Fingerspelling Recognition through Classification of Letter-to-Letter Transitions

Susanna Ricco and Carlo Tomasi

Department of Computer Science
Duke University
Durham, NC 27708
{sricco, tomasi}@cs.duke.edu

Abstract. We propose a new principle for recognizing fingerspelling sequences from American Sign Language (ASL). Instead of training a system to recognize the static posture for each letter from an isolated frame, we recognize the dynamic gestures corresponding to transitions between letters. This eliminates the need for an explicit temporal segmentation step, which we show is error-prone at speeds used by native signers. We present results from our system recognizing 82 different words signed by a single signer, using more than an hour of training and test video. We demonstrate that recognizing letter-to-letter transitions without temporal segmentation is feasible and results in improved performance.

1 Introduction

The native language of the Deaf Community in the United States is *American Sign Language* (ASL), which defines a vocabulary of gestures corresponding to frequently used words. When no standard sign exists for a desired word, signers use *fingerspelling*, spelling out the word using gestures that correspond to the letters in the English alphabet. Unlike word-level signs, fingerspelling gestures use a single hand, and most do not require motion. Instead, different letters are primarily distinguished by the positions of the signer's fingers, called the *handshape*.

The naïve approach to fingerspelling recognition is to learn to recognize each letter's handshape in isolation before tackling letters in sequence. We believe a more reliable system recognizes *transitions between letters* rather than the letters themselves. This approach avoids the need to select which frames to classify into letters, a process that is error-prone at conversational speed. In addition, emphasis on transitions leverages information about the shape of a signer's hand *as a letter is being formed* to differentiate between letters that are easily confused in static frames. The naïve solution discards this helpful information.

In this work, we present a system that recognizes transitions between fingerspelled letters. In Sect. 2, we review previous work on fingerspelling recognition. These existing recognition systems rely on an initial time segmentation process to identify a single isolated frame for each letter to be recognized. In Sect. 3, we demonstrate situations where proposed time segmentation techniques fail, necessitating the shift to letter-to-letter transitions. In Sect. 4, we describe a system that uses traditional techniques from word-level ASL and speech recognition to model the transitions. Section 5 illustrates the technique on an example vocabulary. The results show that modeling transitions between letters improves recognition performance when prior temporal segmentation is not assumed.

2 Related Work

The automatic translation of ASL into written English has been an active area of research in computer vision for over a decade. Traditionally, researchers have considered recognition of word-level gestures and fingerspelled letters to be isolated problems and have developed separate techniques to address the two. The two independent systems could eventually be combined to translate sequences containing both word-level and fingerspelled signs by segmenting the video into word-level or fingerspelled only segments using a binary classifier [1] and running the appropriate system on the extracted segments.

Most systems designed to recognize word-level gestures use Hidden Markov Models (HMMs) to model each hand's location and velocity over time. Techniques differ mainly in the degree to which handshape information is considered. Some methods [2, 3] use only very basic handshape information, if any; others [4] use a complete description of the bending angles at 18 joints in the hand, which are measured using an instrumented glove such as a CyberGlove.

In contrast, existing fingerspelling recognition systems classify static handshapes in isolation. The complexity in the handshapes that must be differentiated led some researchers [5, 6] to use joint bending angles from a CyberGlove as the input features. Unfortunately, these gloves are both intrusive and expensive. Hernandez-Rebollar et al. [7] built their own instrumented glove in an attempt to provide a low-cost option. Other researchers [8–11] focused on improving vision-based methods to create systems that are relatively inexpensive and require only passive sensing. These systems have performed well in restricted environments. Birk et al. [12] report recognition rates as high as 99.7% for a single signer when presented with isolated images of each letter.

A related and active area of research is the recovery of arbitrary 3D hand poses from a single image [13]. In theory, one could construct a fingerspelling recognition system by taking a single image of the signer's hand, inferring the corresponding 3D hand pose, and then matching this pose to the static poses defined for each letter. Like traditional systems, however, a technique relying on pose reconstruction still uses an isolated image of a letter as the input to be recognized.

To find the necessary single frame, researchers apply a threshold to the total motion in the image. Recognition is performed on low-motion frames. Different techniques are used to measure the motion of the signer, ranging from the total energy in the difference between two consecutive frames [10] to the velocity of the hand directly measured using the instrumented gloves [6]. Motion-thresholding techniques work well as long as signers pause as they sign each letter. However, they begin to fail when this assumption breaks down and individual letters become hidden in the smooth flow of high-speed fingerspelling gestures.

To our knowledge, Goh and Holden's fingerspelling recognition system [14] is the only current technique that does not require an explicit segmentation into individual letters prior to recognition. This system is trained to recognize fingerspelling using the Australian Sign Language (Auslan) alphabet, with individual HMMs for each Auslan letter chained together using an explicit grammar to form word-level HMMs. A new sequence is classified as the word whose HMM maximizes the probability of the observations, consisting of coarse descriptions of handshape and the velocities of points along the boundary of the silhouette. They report a best word-level accuracy of 88.61% on a test set of 4 examples of 20 different words.

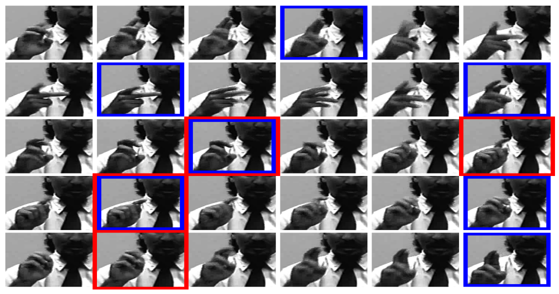
3 The Case for Transitions

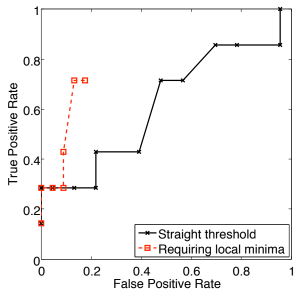
The assumption that signers pause at each letter is consistently violated at conversational speed. Proficient signers commonly fingerspell at 40-45 words per minute (WPM), and it is impossible to pause at every letter at this speed. At 45 WPM, many letters are not formed exactly, but are combined with neighboring letters in fluid motions. Even if a signer does pass through the exact handshape defined for a letter, the aliasing resulting from a comparatively low frame rate can cause this handshape to be missed.

Our experiments show that thresholding methods fail to accurately identify letters at conversational speed. We took clips from interpreter training videos [15] of native signers and identified frames to classify using a method similar to the one described by Lamar et al. [10], which measures motion in each frame by image differencing. In the first version, we select all frames with motion below a set threshold; in the second, we select only frames corresponding to local minima of motion that fall below the threshold. Figure 1(a) shows 30 frames from a man signing *rpreter* (part of *interpreter*), with frames corresponding to local minima below a set threshold surrounded by red boxes. The seven frames that best represent the seven signed letters as determined by a human expert are outlined in blue.

The threshold misses the first three (r, p, and r in frames 4, 8, and 12) and last (r in frame 30) letters completely. Frame 18 is incorrectly identified as a letter frame; it is actually the midpoint of the transitional motion from the letter e to t, where the signer changes the direction of motion of the index finger. Also note that the handshapes selected by the expert for r and e in frames 12 and 15 do not exactly match the handshapes defined in the ASL manual alphabet for these letters.[1] The signer never forms the exact handshapes during his smooth motion from the p in frame 8 to the t in frame 20. This would cause errors in recognition for a system trained on the defined static poses for each letter, even if these frames were selected for classification.

[1] An introduction to the ASL manual alphabet can be found at http://www.lifeprint.com/asl101/fingerspelling/abc.htm.


(a)



(b)

Fig. 1. The last 30 frames (left-to-right, top-to-bottom, in (a)) from the word *interpreter* with the closest frame to each of the seven letters outlined in blue. The four frames outlined in red (numbers 15, 18, 20, and 26) are those selected using a fixed threshold on the motion in the frame. ROC curves (b) show the effect of varying the threshold. Selecting all frames below the threshold (solid black) identifies too many incorrect frames; selecting only local minima below the threshold (dashed red) is incapable of finding letters where the signer does not pause. (Figure best viewed in color.)

The receiver operating characteristic (ROC) curves in Fig. 1(b) show the effect of varying the threshold. The dashed line corresponds to an algorithm that selects only local minima. Because some letters actually occur at local maxima, not minima, this algorithm can never identify all the letters, no matter what the threshold. The solid line corresponds to an algorithm that selects every frame with motion below the threshold. This algorithm eventually finds all the letter frames but includes almost all transition frames as well. Clips of different words from a number of different signers show similar poor performance. In fact, we observed that the more common a specific combination of letters was, the less likely it was for those letters to occur at local minima.

Human experts recognize the difficulty in trying to extract individual letters from continuous fingerspelling, often teaching students to look for the "shape of the word" instead of picking out each letter. Research has shown young deaf children also use this method, initially perceiving fingerspelled words as single units rather than as sequences of individual letters [16]. We adopt a similar approach, recognizing motions between letters and eliminating the need for an initial time segmentation step. As an added benefit, looking at the motion between letters can help differentiate between letters whose static handshapes appear similar. Figure 2 shows consecutive frames from the fingerspelled words *at* and *an*, which have similar final handshapes but contain distinguishing transitional motions.

4 Recognizing Transitions

In this section, we describe a system that recognizes the gestures corresponding to motions between consecutive letters. We model the motion using an HMM

Fig. 2. Hand silhouettes from consecutive frames of the fingerspelled words *at* (left) or *an* (right). The final handshapes (letters *t* and *n*) appear similar, but handshapes are clearly different during the transition. In *at*, only the index finger must move to all for correct placement of the thumb. In *an*, the index and middle fingers must both move.

with an observation model defined over part-based features extracted from single-camera video of an unadorned signer. Because we recognize changes in handshape over time using an HMM, our approach is related to the handshape channel model used by Vogler and Mexatas [17] to recognize word-level signs involving changes in handshape. Our method differs in that it is glove-free. The use of similar recognition techniques is intentional because it allows the two systems to be combined into one that would recognize both aspects of ASL.

4.1 Handshape Representation

We use a part-based method to represent handshape. Part-based methods are sparse representations that match regions of the image to codewords in a specified dictionary. Typically, codewords are learned from training data or provided by a human expert. We learn codewords that capture important information about the position of each finger but that can be easily computed from images recorded by a single camera.

Extracting Hand Silhouettes. Before learning parts, we extract the silhouette of the signer's dominant hand from each frame. Our train and test sets are constructed so that skin can be accurately detecting using an intensity threshold. In realistic environments, a more robust skin detection algorithm [18] would be needed. After locating the region corresponding to the dominant hand and arm based on its position in the frame, we discard the portion corresponding to the arm by finding the wrist, a local minimum in the horizontal thickness of the region. Our algorithm deals with slight errors in wrist detection by learning to include small unremoved arm pieces in the dictionary of parts. Finally, extracted silhouettes from each frame (examples shown in Fig. 2) are translated so that their centroids align and are stored as 201×201-pixel binary masks.

Unsupervised Learning of a Dictionary of Parts. These silhouettes can be partitioned into a small number of mostly convex parts. Each part is defined by its shape and location relative to the centroid of the silhouette. The largest part corresponds to the palm and any bent fingers occluding it. The remainder of the silhouette consists of disconnected parts corresponding to extended or partially extended groups of fingers or to sections of the arm that were not properly removed. In Fig. 3(b), the silhouette from Fig. 3(a) has been broken into parts. The outline of each piece is shown.

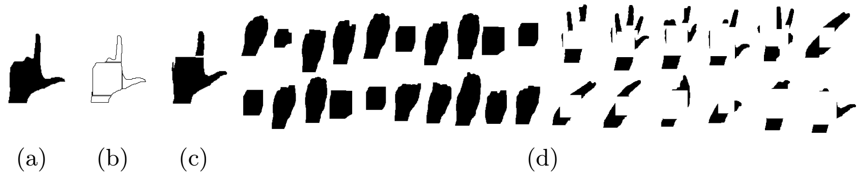


Fig. 3. A hand silhouette (a) is broken into parts, indicated by their outlines in (b). The reconstruction of this silhouette using the dictionary of parts in (d) is shown in (c). This dictionary was learned from the training set described in Sect. 5. Each part is displayed using the corresponding binary mask, with multiple non-overlapping non-palm parts drawn in the same image to conserve space. We successfully learn semantically meaningful palms and groups of fingers in an unsupervised fashion.

We extract parts from silhouettes using morphological operations. The palm part is extracted by performing a sequence of erosions and dilations. After a few erosions, the appendages disappear, indicated by the convexity of the shape exceeding a fixed threshold. The dilations return the shape, corresponding to the palm, to its original size but do not regrow the removed appendages. No morphological operations are performed on the non-palm parts, which remain when the extracted palm is subtracted from the original silhouette. All parts are represented by binary masks with dimensions equal to the size of the input silhouette (201×201 pixels). The location of the non-zero region encodes the location of the part relative to the centroid of the hand.

The final dictionary contains parts representing the most frequently occurring pieces. After extracting pieces from a training set, we cluster the palm pieces and the non-palm pieces separately using k-means clustering. To increase clustering speed, we reduce the dimensionality of each piece using PCA. We include the medioids of each returned cluster in our dictionary. Increasing the size of the dictionary improves the expressiveness of the representation but decreases computational efficiency and requires more training data. The dictionary learned from our training set (see Sect. 5) is shown in Fig. 3(d). Each connected component (20 palms and 40 non-palms) is a separate part.

Reconstruction from Parts. Given a learned dictionary, we compute representations of novel hand silhouettes by reconstructing the new shape as closely as possible while simultaneously using as few parts as possible. We first extract the palm part from the novel silhouette using morphological operations and select the part from the palm section of the dictionary that minimizes the total number of incorrect pixels. Next, we greedily add parts from the remaining (non-palm) portion of the dictionary until adding parts no longer improves the reconstruction. At each iteration, we tentatively add each unused part to the reconstruction by increasing the value of all pixels inside its mask by one, selecting the part which most reduces the total number of incorrect pixels in the reconstruction. To improve the invariance of the representation we search over a small set of affine transformations when finding the best fitting part. At termination, we return a bit-vector indicating which parts make up the final reconstruction. Figure 3(c) shows the reconstruction of the silhouette from Fig. 3(a) that uses five parts from the dictionary in Fig. 3(d).

Our reconstruction procedure is reminiscent of the matching pursuit algorithm [19]. However, our technique forces the reconstruction coefficients to be binary rather than real-valued and our dictionary of parts does not form an over-complete basis.

4.2 Hidden Markov Model

We train separate HMMs to recognize the transition between each pair of letters. To recognize fingerspelling sequences without knowing when each transition begins, we chain together the individual HMMs (called *subunits*). In this section, we describe the topology of the resulting HMM, the observation model, and the training and recognition processes. Rabiner's tutorial [20] provides a good review of HMMs and the related algorithms referenced here.

HMM Topology and Observation Model. Each subunit is a five-state Bakis topology HMM [21] (see Fig. 4). Observations in the first and last states usually correspond to the handshapes of the two letters. Observations in the three internal states capture configurations appearing during the transition. Skip transitions accommodate transitional motions performed at varying rate and phase relative to video sampling times. In the complete HMM, we connect subunits together using a bigram language model over letter transitions, introducing transitions between final and initial states of the subunits that form trigrams.

With our representation of handshape, each frame contains one palm and any combination of the non-palms. Thus, with a dictionary containing P palm parts and F non-palm parts, there are $P \cdot 2^F$ possible observations at each frame. It is too costly to try to learn or store the exact distribution over all possible observations. Instead, we approximate with the factored distribution

$$\mathcal{P}(p, f_1, \ldots, f_F) = \mathcal{P}(p) \prod_{i=1}^{F} \mathcal{P}(f_i|p)\,, \tag{1}$$

which requires only $(P-1)+P\cdot F$ parameters for each state. The $P-1$ parameters define a multinomial distribution over palm parts. The remaining parameters define $P \cdot F$ binomial distributions over the existence or non-existence of each non-palm part conditioned on the palm used.

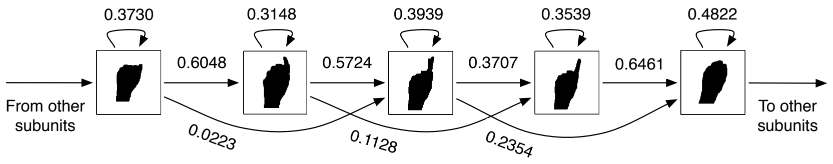


Fig. 4. A letter-to-letter transition HMM trained to recognize the $a \rightarrow t$ transition. Edges correspond to transitions with non-zero probability. States model handshapes found at different points during the transitional motion. The most likely observation is displayed to represent each state.

Training. The subunits are trained independently using isolated sequences corresponding to the desired letter pair. Given a clip of continuous fingerspelling, we hand-label a single frame for each letter signed. (These frames are the ones previous methods use for recognition.) We then use all the frames between the two labeled frames as an example of a given transition. During training, we ensure that each sequence ends in the final state of the subunit by adding a non-emitting state reachable only from the final emitting state. The parameters of each subunit HMM are estimated using the standard Baum-Welch algorithm [20]. Initial observation models are learned by assuming that the true path for each training sequence traverses all five states and remains in each state for $\frac{1}{5}$ of the total number of frames. The state transitions are initialized to be uniform over those allowed by our topology. Figure 4 shows the learned HMM for the $a \rightarrow t$ transition. Each state is represented by its most probable observation.

Recognition. To recognize a sequence of letters we compute the Viterbi path [20] through the full HMM. Our recognized sequence of letters follows from the sequence of subunits the path traverses. The path traverses a subunit only if it reaches one of the final two states, which keeps us from recognizing a letter pair when only the first letter of the pair is actually signed.

5 Results

To construct a challenging test vocabulary, we built a third-order letter-level model of English words (from Joyce's *Ulysses*), and included the 50 most common letter pairs. These 50 digrams account for 48% of all letter pairs, and contain 18 different letters. We then listed all trigrams (a total of 186) containing these 50 digrams that occurred with a frequency of at least 10^{-4}. We built an 82-word vocabulary (listed in Fig. 5) containing each trigram at least once. The perplexity, 2^H (where H is entropy [22]), of this vocabulary is 5.53 per digram. By comparison, the perplexity of an equivalent model built from the 1,000 most common English words is 10.31. Our reduced perplexity results from the prevalence of vowels in the top 50 digrams.

Our training set consists of 15 frame-per-second video of 10 examples of each word (29,957 frames total); a separate testing set contains 10 additional examples of each word (28,923 frames). Training and test data amount to about 65 minutes of video. Each frame is originally 640×480 pixels, with the hand occupying a region no larger than 200×200 pixels. We learn the dictionary of parts using unlabeled frames from the training set and train the HMMs using labeled frames. No portion of the training set was used to test the performance of any algorithm. After blind review of this paper, these data will be made available online.

alas, andes, aroma, atoned, beating, bed, below, berate, bestowal, chased, cheat, cheng, chinese, chisel, chow, coma, conde, contend, coral, corinth, courant, delores, easter, eden, elitist, eraser, halo, handed, hang, hare, healed, helen, hero, hinder, hither, home, hour, lane, larine, latest, lathered, line, long, male, marathon, master, mate, meander, medea, mentor, merited, near, rarest, realist, releases, rise, roman, ron, row, sealer, sentinel, serene, teal, testing, that, then, these, this, thor, tithed, tome, urease, urine, velour, venerate, vera, vest, wales, wand, war, wasteland, water

Fig. 5. The 82-word vocabulary.

5.1 Competing Algorithms

To isolate the effect of recognizing letter transitions from choices of handshape representation and probabilistic model, we compare our performance to two alternate systems (`Alt1` and `Alt2`), both of which share our handshape representation and observation model. Both `Alt1` and `Alt2` use an HMM with one state corresponding to each letter, with observation models trained on isolated instances of the corresponding letters. The HMM for `Alt1` contains a single state modeling all non-letter handshapes. In the `Alt2` HMM, we form 18 identical copies of the non-letter state, one for each letter state. The replicated non-letter states permit transitions between only those pairs of letters that occur in our vocabulary. In both systems, recognition is performed by computing the Viterbi path and discarding the frames assigned to the non-letter state(s).

5.2 Performance Comparison

We classified isolated digrams and entire words using our method (`L2L`) and the comparison methods that recognize letters only (`Alt1` and `Alt2`). Figure 6(a) shows the distribution of recognition performance for the three algorithms over the isolated digrams. To quantify the severity of a recognition error on a word, we compute the letter error rate (LER) for each word by computing the ratio of the number of incorrect letters (insertions, deletions, or substitutions) to the total number of letters recognized. The *per letter performance* for that word

Table 1. Performance of `Alt2` and `L2L` with and without a dictionary, averaged over the entire test set (10 examples each of 82 different words). Most recognition errors in `L2L` without a dictionary are single letter substitutions or missing final letters.

	`Alt2`	`Alt2+dict`	`L2L`	`L2L+dict`
Digrams correct	53.44%	60.61%	69.64%	72.85%
Words recognized with no incorrect letters	31.83%	86.59%	57.32%	92.68%
Per letter performance on full words	76.97%	90.86%	86.85%	94.75%

is then $1 - \text{LER}$. Figure 6(b) shows the distribution of per letter performance over our test words. `L2L` outperforms the alternative techniques on both isolated digrams and full words.

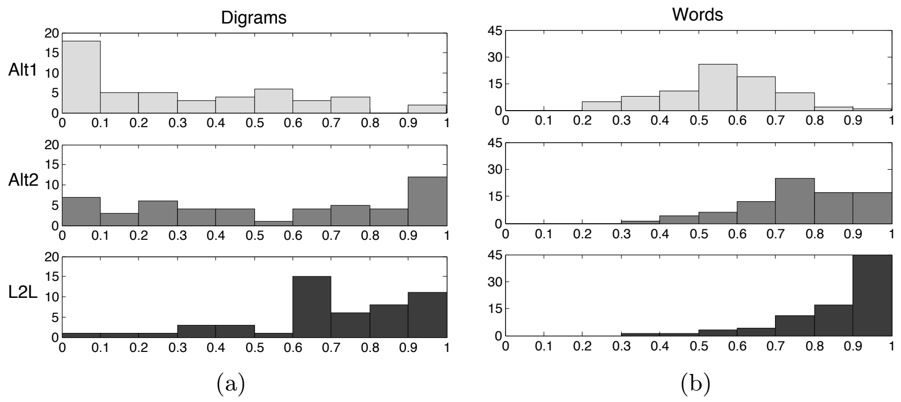


Fig. 6. A comparison of the performance of the proposed method to the two alternatives. All graphs show a count distribution of the fraction of letters recognized correctly. (a) Counts over 50 isolated digrams. (b) Counts over 82 different words. Top row: `Alt1`; middle row: `Alt2`; bottom row: `L2L` (our method). `L2L` recognizes a larger percentage of digrams and words with increased accuracy.

Adding an explicit dictionary to both `Alt2` and `L2L` will improve performance by restricting the space of possible words. Table 1 contains a summary of recognition performance of both techniques with and without a dictionary. While adding a dictionary improves the performance of both `Alt2` and `L2L`, modeling transitions results in better recognition accuracy than modeling letters in isolation with or without the help of a dictionary.

6 Discussion

We have introduced a principle for fingerspelling recognition that bypasses the difficult task of identifying an isolated frame for each letter and no longer ignores the dynamic nature of fingerspelling sequences. Our experiments show that modeling transitions between letters instead of isolated static handshapes for each letter improves recognition accuracy. Modeling transitions results in a recognition system that leverages information available while a letter is being formed to disambiguate between letters whose handshapes appear similar in single-camera video. Additionally, because the letter transition model includes multiple HMM states for each letter depending on the surrounding context, it can learn differences in handshape caused by coarticulation [23].

The benefit of modeling transitions is most apparent when no dictionary is used to aid recognition. While dictionaries are commonly used in deployed speech or word-level ASL recognition, we believe a system that does not rely on an explicit dictionary is more suited to fingerspelling recognition. Signers use fingerspelled signs exactly when the word they need is uncommon enough to not have a word-level sign. Thus, a deployed system would be most useful when it could correctly interpret uncommon words such as proper nouns that are likely not to be included in a reasonably-sized dictionary constructed during training.

The largest drawback to modeling and recognizing transitions between letters instead of isolated letters is the increase in the number of classes from 26 to 26^2. Although this increases the need for training data, it does not pose an insurmountable obstacle. For example, a hybrid method that models interesting transitions in detail but uninformative transitions at the level of `Alt2` would help manage the complexity of the resulting system. Additionally, techniques commonly employed in speech recognition such as tying similar states together could be used to make it possible to train the HMM with a reasonable amount of training data.

Our goal in this paper was not to demonstrate a deployable fingerspelling recognition system, but rather a useful principle for analysis. Much work remains before we reach a practical system, including generalizing to the full alphabet and multiple signers, dealing with cluttered environments, and interfacing with a word-level recognition system. Nonetheless, our demonstration of the feasibility of modeling transitions between letters represents a step toward a system that will recognize native ASL.

References

1. Tsechpenakis, G., Metaxas, D., Neidle, C.: Learning-based dynamic coupling of discrete and continuous trackers. Computer Vision and Image Understanding **104**(2-3) (2006) 140–156
2. Starner, T., Pentland, A., Weaver, J.: Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(12) (1998) 1371–1375

3. Vogler, C., Metaxas, D.: A Framework for Recognizing the Simultaneous Aspects of American Sign Language. Computer Vision and Image Understanding **81**(3) (2001) 358–384
4. Fang, G., Gao, W., Chen, X., Wang, C., Ma, J.: Signer-independent Continuous Sign Language Recognition Based on SRN/HMM. IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time (2001) 90–95
5. Allen, J., Asselin, P., Foulds, R.: American Sign Language Finger Spelling Recognition System. IEEE 29th Annual Northeast Bioengineering Conference (2003) 285–286
6. Oz, C., Leu, M.: Recognition of Finger Spelling of American Sign Language with Artificial Neural Network Using Position/Orientation Sensors and Data Glove. 2nd International Symposium on Neural Networks (2005) 157–164
7. Hernandez-Rebollar, J., Lindeman, R., Kyriakopoulos, N.: A Multi-Class Pattern Recognition System for Practical Finger Spelling Translation. 4th International Conference on Multimodal Interfaces (2002) 185–190
8. Dreuw, P., Keysers, D., Deselaers, T., Ney, H.: Gesture Recognition Using Image Comparison Methods. Sixth International Workshop on Gesture in Human-Computer Interaction and Simulation (2005) 124–128
9. Feris, R., Turk, M., Raskar, R., Tan, K., Ohashi, G.: Exploiting Depth Discontinuities for Vision-Based Fingerspelling Recognition. IEEE Workshop on Real-time Vision for Human-Computer Interaction (2004)
10. Lamar, M., Bhuiyan, M., Iwata, A.: Hand Alphabet Recognition using Morphological PCA and Neural Networks. International Joint Conference on Neural Networks (1999) 2839–2844
11. Tomasi, C., Petrov, S., Sastry, A.: 3D Tracking = Classification + Interpolation. International Conference on Computer Vision (2003) 1441–1448
12. Birk, H., Moeslund, T., Madsen, C.: Real-Time Recognition of Hand Alphabet Gestures Using Principal Component Analysis. 10th Scandinavian Conference on Image Analysis (1997) 261–268
13. Athitsos, V., Sclaroff, S.: Estimating 3D Hand Pose from a Cluttered Image. IEEE Conference on Computer Vision and Pattern Recognition (2003) 432–439
14. Goh, P., Holden, E.: Dynamic Fingerspelling Recognition using Geometric and Motion Features. International Conference on Image Processing (2006) 2741–2744
15. Videos used in experiments include clips from the John A. Logan College Interpreter Training Program (www.jalc.edu/ipp) and the DVDs *Fast Expressive Fingerspelling Practice*, *Fingerspelled Stories from 10 to 45 words per minute* (both available from www.drsign.com), and *Fingerspelling: Expressive and Receptive Fluency* (available from www.dawnsign.com).
16. Padden, C.: Learning to fingerspell twice: Young signing children's acquisition of fingerspelling. In Marschark, M., Schick, B., Spencer, P., eds.: Advances in Sign Language Development by Deaf Children. Oxford University Press, New York (2006) 189–201
17. Vogler, C., Metaxas, D.: Handshapes and Movements: Multiple-Channel American Sign Language Recognition. 5th International Workshop on Gesture and Sign Language Based Human-Computer Interaction (2003) 247–258
18. Jones, M., Rehg, J.: Statistical Color Models with Application to Skin Detection. International Journal of Computer Vision **46**(1) (2002) 81–96
19. Mallat, S., Zhang, Z.: Matching Pursuit in a Time-Frequency Dictionary. IEEE Transactions on Signal Processing **41**(12) (1993) 3397–3415

20. Rabiner, L.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE **77**(2) (1989) 257–286
21. Bakis, R.: Continuous speech recognition via centisecond acoustic states. Journal fo the Acoustical Society of America **59**(S1) (1976) S97
22. Jelinek, F., Mercer, R., Bahl, L., Baker, J.: Perplexity–a measure of the difficulty of speech recognition tasks. Journal of the Acoustical Society of America **62**(S1) (1977) S63
23. Jerde, T., Soechting, J., Flanders, M.: Coarticulation in Fluent Fingerspelling. Journal of Neuroscience **23**(6) (2003) 2383