Local Frame Match Distance: A Novel Approach for Exemplar Gesture Recognition

Radu Tudor Ionescu*, Marius Popescu*, Christopher Conly†, Vassilis Athitsos†
*Department of Computer Science
University of Bucharest, Romania
E-mails: raducu.ionescu@gmail.com, popescunmarius@gmail.com
†Department of Computer Science and Engineering
University of Texas at Arlington, TX
Emails: chris.conly@uta.edu, athitsos@uta.edu

***Abstract*—Gesture recognition using a training set of limited size for a large vocabulary of gestures is a challenging problem in computer vision. With few examples per gesture class, researchers often employ state-of-the-art exemplar-based methods such as Dynamic Time Warping (DTW). This paper makes two contributions in the area of exemplar-based gesture recognition. As an alternative to DTW, we first introduce the Local Frame Match Distance (LFMD), a novel approach for matching gestures inspired by a distance measure for strings, namely Local Rank Distance (LRD). While LRD efficiently approximates the non-alignment of character n-grams between two strings, we employ LFMD to efficiently measure the non-alignment of hand locations between two video sequences. Second of all, we transform LFMD into a kernel and use it in combination with Kernel Discriminant Analysis for sign language recognition with exemplars. The empirical results indicate that our method can generally yield better performance than a state-of-the-art DTW approach on the challenging task of American Sign Language recognition, while reducing the computational time by 30%.**

1. Introduction

Gesture and sign language recognition represent a challenging research area in computer vision. Popular probabilistic methods such as Hidden Markov Models (HMM) [1] and Conditional Random Fields (CRF) [2] require large training sets to learn good probability distributions. This requirement often limits the size of the set of gestures (vocabulary) that can be recognized by such systems. When a large vocabulary is desired, time constraints may force researchers to restrict the size of the training set to only a few examples per gesture class. As using few examples per class prohibits the use of many statistical and machine learning methods, researchers are often limited to exemplar-based recognition and similarity measures. In such cases, Dynamic Time Warping (DTW) [3] is frequently used on hand location or other information to generate scores that serve as a measure of similarity to training examples [4], [5], [6], [7]. DTW has been improved with the use of a well-designed feature vector that includes more than hand positions to represent the state of a gesture at each point in time [8].

In this paper, we propose an alternative solution to DTW, inspired by a distance measure for strings, namely Local Rank Distance (LRD) [9]. LRD has successfully been used for a broad range of tasks from phylogenetic analysis [9] and sequence alignment [10] to native language identification [11], [12], [13] and Arabic dialect identification [14]. LRD essentially measures the non-alignment (displacement) of character n-grams between two strings. Previous results indicate that LRD is more accurate [10] and can be computed faster [12] than the edit distance [15]. Since both DTW and edit distance are solved by dynamic programming, we can obtain a more efficient algorithm by adapting LRD for gesture recognition from video. Hence, we introduce the Local Frame Match Distance (LFMD) algorithm to measure the distance (or similarity) between two gestures. In order to use LFMD for gesture recognition, we first transform it into a kernel function using the squared RBF kernel [16] and then we employ Kernel Discriminant Analysis (KDA) [16] to train our gesture classifier. To the best of our knowledge, KDA has never been used for exemplar-based gesture recognition. We compare our gesture recognition approach with a state-of-the-art approach based on DTW on the American Sign Language Lexicon Video Dataset (ASLLVD) [17]. The empirical results indicate that our approach can yield better performance, while reducing the computational time by 30%.

The paper is organized as follows. Related work on gesture and sign language recognition is presented in Section 2. Our learning framework is described in Section 3. The sign language recognition experiments are presented in Section 4. Finally, we draw our conclusions in Section 5.

2. Related Work

Most recent works have been in action and activity recognition, some from static images [18], others from video [19]. These works tend to focus on classifying small vocabularies of general actions, rather than discriminating between specific actions such as language signs. Some action recognition works do test their methods on gesture

data sets [20], [21], but the vocabularies are limited, and the methods are generally not directly applicable to larger vocabulary gesture sets. A second area of research focuses on generalized gesture recognition. The sets of gestures may be created specifically for this task, and can be chosen so as to minimize similarity between classes. Long Short Term Memory (LSTM) networks have proven successful for this task [22]. With the release of ChaLearn Gesture Challenge data set [23], there have been a number of works in one-shot learning, in which a single training example is used per gesture class [24], [25], [26]. A third focus is on developing methods that work on well-established gesture sets, such as sign languages. One branch of work deals in continuous sign language recognition and fingerspelling [27]. Another branch of sign language recognition research focuses instead on classification of individually segmented signs. One popular intuitive method is to segment a sign into motion or other types of sub-units and then use an HMM to model the temporal changes in sub-units throughout each sign [28], [29]. Dynamic Time Warping has also been used for action and gesture recognition [7], [8], [30], [31] and it has shown its superiority over LSTM and HMM models [31]. Some of these works approach the idea of class variability modeling [30], [31].

3. Method

We propose a gesture and sign language recognition system, given the hand trajectories of the gestures. We first compute a feature matrix for each hand gesture, as described in Section 3.1. Next, we compute the distance of two hand gestures by employing our novel algorithm presented in Section 3.2. Finally, we train a Kernel Discriminant Analysis classifier to recognize new hand gestures based on a kernel derived from the pairwise distances between gestures, as detailed in Section 3.3.

3.1. Feature Representation of Hand Gestures

To represent a hand gesture, we use the feature vector introduced in [8]. The feature vector based on 2D hand position information is built for each video frame in order to describe what is occurring at every point in time. The hand positions are first expressed in a face-centric coordinate system. For one-handed signs, the position of the non-dominant hand is set to $(0, 0)$, hence it will not contribute to the similarity score. The following features compose the vectors for each frame t of gesture video X:

- $L_d(X,t)$ and $L_{nd}(X,t)$: 2D pixel coordinates of the dominant and non-dominant hands.
- $O_d(X,t) = L_d(X,t+1) - L_d(X,t)$ and $O_{nd}(X,t) = L_{nd}(X,t+1) - L_{nd}(X,t)$: motion direction from frame t to frame $t+1$ for the dominant and non-dominant hands.
- $L_\delta(X,t) = L_d(X,t) - L_{nd}(X,t)$: position of the dominant hand relative to the non-dominant hand.
- $O_\delta(X,t) = L_\delta(X,t+1) - L_\delta(X,t)$: direction of change for L_δ from frame t to frame $t+1$.

In total, there are 12 features in the vector representing each video frame. The feature vectors are combined into a single matrix to describe the sign. In the experiments, we use manual annotations of the hand positions. The hand gesture is size-normalized so that the diagonal of the face bounding box is 1. Finally, the frame length is normalized to 24 frames using bicubic interpolation on the feature matrix, as in [8]. Hence, the size of the feature matrix for a hand gesture becomes 12×24.

3.2. Local Frame Match Distance

In this section, we describe a novel algorithm for computing the similarity (or the distance) between the hand trajectories of two gestures. Our algorithm is inspired by LRD [9] which has successfully been applied to phylogenetic analysis [9], sequence alignment [10], native language identification [11], [12], [13] and Arabic dialect identification [14], [32]. We next present how we adapt LRD and obtain a novel algorithm, termed *Local Frame Match Distance*, for the task of gesture recognition. Given the hand locations in each video frame, we match each hand location from the first video sequence to the nearest hand location (in terms of the features derived from pixel coordinates) in the second video sequence. Then, we compute the sum of the absolute differences between the indexes of matched frames. As LRD operates on character n-grams in order to yield better performance, we can also extend the LFMD algorithm to match sets of consecutive hand locations to achieve the same goal. Local Frame Match Distance is formally presented in Algorithm 1. We use the following notations for describing the algorithm. An array (or an ordered set of elements) is denoted by $V = (v_1, v_2, ..., v_n)$ and the length of V is denoted by $|V| = n$. Arrays are considered to be indexed starting from position 1, thus $V[i] = v_i, \forall i \in \{1, 2, ...n\}$. We extend this notation to matrices as well, therefore we consider that $M[i, j]$ represents the element on row i and column j of the matrix M.

The goal of Algorithm 1 is to compute a distance between two hand trajectories represented as features matrices X and Y. As LRD obtains generally better results when matching character n-grams instead of single characters, we also want to match a set of consecutive frames in X with another set of consecutive frames in Y by minimizing a cost function. For the sake of simplicity, we will refer to a set of consecutive frames $X_{i:i+p-1} = \{X_i, X_{i+1}..., X_{i+p-1}\}$ as a *p-frame*, where p denotes the number of frames considered in the set denoted by $X_{i:i+p-1}$. For individual frames X_i and Y_j, we employ the same cost function as in [8], but we assign equal weights to all the features, therefore eliminating the weights and the need to tune them on a validation set:

$$\begin{aligned} cost(X_i, Y_j) = &\|L_d(X,i) - L_d(Y,j)\|_2 + \\ &+ \|L_{nd}(X,i) - L_{nd}(Y,j)\|_2 + \\ &+ \|O_d(X,i) - O_d(Y,j)\|_2 + \\ &+ \|O_{nd}(X,i) - O_{nd}(Y,j)\|_2 + \\ &+ \|L_\delta(X,i) - L_\delta(Y,j)\|_2 + \\ &+ \|O_\delta(X,i) - O_\delta(Y,j)\|_2 , \end{aligned} \tag{1}$$

Algorithm 1: LFMD Algorithm

1 **Input**:
2 X, Y – the input feature matrices for two hand gestures;
3 n – the number of frames of the two hand gestures;
4 p – the number of consecutive frames to be matched;
5 m – the maximum spatial offset ($m \leq n$);
6 **Notations**:
7 C – the $(n-p+1) \times (n-p+1)$ cost matrix of all possible pairs of p-frames from X and Y;
8 M_X – the vector with minimal costs for each p-frame in X;
9 J_X – the indexes of the p-frames in Y corresponding to the minimal costs in M_X;
10 M_Y – the vector with minimal costs for each p-frame in Y;
11 J_Y – the indexes of the p-frames in X corresponding to the minimal costs in M_Y;
12 **Initialization**:
13 $M_X \leftarrow (\infty, \infty, ..., \infty)$, such that $|M_X| = n-p+1$;
14 $M_Y \leftarrow (\infty, \infty, ..., \infty)$, such that $|M_Y| = n-p+1$;
15 $J_X \leftarrow (0, 0, ..., 0)$, such that $|J_X| = n-p+1$;
16 $J_Y \leftarrow (0, 0, ..., 0)$, such that $|J_Y| = n-p+1$;
17 **Computation**:
18 **if** $p \geq 2$ **then**
19 **for** $i \in \{1, ..., min\{m, n-p\}\}$ **do**
20 $C[i+p-2, p-1] \leftarrow cost(X_{i+p-2}, Y_{p-1})$;
21 $C[p-1, i+p-2] \leftarrow cost(X_{p-1}, Y_{i+p-2})$;
22 **for** $j \in \{p-2, ..., 1\}$ **do**
23 $C[i+j-1, j] \leftarrow cost(X_{i+j-1}, Y_j) + C[i+j, j+1]$;
24 $C[j, i+j-1] \leftarrow cost(X_j, Y_{i+j-1}) + C[j+1, i+j]$;
25 **for** $i \in \{1, ..., n-p+1\}$ **do**
26 **for** $j \in \{1, ..., n-p+1\}$ **do**
27 **if** $|i-j| \leq m$ **then**
28 $C[i+p-1, j+p-1] \leftarrow cost(X_{i+p-1}, Y_{j+p-1})$;
29 **for** $k \in \{p-2, ..., 0\}$ **do**
30 $C[i+k, j+k] \leftarrow$
$C[i+k, j+k] + C[i+p-1, j+p-1]$;
31 **if** $C[i,j] < M_X[i]$ **then**
32 $M_X[i] \leftarrow C[i,j]$;
33 $J_X[i] \leftarrow j$;
34 **if** $C[i,j] < M_Y[j]$ **then**
35 $M_Y[j] \leftarrow C[i,j]$;
36 $J_Y[j] \leftarrow i$;
37 $\Delta \leftarrow 0$;
38 **for** $i \in \{1, ..., n-p+1\}$ **do**
39 $\Delta \leftarrow \Delta + M_X[i] \cdot |J_X[i] - i|$;
40 **for** $j \in \{1, ..., n-p+1\}$ **do**
41 $\Delta \leftarrow \Delta + M_Y[j] \cdot |J_Y[j] - j|$;
42 **Output**:
43 Δ - the Local Frame Match Distance between X and Y.

where $\|\cdot\|_2$ represents the L_2-norm. For p-frames $X_{i:i+p-1}$ and $Y_{j:j+p-1}$, we naively consider the cost given by diagonally aligning the individual frames:

$$cost(X_{i:i+p-1}, Y_{j:j+p-1}) = cost(X_i, Y_j) + \\ + cost(X_{i+1}, Y_{j+1}) + ... + \\ + cost(X_{i+p-1}, Y_{j+p-1}). \quad (2)$$

In a similar way to DTW, we build a cost matrix C for each pair of p-frames in X and Y. We first pre-compute some of the components of the matrix C in steps 18-24 of Algorithm 1. The rest of the components are computed in steps 25-30. It is important to note that we compute only those components for which the absolute difference between their indexes is less than a parameter m (step 27). In the experiments, we set $m = 10$ and thus obtain some speed improvement compared to DTW. Right after the cost $C[i,j]$ between two p-frames $X_{i:i+p-1}$ and $Y_{j:j+p-1}$ is computed, we check if this cost is minimal on the row i (step 31) or the column j of C (steps 34), in which case we store the minimal value for future comparisons (steps 32 and 35, respectively) and the corresponding indexes (steps 33 and 36, respectively). We can now compute the distance between X and Y by adding the minimal costs for each row (steps 38-39) and each column (steps 40-41) of C. At this point, we multiply the minimal cost with the offset between the indexes of the matched p-frames (steps 39 and 41, respectively). While the minimal cost accounts for the spatial difference between p-frames, the offset between indexes accounts for the temporal difference between the matched p-frames. Intuitively, a larger temporal difference indicates that the hand trajectories are less similar, hence the distance between them should be greater.

It is interesting to note that LRD measures the spatial non-alignment of two strings as the sum of all the spatial offsets between pairs of identical character n-grams in the two strings. In our case, we consider pairs of p-frames, but we do not want to match their spatial features exactly, as two hand trajectories representing the same gesture class are almost never the same, even if they are performed by the same user. Instead of trying to find identical p-frames, we match p-frames by minimizing the cost defined in Equation (2). While LRD computes a spatial non-alignment of identical character n-grams, LFMD measures the temporal non-alignment of matching p-frames. In practice, LRD works better when n-grams of multiple lengths are combined together [11], [12], [13], [14]. Algorithm 1 can be easily extended to treat the case in which a blended range of p-frames is used. The extension is based on the observation that the cost between shorter p-frames is included in the cost of longer p-frames. Indeed, we can trivially demonstrate this by induction from p to $p+1$ using Equation (2):

$$cost(X_{i:i+p}, Y_{j:j+p}) = cost(X_{i:i+p-1}, Y_{j:j+p-1}) + \\ + cost(X_{i+p}, Y_{j+p}). \quad (3)$$

Moreover, we can perform the algorithm extension to a blended range of p-frames, without increasing its time complexity. However, the space required increases linearly with the range of values considered for p. In the experiments, we consider only 1-frames, 2-frames and 3-frames. Since p and m can be considered as constants, the time complexity of the proposed algorithm is essentially quadratic in terms of the number of frames of the two gestures.

3.3. Kernel Learning Method

Kernel-based learning algorithms work by embedding the data into a Hilbert feature space and by searching for linear relations in that space. The embedding is performed

TABLE 1. THE ACCURACY RATES OF DTW AND 1-NN VERSUS SEVERAL LFMD AND KDA MODELS BASED ON VARIOUS RANGES OF p-FRAMES. THE METHODS ARE COMPARED USING A 3-FOLD CROSS-VALIDATION PROCEDURE. THE BEST RESULT FOR EACH k IS HIGHLIGHTED IN BOLD.

Top k	DTW and 1-NN	LFMD and KDA				
		1-frames	2-frames	3-frames	$\{1,2\}$-frames	$\{1,2,3\}$-frames
1	34.38%	34.65%	35.21%	34.35%	35.65%	**36.42%**
5	60.77%	62.41%	63.13%	61.96%	63.40%	**64.54%**
10	71.88%	72.00%	72.12%	71.94%	73.20%	**74.48%**
20	80.11%	79.31%	80.08%	76.63%	81.04%	**81.28%**
30	84.16%	82.63%	84.07%	83.32%	84.22%	**84.97%**
50	88.41%	86.88%	88.08%	87.72%	88.62%	**89.04%**
100	93.41%	92.15%	92.99%	92.90%	93.14%	**93.80%**

implicitly, that is by specifying the inner product between each pair of points rather than by giving their coordinates explicitly. A kernel matrix that contains the pairwise similarities between every pair of training samples is used in the learning stage. As LFMD needs to be used as a kernel function, we employ the RBF kernel [16] to transform LFMD into a similarity measure:

$$k(X,Y) = e^{-\frac{\Delta(X,Y)}{2\sigma^2}},$$

where Δ is the Local Frame Match Distance between gestures X and Y. The parameter σ is usually chosen such that values of $k(X,Y)$ are well scaled. In this context, we set $\sigma = 0.3$ in the experiments. The resulted similarity matrix is then squared in order to make sure it becomes a symmetric and positive definite kernel matrix.

After embedding the features with a kernel function, a linear classifier is used to select the most discriminant features. Various kernel classifiers differ in the way they learn to separate the samples. There are some classifiers that take the multi-class nature of the gesture recognition problem directly into account, such as Kernel Discriminant Analysis (KDA). The KDA method provides a projection of the data points to a one-dimensional subspace where the Bayes classification error is smallest. It is able to improve accuracy by avoiding the class masking problem [33]. We use KDA in our sign language recognition experiments. We set the regularization parameter of KDA to 2, for increased regularization (less overfitting). Further details about kernel methods can be found in [16].

4. Experiments

4.1. Data Set

The sign language data set used in the experiments is composed of 1113 distinct sign classes. For each sign class there are three examples, each from a different user. In total, there are 3339 examples. All sign videos and annotations have been acquired from the ASLLVD [17].

4.2. Evaluation

We compare our approach based on LFMD and KDA with a state-of-the-art approach based on DTW and k-Nearest Neighbors (k-NN). We train and evaluate the sign language recognition systems in a user-independent setting, by using 3-fold cross-validation. In each fold, the users performing the signs included the training set are different from those performing the signs included the test set. To assess the performance level of the considered systems, we define the measure of accuracy to be the percentage of signs whose correct match is ranked in the top k most similar signs for each $k \in \{1, 5, 10, 20, 30, 50, 100\}$.

4.3. Results and Discussion

Table 1 shows the results of our system based on various ranges of p-frames in comparison with a state-of-the-art approach based on DTW and 1-Nearest Neighbors (1-NN). When using 1-frames, 2-frames and 3-frames alone, the LFMD and KDA approach does not achieve better results than the DTW and 1-NN approach. It generally seems that our approach can bring some performance improvements over DTW when the accuracy is measured for the top 1 or top 5 retrieved signs, but the performance of DTW is higher for values of k larger than 20. Interestingly, we can obtain better results for all k values when we combine p-frames. Indeed, our best performance is obtained when we put 1-frames, 2-frames and 3-frames together. Our performance improvements over DTW are roughly 2.0% for $k = 1$, 3.8% for $k = 5$, 3.4% for $k = 10$, 1.1% for $k = 20$, 0.8% for $k = 30$, 0.6% for $k = 50$ and 0.5% for $k = 100$, respectively. Interestingly, the empirical results presented in [11], [12], [13], [14] also indicate that LRD obtains better performance when using n-grams in a range. For example, n-grams in the range 3-6 have been used for Arabic dialect identification [14].

We also compare the LFMD and the DTW algorithms in terms of time. Both algorithms are implemented in Java and used in similar settings. We measure the time required by the two algorithms to produce the 3339×3339 distance matrix for all the examples in the data set, on a computer with Intel Core i7 2.3 GHz processor and 8 GB of RAM using a single Core. The DTW algorithm takes about 3125 seconds, while the LFMD algorithm takes about 2186 seconds to compute the similarity based on $\{1,2,3\}$-frames. Thus, we obtain a speed improvement of 30%.

5. Conclusion

In this paper, we have presented a novel approach for sign and gesture recognition given the hand trajectories of

the gestures. Our approach is based on comparing the hand trajectories two by two using the novel Local Frame Match Distance algorithm. We have used LFMD in a learning context by transforming it into a kernel and by combining it with the KDA classifier. To the best of our knowledge, we are the first to use KDA for gesture recognition with exemplars. Overall, the sign language results of LFMD are better than the results of DTW, in terms of both accuracy and time. Using automatically annotated hand positions is beyond the scope of this paper, but we aim to use our framework in this rather more realistic setting in future work.

Acknowledgments

This work was partially supported by National Science Foundation grants IIS-1055062, CNS-1338118, and IIS 1565328.

References

[1] L. E. Baum and T. Petrie, "Statistical Inference for Probabilistic Functions of Finite State Markov Chains," *The Annals of Mathematical Statistics*, vol. 37, no. 6, pp. 1554–1563, 12 1966.

[2] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," *Proceedings of ICML*, pp. 282–289, 2001.

[3] J. B. Kruskal and M. Liberman, "The Symmetric Time-Warping Problem: from Continuous to Discrete," in *Time Warps, String Edits, and Macromolecules - The Theory and Practice of Sequence Comparison*, D. Sankoff and J. B. Kruskal, Eds. Stanford, CA 94305: CSLI Publications, 1999, ch. 4.

[4] T. Darrell and A. Pentland, "Space-time gestures," *Proceedings of CVPR*, pp. 335–340, Jun 1993.

[5] T. Darrell, I. Essa, and A. Pentland, "Task-specific Gesture Modeling using Interpolated Views," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 12, 1996.

[6] A. Corradini, "Dynamic Time Warping for Off-Line Recognition of a Small Gesture Vocabulary," *Proceedings of ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in RealTime Systems*, pp. 82–89, 2001.

[7] A. Stefan, H. Wang, and V. Athitsos, "Towards Automated Large Vocabulary Gesture Search," *Proceedings of PETRA*, pp. 16:1–16:8, 2009.

[8] H. Wang, A. Stefan, S. Moradi, V. Athitsos, C. Neidle, and F. Kamangar, "A system for large vocabulary sign search," *Trends and Topics in Computer Vision: ECCV 2010 Workshops, Revised Selected Papers, Part I*, pp. 342–353, 2012.

[9] R. T. Ionescu, "Local Rank Distance," *Proceedings of SYNASC*, pp. 221–228, 2013.

[10] L. P. Dinu, R. T. Ionescu, and A. I. Tomescu, "A rank-based sequence aligner with applications in phylogenetic analysis," *PLoS ONE*, vol. 9, no. 8, p. e104006, 08 2014.

[11] M. Popescu and R. T. Ionescu, "The Story of the Characters, the DNA and the Native Language," *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 270–278, June 2013.

[12] R. T. Ionescu, "A Fast Algorithm for Local Rank Distance: Application to Arabic Native Language Identification," *Proceedings of ICONIP*, vol. 9490, pp. 390–400, 2015.

[13] R. T. Ionescu, M. Popescu, and A. Cahill, "String kernels for native language identification: Insights from behind the curtains," *Computational Linguistics*, vol. 42, no. 3, pp. 491–525, 2016.

[14] R. T. Ionescu and M. Popescu, "UnibucKernel: An Approach for Arabic Dialect Identification based on Multiple String Kernels," *Proceedings of VarDial Workshop of COLING*, pp. 135–144, 2016.

[15] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reverseals," *Cybernetics and Control Theory*, vol. 10, no. 8, pp. 707–710, 1966.

[16] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[17] V. Athitsos, C. Neidle, S. Sclaroff, J. Nash, A. Stefan, Q. Yuan, and A. Thangali, "The American Sign Language Lexicon Video Dataset," *Proceedings of CVPR Workshop on Human Communicative Behaviour Analysis*, pp. 1–8, June 2008.

[18] Y. Wang, D. Tran, Z. Liao, and D. A. Forsyth, "Discriminative hierarchical part-based models for human parsing and action recognition," *Journal of Machine Learning Research*, vol. 13, pp. 3075–3102, 2012.

[19] Y. Tian, R. Sukthankar, and M. Shah, "Spatiotemporal Deformable Part Models for Action Detection," *Proceedings of CVPR*, pp. 2642–2649, June 2013.

[20] Y. Song, L.-P. Morency, and R. Davis, "Action recognition by hierarchical sequence summarization," *Proceedings of CVPR*, pp. 3562–3569, June 2013.

[21] B. Fernando, E. Gavves, J. M. Oramas, A. Ghodrati, and T. Tuytelaars, "Modeling Video Evolution for Action Recognition," *Proceedings of CVPR*, pp. 5378–5387, June 2015.

[22] O. Alsharif, T. Ouyang, F. Beaufays, S. Zhai, T. Breuel, and J. Schalkwyk, "Long short term memory neural network for keyboard gesture decoding," *Proceedings of ICASSP*, pp. 2076–2080, April 2015.

[23] I. Guyon, V. Athitsos, P. Jangyodsuk, B. Hamner, and H. Escalante, "ChaLearn gesture challenge: Design and first results," *Proceedings of CVPR Workshops*, pp. 1–6, 2012.

[24] J. Wan, Q. Ruan, W. Li, and S. Deng, "One-shot Learning Gesture Recognition from RGB-D Data Using Bag of Features," *Journal of Machine Learning Research*, vol. 14, no. 1, pp. 2549–2582, Jan. 2013.

[25] J. Konečný and M. Hagara, "One-shot-learning Gesture Recognition Using HOG-HOF Features," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 2513–2532, Jan. 2014.

[26] F. Jiang, S. Zhang, S. Wu, Y. Gao, and D. Zhao, "Multi-layered Gesture Recognition with Kinect," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 227–254, Jan. 2015.

[27] T. Kim, G. Shakhnarovich, and K. Livescu, "Fingerspelling Recognition with Semi-Markov Conditional Random Fields," *Proceedings of ICCV*, pp. 1521–1528, 2013.

[28] H. Cooper, E.-J. Ong, N. Pugeault, and R. Bowden, "Sign Language Recognition Using Sub-units," *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 2205–2231, Jul. 2012.

[29] H. Wang, X. Chai, Y. Zhou, and X. Chen, "Fast sign language recognition benefited from low rank approximation," *Proceedings of FG*, pp. 1–6, May 2015.

[30] M. Reyes, G. Dominguez, and S. Escalera, "Feature weighting in dynamic time warping for gesture recognition in depth data," *Proceedings of ICCV Workshops*, pp. 1182–1188, 2016.

[31] C. Conly, A. Dillhoff, and V. Athitsos, "Leveraging Intra-Class Variations to Improve Large Vocabulary Gesture Recognition," *Proceedings of ICPR*, 2016.

[32] R. T. Ionescu and A. Butnaru, "Learning to Identify Arabic and German Dialects using Multiple Kernels," *Proceedings of VarDial Workshop of EACL*, pp. 200–209, 2017.

[33] T. Hastie and R. Tibshirani, *The Elements of Statistical Learning*, corrected ed. Springer, Jul. 2003.