Features Extraction for Sketch-Based Recognition

Lecture #8: Feature Extraction
Joseph J. LaViola Jr.
Fall 2007

Recall Pen-Based Interface Dataflow

- Raw Stroke Data
- Preprocessing
- Segmentation
- Sketch Understanding
- Ink Parsing
- Classification
- Feature Extraction And Analysis
- Make Inferences
Feature Extraction and Analysis

- What came first, the feature or the machine learning algorithm?
- Want to distinguish sketch components from one another
- Good features are critical
- Extract important information
  - geometrical, statistical, contextual
- Examples include
  - arc length, histograms, cusps, aspect ratio
  - self-intersections, stroke area, etc…

Finding Features

- Challenging problem
  - need fast algorithms for gathering information
  - features must be good discriminators
- Often trial and error
- Can be domain specific
Geometric Features (1)

- Number of strokes
  - if you know how many strokes a symbol has, you can break up your recognizer into pieces (i.e., recognizer for 1 stroke symbols, recognizer for 2 stroke symbols …)

- Cusps
  - smooth vs. jagged strokes
  - distance between cusps
    - useful for when cusps are close together/far apart

Geometric Features (2)

- Aspect ratio (width / height)
  - tall vs. flat

- Self Intersections
  - loops vs. no loops
  - strokes with write over
  - distance between self intersections also useful
  - use line segment intersection algorithm

loops → 2 3 4 5 → write over
Geometric Features (3)

- First and last distance
  - Strokes where first and last points are close together vs. far apart
  - simple computation – $\|p_n - p_1\|$
- Arc length
  - many different symbols have varying arc lengths
  - simple computation as well –

$$l = \sum_{i=2}^{n} \|p_i - p_{i-1}\|$$

Geometric Features (4)

- Stroke area
  - area defined by the vectors created with the initial stroke point and consecutive stroke points.
  - good discriminator for straight vs. curved lines

Given $\vec{u}_i = p_{i+1} - p_i$ and $\vec{v}_i = p_{i+2} - p_i$

$$s_{\text{area}} = \sum_{i=1}^{n-2} \frac{1}{2} (\vec{u}_i \times \vec{v}_i) \cdot \text{sgn}(\vec{u}_i \times \vec{v}_i)$$

where $\vec{u}_i \times \vec{v}_i$ is a scalar
Geometric Features (5)

- **Fit line feature**
  - sophisticated approach to finding how close a stroke is to a straight line
  - finds a least-squares approximation to a line using principal components and then uses this approximation to find the distance of the projection of the stroke points onto the approximated line
  - outputs a value in \([0,1]\)

- What is another name for this approach?

### Fit Line Feature Implementation

**Input:** A set of stroke points \(P\).

**Output:** A distance measure

\[
\text{FitLine}(P) = \begin{align*}
(1) & \quad x_1 = \sum_{i=1}^{n} X(P_i) \\
(2) & \quad y_1 = \sum_{i=1}^{n} Y(P_i) \\
(3) & \quad x_2 = \sum_{i=1}^{n} X(P_i)^2 \\
(4) & \quad y_2 = \sum_{i=1}^{n} Y(P_i)^2 \\
(5) & \quad x_{y1} = \sum_{i=1}^{n} X(P_i)Y(P_i) \\
(6) & \quad x_3 = x_2 - x_1^2/n \\
(7) & \quad y_3 = y_2 - y_1^2/n \\
(8) & \quad x_{y2} = x_{y1} - (x_1y_1)/n \\
(9) & \quad \text{rad} = \sqrt{(x_3 - y_3)^2 + 4x_{y2}^2} \\
(10) & \quad \text{error} = (x_3 + y_3 - \text{rad})/2 \\
(11) & \quad \text{rms} = \sqrt{\text{error}/n} \\
(12) & \quad \text{if } x_3 > y_3 \\
(13) & \quad a = -2x_{y2} \\
(14) & \quad b = x_3 - y_3 + \text{rad} \\
(15) & \quad \text{else if } x_3 < y_3 \\
(16) & \quad a = y_3 - x_3 + \text{rad} \\
(17) & \quad b = -2x_{y2} \\
(18) & \quad \text{else} \\
(19) & \quad \text{if } x_{y2} = 0 \\
(20) & \quad a = b = c = 0 \\
(21) & \quad \text{error} = +\infty \\
(22) & \quad \text{else} \\
(23) & \quad a = -1 \\
(24) & \quad b = -1 \\
(25) & \quad c = \frac{(a - x_1)(b - y_1)}{n} \\
(26) & \quad \text{mag} = \sqrt{a^2 + b^2} \\
(27) & \quad a = \frac{a}{\text{mag}} \\
(28) & \quad b = \frac{b}{\text{mag}} \\
(29) & \quad \text{min}_1 = +\infty \\
(30) & \quad \text{max}_3 = -\infty \\
(31) & \quad \text{for } i = 1 \text{ to } n \\
(32) & \quad \text{err} = aX(P_i) + bY(P_i) + c \\
(33) & \quad \text{pX} = X(P_i) - a \cdot \text{err} \\
(34) & \quad \text{pY} = Y(P_i) - b \cdot \text{err} \\
(35) & \quad \text{ploc} = b \cdot \text{pX} - b \cdot \text{pY} \\
(36) & \quad \text{min}_1 = \min(\text{min}_1, \text{ploc}) \\
(37) & \quad \text{max}_1 = \max(\text{max}_1, \text{ploc}) \\
(38) & \quad \text{return } \frac{\text{mag} \cdot \text{error}}{\text{max}_1 - \text{min}_1}
\end{align*}
\]
Statistical Features (1)

- **Side ratios**
  - first and last point of strokes have variable locations with respect to the bounding box
  - **Approach**
    - take the x coordinates of the first and last point of a stroke
    - subtract them from the left side of the symbol’s bounding box (i.e., the bounding box’s leftmost x value)
    - divide by the bounding box width.

Statistical Features (2)

- **Top and Bottom ratios**
  - similar to side ratios except we are dealing with y coordinate
  - **Approach**
    - take y coordinate of the first and last point of a stroke
    - subtract from the top of the symbol’s bounding box (i.e., the bounding box’s topmost y value)
    - these values are divided by the bounding box height.
Statistical Features (3)

- **Point Histogram**
  - distribution of point locations in stroke bounding box
  - discrimination where point concentrations are high
  - approach
    - break up box into $n \times m$ grid
    - Count number of points in each sub box
    - divide by total number of points

Statistical Features (4)

- **Angle Histogram**
  - similar to point histogram except dealing with angles
  - Approach
    
    Given $\vec{v}_j = p_j - p_{i-1}$ for $2 \leq i \leq n$ and $\vec{x} = (1,0)$
    
    $\alpha_j = \arccos\left(\frac{\vec{x} \cdot \vec{v}_j}{\|\vec{v}_j\|}\right)$
    
    - put angles into bins of $n$ degrees
The Rubine Feature Set (Rubine 1991)

- Part of Rubine’s gesture recognition system
  - we will see this next class

- Stroke
  - \( P = \) total number of points
  - \( p = \) middle point
  - first point \((x_0, y_0, t_0)\)
  - last point \((x_{P-1}, y_{P-1}, t_{P-1})\)
  - compute \(x_{\text{min}}, y_{\text{min}}, x_{\text{max}}, y_{\text{max}}\)

---

Feature \( f_1 \)

- Cosine of starting angle

\[
f_1 = \cos(\alpha) = \frac{x_2 - x_0}{\sqrt{(x_2 - x_0)^2 + (y_2 - y_0)^2}}
\]
Feature $f_2$

- Sine of starting angle

$$f_2 = \sin(\alpha) = \frac{(y_2 - y_0)}{\sqrt{(x_2 - x_0)^2 + (y_2 - y_0)^2}}$$

Feature $f_3$

- Length of diagonal of bounding box (gives an idea of the size of the bounding box)

$$f_3 = \sqrt{(x_{\text{max}} - x_{\text{min}})^2 + (y_{\text{max}} - y_{\text{min}})^2}$$
Feature $f_4$

- Angle of diagonal
- Gives an idea of the shape of the bounding box (long, tall, square)

\[ f_4 = \arctan \left( \frac{y_{\text{max}} - y_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \right) \]

Feature $f_5$

- Distance from start to end of stroke

\[ f_5 = \sqrt{(x_{P-1} - x_0)^2 + (y_{P-1} - y_0)^2} \]
Feature $f_6$

- Cosine of ending angle

$$f_6 = \cos(\beta) = \frac{(x_{p-1} - x_0)}{f_5}$$

Feature $f_7$

- Sine of ending angle

$$f_7 = \sin(\beta) = \frac{(x_{p-1} - x_0)}{f_5}$$
More Definitions (before we continue)

Let $\Delta x_p = x_{p+1} - x_p$ and $\Delta y_p = y_{p+1} - y_p$

Let $\theta_p = \arctan \frac{\Delta x_p \Delta y_{p-1} - \Delta x_{p-1} \Delta y_p}{\Delta x_p \Delta x_{p-1} + \Delta y_p \Delta y_{p-1}}$ Directional angle

Let $\Delta t_p = t_{p+1} - t_p$ Time delta

Feature $f_8$

- Total stroke length

$$f_8 = \sum_{p=0}^{P-2} \sqrt{\Delta x_p^2 + \Delta y_p^2}$$
Feature $f_9$

- Total rotation (from start to end point)
- (not the same as $\beta - \alpha$ – think of spirals)

$$f_9 = \sum_{p=1}^{P-2} \theta_p$$

Feature $f_{10}$

- Absolute rotation
- How much does it move around

$$f_{10} = \sum_{p=1}^{P-2} |\theta_p|$$
Feature $f_{11}$

- Rotation squared
- How smooth are the turns?
- Measure of sharpness

$$f_{11} = \sum_{p=1}^{P-2} \theta_p^2$$

Feature $f_{12}$

- The maximum speed reached (squared)

$$f_{12} = \max_{p=0}^{p-2} \frac{\Delta x_p^2 + \Delta y_p^2}{\Delta t_p^2}$$
Feature $f_{13}$

- Total time of stroke

$$f_{13} = t_{P-1} - t_0$$

Next Class

- Start discussing machine learning algorithms
  - linear classifiers (e.g., Rubine)
  - template matching
  - SVM
  - AdaBoost
  - etc…