Recognition of Unistroke Gesture Sequences

I. Problem Statement

Gesture recognition is a sizable research area, due to the many uses and applications of gesture detection. One important domain where gestures are frequently encountered is law enforcement, where a significant concern is recognition of gang symbols in graffiti and other forms of hand-written communication. The United States Federal Bureau of Investigation's Safe Streets and Gang Unit commonly encounters handwritten communication using custom gestures [1].

Inspired by this problem of gesture recognition, we focused our efforts on a similar problem with common features. Our problem is recognition of a unistroke gesture sequence, defined as a continuous stroke (unistroke) consisting of multiple gestures defined *a priori* (see Figure 1). In our opinion, this problem description is analogous to identifying known gang symbols in grafitti markings. An additional constraint is to use little to no training data, analogous to the extremely small amount of evidence typically available to law enforcement officials.

II. Related Work

Yang et al [2] present work on recognition of individual gestures in unistroke and multistroke gesture sequences of digits by constructing an exhaustive set of HMMs. They train HMMs to recognize continuous gesture sequences defined *a priori*, which is a deficient approach for our problem. Robust individual gesture recognition systems exist, such as the mouse gesture recognition system developed by Tanguay [3]. However, the implementation is tailored to individual gesture recognition. Additionally, the $1 recognizer [4] is a single gesture recognition system which works with no training (the system has a built-in representative gesture training set), but the primary deficiency is its inability to recognize gesture sequences.

III. Approach

Recognizing a unistroke gesture sequence properly is difficult primarily due to the nature of the problem, described in the literature as an "inverse perception problem", which is an acknowledged "hard problem" [5]. An inverse perception problem, as described by Pizlo, "is about inferring the properties of the distal stimulus X given the proximal stimulus Y", and the problem is "ill-conditioned". A feature of an ill-conditioned problem most relevant here is the possibility of an infinite number of distal stimuli (intended gesture sequences) that produce the proximal stimulus (the 2-D point representation of the gesture sequence).

We created a prototype gesture recognizer with two modes of operation: (1) a training mode, in which the user selects a gesture and subsequently trains by providing additional gesture *templates*, and (2) a recognition mode, in which the user draws a unistroke gesture sequence for subsequent recognition. For the purposes of this study, and also for user convenience, we restrict the maximum number of gestures in the sequence to 3, but the system can accept any finite length input. A screenshot of the system in recognition mode is shown in the following figure:

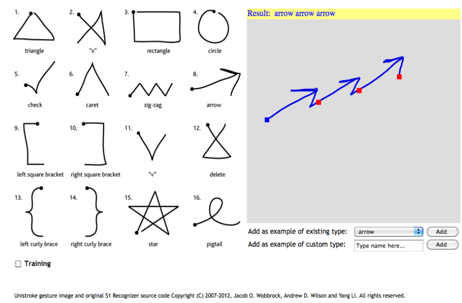

Figure 1: *Recognition GUI*

Our approach, at a high level, is to compare the *visual affinity* of an input with pre-existing templates. The first step is to match each template against a portion of the input in order to identify a "segmentation point" that splits the input into a matched gesture and the remainder. The second step is to remove the matched gesture from the input sequence and iterate, repeatedly searching for additional gestures in the remainder of the user input. A challenge to designing our approach was to account for variability in user input. A feature of the inverse perception problem appears in the form of segmentation ambiguity, as depicted in Figure 2.

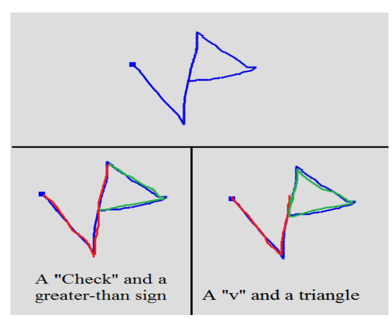

Figure 2: *Ambiguity in gesture recognition*

In the next section, we present details of our segmentation routine that determines the portion of the user's input that matches a template.

Segmentation

During recognition, the user draws a unistroke gesture sequence, which is recorded as an ordered list of (x, y) coordinates. The user's input, as noted above, may contain one or more individual gestures. However, the template, by definition, is one gesture. As a result, we need to compare a portion of the user's input with the template. The question is: what portion? An insight into our problem that helps answer this question is the observation that the same user provided both the template and the input, and that both are drawn to a similar scale. As a result, the user's input, if it contains this template, must have a path length approximately equal to or greater than the path length x of the template. Conversely, if the user's input has a path length less than x, then the input cannot contain the template. We repeatedly perform this calculation for each template; the templates that are a potential match are subsequently scored.

We score the difference between a portion of the user's input and the given template using Dynamic Time Warping (DTW, see next section for details). DTW essentially performs a flexible comparison between two inputs that have been sampled over time. In order to generate input suitable for DTW, we re-sample both the user's input and the template to 64 equidistant points. Although the points are equidistant (due to the matching considerations discussed above), the points are also ordered with respect to time, which allows us to provide them as input to the DTW calculation. Additionally, there are two benefits to re-sampling: 1) re-sampling smooths out noise, and 2) re-sampling with 64 points is enough to retain the resolution necessary to disambiguate gestures but also low enough to be computationally inexpensive. Additionally, this number was successfully used by the $1 Recognizer [4].

Our routine contains one other noteworthy optimization aimed at determining the most accurate segmentation point. After the routine determines an initial candidate segmentation point, as described above, it is possible that a nearby point is actually a better match (indicated by a smaller DTW score). We look for a better segmentation point (indicated by a smaller score) by searching backward and forward from the initial candidate point, repeatedly calculating the score. The segmentation point corresponding to the minimal score is marked as the point of segmentation, and the matched gesture is spliced from the input sequence. An example of the segmentation points for a unistroke gesture sequence is shown in Figure 1.

Dynamic Time Warping

As mentioned above, our algorithm for scoring template gestures is dynamic time warping (DTW). DTW is appropriate due to our need for flexible matching of resampled input and template gestures. DTW "warps" time to map each data point in inputs X to the "closest" matching point in input Y (as measured by the DTW distance calculation). Recall that our inputs have differing numbers of sample points (the template is resampled to 64 points, but a segment of the user's input, less than or equal to 64 points, is chosen to compare with the template). For our 2-D data, the appropriate distance metric is Euclidean distance. Using the minimal cost of neighboring points (we compare the distance score for three "nearby" points), we map each point i in input X to a nearest point j in input Y.

IV. Evaluation

The unique component of our system is our segmentation routine, which is tightly coupled to (and dependent upon) the accuracy of the individual gesture recognition algorithm (DTW). As a result, we evaluated our recognition system via three metrics, which are, as whole, intended to provide insight into the system function.

1. Overall Accuracy: This is an all-or-nothing measure. For input sequence of lengths 1, 2 or 3, we measure the percentage of trials in which the system outputs both the correct number and correct identity of individual gestures (a perfect match).

2. Segmentation Accuracy: For input sequence of lengths 1, 2 or 3, we measure the percentage of trials in which the system outputs the correct *number* of gestures (regardless of correct identification of individual gestures).

3. Relaxed Accuracy: This is a relaxed version of Overall Accuracy relevant only to gesture sequences of length 2 or 3, where we measure the number of gestures in the output that actually appear in the input, accounting for order. The purpose of this metric is to determine the percentage of trials where a portion of the input was correctly segmented and identified. For this metric, we count each gesture that was correctly identified, accounting for gesture order. For example, if the input unistroke gesture sequence is $check, star, \{$, and the system outputs $v, star, \{$, Relaxed Accuracy = 2, counting $star, \{$. Additionally, if the sequence was identified as $v, star, caret, \{$, the Relaxed Accuracy = 2, counting the $star$ and $\{$. The score assigned to a particular trial is:

$$RelaxedAccuracy = \frac{\text{Reported gestures in the input}}{\text{Total gestures in the input sequence}}$$

We conducted a micro-study involving three participants and three recognition phases. Each study participant chose 5 gestures out of the set of 16 to use for the study. Then each participant drew 10 unistroke gesture sequence of lengths 1, 2, and 3 (for a total of 30) after three phases of training for each chosen gesture: 1 template, 3 templates, 5 templates, and 10 templates. We calculated the three aforementioned accuracy metrics for the aggregate data. The results are shown in Tables 1,2 and 3 respectively.

Table 1: *Overall Accuracy*

Sequence Length	Accuracy Rate
1	57.5%
2	32.16%
3	25.72%

Table 2: *Segmentation Accuracy*

Sequence Length	Segmentation Accuracy
1	71.5%
2	71.5%
3	52.27%

Table 3: *Relaxed Accuracy*

Sequence Length	Accuracy Rate
2	56.19%
3	56.37%

Lastly, we calculated recognition accuracy for each training phase. Figure 3 displays Overall Accuracy (for sequences of length 1) and Relaxed Accuracy (for sequences of lengths 2 and 3) for the different recognition phases.

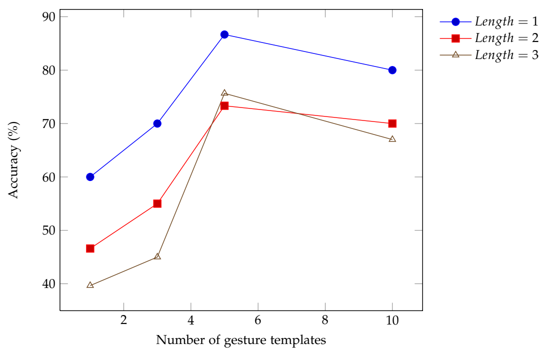

Figure 3: *Recognition Rate Vs. # of Gesture Templates*

V. Discussion

Tables 1, 2 and 3 reveal that our approach yields reasonable, but far from perfect, results. The accuracy rate for sequence of length 1 (57.5%) implies DTW is a useful approach for comparing two data sets that vary in time. This fact becomes clearer when we compare the DTW approach to the HMM approach taken by Tanguay [3], where he achieved a recognition accuracy of $50 - 60\%$ on lower-case English alphabet letters after extensive training. Note that the accuracy is lower for our problem than for the simplified problem of 1-to-1 gesture template matching due to the complexity introduced by allowing a gesture sequence. In the training mode, where only sequences of length 1 are allowed, DTW excels at labeling the input correctly.

Table 1 shows a drop-off in accuracy as the sequence length increases. This drop-off indicates a fundamental shortcoming in our approach: improper recognition of a gesture is concomitant with improper segmentation of the gesture sequence, leading to mis-recognition of subsequent gestures. Yang et al [2] report a nearly perfect recognition rate of up to 99.78% for sequences of digits up to length 2. The difference in approach is a key consideration when comparing these numbers. Yang et al train Hidden Markov models for both individual gestures as well as gesture sequences. It is probable that training on gesture sequences will yield significantly improved recognition rates.

Table 2 shows the percentage of trials that were correctly segmented into the correct number of gestures. Note that, for each length, the trials in which complete recognition was achieved (table 1) are a subset of the trials in which the correct number of gestures was identified (table 2). As a result, one can conclude that the difference in Overall Accuracy as compared to Segmentation Accuracy is due to incorrect recognition of individual gestures. A more robust technique for individual gesture recognition may significantly boost the Overall Accuracy.

The Relaxed Accuracy results in Table 3 can be compared with the corresponding Overall Accuracy results in Table 1. The increase in recognition accuracy indicates that the system "recovers" after a gesture is incorrectly identified. This is noteworthy primarily because it implies that additionally training intended specifically to address gesture ambiguity (including ambiguity with the 11 gestures the user did not choose) may significantly increase Overall Accuracy.

Figure 3 clearly indicates that additional training increases both Overall Accuracy and Relaxed Accuracy metrics. Importantly, "excessive" training (seen for 10 templates) may lead to incorrect recognition due to the "confusion" caused by having too many variations of an individual gesture; put another way, too many templates *adds* ambiguity, which supports our objective of training the system minimally.

We learned a great deal during this project. Most importantly, a significant constraint on possible approaches is added by our problem definition: train only using individual gestures. A relaxed problem definition, in which we would train on gesture sequences, would have allowed us to determine the features that indicate a segmentation point, such as a slight pause between gestures in a gesture sequence (which we observed during the study). A key insight is that the transition data itself is of great use when choosing among potential gesture segmentation points.

Reflecting upon the nature of the inverse perception problem, we hypothesize that the problem is defined by identification of a hidden state (the intended gesture sequence) corresponding to the "distal stimulus X" and observed via "proximal stimulus Y". Phrased in this manner, Hidden Markov Models are an ideal approach to this problem. Although HMMs require significant training, potential future work is to train HMMs on individual gestures and construct HMMs for gesture sequences by chaining the trained HMMs for all possible permutations of gestures. The trade-off is increased training time and computational complexity.

VI. References

[1] Lyddane, Donald. United states attorneys' bulletin. Technical report, United States Department of Justice Executive Office for United States Attorneys, May 2006.

[2] J. Yang, Y. Xu, and C. S. Chen. Gesture interface: Modeling and learning. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 1747–1752, 1994.

[3] D. O. Tanguay Jr. Hidden markov models for gesture recognition. Master's thesis, Massachusetts Institute of Technology, 1995.

[4] J.O. Wobbrock, A.D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 159–168. ACM, 2007.

[5] Zygmunt Pizlo. Perception viewed as an inverse problem. *Vision Research*, 41(24):3145–3161, November 2001.