Perceptual Organization (I)

Introduction to Computational and Biological Vision

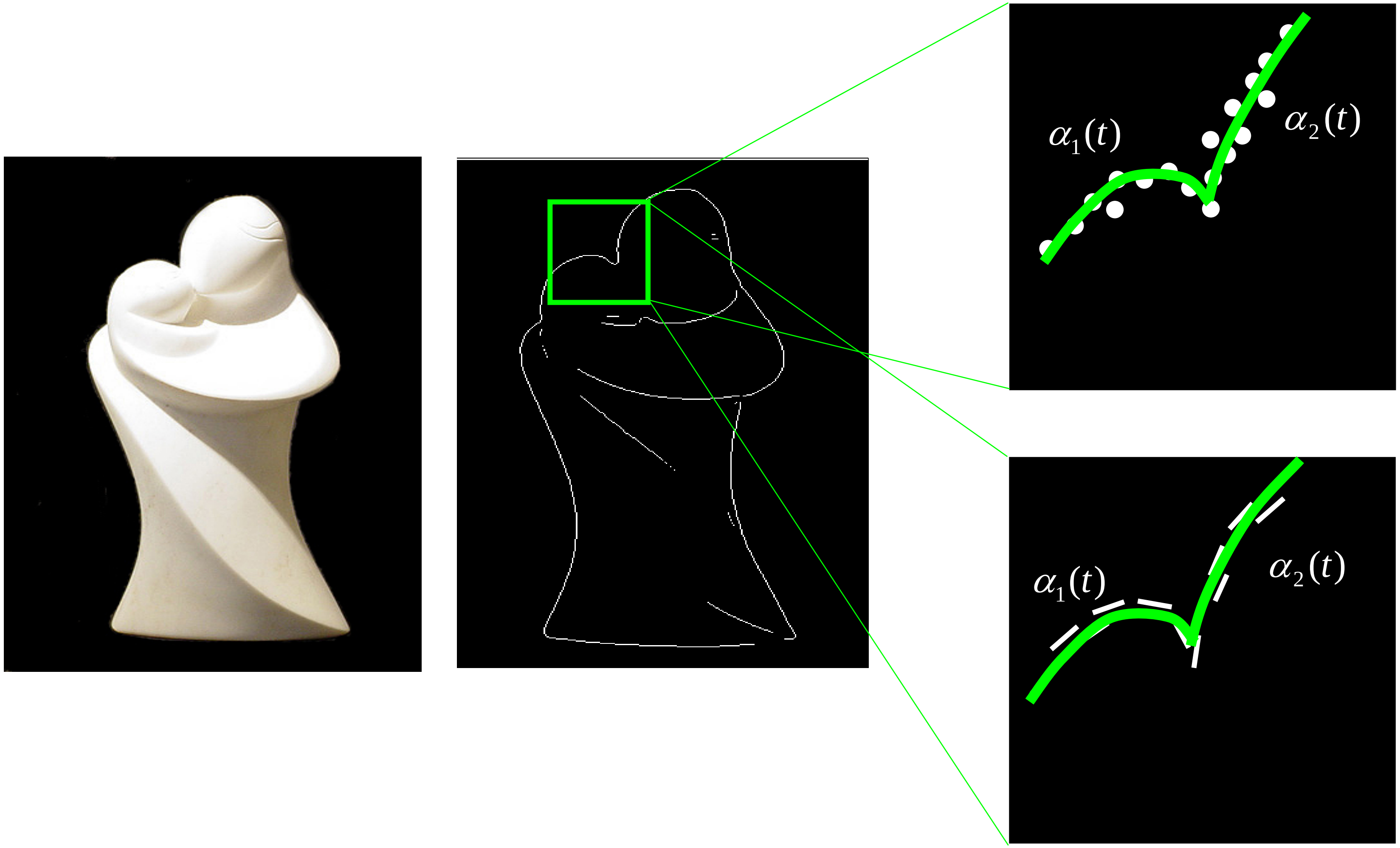
CS 202-1-5261

Computer Science Department, BGU

Ohad Ben-Shahar

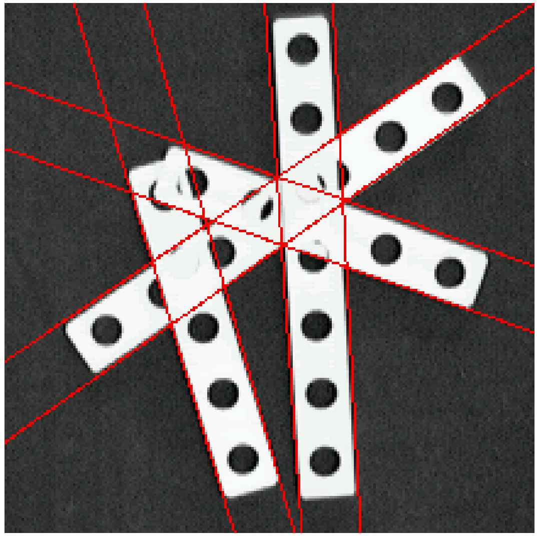
Edge aggregation case study

From local edges to global boundaries



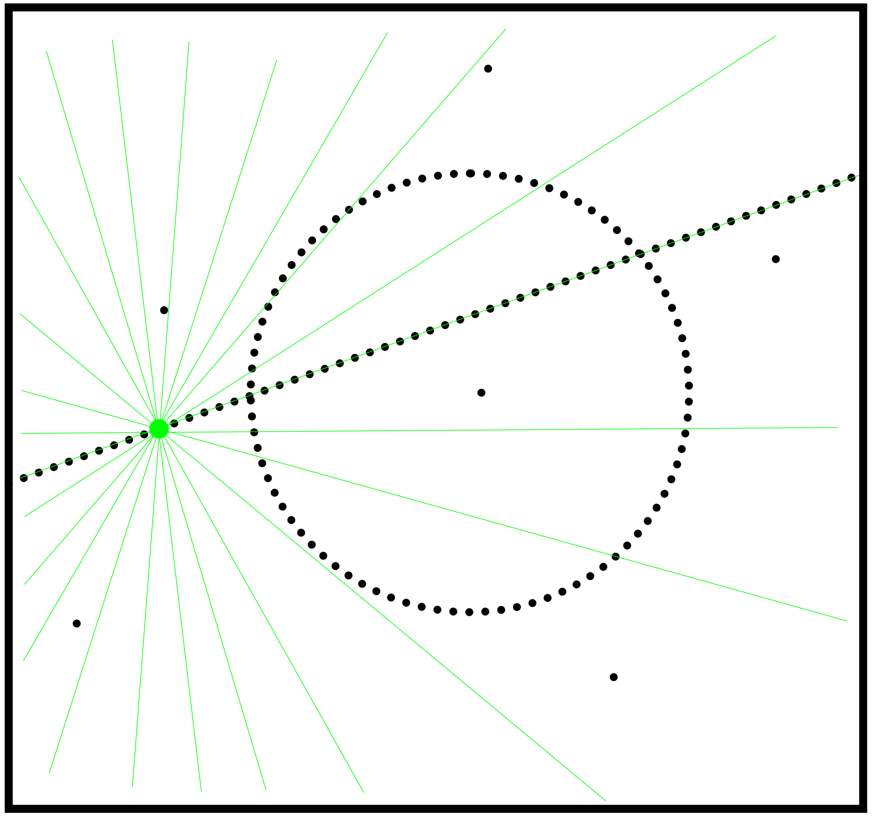
Edge aggregation - a case study

The Hough transform for line detection



Edge aggregation - a case study

The Hough transform for line detection



Given:

List of edge points (arbitrary order)

Compute:

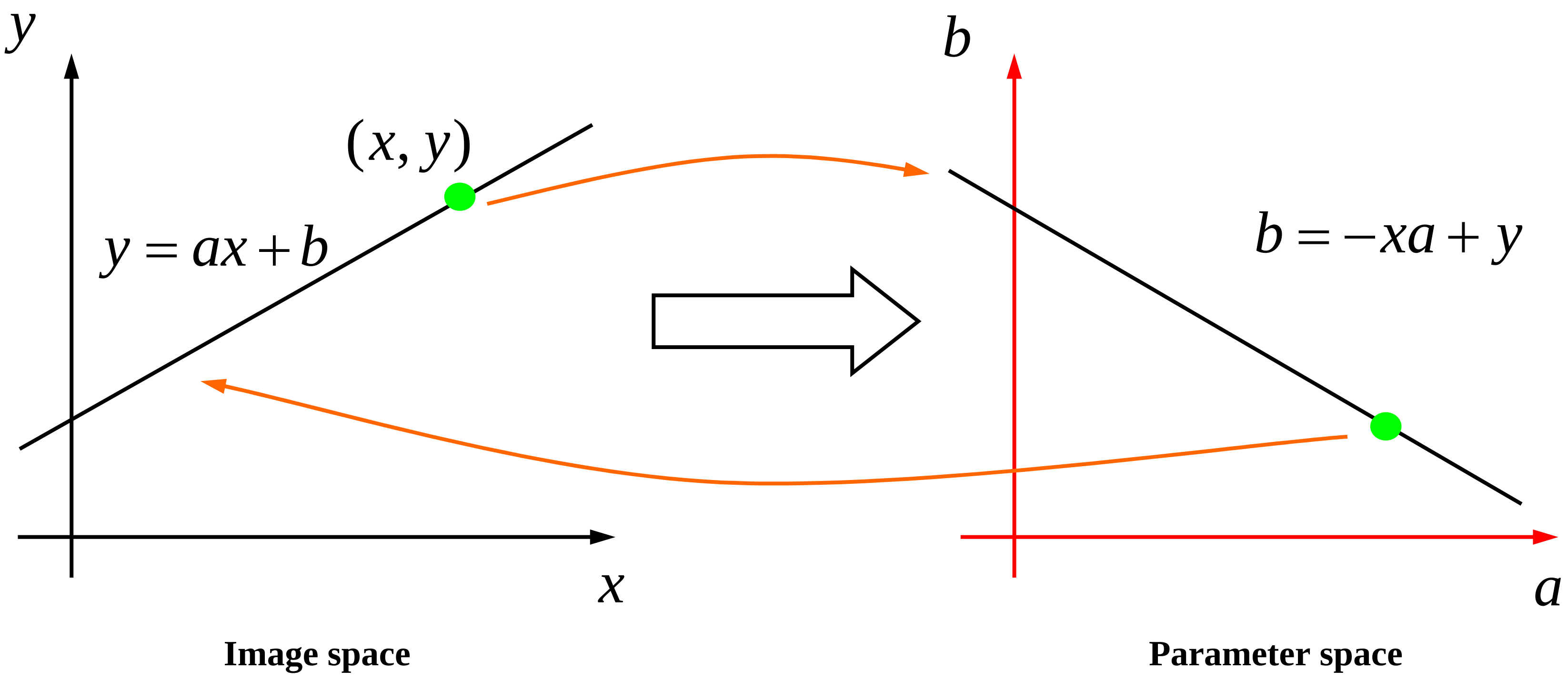
Set of straight lines in the edge map.

Basic idea:

1. **Let each edge point vote for all lines it may belong to.**
2. **Lines with lots of votes “win”**

Edge aggregation - a case study

The Hough transform for line detection



Edge aggregation - a case study

The Hough transform for line detection

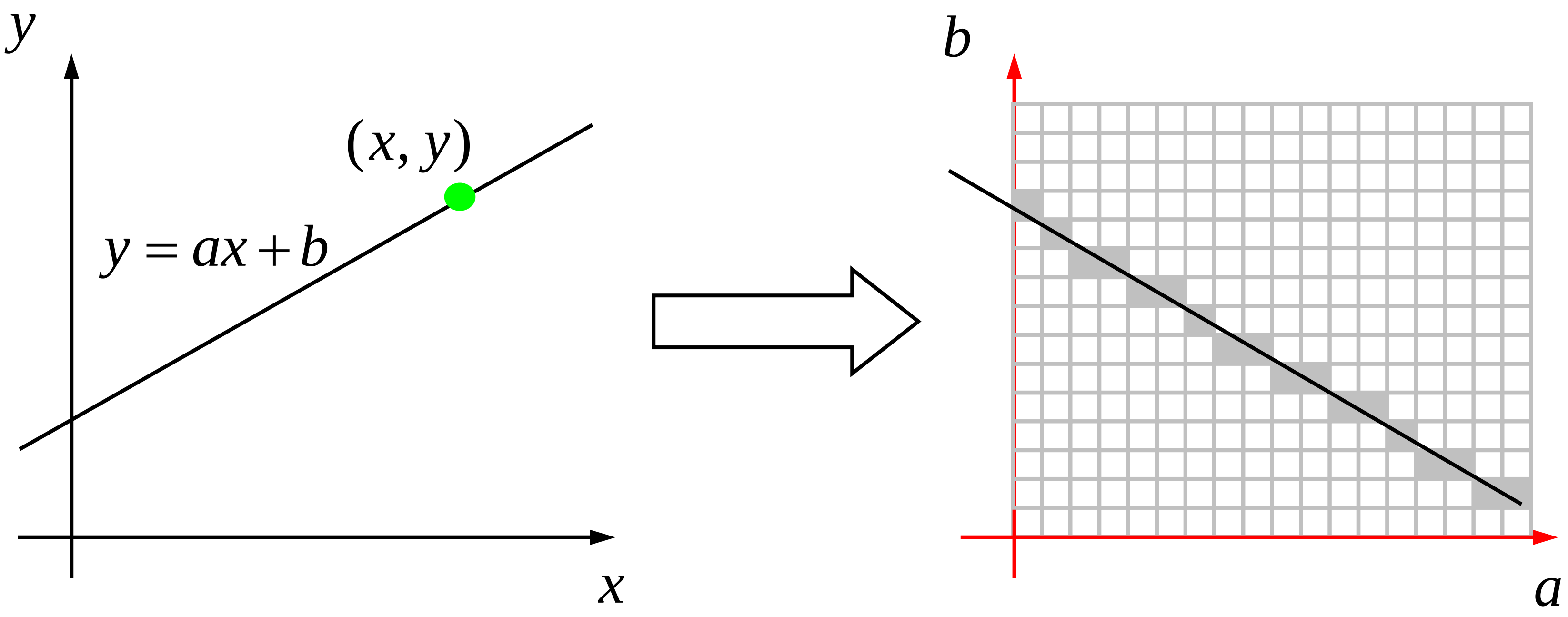
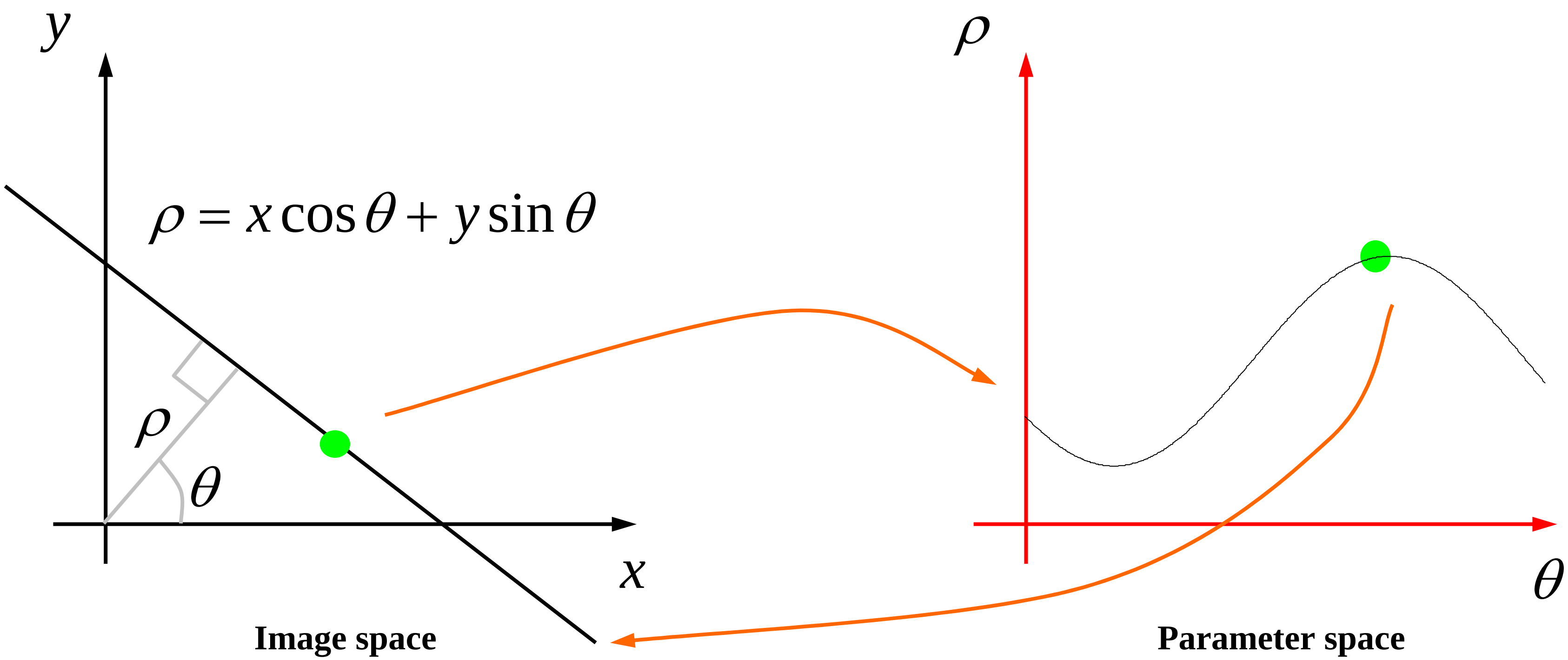


Image space

Parameter space

Edge aggregation - a case study

The Hough transform for line detection



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

Edge aggregation - a case study

The Hough transform for line detection

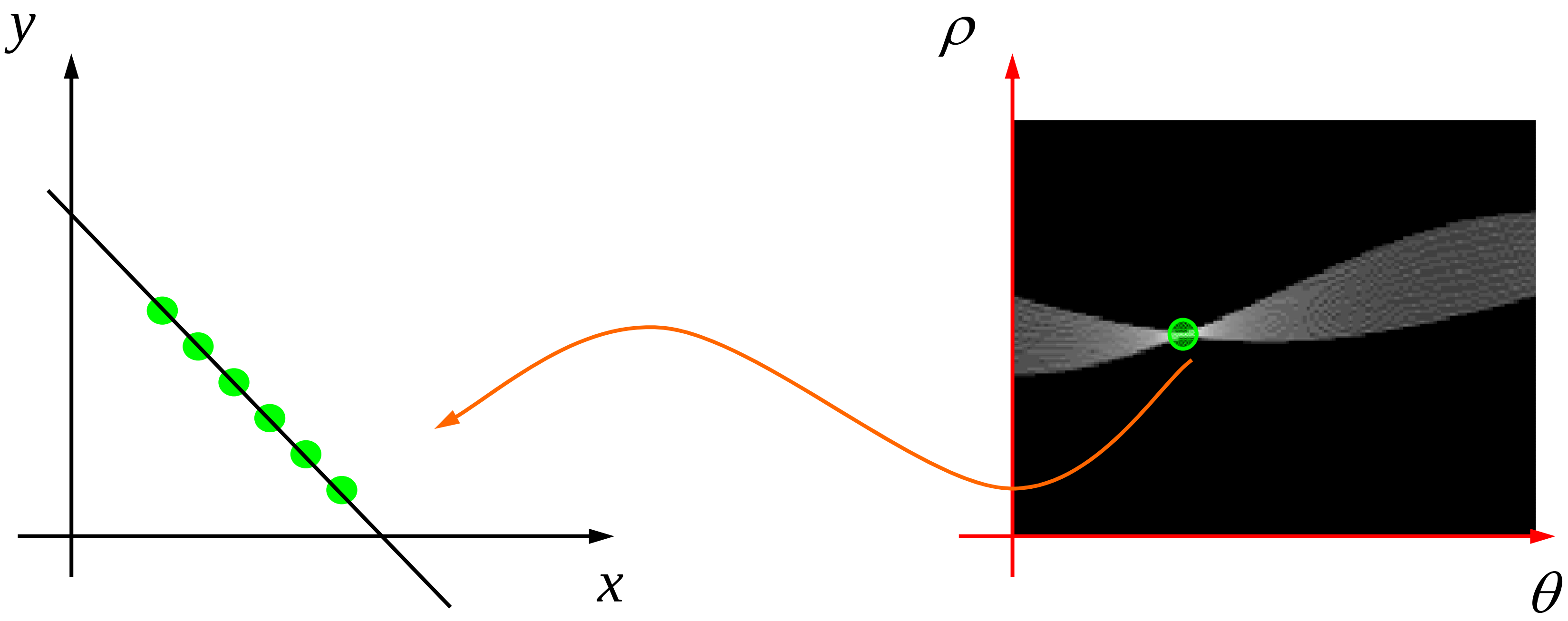
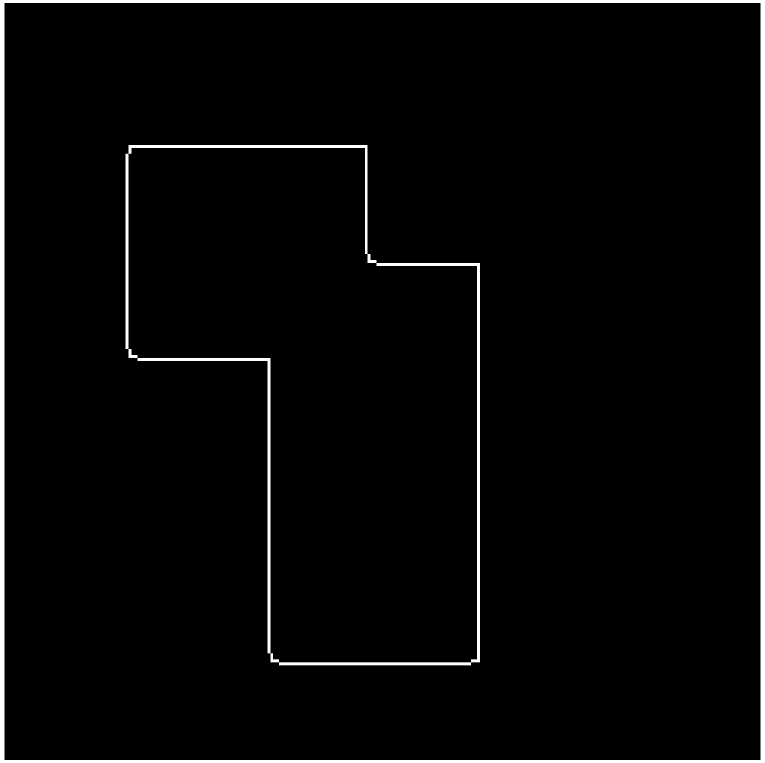


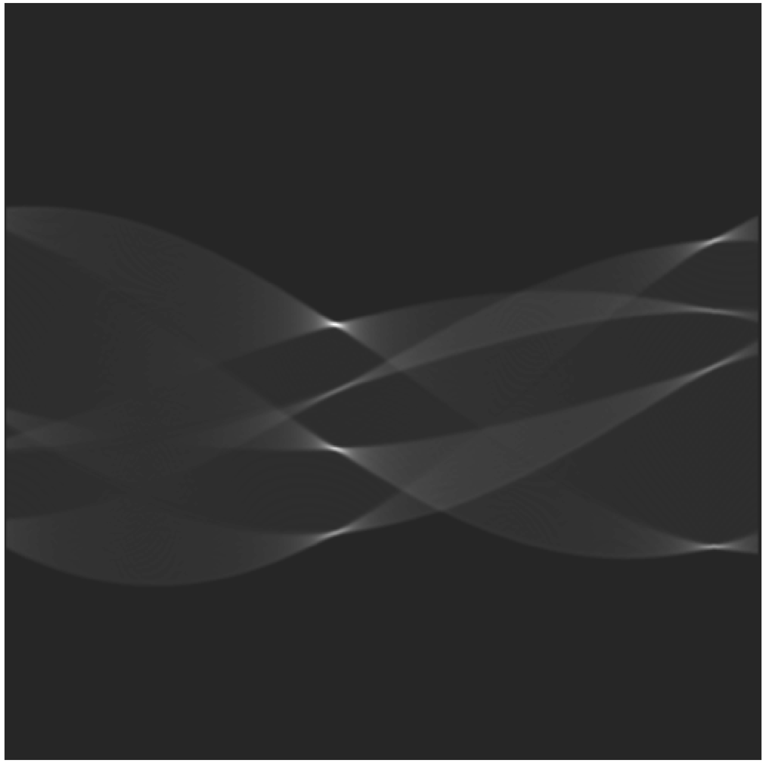
Image space

Parameter space

Edge aggregation - a case study

The Hough transform for line detection





Edge aggregation - a case study

The Hough transform for line detection

possible improvement

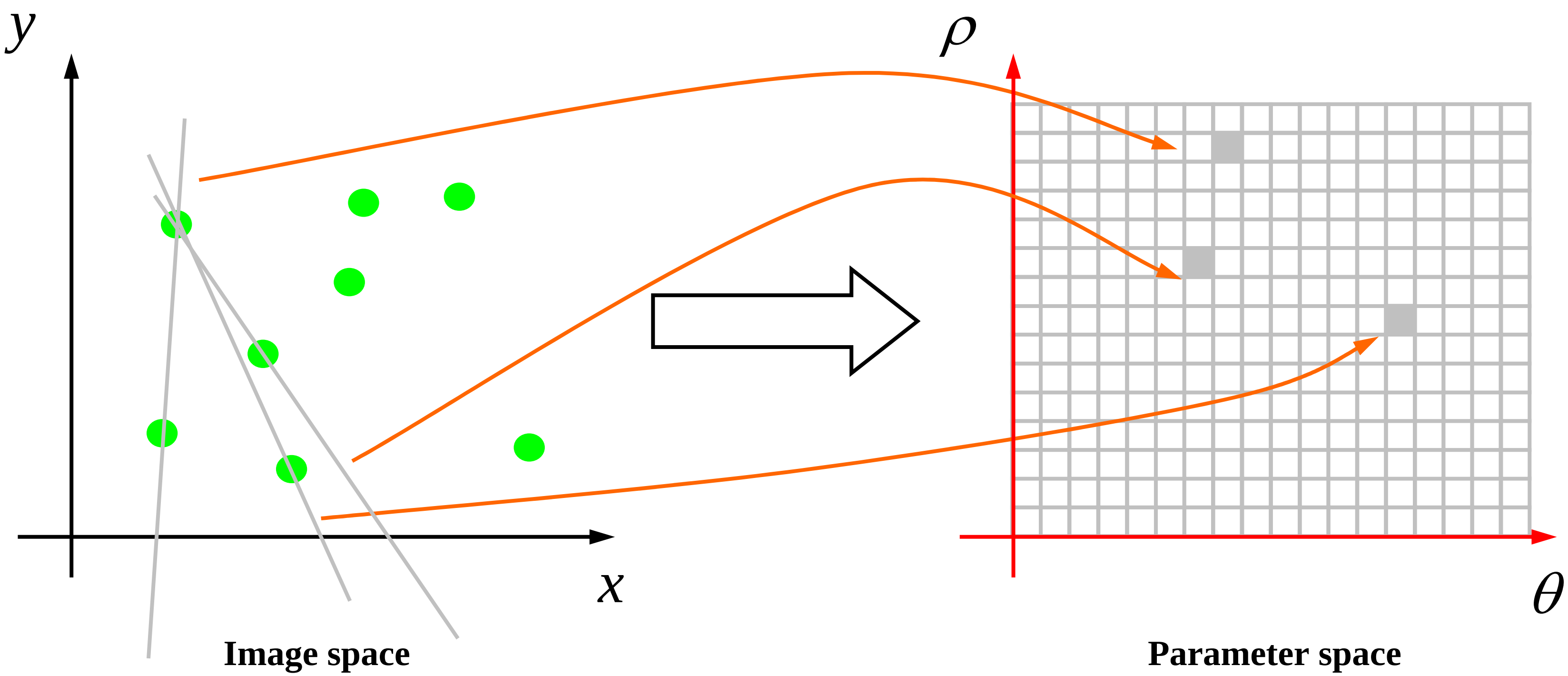


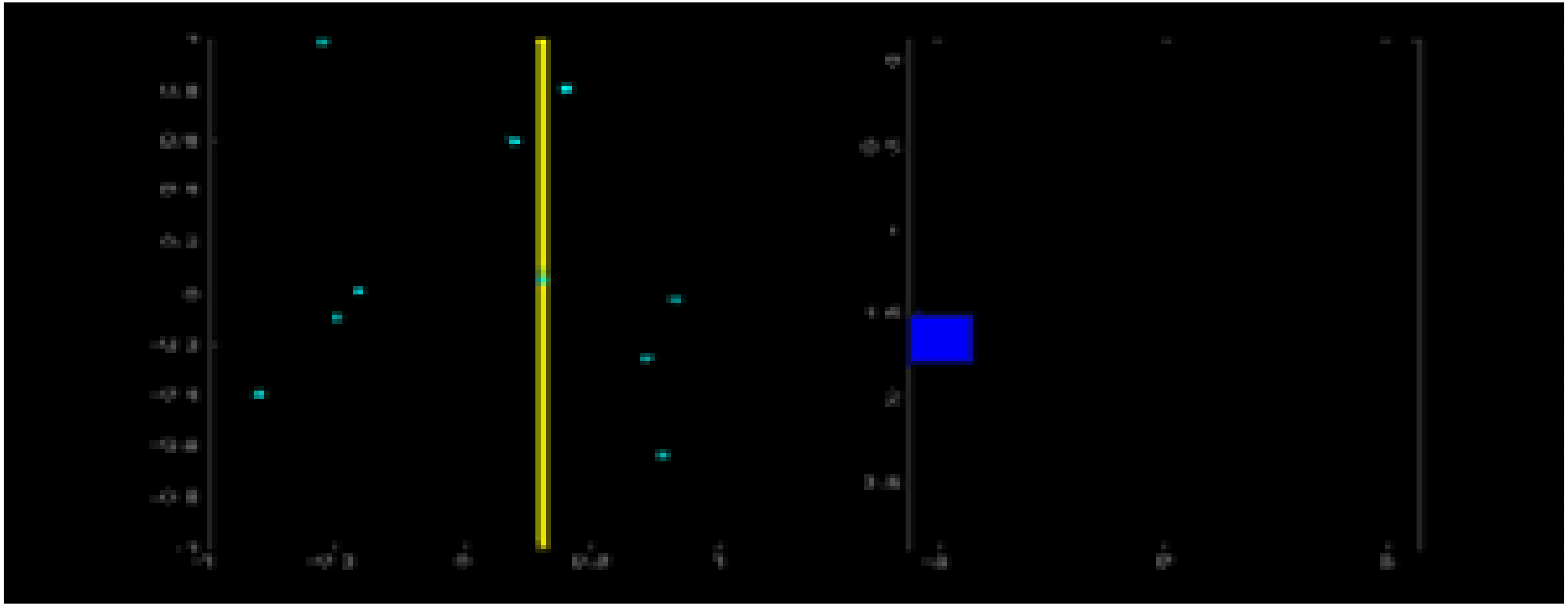
Image space

Parameter space

Edge aggregation - a case study

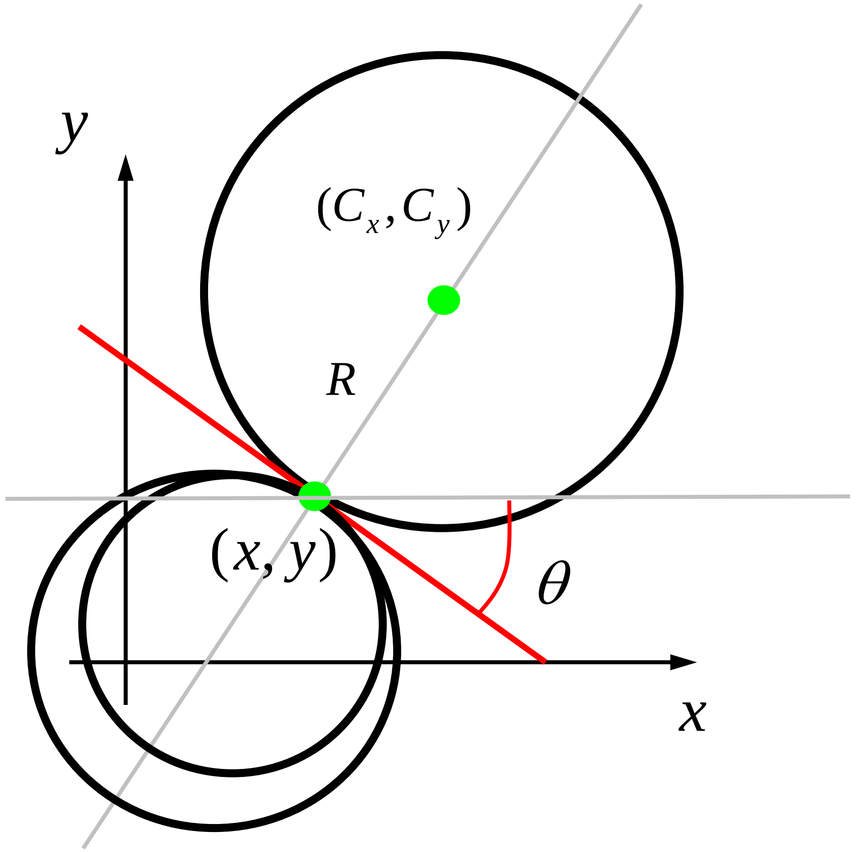
The Hough transform for line detection

possible improvement



Edge aggregation - a case study

Hough transform and circles



$$C_x = x + R\sin\theta$$

$$C_y = y - R\cos\theta$$

Edge aggregation - a case study

Hough transform and circles

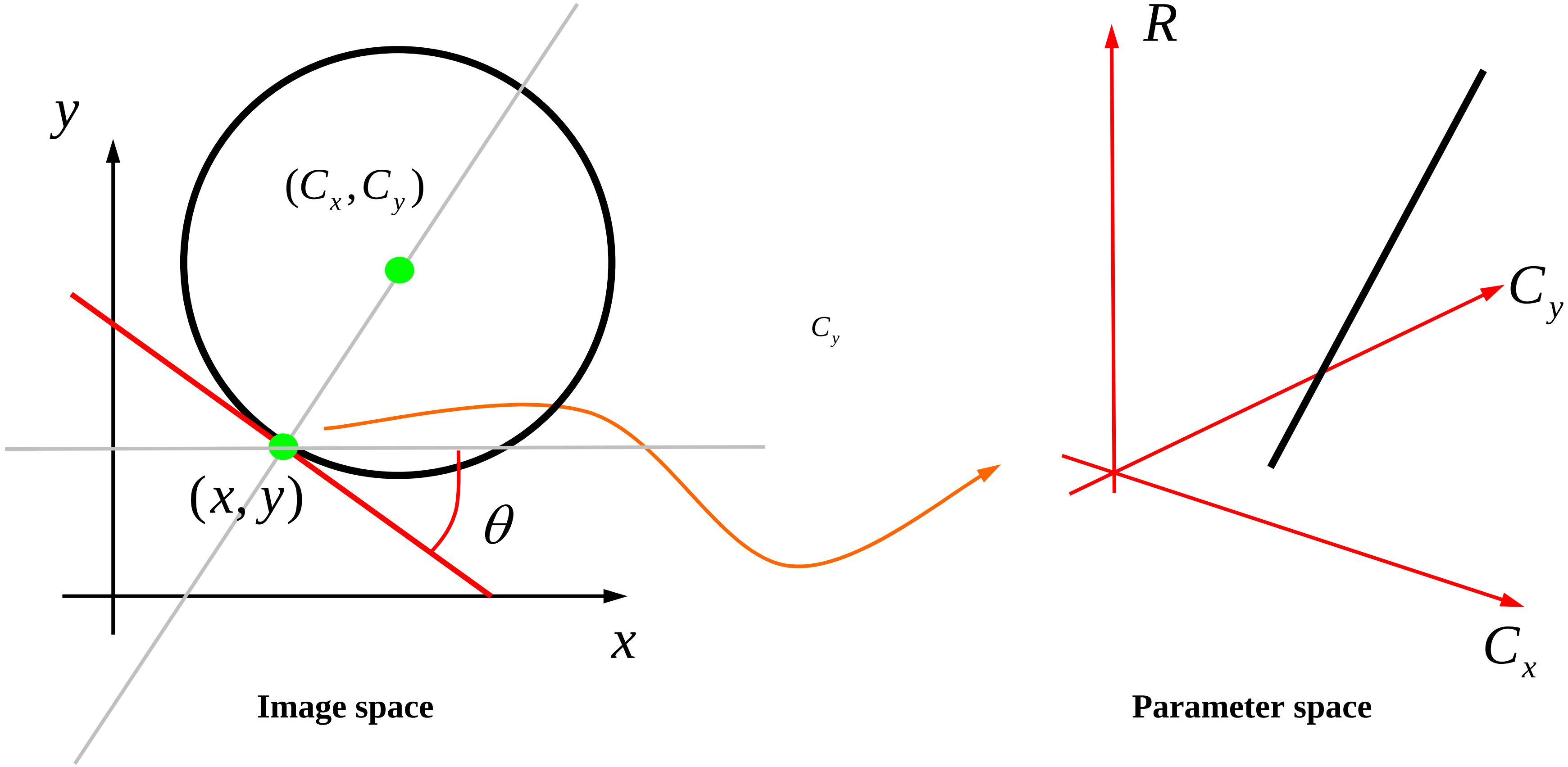


Image space

Parameter space

Edge aggregation - a case study

General Hough transform algorithm

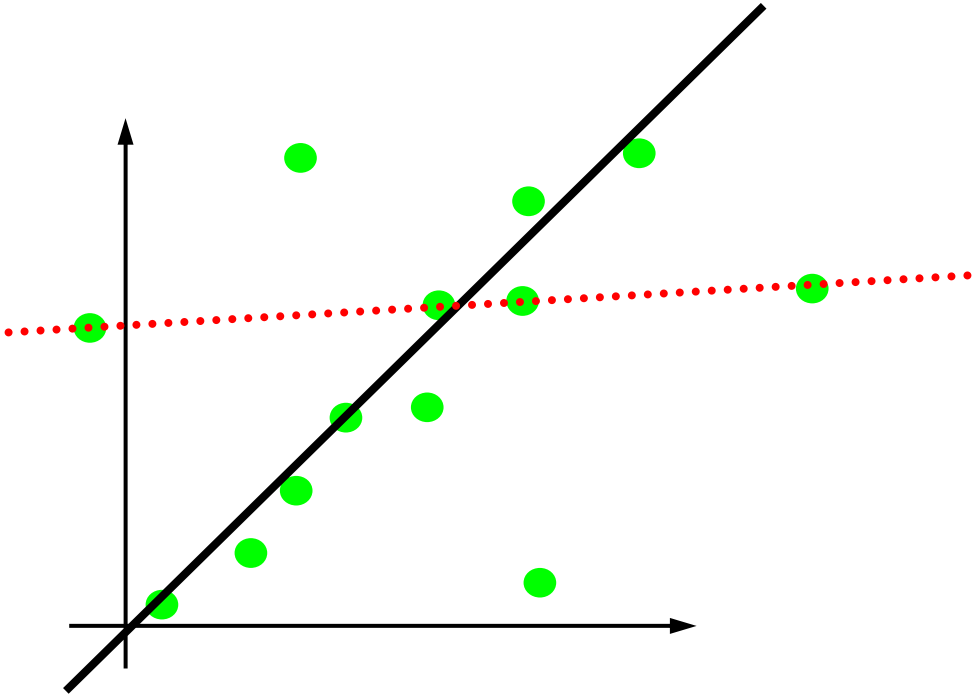
1. **Determine a parametric model for your desired geometrical structure**

$$G(p_1, p_2, \ldots, p_n; x, y) = 0$$

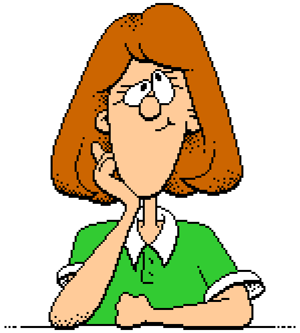
2. **Quantize the parameter space appropriately into bins.**
3. **Initialize each bin to zero.**
4. **For each point (x,y) in the image space, vote (e.g., add 1) to all parameter bins that satisfy the model equation.**
5. **Maxima in bin array correspond to instances of model in the image.**

Edge aggregation - a case study

Hough transform and noisy structures

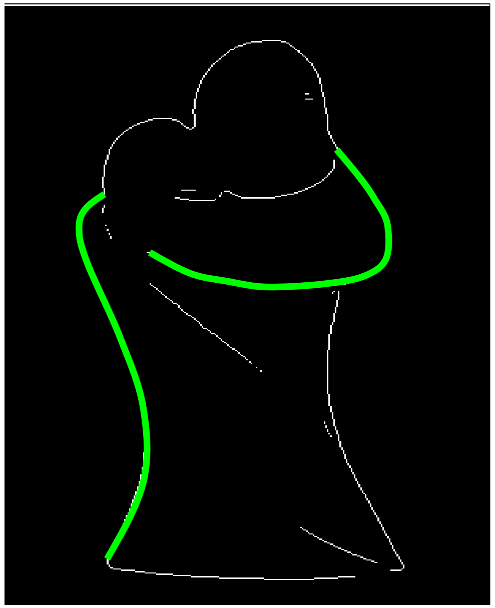


Can we use the Hough transform to detect noisy structures?



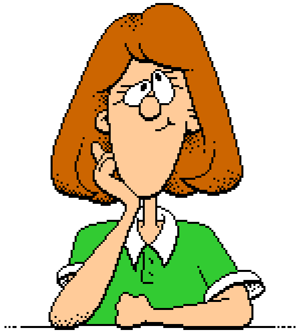
Edge aggregation - a case study

Hough transform and general structures



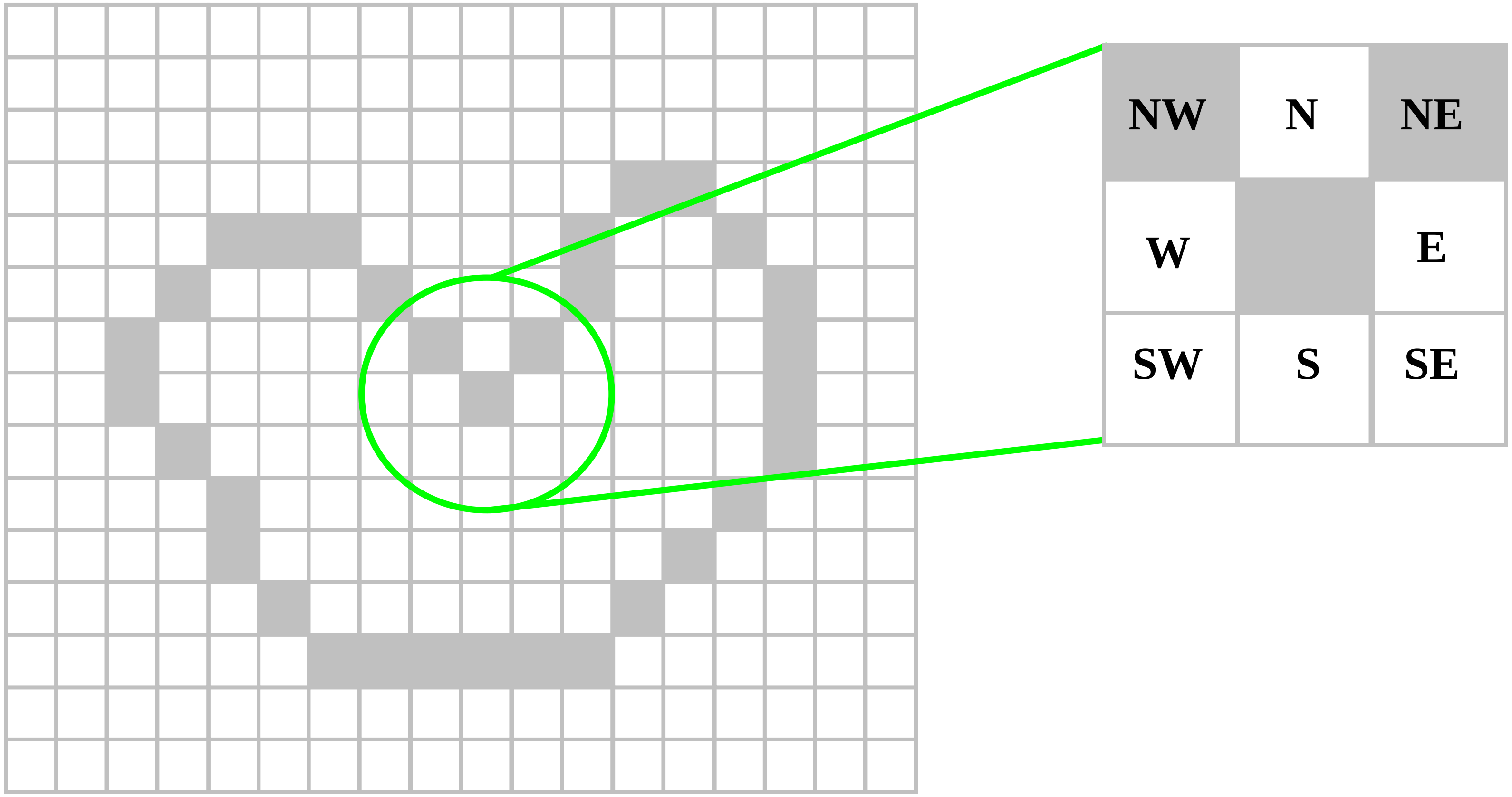
Can we use the Hough transform to detect arbitrary curves?

What parametric model can describe a general curve?



Edge aggregation - a case study

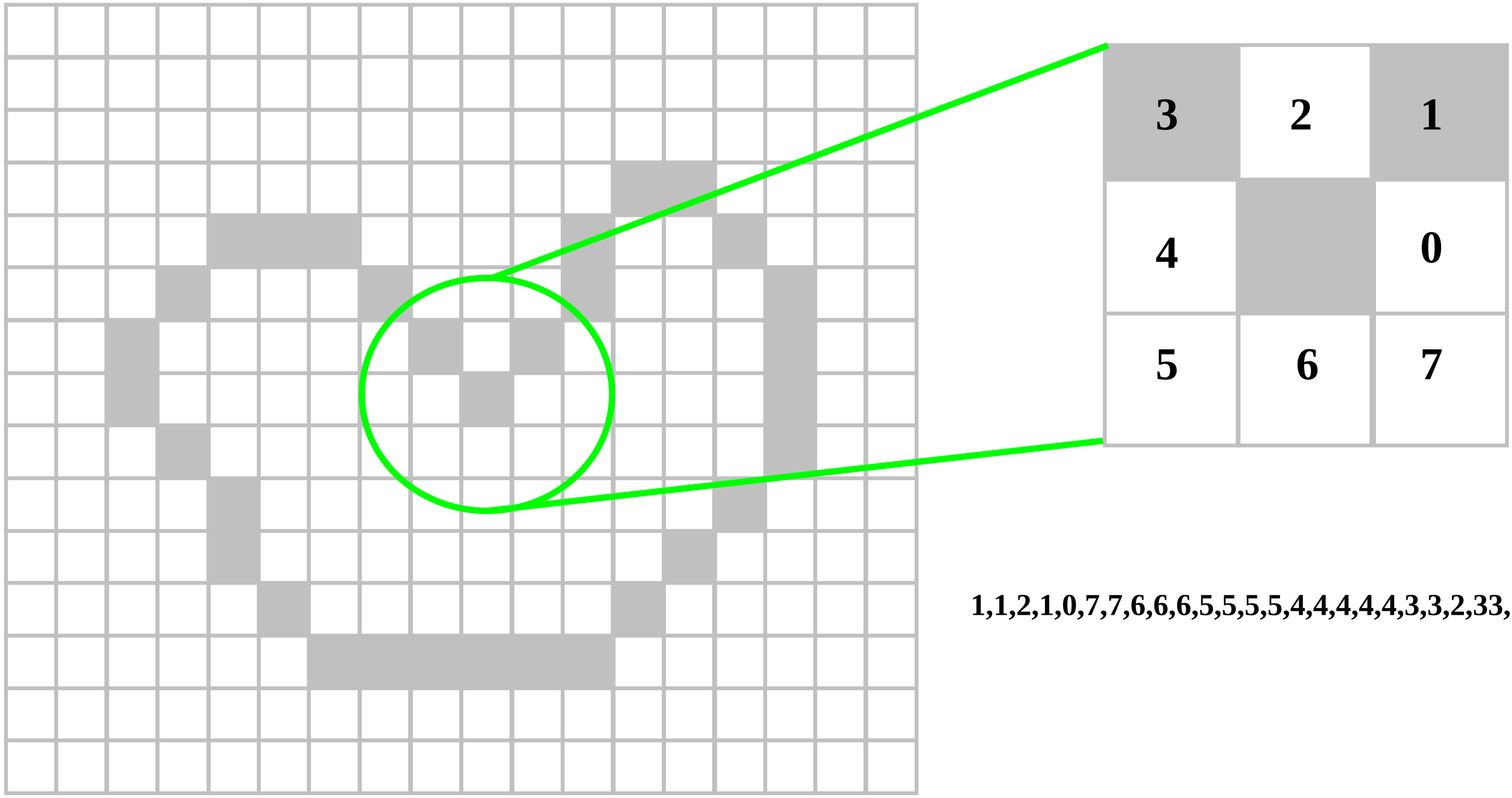
Edge tracing and ordered lists of edges



Absolute chain code

Edge aggregation - a case study

Edge tracing and ordered lists of edges

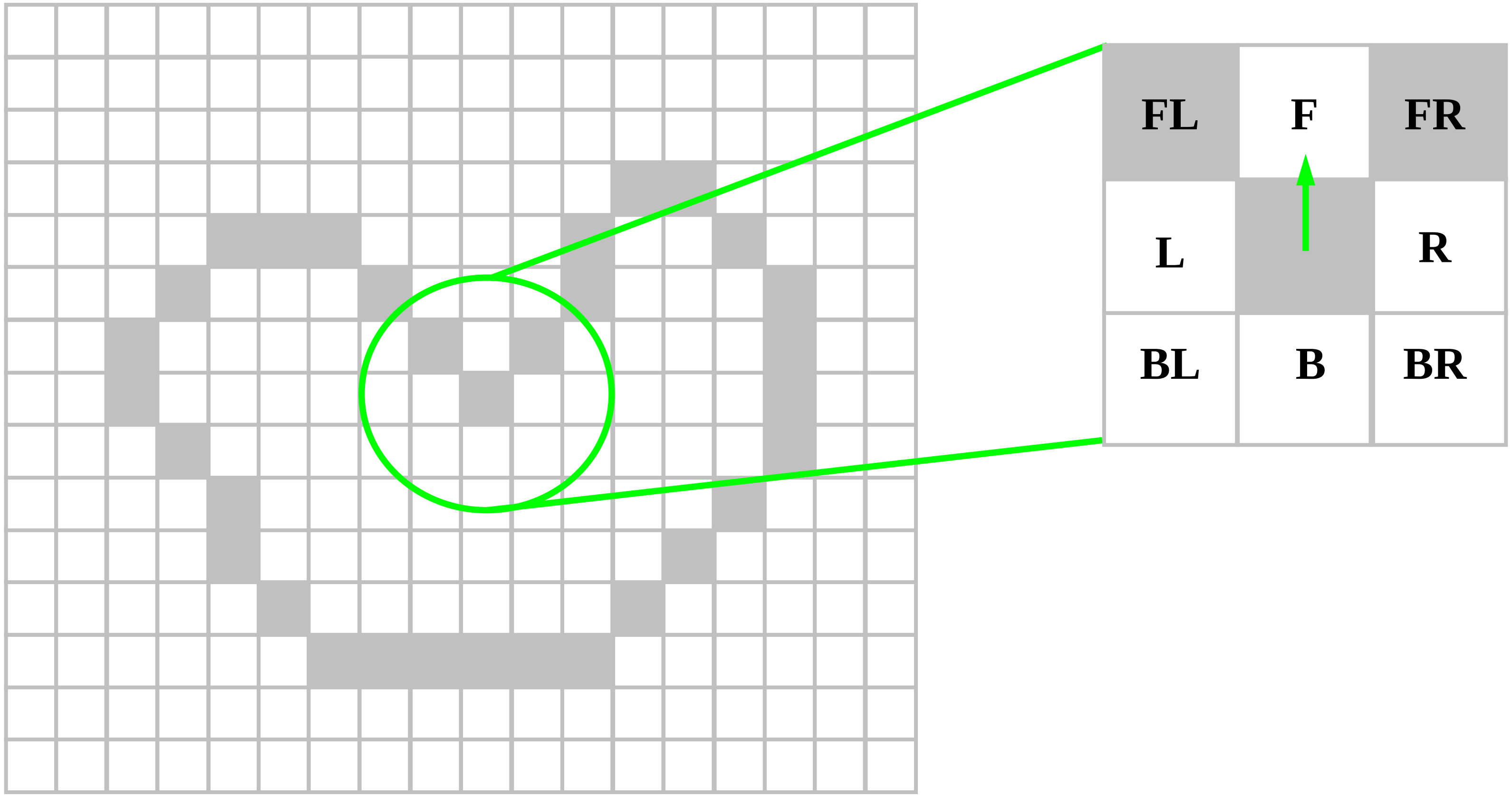


1,1,2,1,0,7,7,6,6,6,5,5,5,5,4,4,4,4,4,3,3,2,33,2,1,1,0,0,7,7,7

Absolute chain code

Edge aggregation - a case study

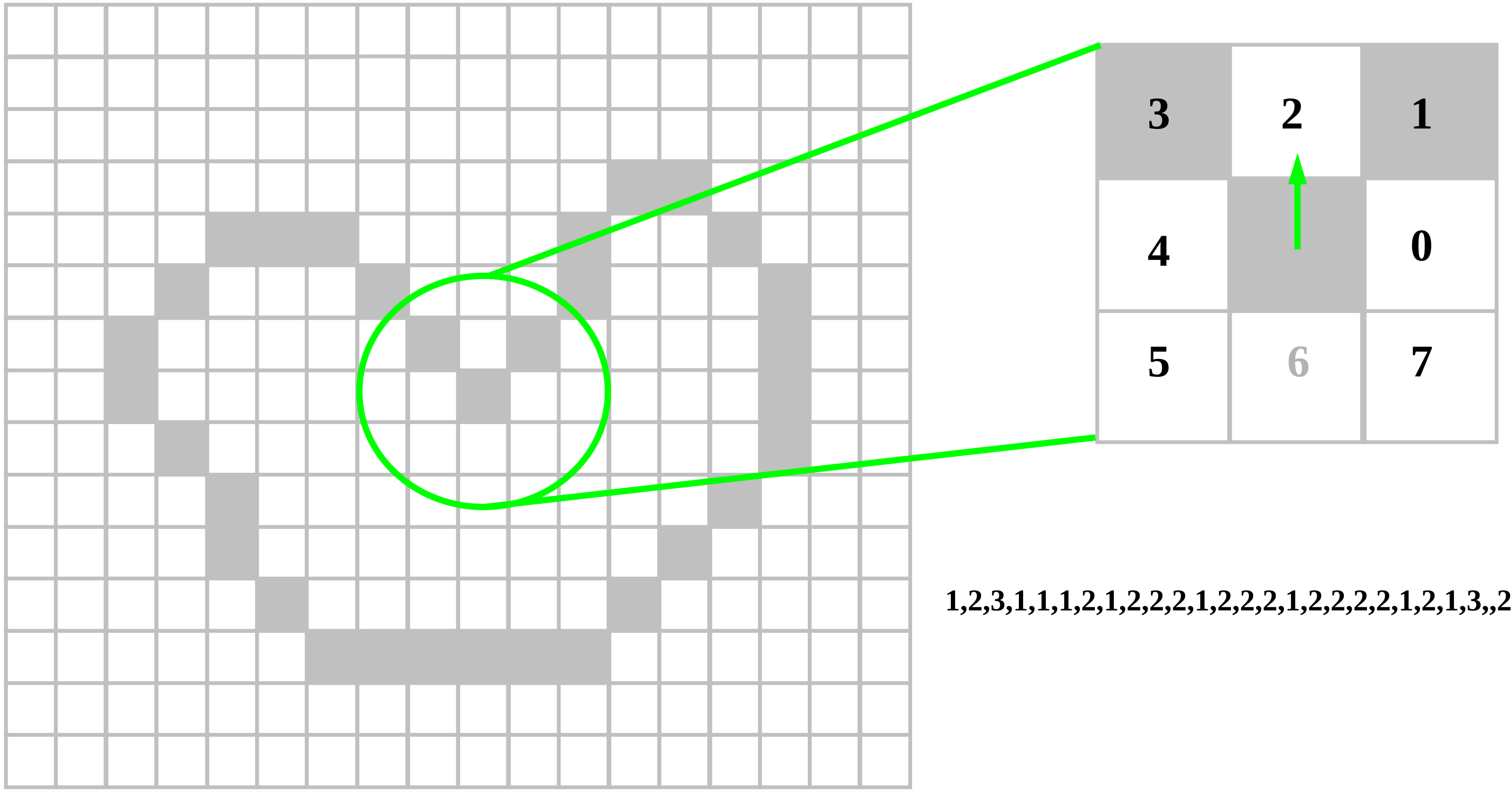
Edge tracing and ordered lists of edges



Relative chain code

Edge aggregation - a case study

Edge tracing and ordered lists of edges

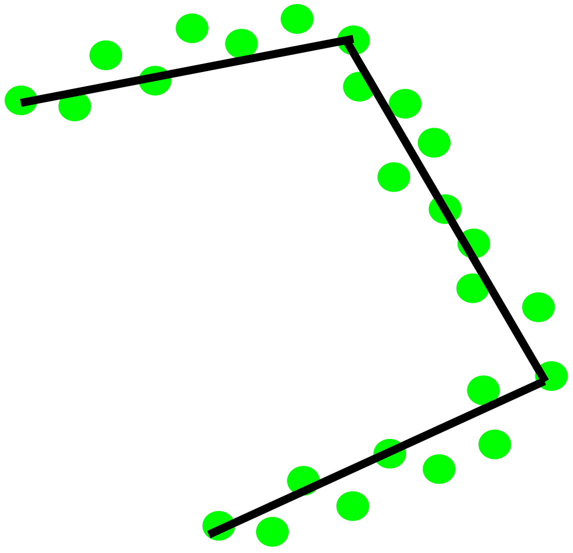


1,2,3,1,1,1,2,1,2,2,2,1,2,2,2,1,2,2,2,2,1,2,1,3,,2,1,1,2,1,2,1,2,2

Relative chain code

Edge aggregation - a case study

Polyline approximation



Given:

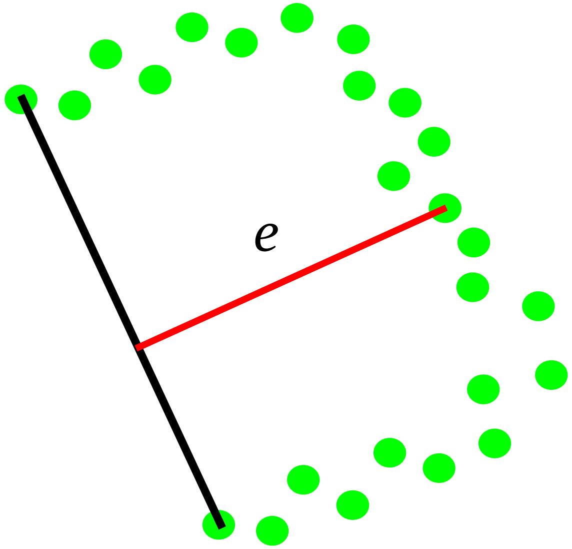
Edge list

Find:

Polygonal approximation that passes no further than distance *d* from any point

Edge aggregation - a case study

Polyline approximation

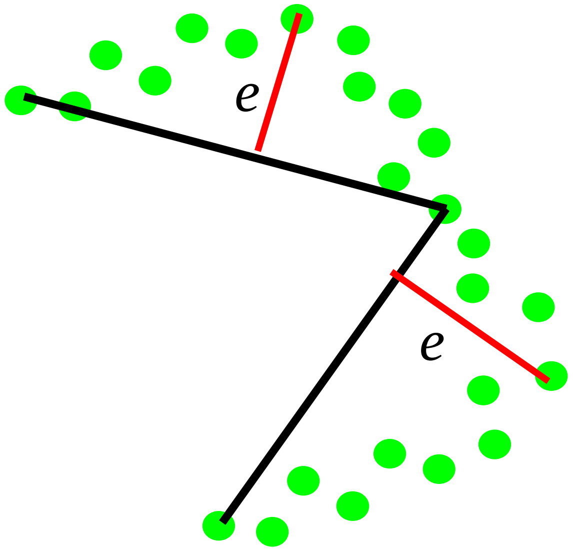


Algorithm:

1. Fit a line between the first and last edge points
2. Split list at point of maximum error
3. Apply recursively until threshold (error<*d*)
4. Merge neighboring segments if error remains within range

Edge aggregation - a case study

Polyline approximation

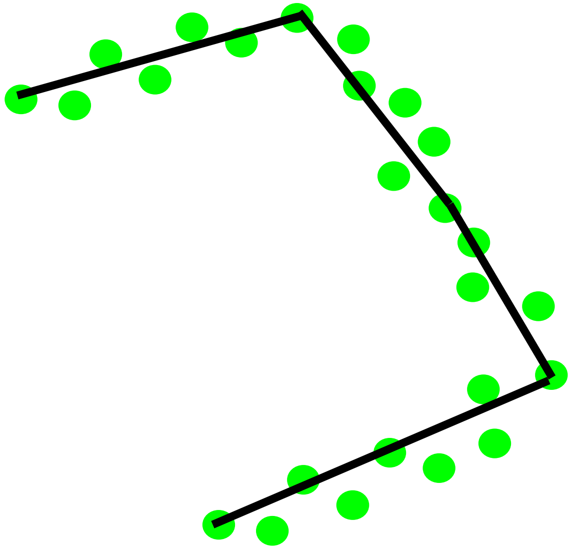


Algorithm:

1. Fit a line between the first and last edge points
2. Split list at point of maximum error
3. Apply recursively until threshold (error<*d*)
4. Merge neighboring segments if error remains within range

Edge aggregation - a case study

Polyline approximation

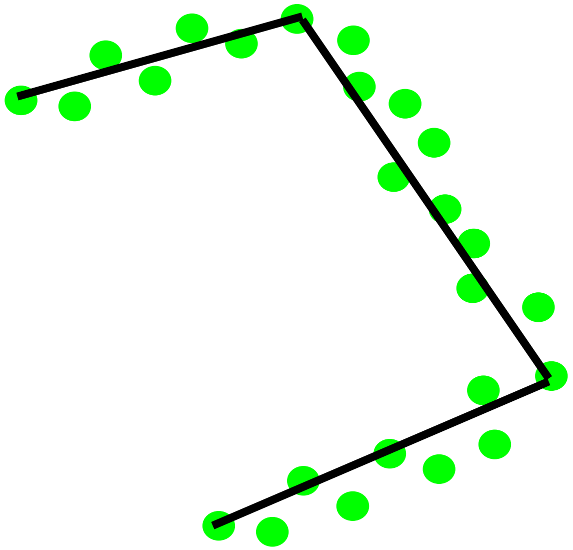


Algorithm:

1. Fit a line between the first and last edge points
2. Split list at point of maximum error
3. Apply recursively until threshold (error<*d*)
4. Merge neighboring segments if error remains within range

Edge aggregation - a case study

Polyline approximation

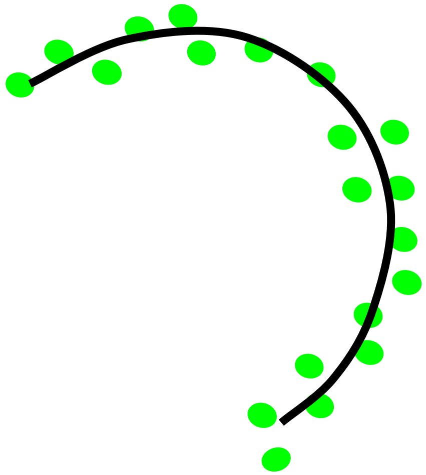


Algorithm:

1. Fit a line between the first and last edge points
2. Split list at point of maximum error
3. Apply recursively until threshold (error<*d*)
4. Merge neighboring segments if error remains within range

Edge aggregation - a case study

Contour approximation via curve fitting



Given:

List of edge points that belong to the same contour

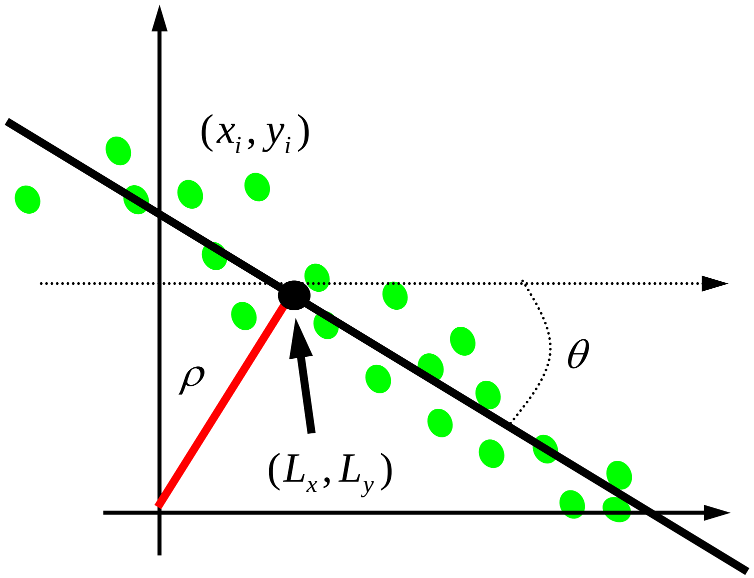
Compute:

Best fit model of a a predefined class *G*

$$\arg\min_{\bar{p}} E\left[\{(x_i, y_i)\} - G(\bar{p}; t)\right]$$

Edge aggregation - a case study

Total regression (fitting) of straight lines



Line representation: $x \sin\theta - y \cos\theta + \rho = 0$

Fit error: $E(\rho,\theta) = \sum_i \left(x_i \sin\theta - y_i \cos\theta + \rho\right)^2$

Normal equations:

$$\frac{\partial E(\rho,\theta)}{\partial \rho} = 0$$

$$\frac{\partial E(\rho,\theta)}{\partial \theta} = 0$$

Solution: $\tan\theta = \dfrac{a}{b+c}$ $\qquad \rho = \bar{y}\cos\theta - \bar{x}\sin\theta$

$$\bar{x} = \frac{1}{n}\sum_i x_i$$

$$\bar{y} = \frac{1}{n}\sum_i y_i$$

$$x'_i = x_i - \bar{x}$$

$$y'_i = y_i - \bar{y}$$

$$a = 2\sum_i x'_i\, y'_i$$

$$b = \sum_i x'^{\,2}_i - \sum_i y'^{\,2}_i$$

$$c = \sqrt{a^2 + b^2}$$