Ink Preprocessing and Preparation

Lecture #5: Preparing Ink
Joseph J. LaViola Jr.
Fall 2016

Recall Pen-Based Interface Dataflow
### 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**

---

Fall 2016  
CAP 6105 – Pen-Based User Interfaces  
© Joseph J. LaViola Jr.
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

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

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

\(\sigma\) is a scaling parameter

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. \(\text{curPt} \leftarrow P_1\)
3. for \(i = 2\) to \(n\)
4. if \(\text{curPt} = P_i\)
5. \(\text{BadPts} \leftarrow P_i\)
6. else
7. \(\text{curPt} = P_i\)
8. \(\text{RemovePointsFromPointList(BadPts, P)}\)
9. \(\text{SelfInts} \leftarrow \text{SelfIntersectionLocations}(P)\)
10. \(\text{prev} \leftarrow -1\)
11. for \(i = 1\) to \(|P|\)
12. 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. \(\text{RemovePointsFromPointList(BadPts, P)}\)
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

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

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

(14) \textbf{for } \(i = P_{n-1}\) \textbf{ down to } \max(hook_{\text{max}}, P_n - hook_{\text{min}}) \\
(15) \quad dist \leftarrow \|P_n - P_i\| \\
(16) \quad \textbf{if } dist > \epsilon_{\text{hook}} \\
(17) \quad \textbf{break} \\
(18) \quad \textbf{if } 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 \text{return } P

Next Class – Discussion

- Readings