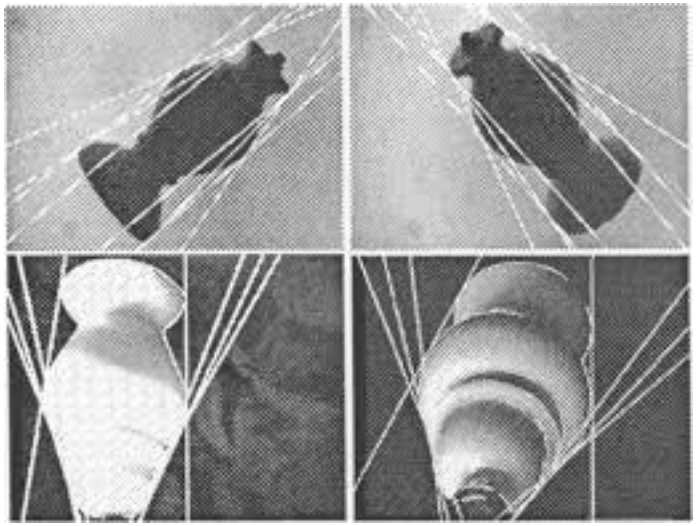
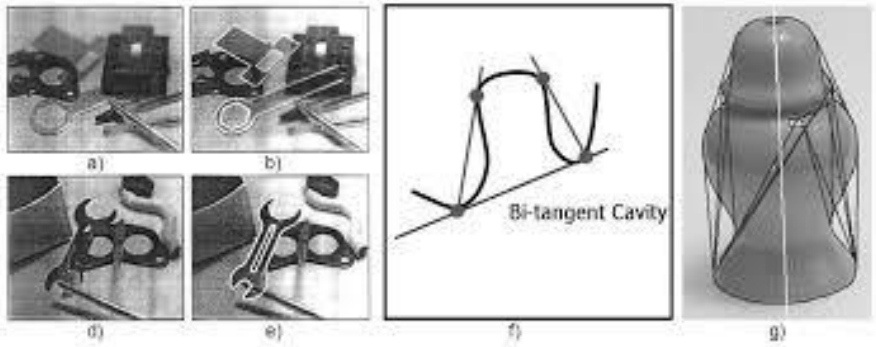
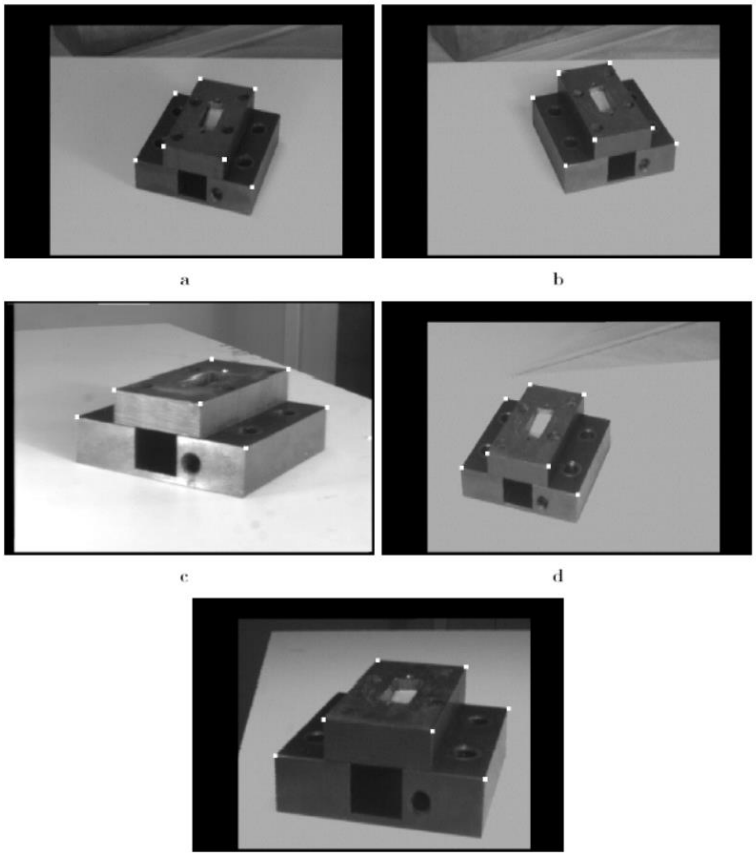
Line fitting, Corner Extraction

Some slides of this lectures are courtesy of profs. S. Lazebnik & K. Grauman





Object recognition in the Geometric Era, Mundy



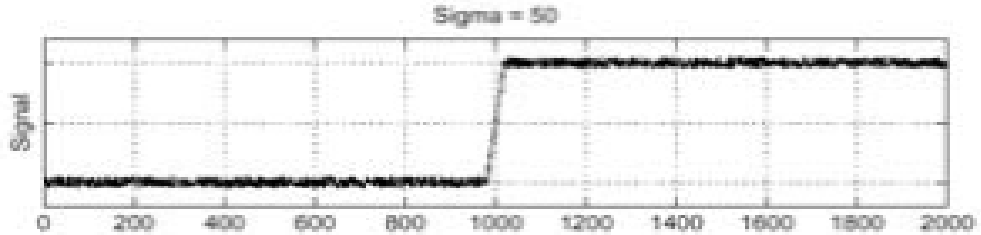
4th ICCV, 1993, Berlin, Germany[edit]

•Charles A. Rothwell, David A. Forsyth, Andrew Zisserman, and Joseph L. Mundy, Extracting Projective Structure from Single Perspective Views of 3D Point Sets

Edge as the zero-crossing of the second order derivative using Laplacian of Gaussian (LoG filter/Marr filter)

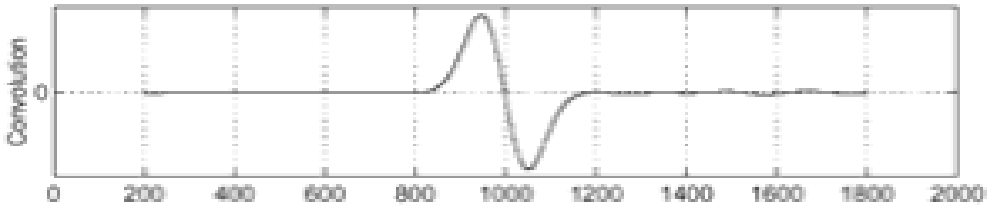
Consider $\frac{\partial^2}{\partial x^2}(h \star f)$

f



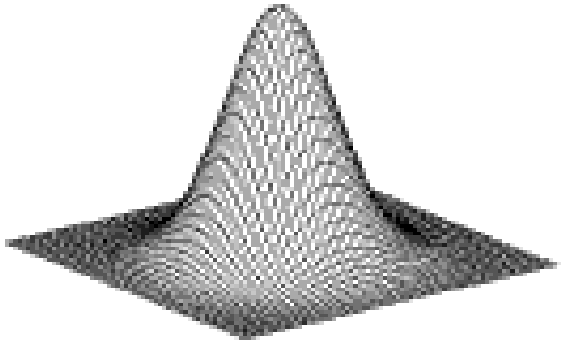
$\frac{\partial^2}{\partial x^2}h$

$(\frac{\partial^2}{\partial x^2}h) \star f$



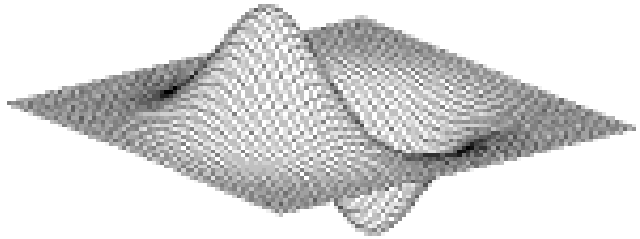
Where is the edge? Zero-crossings of bottom graph

2D edge detection filters



Gaussian

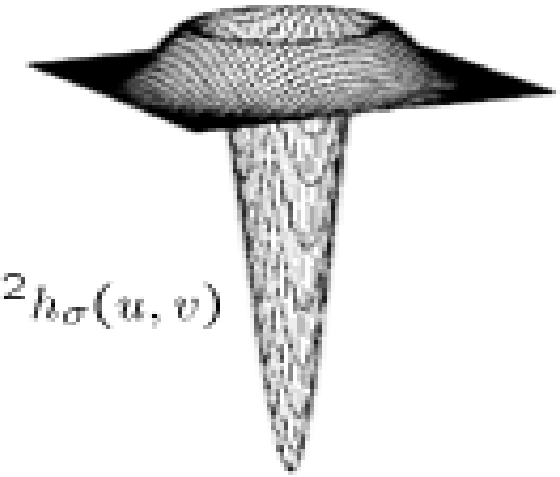
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian

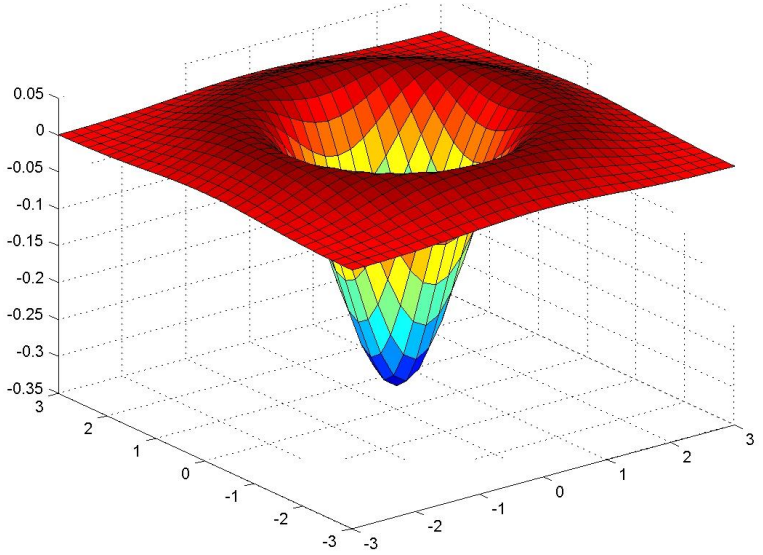


$$\nabla^2 h_\sigma(u,v)$$

- ∇^2 **is the Laplacian operator:**

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Laplacian of Gaussian (LoG): Marr edge detector



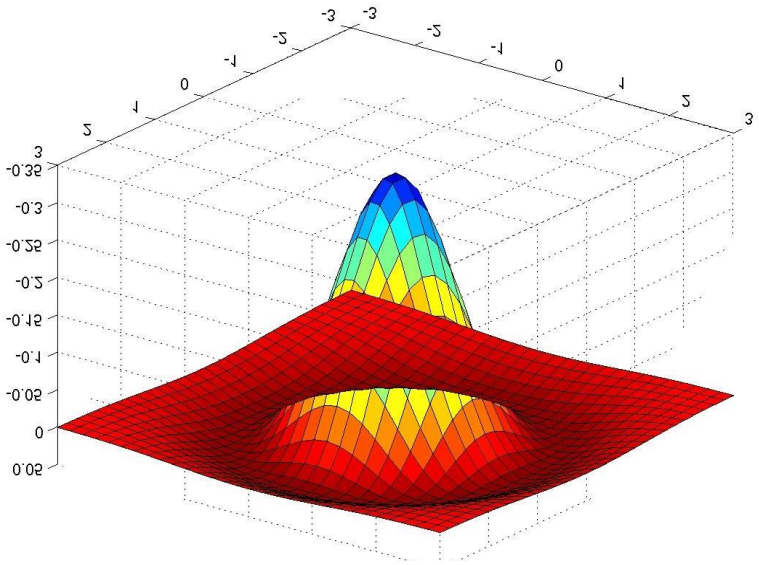
$$LoG_\sigma = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

- Matlab: `fspecial('log', sigma)`

LoG kernel — Mexican hat

An example of an 11×11 mask approximating the Laplacian of Guassian with $\sigma^2 = 2$ is given below:

0	0	0	−1	−1	−2	−1	−1	0	0	0
0	0	−2	−4	−8	−9	−8	−4	−2	0	0
0	−2	−7	−15	−22	−23	−22	−15	−7	−2	0
−1	−4	−15	−24	−14	−1	−14	−24	−15	−4	−1
−1	−8	−22	−14	52	103	52	−14	−22	−8	−1
−2	−9	−23	−1	103	180	103	−1	−23	−9	−2
−1	−8	−22	−14	52	103	52	−14	−22	−8	−1
−1	−4	−15	−24	−14	−1	−14	−24	−15	−4	−1
0	−2	−7	−15	−22	−23	−22	−15	−7	−2	
0	0	−2	−4	−8	−9	−8	−4	−2	0	
0	0	0	−1	−1	−2	−1	−1	0	0	



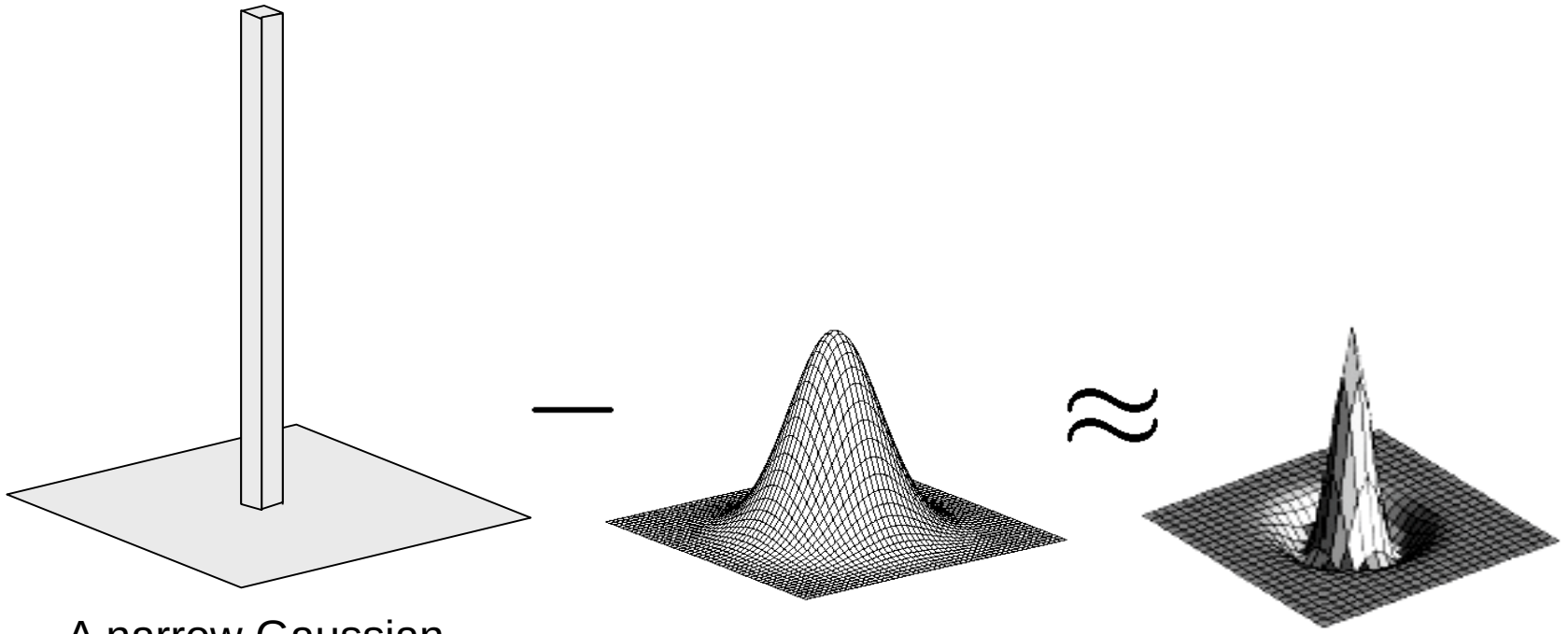
Detecting zero-crossings

- The edge locations occur at points in the LoG-filtered image where the value passes through zero. i.e., where there is a sign change.

- The number of zero crossings is influenced by the size of the Gaussian; the greater the smoothing, the smaller the number of zero crossings.

- A simple algorithm is to declare a pixel p to be zero crossing if any one of its 8-neighbors is of opposite sign. To restrict the detected zero crossings to the more significant ones, we can impose a threshold T.

“Laplacian of Gaussian” (LoG) can be approximately computed as “Difference of Gaussians (DoG).

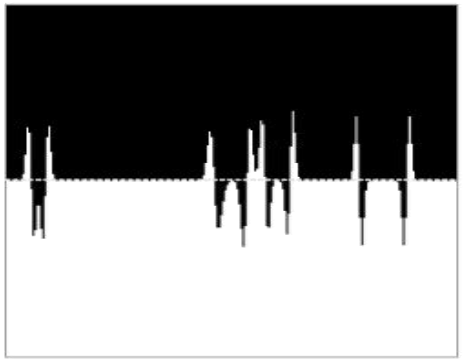


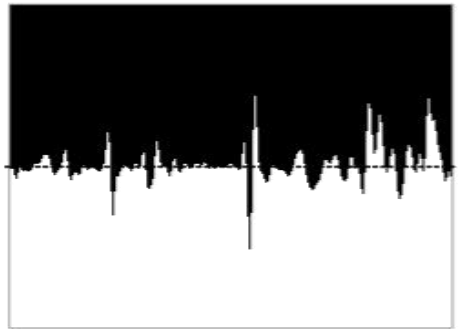
A narrow Gaussian
(or an impulse function)

A wide Gaussian

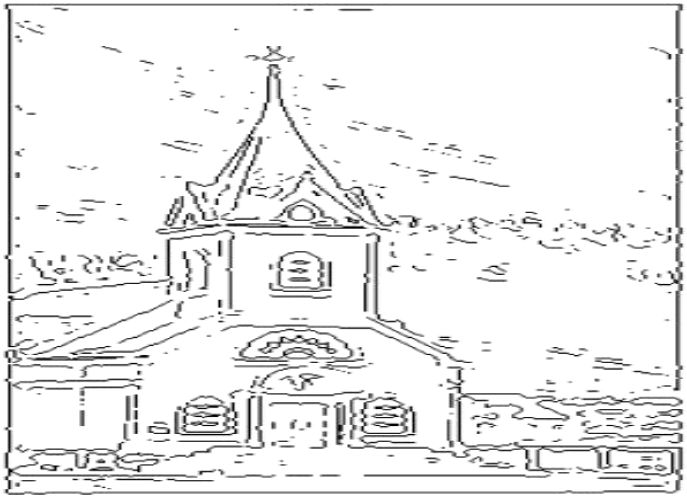
Laplacian of Gaussian

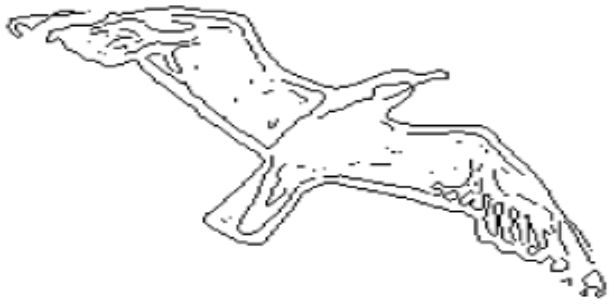
Example: 2-D LoG filtering





Example: LoG/Marr edge detector

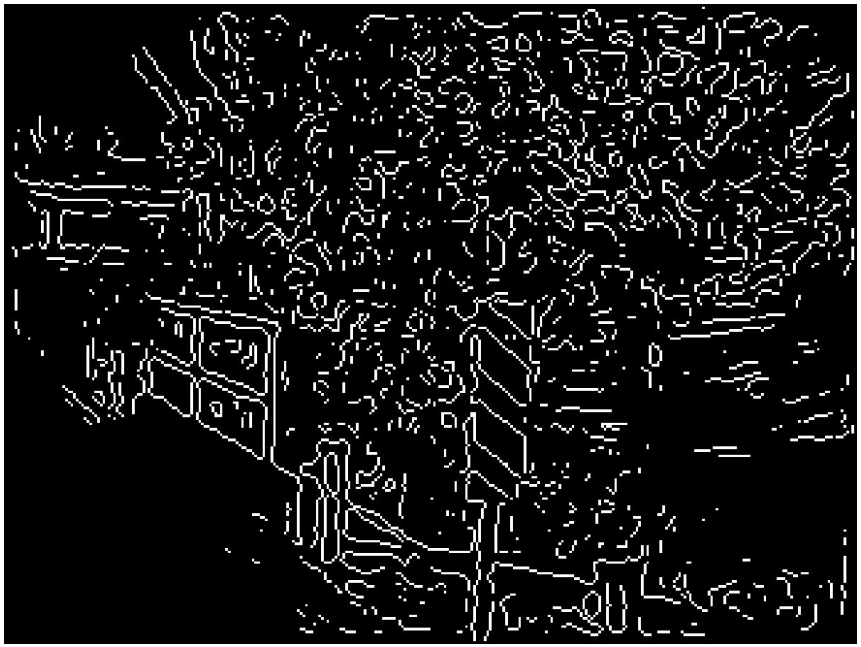




Change of Scale (sigma in Gaussian/LoG)

Sobel vs. LoG

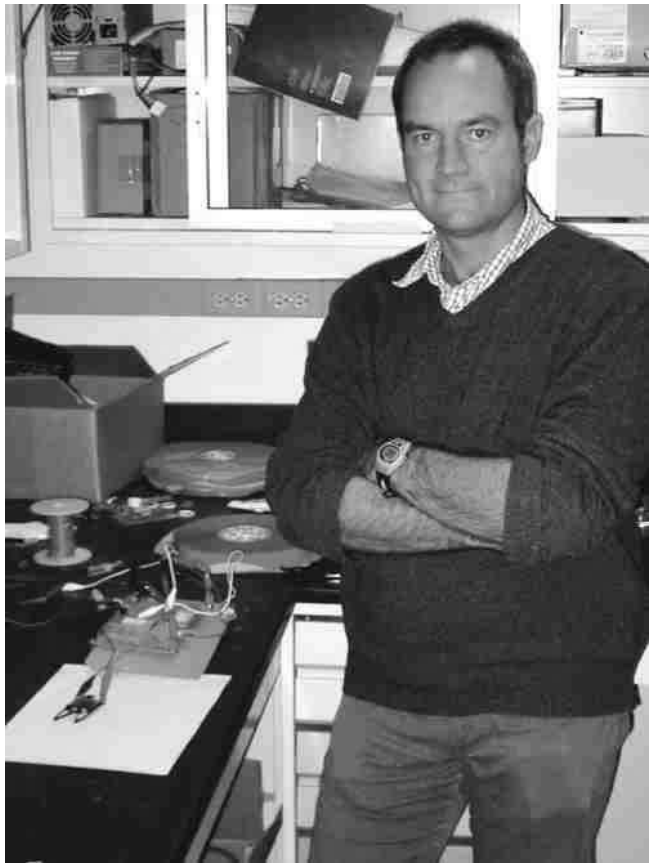
Sobel



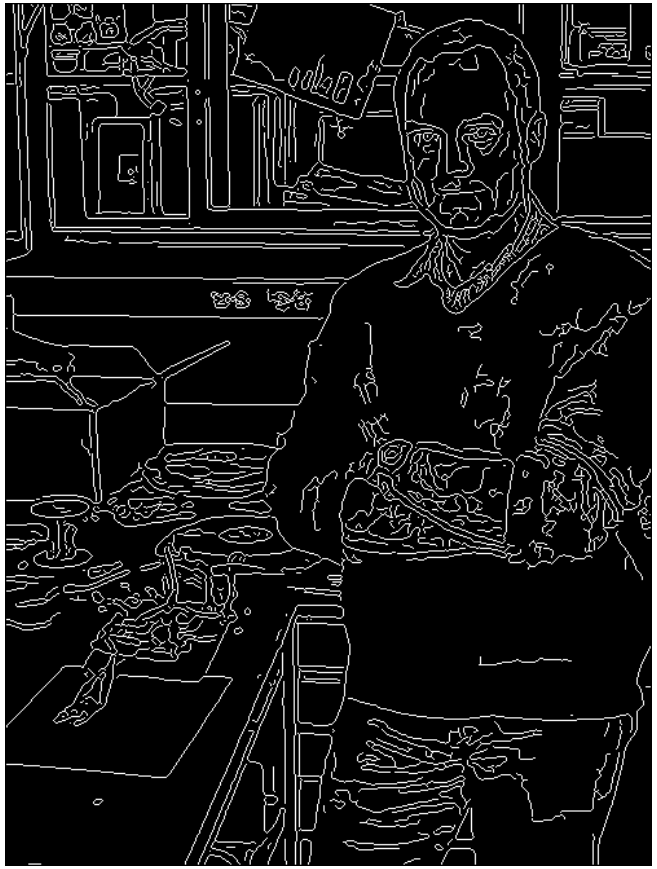
LoG

Example: Canny edge detector

Canny Edge Detector



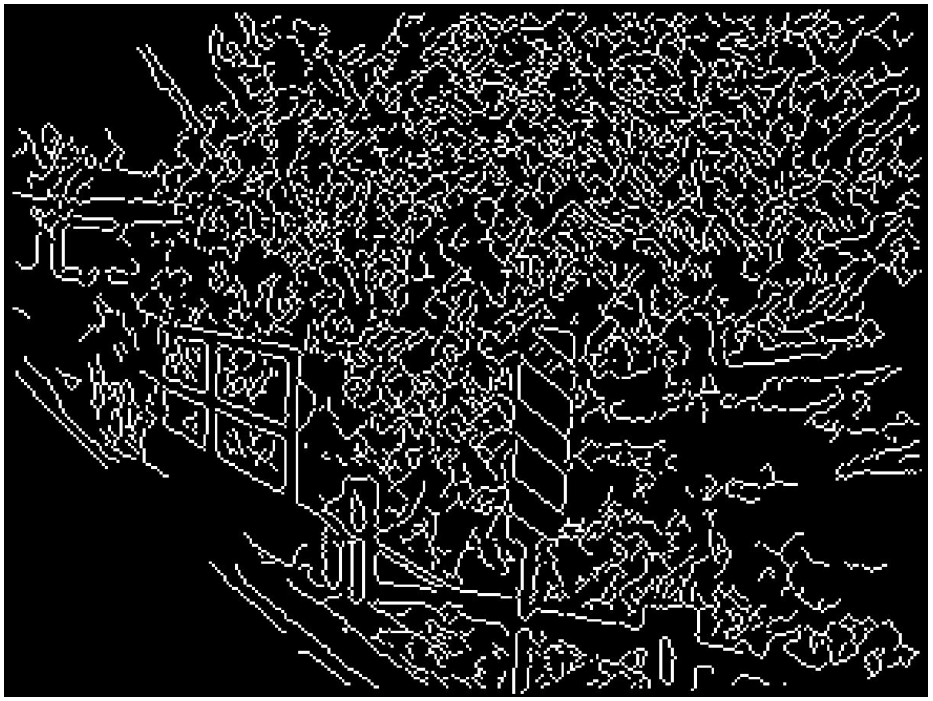
John Canny



Canny Edge Detector is one of the most successful edge detection method.

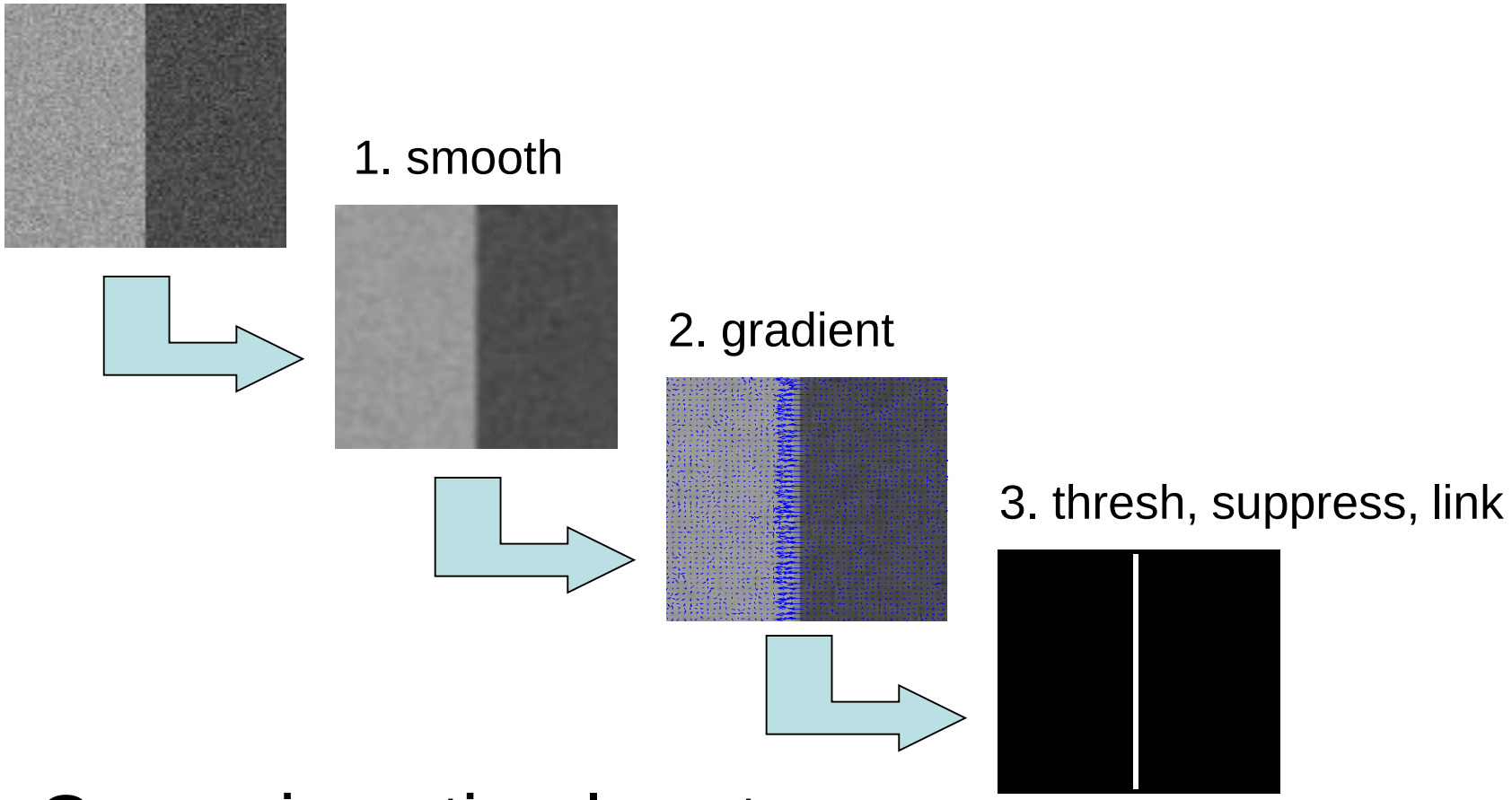
Ref: John F. Canny, "A computational approach to edge detection," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 8, No. 16, pp. 679-698, Nov. 1986.

Canny Edge Detector



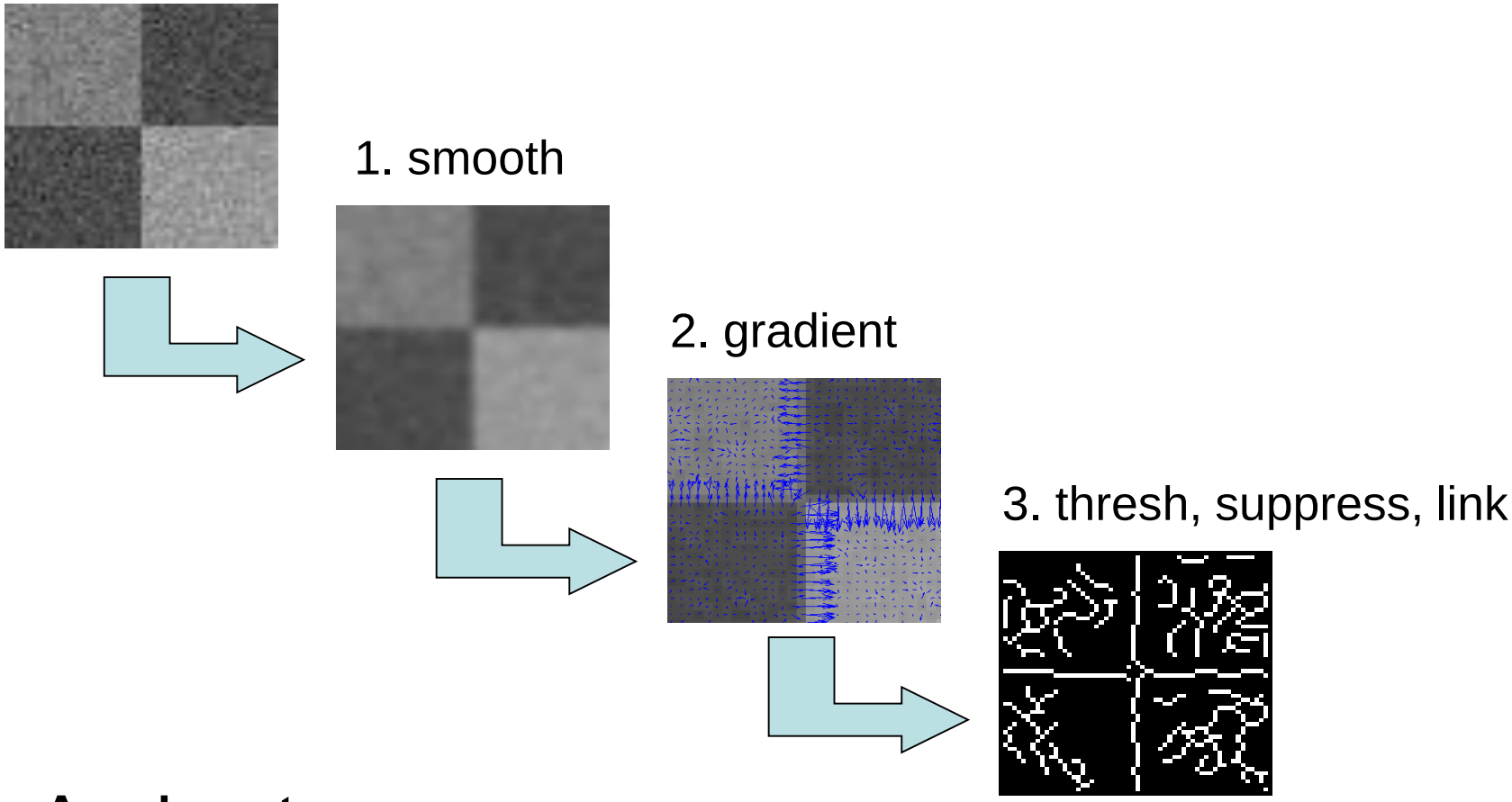
(Matlab automatically set thresholds)

Canny edge detection



Canny is optimal w.r.t. some model.

Canny edge detection



And yet…

The Canny edge detector

- norm of the gradient

The Canny edge detector

- thresholding

The Canny edge detector

- thinning

- (non-maximum suppression, edge following)

- Make use of Derivative of Gaussian
- Non-maximum suppression
  - Thin multi-pixel wide “ridges” down to single pixel
- Optimal thresholding
  - Low, high edge-strength thresholds
  - Accept all edges over low threshold that are connected to edge over high threshold
- Optimal decision making.
- Matlab: `edge(I, 'canny', sigma)`

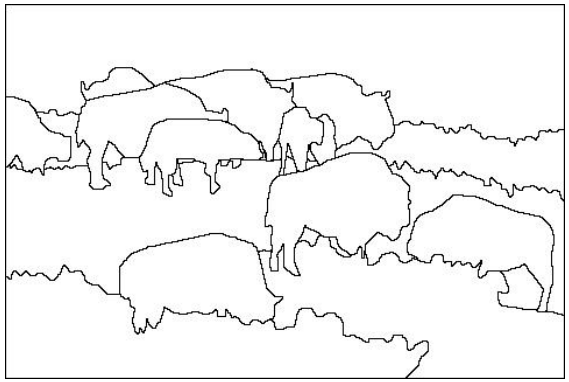
Modern edge detectors

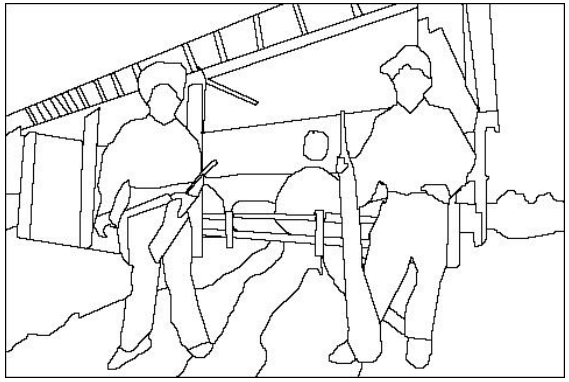
- Machine learning based edge detector.
- To learn how human label edges of natural images.

Reference:
Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues,
David R. Martin, Charless C. Fowlkes, and Jitendra Malik, TPAMI, 2004

Learning to detect boundaries

image | human segmentation | gradient magnitude




- Berkeley segmentation database: http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

Slides taken from Pb edge-detection: Martin, Malik et al.

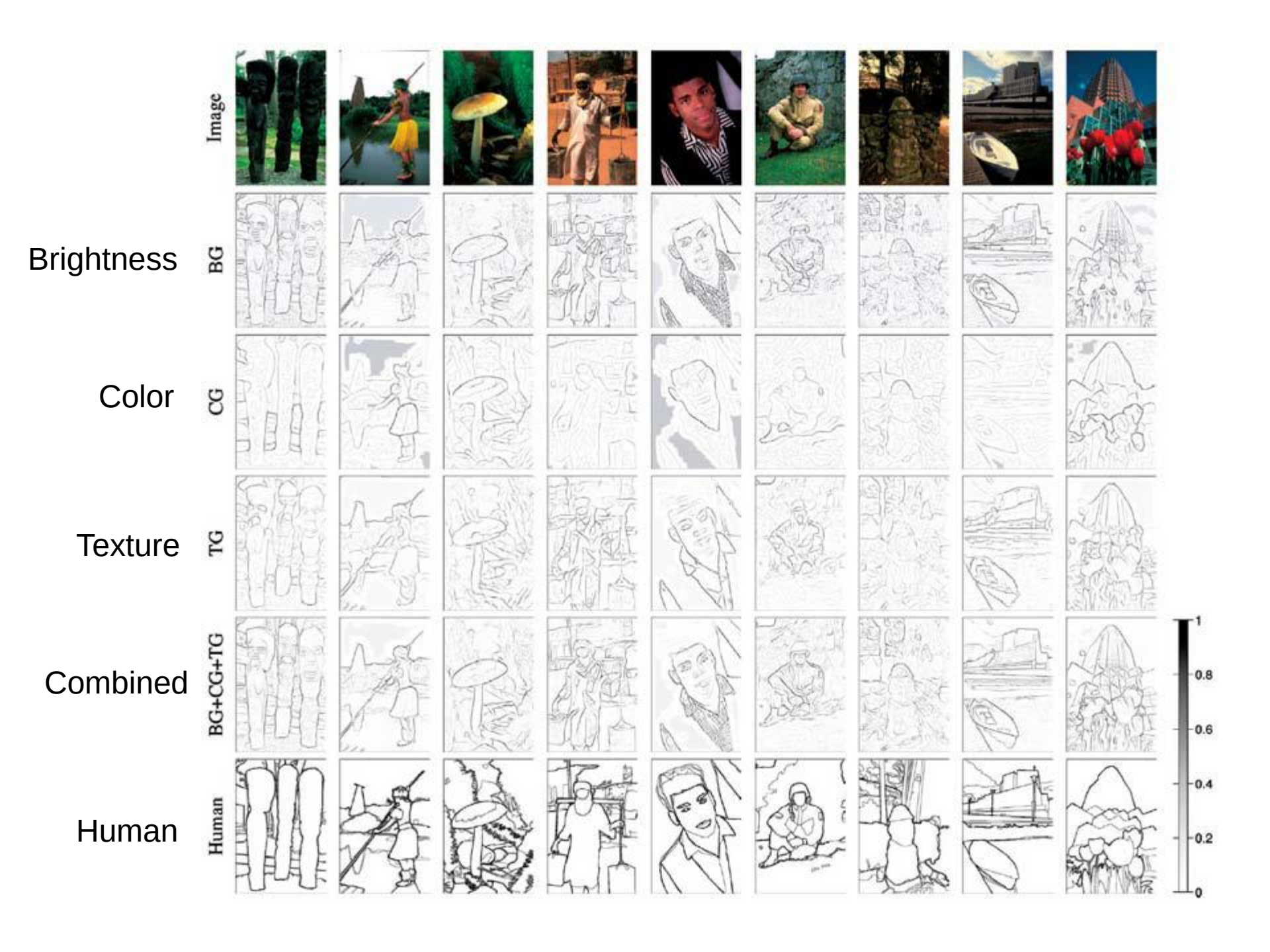
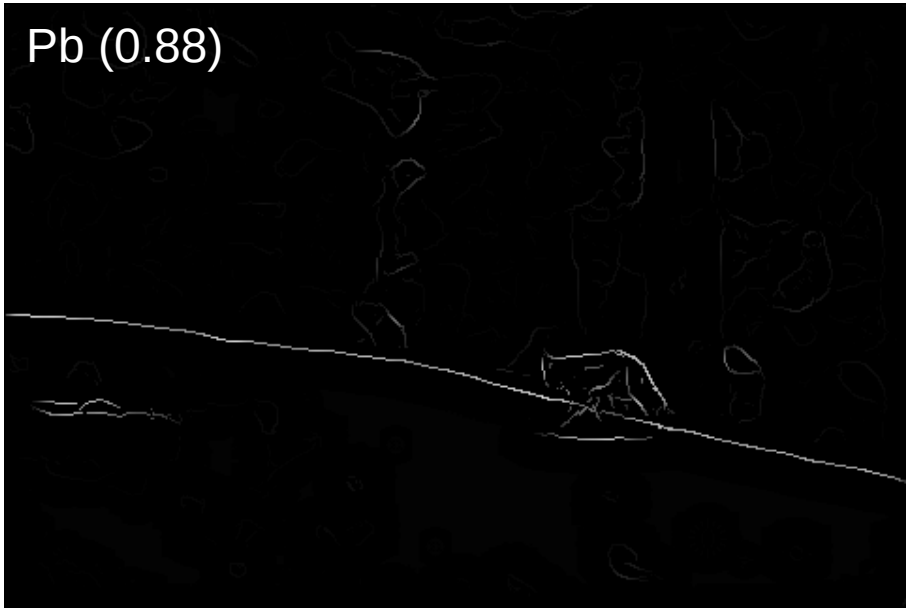
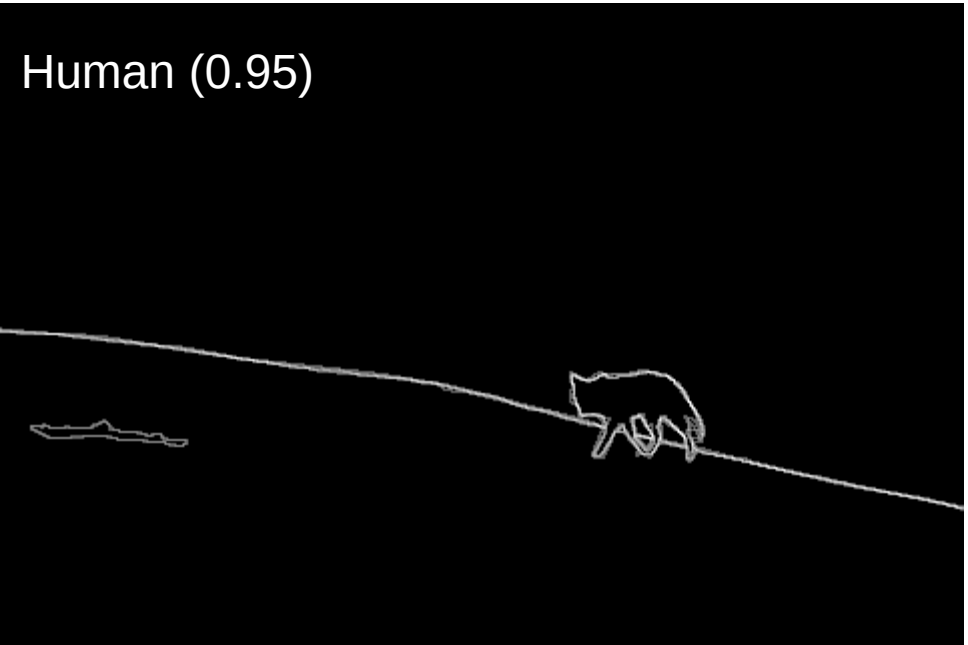


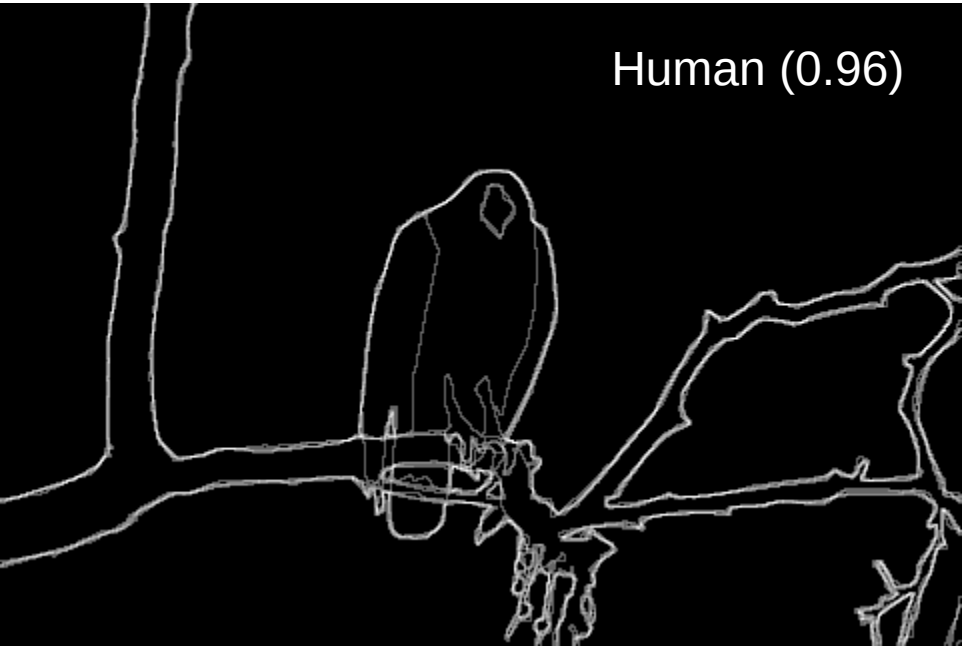
Image
BG
CG
TG
BG+CG+TG
Human
Brightness
Color
Texture
Combined
Human
1
0.8
0.6
0.4
0.2
0

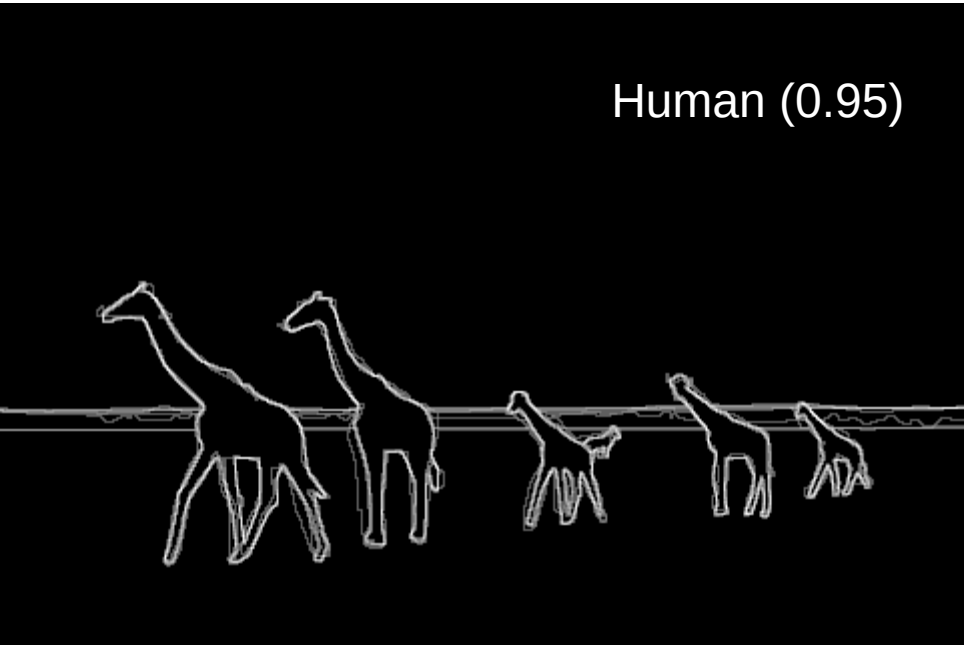
Results



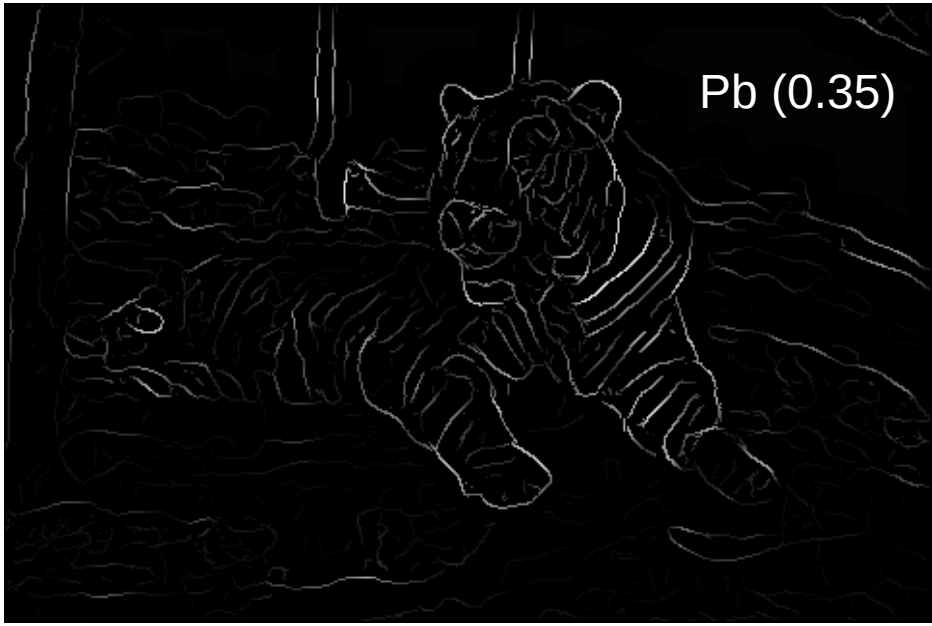


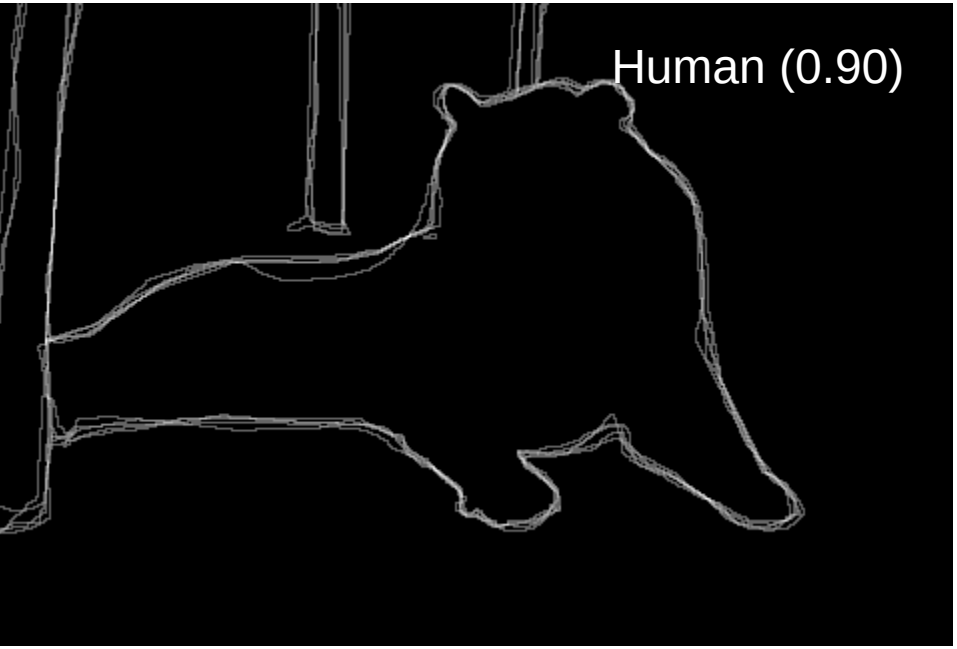
Results




Human (0.95)


Pb (0.63)

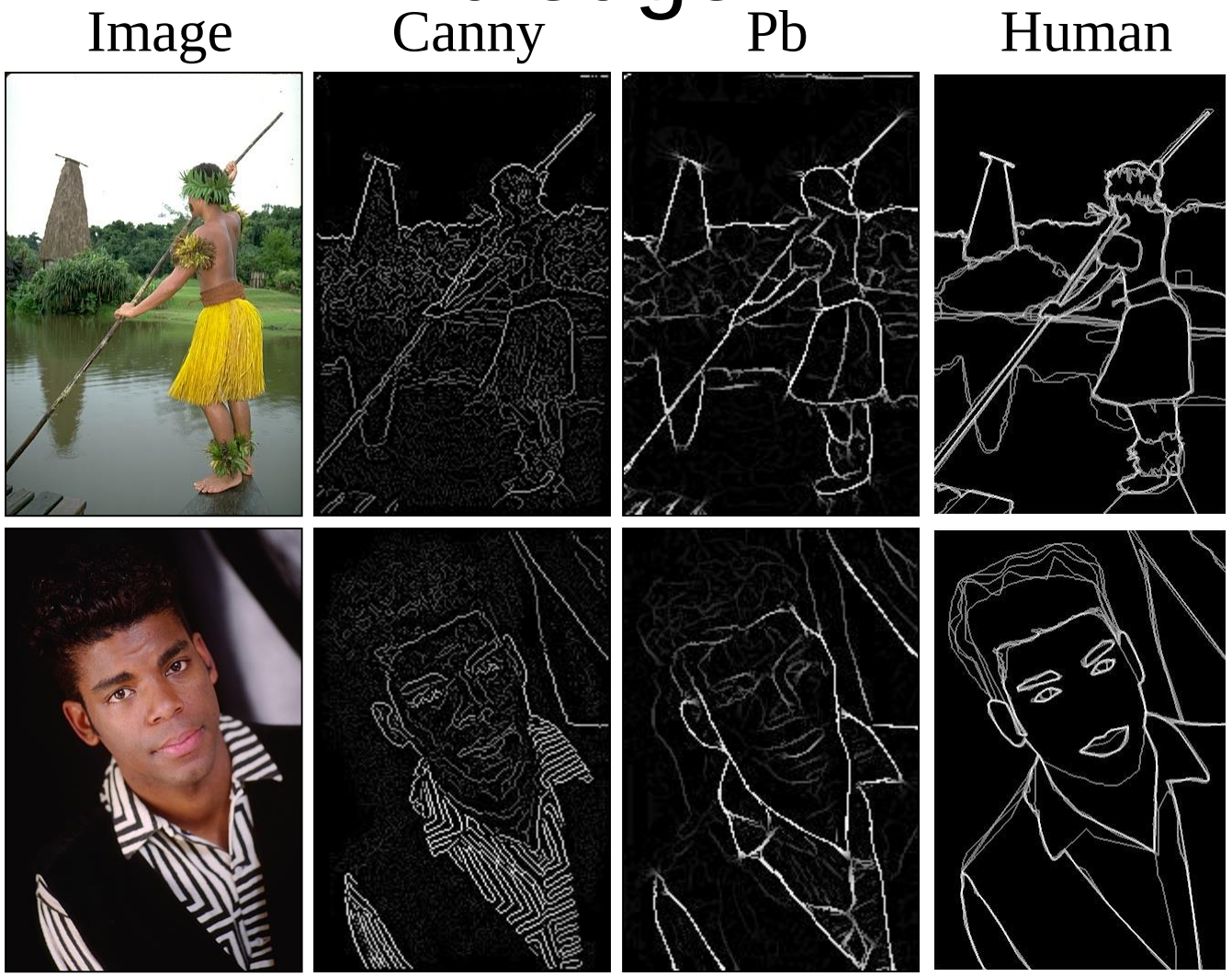




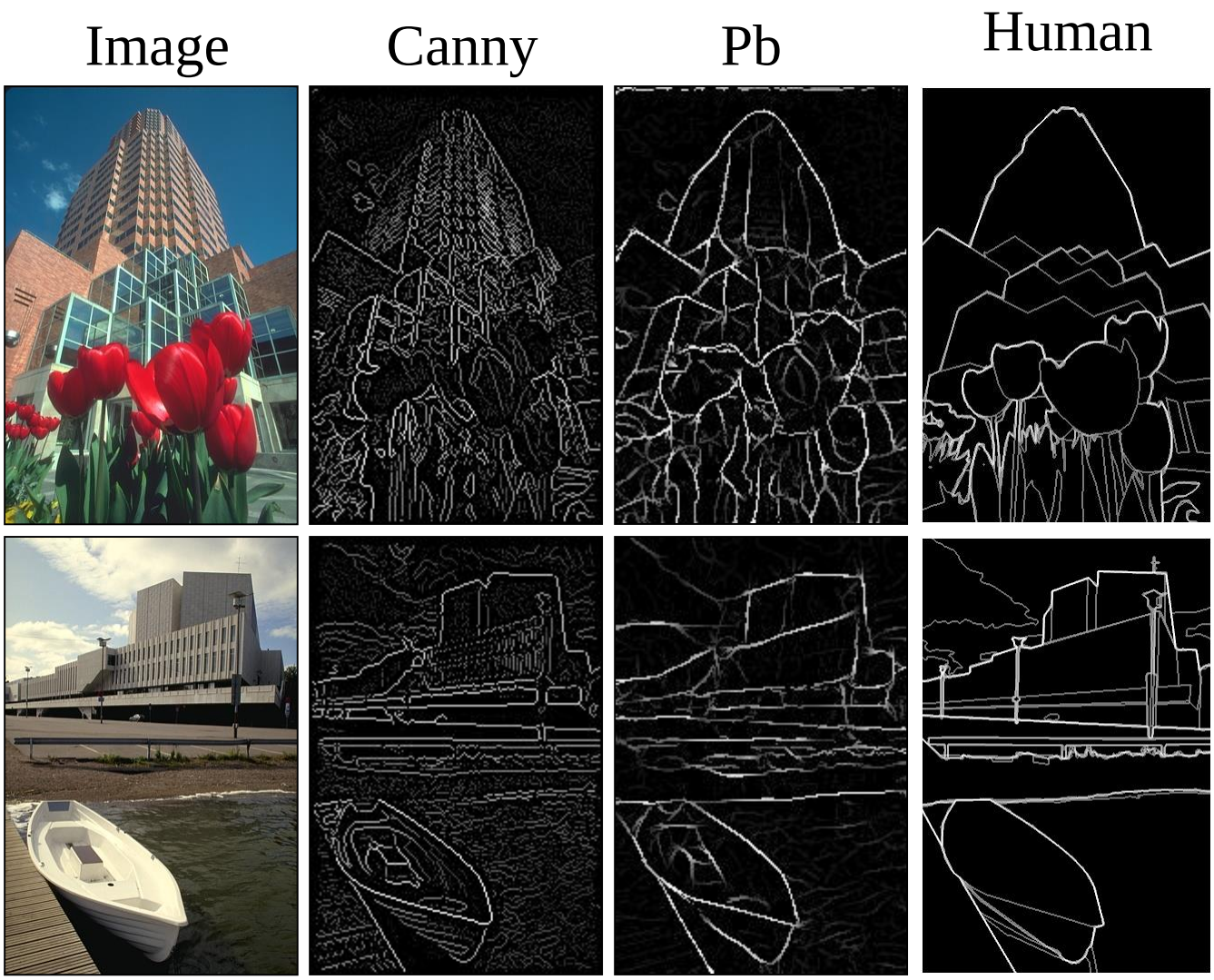
For more:
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/bench/html/108082-color.html

Sample human-labeled training images and their corresponding edge maps

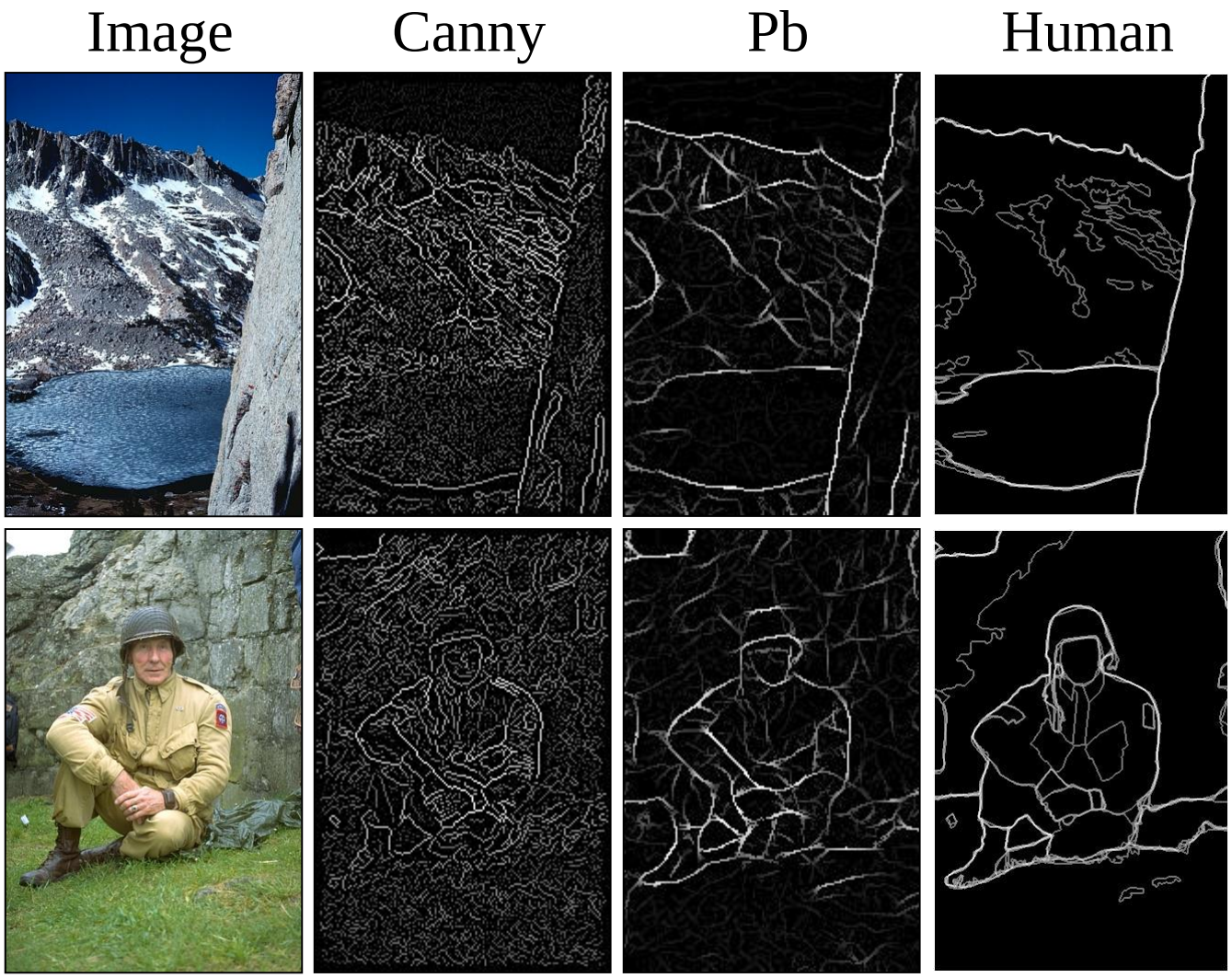
Learned edge detector: Pb edge I



Pb result II



Pb result III

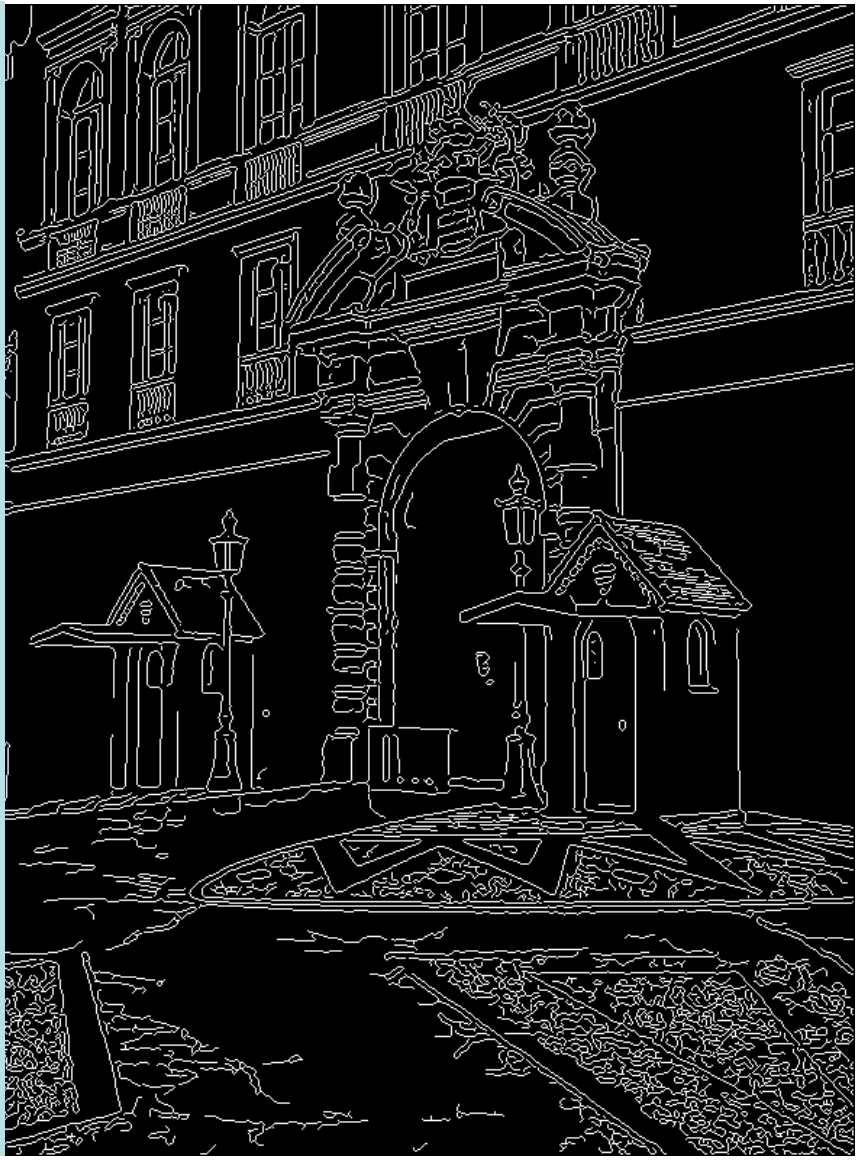


Line Fitting

A binary edge map is not enough

A binary edge map is not enough

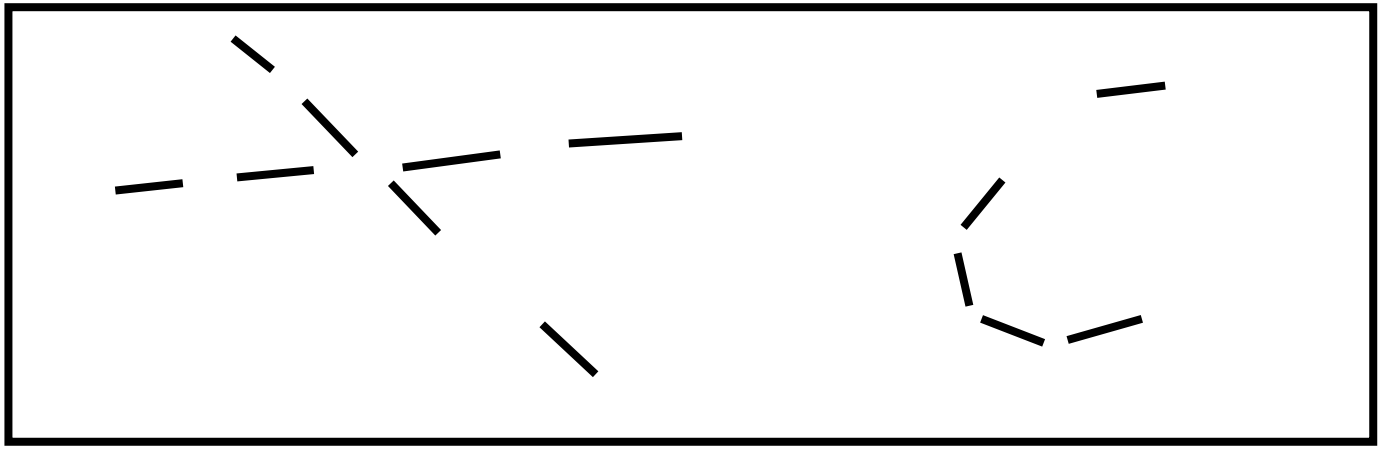
- Fragmented edges
- Many broken lines.
- Noise in edge orientations.
- Membership not clear:
  which points on which line ?



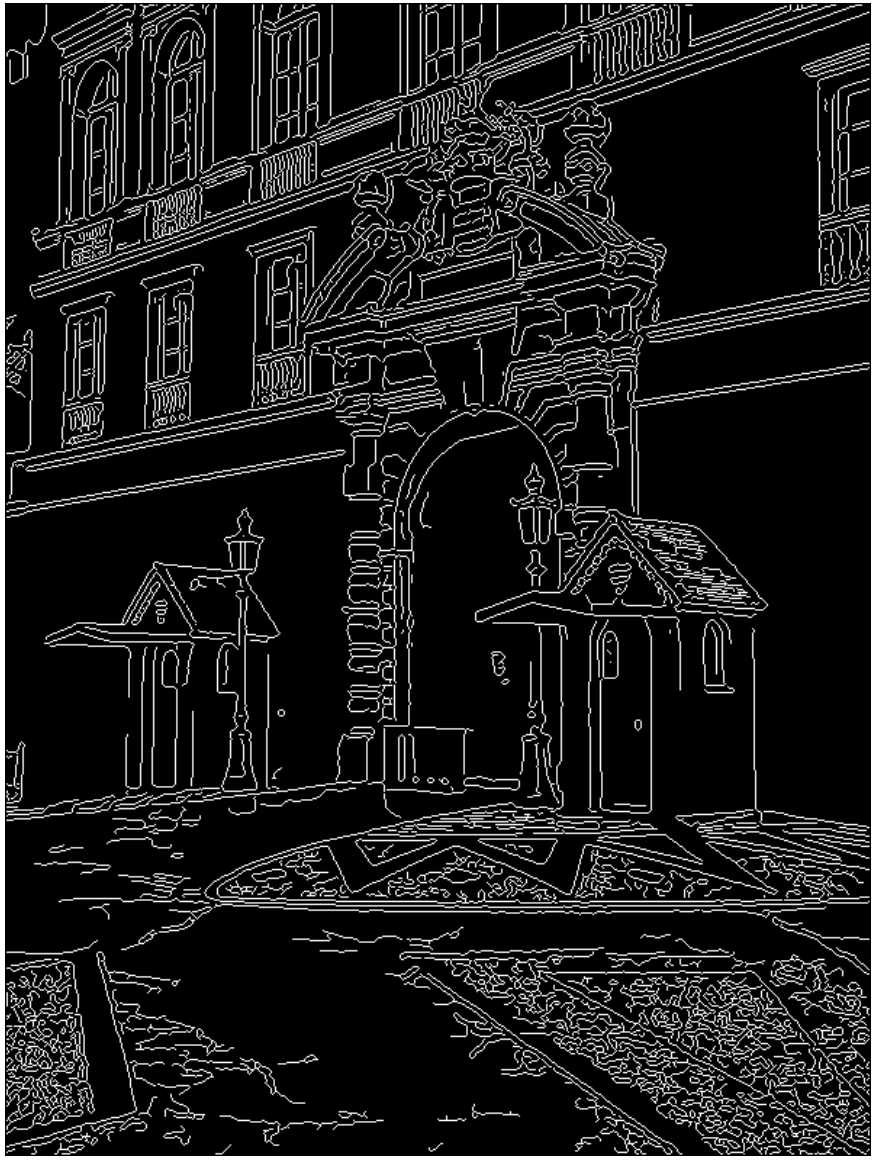
Vanishing lines (3D Vision)

Goal of Line Fitting:

from fragmented edges to straight lines or curves



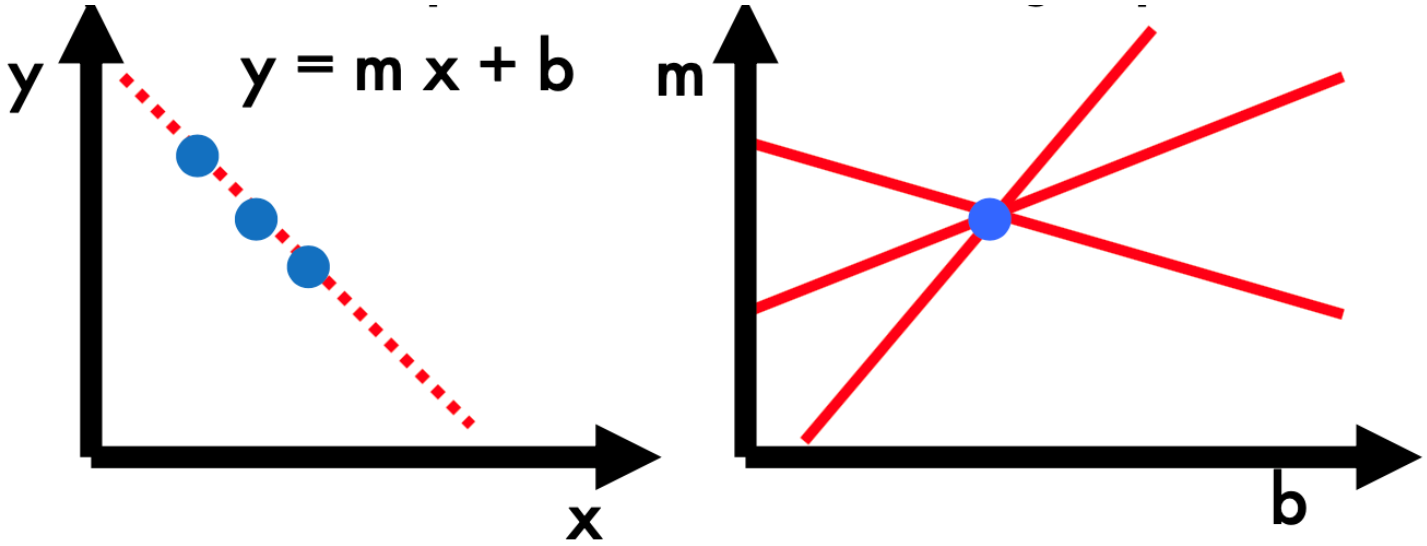
Goal of Line **Fitting**



Hough Transform

Fitting Multiple Lines using Hough Transform

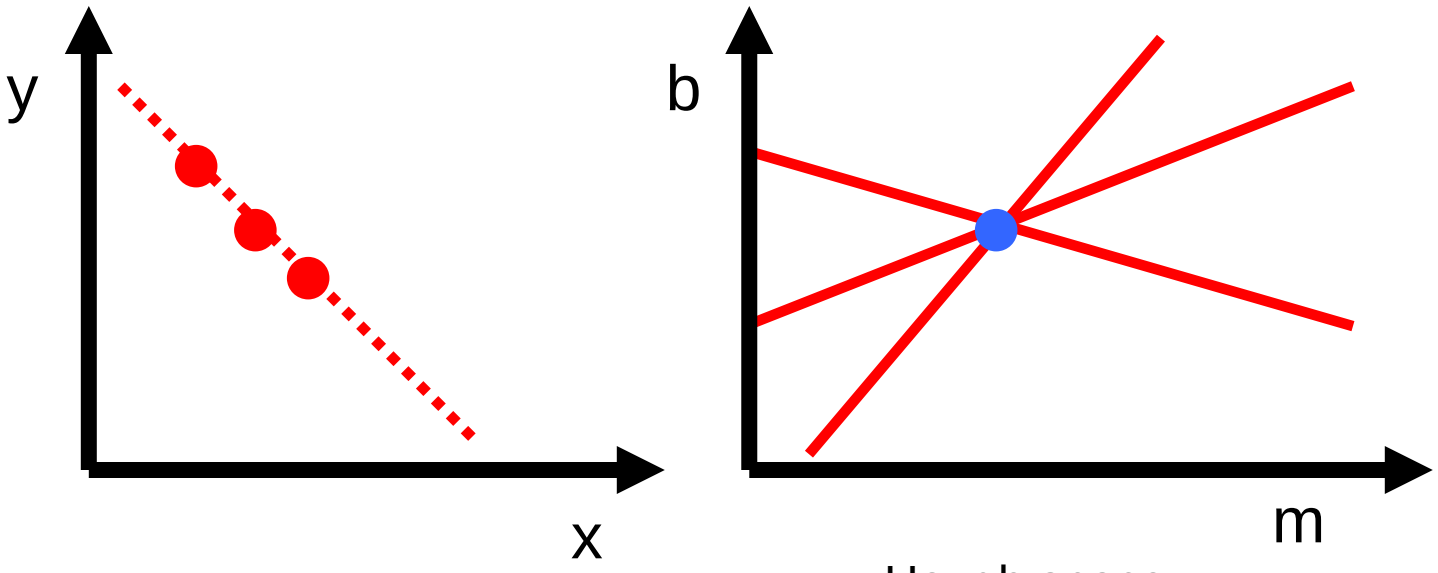
- Given a binary edge image, find the lines (or curves) that explain the data points best in the parameter space.
- This parameter space is called a Hough space



Hough transform

P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959. (Patented).

Given a set of points, find the line (or curve) that explains the data points best



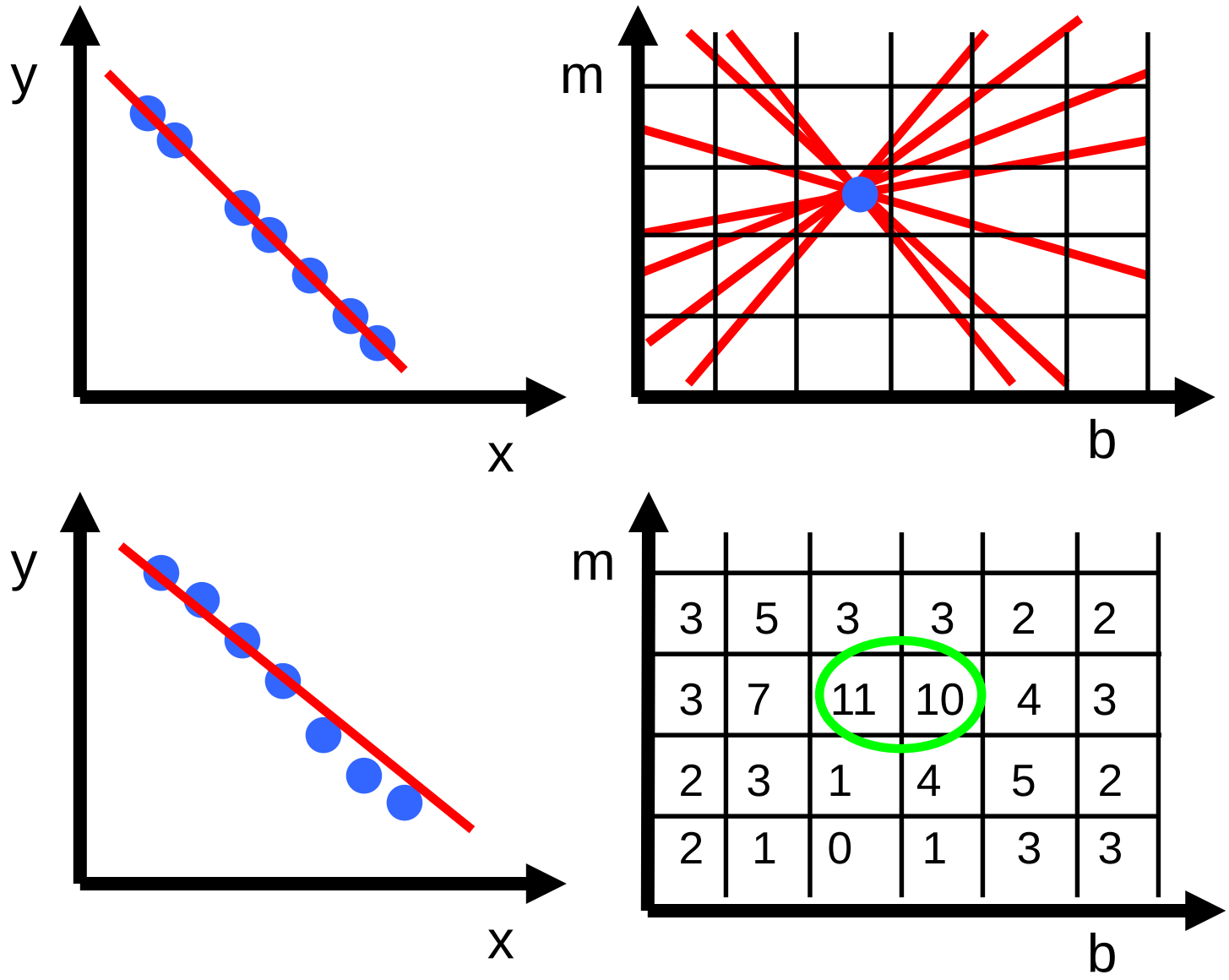
Hough space

$y = m\,x + b$

$b = m\,x - y$

Slide from S. Savarese

Hough transform



Slide from S. Savarese

Hough Transform

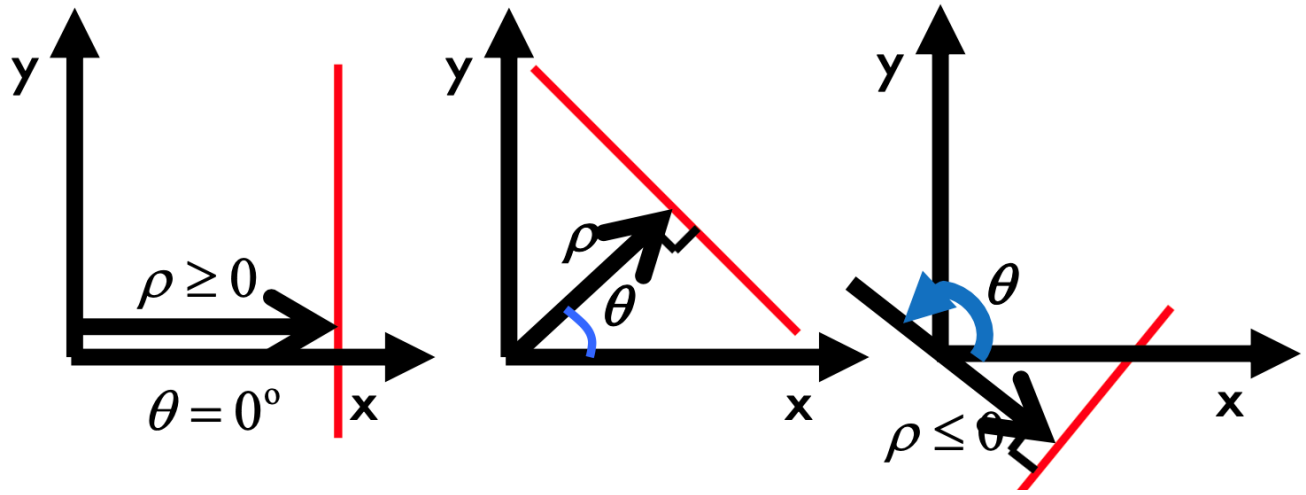
- Can detect multiple lines in an image (from multiple local maxima)
- Can be easily extended for circles and ellipses
- Computationally efficient
- Problem: (*m, b*) are unbounded. For example, the slope parameter *m* can have an infinite value.

Slide from S. Savarese

Hough Transform: Using Polar Form

- Use a polar representation for the parameter space

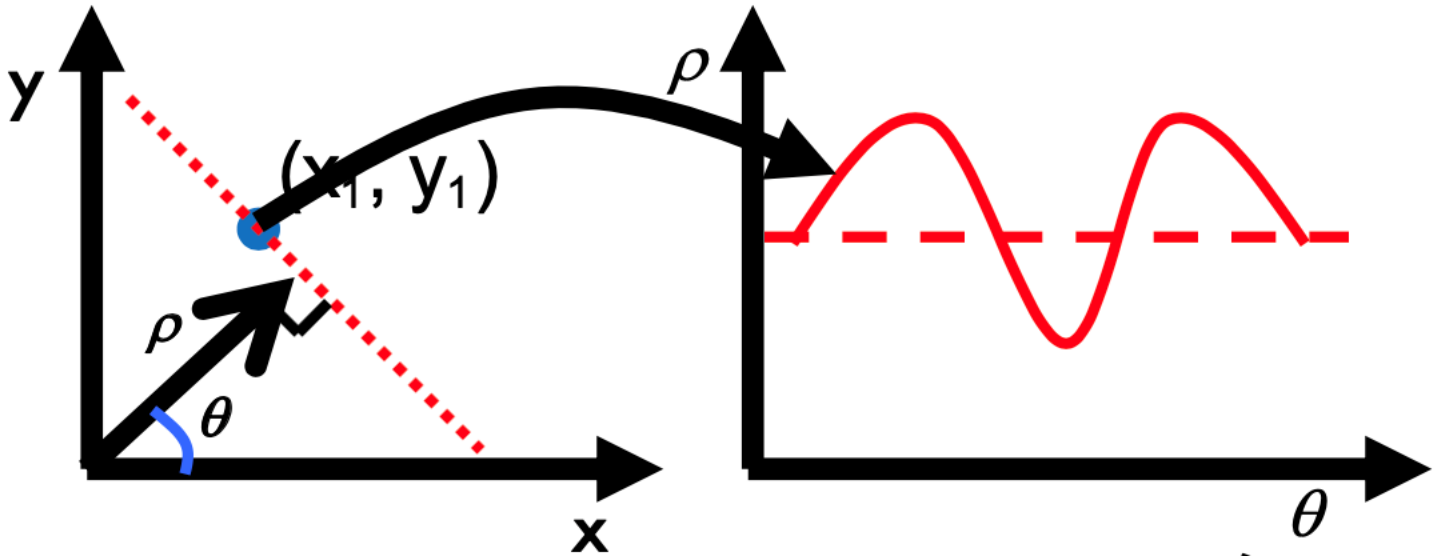
$$x\cos\theta + y\sin\theta = \rho, \quad (0 \le \theta \le 180)$$



A point (x_1,y_1) is mapped to a sinusoid in the polar parameter space

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

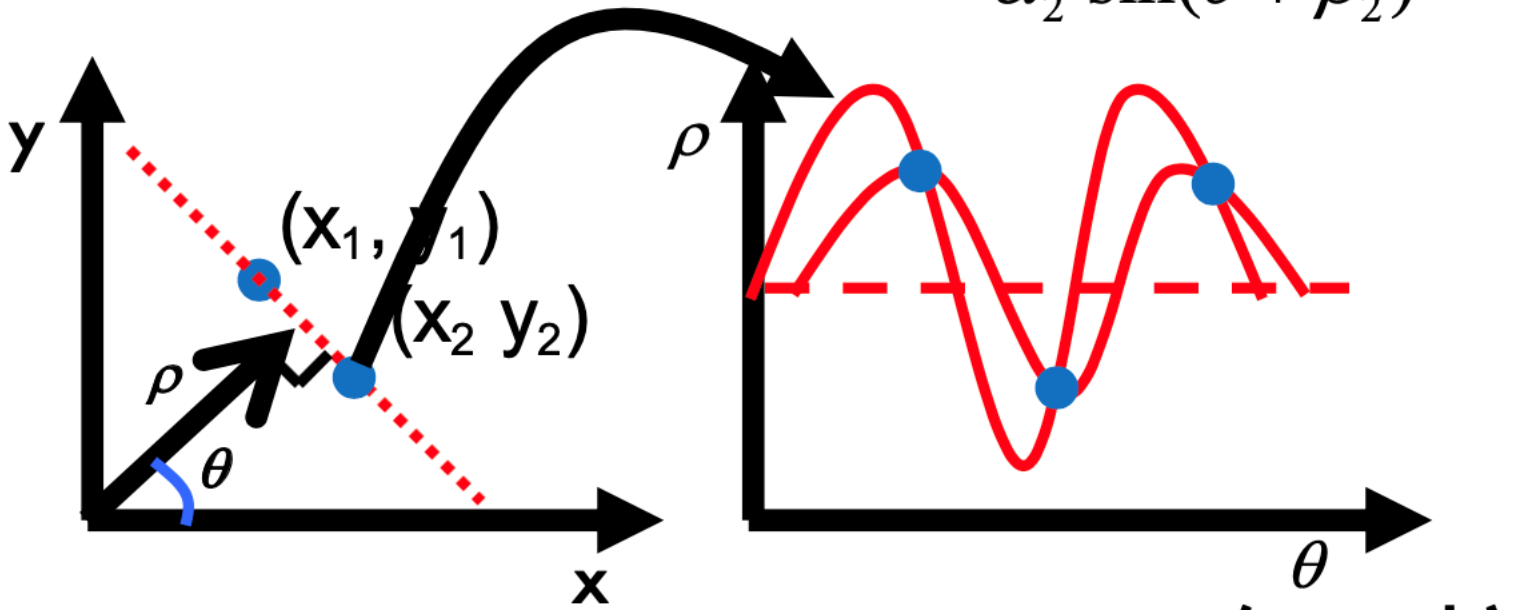
$$\rho = x_1 \cos\theta + y_1 \sin\theta$$
$$= \alpha_1 \sin(\theta + \beta_1)$$



A point (x_2,y_2) is mapped to a sinusoid in the polar parameter space

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

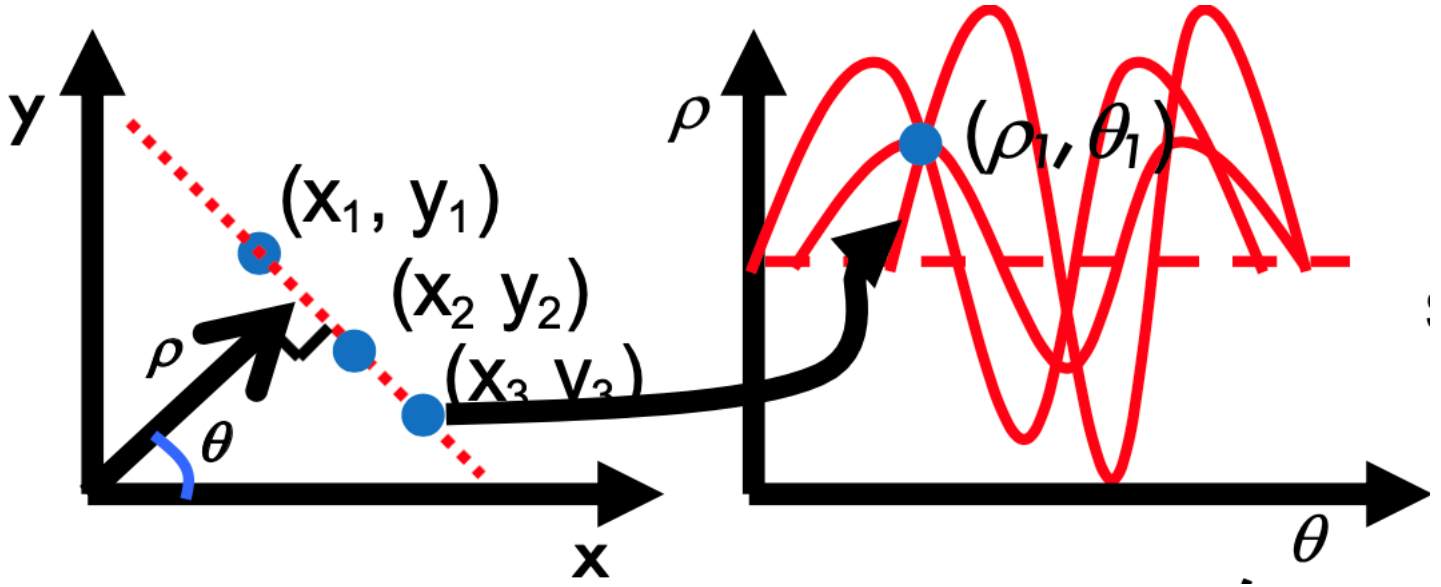
$$\rho = x_2 \cos\theta + y_2 \sin\theta$$
$$= \alpha_2 \sin(\theta + \beta_2)$$



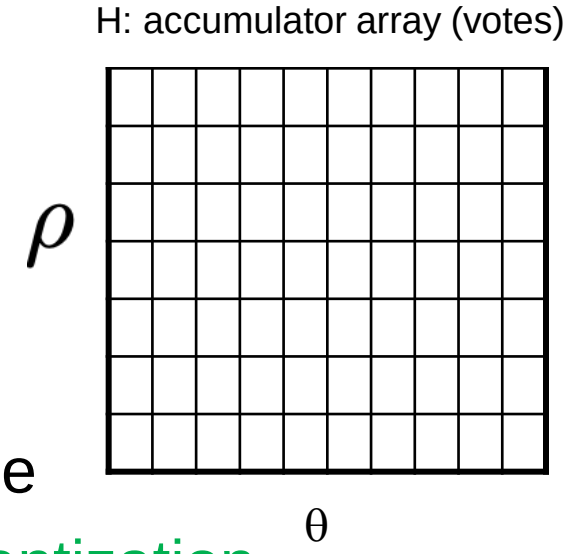
A point (x3,y3) is mapped to a sinusoid in the polar parameter space

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

$$\rho = x_3 \cos\theta + y_3 \sin\theta$$
$$= \alpha_3 \sin(\theta + \beta_3)$$



Hough Transform algorithm:



Using the polar parameterization:

Basic Hough transform algorithm

1. Initialize H[ρ, θ]=0
2. for each edge point I[x,y] in the image

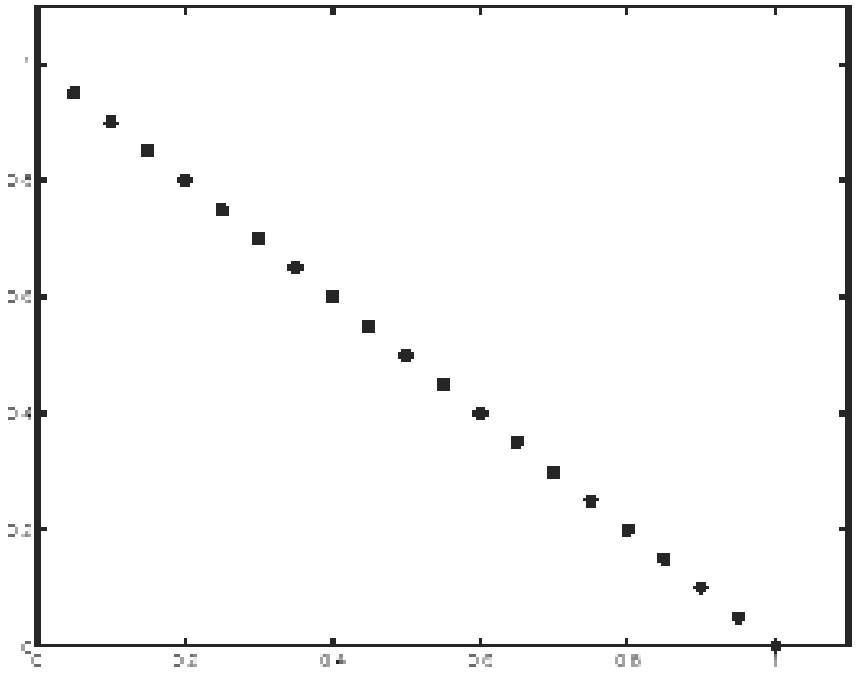
   for θ = [θ_{min} to θ_{max}] // some quantization

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

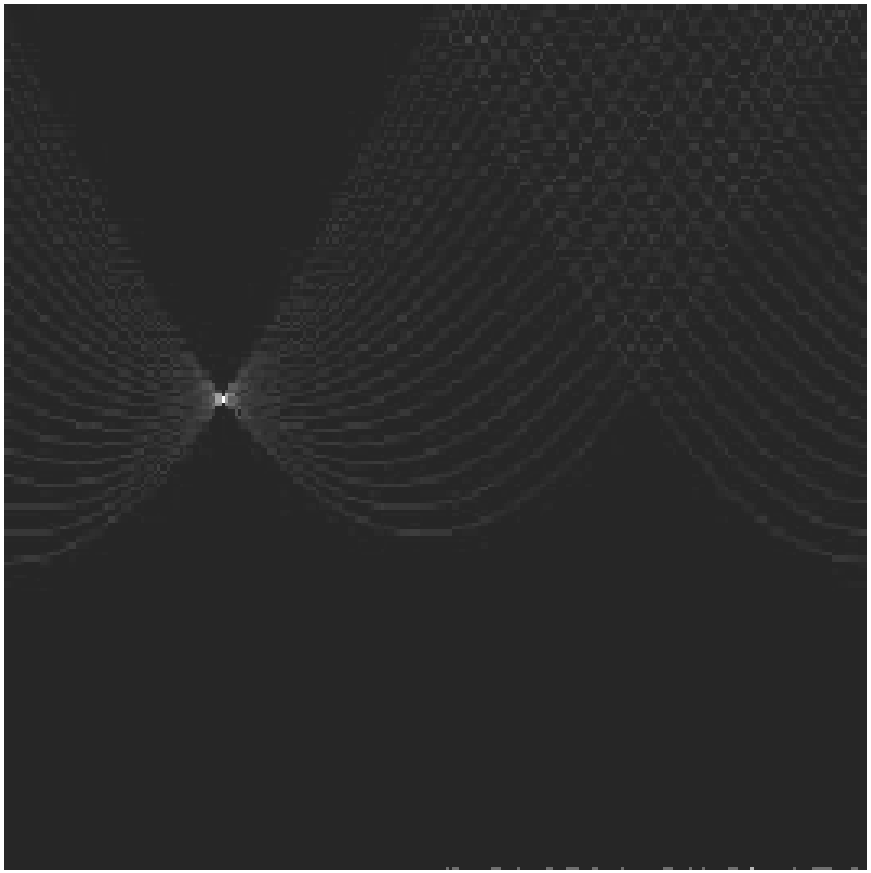
   H[ρ, θ] += 1
3. Find the value(s) of (ρ, θ) where H[ρ, θ] is maximum
4. The detected line in the image is given by

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

Hough transform - experiments



features

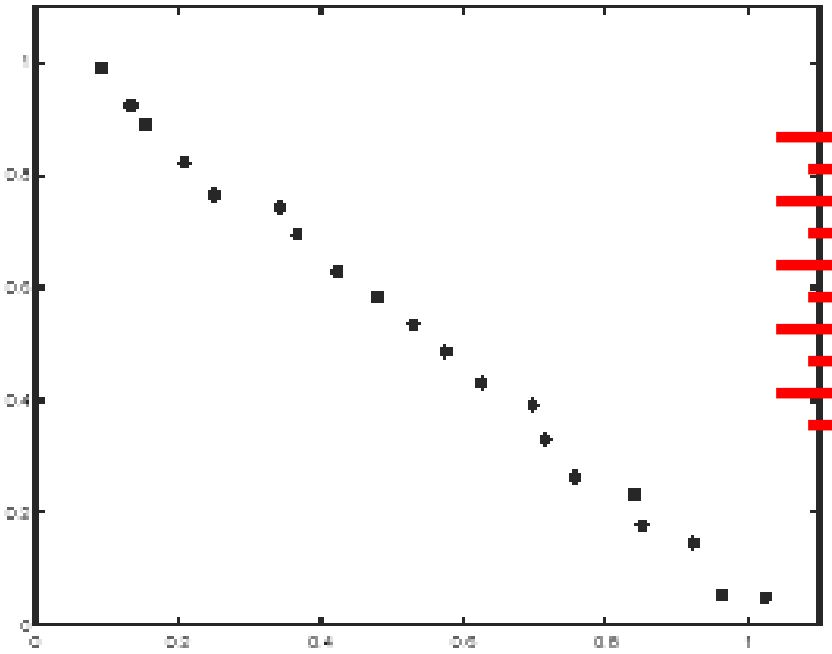


votes

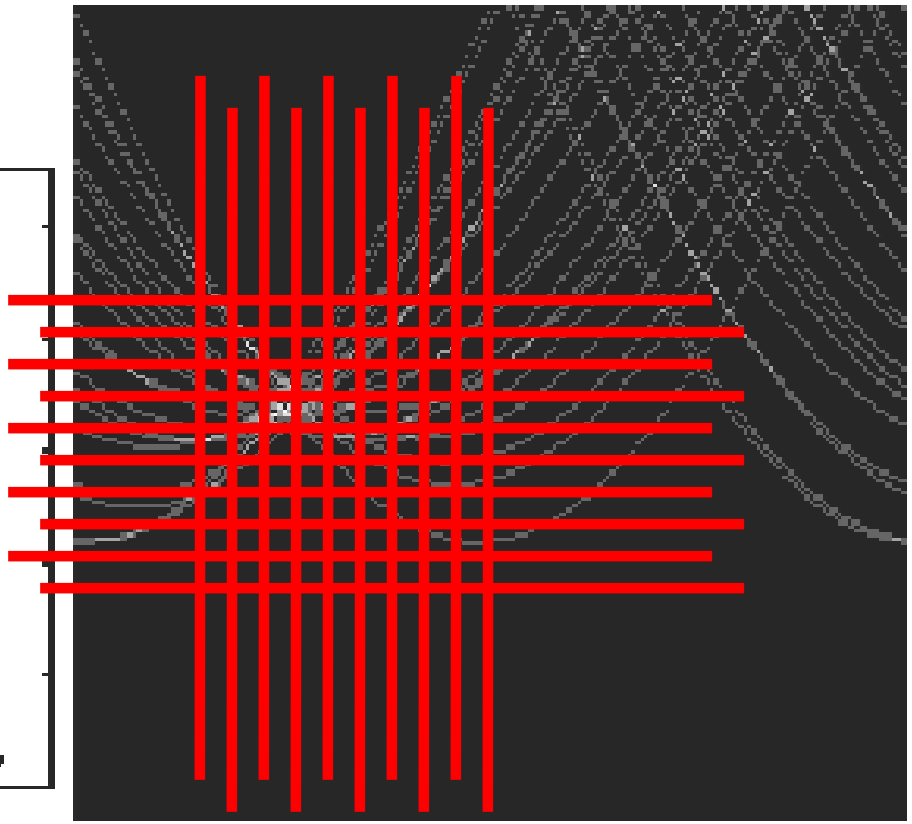
Slide from S. Savarese

Hough transform - experiments

Noisy data



features

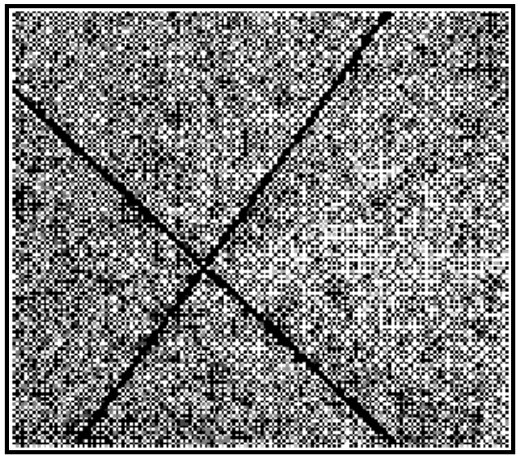


votes

Issue: Grid size needs to be adjusted…

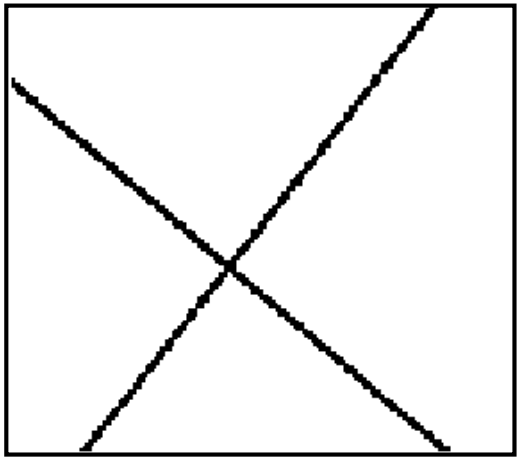
Slide from S. Savarese

Example: Hough Transform



Image

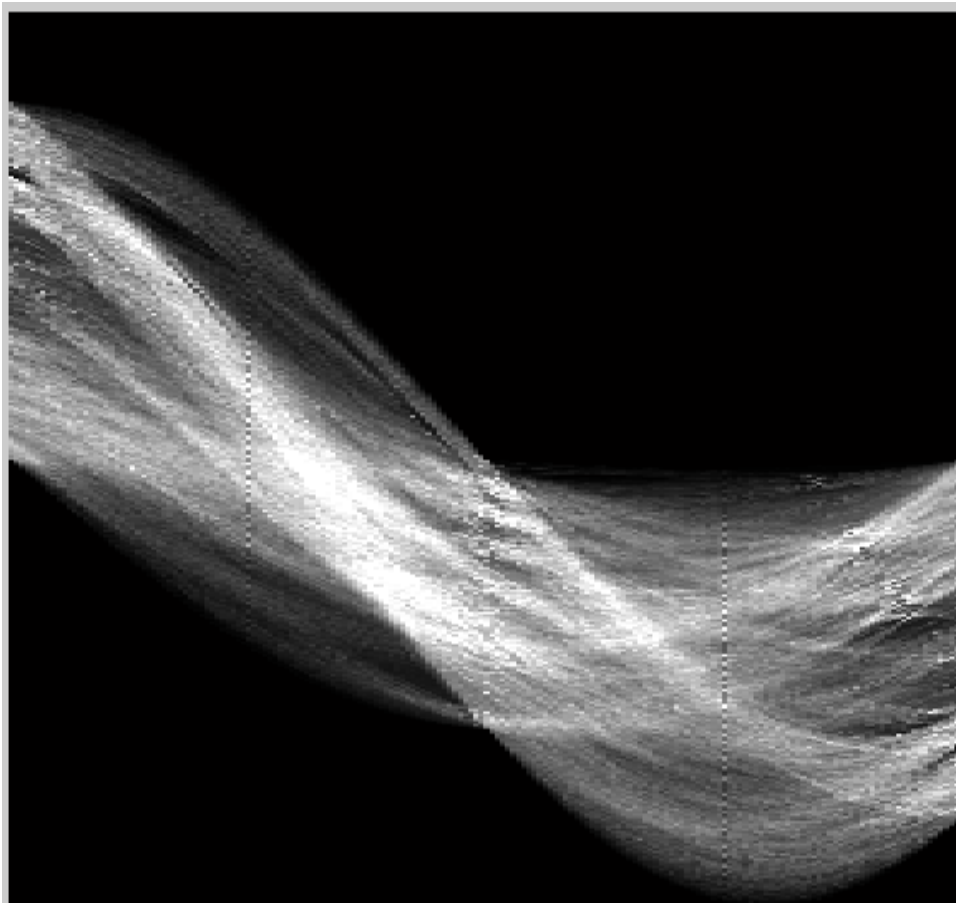
Edge detection



Hough Transform

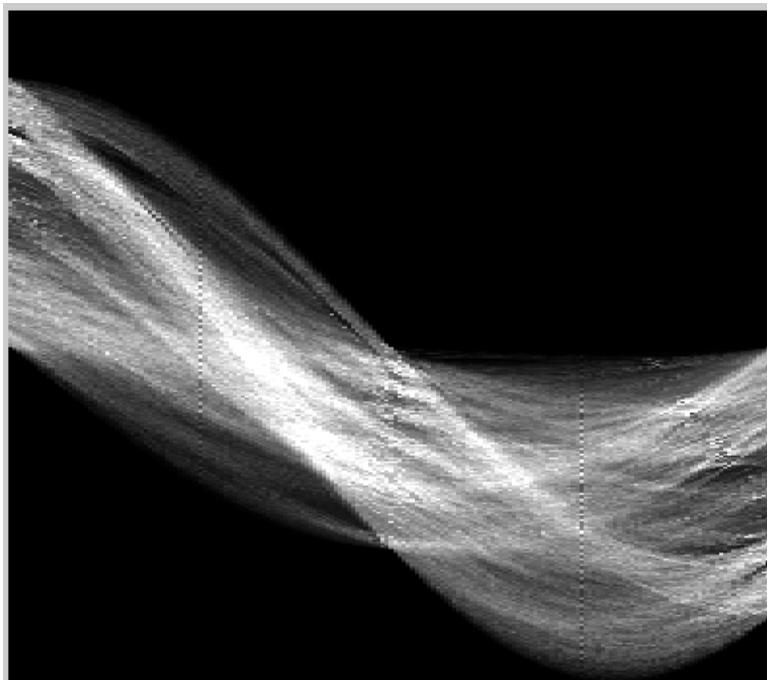
1. Image → Canny

2. Canny → Hough votes

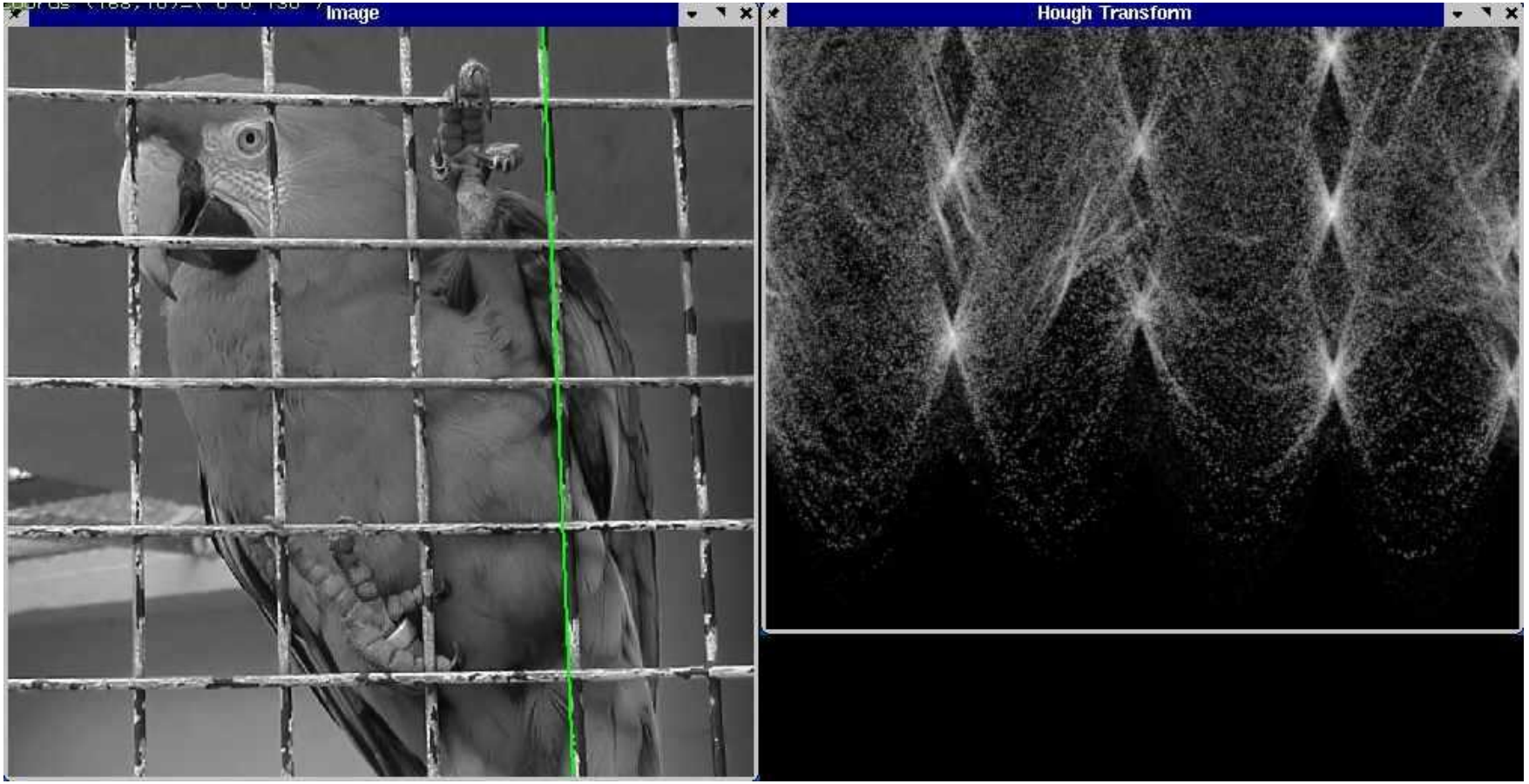


3. Hough votes → lines

Find peaks and post-process

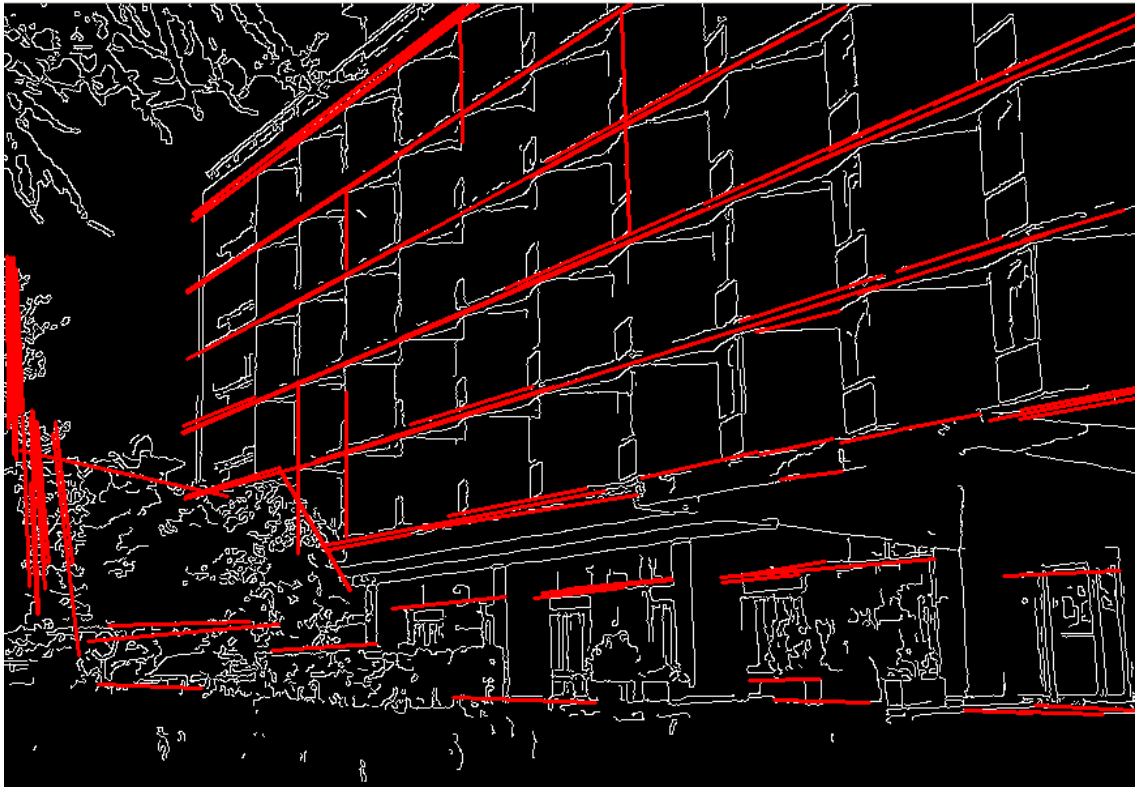


Hough transform example



http://ostatic.com/files/images/ss_hough.jpg

Hough transform examples



OpenCV: probabilistic hough transform

Hough transform pros and cons

Pros

- All points are processed independently, so can cope with occlusion
- Some robustness to noise: noise points unlikely to contribute consistently to any single bin
- Can detect multiple instances of a model in a single pass

Cons

- Complexity of search time increases exponentially with the number of model parameters
- Non-target shapes can produce spurious peaks in parameter space
- Quantization: hard to pick a good grid size

Content

- Least Squares Line Fitting
- Robust-M Estimator Line Fitting
- RANSAC

“Fitting” in general context

- Choose a parametric model to represent a set of features.

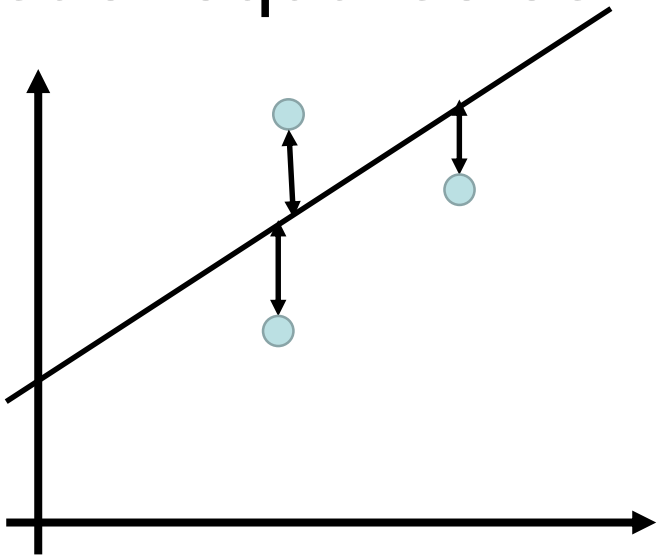
- Membership determination
  - Decide which feature belongs to which model.
  - Not a local decision; must look at the full image globally.

- Key questions
  - What is the best parametric model
  - How many models (e.g. lines) are there
  - Which point belongs to which line
  - Remove noisy points, wrong edge points.

Line fitting examples

Basic Line Fitting-Least Square

- Given a set of points *belong to* a straight line, find the equation of the line.
- Find the line parameters that yield minimal error in a least-squares sense.

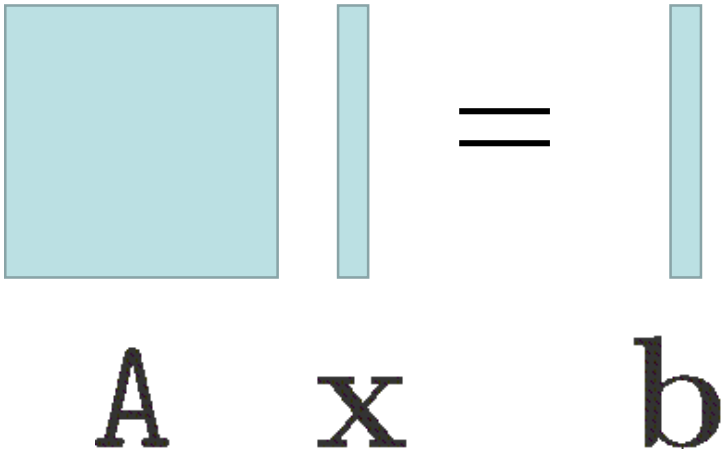


Least Square

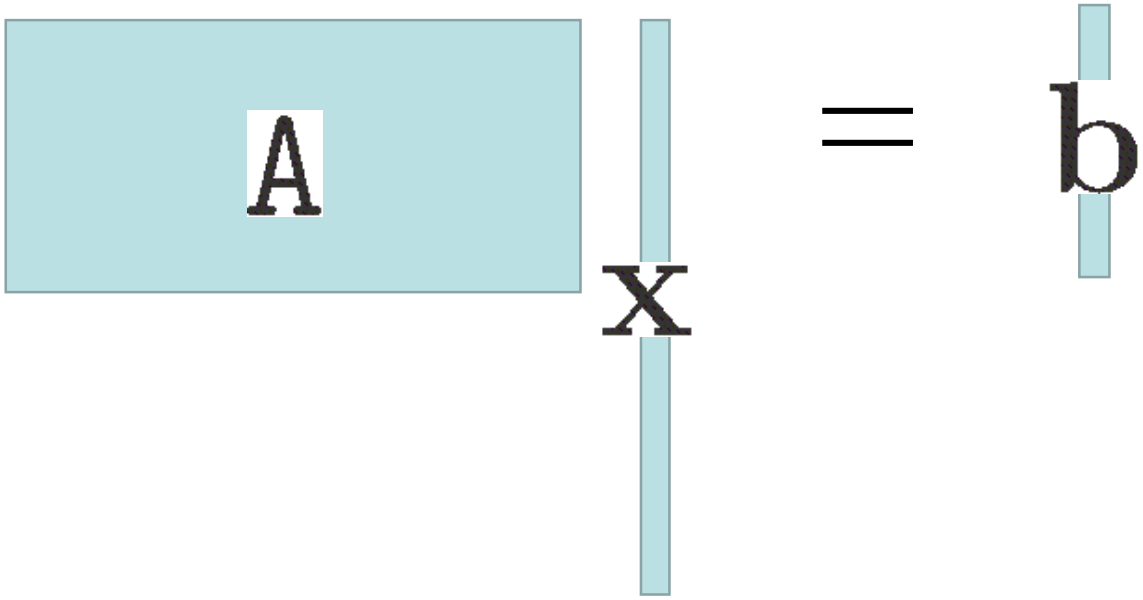
Solving equations

How do I solve an equation of the form

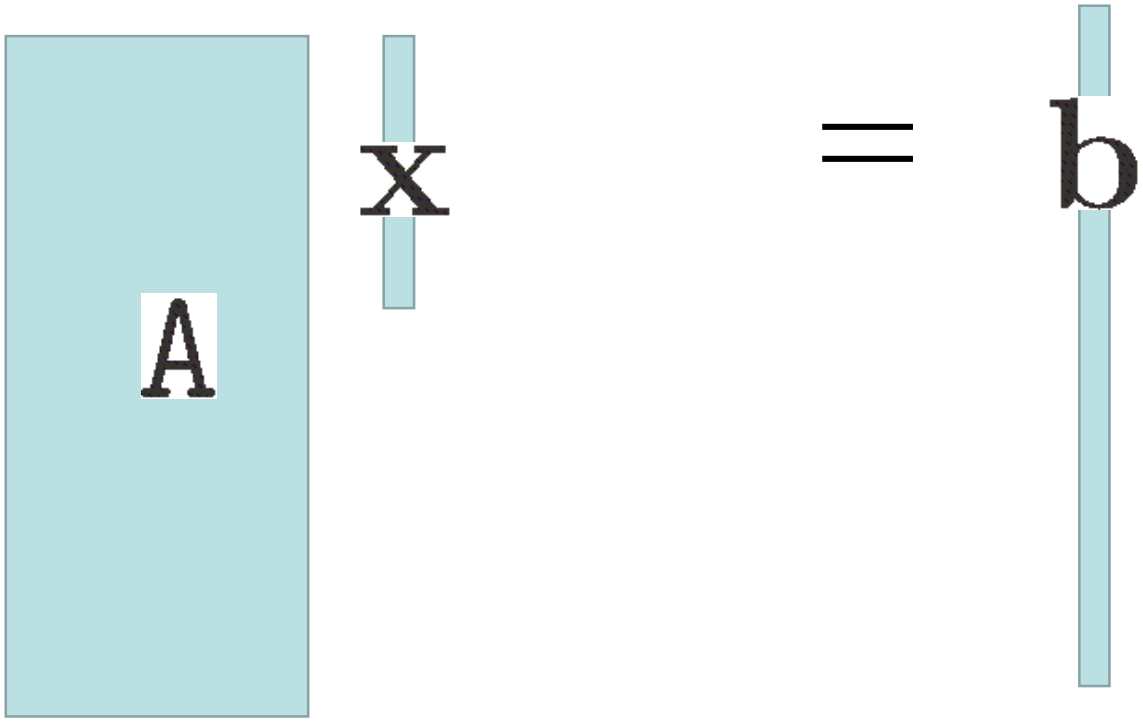
$$\mathbf{Ax} = \mathbf{b}$$



$$x = \mathbf{A}^{-1}\mathbf{b}$$

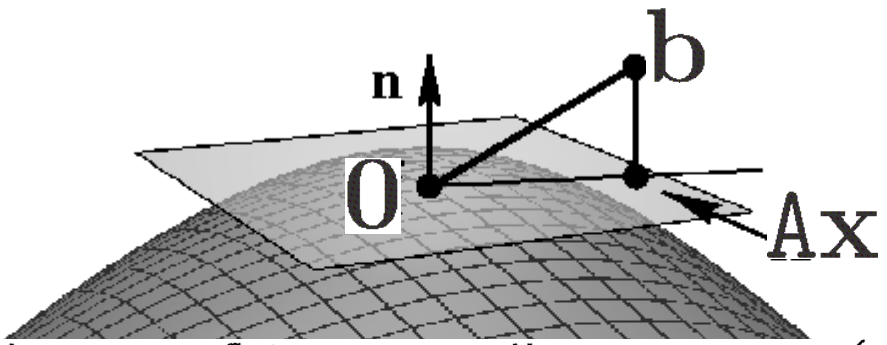


minimize $\|\mathbf{A}\mathbf{x} - \mathbf{b}\| + \lambda\|\mathbf{x}\|_1$



- Solution exists only if $\mathbf{b}$ is in the span of the columns of $\mathtt{A}$.
- Find a least-squares solution:

$$\text{minimize} \quad \|\mathbf{Ax} - \mathbf{b}\|^2$$



- Columns of $\mathtt{A}$ span a linear space (plane)
- $\mathtt{Ax}$ represents a point in the plane.
- Required point such that $\mathtt{Ax} - \mathbf{b}$ is perpendicular to plane.
- So

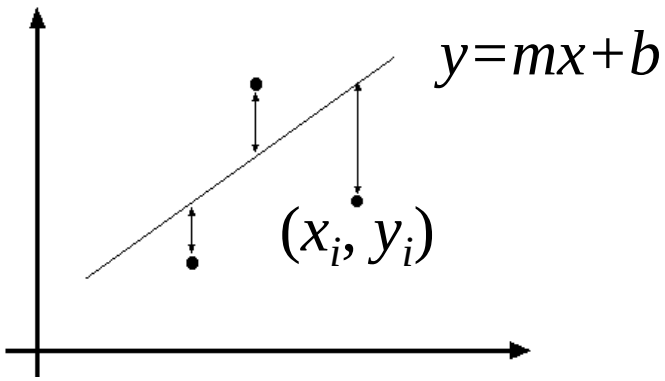
$$\mathtt{A}^\top(\mathtt{Ax} - \mathbf{b}) = 0$$

- So

$$\mathbf{x} = (\mathtt{A}^\top\mathtt{A})^{-1}\mathtt{A}^\top\mathbf{b}$$

Least squares line fitting

- Data: $(x_1, y_1), \ldots, (x_n, y_n)$
- Line equation:
  - $y_i = m\,x_i + b$

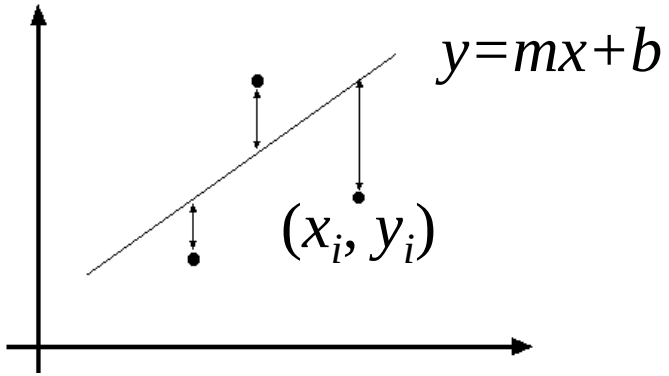


- Find **(m, b)** to minimize

$$E = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$

Least squares line fitting

$$E = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$



Solution:

Step 1: Define fitting error for each point as

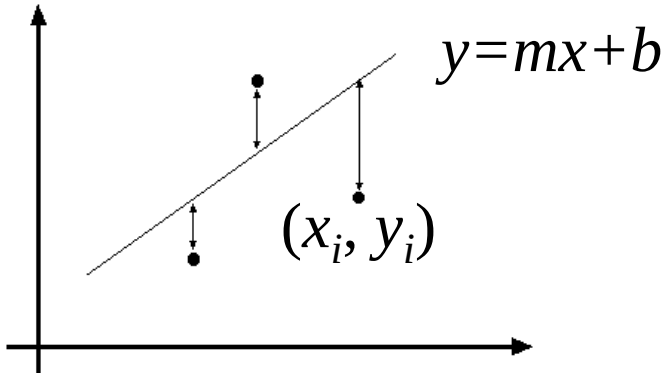
$$e_i = \begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - y_i$$

Step 2: The total fitting error for all the points are

$$\mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_n \end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix}, \mathbf{p} = \begin{bmatrix} m \\ b \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}$$

Least squares line fitting

$$E = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$



Solution:

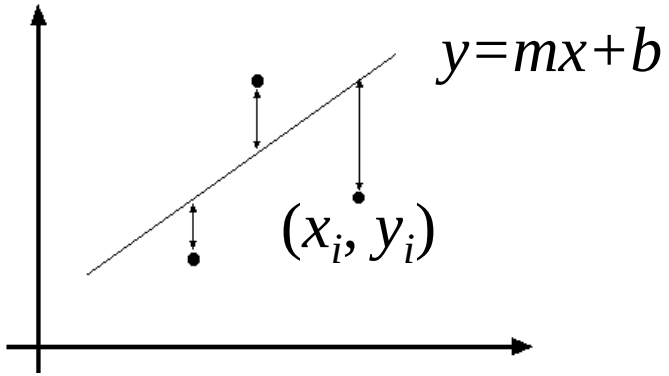
Step 2: The total fitting error for all the points are

$$\mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_n \end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \mathbf{Ap} - \mathbf{y}$$

Where, $\mathbf{A} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix}, \mathbf{p} = \begin{bmatrix} m \\ b \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}$

Least squares line fitting

$$E = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$



Solution:

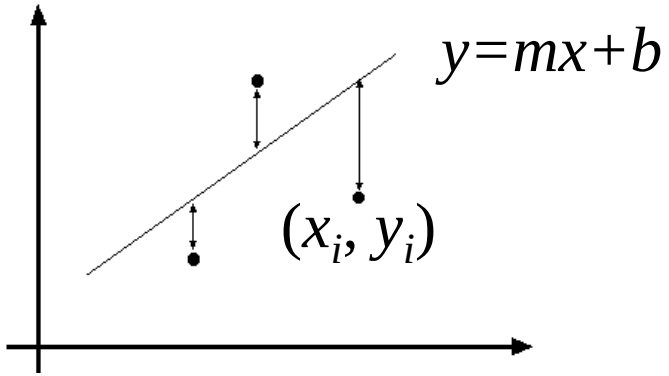
Step 3: The errors for all points are

$$E = \sum_{i=1}^{n} (mx_i + b - y_i)^2 = \mathbf{e}^T\mathbf{e} = (\mathbf{Ap} - \mathbf{y})^T(\mathbf{Ap} - \mathbf{y})$$

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

Least squares line fitting

$$E = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$



Solution:

Step 3: find solution to $\mathbf{p}$, where $\mathbf{p} = \begin{bmatrix} m \\ b \end{bmatrix}$

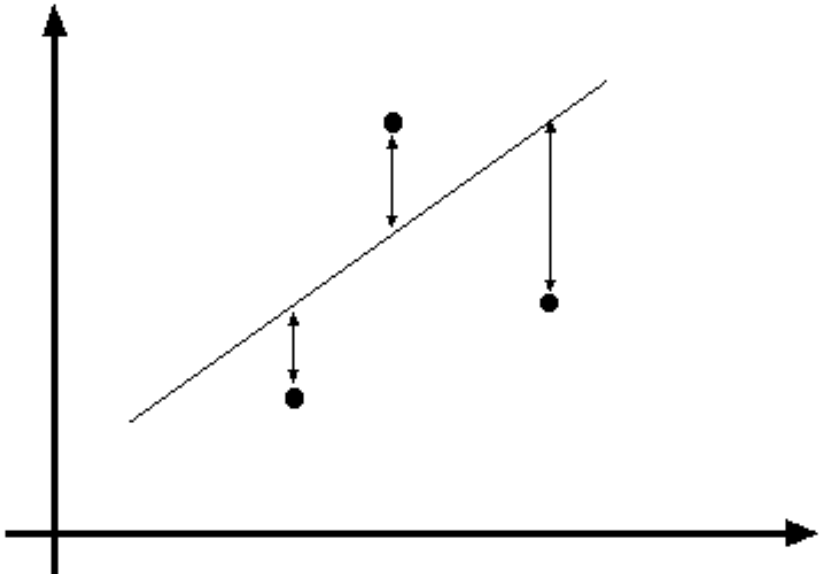
$$\frac{dE}{d\mathbf{p}} = 2\mathbf{A}^T\mathbf{A}\mathbf{p} - 2\mathbf{A}^T\mathbf{y} = 0$$

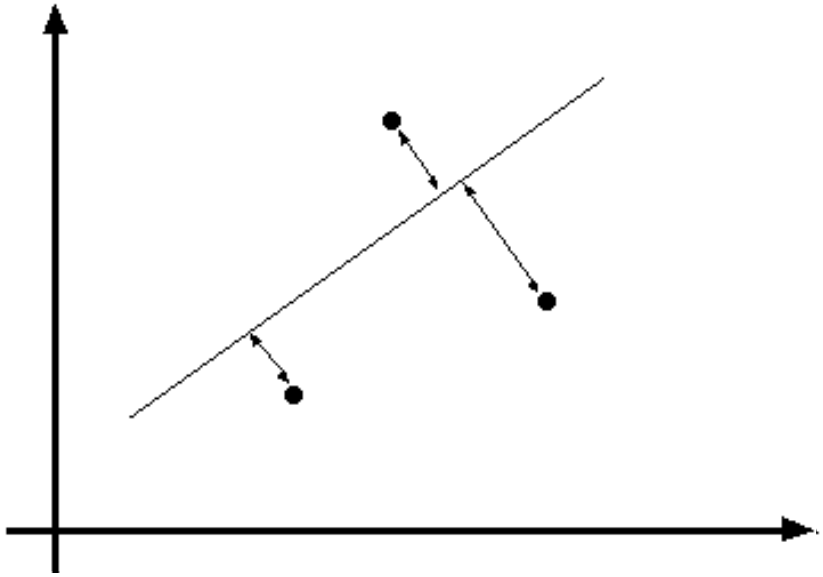
$$\Longrightarrow \quad \mathbf{p} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}$$

https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf

Issues with LS fitting

- Not rotation-invariant
- Fails completely for vertical lines
- Modified least squares using a perpendicular distance



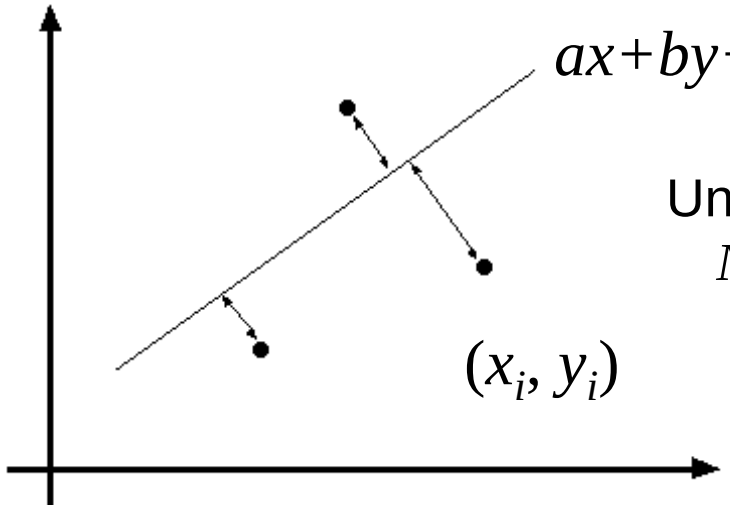


Slide from S. Lazebnik

Orthogonal regression

If ($a^2+b^2=1$) then the distance between point (x_i, y_i) to line is

$$|ax_i + by_i + c|$$



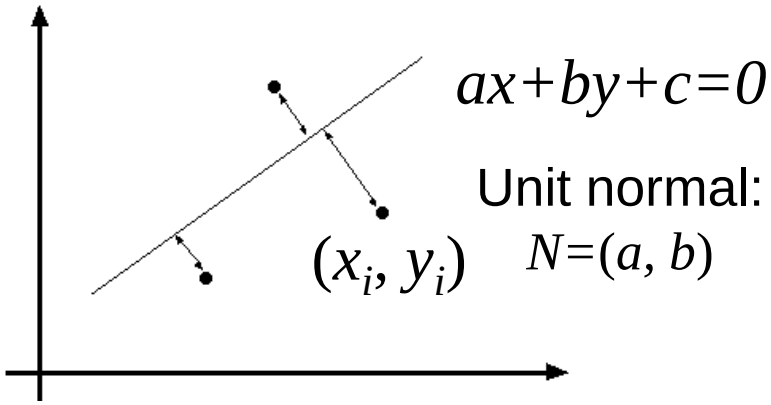
Total least squares

If ($a^2+b^2=1$) then the distance between point (x_i, y_i) to the line is

$$|ax_i + by_i + c|$$

Find (a, b, c) to minimize the sum of squared perpendicular distances

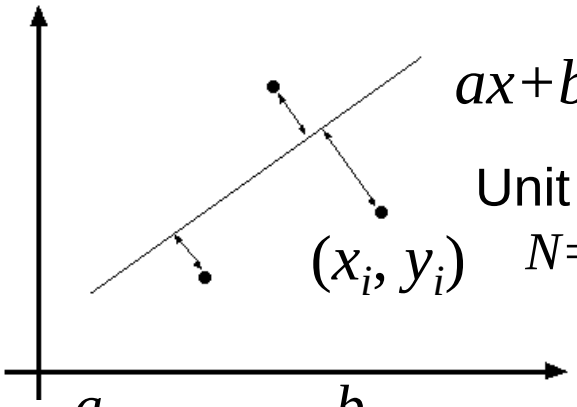
$$E = \sum_{i=1}^{n} (a x_i + b y_i + c)^2$$



Total least squares

Find (a, b, c) to minimize the sum of squared perpendicular distances

$$E = \sum_{i=1}^{n} (a x_i + b y_i + c)^2$$



$$\frac{\partial E}{\partial c} = \sum_{i=1}^{n} -2(a x_i + b y_i + c) = 0$$

$$c = -\frac{a}{n}\sum_{i=1}^{n} x_i - \frac{b}{n}\sum_{i=1}^{n} x_i = -a\bar{x} - b\bar{y}$$

$$E = \sum_{i=1}^{n} (a(x_i - \bar{x}) + b(y_i - \bar{y}))^2 = \left\| \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\|^2 = \mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p}$$

$$\text{minimize} \mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p} \quad \text{s.t. } \mathbf{p}^T \mathbf{p} = 1 \quad \Rightarrow \quad \text{minimize} \frac{\mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p}}{\mathbf{p}^T \mathbf{p}}$$

Solution is the eigenvector corresponding to the smallest eigenvalue of A^TA.

Recap: Two Common Optimization Problems

LS Problem statement

$$\text{minimize } \|\mathbf{Ax}-\mathbf{b}\|^2$$

$$\text{least squares solution to } \mathbf{Ax}=\mathbf{b}$$

Solution

$$\mathbf{x}=\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$$

$$\mathbf{x}=\mathbf{A}\setminus\mathbf{b} \quad \text{(matlab)}$$

TLS Problem statement

$$\text{minimize } \mathbf{x}^T\mathbf{A}^T\mathbf{Ax} \quad \text{s.t. } \mathbf{x}^T\mathbf{x}=1$$

$$\text{minimize}\frac{\mathbf{x}^T\mathbf{A}^T\mathbf{Ax}}{\mathbf{x}^T\mathbf{x}}$$

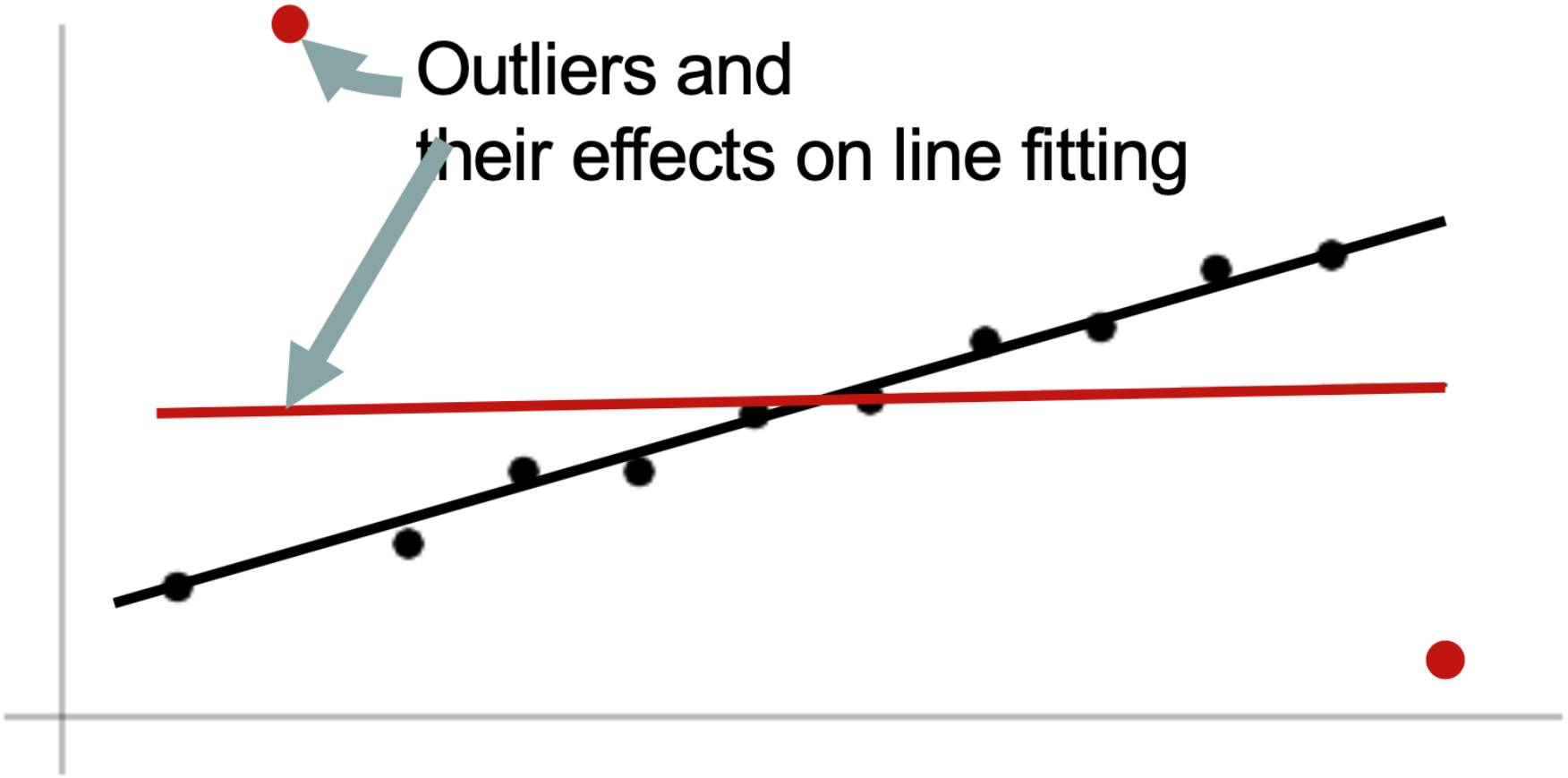
$$\text{non-trivial TLS solution to } \mathrm{A}\,\mathbf{x}=0$$

Solution

$$[\mathbf{v},\lambda]=\text{eig}(\mathbf{A}^T\mathbf{A})$$

$$\lambda_1<\lambda_{2..n}:\mathbf{x}=\mathbf{v}_1$$

Basic line fitting: Not Robust to Outliers

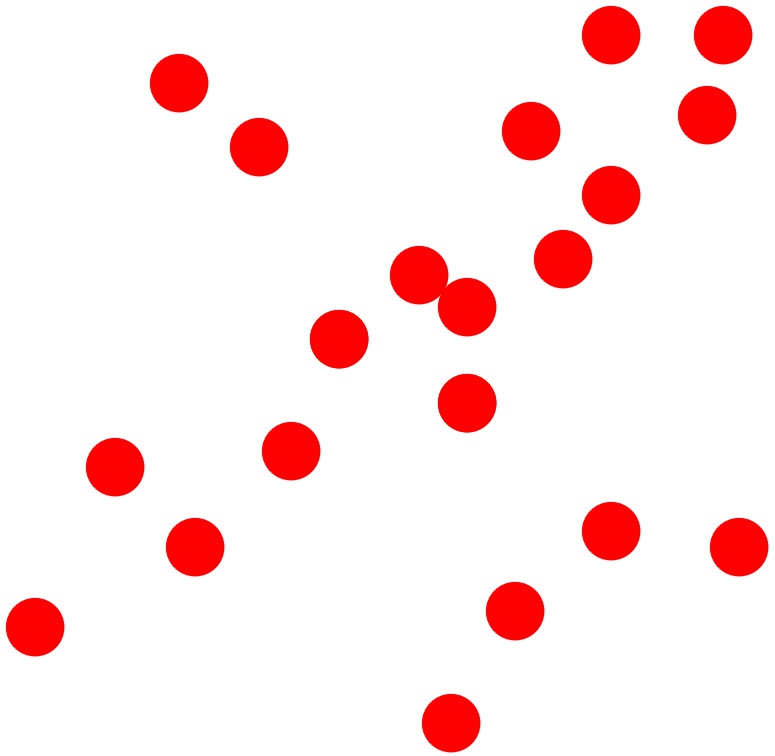


RANSAC Line fitting

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in ‘81.



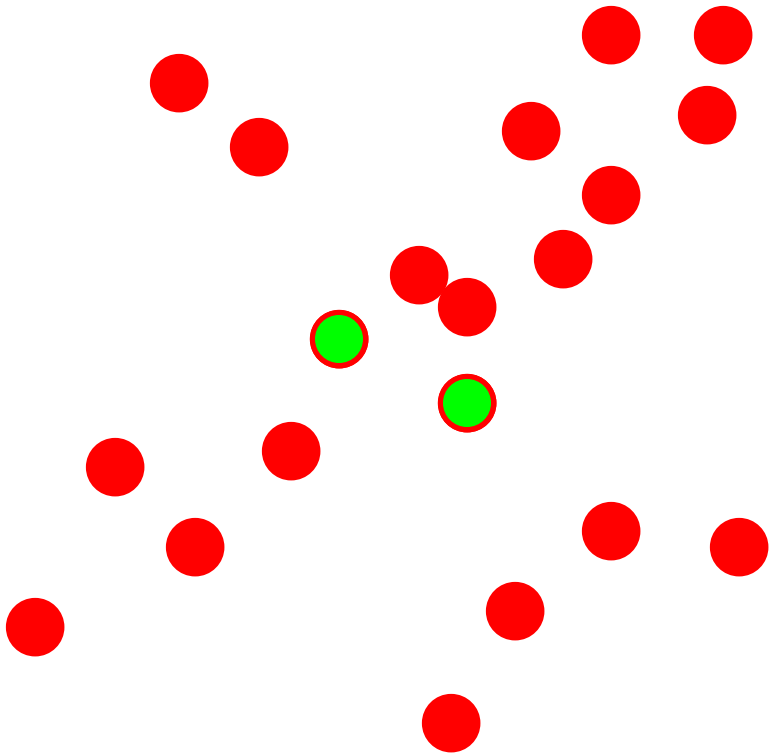
Algorithm:

1. Randomly **sample** the minimum number of points required to fit the model
2. **Solve** for model parameters using these minimum set of samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



Algorithm:

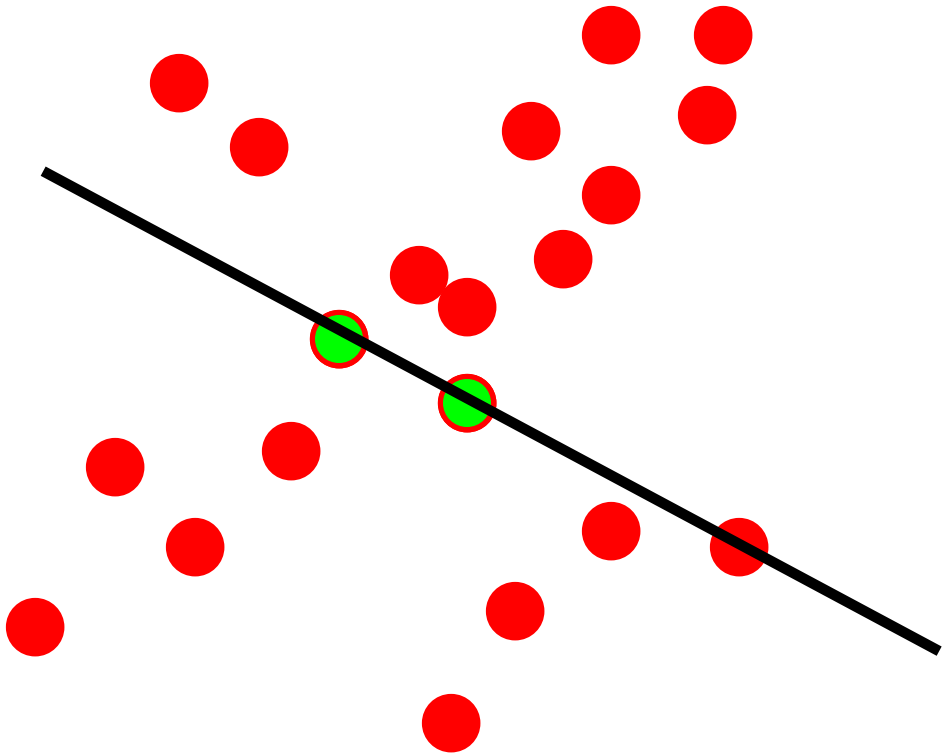
1. **Randomly sample** the minimum number of points required to fit the model (#=2)
2. **Solve** for model parameters using the minimum set of samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Illustration by Savarese

RANSAC

Line fitting example



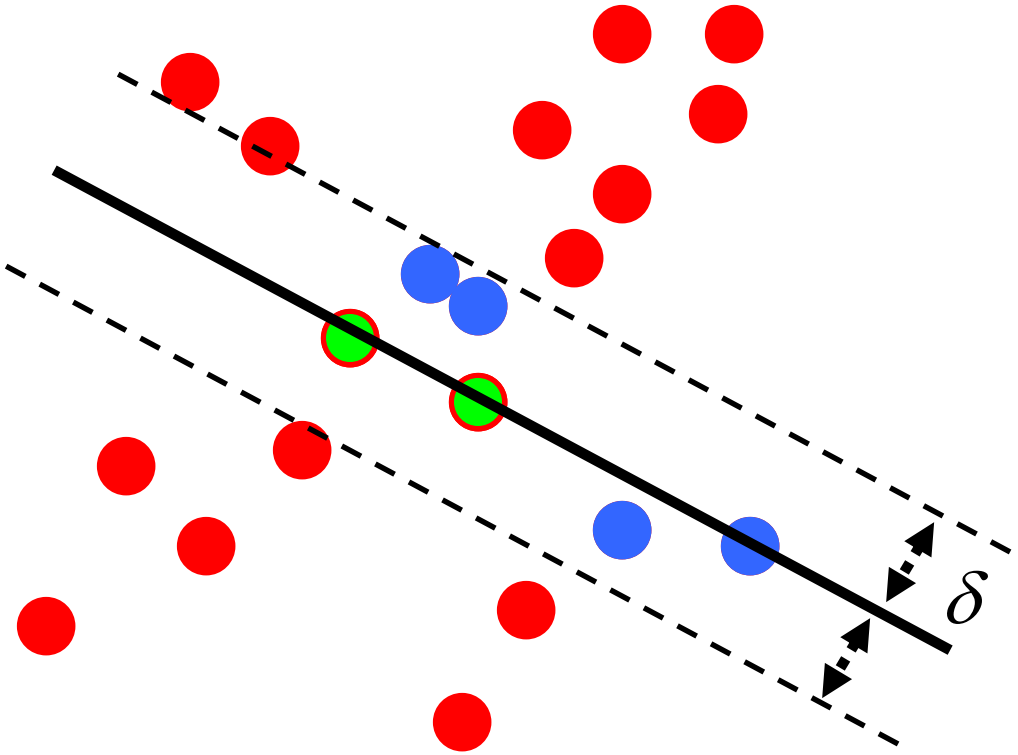
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



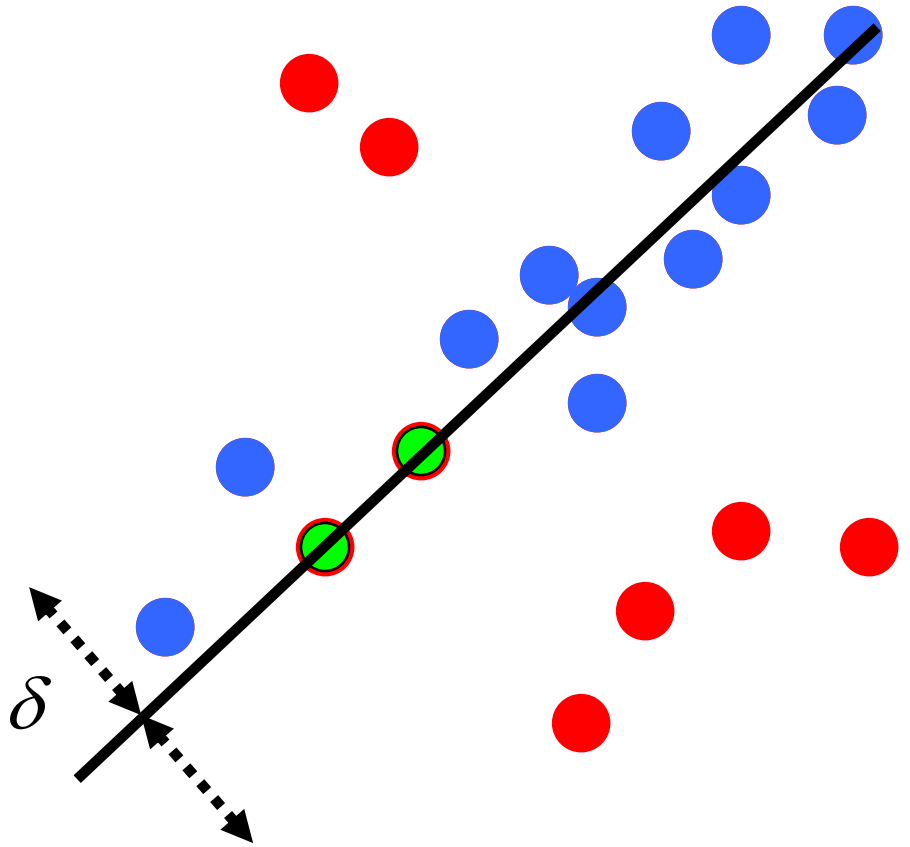
$$N_I = 6$$

Algorithm:

1. **Sample** randomly the number of points required to fit the model (#=2)
2. **Solve** for model parameters using the minimum set of samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC



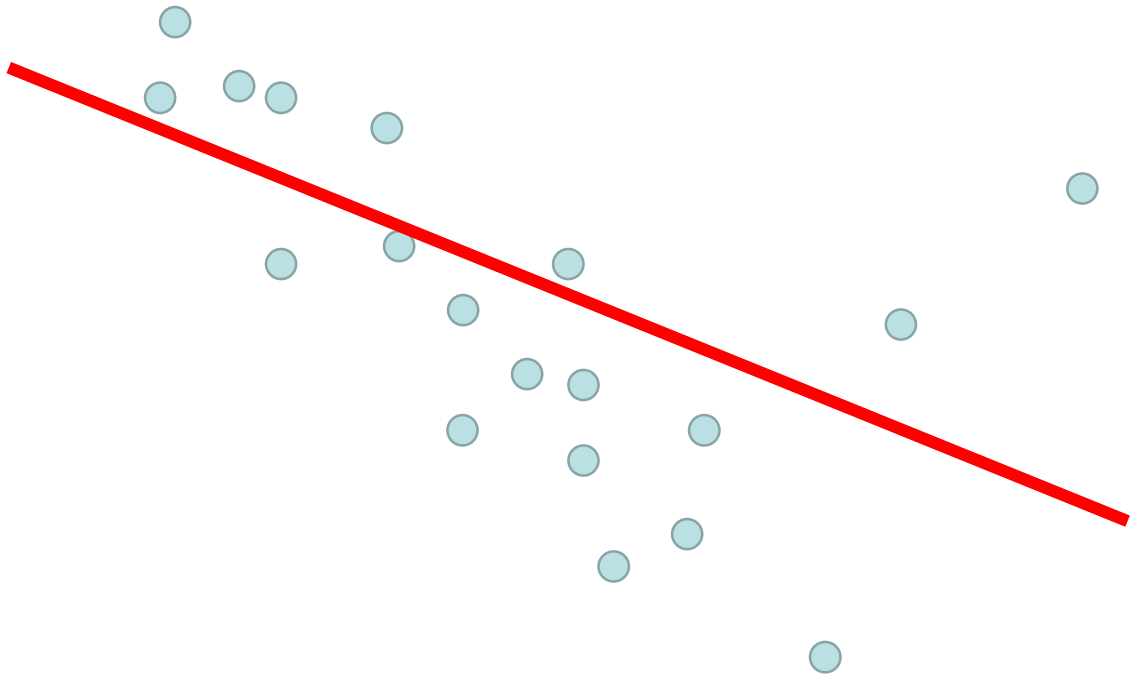
$N_I = 14$

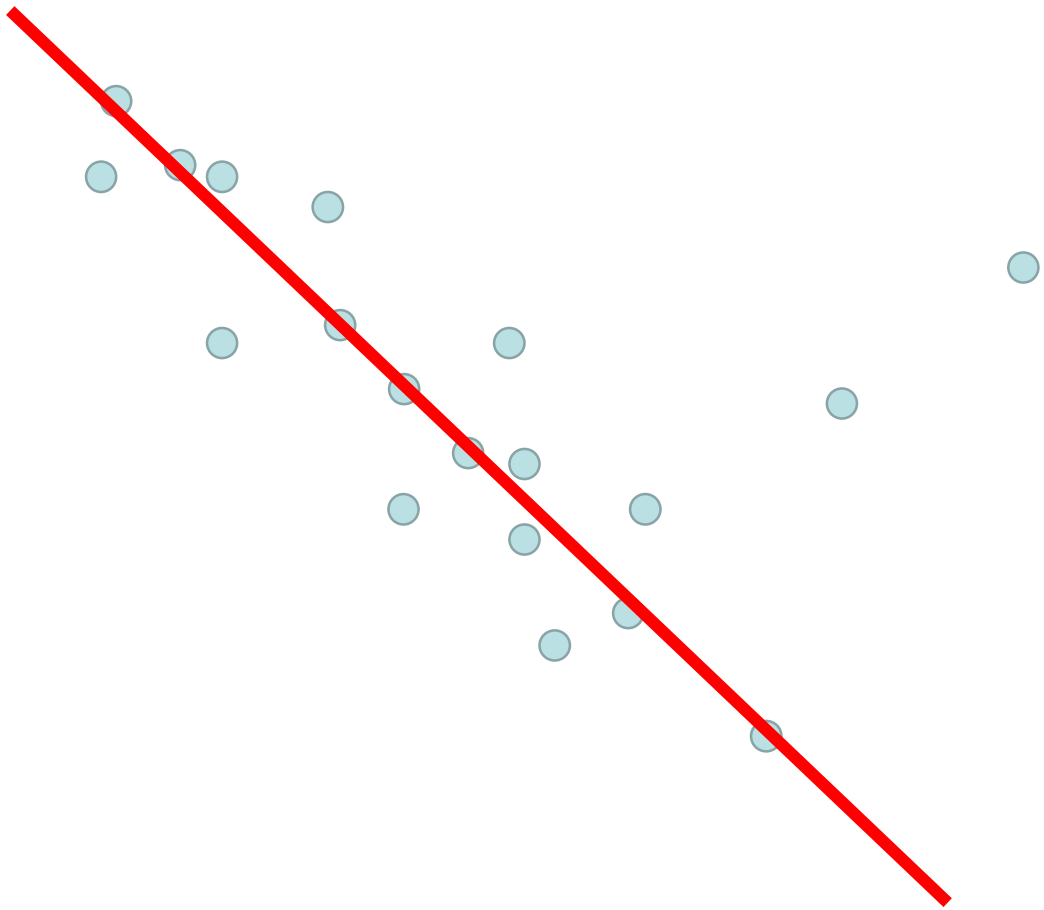
Algorithm:

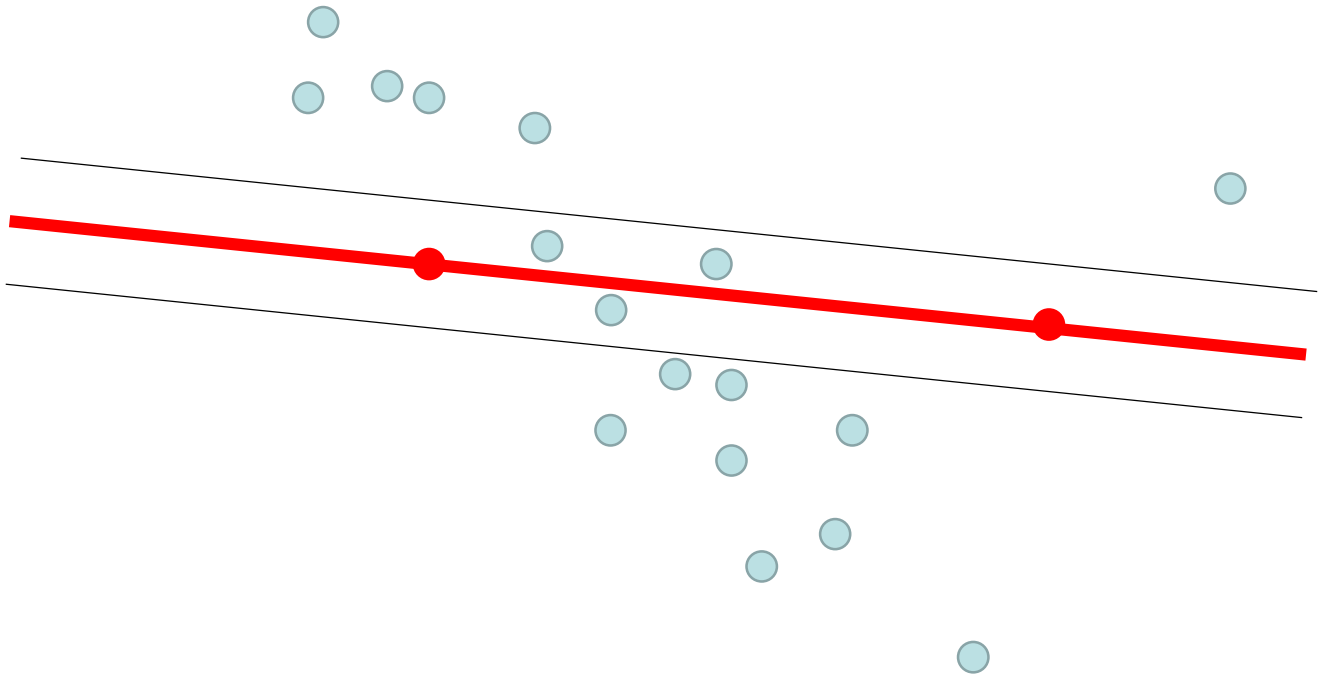
1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using the minimum set of samples
3. **Score** by the fraction of inliers within a preset threshold of the model

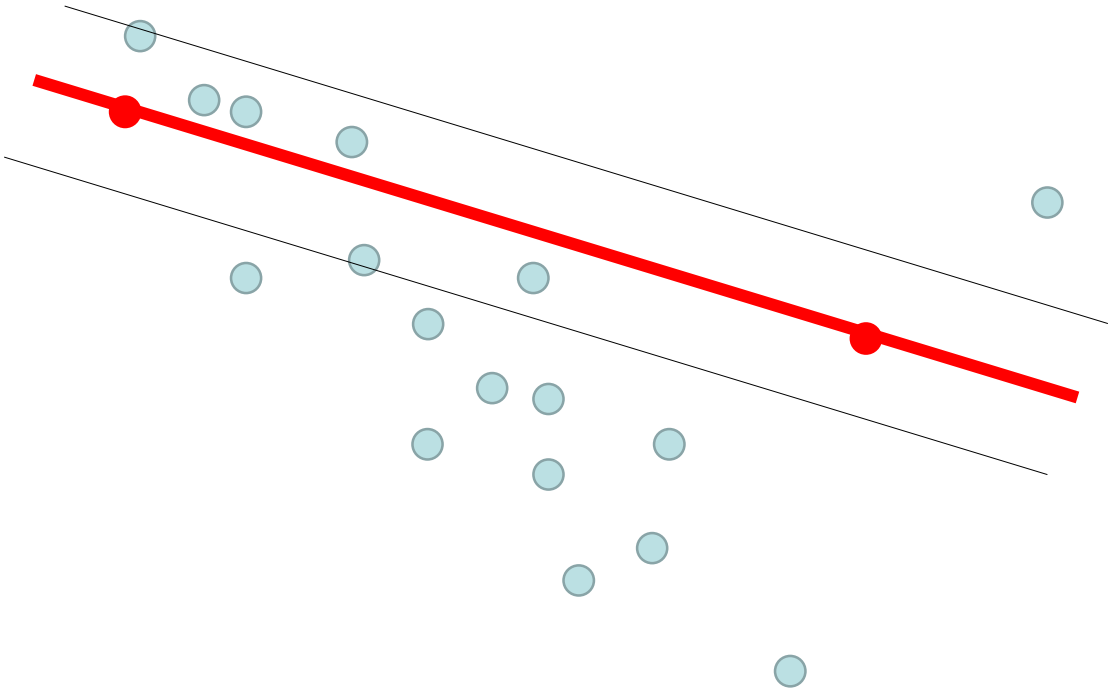
Repeat 1-3 until the best model is found with high confidence

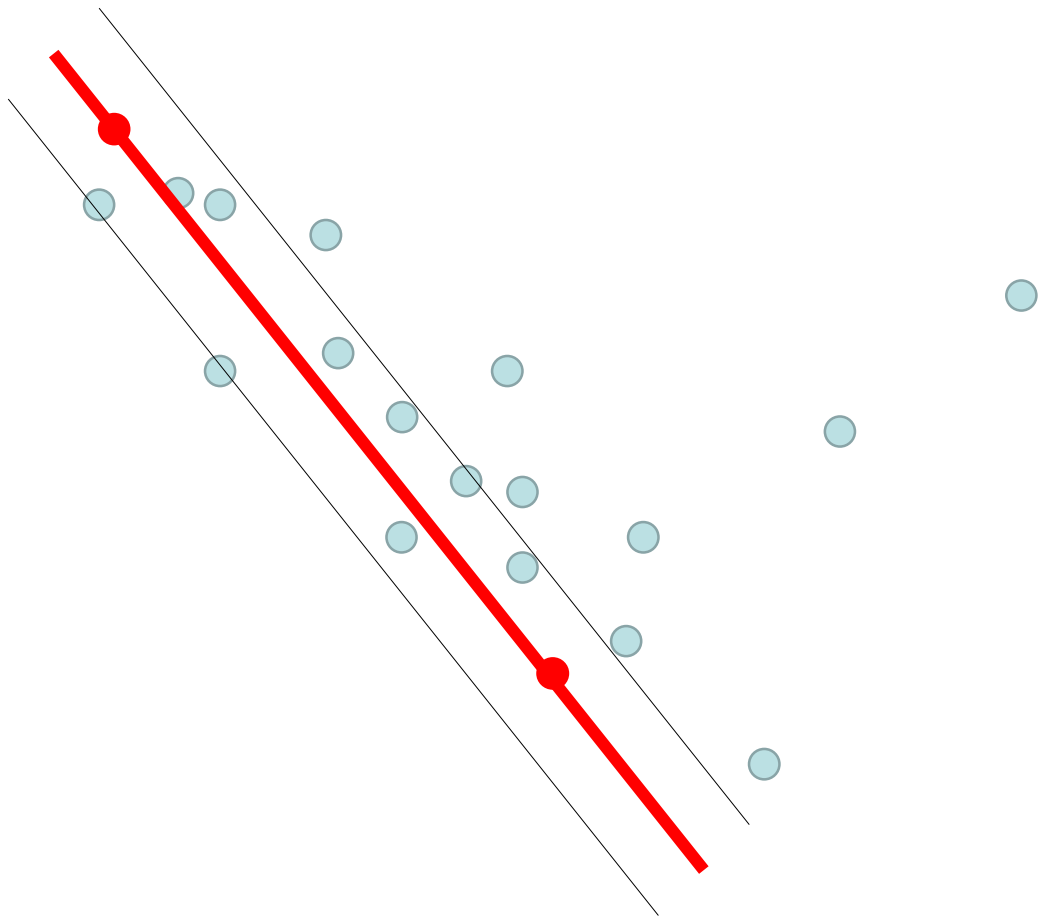
RANSAC -- Random Sample Consensus











How to choose parameters?

- Number of samples *N*
  - Choose *N* so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)
- Number of sampled points *s*
  - Minimum number needed to fit the model
- Distance threshold δ
  - Choose δ so that a good point with noise is likely (e.g., prob=0.95) within threshold

$$N = \log(1-p) / \log\left(1-(1-e)^s\right)$$

Issues.

- Model should be small (small number of d.o.f)
- Adaptive RANSAC (adjusting the threshold)
- Premptive RANSAC (saves time)

RANSAC pros and cons

Pros

- Robust to outliers
- Applicable for large number of parameters
- Parameters are easy to choose

Cons

- Computational time grows quickly with fraction of outliers and number of parameters
- Not good for getting multiple fits

Common applications (for 3D vision)

  - Computing a homography (e.g., image stitching)
  - Estimating fundamental matrix (relating two views)

Iterative Reweighted Least Squares

Slides available on my home page:

Navigate to

***My papers*:**

- Publishers' versions of my papers.

ECCV, Sept 7, 2014

Line fitting – simple regression

- $y_i = mx_i + b$. Given (x_i, y_i), solve for a and b
- Write as

$$(x_i, 1) \begin{pmatrix} m \\ b \end{pmatrix} = y_i$$

- Several equation give

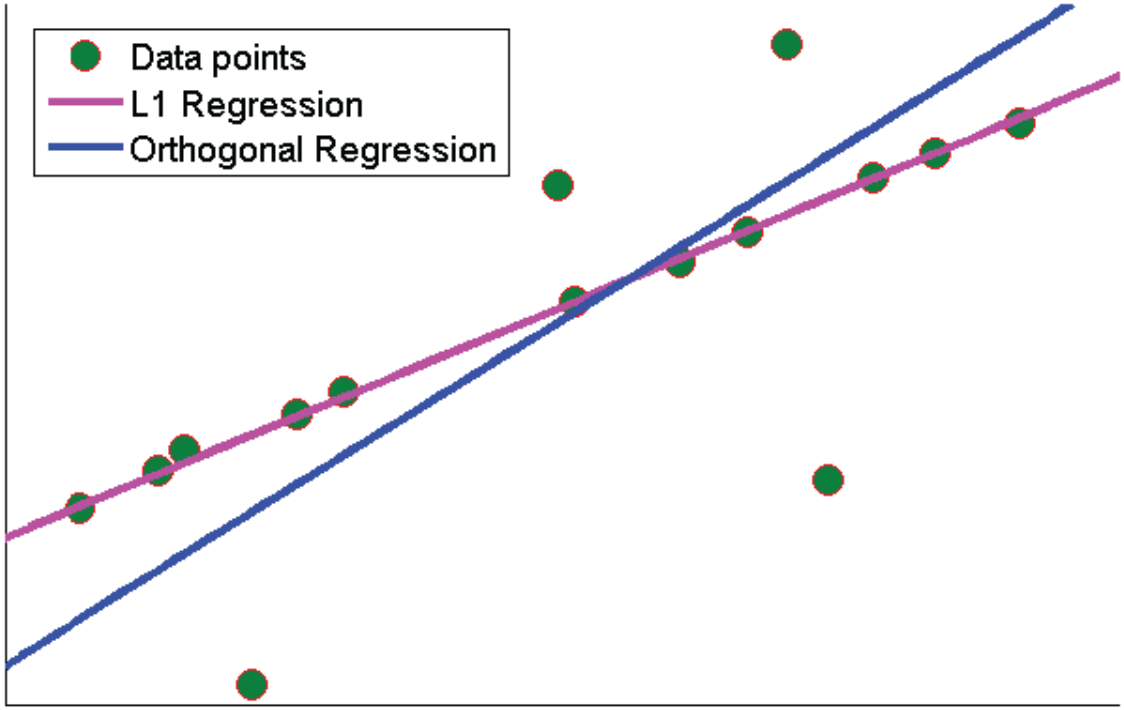
$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \\ x_n & 1 \end{bmatrix} \begin{pmatrix} m \\ b \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} .$$

- Least-squares regression.

NICTA

Line Fitting:

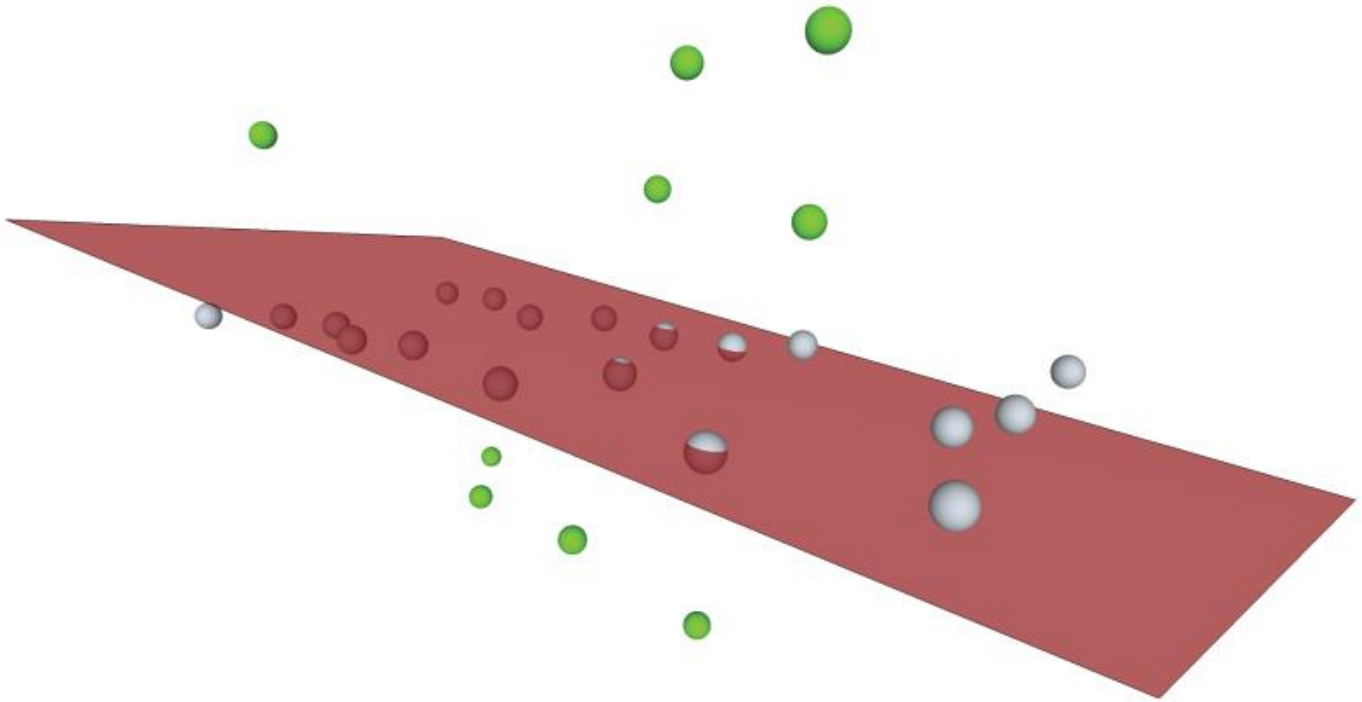
- L_1 regression is compared with L_2 regression.



Example: Regression

NICTA

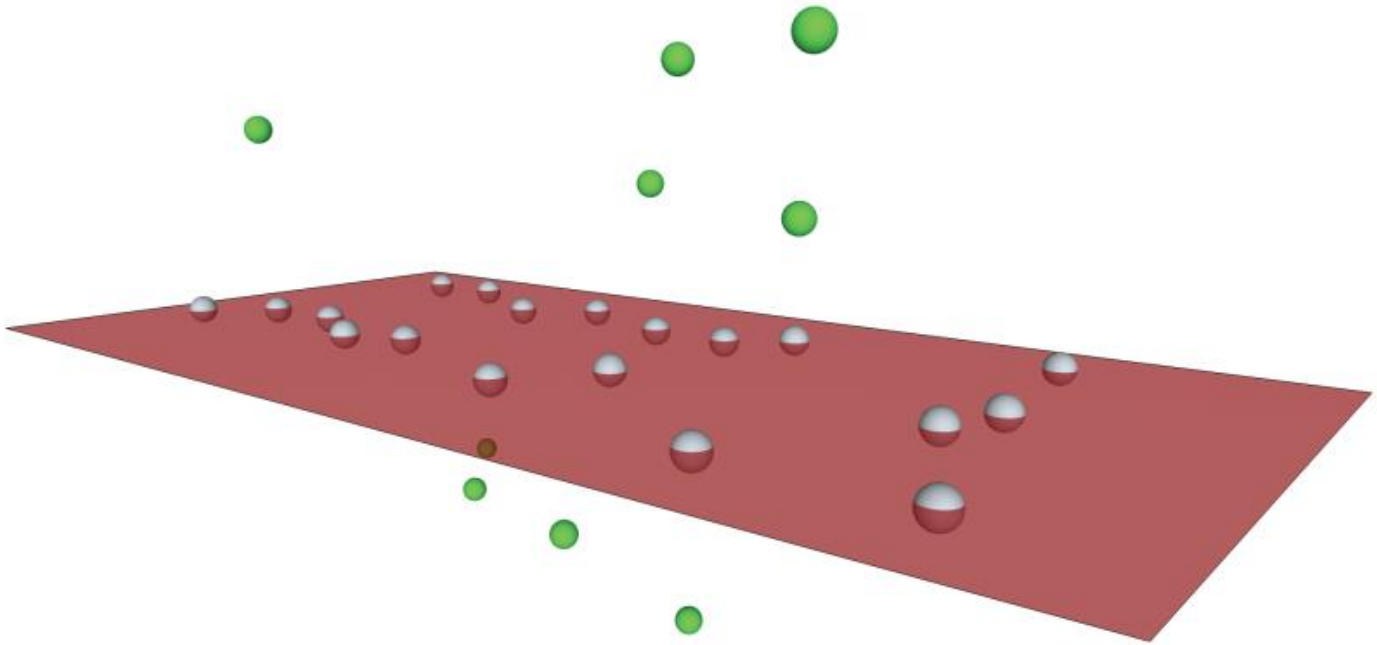
- Squared distance does not work in the presence of outliers.



Example: Regression

NICTA

- The ideal model should be like this

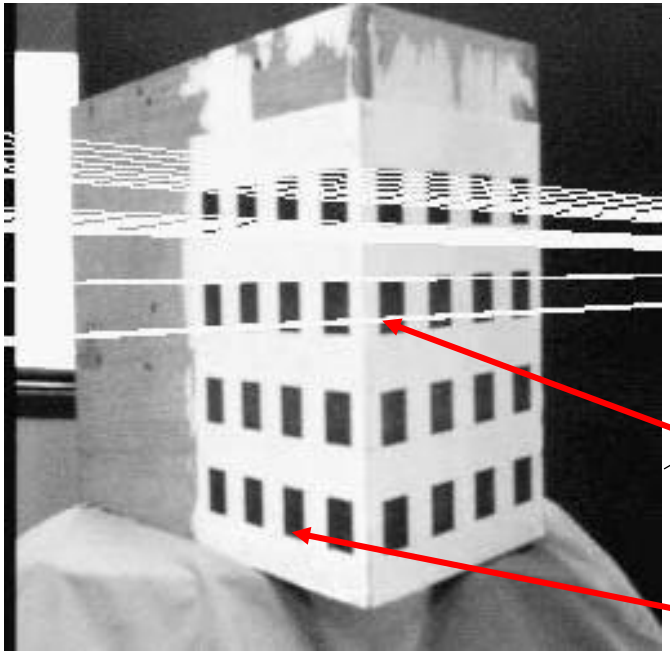


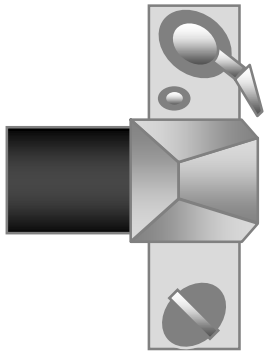
Non-linear regression

Given a function $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$ find $\mathbf{x}$ that minizes the function.

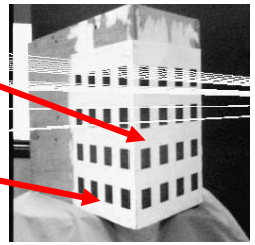
$$\mathbf{x}^* = \text{argmin}_{\mathbf{x} \in \mathbb{R}^m} \|f(\mathbf{x})\|^2$$

$$= \sum_{i=1}^{n} f_i(\mathbf{x})^2$$





Camera



Calibration grid

Image

Compute the pose and calibration (focal length, etc) of the camera.

M-estimators

m-estimators

Least-squares:

$$\text{minimize} \quad \|\mathbf{Ax} - \mathbf{b}\|^2$$

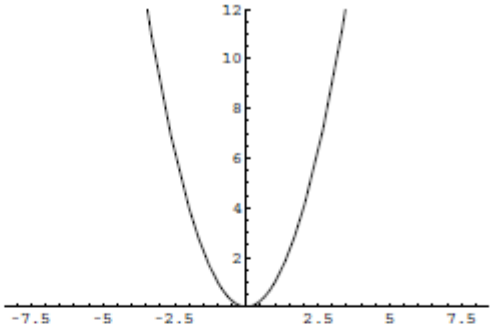
M-estimator

$$\text{minimize} \quad h(\|\mathbf{Ax} - \mathbf{b}\|)$$

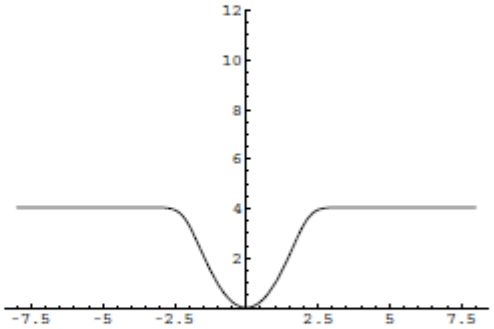
or given $f_i : \mathbb{R}^m \to \mathbb{R}$ for $i = 1, \ldots n$:

$$\text{minimize} \quad \sum_{i=1}^{n} h \circ f_i(\mathbf{x})$$

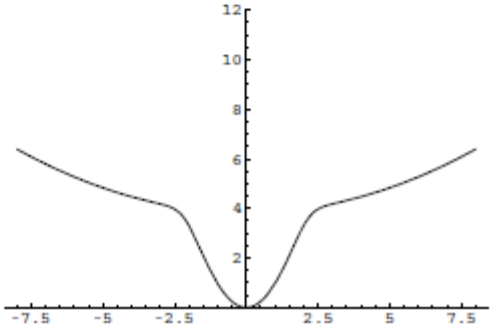
Squared-error



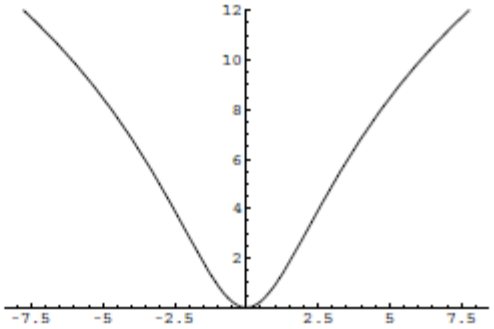
Blake-Zisserman



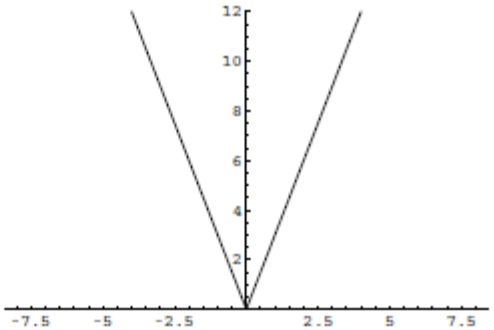
corrupted Gaussian



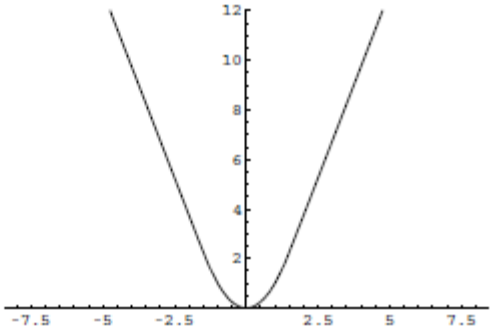
Cauchy



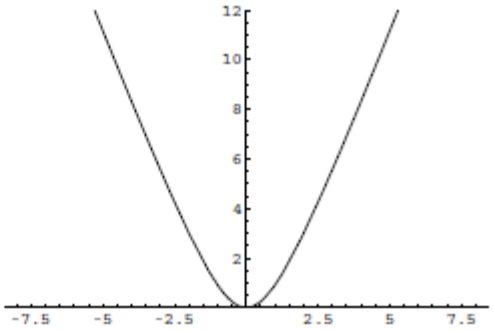
$L1$



Huber

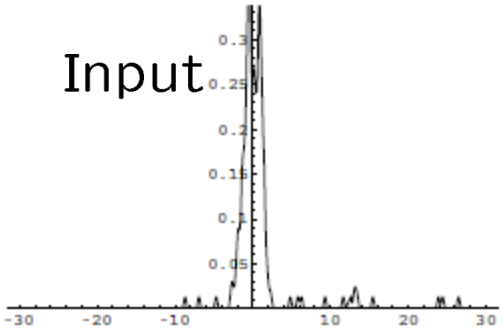


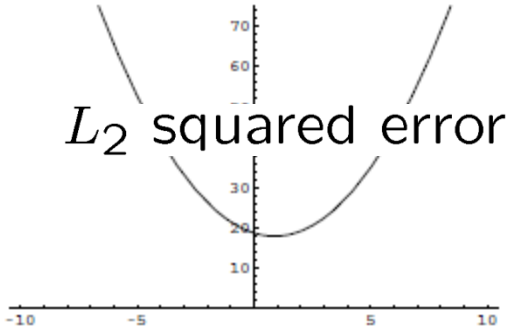
pseudo-Huber

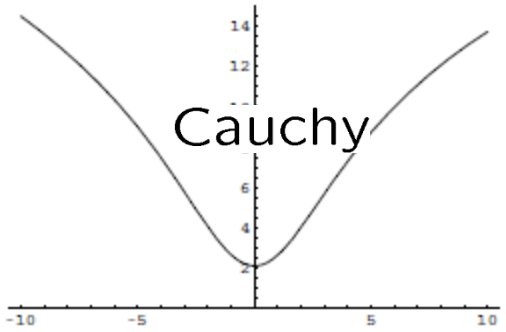


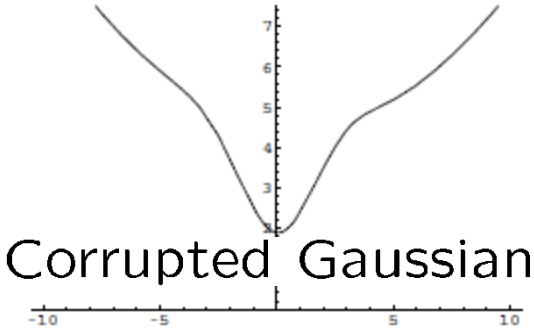
Various robust measures

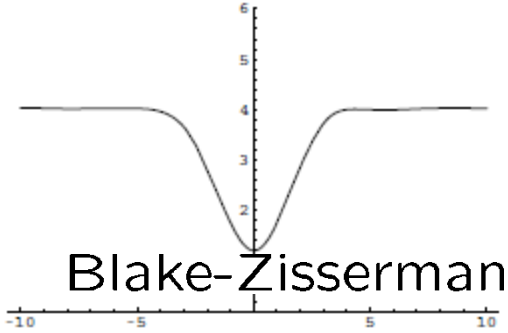
Resistance to Outliers

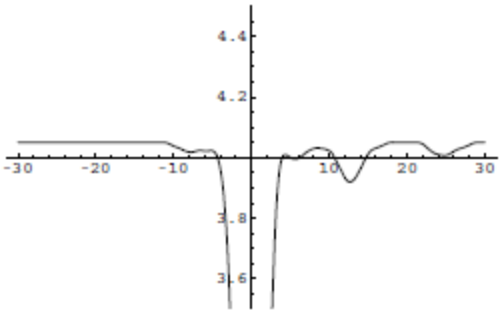


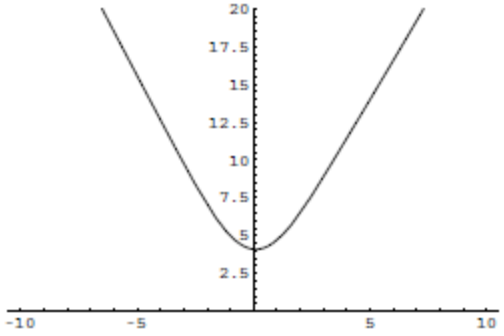


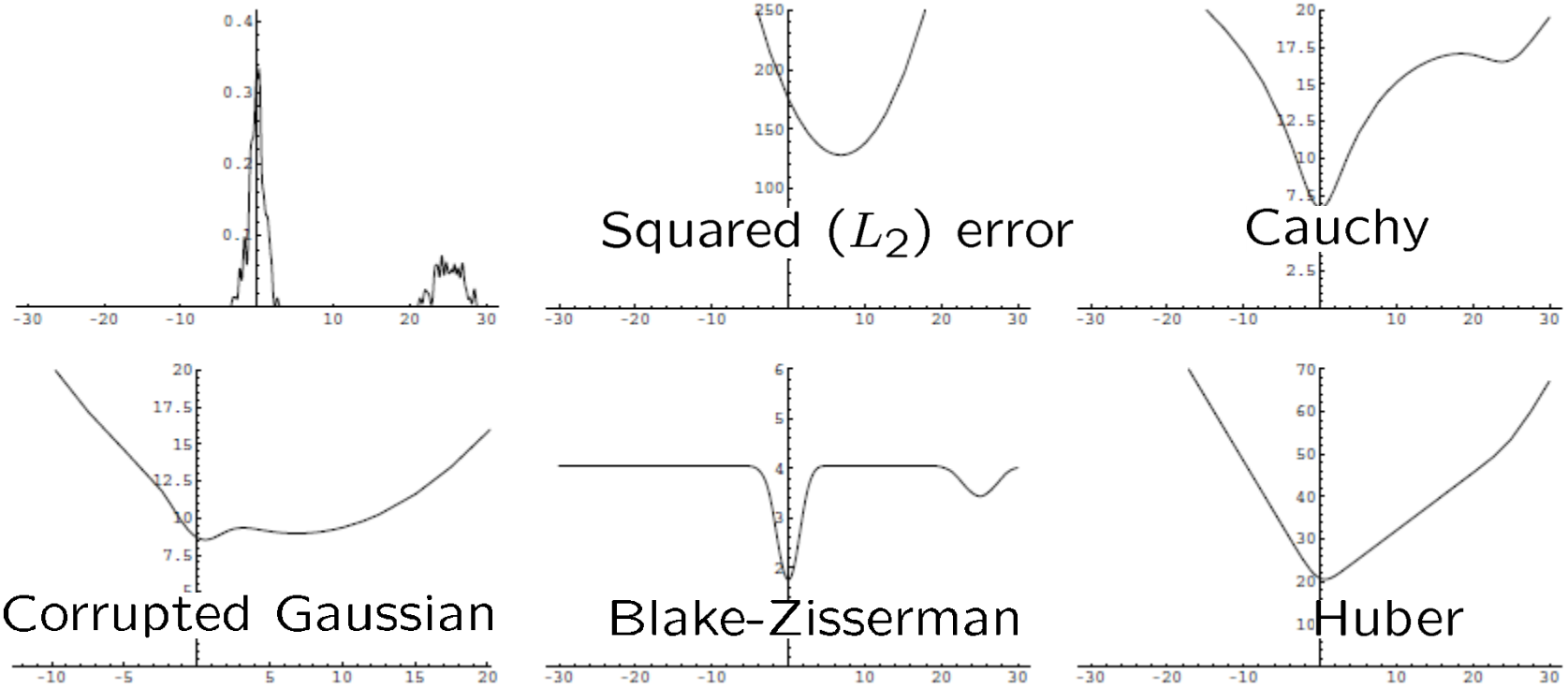








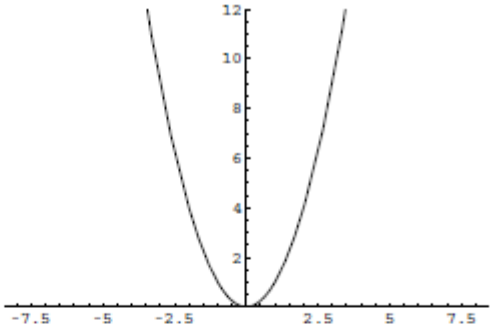




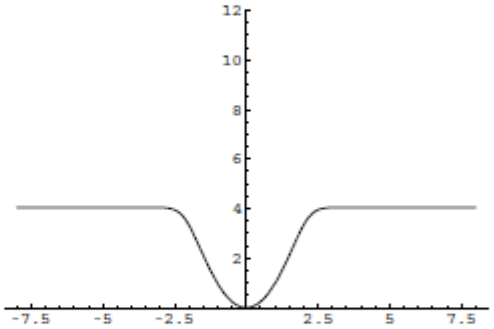
Robustness to systematic outliers
(e.g. ghost edges)

Generalizing IRLS

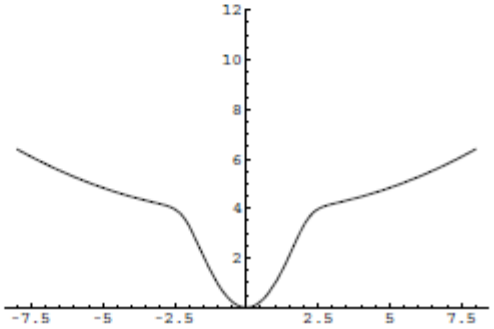
Squared-error



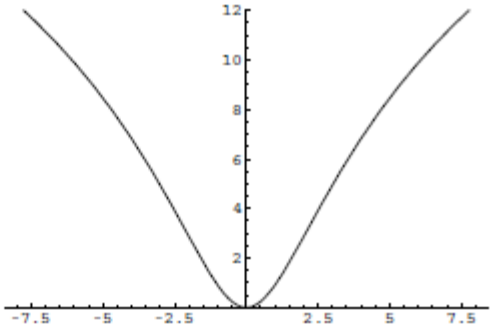
Blake-Zisserman



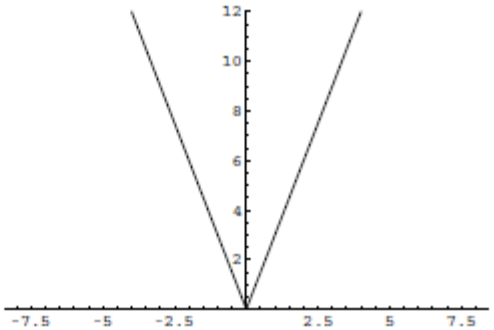
corrupted Gaussian



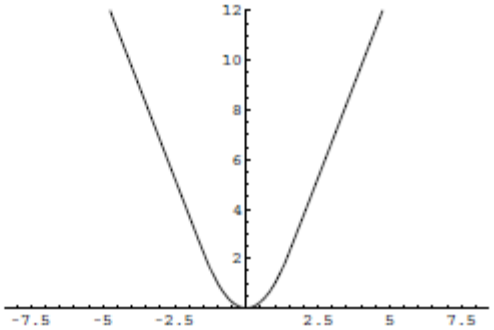
Cauchy



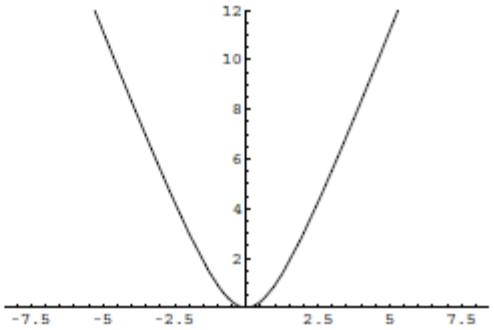
$L1$



Huber

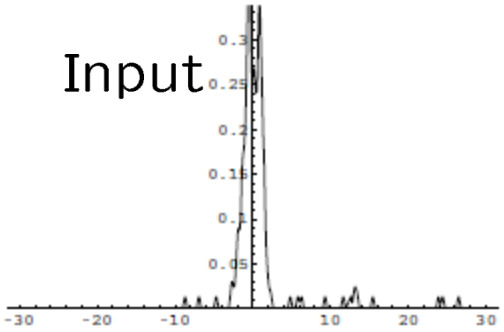


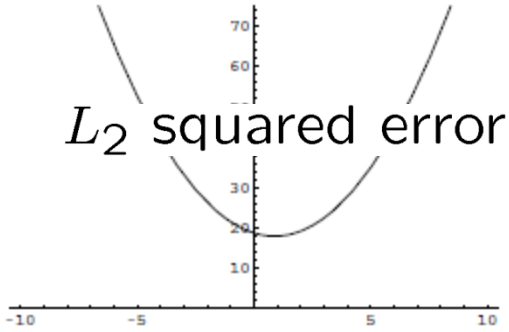
pseudo-Huber

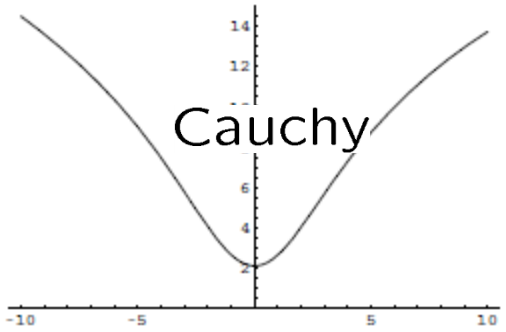


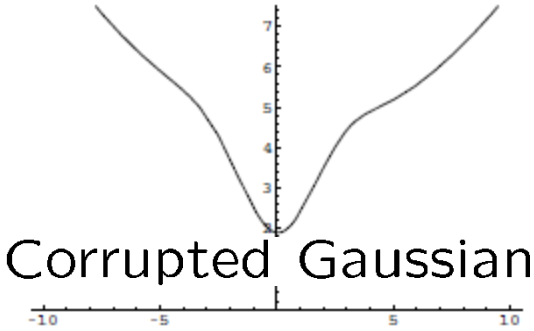
Various robust measures

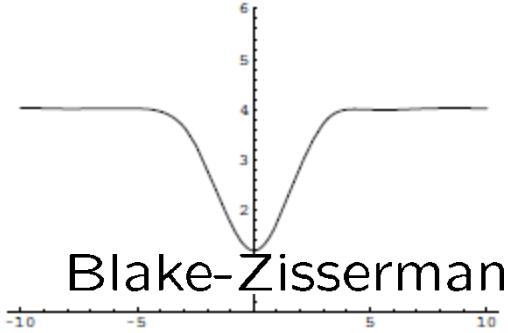
Resistance to Outliers

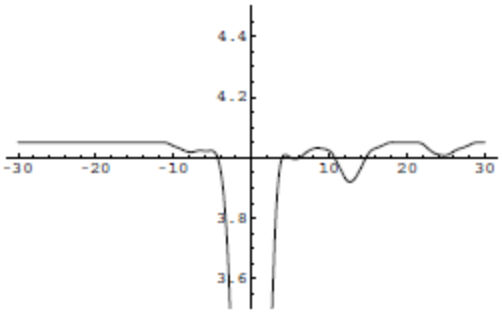


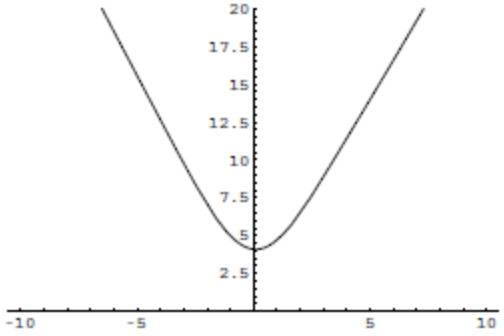


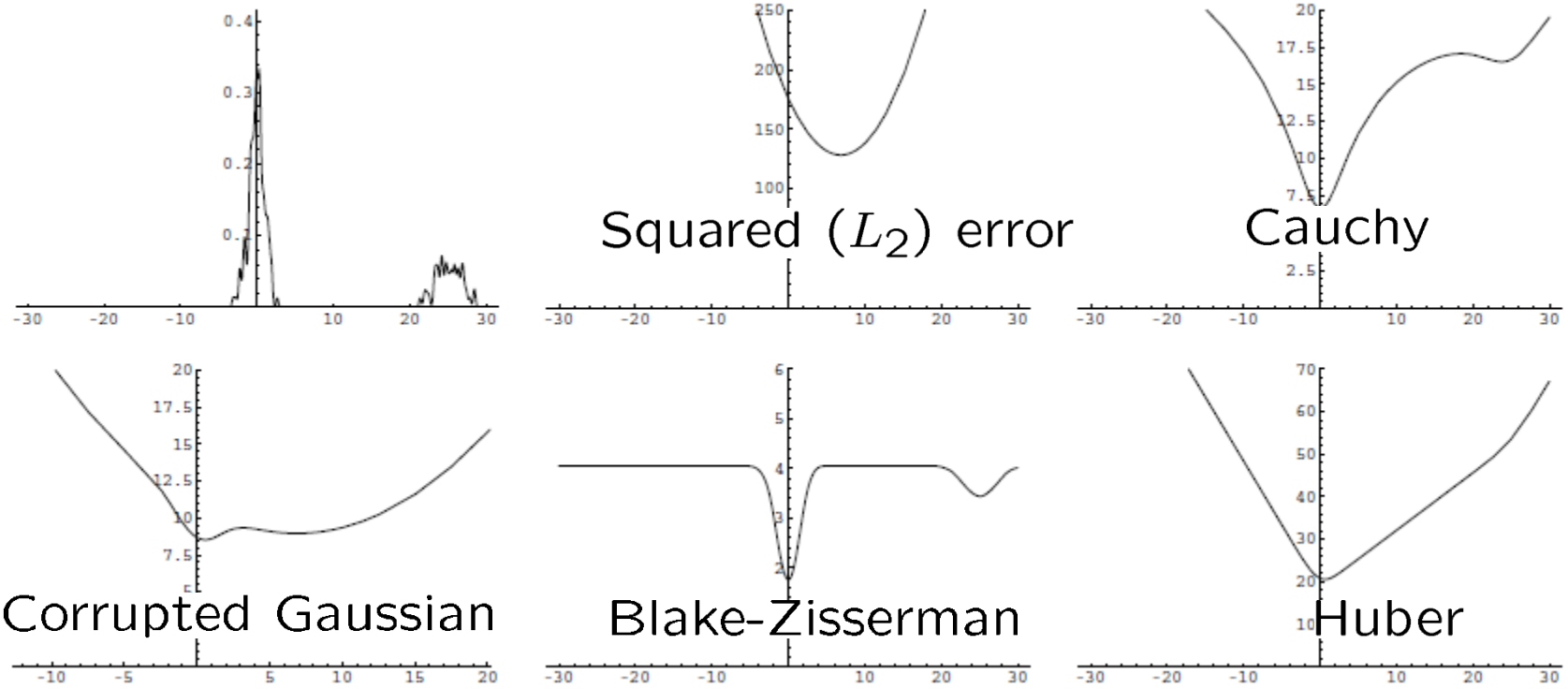




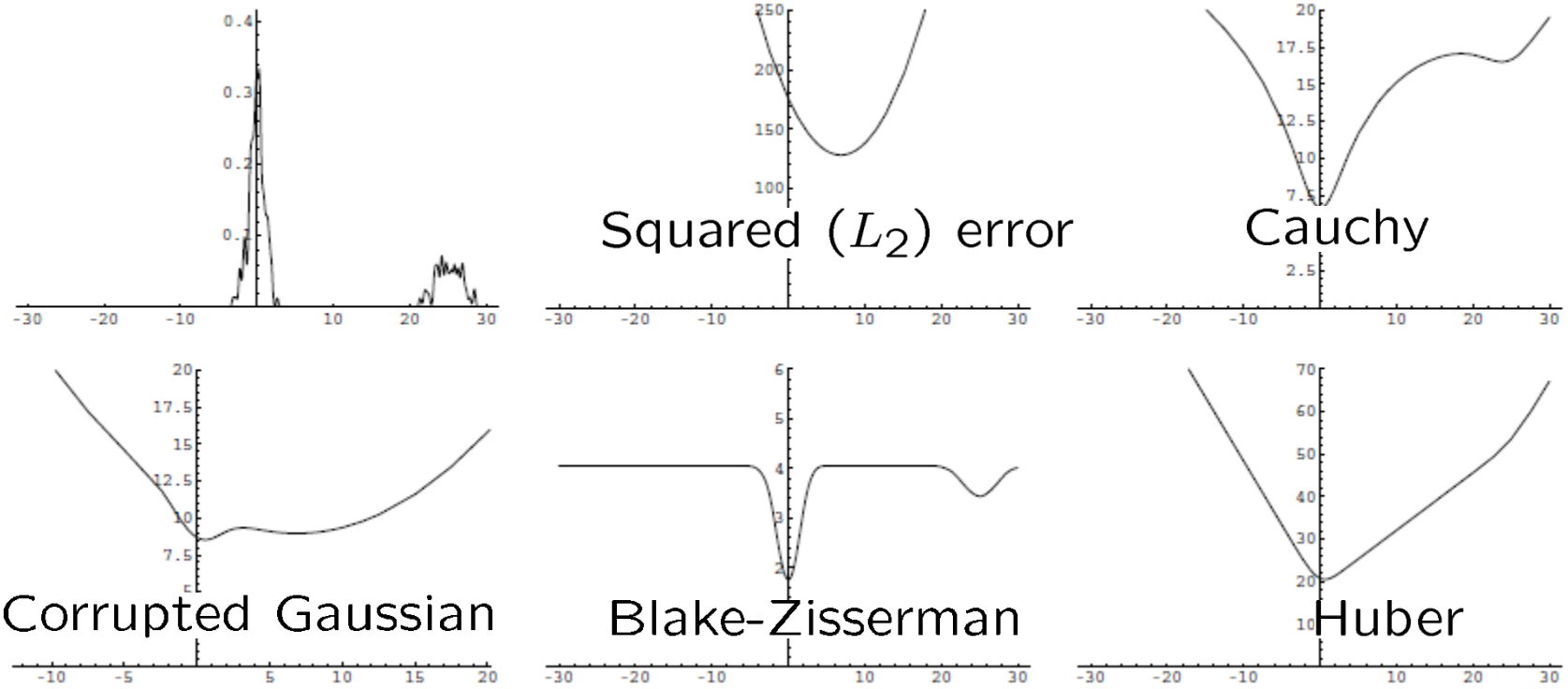








Robustness to systematic outliers
(e.g. ghost edges)



Robustness to systematic outliers
(e.g. ghost edges)

A general IRLS algorithm

1. Identify a weighted optimization problem that can be solved optimally (e.g. in closed form)

$$C(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{n} w_i \, f_i(\mathbf{x})$$

Written without the squares

2. Solve iteratively: At each step, define weights (how)

$$w_i^t = w_i(\mathbf{x}^t)$$

Define weights

and define

$$\begin{aligned} \mathbf{x}^{t+1} &= \text{argmin}_{\mathbf{x}} \, C(\mathbf{x}, \mathbf{w}^t) \\ &= \text{argmin}_{\mathbf{x}} \sum_{i=1}^{n} w_i^t \, f_i(\mathbf{x}) \end{aligned}$$

Minimize weighted cost

3. Hope that it converges to what you want.

Image Feature Extraction

Feature Extraction

- Edge detection/ Line Fitting
- Interest Point Detection (corner, SIFT)
- Image Region Segmentation (next week)

Three levels of Vision Processing

- **Low-level vision:**

  – image processing, denoise, filtering, image restoration.

- **Mid-level vision:**

  – image feature detection, image segmentation, edge, contour extraction, perceptual organization,

  – 3D vision reconstruction: Multiview Geometry: 2-1/2D representation, 3D information recovery.

- **High level vision:**

  – visual recognition, classification, object localisation, semantic understanding and labelling, action, activity, event detection.

Feature Extraction

- Edge detection/ Line Fitting---- 1D
- Interest Point Detection (corner, SIFT)-0D
- Image Region Segmentation----2D

Content

- Harris Corner Detector
  - The most widely used corner point detector
- SIFT-Scale Invariants Feature Transform

Motivation

- Find “interesting” parts/pieces inside an image
  - e.g. corners, salient patches
  - Focus of attention, fixation.
  - Speed up computation.
  - Compress/extraction of information.
- Applications of interest points
  - Image Matching, Search
  - Object Detection, Object Recognition
  - Image Alignment & Stitching
  - Stereo
  - Tracking

Photo Tourism

Exploring photo collections in 3D

Noah Snavely Steven M. Seitz Richard Szeliski

University of Washington *Microsoft Research*

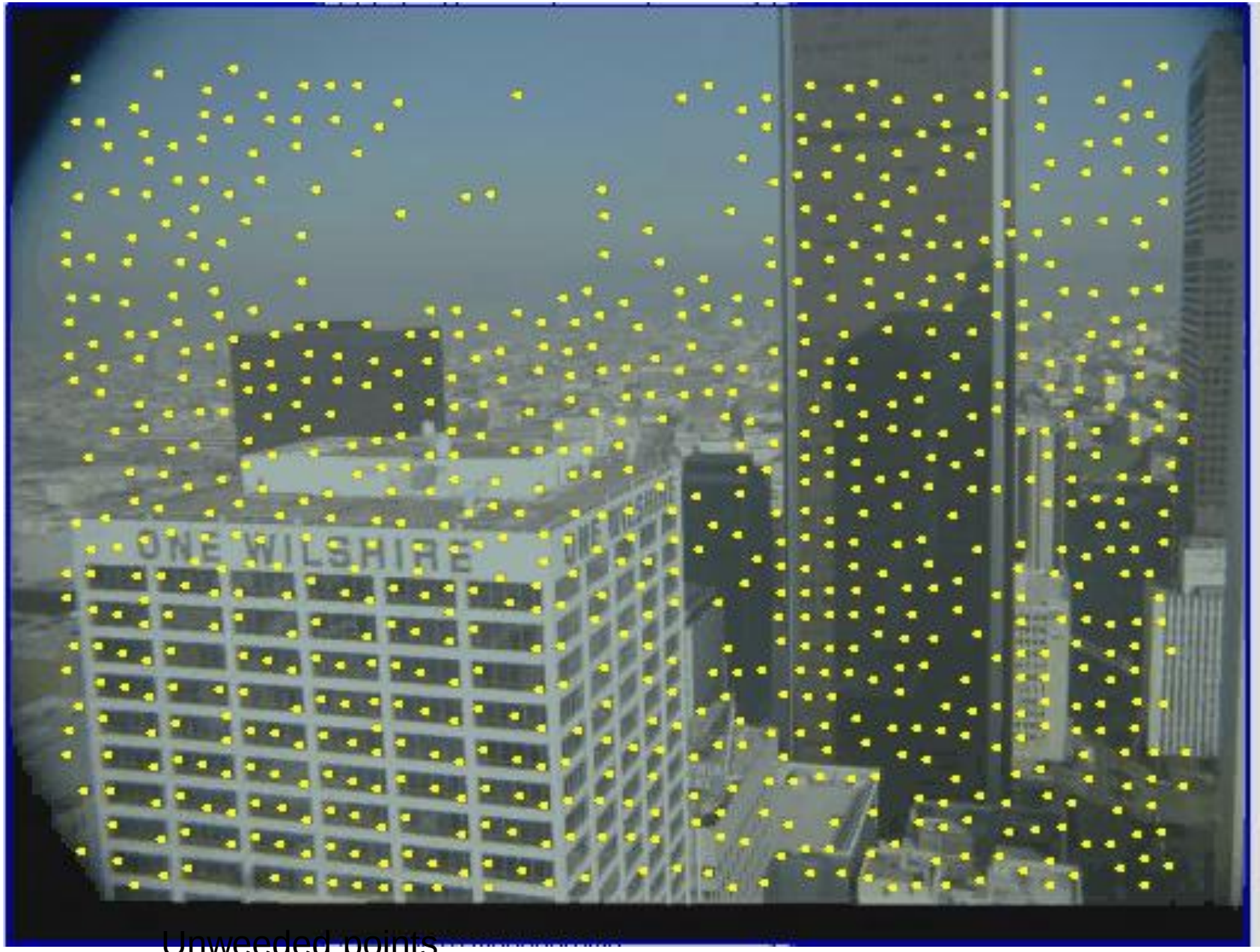
SIGGRAPH 2006

Image sequence of Wilshire Boulevard, LA

Courtesy Oxford Visual Geometry group

Steps of reconstruction:

1. Tracking of points in the video
2. Weeding out bad tracks
3. Projective reconstruction
4. Self calibration / Euclidean reconstruction
5. Model building

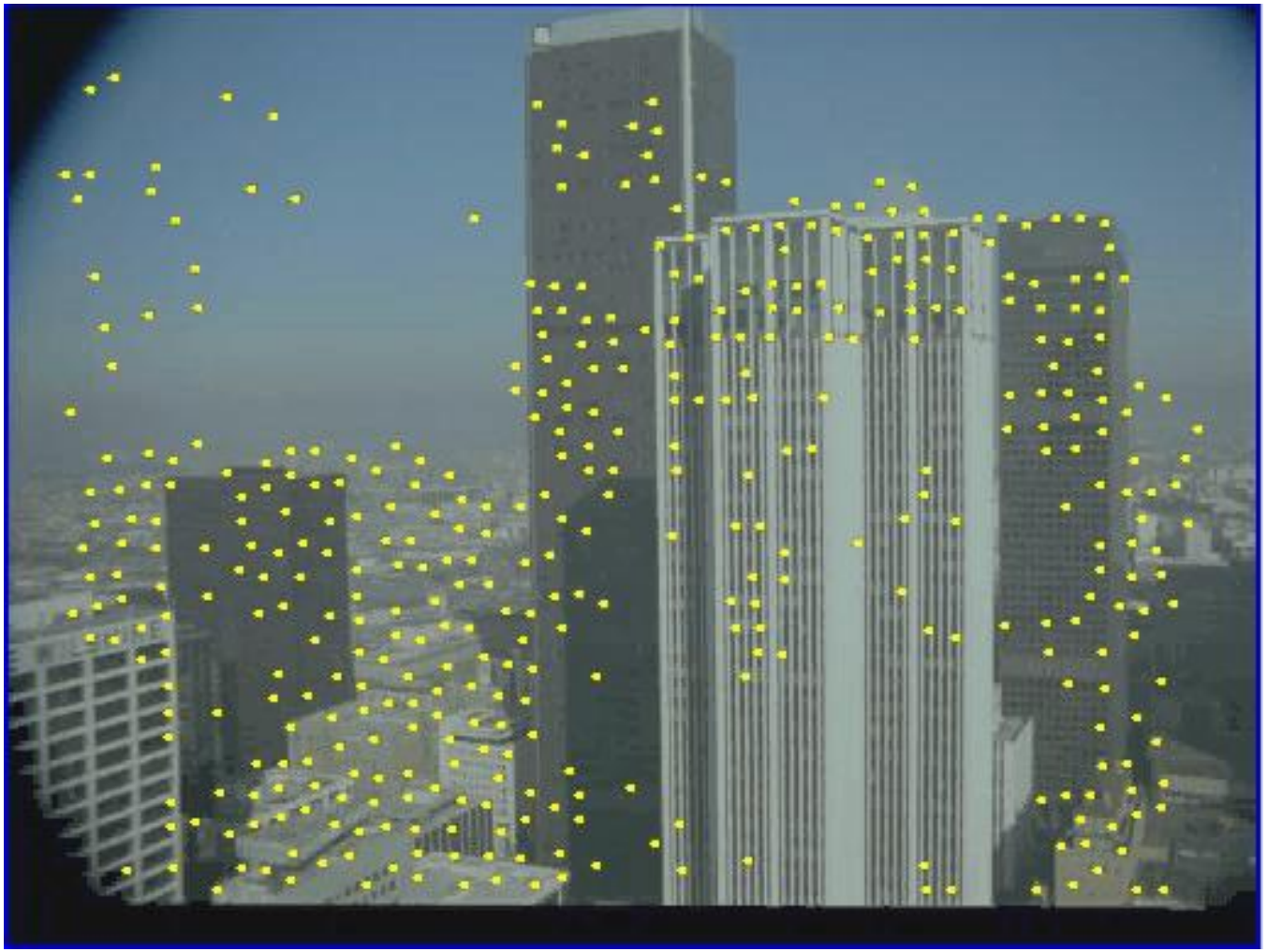


Unweeded points

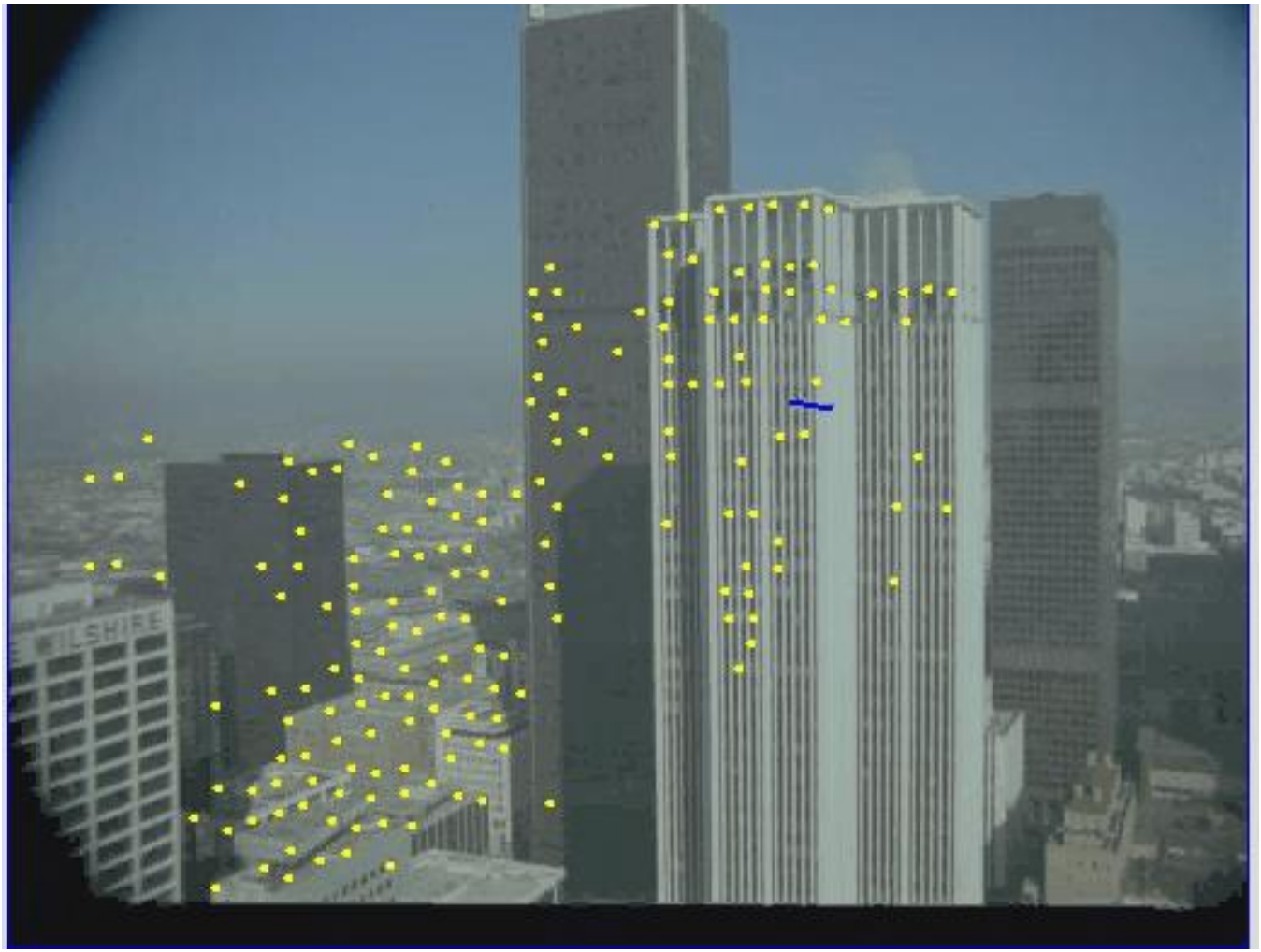
Tracking

First step is to weed the points

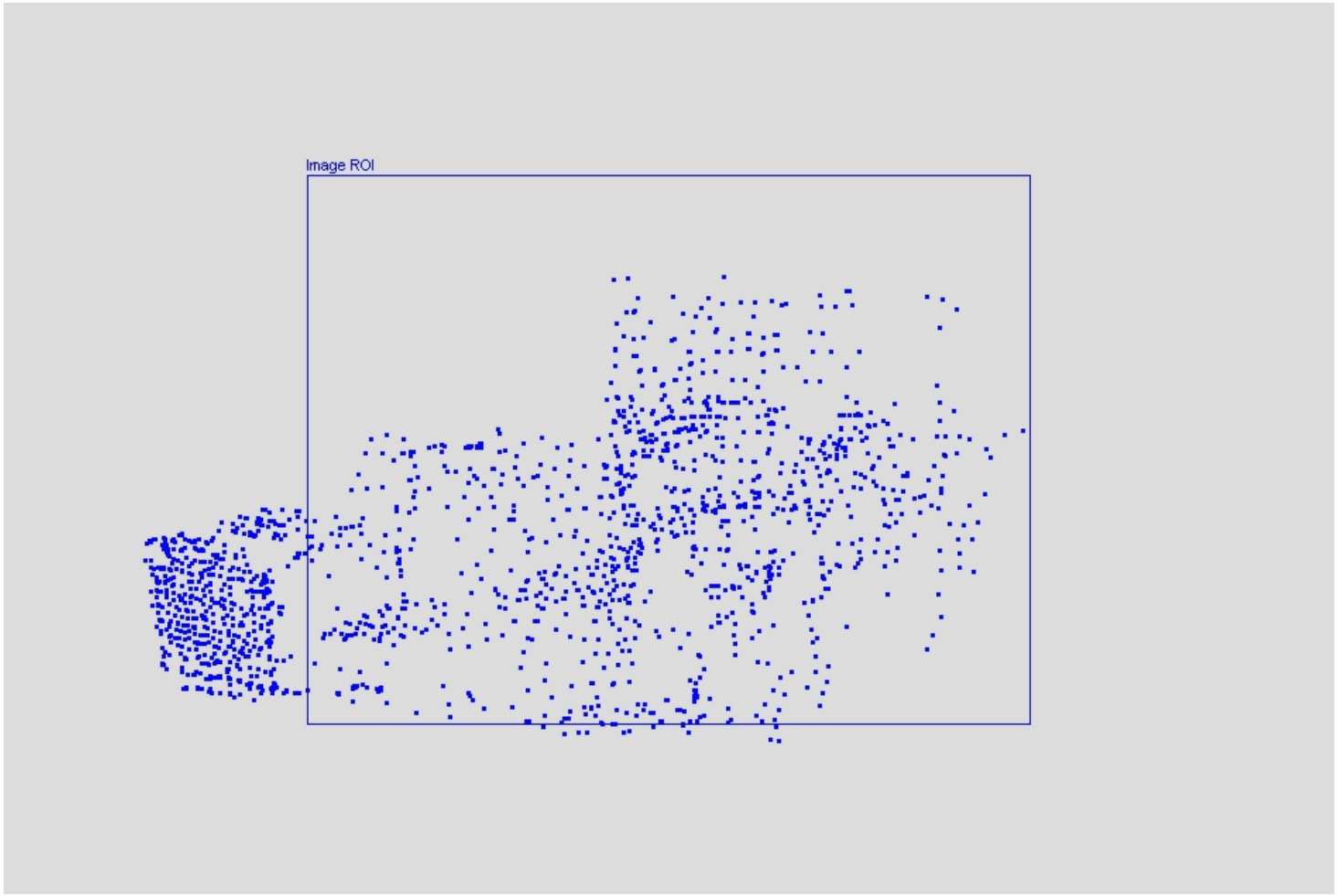
1. Produces lots of bad matches
2. Must be weeded out using a weeding program



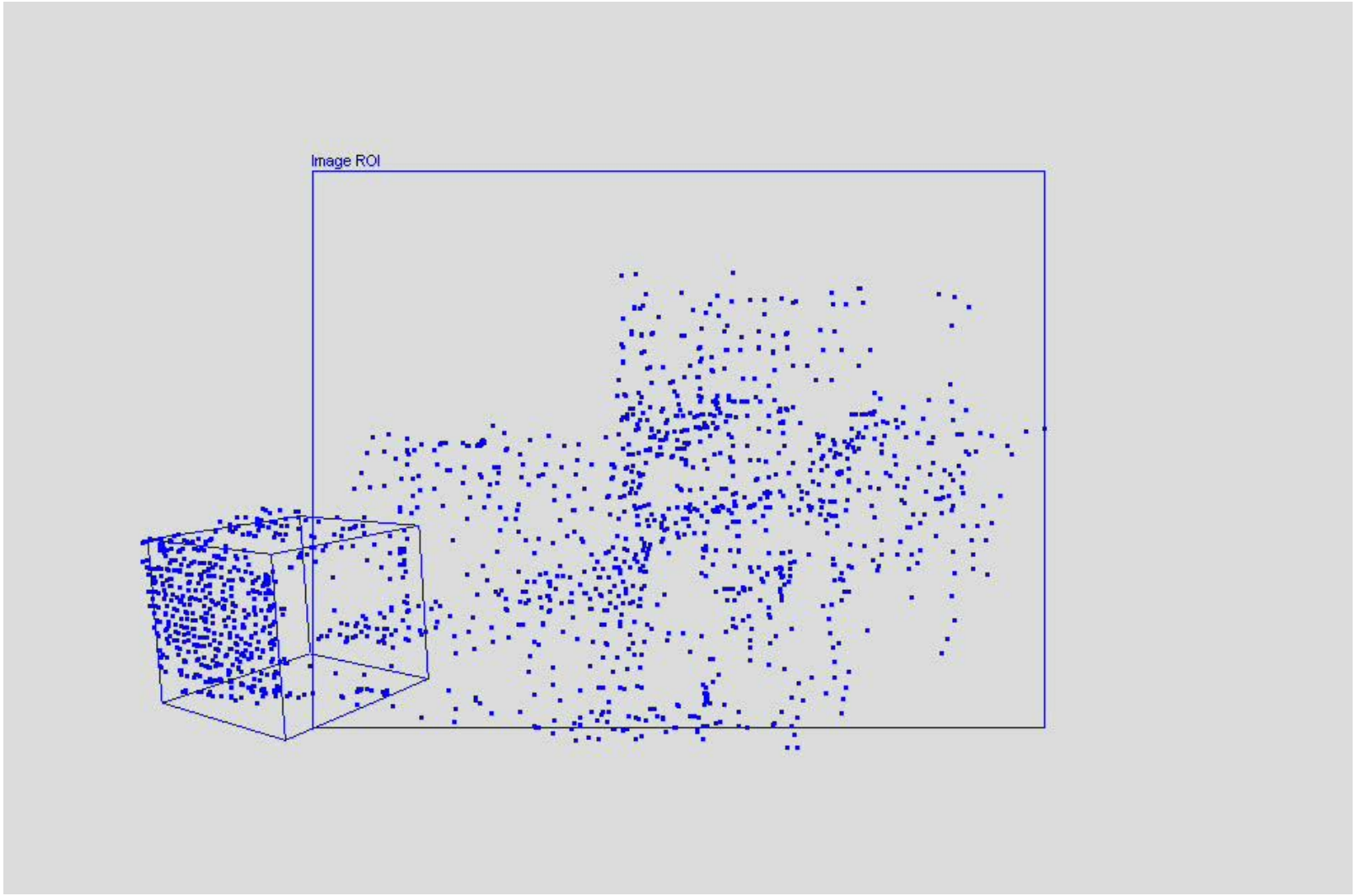
Outliers removed using the fundamental matrix



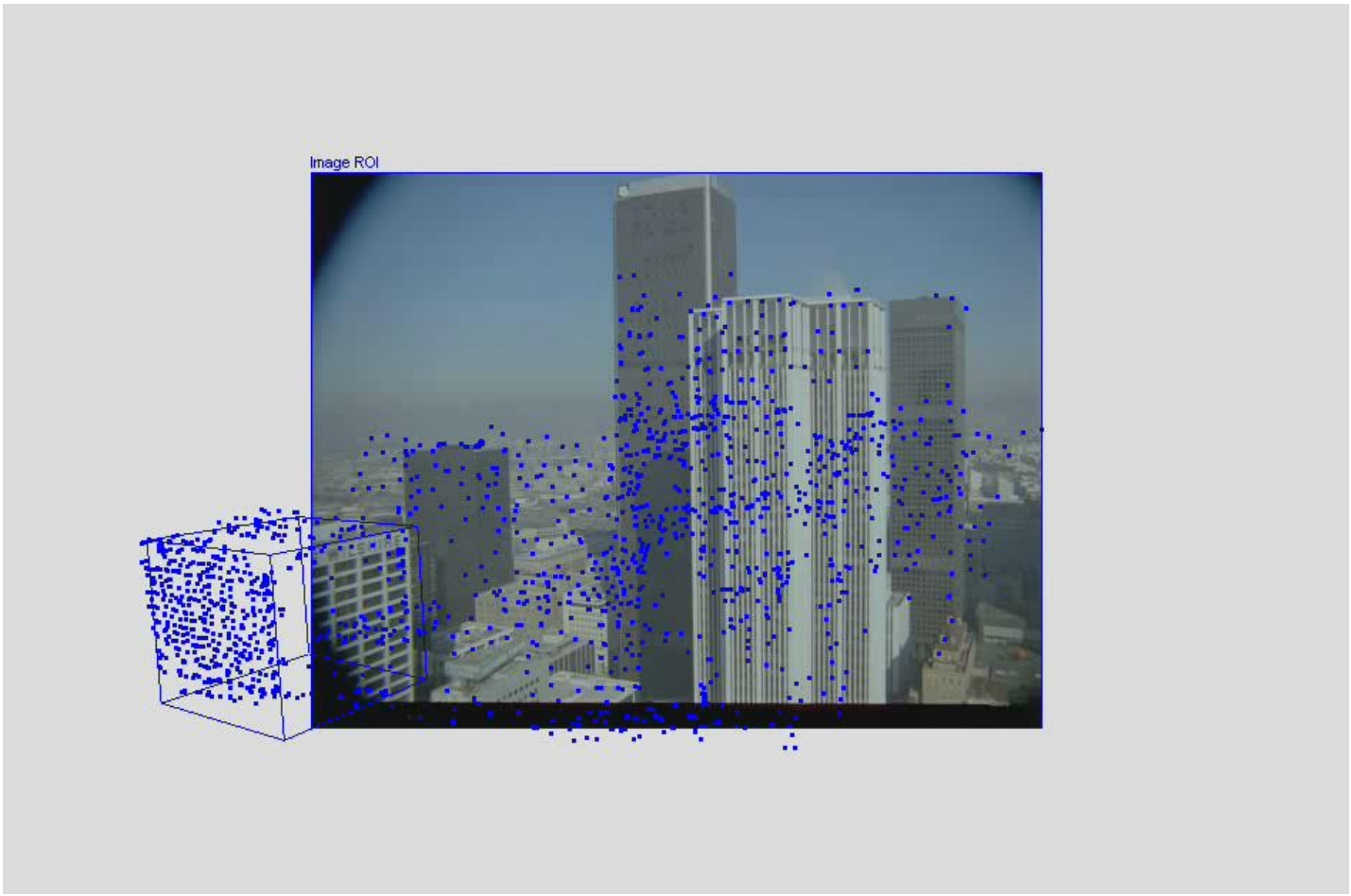
Outliers removed using the trifocal tensor



Results of reconstruction of points



Rectangular reference frame



Rectangular reference frame

Enhanced video

Courtesy Oxford Visual Geometry group

Tracking a plane section of the image

Stabilization on the plane

Image enhancement using homographies

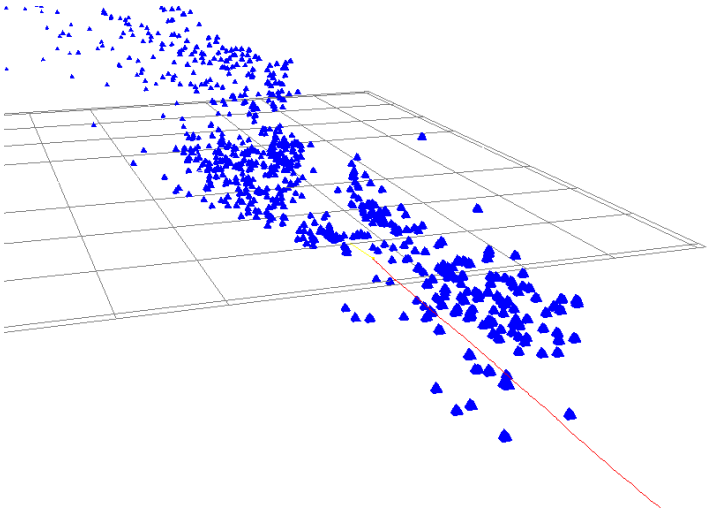
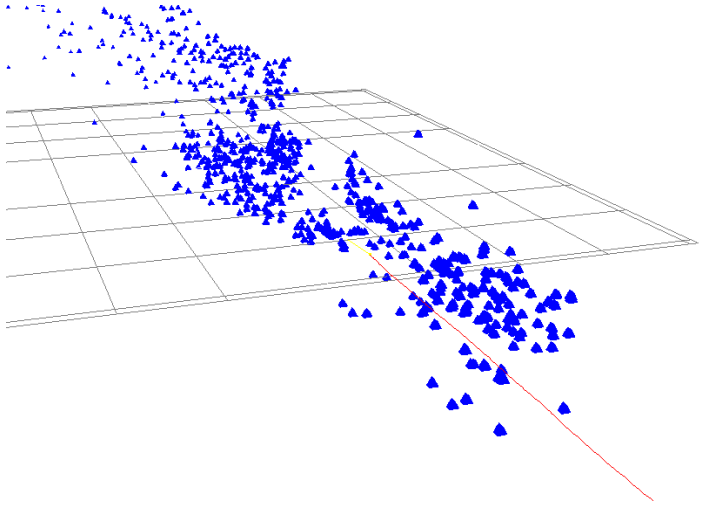
Courtesy Oxford Visual Geometry group


ONE WILSHIRE
ONE WILSHIRE



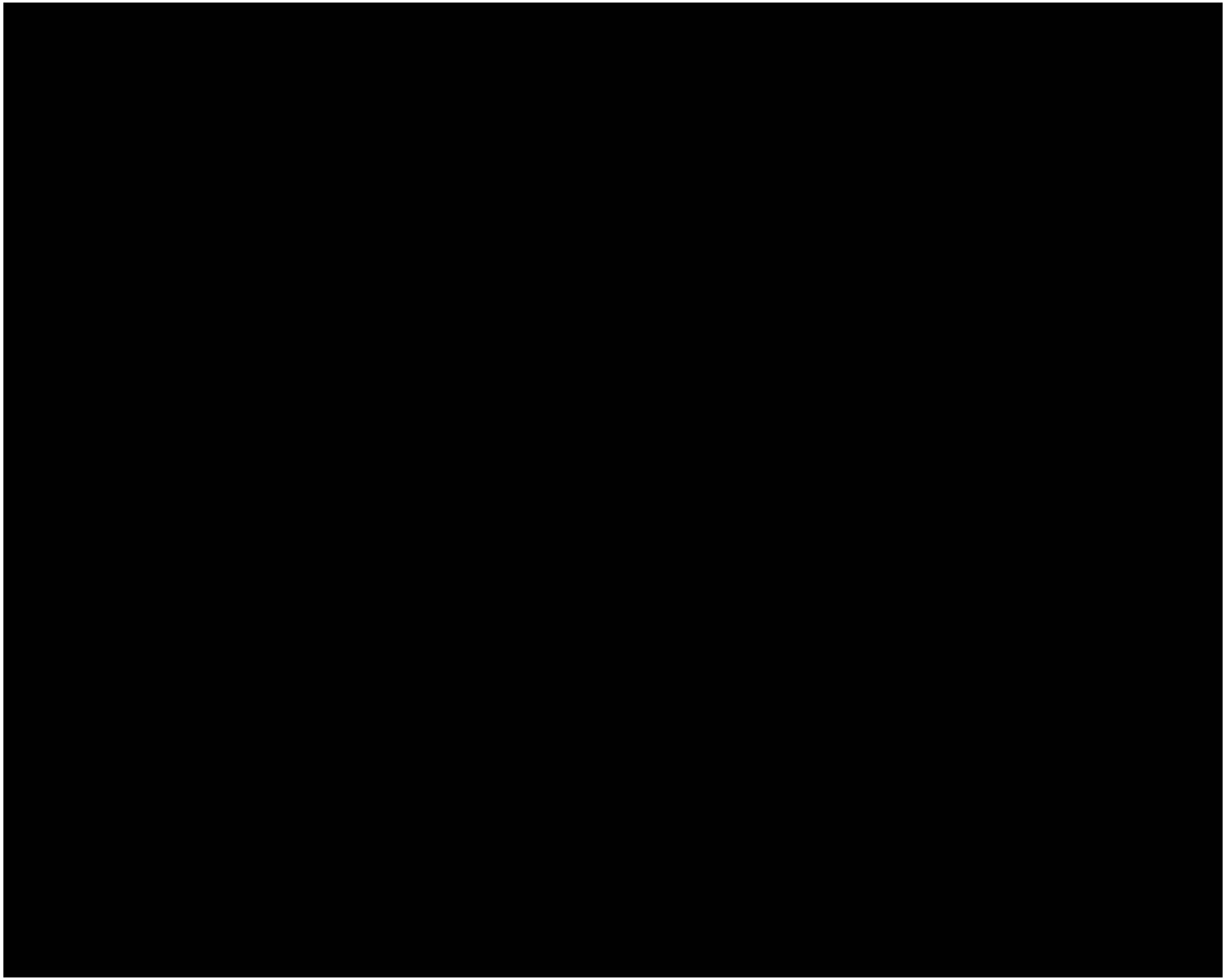




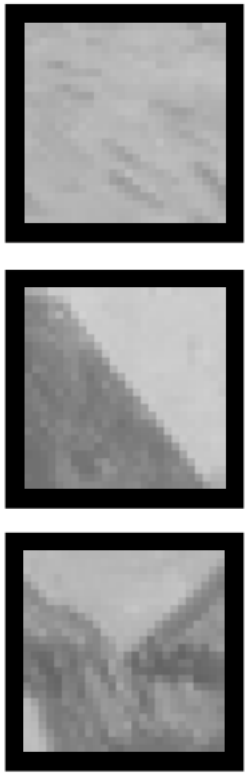




2D3 Movie

Interest Points



isotropic structure: flat region

➡ not interesting, 0D, not useful for matching

linear structure: edges, lines

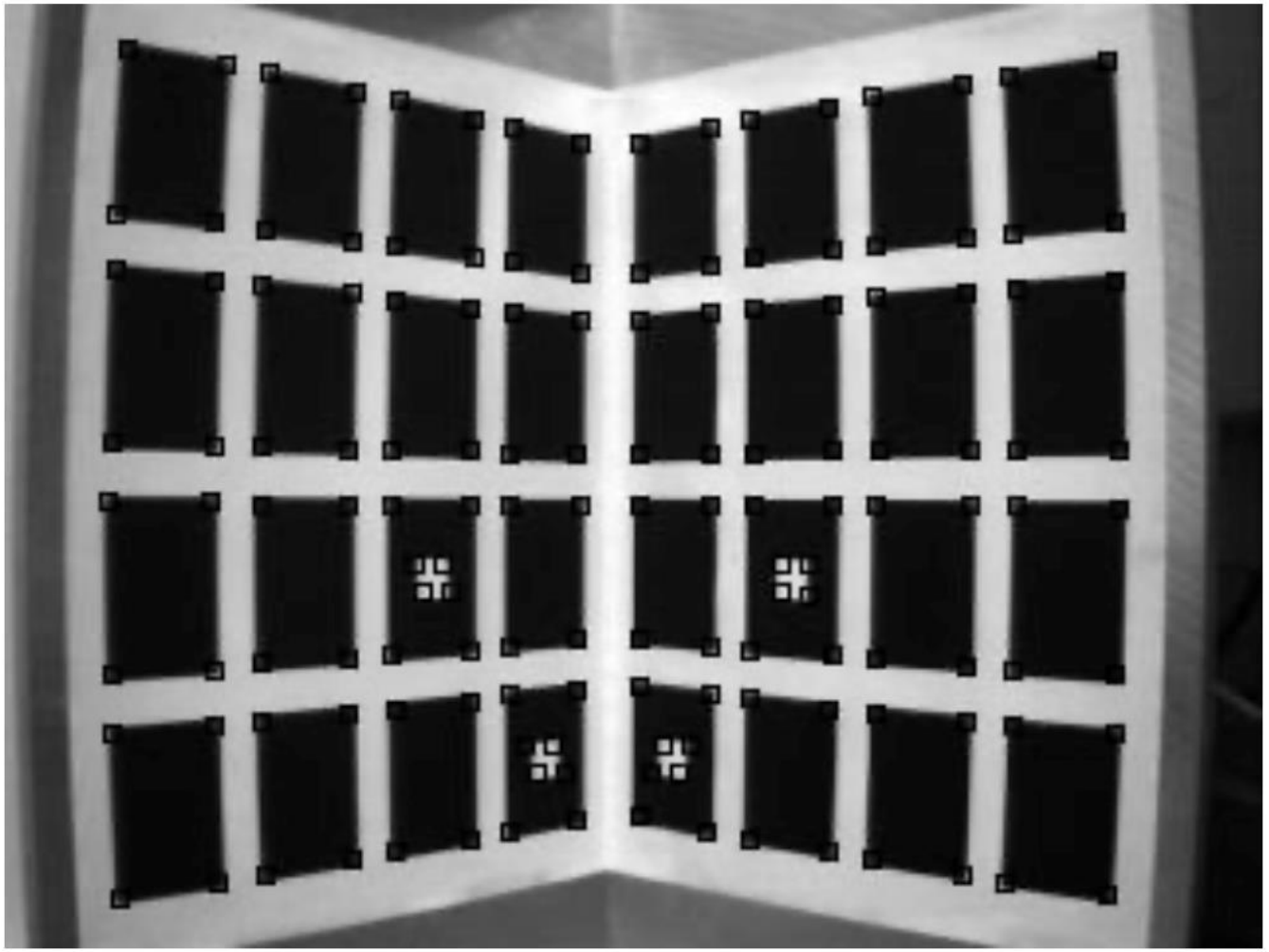
➡ edge, can be localized in 1D, subject to the aperture problem

bi-directional structure: corners

➡ corner, or **interest point**, can be localised in 2D, good for matching

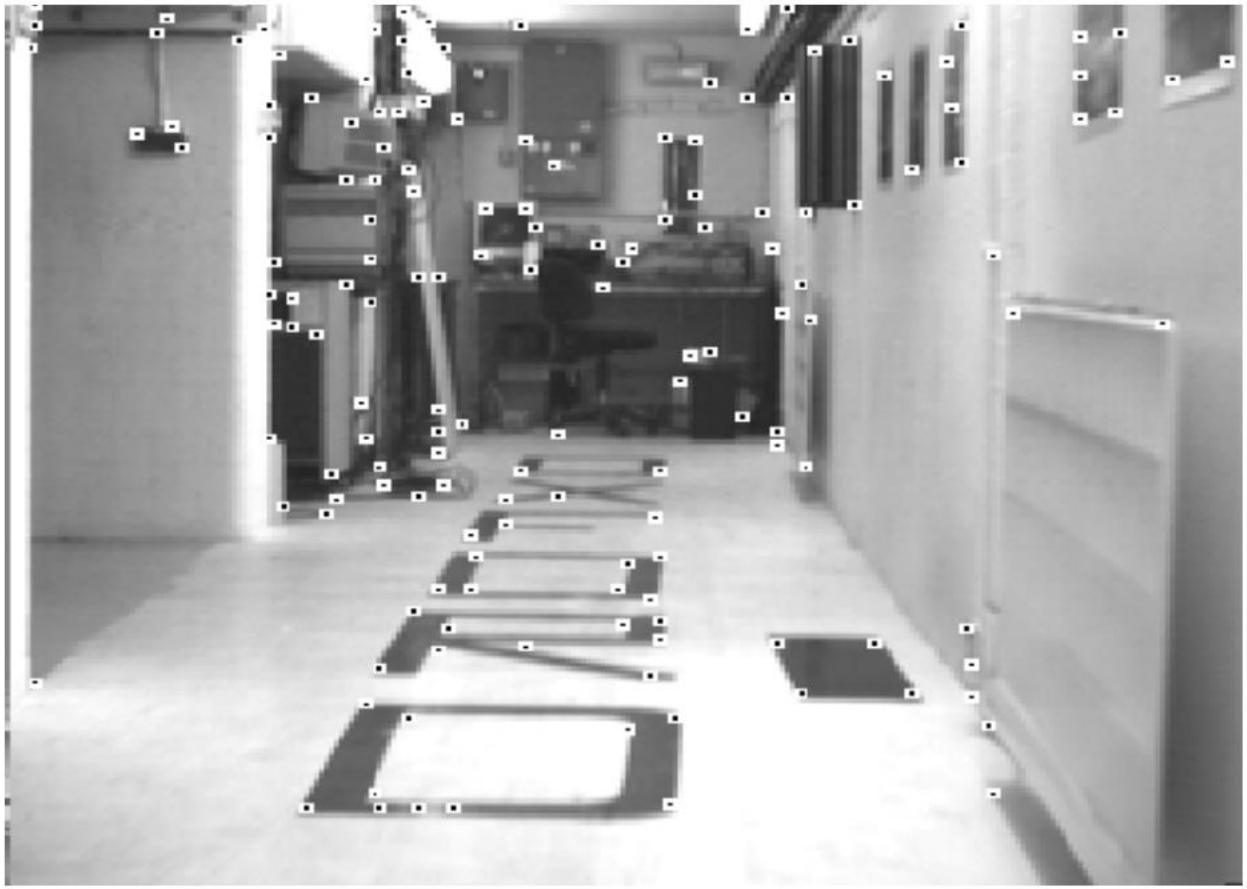
Interest Points have 2-directional structure.

Application: Corner Detection (for camera calibration)



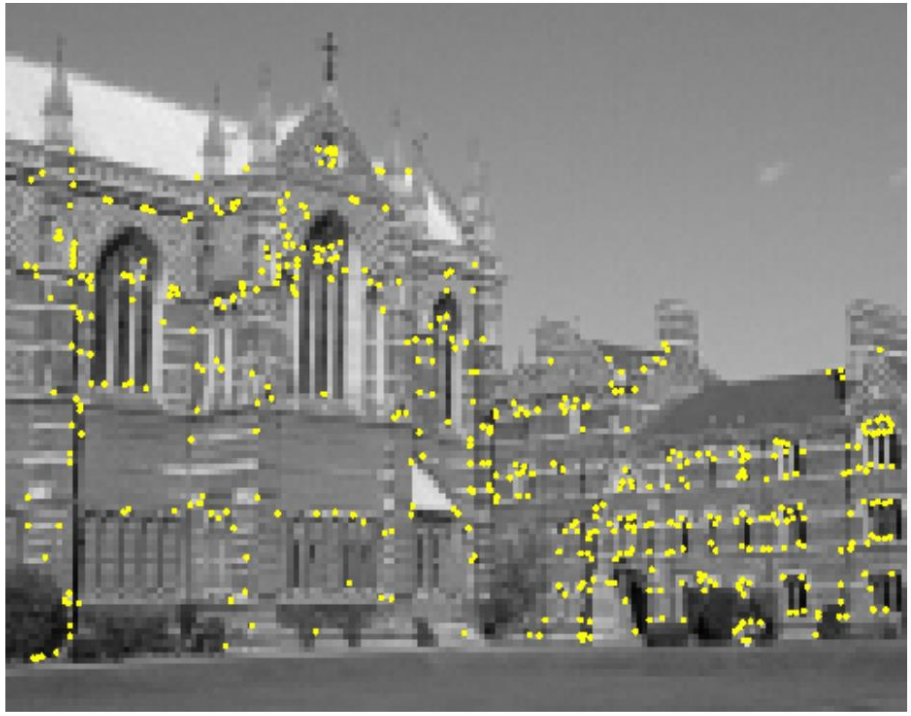
courtesy of B. Wilburn

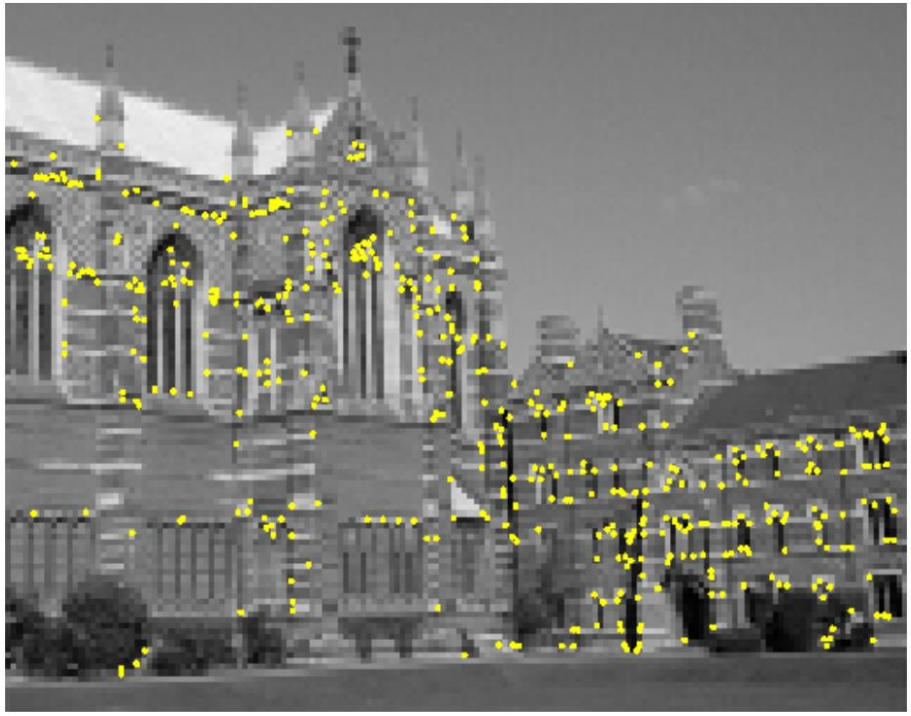
Application: Robot navigation



courtesy of S. Smith

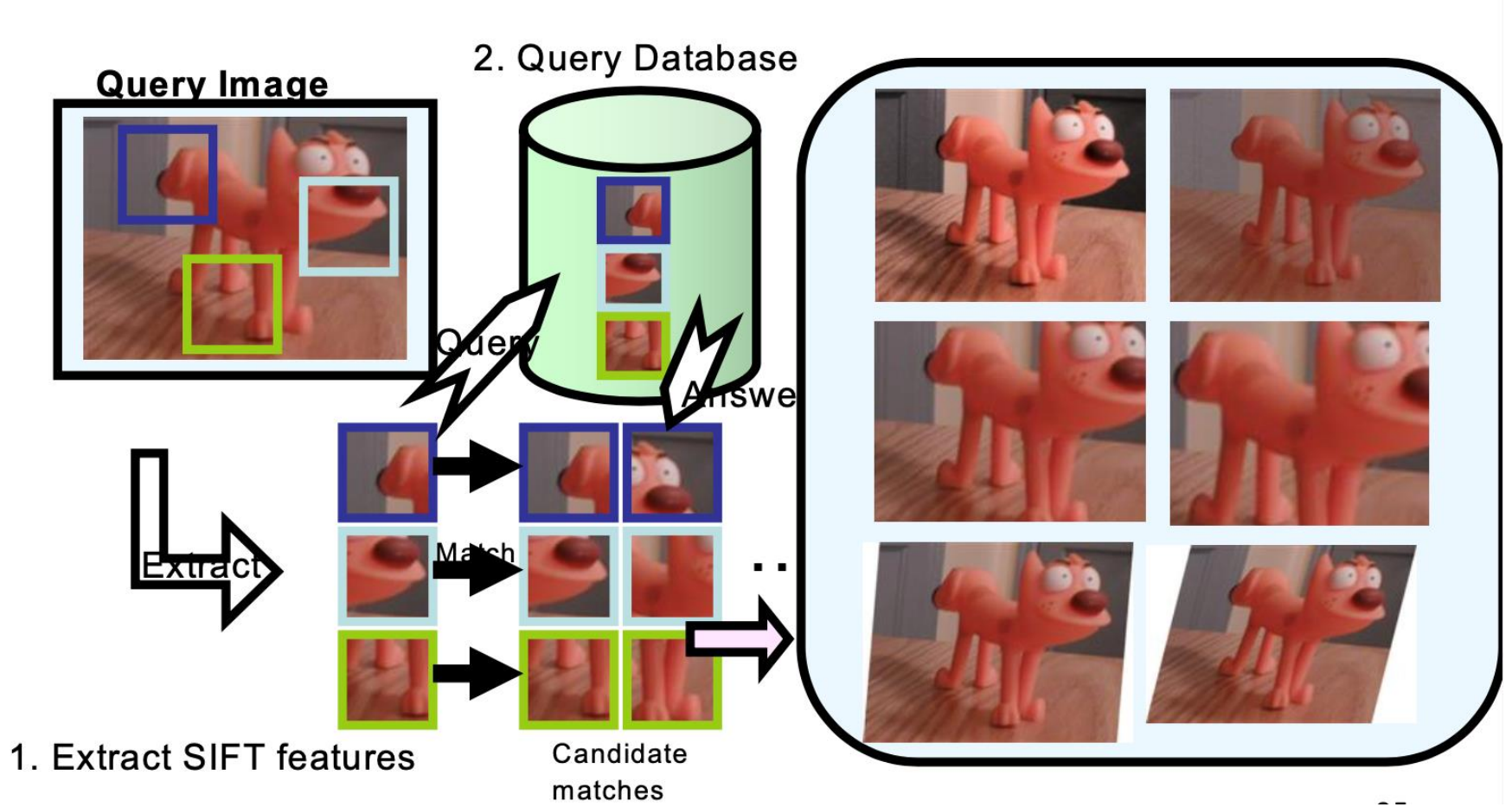
Application: Matching between two images



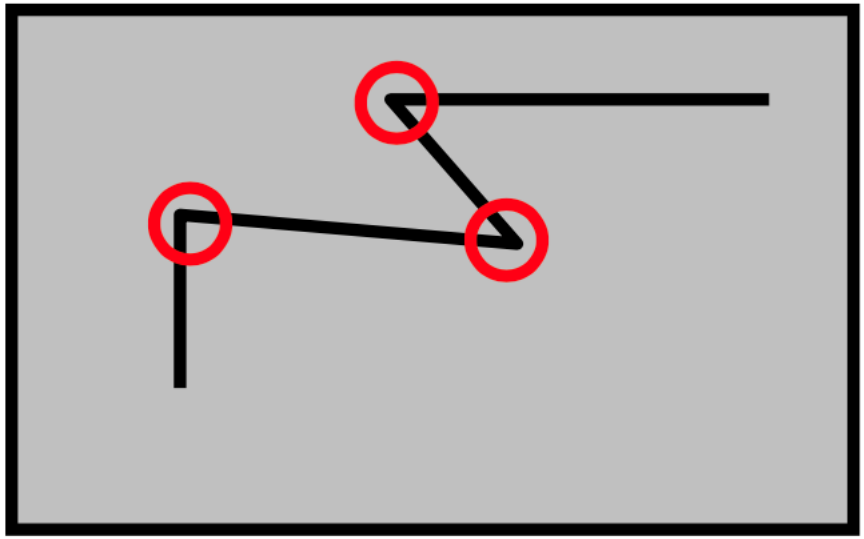


Interest points extracted with Harris (~ 500 points)

Content Based Image Retrieval (CBIR)



Harris Corner Detector



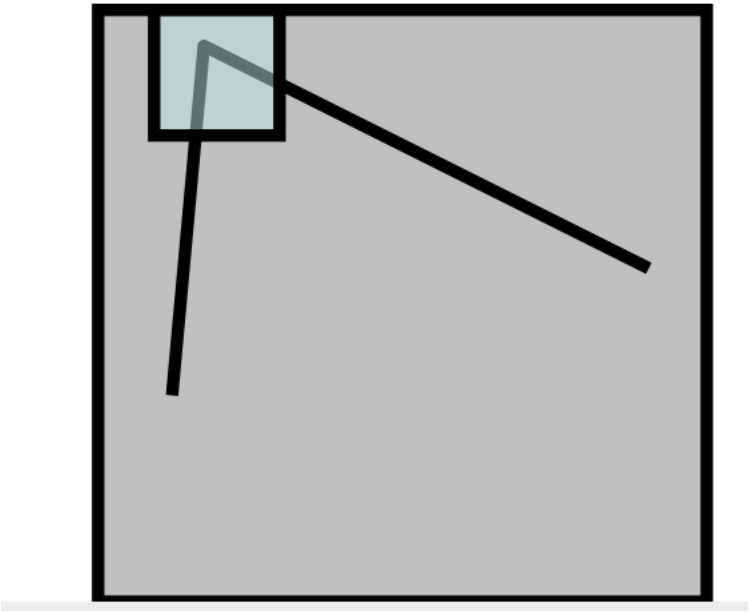
Reference

Chris Harris & Mike Stephens, CVIU, 1988

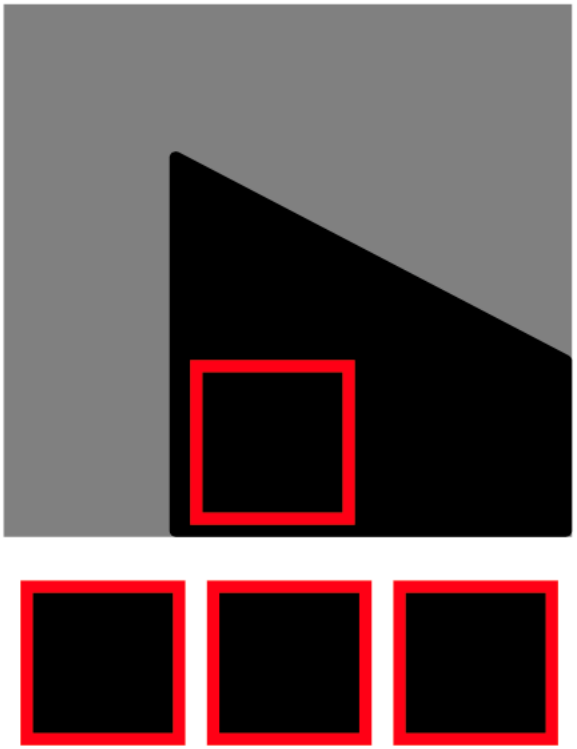
“A Combined Corner and Edge Detector”

Harris Corner: Intuition

- We should easily recognize a corner point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity

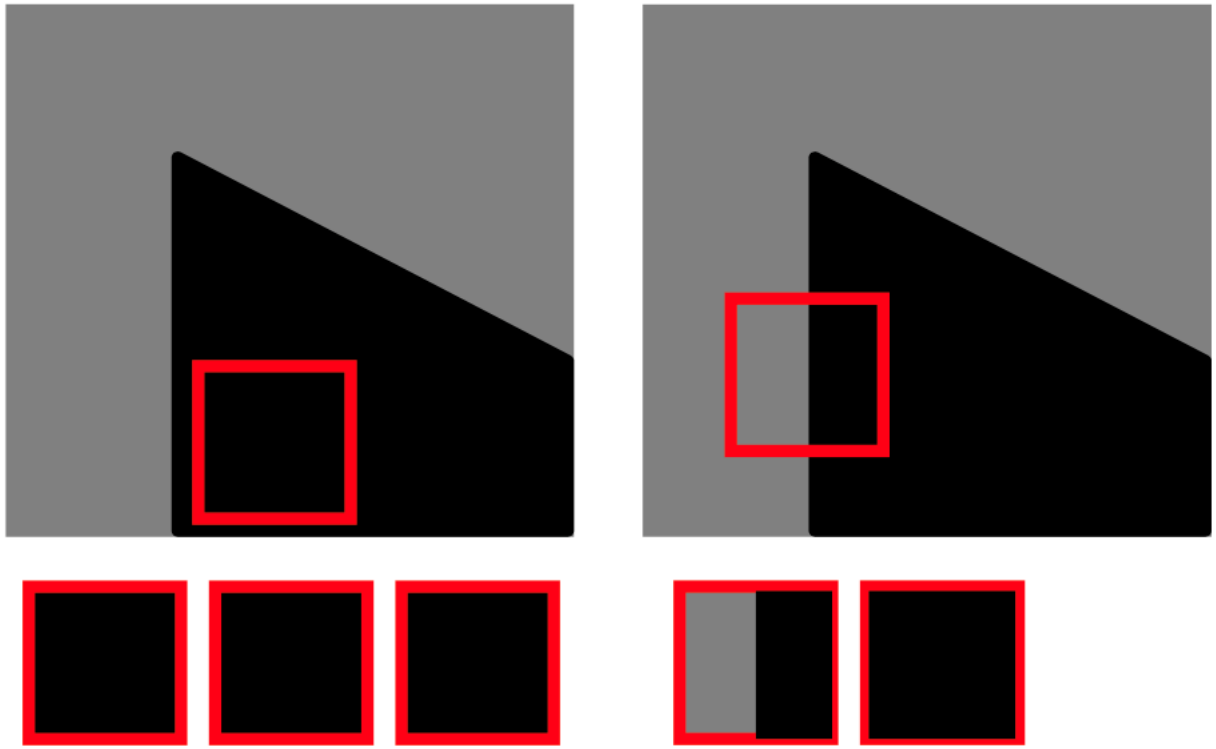


Corner detector



Flat

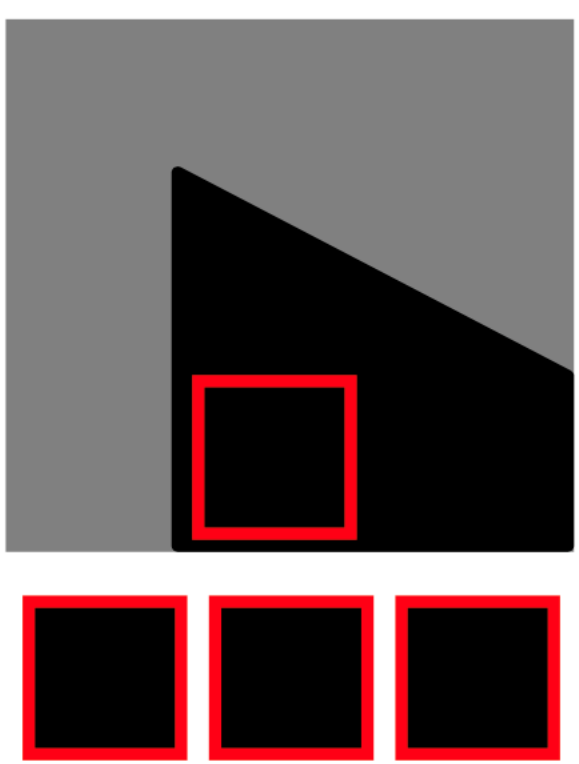
Corner detector



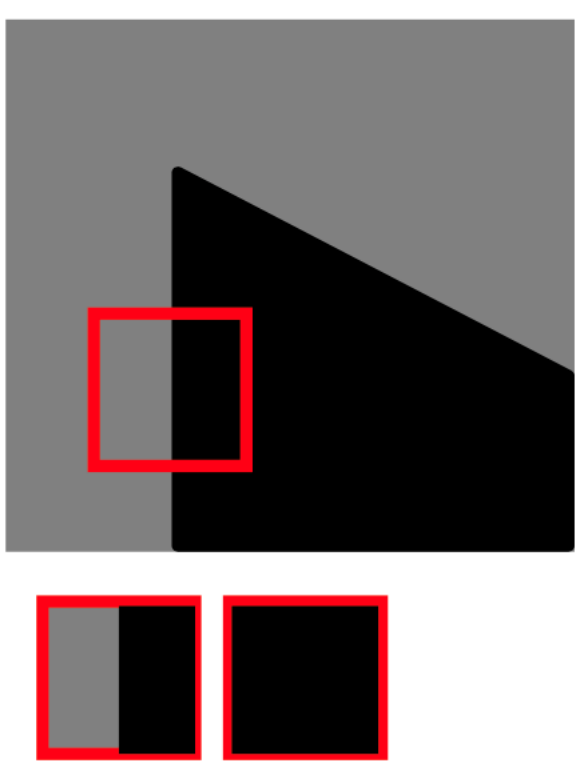
Flat

Edge

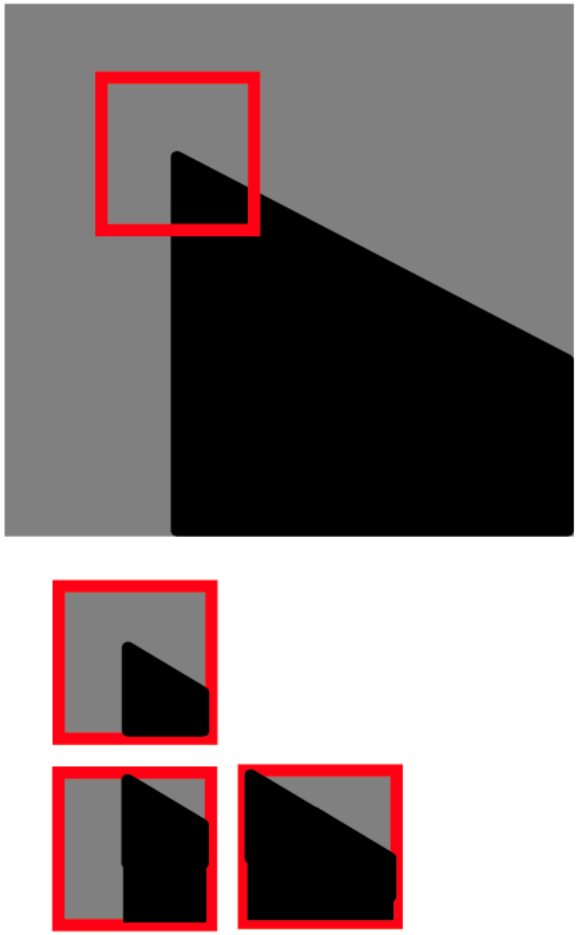
Corner detector



Flat



Edge



Corner

Harris Corner: Mathematics

Change of intensity for the shift $[u,v]$
(local auto-correlation analysis):

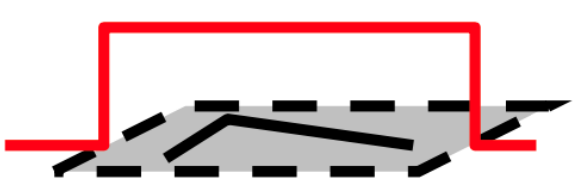
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

Intensity

Window function $w(x,y)$ =



1 in window, 0 outside

or

Gaussian

Harris Corner: Mathematics

- For small shifts [u,v], we have first-order Taylor approximation for I(x+u,y+v)

$$I(x+u, y+v) \approx I(x,y) + uI_x + vI_y$$

Harris Corner: Mathematics

$$
\begin{aligned}
E(u,v) &= \sum_{x,y} w(x,y)(I(x+u,y+v) - I(x,y))^2 \\
&= \sum_{x,y} w(x,y)(I(x,y) + uI_x + vI_y - I(x,y))^2 \\
&= \sum_{x,y} w(x,y)(uI_x + vI_y)^2 \\
&= \sum_{x,y} w(x,y)(u^2 I_x^2 + v^2 I_y^2 + 2uvI_x I_y) \\
&= [u,v]\, M \begin{bmatrix} u \\ v \end{bmatrix}
\end{aligned}
$$

Harris Corner: Mathematics

- For small shift [u,v], we have 1st order Taylor approximation

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

- Where M is a 2x2 matrix computed from image derivatives:

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

M is called the second order moments (or auto correlation) matrix of gradients.

Readings

- Chapter 4.3 Lines in
- Book “Computer Vision: Algorithms and Applications”

Harris Corner: Mathematics

Intensity change in shifting window: eigenvalue analysis

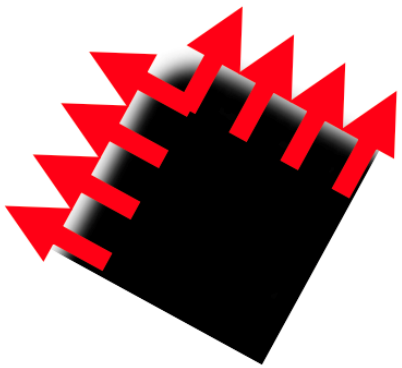
$$E(u,v) \cong [u,v] \quad M \quad \begin{bmatrix} u \\ v \end{bmatrix}$$

λ_1, λ_2 – eigenvalues of M

If we try every possible orientation **n**,
the biggest change and smallest changes in intensity happen in λ_s.

Recall: eigenvalue decomposition in linear algebra

Since *M* is symmetric, we have



$$M = X \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} X^T$$

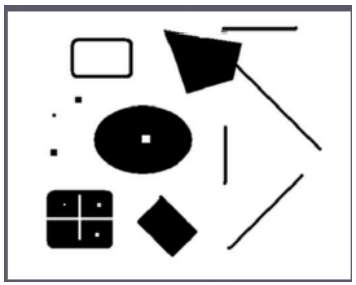
$$Mx_i = \lambda_i x_i$$

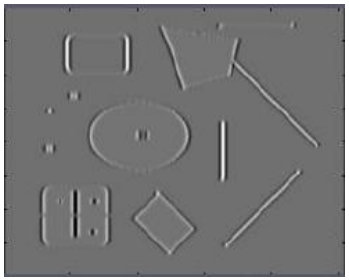
The *eigenvalues* of *M* reveal the amount of intensity change in the two principal orthogonal gradient directions in the window.

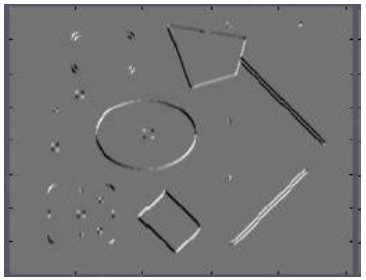
Corners as distinctive interest points

$$M = \sum w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

2 x 2 matrix of image derivatives (averaged in neighborhood of a point).







Notation:

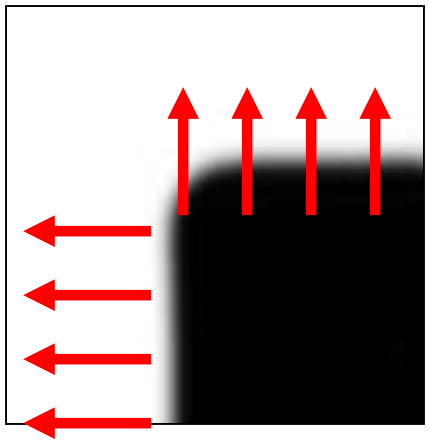
$$I_x \Leftrightarrow \frac{\partial I}{\partial x}$$

$$I_y \Leftrightarrow \frac{\partial I}{\partial y}$$

$$I_x I_y \Leftrightarrow \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$$

What does this matrix reveal?

First, consider an axis-aligned corner:



What does this matrix reveal?

First, consider an axis-aligned corner:

$$M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

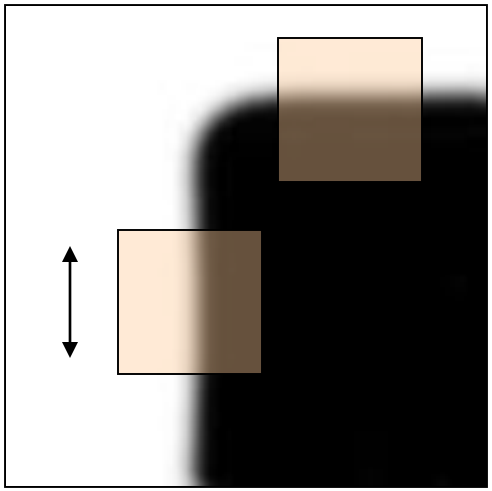
This means dominant gradient directions align with x or y axis

Look for locations where **both** λ's are large.

If either λ is close to 0, then this is **not** corner-like.

What if we have a corner that is not aligned with the image axes?

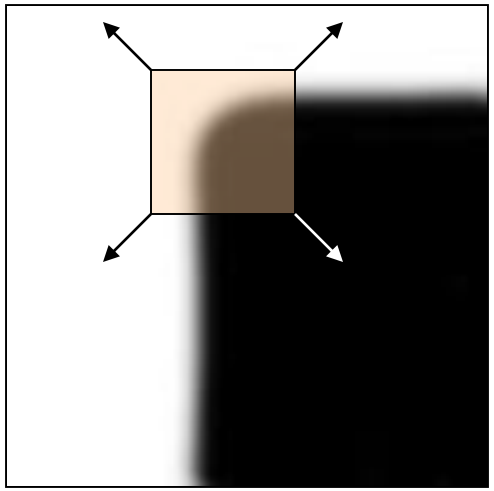
Corner response function



"edge":

$\lambda_1 >> \lambda_2$

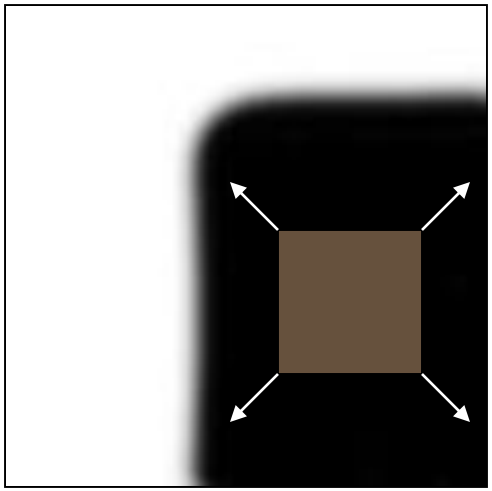
$\lambda_2 >> \lambda_1$



"corner":

λ_1 and λ_2 are large,

$\lambda_1 \sim \lambda_2$;



"flat" region

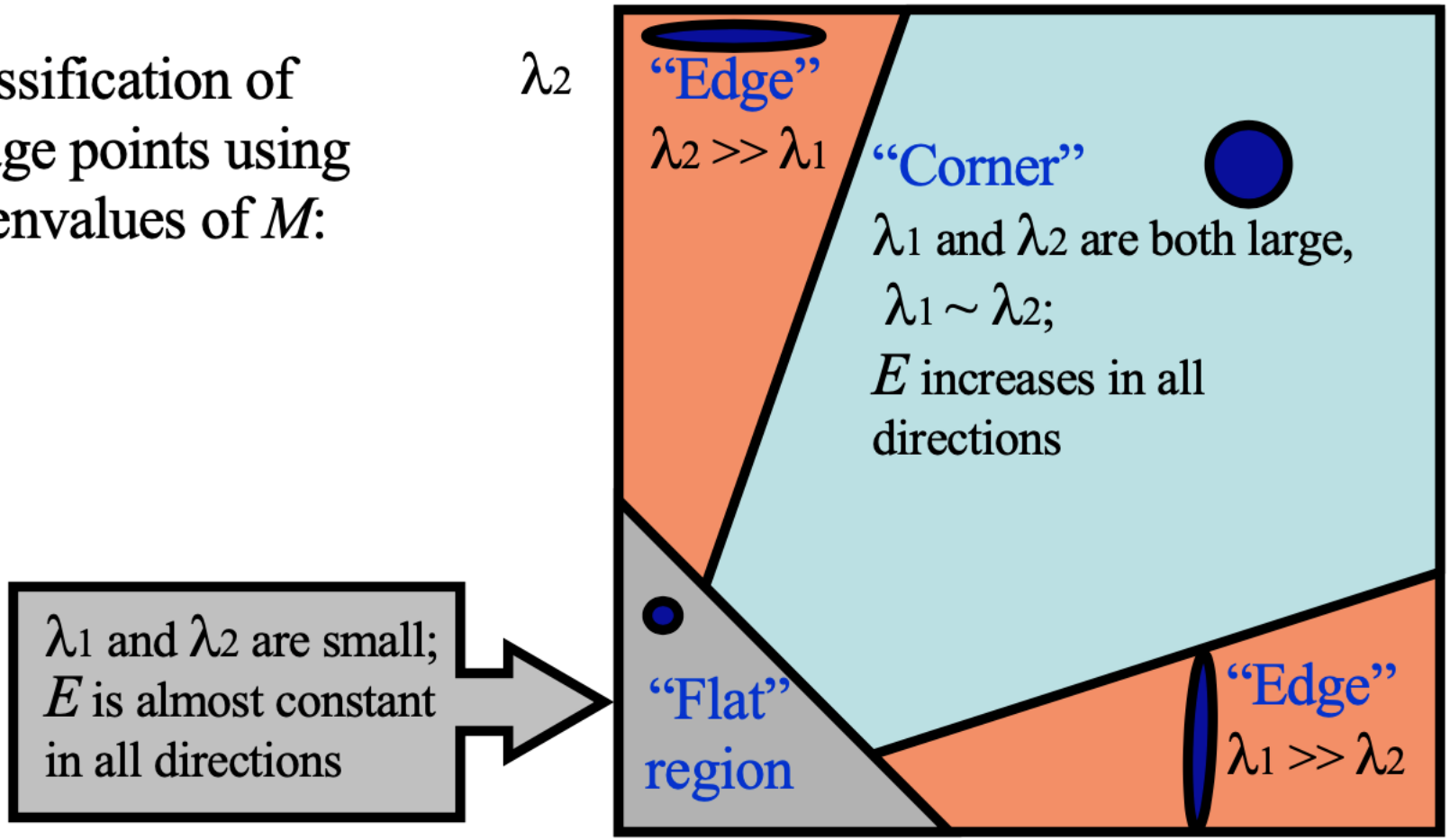
λ_1 and λ_2 are small;

Harris corner detector

1) Compute *M* matrix for each image window to get their *cornerness* scores.
2) Find points whose surrounding window gave large corner response (*R*> threshold)
3) Take the points of local maxima, i.e., perform non-maximum suppression

Harris Corner Detector

Classification of image points using eigenvalues of M:



Harris Corner Detector: Mathematics

Measure of corner response: (Cornerness)

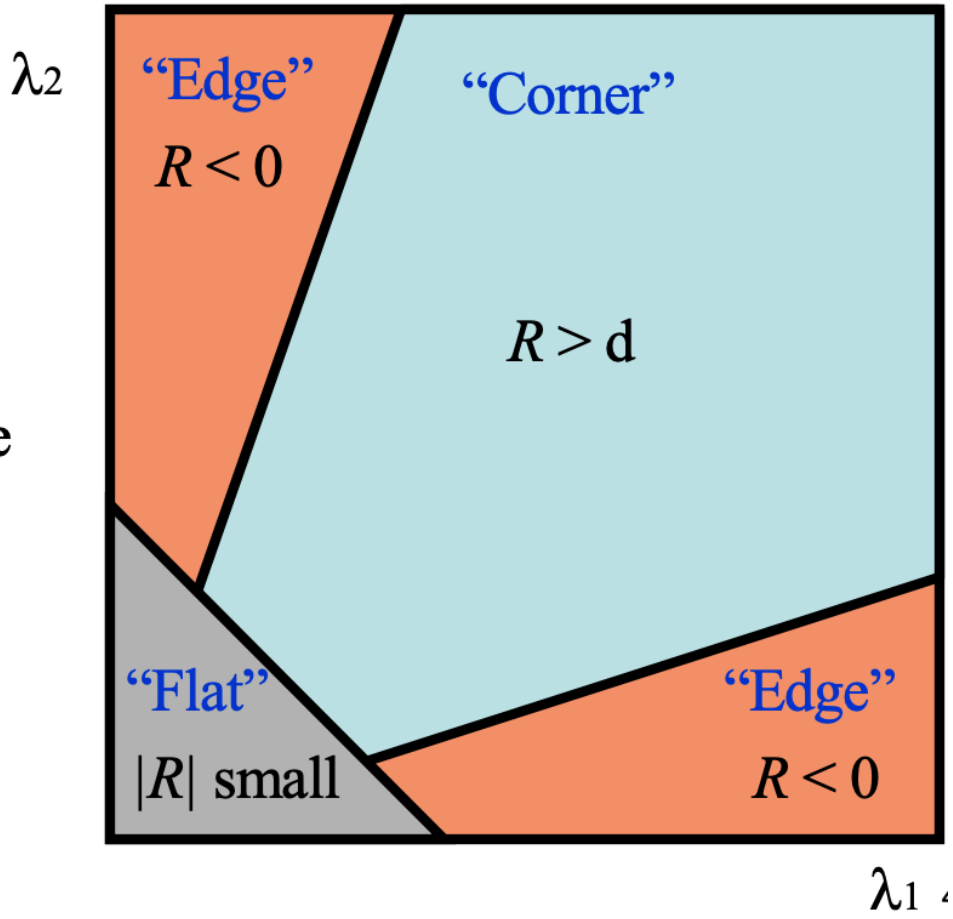
$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

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

(k – empirical constant, $k = 0.01\text{-}0.1$)

Harris Corner Detector: Mathematics

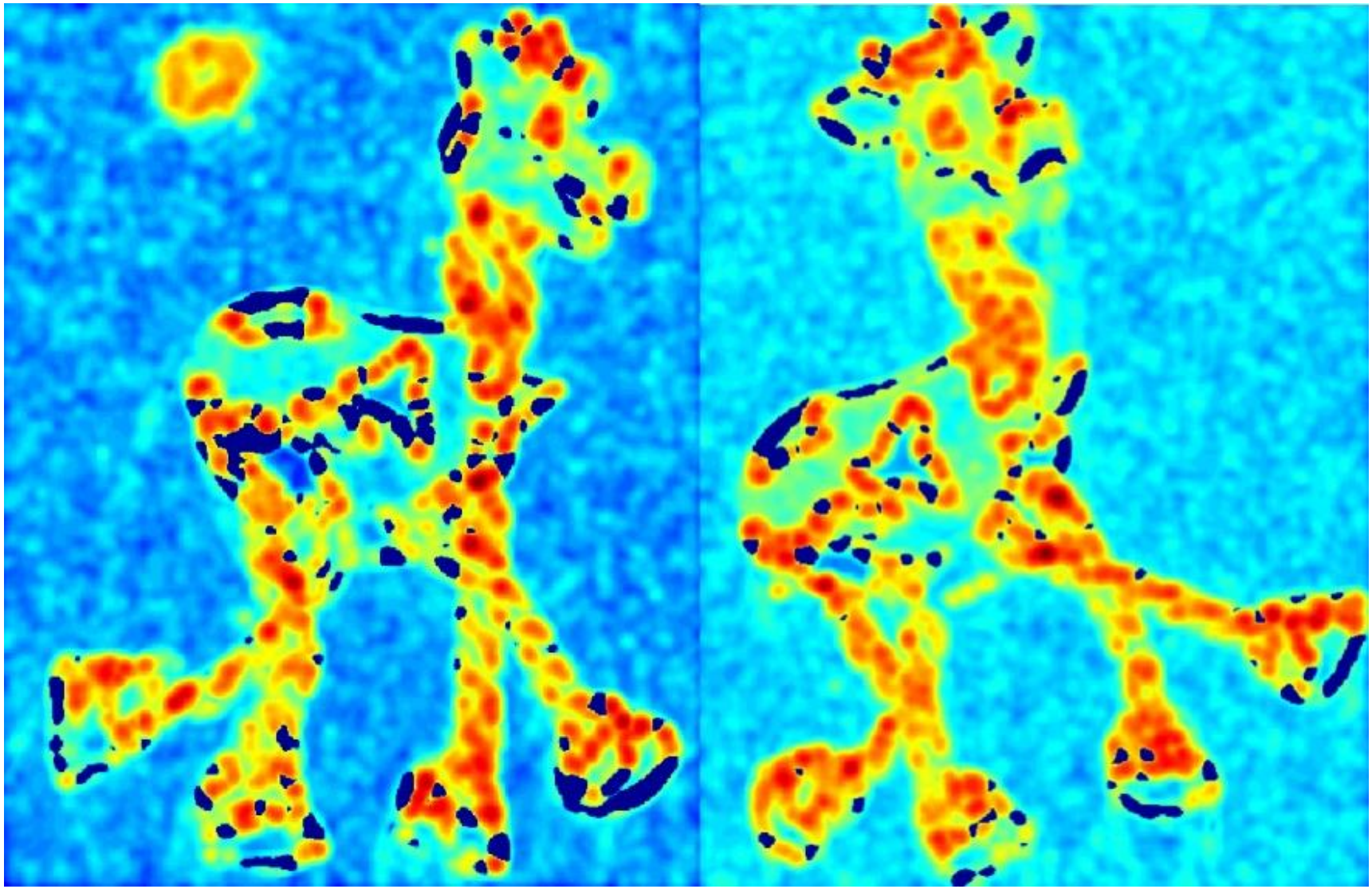
- R depends only on eigenvalues of M
- R is large for a corner
- R is negative with large magnitude for an edge
- $|R|$ is small for a flat region



Harris Detector: Steps

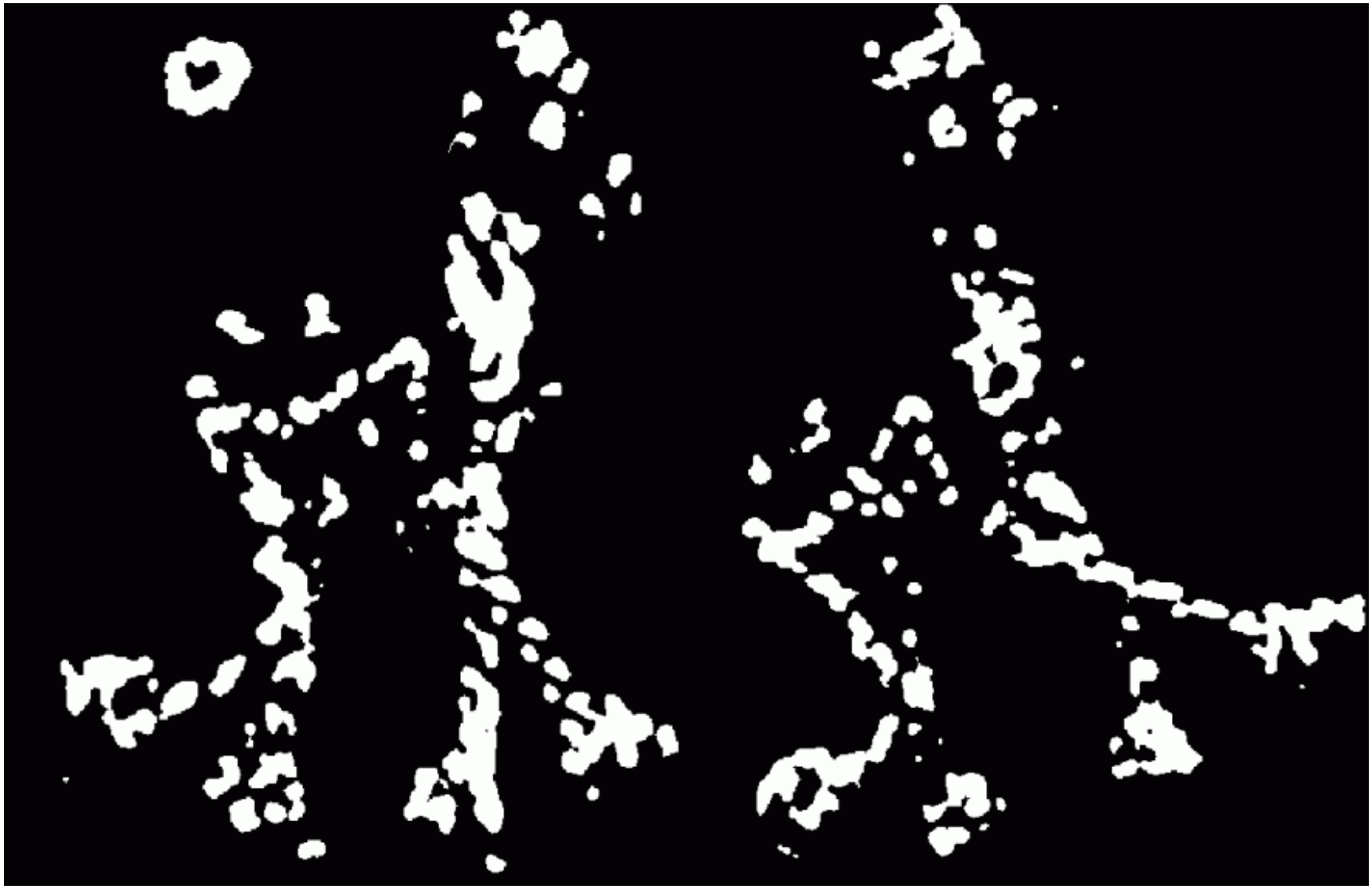
Harris Detector: Steps

Compute corner response *R (i.e. the "cornerness")*



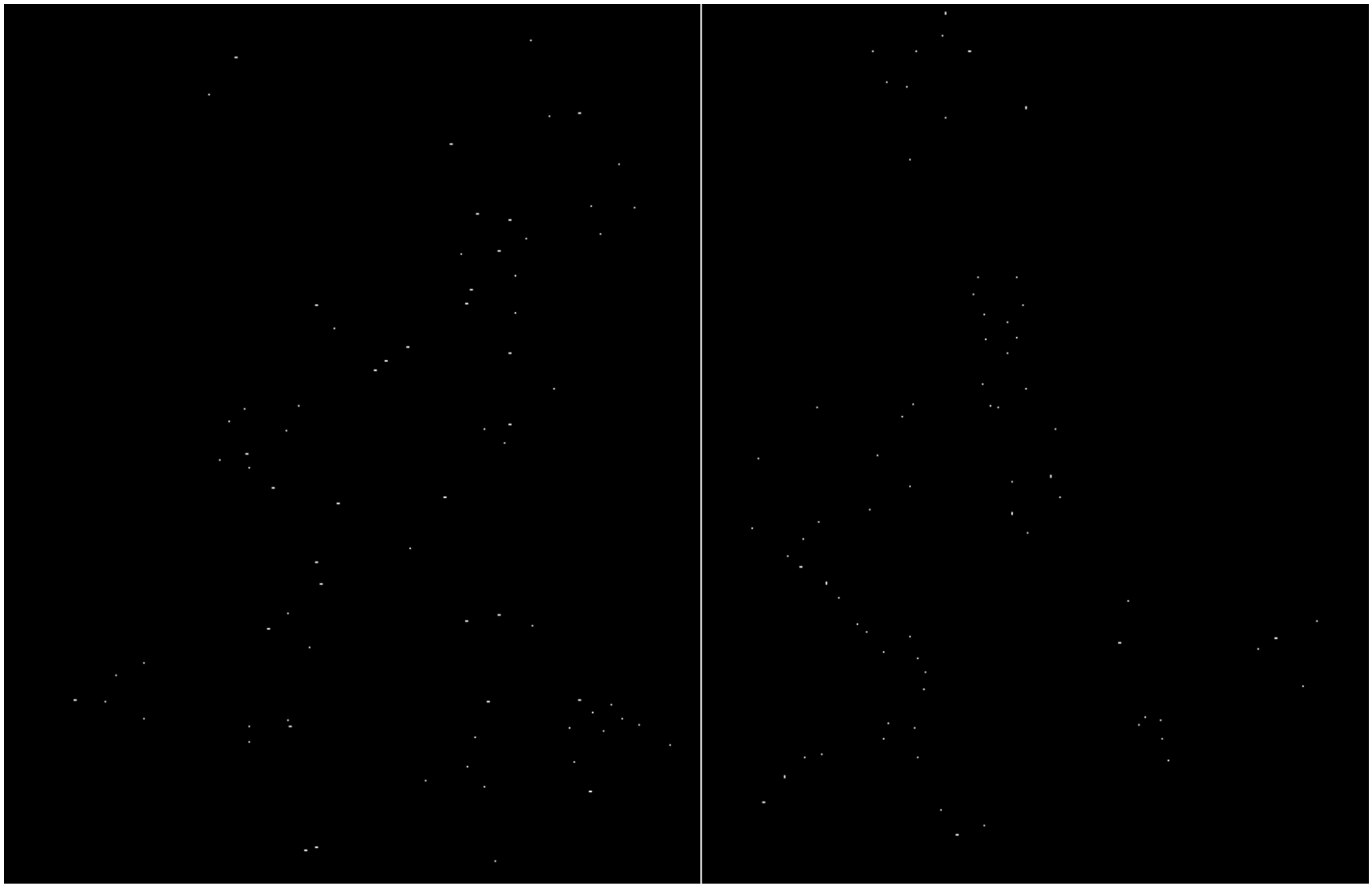
Harris Detector: Steps

Find points with large corner response: R> threshold

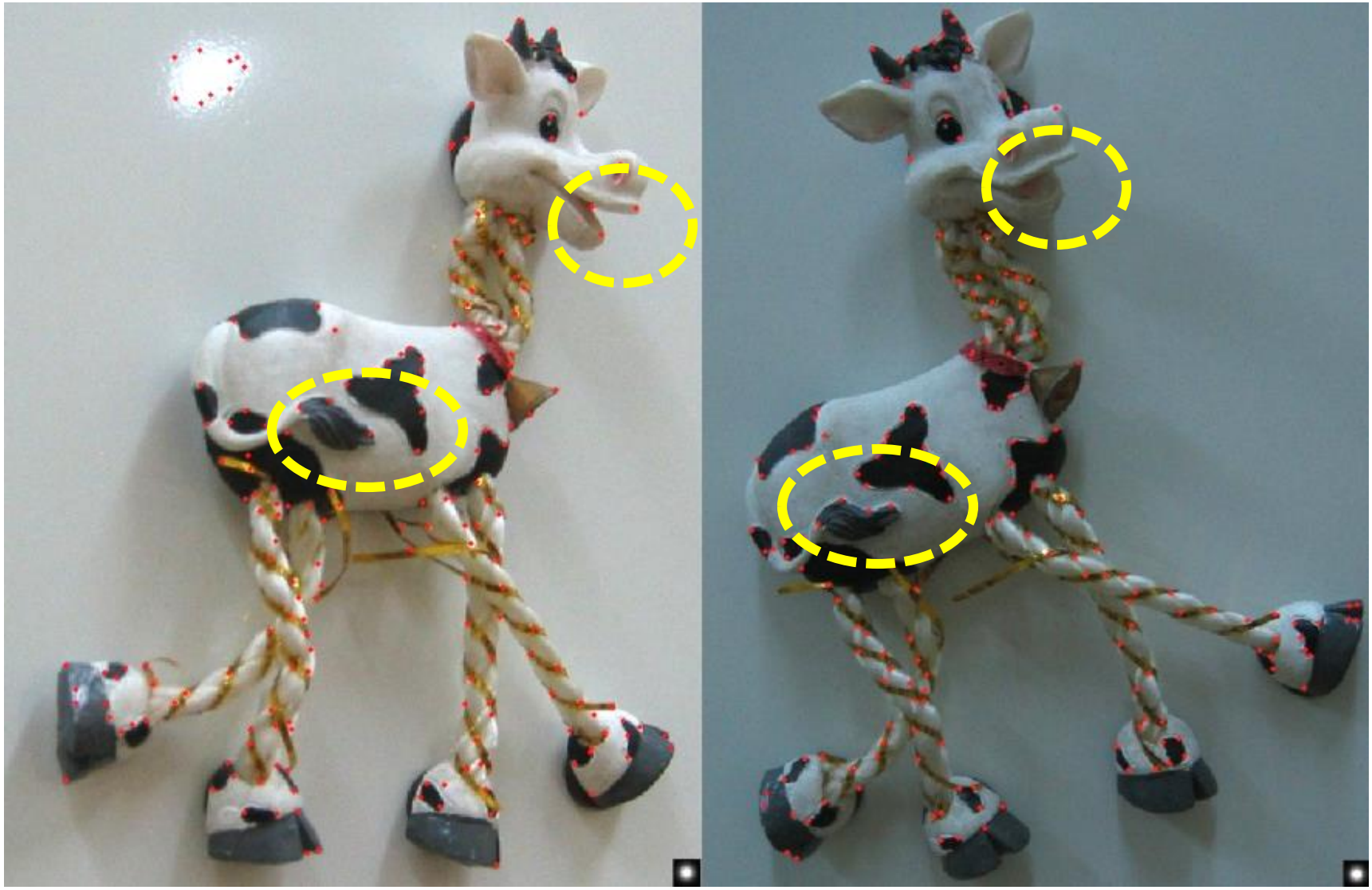


Harris Detector: Steps

Take only the points of local maxima of *R*



Harris Detector: Steps



Recap: Harris Corner

- Compute the moment (auto-correlation) matrix M
  - captures the structure of the local neighborhood
  - measure based on **eigenvalues** of M
    - 2 strong eigenvalues => interest point
    - 1 strong eigenvalue => edge or contour
    - 0 or very weak eigenvalues => flat region

- Interest point detection
  - threshold on the eigenvalues
  - local maximum for localization

Recap: Harris Corner

- Corner strength $R = \det(M) - k\,\mathrm{Tr}(M)^2$
- Let α and β be the two eigenvalues. We don't have to calculate them! Instead, use trace and determinant:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \qquad \det(\mathbf{A}) = a_{11}a_{22} - a_{12}a_{21}$$
$$\mathrm{tr}(\mathbf{A}) = a_{11} + a_{22}$$

- R is positive for corners, negative for edges, and small for flat regions
- **Non-maximal suppression**: select corners that are 8-way local maxima

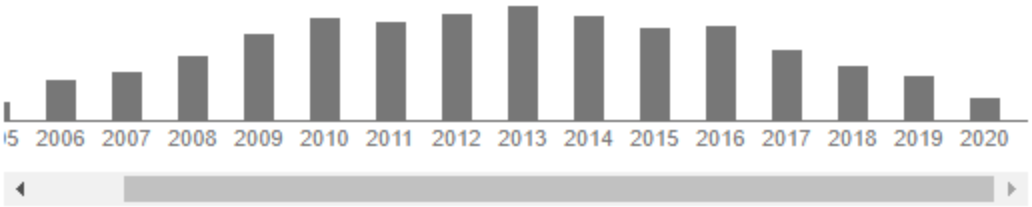
Harris Corner in Matlab Code

```
% Harris Corner detector

sigma=2; thresh=0.1; sze=11; disp=0;

% Derivative masks
dy = [-1 0 1; -1 0 1; -1 0 1];
dx = dy'; %dx is he transpose matrix of dy

% Calculate the coordinates of the image matrix
maxima

[rws,cols] = find(cornerness); % Find row,col coords.  clf ;
imshow(bw);
hold on;
p=[cols rws];
plot(p(:,1),p(:,2),'or');
title('\bf Harris Corners')
```

In your C-Lab-2:

A performance evaluation of local descriptors

Authors	Krystian Mikolajczyk, Cordelia Schmid
Publication date	2005/8/22
Journal	IEEE transactions on pattern analysis and machine intelligence
Volume	27
Issue	10
Pages	1615-1630
Publisher	IEEE
Description	In this paper, we compare the performance of descriptors computed for local interest regions, as, for example, extracted by the Harris-Affine detector [Mikolajczyk, K and Schmid, C, 2004]. Many different descriptors have been proposed in the literature. It is unclear which descriptors are more appropriate and how their performance depends on the interest region detector. The descriptors should be distinctive and at the same time robust to changes in viewing conditions as well as to errors of the detector. Our evaluation uses as criterion recall with respect to precision and is carried out for different image transformations. We compare shape context [Belongie, S, et al., April 2002], steerable filters [Freeman, W and Adelson, E, Setp. 1991], PCA-SIFT [Ke, Y and Sukthankar, R, 2004], differential invariants [Koenderink, J and van Doorn, A, 1987], spin images [Lazebnik, S, et al., 2003], SIFT [Lowe, D. G., 1999], complex filters ...
Total citations	Cited by 9510



Scholar articles
A performance evaluation of local descriptors
K Mikolajczyk, C Schmid - IEEE transactions on pattern analysis and machine ..., 2005
Cited by 9510 Related articles All 62 versions