What!?! No Rubine Features?: Using Geometric-based Features to Produce Normalized Confidence Values for Sketch Recognition

Brandon Paulson[†], Pankaj Rajan[†], Pedro Davalos[§], Ricardo Gutierrez-Osuna[‡], Tracy Hammond[†]
Sketch Recognition Lab[†]
Pattern Recognition and Intelligent Sensor Machines Lab[‡]
Spacecraft Technology Center[§]
Texas A&M University
3112 TAMU
College Station, TX 77843 USA
{bpaulson, pankaj, p0d9861, rgutier, hammond}@cs.tamu.edu

Abstract

As pen-based interfaces become more popular in today's applications, the need for algorithms to accurately recognize hand-drawn sketches and shapes has increased. In many cases, complex shapes can be constructed hierarchically as a combination of smaller primitive shapes meeting certain geometric constraints. However, in order to construct higher level shapes, it is imperative to accurately recognize the lower-level primitives. Two approaches have become widespread in the sketch recognition field for recognizing lower-level primitives: gesture-based recognition and geometric-based recognition. Our goal is to use a hybrid approach that combines features from both traditional gesture-based recognition systems and geometric-based recognition systems. In this paper, we show that we can produce a system with high recognition rates while providing the added benefit of being able to produce normalized confidence values for alternative interpretations; something most geometric-based recognizers lack. More significantly, results from feature subset selection indicate that geometric features aid the recognition process more than gesture-based features when given naturally sketched data.

1. Introduction

Hardware supporting pen-based input has become popular in recent years as Tablet PCs, SmartBoards, and touch screens are becoming common modes of input for many applications. Sketches are used in a variety of domains to help convey ideas and designs. Computer-aided design (CAD) tools have been created to allow the visualization of abstract ideas; however, traditional CAD applications use less than intuitive interfaces. Sketching is a very natural choice as an alternative to the multi-tool selection paradigm. Many tools have been created which allow sketching to be easily incorporated into user interfaces [12, 16]. However, in order for sketching to be an effective means of input, we must develop recognizers that can accurately determine the intention of a sketcher.

Two approaches have become standard to solving the sketch recognition problem. The first approach treats input sketches simply as two-dimensional gestures [1, 15, 18]. These gesture-based recognition techniques typically focus on how a sketch was drawn rather than on what the final sketch actually looks like. The typical goal of these systems is to take an input stroke (a sampling of points in the form of x, y, and time value) and classify each one into a set of pre-defined gestures. This approach has the benefit of using mathematically sound classifiers which produce fast classifications along with normalized confidence values, but has the disadvantage of using feature sets which are user-dependent and require individual training by each user to give good recognition results. Furthermore, many of these gesture-based features produce systems which are very sensitive to changes in scale and rotation.

The second approach has been to describe shapes geometrically, focusing on what the sketch looks like and less on how it was actually drawn [14, 17, 19]. Essentially, these geometric-based techniques are meant to take a single stroke as input and classify it as one of the predefined geometric primitives. These techniques are geometric in nature because they compare a stroke

VL/HCC Workshop: Sketch Tools for Diagramming
Herrsching am Ammersee, Germany
15 September 2008
Editors: Beryl Plimmer & Tracy Hammond

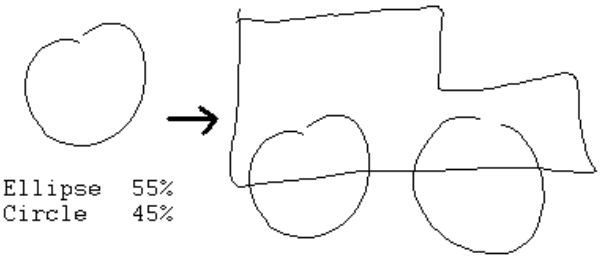

Figure 1. Example scenario in which higher-level context can disambiguate a lower-level interpretation. In this example, the high-level recognizer may realize that the ambiguous stroke is more likely to be a circle (which represents a wheel in this domain) than an ellipse because of context.

to an ideal representation of each primitive using formulas based on geometry. Primitives can then be combined hierarchically to form more complex shapes using specialized grammars [7]. Since the geometric tests used by these systems focus more on what the sketch looks like, these recognizers are typically more user- and style-independent. This also means that no individual (per user) training is necessary; the only training required is that which is necessary to determine numerous geometric thresholds. The disadvantage of such a system is that geometric-based recognizers typically use numerous thresholds and heuristic hierarchies which are difficult to analyze and optimize in a systematic fashion. Inferences about generalization are hard to determine because classification is not statistical. Furthermore, the use of multiple error measures on a per shape basis makes ranking alternative interpretations difficult [14].

Ranking alternative interpretations with a normalized confidence value can be important in aiding a higher-level recognition system, which often has access to context that can help resolve ambiguity in a lower-level interpretation [3, 6]. For example, imagine that a user draws a shape and the low-level recognizer returns a circle interpretation with an ellipse listed as an alternative interpretation. In addition, the low-level recognizer can also supply confidence values: 55% chance of an ellipse, 45% chance of a circle. The higher-level recognizer may now be more likely to choose the circle interpretation over the ellipse interpretation given that these confidence values are so close, if context indicates that an circle is a more likely interpretation. An example of this is shown in Figure 1.

Our initial goal for this work was to find a way to combine these gesture-based and geometric-based approaches in such a way as to take advantage of the positive aspects of each (accurate classification, user independent, mathematically sound, ability to produce normalized confidence values, etc.) while avoiding many of the disadvantages. During this process, we discovered that gesture-based features were less significant in aiding recognition on freely sketched data. This discovery, along with the production of normalized confidence values, is our main contribution.

As is the case in many previous works [14, 17, 19], we focused simply on classifying low-level primitive shapes. We have chosen to classify single strokes into one of nine different shape classes (arc, line, curve, circle, ellipse, helix, spiral, polyline, and complex). This is the shape set used in a previous geometric-based recognizer, PaleoSketch [14]. In this paper we show that combining the gesture-based features from Rubine [15] with geometric-based features and using a statistical classifier can yield accurate recognition results while producing normalized confidence values. We also perform feature subset selection to determine what features are the most significant for recognition. Our findings indicate that geometric-based features contribute more significantly to the recognition of naturally drawn sketch data than gesture-based features.

2 Previous Work

2.1 Gesture-based Recognition

In 1991, Dean Rubine introduced one of the first pen-based input gesture recognition systems, GRANDMA [15]. This toolkit allowed users to specify single stroke gestures that could be trained and learned through a simple linear classifier. Rubine proposed thirteen features which could be used to classify simple gestures with an accuracy of 98% on a fifteen-class gesture set when trained with at least fifteen examples per class.

Rubine's work was extended by [1]. In this paper, Long et al. added nine new features to Rubine's existing set. They performed multi-dimensional scaling to identify correlated features and ultimately found an optimal subset that contained eleven of Rubine's original thirteen features along with six of their own. Both of these works proved to be helpful in recognizing two-dimensional gestures, but when applied to natural sketch recognition problems, the accuracy of these approaches is not optimal.

The nature of the feature sets used by these recognizers requires that gestures be drawn the same way and to the same scale every time they are drawn. For example, a clockwise, circular gesture would not be the

```
(define shape Pendulum ...
  (components
    (Circle mass)
    (Line arm))
  (constraints
    (concentric arm.p1 mass) ...)
)
```

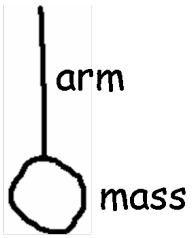

Figure 2. Example of a higher-level shape (pendulum) being constructed from lower-level primitives (line, circle).

same as a counter-clockwise, circular gesture. When treating sketched shapes as gestures these approaches do not perform well because they put constraints on how users must draw. Our goal is to be able to create recognition systems which are user-independent and allow users to draw as they would naturally, without having to worry about issues such as where to start a stroke or which direction to draw certain shapes. This is typically referred to as "free" or "natural" sketch recognition [8].

2.2 Geometric-based Recognition

Because of the drawing constraints imposed by gesture-based classifiers, the most recent shift in sketch recognition has been toward a geometric approach, which puts virtually no drawing constraints on the user. Essentially, shape grammars such as LADDER [7] can be used to define higher level shapes as a combination of lower level primitive shapes meeting certain geometric constraints. Figure 2 gives an example of how higher-level shapes can be constructed from lower-level primitives.

In order for these higher-level recognition schemes to be effective, it is important that the low-level primitive shapes are accurately recognized. Many geometric-based recognizers have been developed to recognize low-level primitive shapes [14, 17, 19]. Unlike gesture-based techniques, these recognizers do not use statistical classifiers. Instead, they focus on determining the error between a sketched shape and its ideal version using a series of geometric tests and formulas. Some recognizers focus on developing a universal error metric such as the feature area error metric [19]. However, universal error metrics can be hard to describe and compute for more complex primitive shapes such as spirals and helixes.

In order to support more primitives, some recognizers use different error metrics for each primitive [14]. These recognizers then rely upon heuristic hierarchies and numerous thresholds to order shape interpretations. Such an approach makes it hard to generalize and prove a recognizer's success for future sketches. It also makes producing normalized confidence values difficult (the biggest downfall of such an approach).

3. Features

As mentioned before, our goal is to combine gesture-based and geometric-based approaches in such a way as to take advantage of the benefits of both techniques. We will use a statistical classifier - namely a quadratic classifier [5] - that uses a feature set containing both gestural and geometric features. Our hypothesis is that this approach will lend itself more naturally to systematic optimization than using heuristic hierarchies, will maintain user independence because of the addition of geometric features, and will still enable us to return multiple, ranked interpretations with normalized confidence values.

Our feature set initially contained 44 total features. The first 31 features came from geometric tests described in [14]. The remaining 13 features are the classical gesture-based features used by Rubine [15]. The descriptions of each of these features are beyond the scope of this paper, so the interested reader is referred to [14, 15]. Table 1 gives a list of the features we used.

4. Data

To perform our tests, we used a dataset consisting of 1800 total sketch examples - the same dataset from [14]. Each example consists of a single stroke and is labeled with the intention of the original sketcher. A stroke is defined as the set of points (x-coordinate, y-coordinate, and time stamp) sampled between pen-down and pen-up events. The data samples came from 20 different users. Each user provided 90 samples (10 of each shape class). Figure 3 shows some examples from the dataset.

For our experiments we split the data set into two halves. The first half consisted of 900 examples sketched by 10 distinct users. The second half consisted of 900 examples from 10 different users. Rather than use a random 50/50 split of the overall data, we chose to split based on user to more accurately reflect how a sketch system would be used (i.e. the classifier is trained with data offline and then new users interact with the system without providing their own training data). This testing procedure allows us to determine the extent to which our algorithm is user-independent. The first half of the data was used to perform feature subset selection and training while the second set was left untouched until the validation of our final model.

1. Endpoint to stroke length ratio (100%)	**12. Curve least squares error (90%)**	**23. Spiral fit: avg. radius/bounding box radius ratio (60%)**	34. Length of bounding box diagonal (20%)
2. NDDE (90%)	**13. Polyline fit: # of sub-strokes (70%)**	**24. Spiral fit: center closeness error (70%)**	35. Angle of the bounding box diagonal (40%)
3. DCR (90%)	**14. Polyline fit: percent of sub-strokes pass line test (50%)**	25. Spiral fit: max distance between consecutive centers (20%)	36. Distance between endpoints (10%)
4. Slope of the direction graph (20%)	**15. Polyline feature area error (80%)**	26. Spiral fit: average radius estimate (10%)	37. Cosine of angle between endpoints (0%)
5. Maximum curvature (40%)	16. Polyline least squares error (30%)	27. Spiral fit: radius test passed (1.0 or 0.0) (40%)	38. Sine of angle between endpoints (10%)
6. Average curvature (30%)	17. Ellipse fit: major axis length estimate (20%)	**28. Complex fit: # of sub-fits (60%)**	39. Total stroke length (20%)
7. # of corners (30%)	18. Ellipse fit: minor axis length estimate (30%)	**29. Complex fit: # of non-polyline primitives (50%)**	**40. Total rotation (100%)**
8. Line least squares error (0%)	19. Ellipse feature area error (10%)	**30. Complex fit: percent of sub-fits that are lines (90%)**	41. Absolute rotation (10%)
9. Line feature area error (40%)	20. Circle fit: radius estimate (30%)	**31. Complex score / rank (50%)**	42. Rotation squared (10%)
10. Arc fit: radius estimate (0%)	**21. Circle fit: major axis to minor axis ratio (80%)**	32. Cosine of the starting angle (30%)	43. Maximum speed (20%)
11. Arc feature area error (20%)	22. Circle feature area error (0%)	33. Sine of the starting angle (10%)	44. Total time (30%)

Table 1. Features used by our recognizer. Implementation details for features 1-31 can be found in [14]. Details for features 32-44 can be found in [15]. Bold features are those chosen as the optimal subset through feature subset selection. Percentage values indicate how often a feature was chosen as optimal through various folds of subset selection.

5. Results

Our goal was to determine whether or not we could use a statistical classifier to classify single-stroke sketched primitives using a combination of gesture-based and geometric-based features. In addition, we wanted to determine if our approach produced classification rates that are comparable to the current best low-level system, PaleoSketch. In order to compare our approach directly to the PaleoSketch system, we have presented our results using the same 50/50 user split from [14]. However, we also wanted to determine if our final classifier was robust to other splits. Therefore, we have also presented results based on 25 folds of cross-validation where we combined the training and testing data into a single set and randomly selected 10 users for training and 10 users for testing. Table 2 shows the results of using the full feature set along with a quadratic classifier for each of the cases just mentioned. In this table, we also present the improved accuracy achieved through feature subset selection. Because of singularities in the data, the covariance matrix was regularized by adding a small value (0.001) to its diagonal [4].

As Table 2 shows, the full feature set alone did not provide the same accuracy reported in [14]. We can also see that the split used by the original PaleoSketch system is not favorable to the quadratic classifier using the full feature set. Fortunately, the quadratic classifier can be optimized by removing features which contribute negatively to recognition.

5.1. Feature Subset Selection

To determine relevant features, we employed a greedy, sequential forward selection (SFS) technique to perform feature subset selection [2]. The subset se-