Ink Preprocessing and Preparation

Lecture #5: Preparing Ink
Joseph J. LaViola Jr.
Fall 2010

Recall Pen-Based Interface Dataflow

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

- Points and strokes
  \[ s = p_1, p_2, \ldots, p_n \]
  where
  \[ p_i = (x_{i1}, y_{i1}, t_i), \quad 1 \leq i \leq n \]
  \[ S = s_1, s_2, \ldots, s_m \]

- Image
  - pixel matrix
  - not as popular

Preprocessing

- Often required to clean raw data
- Stroke Invariance
  - scale
  - position
  - orientation
  - slant/skew
  - order/direction
- Filtering and Smoothing
- Dehooking

Normal view of stroke

Zoomed in view of stroke showing unwanted cusps and self-intersections
Scale Invariance

- Why? – want to ensure stroke has a canonical representation so its size makes no difference in recognition
- Approach
  - define constant width or height
  - scale stroke maintaining aspect ratio
  - choose constant width or height based on stroke

Translation Invariance

- Why? – want to ensure stroke has canonical representation so its position makes no difference in recognition
- Approach
  - translate stroke to origin
  - use stroke bounding box
  - possible translation points
    - top left point
    - center point
Rotation Invariance

- Primarily used when for handwriting (sometimes for shapes)
- Why? – want to remove baseline drift which could affect recognition
- Baseline drift – deviation between baseline and horizontal axis
- Difficult problem to deal with
  - ambiguous baseline locations
- One approach (Guerfali and Plamondon 1993)
  - uses center of mass of word regions
  - least squares for baseline construction

Slant/Skew Invariance

- Important in handwriting recognition
- Handwriting slant – deviation between the principal axis of strokes and vertical axis
  - Often referred to as deskewing process
- Why? – can be important for segmentation
- Difficult problem – very subjective
- One approach (Guerfali and Plamondon 1993)
  - zone extraction
  - observation windows
  - local and global slants
Stroke Direction and Ordering Invariance

- Can be large variation in ways a symbol is drawn
  - order of strokes
  - direction of strokes
- Possible approach is to model each possible combination
  - combinatorially expensive
  - could hurt recognition accuracy
- Want to assign canonical ordering and direction
  - see Matsakis (1999)

Stroke Invariance Summary

- Want to have canonical representation
- Makes calculating features easier
- Makes recognition easier

\[ y = x^3 + 3x \]
Resampling

- Why? – sometimes we want to have all strokes have the same number of points
  - helps deal with some recognition algorithms
- Approach
  - linear interpolation between points

Filtering and Smoothing

- Remove duplicate points
- Remove unwanted cusps and self-intersections
- Thinning – reduce points
- Dot reduction – reduce dots to single point
- Stroke connection- deal with extraneous pen lifts (e.g., stroke segmentation)
Gaussian Smoothing

\[
p_i^{\text{filt}} = \sum_{j=-3\sigma}^{3\sigma} w_j p_{j+i} \quad \sigma \text{ is a scaling parameter}
\]

\[
w_j = \frac{3\sigma}{\sum_{k=-3\sigma}^{3\sigma} e^{-\frac{k^2}{2\sigma^2}}} \quad \frac{1}{e^{-\frac{j^2}{2\sigma^2}}}
\]

Should try to maintain cusps when filtering

A Filtering Algorithm

Input: Stroke \( s_i \) and a self-intersection threshold \( \alpha \).
Output: A filtered list of points.

\[
\text{FILTERSTROKE}(s_i, \alpha)
\]

1. \( P \leftarrow \text{Points}(s_i) \)
2. \( \text{cur}_p \leftarrow p_1 \)
3. for \( i = 2 \) to \( n \)
4. \hspace{0.5cm} if \( \text{cur}_p = p_i \)
5. \hspace{1cm} \( \text{BadPts} \leftarrow P_i \)
6. \hspace{1cm} else
7. \hspace{1.5cm} \( \text{cur}_p = p_i \)
8. RemovePointsFromPointList(BadPts, P)
9. \( \text{SelfInts} \leftarrow \text{SelfIntersectionLocations}(P) \)
10. \( \text{prev} \leftarrow -1 \)
11. for \( i = 1 \) to \( |P| \)
12. \hspace{0.5cm} if \( \text{prev} \neq -1 \) and \( \text{SelfInts}_i - \text{prev} > \alpha \)
13. \hspace{1cm} for \( j = \text{prev} \) to \( \text{SelfInts}_i \)
14. \hspace{1.5cm} \( \text{BadPts} \leftarrow P_j \)
15. \hspace{0.5cm} \( \text{prev} \leftarrow \text{SelfInts}_i \)
16. RemovePointsFromPointList(BadPts, P)
17. Return P
Dehooking

- Want to eliminate hooks that can occur at the end of strokes (sometimes at the beginning)
- Hooks come from
  - inaccuracies in pen-down detection
  - rapid and erratic stylus motion
- Hooks vary depending on user and on stroke

A Dehooking Algorithm

**Input:** Stroke $s_i$, minimum and maximum hook threshold $hook_{\text{min}}$ and $hook_{\text{max}}$, and a dehooking distance threshold $\epsilon_{\text{hook}}$.

**Output:** A dehooked list of points

$$\text{DEHOOK}(s_i, hook_{\text{min}}, hook_{\text{max}}, \epsilon_{\text{hook}})$$

1. $P' \leftarrow \text{Points}(s_i)$
2. $\text{maxdist} \leftarrow 0$
3. for $i = 2$ to $\min(hook_{\text{min}}, P_n - hook_{\text{max}})$
   4. $\text{dist} \leftarrow \|P_i - P_1\|$
   5. if $\text{dist} > \epsilon_{\text{hook}}$
      6. break
   7. if $\text{dist} \geq \text{maxdist}$
      8. $\text{maxdist} = \text{dist}$
   9. else
      10. for $j = 1$ to $i$
         11. $\text{BadPts} \leftarrow P_j$
         12. break
      13. $\text{maxdist} \leftarrow 0$
Dehooking Algorithm Cont’d

\begin{align*}
(14) \quad & \text{for } i = P_{n-1} \text{ down to } \max(hook_{max}, P_n - hook_{min}) \\
(15) \quad & \text{dist} \leftarrow \|P_n - P_i\| \\
(16) \quad & \text{if } \text{dist} > \epsilon_{\text{hook}} \\
(17) \quad & \text{break} \\
(18) \quad & \text{if } \text{dist} \geq \maxdist \\
(19) \quad & \maxdist = \text{dist} \\
(20) \quad & \text{else} \\
(21) \quad & \text{for } j = n \text{ down to } i \\
(22) \quad & \text{BadPts} \leftarrow P_j \\
(23) \quad & \text{break} \\
(24) \quad & \text{RemovePointsFromPointList(BadPts, P)} \\
(25) \quad & \text{return } P
\end{align*}

Next Class – Discussion

- Assignment 1 – out
- Readings