2014 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS 2014)
September 14-18, 2014, Chicago, IL, USA

A Gesture Recognition System for Mobile Robots that Learns Online

Alan J. Hamlet[1], Patrick Emami[2], and Carl D. Crane[3]

Abstract— **This paper presents a gesture recognition framework designed for controlling mobile robots using dynamic hand gestures. The system is unique in that it can be trained to recognize a new gesture by simply demonstrating the gesture once. From then on, the system improves its recognition accuracy over time by learning from experience. The classification scheme utilizes a nearest neighbor approach with a dynamic time warping distance metric. The approach robustly handles both temporal and spatial variations in executions of highly dynamic and complex gestures. One product of the research is an open source software package that will work with little to no modifications on robots utilizing a 3D depth sensor and the Robot Operating System (ROS). Experimental results show the ability of novice users to train and utilize the system effectively, as well as the system's ability to increase its recognition accuracy over time by learning from experience.**

I. INTRODUCTION

The recent advances in hardware and software for mobile robots are enabling robots to become useful and, more importantly, practical in home and work environments. As these robots' behavioral capabilities increase, so does the need for an easy and natural way for humans to interact with them. This is especially true for non-technical users who are hesitant to adopt such technologies due to the difficulty of using robots. While speech recognition technology is improving [1], it still has trouble in environments with background noise. Additionally, robots may be operating in envronments not conducive to speech recognition such as loud construction sites and offices, or places where it is inappropriate for the operator to enunciate and project their voice properly such as libraries, movie theaters, or even stealth military operations. Gesture recognition can be an alternative or complementary form of human robot interaction (HRI) in these situations in order to increase classification accuracy.

Depth cameras such as the Microsoft Kinect, the Asus Xtion, and others made by Primesense are becoming increasingly affordable and popular on robotics platforms. Some commercially available robots currently utilizing such depth sensors include the Willow Garage PR2 robot [2], the recently introduced UBR-1 [3], and the Turtlebot and Turtlebot 2 [4] robots. These depth cameras can utilize the open source OpenNI middleware packages to allow for skeleton tracking applications. The work presented in this paper builds on such applications to create an open source software package that lets the user train robots to robustly recognize dynamic gestures of the user's choosing in an easy and intuitive way.

Many methods have been proposed for dynamic gesture recognition. Some research, such as [5] and [6], implement sensors that the user must physically interact with. While these methods are quite accurate, they are expensive and impractical for use on a mobile robot platform. Gesture recognition using 'visual' methods, either depth or video cameras, has been the subject of much of the recent research in the literature. In [7], the authors experiment with both a template matching technique and a neural network approach to classify arm poses from camera images. Then the authors utilize the Viterbi algorithm to align input sequences with the template gestures. While the results were promising, the study tested simple psuedo-static gestures and did not demonstrate the ability of non-technical persons to utilize the system. Other studies have implemented hidden Markov models [8][9][10][11] or neural network techniques [12][13]. Hidden Markov models utilize an offline training procedure which uses a form of expectation maximization to determine the model parameters. This training procedure requires a relatively large number of example sequences in order to approximate the model parameters sufficiently. Similarly, typical neural networks used for gesture recognition use backpropagation or some other form of supervised learning where the network weights are learned offline from a large collection of example sequences. Similarly, [14] trains a support vector machine (SVM) classifier on a set of camera images in order to recognize hand signals from Kinect RGB data on a care-giver robot. Since the most important aspect of a human-robot interface is the ability of users to effectively utilize the interface and communicate with the robot, there is a need for a robust and easy to use gesture recognition system. This study introduces a gesture recognition system that a user can train by simply demonstraing a gesture one time, eliminating the need for an offline training phase.

The remainder of this paper is laid out as follows. First, in section II, the need for a nonlinear distance metric to compare time series is motivated. Then, the dynamic time warping algorithm is introduced and the constraints for the algorithm used in this study are presented. Next, section III discusses the mechanism for determining the classification thresholds and the method for learning from experience. Then the experimental set up and results are presented in section IV. Finally, the conclusion in section V summarizes the results and contributions of the work presented.

[1]Alan J. Hamlet is a PhD candidate in the Mechanical Engineering Department at the University of Florida, Gainesville, FL, 32601, USA. Correspondence should be addressed to AJHamlet@ufl.edu

[2]Patrick Emami is a student in the Electrical and Computer Engineering Department at the University of Florida, Gainesville, FL, 32601, USA PatrickEmami@gmail.com

[3]Carl D. Crane is Faculty in the Mechanical Engineering Department at the University of Florida, Gainesville, FL, 32601, USA Carl.Crane@gmail.com

II. TIME SERIES CLASSIFICATION

Gesture recognition is an example of a time series classification problem. Typically, data (position, speed, orientation) about the users hand or body is sampled in discrete time steps, resulting in a discrete time series. Classification is then performed to see which, if any, known gesture the input sequence matches. When looking for patterns in time series data, often the speed of the pattern is of no consequence while the shape of the data is the salient feature. Therefore, a notion of distance must be used to measure the similarity of two time series that does not punish differences in the timing or speed of the data. An intuitive way to measure the similarity between two temporal sequences is to sum over the distances between the two series at each time step. Even if the shape of the data is the same, this method will result in large distances if the timing is off. For example, a plot of the time series data acquired by sampling the equations

$$y = sin(t) + sin(2t) \tag{1}$$

$$y = sin(0.75t) + sin(1.5t) \tag{2}$$

at a frequency of 10 Hz for 4.8 seconds and 6.4 seconds, respectively, is shown in Fig. 1. The blue and red data sets have the same shape but the former is 0.75 times slower than the latter. Using a pairwise Euclidean distance metric the distance between the two time series in Fig. 1 is 24.54 while it is only 2.00 when using a distance metric based on the dynamic time warping algorithm.

A. Dynamic Time Warping

One method for compensating for nonlinear variations in time is the dynamic time warping algorithm. Dynamic time warping (DTW) is a dynamic programming algorithm that finds the optimal alignment of two temporal sequences that minimizes an assigned cost. Instead of comparing points in the two series pairwise in time, the algorithm allows for a certain amount of 'warping' in the time dimension so points are matched with nearby points that will minimize the distance between them without violating local and global constraints. DTW can also directly compare sequences of different lengths without first requiring interpolation of the shorter sequence. The remainder of this section will provide a general overview of the DTW algorithm. For a more formal explaination of the algorithm, please see [15].

To find the DTW distance between two sequences $X := (x_1, x_2, ..., x_N)$ of length $N \in \mathbb{N}$ and $Y := (y_1, y_2, ..., y_M)$ of length $M \in \mathbb{N}$, the DTW algorithm finds the lowest cost path through the $N \times M$ cost matrix. The *(n,m)* element of the cost matrix represents the local distance, or cost, between the n^{th} element in X and m^{th} element in Y. Euclidean distance was chosen for this local cost since the quantity of interest is the similarity between points in 3 dimensional space.

B. Local and Global Constraints

The path through the cost matrix must satisfy three conditions: a boundary condition, the monotonicity condition, and the step size condition. Let the warping path be denoted as

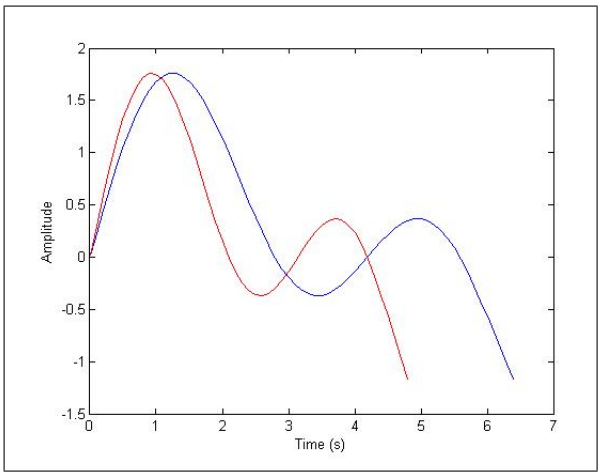

Fig. 1. Two time series with the same shape but differing frequency; the red wave has a speed 1.33 times that of the blue wave. The Euclidean distance between the two time series is 24.54 while the dynamic time warping distance is only 2.00.

the sequence $p = (p_1, p_2, ..., p_L)$ with $p_l = (n_l, m_l) \in [1 : N] \times [1 : M]$ for $l \in [1 : L]$. The three conditions are as follows:

1) Boundary condition: $p_1 = (1, 1)$ and $p_L = (N, M)$

2) Monotonicity condition: $n_1 \leq n_2 \leq ... \leq n_L$ and $m_1 \leq m_2 \leq ... \leq m_L$

3) Step size condition: $p_{l+1} - p_l \in [(1, 0), (0, 1), (1, 1)]$ for $l \in [1 : L - 1]$

It should be noted that the step size condition implies the monotonicity condition. A warping path defines an alignment between the sequences X and Y by assigning the x_{n_l} element of X to the element y_{m_l} of Y. The boundary condition requires that the first and last elements of the two sequences are aligned with each other, the monotonicity condition ensures that the alignment goes forward in time (sequence elements cannot be reversed) and the step size condition ensures continuity so no element in either of the sequences is omitted.

Applying global constraints to the DTW algorithm not only decreases computation time but also improves the algorithm's accuracy for real world problems [16]. Without global constraints, DTW can find unrealistic warping paths that result in small distances between dissimilar sequences. The common global constraint known as the Sakoe-Chiba band was found to increase the accuracy of the gesture recognition system and reduce the number of false positive classifications [17]. The Sakoe-Chiba band runs along the main diagonal of the cost matrix with fixed horizontal and vertical width $R \in \mathbb{N}$. Fig. 2 demonstrates a path through the cost matrix which does not violate the boundary, monotonicity, or step size conditions and also stays within the Sakoe-Chiba band. The global constraint region imposed by the Sakoe-Chiba band is shown in Fig. 2 as the two straight diagonal lines above and below the path through the matrix. The details of the procedure for determining the path that minimizes cost will not be presented here but by using dynamic programming it can be done in $O(NM)$ time

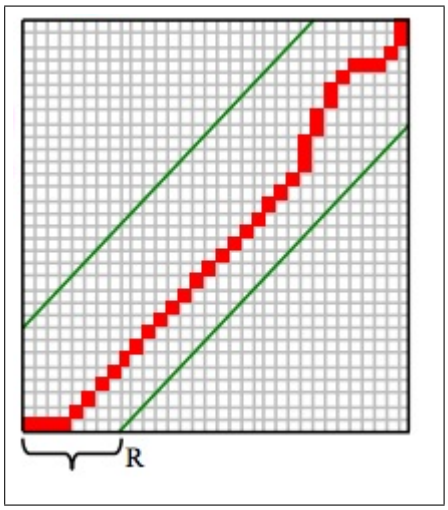

Fig. 2. A DTW path that satisfies the monotonicity condition, the boundary condition, and the step size condition. The diagonal lines above and below the path represent the edges of the Sakoe-Chiba band global constraint with a horizontal and vertical width, R.

[15]. For recognizing gesture sequences in the current work of length $N \leq M \leq 100$ this computation time is trivial. This gesture length was chosen since the data collection is performed at a frequency of 10 Hz and the maximum acceptable gesture length was set to be 10 seconds.

III. Experiential Learning

The DTW algorithm allows for a one-to-one comparison of an input sequence to a template sequence. If the gesture recognition system had a way to generate appropriate distance thresholds for gestures, it could begin classifying input gesture sequences as soon as it had a single labeled sequence. If the distance between an input sequence and a template sequence is below the threshold, the input sequence is recognized as the gesture corresponding to that template sequence. Determining a recognition threshold that prevents false positive classifications while minimizing false negatives is paramount.

As not to limit the variety of gestures available to the end users, the gesture recognition system allows for gestures to be from anywhere between 1 and 10 seconds long. During experimental testing of the system, it was found that gestures with more movement tended to have higher intra-class distances (distances between sequences generated from the same gesture). This variability suggestes that the thresholds for these gestures should be higher. By plotting the average intra-class distances for several gestures having varying amounts of movement, a linear correlation was found. It should be noted that the duration of the gesture is not correlated to the average intra-class distance since a long static gesture requiring little movement will be very repeatable and have a small intra-class distance. Upon examing the correlations between the amount of movement in a behavior and the intra- and inter-class distances for that behavior, a linear relation was selected to calculate the threshold distance for gestures as a function of the distance the hands move during the gesture. The threshold for the i^{th} gesture, $threshold_i$, is given by (3). The amount of movement, $movement_i$, in the i^{th} gesture sequence, X, is calulated by (4).

$$threshold_i = 0.47 \times movement_i + 11 \tag{3}$$

$$movement_i = \Sigma_{n=2}^{N} \Sigma_{j=1}^{D} |X_n^j - X_{n-1}^j| \tag{4}$$

The sequence X contains $N \in \mathbb{N}$ data points each of dimension $D \in \mathbb{N}$. Equation 4 sums over each data point as well as each dimension of the points to return a scalar quantity. In the current work, each data point is a 6 dimensional vector corresponding to the position of each of the user's hands (with respect to their torso) in 3D space.

The threshold equation favors false negative classifications over false positive ones. This is so the robot does not perform actions the user did not want it to; more importantly, this allows the true positive classifications to be used to enhance future performance. When a sequence is below the threshold, the system will be confident that the sequence was in fact generated by the corresponding gesture and the input sequence can be given the same label as the template sequence. Then, the newly labeled sequence can be used as another example of the gesture to make classification more robust. Using a 1 nearest neighbor approach, the gesture recognition system compares an input sequence to every labeled sequence it has stored in memory. When the system is first trained on a gesture it will only have one example of that gesture. Every time a gesture is recognized the system will accumulate another positive example of that gesture. Using this method, the system learns to generalize more effectively through experience.

The nearest neighbor algorithm utilizes a normalized cost, J, given by the following equation:

$$J_i = dist_i / threshold_i \tag{5}$$

where J_i is the cost of the input sequence with respect to the i^{th} template sequence, $dist_i$ is the DTW distance between the input sequence and the i^{th} template sequence, and $threshold_i$ is the threshold for the i^{th} template sequence as given by (3). The algorithm returns the gesture with the lowest cost. If the cost of that gesture is less than one, then the DTW distance is below the threshold and the robot executes the corresponding behavior.

IV. Implementation

The gesture recognition system was implemented on a TurtleBot, which has an iRobot Create base, a Kinect sensor, and a netbook laptop running all the software. The base is differential drive, has encoders for wheel odometry, and has an inertial measurement unit for dead reckoning. The hardware setup is shown in Fig. 3. The robot's software utilizes the Robot Operating System (ROS) with the human tracking being performed by a modified version of the open source package `openni_tracker`, which tracks users with the Xbox Kinect depth camera data. While the software tracks 15 different joints on the human body, the implementation

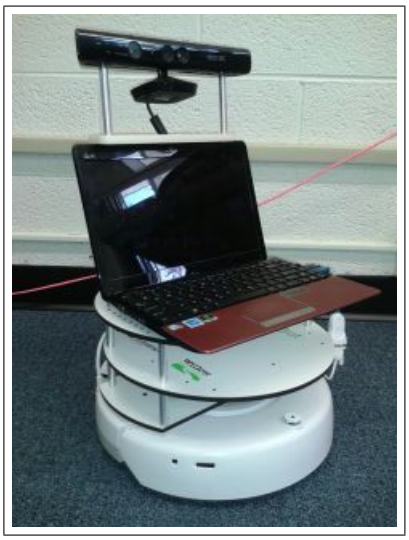

Fig. 3. The Turtlebot platform on which the gesture recognition system was tested. The platform is composed of a differential drive base, netbook laptop, and Xbox Kinect sensor.

of the gesture recognition system described here only looks at the position of the users hands with respect to their torso. The software could easily be modified to incorporate other joints for whole body gestures. The position of the user's torso is subracted from the position of their hands so that the gestures are independent of the user's distance from the robot. In practice, gestures are recognized no matter where the user is with respect to the robot as long as they are in its field of view. When recording a gesture sequence, the system calculates the relative X,Y, and Z positions of each of the user's hands at a rate of 10 Hz.

The system does not have any hard coded dynamic gesture models, but it has 3 static poses that it recognizes. When the user wants to train the robot to learn a new gesture, the user simply assumes the 'T-pose'; they hold their arms straight out to their sides at shoulder height. The robot will then record the user's dynamic gesture and ask the user which of the built-in behaviors the user wants it to execute when the gesture is seen. This is the only time the system requires any physical contact between the user and the robot. The user simply enters the number of the desired gesture as shown in a displayed list. When the user wants to give the robot a command, they intuitively raise their left hand to get the robot's attention and then perform the dynamic gesture. The gesture recognition system recognizes that gestures are finished when the user is in the 'idle pose'. The 'idle pose' is recognized when the user's hands are down at their sides for approximately one second.

V. EXPERIMENTS AND RESULTS

A. Experimental Setup

In order to measure the system's overall performance and ability to be utilized by novice users, experiments were conducted with 10 volunteers who were initially unfamiliar with the system. After a brief (5-10 minute) training period, the subjects were instructed to train the robot so that the robot would perform four different behaviors when it saw four corresponding gestures. The behaviors were: move-to-the-right, move-to-the-left, turn-back-and-forth, and rotate-in-a-circle. These simple behaviors were chosen to demonstrate the ability of the gesture recognition system to work on a moving robot platform. More complex behaviors such as searching an area or fetching the user a soda could easily be incorporated into the gesture recognition system, given a more capable hardware platform. The gesture corresponding to the move-to-the-right behavior was to move the right hand to the side with the palm facing up, then move it toward the centerline rotating the palm downward, and then back to the side with the palm facing up again. Similarly, the robot was to be trained to move to the left if the same gesture was done with the left hand. The third gesture was to rotate both forearms upward at the elbows, bringing the palms up toward the face two consecutive times. This gesture corresponded to the turn-back-and-forth behavior. Finally, the last gesture was to take the right hand and make a circle in the air, which corresponded to the rotate-in-a-circle behavior. After the volunteer trained the robot, they commanded the robot to perform the four behaviors by using the dynamic hand gestures four times each for a total of 16 commands.

B. Results

The classification results, expressed in ratios, are shown in Table 1. The overall accuracy of the system on the 160 tested commands given by the 10 novice volunteers was 92.5%. As designed, no commands resulted in a false positive classification (a gesture being recognized when that gesture was not actually performed). This is crucial for the system since it learns from experience by using all positively recognized sequences to determine future gesture labels. Even with the strict threshold to eliminate false positive classifications, the system only had 7.5% false negative classifications (not recognizing any gesture when one was actually performed). All of these false negative classifications occured with three of the ten volunteers. Most of the misclassifications with these three individuals came from repeated failures on the same gesture. With one volunteer, the system did not recognize gesture one (move to the right) on any of the four attempts. This is believed to be caused by a tracking error during the training phase for that gesture. On another volunteer, the system misclassified the first three attempts of the 'rotate' gesture. This was caused by the user performing a much wider circle than what he trained the system on originally for the first three attempts and then recognizing his error on the last attempt to correctly communicate the command to the robot. True negative classifications are not listed in Table 1 since they were not explicitly tested in the experiments.

TABLE I
CLASSIFICATION RESULTS

	TRUE	FALSE
POSITIVE	0.925	0
NEGATIVE	N/A	0.075

For all four gestures, the average cost of the closest gesture (of the same class) decreased as the system gained more training examples. As can be seen in Fig. 4, the costs for gesture 2 and gesture 3 decreased with every additional example. All four decreased significantly once a second example was obtained and seemed to begin leveling out at around three or four examples. This suggests it may be optimal to store a total of four or five sequences for each gesture, thereby increasing recognition accuracy without taking up superfluous memory.

Table 2 shows the average intra- and inter-class distances for all gesture combinations. The within class distances for all four gestures were well below the threshold of 1.0 while all of the between class distances were over the threshold. This means that even if the system was not trained on a gesture, it would not have misclassified the gesture as one of the other gestures if it was executed. The only gesture that was ever incorrectly below the threshold on any attempt, was gesture 4. Since both gesture 1 and gesture 4 involved only motion of the right hand, they tended to have closer distances. Gesture 1 involved less motion, though, so it had a lower threshold and was not mistaken for gesture 4. Gesture 4 had more motion and therefore a higher threshold, so when the shorter gesture was actually performed, the cost of gesture 4 was sometimes lower than 1.0, especially if gesture 1 was executed at the same height as gesture 4. As shown in Table 2, though, the average distance between gestures 1 and 4 was above 1.0 and the wrong gesture was never recognized.

C. Further Discussion

As with any human robot interaction interface, the human psychology and attitude had a large impact on the efficacy of the gesture recognition system. It was clear that some of the novice users were uncomfortable gesturing to a robot and/or nervous in the testing environment, and therefore moved in very forced and unnatural ways. These forced, unnatural movements were difficult for the user to repeat accurately, so the gesture recognition system had a more difficult time recognizing them. On the other hand, individuals who seemed comfortable in the testing environment and with the robot tended to move fluidly and more naturally.

Fig. 4. The average cost of the closest intra-class gesture versus the number of available examples of the gesture.

TABLE II
AVERAGE INTER- AND INTRA- CLASS DISTANCES

		Gesture Tested			
		1	2	3	4
Gesture Executed	1	0.48	1.31	1.17	1.04
	2	1.34	0.47	1.27	1.73
	3	1.41	1.48	0.61	1.32
	4	1.20	1.75	1.28	0.58

This resulted in the system achieving better performance. In application, the user would pressumably be in a more comfortable environment than an unfamiliar lab and would become more comfortable with the interface with repeated use. Either way, an important conclusion of the experiment is that human comfort is critical to acceptance of service robots and to an effective HRI method.

While the system performed well in the experiments, it should be noted that the system had difficulties when several humans were in the field of view of the robot. Additionally, the robot tended to lose the user when performing the behaviors and would require a few seconds to recalibrate to the user. This is because the human tracking software is not optimized for a moving sensor and further extensions of the research could include 1) tracking and recognizing gestures of multiple individuals simultaneously and 2) efficient tracking while the robot is moving.

VI. CONCLUSION

This paper presented a gesture recognition framework that is novel for three main reasons. First, a non-technical user can program the sytem to recognize a new dynamic gesture by simply demonstrating the gesture a single time. Second, the system uses a dynamically calculated threshold to determine class labels of input gestures. Finally, the system improves its recognition accuracy over time by accumulating and labeling example gesture sequences. Experiments with 10 novice users training and utilizing the gesture recognition system resulted in a recognition accuracy of 93%, with perfect classification results on 7 of the 10 users. The experimental results demonstrate the ability of non-technical users to efficiently utilize the system without extensive training. The results also show an increase in the system's ability to correctly classify gestures with increased experience. An open source software package was developed that can be utilized by robots with depth cameras that are operating with the Robot Operating System (ROS). Individuals interested in the software should contact the authors.

REFERENCES

[1] J. Fasola, M. Mataric, "Using semantic fields to model dynamic spatial relations in a robot architecture for natural language instruction of service robots." *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on,* IEEE, 2013. pp. 143-150.

[2] Garage, Willow. "Personal robot 2 (pr2)." Online: www.willowgarage.com (2013).

[3] Robotics, Unbounded. "UBR-1." Online: www.http://unboundedrobotics.com (2014).

[4] Garage, Willow. "Turtlebot 2." Online: www.willowgarage.com/turtlebot (2013).

[5] F. Camastra and D. De Felice, "LVQ-Based Hand Gesture Recognition Using a Data Glove," *Neural Nets and Surroundings.* pp. 159-168, 2013.

[6] S. Rajko, et al. "Real-time gesture recognition with minimal training requirements and on-line learning." Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on. IEEE, 2007.

[7] S. Waldherr, R. Romero, and S. Thrun, "A gesture based interface for human robot interaction," *Autonomous Robots 9*, no. 2. pp. 151-173, 2000.

[8] S. Shinji and T. Kitamura, "Subunit modeling for Japanese sign language recognition based on phonetically dependent multi-stream hidden Markov models," *Universal Access in Human-Computer Interaction. Design Methods, Tools, and Interaction Techniques for eInclusion*, pp. 548-555. Springer Berlin-Heidleberg, 2013.

[9] P. Haibo and Y. Ding, "Dynamic Hand Gesture Recognition Using Kinematic Features Based on Hidden Markov Model," *Proceedings of the 2nd International Conference on Green Communications and Networks 2012*, Springer Berlin-Heidleberg, 2013.

[10] H. Park et. al. "HMM-based gesture recognition for robot control." *Pattern recognition and image analysis.* Springer Berlin Heidelberg, 2005. pp. 607-614.

[11] A. Gaschler et al. "Social behavior recognition using body posture and head pose for human-robot interaction." *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on.* IEEE, 2012.

[12] P. Haibo and Y. Ding, "Dynamic Hand Gesture Recognition Using Kinematic Features Based on Hidden Markov Model," *Proceedings of the 2nd International Conference on Green Communications and Networks 2012*, Springer Berlin-Heidleberg, 2013.

[13] C. Chen et al. "Dynamic Gesture Recognition Based on Fuzzy Neural Network Classifier," *ACHI 2013, The 6th International Conference on Advances in Computer-Human Interactions*, 2013.

[14] T. Tabata, Y. Kobayashi, and Y. Kuno, "Recognition of Request through Hand Gesture for Mobile Care Robots", *Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE*, pp. 8312-8316, 2013.

[15] M. Muller, Information Retreival for Music and Motion. Berlin: Springer-Verlag, 2007, ch. 4.

[16] G. Tomasi, F. van den Berg, and C. Andersson, "Correlation Optimized Warping and DTW as Preprocessing Methods for Chromatographic Data," *Journal of Chemometrics*, 2004.

[17] H. Sakoe and S. Chiba. "Dynamic programming algorithm optimization for spoken word recognition." *Acoustics, Speech and Signal Processing*, IEEE Transactions on 26.1 (1978): 43-49.