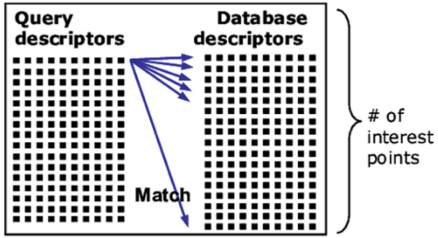
- ***position*** (x, y) in image
- ***scale*** s of detection
- ***orientation*** θ
- ***128D feature descriptor*** $\bar{d}$

2. **Compute** a **feature** per keypt

3. **Establish** keypt *correspondences*

$$(x, y, s, \theta, \bar{d})_i$$

$$(x, y, s, \theta, \bar{d})_k \xleftarrow{F_1} \quad \xrightarrow{F_2} (x', y', s', \theta', \bar{d}')_l$$



>Will be covered<
>Next week<

SIFT for Image Stitching

1. **Detect keypoints** – OpenCV tuple

$$(x, y, s, \theta, \bar{d})_i$$

SIFT for Image Stitch.

>Will be covered<
>Next week<

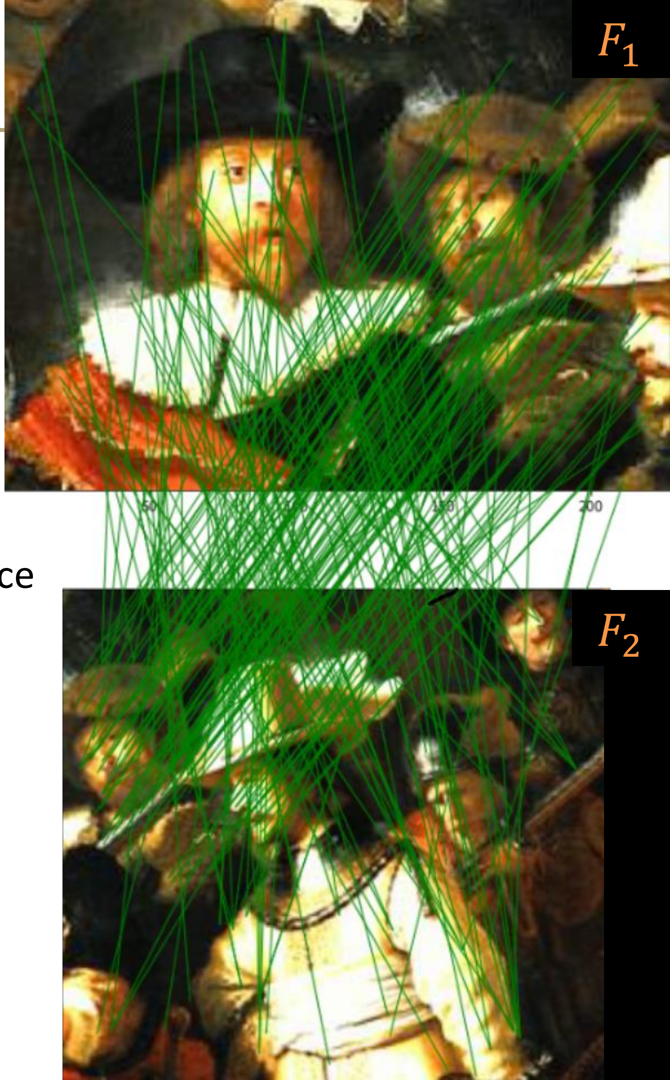
1. **Detect keypoints** – OpenCV tuple

$$(x, y, s, \theta, \bar{d})_i$$

2. **Compute** a **feature** per keypt

3. **Establish** keypt ***correspondences*** → Can be ***noisy***

4. **Filter** corresp. & find transf. with ***RANSAC*** ← *Solution* ***Random*** subsets produce ***diff.*** Projective Trans. T



>Will be covered<
>Next week<

SIFT for Image Stitch.

RANSAC

Goal: Find ***subset*** of corrsp. that produce a ***good*** Proj. Transform T

A *candidate* corresp. $\{(x, y), (x', y')\}$ is ***true positive*** if it complies with T :

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \approx T \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

within a small error margin

Steps

a. Randomly select 4 corresp: $\{(x, y), (x', y')\}_{i \in \{1,2,3,4\}}$ → Compute (hypothesis) Proj. Transf. T

b. Count number # of *inliers* for current hypothesis/model → For other corrsp. evaluate whether:

$$\left\| \begin{bmatrix} x \\ y \end{bmatrix} - h2e\left(T \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \right) \right\| \leq \varepsilon$$

c. Store model with max inliers (corr., # inliers, fit quality, trnsf. T)

d. Iterate max N times

e. For best model – use ***all*** *inliers*

>Will be covered<
>Next week<

SIFT for Image Stitching

1. **Detect keypoints** – OpenCV tuple

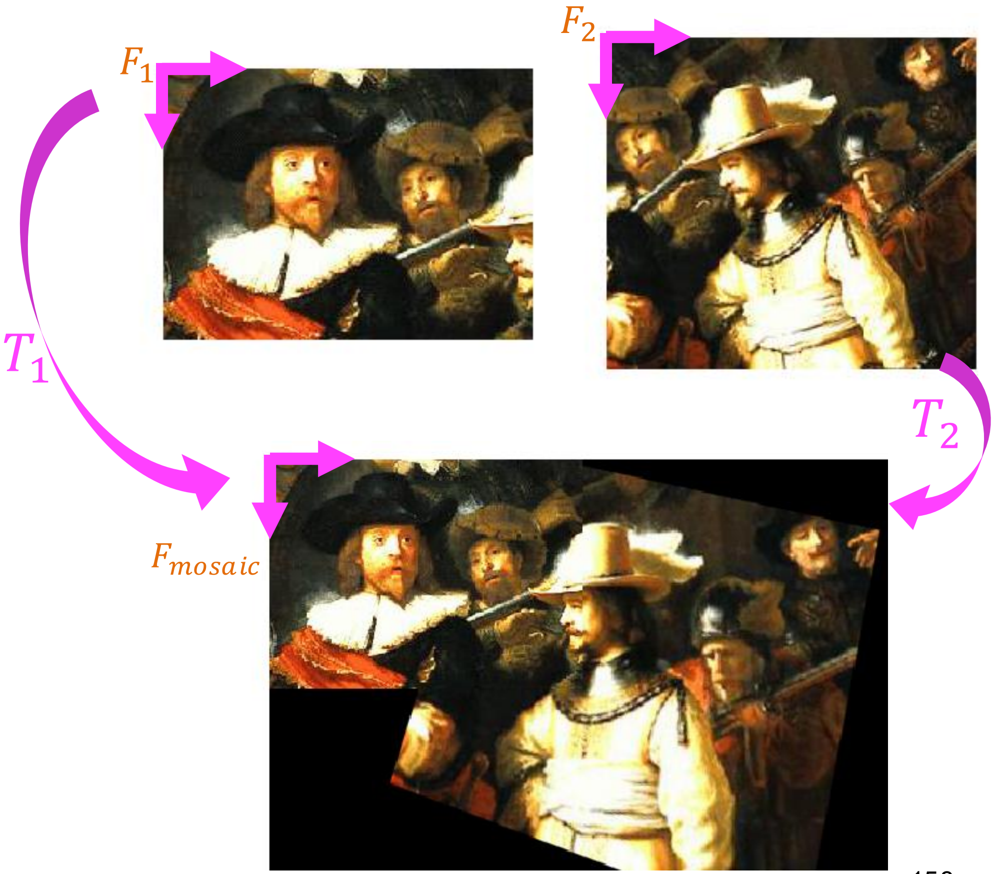
$$\left(x, y, s, \theta, \bar{d}\right)_i$$

2. **Compute** a **feature** per keypt

3. **Establish** keypt *correspondences*

4. **Filter** corresp. & find transf. with ***RANSAC***

5. **Warp** F_1 and F_2 into frame F_{mosaic} with their respective *final transf. T*



Outline

- Edge Detection
  - Derivatives of Image, Derivatives of Gaussian
  - Canny Edge Detector
- Line Fitting
  - Least Squares
  - RANSAC
  - Hough Transform
- Corners
  - Harris Corners
  - SIFT
  - Applications

Applications

- Feature points are used for:
  - Image alignment
  - 3D reconstruction
  - Motion tracking
  - Robot navigation
  - Indexing and database retrieval
  - Object recognition

Stitching & Multi-band Blending

>Will be covered<
>Next week<

P. J. Burt and E. H. Adelson
[A multiresolution spline with application to image mosaics]
ACM Transaction on Graphics (ToG), 1983

Stitching & Multi-band Blending

>Will be covered<
>Next week<

P. J. Burt and E. H. Adelson
[A multiresolution spline with application to image mosaics]
ACM Transaction on Graphics (ToG), 1983

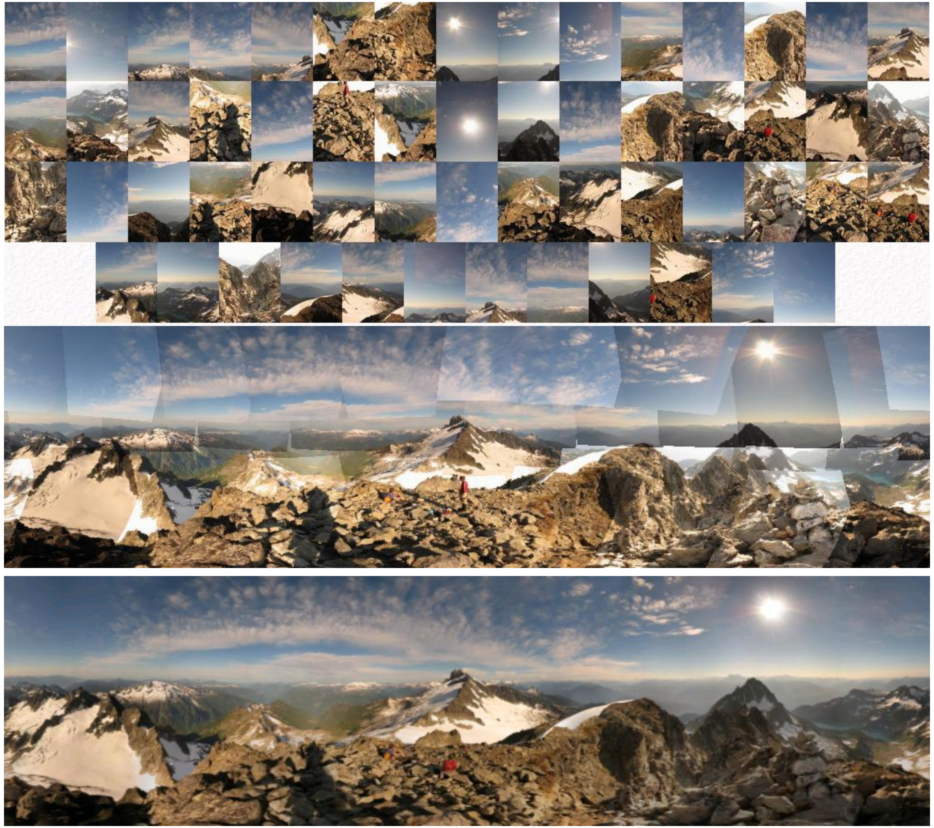
Stitching & Multi-band Blending

>Will be covered<
>Next week<

P. J. Burt and E. H. Adelson
[A multiresolution spline with application to image mosaics]
ACM Transaction on Graphics (ToG), 1983

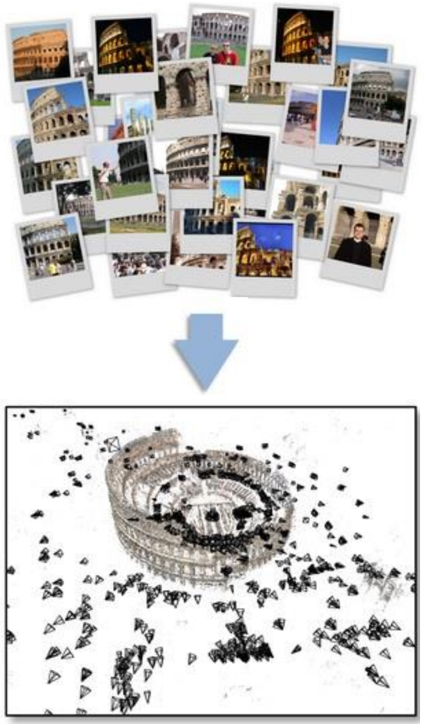
Stitching & Multi-band Blending

>Will be covered<
>Next week<



http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html

Example: Structure from Motion



Jan-Michael Frahm et al.
[Building rome on a cloudless day]
ECCV 2010

https://youtu.be/4cEQZreQ2zQ

Closing Remarks

Action Points

- This week:
  - Thursday Lecture: Optical Flow & Motion
  - Finish Quiz-2
  - Finish Lab-2 Assignment
  - Start Lab-3 Assignment

Disclaimer

Many of the slides used here are obtained from online resources (including many open lecture materials) without appropriate acknowledgement. They are used here for the sole purpose of classroom teaching. All the credit and all the copyrights belong to the original authors. You should not copy it, redistribute it, put it online, or use it for any other purposes than for this course.