Ink Segmentation

Recall Pen-Based Interface Dataflow

- Raw Stroke Data
  - Preprocessing
  - Segmentation
    - Sketch Understanding
      - Ink Parsing
      - Classification
      - Feature Extraction and Analysis
      - Make Inferences
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

\[
\begin{align*}
y &= \frac{1}{2} x^2 \\
y &= x^2 e^{-\frac{1}{2}t}
\end{align*}
\]

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
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

**ROBUSTINTERSECTION**($s_i, CS$)

1. $P \leftarrow \text{Points}(s_i)$
2. $cs_1 \leftarrow \text{Circle}(P_i, \frac{PenInkWidth}{2})$
3. $cs_2 \leftarrow \text{Circle}(P_n, \frac{PenInkWidth}{2})$
4. $sil_1 \leftarrow \text{Polygon}(\text{ComputeStrokeEdges}(s_i))$
5. foreach Stroke $stk \in CS$
6. $Q \leftarrow \text{Points}(stk)$
7. $cstk_1 \leftarrow \text{Circle}(Q_1, \frac{PenInkWidth}{2})$
8. $cstk_2 \leftarrow \text{Circle}(Q_n, \frac{PenInkWidth}{2})$
9. $silk_2 \leftarrow \text{Polygon}(\text{ComputeStrokeEdges}(stk))$
10. if $cs_1 \cap cstk_1$ or $cs_1 \cap cstk_2$ or $cs_1 \cap sil_2$ or $cs_2 \cap cstk_1$ or $cs_2 \cap cstk_2$ or $cs_2 \cap sil_2$
11. return true
12. return false

Robust Stroke Intersection (Part 2)

**Input:** Stroke $s_i$

**Output:** A list of silhouette points

**COMPUTESTROKEEDGES**($s_i$)

1. $P \leftarrow \text{Points}(s_i)$
2. $penw \leftarrow \frac{PenInkWidth}{2}$
3. if $n < 3$
4. return $P$
5. foreach $i = 1$ to $n - 1$
6. $v_i^1 \leftarrow \text{Vector}(Y(P_{i+1}) - Y(P_i), -(X(P_{i+1}) - X(P_i)))$
7. $v_i^2 \leftarrow \text{Vector}(-(Y(P_{i+1}) - Y(P_i)), X(P_{i+1}) - X(P_i))$
8. $Ppts_{1i} \leftarrow P_i + penw \frac{v_i^1}{\|v_i^1\|}$
9. $Ppts_{2i} \leftarrow P_i + penw \frac{v_i^2}{\|v_i^2\|}$
10. if $i = n - 1$
11. $Ppts_{1i} \leftarrow P_{i+1} + penw \frac{v_i^1}{\|v_i^1\|}$
12. $Ppts_{2i} \leftarrow P_{i+1} + penw \frac{v_i^2}{\|v_i^2\|}$
Robust Intersection (Part 2) – cont’d

for $i = 1$ to $n - 1$
if $i = 1$
  $Silpts_{i-1} = Ppts_{i}$
  $Silpts_{i} = Ppts_{i}$
continue
if $i = n - 1$
  $Silpts_{i-1} = Ppts_{i-1}$
  $Silpts_{i} = Ppts_{i+1}$
continue
$v_0 = \text{Vector}(X(Ppts_{i-1}), Y(Ppts_{i-1}), X(Ppts_{i}), Y(Ppts_{i}))$
$v_1 = \text{Vector}(X(Ppts_{i}), Y(Ppts_{i}), X(Ppts_{i+1}), Y(Ppts_{i+1}))$
inp = $\text{LineIntersection}(Ppts_{i}, \frac{v_0}{|v_0|}, Ppts_{i+1}, \frac{v_1}{|v_1|})$
if inp = $\emptyset$
  $Silpts_{i+1} = Ppts_{i}$
else
  $Silpts_{i+1} = inp$
$v_2 = \text{Vector}(X(Ppts_{i+2}), Y(Ppts_{i+2}), X(Ppts_{i+3}), Y(Ppts_{i+3}))$
$v_3 = \text{Vector}(X(Ppts_{i+2}), Y(Ppts_{i+2}), X(Ppts_{i+3}), Y(Ppts_{i+3}))$
inp = $\text{LineIntersection}(Ppts_{i+2}, \frac{v_2}{|v_2|}, Ppts_{i+3}, \frac{v_3}{|v_3|})$
if inp = $\emptyset$
  $Silpts_{i+3} = Ppts_{i+2}$
else
  $Silpts_{i+3} = inp$
return CreatePointList($Silpts_{1}, Silpts_{2}, Silpts_{i+3}$)

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

\[
\begin{align*}
y &= 3x^2 + 6 \\
y &= 5x^2 - 4
\end{align*}
\]
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