Extracting Commands From Gestures: Gesture Spotting and Recognition for Real-time Music Performance

Jiuqiang Tang[1] and Roger B. Dannenberg[2],

[1 2] Carnegie Mellon University
jiuqiant@andrew.cmu.edu
rbd@cs.cmu.edu

Abstract. Our work allows an interactive music system to spot and recognize "command" gestures from musicians in real time. The system gives the musician gestural control over sound and the flexibility to make distinct changes during the performance by interpreting gestures as discrete commands. We combine a gesture threshold model with a Dynamic Time Warping (DTW) algorithm for gesture spotting and classification. The following problems are addressed: i) how to recognize discrete commands embedded within continuous gestures, and ii) an automatic threshold and feature selection method based on F-measure to find good system parameters according to training data.

Keywords: Gesture Spotting and Recognition, Automatic Threshold and Feature Selection, Dynamic Time Warping, Threshold Model, F-Measure Evaluation.

1 Introduction

Musicians' gestures can be interpreted as either discrete commands or as continuous control over a set of parameters. For instance, a temporal gesture, which consists of a cohesive sequence of movements, could represent a command to start a sound, enter a new state, or make a selection among several discrete choices. Alternatively, continuous control gestures may communicate sound parameters such as gain, pitch, velocity, and panning. Continuous parameter sensing is relatively easy since most physiological parameters captured by sensors are represented as a continuous stream. Converting this stream to a continuous parameter is often a simple matter of applying a mapping function to each sensor value to compute a control value. On the other hand, it can be very difficult to spot and recognize temporal gesture patterns within a continuous signal stream. Common solutions for command interfaces include special sensors such as buttons on a keyboard or multiple spatial positions as command triggers. However, locating multiple physical keys or learning to reproduce absolute positions in live performance can be risky. Some "natural" gestures, such as nodding the head or pointing, offer an alternative way to communicate commands.

Achieving a high recognition rate for continuous gestures is more challenging than recognizing static poses or positions because a sequence of sensor values must be considered. In this paper, we explore the recognition of discrete command gestures within a stream of continuous sensor data. The approach is based on the well-known

dynamic time warping (DTW) sequence matching algorithm. While DTW provides a powerful mechanism for comparing two temporal sequences, it does not tell us what features will work best, how to pre-process the data to avoid outliers, how to spot a finite gesture in an infinite stream of input, or how to set thresholds to optimize recognition and classification performance. The goal of our study is to design and evaluate a gesture spotting and recognition strategy especially for music performance. We select features by searching over feature combinations, obtain optimal thresholds for each gesture pattern, and automatically generate a gesture recognizer.

2 Related Work

Recent research focuses on machine learning approaches for gesture-to-sound mapping in interactive music performance [1, 2, 3, 4]. Fiebrink's Wekinator [2] explores how end users can incorporate machine learning into real-time interactive systems, but the API/ Toolbox she provides uses a feature vector as input and creates a class as output. Users have to select features and models manually. Francoise built two models based on Hierarchical Hidden Markov Models to segment complex music gestures [1] in real time. He applied two strategies, a forward algorithm and Fixed-Lag smoothing, to recognize violin-bowing techniques during a real performance. His research mainly focused on how different strategies affect segmentation accuracy and recognition latency but did not clearly point toward a good combination over parameters that can increase detection accuracy.

In the field of gesture recognition, major approaches for analyzing spatial and temporal gesture patterns include Dynamic Time Warping [5, 6, 7, 8, 18], Neural Networks [9, 10, 11], Hidden Markov Models [12, 13] and Conditional Random Fields [14]. Many existing gesture spotting and recognition systems apply a threshold model for discriminating valid gesture patterns from non-gesture patterns [12, 15]. Lee and Kim developed an HMM-based adaptive threshold model approach [12]. This method runs off-line and is not a suitable threshold model for real-time musical gesture recognition and identification. Krishnan et al. proposed an adaptive threshold model [15]. In their model, several weakest classifiers, which can satisfy a majority of samples of all the training classes, are used to test whether the input is too general. Additionally, in order to deal with multiple deformations in training gesture data, Miguel A Bautista et al. proposed a probability-based DTW for gestures on RGB-D data [8], in which the pattern model is discerned from several samples of the same gesture pattern. They used different sequences to build a Gaussian-based probabilistic model of the gesture pattern. Similar to our work, Gillian et al. applied DTW to musical gesture recognition [18] and achieved over 80% recognition accuracy spotting command gestures in a continuous data stream. Our work extends this approach with automated feature selection and threshold optimization.

3 System Overview

Our music gesture recognition system uses training data to learn gestures. In our terminology, a *feature* is a sensor reading or a processed sensor reading, a *feature*

vector, or simply *vector*, is a set of simultaneous *features*, which typically include x, y, and z joint position coordinates. A gesture *sample* is a sequence of these *vectors*. A *template* is a *sample* that is representative of a particular command gesture. In operation, input is a continuous sequence of feature vectors. Overlapping windows of input are compared to templates using DTW, and output commands are issued when gestures are detected. We describe how input is compared to templates in Section 4.

Initially, the musician defines a gesture vocabulary, such as nodding their head, waving their hands, or making a circle motion. After defining a gesture vocabulary, the user records both gesture templates and test samples by using a motion capture sensor such as Microsoft Kinect or an on-body acceleration sensor.

Figure 1 illustrates the general architecture. Here we see the training process on the left (described in Section 5), where training samples are compared to templates using DTW using different parameters and features to obtain the best settings. The system in operation is shown on the right where the results of comparing continuous gesture input to templates are used to recognize gestures.

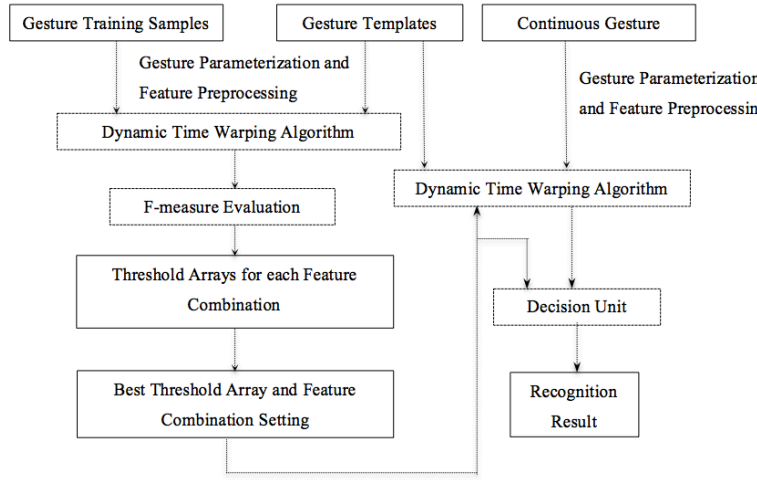

Fig. 1. General architecture of training and recognition processes.

4 Gesture Comparison with Dynamic Time Warping

Since the lengths of the template and real gesture are variable, Dynamic Time Warping (DTW) is one of the best algorithms for defining and computing the distance between these sequences. Moreover, DTW has the advantage of good performance with limited training data. The system stores many prerecorded templates for each gesture and each template is compared to the input data with a separate DTW computation. Multiple templates are expected to cover the variation of a particular gesture pattern. However, the use of multiple templates increases the overall classification time of the system, and some templates could be redundant.

The original DTW algorithm was defined to measure the distance or dissimilarity between two time series T and I based on an Euclidean distance function $dist(i,j)$ that measures the mismatch between any two vectors T_i and I_j, and penalties for

inserting and deleting sequence elements. A cost matrix of size $m \times n$ is filled with values that represent distances between the first *i* vectors of *T* and *j* vectors of *I*. After computing a full cost matrix, the DTW algorithm can do backtracking from the last cell of the cost matrix, *C*[*m*][*n*], to find a least distance path from the end to the beginning. The value of any cell $C(i,j)$ in the cost matrix can be calculated by following the simple rule:

$$C(i,j) = dist(i,j) + \min \begin{cases} C(i-1,j) + insert\ cost \\ C(i-1,j-1) \\ C(i,j-1) + deletion\ cost \end{cases} \quad (1)$$

Unfortunately, in our case we want continuous input *I* to match template *T* even if only the end of *I* is a good match. The standard DTW can be modified to remove penalties for skipping an initial prefix of the input; however, we find it useful to compute spatial coordinates relative to the gesture starting point to achieve position-independent gesture features. Therefore, we use a sliding window method that considers many gesture starting points within the continuous input *I*.

4.1 Dynamic Time Warping with Sliding Window

For each gesture template, the system allocates a queue Q with a capacity of *n* representing instances of DTW matchers. A new matcher is created every *t* input samples (the hop size in Figure 2). Each new matcher replaces the oldest matcher in Q. Each matcher also retains the spatial coordinates of the first input; typically these are subtracted from successive samples to make them position independent before computing a new column of $C(i,j)$.

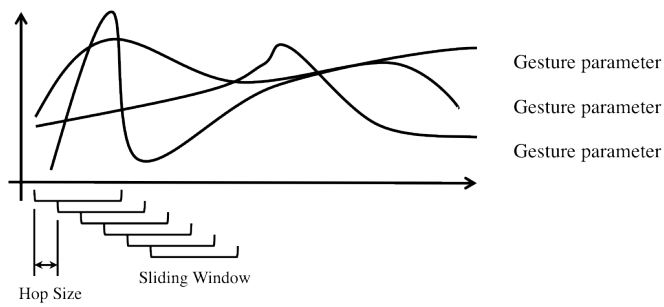

Fig. 2. The system runs a new DTW instance for each template in each window.

The system recognizes a gesture when the DTW-based distance to a template falls below the gesture threshold. The timestamp where the minimum distance occurred will be treated as the end point of this gesture. In addition, the start point of the gesture can be estimated by applying backtracking. Using this information, an output message is generated to report the type of gesture, the start time, and the end time. This message may then trigger some music events according to the detected gesture pattern. If more than one gesture type is matched at the same time, we use a KNN-based resolution algorithm to pick only one.

5 Training

We use the users' personal gesture patterns and train the threshold model by trying each reasonable feature combination to find the best combination. In our implementation, training on examples is used to select among the following parameters and features:

Input Feature. Sensors capture the motion of the users as a vector of N features consisting of values such as joint angles, spatial coordinates, and acceleration. Rather than search every combination (all 2^N), the system searches only a set of user-predefined feature combinations.

Preprocessing Function and Parameters. The system can also apply preprocessing functions to smooth or transform data. In our implementation we normally use an exponential smoothing method, testing four or five different smoothing values.

Insertion and Deletion Costs. For the DTW Algorithm, insertion and deletion costs can have a large effect on performance. To limit the search, insertion and deletion costs are chosen from 4 possible values: 0.0, 0.1, 0.3 and 0.5.

Thresholds. For each gesture, we store n templates, and a gesture is considered recognized when one of its n templates matches the input with a distance less than a per-gesture threshold. For each gesture, we search for the threshold that maximizes the F-measure. Note that once the relatively expensive DTW distances are computed, we can evaluate the F-measure for different thresholds quickly, so the search for the best threshold is a fast operation. In practice, greater thresholds may be desired, e.g. to allow for greater variation in real-time performance.

Our training system performs a grid search on every reasonable combination of Input Feature, Preprocessing Function and Parameters, Insertion and Deletion Costs, and Thresholds to obtain the best result as evaluated by summing the F-measures obtained for all gestures. Moreover, users can customize and weight the feature combinations and adjust F-measure criteria to meet their needs. The F_2-measure, for example, can be used to favor recall over precision, leading to fewer false negatives.

6 Experiments and Results

To evaluate continuous gesture recognition, a human tester is asked to perform eight simple hand gestures (see Figure 3) in front of a Microsoft Kinect sensor. Each of these "command" gestures is repeated five times to create templates. The tester is also asked to perform ten training gestures for each pattern. In order to reject the false detection of user's spontaneous motions, the system also requires the user perform several spontaneous motion samples, such as waving arms and shaking the body. Those non-gesture training samples are labeled with 0. Each sample in the template set and training set has the same approximate duration, which is about 2 seconds or 60 samples. The system estimates the best feature combination and threshold values based on the DTW comparisons between the template and training sets.

To evaluate the system, a tester performed 25 samples lasting 10 to 20 s. Each sample contains 1 to 3 gestures in the vocabulary and those gestures are hand labeled after recording. Figure 4 shows the two-dimensional hand trajectory of one of the test samples, which contains gesture 8 and gesture 1 in the vocabulary. The evaluation is

based on recognition accuracy. A gesture is correctly recognized if the system finds the correct gesture within 6 samples (0.2 s) of the end point of the "true" label. Figure 5 shows how the alignment scores for Gestures 1 and 8 fall below the respective recognition thresholds (dotted lines) when the gestures are made.

When testing 55 gestures in 25 continuous sequences, we find there is a big difference in the recognition accuracy with different feature combination (See Table 1). Generally, the feature combination with the higher F-measure score achieves higher recognition accuracy. The highest recognition accuracy is 89%, where the F-measure score is 7.83/8.0. (8 F-measures are summed, so the maximum score is 8.)

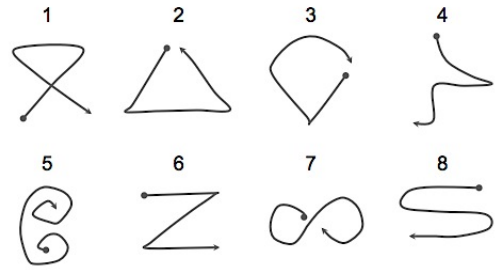

Fig. 3. Gesture vocabulary for testing. The start point of each gesture is labeled as a dot and the end of the gesture is labeled as an arrow.

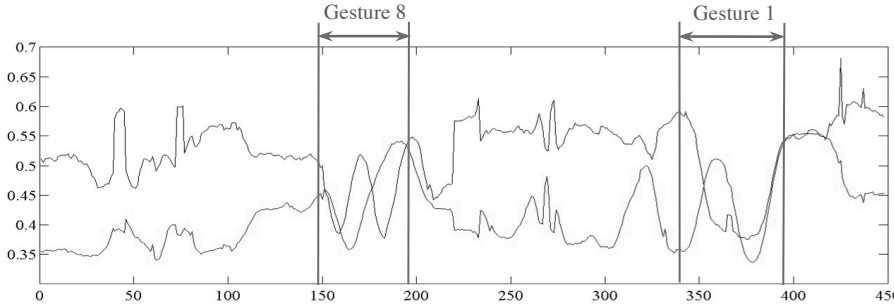

Fig. 4. Two-dimensional trajectory of one test sample, which contains gesture 8 and gesture 1.

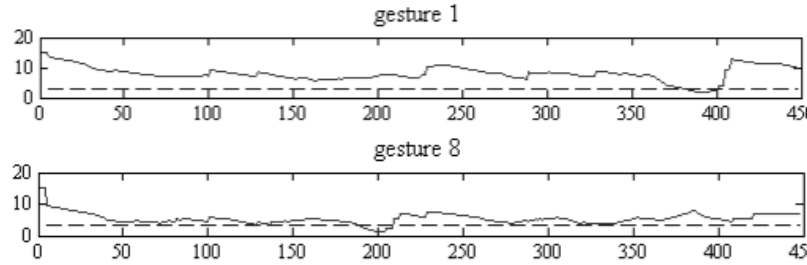

Fig. 5. Solid curves are the best alignment score between each template and each input vector. The dotted lines illustrate the thresholds of each gesture template.

Table 1. Continuous Gesture Recognition Accuracy.

Feature Vector	Relative Position	Insertion cost	Deletion Cost	F-Score Value	Recognition Accuracy
x-,y-axis left hand	False	0.0	0.0	6.37	72%
x-,y-axis left hand	False	0.1	0.3	6.26	70%
x-,y-axis left hand	False	0.0	0.0	6.44	75%
x-,y-axis left hand	False	0.1	0.3	7.63	82%
x-,y-axis elbow & left hand	False	0.1	0.3	6.10	67%
x-,y-axis elbow & left hand	False	0.1	0.4	7.21	80%
x-,y-axis elbow & left hand	True	0.1	0.4	6.67	75%
x-,y-axis left hand	True	0.0	0.1	7.83	89%

Execution time depends on several variables. Assume the system has m gestures and n templates of length of l, we create a DTW comparison every t vectors, and the computation for one DTW matrix cell is c. The total computation is $m \times n \times \frac{l}{t} \times c$. In our implementation, $m = 8$, $n = 5$, $l = 60$ (2 seconds), $t = 4$, $c = 2.6$ μs, and for one DTW cell and the total cost for comparing one input vector with all gesture templates is 1.56 ms on average. This represents less than 5% of a core at a 30 Hz input rate.

7 Conclusion and Future Work

In this paper, a Dynamic Time Warping based gesture spotting and recognition system is built for real-time "command" gestures. The system evaluates different feature combinations to maximize the F-measure for the training data. The system sends recognized commands via OSC messages. It currently supports Microsoft Kinect and an on-body wireless sensor as the input. The system achieves 89% accuracy on continuous gesture recognition. A working implementation has been written in Java. Source code and a demonstration video are available: `http://www.cs.cmu.edu/ ~music/mat/projects.html#gesture-jt`.

However, there is still work to be done in the future to improve the performance of the system. The system still does not get a perfect performance on continuous gesture spotting and recognition. In our testing, some particular non-gesture patterns can easily match one of the gesture patterns and make a false triggering. Good gesture design is critical but further work could explore better distance functions (e.g. using Mahalanobis distance [16] [17]) and sequence matching algorithms (e.g. Continuous Dynamic Programming (CDP), Hidden Markov Models (HMMs) or Conditional Random Fields (CRFs)). Although not currently a problem, various optimizations could be applied to reduce the real-time computational load.

Moreover, further research on optimizing the searching routine for feature combinations should be done to eliminate unnecessary search. An analysis on the individual contribution of each factor in the search space might help us rank the impact of each factor and use a greedy algorithm to reduce search time. Furthermore, we will conduct user studies to evaluate our recognition system in practical contexts and develop systems with more gesture types providing a larger command set.

8 References

1. Francoise, J.: Real Time Segmentation and Recognition of Gestures using Hierarchical Markov Models. Master's Thesis, Universite Pierre et Marie Curie, Ircam (2011)
2. Fiebrink, R.: Real-time Human Interaction with Supervised Learning Algorithms for Music Composition and Performance. Ph.D. dissertation, Faculty of Princeton University (2011)
3. Bevilacqua, F., Schnell, N.: Wireless Sensor Interface and Gesture-follower for Music Pedagogy. In: Proceedings of the 7th International Conference on New Interfaces for Musical Expression, pp. 124–129 (2007)
4. Gillian, N., Knapp, R.: A Machine Learning Tool-box for Musician Computer Interaction. In: Proceedings of the 2011 Conference on New Interfaces for Musical Expression (2011)
5. Takahashi, K., Seki, S., Oka, R.: Spotting Recognition of Human Gestures from Motion Images (in Japanese). Technical Report IE92 134, The Inst. of Electronics, Information, and Comm. Engineers, Japan, pp. 9-16 (1992)
6. Akl, A., Valaee, S.: Accelerometer-based Gesture Recognition via Dynamic-Time Warping, Affinity Propagation, and Compressive Sensing. In: ICASSP, pp. 2270–2273 (2010)
7. Corradini, A.: Dynamic Time Warping for Off-line Recognition of a Small Gesture Vocabulary. In: RATFG-RTS '01: Proceedings of the IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems (RATFG-RTS'01). Washington, DC, USA: IEEE Computer Society (2001)
8. Bautista, M.A., Hernández, A., Ponce, V., Perez-Sala, X., Baró, X., Pujol, O., Angulo, C., Escalera, S.: Probability-based Dynamic Time Warping for Gesture Recognition on RGB-D data. In: Proceedings of the 21st International Conference on Pattern Recognition, International Conference on Pattern Recognition Workshops, WDIA (2012)
9. Kjeldsen, R., Kender, J.: Visual Hand Gesture Recognition for Window System Control. In: Proceedings of Int'l Workshop Automatic Face- and Gesture-Recognition, pp. 184-188, Zurich, Switzerland (1995)
10. Fels, S., Hinton, G.: Glove-talk: A Neural Network Interface between a Data-glove and a Speech Synthesizer. In: IEEE Transactions on Neural Networks, Vol. 4, No. 1, pp. 2–8 (1993)
11. Xu, D.: A Neural Approach for Hand Gesture Recognition in Virtual Reality Driving Training System of SPG. In: International Conf. on Pattern Rec, Vol. 3, pp. 519-522 (2006)
12. Lee, H., Kim, J.: An HMM-Based Threshold Model Approach for Gesture Recognition. In: IEEE Trans. on Pattern Analysis and Mach. Intel, Vol. 21, No. 10, pp. 961-973 (1999)
13. Murph, K.P., Paskin, M.A.: Linear Time Inference in Hierarchical HMMs. In: Advances in Neural Information Processing Systems, Vol.2 (2001)
14. Elmezain, M., Al-Hamadi, A, Michaelis, B.: Robust Methods for Hand Gesture Spotting and Recognition Using Hidden Markov Models and Conditional Random Fields. In: Signal Processing and Information Technology (ISSPIT), 2010 IEEE Int. Symposium. (2010)
15. Krishnan, N.C., Lade, Pr., Panchanathan, S.: Activity Gesture Spotting Using a Threshold Model Based On Adaptive Boosting. In: Multimedia and Expo (ICME), 2010 IEEE International Conference, Vol.1, pp. 155-160 (2010)
16. Alon, J., Athitsos, V., Yuan, Q., Sclaroff, S.: A Unified Framework for Gesture Recognition and Spatiotemporal Gesture Segmentation. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 31, pp. 1685–1699 (2008)
17. Liu, J., Wang, Z., Zhong, L., Wickramasuriya, J., Vasudevan, V.: uwave: Accelerometer-based Personalized Gesture Recognition and Its Applications. In: IEEE International Conference on Pervasive Computing and Communications, pp. 1–9 (2009)
18. Gillian N., Knapp R B., O'Modhrain S.: Recognition Of Multivariate Temporal Musical Gestures Using N-Dimensional Dynamic Time Warping. In: Proceedings of the 11th International Conference on New Interfaces for Musical Expression pp. 337-342 (2011)