Recognition of Continuous Mouse Gesture Sequences

I. Problem Statement

Gestures, ranging from simple mouse movement to complex full-body motion, are a common form of input in human-centric user interfaces [1]. However, a significant barrier to widespread use of gesture input is the problem of accurate gesture recognition, both in isolation and in sequence. Gesture recognition frequently involves solving two problems: segmentation of a gesture sequence and recognition of individual gestures.

We are focused on creating a quick and flexible prototyping environment, in which the user trains the system on individual mouse gestures. However the user can then create custom unistroke gesture sequences, which will be segmented into individual gestures and identified. The main goal of this project is to get an efficient recognition system up-and-running in the shortest possible time without the need to excessively train it. Our work addresses one part of the larger problem of recognition of custom symbols, which is an important problem in law enforcement. The United States Federal Bureau of Investigation's Safe Streets and Gang Unit commonly encounters handwritten communication involving custom symbols [2].

II. Related Work

Much of the literature related to the problem of gesture recognition involve training Hidden Markov Models for various gestures in the "gesture grammar". Yang et al [3] present work on recognition of individual gestures in continuous gesture sequences, in which they train Hidden Markov Models (HMMs) on continuous gestures. This approach involves defining gesture sequences *apriori*, as well as a prohibitively long training period, both of which we see as deficiencies. Hong et al, in their paper on Chinese character recognition [4], use an iterative segmentation technique that uses whitespace separation to split character sequences into individual characters. This approach does not work if the sequences are drawn in one continuous motion with no space between them, which is the type of input that our problem seeks to address. Robust systems, implemented using HMMs exist, such as the mouse gesture recognition system developed by Tanguay [5]. However this implementation does not lend well to be extended to recognize multiple gestures.

The $1 recognizer [6] is another single gesture recognition system, which uses a simple geometric match algorithm. In our project, we extend this system to recognize multiple gestures in a sequence, by progressively applying the geometric match over the input sequence, which offers fairly robust recognition results with minimal training models.

III. Approach

In our approach, we categorize the gesture sequence given as input by the user as a set of candidate points, which need to be matched against one or more sets of template points, in order to spot a gesture. The main challenge in our approach was to come up with a technique which was neither too strict (as users generally cannot repeat the exact same pattern again and again) not too lenient (we do not want to recognize some random squiggle as any particular gesture). There are two stages in the recognizer, (a) A brief training stage where the user draws a particular gesture and adds its model to the list of "recognizable" gestures (b) A recognition phase where the user can draw any sequence of gestures, from which those that have a corresponding model in the system will be recognized.

Training Phase

Depending on the speed at which a gesture is drawn, there can be a lot of variance in the number of input points to be considered as part of the template. For this reason, irrespective of the number of raw pixels drawn by the user, we quantize it into N equidistant points, where N=64 proved to give adequately accurate as well as computationally inexpensive results. The user is required to give a name for this gesture. For each of the labeled gesture, we find its *indicative angle* [6], which is the angle subtended between the first point and the centroid of the gesture. We define the centroid of a gesture to be the point which has the maximum density of neighboring pixels. If the user draws more instances of the same gesture, the centroid will be computed as an average over all the instances. However, we have found that even with very few instances of each gesture (sometimes even just 1), we get a high accuracy rate for recognizing one or more individual gestures.

Recognition Phase

As stated previously, the foremost challenge in our project was that of segmenting the input sequence in order to obtain the correct number of gestures drawn by the user. In addition, since users are not constrained at the time of defining new gestures, ambiguity can exist based on where a gesture sequence is segmented. Figure 1 depicts this problem of ambiguity. However in our approach we concern ourself only with the first problem; we consider either one of these two possibilities to be correct.

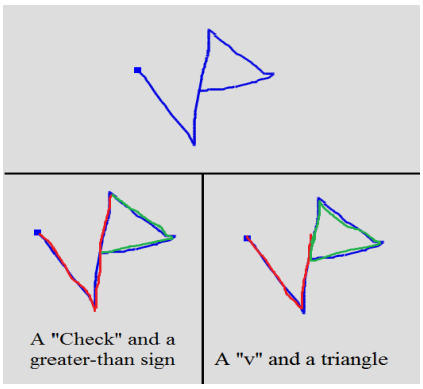

Figure 1: *Ambiguity in gesture recognition after*

We obtain the raw pixels which are part of the gesture sequence drawn by the user. We pick out the first 64 equidistant points (the candidate points) in the input sequence and find its centroid and the *indicative angle*. This subset of points is rotated by the *indicative angle* and is scaled to fit into a square of fixed dimensions. This step is then repeated for all the gesture models recorded into the system by the user. It is necessary to fit both the template points as well as the candidate points into a fixed square in order to make the recognizer invariant to scale. For each predefined gesture, we calculate the euclidean distance between each template point T[k] and the candidate point C[k] and average it over the 64 points. Equation 1 computes this distance:

$$d_k = \frac{\sum_{k=1}^{N} \sqrt{(C[k]_x - T[k]_x)^2 + (C[k]_y - T[k]_y)^2}}{N}$$

This way we have obtained a "distance" measure for the input points with respect to each of the predefined gestures. However, it would be incorrect to simply pick the gesture which gives the lowest distance, as this might give a drastically incorrect result. So we keep reading pixels from the raw input, in groups of 32, and add it to the candidate points. We re-evaluate this distance measure for the new set of points. We keep performing this process of "window sliding" until there is a value of d_k which falls below an empirically determined threshold value. The system then adds the gesture corresponding to this value of d_k to the list of gestures present in the gesture sequence. This gesture is then "spliced" off from the user input and the above process is repeated until there are no more points left in the input. The list of gestures thus obtained corresponds to the individual gestures present in the gesture sequence.

Obviously due to the number of computations involved at every iteration of the window sliding process, its necessary to have a limit on the number of input gestures. We have performed our evaluations by defining 16 gestures, but we believe that this can go upto 50 and the recognizer will still give a reasonable performance.

IV. Evaluation

We have evaluated the system on three parameters:

1. Accuracy of the recognition system: This is an all-or-nothing measure. For input sequence of lengths 1, 2 or 3, we measured the number of times our system got all the gestures in the input correctly.
2. Segmentation efficiency: We have tried to quantify the efficiency of our segmentation routine by finding out the number of instances in which the system has segmented the gesture sequence into the correct *number* of gestures. Even if the system recognizes one or more of the gestures wrongly, if it manages to report the correct number of individual gestures, we consider that to be a positive result for this metric.
3. Relaxed gesture recognition accuracy: This is a relaxed version of the first metric. This parameter will not take into consideration the ordering of the gestures in the input sequence and will also consider partially recognized sequences as favorable. For instance, if the user has drawn a "check" followed by a "star" and the program reports it as a "star" and "arrow", this metric will give it a score of 1/2. That is, the score assigned to a particular solution provided by the system is:

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

Obviously this measure will be different only for input sequences of length greater than 1.

The results of these three metrics were obtained from a micro-study involving a single third-party user and are represented as a percentage in Tables 1,2 and 3 respectively. 20 trials were performed each for input of lengths 1,2 and 3.

Table 1: *Accuracy rate of the system for a given input length*

Sequence Length	Accuracy Rate
1	90%
2	50%
3	20%

Table 2: *Segmentation accuracy of the system for a given input length*

Sequence Length	Segmentation Accuracy
1	100%
2	80%
3	60%

Table 3: *Relaxed Accuracy rate of the system for a given input length*

Sequence Length	Accuracy Rate
2	50%
3	50%

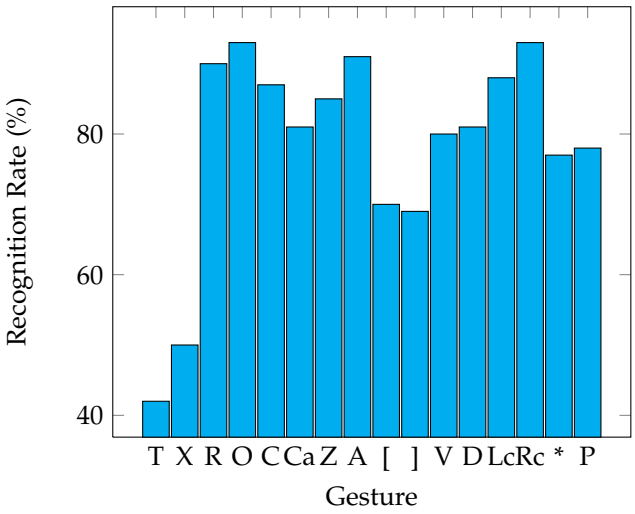

Figure 2: *Recognition rates for individual gestures*

Also we were interested to find out the recognition rate for each of the predefined gestures individually. This is represented in Figure 2.

In Figure 3, the time taken for segmenting and recognizing gestures is plotted for different number of predefined gestures ("Gesture Library").

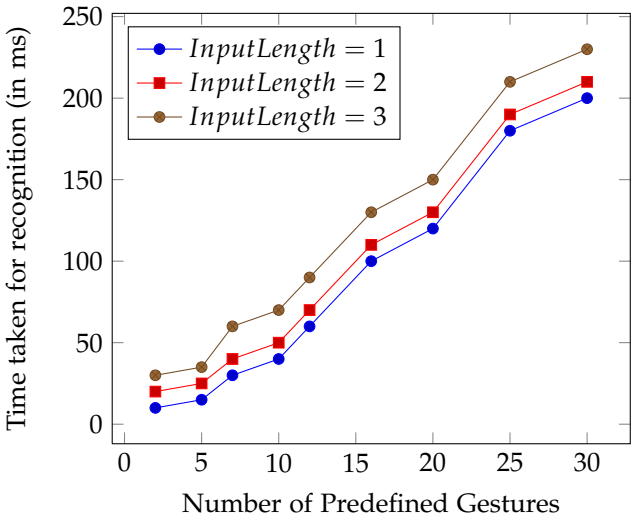

Figure 3: *Time taken for segmenting and recognizing gesture sequences*

Figure 4 is a plot of the variation of the recognition rate with respect to the number of times each predefined gesture was drawn during the training phase. Each of the gesture in the "Gesture Library" were trained equally, and this is represented by the x-axis of the plot.

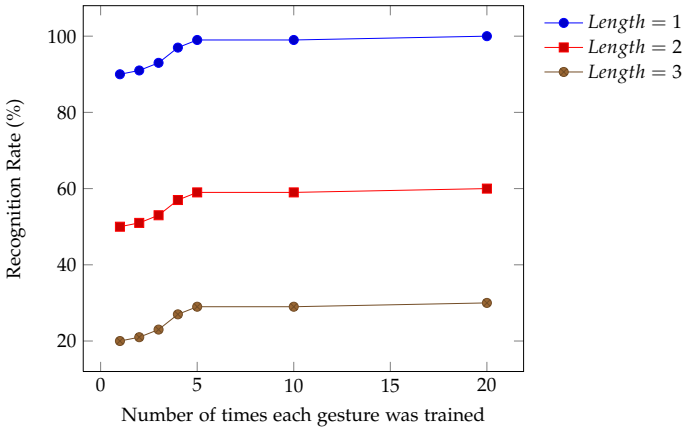

Figure 4: *Recognition Rate Vs. Size of Training Data*

V. Discussion

From Tables 1,2 and 3 it is apparent that the method of progressively applying dynamic time warping over the input sequence of points gives fairly average results. It is clear that the process of geometric matching works much better for recognizing individual gestures than for gesture sequences. However, it is noteworthy to observe that our algorithm was able to segment the input sequence correctly, on an average, about 80% of the time. There was no case where the user drew just one gesture and our system reported it to be more than one. This complete absence of a "false positive" is an encouraging sign, which leads us to believe that our segmentation routine is robust. Although the trials were conducted by only one person, we discovered that our system was fairly invariant to spatio-temporal factors. In such problems, author bias tends to play a big role, and hence we would have got a more complete picture had we been able to test it with multiple users. The results from Table 3 indicate that 75% of the time the system is able to correctly recognize atleast one gesture in sequences of length greater than 1. This is a significant achievement considering how little training is required to get the system up-and-running.

From Figure 2 it is possible to obtain gestures which are "easily recognizable" or in other words "easily distinguishable" from one another (by simply picking out the gestures with the highest recognition rates). The purpose of this exercise was to find out a set of gestures which are very different from one another. If the input were restricted to sequences composed of only these allowable gestures, we posit that we can obtain a much higher recognition accuracy by using the same algorithm.

The need to have a restricted set of gestures as part of the input is also justified by the results compiled in Figure 3. It shows that there is almost a linear relationship between the number of predefined gestures and the time taken to segment and recognize individual gestures in an input sequence. This is understandable since every iteration of the segmentation routine compares the candidate points with all template models available to the system. Therefore by fixing the number of possible gestures, we can obtain a fairly good (and constant) estimate for an upper limit on the time complexity for the recognition process, for a input sequence of a given length.

Figure 4 shows that we do not require a lot of training data to get a good recognition rate. Even if each gesture has just one template instance during the training phase, we were able to achieve 90%, 50% and 20% recognition accuracy for 1-gesture, 2-gesture and 3-gesture input sequences. Also we observe that by increasing the number of training samples, we were able to improve on the accuracy, but this improvement was incremental most of the time. It goes to show that the role of the training phase in our approach is minimal.

We considered using the HMM approach for solving our problem. In order to recognize gesture sequences, we would have had to construct HMMs for each of the individual gestures. Then we can build (not by training, but programatically) models of all possible 2-gesture sequences from our "Gesture Library" by simply chaining the various permutations of the available gestures. For instance, if the number of predefined gestures is 4, and the input sequence is of length 2, then we would have 4 HMMs for the individual gestures and 4^2 HMMs for all possible 2-gesture sequences. We could then simply run the features extracted from the candidate points through each of these HMMs and get the result in a similar manner as Yang et al. However, this solution suffers from combinatorial explosion and will not scale well to handle longer input sequences.

For instance, consider the system to have M predefined gestures and an input sequence which is N gestures long. Then the number of HMMs which need to be constructed in this case, which will give a good approximation of the time complexity of this approach, is

$$M + M^2 + M^3 + ... + M^N = \mathcal{O}(M^N)$$

However our approach, as proven by the results of Figure 3, has a time complexity = $\mathcal{O}$(M*N), which is far less computationally demanding. Nonetheless, we have compared our results with some approaches which have used HMMs in attempting to solve problems similar to ours.

Tanguay [5] in his thesis, has performed identification of lower-case English alphabets and has reported an accuracy of 60-70% on individual characters after extensive training by extracting 5 features and throwing it at an HMM. In our approach, after adding just one training sample for each of the lower-case alphabets, individual gesture recognition was close to 85%. Again this comparison may not be complete due to user bias, but it indicates that dynamic time warping is also a feasible approach in gesture recognition.

Since Tanguay does not recognize multiple gestures in a gesture sequence, we compare our results with that of Yang et al [3]. Although they do not mention results for gesture sequences of length 3, they report a really high accuracy rate of upto 99.78%. Our results from Table 1 are much less impressive, as we were only able to get 50% accuracy. However there are certain key differences in both the approaches. Yang et al train Hidden Markov models for both individual gestures as well as for all possible 2-gesture combinations. Moreover, their gesture set is restricted to the numerals 0-9. Hence in the recognition phase of their approach, they simply pass the raw input to each of the available HMMs and the gesture(s) corresponding to the model that gives the highest probability is selected.

From our work we have learnt that the dynamic time warping approach is a good alternative to HMMs in problems of gesture recognition; more so if computational power is a constraint. However there is potential for further research and work to be done in this field. An immediate work for the future is to be able to carry forward our assumption about a restricted set of input gestures and implement it. It would be worthwhile in investigating the performance measure of the system when the possible input sequence is made up of a finite "alphabet". Also, given sufficient computational horsepower, it would be interesting to construct HMMs for gesture sequences of varying lengths, say upto 5, using the chaining approach and compare the results with those that we have obtained. A fundamental constraint of our implementation is that the input needs to be constructed in one stroke, which does not give much leeway to the user constructing the gesture sequence. Another potential problem for the future is to extend this system, allowing the user to input gesture sequences in a more free-form manner using multiple strokes.

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] C. Hong, G. Loudon, Y. Wu, and R. Zitserman. Segmentation and recognition of continuous handwriting chinese text. *International journal of pattern recognition and artificial intelligence*, 12(02):223–232, 1998.

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

[6] 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.