Graph-based Symbol Recognizer (Thesis of Levent Burak Kara from CMU)

WeeSan Lee
Feb 15, 2005

Outline

- Background
- Symbol Representation
- Training and Recognition
- Error Metrics
- Graph Matching Problem
- Potential Solutions
- Conclusion

Background

1. Image-based symbol recognizer
   – Processes on bitmap images
   ✔ Ability to train with one sample
   ✗ Computation expensive
2. Feature-based symbol recognizer
   – Needs ink segmentation
   – Extracts geometric properties of lines and arcs
   – Based on Gaussian Distribution
   ✔ Fast recognition after training
   ✗ Different symbols might share the same features

Background (cont)

3. Graph-based symbol recognizer
   – Needs ink segmentation
   – Creates attributed relational graph
   ✔ Ability to recognize symbols with small details
   ✗ Slower than feature-based, difficult to train

Background (cont)

- Comparison of 3 recognizers:

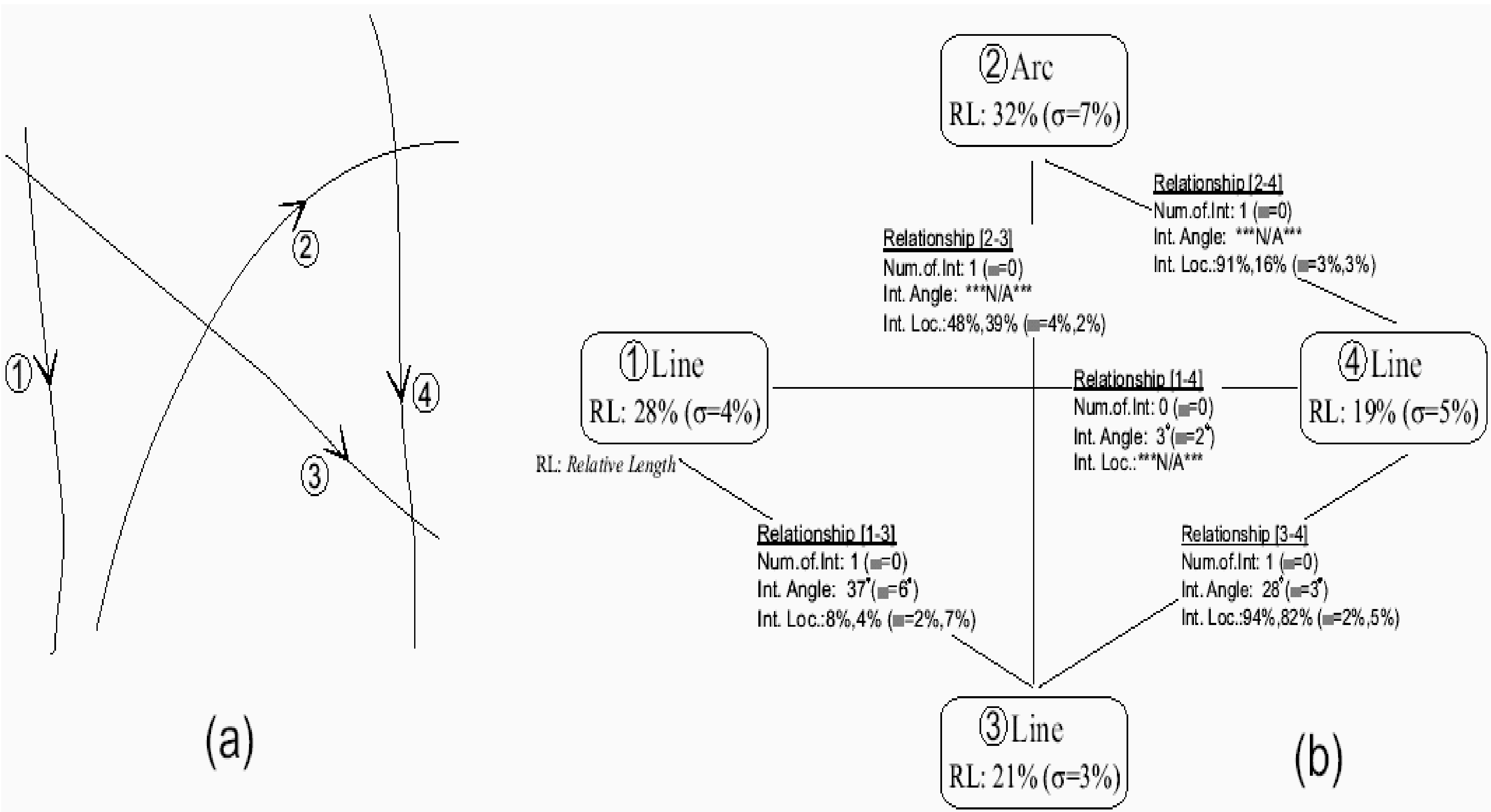
		Image-based	Feature-based	Graph-Based
Attributes	Trainable	✓	✓	✓
	Rotation invariant	✓	✓	✓
	Size invariant	✓	✓	✓
	Non-uniform scale invariant	✗	✓	✓
	Requires segmentation	✗	✓	✓
	Handles overtracing	✓	✗	✗
Performance Metrics	Ease of trainability	★★★★★	★★★★	★★
	Recognition speed	★★	★★★★★	★★★★

http://www.cs.ucr.edu/~weesan/smart_tools/graph_based_symbol_recognizer.pdf

Symbol Representation

- Intrinsic Properties
  - Primitive/Segment Type: line or arc
  - Relative Length
- Relationship Properties
  - Number of Intersections: 0, 1 or 2
  - Intersection Angle: 0 – 90°, only between 2 lines
  - Intersection Location
- Use lines and arcs as nodes, relationship properties as links in the relational graph

Symbol Representation (cont)



http://www.cs.ucr.edu/~weesan/smart_tools/graph_based_symbol_recognizer.pdf

Symbol Representation (cont)

	Prim. 1	Prim. 2	Prim. 3	Prim. 4
Prim. 1	Line RL: 28% (σ=4%)	Line-Arc Num.of.Int:0(s=0) Int.Angle:***N/A*** Int.Loc.:***N/A***	Line-Line Num.of.Int:1(s=0) Int.Angle:37°,σ=6° Int.Loc.:8%,4% (σ=2%,7%)	Line-Line Num.of.Int:0(s=0) Int.Angle:3°,σ=2° Int.Loc.:***N/A***
Prim. 2		Arc RL: 32% (σ=7%)	Arc-Line Num.of.Int:1(s=0) Int.Angle:***N/A*** Int.Loc.:48%,39% (σ=4%,2%)	Arc-Line Num.of.Int:1(s=0) Int.Angle:***N/A*** Int.Loc.:91%,16% (σ=3%,3%)
Prim. 3			Line RL: 21% (σ=3%)	Line-Line Num.of.Int:1(s=0) Int.Angle:28°,σ=3° Int.Loc.:91%,16% (σ=3%,3%)
Prim. 4				Line RL: 19% (σ=5%)

Training and Recognition

- Training
  - A process to build a relationship matrix from several drawing samples and store it into a DB
  - Require same drawing order for the same symbol
- Recognition
  - Same as training, but for unknown symbol
  - Instead of storing, the relationship matrix is used to compare against those in the DB
  - Uses 6 error metrics
  - With least Dissimilarity Score = $\sum_{i=1}^{6} w_i \cdot E_i$

Error Metrics

- 6 error metrics & theirs weights
  - Primitive Count Error (20%)
  - Primitive Type Error (20%)
  - Relative Length Error (20%)
  - Number of Intersections Error (15%)
  - Intersection Angle Error (15%)
  - Intersection Location Error (10%)
- The weights are assigned empirically
- All errors are in the range of [0, 1], 0 is better

Primitive Count Error

- Measures the differences in the number of primitives in 2 symbols

$$Primitive\ Count\ Error = \min(1.0, \frac{|N_U - N_D|}{\min(N_U, N_D)})$$

- N_U – the # of primitives in unknown symbol
- N_D – the # of primitives in definition symbol
- The min with 1.0 is to make sure the value is between [0,1]

http://www.cs.ucr.edu/~weesan/smart_tools/graph_based_symbol_recognizer.pdf

Primitive Type Error

- Measures the difference between the types (line vs arc) of the primitives in 2 symbols

$$Primitive\ Type\ Error = \frac{\sum_{i=1}^{\min(N_U,N_D)} \delta(Type(U_i), Type(D_i))}{\min(N_U, N_D)}$$

- $\delta(x,y)$ = 1 when x != y, 0 otherwise
- Sensitive to drawing order & missing segment
- Eg. Primitive Type Error = 4/11

Unknown	L-L-A-L-L-A-A-A-L-L-L-*I*-*I*-*I*-*I*
Definition	L-A-L-L-L-L-A-A-L-L-A-L-L-A-A

http://www.cs.ucr.edu/~weesan/smart_tools/graph_based_symbol_recognizer.pdf

Relative Length Error

- Compares the relative lengths of each unknown primitive to those in the definition's
- Probabilistic Definition Function:

$$P(x) = exp[-\frac{1}{50.0} \cdot \frac{(x-m)^4}{\sigma^4}]$$

- *m* and *σ* are the mean and standard deviation from the training samples

$$Relative\ Length\ Error = \frac{\sum_{i=1}^{\min(N_U,N_D)} 1 - P(u^i_{RL})}{\min(N_U, N_D)}$$

- U^i_{RL} is the relative length of the i primitives of the unknown symbol

Number of Intersections Error

- Compare the difference between the number of intersections of 2 symbols

$$Average\ Intersection\ Difference = \frac{\sum_{i=1}^{\min(N_U,N_D)} \sum_{j=1}^{i} |NumInt(U_i,U_j) - NumInt(D_i,D_j)|}{\min(NR_U, NR_D)}$$

- The average above ranges from [0,2]
- Need a Squashing Function to make it [0,1]

$$S(x) := \frac{1}{1 + \exp[6(1 - x)]}$$

- Thus, the *error* is defined as:
  - *S(Average Intersection Difference)*

Intersection Angle Error

- Compare the acute angles of 2 lines in 2 symbols
- Uses Probabilistic Definition Function
- The error is defined as:
  - The average of the errors accumulated from the line pairs considered

Intersection Location Error

- Measures the average difference between the location of intersections in 2 symbols
- Uses Probabilistic Definition Function
- Thus, the error is defined as:

$$Primitive\ Location\ Error = \frac{[1 - P(A)] + [1 - P(B)]}{2}$$

- Sensitive to drawing direction, thus, consider reverse direction as well
- Take the minimum error

http://www.cs.ucr.edu/~weesan/smart_tools/graph_based_symbol_recognizer.pdf

Graph Matching Problem

- 5 out of 6 error matrices are sensitive to drawing order
- Swapping 2 primitives in definition symbol to find better match with unknown symbol
- Uses Stochastic Swapping Algorithm
- Hardcoded 100 tries, returns definition symbol with minimum errors

Potential Solutions

- Swaps primitives with the highest errors first
- Establish a “graph order”
  - Sorted by primitive types then node degree
  - Swaps primitives with same/similar node degree
- Replace Error Metrics sensitive to drawing order with something else
  - Become feature-based recognizer?
- Other suggestions?

Conclusion

- Needs ink segmentation
- Abstract intrinsic and relationship properties
- Needs at least 10-15 training samples
- Uses 6 Error Metrics for recognition
- Reasonable accuracy & response time
- Sensitive to drawing order, need a way to fix it
- Not sure if it's better than feature-based recognizer

http://www.cs.ucr.edu/~weesan/smart_tools/graph_based_symbol_recognizer.pdf