

Recognition of Unistroke Gesture Sequences

I. PROBLEM STATEMENT

Gestures are a popular and growing form of input in human-centric user interfaces, primarily due to their natural use in everyday communication [1]. 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 [2]. We hope to address such a problem on a much smaller scale in our project, where an input gesture sequence, consisting of a set of known gestures, is segmented into its individual components, which are then recognized. Our objective is to solve this problem without constructing an exhaustive set of training data.

Recognizing a sequence of symbols is difficult primarily because of the segmentation problem, that is, the problem of analyzing a gesture sequence and identifying where each individual gesture begins and ends. Unlike the similar problem of cursive handwriting recognition, the sequence of gestures have no well-defined grammar (which English possesses to an extent), and thus cannot be leveraged during the process of recognition. Also gesture sequences do not have any obvious "breaks" to demarcate individual gestures, which makes segmentation more difficult to accurately implement.

II. RELATED WORK

Yang et al [3] present work on recognition of individual gestures in continuous gesture sequences containing arbitrary gestures (e.g., digits), by constructing an exhaustive set of HMMs. They train HMMs to recognize continuous gesture sequences defined *a priori*, and, because of their use of HMMs, require an intensive training period, both of which we see as deficiencies.

Robust individual gesture recognition systems exist, such as the mouse gesture recognition system developed by Tanguay [4]. However, the implementation is tailored to individual gesture recognition, and does not extended easily to recognize unistroke gesture sequences.

The \$1 recognizer [5] is a single gesture recognition system which works even without any training by the user (the system has a built-in gesture set which can be supplemented with additional training data tailored for the individual user). The primary deficiency of this system is its inability to recognize sequences of gestures.

III. APPROACH

There are two states in our prototype gesture recognizer: (1) a training state, where the user selects a gesture, and trains the chosen gesture by replicating it (usually not more than 5 times) and adding the data as a gesture template, and (2) a recognition state, where 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 be 3, but the system will work for any finite length of the input sequence as well.

A screenshot of the system in the recognition state is shown in the following figure:

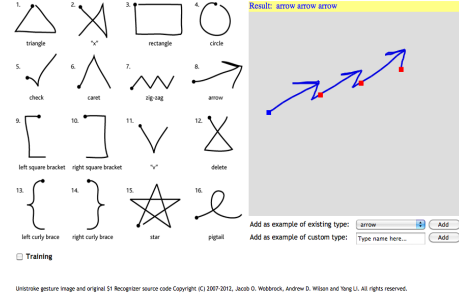


Figure 1: Recognition GUI

Our approach, at a high level, is very straightforward, and divided into two steps. The first step is to match each individual gesture against a portion of the input data, thereby segmenting the unistroke gesture sequence into the first gesture, and the remaining input. The second is to remove the first gesture from the input sequence and iterate, searching for additional gestures. The difficult part is accurately segmenting the unistroke sequence. The following section details our segmentation approach.

Segmentation

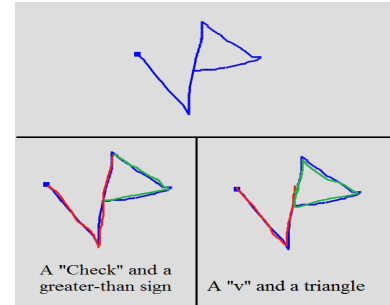


Figure 2: Ambiguity in gesture recognition

The main challenge in our approach was to come up with a segmentation technique which was neither too strict (as users generally are unable to repeatedly reproduce gestures exactly) nor too lenient (we do not want to recognize noise as a gesture). Additionally, we need an approach that is fairly robust to segmentation ambiguities. Because users are not constrained in their choice of gestures, ambiguity can exist regarding the appropriate segmentation of a gesture sequence, as shown in Figure 2.

During recognition, the user draws a unistroke gesture sequence, which is recorded as an ordered list of (x, y) coordinates. It is this sequence of ordered points that is

inspected for potential matches of individual gestures. First, each gesture recorded in the system is logically identified, and each sample of that logical gesture (a “template”) is compared with the user’s input, as follows. The template gesture is resampled to 64 equidistant points in order to provide a basis for direct pointwise comparison with the user input. Resampling is essential when comparing templates of *individual* gestures against the user input, which may contain *one or more* individual gestures. A fundamental insight into the problem that justifies our approach is that a sequence of gestures is going to “contain more data” (i.e., have a longer 2-D path length) than a single gesture. In addition to resampling the template gesture, the user’s input is also resampled, primarily to smooth out natural variations (jagged edges, etc.) in user input. We found that 64 points was both high enough to retain the resolution necessary to disambiguate gestures and low enough to be computationally inexpensive. Additionally, this number has been used and suggested in the \$1 Recognizer [5].

After resampling, the path length of the template is calculated and a threshold is set at 90% of the full path length. The purpose of the threshold is to ensure that we don’t require the full template data set to appear in the user’s input, which we felt was an intuitive and valid consideration (to see this, simply consider the scenario for a gesture sequence of length 1). The next step is to calculate the corresponding number of points in the user’s input whose path length approximately equals the thresholded template path length. Note that if 90% of the full path length is longer than the path length of the user input, the candidate gesture is deemed to not be a potential match, and skipped.

We now have a portion of the user input that may match a gesture template. To determine a quantitative measure, we first translate both the user’s input and the template to the origin and compare the segments pointwise using dynamic time warping (briefly reviewed in the next section). At this point, we have a candidate segment of user input with a score, but not necessarily a minimal score. To determine a minimal score, we search both backward and forward in the user input for a better match (indicated by a lower score). As long as the score of the tested segment is better than the previous score, we continue searching backward (forward) in the user input. The minimal score is computed, and retained for comparison against all other templates.

All of the candidate templates for the possible individual gestures are identified in this manner and then sorted in order of ascending score. The minimal score is the best match, and this is the gesture chosen as the first in the user’s input. An example of the results of this segmentation process is shown in Figure 1.

Dynamic Time Warping

As mentioned above, our algorithm for matching individual 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 sample in inputs X and Y to the “closest” match (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 the resampled 64 points, is chosen to compare with the template). For our 2-D data, the appropriate distance metric is Euclidean distance:

$$dist = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Using the minimal cost of neighboring points (we compare three nearby DTW calculations), 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, but it is tightly coupled (and dependent upon) the accuracy of the individual gesture recognition (DTW). As a result, we evaluated our recognition system via three metrics:

1. Overall Accuracy: This is an all-or-nothing measure. For input sequence of lengths 1, 2 or 3, we measured 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 measured 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 input containing multiple gestures. For input sequences of length 2 or 3, we measured the number of gestures in the output that actually appear in the input, accounting for order. The purpose of this metric is to determine where the segmentation routine did not “correctly segment” the input sequence. For this metric, we count the first gesture in the input that is correctly identified, count that gesture as +1 and continue summing over the rest of the output. For example, if the input unistroke gesture sequence is *check, star, {*, and the system outputs the sequence as *v, star, {*, the Relaxed Accuracy = 2, counting both the *star* and the *{*. If the same sequence was identified as *check, v, {*, the Relaxed Accuracy = 1 (counting the *check*). Finally, 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 third-party users who conducted multiple recognition phases. Each study participant first chose 5 gestures out of the set of 16 to use for the study. Then the participant entered a recognition phase, where they drew 10 unistroke gesture sequences of lengths 1, 2, and 3 (using the 5 chosen gestures). After the first recognition phase, the participant trained each of the 5

chosen gestures once, and then re-entered the recognition phase. The user alternated training and recognition phases two more times, training each of the 5 gestures a total of 3 times (during the second training phase) and then 5 times (during the third training phase).

We calculated the three 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%

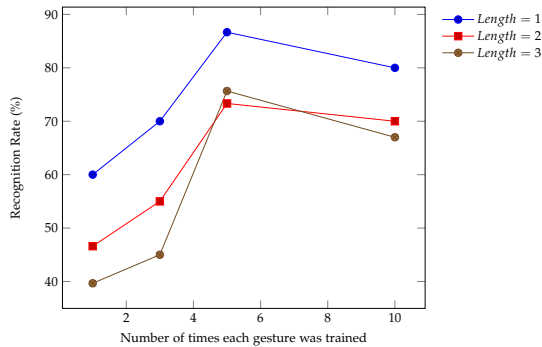


Figure 3: Recognition Rate Vs. Size of Training Data

Lastly, we gained insight into the effect of training additional templates for each of the chosen gestures by measuring the Overall Accuracy (for sequences of length 1) and Relaxed Accuracy (for sequences of lengths 2 and 3) for the different phases of the study. The results are shown in Figure 3.

V. DISCUSSION

Glancing at Tables 1, 2 and 3 together, it is apparent that our segmentation method yields reasonable, but far from perfect, results. The accuracy rate for sequence of length 1 as

shown in Table 1 confirms that DTW is a good algorithm for matching two data sets that vary in time, which is exactly the variance we encounter when recognizing individual gestures. This fact becomes clearer when we compare the DTW approach to the HMM approach taken by Tanguay [4], where he achieved a recognition accuracy of 50 – 60% on lower-case English alphabet letters after extensive training.

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 results in an improper segmentation of the sequence, leading to misrecognition of subsequent gestures. Yang et al [3] report a nearly perfect recognition rate of up to 99.78% for sequences of length 2. The differences in the approaches (motivated by the differences problem statements, as detailed above), are a key consideration when comparing these numbers. Yang et al train Hidden Markov models for both individual gestures as well as for all possible 2-gesture combinations. The comparison clearly shows that reducing our problem “complexity” by training on gesture sequences should yield significantly improved recognition rates.

Table 2 shows the percentage of trials (broken down by length) that were correctly segmented into the appropriate number of gestures. The data in Table 1 shows the percentage of trials that were completely recognized which is a proper subset of the trials counted in 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. Thus, this implies that a more robust technique for individual gesture recognition would significantly boost the Overall Accuracy.

The Relaxed Accuracy results in Table 3 indicate that the system is “recovering” after it incorrectly identifies a gesture in the sequence. Specifically, when comparing the results for gestures of length 2 in Table 3 to the corresponding results in Table 1, the higher Relaxed Accuracy rate indicates that the system is more often able to correctly recognize one gesture in the sequence. This is noteworthy primarily because it implies that training to “dis-ambiguate” gestures (in particular, gestures that the user did not use but that were available in the system) may prove effective. The results for length 3 are similar.

Figure 3 shows that additional training benefits recognition accuracy (Overall Accuracy for sequences of length 1 and Relaxed Accuracy for 2 and 3 gesture sequences) to an extent. Better recognition rates increase the ability for the segmentation routine to properly match and segment each gesture in the input. However “excessive” training may lead to some incorrect recognition as the algorithm may overfit the input with respect to the trained templates. Figure 3 indicates that the current system gives optimal results if each gesture is trained a maximum of 5 times. Fortunately, this constraint helps us with our goal of having a minimally trained system for gesture recognition.

Of the many lessons we learnt, we observed during the study that the gestures themselves were drawn differently during training (when gestures were drawn individually) and during recognition (when gestures were drawn as part of a gesture sequence). This observation is consistent with other research results that indicate that a complex motion is

“more than the sum of its parts”, or more formally, that the transition between gestures in a gesture sequence is critical data that can be used to dis-ambiguate potential gesture segmentation choices.

Regarding future work, our preferred approach to achieving the highest accuracy rates for our current problem definition would be to use HMMs and train on gesture sequences. Additionally, it would be worthwhile to ease the problem constraints and allow the user to intentionally provide “segmentation cues” such as a deliberate pause between gestures in a sequence (which, as an aside, occurred naturally during the study). Another possible avenue for exploration is to allow users to input the gestures in a “free-form” manner and not constraining themselves to unistroke sequences. This mode of input would allow us to leverage the spatial information (such as a space between two words in cursive script) to obtain better segmentation results.

VI. REFERENCES

- [1] S. Mitra and T. Acharya. Gesture recognition: A survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(3):311–324, 2007.
- [2] Lyddane, Donald. United states attorneys’ bulletin. Technical report, United States Department of Justice Executive Office for United States Attorneys, May 2006.
- [3] 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.
- [4] D. O. Tanguay Jr. Hidden markov models for gesture recognition. Master’s thesis, Massachusetts Institute of Technology, 1995.
- [5] 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.