Ink Segmentation

Lecture #7: Ink Segmentation
Joseph J. LaViola Jr.
Fall 2012

Recall Pen-Based Interface Dataflow

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

\[
y = \frac{1}{2} x^4 \\
y = x^4 e^{-\frac{1}{2} t}
\]

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

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

\[
\text{ROBUSTINTERSECTION}(s_i, CS) \\
\begin{align*}
(1) & \quad P \leftarrow \text{Points}(s_i) \\
(2) & \quad cs_1 \leftarrow \text{Circle}(P_1, \frac{\text{PenInkWidth}(i)}{2}) \\
(3) & \quad cs_2 \leftarrow \text{Circle}(P_n, \frac{\text{PenInkWidth}(i)}{2}) \\
(4) & \quad sil_1 \leftarrow \text{Polygon}(	ext{ComputeStrokeEdges}(s_i)) \\
(5) & \quad \text{foreach Stroke } stk \in CS \\
\end{align*}
\]

\[
\begin{align*}
(6) & \quad Q \leftarrow \text{Points}(stk) \\
(7) & \quad cstk_1 \leftarrow \text{Circle}(Q_1, \frac{\text{PenInkWidth}(i)}{2}) \\
(8) & \quad cstk_2 \leftarrow \text{Circle}(Q_n, \frac{\text{PenInkWidth}(i)}{2}) \\
(9) & \quad sil_2 \leftarrow \text{Polygon}(	ext{ComputeStrokeEdges}(stk)) \\
(10) & \quad \text{if } cs_1 \cap cstk_1 \text{ or } cs_1 \cap cstk_2 \text{ or } cs_1 \cap sil_2 \text{ or } cs_2 \cap cstk_1 \text{ or } cs_2 \cap cstk_2 \\
& \quad \text{ or } cs_2 \cap sil_2 \text{ or } sil_1 \cap cstk_1 \text{ or } sil_1 \cap cstk_2 \text{ or } sil_1 \cap sil_2 \\
& \quad \quad \text{return true} \\
(11) & \quad \text{return false}
\end{align*}
\]

Robust Stroke Intersection (Part 2)

**Input:** Stroke \( s_i \)

**Output:** A list of silhouette points

\[
\text{COMPUTESTROKEEDGES}(s_i) \\
\begin{align*}
(1) & \quad P \leftarrow \text{Points}(s_i) \\
(2) & \quad penw \leftarrow \frac{\text{PenInkWidth}(i)}{2} \\
(3) & \quad \text{if } n < 3 \\
(4) & \quad \text{return } P \\
(5) & \quad \text{for } i = 1 \text{ to } n - 1 \\
(6) & \quad v_1 \leftarrow \text{Vector}(Y(P_{i+1}) - Y(P_i), -(X(P_{i+1}) - X(P_i))) \\
(7) & \quad v_2 \leftarrow \text{Vector}(-Y(P_{i+1}) - Y(P_i), X(P_{i+1}) - X(P_i)) \\
(8) & \quad Ppts1_i \leftarrow P_i + penw \frac{v_1}{||v_1||} \\
(9) & \quad Ppts2_i \leftarrow P_i + penw \frac{v_2}{||v_2||} \\
(10) & \quad \text{if } i = n - 1 \\
(11) & \quad Ppts1_i \leftarrow P_{i+1} + penw \frac{v_1}{||v_1||} \\
(12) & \quad Ppts2_i \leftarrow P_{i+1} + penw \frac{v_2}{||v_2||}
\end{align*}
\]
Robust Intersection (Part 2) – cont’d

for \( i = 1 \) to \( n - 1 \)

\( i = 1 \)

\( Silpts1_{i} = Ppts1_{i} \)

\( Silpts2_{i} = Ppts2_{i} \)

continue

\( i = n - 1 \)

\( Silpts1_{n+1} = Ppts1_{n+1} \)

\( Silpts2_{n+1} = Ppts2_{n+1} \)

continue

\( v_1 \leftarrow \text{Vector}(X(Ppts1_{i-1}) - X(Ppts1_{i}), Y(Ppts1_{i-1}) - Y(Ppts1_{i})) \)

\( v_2 \leftarrow \text{Vector}(X(Ppts1_{i+1}) - X(Ppts1_{i}), Y(Ppts1_{i+1}) - Y(Ppts1_{i})) \)

\( \text{intpt} \leftarrow \text{LineIntersection}(Ppts1_{i}, \frac{v_1}{||v_1||}, Ppts1_{i+1}, \frac{v_2}{||v_2||}) \)

if \( \text{intpt} = \emptyset \)

\( Silpts1_{i} = Ppts1_{i} \)

else

\( Silpts1_{i} = \text{intpt} \)

\( v_3 \leftarrow \text{Vector}(X(Ppts2_{i-1}) - X(Ppts2_{i}), Y(Ppts2_{i-1}) - Y(Ppts2_{i})) \)

\( v_4 \leftarrow \text{Vector}(X(Ppts2_{i+1}) - X(Ppts2_{i}), Y(Ppts2_{i+1}) - Y(Ppts2_{i})) \)

\( \text{intpt} \leftarrow \text{LineIntersection}(Ppts2_{i}, \frac{v_3}{||v_3||}, Ppts2_{i+1}, \frac{v_4}{||v_4||}) \)

if \( \text{intpt} = \emptyset \)

\( Silpts2_{i} = Ppts2_{i} \)

else

\( Silpts2_{i} = \text{intpt} \)

return CreatePointList(Silpts1, Silpts2, Silpts1_{0})

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

\[
y = 3x^3 + 6
\]

\[
y = 5x^2 - 4
\]
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

- Peterson, E., Stahovich, T., Doi, E., and Alvarado, C. Grouping Strokes into Shapes in Hand-Drawn Diagrams Proc. of the 24th AAAI Conference on Artificial Intelligence (AAAI-10), 2010, pp. 974-979