Ink Segmentation

Lecture #7: Ink Segmentation
Joseph J. LaViola Jr.
Fall 2015

Recall Pen-Based Interface Dataflow
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

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

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

Robust Stroke Intersection (Part 2)

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

ComputeStrokeEdges($s_i$)
1. $P \leftarrow Points(s_i)$
2. $pen_w \leftarrow PenInkWidth()/2$
3. if $n < 3$
4. return $P$
5. for $i = 1$ to $n - 1$
6. $v_1 \leftarrow Vector(Y(P_{i+1}) - Y(P_i), -(X(P_{i+1}) - X(P_i)))$
7. $v_2 \leftarrow Vector(-(Y(P_{i+1}) - Y(P_i)), X(P_{i+1}) - X(P_i))$
8. $Pts1_i \leftarrow P_i + pen_w \|v_1\|$
9. $Pts2_i \leftarrow P_i + pen_w \|v_2\|$
10. if $i = n - 1$
11. $Pts1_i \leftarrow P_{i+1} + pen_w \|v_1\|$
12. $Pts2_i \leftarrow P_{i+1} + pen_w \|v_2\|$
Robust Intersection (Part 2) – cont’d

for $i = 1$ to $n - 1$

if $i = 1$

$S_i = P_{i,1}$

$S_i = P_{i,2}$

continue

if $i = n - 1$

$S_i = P_{i,1} + 1$

$S_i = P_{i,2} + 1$

continue

$v_i \leftarrow \text{Vector}(X(P_{i-1,1}) - X(P_{i,1}), Y(P_{i-1,1}) - Y(P_{i,1}))$

$v_i \leftarrow \text{Vector}(X(P_{i+1,1}) - X(P_{i,1}), Y(P_{i+1,1}) - Y(P_{i,1}))$

$\text{intpt} \leftarrow \text{LineIntersection}(P_{i,1}, \frac{v_i}{|v_i|}, P_{i+1,1}, \frac{v_i}{|v_i|})$

if $\text{intpt} = \emptyset$

$S_i = P_{i,1}$

else

$S_i = \text{intpt}$

$v_i \leftarrow \text{Vector}(X(P_{i+1,2}) - X(P_{i,2}), Y(P_{i+1,2}) - Y(P_{i,2}))$

$v_i \leftarrow \text{Vector}(X(P_{i+2,2}) - X(P_{i,2}), Y(P_{i+2,2}) - Y(P_{i,2}))$

$\text{intpt} \leftarrow \text{LineIntersection}(P_{i,2}, \frac{v_i}{|v_i|}, P_{i+2,2}, \frac{v_i}{|v_i|})$

if $\text{intpt} = \emptyset$

$S_i = P_{i,2}$

else

$S_i = \text{intpt}$

return $\text{CreatePointList}(S_i, S_i, S_i, S_i)$

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
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