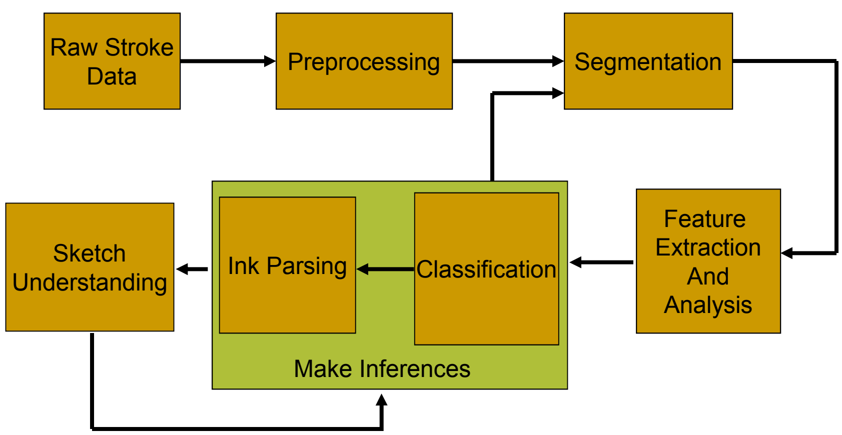
Ink Preprocessing and Preparation

Lecture #5: Preparing Ink

Joseph J. LaViola Jr.

Fall 2016

Recall Pen-Based Interface Dataflow



Representing Data

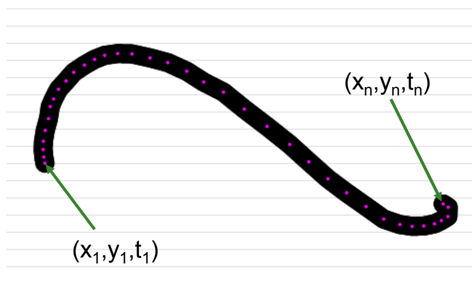
- Points and strokes

  $s = p_1 p_2 \ldots p_n$

  where

  $p_i = (x_i, y_i, t_i),\ 1 \le i \le n$

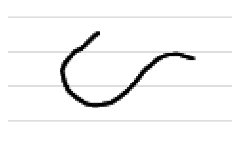
  $S = s_1 s_2 \ldots s_m$

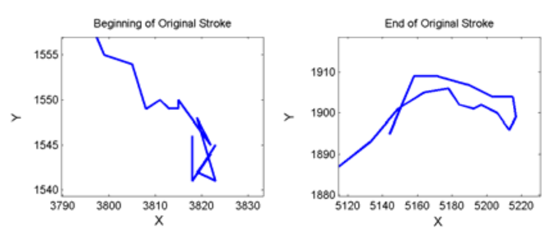


- Image
  - pixel matrix
  - not as popular

Preprocessing

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

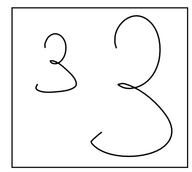




Zoomed in view of stroke showing unwanted cusps and self-intersections

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

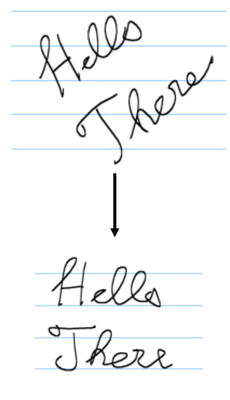


Translation Invariance

- 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

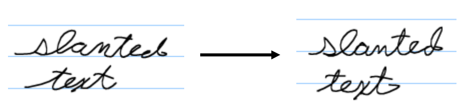
- Primarily used when for handwriting (sometimes for shapes)
- Why? – want to remove baseline drift which could affect recognition
- Baseline drift – deviation between baseline and horizontal axis
- Difficult problem to deal with
  - ambiguous baseline locations
- One approach (Guerfali and Plamondon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



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

Slant/Skew Invariance

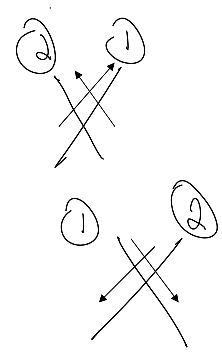
- Important in handwriting recognition
- Handwriting slant – deviation between the principal axis of strokes and vertical axis
  - Often referred to as *deskewing process*
- Why? – can be important for segmentation
- Difficult problem – very subjective
- One approach (Guerfali and Plamondon 1993)
  -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



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

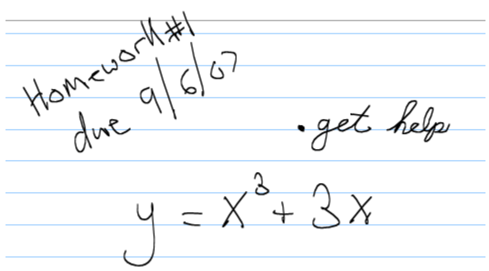
Stroke Direction and Ordering Invariance

- 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

$$p_i^{filt} = \sum_{j=-3\sigma}^{3\sigma} w_j \, p_{j+i}$$

$$w_j = \frac{e^{-\frac{j^2}{2\sigma^2}}}{\sum_{k=-3\sigma}^{3\sigma} e^{-\frac{k^2}{2\sigma^2}}}$$

σ is a scaling parameter

Should try to maintain cusps when filtering

A Filtering Algorithm

Input: Stroke s_i and a self-intersection threshold α.
Output: A filtered list of points

```
FilterStroke(s_i, α)
    P ← Points(s_i)
    cur_pt ← P_1
    for i = 2 to n
        if cur_pt = P_i
            BadPts ← P_i
        else
            cur_pt = P_i
    RemovePointsFromPointList(BadPts, P)
    SelfInts ← SelfIntersectionLocations(P)
    prev ← −1
    for i = 1 to ‖P‖
        if prev ≠ −1 and SelfInts_i − prev > α
            for j = prev to SelfInts_i
                BadPts ← P_j
        prev ← SelfInts_i
    RemovePointsFromPointList(BadPts, P)
    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 ϵ_{hook}.
Output: A dehooked list of points

DeHook(s_i,$hook_{min}$,$hook_{max}$,ϵ_{hook})
    $P \leftarrow Points(s_i)$
    $maxdist \leftarrow 0$
    for $i = 2$ **to** $\min(hook_{min}, P_n - hook_{max})$
        $dist \leftarrow \|P_i - P_1\|$
        if $dist > \epsilon_{hook}$
            break
        if $dist \geq maxdist$
            $maxdist = dist$
        else
            for $j = 1$ **to** i
                $BadPts \leftarrow P_j$
            break
        $maxdist \leftarrow 0$

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

Dehooking Algorithm Cont'd

```
for i = P_{n-1} down to max(hook_max, P_n - hook_min)
    dist ← ||P_n - P_i||
    if dist > ε_hook
        break
    if dist ≥ maxdist
        maxdist = dist
    else
        for j = n down to i
            BadPts ← P_j
        break
RemovePointsFromPointList(BadPts, P)
return P
```


Next Class – Discussion

- Readings
  - Wolin, A., Eoff, B., and Hammond, T. ShortStraw: A Simple and Effective Corner Finder for Polylines. Eurographics 5th Annual Workshop on Sketch-Based Interfaces and Modeling, Annecy, France, June, 2008, pp. 33-40.
  - Xiong, Y. and LaViola, J. "Revisiting ShortStraw – Improving Corner Finding in Sketch-Based Interfaces", Proceedings of the Sixth Eurographics/ACM Symposium on Sketch-Based Interfaces and Modeling 2009, 101-108, August 2009.
  - Herold, J. and Stahovich, T. SpeedSeg: A Technique for Segmenting Pen Strokes Using Pen Speed Computers and Graphics, Volume 35, Issue 2, 2011, pp. 250-264