Discriminant Boosted Dynamic Time Warping and Its Application to Gesture Recognition

Tarik Arici[1], Sait Celebi[2], Ali Selman Aydin[1] and Talha Tarik Temiz[1]
[1]*Department of Electrical Engineering, Istanbul Sehir University, Istanbul, Turkey*
[2]*Graduate School of Natural and Applied Sciences, Istanbul Sehir University, Istanbul, Turkey*
tarikarici@sehir.edu.tr, {saitcelebi,aliaydin,talhatemiz}@std.sehir.edu.tr

Keywords: Dynamic Time Warping, Linear Discriminant Analysis, Gesture Recognition, Kinect

Abstract: Dynamic time warping (DTW) measures similarity between two data sequences by minimizing an accumulated distance between two sequence samples at each iteration and a cost is computed to assess the level of the similarity. The DTW cost may then be used to assign a sequence to a class if the problem is a classification problem. In machine learning, classification problems are solved using features with good discrimination power, which are generated by exploiting the distribution of data vectors. Linear Discriminant Analysis (LDA) is such a technique and finds discriminative projection directions which are used to generate features as projections of sequence vectors on to these directions. Unfortunately, these techniques are not applicable to warped sequences because the mapping between the test sequences and the training sequences is not known. To solve this problem, we propose a constrained LDA framework that produces direction vectors that repeat unit vectors that have dimensions equal to the dimensions of a single sequence sample. Such projection vectors can be used without knowing the mapping of test sequence vectors to training sequence vectors. Experiment results show that generating features by discriminant analysis improves the performance significantly.

1 INTRODUCTION

Dynamic time warping (DTW) measures similarity between two data sequences which might be obtained by sampling a source with varying sampling rates or by recording the same phenomenon occurring with varying speeds (Wikipedia, 2012). For example, DTW is used in speech recognition to warp speech in time to be able to cope with different speaking speeds (Amin and Mahmood, 2008; Myers and Rabiner, 1981; Sakoe and Chiba, 1978). DTW is also used in data mining and information retrieval to deal with time-dependent data (Rath and Manmatha, 2003; Adams et al., 2004). Another application of DTW is in gesture recognition to time-warp an observed motion sequence of human body joints to pre-stored gesture sequences (Rekha et al., 2011; Wenjun et al., 2010; Kuzmanic and Zanchi, 2007; Corradini, 2001).

A classifier based on dynamic time warping reduces to a nearest-neighbor (NN) classifier based on a (Euclidean) distance performed directly on the sequence samples if the two sequences were aligned in time. However, sequences are not always time-aligned and DTW time-warps sequences to achieve a satisfactory alignment and paves the way for a simple NN classifier. NN classifier is probably the simplest classifier in machine learning. It is a type of instance-based classification which only uses local information and classifies based on closest training examples. One way to improve the classification performance is to generate *good* features, which are learnt from the training set. Good features can be generated by first finding directions along which the classes are best separated in some way and then projecting data points to these directions. For example, linear discrimination or linear discriminant analysis (LDA) seeks directions in which the classes are separated in a way that maximizes the class separation while minimizing the in-class variation using training data set. Such directions are vectors that have the same dimensionality as the data point vectors obtained by vectorizing the data sequences. The features are projections of data points on to these vectors. Hence, all the sequence samples each of which constitute a dimension or a block of dimensions of a data point vector is used to compute the projection. However, feature generation based on these projections will not work in the case of misaligned sequences since the index will likely be different than the corresponding index of a specific dimension in the training phase. In other words, the mapping of the indices of training sequences to test sequences is not known, and paradoxically this was the reason while DTW is needed at the beginning. If it were known, feature generation and/or dimensionality reduction techniques from machine learning could have been applied directly. DTW aims to find this mapping but can not use machine learning tools because of the aforementioned reason and hence boils down to a per time-sample based similarity between two time sequences. In other words, it does not utilize the distribution of data in the feasible space and therefore the DTW cost is not as discriminative as it would have been.

We propose a method to utilize tools from machine learning such as feature generation (dimensionality reduction) in dynamic time warping. Specifically, we constrain LDA to generate discriminative features learnt from a training set, which can still be applied to sequences warped in time. Solving a constrained LDA problem on a training set with time-aligned sequences, we obtain discriminative vectors to be projected on to. The constraint on these vectors enforce them to be independent of the unknown time-warping by making such vectors to be repetitions of a smaller unit vector with dimensionality required by a single sample of the time sequence. Since a discriminative vector constitutes repetitions of this unit vector corresponding to a single sample, DTW can perform its recursive cost-computation without requiring to know which time sample it is operating on. Although the unconstrained (conventional) LDA would have led to more discriminative projection directions, which fully take advantage of the data distribution, our constrained LDA will still lead to projection directions with discriminative power better than an NN classifier using DTW directly on sequence samples. Moreover, constrained LDA features can also be used on time-warped sequences.

Gesture recognition is an example application that can utilize DTW. Gesture recognition has wide-ranging applications such as navigating in virtual environments, controlling devices in a smart environment, interacting with computer games via player's gestures, recognizing sign language, etc (Bleiweiss et al., 2010; Raptis et al., 2011; Mitra and Acharya, 2007; Keskin et al., 2011).

Microsoft launched Kinect in 2010, which is a depth sensor and subsequently released Kinect Software Development Kit (SDK) in 2011, which includes machine learning algorithms such as random forests to accurately predict 3D positions of body joints from a single depth image (Shotton et al., 2011). Although Kinect is developed for Xbox 360 video game console to enable natural interaction between users and video games, Kinect's robustness in 3D human joint position prediction and its affordability for consumers have generated numerous research projects and applications in human-computer interfaces (Bleiweiss et al., 2010; Raptis et al., 2011). Kinect enables gesture recognition applications such as interactive gaming, which performs gesture recognition followed by gesture animation (Reyes et al., 2011), touch detection using depth data (Wilson, 2010), human pose estimation (Jain et al., 2011), implementation of real-time virtual fixtures (Ryden et al., 2011), real-time robotics control applications (Stowers et al., 2011) and the physical rehabilitation of young adults with motor disabilities (Chang et al., 2011). We apply our proposed method to recognize gestures.

2 BACKGROUND

2.1 Related Work

One commonly used technique for gesture recognition is using HMMs for modeling gesture sequences. HMMs are especially known for their application to speech recognition, gesture recognition, bioinformatics, etc. HMMs are statistical models for sequential data (Baum et al., 1970; Baum, 1972), and therefore can be used in gesture recognition (Gehrig et al., 2009; Starner and Pentland, 1996; Lee and Kim, 1999). The states of an HMM are hidden and state transition probabilities are to be learned from the training data. However, defining states for gestures is not an easy task since gestures can be formed by a complex interaction of different joints. Also, learning the model parameters *i.e.*, transition probabilities, requires large training sets, which may not always be available. On the other hand, DTW does not require training but needs good reference sequences to align with.

Using a weighting scheme in DTW cost computation has been proposed for gesture recognition (Reyes et al., 2011) (Celebi et al., 2013), which does not generate features as proposed in this paper but finds weights for the Euclidean distance computation. The method proposed in (Reyes et al., 2011) uses DTW costs to compute between and within class variations to find a weight for each body joint to be used in distance calculation. The distances between joint positions in different time instances are weighted according to these weights. However, our proposed method iteratively computes the overall distance between two sequences and finds a constrained projection direction. In our method, a projection operation is made on to pre-computed discriminative directions for each sequence sample. Hence, we compute a linear combination of joint positions, which creates features that are spatial distances between x and y positions of the joints at the same time instant. The proposed method creates stronger features because features that are spatial distances between joint positions have potentially stronger discriminative power while describing a gesture with respect to another gestures. For example, a *zoom-out* gesture may be better described using the distance between the x positions of hand positions compared to a *swipe-left* gesture. To our knowledge, there is no prior work in the literature that proposes using feature generation techniques from machine learning for dynamic time warping.

2.2 DYNAMIC TIME WARPING

DTW is a template matching algorithm to find the best match for a test pattern out of the reference patterns, where the patterns are represented as a time sequence of measurements or features obtained from measurements.

Let $\mathbf{r}(i),\ i = 1,2,\ldots,I$, and $\mathbf{t}(j),\ j = 1,2,\ldots,J$ be reference and test vector sequences, respectively. The objective is to align the two sequences in time via a nonlinear mapping (warping). Such a warping is an ordered set of tuples as given below

$$(i_0, j_0),(i_1, j_1),\ldots,(i_f, j_f),$$

where tuple (i, j) denotes mapping of $\mathbf{r}(i)$ to $\mathbf{t}(j)$, and $f+1$ is the number of mappings. The total cost D of a mapping between $\mathbf{r}$ and $\mathbf{t}$ with respect to a *distance* function $d(i, j)$, is defined as the sum of all distances between the mapped sequence elements

$$D(\mathbf{r},\mathbf{t}) = \sum_{k=0}^{f} d(i_k, j_k), \tag{1}$$

where $d(i, j)$ measures the distance between elements $\mathbf{r}(i)$ and $\mathbf{t}(j)$.

A mapping can also be viewed as a path on a two-dimensional (2D) grid of size $I \times J$, where grid node (i, j) denotes a correspondence between $\mathbf{r}(i)$ and $\mathbf{t}(j)$. Each path on the 2D grid is associated with a total cost D given in (1). If the path is a *complete* path defined by

$$(i_0, j_0) = (0,0),\ (i_f, j_f) = (I,J), \tag{2}$$

then a complete path aligns the entire sequences $\mathbf{r}$ and $\mathbf{t}$.

The minimum cost path on the 2D grid is the optimal alignment between two sequences. One way to find the minimum cost path is to test every possible path from the left-bottom corner to right-top corner. However, this has exponential computational complexity. Dynamic programming reduces the complexity by taking advantage of Bellman's principle. Bellman's optimality principle states that the optimal path from the starting grid node (i_0, j_0) to the ending node (i_f, j_f) through an intermediate point (i, j) can be expressed as the concatenation of the optimal path from (i_0, j_0) to (i, j), and the optimal path from (i, j) to (i_f, j_f). This implies that if we are given the optimal path from (i_0, j_0) to (i, j), we only need to search for the optimal path from (i, j) to (i_f, j_f) rather than searching for paths from (i_0, j_0) to (i_f, j_f).

Let's use Bellman's principle in total cost computation. If we denote the minimum total cost at node (i_k, j_k) by $D_{min}(i_k, j_k)$, then by Bellman's principle $D_{min}(i_k, j_k)$ can be computed by using the costs of the predecessor nodes, *i.e.* the set of i_{k-1}, j_{k-1}s, as below

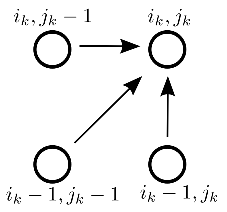
$$D_{min}(i_k, j_k) = \min_{i_{k-1},j_{k-1}} D_{min}(i_{k-1}, j_{k-1}) + d(i_k, j_k). \tag{3}$$

The distance between two sequence elements $d(i_k, j_k)$ can also be designed to depend on the transition between the nodes and can be denoted as $d(i_k, j_k|i_{k-1}, j_{k-1})$ to show that it depends on the predecessor node. Since all the elements are ordered in time, the set of predecessor nodes are to the east and south of a current node. An example set of predecessors which includes only its immediate neighbors is given in Figure 1(a). Finally, the minimum cost path aligning two sequences has cost $D_{min}(i_f, j_f)$, which is equal to $D(\mathbf{r},\mathbf{t})$, the total cost or the dissimilarity between $\mathbf{r}$ and $\mathbf{t}$. The test sequence is matched to the reference sequence that has the minimum dissimilarity.

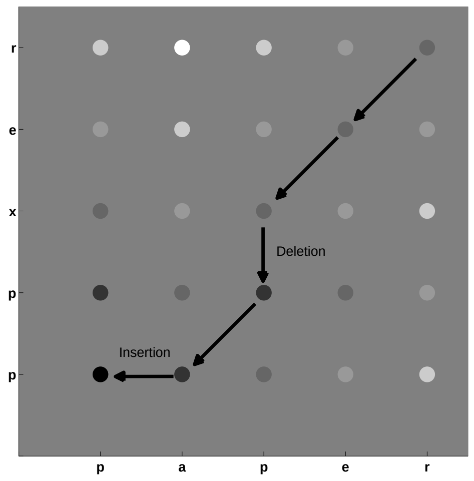
Although Equation (3) outputs the minimum cost between two sequences, it does not output the optimal path. To find the optimal path, which can be used to map test sequence elements to reference sequence elements, one needs to backtrack the optimal path starting with the final node. If the the whole test sequence is to be mapped to the whole reference sequence than $(i_f, j_f) = (I,J)$.

2.3 A STRING MATCHING EXAMPLE

String matching may be required in text retrieval and automatic editing applications (Pollock and Zamora, 1984; Morita and Shinoda, 1994). A sequence of characters (a test string) are read by a device or a program (*e.g.* optical character recognition (OCR) software), which is subsequently matched to a set of stored reference strings. In this simple example, a noisy image of the string "paper" is recognized as



(a) Local constraint on transitions



(b) 2D grid with D_{min} modulating the node intensity

Figure 1: **String pattern matching example:** The test string "ppxer" is placed in the vertical axis, and the reference string "paper" in the horizontal axis. The minimum cost path unravels the alignment to the reference string by performing one insertion and one deletion operation. The grid nodes are modulated with their corresponding D_{min}'s.

"ppxer" by an OCR software and then matched to a set of reference string using dynamic time warping[1]. The total cost $D_{min}(i,j)$ is computed using (3) and shown on the 2D grid as given in Figure 1(b). The intensity of each node (i,j) on the grid is modulated with $D_{min}(i,j)$. A lower cost is represented with a lower intensity. The optimal path is obtained by back-tracking $D_{min}(I,J)$, as depicted by arrows pointed towards the left-bottom corner of the grid. The optimal path is constructed by choosing the "darkest" nodes among the nodes in local constraint neighborhood as given in Figure 1(a). Since the OCR software is expected to miss fainted characters or misrecognize some characters, insertion (horizontal transition) and deletion (vertical transition) of a character is allowed. Insertion and deletion operations should be penalized since they also imply a mismatch but to a lesser degree than mismatch of two characters (diagonal transition). The distance function d is given below, and therefore depends on the predecessor nodes to incorporate this logic.

$$d(i,j|i-1,j-1) = \begin{cases} 0 & \text{if } r(i) = t(j) \\ 1 & \text{if } r(i) \neq t(j), \end{cases} \tag{4}$$

specifies the distance function for diagonal transitions, which maps a recognized character to a character in the reference string. For horizontal and vertical transitions

$$d(i,j|i-1,j) = d(i,j|i,j-1) = 0.5. \tag{5}$$

Insertion and deletion operations on a time sequence corresponds to stretching (slowing down) or compressing (speeding up) the sampled time signal to align with the reference time signal. For example in gesture recognition the same motion pattern of joints defining a gesture can be performed with varying speeds depending on the person. These variations are corrected by warping the test sequence in time.

[1]Dynamic programming is the more appropriate name for this example.

3 CONSTRAINED LINEAR DISCRIMINANT ANALYSIS FOR DTW

As can be seen in the string matching example, DTW is a nonlinear mapping, which renders linear techniques from machine learning useless in exploiting the data distribution. However, useful information can be learned by analyzing how sequences are distributed with respect to sequences of the same class or of other classes. This information can be utilized to find projection directions that best separate the data. To do so, we work on a training set in which all the sequences are of the same length. This assumption enables us to compute within-class scatter matrices and between-class scatter matrices to be used in our proposed constrained LDA. To equalize the length of the sequences in the training set, sequence vectors can be scaled up via interpolation.

3.1 DYNAMIC TIME WARPING ON ALIGNED SEQUENCES

DTW reduces to computing L_1 distance between two vectors created from the two sequences if the sequences were aligned in time and L_1 norm is used as the distance function between two sequence samples *i.e.*,

$$d(i_k, j_k) = |\mathbf{r}(i_k) - \mathbf{t}(j_k)|. \tag{6}$$

If the two sequences are aligned in time then $i_k = i_{k-1}+1$, $j_k = j_{k-1}+1$, hence $i_k = j_k = k$. By using this and (6) in (3),

$$D_{min}(k,k) = D_{min}(k-1,k-1) + |\mathbf{r}(k) - \mathbf{t}(k)|, \quad (7)$$

$$D_{min}(k,k) = \sum_{t=1}^{k} |\mathbf{r}(t) - \mathbf{t}(t)|. \quad (8)$$

If both sequences are of length L (*i.e.*, $i_f = j_f = m$), then

$$D(\mathbf{r},\mathbf{t}) = \sum_{t=1}^{m} |\mathbf{r}(t) - \mathbf{t}(t)|, \quad (9)$$

$$D(\mathbf{r},\mathbf{t}) = ||\mathbf{r} - \mathbf{t}||_1. \quad (10)$$

Similarly, the dissimilarity between two aligned sequences using the square of the L_2 norm as the distance function $d(i, j)$ leads to the L_2 distance between **r** and **t**.

3.2 LINEAR DISCRIMINANT ANALYSIS

Linear discriminant analysis is a discriminative feature generation technique. The method goes back to the work of Fisher on linear discrimination (Fisher, 1936), and is widely used in many machine learning problems such as face recognition (Liu and Wechsler, 2001). LDA has its roots in *principal component analysis* (PCA). PCA is one of the most popular methods in feature generation and dimensionality reduction in machine learning. PCA searches directions that have the largest variation, and projects on to these directions hence resulting in a lower dimensional representation of the data. Data projected on to each direction becomes an informative feature about the data describing it with respect to the overall data distribution. Although PCA is effective in finding directions with the largest variation in unsupervised learning, it does not search for directions that discriminate members of a class from another class, which is important in supervised learning. LDA achieves what PCA fails: By utilizing the label information of training data it finds discriminative directions that can be used to classify test data.

3.2.1 MEASURING CLASS SEPARABILITY

Let's formulate LDA as a maximization problem that finds discriminative projection directions. Our goal is to generate features that are well clustered around their class means. First, we will present the criteria to assess if a given set of features are scattered as such. To this end, the following scatter matrices are defined. *Within-class scatter matrix* is the average of the feature variances of each class out of M total classes as given below

$$S_{xw} = \sum_{i=1}^{M} P_i \Sigma_i, \quad (11)$$

where Σ_i is the covariance for class i, and

$$\Sigma_i = E[(\mathbf{x} - \mu_i)(\mathbf{x} - \mu_i)^T], \quad (12)$$

where P_i is the *a priori* probability of class i. *Between-class scatter matrix* is the average distance of each class mean from the global mean vector:

$$S_{xb} = \sum_{i=1}^{M} P_i (\mu_i - \mu_0)(\mu_i - \mu_0)^T, \quad (13)$$

where the global mean vector μ_0, and is defined as

$$\mu_0 = \sum_{i}^{M} P_i \mu_i. \quad (14)$$

Using these two scatter matrices, we can now define three criteria that asses if the generated features are scattered as desired. The first criterion is

$$J_1 = \frac{trace\{S_{xb}\}}{trace\{S_{xw}\}} \quad (15)$$

If the features are well clustered around their class mean, and classes are separated from each other, J_1 will take large values. Trace of a matrix is equal to the sum of its eigenvalues, while determinant is the product of its eigenvalues. Since our scatter matrices are symmetric and positive definite, and thus have positive eigenvalues (Hartley and Zisserman, 2004), both the trace and the determinant of scatter matrices will take on large values when their eigenvalues are large. Hence, a second criterion would be

$$J_2 = \frac{|S_{xb}|}{|S_{xw}|} = |S_{xw}^{-1} S_{xb}|, \quad (16)$$

and will produce similar values when all the eigenvalues are large.

To simplify the maximization problem, determinant is replaced with the trace[2], which leads to

$$J_3(\mathbf{x}) = trace\{S_{xw}^{-1} S_{xb}\}. \quad (17)$$

3.2.2 DISCRIMINATIVE FEATURE GENERATION

Let $\mathbf{x}$ be an m-dimensional vector of measurements, our aim is to extract l discriminative directions from the scatter of data vectors by maximizing J_3. A feature corresponding to a data vector is obtained by projecting on to l such informative directions. Then, the l-dimensional feature vector can be computed by

$$\mathbf{y} = A^T \mathbf{x}, \quad (18)$$

[2]However, $S_w^{-1} S_b$ is not necessarily symmetric and the eigenvalues are not guaranteed to be positive as it was in the justification of using J_2 instead of J_1.

where the discriminative vectors constitute the column-vectors of $m \times l$ matrix A. Our goal is to find l such features, *i.e.* $\mathbf{y}$, that maximizes J_3. The corresponding scatter matrices of $\mathbf{y}$ are obtained by using (18) in (11), (12), (13), and (14)

$$S_{yw} = A^T S_{xw} A, \quad S_{yb} = A^T S_{xb} A. \tag{19}$$

By substituting the scatter matrices of $\mathbf{y}$ in (17), A can be found as the maximizer of $J_3(\mathbf{y})$ as below

$$A = \arg\max \quad trace\{(A^T S_{xw} A)^{-1}(A^T S_{xb} A)\}. \tag{20}$$

Instead of solving for $\mathbf{y}$ directly, first transforming y to $\hat{\mathbf{y}} = B^T \mathbf{y}$, where B is a linear transformation which diagonalizes the scatter matrices (S_{yw}, S_{yb}), does not change the criterion value, *i.e.*, $J_3(\mathbf{y}) = J_3(\hat{\mathbf{y}})$. However, $\mathbf{y}$ is found to be

$$\hat{\mathbf{y}} = C^T \mathbf{x}, \tag{21}$$

where C is an $m \times l$ dimensional matrix that consists of l eigenvectors of $S_{xw}^{-1} S_{xb}$. By definition, between-class scatter matrix is of rank $M-1$. Hence, there can be at most $M-1$ eigenvectors with nonzero eigenvalues. We pick the maximum possible number of discriminative directions, thus $l = M-1$.

3.3 LINEAR DISCRIMINANT ANALYSIS WITH CONSTRAINTS

To find discriminative features that can be used for sequences warped in time, the projection operation for finding the feature vector must be performed so that the time information is not required. Let $\mathbf{x}$ denote the data vector which is the vectorized time sequence, the projection operation to the i^{th} discriminative direction is given by

$$y^p = (\mathbf{c}^p)^T \mathbf{x}, \tag{22}$$

where $\mathbf{c}^p$ corresponds to the p^{th} column of C, and y^p is the p^{th} feature. If the time sequence consists of n samples of size d, then the length of $\mathbf{x}$ is $m = n \times d$. If $\mathbf{x}$ and $\mathbf{c}_i$ is divided into smaller vectors of size d such that $\mathbf{x} = [(\mathbf{x}_1)^T, (\mathbf{x}_2)^T, \ldots, (\mathbf{x}_n)^T]^T$, and $\mathbf{c}^p = [(\mathbf{c}_1^p)^T, (\mathbf{c}_2^p)^T, \ldots, (\mathbf{c}_n^p)^T]^T$, where $\mathbf{x}_t$ represents the sample at time index t, and $\mathbf{c}_t^p$ represents the corresponding elements of the projection vector $\mathbf{c}^p$. Using this notation, the projection operation in (22) can be rewritten as

$$y^p = \sum_{t=1}^{n} (\mathbf{c}_t^p)^T \mathbf{x}_t$$

Given a reference sequence vector $\mathbf{r}$ and a test sequence vector $\mathbf{t}$, projection on to p^{th} discriminative vector is obtained by

$$\hat{r}^p = \sum_{t=1}^{n} (\mathbf{c}_t^p)^T \mathbf{r}_t,$$

$$\hat{t}^p = \sum_{t=1}^{n} (\mathbf{c}_t^p)^T \mathbf{t}_t,$$

where $\hat{r}^p$ and $\hat{t}^p$ constitute the p^{th} dimensions of the l-dimensional feature vectors $\hat{\mathbf{r}}$ and $\hat{\mathbf{t}}$, respectively. The difference $\hat{r}^p - \hat{t}^p$ between $\hat{r}^p$ and $\hat{t}^p$ is

$$\hat{r}^p - \hat{t}^p = \sum_{t=1}^{n} (\mathbf{c}_t^p)^T \mathbf{r}_t - \sum_{t=1}^{n} (\mathbf{c}_t^p)^T \mathbf{t}_t, \tag{23}$$

where we kept the summations separately to make it easier to introduce the warping concept in the following. The above summation can be computed iteratively by

$$D^p(k,k) = D^p(k-1,k-1) + (\mathbf{c}_k^p)^T \mathbf{r}_k - (\mathbf{c}_k^p)^T \mathbf{t}_k, \tag{24}$$

where $D^p(k,k)$ represents the two sums on the right-hand side of (23) and the first input k denotes that the first sum goes up to the k^{th} term and similarly the second input k denotes that the second sum goes up to the k^{th} term. Note that $D^p(n,n) = \hat{r}^p - \hat{t}^p$ since all the n terms in the two summations are included. In a warping between two sequences, sequence elements with different indices are mapped to each other as in (3). Similarly, (24) can be modified as below to obtain a warping

$$D^p(i_k, j_k) = D^p(i_{k-1}, j_{k-1}) + (\mathbf{c}_{i_k}^p)^T \mathbf{r}_{i_k} - (\mathbf{c}_{j_k}^p)^T \mathbf{t}_{j_k}. \tag{25}$$

Note that if $i_k = i_{k-1} + 1$ and $j_k = j_{k-1} + 1$, no warping is done and (25) boils down to (24).

Equation (25) can pair up sequence elements from different time instances so that temporal sequences can be aligned in time. Since during the DTW computation we can not do any inference on the temporal position of a sample of $\mathbf{r}$ or $\mathbf{t}$, discriminative vectors should not discriminate sequences using time information. Hence, $\mathbf{c}_{i_k}^p$'s should not depend on time, therefore $\mathbf{c}_{i_k}^p = \bar{\mathbf{c}}^p$, which reduces (25) to

$$D^p(i_k, j_k) = D^p(i_{k-1}, j_{k-1}) + (\bar{\mathbf{c}}^p)^T (\mathbf{r}_{i_k} - \mathbf{t}_{j_k}). \tag{26}$$

The above iterative computations are presented for the p^{th} feature. However, our feature vectors $\hat{\mathbf{r}}$ and $\hat{\mathbf{t}}$ have l dimensions. Thus, DTW cost needs to be computed for l features in parallel as below

$$\mathbf{D}(i_k, j_k) = \mathbf{D}(i_{k-1}, j_{k-1}) + \bar{C}^T (\mathbf{r}_{i_k} - \mathbf{t}_{j_k}), \tag{27}$$

where $\mathbf{D}(i_k, j_k) = [D^1(i_k, j_k), D^2(i_k, j_k), \ldots, D^l(i_k, j_k)]^T$ is the feature vector at grid point (i_k, j_k) and $\bar{C}$ is an augmented matrix of size $d \times l$ such that $\bar{C} = [\bar{\mathbf{c}}^1, \bar{\mathbf{c}}^2, \ldots, \bar{\mathbf{c}}^l]$.

At this stage we are equipped with an iterative computation method that computes l-dimensional feature vector differences at any point on the alignment grid, however we still need to determine which warping path out of all possible paths is a good warping path. A good warping path specifies a good temporal alignment between two sequences. Unfortunately, we can not use Bellman's principle to do so as we did to arrive at (3). This is because we are operating on differences between features not the distances and differences can be negative. To avoid exhaustively searching all warping paths, we make an assumption which drastically reduces the computational complexity. First, we make the following observation: for sequences that are similar to each other (*e.g.*, sequences in the same class), if projection directions that are optimized for discrimination is used than the distance between the projections given by $|\hat{\mathbf{r}} - \hat{\mathbf{t}}| = |\mathbf{D}(n,n)|$ will be small. This is because discriminative directions minimize within-class variance while maximizing between-class variance. Using this observation, we further assume that not only $|\mathbf{D}(n,n)|$ is small but also the cost $|\mathbf{D}(i_k, j_k)|$ of a *good* incomplete-warping-path $p = \{(i_0, j_0), (i_1, j_1), \dots (i_k, j_k)\}$ is also small. Hence, a *good* warping path ending in node (i_k, j_k) can be found by searching for a good neighbor node (i^*_{k-1}, j^*_{k-1}) as below

$$(i^*_{k-1}, j^*_{k-1}) = \arg \min_{(i_{k-1}, j_{k-1})} |\mathbf{D}^*(i_{k-1}, j_{k-1}) + \\ + \bar{C}^T(\mathbf{r}_{i_k} - \mathbf{t}_{j_k})|, \quad (28)$$

where $\mathbf{D}^*(i_k, j_k)$ denotes the associated l-dimensional cost vector of node (i_k, j_k) using a *good* path given by $p^* = \{(i^*_0, j^*_0), (i^*_1, j^*_1), \dots (i_k, j_k)\}$. $D^*(i_k, j_k)$ can then be found by

$$\mathbf{D}^*(i_k, j_k) = \mathbf{D}^*(i^*_{k-1}, j^*_{k-1}) + \bar{C}^T(\mathbf{r}_{i_k} - \mathbf{t}_{j_k}). \quad (29)$$

Since $\bar{C}^T(\mathbf{r}_{i_k} - \mathbf{t}_{j_k})$ can be negative, unlike $d(i_k, j_k)$ in the conventional DTW our cost computation iterations have been separated into two given by (28) and (29) as compared to (3).

Next, we pose a constrained version of the LDA problem, which requires the l discriminative vectors to be indifferent to temporal characteristics of data in the training set.

3.4 SOLVING FOR CONSTRAINED LINEAR DISCRIMINANT VECTORS

Given a discriminative vector $\mathbf{c}^p = [\mathbf{c}^p_1, \mathbf{c}^p_2, \dots, \mathbf{c}^p_n]$ of length nd, where d is the length of each $\mathbf{c}^p_t$, we want to impose constraints so that $\mathbf{c}^p_t = \bar{\mathbf{c}}^p$, and $\mathbf{c}^p$, becomes concatenation of n copies of $\bar{\mathbf{c}}^p$: $\mathbf{c}^p = [\bar{\mathbf{c}}^p, \bar{\mathbf{c}}^p, \dots, \bar{\mathbf{c}}^p]$. To achieve this, we need to impose $n-1$ equality constraints given by $\mathbf{c}^p_1 = \mathbf{c}^p_2$, $\mathbf{c}^p_2 = \mathbf{c}^p_3$, up to $\mathbf{c}^p_{n-1} = \mathbf{c}^p_n$, each of which is of size d. Hence, there will be $(n-1)d$ constraints on $\mathbf{c}^p$, which is the p^{th} column-vector of the transformation matrix C. This can be formulated as

$$PC = \mathbf{0}, \quad (30)$$

where P is the constraint matrix and is of size $(n-1)d \times m$ and C is of size $m \times l$, where d is the dimensionality of a sequence sample, which consists of n samples. Each row of P has only two non-zero elements of which one element is 1, and the other is -1. This imposes an equality constraint between the corresponding elements of $\mathbf{c}^p$.

The constraint $PC = \mathbf{0}$ implies that each column of C lies perpendicular to each of the rows of P. Hence, column vectors of C must lie in the orthogonal complement of the row-space of P. To enforce this, we make use of the Singular Value Decomposition (SVD) of P. Let $P = UDV^T$, where D is a diagonal matrix with r non-zero diagonal entries, *i.e.* P has rank r. The orthogonal complement of the row-space of C consists of the $n-r$ columns of V corresponding to the zero diagonal entries of D, which we denote by $P^\perp$. Hence any $C = P^\perp C'$, where C' is a suitable matrix of the same size with C will satisfy $PC = 0$. Substituting this in (21),

$$\hat{\mathbf{y}} = C^T(P^\perp)^T\mathbf{x}, \quad (31)$$

If we define $\mathbf{x}' = (P^\perp)^T\mathbf{x}$, and maximize J_3 on $\mathbf{x}'$ then the constrained LDA problem reduces to the unconstrained LDA discussed above. Accordingly, the scatter matrices have to be computed for $\mathbf{x}'$, *i.e.* $S_{x'w} = (P^\perp)^T S_{xw} P^\perp$ and $S_{x'b} = (P^\perp)^T S_{xb} P^\perp$.

4 OPTIMIZING DISCRIMINATIVE FEATURES FOR A MULTI-CLASS PROBLEM

As discussed in Section 1, a classifier based on minimizing the DTW cost reduces to an NN classifier based on a Euclidean distance if the two sequences were perfectly aligned in time. In this case DTW will be a redundant operation matching samples with the same indices. However, the sequences are warped in time and this is the reason an alignment is required in the first place. But this suggests the idea that using distances other than an Euclidean distance can lead to an improvement in the performance, which we have tackled in the previous section. By maximizing a criterion for discrimination we have shown that it is possible to find discriminative directions and we presented an iterative computation method which can be used for dynamically warping the two sequences. After the alignment between $\mathbf{t}$ and $\mathbf{r}^k$s for each class k has been performed, the computed costs will be utilized to pick the best matching reference sequence $\mathbf{r}^k$ as given below

$$k^* = \arg\min_k D(\mathbf{r}^k, \mathbf{t}), \tag{32}$$

which means that $\mathbf{t}$ is assigned to class k^*. This shows that we are solving our multi-class classification problem, and the hypothesis for each class k is denoted by $h^k_{(C,\mathbf{r^k})}(\mathbf{t}) = D(\mathbf{r}^k, \mathbf{t})$, in which we have shown the implicit hypothesis parameter C (*i.e.*, projection matrix that consists of l projection directions as its column vectors), in addition to the explicit hypothesis parameter $\mathbf{r}^k$.

4.1 ONE-VERSUS-OTHERS STRATEGY

There is no reason that the hypothesis parameter C should be independent of class type, hence the hypothesis parameter C should be learnt for each class separately. We employ a modified version of one-versus-all strategy to solve our multi-class classification problem. Hence our original problem is separated into M binary classification problems. Training for each binary classification is performed to learn for the projection matrix:

$$C^k = \max J_3^k \quad \text{such that} \quad PC = 0, \tag{33}$$

where J_3^k denotes that the J_3 criterion is computed for class k considering that there are only two classes: class k and the remaining classes. It is important to note that while learning for C^k of the k^{th} binary classification problem, we need to compute the within-class and between-class scatter matrices as explained in Section 3.2. The between-class scatter is given by (13) and finds the scatter between the global mean μ_0 and each class mean *i.e.*, μ_1 and μ_2. Hence, the between-class scatter matrix for two classes becomes

$$S_{xb} = P_1(\mu_1 - \mu_0)(\mu_1 - \mu_0)^T + P_2(\mu_2 - \mu_0)(\mu_2 - \mu_0)^T, \tag{34}$$

Although the between-class scatter matrix computed as above for each binary classification problem is correct, potentially a better scatter matrix can be computed to learn the data distribution. We know that each binary classification is a subproblem obtained by separating our original multi-class classification problem into k subproblems via one-versus-all strategy. By exploiting this fact, the between-class scatter matrix for the k subproblems can be computed using the original equation in (13) as if each of our binary classification is a multi-class classification problem. This method still takes advantage of solving binary classification problems instead of a multi-class problem but at the same does not forget that the true scatter is not between *one* versus *all*, but *one* versus the *other* remaining $M-1$ classes. Hence, we call this strategy one-versus-others and utilize it to compute the features.

	BHZI	BHZO	LHCCW	LHCW	LHW	RHCCW	RHCW	RHW
BHZI	100	0	0	0	0	0	0	0
BHZO	0	100	0	0	0	0	0	0
LHCCW	0	0	87.5	0	12.5	0	0	0
LHCW	0	0	0	100	0	0	0	0
LHW	25	0	0	0	75	0	0	0
RHCCW	0	0	0	0	0	87.5	0	12.5
RHCW	0	0	0	0	0	0	75	25
RHW	0	0	0	0	0	0	0	100

Table 1: Reference dataset, Conventional DTW

	BHZI	BHZO	LHCCW	LHCW	LHW	RHCCW	RHCW	RHW
BHZI	100	0	0	0	0	0	0	0
BHZO	0	100	0	0	0	0	0	0
LHCCW	0	0	75	12.5	12.5	0	0	0
LHCW	0	0	0	100	0	0	0	0
LHW	0	0	0	0	100	0	0	0
RHCCW	0	0	0	0	0	100	0	0
RHCW	0	0	0	0	0	0	87.5	12.5
RHW	0	0	0	0	0	0	0	100

Table 2: Reference dataset, Discriminant Boosted DTW

	BHZI	BHZO	LHCCW	LHCW	LHW	RHCCW	RHCW	RHW
BHZI	100	0	0	0	0	0	0	0
BHZO	12.5	87.5	0	0	0	0	0	0
LHCCW	12.5	0	75	0	12.5	0	0	0
LHCW	0	0	0	50	50	0	0	0
LHW	37.5	0	0	0	62.5	0	0	0
RHCCW	0	0	0	0	0	37.5	0	62.5
RHCW	0	25	0	0	0	0	50	25
RHW	0	0	0	0	0	0	0	100

Table 3: Real-time data, Conventional DTW

5 RESULTS AND DISCUSSION

We tested the performance of our proposed discriminant boosted DTW method on our gesture database which has eight gesture classes: Both Hands Zoom In, Both Hands Zoom Out, Left Hand Circle Counter Clock Wise/Clock Wise, Left Hand Wave,

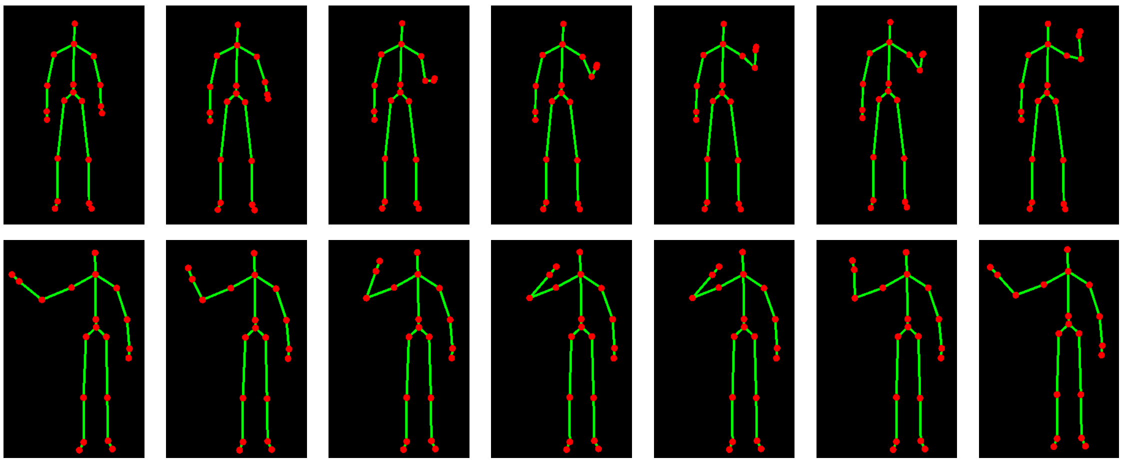


Figure 2: Two sample reference gestures from the gesture database visualized by our OpenGL Gesture Viewer: Right Hand Push Up and Left Hand Wave.

	BHZI	BHZO	LHCCW	LHCW	LHW	RHCCW	RHCW	RHW
BHZI	87.5	0	0	0	0	0	0	12.5
BHZO	0	75	0	25	0	0	0	0
LHCCW	0	0	42.5	45	0	0	12.5	0
LHCW	0	0	25	75	0	0	0	0
LHW	0	0	12.5	0	87.5	0	0	0
RHCCW	0	0	0	0	0	75	12.5	12.5
RHCW	0	0	0	0	0	25	62.5	12.5
RHW	0	0	0	0	0	12.5	0	87.5

Table 4: Real-time data, Discriminant Boosted DTW

Method	Accuracy
Conventional DTW	90.6250 %
Discriminant Boosted DTW	95.3125 %

Table 5: Reference dataset accuracy

Method	Accuracy
Conventional DTW	70.3125 %
Discriminant Boosted DTW	74.0625 %

Table 6: Real-time data accuracy

Method	Disc. Ratio
Conventional DTW	1.562019 %
Discriminant Boosted DTW	2.748761 %

Table 7: Reference dataset discriminant ratio

Method	Disc. Ratio
Conventional DTW	1.523670 %
Discriminant Boosted DTW	2.223496 %

Table 8: Real-time data discriminant ratio

Right Hand Circle Counter Clock Wise/Clock Wise, Right Hand Wave. The database has two datasets, both data sets record the XYZ locations of user skeleton joints performing the gestures by using Microsoft's Kinect sensor. Users are positioned at a fixed distance from the camera. First dataset, the reference-dataset, is performed properly by trained users, then a post-processing is applied including the deletion of gesture samples in which Kinect failed to detect a (correct) skeleton and deletion of redundant frames at the beginning and ending in which user does not perform the gesture. We used a gesture visualizer which is programmed with OpenGL to manually do this post-processing of gesture samples. The second database is the real-time dataset and it does not have any post-processing. Two sample reference gestures are shown in Figure 2.

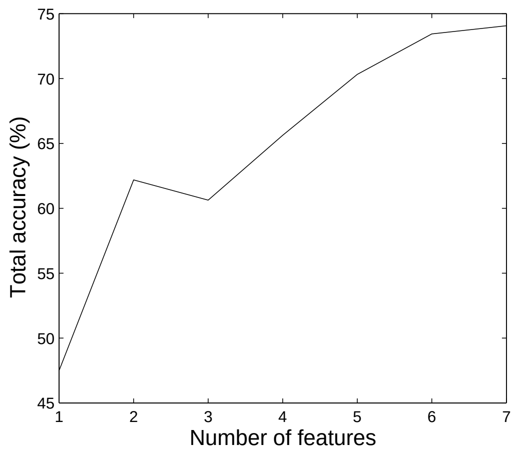


Figure 3: Accuracy increases when more features are used

Each gesture sample entry in the database includes 20 joint positions per frame and the time difference between two consecutive frames. We used only four joint positions in this work, Left/Right Hand, Left/Right Elbow for gesture recognition. The gesture databases used in the experiments, source code for visualization of gestures, and for producing the results in this paper and further results are publicly available[3].

In the first experiment set, we test our method with Conventional DTW in terms of the accuracy using the reference dataset and real-time dataset, the confusion matrices are given in Table 1 and 2. The overall accuracy of our proposed discriminant boosted DTW method outperforms the conventional DTW method, especially improving on difficult gestures such as moving the hand in a circular motion or waving the hands. The overall performances are given in Table 5 and 6 and discriminant ratios in 7 and 8. On real-time experiments the overall accuracies of both methods drop down significantly due to gestures performed by untrained users especially including errors at the starting and ending times of a gesture performance. However, our method still has a higher overall accuracy but interestingly has much larger discriminant ratio, which is defined as the average between-class costs divided into within-class costs. The significantly larger discriminant ratio implies that a higher accuracy rate can potentially be achieved by a cost computation method that weights down the beginning and ending of gesture sequences. However to do this, one needs to design a separate algorithm for detecting beginning and ending times.

To see the effect of the number of features used, we tested our algorithm by varying the number of features used. The results which are given in Figure 3 show that all features used are relevant and help to discriminate gesture samples. There is a small glitch observed by increasing the number of features from two to three. We think this is a noise created by the randomness and due to the under performance of the method when an insufficient number of features.

6 CONCLUSION

We proposed a framework to use linear feature generation techniques from machine learning to boost the discriminative power of nonlinear warping costs obtained by dynamic programming. To do so, we constrained the features to be independent of the sequence index (time) and solved the constrained LDA problem. Also, we modified the dynamic program to enable the use of these discriminative features. Furthermore, we proposed a new strategy for computing scatter matrices to be used in the one-versus-all strategy, which we named as one-versus-others.

[3]`http://mll.sehir.edu.tr/visapp2014`

ACKNOWLEDGMENT

The authors would like to thank to all Sehir University members who participated our gesture database recordings and patiently performed all gestures to help our experiments.

REFERENCES

Adams, N. H., Bartsch, M. A., Shifrin, J., and Wakefield, G. H. (2004). Time series alignment for music information retrieval. In *ISMIR*.

Amin, T. B. and Mahmood, I. (2008). Speech Recognition using Dynamic Time Warping. In *International Conference on Advances in Space Technologies*.

Baum, L. (1972). An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics*, 41:164–171.

Bleiweiss, A., Eshar, D., Kutliroff, G., Lerner, A., Oshrat, Y., and Yanai, Y. (2010). Enhanced interactive gaming by blending full-body tracking and gesture animation. In *ACM SIGGRAPH ASIA 2010 Sketches*, page 34. ACM.

Celebi, S., Aydin, A. S., Temiz, T. T., and Arici, T. (2013). Gesture Recognition Using Skeleton Data with Weighted Dynamic Time Warping. In *Computer Vision Theory and Applications*. Visapp.

Chang, Y.-J., Chen, S.-F., and Huang, J.-D. (2011). A kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities. *Research in Developmental Disabilities*, 32(6):2566 – 2570.

Corradini, A. (2001). Dynamic time warping for off-line recognition of a small gesture vocabulary. In *Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 2001. Proceedings. IEEE ICCV Workshop on*, pages 82–89. IEEE.

Fisher, R. (1936). The use of multiple measurements in taxonomic problems. *Annals of Human Genetics*, 7(2):179–188.

Gehrig, D., Kuehne, H., Woerner, A., and Schultz, T. (2009). Hmm-based human motion recognition with optical flow data. In *IEEE International Conference on Humanoid Robots (Humanoids 2009)*, Paris, France.

Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition.

Jain, H. P., Subramanian, A., Das, S., and Mittal, A. (2011). Real-time upper-body human pose estimation using a depth camera. In *Proceedings of the 5th international conference on Computer vision/computer graphics collaboration techniques*, MIRAGE'11, pages 227–238, Berlin, Heidelberg. Springer-Verlag.

Keskin, C., Kiraç, F., Kara, Y., and Akarun, L. (2011). Real time hand pose estimation using depth sensors. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1228–1234. IEEE.

Kuzmanic, A. and Zanchi, V. (2007). Hand shape classification using dtw and lcss as similarity measures for vision-based gesture recognition system. In *EUROCON, 2007. The International Conference on" Computer as a Tool"*, pages 264–269. IEEE.

Lee, H.-K. and Kim, J. (1999). An hmm-based threshold model approach for gesture recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(10):961 –973.

Liu, C. and Wechsler, H. (2001). A shape- and texture-based enhanced Fisher classifier for face recognition. *IEEE Transactions on Image Processing*, 10:598–608.

Mitra, S. and Acharya, T. (2007). Gesture recognition: A survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(3):311–324.

Morita, M. and Shinoda, Y. (1994). Information filtering based on user behavior analysis and best match text retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 272–281. Springer-Verlag New York, Inc.

Myers, C. and Rabiner, L. (1981). A comparative study of several dynamic time-warping algorithms for connected-word. *Bell System Technical Journal*.

Pollock, J. and Zamora, A. (1984). Automatic spelling correction in scientific and scholarly text. *Communications of the ACM*, 27(4):358–368.

Raptis, M., Kirovski, D., and Hoppe, H. (2011). Real-time classification of dance gestures from skeleton animation. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 147–156. ACM.

Rath, T. and Manmatha, R. (2003). Word image matching using dynamic time warping. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–521 – II–527 vol.2.

Rekha, J., Bhattacharya, J., and Majumder, S. (2011). Shape, texture and local movement hand gesture features for indian sign language recognition. In *Trendz in Information Sciences and Computing (TISC), 2011 3rd International Conference on*, pages 30 –35.

Reyes, M., Dominguez, G., and Escalera, S. (2011). Feature weighting in dynamic time warping for gesture recognition in depth data. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1182 –1188.

Ryden, F., Chizeck, H. J., Kosari, S. N., King, H., and Hannaford, B. (2011). Using kinect and a haptic interface for implementation of real-time virtual fixtures. In *Robotics Sciences and Systems, Workshop on RGB-D: Advanced Reasoning with Depth Cameras,*, Los Angeles,.

Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 26(1):43 – 49.

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. (2011). Real-time human pose recognition in parts from single depth images. In *CVPR*, volume 2, page 7.

Starner, T. and Pentland, A. (1996). Real-Time American Sign Language Recognition from Video Using Hidden Markov Models. In *International Symposium on Computer Vision*.

Stowers, J., Hayes, M., and Bainbridge-Smith, A. (2011). Altitude control of a quadrotor helicopter using depth map from microsoft kinect sensor. In *Mechatronics (ICM), 2011 IEEE International Conference on*, pages 358 –362.

Wenjun, T., Chengdong, W., Shuying, Z., and Li, J. (2010). Dynamic hand gesture recognition using motion trajectories and key frames. In *Advanced Computer Control (ICACC), 2010 2nd International Conference on*, volume 3, pages 163 –167.

Wikipedia (2012). Dynamic Time Warping. `http://en.wikipedia.org/wiki/Dynamic_time_warping`. [Online;accessed 01-September-12].

Wilson, A. D. (2010). Using a depth camera as a touch sensor. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 69–72, New York, NY, USA. ACM.