Proc. of ACM VRST'96 (ACM International Symposium on Virtual Reality and Software Technology), pp. 59-66, July, Hong Kong, 1996

A Sign Language Recognition System Using Hidden Markov Model and Context Sensitive Search

Rung-Huei Liang Ming Ouhyoung
Communication and Multimedia Lab.,
Dep. of Computer Science and Information Engineering,
National Taiwan University, Taipei, R.O.C.
email: ming@csie.ntu.edu.tw

Abstract

Hand gesture is one of the most natural and expressive ways for the hearing impaired. However, because of the complexity of dynamic gestures, most researches are focused either on static gestures, postures, or a small set of dynamic gestures. As real-time recognition of a large set of dynamic gestures is considered, some efficient algorithms and models are needed. To solve this problem in Taiwanese Sign Language, a statistic based context sensitive model is presented and both gestures and postures can be successfully recognized. A gesture is decomposed as a sequence of postures and the postures can be quickly recognized using hidden Markov model. With the probability resulted from hidden Markov model and the probability of each gesture in a lexicon, a gesture can be easily recognized in a linguistic way in real-time.

Keyword: Gesture recognition, sign language, gesture interface, virtual reality.

1. Introduction

1.1 Gesture and Posture

Gestures are usually understood as hand and body movement which can pass information from one to another. Since we are interested in hand gesture and so the term “gesture” is always referred to hand gesture in this paper. To be consistent with previous researches[1], gestures are defined as dynamic hand gestures and postures as static hand gestures. We call postures those not time varying and a gesture a sequence of postures over a short time span.

1.2 Methods of Gesture Recognition

According to the analysis of Watson[2], there are four methods used for gesture recognition: neural network, template-matching, statistical classification, and time-space curves spline matching models. Without time variance, neural network and template matching are suitable for posture matching. However, template-matching is strictly limited to fixed patterns and is very inflexible. VPL™'s posture recognizer[3] and our previous alphabet recognizer[4] are of template-matching. In our previous system, to solve the ambiguity in the alphabet set of the American Sign Language, five tact switches, served as contact-points are sewn on DataGlove™ to provide the abduction information and real-time continuous posture recognition can be achieved. Both Fels' Glove Talk[5] and Beale's system[6] use neural networks. These systems need thousands of labeled examples and output few kinds of recognized symbols, for examples, five symbols, a, e, i, o, and u from the American Sign Language in Beale's system. Watson[2] also discussed pros and cons of neural networks in gesture recognition. In short, retraining is necessary if inserting or deleting one gesture and training often costs much in neural networks. Large amount of processing power is needed, and above all, the structure and results of network are usually not systematically generated, including learning strategy, learning rate, topology, and activation function. Bubine's stroke recognizer[7] is of statistical classification and we classify it to be 2D oriented strokes. Time-space curves spline matching models are still too expensive for real-time applications.

The above systems are all posture recognizers. About gestures, Väänänen[1] proposed the GIVEN concept to recognize postures and gestures using neural networks. GIVEN employs several networks to take time dependency into account. Five time steps of an input stream is suggested. As the sampling rate of input device and the time of gesticulation are considered, hundreds of time steps are very often in one gesture duration. Thus, if one neural network need thousands of labeled examples, the complexity of real gesture is enormous. The critical problem, how to determine the starting and ending point in an input stream of a gesture is not considered by GIVEN and is taken into account by Watson[8]. Discontinuities in a input stream is defined and sequences of discontinuities are processed by template matching. Several pose-based gestures can be recognized and interpreted into artificial reality commands.

1.3 Gesture Recognition with Hidden Markov Model

Inspired by our previous work[4] on American Sign Language (alphabet set only), since full set sign language understanding is our goal, new approaches are proposed. We define that the time-varying parameter (TVP) in a stream of data is the parameter which changes its value along time axis. A discontinuity occurs when the number of TVPs of a stream of gesture input is under a threshold. A posture recognition takes place if and only if a discontinuity is detected. Then hidden Markov models (HMMs) are employed for posture recognition. Every known posture has a distribution in angles for each joint of finger; since eleven inputs are received from DataGlove™ plus a 3D tracker, eleven distributions are constructed for each posture. For example, in Taiwanese Sign Language (TSL), there are 50 fundamental postures; thus 50×11 distributions are established during learning phase. A frame of input data is send to 50 HMMs and the posture with the maximal evaluation resulted from its corresponding HMM is the winner. HMM is briefly explained in section 4, and posture recognition is described in section 5.

Gesture recognition is based on looking up the vocabularies composed of several possible paths of sequences of postures. The final solution is resulted from dynamic programming. Since the evaluations of these competing vocabularies are simple lookups from a lexicon, the processing time of dynamic programming is greatly reduced. The whole recognition procedure is shown in section 3, and details are given in section 6 and 7.

2. Taiwanese Sign Language

We choose a specific sign language for our study, and our goal is to study the full set. The Taiwanese Sign Language (TSL) includes 50 fundamental postures and is shown in Figure 1. Even though this sign language is not fully compatible to international standard, the experiences gained from studying TSL will also contribute to the study of other sign languages, such as ASL (American Sign Language). A gesture in TSL consists of one or more postures sequentially moved or posed to some position or direction. For example, in figure 2, "Father" in TSL is gesticulated as making the posture numbered one in Figure 1, which means one, to the cheek and then making posture numbered twenty-six in Figure 1, which means "male".

Each gesture can be thought as a vocabulary in a lexicon, and a sentence is a sequence of gestures. Moreover, There are two styles of sequences of gesture in TSL: natural sign language and grammatical sign language. Natural style is mostly used by the disabled, especially the older generations. Its order of sequence of vocabularies is different from that of Chinese. The grammatical style is taught in the elementary school and is consistent with the order of word sequence of Chinese. Because of the consistency and regularity, we tentatively only take grammatical style into account in our research.

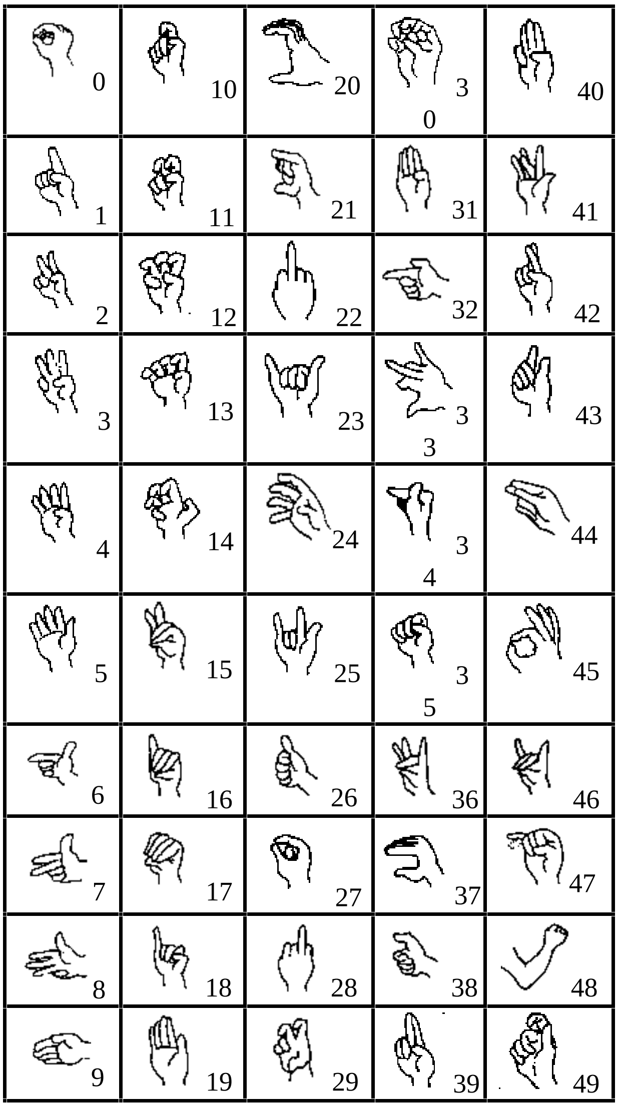


Figure 1 50 fundamental postures in TSL.

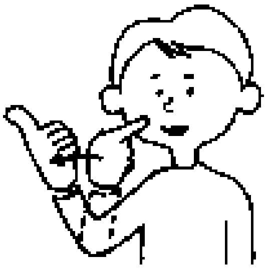


Figure 2 "Father" in TSL.

3. System Overview

The proposed system architecture is shown in Figure 3 and is similar to that in speech recognition [9][10][11] because of the similarity between gesture and speech recognition.

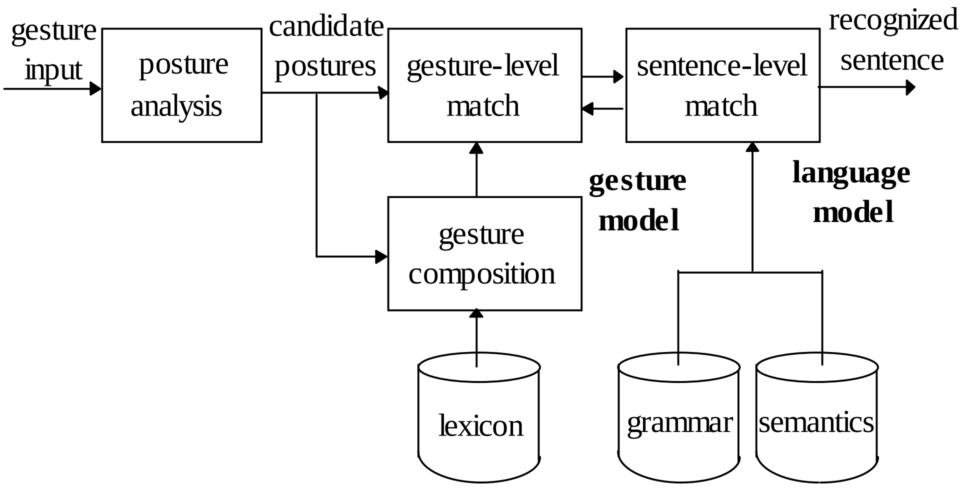


Figure 3 Block diagram of posture-based continuous gesture recognition.

This posture-based continuous gesture recognition system consists of three major function units, i.e., posture analysis, gesture-level match, and sentence-level match. Posture analysis is illustrated in Figure 4 and is described later. After posture analysis, the results are decoded into several candidate postures. The gesture composition composes several possible gestures according to lexicon. Every one of these possible gestures may contain one or more postures, that is, each candidate posture may be combined with those candidate postures in the previous one or more frames. Gesture-level match evaluates these gestures according to the probabilities of associated postures, and their corresponding probabilities in this language. The above process is defined here as the gesture model and can produce some recognized gestures with their corresponding probabilities. In Figure 3, the two arrows between gesture-level and sentence-level matches indicate the necessary backward and forward processes of dynamic programming. Sentence-level match takes responsibility of higher level match, from the view of language model. The relationship of several adjacent gestures is explained from storage grammar. An *N*-gram strategy estimates the probability of specific *N* adjacent gestures, while bi-gram only considers two adjacent gestures and is adopted in this paper. The probability looked up in storage grammar is combined with the probability in gesture-level match, and the sentence-level match will generate a sentence with the highest probability and output it according to semantics.

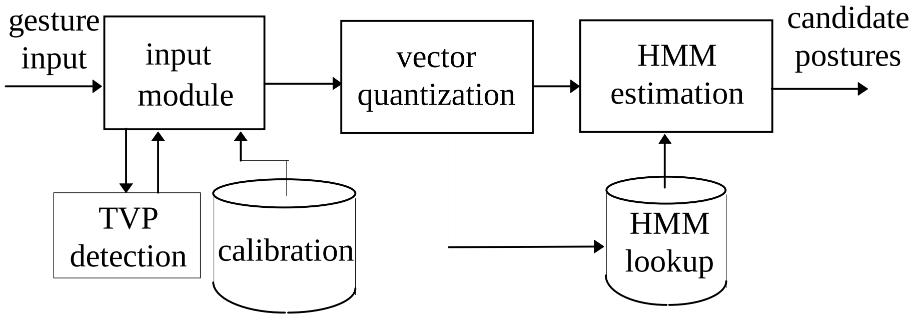


Figure 4 Block diagram of posture analysis. 3 major modules are included: input module, vector quantization, and hidden Markov model (HMM) estimation.

Posture analysis comprises three major modules: input module, vector quantization, and HMM estimation. The input module receives a stream of gesture input, and if a discontinuity happens, the input module forwards a frame of raw data, adjusted by calibration file, to vector quantization. The discontinuity detection is done by time-varying parameter (TVP) detection, and whenever the number of TVPs of the hand motion begins to reduce to below a threshold, the motion is thought to be quasi-stationary, and its corresponding frame of data is taken to be recognized. It is reasonable to stay for a while to make a posture, and no matter how short the duration of stationary state is, the TVP detection always works because the motion of a sophisticated sign language user is still far below the sampling rate of the input device, which is usually above 10 Hz in our system. Because of the characteristics of a gesture, this method solves the end point detection in a stream of data, also a nontrivial problem in speech recognition.

To estimate the probability of a frame of data, vector quantization is needed because the density function we built in HMM estimation is discrete. Ideally, the data distribution in angles of a specific hand joint for a particular posture can be presented as Figure 5(a), and $0^{\Leftrightarrow}$-$90^{\Leftrightarrow}$ is the range of the angle a joint bends. However, this continuous function is not trivial to obtain. A discrete density function, Figure 5(b), is both convenient to implement and efficient to estimate. Each hidden Markov model of 50 postures reports the probability for the specific input flame respectively, by simply multiplying ten probabilities inside each HMM together. Finally the system selects several candidate postures according to the probabilities estimated above.

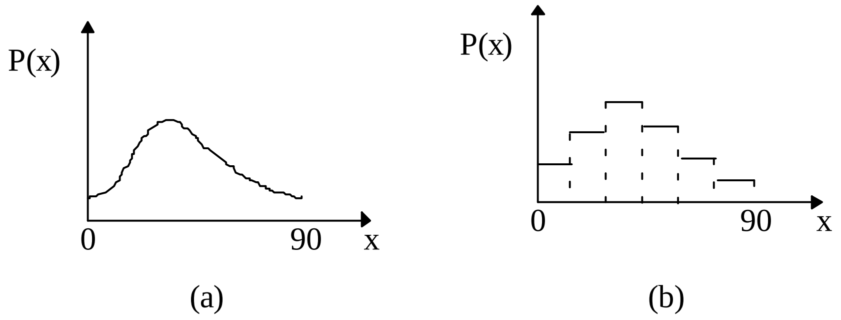


Figure 5 (a) Continuous density function and (b) discrete density function.

4. Hidden Markov Model

Hidden Markov model (HMM) is a well-known and widely used statistical method. The basic theory of hidden Markov models was implemented for speech-processing applications in 1970s[9]. Similar to speech, the underlying assumption of a statistical model, the hidden Markov model, is that gesture signal can be well characterized as a parametric random process, and that the parameters of the stochastic process can be determined in a precise, well-defined manner.

Consider a system that may be in one of N distinct states at any time indexed by $\{1, 2,\ldots, N\}$. We denote the time instants as $t = 1, 2,\ldots, T$, and the actual state at time t as q_t. Assume that the current state is just correlated to the preceding state, that is, the first order Markov chain,

$$P[q_t = j|q_{t-1} = i, q_{t-2} = k,...] = P[q_t = j|q_{t-1} = i]. \tag{4.1}$$

Then, the state transition probabilities a_{ij} is defined as

$$a_{ij} = P[q_t = j|q_{t-1} = i], \qquad 1\leq i, j \leq N \tag{4.2}$$

with the following properties

$$a_{ij} \geq 0 \qquad \forall j, i \tag{4.3.a}$$

$$\sum_{j=1}^{N} a_{ij} = 1 \qquad \forall i \tag{4.3.b}$$

Furthermore, the joint probability of observation sequence **O** and a state sequence **q** can be written as

$$P(\mathbf{O}, \mathbf{q}|\lambda) = P(\mathbf{O}|\mathbf{q}, \lambda)P(\mathbf{q}|\lambda), \tag{4.4}$$

where λ is the given hidden Markov model.

The probability of observation **O** (given the model λ) is obtained by summing this joint probability over all possible state sequences **q**, resulting

$$P(\mathbf{O}|\lambda) = \sum_{all\ \boldsymbol{q}} P(\mathbf{O}|\mathbf{q}, \lambda)P(\mathbf{q}|\lambda) \tag{4.5}$$

$$= \sum_{q_1, q_2, \ldots, q_T} \pi_{q1} b_{q1}(\mathbf{o}_1) a_{q1q2} b_{q2}(\mathbf{o}_2) \ldots a_{qT-1qT} b_{qT}(\mathbf{o}_T).$$

(4.6)

Equation (4.6) can be interpreted as the following. q_1 is the initial state with probability π_{q1} , and $\mathbf{o}_1$ is the generated observation with probability $b_{q1}(\mathbf{o}_1)$. Time changes from time step 1 to 2 and a transition is made from state q_1 to q_2 with probability a_{q1q2} , and $\mathbf{o}_2$ is generated. This process continues in the same manner until time T.

To find the best state sequence, $\boldsymbol{q} = (q_1 q_2 \ldots q_T)$, for the given observation sequence $\boldsymbol{O} = (o_1 o_2 \ldots o_T)$, consider the following quantity

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P[q_1 q_2 \ldots q_{t-1}, q_{t=i}, \boldsymbol{o}_1 \boldsymbol{o}_2 \ldots \boldsymbol{o}_t|\lambda]. \tag{4.7}$$

$\delta_t(i)$ is the best score (highest probability) along a single path, at time t, which accounts for the first t observations and ends in state i. By induction,

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] \exists b_j(\boldsymbol{o}_{t+1}), \tag{4.8}$$

and the goal is to find a single path which can result

$$P^* = \max_{1 [i [N} [\delta_T(i)]. \tag{4.9}$$

This can be solved by the Viterbi algorithm which is used in speech recognition[9].

5. Posture Recognition

The approach of posture recognition is to assume a simple probabilistic model of posture input data. The model of each specific posture P produces an observation Y with probability $P(P,Y)$. The goal is to decode the posture data with the observation so that the decoded posture has the maximum a posteriori (MAP) probability, that is,

$$\hat{P} \ni P(\hat{P}|Y) = \max_P P(P|Y) \tag{5.1}$$

Equation (5.1) can be written as

$$P(P|Y) = P(Y|P)P(P)/P(Y) \tag{5.2}$$

Since $P(Y)$ is independent of P, Equation (5.1) is equivalent to

$$\hat{P} = \arg\max_P P(Y|P)P(P). \tag{5.3}$$

The first term in Equation (5.3), $P(Y|P)$ estimates the probability of a observation conditioned on a specific posture, and is called the acoustic model in speech recognition [9]. The second term in (5.3), *P(P),* is called the language model since it describes the probability associated with *P* in the specific set of sign language.

Figure 6 shows a hidden Markov model for a posture. Each state represents an evaluation of an observation data reported from DataGlove™, and a total of 10 data of joints are send to estimate *P(Y|P)*. Since these 10 data can be thought mutually independent, the type of HMM we use here is simply a left-right model with all state transition probabilities being 1, i.e., $a_{1,2} = a_{2,3} = \ldots = a_{9,10} = 1$. As a example in TSL, there are 50 fundamental postures, and therefore, 50 hidden Markov models are built in training phase to evaluate their corresponding *P(Y|P)*.

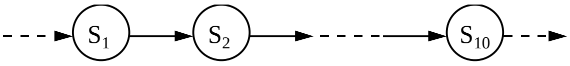


Figure 6 HMM representation of a posture.

6. Gesture Recognition with Lexicon

Since a gesture is a sequence of postures with maximal length 3 in TSL, HMM of a gesture can be represented as Figure 7. Each state among S_2, S_3, and S_4 represents an evaluation of a posture.

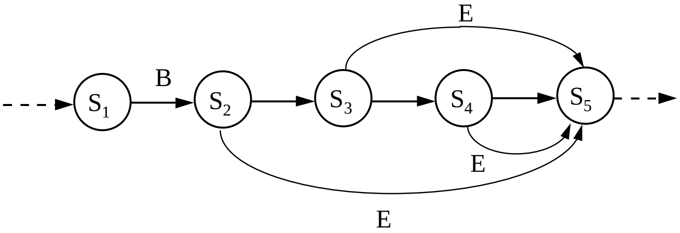


Figure 7 HMM representation of a gesture. B means beginning and E, ending.

Given an observation sequence $\mathbf{O} = (\mathbf{o}_1\mathbf{o}_2...\mathbf{o}_T)$, our goal is to find a single best state sequence, $\mathbf{q} = (q_1q_2...q_T)$. Consider ending frame *e*, as illustrated in Figure 8. Each frame may be recognized as several posture candidates in posture recognition phase and are composed into some gestures according to the gesture lexicon. Then a typical dynamic programming is employed to evaluated 3 conditions in Figure 8.

To solve this dynamic programming problem, we first let *Solution(e)* represent the best solution ending at frame *e*. We assume that g_1 is the gesture in Figure 8(a), g_2 in Figure 8(b), and g_3 in Figure 8(c); g_{l1} is the last gesture in *Solution(e-1)*, g_{l2} is the last gesture in *Solution(e-2)*, and g_{l3} is the last gesture in *Solution(e-3)*. The probability of the best solution ending at frame *e* can be written as

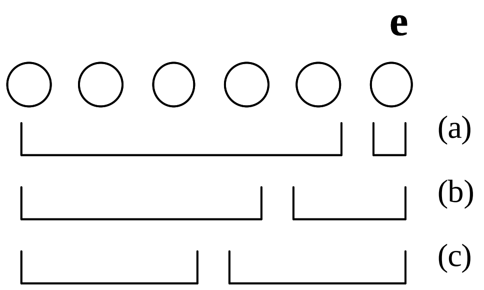


Figure 8 Possible gestures ending at frame e, with gesture of length 1 ending at *e* (a), length 2 (b), and length 3 (c).

$$\begin{aligned} P(Solution(e)) = \ & max(P(Solution(e\text{-}1))a_{gl1g1}P(g_1), \\ & P(Solution(e\text{-}2))a_{gl2g2}P(g_2), \\ & P(Solution(e\text{-}3))a_{gl3g3}P(g_3)) \end{aligned} \qquad (6.1)$$

where a_{ij} is the probability that gesture *i* and gesture *j* are adjacent and is called grammar model. $P(g_1)$ is the probability of gesture g_1 in the specific sign language system and is called the language model. To emphasize the importance of the influence of a_{ij}, *Solution(k)* ending at frame *k* keeps 3 candidates with best scores. Besides, for each candidate solution, the data structure *Solution(k)* records one gesture, as the last gesture ending at frame *k*, and an index to the previous ending frame number. Therefore, the single path of a solution can be easily obtained by a backward procedure. Thus each argument in *max* in Equation (6.1) generates 3 possible solutions to compete, and also g_1, g_2, and g_3 may be composed of different candidate postures.

To show the feasibility of real-time recognition, a brief complexity analysis is given below. As the worst case is considered, g_1 may be 3 different gestures composed from 3 different postures and all of these gestures are valid in the lexicon, and in the same way, g_2 may be 9 different gestures and g_3 27 different gestures. Combined with the structure *Solution*, the total number of competing solutions is (3%3 + 3%9 + 3%27) = 117 in the worst case. That is, whenever a discontinuity occurs, at most 117%2 multiplications (one between $P(Solution(k))a_{ij}$ and another between $a_{ij}P(g)$ in (6.1)) and 116 comparisons (to find 3 maximal values) are needed. Roughly estimating, whenever a posture recognition happens, 117%3 float-point computations are needed for dynamic programming of gesture recognition.

Taking a 1.0 Mega FLOP CPU as an example, assume a very experienced sign language user can make 5 postures per second, which is very difficult, the average duration of a posture is 0.2 seconds. Thus, in this time duration, the CPU can perform 1*M* % 0.2 = 200*K* floating-point computations, which is about 300 to 400 times of 117%3 multiplications, as the required computation power. As analyzed above, the solution time of the dynamic programming used here is very short comparing to the duration of making a posture, and the real-time requirement is therefore met.

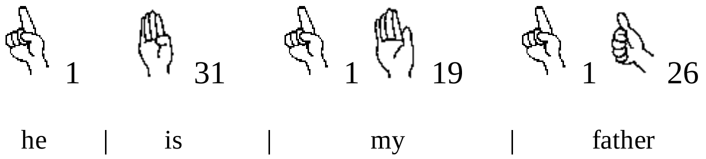


Figure 9 An input stream of postures and its semantics in TSL.

In figure 9, a possible sequence of postures *(posture index: 1, 31, 1, 19, 1, 26)* ending at frame 6 is input to our system, and the goal is to find suitable sequence of gestures and to generate its corresponding semantics. When posture 26 arrives, gesture *(1, 26)*, which means "father", is one of the possible vocabularies in figure 8(b), and the solution *Solution(6) = Solution(4)|(1, 26)* may be the winner among 117 possible solutions since most of the gestures composed by candidate postures are invalid in the lexicon. Therefore *Solution(6)* records gesture *(1, 26)* as its last gesture in that sentence of length 6, and an index to frame 4. Because *Solution(4)* has been processed in the same manner, *(1, 19)* may be the last gesture of the sentence of length 4, and its corresponding index 2 refers to the previous optimal solution frame number. Also by the adjustment of a_{ij} in Equation (6.1), the dynamic programming algorithm generates an optimal path based on bi-gram strategy described in the next section.

7. Grammar Model

The goal of the grammar model is to provide an estimate of the probability of a gesture sequence *G*. Assume *G* is a sequence of gestures,

$$G = g_1 g_2 ... g_K , \qquad (7.1)$$

then the probability of G can be written as

$$P(G) = P(g_1 g_2 ... g_K) = P(g_1)P(g_2|g_1)P(g_3|g_1 g_2)... \\ P(g_N|g_1 g_2 ... g_{K-1}). \qquad (7.2)$$

However, it is not likely to estimate $P(g_j|g_1 g_2 ... g_{j-1})$ for all gestures and all sequence lengths in a language. Therefore, an *N*-gram model is convenient to approximate $P(g_j|g_1 g_2 ... g_{j-1})$ as

$$P(g_j|g_1 g_2 ... g_{j-1}) \approx P(g_j|g_{j-N+1} ... g_{j-1}) \qquad (7.3)$$

Even *N*-gram probabilities are difficult to estimate, thus if $N = 2$, Equation (7.2) can be approximated as

$$P(G) = P(g_1)P(g_2|g_1)\ P(g_3|g_2)...\ P(g_k|g_{k-1}), \qquad (7.4)$$

and this is called bi-gram and is much easier to implement and more efficient to estimate.

8. Results

To demonstrate our system capability, lesson one of the Taiwanese Sign

Language (TSL) textbook for elementary school is implemented as our first step. It contains 71 most frequently used vocabularies in TSL and 30 sample sentences. By trial and error, we divide $0^{\Leftrightarrow}$-$90^{\Leftrightarrow}$ into 5 quantization steps with equal quantization size in posture recognition. The discrete density function is simply obtained by counting the frequency in each quantization step over the total number of training samples of the corresponding posture.

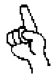
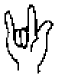
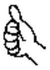
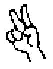
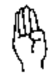
 1	 25	 26	 2	 31
1: 1.0000000	*25: 1.00000000*	*26: 1.00000000*	*2: 1.00000000*	*31: 1.00000000*
32: 0.0076246	*23: 0.05861891*	*35: 0.01308350*	*39: 0.02032957*	*4: 0.78576466*
10: 0.0024320	*6: 0.05381770*	*6: 0.00918933*	*42: 0.00646599*	*5: 0.08633067*

Figure 10 Five examples of recognized postures. For each posture, three most possible posture candidates and the probabilities related to the highest score are listed. In the fifth column, the probabilities of posture 31 and posture 4 are nearly the same (see Figure 1 for posture 4), and this ambiguity can be solved by the context sensitive search.

HMMs used in posture recognition can report three candidates for each posture group (Figure 10). Whenever a discontinuity is detected, by means of time-varying parameter (TVP) detection, 9 multiplications are needed for each HMM of posture, and 50 comparisons are performed, and thus 9 % 50 + 50 = 500 computations are needed; according to the analysis in Section 6, i.e., using a 1*M* FLOP computer, the above computations take 500/ 1*M* = 0.5 *m* seconds. This is also verified to meet the real-time requirement by our TSL posture recognition module.

Gesture-level match and sentence-level match always keep three solutions with highest scores ending at each frame. As estimated in Section 6, the dynamic programming needs 117 % 3 = 351 floating-point computations in the worst case, and usually far smaller than this quantity because the number of valid vocabularies among these 117 possible conditions is usually under 50. One thing to notice is that the time needed for dynamic programming is shorter than that of posture recognition, and the total computation time of gesture recognition is 1 *m* seconds at most. We have verified the above estimation in our prototype TSL recognition system.

The experience we learned here is that we can spend a little more computation time on dynamic programming instead of investing in improving the posture recognition rate; because the ambiguity resulted from a faster posture recognizer can be easily solved by dynamic programming in our system, being implemented on a Pentium100 PC.

The table lookup time which is not yet discussed may take the biggest part of the whole recognition time; since the lexicon, the grammar, and the semantics files are small enough to load into memory, this issue is not considered in this paper tentatively. When the number of vocabularies grows up, this has to be addressed.

Furthermore, to recognize gestures more precisely, the postures of both hands should be considered, that is, there should be two sets of input devices to capture the postures of both left hand and right hand; while in this paper, we only take the postures of right hand as illustration. Although processing the motion of two hands may double the posture recognition time we analyzed above, the solution time for dynamic programming is the same, and thus the total computation time takes 1.5 *m* seconds at most. The information reported from attached 3D tracker can also help to solve the ambiguity of gestures, since the former are of the same sequence of postures but different in motion trajectory.

9. Conclusions

The model proposed in this paper is suitable for the problem of sign language recognition. Based on the assumption that a discontinuity occurs whenever a posture is performed, the time-varying parameters (TVP) detection can solve the ending point problem of postures, and is proved to be feasible in our implementation. Moreover, posture recognition with hidden Markov models (HMM) is also superior in several aspects. First, it is always statistically correct; unlike neural network, innocent of the procedure and results as described in Section 1, each HMM estimates based on the statistical facts. Second, it is easy to insert a new class of posture or to delete an existing class of posture and also new training for a specific posture is simply to accumulate new samples into its density function. This advantage is hard to achieve by other methods. Finally, because of the efficiency in time and space, it is very easy to deal with large gesture “vocabularies”, while VPL™‘s posture recognizer tends to overlap when the classes of postures are over 15[2].

Our prototype TSL recognition system can recognize dynamic gestures in real-time; and the experience of developing posture and gesture models can be applied to other sign languages, and may help to interpret a large set of virtual reality commands.

Reference

[1] Kaisa Väänänen, Klaus Böhm, “Gesture Driven Interaction as a Human Factor in Virtual Environments - An Approach with Neural Networks,” pp. 93-106, *Virtual Reality System*, Academic Press, 1993.

[2] Richard Watson, “A Survey of Gesture Recognition Techniques,” Technical Report TCD-CS-93-11, Trinity College, Dublin 2, 1993.

[3] Tomas G. Zimmerman, Jaron Lanier, "A Hand Gesture Interface Device," pp. 189-192, *ACM SIGCHI/GI*, 1987.

[4] Rung-Huei Liang, Ming Ouhyoung, "A Real-time Continuous Alphabetic Sign Language to Speech Conversion VR System," *Computer Graphics Forum*, pp. C67-C77, Vol. 14, No. 3, 1995. (also in EUROGRAPHICS‘95, Holland). *http://www.cmlab.csie.ntu.edu.tw/~f1506028*.

[5] S. Sidney Fels, Geoffrey E. Hinton, "Building Adaptive Interfaces with Neural Networks: The Glove-talk Pilot Study," pp. 683-688, *Human-Computer Interaction-INTERACT 90*, Elservier Science Publisher B.V., North-Holland, 1990.

[6] R. Beale, A. Edwards, "Recognising Postures and Gestures Using Neural Networks," *Neural Networks and Pattern Recognition in Human Computer Interaction*, E. Holland, 1992.

[7] Dean Rubine, "The Automatic Recognition of Gestures," Ph. D thesis, Carnegie Mellon University, December 1991.

[8] Richard Watson, Paul O'Neill, "Gesture Recognition for Manipulation in Artificial Realities," *Proc. of the 6th International Conference on Human-Computer Interaction*, Pacifico Yokahama, Yokahama, Japan, July 1995.

[9] Lawrence Rabiner, Biing-Hwang Juang, *Fundamentals of Speech Recognition*, pp. 321-389, pp. 434-481, Prentice-Hall International, Inc., 1993.

[10]H. M. Wang, L. S. Lee, “Mandarin Syllable Recognition in Continuous Speech Under Limited Training Data with Sub-syllabic Acoustic Modeling,” *Computer Processing of Chinese and Oriented Language*, Vol. 8, pp. 1-6, December 1994.

[11]L. S. Lee, et al, “Golden Mandarin (III)- A User Adaptive Prosodic-segment-based Mandarin Dictation Machine for Chinese Language with Very Large Vocabulary,” ICASSP, 1995.