Ink Segmentation

Lecture #7: Ink Segmentation
Joseph J. LaViola Jr.
Fall 2008

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

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

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. $cs_1 \leftarrow Circle(P_1, \text{PenInkWidth}/2)$
3. $cs_2 \leftarrow Circle(P_n, \text{PenInkWidth}/2)$
4. $sil_i \leftarrow Polygon(ComputeStrokeEdges(s_i))$
5. **foreach** Stroke $stk \in CS$
6. $Q \leftarrow Points(stk)$
7. $cstk_1 \leftarrow Circle(Q_1, \text{PenInkWidth}/2)$
8. $cstk_2 \leftarrow Circle(Q_n, \text{PenInkWidth}/2)$
9. $sil_2 \leftarrow Polygon(ComputeStrokeEdges(stk))$
10. **if** $cs_1 \cap cstk_2 \text{ or } cs_1 \cap cstk_1 \text{ or } cs_2 \cap cstk_1 \text{ or } cs_2 \cap cstk_2$
    
    or $cs_2 \cap sil_2 \text{ or } sil_1 \cap cstk_1 \text{ or } sil_1 \cap cstk_2 \text{ or } sil_1 \cap sil_2$
11. \hspace{1cm} return true
12. \hspace{1cm} 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. $pen_w \leftarrow \text{PenInkWidth}/2$
3. **if** $n < 3$
4. \hspace{1cm} return $P$
5. **for** $i = 1 \text{ to } n - 1$
6. \hspace{1cm} $\vec{v}_1 \leftarrow \text{Vector}(Y(P_{i+1}) - Y(P_i), -(X(P_{i+1}) - X(P_i)))$
7. \hspace{1cm} $\vec{v}_2 \leftarrow \text{Vector}(-Y(P_{i+1}) - Y(P_i), X(P_{i+1}) - X(P_i))$
8. \hspace{1cm} $P_{pts1_i} \leftarrow P_i + pen_w \frac{\vec{v}_1}{||\vec{v}_1||}$
9. \hspace{1cm} $P_{pts2_i} \leftarrow P_i + pen_w \frac{\vec{v}_2}{||\vec{v}_2||}$
10. **if** $i = n - 1$
11. \hspace{1cm} $P_{pts1_i} \leftarrow P_{i+1} + pen_w \frac{\vec{v}_1}{||\vec{v}_1||}$
12. \hspace{1cm} $P_{pts2_i} \leftarrow P_{i+1} + pen_w \frac{\vec{v}_2}{||\vec{v}_2||}$
Robust Intersection (Part 2) – cont’d

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

if $i = 1$
  $SIpts[1] = Ppts[1]$
  continue

if $i = n - 1$
  $SIpts[i+1] = Ppts[1+1]$
  $SIpts[i+2] = Ppts[2+1]$
  continue

$v_1 \leftarrow$ Vector($X(Ppts[i-1]), Y(Ppts[i-1]) - X(Ppts[i]), Y(Ppts[i])$)
$v_2 \leftarrow$ Vector($X(Ppts[i+1]), Y(Ppts[i+1]) - X(Ppts[i+2]), Y(Ppts[i+2])$)

intpt $\leftarrow$ LineIntersection($Ppts[1], \overrightarrow{v_1}, Ppts[1+1], \overrightarrow{v_2}$)

if intpt $\neq \emptyset$
  $SIpts[1] = Ppts[1]$
else
  $SIpts[1] = intpt$

$v_1 \leftarrow$ Vector($X(Ppts[i+1-1]), Y(Ppts[i+1-1]) - X(Ppts[i+1]), Y(Ppts[i+1])$)
$v_2 \leftarrow$ Vector($X(Ppts[i+2+1]), Y(Ppts[i+2+1]) - X(Ppts[i+2]), Y(Ppts[i+2])$)

intpt $\leftarrow$ LineIntersection($Ppts[i+1], \overrightarrow{v_1}, Ppts[i+2+1], \overrightarrow{v_2}$)

if intpt $\neq \emptyset$
  $SIpts[i+1] = Ppts[i+1]$
else
  $SIpts[i+1] = intpt$

return CreatePointList($SIpts[1], SIpts[2], SIpts[1+2]$)

---

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