Multiple Multidimensional Sequence Alignment Using Generalized Dynamic Time Warping

PARINYA SANGUANSAT
Faculty of Engineering and Technology
Panyapiwat Institute of Management
NONTHABURI
THAILAND
parinyasan@pim.ac.th

Abstract: - Nowadays, the alignment of multidimensional sequences is required in many applications especially for multimedia data. The ordinary DTW is not well satisfied this condition because it can only align a pair of 1D sequences at once. Applying traditional DTW to these tasks not only makes each pairwise alignments are independent but also causes each dimensions are compared separately. In this paper, the generalized algorithm of Dynamic Time Warping (DTW) is proposed for multiple multidimensional time series alignments, called Multiple Multidimensional Dynamic Time Warping (MM-DTW). This algorithm utilizes all dimensions to obtain the optimal path and aligns multiple signals simultaneously. After alignment, the regular similarity measurement can be used to these aligned signals. The performances of MM-DTW are investigated by the nearest neighbor classifier compared to the ordinary DTW and its multidimensional modified algorithm. Experiments on real-world application, Query by humming, demonstrate the improved performance of the proposed method over the other algorithms.

Key-Words: - Multiple multidimensional dynamic time warping, Dynamic time warping, Multidimensional sequences, Dynamic programming, Signal processing, Query by humming

1 Introduction

Signal alignment is significant in many research areas, such as bioinformatics, speech recognition, time-series analysis, content-based retrieval etc. The characteristic of these signals is the varying in some dimensions, especially in time. For example, similarities in the speech of the same sentence of the same subject would be matched, even if there were accelerations and decelerations during speaking in the different times. Basically, Dynamic Time Warping (DTW) [1] is used to align a pair of sequences with optimal match and then measure the similarity. Unlike the traditional distance such as Euclidean distance, the DTW algorithm calculates the distance between each associated pairs of two sequences in order to minimize the distance while the point to point distance is used in traditional one. To manipulate this algorithm, dynamic programming is applied to find the least expensive path through a cumulative distance matrix. The obtained path can be used to align these two sequences into same length with synchronized points by insertion, deletion and match. However, this algorithm is available for only one attribute per point or 1D sequence while some features, which are multidimensional signals such as Mel-Frequency Cepstral Coefficients (MFCC), cannot be aligned.

For multidimensional sequence alignment, the Multidimensional Dynamic Time Warping (MD-DTW) was proposed in [2]. This technique was applied to gesture recognition in original work [2] and then another application for image texture similarity measurement in [3]. In MDDTW, only the distance matrix is changed and then the traditional DTW algorithm is applied on modified distance matrix. A distance measure for two multi-dimensional points must be calculated by any Lp-norm. The L1-norm, which is the sum of the absolute differences in all dimensions, was selected to use in [2] and [3]. Because of the different scale in each dimension, the signal is necessary to whiten by remove the first two statistical moments before combining. For Multiple Sequence Alignment (MSA), there are many proposed works in area of biological analysis [4, 5, 6, 7]. Unfortunately, none of these works can be applied to multidimensional signals.

Even through Hidden Markov Model (HMM) trends to outperform DTW in many applications but it can be utilized when the number of training samples is sufficient. In this paper, the performance of the proposed algorithm is investigated in the application of music information retrieval, query by humming. The number of training set is not sufficient in this application; there is only one MIDI per song in the database. Moreover, the humming sound and the song template are in different domains. Therefore, HMM was only used for note segmentation but not for matching in this application.

The remainder of this paper is organized as follows. In Section 2, the original DTW are described. The Multidimensional Dynamic Time Warping (MD-DTW) is presented in Section 3 then Multiple Multidimensional Dynamic Time Warping (MMD-DTW) is introduced in Section 4. The Query by Humming (QbH) is described in Section 5 with the proposed features. In Section 6, the experimental results are presented to demonstrate the effectiveness of our proposed techniques compared to DTW and MD-DTW. Finally, conclusions are presented in Section 7.

2 Dynamic Time Warping (DTW)

Due to the tempo variation of length of sequence, we cannot measure the similarity by any tradition distances. Dynamic Time Warping (DTW) is adopted to fill the gap caused by tempo variation between two sequences. For similarity measurement, DTW is used to compute the warping distance between the pair of sequences. Suppose that the input observation sequence is represented by $t(i), i=1,\ldots,m$, and the reference vector by $r(j), j=1,\ldots,n$. These two vectors are not necessarily of the same size (*m*, *n*). The distance in DTW is defined as the minimum distance starting from the beginning of the DTW table to the current position (*i; j*). According to the dynamic programming algorithm, the DTW table *D*(*i; j*) can be calculated by:

$$D(i,j)=d(i,j)+\min\begin{cases}D(i-1,j)\\D(i,j-1)\\D(i-1,j-1)\end{cases}, \qquad (1)$$

where *D*(*i; j*) is the node cost associated with *t*(*i*) and *r*(*j*) and can be defined from the L2-norm (the square root is not necessary for comparison purpose) as

$$d(i,j)=\left(t(i)-r(j)\right)^2. \qquad (2)$$

The best path is the one with the least global distance, which is the sum of cells alone the path. According to this algorithm, only two sequences can be compared and only signal with one attribute can be applied.

3 Multidimensional Dynamic Time Warping (MD-DTW)

If we need to apply the traditional DTW to align the multidimensional sequence, the simplest way is to perform DTW separately in each dimension. In this way, the summation of distance of all dimensions can be used to measure the similarity of a pair of sequences. However, the synchronized points in each dimension will be not the same position and uncorrelated.

To generalize the DTW for multidimensional sequence alignment, the Multidimensional Dynamic Time Warping (MD-DTW) was proposed in [2]. The idea of MD-DTW is the modified distance matrix *d* in (2) by using the vector norm to calculate the distance between a pair of points. To understand this algorithm, consider two series $\mathbf{t}\in\mathbf{R}^{K\times L_t}$ and $\mathbf{r}\in\mathbf{R}^{K\times L_r}$, where *K* is the number of attributes or the dimension of one point and L_t and L_r are the length of the sequences **t** and **r** respectively. The first step is the normalization of sequence in each dimension to zero mean and unit variance. Next step, calculating distance matrix *d* by

$$d(i,j)=\sum_{k=1}^{K}\left(t(k,i)-r(k,j)\right)^2 \qquad (3)$$

where the vector norm which normally use the L2-norm without square root operation is used. Finally, the minimal path is obtained by the traditional method as in the original DTW. According to this method, the feature values in all dimensions in each point of a pair of sequences will be synchronized at the same position. The Algorithm 1, in Table 1, describes the MD-DTW calculation in each step.

Table 1. MD-DTW algorithm

```
Algorithm 1 MD-DTW

Require: {t, r}
Ensure: path, distance
1: Normalize each dimension of t and r separately to a zero mean and unit variance
2: Calculate distance matrix according to (3)
3: Compute the matrix D by the traditional DTW algorithm in (1) , instead of the matrix D of the traditional approach
```

```
4: find path and distance along the minimal distance in D
5: return path, distance
```

4 Multiple Multidimensional Dynamic Time Warping (MMD-DTW)

For multiple alignments of multidimensional sequences, the distance of multiple sequences should be redefined. The modified distance between all points of all sequences can be illustrated as in Fig. 1. To simplify the implementation, all sequences were concatenated together in row direction before find the distance between all point pairs by,

$$d(i_1, i_2, \ldots, i_N) = \sum_{a=1}^{N} \sum_{b=a+1}^{N} \sum_{k=1}^{K} \left\| x_{a_{i_a}}(k) - x_{b_{i_b}}(k) \right\|, \quad (4)$$

where $x_{a_{i_a}}$ is the feature value at a^{th} point of k^{th} attribute of i_a^{th} sequence and N is the number of sequences. It is the cumulated distance of all point pair distances, which is no meaning in term of distance but it reflects the trend of data. Afterward, the distance matrix D can be computed by extending the idea of the original DTW in Fig. 2 to generalized way, multiple dimensions, as example in Fig. 3 for 3D-DTW. The calculation of D can be redefined as follow

$$D(i_1, i_2, \ldots, i_N) = d(i_1, i_2, \ldots, i_N) + \min \begin{cases} D(i_1 - 1, i_2, \ldots, i_N) \\ D(i_1, i_2 - 1, \ldots, i_N) \\ \vdots \\ D(i_1 - 1, i_2 - 1, \ldots, i_N) \\ \vdots \\ D(i_1 - 1, i_2 - 1, \ldots, i_N - 1). \end{cases} \quad (5)$$

The Algorithm 2, in Table 2, describes how to implement the MMD-DTW. It should be noted that MMD-DTW will return only the optimal path because the distance, which use in procedure in (4), has no meaning for comparing the multiple sequences. Each sequence must be previously rearranged as the obtained path before calculating the distance, which can computed by any classical method such as Euclidean distance.

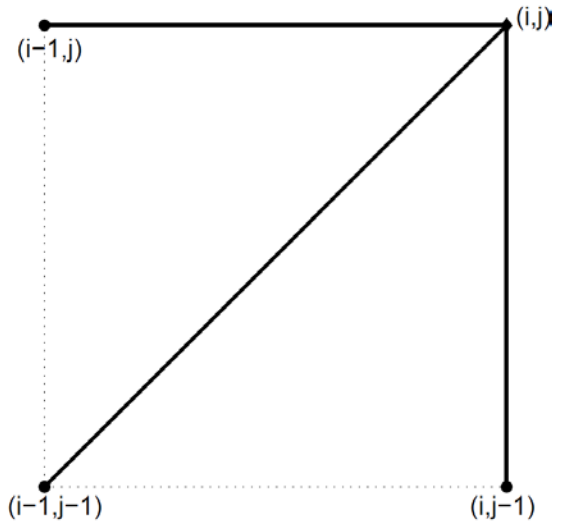

Fig. 2. Distance pairs in original DTW

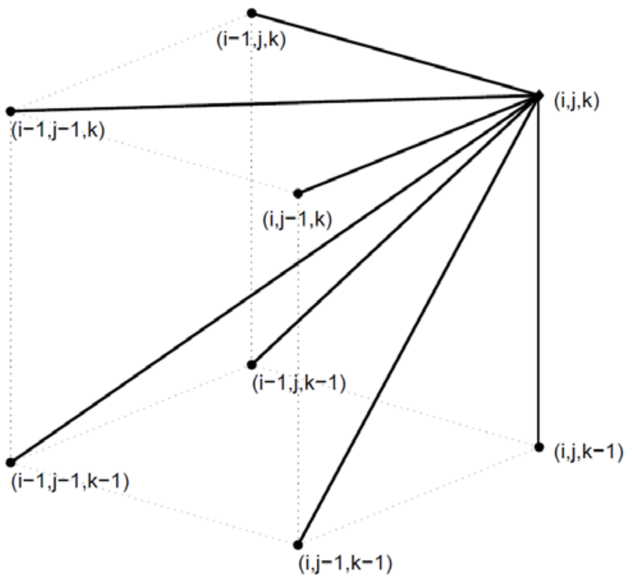

Fig. 3. Distance pairs in generalized DTW

Table 2. MMD-DTW algorithm

```
Algorithm 2 MMD-DTW

Require: $\{\mathbf{x}_i\}_{i=1}^{N}$
Ensure: path
1: Initial $D(1:L_1+2, 1:L_2+2, \ldots, 1:L_N+2) \leftarrow \infty$ { $L_i$ is the length of $\mathbf{x}_i$ }
2: $D(1) \leftarrow 0$
3: for $i_1 = 2 : L_1 + 1$ do
```

```
4:        ⋮
5:        for $i_N = 2 : L_N + 1$ do
6:            $\mathcal{X} \leftarrow [\mathbf{X}_1(i_1 - 1), \ldots, \mathbf{X}_N(i_N - 1)]$
              {Concatenate $\mathbf{x}_i$}
7:            $d \leftarrow dist(\mathcal{X})$
              {Calculate distance of $\mathcal{X}$ by (4)}
8:            $\mathbf{B} \leftarrow D(i_1 - 1 : i_1, i_2 - 1 : i_2, \ldots, i_N - 1 : i_N)$
9:            $min_{\mathbf{B}} \leftarrow \min(vec(\mathbf{B}))$
10:           $D(i_1, i_2, \ldots, i_N) \leftarrow d + min_{\mathbf{B}}$
11:       end for
12:       ⋮
13: end for
14: find path along the minimal distance in $D$
15: return path
```

5 Implementation

This section describes the implementation of DTW, MD-DTW and the proposed MMD-DTW in MATLAB [8] (It should be compatible with Octave [9]). The original DTW was implemented by Timothy Felty [10] as follow,

```
function [Dist,D,k,w]=DTW(t,r)
%Dynamic Time Warping Algorithm
%Dist is unnormalized distance between t and r
%D is the accumulated distance matrix
%k is the normalizing factor
%w is the optimal path
%t is the vector you are testing against
%r is the vector you are testing
%The orinal code by T. Felty [10]

[rows,N]=size(t);
[rows,M]=size(r);

d=(repmat(t(:),1,M)-repmat(r(:)',N,1)).^2;
%this replaces the nested for loops from above
Thanks Georg Schmitz

D=zeros(size(d));
D(1,1)=d(1,1);

for n=2:N
    D(n,1)=d(n,1)+D(n-1,1);
end
for m=2:M
    D(1,m)=d(1,m)+D(1,m-1);
end
for n=2:N
    for m=2:M
        D(n,m)=d(n,m)+ ...
           min([D(n-1,m),D(n-1,m-1),D(n,m-1)]);
    end
end

Dist=D(N,M);
n=N;
m=M;
k=1;
w=[];
w(1,:)=[N,M];
while ((n+m)~=2)
    if (n-1)==0
        m=m-1;
    elseif (m-1)==0
        n=n-1;
    else
        [values,number]=min([D(n-1,m), ...
                        D(n,m-1),D(n-1,m-1)]);
        switch number
            case 1
                n=n-1;
            case 2
                m=m-1;
            case 3
                n=n-1;
                m=m-1;
        end
    end
    k=k+1;
    w=cat(1,w,[n,m]);
end
```

For MD-DTW, only distance calculation of the original DTW is replaced by (3) as follow,

```
function [Dist,D,k,w]=dtwk(t,r)
%Dynamic Time Warping Algorithm
%Dist is unnormalized distance between t and r
%D is the accumulated distance matrix
%k is the normalizing factor
%w is the optimal path
%t is the vector you are testing against
%r is the vector you are testing
%The orinal code by T. Felty [10]
%Modify by Parinya Sanguansat


if nargin < 3
    L = 1;
end


[rows,N]=size(t);
[rows,M]=size(r);
d = 0;
for i=1:rows
    tt = t(i,:);
    rr = r(i,:);
    tt = (tt-mean(tt))/std(tt);
    rr = (rr-mean(rr))/std(rr);
    d = d + (repmat(tt(:),1,M) - ...
             repmat(rr(:)',N,1)).^2;
end


D=zeros(size(d));
D(1,1)=d(1,1);


for n=2:N
    D(n,1)=d(n,1)+D(n-1,1);
end
for m=2:M
    D(1,m)=d(1,m)+D(1,m-1);
end
for n=2:N
    for m=2:M
        D(n,m)=d(n,m)+min([D(n-1,m), ...
                      D(n-1,m-1),D(n,m-1)]);
    end
end


Dist=D(N,M);
```

```
n=N;
m=M;
k=1;
w=[];
w(1,:)=[N,M];
while ((n+m)~=2)
    if (n-1)==0
        m=m-1;
    elseif (m-1)==0
        n=n-1;
    else
        [values,number]=min([D(n-1,m), ...
                D(n,m-1),D(n-1,m-1)]);
        switch number
            case 1
                n=n-1;
            case 2
                m=m-1;
            case 3
                n=n-1;
                m=m-1;
        end
    end
    k=k+1;
    w=cat(1,w,[n,m]);
end
```

For the proposed MMD-DTW, it is not easy to implement this algorithm directly because the variable of for loop. The one way to manipulate this algorithm is to program the code for writing another code and then evaluate it later as follow,

```
function [y p] = MMD_DTW(x)
% MMD-DTW for multiple multidimensional signals
% x is a cell array of multiple n-D signals.
% p is the optimal path.
% y is a matrix of the normalized signals in
direction of column.
% 5/1/2011 22:11
% Parinya Sanguansat
if ~exist('x','var')
    x{1} = [1 2 3;1 2 3];
    x{2} = [1 3 3 4;1 3 3 4 ];
    x{3} = [1 2 2 3 4;1 2 2 3 4];
    %x{4} = [1 2 3 4 5];
    show = 1;
end
if ~exist('show','var')
    show = 0;
end
N = length(x);
% Initial e
sizee = zeros(1,N);
for i=1:N
    sizee(i) =  size(x{i},2)+2;
end
e = realmax*ones(sizee);
e(1) = 0;
% Find accumulated distance
str = '';
for q=1:N
    str = [str sprintf('for
i%d=2:size(x{%d},2)+1\n',q,q)];
end

str = [str 'cost =  dist(['];
for q=1:N
    str = [str sprintf('x{%d}(:,i%d-1) ',q,q)];
end
str = [str ']);'];

str = [str 'block = e('];
for q=1:N
    str = [str sprintf('i%d-1:i%d,',q,q)];
end
str = str(1:end-1);
str = [str ');mine = min(block(:));'];

str = [str 'e('];
for q=1:N
    str = [str sprintf('i%d,',q)];
end
str = str(1:end-1);
str = [str ') = cost + mine;'];

for q=1:N
    str = [str sprintf('\nend\n')];
end

str = [str 'e = e('];
for q=1:N
    str = [str sprintf('2:size(x{%d},2)+2,',q)];
end
str = str(1:end-1);
str = [str ');'];
eval(str);
% Find optimal path
p = ones(1,N);
p = minP(p,e,x);
if show
    if N==3
        plot3(p(:,1),p(:,2),p(:,3));
        xlabel(sprintf('S_1'));
        ylabel(sprintf('S_2'));
        zlabel(sprintf('S_3'));
        xlim([min(p(:,1)) max(p(:,1))]);
        ylim([min(p(:,2)) max(p(:,2))]);
        zlim([min(p(:,3)) max(p(:,3))]);
        grid on
    else
        if N==2;

imshow(e(1:size(x{1},2),1:size(x{2},2)),[]);
            figure
        end
        combos = combntns(1:N,2);
        mn = factor(size(combos,1));
        mn1 = floor(length(mn)/2);
        m = prod(mn(1:mn1));
        n = prod(mn(mn1+1:end));


        for k=1:size(combos,1)
            subplot(m,n,k);

plot(p(:,combos(k,1)),p(:,combos(k,2)));
            xlabel(sprintf('S_%d',combos(k,1)));
            ylabel(sprintf('S_%d',combos(k,2)));
        end
    end
end
% Normalized signals
for i=1:N
    y{i} = x{i}(:,p(:,i));
end

function p = minP(p,e,x)
% Find minimum e
% Last step
laststep = p(size(p,1),:);
N = length(laststep) ;
% Find minimum e
str = 'block = e(';
for q=1:N
    str = [str,sprintf('%d:%d,',laststep(q), ...
           laststep(q)+1)];
end
str = str(1:end-1);
str = [str,');'];
eval(str)
block(1) = realmax;
[mine idxe] = min(block(:));

str = '[';
for q=1:N
    str = [str,sprintf('move5(%d) ',q)];
end
str = [str , ']=ind2sub(size(block),idxe);'];
eval(str);
step = laststep+(move5-1);
p = [p;step];

str = 'fin = sum(step == [';
for q=1:N
    str = [str,sprintf('size(x{%d},2) ',q)];
end
str = [str , ']);'];
eval(str);
if fin == N
    return
end
% recursive
p = minP(p,e,x);

function d = dist(x)
% Find distance
d = 0;
for i=1:size(x,2)
    for j=i+1:size(x,2)
        d = d+sum(abs(x(:,i)-x(:,j))); % L1
    end
end
```

6 Query by Humming (QbH)

Query by Humming is the one of musical information retrieval application which uses the humming tune to search the song in database. Normally, the songs in database are stored in MIDI format to simplify the task. Because of the domain difference between sound wave (humming) and music scroll (MIDI), the pitch determination is necessary to extract the humming sound to melody sequence. Pitch extraction can be done by many methods such as autocorrelation, maximum likelihood cepstrum analysis [11] or Subharmonic-to-Harmonic Ratio (SHR) [12]. The first attempt in QbH use the three alphabets which indicate whether a note in sequence is up (U), down (D), or the same (S) as the previous note [11]. It seems simple but not accurate because this feature is too low level resolution. To improve this detail, the techniques based on continuous pitch contour were proposed in [13], [14], [15]. Normally, continues pitch contour is composed of pitch and duration time that makes this feature is sensitive to noise and error. In this paper, the pitch and duration time are considered separately in different dimensions. The pitch is determined by SHR after applying note segmentation. The corresponding duration time of each pitch is normalized by the length of hum. Therefore, this feature is the variable length multidimensional sequence that needs special technique as MDD-DTW to measure the similarity.

7 Feature Extraction

In this section, the feature extraction of humming sound is described. First of all, the note segmentation is applied using Hidden Markov Model (HMM). The model of silence and sound are formulated by training set to classify the note from the silence parts.

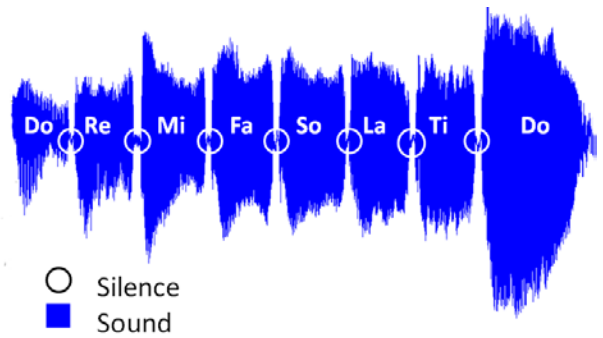

Fig 4. Sound wave from humming standard note in C major scale (do, re, me,..., do)

From the sound wave in Fig. 4, the silence interval is removed manually as pre-processing before being fed to the HMM. As shown in Fig. 5, the HMM contain 3 states with left-to-right topology using 2 Gaussian mixture distributions. Both the note and the silence are used to train these HMMs

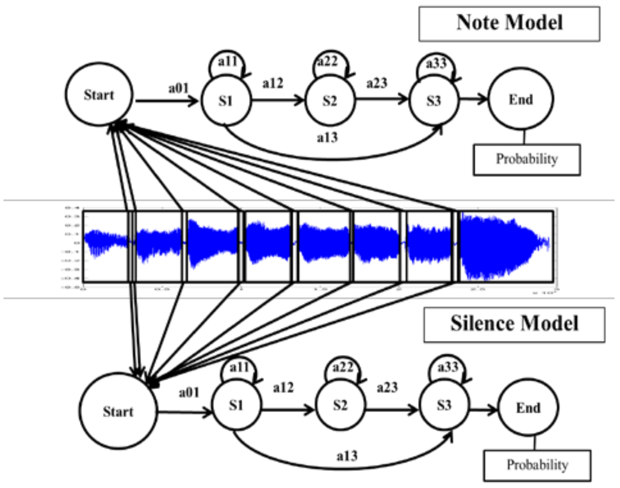

Fig. 5. Note and Silence Models

After note segmentation, each sound part is assumed that contains only one unknown note. To determine the actual note of this sound the SHR [12] is used to detect the note from its pitch. For each short-term signal, let $A(f)$ represents the amplitude spectrum, and let f_0 and $f_{\max}$ be the fundamental frequency and the maximum frequency of $A(f)$, respectively. Then the Sum of Harmonic (SH) amplitude is defined as

$$SH = \sum_{n=1}^{N} A(nf_0) \quad (6)$$

where N is the maximum number of harmonics contained in the spectrum, and $A(f) = 0$ if $f > f_{\max}$. If they confine the pitch search range in $[\mathrm{F0}_{\min}\ \mathrm{F0}_{\max}]$, then $N = \mathrm{floor}\left(f_{\max} / \mathrm{F0}_{\min}\right)$.

Assuming the lowest subharmonic frequency is one half of f_0. The Sum of Subharmonic (SS) amplitude is defined as

$$SS = \sum_{n=1}^{N} A((n-1/2) f_0) \quad (7)$$

Let $LOGA(\bullet)$ denotes the spectrum with log frequency scale, then they can represent SH and SS as

$$\begin{aligned} SH &= \sum_{n=1}^{N} LOGA(\log(nf_0)) \\ &= \sum_{n=1}^{N} LOGA(\log(n) + \log(f_0)) \end{aligned} \quad (8)$$

$$SS = \sum_{n=1}^{N} LOGA(\log(n-1/2) + \log(f_0)) \quad (9)$$

To obtain SH, the spectrum is shifted leftward along the logarithmic frequency abscissa at even orders, i.e., log(2), log(4),…, log(4N). These shifted spectra are added together and denoted by

$$SUMA(\log f)_{even} = \sum_{n=1}^{2N} LOGA(\log f + \log(2n)) \quad (10)$$

Similarly, by shifting the spectrum leftward at log(1), log(3), log(5), …, log(4N-1), they have

$$SUMA(\log f)_{odd} = \sum_{n=1}^{2N} LOGA\left(\log f + \log\left(2n-1\right)\right) \quad (11)$$

Next, they define a difference function as

$$DA(\log f) = SUMA\left(\log f\right)_{even} - SUMA\left(\log f\right)_{odd} \quad (12)$$

In searching for the maximum value, they first locate the position of the global maximum denoted as $\log(f_1)$. Then, starting from this point, the position of the next local maximum denoted as $\log(f_2)$ is selected in the range of $\left[\log(1.9375 f_1), \log\left(2.0625 f_1\right)\right]$ SHR equation is defined as

$$SHR = \frac{DA\left(\log f_1\right) - DA\left(\log f_2\right)}{DA\left(\log f_1\right) + DA\left(\log f_2\right)} \quad (13)$$

If SHR is less than a certain threshold value, it indicates that subharmonics are weak and they should favour the harmonics. Thus, f_2 is selected and the final pitch value is $2f_2$ Otherwise, f_1 is selected and the pitch is $2f_1$. SHR can be effectively used for pitch tracking.

After SHR process, the sequence of pitch (p) is extracted, including noise. The following algorithm, in Table 3, describes how to extract only useful pitch from humming sound to obtain the melody contour. Melody Contour Extraction, we have proposed in [16]. To make feature extraction, it can extract the significant feature and also reduce the dimension of feature. The humming sound consists of pitch in several values and also has noise fused in the pitch. Normally, the humming sound is usually reduced noise by median filtering which makes the signal is better smooth. However, it usually makes the discriminant information of the signal be lost at the same time. It is also applied for filtering part of signals prior to further processing with small window. We can reduce noise meanwhile the information of the signal is still reserved by melody contour extraction algorithm.

Let **m** represents melody contour and let **p** be the pitch. The variables of algorithm are described as follows: S is the size of the window for filtering, g is the gap of pitch difference, T is the threshold of standard deviation, and v is the variance of pitch interval.

The first step of this technique is to take a pitch to pass through the noise filtering process which uses the median filter in order to make the signal go smoothly. Then, find the different value of p by comparing with the defined g value by selecting only the exceed value. The value of s is determined in order to apply to find the range of signal that changes a little at that period of time. In other words, it discards the signal that changes rapidly in a short time comparing with this interval. There is the spread around the signal and it only needs the group of significant signals.

Table 3. Melody Contour Extraction algorithm

```
Algorithm 3 Melody Contour Extraction Algorithm
Require: p, $g$, $T$, $s$
Ensure: m
1: smoothing p by median filter
2: initial $m_1 \leftarrow p_1$
3: $N \leftarrow$ length of p
4: $j \leftarrow 1$
5: while $t \le N$ do
6:     $d = |p_t - p_{t-1}|$
7:     $Y \leftarrow \{p_{t-v}, p_{t-v+1}, \dots, p_{t+v-1}, p_{t+v}\}$
8:     $S_Y \leftarrow$ Standard deviation of $Y$
9:     if $d > g$ and $S_Y < T$ then
10:        $m_j \leftarrow p_t$
11:    end if
12:    $t \leftarrow t + s$
13:    $j \leftarrow j + 1$
14: end while
15: return m
```

Hence, it finds the range of signal which has a small value of the spread when comparing with the threshold of standard deviation (*T*) as shown in Fig. 6. The output of the algorithm melody contour contains significant pitch. Finally, when this technique is applied to retrieval task, it to do retrieval process, the result will be more correct than the median filter as shown in Fig. 7.

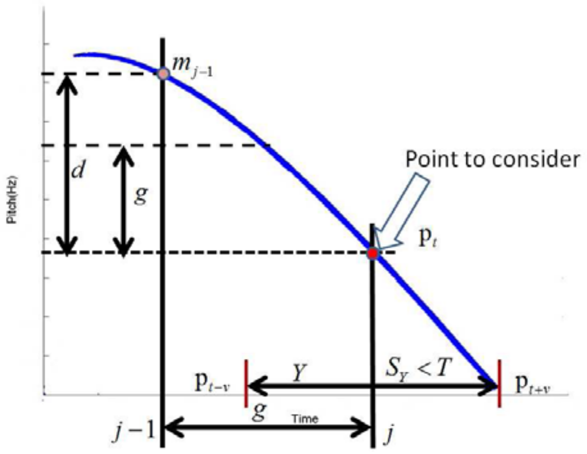

Fig. 6. Example of pitch extraction by Melody contour extraction

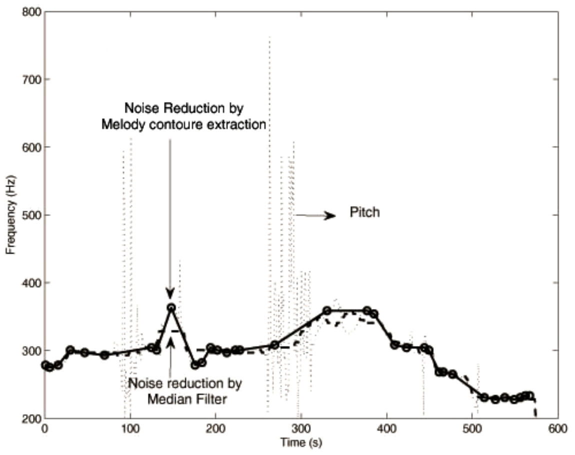

Fig. 7. Example of pitch sequence which is obtained by our technique, compare to median filter

The sequence of pitch is normalized to zero mean and unit variance. Next, the duration time of each pitch is extracted from the sequence of pitch and normalized by total time of this sequence as shown in Fig. 8.

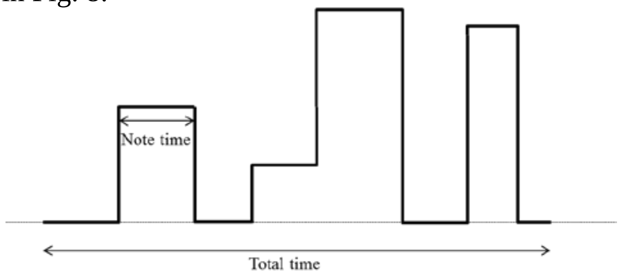

Fig. 8. The duration time normalization

Finally, the normalized pitch (note) and duration time are combined to a feature matrix, for example

0.45	-0.73	0.44	-0.94	-0.82	2.78
0.08	0.1	0.2	0.42	0.15	0.05

Each element in the first row is the mean of normalized pitch which calculated from the pitches in each interval. The second row is the normalized time (Note time / Total time) of each note. This feature is not simple to compare with another because the number of columns of this feature matrix is variable. The signal alignment is applied to enable this feature comparable.

8 Experimental Results

In this section, the efficiency of the proposed MMD-DTW was examined in query by humming problem. To demonstrate the graph of optimal path as in Fig. 6, only two songs were chosen to compare with a humming sound. The songs, which were used in these experiments, are MIDI format of popular Thai pop song. Ten humming sounds are hummed ("Da Da Da") at the beginning of the song around ten seconds by difference person for each song. As discussed in Section 6, the feature of each humming is two-dimensional sequence, pitch and duration time. The pitch and duration time were extracted from MIDI song as same manner as the humming sound to construct the sequence.

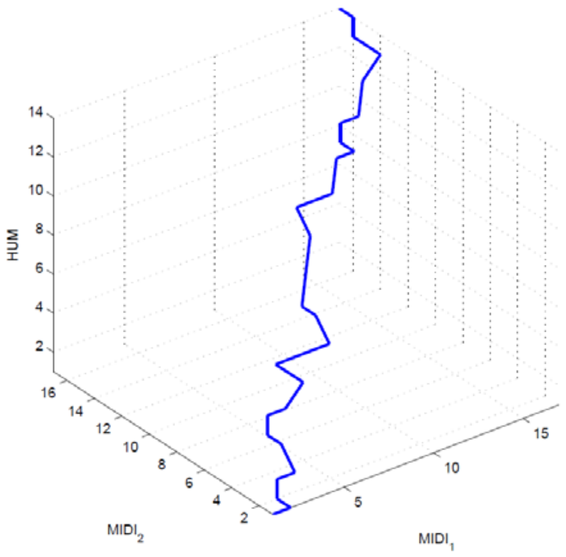

Fig.9. The optimal path by MMD-DTW

In this paper, the three methods of DTW were used to align these feature matrices. Firstly, the original DTW was used for aligning the multiple multidimensional sequences. Since DTW can align for sequence which has only one attribute therefore it cannot apply to our feature directly. For this limitation, the distance of each attribute is computed separately and then sum all distances together. For MD-DTW, our feature can be used directly because it supports the multidimensional sequence or the sequence with multiple attributes. However, Alignment by both DTW and MD-DTW are available only pairwise alignment scheme. That means the distances of all sequence pairs need to be computed $n(n+1)/2$ times (if n is the number of sequences). Finally, multiple alignments can be used only for MMDDTW. Afterward, the nearest neighbor used to classify the observation sequences.

The Tables 4, 5 and 6 are presenting the confusion matrices of DTW, MD-DTW and MMD-DTW, respectively. As the results, the accuracy rate of MMD-DTW is better than other methods. However, the processing time of MMD-DTW is more than the others. It depends on the number and length of sequences.

Table 4. Confusion matrix of DTW

Song	MIDI$_1$	MIDI$_2$
Hum$_1$	8	2
Hum$_2$	2	8

Table 5. Confusion matrix of MD-DTW

Song	MIDI$_1$	MIDI$_2$
Hum$_1$	7	3
Hum$_2$	0	10

Table 6. Confusion matrix of MMD-DTW

Song	MIDI$_1$	MIDI$_2$
Hum$_1$	9	1
Hum$_2$	1	9

9 Conclusions

This paper has proposed the generalized Dynamic Time Warping (DTW) for multiple alignments of multidimensional sequences, called Multiple Multidimensional Dynamic Time Warping (MMD-DTW). It is not only utilized all dimensions of signal to synchronize the alignment but also can align the multiple sequences at once. Because of the number of for-loop depends on the number of sequences that need to alignment, therefore the implementation can be done by the evaluation function which is available in many programming languages. This algorithm was evaluated on musical information retrieval, query by humming. The experimental results show the improvement over the traditional one and its extension Multidimensional Dynamic Time Warping (MD-DTW). The major problem of this proposed algorithm is the complexity which will be increasing exponentially when the number of sequences is increased.

References:

[1] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, *Acoustics, Speech and Signal Processing, IEEE Transactions on* 26 (1) (1978) 43 – 49. doi:10.1109/TASSP.1978.1163055.

[2] G. ten Holt, M. Reinders, E. Hendriks, Multi-dimensional dynamic time warping for gesture recognition, in: *Annual Conference on the Advanced School for Computing and Imaging*, 2007.

[3] R. F. Mello, I. Gondra, Multi-dimensional dynamic time warping for image texture similarity, in: *Proceedings of the 19th Brazilian Symposium on Artificial Intelligence*: Advances in Artificial Intelligence, SBIA ’08, 2008, pp.23–32.

[4] S. Wu, M. Lee, Y. Lee, T. M. Gatton, Multiple sequence alignment using ga and nn, International *Journal of Signal Processing, Image Processing and Pattern Recognition*, 1 (1) (2008) 21–30.

[5] J. Kim, S. Pramanik, M. J. Chung, Multiple sequence alignment using simulated annealing, *Bioinformatics* 10 (4) (1994) 419–426.

[6] C. Notredame, D. G. Higgins, Saga: Sequence alignment by genetic algorithm, *Nucl. Acids Res.* 24 (8) (1996) 1515–1524.

[7] C. B. Do, M. S. Mahabhashyam, M. Brudno, S. Batzoglou, Probcons: Probabilistic consistency-based multiple sequence alignment, *Genome Res.* 15 (2005) 330–340.

[8] The MathWorks, Inc. MATLAB [Computer program]. Available at http://www.mathworks.com/products/matlab (Accessed 1 March 2012)

[9] J. W. Eaton (2012), GNU Octave version 3.6.1 [Computer program]. Available at http://www.gnu.org/software/octave Retrieved 1 March 2012)

[10] T. Felty (2005). Dynamic Time Warping [Computer program]. Available at http:// www.mathworks.com/matlabcentral/fileexchange/6516-dynamic-time-warping (Retrieved 1 March 2012)

[11] A. Ghias, J. Logan, D. Chamberlin, B. C. Smith, Query by humming: musical information retrieval in an audio database, in: MULTIMEDIA ’95: Proceedings of the third ACM international conference on Multimedia, ACM, New York, NY, USA, 1995, pp. 231–236. doi:http://doi.acm.org/10.1145/217279.215273.

[12] X. Sun, Pitch determination and voice quality analysis using subharmonic-to-harmonic ratio, in: *Proceedings of the IEEE*, 2002, pp. 333-336.

[13] Y. Zhu, M. Kankanhalli, Similarity matching of continuous melody contours for humming querying of melody databases, in: of Melody Databases, *International Workshop on Multimedia Signal Processing*, USVI, 2002.

[14] T. Nishimura, J. X. Zhang, H. Hashiguchi, Music signal spotting retrieval by a humming query using start frame feature dependent continuous dynamic programming, in: *Continuous Dynamic Programming*, *Proc.* 3 rd *International Symposium on Music Information Retrieval*, 2001, pp. 211–218.

[15] Y. Zhu, M. S. Kankanhalli, C. Xu, Pitch tracking and melody slope matching for song retrieval, in: *PCM ’01: Proceedings of the Second IEEE Pacific Rim Conference on Multimedia*, Springer-Verlag, London, UK, 2001, pp. 530–537.

[16] N. Phiwma and P. Sanguansat, A music information system based on improved melody contour extraction, *in Signal Acquisition and Processing, 2010. ICSAP ’10*. International Conference on, 2010, pp.85 –89.

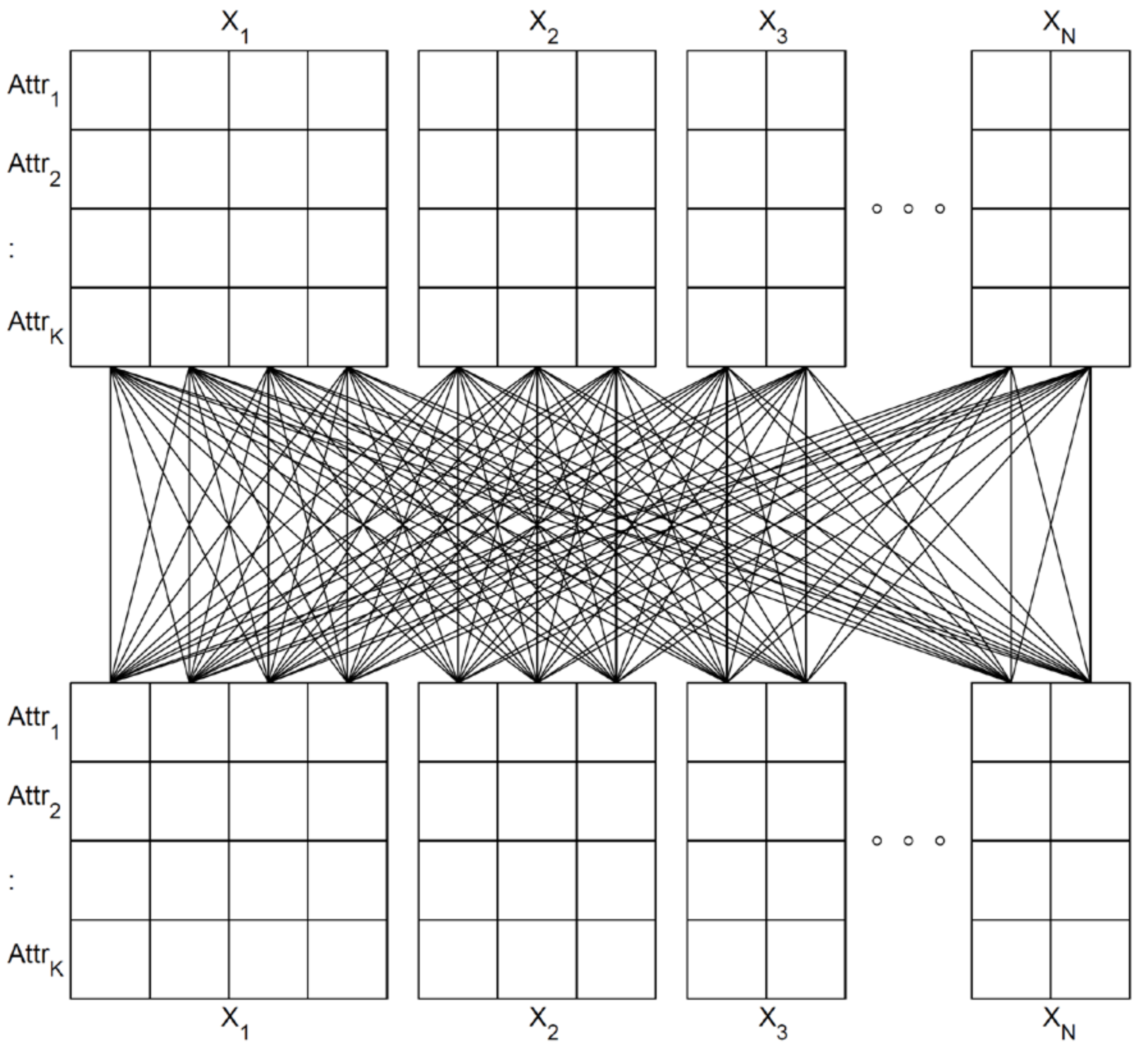

Fig. 1. Distance for MMD-DTW