Recall Pen-Based Interface Dataflow

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

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

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. \( cs_1 \leftarrow \text{Circle}(P_1, \frac{\text{PenInkWidth}}{2}) \)
3. \( cs_2 \leftarrow \text{Circle}(P_n, \frac{\text{PenInkWidth}}{2}) \)
4. \( sil_1 \leftarrow \text{Polygon} (\text{ComputeStrokeEdges}(s_i)) \)
5. **foreach** Stroke \( stk \in CS \)
6. \( Q \leftarrow \text{Points}(stk) \)
7. \( cstk_1 \leftarrow \text{Circle}(Q_1, \frac{\text{PenInkWidth}}{2}) \)
8. \( cstk_2 \leftarrow \text{Circle}(Q_n, \frac{\text{PenInkWidth}}{2}) \)
9. \( sil_2 \leftarrow \text{Polygon} (\text{ComputeStrokeEdges}(stk)) \)
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 \( cs_2 \cap sil_2 \) or \( sil_1 \cap cstk_1 \) or \( sil_1 \cap cstk_2 \) or \( sil_1 \cap sil_2 \)
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. \( \text{pen}_{\mu} \leftarrow \frac{\text{PenInkWidth}}{2} \)
3. **if** \( n < 3 \)
4. **return** \( P \)
5. **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. \( P_{pts1_i} \leftarrow P_i + \text{pen}_{\mu} \left| \vec{v}_1 \right| \)
9. \( P_{pts2_i} \leftarrow P_i + \text{pen}_{\mu} \left| \vec{v}_2 \right| \)
10. **if** \( i = n - 1 \)
11. \( P_{pts1_{n}} \leftarrow P_{i+1} + \text{pen}_{\mu} \left| \vec{v}_1 \right| \)
12. \( P_{pts2_{n}} \leftarrow P_{i+1} + \text{pen}_{\mu} \left| \vec{v}_2 \right| \)
Robust Intersection (Part 2) – cont’d

for $i = 1$ to $n - 1$
if $i = 1$
$S\ell pts_{i-1} = P\ell pts_{1}$
$S\ell pts_{i} = P\ell pts_{2}$
continue
if $i = n - 1$
$S\ell pts_{n,1} = P\ell pts_{i+1}$
$S\ell pts_{n,1} = P\ell pts_{i+2}$
continue
$v_{3} \leftarrow Vector(X(P\ell pts_{i-1}) - X(P\ell pts_{i}), Y(P\ell pts_{i-1}) - Y(P\ell pts_{i}))$
$v_{4} \leftarrow Vector(X(P\ell pts_{i+1}) - X(P\ell pts_{i}), Y(P\ell pts_{i+1}) - Y(P\ell pts_{i}))$
$\text{inplt} \leftarrow LineIntersection(P\ell pts_{i}, \frac{v_{3}}{|v_{3}|}, P\ell pts_{i+1}, \frac{v_{4}}{|v_{4}|})$
if $\text{inplt} = \emptyset$
$S\ell pts_{i} = P\ell pts_{i}$
else
$S\ell pts_{i-1} = \text{inplt}$
$v_{3} \leftarrow Vector(X(P\ell pts_{i+2}) - X(P\ell pts_{i+1}), Y(P\ell pts_{i+2}) - Y(P\ell pts_{i+1}))$
$v_{4} \leftarrow Vector(X(P\ell pts_{i}) - X(P\ell pts_{i+1}), Y(P\ell pts_{i}) - Y(P\ell pts_{i+1}))$
$\text{inplt} \leftarrow LineIntersection(P\ell pts_{i+1}, \frac{v_{3}}{|v_{3}|}, P\ell pts_{i+2}, \frac{v_{4}}{|v_{4}|})$
if $\text{inplt} = \emptyset$
$S\ell pts_{i+1} = P\ell pts_{i+1}$
else
$S\ell pts_{i+1} = \text{inplt}$
$S\ell pts_{i+2} = \text{inplt}$
return CreatePointList($S\ell pts_{1}, S\ell pts_{2}, S\ell pts_{n}$)

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