Ink Preprocessing and Preparation

Lecture #5: Preparing Ink
Joseph J. LaViola Jr.
Fall 2015

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
  \[ p_i = (x_i, y_i, t_i), \quad 1 \leq i \leq n \]
  \[ S = s_1, s_2, \ldots, s_m \]
- 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
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
**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

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

\[ \sigma \text{ is a scaling parameter} \]

Should try to maintain cusps when filtering

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

A Filtering Algorithm

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

\begin{align*}
\text{FILTERSTROKE}(s_i, \alpha) & = \text{Points}(s_i) \\
1 & \text{ num } \leftarrow 1 \\
2 & \text{ curpt } \leftarrow P_1 \\
3 & \text{ for } i = 2 \text{ to } n & \leftarrow \text{Points}(s_i) \\
4 & \text{ if } \text{ curpt } = P_i & \text{SelfInts } \leftarrow \text{SelfIntersectionLocations}(P) \\
5 & \text{ BadPts } \leftarrow P_i & prev \leftarrow -1 \\
6 & \text{ else } & \\
7 & \text{ curpt } \leftarrow P_i & \\
8 & \text{ RemovePointsFromPointList(BadPts, P)} & \\
9 & \text{ SelfInts } \leftarrow \text{SelfIntersectionLocations}(P) & \\
10 & prev \leftarrow -1 & \\
11 & \text{ for } i = 1 \text{ to } |P| & \\
12 & \text{ if } prev \neq -1 \text{ and } prev > \alpha & \\
13 & \text{ for } j = prev \text{ to } SelfInts_i & \\
14 & \text{ BadPts } \leftarrow P_j & prev \leftarrow SelfInts_i \\
15 & \text{ RemovePointsFromPointList(BadPts, P)} & \\
16 & \text{return } P & \\
\end{align*}
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

```plaintext
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. dist $\leftarrow \| P_i - P_1 \|
5. if dist $> \epsilon_{hook}$
6. break
7. if dist $> maxdist$
8. maxdist $= dist$
9. else
10. for $j = 1$ to $i$
11. $BadPts \leftarrow P_j$
12. break
13. maxdist $\leftarrow 0$
```
Dehooking Algorithm Cont’d

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

Next Class – Discussion

- Readings