

Feature Selection for Character Recognition (Homework 01)

Daniel Dixon

Department of Computer Science
Texas A&M University
dixondm@cs.tamu.edu

Abstract

Computer recognition of the hand-drawn sketches has numerous applications across many domains (animation, brainstorming, design, etc.). The input of characters for writing also applies. An accepted way of performing sketch recognition is to provide examples of the desired forms, determine the unique features of each, and use these features to build a classifier for any given candidate gesture. This paper presents an implementation of one such accepted method with addition of the new features for improved character recognition. With these features, the author was able to consistently achieve a 91.1% recognition rate on a training set of English character.

Introduction

Any computer user with almost any range of dexterity is able to take advantage of sketch recognition. It seeks to bridge the divide between natural hand motion and the computer interface by replacing a writing utensil, such as a pen or pencil, with a stylus or even a finger (though a mouse is of course quite capable). The education domain is one area that stands to gain much from such a tool. For example, a sketch recognition device, such as a consumer-ready tablet PC with the appropriate software, could be used for tutoring young students in learning handwriting, mathematical notation and formulas, or even basic drawing skills.

The particular domain that this paper focuses on is that of English character recognition, probably the most prevalent area of application. Character

recognition is used for personal digital assistants (PDAs), tablet PCs, and game consoles. Successful recognition of hand-drawn characters can be a replacement for a small portable device without a keyboard, or transcription of handwritten letters and notes.

Issues of importance with any gesture recognition algorithm are that computation not be excessive while accuracy and trainability are high. These are taken into consideration in this paper as the suggestions made do not hinder speed while improving recognition rates.

Previous Work

A feature-based, linear classifier for single-stroke recognition was introduced by [2] in 1991 under the acronym, GRANDMA. His implementation is trainable through example gestures and supports “eager recognition”, where it intends to determine the gesture as soon as possible while the user is drawing. A vector of features is abstracted from each gesture, whether for training or as input. As input, this vector is then used to calculate the weight for each class of gestures, with the highest weight being the recognizer’s class determination for that gesture. During training, these weights are determined after calculating the common covariance matrix of the corpus from those of each class of gestures. More of this implementation and the features selected is discussed in the following section. According to the author, his algorithm achieved rates of 98% accuracy, though such rates are arguable when considering his criteria for successful recognition. Rubine’s algorithm served as a basis for this work.

The goal of [1] was to determine the common attributes of gestures that might lead them to be considered visually similar by a human observer. This was attempted through two user studies where each participant was asked to determine the least similar gesture in a side-by-side comparison of three. Statistical analysis on the results led to the creation of two data models for determining visually similar gestures, which could then be used as advice for a gesture designer. The researchers implemented many of Rubine’s features and added six of their own. This paper used this work as a basis for comparison of recognition results.

In [3], the authors sought to build a recognizer that would work on the large domain of mathematical symbols. They deviated from the two aforementioned works with their recognizer, using elastic matching and a HMM, and their feature selection, using attributes such as the number of cusps and regional point density. To improve computation, “prototype pruning” was introduced to eliminate classes of gestures that a candidate gesture would likely not belong to. The authors claimed accuracy rates of 94% without prototype pruning and 91% with. The drop of the latter is warranted by the author with the algorithms dramatic decrease in computation. The choice of features in this work served as inspiration for this paper.

Implementation

IMPLEMENTATION OF RUBINE

Each gesture made by a user is a collection of data points captured by a drawing application. Each data point processed by the algorithm is a tuple of the x and y position, and the time. No pressure information was used. A class of gestures is those gestures that are considered to the same and is what the sketch recognition seeks to determine for a candidate gesture (i.e. the gesture drawn as input).

The author first reproduced the algorithms of Rubine and the feature selection of Long.

On the addition of each data point during the

drawing of a gesture, the features of that gesture are calculated. These features seek to describe the gesture for numerical comparison. Straight from Rubine’s implementation, these features are:

1. Cosine of the initial angle
2. Sine of the initial angle
3. Length of bounding box diagonal
4. Angle of the bounding box diagonal
5. Distance between the first and last point
6. Cosine of the angle between the first and last point
7. Sine of the angle between the first and last point
8. Total length of the gesture
9. Total angle traversed
10. Summation of the absolute value of the angle at each point
11. Summation of the square value for #10
12. Maximum speed of the gesture
13. Duration of the gesture

As Rubine states, these features were chosen empirically. The equations for each can be found in [2]. Each feature f combines to form a gesture’s feature vector. Classification is completed by the weights w of the class c that is able to maximize the Equation 1.

$$v_c = w_{c0} + \sum_{i=1}^F w_{ci} f_i \quad 0 \leq c < C$$

Equation 1

To train this linear classifier, the weights are determined for each class c in the total number of classes C in the gesture corpus. This begins by determining the mean feature vector of each class by averaging the feature vectors together. A covariance matrix is then calculated for each class by looping through the training gestures E of that class and determining the value of feature f_i in light of feature f_j for all the features.

$$\Sigma_{cij} = \sum_{e=0}^{E_c-1} (f_{c ei} - \bar{f}_{ci})(f_{c ej} - \bar{f}_{cj})$$

Equation 2

The common covariance matrix for the features across the entire course is then computed through a summation of each class' covariance matrix, normalized by the number of examples minus the number of classes C .

$$\Sigma_{ij} = \frac{\sum_{c=0}^{C-1} \Sigma_{\epsilon ij}}{-C + \sum_{c=0}^{C-1} E_{\epsilon}}$$

Equation 3

Lastly, this matrix is inverted and the weights w of each class c for each feature f is calculated as such:

$$w_{\epsilon j} = \sum_{i=1}^F (\Sigma^{-1})_{ij} \bar{J}_{\epsilon i} \quad 1 \leq j \leq F$$

$$w_{\epsilon 0} = -\frac{1}{2} \sum_{i=1}^F w_{\epsilon i} \bar{J}_{\epsilon i}$$

Equation 4

IMPLEMENTATION OF LONG

[1] also implemented Rubine's algorithms, but added a few features. The feature selection for this implementation became:

1. Cosine of the initial angle
2. Sine of the initial angle
3. *Length of bounding box diagonal*
4. *Angle of the bounding box diagonal*
5. Distance between the first and last point
6. Cosine of the angle between the first and last point
7. Sine of the angle between the first and last point
8. *Total length of the gesture*
9. Total angle traversed
10. Summation of the absolute value of the

angle at each point

11. Summation of the square value for #10
12. Aspect, or the absolute value of 45 degrees minus #4
13. Curviness, or the summation of all intersegment angles within the gesture whose absolute value was below the threshold of 19 degrees.
14. *Total angle traversed / total length*
15. Density metric #1 (#8/ #5)
16. Density metric #2 (#8/#3)
17. Non-subjective openness (#5/#3)
18. *Area of the bounding box*
19. *Logarithm of the area of the bounding box*
20. *Total angle / total absolute angle*
21. Logarithm of the total length
22. Logarithm of the aspect

The features that are not italicized are those chosen by [1] to be "significant" for determining visual similarity, based on their statistical analysis. All features were implemented since this paper sought to compare accuracy rates for recognition, and not which features were best for determining visual similarity.

IMPLEMENTATION OF NEW FEATURES

Considering the domain of character recognition, the author sought to create features that took advantage of the attributes of its members. Thus the following features were added to Rubine's list:

14. Cosine of the ending angle
15. Sine of the ending angle
16. Quadrant of the first point
17. Quadrant of the last point
18. Number of cusps
19. Cosine of the centroid to the starting point
20. Sine of the centroid to the starting point
21. Cosine of the centroid to the end point
22. Sine of the centroid to the end point

The equations for each are as follows:

$$f14 = \cos \theta = (x_n - x_{n-1}) / \sqrt{(x_n - x_{n-1})^2 + (y_n - y_{n-1})^2}$$

$$f15 = \sin \theta = (y_n - y_{n-1}) / \sqrt{(x_n - x_{n-1})^2 + (y_n - y_{n-1})^2}$$

$$f16 = \begin{cases} 1 = \textit{northwest} \\ 2 = \textit{northeast} \\ 3 = \textit{southwest} \\ 4 = \textit{southeast} \end{cases}$$

$f17$ = repeat #16 for end point

$f18$ = see [3] for reference

$$f19 = \cos \theta = (x_{cent} - x_1) / \sqrt{(x_{cent} - x_1)^2 + (y_{cent} - y_1)^2}$$

$$f20 = \sin \theta = (y_{cent} - y_1) / \sqrt{(x_{cent} - x_1)^2 + (y_{cent} - y_1)^2}$$

$$f21 = \cos \theta = (x_{cent} - x_n) / \sqrt{(x_{cent} - x_n)^2 + (y_{cent} - y_n)^2}$$

$$f22 = \sin \theta = (y_{cent} - y_n) / \sqrt{(x_{cent} - x_n)^2 + (y_{cent} - y_n)^2}$$

The centroid is the average of all the data points, not the center of the bounding box. The quadrants for $f16$ and $f17$ are determined using the bounding box of the gesture. The size of the quadrants, which quadrant the first and last point lie in, and the position of the centroid are recomputed with each new data point.

DATA COLLECTION

The implementation was performed offline from any type of drawing tool. A third-party tool was used for gathering samples from classmates and the general public on the Internet. This tool instructed the participant how to draw the sketch through an example image, thus all samples were drawn with the same motion (see Figure 1). One gesture was defined as the points collected between the actions of pen down and pen up, thus only single strokes are considered. To the best of the author's knowledge, no known processing was done on the points collected by the third-party tool. The training set consisted of 12-15 samples for each character of the English alphabet, with an additional three for the testing set. Each implementation pre-processed the data in the same way, which was to remove data points with duplicate times or positions.



Figure 1 - An example image of the character 'E' that a participant would repeat for data collection.

Results

With all the aforementioned features implemented along side Rubine's, this author's feature selection achieved an accuracy rate of 91.1% on the testing corpus of 88 gestures (26 characters x 3 gestures each).

Features Included	Recognition Rate
Rubine Only	68.4%
Add $f14$	70.9%
$f15$	77.2%
$f16$	83.5%
$f17$	84.8%
$f18$	86.1%
$f19$	88.6%
$f20$	89.9%
$f21$	91.1%
$f22$	91.1%

For this set of data, it was found that the number of cusps $f18$ did not make a difference when $f19-22$ were present. For comparison, this author was only able to achieve a recognition rate of 70.9% with Long's feature selection.

Discussion

This work was able to achieve a 91.1% recognition rate because its selection of features

that took advantage of common character attributes found in the English alphabet. Since the characters were drawn the same way each time, angles between select points make for unique features. Had the participants generating sample data not be instructed on how to make the gesture, these results may have been less. In fact, the measurement of angles at the end of a gesture might cause a lot of false negatives since a person would probably only draw more intently at the beginning.

These attributes also helped disregard other features. For example, most characters have a horizontal or vertical symmetry (ex: 'H', 'M'), therefore dividing a normalized bounding box into halves or quadrants to determine point density would have not be useful.

Future Work

To truly prove the worth of these features, they need to be tested against another corpus of sample gestures.

On an aside, some of the implementation time during this work was actually spent verifying the capabilities of the Adobe Flex 3 framework for sketch recognition (i.e. the sampling rate). The author is excited about the potential for such a platform.

Conclusion

Sketch recognition is an important component of

human-computer interaction, being a bridge interface between normal dexterity and computer operation. In this paper, the author implements Rubine's algorithm for sketch recognition and then presents a side-by-side comparison of features selected for the recognition of hand-drawn English characters. It was found that for this particular domain, the author's choice of features improved the recognition rates dramatically over those presented in [1] and [2].

REFERENCES

1. Long, J., Landay, J. A., Rowe, L. A., and Michiels, J. 2002 Visual Similarity of Pen Gestures. Technical Report. UMI Order Number: CSD-99-1069., University of California at Berkeley.
2. Rubine, D. 1991. Specifying gestures by example. SIGGRAPH Comput. Graph. 25, 4 (Jul. 1991), 329-337. DOI=<http://doi.acm.org/10.1145/127719.122753>
3. Watt, Stephen M., Xie, Xiaofang, Prototype pruning by feature extraction for handwritten mathematical symbol recognition, Maple Conference 2005, Maplesoft, 2005, pp. 423-437.