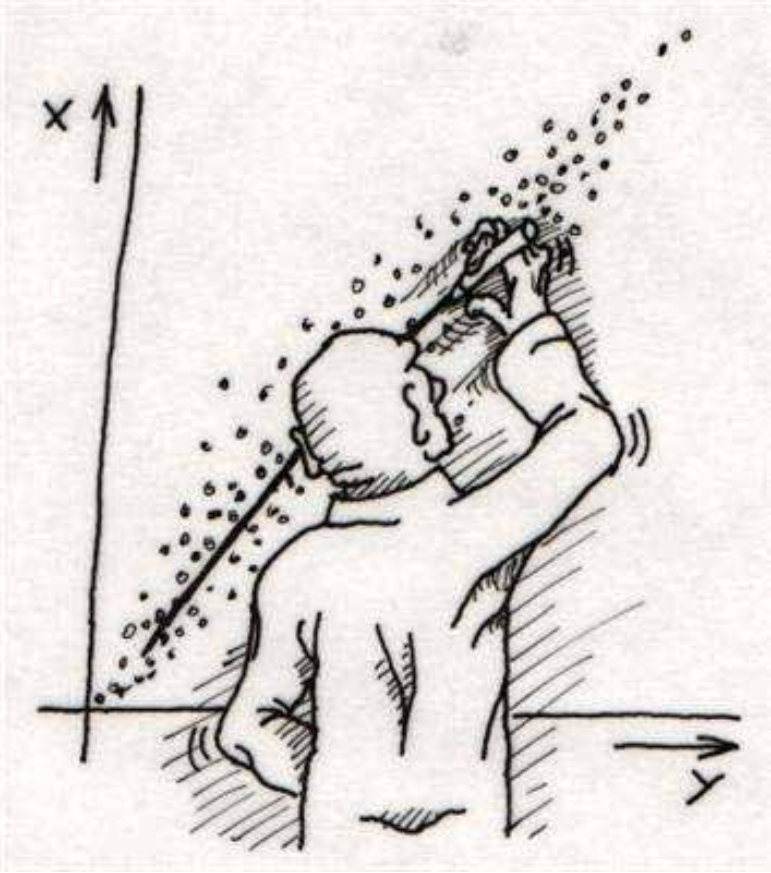
Model Fitting: The Hough transform I

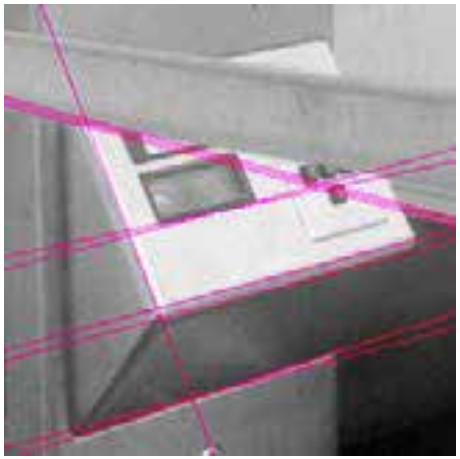
Guido Gerig, CS6640 Image Processing, Utah



Credit: Svetlana Lazebnik (Computer Vision UNC Chapel Hill, 2008)

Fitting Parametric Models: Beyond Lines

- Choose a parametric model to represent a set of features



simple model: **lines**

simple model: **circles**

complicated model: **car**

Source: K. Grauman

Fitting

- Choose a parametric model to represent a set of features
- Membership criterion is not local
  - Can’t tell whether a point belongs to a given model just by looking at that point
- Three main questions:
  - What model represents this set of features best?
  - Which of several model instances gets which feature?
  - How many model instances are there?
- Computational complexity is important
  - It is infeasible to examine every possible set of parameters and every possible combination of features

Fitting: Issues

- **Noise** in the measured feature locations
- **Extraneous data:** clutter (outliers), multiple lines
- **Missing data:** occlusions

Case study: Line detection

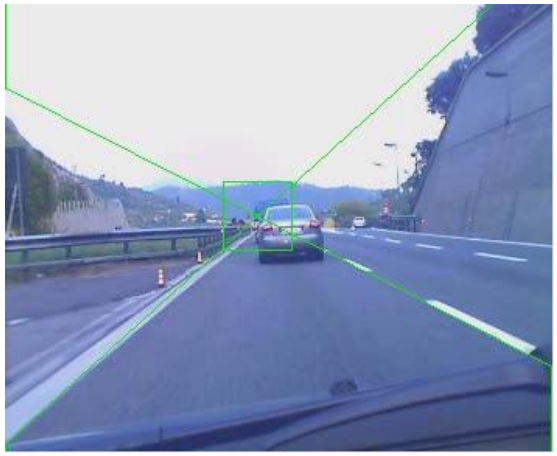


IMAGE PROCESSING LANE DEPARTURE SYSTEM WITH EDGE DETECTION TECHNIQUE USING HOUGH TRANSFORM – A.RAJYA LAKSHMI & J. MOUNIKA (link)

Voting schemes

- Let each feature vote for all the models that are compatible with it
- Hopefully the noise features will not vote consistently for any single model
- Missing data doesn’t matter as long as there are enough features remaining to agree on a good model

Hough transform

- An early type of voting scheme
- General outline:
  - Discretize parameter space into bins
  - For each feature point in the image, put a vote in every bin in the parameter space that could have generated this point
  - Find bins that have the most votes

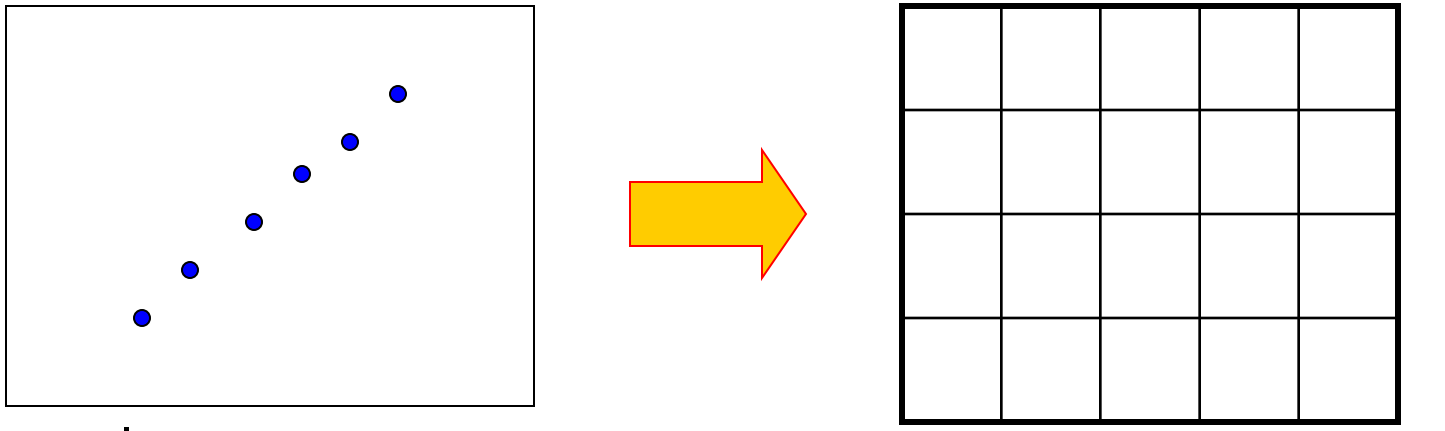


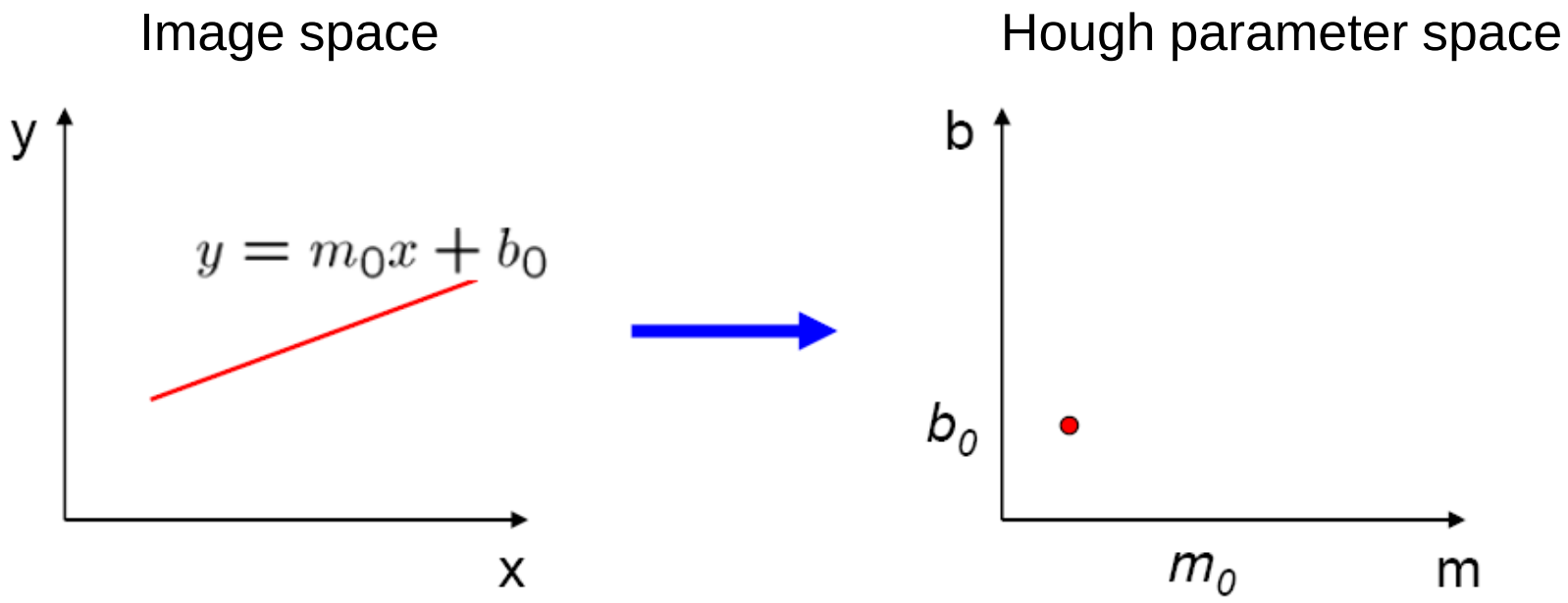
Image space

Hough parameter space

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

Parameter space representation

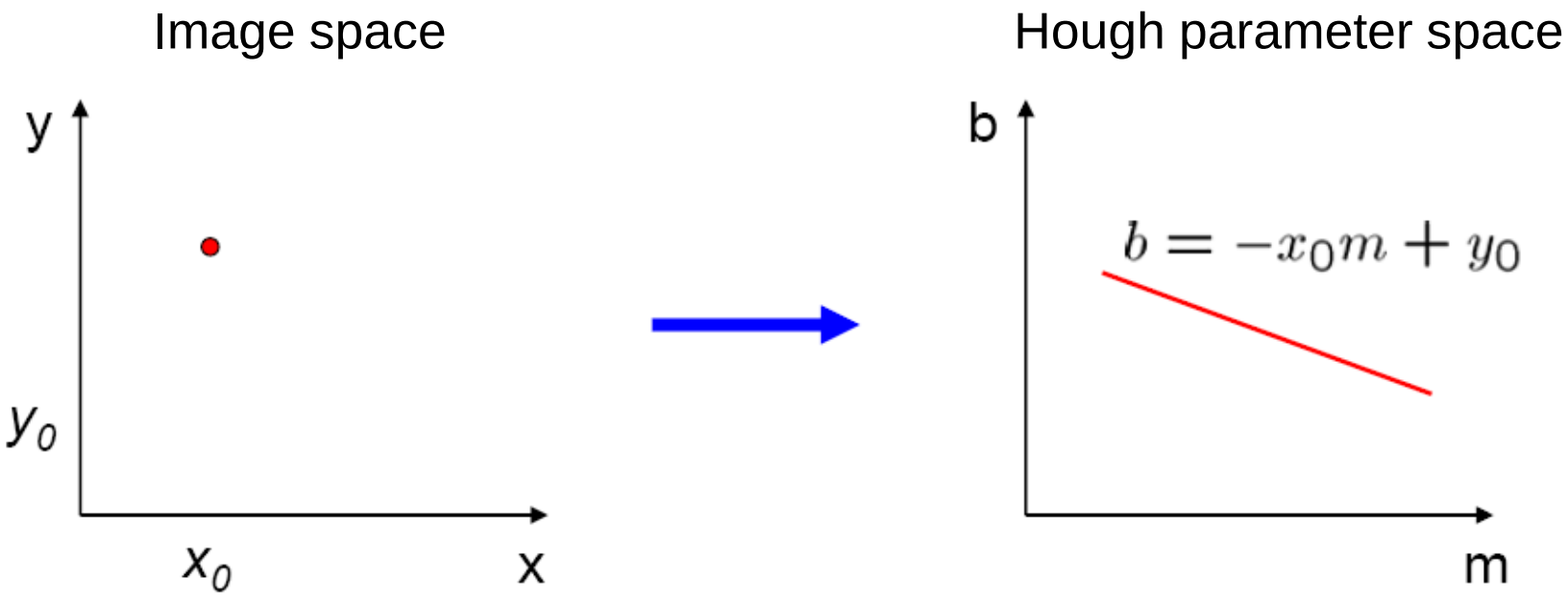
- A line in the image corresponds to a point in Hough space



Source: K. Grauman

Parameter space representation

- What does a point (x_0, y_0) in the image space map to in the Hough space?
  - Answer: the solutions of $b = –x_0m + y_0$
  - This is a line in Hough space

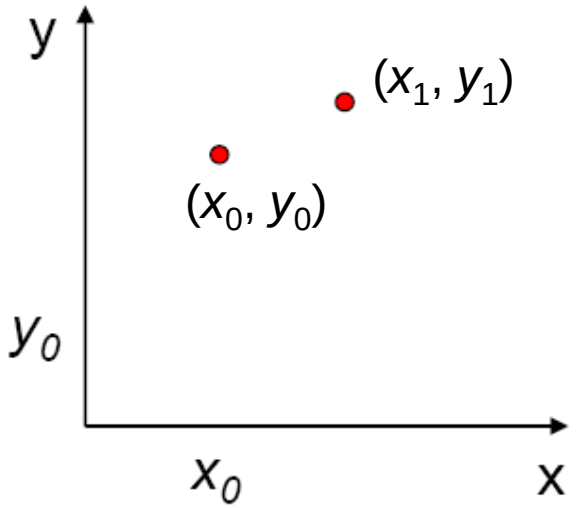


Source: K. Grauman

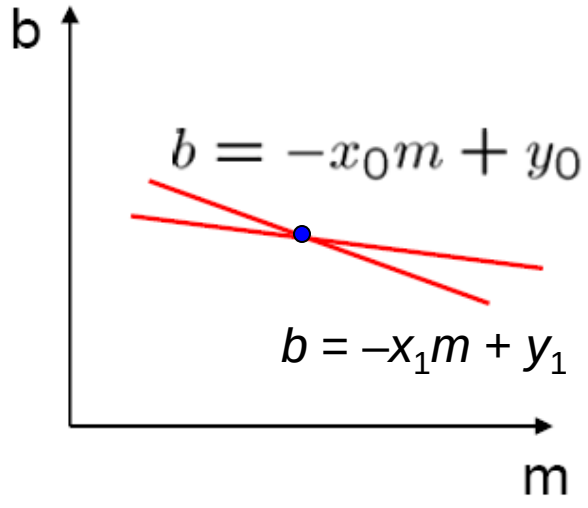
Parameter space representation

- Where is the line that contains both (x_0, y_0) and (x_1, y_1)?
  - It is the intersection of the lines $b = –x_0m + y_0$ and $b = –x_1m + y_1$

Image space



Hough parameter space

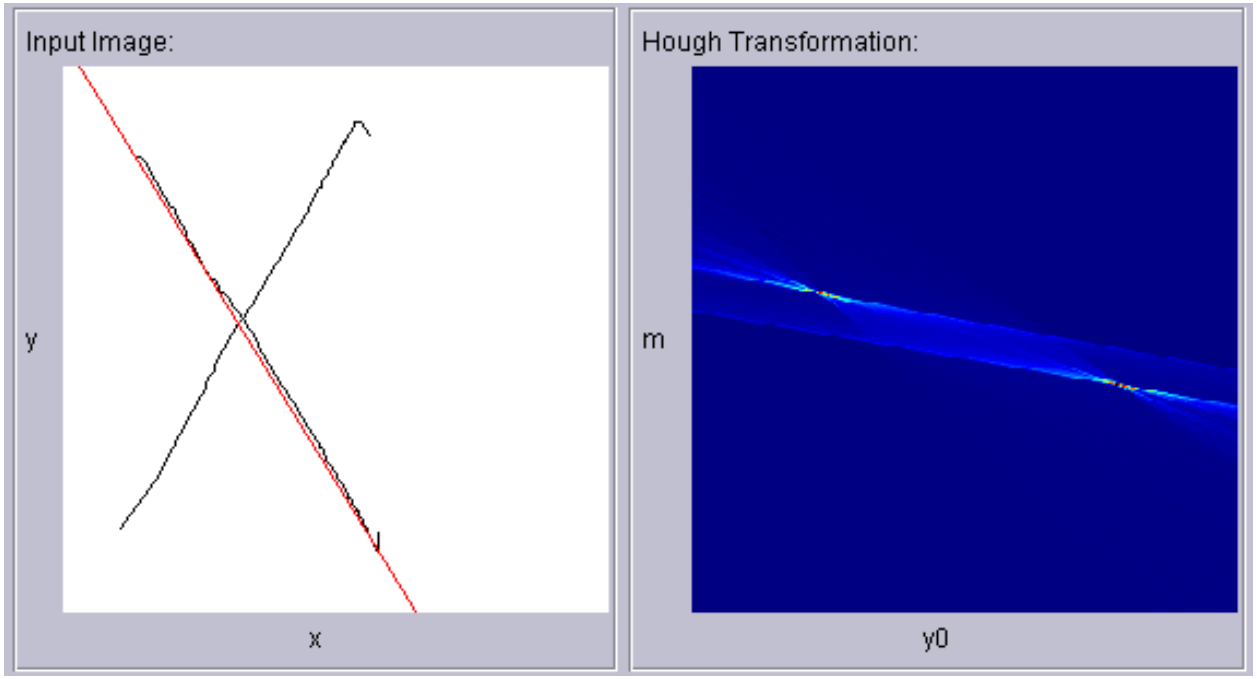


Source: K. Grauman

Java Demo

Java Demo: http://www.physik.uni-osnabrueck.de/nonlinop/Hough/LineHough.html

Image space **Parameter Space**



- Parametrization: $\mathbf{y = y_0 + m*x}$
- Each pixel (x,y) represents line $\mathbf{y_0 = y - x*m}$ in parameter space
- Two lines create two clusters
- Two cluster centers (m,y0) represent the two lines
- → Detecting clusters in parameter space provides solutions for lines in image space

Hough transform

- An early type of voting scheme
- General outline:
  - Discretize parameter space into bins
  - For each feature point in the image, put a vote in every bin in the parameter space that could have generated this point
  - Find bins that have the most votes

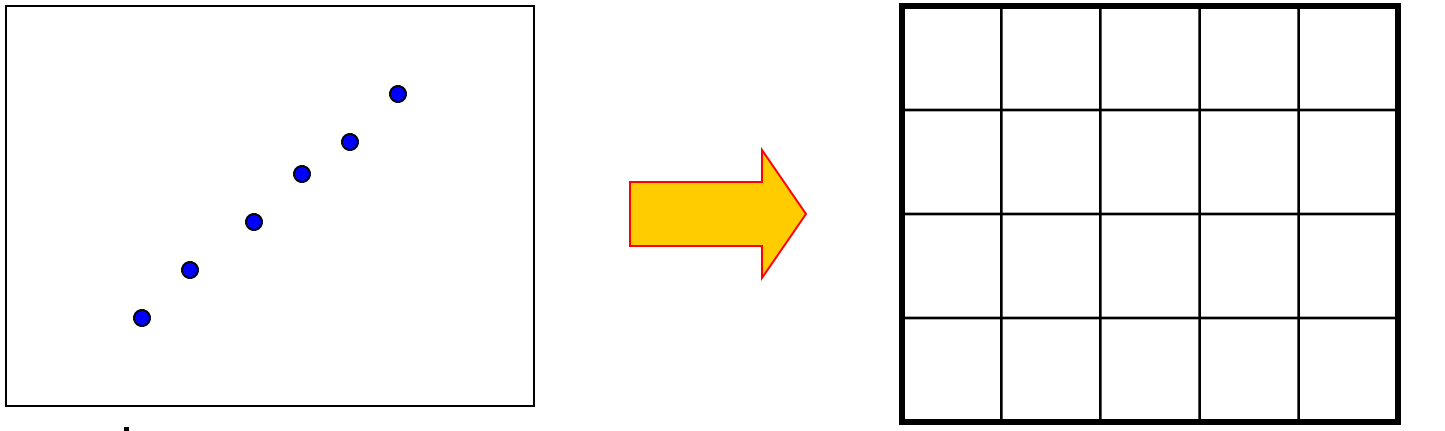
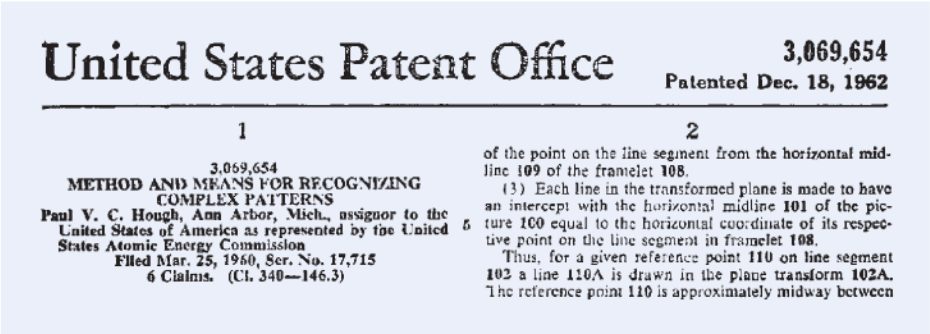


Image space

Hough parameter space

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

Hough Transform: PVC Hough, Patent 1962



United States Patent Office

3,069,654
Patented Dec. 18, 1962

1

3,069,654
METHOD AND MEANS FOR RECOGNIZING COMPLEX PATTERNS
Paul V. C. Hough, Ann Arbor, Mich., assignor to the United States of America as represented by the United States Atomic Energy Commission
Filed Mar. 25, 1960, Ser. No. 17,715
6 Claims. (Cl. 340—146.3)

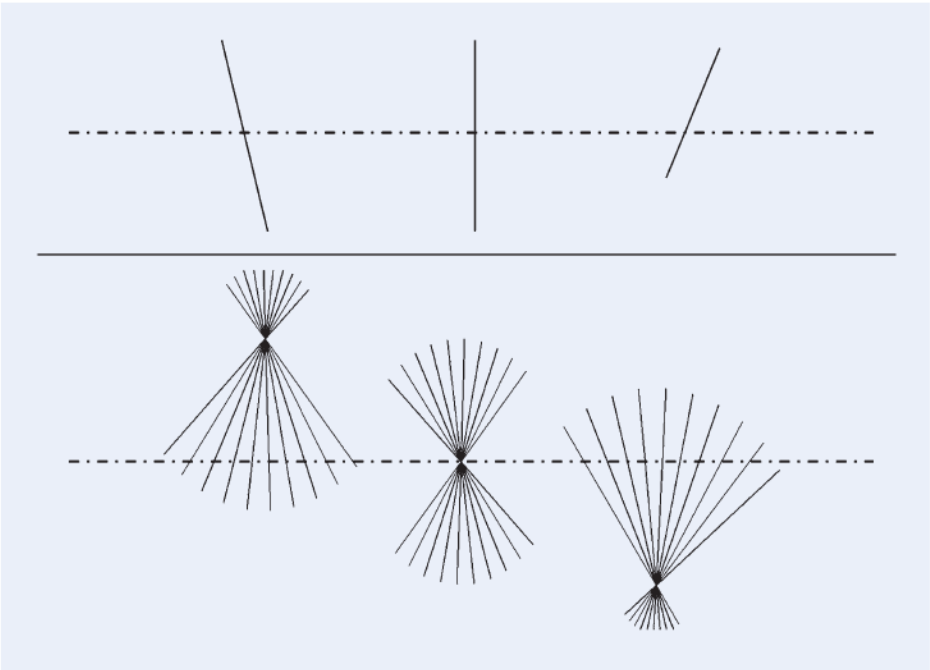
2

of the point on the line segment from the horizontal midline 109 of the framelet 108.

(3) Each line in the transformed plane is made to have an intercept with the horizontal midline 101 of the picture 100 equal to the horizontal coordinate of its respective point on the line segment in framelet 108.

Thus, for a given reference point 110 on line segment 102 a line 110A is drawn in the plane transform 102A. The reference point 110 is approximately midway between

[FIG1] From the title page of Paul V.C. Hough's patent.



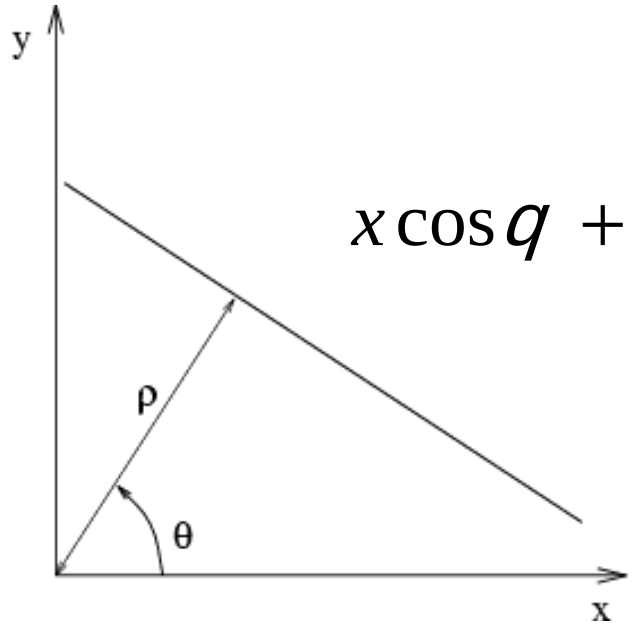
[FIG2] Graphical description of the transform in the Hough patent. A point in the upper image space or "framelet" maps to a line in the lower "transformed space." Colinear points map to lines that intersect at a "knot." [The original figure in the patent has been redrawn and slightly simplified here for clarity.]

Polar representation for lines

- Problems with the (m,b) space:
  - Unbounded parameter domain: m=[0….inf]
  - Vertical lines require infinite **m**
  - Parameter **m** does not change linearly with orientation of line → Parameter space shows different cluster representations for different lines

Polar representation for lines

- Problems with the (m,b) space:
  - Unbounded parameter domain
  - Vertical lines require infinite m
- Alternative: polar representation



$$x\cos q + y\sin q = r$$

Each point will add a sinusoid in the (q,r) **parameter space**

Polar representation / Normal Form

- Parametrization: $x\cos\theta + y\sin\theta = \rho$
- Image point (x,y), parameters (θ,ρ)
- What function is described by this parametrization?

Polar representation / Normal Form

- **Parametrization**: $x\cos\theta + y\sin\theta = r$
- Image point (x,y), parameters (θ,ρ)
- What function is described by this parametrization?

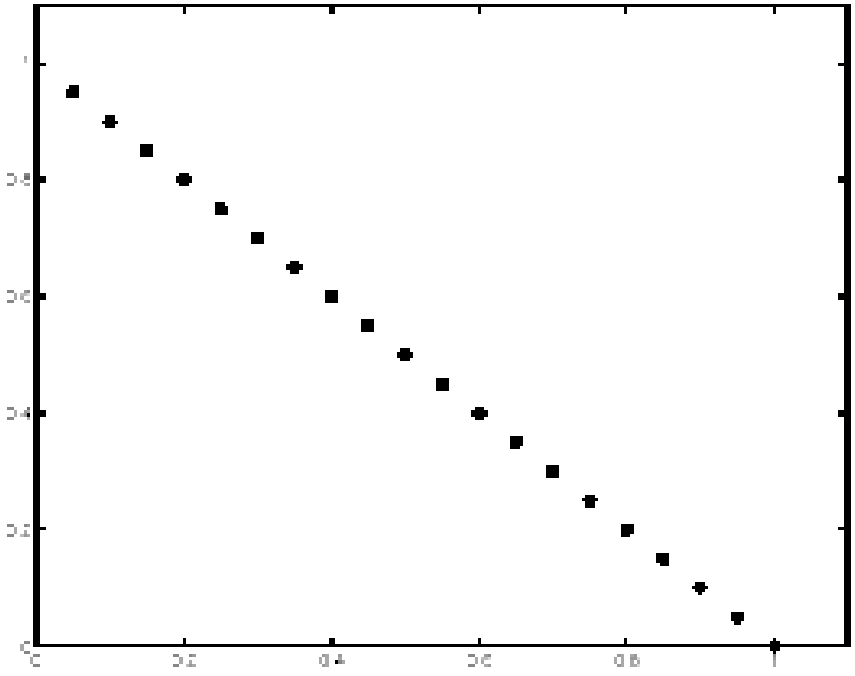
$$r = A\cos(\theta - \delta)$$

$$A = \sqrt{x^2 + y^2}$$

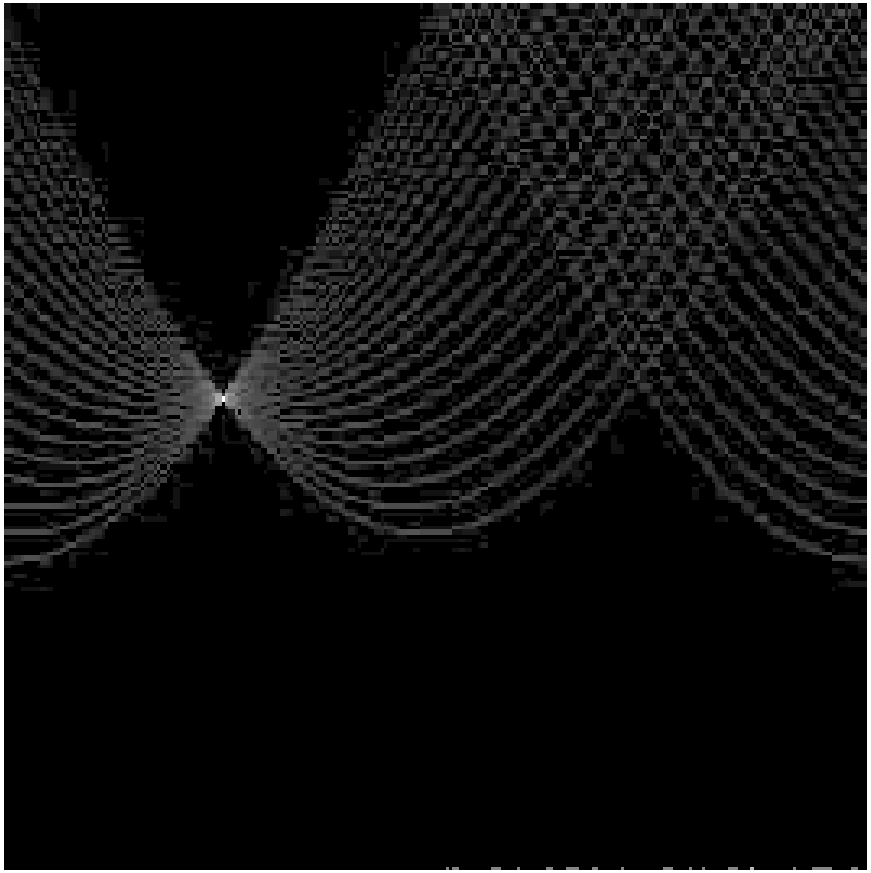
$$\delta = \tan^{-1}\left(\frac{y}{x}\right)$$

- **Result**: Given (x,y) → (A,δ) → Cos function in parameter space (see handout notes)

Basic illustration



features

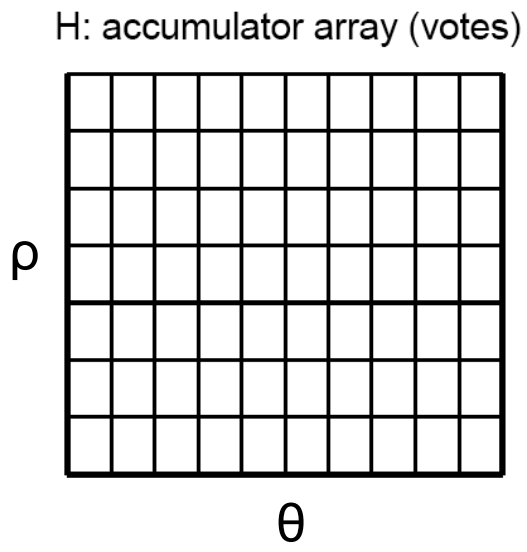


votes

Java Demo 1: http://www.dis.uniroma1.it/~iocchi/slides/icra2001/java/hough.html
Java Demo 2: ETH Zurich: http://www.vision.ee.ethz.ch/~cvcourse/brechbuehler/hough.html

Algorithm outline

- Initialize accumulator H to all zeros
- For each edge point (x,y) in the image
  For θ = 0 to 180
    ρ = x cos θ + y sin θ
    H(θ, ρ) = H(θ, ρ) + 1
  end
end
- Find the value(s) of (θ, ρ) where H(θ, ρ) is a local maximum
  - The detected line in the image is given by
    ρ = x cos θ + y sin θ



Properties

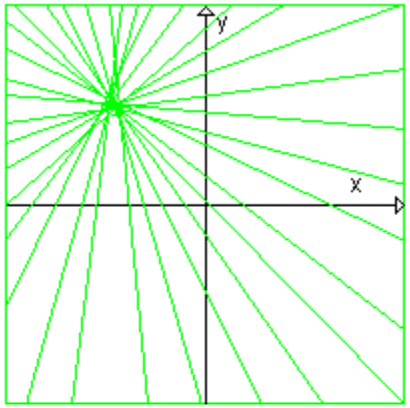
- Point (x,y) maps into Cos-curve. What do all these points represent?

- What happens with parallel lines?

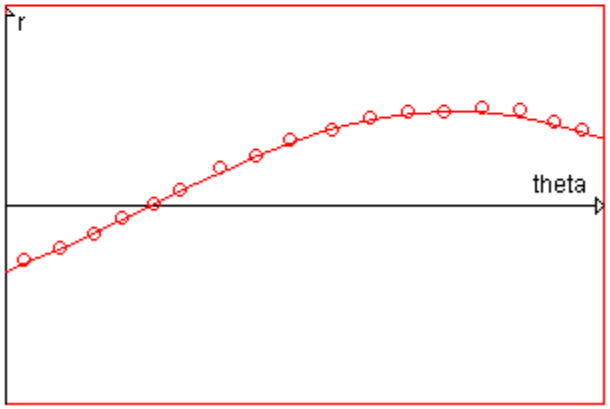
- What happens with perpendicular lines?

Properties

- Point (x,y) maps into Cos-curve. What do all these points represent?



- arallel lines?

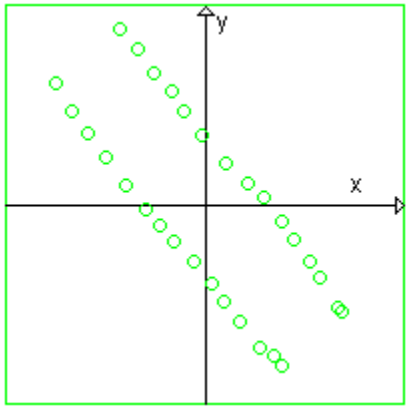


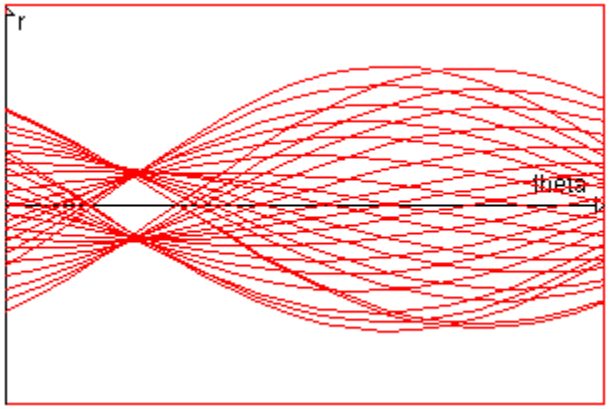
- erpendicular lines?

Java Demo 2: ETH Zurich: http://www.vision.ee.ethz.ch/~cvcourse/brechbuehler/hough.html

Properties

- Point (x,y) maps into Cos-curve. What do all these points represent?
- What happens with parallel lines?





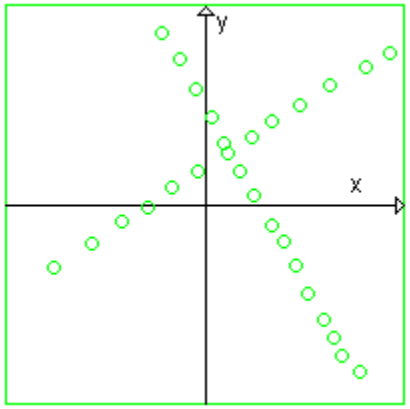
Same angles θ, different distances ρ

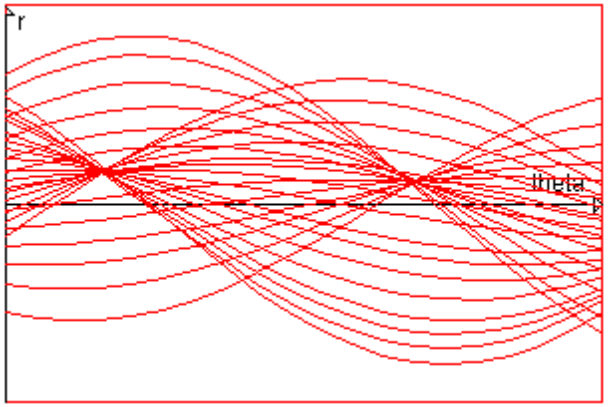
- What happens with perpendicular lines?

Java Demo 2: ETH Zurich: http://www.vision.ee.ethz.ch/~cvcourse/brechbuehler/hough.html

Properties

- Point (x,y) maps into Cos-curve. What do all these points represent?
- What happens with parallel lines?
- What happens with perpendicular lines?



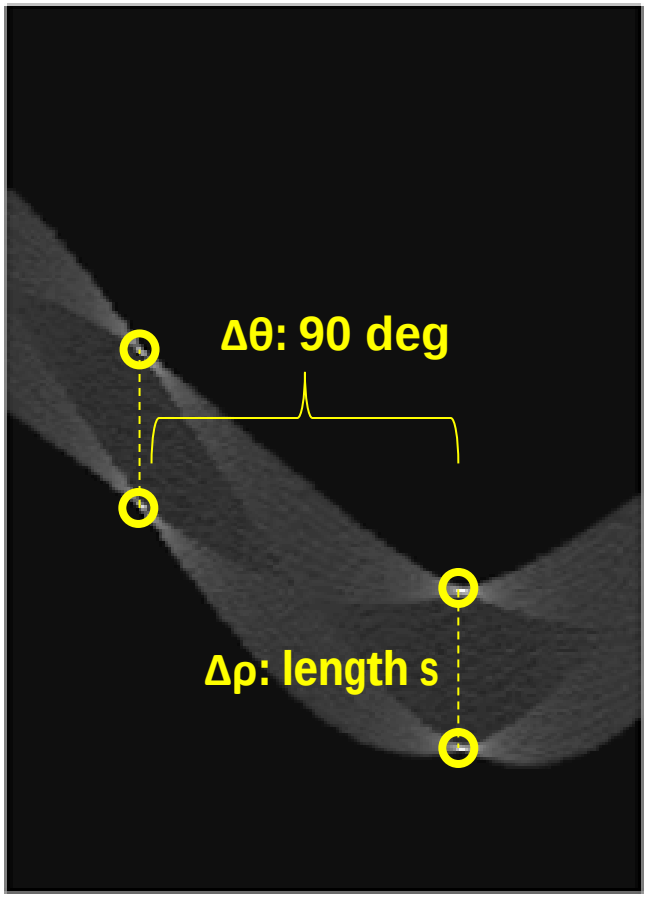


Angles θ are 90 deg apart, distances ρ are different

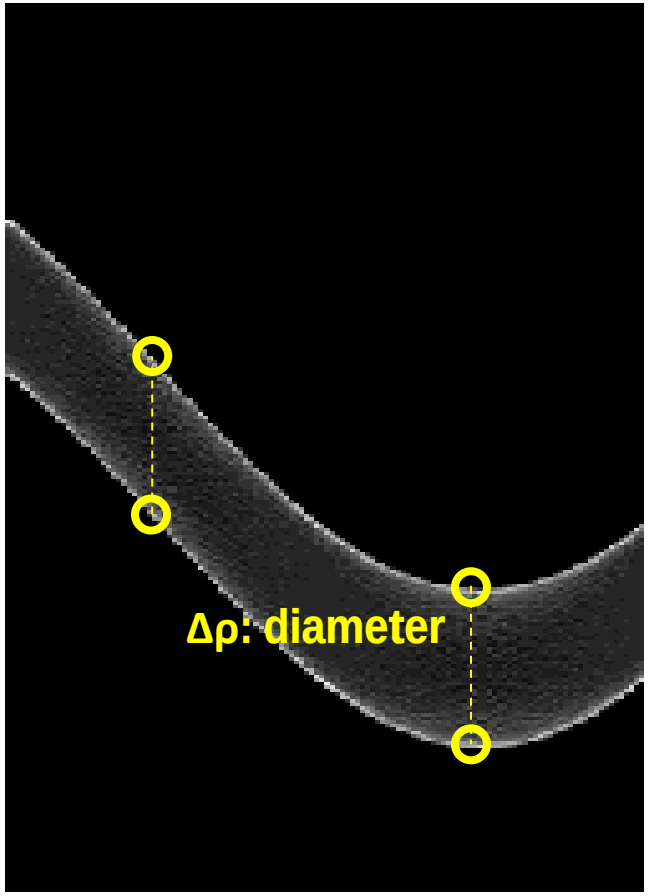
Java Demo 2: ETH Zurich: http://www.vision.ee.ethz.ch/~cvcourse/brechbuehler/hough.html

Other shapes

Square

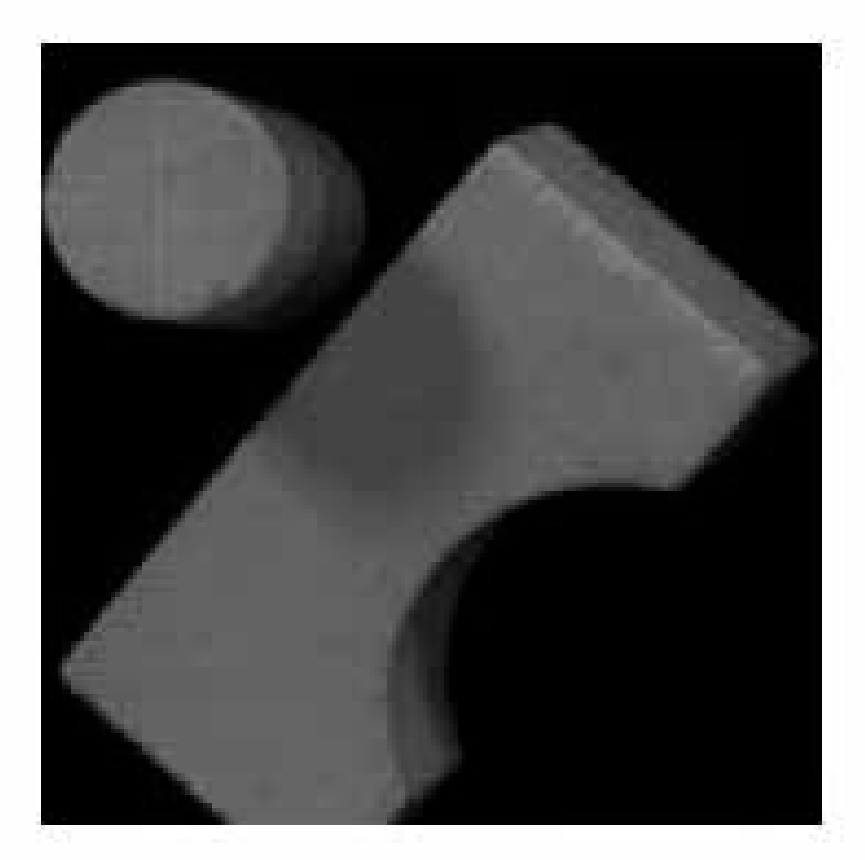


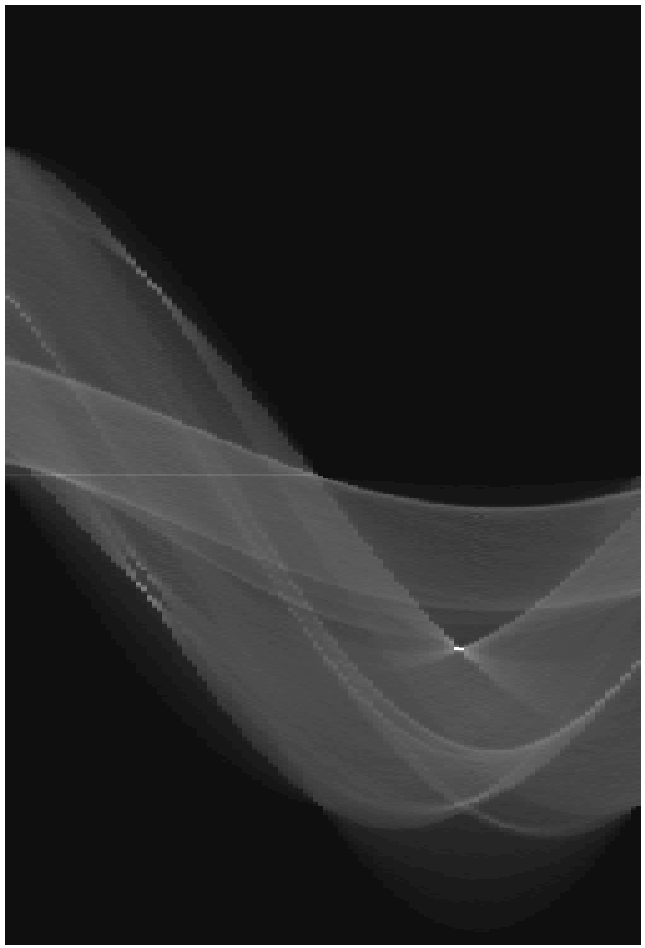
Circle



Java Demo 2: ETH Zurich: http://www.vision.ee.ethz.ch/~cvcourse/brechbuehler/hough.html

Parameter Space has Structure!





Another Java Demo: U of Southhamptom

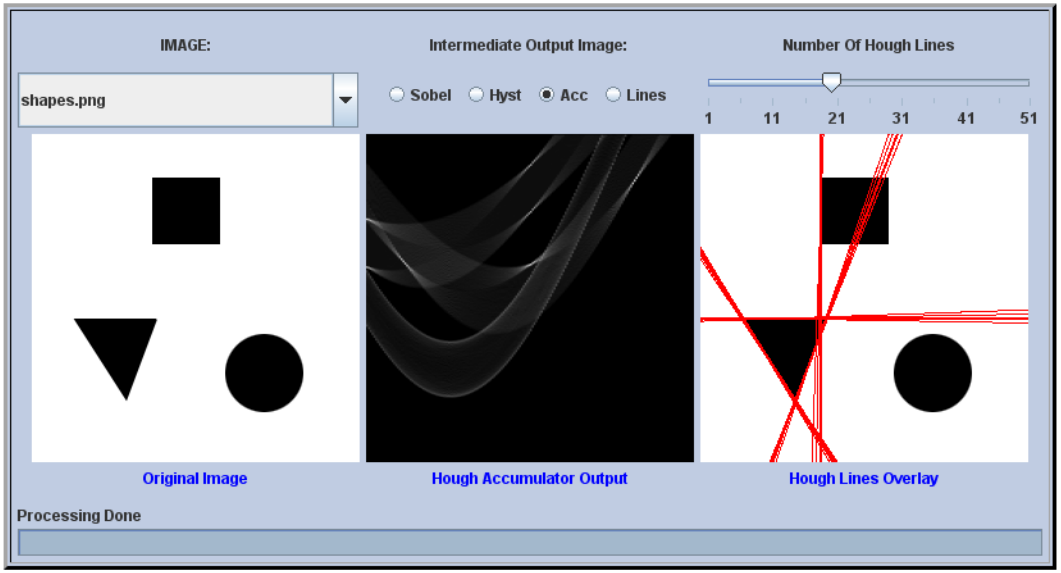
Computer Vision Demonstration Website

Electronics and Computer Science
University of Southampton

- Main Menu
- Home Page
- Filtering
- Fourier Transform
- Averaging
- Gaussian
- Median
- Thresholding
- Non-Maximum Suppression
- Low Level Feature Extraction
- Sobel Edge Detection
- Marr Hildreth Edge Detection
- Harris Corner Detection
- Optical Flow
- Finding Shapes
- Hough Transform for Lines
- Hough Transform for Circles
- Active Contours & Snakes
- Symmetry Operator
- Biometrics
- Face Recognition
- Iris Recognition

Hough Transform

The Hough Transform is a method for finding shapes in an image. The Line Hough has been implemented here, which is used to find lines within an image. The Hough Transform can also be used to find circles, or even generalised to find any shapes. The applet below allows the number of output lines to be selected. The "Acc" option shows the output of the accumulator (polar space output image).

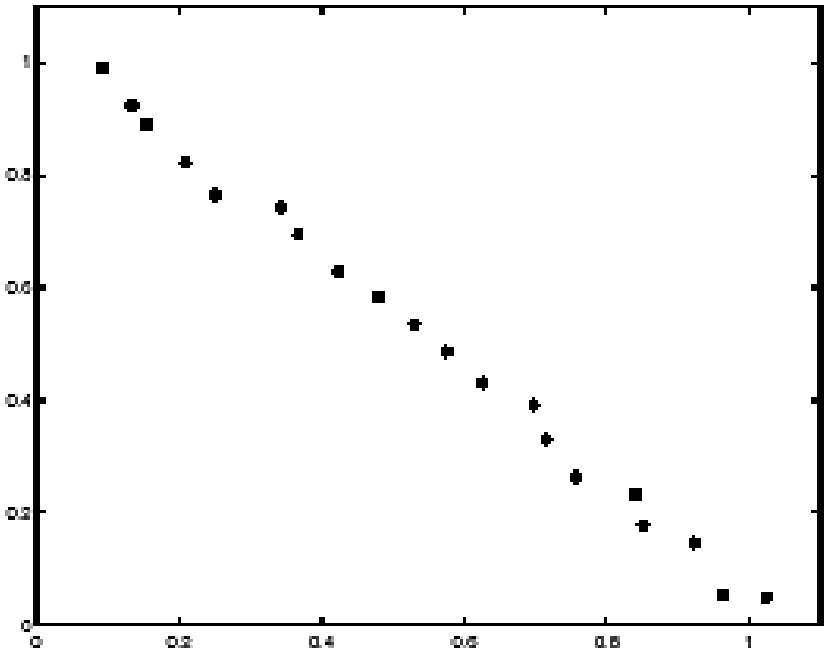


How it works

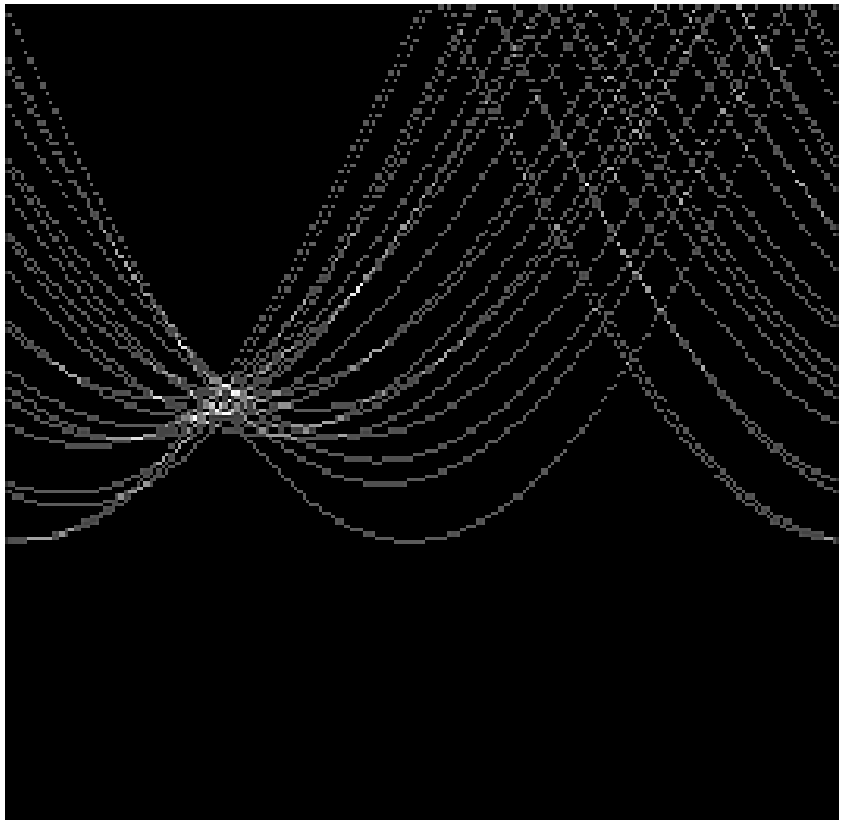
The line Hough Transform generates a curve in polar space for each edge point in Cartesian space. This polar HT accumulator array contains votes for each parameter pair. The peaks in this image are then inverse transformed to give a set of lines that represent these edges.

http://users.ecs.soton.ac.uk/msn/book/new_demo/hough/

Effect of noise



features

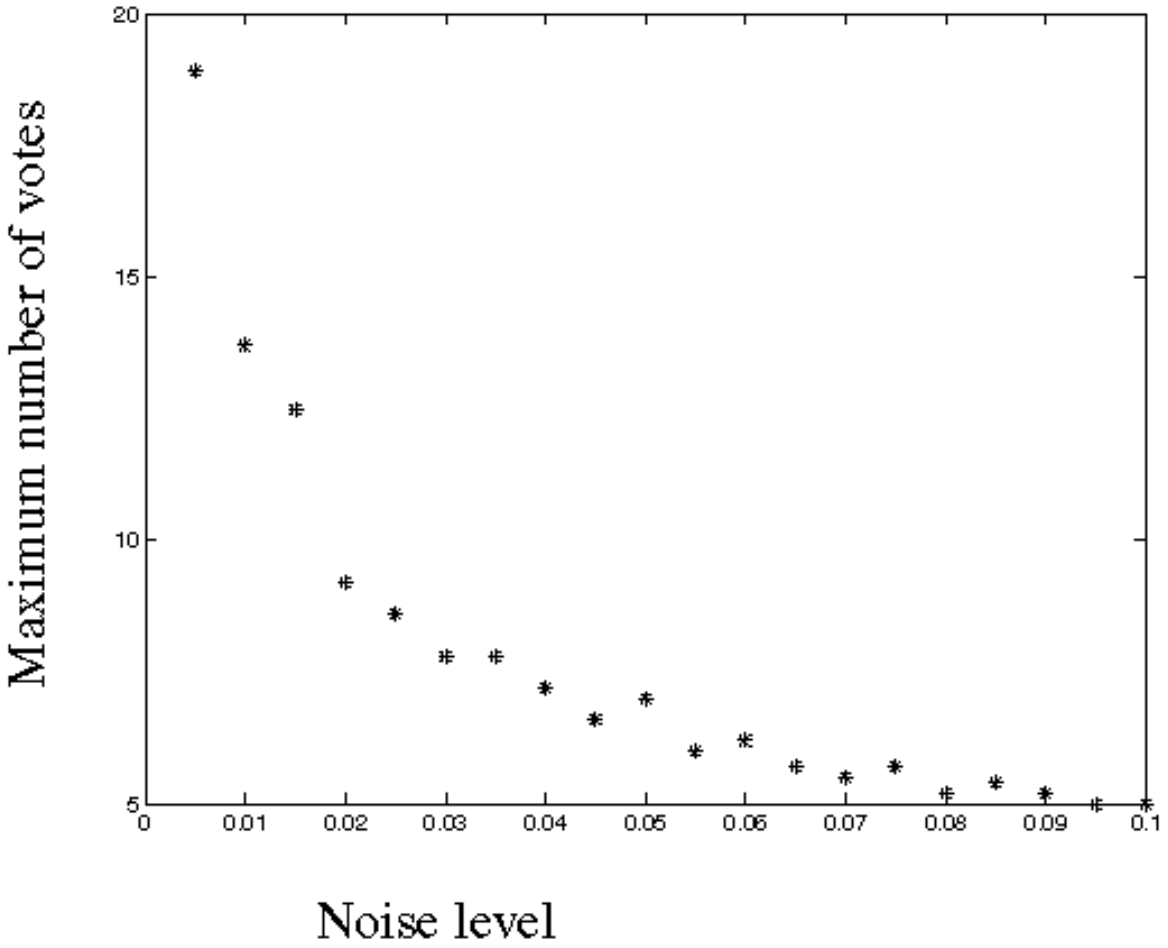


votes

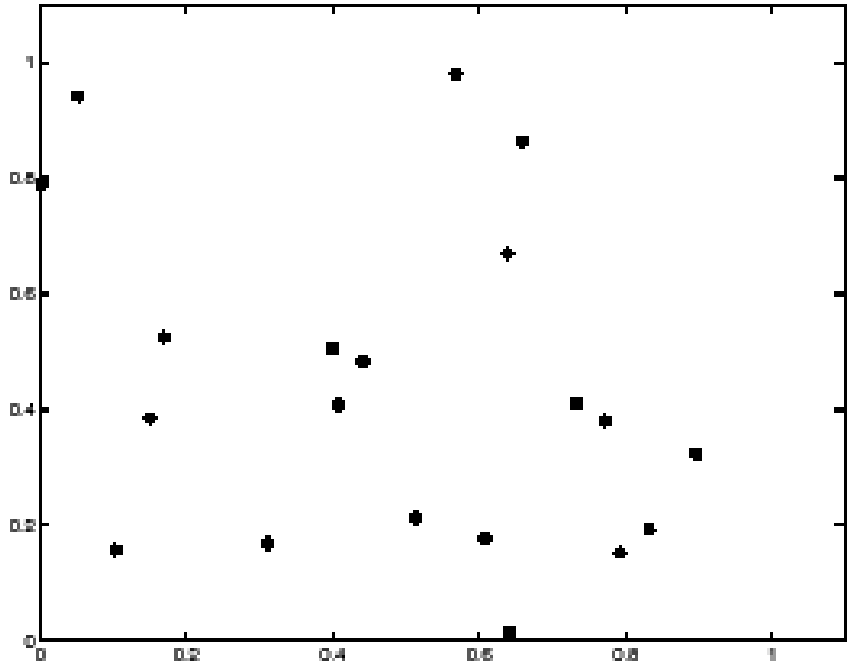
Peak gets fuzzy and hard to locate

Effect of noise

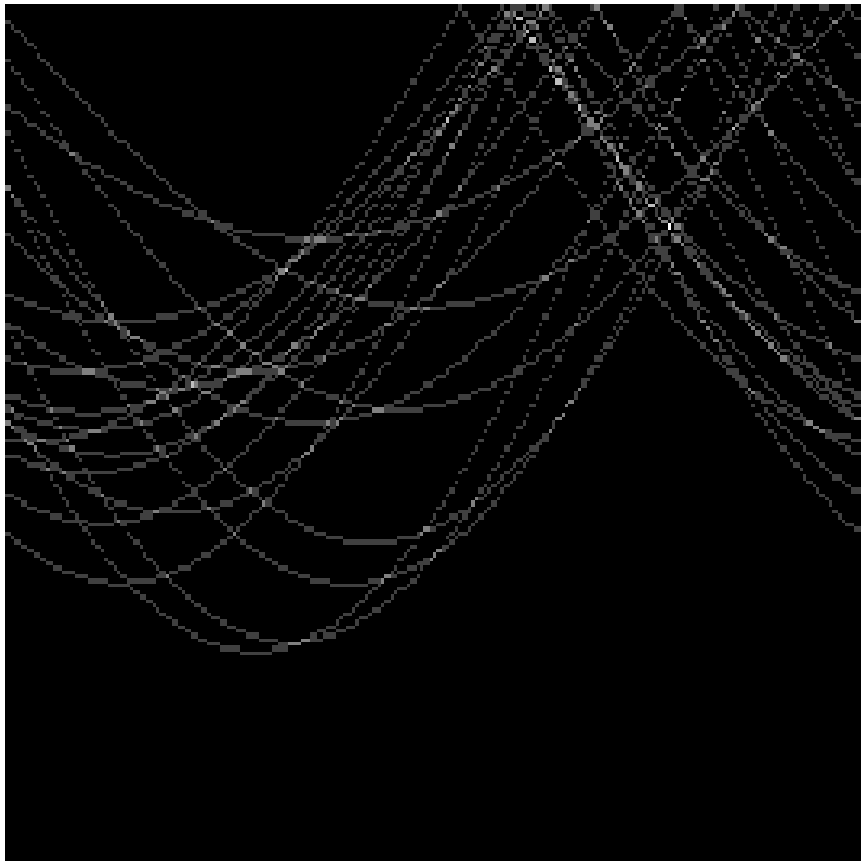
- Number of votes for a line of 20 points with increasing noise:



Random points



features

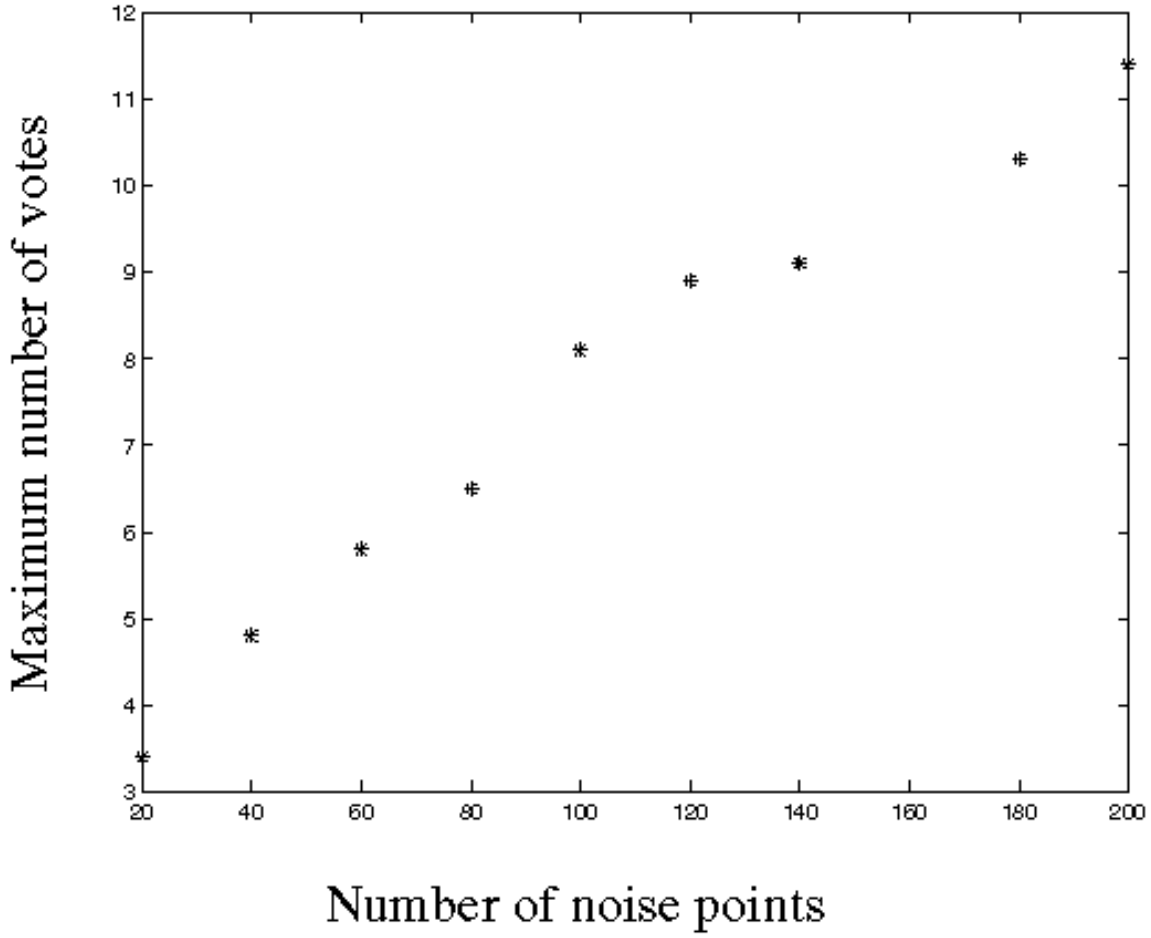


votes

Uniform noise can lead to spurious peaks in the array

Random points

- As the level of uniform noise increases, the maximum number of votes increases too:



Practical details

- Try to get rid of irrelevant features
  - Take edge points with significant gradient magnitude
- Choose a good grid / discretization
  - Too coarse: large votes obtained when too many different lines correspond to a single bucket
  - Too fine: miss lines because some points that are not exactly collinear cast votes for different buckets
- Increment neighboring bins (smoothing in accumulator array)
- Who belongs to which line?
  - Tag the votes

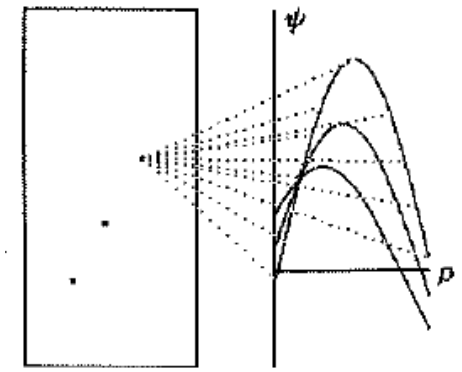
Hough transform: Pros

- 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 deal with non-locality and occlusion
- Can detect multiple instances of a model in a single pass

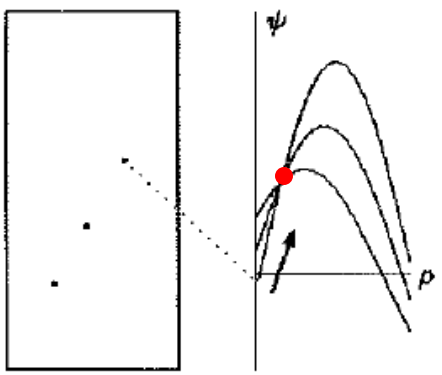
Hough transform: Cons

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

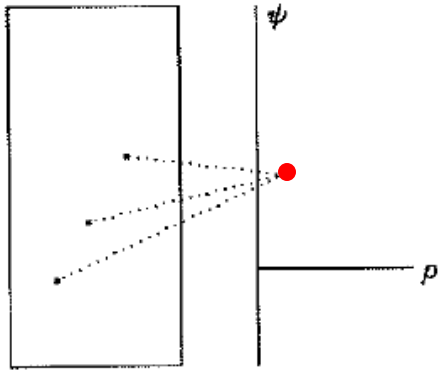
Simplify Accumulator Space: Backtransform



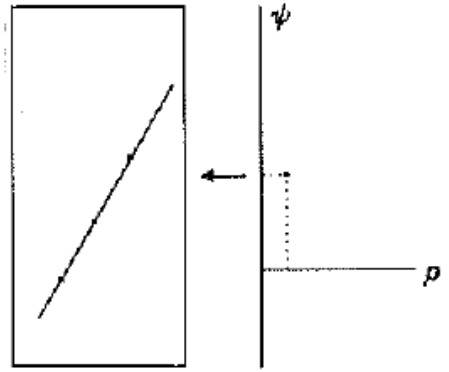
Mapping: Point to curve



Backmapping: Trace curve, select maximum peak.



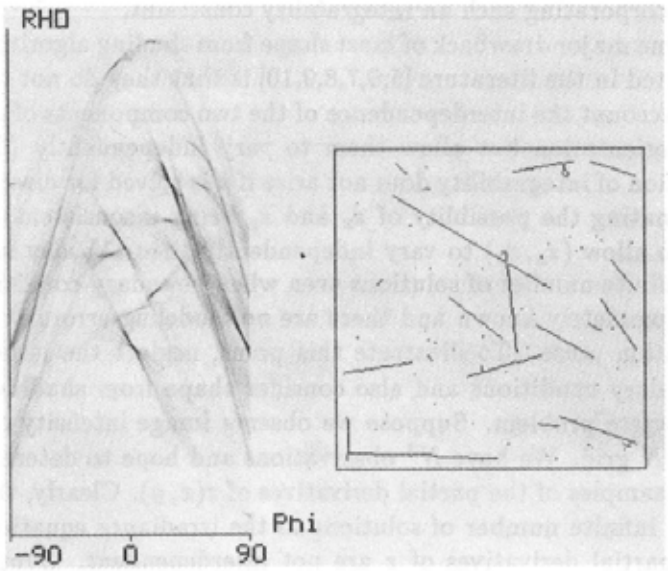
Backmapping: Keep only location of maximum peak per point.

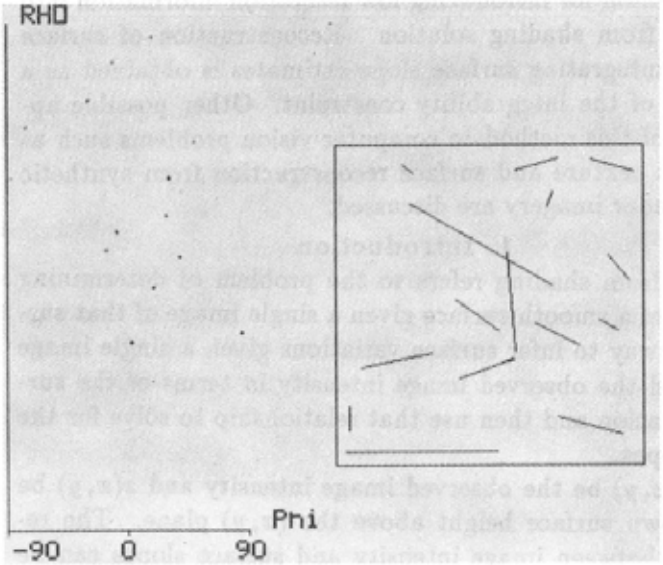


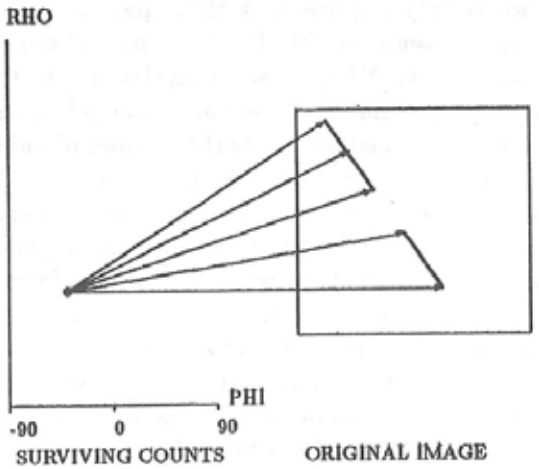
Result: Points on line map to one parameter cell.

See: Gerig et al., ICCV'87

Simplify Accumulator Space: Backtransform







a | b
c

fig. 5: Detection of straight lines

a) original image (right) and corresponding accumulator space (left)

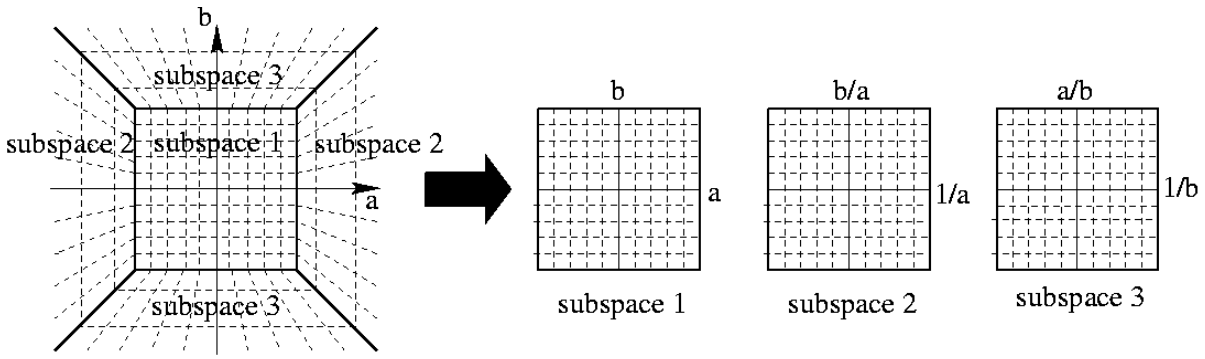
b) surviving counts after backtransform and reconstructed lines (minimum length 10 pixels)

c) example of link between surviving evident count and corresponding line-points

See: Gerig et al., ICCV'87

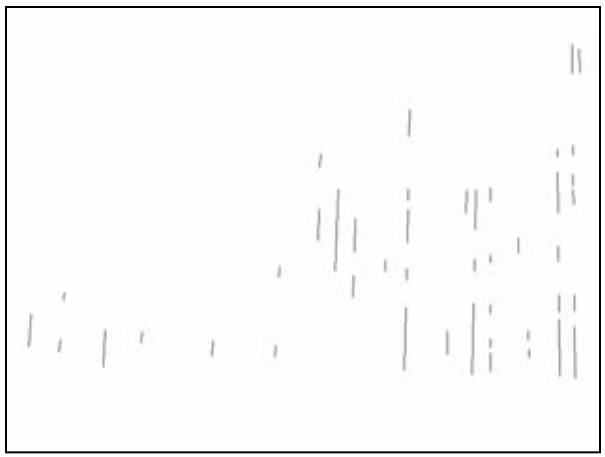
Extension: Cascaded Hough transform

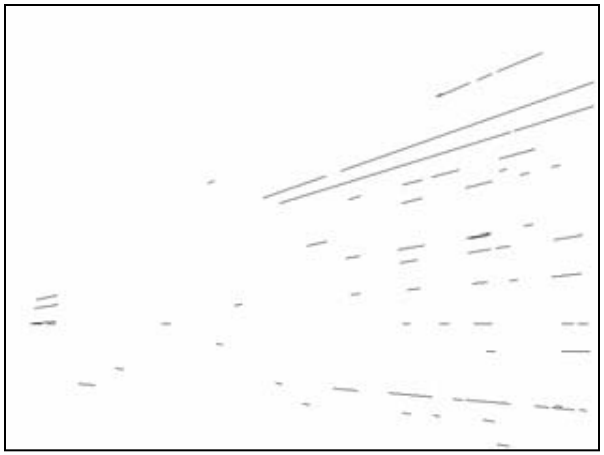
- Let's go back to the original (m,b) parametrization
- A line in the image maps to a pencil of lines in the Hough space
- What do we get with parallel lines or a pencil of lines?
  - Collinear peaks in the Hough space!
- So we can apply a Hough transform to the output of the first Hough transform to find vanishing points
- Issue: dealing with unbounded parameter space

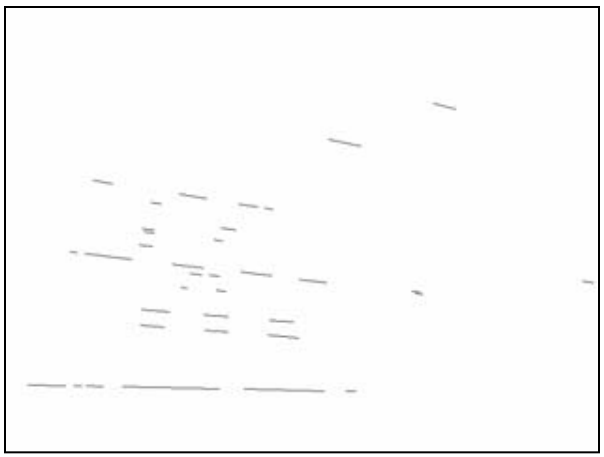


T. Tuytelaars, M. Proesmans, L. Van Gool ***"The cascaded Hough transform,"*** ***ICIP, vol. II, pp. 736-739, 1997.***

Cascaded Hough transform



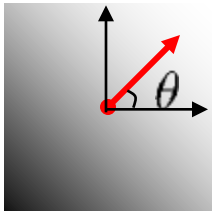




T. Tuytelaars, M. Proesmans, L. Van Gool ***"The cascaded Hough transform,"*** ***ICIP, vol. II, pp. 736-739, 1997.***

Extension: Incorporating image gradients

- Recall: when we detect an edge point, we also know its gradient direction
- But this means that the line is uniquely determined!



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

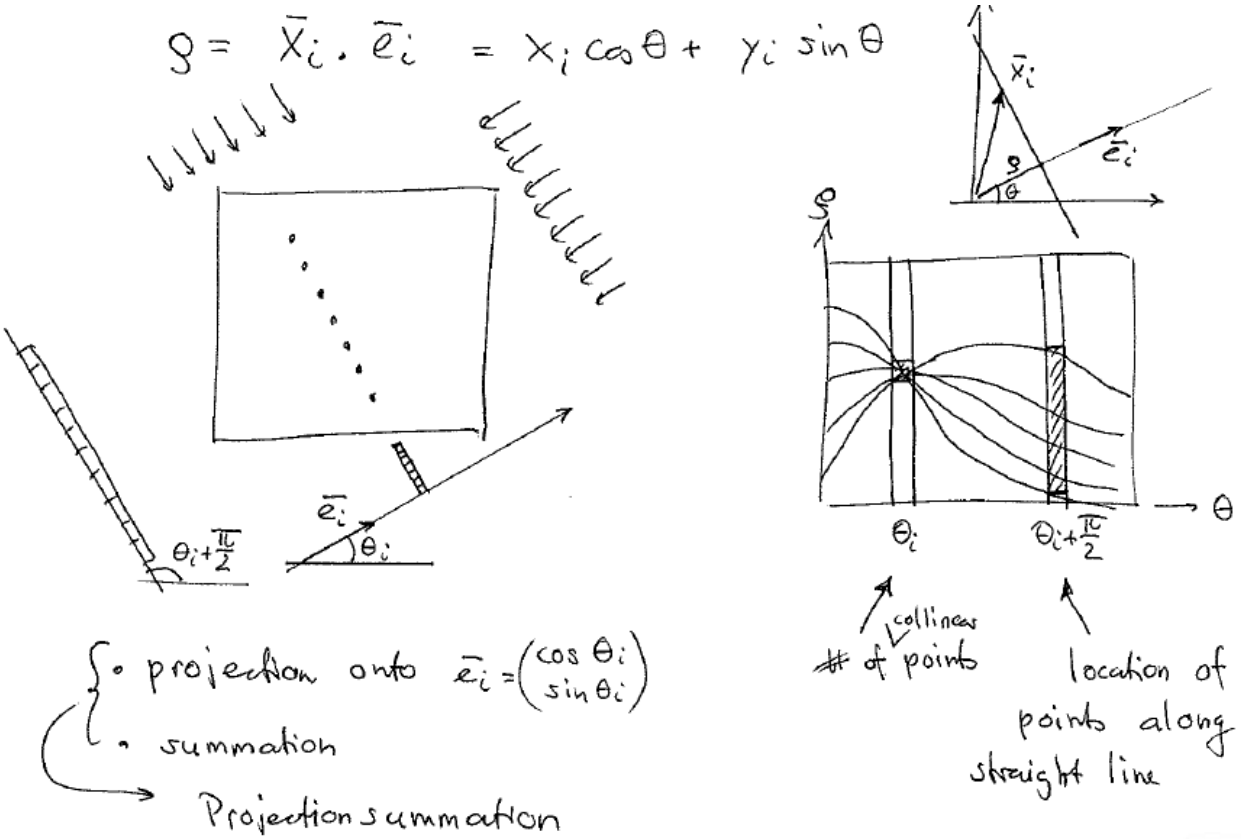
$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- Modified Hough transform:

For each edge point (x,y)
    θ = gradient orientation at (x,y)
    ρ = x cos θ + y sin θ
    H(θ, ρ) = H(θ, ρ) + 1
end

Alternative View of HT: Radon Transform

For details please see document **HT-notes-GG-I.pdf**



Excellent online demonstration:
http://bigwww.epfl.ch/demo/jtomography/demo.html