Ink Segmentation

Recall Pen-Based Interface Dataflow

1. Raw Stroke Data → Preprocessing
2. Preprocessing → Segmentation
3. Segmentation → Feature Extraction and Analysis
4. Feature Extraction and Analysis → Sketch Understanding
5. Sketch Understanding → Ink Parsing
6. Ink Parsing → Classification
7. Classification → Make Inferences
8. Make Inferences → Segmentation
9. Segmentation → Preprocessing
10. Preprocessing → Raw Stroke Data
Segmentation

- Determine which strokes go together
- Determine which strokes should be apart
- Can be done in real-time or in batch
- Often uses proximity and timing information

\[ y = \frac{1}{2} x^2 \]
\[ y = x^2 e^{-\frac{t}{2}} \]

Grouping Strokes Together

- Why? – Multiple strokes can form one symbol
  - math symbols, shapes, etc…
  - want to pass all strokes that make up a symbol to recognizer

4 strokes

2 strokes

3 strokes
Grouping Strokes Together – Basic Approach

- Check to see if two or more strokes intersect
  - if they do then group them together
- Can use simple line segment intersection tests
- Problems
  - ink strokes – ink ≠ polyline
  - what if two strokes do not intersect but should be grouped together?
  - what if two strokes intersect but should not be grouped together?

Ink Strokes and Polylines

- Polylines are internal representation
- Ink has width
  - need requires more robust intersection
- One approach
  - find silhouettes
  - do intersection testing on them
Robust Stroke Intersection (Part 1)

Input: Stroke $s_i$, a set of candidate strokes $CS = \{s_1, s_2, \ldots, s_n\}$.
Output: True or false

\textsc{RobustIntersection}(s_i, CS)
(1) $P \leftarrow \text{Points}(s_i)$
(2) $cs_1 \leftarrow \text{Circle}(P_1, \frac{\text{PenInkWidth}()}{2})$
(3) $cs_2 \leftarrow \text{Circle}(P_0, \frac{\text{PenInkWidth}()}{2})$
(4) $sil_1 \leftarrow \text{Polygon}(\text{ComputeStrokeEdges}(s_i))$
(5) \text{foreach} Stroke $stk \in CS$
(6) $Q \leftarrow \text{Points}(stk)$
(7) $csstk_1 \leftarrow \text{Circle}(Q_1, \frac{\text{PenInkWidth}()}{2})$
(8) $csstk_2 \leftarrow \text{Circle}(Q_n, \frac{\text{PenInkWidth}()}{2})$
(9) $sil_2 \leftarrow \text{Polygon}(\text{ComputeStrokeEdges}(stk))$
(10) \text{if} $cs_1 \cap csstk_2$ or $cs_1 \cap sil_2$ or $cs_2 \cap sil_1$ or $cs_2 \cap csstk_2$
\text{or} $cs_2 \cap sil_2$ or $sil_1 \cap csstk_1$ or $sil_1 \cap csstk_2$ or $sil_1 \cap sil_2$
\text{then}
\text{return} true
\text{else}
\text{return} false

Robust Stroke Intersection (Part 2)

Input: Stroke $s_i$
Output: A list of silhouette points

\textsc{ComputeStrokeEdges}(s_i)
(1) $P \leftarrow \text{Points}(s_i)$
(2) $penw \leftarrow \frac{\text{PenInkWidth}()}{2}$
(3) \text{if} $n < 3$
\text{then}
\text{return} $P$
\text{else}
\text{for} $i = 1$ to $n - 1$
(6) $\vec{v}_1 \leftarrow \text{Vector}(Y(P_{i+1}) - Y(P_i), -(X(P_{i+1}) - X(P_i)))$
(7) $\vec{v}_2 \leftarrow \text{Vector}(-Y(P_{i+1}) + Y(P_i), X(P_{i+1}) - X(P_i))$
(8) $Ppts_{1i} \leftarrow P_i + penw \frac{\vec{v}_1}{\|\vec{v}_1\|}$
(9) $Ppts_{2i} \leftarrow P_i + penw \frac{\vec{v}_2}{\|\vec{v}_2\|}$
\text{if} $i = n - 1$
(10) $Ppts_{1i} \leftarrow P_{i+1} + penw \frac{\vec{v}_1}{\|\vec{v}_1\|}$
(11) $Ppts_{2i} \leftarrow P_{i+1} + penw \frac{\vec{v}_2}{\|\vec{v}_2\|}$
Robust Intersection (Part 2) – cont’d

```
for i = 1 to n - 1
    if i = 1
        Sip1 = Ppts1
    Sip2 = Ppts2
    continue
    if i = n - 1
        Sip1n+1 = Ppts1n+1
        Sip2n+1 = Ppts2n+1
    continue
    v1 ← Vector( X(Ppts1i-1) - X(Ppts1i), Y(Ppts1i-1) - Y(Ppts1i) )
    v2 ← Vector( X(Ppts1i) - X(Ppts1i+1), Y(Ppts1i) - Y(Ppts1i+1) )
    intpt ← LineIntersection( Ppts1i, v1, Ppts1i+1, v2 )
    if intpt = ∅
        Sip1i = Ppts1
    else
        Sip1i = intpt
    v1 ← Vector( X(Ppts2i-1) - X(Ppts2i), Y(Ppts2i-1) - Y(Ppts2i) )
    v2 ← Vector( X(Ppts2i) - X(Ppts2i+1), Y(Ppts2i) - Y(Ppts2i+1) )
    intpt ← LineIntersection( Ppts2i, v1, Ppts2i+1, v2 )
    if intpt = ∅
        Sip2i = Ppts2
    else
        Sip2i = intpt
    return CreatePointList(Sip1, Sip2, Sip10)
```

Grouping Strokes Together – Extending Basic Approach

- What if two or more strokes should be grouped together but do not intersect?
- Need other information
  - timing info
  - spatial info
- If two strokes are close together and they have been drawn consecutively then there is a good chance they should be grouped together
  - still has problems
Grouping Strokes Together – Using Recognition

- To help with segmentation – use recognizer (Smithies et. al 1999)
- For each stroke
  - take last k strokes and send to recognizer
  - look for symbol recognitions with highest confidence level
  - group based on highest confidence level
- When all else fails
  - use domain knowledge
  - easy to use UI correction techniques

Inadvertent Stroke Grouping

- What if strokes are intersecting but should not be grouped together?
- Must look at context
  - would such a symbol make sense in its surroundings?
  - example – perpendicular symbol over 6 does not make sense (so ungroup to make 1 and division line)
- UI correction also important (tools for breaking strokes apart)
Breaking Strokes Apart

- Why? – Want to break symbols (groups of strokes) into logical blocks
  - Examples include mathematical expressions on a page, multiple diagrams or drawings
- Starts moving into sketch understanding and sketch parsing
- As with grouping, using recognition engine can help
- Domain knowledge also important

Breaking Strokes Apart – Basic Approach

- Lines of math
- Do a horizontal line sweep, if white space is found, break up strokes into expressions
  - a threshold could be used just in case of a few pixels found in sweep
- Another approach
  - Look at histogram of points
    - rotate ink 90 degrees
    - project onto x-axis
    - find minima

\[ y = 3x^2 + 6 \]
\[ y = 5x^2 - 4 \]
Strategy Summary

- Can go a long way with speed data, proximity info, and intersection testing
  - does not work every time
- Use recognizer to help find segmentations that make sense
- Make use of domain knowledge
- Have easy to use UI techniques for corrections
- More on this when we get to sketch understanding

Readings