Boundary Detection: Hough Transform

Lecture #9

Reading: Computer Vision (Ballard and Brown): Chapter 4
“Use of the Hough Transform to detect lines and curves in pictures”, Comm. ACM 15, 1, January 1972 (pgs 112-115)
Boundaries of Objects

Marked by many users

http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/bench/html/images.html

Boundaries of Objects from Edges

Brightness Gradient (Edge detection)

- Missing edge continuity, many spurious edges
Boundaries of Objects from Edges

- But, low strength edges may be very important

Multi-scale Brightness Gradient

Image

Machine Edge Detection
Human Boundary Marking
Boundaries in Medical Imaging

Detection of cancerous regions.

[Foran, Comaniciu, Meer, Goodell, 00]

Boundaries in Ultrasound Images

Hard to detect in the presence of large amount of speckle noise.
Boundaries of Objects

Sometimes hard even for humans!

Topics

• Preprocessing Edge Images
• Edge Tracking Methods
• Fitting Lines and Curves to Edges
• The Hough Transform
Preprocessing Edge Images

- Image
  - Edge detection and Thresholding
  - Noisy edge image
  - Incomplete boundaries
  - Shrink and Expand
  - Thinning

Edge Tracking Methods

Adjusting a priori Boundaries:

Given: Approximate Location of Boundary
Task: Find Accurate Location of Boundary

- Search for STRONG EDGES along normals to approximate boundary.
- Fit curve (e.g., polynomials) to strong edges.
Edge Tracking Methods

Divide and Conquer:

**Given:** Boundary lies between points A and B  
**Task:** Find Boundary

- Connect A and B with Line  
- Find strongest edge along line bisector  
- Use edge point as break point  
- Repeat

Fitting Lines to Edges (Least Squares)

**Given:** Many \((x_i, y_i)\) pairs  
**Find:** Parameters \((m, c)\)

**Minimize:** Average square distance:

\[
E = \frac{1}{N} \sum_{i} (y_i - mx_i - c)^2
\]

Using:

\[
\frac{\partial E}{\partial m} = 0 \quad \text{and} \quad \frac{\partial E}{\partial c} = 0
\]

**Note:**

\[
y = \bar{y} = \frac{1}{N} \sum_{i} y_i \quad \bar{x} = \frac{1}{N} \sum_{i} x_i
\]

\[
c = y - mx = \sum_{i} (x_i - \bar{x})(y_i - \bar{y}) \sum_{i} (x_i - \bar{x})^2
\]

\[
m = \frac{\sum_{i} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i} (x_i - \bar{x})^2}
\]
Problem with Parameterization

Solution: Use a different parameterization
(same as the one we used in computing Minimum Moment of Inertia)

\[
E = \frac{1}{N} \sum_i (\rho - x_i \cos \theta + y_i \sin \theta)^2
\]

Note: Error E must be formulated carefully!

Line fitting can be max. likelihood - but choice of model is important
Curve Fitting

Find Polynomial:

\[ y = f(x) = ax^3 + bx^2 + cx + d \]

that best fits the given points \((x_i, y_i)\)

Minimize:

\[ \frac{1}{N} \sum_i \left[ y_i - (ax_i^3 + bx_i^2 + cx_i + d) \right]^2 \]

Using:

\[ \frac{\partial E}{\partial a} = 0 \quad , \quad \frac{\partial E}{\partial b} = 0 \quad , \quad \frac{\partial E}{\partial c} = 0 \quad , \quad \frac{\partial E}{\partial d} = 0 \]

Note: \( f(x) \) is LINEAR in the parameters \((a, b, c, d)\)

Line Grouping Problem

Slide credit: David Jacobs
This is difficult because of:

- Extraneous data: clutter or multiple models
  - We do not know what is part of the model?
  - Can we pull out models with a few parts from much larger amounts of background clutter?
- Missing data: only some parts of model are present
- Noise

- Cost:
  - It is not feasible to check all combinations of features by fitting a model to each possible subset

Hough Transform

- Elegant method for direct object recognition
  - Edges need not be connected
  - Complete object need not be visible
  - Key Idea: Edges VOTE for the possible model
Image and Parameter Spaces

Equation of Line: \( y = mx + c \)
Find: \( (m, c) \)

Consider point: \( (x_i, y_i) \)
\[ y_i = mx_i + c \quad \text{or} \quad c = -x_i m + y_i \]

Parameter space also called Hough Space

---

Line Detection by Hough Transform

Algorithm:
- Quantize Parameter Space \((m, c)\)
- Create Accumulator Array \(A(m, c)\)
- Set \(A(m, c) = 0\) \(\forall m, c\)
- For each image edge \((x_i, y_i)\) increment:
  \[ A(m, c) = A(m, c) + 1 \]
- If \((m, c)\) lies on the line:
  \[ c = -x_i m + y_i \]
- Find local maxima in \(A(m, c)\)
Better Parameterization

NOTE: \(-\infty \leq m \leq \infty\)

Large Accumulator

More memory and computations

Improvement: (Finite Accumulator Array Size)

Line equation: \(\rho = -x \cos \theta + y \sin \theta\)

Here \(0 \leq \theta \leq 2\pi\)
\(0 \leq \rho \leq \rho_{\text{max}}\)

Given points \((x_i, y_i)\) find \((\rho, \theta)\)

Hough Space Sinusoid

Hough Space

\(\theta\)

\(\rho\)

\((x_i, y_i)\)

Image Space

\(x\)

\(y\)

Vote

Horizontal axis is \(\theta\),
vertical is \(\rho\).
Mechanics of the Hough transform

- Difficulties
  - how big should the cells be? (too big, and we merge quite different lines; too small, and noise causes lines to be missed)

- How many lines?
  - Count the peaks in the Hough array
  - Treat adjacent peaks as a single peak

- Which points belong to each line?
  - Search for points close to the line
  - Solve again for line and iterate

Fewer votes land in a single bin when noise increases.
Adding more clutter increases number of bins with false peaks.

Real World Example

Original  Edge Detection  Found Lines

Parameter Space
Finding Circles by Hough Transform

Equation of Circle:
\[(x_i - a)^2 + (y_i - b)^2 = r^2\]

If radius is known: (2D Hough Space)

Accumulator Array \( A(a, b) \)

Finding Circles by Hough Transform

Equation of Circle:
\[(x_i - a)^2 + (y_i - b)^2 = r^2\]

If radius is not known: 3D Hough Space!
Use Accumulator array \( A(a, b, r) \)

What is the surface in the hough space?
Using Gradient Information

- Gradient information can save lot of computation:

  Edge Location \((x_i, y_i)\)
  Edge Direction \(\phi_i\)

  Assume radius is known:

  \[
  a = x - r \cos \phi \\
  b = y - r \sin \phi
  \]

  Need to increment only one point in Accumulator!!

Real World Circle Examples

Crosshair indicates results of Hough transform, bounding box found via motion differencing.
Finding Coins

Original

Edges (note noise)

Finding Coins (Continued)

Penn

Quarters
Finding Coins (Continued)

Note that because the quarters and penny are different sizes, a different Hough transform (with separate accumulators) was used for each circle size.

Coin finding sample images from: Vivek Kwatra

Generalized Hough Transform

- Model Shape NOT described by equation

Model:
**Generalized Hough Transform**

- Model Shape NOT described by equation

<table>
<thead>
<tr>
<th>Edge Direction</th>
<th>$\bar{r} = (r, \alpha)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi_1$</td>
<td>$\bar{r}_1$, $\bar{r}_2$, $\bar{r}_3$</td>
</tr>
<tr>
<td>$\phi_2$</td>
<td>$\bar{r}_4$, $\bar{r}_5$, $\bar{r}_6$</td>
</tr>
<tr>
<td>$\phi_3$</td>
<td>$\bar{r}_7$, $\bar{r}_8$</td>
</tr>
<tr>
<td>$\phi_n$</td>
<td>$\bar{r}<em>9$, $\bar{r}</em>{10}$</td>
</tr>
</tbody>
</table>

**Generalized Hough Transform**

Find Object Center $(x_c, y_c)$ given edges $(x_i, y_i, \phi_i)$

Create Accumulator Array $A(x_c, y_c)$

Initialize: $A(x_c, y_c) = 0 \ \forall (x_c, y_c)$

For each edge point $(x_i, y_i, \phi_i)$

For each entry $\bar{r}_k^i$ in table, compute:

- $x_c = x_i + r_k^i \cos \alpha_k^i$
- $y_c = y_i + r_k^i \sin \alpha_k^i$

Increment Accumulator: $A(x_c, y_c) = A(x_c, y_c) + 1$

Find Local Maxima in $A(x_c, y_c)$
Hough Transform: Comments

- Works on Disconnected Edges
- Relatively insensitive to occlusion
- Effective for simple shapes (lines, circles, etc)
- Trade-off between work in Image Space and Parameter Space
- Handling inaccurate edge locations:
  - Increment Patch in Accumulator rather than a single point
Next Class

- Lightness and Retinex.
- Reading: Horn, Chapter 9.
- Research webpages of Edward Adelson (MIT).
- Google for illusions.