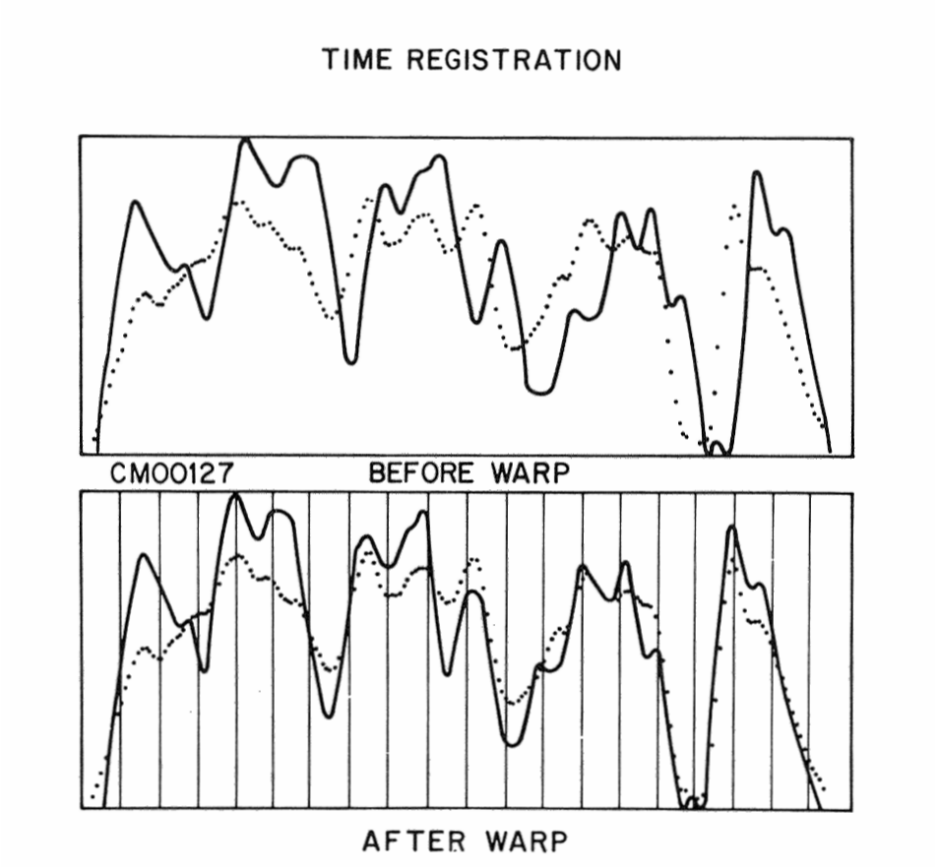
DTW. Alineamiento Temporal Dinámico (DTW)

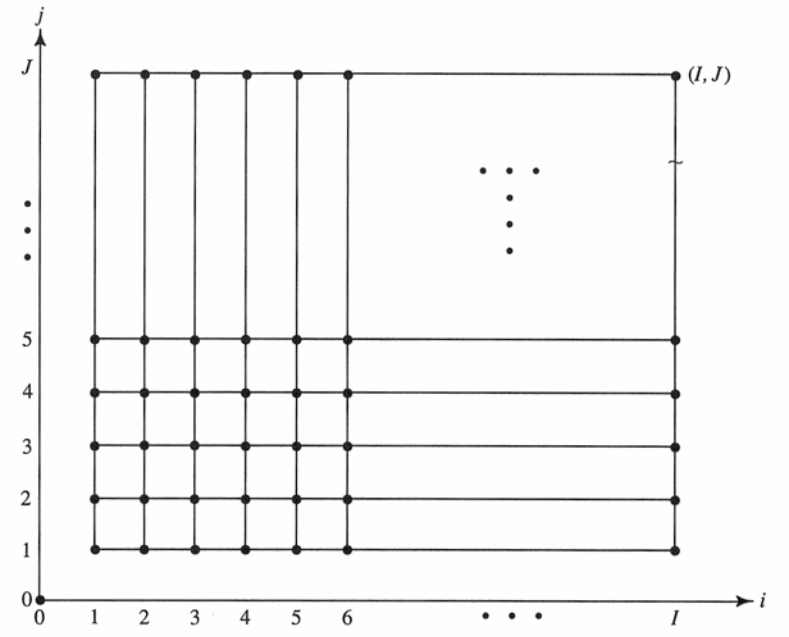
El Alineamiento Temporal Dinámico (*Dynamic Time Warping*, DTW), es una técnica surgida de la problemática inherente a diferentes realizaciones de una misma locución, en las que se observa una variabilidad interna en la duración de los grupos fónicos que la forman, de modo que no existe una sincronización temporal (alineamiento temporal). Además, esta falta de alineamiento no obedece a una ley fija (p. e., un retardo constante), sino que se da de forma heterogénea, produciéndose así variaciones localizadas que aumentan o disminuyen la duración del tramo de análisis.

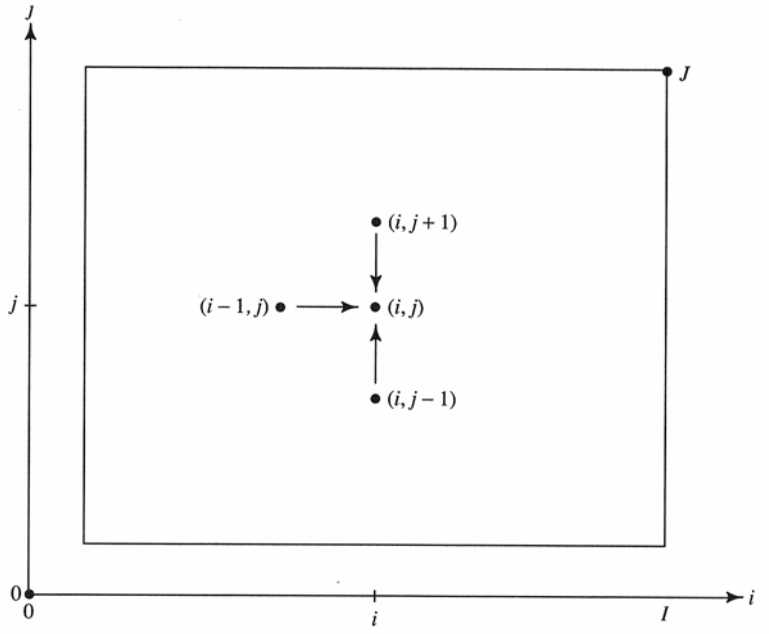
La problemática asociada hace referencia a la dificultad añadida en el proceso de medida de distancia entre patrones, puesto que se estarán comparando tramos que pueden corresponder a unidades fónicas distintas. Será necesario alinear

temporalmente la locución para proceder a realizar una medida de distancia entre patrones cuyo nuevo eje temporal haya homogeneizado las variaciones iniciales. La figura siguiente muestra dos realizaciones de la misma locución (contorno de energía localizada) antes y después de ser alineadas:

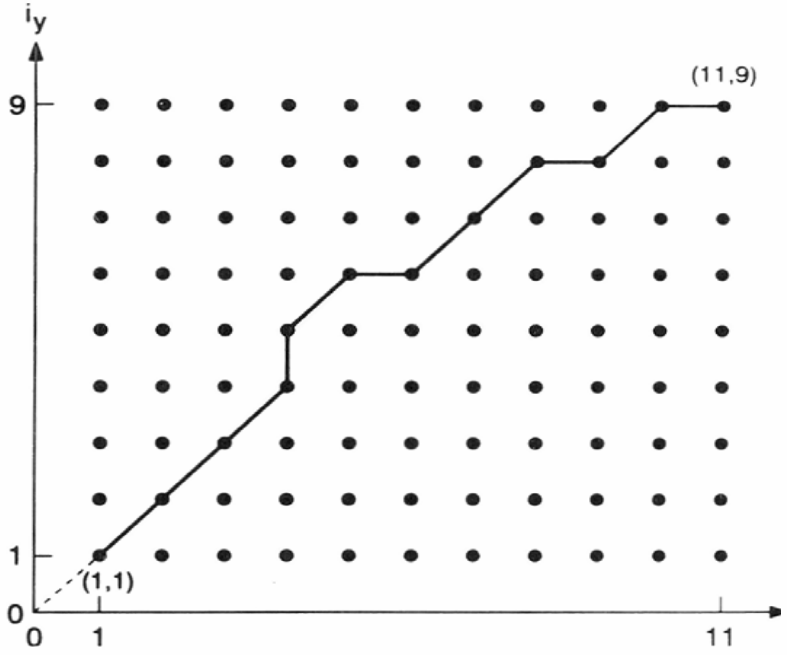


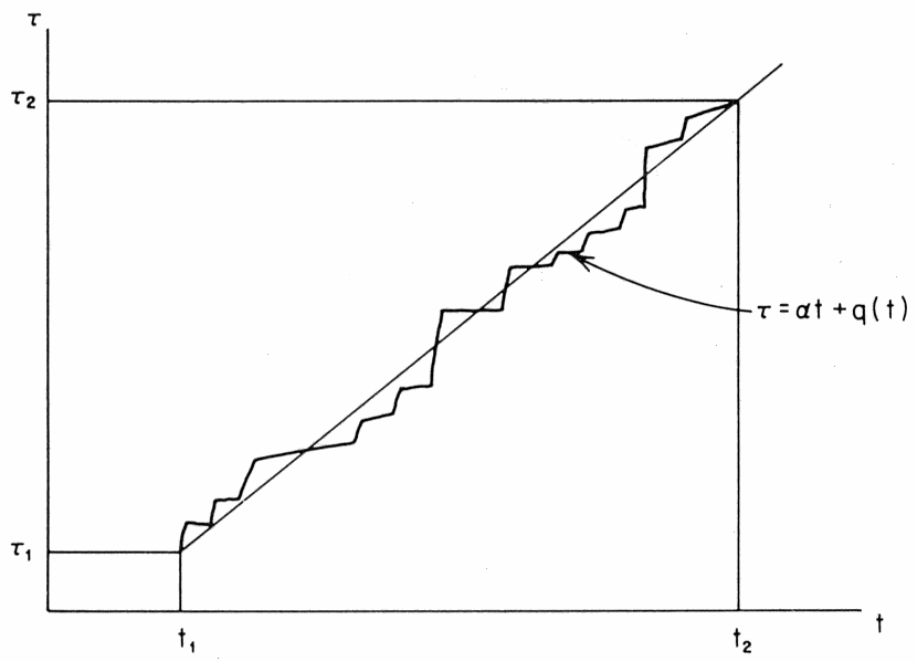
Alineamiento temporal (temporal warping): en este proceso, el eje temporal de la señal de test se comprime y expande de forma no lineal para alinear los vectores de características entre patrón y test.



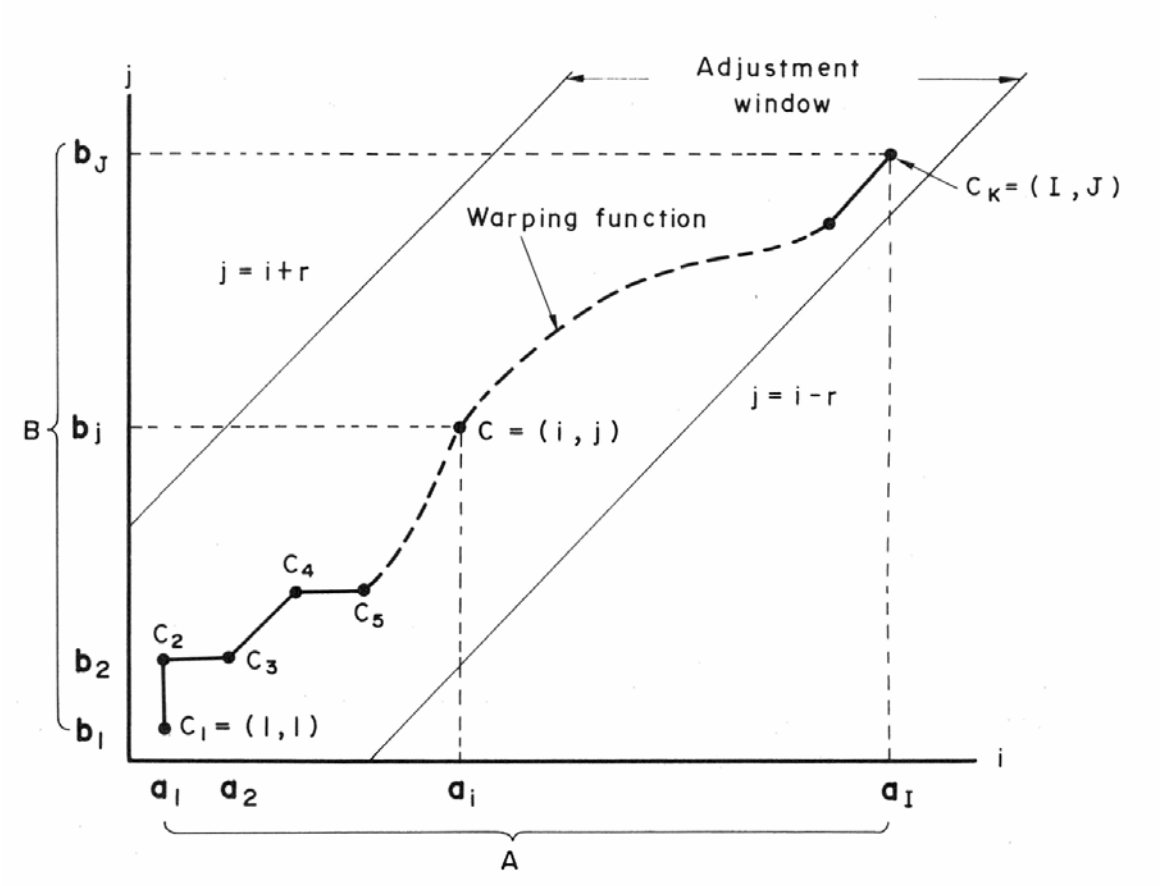


Fruto del proceso de alineamiento, surge lo que se conoce como camino de alineamiento. Las figuras siguientes muestran de forma simplificada (izqda.) y de forma real (drcha.) dos posibles caminos de alineamiento entre realizaciones

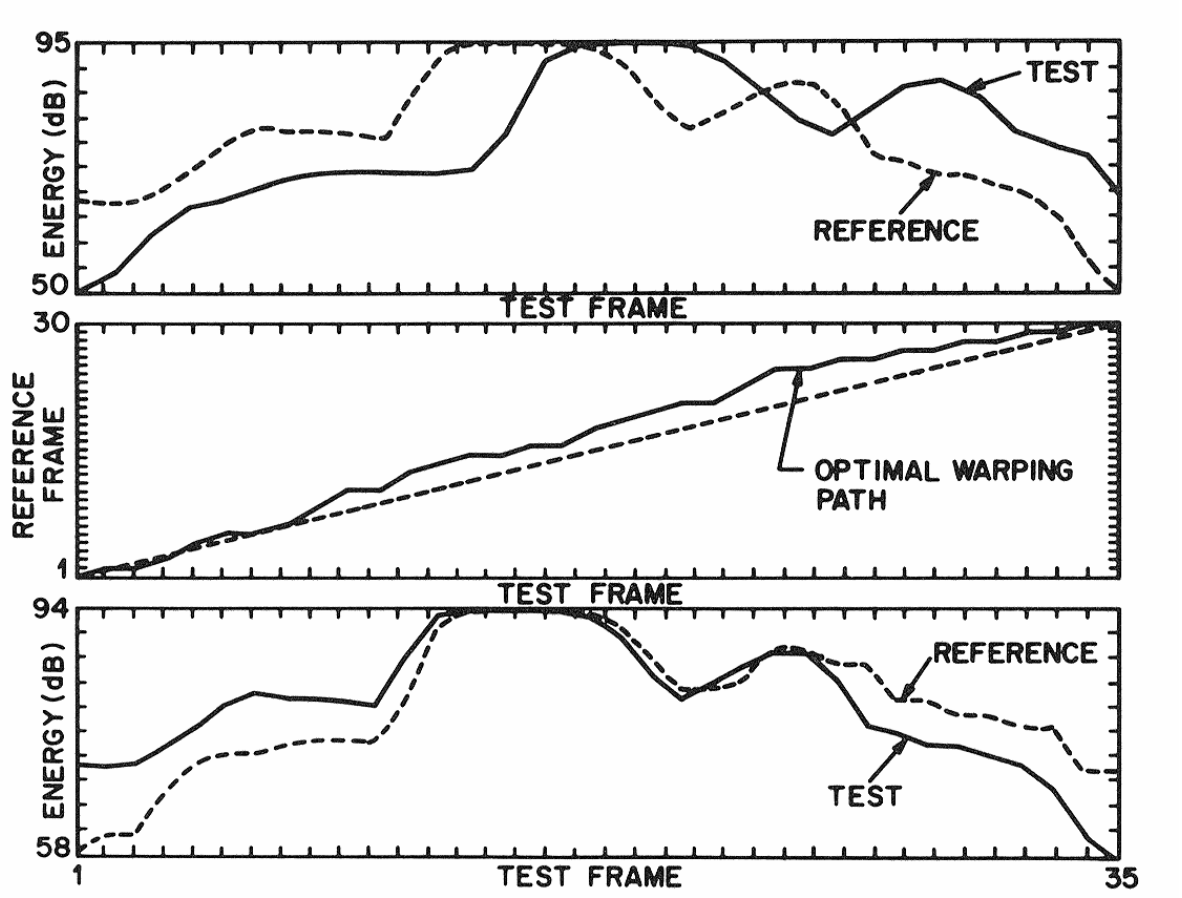




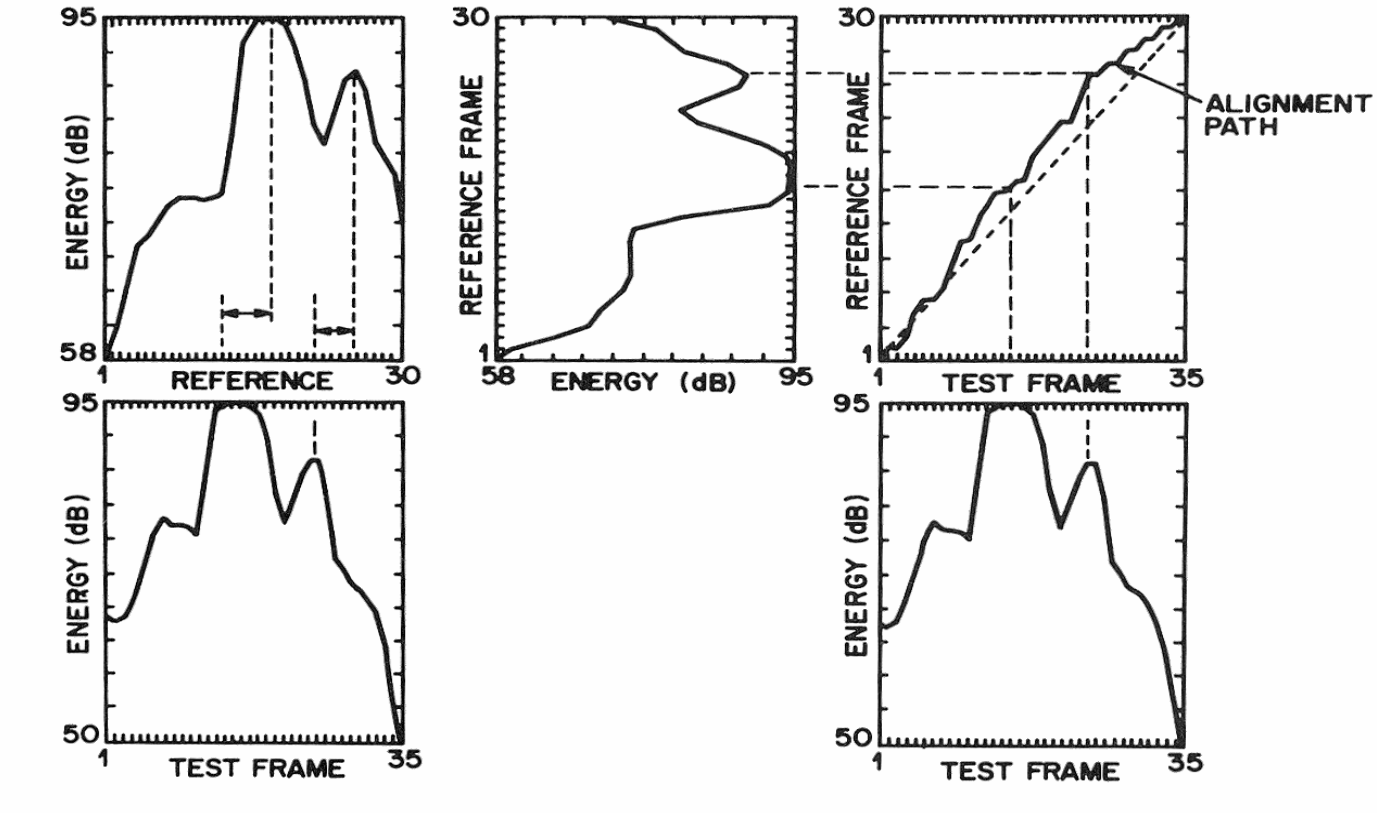
De forma genérica, podemos presentar la función de alineamiento a través de una representación del tipo siguiente:



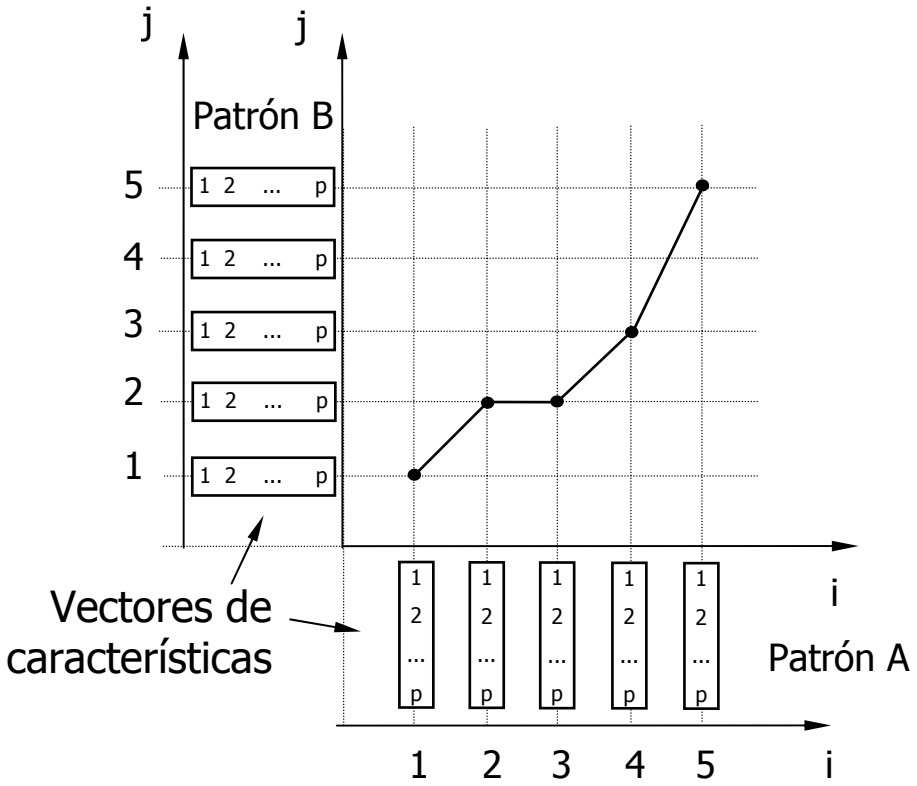
La figura muestra dos realizaciones desalineadas, la función de alineamiento correspondiente y el efecto de alineamiento sobre las locuciones anteriores:



En esta otra figura, se observa con detalle el alineamiento entre dos locuciones, haciendo hincapié en la correspondencia precisa entre los dos máximos relativos encontrados en las mismas:



Si observamos la función de alineamiento en detalle, tendremos que:



$\Rightarrow$ si la función de alineamiento pasa por (i,j), comparamos el vector 'i' de A con la 'j' de B.

DTW.1. Cálculo de la función de alineamiento

Nuestro problema consistirá en encontrar la función de alineamiento a partir de los dos vectores de características (patrones) 'A' y 'B'.

Si tenemos: $A = \{ a_1, a_2, .. , a_i, .. , a_M \}$

$B = \{ b_1, b_2, .. , b_j, .. , a_N \}$

Llamando 'C' a la función de alineamiento:

$C = \{ c(1), c(2), .. , c(k), .. , c(K) \}$

donde c(k) es un par de punteros a los elementos a comparar:

$$c(k) = [\, i(k), j(k) \,]$$

Para cada c(k) tenemos una función de coste:

$$d\{ c(k) \} = \delta (a_{i(k)}, b_{j(k)})$$

que refleja la discrepancia entre los elementos comparados. Una función de coste típica es la distancia euclídea:

$$d\{ c(k) \} = (a_{i(k)} - b_{j(k)})^2$$

$\Rightarrow$ la <u>función de alineamiento</u> será aquella que minimice la función de coste total:

$$D(C) = \sum_{k=1}^{K} d\{c(k)\}$$

- <u>Las LIMITACIONES que debe cumplir dicha función serán:</u>

1.- La función de alineamiento debe ser monótona creciente:

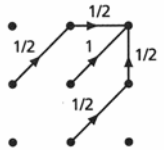
$i(k) \geq i(k-1)$ y $j(k) \geq j(k-1)$

2.- La f. debe alinear los puntos finales de A y B:

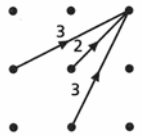
$i(1) = j(1) = 1$ y $i(K) = M, j(K) = N$

3.- La f. debe cumplir unos límites 'locales', como los siguientes:

$$\min \left\{ \begin{array}{l} D(i_x-1,i_y)+d(i_x,i_y), \\ D(i_x-1,i_y-1)+2d(i_x,i_y), \\ D(i_x,i_y-1)+d(i_x,i_y) \end{array} \right\}$$

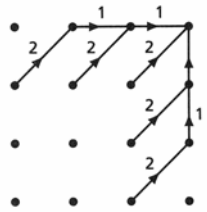


$$\min \left\{ \begin{array}{l} D(i_x-2,i_y-1)+\frac{1}{2}[d(i_x-1,i_y)+d(i_x,i_y)], \\ D(i_x-1,i_y-1)+d(i_x,i_y), \\ D(i_x-1,i_y-2)+\frac{1}{2}[d(i_x,i_y-1)+d(i_x,i_y)] \end{array} \right\}$$

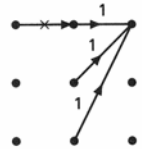


$$\min \left\{ \begin{array}{l} D(i_x-2,i_y-1)+3d(i_x,i_y), \\ D(i_x-1,i_y-1)+2d(i_x,i_y), \\ D(i_x-1,i_y-2)+3d(i_x,i_y), \end{array} \right\}$$

$$\min \left\{ \begin{array}{l} D(i_x-2,i_y-1)+d(i_x-1,i_y)+d(i_x,i_y), \\ D(i_x-2,i_y-2)+d(i_x-1,i_y)+d(i_x,i_y), \\ D(i_x-1,i_y-1)+d(i_x,i_y), \\ D(i_x-1,i_y-2)+d(i_x,i_y), \end{array} \right\}$$



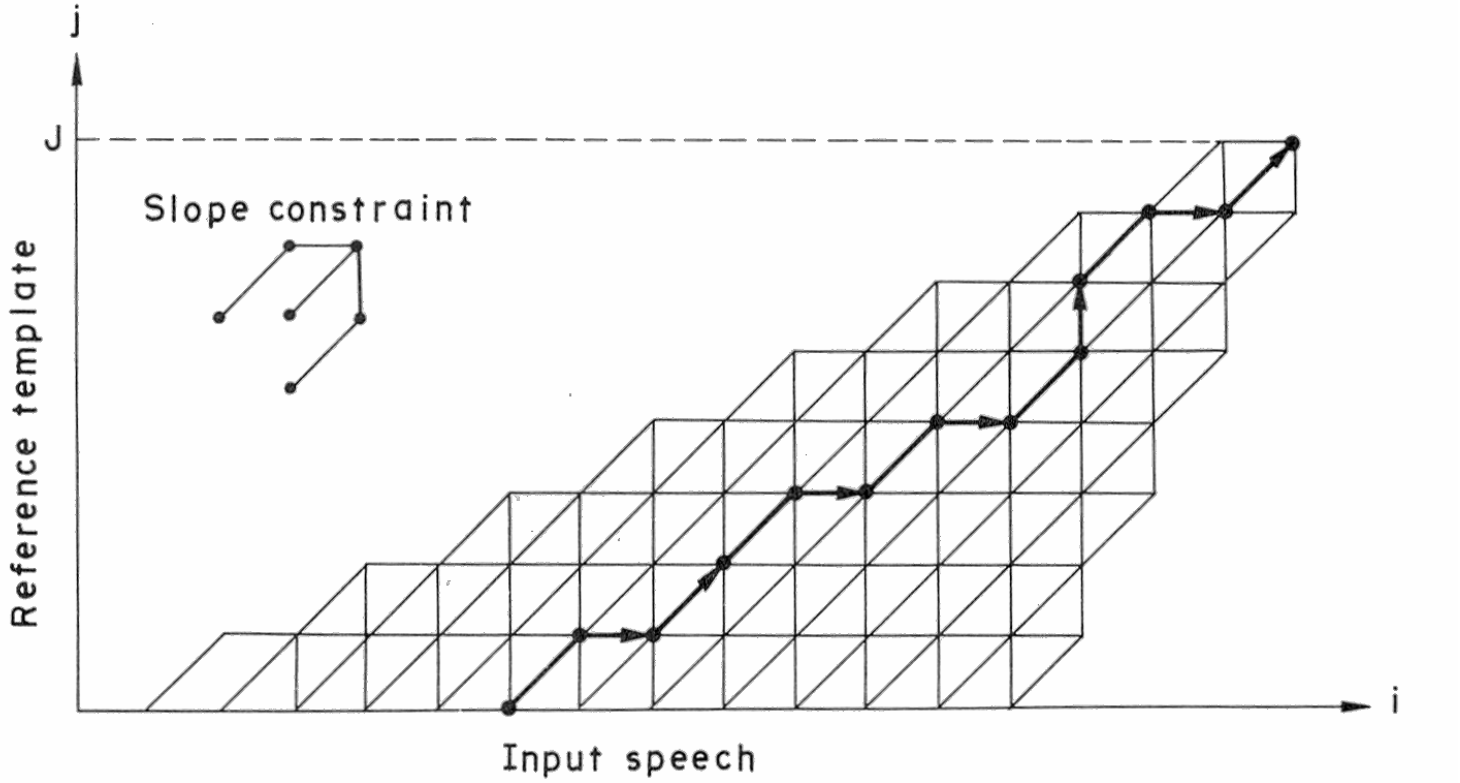
$$\min \left\{ \begin{array}{l} D(i_x-3,i_y-1)+2d(i_x-2,i_y)+d(i_x-1,i_y)+d(i_x,i_y), \\ D(i_x-1,i_y-1)+2d(i_x-1,i_y)+d(i_x,i_y), \\ D(i_x-1,i_y-1)+2d(i_x,i_y), \\ D(i_x-1,i_y-2)+2d(i_x,i_y-1)+d(i_x,i_y), \\ D(i_x-1,i_y-2)+2d(i_x,i_y-1)+d(i_x,i_y), \\ D(i_x-1,i_y-3)+2d(i_x,i_y-2)+d(i_x,i_y-1)+d(i_x,i_y), \end{array} \right\}$$



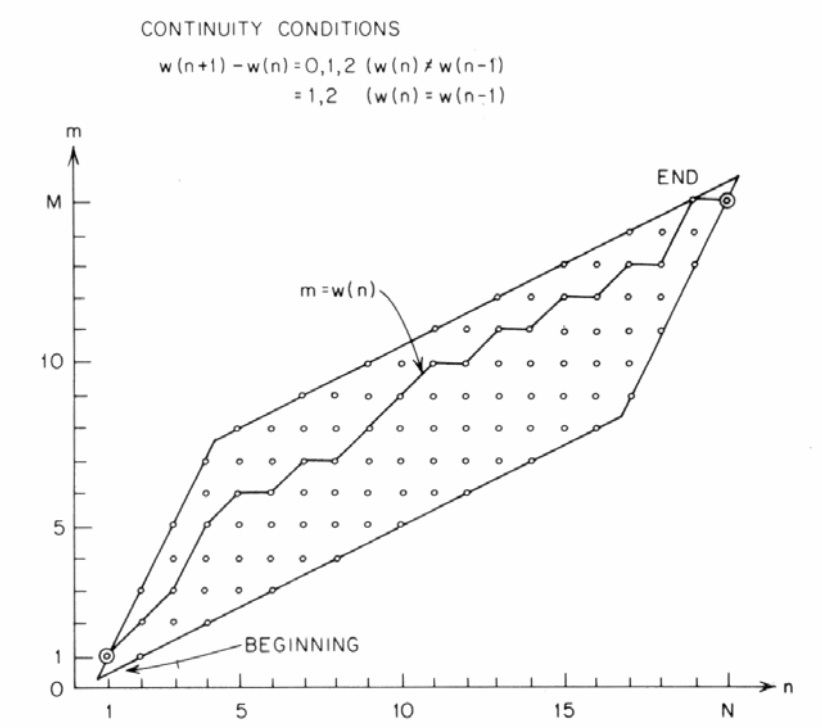
$$\min \left\{ \begin{array}{l} D(i_x-1,i_y)g(k)+d(i_x,i_y), \\ D(i_x-1,i_y-1)+d(i_x,i_y), \\ D(i_x-1,i_y-2)+d(i_x,i_y), \end{array} \right\}$$

$$\text{with} \quad g(k)=\left\{ \begin{array}{ll} 1 & \phi(k-1)\neq\phi_y(k-2) \\ \infty & \phi(k-1)=\phi_y(k-2) \end{array} \right.$$

Estos límites locales dan lugar a la denominada 'restricción de pendiente'. A continuación se muestra un caso particular:



4.- Generalmente existe algún tipo de límite global en la máxima cantidad de alineamiento. Esto se puede expresar como $|i(k) - j(k)| < Q$, lo que da lugar a una restricción denominada de 'ventana' o rombo, siendo Q el ancho de la ventana:

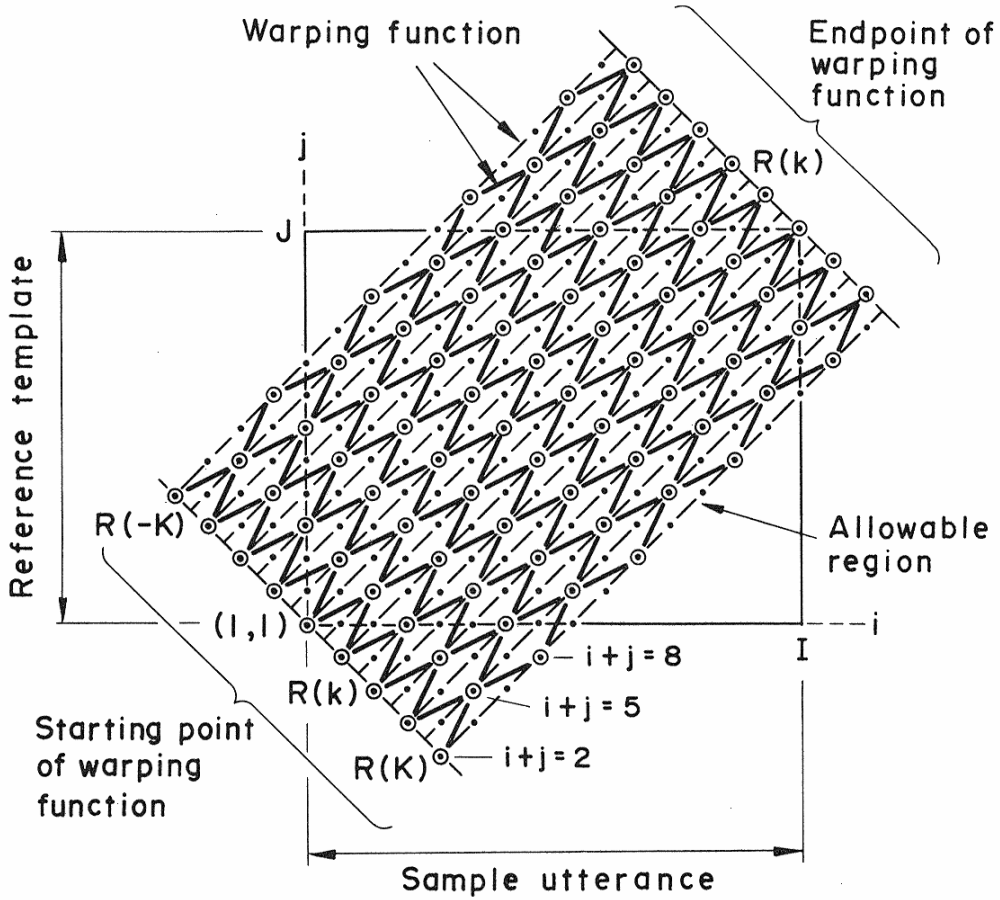


DTW.2. El algoritmo de Programación Dinámica

El problema de calcular la función de alineamiento se resuelve mediante técnicas de programación dinámica, motivo por el que se conoce a esta técnica como alineamiento temporal dinámico.

Una posible idea para la resolución del problema estribaría en el cálculo de D(C) por todos los caminos posibles, eligiendo posteriormente el de coste mínimo, si bien esto resulta impracticable a efectos de cálculo.

La figura siguiente muestra la multiplicidad posible para la función de alineamiento para unos determinados límites locales y una restricción de ventana dada:



La programación dinámica nos dice que el mejor camino entre (1,1) y cualquier punto (i,j) es independiente de lo que suceda mas allá de este punto.

Por tanto, el coste total del punto [i(k),j(k)] es el propio de este punto más el camino más 'barato' hasta él:

$$D(C_k) = d\{c(k)\} + \min_{\text{legal } c(k-1)} \{ D(C_{k-1}) \}$$

donde 'legal c(k-1)' representa todos los predecesores permitidos de c(k).

Sin embargo, por la limitación local tenemos un conjunto muy limitado de predecesores posibles.

Podemos resumir el algoritmo de programación dinámica como:

1.- Inicialización

$D(C_1) = d\{ c(1) \}$

2.- Recorremos el índice 'i':

para cada i desde i=1 hasta i=M

para cada j permitido

Calculo y guardo la distancia acumulada según (dtw.1) y la posición del predecesor

fin para j

fin para i

3.- Final

$d(A,B) = D(C_K) / K$ $\quad$ $c(K) = [i(M), j(N)]$

Para implementar el algoritmo sólo necesito disponer de dos arrays adicionales:

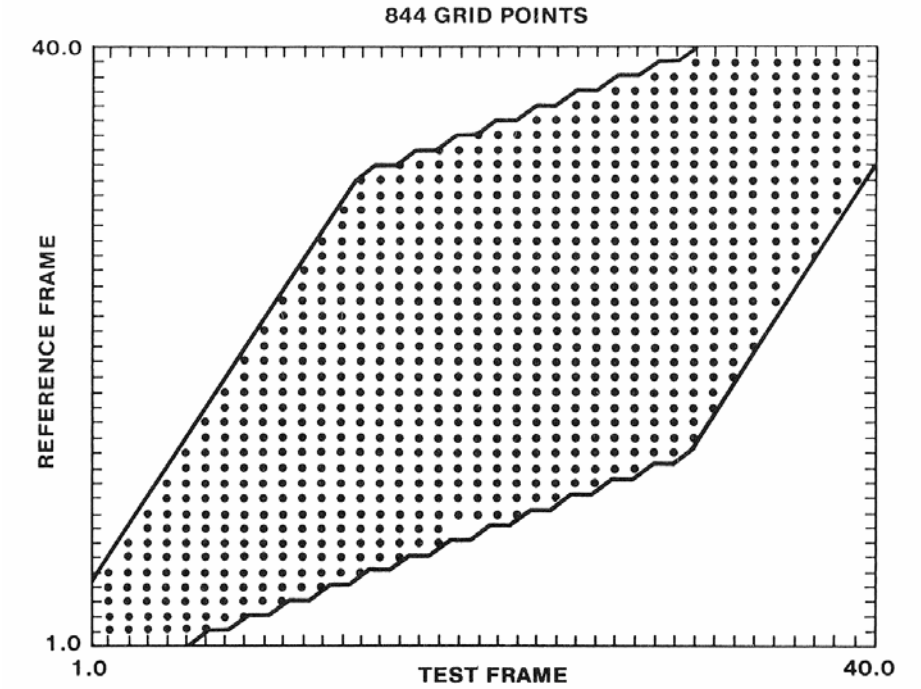
- Uno de tamaño M x N para guardar el predecesor de cada punto
- Otro de tamaño 2 x N para guardar los costes acumulados de la columna anterior y los de la presente, actualizándolos en cada paso de i

Si representamos gráficamente el predecesor de cada punto y recorremos hacia atrás (‘backtracking’) desde el punto final (M,N) hasta el inicial (1,1) obtenemos el camino de alineamiento.

fig. 11.5, pag 301, VSP

DTW.3. Extensiones del DTW

El hecho de exigir que el camino de alineamiento comience en (1,1) y termine en (M,N) hace muy dependiente al sistema de la detección de voz previa al sistema de reconocimiento, por lo que podremos 'relajar' las restricciones de punto inicial y final. La figura muestra una restricción inicial de 5 tramas y una final de 9:



Ejemplo de cálculo: Dada la siguiente tabla de distancias entre las seis tramas correspondientes al patrón de test (i) y al patrón de referencia (j):

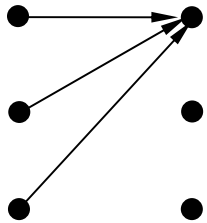
Tabla de d(i, j)

6	3.2	2.5	3.9	2.2	2.1	2.6
5	3.8	2.1	3.0	1.3	1.1	1.7
4	2.3	2.5	2.6	1.4	2.7	2.1
3	2.1	1.2	2.0	2.2	1.4	3.0
2	1.9	1.1	1.3	1.2	2.8	3.1
1	1.5	1.6	3.3	3.9	3.3	3.6
	1	2	3	4	5	6

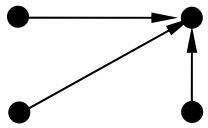
j

i

Considérense dos posibles **<u>restricciones locales</u>**, a saber:



(a)



(b)

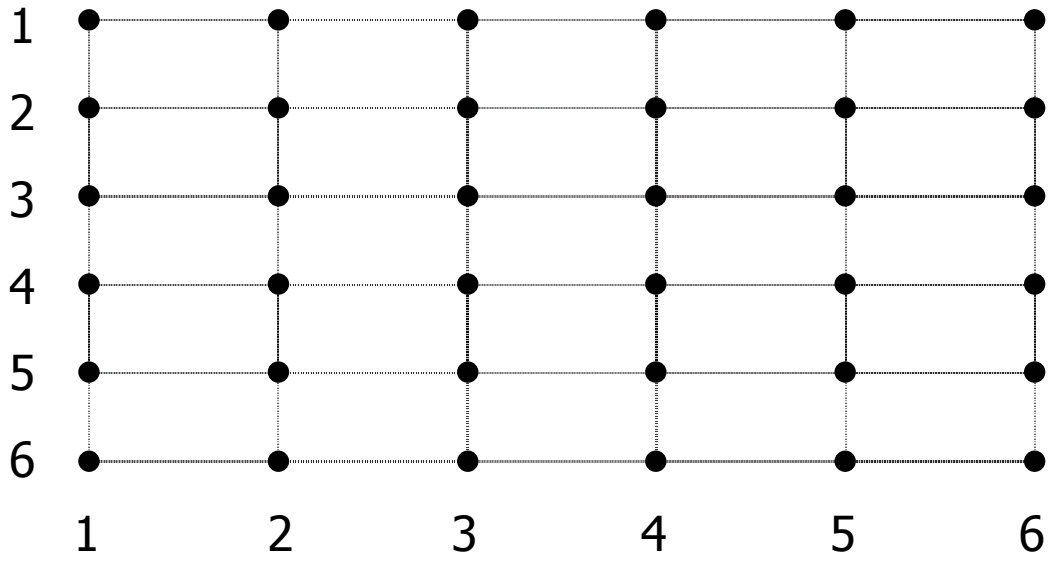
Calcular:

1) La distancia total y el camino de alineamiento comenzando en (1,1) y terminando en (6,6)

2) Ídem, comenzando en (1,1), (1,2) ó (1,3) y terminando en (6,4), (6,5) ó (6,6)

Hoja de cálculo para el ejemplo:

Caso 1):



Caso 2):