Manzo-Martínez, Alain; Camarena-Ibarrola, José A.
A New and Efficient Alignment Technique by Cosine Distance
International Journal of Combinatorial Optimization Problems and Informatics
Morelos, México

Available in: http://www.redalyc.org/articulo.oa?id=265225625003
A New and Efficient Alignment Technique by Cosine Distance

Alain Manzo-Martínez, José A. Camarena-Ibarrola
Universidad Michoacana de San Nicolás de Hidalgo
División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica
alainmama@gmail.com, jacibarrola@gmail.com

Abstract. In this paper we describe a new technique to measure the similarity or distance between time series. We have called it, Alignment Technique by Cosine Distance (ATCD). Important features about the technique are that it requires neither a-priori knowledgement of the time series nor training stages. ATCD is based on cosine distance and least squares, and requires as a parameter the dimension of two support vectors. When we consider high dimensionality on these vectors, ATCD achieves its best performance providing the smallest measure of similarity (distance) as possible. ATCD can be used on applications of medical signal processing, audio and speech recognition, among others. We proved ATCD’s efficiency on an isolated-words speech recognition system by comparing ATCD against Dynamic Time Warping.

Keywords: Time Series, Cosine Distance, Least Squares, Alignment, speech recognition, dynamic time warping.

1 Introduction

At present there is a variety of alignment techniques that measure the similarity between time series (TS). Depending on the application and how the TS are represented, we can use techniques to align them as based on dynamic programming such as Dynamic Time Warping (DTW) [1], also there are ones based on stochastic basis such as Hidden Markov Models (HMM) [2], the ones used on strings processing such as Levenshtein Distance (LD) [3], among others. Many recognition systems require making alignment of TS in order to determine whether the information provided to the system fits with the information stored on their data bases. Therefore, alignment is an important and at the same time difficult task, since TS can be distorted by both, noise and scaling. Some applications where noise and scaling are present are: Speech recognition [4], audio recognition [5], averaged of electroencephalography signals to generate evoked potentials [6], morphological variability beat to beat on electrocardiography signals [7], among others. In these applications, we have two or more TS for aligning, namely, X, Y₁, Y₂, ..., Yₙ, where X is the query TS that contains the observations about a quantitative variable and Y₁, Y₂, ..., Yₙ are pattern TS that contain the observations on the same quantitative variable, but they were generated from the same experiment at different time instances. Therefore, for an observer, the series Y₁, Y₂, ..., Yₙ will be similar to X but with distorted values, stretched or compressed. An example of this occurs on speech recognition systems, where a person can say several times the same elocution through a day in order to access to a restricted place. The utterances can vary in time and shape due to all the involved factors. Speech recognition system should compare the TS produced by the utterances (Y₁, Y₂, ..., Yₙ) with the stored on its data base (X). The alignment process and the distance will determine if the person can access with success to the restricted place.

The isolated-words speech recognition systems have used methods based on vocal tract models, spectral analysis, among others, to obtain characteristic parameters of speech-signal. Some of these are: Linear Prediction Coding (LPC) [8], which estimates the basic speech parameters, e.g., pitch, formants, spectra, vocal tract area functions, and for representing speech for low bit rate transmission or storage. Mel-Frequency Cepstrum Coefficients (MFCC) [9], which is based on the known variation of the human ear’s critical bandwidths with frequency. The MFCC method makes use of two types of filter, namely, linearly spaced filters and logarithmically spaced filters. To capture the phonetically important characteristics of speech, signal is expressed in the Mel frequency scale. In this paper we are only focused on MFCC as method of features extraction.

This paper is organized as follow: A review of backgrounds on which this paper is supported is provided in section 2. On section 3, we describe how speech signal is processed. On section 4, we explain the ATCD and the algorithm is provided. On section 5, the experiments are described as well as the outcomes. Finally, on section 6, we provide conclusions.
2 Backgrounds

In this paper we present a new alignment technique to align TS in a straightforward and fast way, without require neither a-priori knowledge of the TS nor training stage, since, alignment is performed data by data between two TS. The technique is based on cosine distance and least squares, and it requires some heuristic to keep the smallest distance between both series.

2.1 Cosine Distance

Cosine distance has its fundamentals on scalar product. The scalar product among two vectors is always a scalar. The scalar product on d-dimensions is defined as shows equation (1)

\[ A \cdot B = |A||B| \cos(\theta) . \]  

(2)

where \(|A|\) and \(|B|\) are the magnitude of A and B vectors, respectively, and \(\theta\), the angle between both vectors. For a d-dimensional vector its magnitude is given by (2)

\[ |A| = \sqrt{a_1^2 + a_2^2 + \ldots + a_d^2} = \sqrt{\sum_{i=1}^{d} a_i^2} . \]  

(2)

If A and B vectors are represented in terms of their components, then

\[ A = a_1 \hat{i} + a_2 \hat{j} + a_3 \hat{k} + \ldots + a_d \hat{d} , \]

\[ B = b_1 \hat{i} + b_2 \hat{j} + b_3 \hat{k} + \ldots + b_d \hat{d} , \]

therefore, the scalar product is given by (3)

\[ A \cdot B = a_1 b_1 + a_2 b_2 + a_3 b_3 + \ldots + a_d b_d = \sum_{i=1}^{d} a_i b_i . \]  

(3)

Cosine distance is a function that measures the angle among two vectors on a d-dimensional space. From equations (1), (2) and (3), we can compute the cosine distance by (4)

\[ \theta = \cos^{-1} \left( \frac{\sum_{i=1}^{d} a_i b_i}{\sqrt{\sum_{i=1}^{d} a_i^2 \sum_{i=1}^{d} b_i^2}} \right) . \]  

(4)
2.2 Least Squares

The procedure more objective to fit a data set, \((x_1,y_1), (x_2,y_2), \ldots, (x_n,y_n)\), to a straight line on a dispersion diagram is least squares. The straight line has equation \(y = mx + b\), where \(b\) and \(m\) are coefficients that represent the intersection with the abscissas axis and the slope, respectively. To find the coefficients \(b\) and \(m\), the equations (5) and (6) are used.

\[
\begin{align*}
    m &= \frac{n \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \\
    b &= \frac{\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}.
\end{align*}
\]

2.3 Dynamic Time Warping

In our experiments we compare ATCD against DTW, so, we explain DWT briefly. Aligning two TS, \(X\) and \(Y\), with lengths \(n\) and \(m\), respectively, where \(X = \{x_1, x_2, \ldots, x_n\}\) and \(Y = \{y_1, y_2, \ldots, y_m\}\), implies finding a warping function \(j = w(i)\), that maps \(i\) and \(j\) indices so that the register between both TS is obtained. \(w(i)\) function is restricted to the limits \(w(0) = 0\) and \(w(n) = m\), and can be attached to several local conditions with different weights for each of them [1]. Generally [10] [11], to align these two TS, DTW build a matrix \(D\) of dimension \(n \times m\), which contains the distances of the best path as possible, regarding the local conditions. An example of such local conditions is that if the optimal warping function goes to the point \((i, j)\), it must go through either \((i-1, j-1)\), \((i, j-1)\) or \((i-1, j)\) as depicted in Fig. 1. A penalization of 2 is charged when choosing \((i-1, j-1)\) and of 1 if \((i, j-1)\) or \((i-1, j)\) are chosen, this way the three possible paths from \((i-1, j-1)\) to \((i, j)\) will all have the same cost of 2.

![Fig. 1. Symmetric local conditions.](image)

Let \(d_{nm}\) be the distance between the \(x_n\) and \(y_m\) components of \(X\) and \(Y\), respectively, then the optimal warping function between both series is defined by the minimum accumulated distance \(D_{nm}\) as in (7).

\[
D_{nm} = \sum_{n=1}^{m} d_{w(n),w(n)}.
\]

Once a local condition is selected, \(D_{nm}\) can be computed using the recurrence defined in equations (8), (9) and (10), which correspond to local condition shown in Fig. 1. Based on this recurrence \(D_{nm}\) can be efficiently obtained using dynamic programming.

\[
D_{nm} = \sum_{k=0}^{m} d_{k}. \tag{9}
\]
Zero Crossing Rate (ZCR) and energy. ZCR and energy identify unvoiced and voiced segments, respectively.

ZCR’s analysis usually is done under a framework of short-time. Short-time ZCR is defined as in (11), (12) and (13)

$$Z_n = \sum_{m} |\operatorname{sgn}[x(m)] - \operatorname{sgn}[x(m-1)]|w(n-m) \cdot$$

where

$$\operatorname{sgn}[x(n)] = \begin{cases} 1, & x(n) \geq 0 \\ -1, & x(n) < 0 \end{cases}$$

3 Speech Signal processing

In this section we describe how speech-signal is processed. Each elocation is registered via computer’s microphone and is stored in WAV format, without compression, sampling frequency of 8000Hz, quantized to 16 bits and single channel. Before to segment the speech-signal, we apply to it a pre-emphasis filter \( h(n) = x(n) - ax(n - 1) \) in order to highlight the high frequencies by using \( a = 0.9 \).

3.1 Segmentation

Fig. 2 shows the register of the elocation “processing”. We have observed that the amplitude of the speech-signal varies appreciably with time. In particular, the amplitude of unvoiced segments is generally much lower than the amplitude of voiced segments, therefore, to segment the elocation “processing”, we use Zero Crossing Rate (ZCR) and Energy. ZCR and energy identify unvoiced and voiced segments, respectively.

$$d_{ij} = \sum_{i=1}^s d_{ij}. \quad (9)$$

$$d_{ij} = \min \left\{ \frac{d_{i-1,j-1} + 2d_{ij}}{d_{ij} + d_{i+1,j}} \right\}. \quad (10)$$

Fig. 2. Register of the elocation “processing”.

$$Z_n = \sum_{m} |\operatorname{sgn}[x(m)] - \operatorname{sgn}[x(m-1)]|w(n-m) \cdot$$

where

$$\operatorname{sgn}[x(n)] = \begin{cases} 1, & x(n) \geq 0 \\ -1, & x(n) < 0 \end{cases}$$

$$\operatorname{sgn}[x(n)] = \begin{cases} 1, & x(n) \geq 0 \\ -1, & x(n) < 0 \end{cases}$$
and

\[
    w(n) = \begin{cases} 
        \frac{1}{2N}, & 0 \leq n \leq N - 1 \\ 
        0, & \text{otherwise}
    \end{cases}
\]  \quad (13)

The short-time energy of the speech-signal provides a convenient representation that reflects voiced sounds, it can be defined as in (14)

\[
    E_s = \sum_{n=-\infty}^{\infty} [x(m)w(n-m)]^2 
\]  \quad (14)

where \(x(m)\) is the speech-signal and \(w(n)\) a rectangular window.

An analysis in short-time entails to segment the signal in frames. We use a frame size of 30ms corresponding to 240 samples of signal and we use an overlap among frames of 80%. ZCR and energy are computed by using these considerations. A signal of reference (sr) is used in order to sum the weighted contributions of ZCR and energy. This signal is given by (15)

\[
    sr = bSZC_R + cSE 
\]  \quad (15)

where \(SZC\) is the ZCR signal, \(SE\) the energy signal, \(b\) and \(c\) are weights with value \(b = 0.1\) and \(c = 0.9\). With these weights we give more importance to the energy signal than ZCR signal. The \(sr\) must have amplitude at the most of 1. A threshold of 0.2 is then used to determine where begins and ends (red lines) the elocution on time. Fig. 3 shows elocution’s segmentation taking into account the signal of reference and the threshold.

![Speech Signal and Reference Signal](image)

**Fig. 3.** Segmentation of elocution “processing”.
3.2 Mel Frequency Cepstral Coefficients

We used MFCC for extracting the most relevant features of speech-signal. First the speech-signal is divided into short-time windows (i.e. frames with 240 samples), where we compute the Discrete Fourier Transform (DFT) of each time window for the signal x(n) with length N, given by (16)

$$X(k) = \sum_{n=0}^{N-1} w(n)x(n)e^{-j2\pi km/N}.$$  \hspace{1cm} (15)

for \( k = 0, 1, \ldots, N-1 \), where \( k \) correspond to the frequency \( f(k) = kf_s/N \), \( f_s \) is the sampling frequency in Hertz and \( w(n) \) is a time window. We chose Hann window as a time window, given by \( w(n) = 0.5(1 - \cos(2\pi n/N)) \). The magnitude spectrum \(|X(k)|\) is now scaled in both frequency and magnitude. First, the frequency is scaled logarithmically using the so-called Mel filter bank \( H(k,m) \) and then the logarithm is taken, giving (17)

$$X'(m) = \ln \left( \sum_{k=0}^{N-1} |X(k)| H(k,m) \right).$$  \hspace{1cm} (17)

for \( m = 1, 2, \ldots, M \), where \( M \) is the number of filter banks and \( M \ll N \). The Mel filter bank is a collection of triangular filters defined by the center frequencies \( f_c(m) \), written as in (18)

$$H(k, m) = \begin{cases} 
0, & \text{for } f_c(m - 1) \leq f(k) < f_c(m) \\
\frac{f_c(m) - f_c(m - 1)}{f_c(m) - f_c(m + 1)}, & \text{for } f_c(m - 1) \leq f(k) < f_c(m) \\
\frac{f_c(m) - f_c(m + 1)}{f_c(m) - f_c(m - 1)}, & \text{for } f_c(m) \leq f(k) < f_c(m + 1) \\
\frac{f_c(m) + f_c(m + 1)}{2}, & \text{for } f(k) \geq f_c(m + 1) \\
\frac{f_c(m) + f_c(m - 1)}{2}, & \text{for } f(k) \leq f_c(m - 1) \\
\frac{f_c(m - 1) + f_c(m + 1)}{2}, & \text{for } f(k) \geq f_c(m + 1) \\
0, & \text{for } f(k) \leq f_c(m - 1) \\
\frac{f_c(m) + f_c(m - 1)}{2}, & \text{for } f(k) \geq f_c(m + 1) \
\end{cases}. \hspace{1cm} (18)

The center frequencies of the filter bank are computed by approximating the Mel scale with (19)

$$\phi = 2595\log_{10}\left(\frac{f}{700} + 1\right).$$  \hspace{1cm} (19)

which is a common approximation. Note that this equation is non-linear for all frequencies. Then a fixed frequency resolution in the Mel scale is computed, corresponding to a logarithmic scaling of the repetition frequency, using \( \Delta \phi = (\phi_{\text{max}} - \phi_{\text{min}})/(M+1) \) where \( \phi_{\text{max}} \) is the highest frequency of the filter bank on the Mel scale, computed from \( f_{\text{max}} \) using equation (19), \( \phi_{\text{min}} \) is the lowest frequency in Mel scale, having a corresponding \( f_{\text{min}} \), and \( M \) is the number of filter banks. The values for the implementation are \( f_{\text{max}} = 4000\text{Hz} \), \( f_{\text{min}} = 0\text{Hz} \) and \( M = 5 \). The center frequencies on the Mel scale are given by \( \phi_c(m) = m\Delta \phi \) for \( m = 1, 2, \ldots, M \). To obtain the center frequencies in Hertz, we apply the inverse of equation (19) given by \( f_c(m) = 700(10^{\phi_c(m)/2595} - 1) \), which are inserted into equation (18) to give the Mel filter bank. Finally, the MFCCs are obtained by computing the Discrete Cosine Transform (DCT) of \( X'(m) \) using (20)

$$c(l) = \sum_{m=0}^{N} X'(m) w\left(l\frac{\pi}{M}(m-1/2)\right).$$  \hspace{1cm} (20)
for $l = 1, 2, \ldots, M$, where $c(l)$ is the $l$th MFCC. Thus, a vector of MFCC is obtained for each frame. Each $l$th MFCC is quantized to a value between 0 and 1. Thus, each elocution will be represented through a matrix with size $M \times W$, where $W$ is related to elocution’s duration. Fig. 4 shows a graphic representation of the elocution “processing”.

![Graphic representation for each MFCC obtained from the elocution “processing”](image)

Fig. 4. Graphic representation for each MFCC obtained from the elocution “processing”.

4 Alignment Technique by Cosine Distance

The $\theta$ angle between two $d$-dimensional vectors is relatively small, if the majority of their components present some similarity. Taking any two series, $X$ and $Y$, with same length as two vectors, $A$ and $B$, the angle between them will be among $0^\circ \leq \theta \leq 10^\circ$ if TS's elements are similar (this outcome was taken over the experimentation process). Whereas this similarity is held, the previous relation will always be true. If TS's elements are very different, the angle will be bigger than 10 degrees and smaller than 90 degrees, namely, $10^\circ < \theta < 90^\circ$. Considering this as true, the similarity measure or distance between both TS will be determined by $\theta$.

Now, when we consider two TS with different length, it is necessary an alignment process to measure the distance between them. ATCD considers two TS, $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_m\}$, where $n$ and $m$ are the lengths of $X$ and $Y$, respectively. It’s important to note that any TS can be the pattern or query series. The procedure begins defining two $d$-dimensional zero vectors, $A_0$ and $B_0$, in order to realize the alignment data by data between both TS. The $d$ dimension should be a value among 3 and $n$ or 3 and $m$. ATCD doesn't require negative elements on the TS, therefore, an offset can be added to the series if we have this case. Fig. 5 shows two TS with their vectorial representations, the zero vectors with $d = 4$, and two vectors, $A_x$ and $B_y$, where the aligned data are placed.

![Vectorial representation of TS and initial steps](image)

Fig. 5. Vectorial representation of TS and initial steps.
The procedure continues with the next steps:

Step 1. We set \( x_1 \) and \( y_1 \) on \( A_0 \) and \( B_0 \), respectively.

Step 2. Since the angle between \( A_0 \) and \( B_0 \) is \( \theta_r = 0^\circ \), the \( x_1 \) and \( y_1 \) elements are moved to \( A_x \) and \( B_y \), respectively, such as Fig. 5 shows. \( \theta_r \) angle is used as a reference measure for determining alignment data by data among two TS. Three additional variables are necessary and they should be initialized with \( \theta_i = 90^\circ \), \( \phi_1 = 0^\circ \) and \( \phi_2 = 0^\circ \). Then, taking into account the data, \( x_1 \), \( x_2 \), \( y_1 \) and \( y_2 \), the \( z_2 \) value is calculated by the best straight line that fits the data, Fig. 6 shows this step. It can be demonstrated that the \( z_2 \) value can be determined by the equation (21)

\[
x_1 = x_2 - \frac{(x_2 - y_2)}{2}
\]  

(21)

Step 3. A new \( \theta_i \) should be calculated putting the \( z_2 \) value on \( A_0 \) and the \( y_2 \) component on \( B_0 \). This angle is denoted by \( \theta_1 = \theta_r \).

Step 4. It’s necessary to calculate \( \theta_i \), but now regarding \( x_3 \) on \( A_0 \) and \( z_2 \) on \( B_0 \). This angle is denoted by \( \theta_2 = \theta_r \).

\[
\text{Fig. 6. Obtaining the } z_2 \text{ value by the best straight line that fits the data.}
\]

Step 5. If \( \theta_1 < \theta_2 \) and \( \theta_1 < \theta_i \), then, \( z_2 \) and \( y_2 \) will be on \( A_x \) and \( B_y \), respectively. On the other hand, if \( \theta_2 < \theta_1 \) and \( \theta_2 < \theta_i \), then, \( x_2 \) and \( z_2 \) will be on \( A_x \) and \( B_y \), respectively. \( \theta_i \) is updated to the smallest angle, this is, \( \theta_i = \theta_1 \) or \( \theta_i = \theta_2 \). Assuming that \( \theta_2 \) was smaller than \( \theta_1 \), the procedure will be continued calculating the \( z_3 \) value such as it was done for \( z_2 \), but now regarding \( x_3 \) and \( y_3 \). Once more \( \theta_1 \) and \( \theta_2 \) are calculated but now with the \( z_3 \) value. This procedure is showed in the Fig. 7.

\[
\text{Fig. 7. Procedure for the steps 3, 4 and 5.}
\]
Step 6. Assuming that $\theta_1$ was the smallest angle, this is, $\theta_1 < \theta_2$ and $\theta_1 < \theta_i$, the components on $A_x$ and $B_y$ will be $z_3$ and $y_3$, respectively. On the other hand, if $\theta_1 < \theta_2$ but $\theta_1 > \theta_i$, a new consideration should be taken. Due to the fact that $\theta_1$ was bigger than $\theta_i$, it is necessary to fit the alignment regarding the previous values (in this case $x_2$ and $z_2$) to reduce the distance between both TS.

Step 7. There are two new possibilities for considering; the first regards $z_3$ and $z_2$ on $A_0$ and $B_0$, respectively, and the second regards $x_2$ and $y_3$. For the first possibility, we denote with $\phi_1$ the angle calculated between $A_0$ and $B_0$. We denote with $\phi_2$ the angle taking the second possibility.

Step 8. When the angles are compared, if it results in $\theta_1 < \phi_1$ and $\theta_1 < \phi_2$, then, $z_3$ and $y_3$ are placed on $A_x$ and $B_y$, respectively. If $\phi_1 < \theta_1$ and $\phi_2 < \theta_1$, then, $z_3$ and $z_2$ are considered on $A_x$ and $B_y$, respectively. Finally, if $\phi_1 < \theta_1$ and $\phi_2 < \phi_1$, then, $x_2$ and $y_3$ are placed on $A_x$ and $B_y$, respectively. $\theta_1$ is updated adopting the biggest angle of $\theta_1$, $\phi_1$ y $\phi_2$. Fig. 8 shows the procedure of the steps 6, 7 and 8.

The 6, 7 and 8 steps, are applied until the last component of the TS is aligned. $A_0$ and $B_0$ vectors are behaved like queues, namely, once the cells are filled, the element that first in, first out. We notice that equation (4) can be computed easier, if we subtract and add only the contributions of the elements that get in and get out of the queue.

Step 9. Finally, the distance between both TS is obtained by computing cosine distance on the $A_x$ and $B_y$ vectors. The complete ATCD’s algorithm is shown in Table 1.

![Fig. 8. Graphic Representation for the steps 6, 7 and 8.](image-url)

Table 1. ATCD algorithm.

<table>
<thead>
<tr>
<th>Algorithm: Algorithm to makes alignment between two TS.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inputs:</strong> X and Y series and vector_size</td>
</tr>
<tr>
<td><strong>Outputs:</strong> Distance $\theta$, $A_x$ and $B_y$ vectors</td>
</tr>
<tr>
<td><strong>Initializations:</strong></td>
</tr>
<tr>
<td>$n = \text{series_size}_X$</td>
</tr>
<tr>
<td>$m = \text{series_size}_Y$</td>
</tr>
<tr>
<td>$A_0(1 \text{ until vector_size}) = 0$</td>
</tr>
</tbody>
</table>
\[ B_{\theta}(1 \text{ until vector}_\text{size}) = 0 \]
\[ \theta_i = 90^\circ, \phi_i = 0^\circ, 0^\circ, \theta_1 = 0, \theta_2 = 0 \]
\[ i = 1, j = 1, k = 1, l = 1, \text{pointer} = 1 \]

while \( i < n \) or \( i < m \) do

if \( i = 1 \) then

\[ A_0(l) \leftarrow X(j) \]
\[ B_0(l) \leftarrow Y(k) \]
\[ A_0(\text{pointer}) \leftarrow X(j) \]
\[ B_0(\text{pointer}) \leftarrow Y(k) \]
\[ \theta_1 = 0^\circ, i++; j++; k++, l++, \text{pointer}++ \]

end

if \( i > 1 \) then

\[ z = X(j) - (X(j) \cdot Y(k))/2 \] or \( z = Y(k) - (Y(k) \cdot X(j))/2 \)

To check the queues (\( A_0, B_0 \)) before to assign data. If queue filled does \( l = 1 \).

\[ A_0(l) \leftarrow z \]
\[ B_0(l) \leftarrow Y(k) \]
\[ \theta_1 \leftarrow \text{cosine_distance}(A_0, B_0) \]
\[ A_0(l) \leftarrow X(j) \]
\[ B_0(l) \leftarrow z \]
\[ \theta_2 \leftarrow \text{cosine_distance}(A_0, B_0) \]
\[ \theta_i \leftarrow \min(\theta_1, \theta_2) \]

if \( \theta_i < \theta_1 \) then

\[ A_0(l) \leftarrow z \] or \( X(j) \)
\[ B_0(l) \leftarrow Y(k) \] or \( z \)
\[ A_0(\text{pointer}) \leftarrow z \] or \( X(j) \)
\[ B_0(\text{pointer}) \leftarrow Y(k) \] or \( z \)
\[ i++; j++; k++, l++, \text{pointer}++ \]

end

if \( \theta_i > \theta_1 \) then

\[ \phi_1 \leftarrow \text{cosine_distance}(A_0, B_0) \]
\[ A_0(l) \leftarrow A_0(l - 1) \]
\[ B_0(l) \leftarrow z \] or \( Y(k) \)
\[ \phi_2 \leftarrow \text{cosine_distance}(A_0, B_0) \]

if \( \theta_i < \phi_1 \) and \( \theta_i < \phi_2 \) then

\[ A_0(l) \leftarrow z \] or \( X(j) \)
\[ B_0(l) \leftarrow Y(k) \] or \( z \)
\[ A_0(\text{pointer}) \leftarrow z \] or \( X(j) \)
\[ B_0(\text{pointer}) \leftarrow Y(k) \] or \( z \)
\[ i++; j++; k++, l++, \text{pointer}++ \]

end

if \( \phi_1 < \theta_i \) and \( \phi_1 < \phi_2 \) then

\[ A_0(l) \leftarrow z \] or \( X(j) \)
\[ B_0(l) \leftarrow B_0(l - 1) \]
\[ A_0(\text{pointer}) \leftarrow z \] or \( X(j) \)
\[ B_0(\text{pointer}) \leftarrow B_0(l - 1) \]
\[ i++; k++, l++, \text{pointer}++ \]

end

if \( \phi_2 < \theta_i \) and \( \phi_2 < \phi_1 \) then

\[ A_0(l) \leftarrow A_0(l - 1) \]
\[ B_0(l) \leftarrow z \] or \( Y(k) \)
\[ A_0(\text{pointer}) \leftarrow A_0(l - 1) \]
By (pointer) ← z or Y(k)
i++, j++, l++, pointer++
end
end
end
θi ← max(θr, φ1, φ2)
end
θ ← cosine_distance(Ax, By)
return θ, Ax and By

5 Experiments and Outcomes

In this paper we compare the performance of ATCD against DTW. The distance used for DTW was Euclidean distance given by (22), where p and q are d-dimensional vectors.

\[ d(p, q) = \sqrt{\sum_{l=1}^{d} (p_l - q_l)^2} \]  

(22)

A set of 500 different words (utterances) was considered and for each word, its respective MFCC matrix was obtained and stored on a data base. 50 words were randomly selected for matching. The MFCC matrices of these 50 words were used as queries. The matching test consisted in searching these queries in the whole data base.

On an isolated-words speech recognition system the matching is determined by measuring the distance among MFCC matrices. We measure the distance among these matrices with ATCD as follow: We consider all the rows as TS, therefore, the matching is performed by taking the first rows of each matrix and computing ATCD on them, then, the second rows are considered and so on. The total distance among two matrices will be the sum of the distances gotten for each row, namely, we need to make M alignments for that matter. Fig 9 shows the graphic of the five rows of two MFCC matrices that correspond to the elocution “processing”.

Fig. 9. Representation in TS of each row of the MFCC matrix.
As described in section 4, ATCD requires as a parameter the dimension of the $A_0$ and $B_0$ vectors. It is necessary to observe the behavior of these vectors for various dimensions in order to characterize this parameter. Fig. 10 shows the behavior of this parameter obtained from the alignment of two TS that corresponds to the first row of the MFCC matrices. In this figure we observe that when the dimension grows up, the distance oscillates but there is a dimension value where the distance doesn’t change a lot (over the dimension value of 70). Fig. 11 shows the previous said but considering two TS totally different. It is interesting to notice that in this case the distance grows up even more when we increase the dimension of the vectors.

For measuring the performance of ATCD and DTW, we made an analysis of sensitivity that consisted in comparing each query matrix with all matrices stored in the data base. When we compare two matrices corresponding to the same elocution, if the distance between them is lower than a threshold, it is a true positive (TP). If the comparison delivers a distance greater than the threshold then, it is a false negative (FN). Now, when we compare two different utterances and the distance between them is lower than the threshold, it is a false positive (FP), and if the distance is greater than the threshold, then, is a true negative (TN). The true prediction rate (TPR) is computed using (23). The false prediction rate (FPR) is determined with (24).

$$\text{TPR} = \frac{TP}{TP + FN}$$  \hspace{1cm} (23)

$$\text{FPR} = \frac{FP}{FP + TN}$$  \hspace{1cm} (24)

The TPR and FPR for different thresholds yield a ROC curve (Receiver Operating Characteristic curve). Fig. 12 shows the resulting ROC curves for ATCD and DTW. The analysis of sensitivity shows that ATCD and DTW had almost the same...
performance but ATCD is faster than DTW in computing. ATCD’s performance is represented with the black line, whereas DTW’s performance is represented with the red line.

![ROC curves for ATCD (black line) and DTW (red line).](image)

Fig. 12. ROC curves for ATCD (black line) and DTW (red line).

6 Conclusions

We have introduced a new alignment technique and showed that it can be used in applications of speech recognition. Our algorithm is straightforward and fast to implement so, it can be used under another applications. When we considered high dimensionality on \( \mathbf{A}_0 \) and \( \mathbf{B}_0 \) vectors, ATCD achieved its best performance providing the distance lowest if the TS are similar, on the other hand, if the TS are totally different ATCD delivers a distance even greater which is good in matching applications. ATCD and DTW had almost the same performance as showed Fig. 12, but ATCD can be computed faster than DTW.

References