Ink Preprocessing and Preparation

Lecture #5: Preparing Ink
Joseph J. LaViola Jr.
Fall 2008

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
  \( s = p_1p_2\ldots p_n \)
  where
  \( p_i = (x_i, y_i, t_i), \ 1 \leq i \leq n \)
  \( S = s_1s_2\ldots s_m \)

- 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

\[ y = x^2 + 3x \]
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

\[ p_{i}^{\text{filt}} = \sum_{j=-3\sigma}^{3\sigma} w_{j} p_{j+i} \]

\( \sigma \) is a scaling parameter

Should try to maintain cusps when filtering

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

A Filtering Algorithm

**Input:** Stroke \( s \), and a self-intersection threshold \( \alpha \).

**Output:** A filtered list of points.

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

1. \( P \leftarrow \text{Points}(s) \)
2. \( \text{curr} \leftarrow P_{1} \)
3. For \( i = 2 \) to \( n \)
4. \( \text{if} \ \text{curr} = P_{i} \)
5. \( \text{BadPts} \leftarrow P_{i} \)
6. Else
7. \( \text{curr} = P_{i} \)
8. RemovePointsFromPointList(BadPts, P)
9. SelfInts \leftarrow SelfIntersectionLocations(P)
10. \( \text{prev} \leftarrow -1 \)
11. For \( i = 1 \) to \( ||P|| \)
12. \( \text{if} \ \text{prev} \neq -1 \) and \( \text{SelfInts}_{i} - \text{prev} > \alpha \)
13. For \( j = \text{prev} \) to \( \text{SelfInts}_{i} \)
14. \( \text{BadPts} \leftarrow P_{j} \)
15. \( \text{prev} \leftarrow \text{SelfInts}_{i} \)
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 $hook_{min}$ and $hook_{max}$, and a dehooking distance threshold $\epsilon_{hook}$.

**Output:** A dehooked list of points

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

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

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

Next Class – Discussion

- Assignment 1 – out

- Readings