Finding region boundaries
Object boundaries

- How to detect?
- How to represent?
- Problems relative to region/area segmentation
Advantages

- Contrast often more reliable under varying lighting
- Boundaries give precise location
- Boundaries have useful shape information
- Boundary representation is sparse
- Humans use edge information
Problems

- Detection weakens at boundary corners
- Topology of result is imperfect
Topics

- Aggregating chains of boundary pixels
- Line and curve fitting
- Angles, sides, vertices
- Ribbons
- Imperfect graphs
Canny edge detector/tracker

- Detect, then follow high gradient boundary pixels
- Track along a “thin” edge
- Bridge across weaker contrast
Canny algorithm data

$I[x, y]$ : 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[], σ);
{
    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}, E[] );
}
Create thin edge response by suppressing weaker neighbors

- Find neighbors in direction of the gradient (across the edge)
- If neighbor response is weaker, delete it.

```plaintext
procedure Suppress_Nonmaxima( Mag[], Dir[] );
{
    “zero out all dominated edge elements”
}
Begin tracking only at high gradient points

```plaintext
procedure Edge_Detect( Mag[], T_low, T_high, E[] );
{
    for x := 0 to MaxX - 1;
    for y := 0 to MaxY - 1;
    {
        if (Mag[x, y] ≥ T_high) then Follow_Edge( x, y, Mag[], T_low, T_high, E[] );
    }
}

procedure Follow_Edge( x, y, Mag[], T_low, T_high, 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];
    }
}
```

MSU/CSE Fall 2016
Results with two spreads

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).
Notes/Variations

- Very few parameters needed
- Can use multiple spreads and correlate the results in multiple edge images
- Can carefully manage the use of low and high thresholds, perhaps depending on size and quality of the current track
Aggregating pixels into curves

Producing a discrete data structure with meaningful parts.
Tracking tasks

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

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

Canny algorithm does some of these.
Possible output from tracker

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Segment ID</th>
<th>Length</th>
<th>List</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>(1,1)(2,2)(3,3)</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>(1,5)(2,4)(3,3)</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>(3,3)(4,3)(5,3)</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>(5,3)(5,4)(5,5)</td>
</tr>
</tbody>
</table>
The Hough Transform

Detecting a straight line segment as a cluster in rho-theta space.
(Can do circles or ellipses too)
Voting evidence in cluster space

- gather evidence in image neighborhoods — edge element
- local evidence supports some global shapes — line or circle
- global shape parameterized by a parameter set $\alpha = [a_1, \ldots, a_d]$
  - line: $\alpha = [d, \theta]$ are the polar coordinates of the line through $[x, y]$ with normal direction $\theta$
  - 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
Mapping pixels \((r, c)\) to polar parameter \((d, \Theta)\)

Row-column system

The parameters \(d\) and \(\theta\) 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.

Cartesian coordinate system

\[d = x \cos \Theta + y \sin \Theta\]
Mapping example

<table>
<thead>
<tr>
<th>Angle</th>
<th>Dist.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>30</td>
<td>69.6</td>
</tr>
<tr>
<td>60</td>
<td>81.2</td>
</tr>
<tr>
<td>90</td>
<td>70</td>
</tr>
<tr>
<td>120</td>
<td>40.6</td>
</tr>
<tr>
<td>150</td>
<td>0.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Angle</th>
<th>Dist.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>57.1</td>
</tr>
<tr>
<td>30</td>
<td>79.5</td>
</tr>
<tr>
<td>60</td>
<td>80.5</td>
</tr>
<tr>
<td>90</td>
<td>60</td>
</tr>
<tr>
<td>120</td>
<td>23.4</td>
</tr>
<tr>
<td>150</td>
<td>-19.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Angle</th>
<th>Dist.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>74.6</td>
</tr>
<tr>
<td>30</td>
<td>89.6</td>
</tr>
<tr>
<td>60</td>
<td>80.6</td>
</tr>
<tr>
<td>90</td>
<td>50</td>
</tr>
<tr>
<td>120</td>
<td>6.0</td>
</tr>
<tr>
<td>150</td>
<td>-39.6</td>
</tr>
</tbody>
</table>
Mapping example

Input Image

Rendering of Transform Results

Distance from Centre

Angle

MSU/CSE  Fall 2016
Simple algorithm overview

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
Possible implementation

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. $A[DQ, \text{THETAQ}]$ is the accumulator array. $DQ$ is the quantized distance from a line to the origin. $\text{THETAQ}$ 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);
                        PTLIST(DQ, THETAQ) := append(PTLIST(DQ, THETAQ), [R, C])
                    }
            }
    }
}

Binning evidence: note that original pixels [R, C] are saved in the bins.
Real world considerations

- The gradient direction $\Theta$ is bound to have error (due to what?)
- Since $d$ is computed from $\Theta$, $d$ will also have error
- Thus, evidence is spread in parameter space and may frustrate binning methods (2D histogram)
Solutions

- Bins for \((d, \Theta)\) can be smoothed to aggregate neighbors.
- Multiple evidence can be put into the bins by adding deltas to \(\Theta\) and computing variations of \((d, \Theta)\).
- Clustering can be done in real space; e.g. do not use binning.
- Busy images should be handled using a grid of (overlapping) windows to avoid a cluttered transform space.
What is the space of d and Θ?

- Many apps range Θ from 0 to π only.
- However, gradient direction can be detected from 0 to 2π.
- Imagine a checkerboard – there are 4 prominent directions, not 2.
- Using full range of direction requires negative values of d – consider checkerboard with origin in the center.
There are 3 parameters for a circle

1. circle is parameterized by $\alpha = [x_c, y_c, \text{radius}]$

2. for each edgel: $[x, y, \text{Mag}, \text{Dir}]$ with $\text{Mag} \geq t$
   
   (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
Example


MSU/CSE  Fall 2016
Possible implementation

procedure accumulate_circles(S,A);
{
  A := 0;
PagedList := 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;
PagedList(R0,C0,RAD) := append(PagedList(R0,C0,RAD),[R,C])
      }
}

MSU/CSE  Fall 2016
Some notes

- Circular and elliptical Hough Transforms are interesting, but of dubious value by themselves.
- Smarter tracking methods, such as Snakes, balloons, etc. might be needed along with region features.
- There are weak publications that will not help in real applications.
Fitting models to edge data

Can use straight line segments for data compression, or parametric curves for shape detection.
Fitting curves

- we have a candidate mathematical shape model
  - line: \( y = c_1 x + c_0 \) or \( f(x, y) = 0 \)
  - parabola: \( y = c_2 x^2 + c_1 x + c_0 \) or \( g(x, y) = 0 \)
  - circle: \( (x - x_c)^2 + (y - y_c)^2 = r^2 \)
- we have data points \( \{(x_j, y_j)_{j=1,...,n}\} \)
- 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
An unlimited number of models

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 fitting

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 \( y = 2.971x - 6.962 \).

<table>
<thead>
<tr>
<th>Data Pts ((x_j, y_j))</th>
<th>(0.0, -6.8)</th>
<th>(1.0, -4.1)</th>
<th>(2.0, -1.1)</th>
<th>(3.0, 1.8)</th>
<th>(4.0, 5.1)</th>
<th>(5.0, 7.9)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Residuals (y - y_j)</td>
<td>-0.162</td>
<td>0.110</td>
<td>0.081</td>
<td>0.152</td>
<td>-0.176</td>
<td>-0.005</td>
</tr>
</tbody>
</table>
Best fit minimizes sum of squared errors

- expression for least squares error

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

- error function \( \varepsilon \) 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 \( \partial \varepsilon / \partial c_0 = 0 \)
\[ \frac{\partial \varepsilon}{\partial c_1} = \sum_{j=1}^{n} 2(c_1 x_j + c_0 - y_j)x_j = 0 \]  
\[ = 2\left(\sum_{j=1}^{n} x_j^2\right)c_1 + 2\left(\sum_{j=1}^{n} x_j\right)c_0 - 2\sum_{j=1}^{n} x_jy_j \]  
\[ \frac{\partial \varepsilon}{\partial c_0} = \sum_{j=1}^{n} 2(c_1 x_j + c_0 - y_j) = 0 \]  
\[ = 2\left(\sum_{j=1}^{n} x_j\right)c_1 + 2\sum_{j=1}^{n} c_0 - 2\sum_{j=1}^{n} y_j \]

- These normal equations are nicely represented in matrix form.

\[ \begin{bmatrix} \sum_{j=1}^{n} x_j^2 & \sum_{j=1}^{n} x_j \\ \sum_{j=1}^{n} x_j & \sum_{j=1}^{n} 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{n} x_jy_j \\ \sum_{j=1}^{n} y_j \end{bmatrix} \]
Axis of least inertia 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
Angles, corners, 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
The downspout as a ribbon

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.
Road and canal as ribbons
More boundary detection later

- Will investigate “physics-based models”
- Can fit to 2D (snakes) or 3D data (balloons)
- Models do not allow the object topology to break
- Models enforce object smoothness