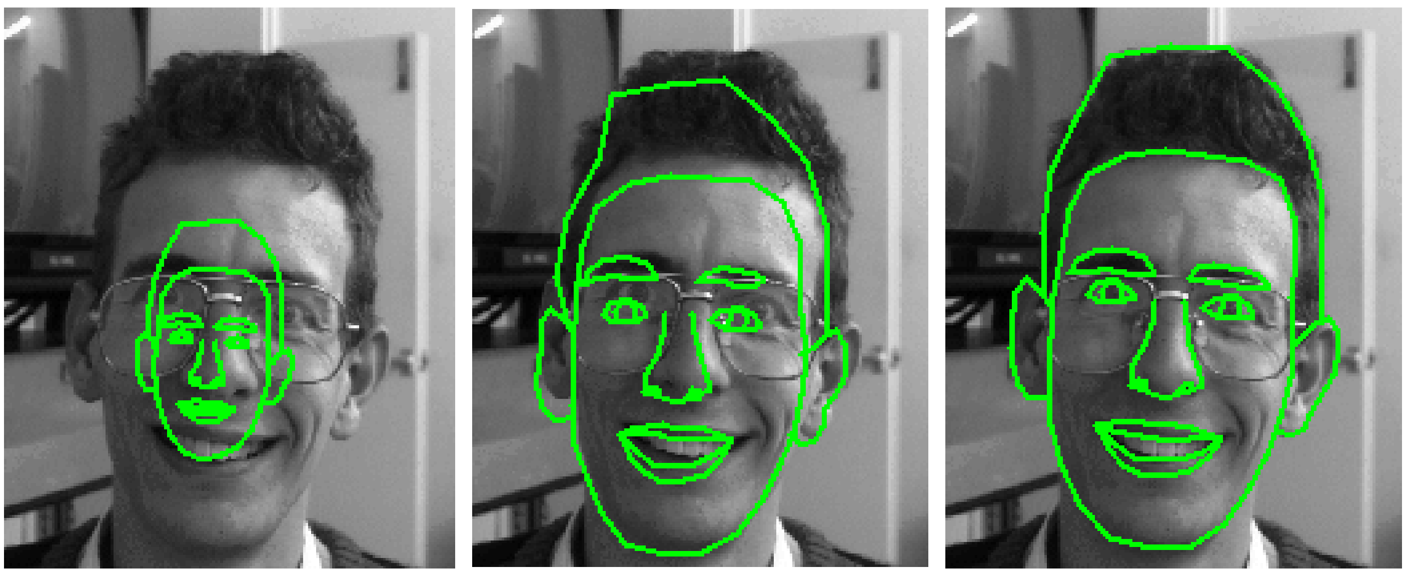
CSE 586, Spring 2015

Shape Fitting

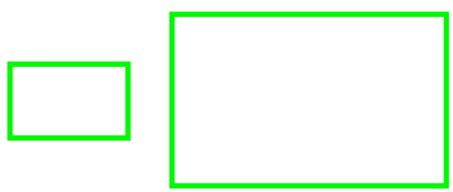
Motivation

Want to align a shape (points; contours) to observed data in an image.

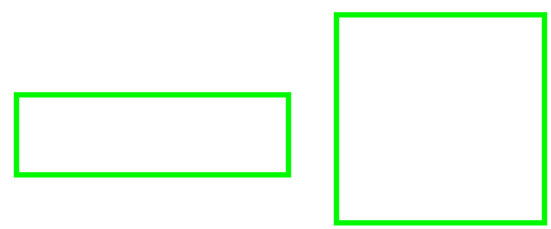


Shape

- What is shape?
  - *Geometric information that remains when location, scale and rotational effects removed* (Kendall)



Same Shape

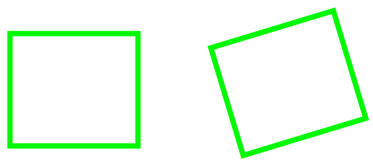
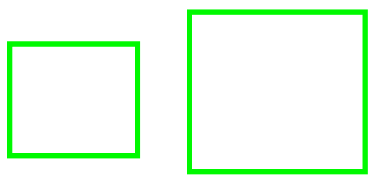
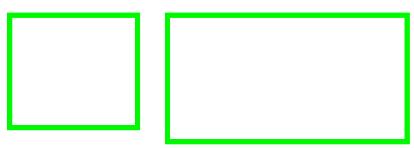
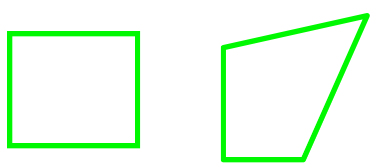


Different Shape

Shape

- More generally
  - *Shape is the geometric information invariant to a particular class of transformations*
- Transformations:
  - Euclidean (translation + rotation)
  - Similarity (translation+rotation+scaling)
  - Affine or projective (deformation due to viewpoint)
  - Articulated motions (e.g. human limbs)
  - General Deformation (mesh models; medical shapes)

What transformation to use?

Shapes	Euclidean	Similarity	Affine	Projective
	✓			
		✓		
			✓	
				✓

Shape Models

- represent the shape using a set of points

Point-based Shape Models

- Landmarks: points that can be reliably located.

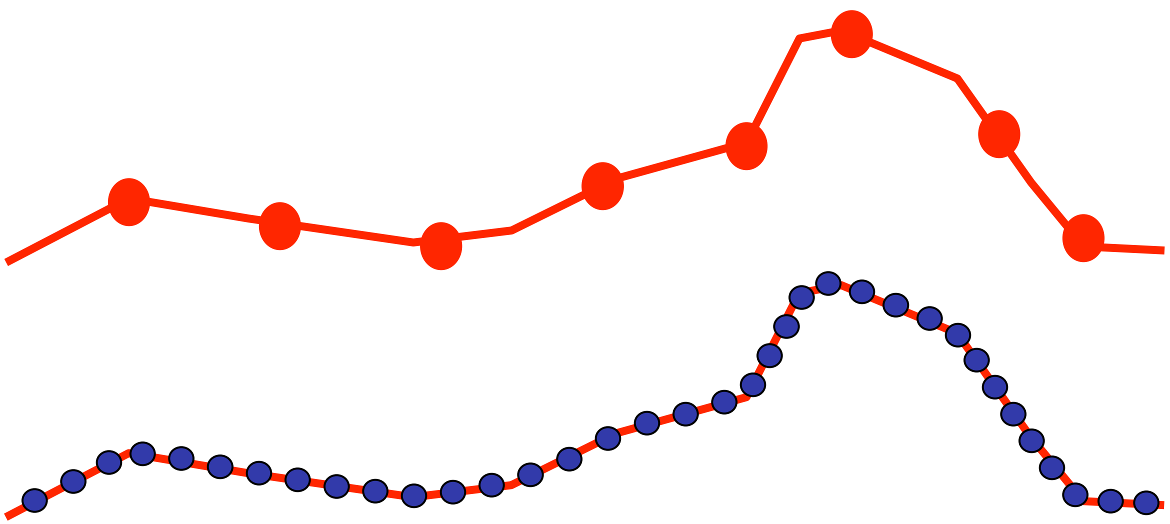
well defined corners, “T” junctions; easily located biological landmarks

e.g. point 17 is right corner of the mouth

Point-based Shape Models

- Sampled points along a curve

  Individual points no longer reliably located, but if curve is sampled densely enough that is not a problem.



Procrustes Analysis

- Bring two or more shapes into alignment using some transformation (usually Euclidean rot+trans or similarity rot+trans+scale)

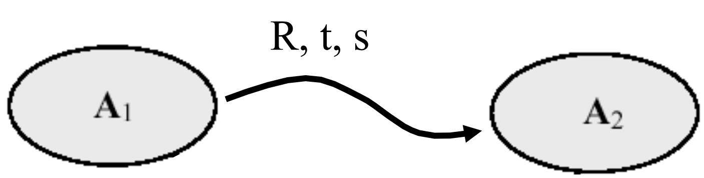
- Estimate a statistical shape distribution p(x) describing intrinsic variation among the shapes

- We will only talk about shape alignment today.

Robert Collins
CSE586

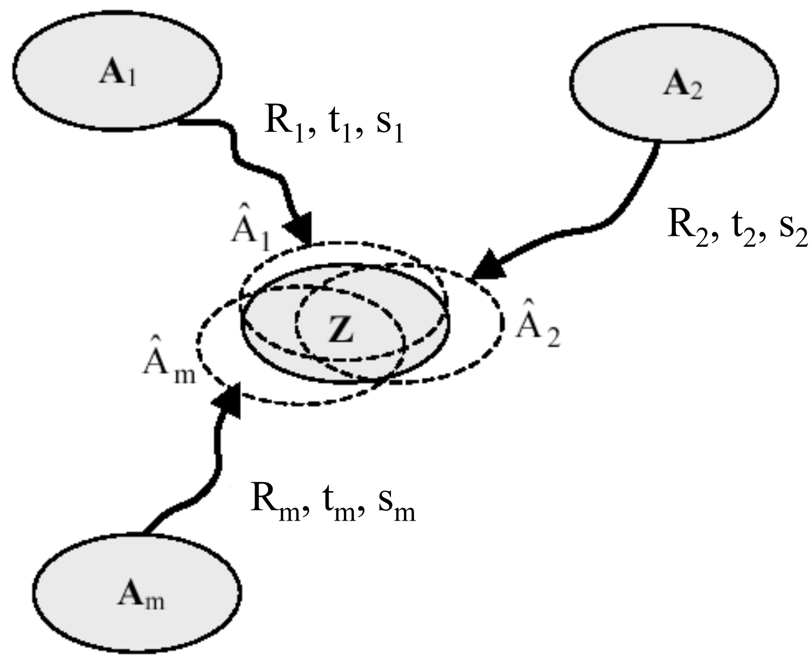
Procrustes Analysis

Procrustes Analysis



Align one shape
with another
(not symmetric)

General Procrustes Analysis



Align a set of shapes with respect
to some unknown "mean" shape
(independent of ordering of shapes)

Why called “Procrustes”?

http://www.procrustes.nl/gif/illustr.gif

Aligning Two Shapes

- Procrustes analysis:
  - Find transformation which minimises

$$|\mathbf{x}_1 - T(\mathbf{x}_2)|^2$$

  - T is a particular type of transformation that you want "shape" to be invariant to

Go to board. Insert my handwritten notes here.

Aligning Two Shapes

- Procrustes analysis:
  - Find transformation which minimises

$$|\mathbf{x}_1 - T(\mathbf{x}_2)|^2$$

  - If T is a similarity transformation, the resulting shapes have
    - Identical center of mass
    - approximately the same scale and orientation

Steps in Similarity Alignment

Given a set of K points: Configuration

Translation normalization: Centered Configuration
(center of mass at origin)

Scale normalization: Pre-shape
(divide by Sqrt of SSQ centered coordinates)

Rotation normalization: Shape
(rotate to alignment with ref shape)

Aligning Pre-Shapes by Rotation

Sketch:

A, B are two Kx2 preshapes, R is unknown rotation

Want to minimize ||(AR - B)^2|| subject to $R^TR=I$

$$\overbrace{\underset{Kx2}{\begin{pmatrix} x1 & y1 \\ x2 & y2 \\ \vdots & \\ xk & yk \end{pmatrix}} \underset{2x2}{\begin{pmatrix} R \end{pmatrix}} - \underset{Kx2}{\begin{pmatrix} p1 & q1 \\ p2 & q2 \\ \vdots & \\ pk & qk \end{pmatrix}}}$$

Minimize sum of squared errors

Aligning Pre-Shapes by Rotation

Sketch:

A, B are two Kx2 preshapes, R is unknown rotation

Want to minimize $||(AR - B)^2||$ subject to $R^TR=I$

After some manipulation, we find that

$$(A^TB)(A^TB)^T = R\,(A^TB)^T(A^TB)R^T$$

Note, both sides are symmetric and have same eigenvalues. This is a job for SVD!

Aligning Pre-Shapes by Rotation

Sketch continued:

$$\underbrace{(A^TB)(A^TB)^T}_{\text{SVD}} = R\,\underbrace{(A^TB)^T(A^TB)}_{\text{SVD}}R^T$$

$$V\ D\ V^T = R\ W\ D\ W^T\ R^T$$

So... $V = R\ W$

and therefore $R = V\ W^T$

Aligning a Set of Shapes

- Generalised Procrustes Analysis
  - Find the transformations T_i which minimise

$$\sum |\mathbf{m} - T_i(\mathbf{x}_i)|^2$$

  - Where $\mathbf{m} = \frac{1}{n}\sum T_i(\mathbf{x}_i)$
  - Under the constraint that $|\mathbf{m}| = 1$

think about why
this is needed...

Aligning a Set of Shapes

- Generalised Procrustes Analysis
  - Find the transformations T_i which minimise

$$\sum |\mathbf{m} - T_i(\mathbf{x}_i)|^2$$

  - Where $\mathbf{m} = \frac{1}{n}\sum T_i(\mathbf{x}_i)$
  - Under the constraint that $|\mathbf{m}| = 1$

Otherwise, you can shrink all shapes to an infinitesimal ball to minimize distance error!

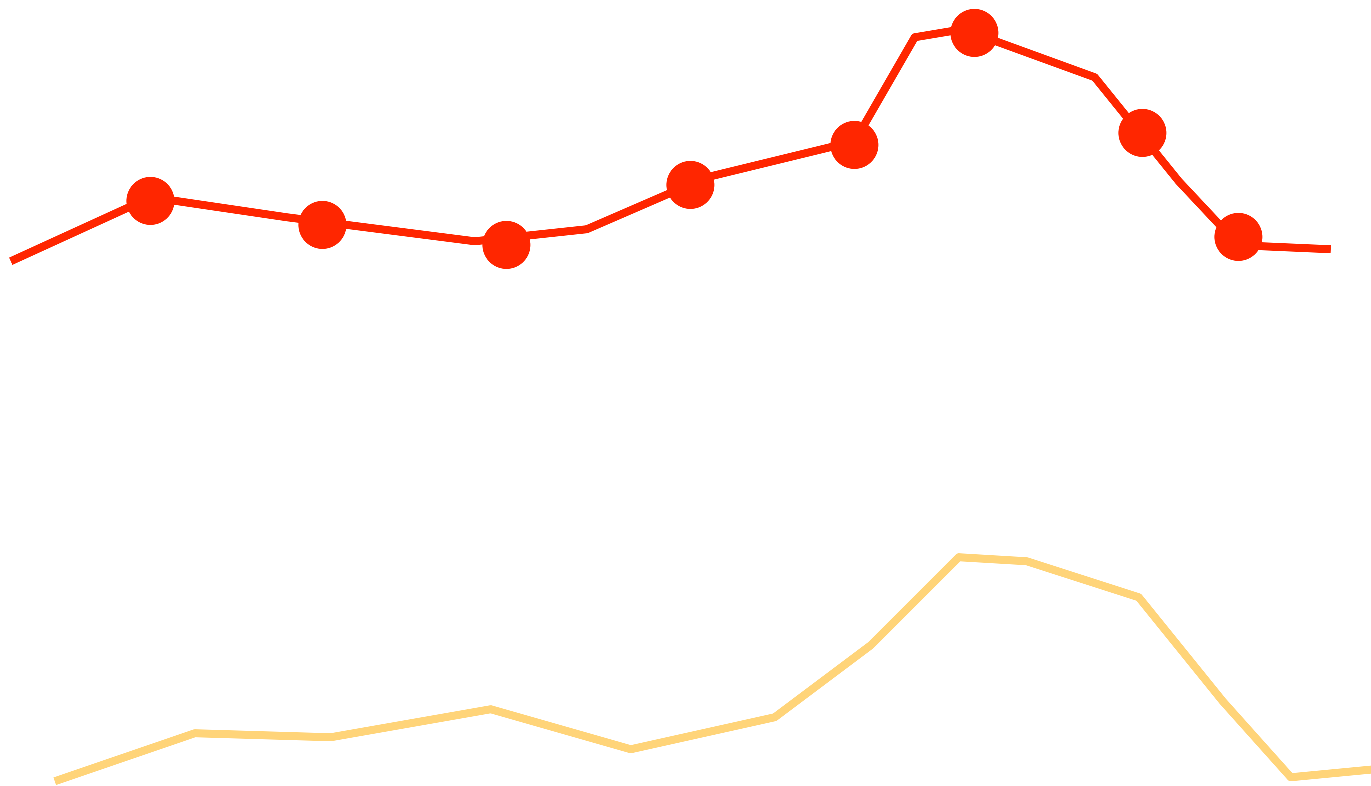
Aligning Shapes : Algorithm

- Normalise all so center of mass is at origin, and size=1
- Let $\mathbf{m} = \mathbf{x}_1$
- Align each shape with **m** (via a rotation)
- Re-calculate $\mathbf{m} = \frac{1}{n} \sum T_i(\mathbf{x}_i)$
- Normalise **m** to default size, orientation
- Repeat until convergence

But what if we don't know the point correspondences?

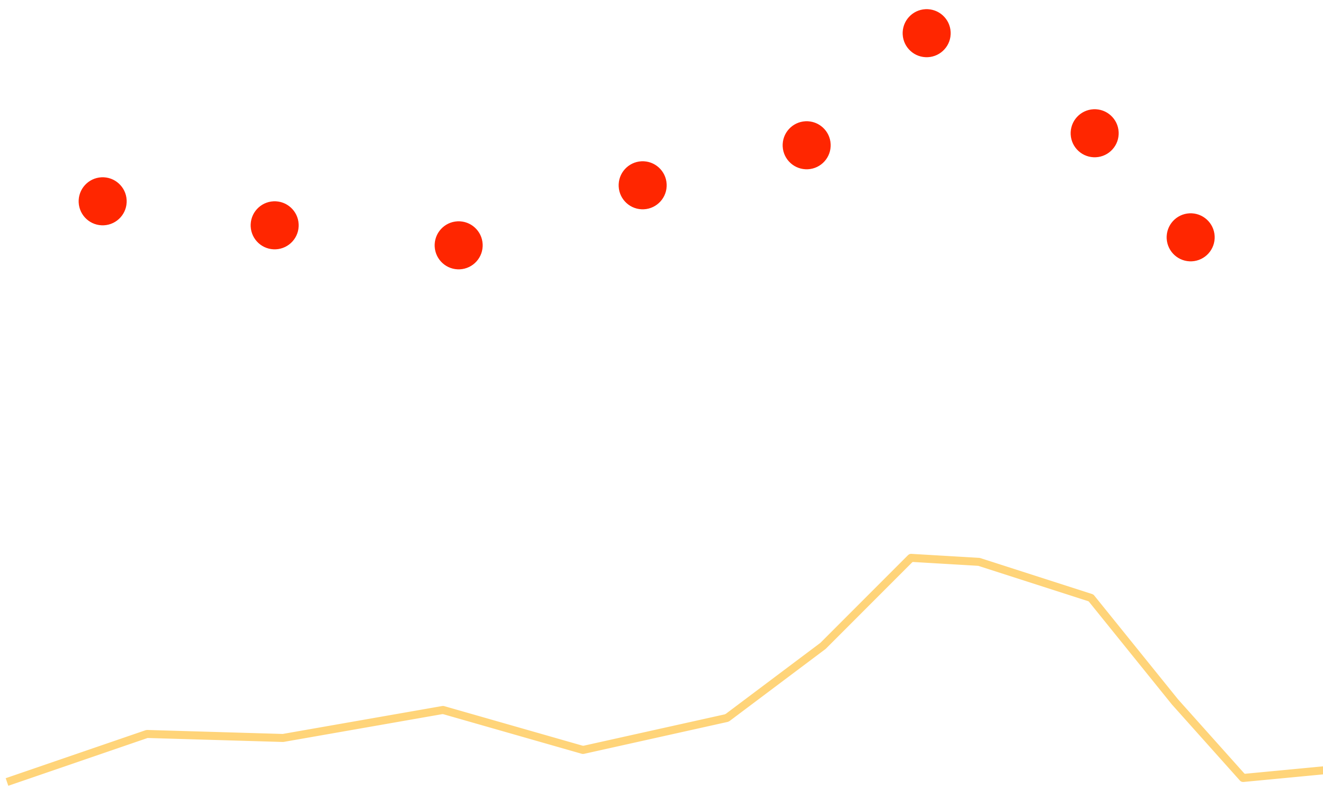
need to do simultaneous localization and matching

Iterative Closest Point (ICP)



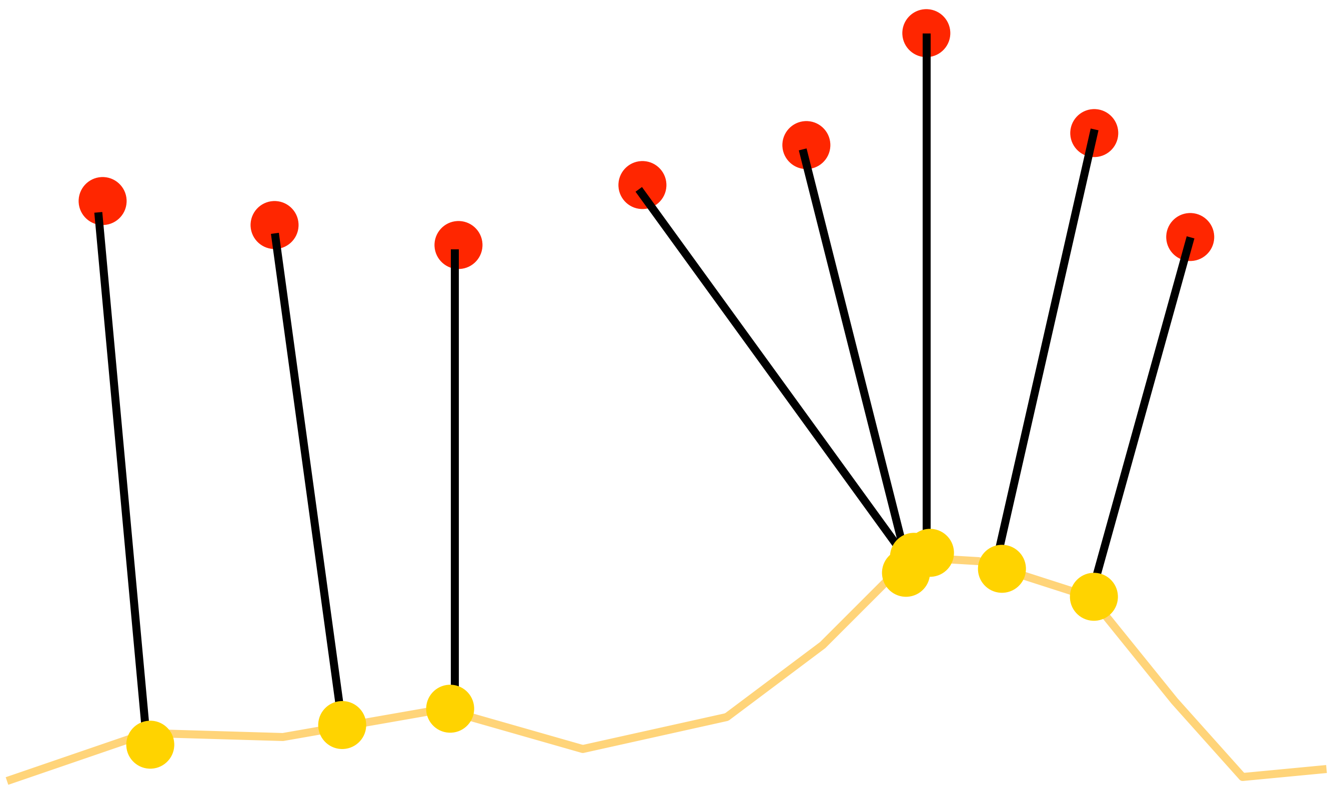
Sample points on source curve/surface

Iterative Closest Point (ICP)



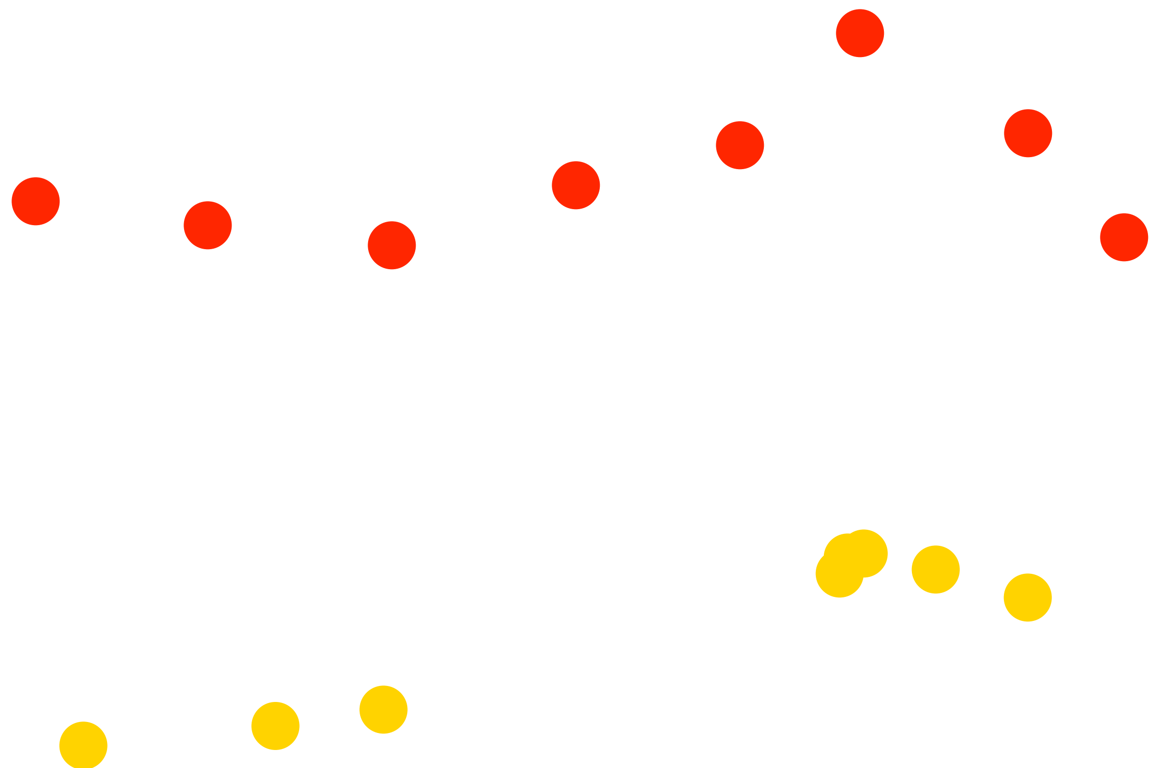
Points become proxy for surface

Iterative Closest Point (ICP)



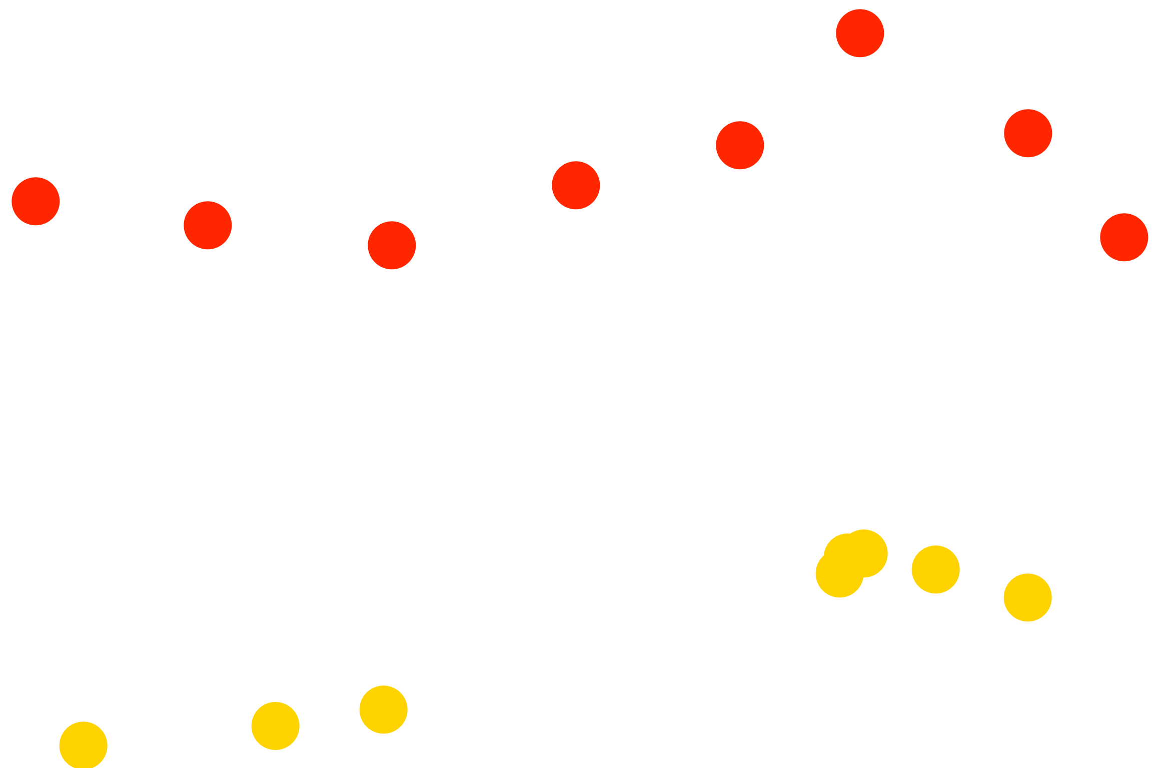
Find closest points on target surface

Iterative Closest Point (ICP)



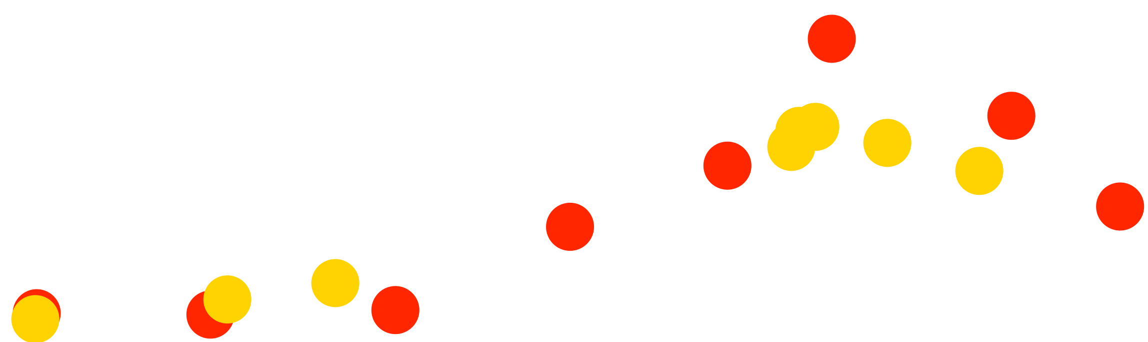
Those points become proxy for target surface

Iterative Closest Point (ICP)



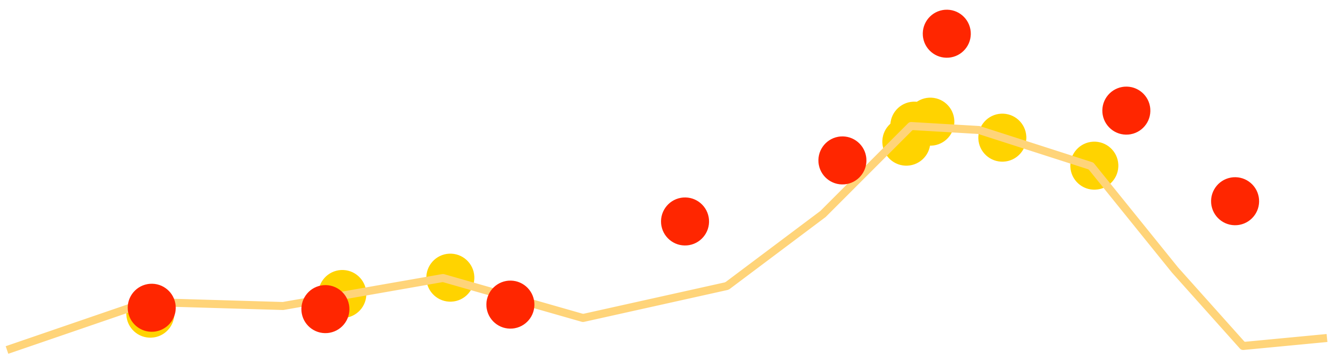
Register point sets (rigid)

Iterative Closest Point (ICP)



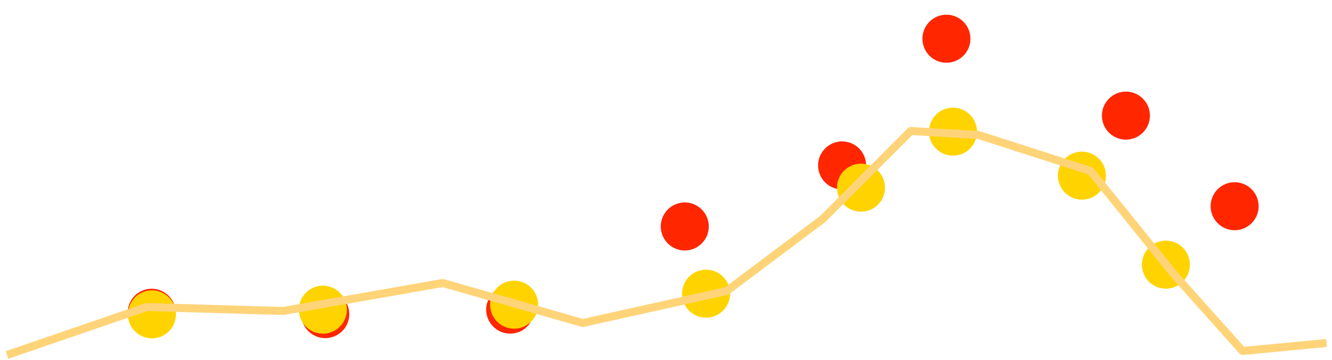
Register point sets (rigid)

Iterative Closest Point (ICP)



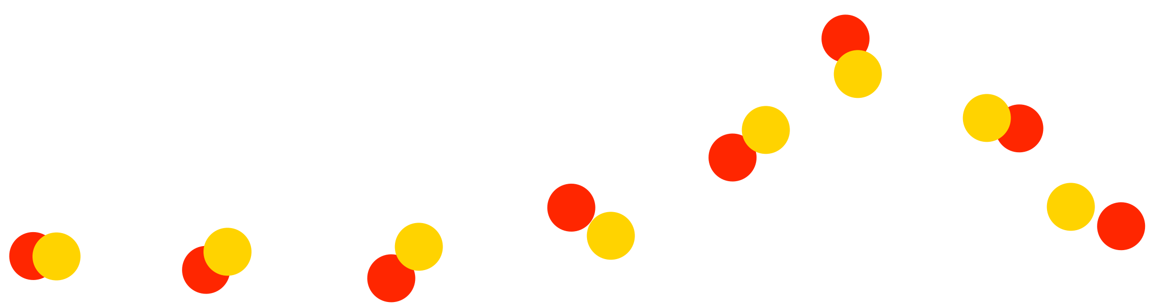
Restore underlying target surface

Iterative Closest Point (ICP)



Find (new) closest points on target surface

Iterative Closest Point (ICP)

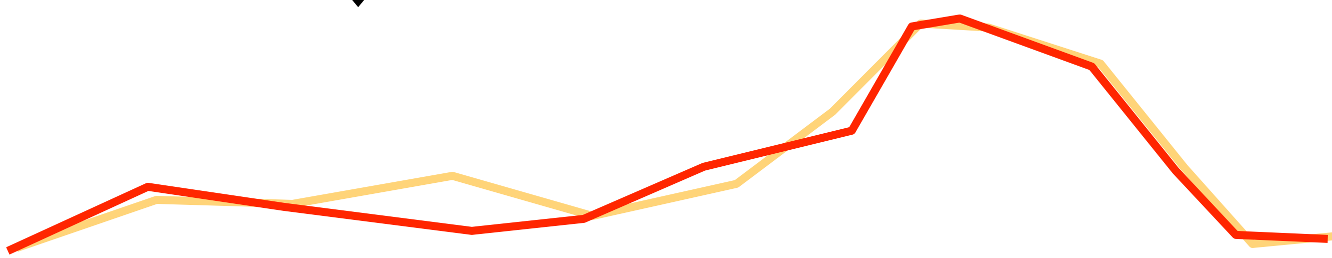


Align the two point sets, and so on…

ICP Algorithm

- Find closest points
- Measure total distance
- Register points

Stop when distance change is small.



Works very well in 3D too!
Guaranteed to converge.
Not guaranteed to find globally best alignment

What if there are multiple copies of the same shape in an image, and you want to detect and align your shape model to each one? Assume there is also other random junk you want to ignore (outliers points).

- Ransac
- Hough transform

We will illustrate both of these in regards to fitting straight lines to points in an image

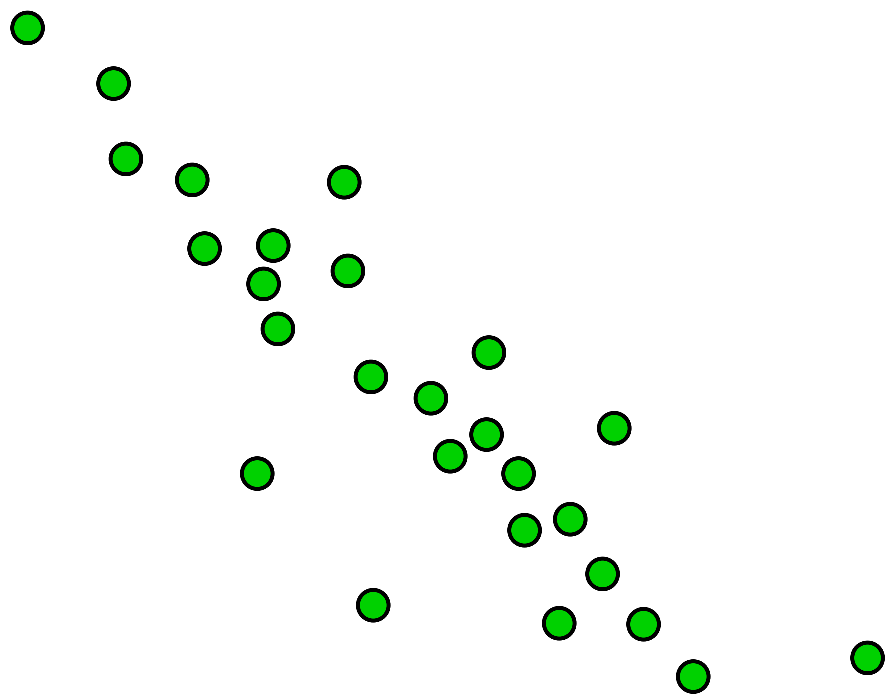
Robust Estimation

- View model fitting as a two-stage process:
  - Classify data points as outliers or inliers
  - Fit model to inliers while ignoring outliers

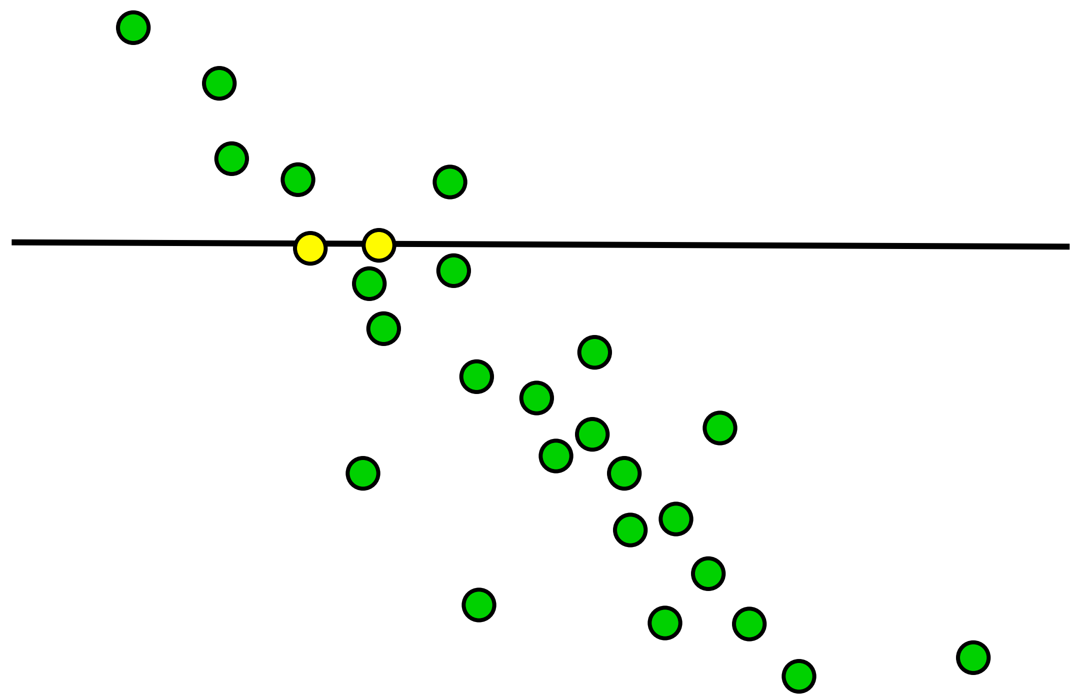
- Example technique: RANSAC
  (RANdom SAmple Consensus)

Ransac Procedure

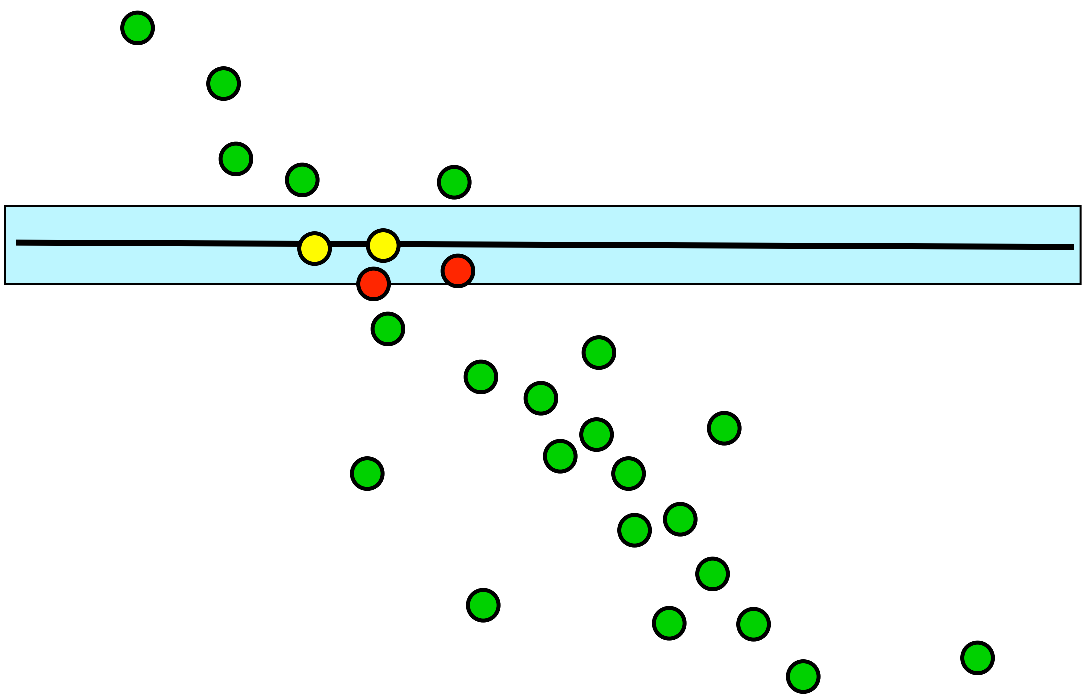
Example: Want to fit a line to a set of points



Ransac Procedure

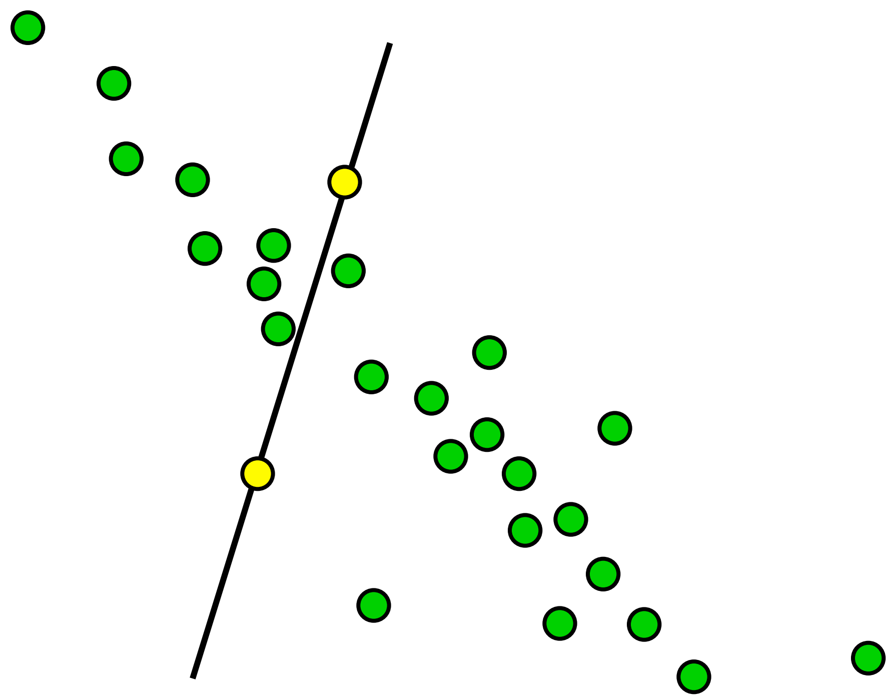


Ransac Procedure

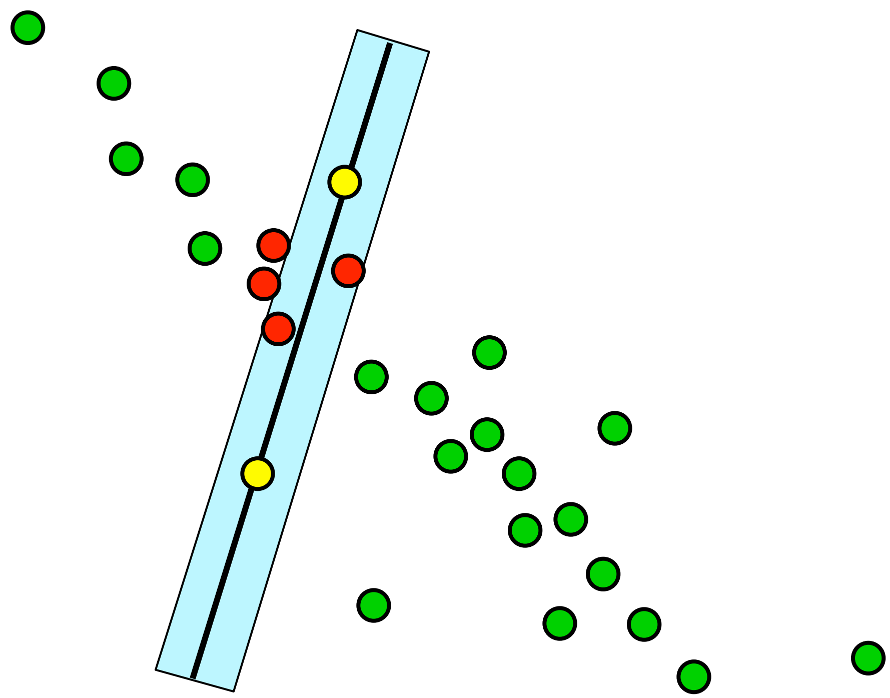


Count = 4

Ransac Procedure



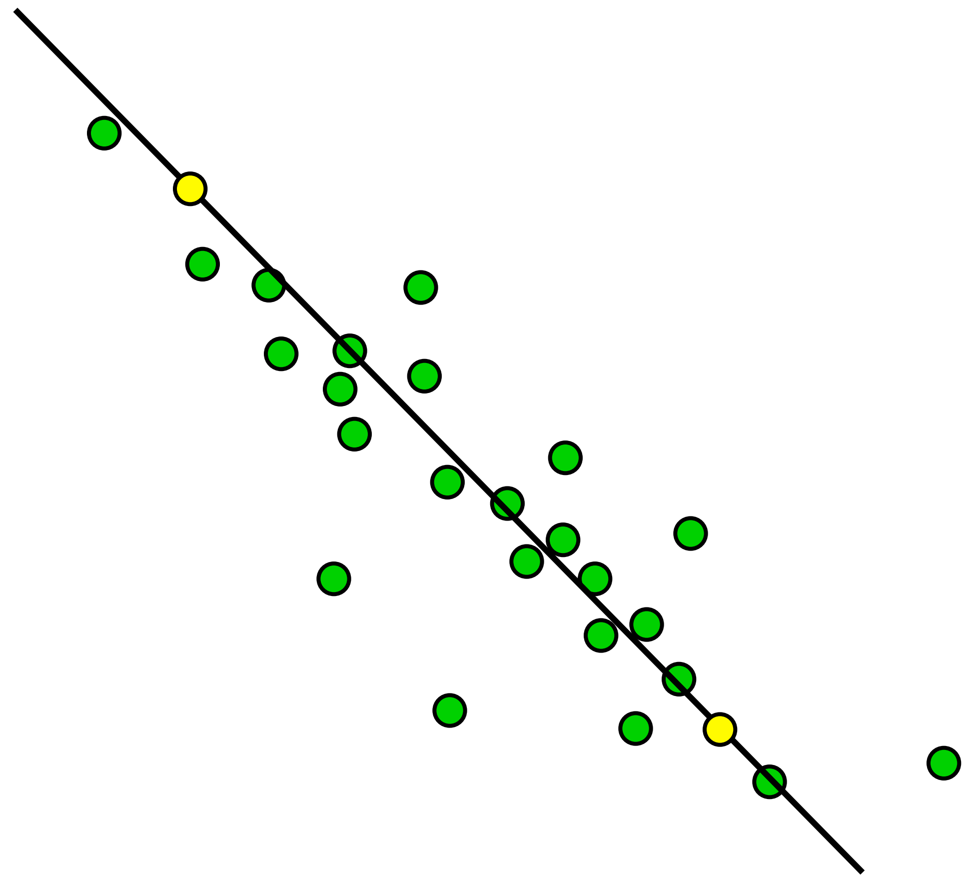
Ransac Procedure



Count = 6

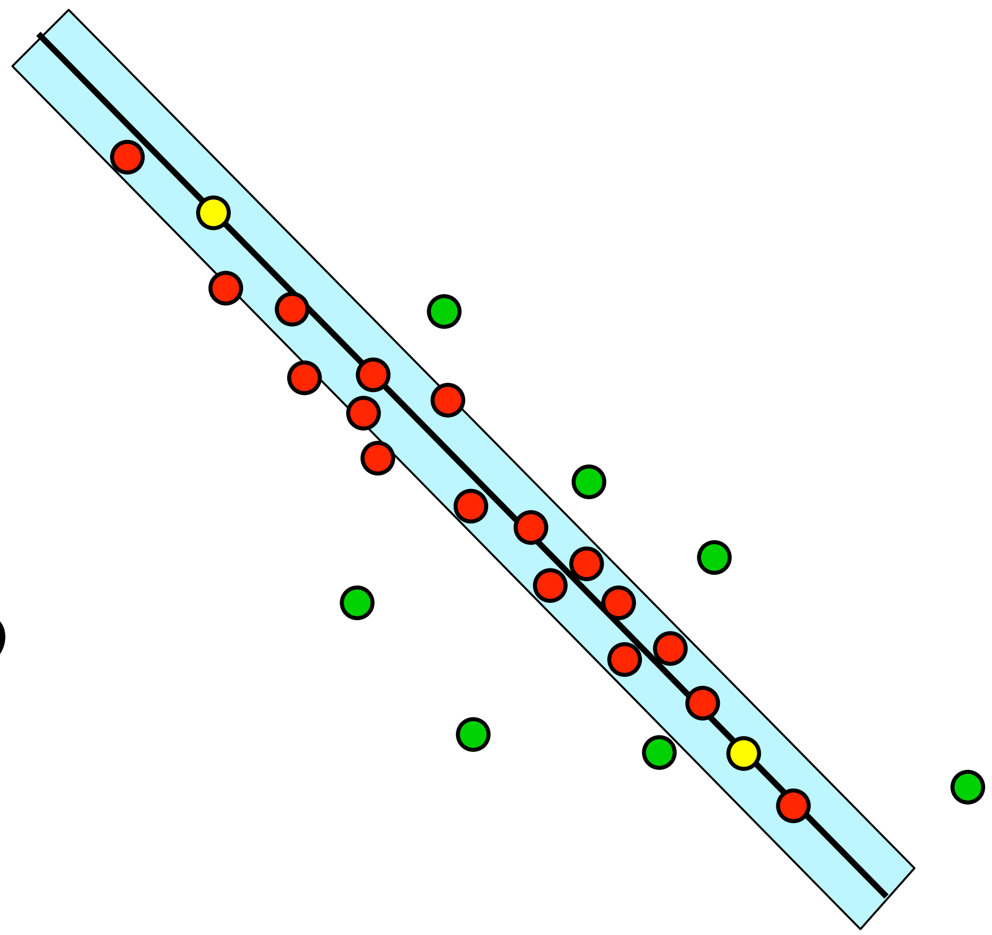
Robert Collins
CSE586

Ransac Procedure



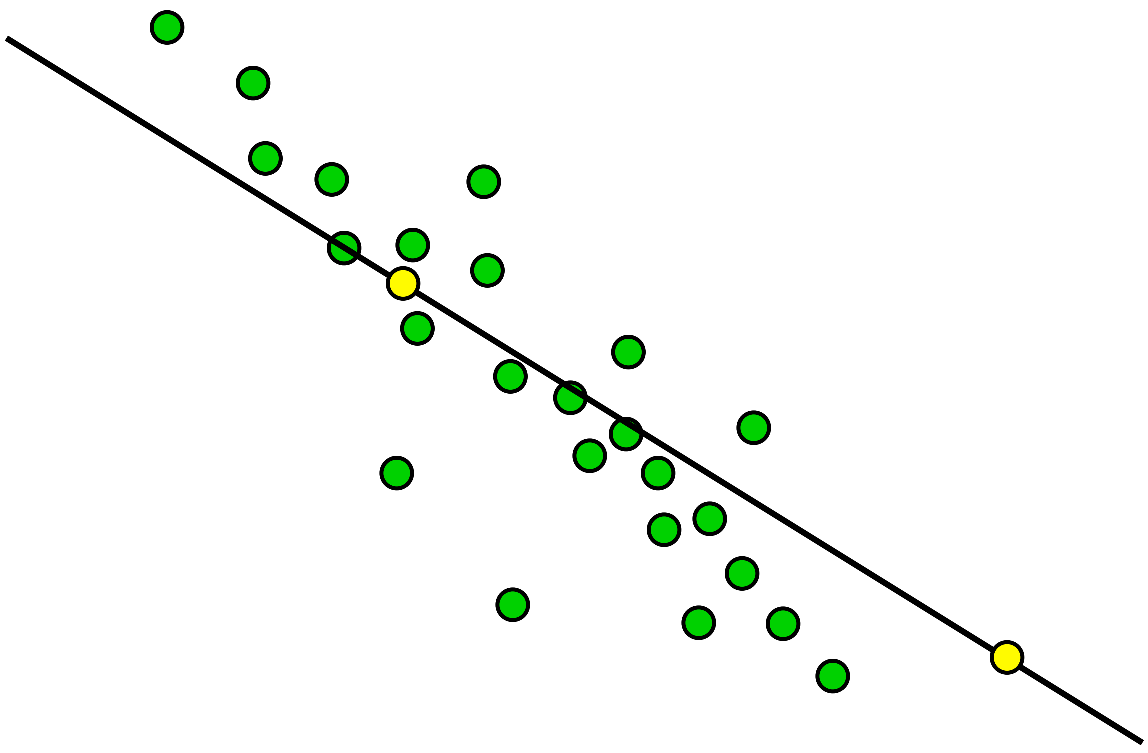
Robert Collins
CSE586

Ransac Procedure

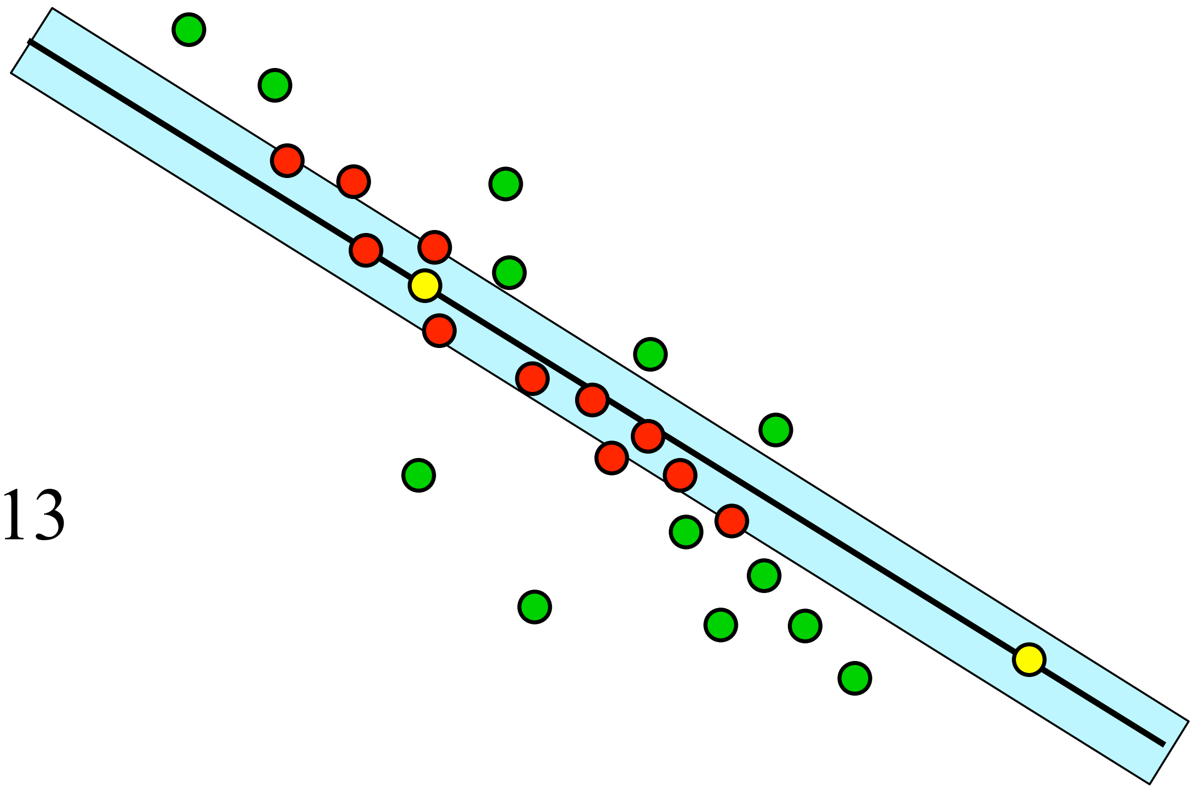


Count = 19

Ransac Procedure



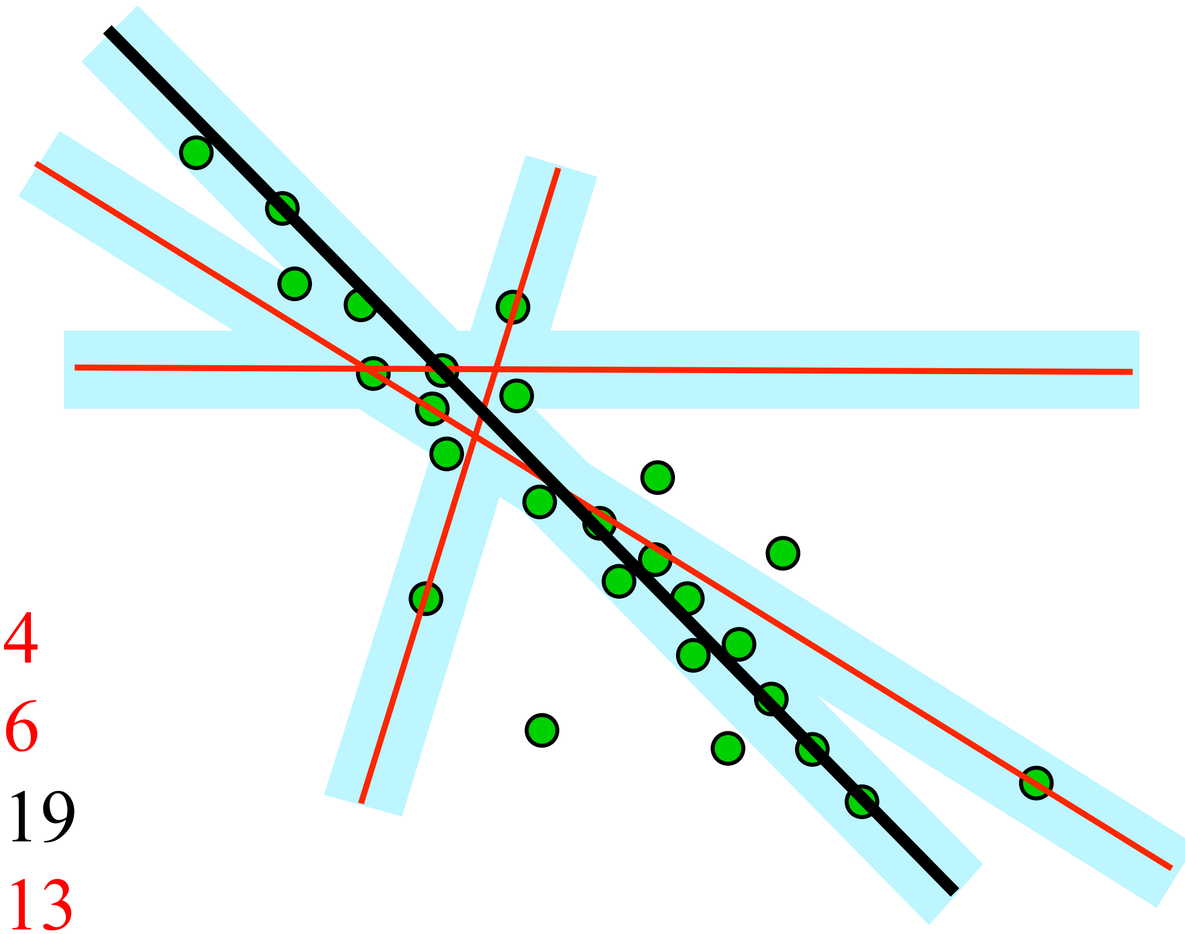
Ransac Procedure



Count = 13

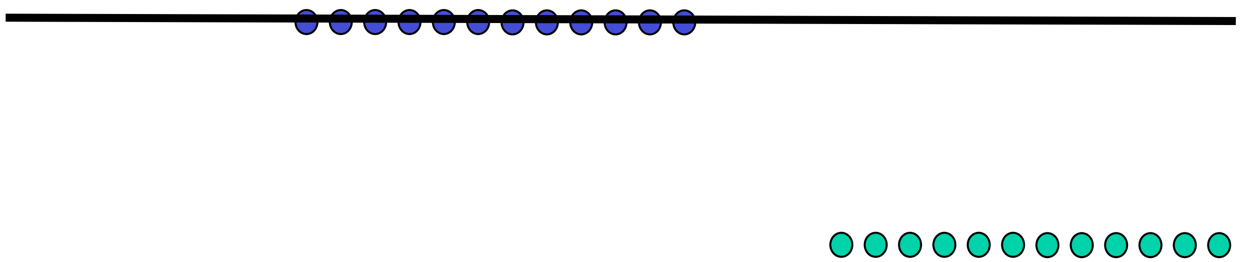
Robert Collins
CSE586

Ransac Procedure



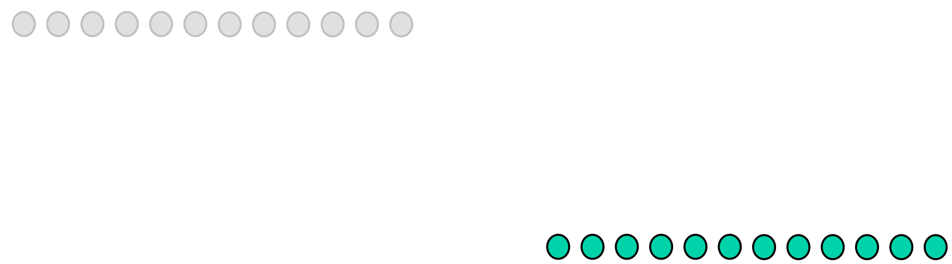
Count = 4
Count = 6
Count = 19
Count = 13

Ransac for fitting multiple shapes



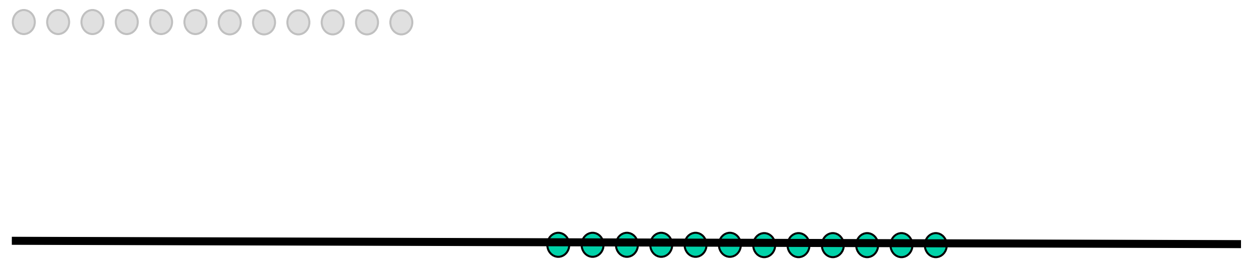
Find best fit line...

Ransac for fitting multiple shapes



Remove those data points

Ransac for fitting multiple shapes



Find best fit to remaining data
and continue

Hough Transform

The Hough transform is a histogram-based voting scheme that addresses three problems:

- how many shapes are there?
- which points belong to which shape?
- what are the parameter values of each shape?

Hough Transform (HT)

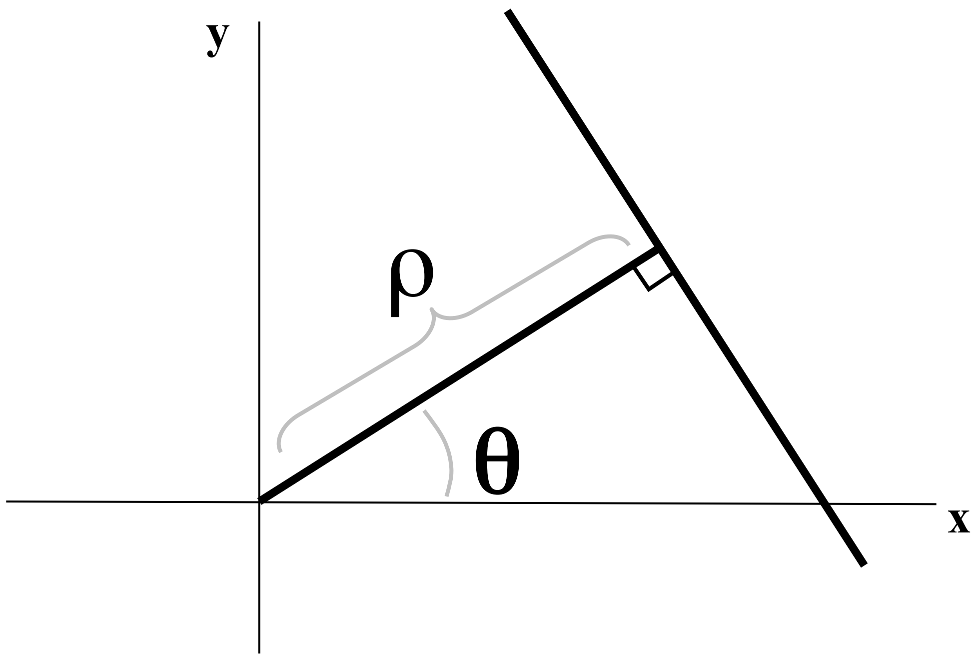
- Basic idea: Change problem from shape fitting to peak finding in parameter space of the shape
  - Each pixel can lie on a family of possible shapes (e.g., for lines, the set of lines through that point)
  - Shapes with more pixels on them have more evidence that they are present in the image
  - Thus every pixel “votes” for a set of shapes and the one(s) with the most votes “win”—i.e., exist

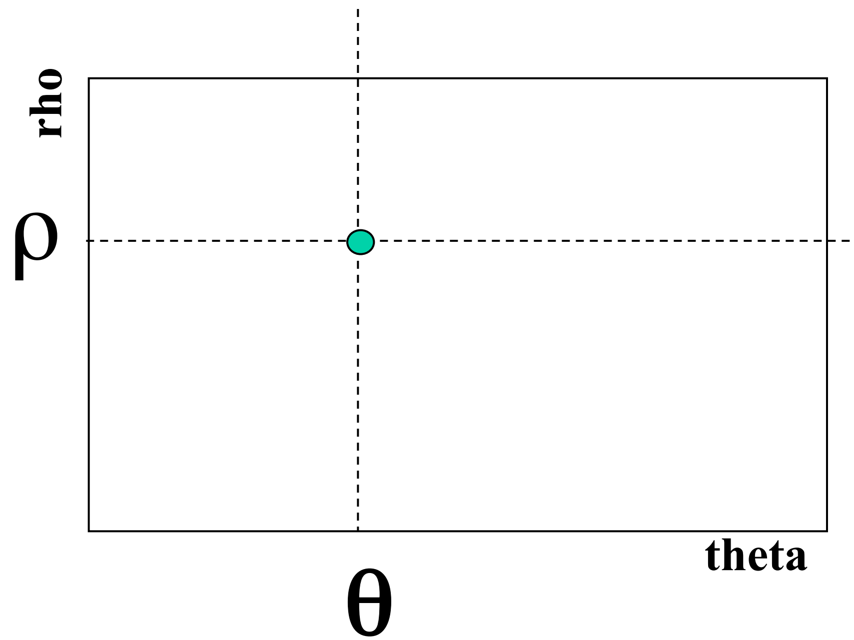
- Original application was detecting lines in time lapse photographs of bubble chamber experiments
  - elementary particles move along straight lines, collide, and create more particles that move along new straight trajectories
  - Hough was the name of the physicist who invented the method

The Hough Transform for Lines

Parameterization: $\cos(\theta)\, x + \sin(\theta)\, y = \rho$

Params are (θ, ρ) => 2 dimensional parameterization





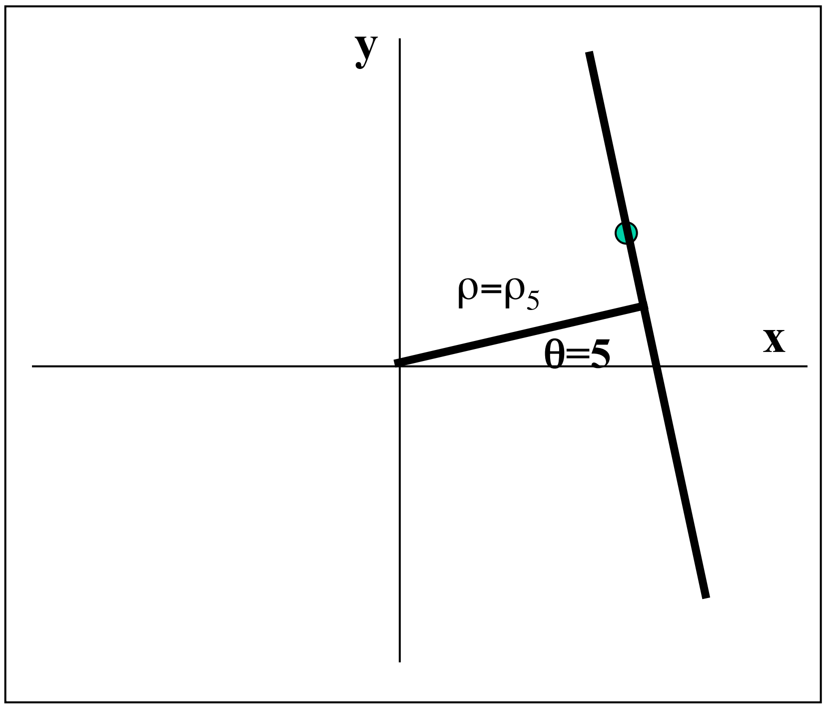
Line in image space

Point in parameter space

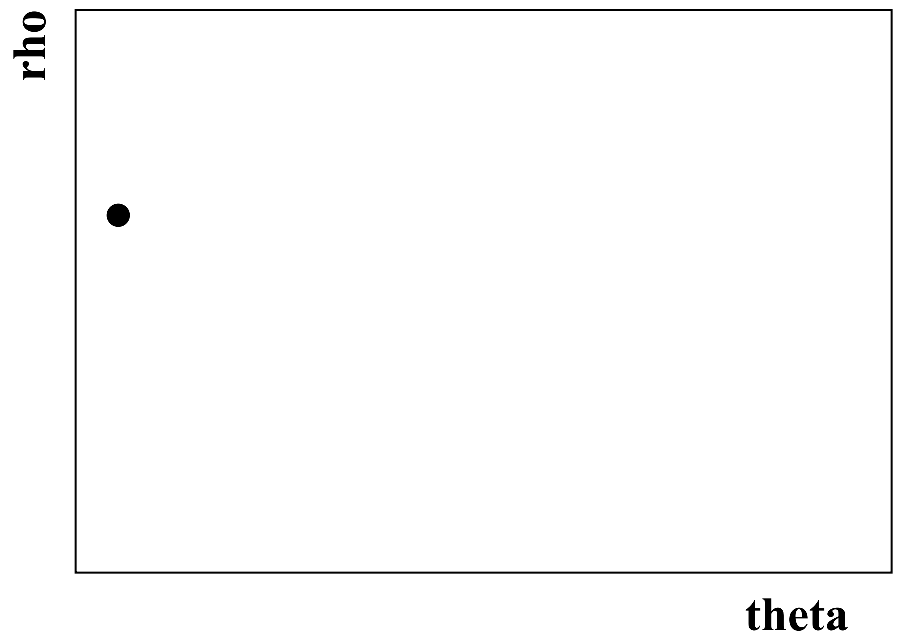
Robert Collins
CSE486, Penn State

Hough Transform for Lines

Image Space



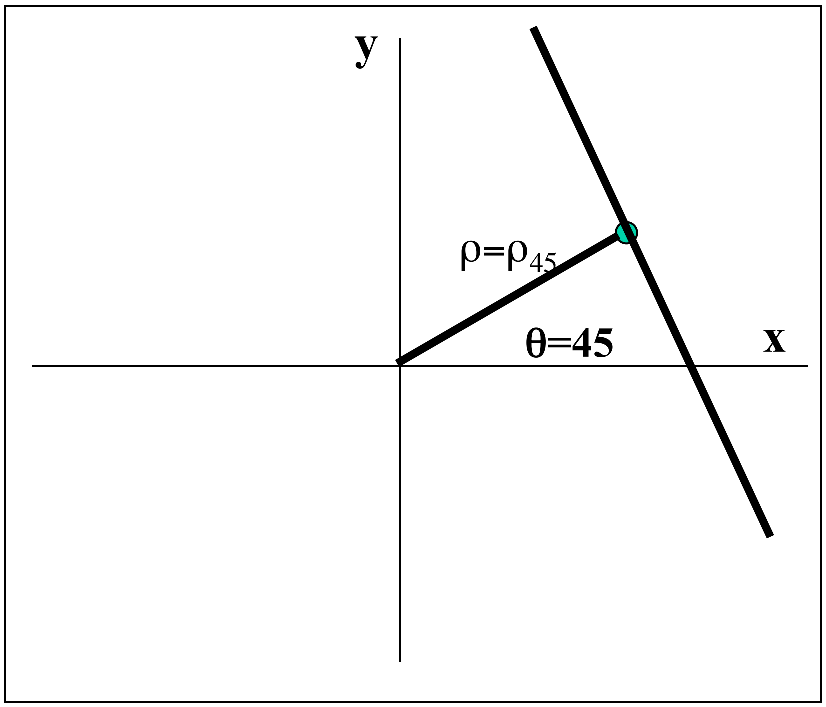
Hough Parameter Space



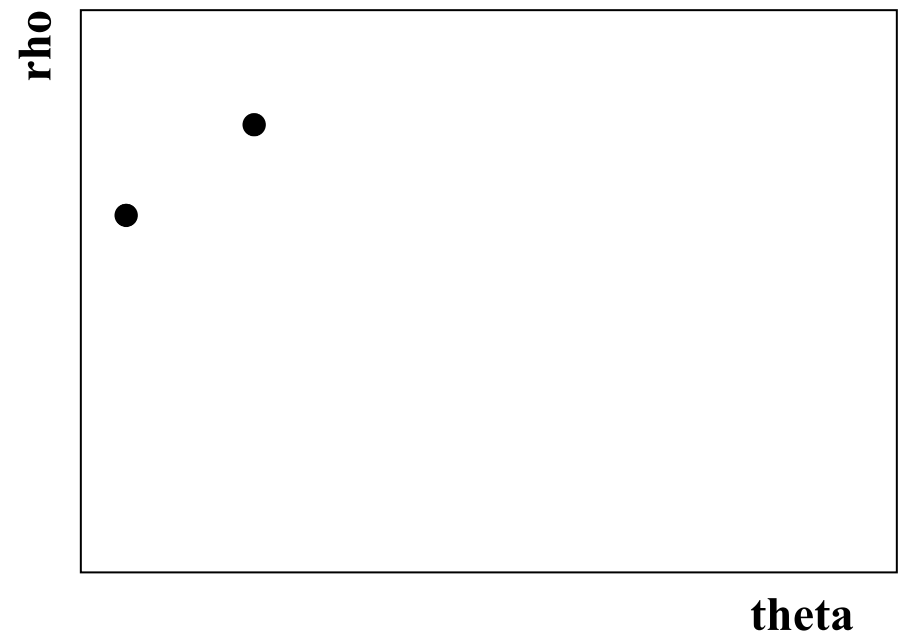
Given a point, vote for (θ,ρ) params representing all lines that pass through that point.

Hough Transform for Lines

Image Space



Hough Parameter Space

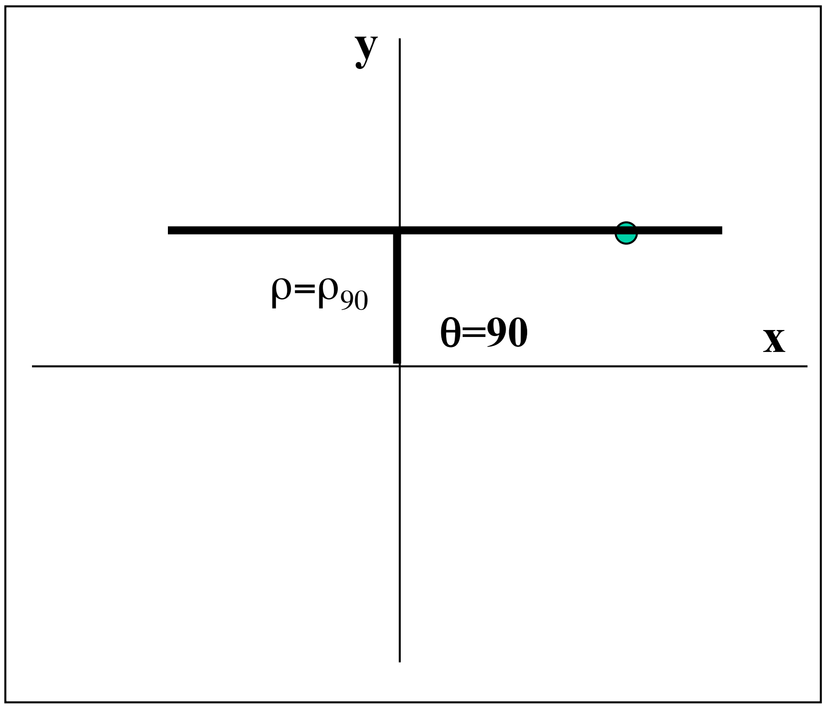


Given a point, vote for (θ,ρ) params representing all lines that pass through that point.

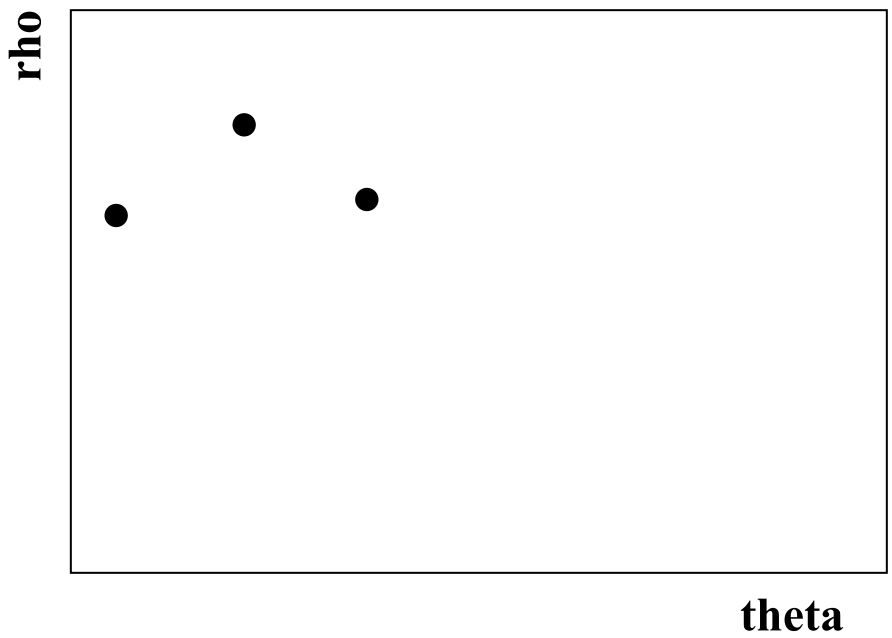
Robert Collins
CSE486, Penn State

Hough Transform for Lines

Image Space



Hough Parameter Space

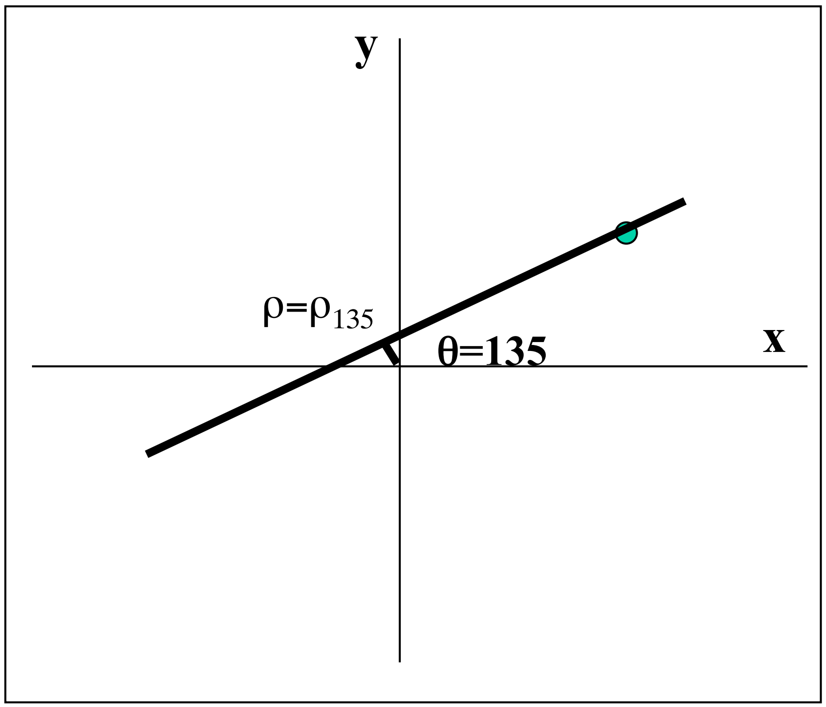


Given a point, vote for (θ,ρ) params representing all lines that pass through that point.

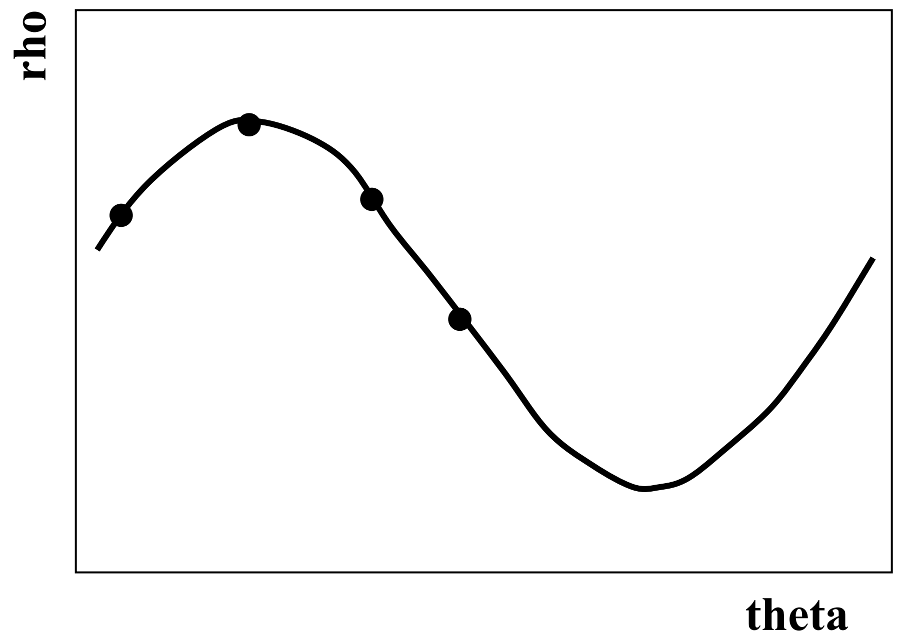
Robert Collins
CSE486, Penn State

Hough Transform for Lines

Image Space



Hough Parameter Space

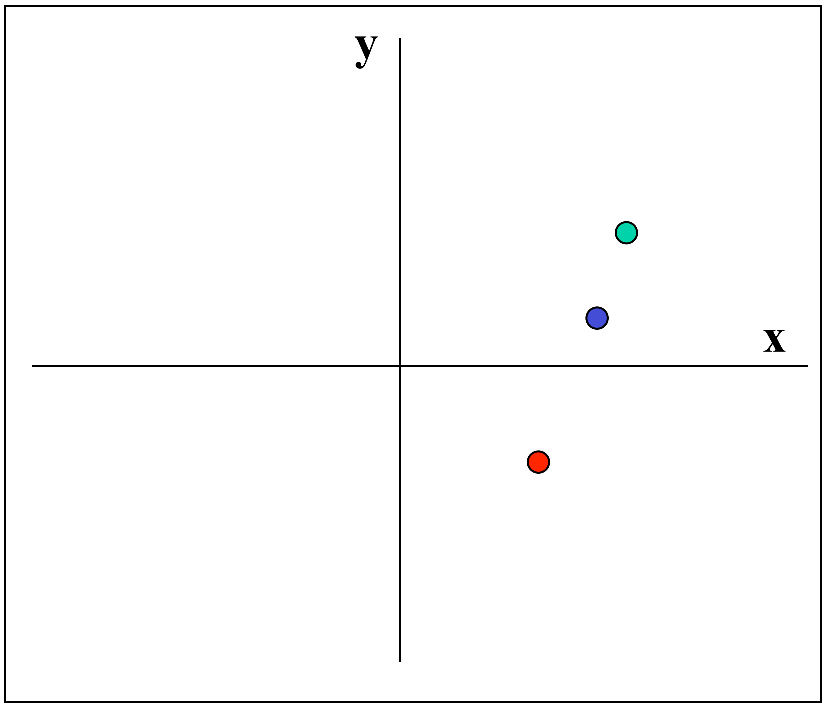


Given a point, vote for (θ,ρ) params representing all lines that pass through that point.

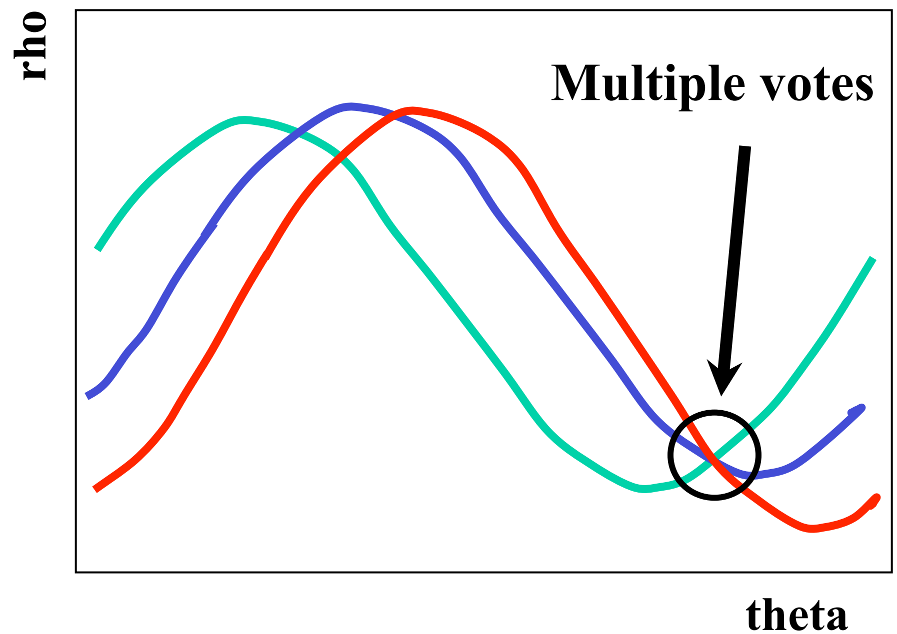
Robert Collins
CSE486, Penn State

Hough Transform for Lines

Image Space

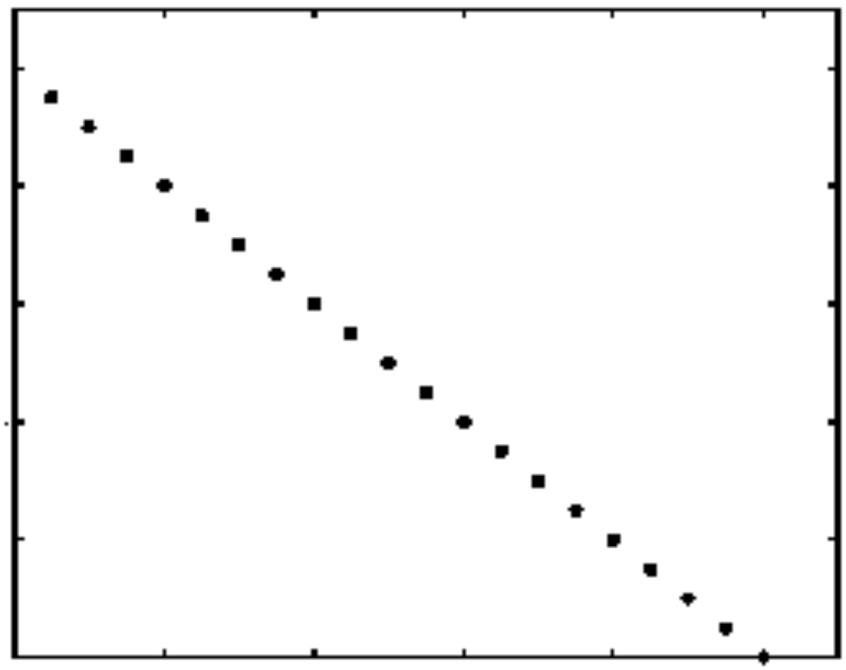


Hough Parameter Space

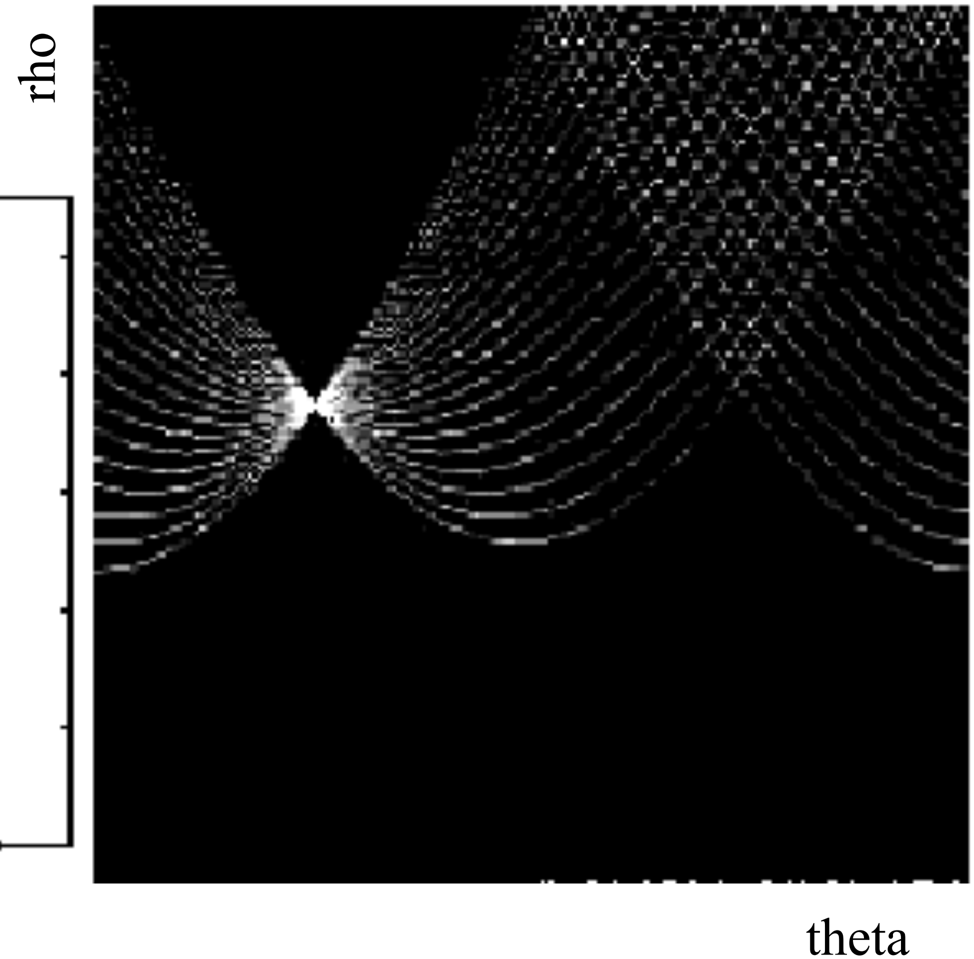


Given a point, vote for (θ,ρ) params representing all lines that pass through that point.

Examples



points

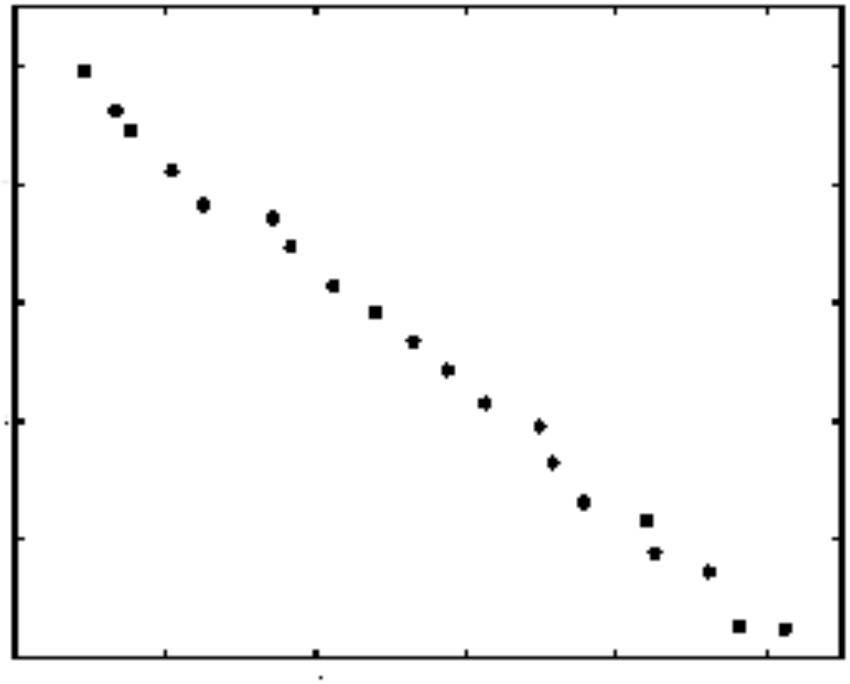


votes

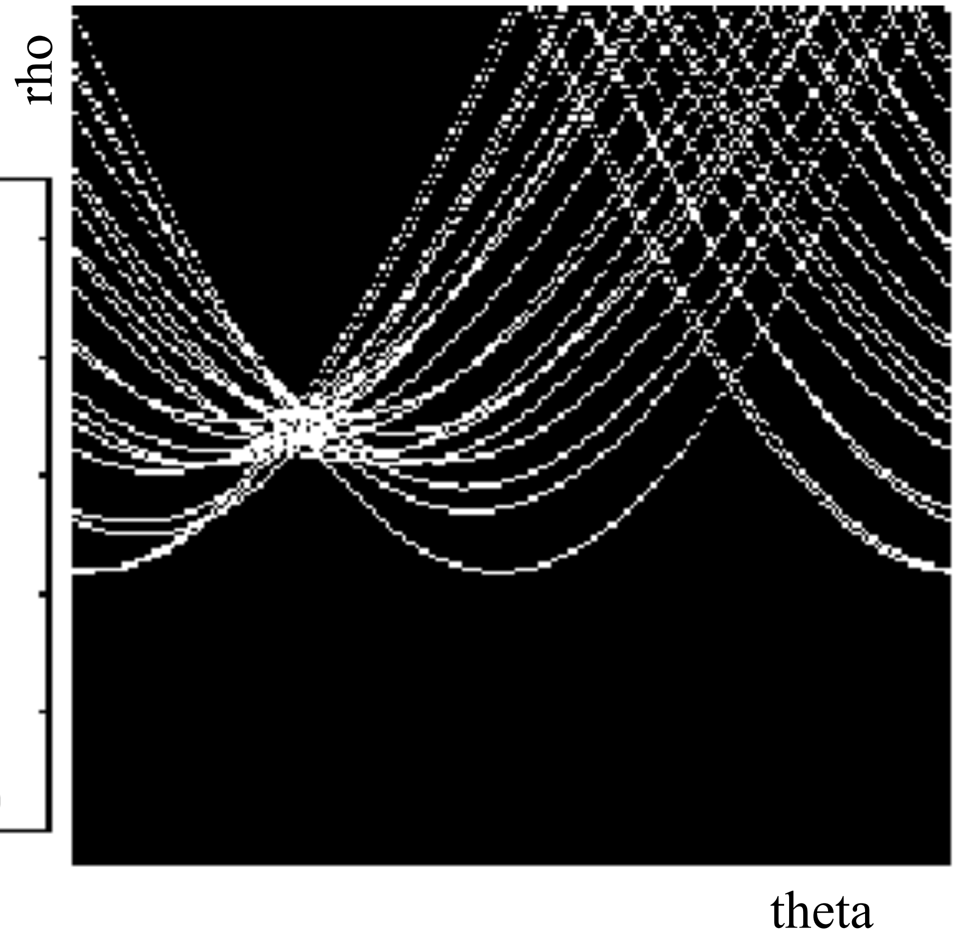
Note: we only need to vote for theta in range 0 to 180

Robert Collins
CSE486, Penn State

Examples

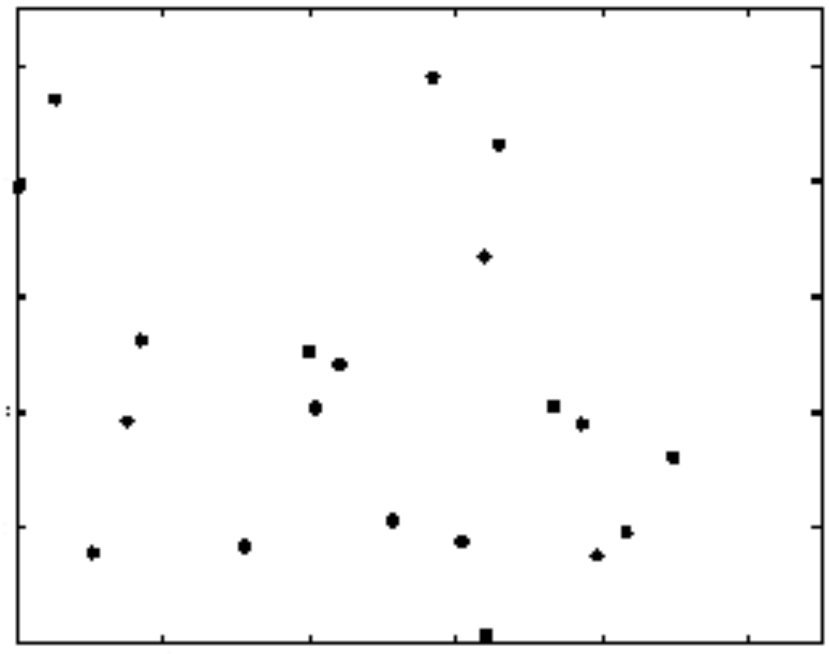


points

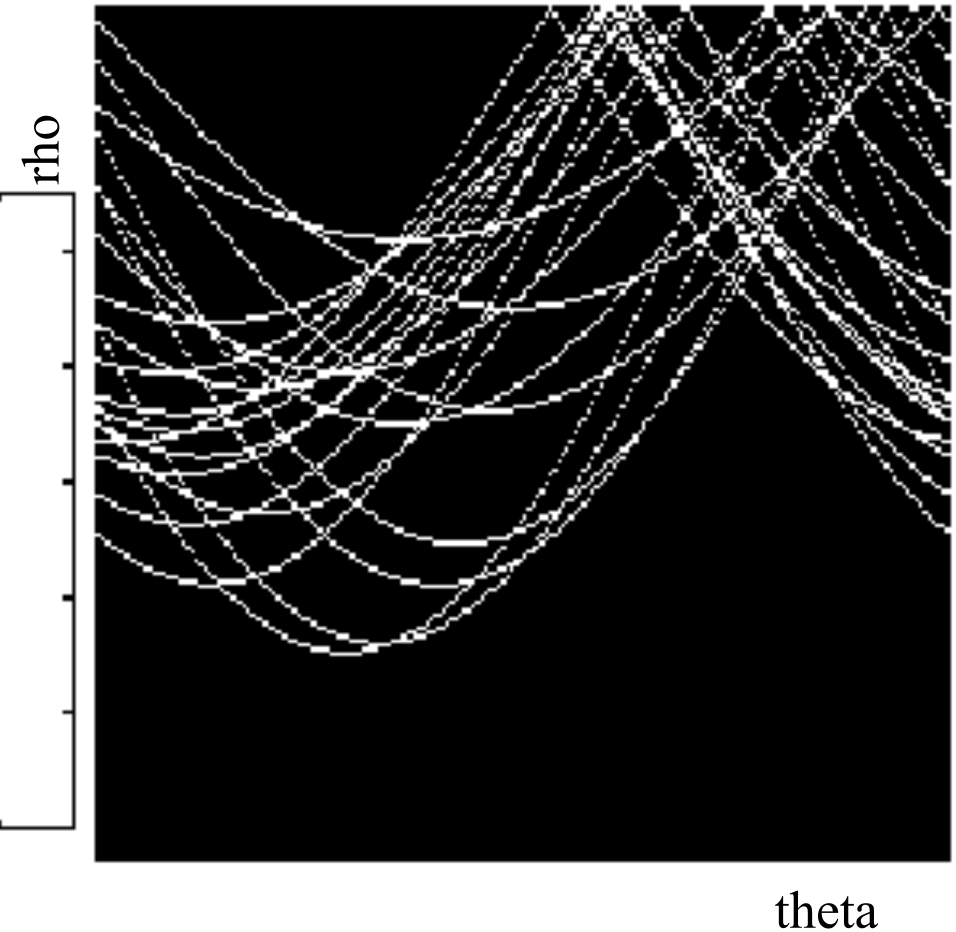


votes

Examples

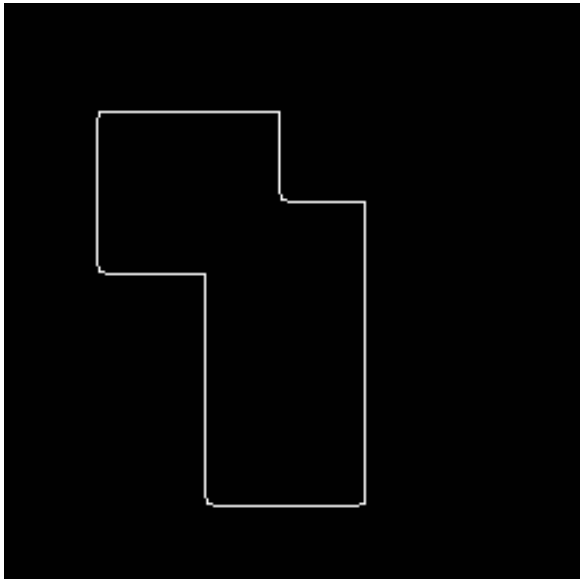


points



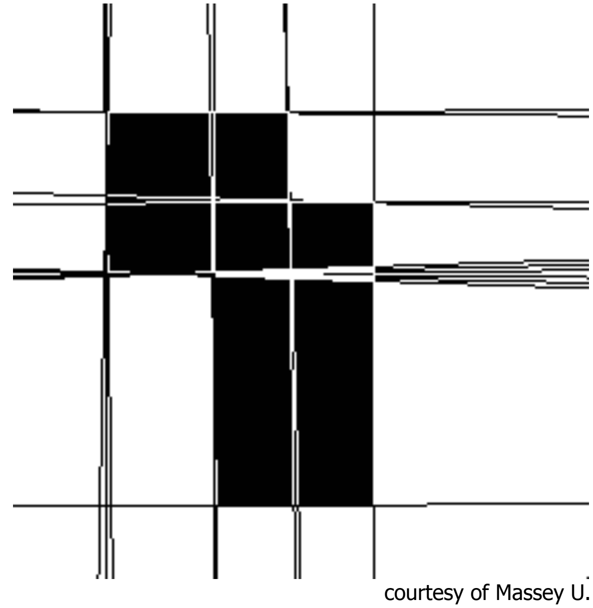
votes

Examples



Canny Edge Image

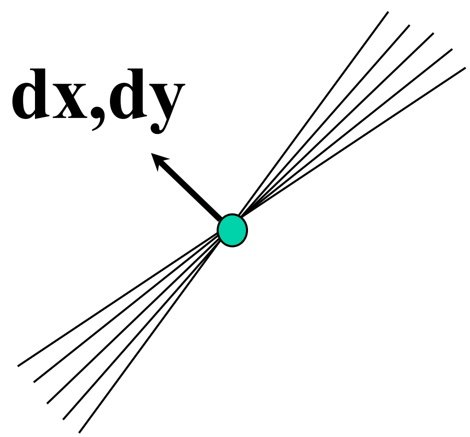
Accumulator array



Lines from accum cells <= 70% of max votes

Using Edge Orientation

Typically, when we extract edges we have more than just point locations. We also have estimates of the gradient (orientation) of each edge.



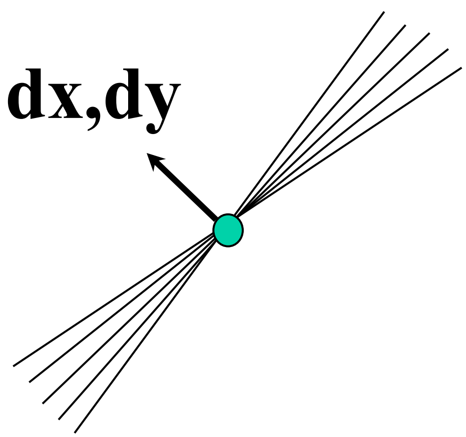
Line passing through point should be perpendicular to gradient. This constraint uniquely identifies a (θ,ρ) pair in accumulator space.

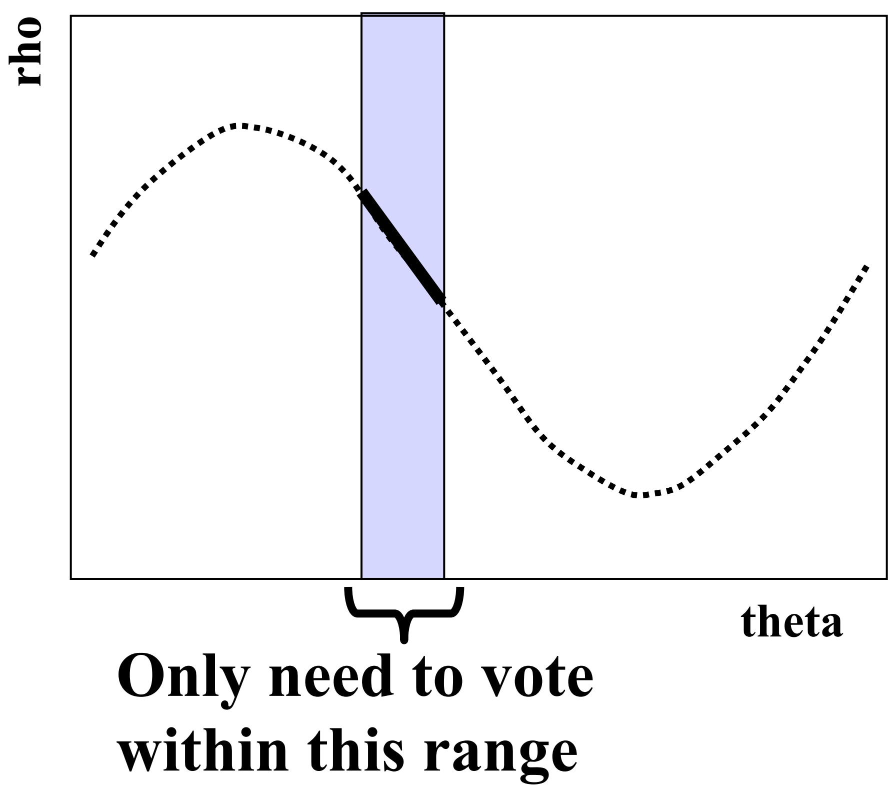
However, to account for noise in gradient estimate, we should vote for a range of θ,ρ pairs

Robert Collins
CSE486, Penn State

Using Edge Orientation

Typically, when we extract edges we have more than just point locations. We also have estimates of the gradient (orientation) of each edge.





Hough Transform for Circles

Parameterization: $(x - a)^2 + (y - b)^2 = r^2$

Params are (a,b,r) => 3 dimensional parameterization

First consider: r is known. Then 2 params (a,b)

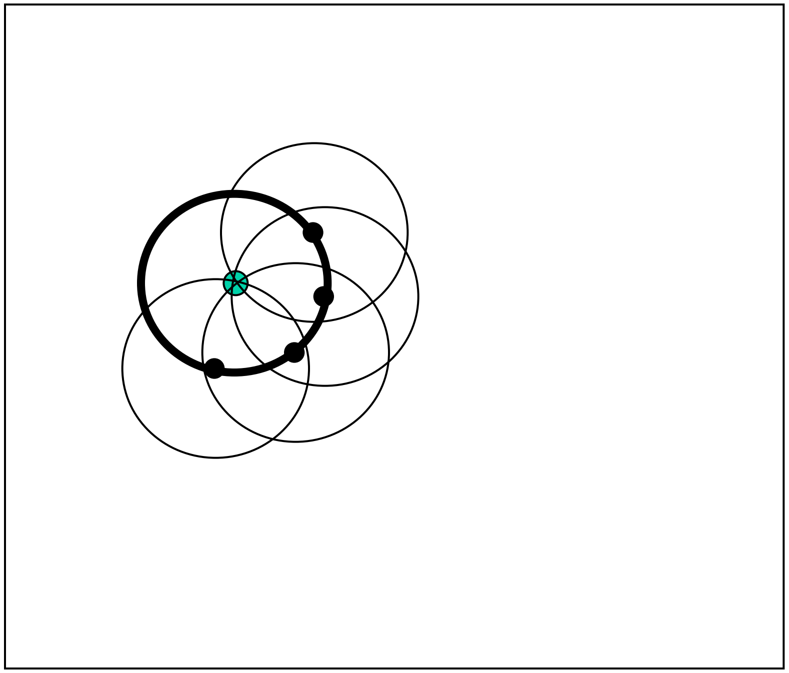


Image space

What is set of points that can be center of a circle of radius r passing through this point?

Parameter space

Robert Collins
CSE486, Penn State

Hough Transform for Circles

Parameterization: $(x - a)^2 + (y - b)^2 = r^2$

Params are (a,b,r) => 3 dimensional parameterization

First consider: r is known. Then 2 params (a,b)

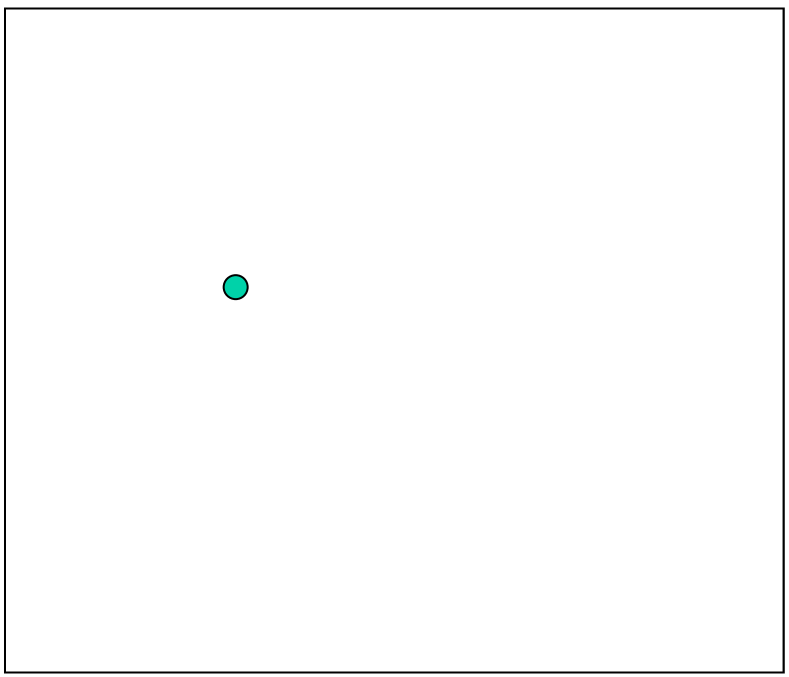
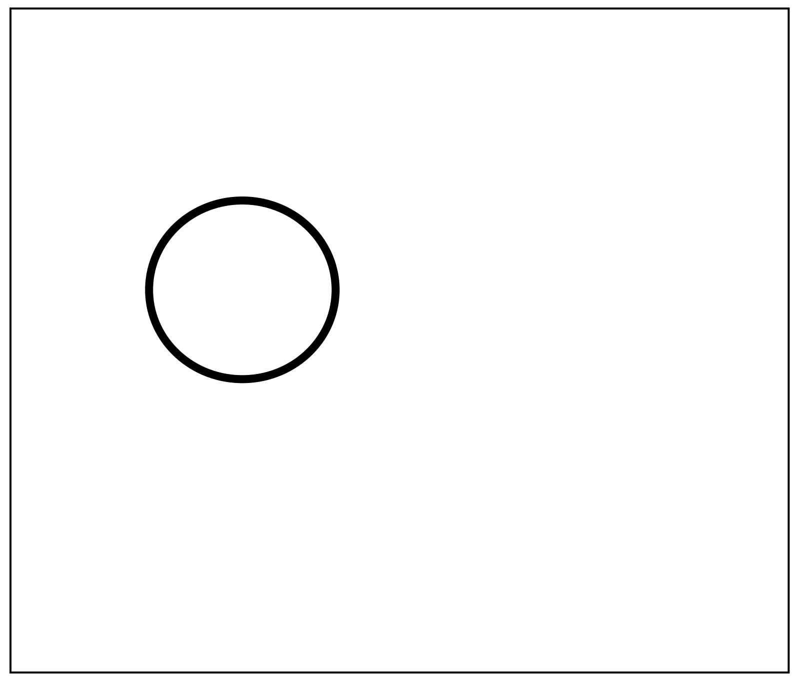


Image space



Parameter space

Robert Collins
CSE486, Penn State

Hough Transform for Circles

Parameterization: $(x - a)^2 + (y - b)^2 = r^2$

Params are (a,b,r) => 3 dimensional parameterization

First consider: r is known. Then 2 params (a,b)

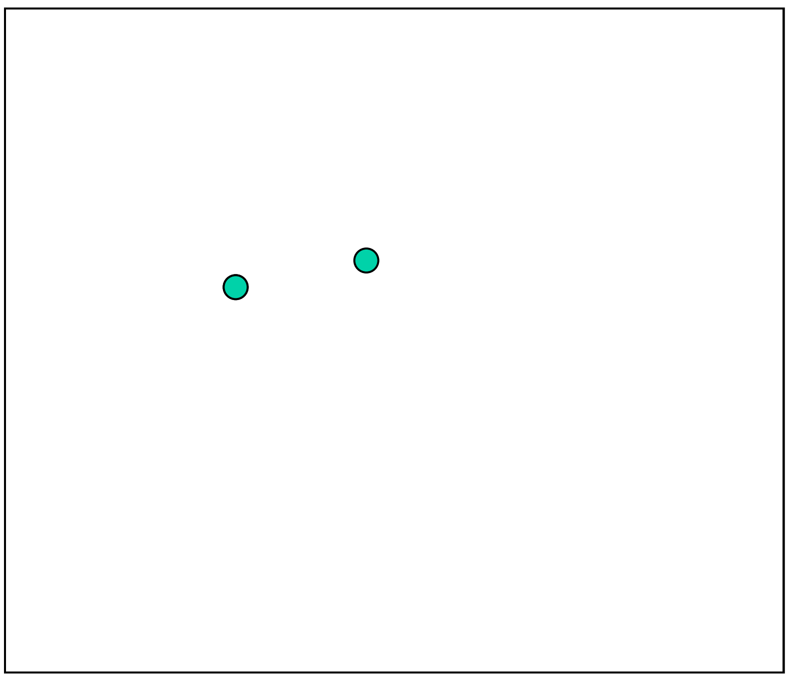
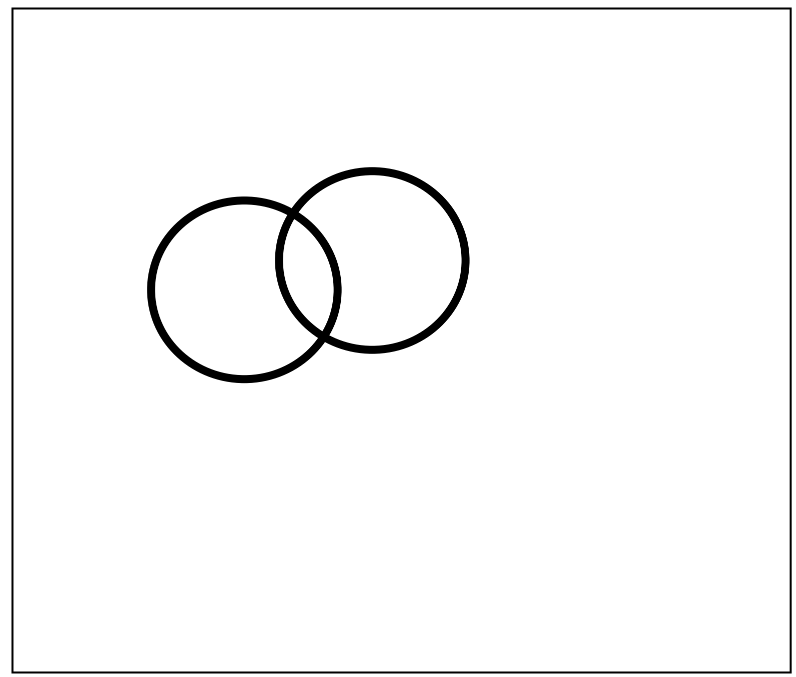


Image space



Parameter space

Hough Transform for Circles

Parameterization: $(x - a)^2 + (y - b)^2 = r^2$

Params are (a,b,r) => 3 dimensional parameterization

First consider: r is known. Then 2 params (a,b)

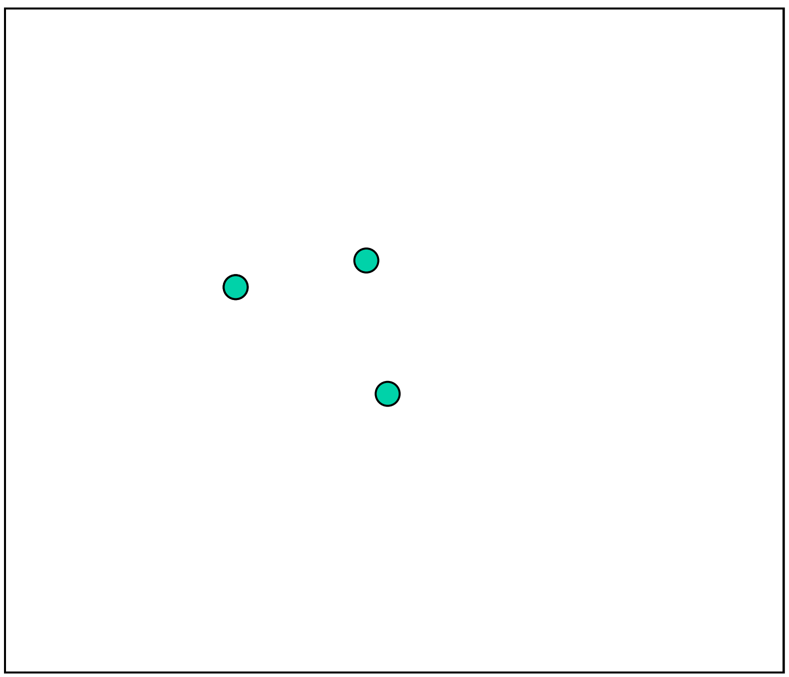
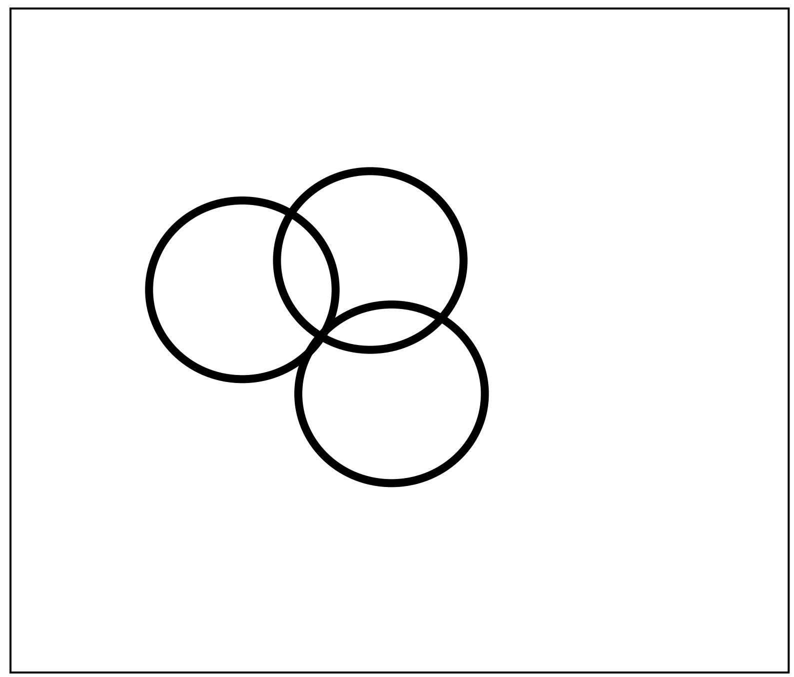


Image space



Parameter space

Robert Collins
CSE486, Penn State

Hough Transform for Circles

Parameterization: $(x - a)^2 + (y - b)^2 = r^2$

Params are (a,b,r) => 3 dimensional parameterization

First consider: r is known. Then 2 params (a,b)

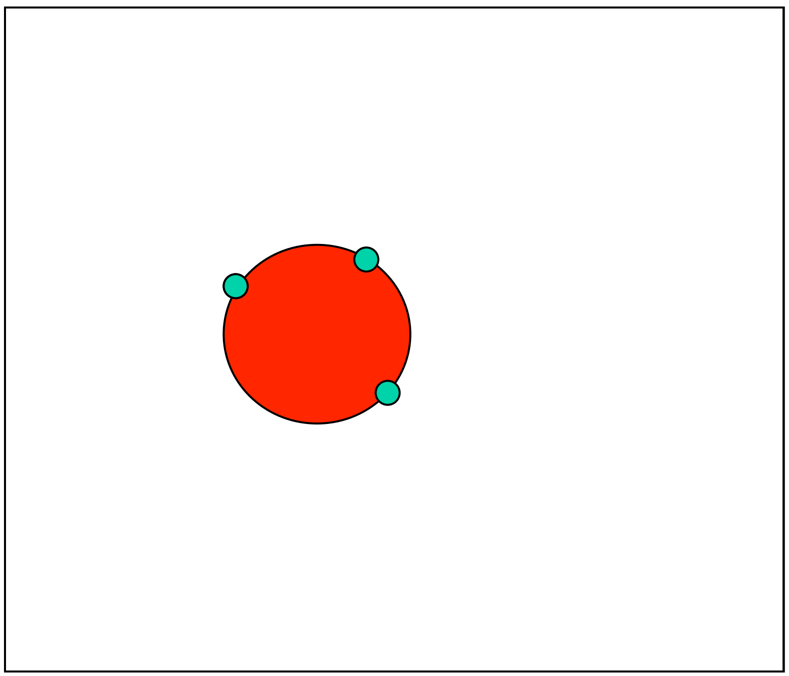
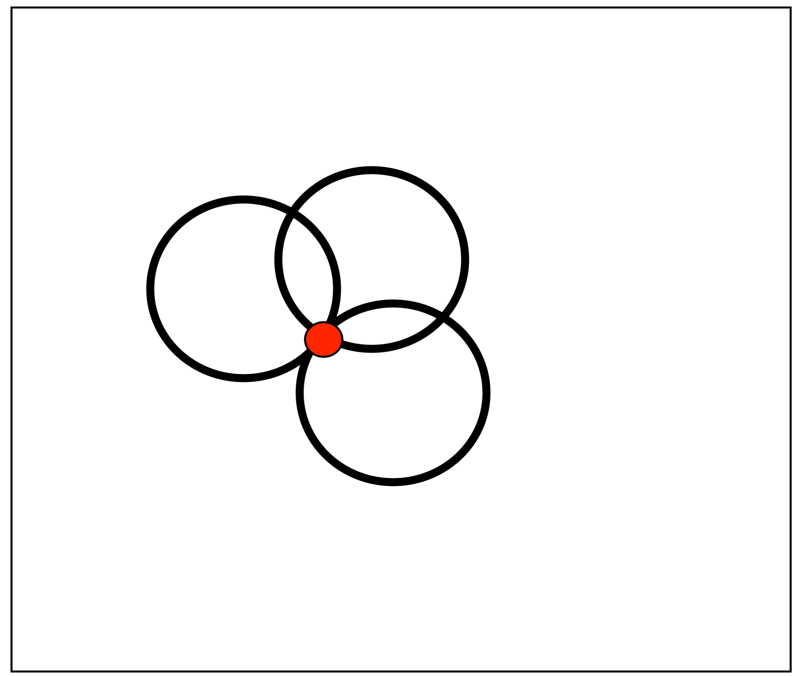


Image space



Parameter space

Robert Collins
CSE486, Penn State

Hough Transform for Circles

Parameterization: $(x - a)^2 + (y - b)^2 = r^2$

Params are (a,b,r) => 3 dimensional parameterization

Now what if radius r is not known?

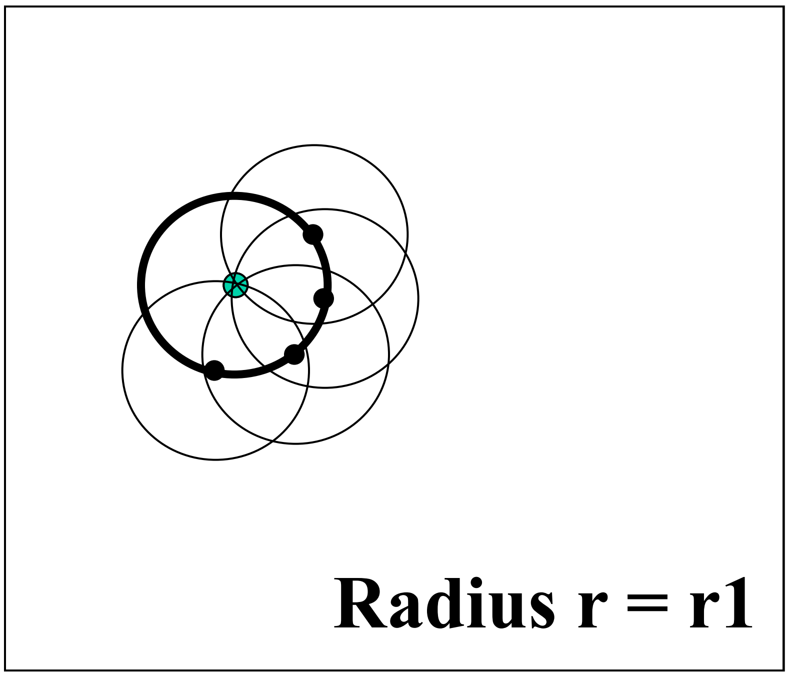
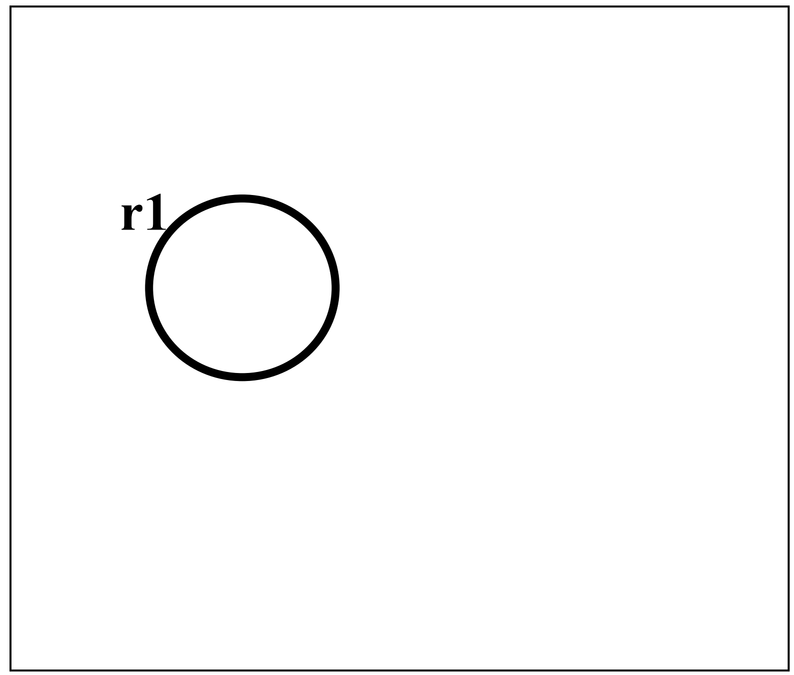


Image space



Parameter space

Robert Collins
CSE486, Penn State

Hough Transform for Circles

Parameterization: $(x - a)^2 + (y - b)^2 = r^2$

Params are (a,b,r) => 3 dimensional parameterization

Now what if radius r is not known?

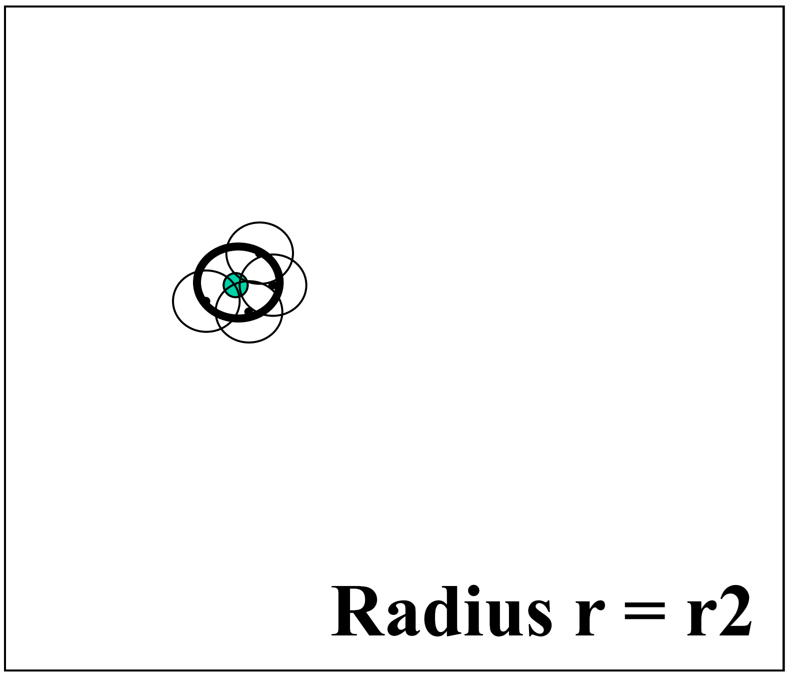
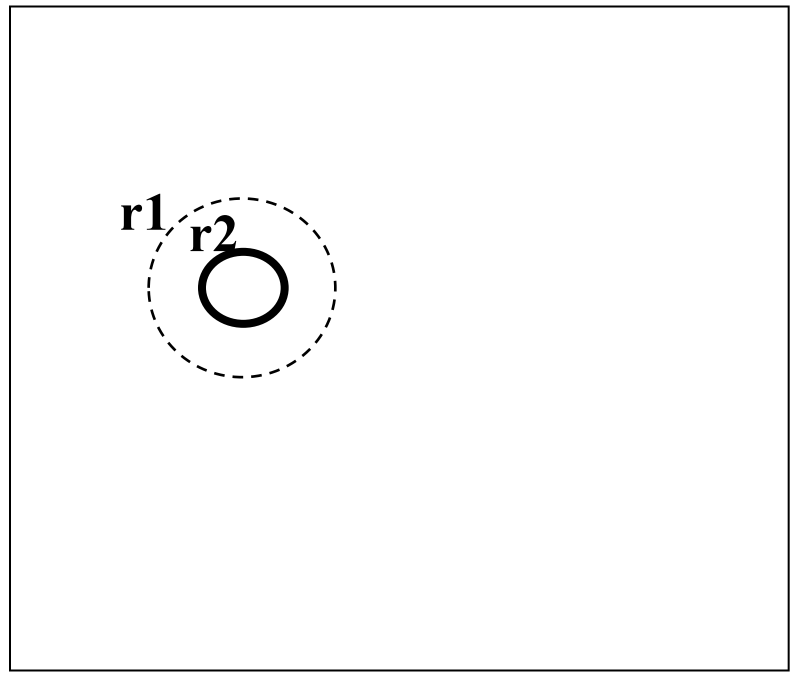


Image space



Parameter space

Robert Collins
CSE486, Penn State

Hough Transform for Circles

Parameterization: $(x - a)^2 + (y - b)^2 = r^2$

Params are (a,b,r) => 3 dimensional parameterization

Now what if radius r is not known?

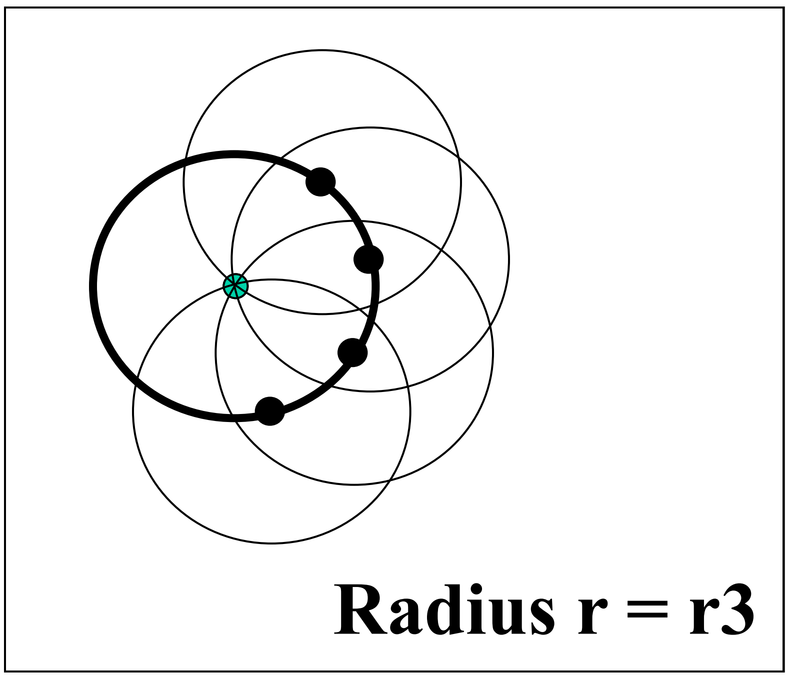
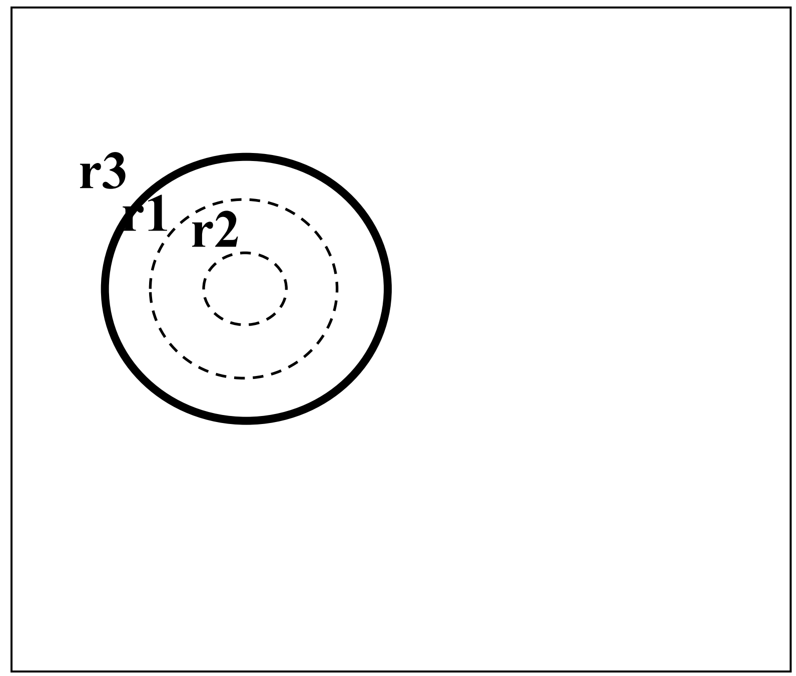


Image space



Parameter space

Hough Transform for Circles

Parameterization: $(x - a)^2 + (y - b)^2 = r^2$

Params are (a,b,r) => 3 dimensional parameterization

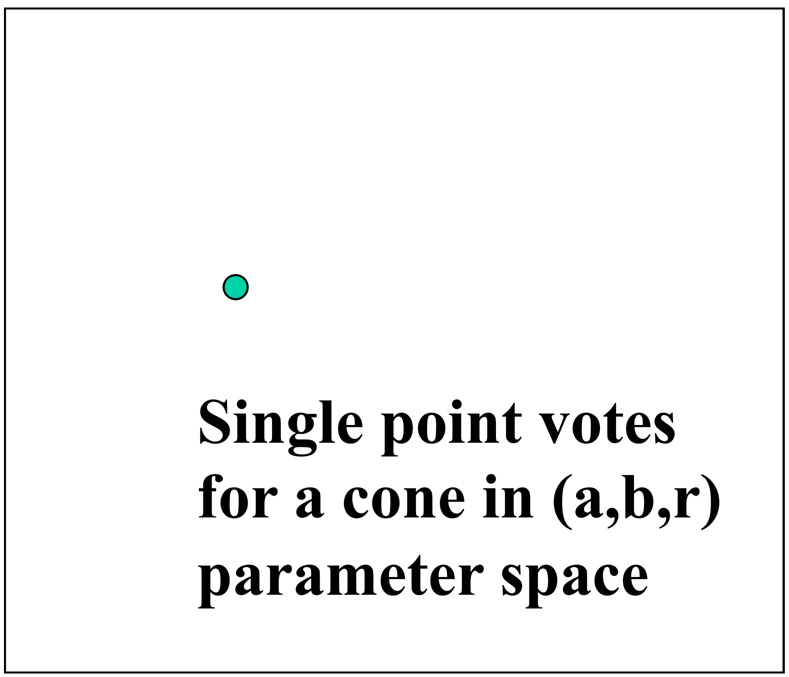
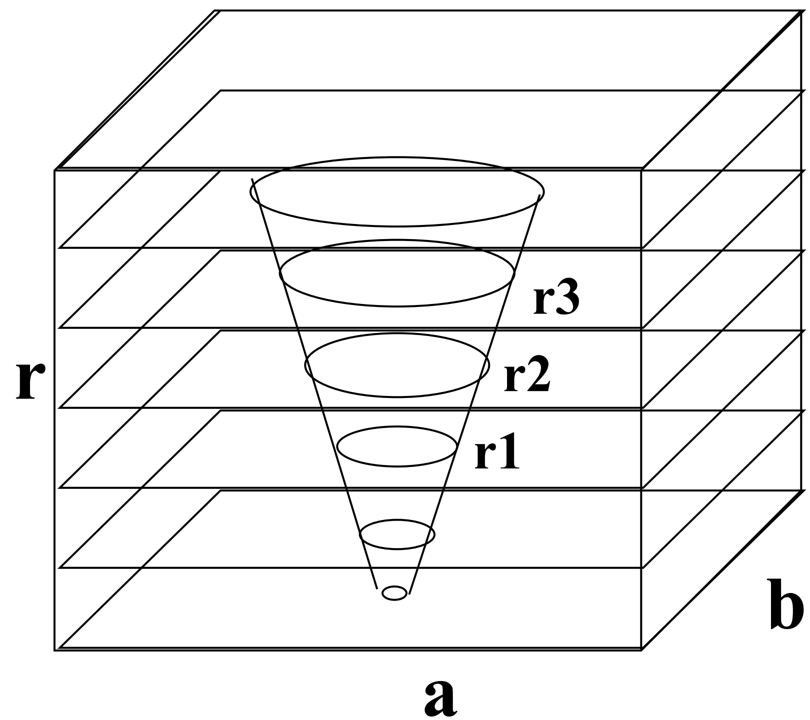


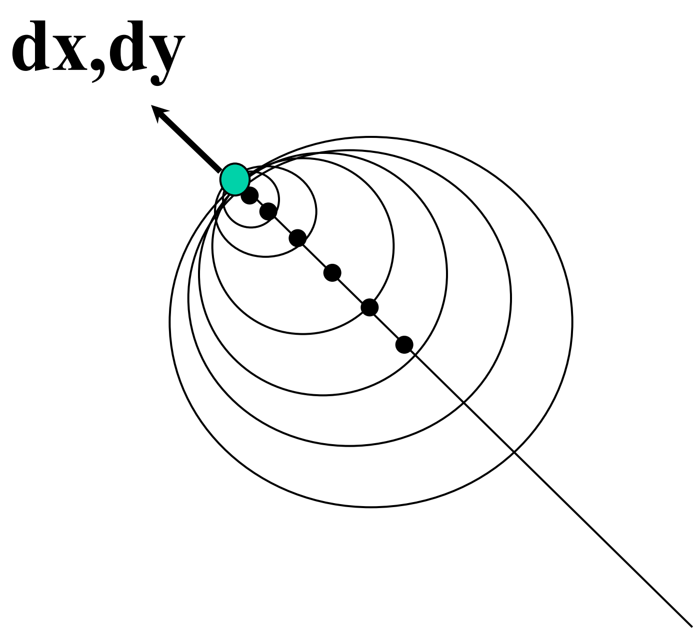
Image space



Parameter space

Hough Transform for Circles

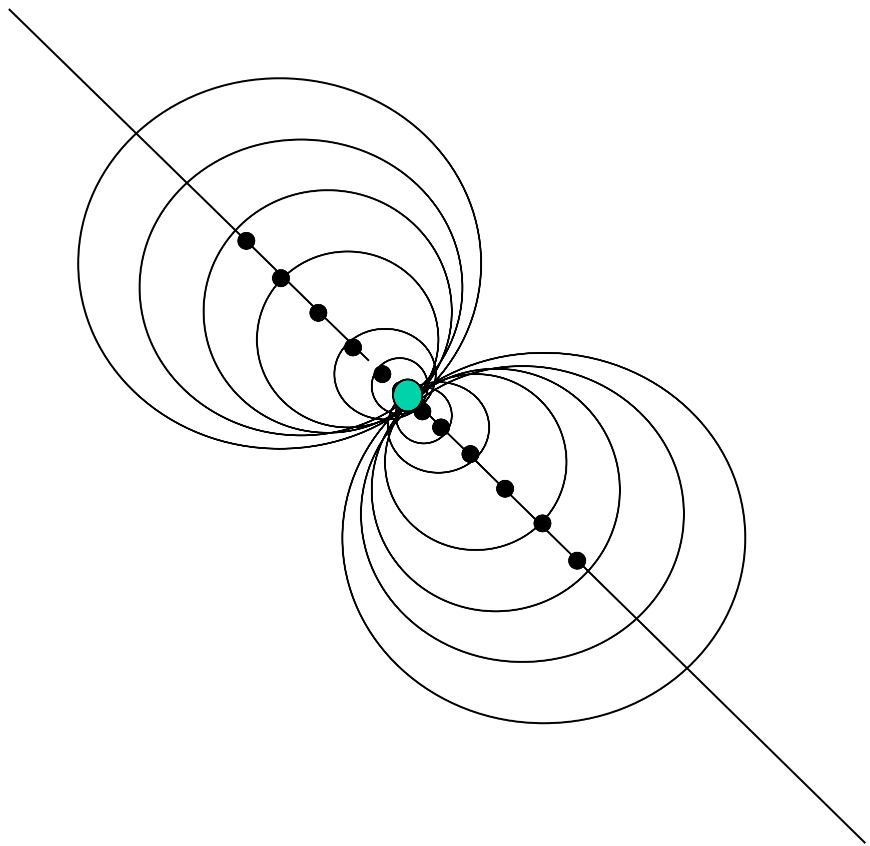
What if we also have gradients at each point?

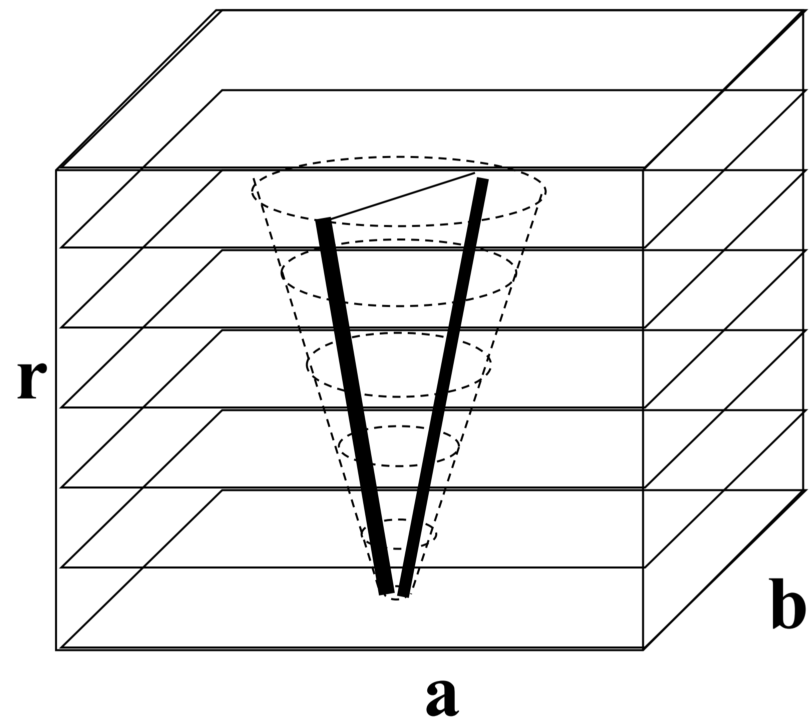


Robert Collins
CSE486, Penn State

Hough Transform for Circles

What if we also have gradients at each point?





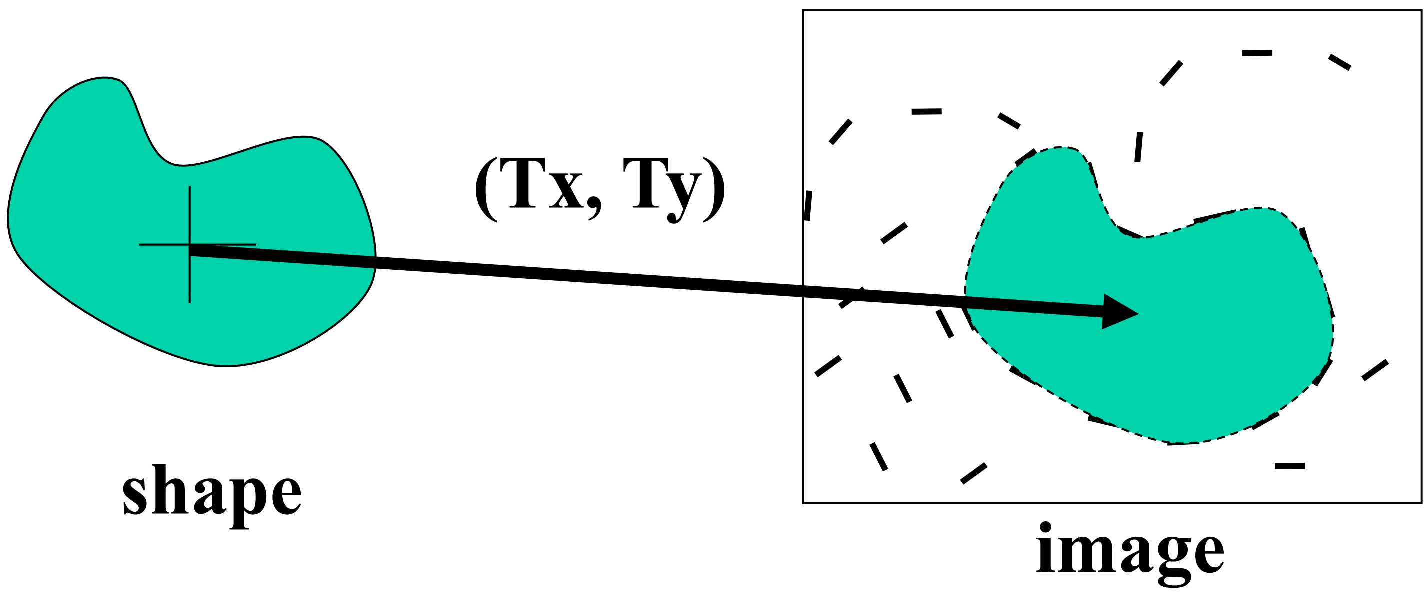
Parameter space

We only have to vote for two bands of the cone in (a,b,r) space.

Arbitrary Shapes

D. H. Ballard
Generalizing the Hough Transform to Detect Arbitrary Shapes
Pattern Recognition, **13**(2):111-122, 1981.

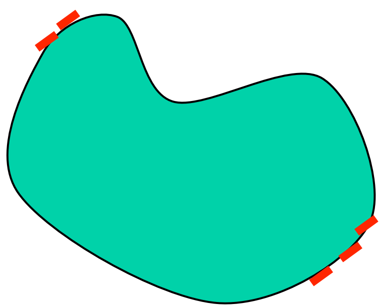
Lets say we just want to find the location (offset in image) of an arbitrary shape. We thus have a 2D parameter space (Tx, Ty).



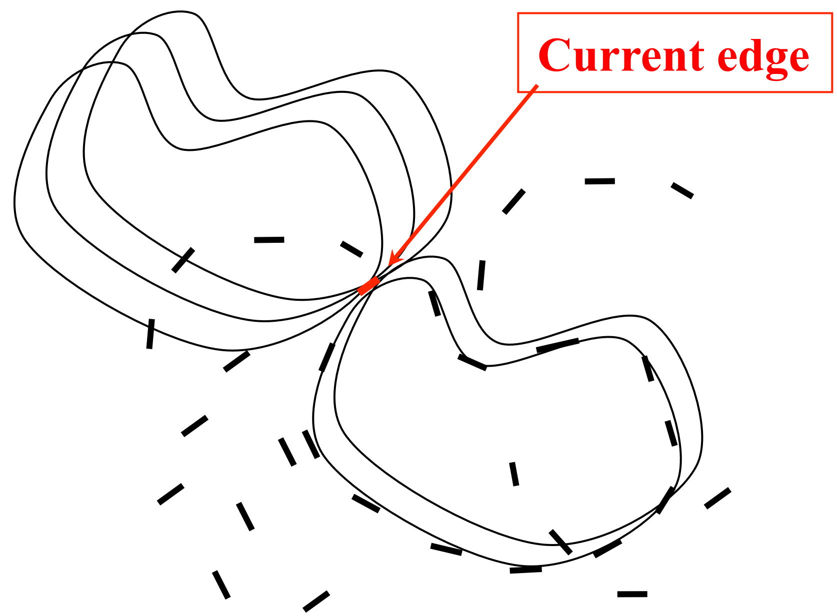
Robert Collins
CSE486, Penn State

Arbitrary Shapes

General idea: for each edge in image, find edges of similar orientation on shape boundary, and use each to vote for an offset (Tx, Ty).

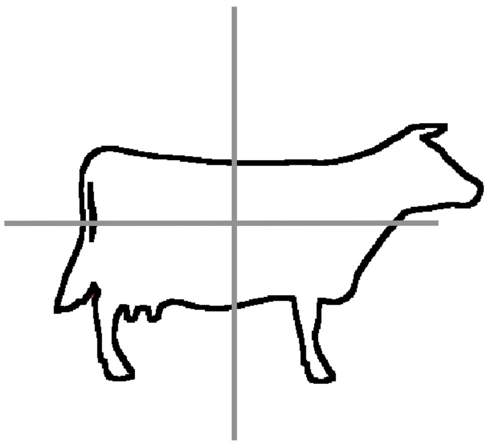


shape

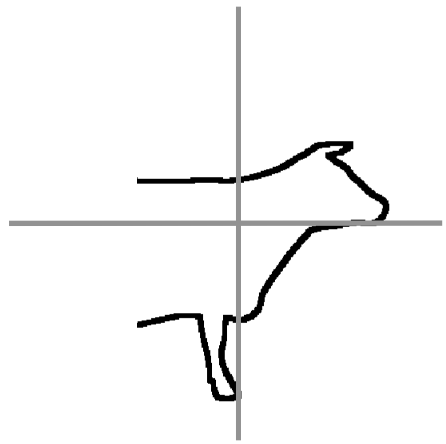


Increment centers of each.
Then, do for all edges.

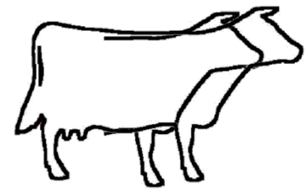
GHT Example



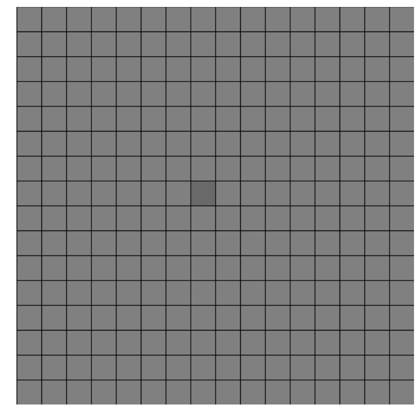
image



shape



offset

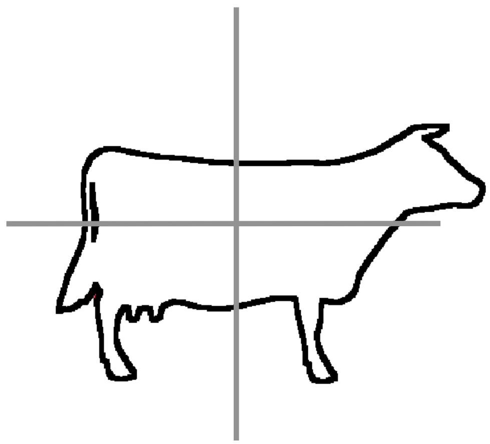


vote

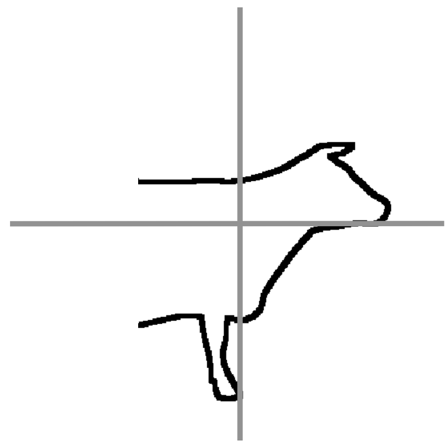
Example from Michael Kazhdan
Johns Hopkins Univ.

Robert Collins
CSE486, Penn State

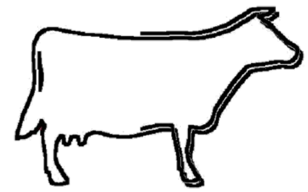
GHT Example



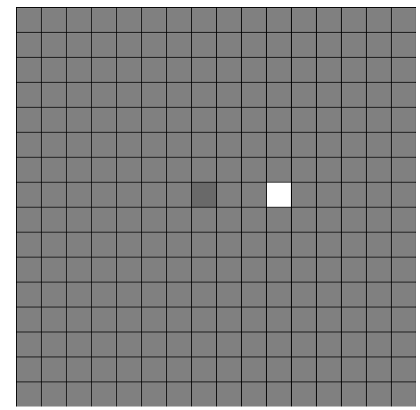
image



shape

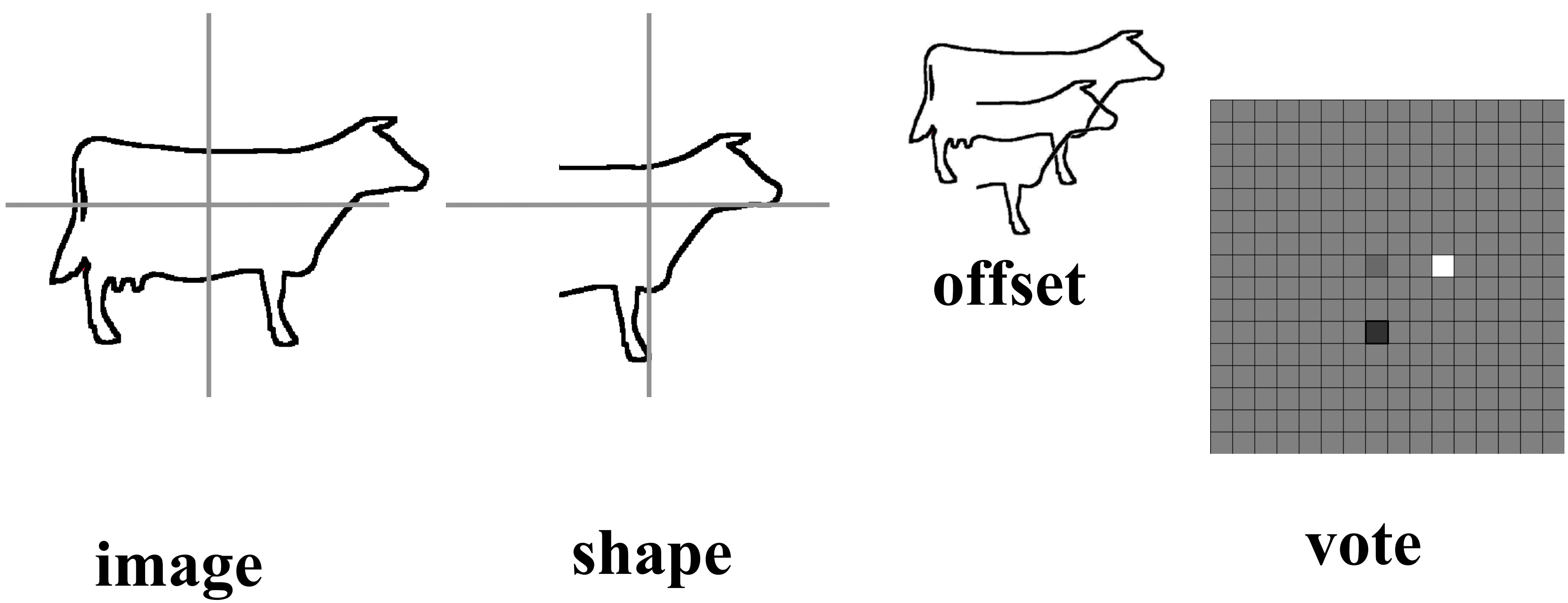


offset



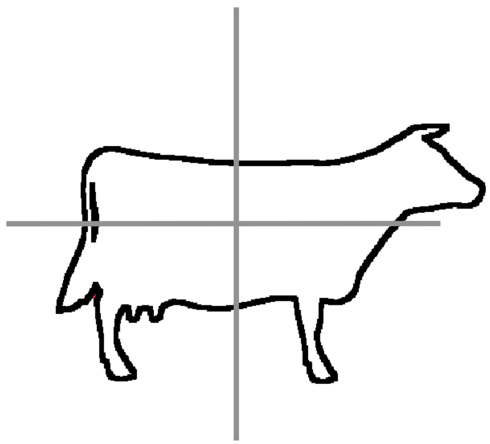
vote

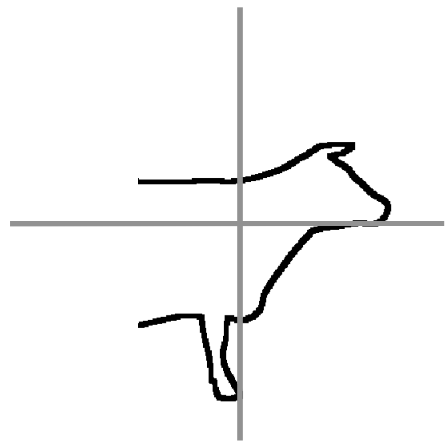
GHT Example

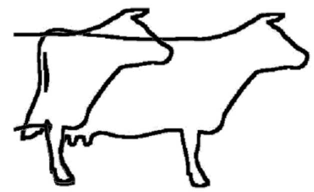


Robert Collins
CSE486, Penn State

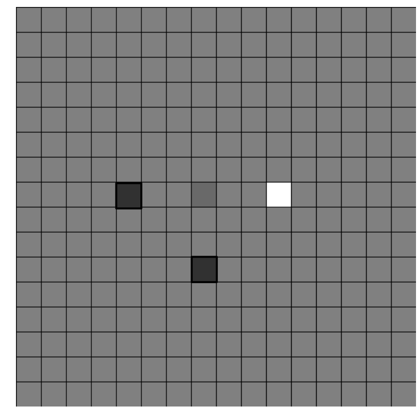
GHT Example







offset

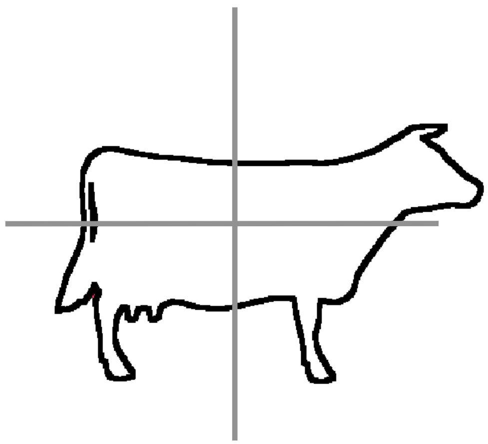


image

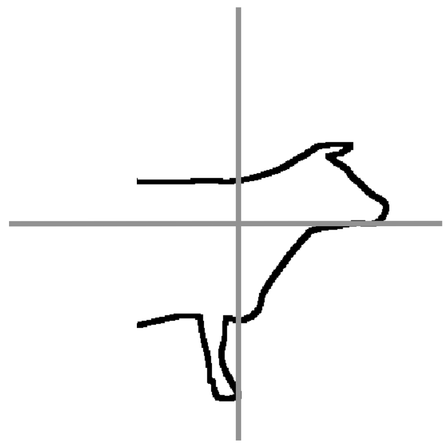
shape

vote

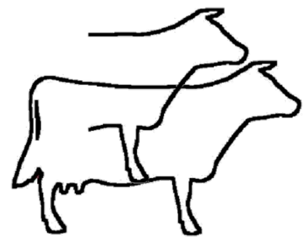
GHT Example



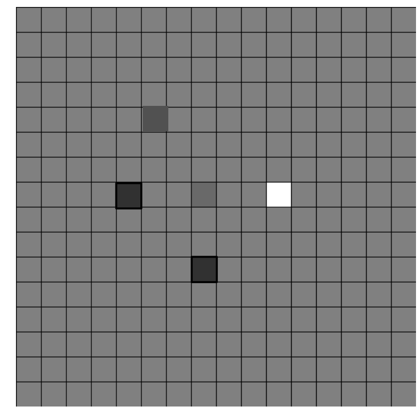
image



shape



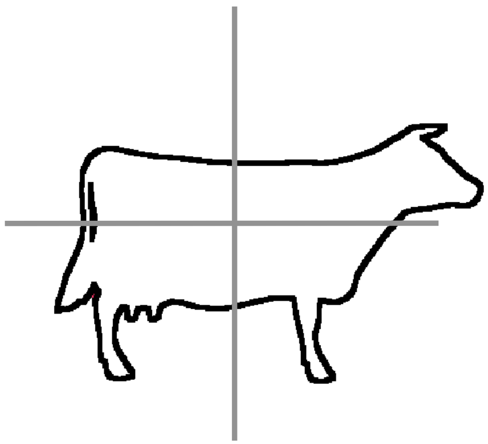
offset



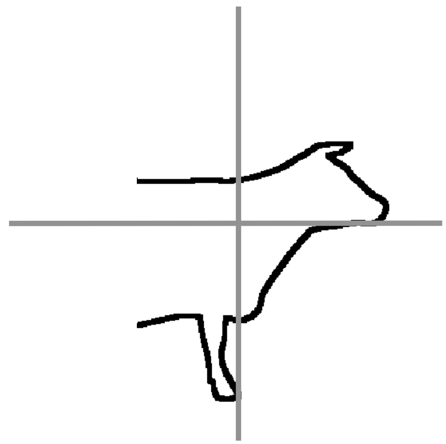
vote

And so on….

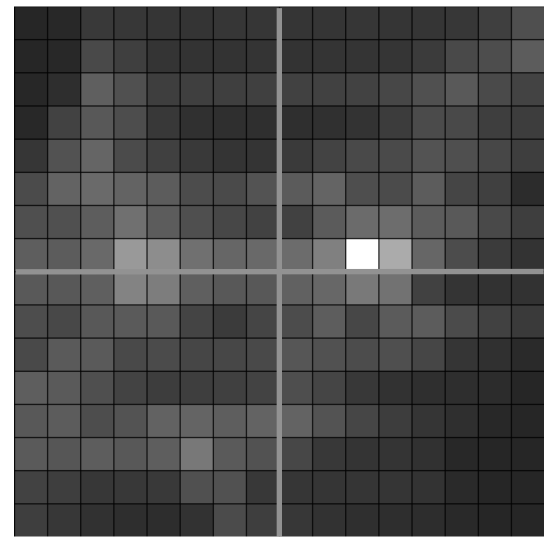
GHT Example



image



shape



offset vote

Insight

Imagine doing GHT with binary edge maps (no orientation available).

So, for each edge in the image, you would loop through each edge in the shape contour, and increment an offset counter in the accum array.

Claim: the result is the same as doing cross correlation of the binary shape with the binary image.

$\boxed{\text{So why do it? Think about number of arithmetic operations involved for each.}}$

GHT vs Correlation

For sparse sets of edges, GHT involves much fewer operations than correlation (particularly if boundaries are represented with linked lists)

Also, if we have unknown rotation and scale, we add two new dimensions to our accumulator space, and it immediately generalizes.