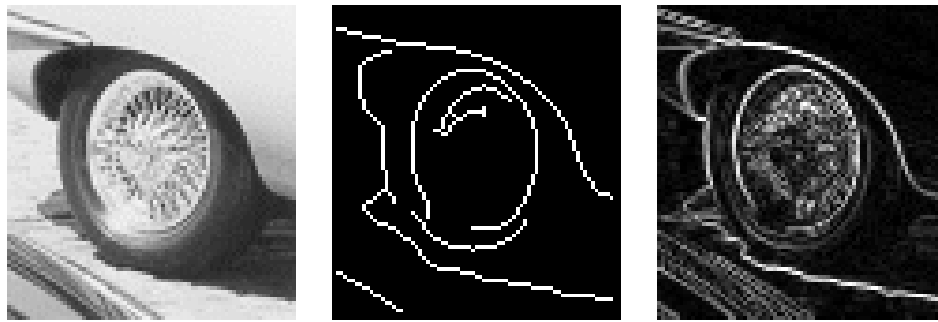


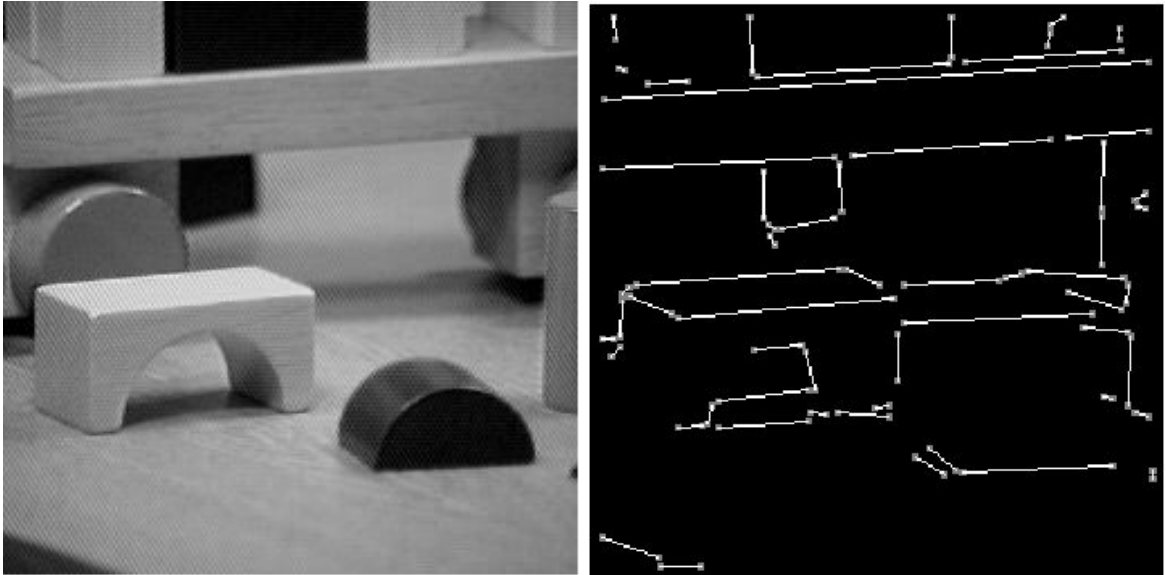
Segmentation of Images: Boundaries

- represent the boundaries of objects
- represent curvilinear objects
- + can yield precise location information
- + can be obtained by edge detection and are more immune to illumination change
- - thin features are easily broken
- - thin features easily have wrong continuations



(Left) image of car wheel; (center) results of Canny operator with $\sigma = 1$; (right) results of Roberts operator. Note how the shadow of the car connects to the tire which connects to the fender: neither the tire nor the spokes are detected well.

Region versus Border Segments



Blocks image (left) and extracted set of straight line segments (right). The line segments were extracted by the ORT (Object Recognition Toolkit) package.

Structures to be Covered

- chains of connected detected edge elements
 1. boundary following (as in Project #1)
 2. Canny Edge Operator
- straight lines (line fitting)
- straight lines (Hough transform)
- circular arcs (Hough transform)
- angles and corners by combining lines
- ribbons by combining “parallel” curves

Canny Edge Operator Tracks Step Edges

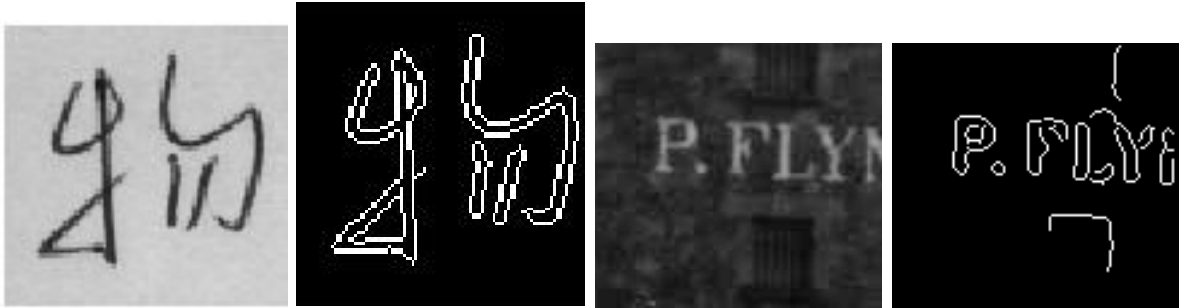
```

I[x, y] : input intensity image;  $\sigma$  : spread used in Gaussian smoothing;
E[x, y] : output binary image;
IS[x, y] : smoothed intensity image;
Mag[x, y] : gradient magnitude; Dir[x, y] : gradient direction;
 $T_{low}$  is low intensity threshold;  $T_{high}$  is high intensity threshold;

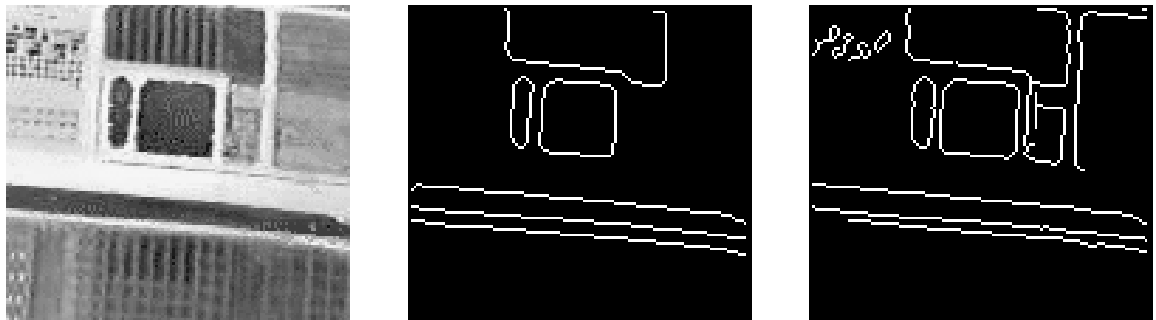
procedure Canny( I[],  $\sigma$ );
{
    IS[] = image I[] smoothed by convolution with Gaussian  $G_{\sigma}(x, y)$ ;
    use Roberts operator to compute Mag[ $x, y$ ] and Dir[ $x, y$ ] from IS[];
    Suppress_Nonmaxima( Mag[], Dir[],  $T_{low}, T_{high}$  );
    Edge_Detect( Mag[],  $T_{low}, T_{high}, \mathbf{E}$ [] );
}
procedure Suppress_Nonmaxima( Mag[], Dir[] );
{
    “zero out all dominated edge elements”
} procedure Edge_Detect( Mag[],  $T_{low}, T_{high}, \mathbf{E}$ [] );
{
    for x := 0 to MaxX - 1;
    for y := 0 to MaxY - 1;
        {
            if (Mag[ $x, y$ ]  $\geq T_{high}$ ) then Follow_Edge(  $x, y, \mathbf{Mag}$ [],  $T_{low}, T_{high}, \mathbf{E}$ [] );
        } ;
}
procedure Follow_Edge(  $x, y, \mathbf{Mag}$ [],  $T_{low}, T_{high}, \mathbf{E}$ [] );
{
    E[ $x, y$ ] := 1;
    while Mag[ $u, v$ ]  $> T_{low}$  for some 8-neighbor [ $u, v$ ] of [ $x, y$ ]
        {
            E[ $u, v$ ] := 1;
            [ $x, y$ ] := [ $u, v$ ];
        } ;
}

```

Some Results of Canny Operator



Identifying regions corresponding to symbols on surfaces is often easy because they are created with good contrast: (left set) character carefully written with ink on paper; (right set) weathered signage on a brick wall.



Contours from an aerial image of farm fields defined using the Canny operator with $\sigma = 2$ and $\sigma = 1$ respectively. Note that five major structures are well represented – three fields and two straight horizontal bands in the bottom of the image (a canal and a road alongside it).

Aggregating Neighboring Edgels into Curves

	1	2	3	4	5
1	1	0	0	0	1
2	0	1	0	1	0
3	0	0	1	0	0
4	0	0	1	0	0
5	0	0	1	1	1

Tracking algorithm must perform the following tasks:

1. starting a new segment
2. adding an interior pixel to a segment
3. ending a segment
4. finding a junction
5. finding a corner

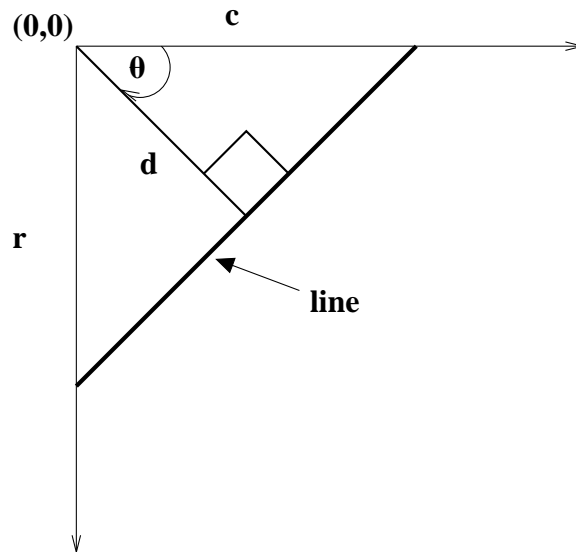
Segment ID	Length	List
1	3	(1,1)(2,2)(3,3)
2	3	(1,5)(2,4)(3,3)
3	3	(3,3)(4,3)(5,3)
4	3	(5,3)(5,4)(5,5)

Output of the *edge_track* procedure on the image, assuming the point (5,3) is judged to be a corner point. If corner points are not used to terminate segments, then segment 3 would have length 5 and list ((3,3)(4,3)(5,3)(5,4)(5,5)).

Hough Transform (Clustering)

- gather evidence in image neighborhoods — edge element
- local evidence supports some global shapes — line or circle
- global shape parameterized by a parameter set $\alpha = [a_1, \dots, a_d]$
 - line: $\alpha = [d, \theta]$ are the polar coordinates of the line through $[x, y]$ with normal direction θ
 - circle: parameterized by $\alpha = [x_c, y_c, radius]$
- edge element *votes for* particular parameter values
- global shape detected by many votes for same parameter values

evidence for straight line segment

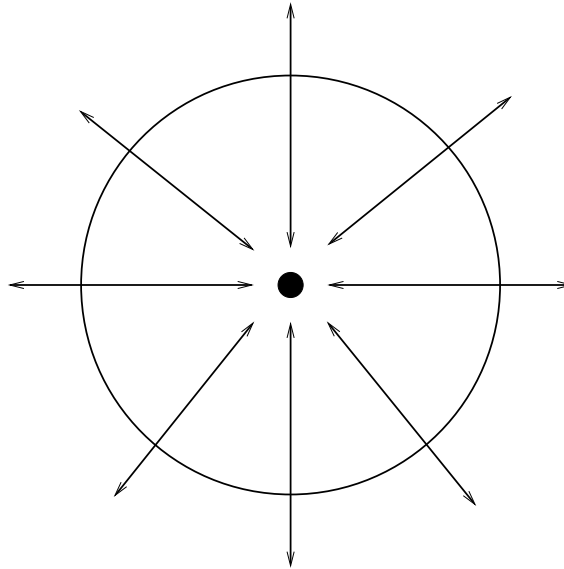


The parameters d and θ used in the equation $d = -r \sin \theta + c \cos \theta$ of a straight line. d is the shortest distance to the origin along a perpendicular to the line, of course.

Hough Transform for Line Detection

1. map each edgel: $[x, y, Mag, Dir = \theta] \rightarrow [\rho, \theta]$
 where $[\rho, \theta]$ are the polar coordinates of the line through $[x, y]$
2. detect clusters in parameter space $[\rho, \theta]$
3. each cluster provides significant evidence of a straight line or edge

accumulating evidence for a circular arc



The inward pointing gradients of edgels on the circle will provide evidence for the center of the circle.

1. circle is parameterized by $\alpha = [x_c, y_c, radius]$
2. for each edgel: $[x, y, Mag, Dir]$ with $Mag \geq t$
 - (a) compute the line $[\rho, \theta]$ **perpendicular** to the edgel and through $[x, y]$
 - (b) for each point $[u, v]$ on this line, vote $[u, v, r]$, where r is the distance between $[x, y]$ and $[u, v]$
3. each cluster in the $[x_c, y_c, r]$ space gives evidence of a circular arc

Straight Line Hough Transform

Accumulate the lines in gray-tone image S to accumulator A.

S[**R**, **C**] is the input gray-tone image.

NLINES is the number of rows in the image.

NPIXELS is the number of pixels per row.

A[**DQ**, **THETAQ**] is the accumulator array.

DQ is the quantized distance from a line to the origin.

THETAQ is the quantized angle of the normal to the line.

```

procedure accumulate_lines(S,A);
{
  A := 0;
  PTLIST := NIL;
  for R := 1 to NLINES
    for C := 1 to NPIXELS
      {
        DR := row_gradient(S,R,C);
        DC := col_gradient(S,R,C);
        GMAG := gradient(DR,DC);
        if GMAG > gradient_threshold
          {
            THETA := atan2(DR,DC);
            THETAQ := quantize_angle(THETA);
            D := abs(C*cos(THETAQ) - R*sin(THETAQ));
            DQ := quantize_distance(D);
            A[DQ,THETAQ] := A[DQ,THETAQ]+GMAG;
            PTLIST(DQ,THETAQ) := append(PTLIST(DQ,THETAQ),[R,C])
          }
      }
}

```

Circular Hough Transform

Accumulate the circles in gray-tone image S to accumulator A .

$S[R, C]$ is the input gray-tone image.

$NLINES$ is the number of rows in the image.

$NPIXELS$ is the number of pixels per row.

$A[R, C, RAD]$ is the accumulator array.

R is the row index of the circle center.

C is the column index of the circle center.

RAD is the radius of the circle.

```

procedure accumulate_circles(S,A);
{
  A := 0;
  PTLIST := 0;
  for R := 1 to NLINES
    for C := 1 to NPIXELS
      for each possible value RAD of radius
        {
          THETA := compute_theta(S,R,C,RAD);
          R0 := R - RAD*cos(THETA);
          C0 := C + RAD*sin(THETA);
          A[R0,C0,RAD] := A[R0,C0,RAD]+1;
          PTLIST(R0,C0,RAD) := append(PTLIST(R0,C0,RAD),[R,C])
        }
    }
}

```

Detecting Lines of Symbols as Words

Algorithm due to R. Kasturi (see auxiliary slides)

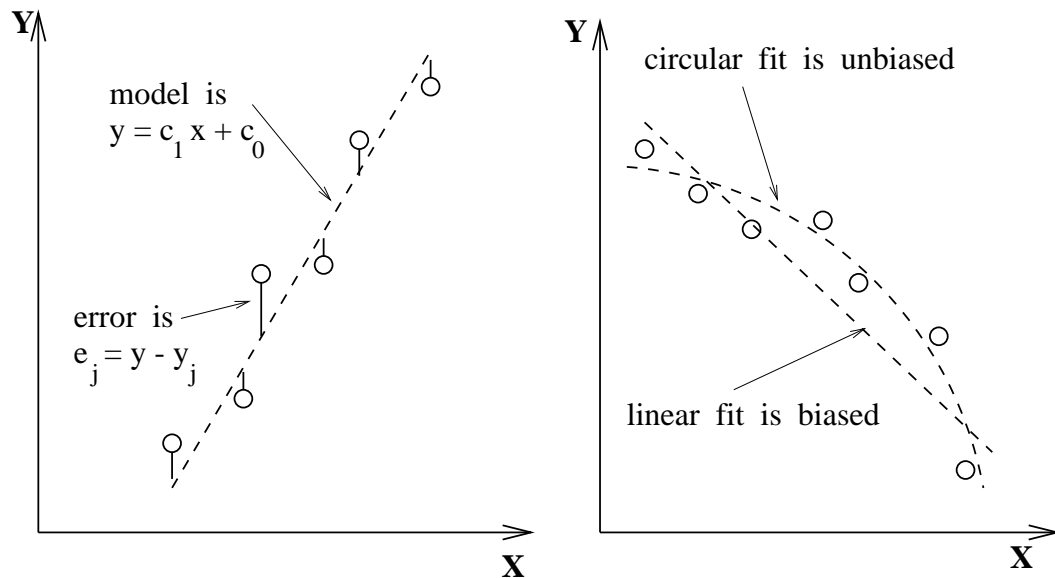
1. binarize an image of a page or map to get $B[x, y]$
2. extract symbols from $B[x, y]$ using connected components
3. for each symbol centroid $B[\bar{x}, \bar{y}]$ set $S[x, y] = 255$
4. Hough transform of $S[x, y]$ detects symbols on same line
5. O’Gorman/Clowes version gives all points on detected line
6. for each point, extract from $B[x, y]$ the bounding box
7. rotate all symbols to 0 baseline for possible recognition

Line Detectors in Animal Visual Systems

- hypothesis supported by experiments with animals
- edgels detected by LOG or Gabor filters
- linear arrangements of detecting cells integrated at higher level
- neighboring line detecting cells have similar ρ, θ
- see papers by Hubel and Wiesel in the 1970's

Fitting Models to Segments

- we have a candidate mathematical shape model
 - line: $y = c_1x + c_0$ or $f(x, y) = 0$
 - parabola: $y = c_2x^2 + c_1x + c_0$ or $g(x, y) = 0$
 - circle: $(x - x_c)^2 + (y - y_c)^2 = r^2$
- we have data points $\{ (x_j, y_j)_{j=1, \dots, n} \}$
- need to find the **best fitting** model to the data
- need to have criteria for *good enough* and *better*
- least squares theory provides practical techniques



Fit of model $y = f(x)$ to six data points; (right) competing straight line and circular models: the signs of the residual errors show that the line fit is biased and the circular fit is unbiased.

Least Squares Fit Criteria

1 **DEFINITION Least-Squares Error Criteria:** *The measure of how well a model $y = f(x)$ fits a set of n observations $\{(x_j, y_j), j = 1, n\}$ is*

$$LSE = \sum_{j=1}^n (f(x_j) - y_j)^2$$

The best model $y = f(x)$ is the model with the parameters minimizing this criteria.

Least-squares fit of data generated using $y = 3x - 7$ plus noise gives fitted model $y = 2.971x - 6.962$.

Data Pts (x_j, y_j)	(0.0, -6.8)	(1.0, -4.1)	(2.0, -1.1)	(3.0, 1.8)	(4.0, 5.1)	(5.0, 7.9)
Residuals $y - y_j$:	-0.162	0.110	0.081	0.152	-0.176	-0.005

Closed Form Solution for Line Fit

- expression for least squares error

$$LSE = \varepsilon(c_1, c_0) = \sum_{j=1}^n (c_1 x_j + c_0 - y_j)^2 \quad (1)$$

- error function ε is a smooth non-negative function of the two parameters c_1 and c_0 and will have a global minimum at the point (c_1, c_0) where $\partial\varepsilon/\partial c_1 = 0$ and $\partial\varepsilon/\partial c_0 = 0$

$$\partial\varepsilon/\partial c_1 = \sum_{j=1}^n 2(c_1 x_j + c_0 - y_j) x_j = 0 \quad (2)$$

$$= 2\left(\sum_{j=1}^n x_j^2\right)c_1 + 2\left(\sum_{j=1}^n x_j\right)c_0 - 2\sum_{j=1}^n x_j y_j \quad (3)$$

$$\partial\varepsilon/\partial c_0 = \sum_{j=1}^n 2(c_1 x_j + c_0 - y_j) = 0 \quad (4)$$

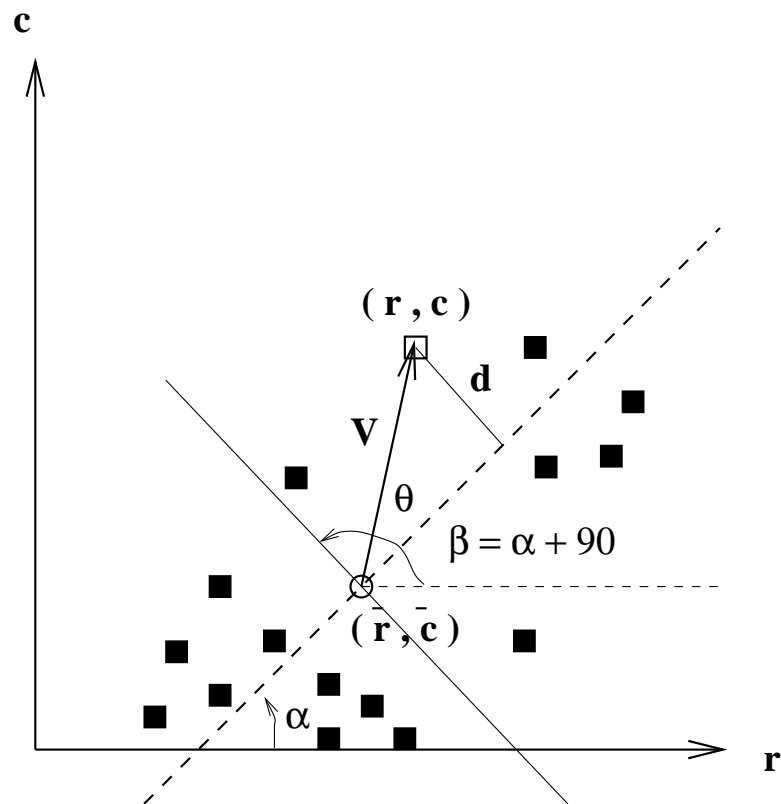
$$= 2\left(\sum_{j=1}^n x_j\right)c_1 + 2\sum_{j=1}^n c_0 - 2\sum_{j=1}^n y_j \quad (5)$$

- These *normal equations* are nicely represented in matrix form.

$$\begin{bmatrix} \sum_{j=1}^n x_j^2 & \sum_{j=1}^n x_j \\ \sum_{j=1}^n x_j & \sum_{j=1}^n 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^n x_j y_j \\ \sum_{j=1}^n y_j \end{bmatrix} \quad (6)$$

- pattern is general for fitting polynomials of all degrees

Axis of Least Inertia Might Be Better



- axis of least inertia rotates with the data/object
- y need not be a function of x
- linearity detected by low moment about axis; high perpendicular moment
- line has *intrinsic dimensionality* = 1 plus noise

Perimeter Set \longrightarrow Polygon

Ramer's Algorithm:

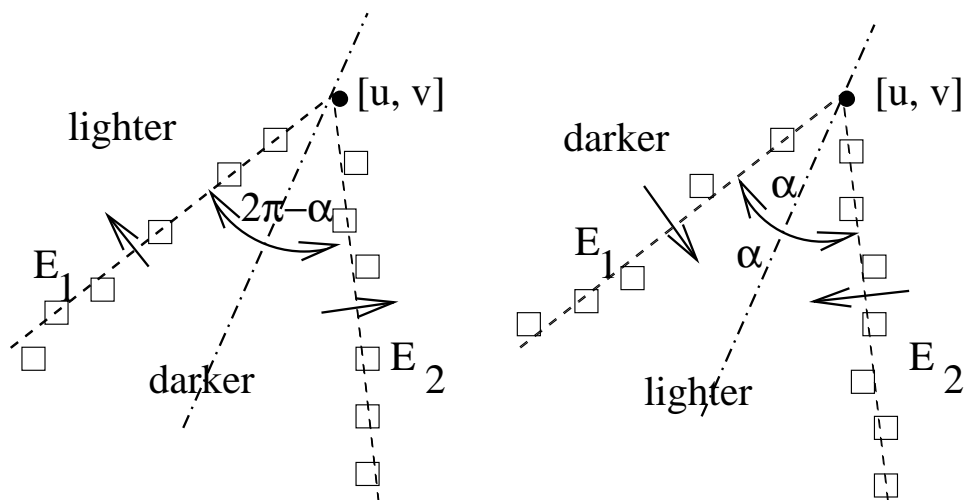
input perimeter set and tolerance

output polygon fitting perimeter set within tolerance

1. input ordered perimeter set $P = \langle (r_o, c_o), \dots, (r_{K-1}, c_{K-1}) \rangle$
 2. input tolerance tol
 3. compute the equation of the line from the first to last point
 4. compute the distance of each perimeter point from the line
 5. if all points are within tol of the line return
 6. else recurse on the two halves of the perimeter set
- + creates open or closed polygons
- requires all data to be in memory (other algs don't)

Angles, Corners and Ribbons

- **Angles and corners** are formed by coterminating line segments
- **Ribbons** are projections of **generalized cylinders**
 - electric cord or rope
 - road or river
 - bottle, lamp or lamppost



Corners are detected as pairs of detected edge segments appropriately related.

Detecting Straight Ribbons

2 DEFINITION *A ribbon is an elongated region that is approximately symmetrical about its major axis. Often, but not always, the edges of a ribbon contrast symmetrically with its background.*



Region of an image of a house showing a downspout and strong shadows; (center) the highest 10% gradient magnitudes computed by the Prewitt 3x3 operator; (right) sketch of ribbons and corners evident. The projection causes incorrect connections.