SEGUIMIENTO DE AUDIO MEDIANTE UN ÍNDICE DE PROXIMIDAD

TESIS

Que para obtener el grado de

MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA

presenta

Luis Fernando Guzmán Nateras

José Antonio Camarena Ibarrola

Director de Tesis

Universidad Michoacana de San Nicolás de Hidalgo

Abril 2014

Resumen

El Seguimiento de Audio (*AF*, del Inglés *Audio Following*) es un proceso mediante el cual se mapea la interpretación de una pieza musical realizada por un músico, usualmente en tiempo real, con una interpretación base utilizada como referencia. Esta interpretación base se considera como "bien interpretada" y es por ello que la interpretación en vivo debería alinearse con ella.

Este tipo de seguimiento tiene muchas aplicaciones como pueden ser el acompañamiento automatizado, la sincronización de efectos visuales o de audio y la enseñanza, entre otros.

Anteriormente se han utilizado diversos esquemas para implementar los sistemas de seguimiento: inicialmente se utilizó programación dinámica, posteriormente funciones de densidad de probabilidad, filtro de Kalman y, más recientemente, Modelos Ocultos de Markov (HMM, del Inglés *Hidden Markov Model*). Usualmente se ha utilizado un enfoque de nota-a-nota y la alienación se realiza con una representación simbólica de la partitura. A dicho enfoque se le denomina Seguimiento de Partitura (*SF*, del Inglés *Score Following*).

En este trabajo se propone el uso de un índice de proximidad para realizar el seguimiento. En particular se utiliza un índice aproximado basado en *Hashing* Sensible a la Localidad (LSH) en conjunto con una firma de audio (AFP, del Inglés *Audio Fingerprint*) que usa como característica de extracción la entropía de la señal. A este enfoque se le denominó *Seguimiento de Audio mediante Índices* (IAF, del Inglés *Index Audio Following*). Para efectuar el seguimiento se extrae la huella de audio de una interpretación que después es indexada usando el índice LSH, esta interpretación se utilizará como interpretación de referencia. Posteriormente, se toman subfirmas equivalentes a medio segundo de audio de una segunda interpretación y se busca su posición correspondiente en la firma de la interpretación de referencia utilizando el índice creado previamente. Así, el enfoque propuesto no utiliza un enfoque de nota a nota puesto que no se mapea a la partitura de la interpretación. En su lugar, se utiliza la firma de audio de una interpretación como referencia para la alineación. Para validar el desempeño se propone un esquema novedoso que consiste en la creación de huellas de audio sintéticas, las cuales son modificadas en puntos específicos conocidos con lo cual se conoce de antemano el comportamiento esperado del seguimiento. Se obtuvieron resultados satisfactorios de seguimiento en los experimentos realizados utilizan-

do búsqueda secuencial con el algoritmo propuesto. Al utilizar el índice LSH, se logró una reducción de entre el 60 % y el 85 % de las comparaciones entre firmas realizadas respecto al número de comparaciones hechas al utilizar la búsqueda secuencial, siempre manteniendo un comportamiento equivalente al obtenido con búsqueda secuencial.

Abstract

Audio Following (AF) is the process of mapping a musician's live performace, usually in realtime, to a base performace that is used as a reference. Such base performace is considered a "correct performace" and thus, the live performace has to be aligned to it.

Audio Following is very usefull in practice as in automatic accompaniment, visual and audio effects synchronization, teaching environments, among others.

Previously, several approaches have been used to implement AF systems: the first approach was dynamic programming, then probability density functions were used, Kalman filter and, more recently, Hidden Markov Models (HMM). Furthermore, a note-by-note approach is frequently used and the mapping is performed with the corresponding musical score which has been previously stored in a computer using a symbolic representation. This approach is known as Score Following (SF).

This project discusses the use of a proximity index to perform audio following. In particular, we use an approximate search index based on *Locality Sensitive Hashing* (LSH) in conjuction with an audio fingerprint (AFP) that uses the signal's entropy as a feature for extraction. I refer to this concept as *Index Audio Following* (IAF). To perform audio following, first the audio fingerprint of a performance is extracted. Then, it is indexed using the LSH index, this performance's AFP will be used as a reference to align any other performance of the same music. Afterwards, half-a-second subAFPs of another performance are extracted and their correspondig positions in the reference AFP are searched for using the LSH index. Thus, the proposed schema does not use a note-by-note approach due to the fact that the mapping is not performed againts the performance's score. Instead, the audio fingerprint of a performance is used as a reference for the alignment.

Audio following validation is performed using a proposed novel method that consists of the creation of synthetic AFPs that are modified in specific known points so that the expected following behavior is known beforehand. Satisfactory results were obtained with the proposed algorithm when a sequential search was used. When the LSH index was implemented and tested, a reduction of 60 % to 85 % in the number of comparisons needed between AFPs with respect to the number of comparisons performed using a sequential search, while maintaining an equivalent following to the one obtained through sequential search.

Contenido

Lista de Figuras

Lista de Tablas

Lista de Algoritmos

Lista de Símbolos

AF	Seguimiento de Audio, *Audio Following*.
SF	Seguimiento de Partitura, *Score Following*.
AA	Alineación de Audio, *Audio Alignment*.
AFP	Huella de Audio, *Audio Fingerprint*.
$MIDI$	Interfaz Digital para Instrumentos Musicales. *Musical Instrument Digital Interface*.
$MFCC$	Coeficientes Cepstrales de Frecuencia de Mel. *Mel Frequency Cepstral Coefficients*.
DFT	Transformada de Fourier Discreta. *Discrete Fourier Transform*.
MFT	Transformada a la Frecuencia de Mel. *Mel Frequency Transform*.
LSH	Hashing Sensible a la Localidad. *Locality Sensitive Hashing*.
HMM	Modelo Oculto de Markov. *Hidden Markov Model*.
$HHMM$	Modelo Oculto de Markov Jerárquico. *Hierarchichal Hidden Markov Model*.
PDF	Función de Densidad de Probabilidad. *Probability Density Function*.
DP	Programación Dinámica. *Dynamic Programming*.
DTW	Doblado Dinámico del Tiempo. *Dynamic Time Warping*.
OTW	Doblado En Línea del Tiempo. *On − line Time Warping*.
IAF	Seguimiento de Audio mediante Índices. *Index Audio Following*.
$\mathbb{R}^b$	Espacio vectorial de dimensión b.
$H^{b'}$	Espacio métrico de Hamming de dimensión b'.
$d(x, y)$	Distancia entre los objetos x y y.
b	Dimensión del espacio.
l	Número de instancias LSH.
n	Número de bits/posiciones utilizados para crear el hash de un vector.
p, q	Elementos de consulta
T	Conjunto de tablas hash.

s_f	Subfirma a buscar.
$hash()$	Función hash.
h	Valor hash calculado.
k	Número de vecinos más cercanos.
knn	Lista de vecinos más cercanos.
$distanciasKnn$	Lista de distancias a los vecinos más cercanos.
p_c	Posición candidato, *candidate position*.
p_s	Posición inicial, *starting position*.
p_p	Posición previa o *previous position*.
p_n	Posición siguiente o *next position*.
LCS	Subsecuencia común más larga, *Longest Common Subsequence*.
p_{np}	Posición siguiente probable o *probable next position*.
p_{pp}	Posición previa a la anterior o *previous to previous position*.
p_{pred}	Predicción de posición siguiente o *next position prediction*.
p_{nt}	Posición siguiente en tiempo
p_{nd}	Posición siguiente en distancia.
d_t	Diferencia en tiempo, distancia reportada por la subfirma de p_{nt}.
d_d	Diferencia en distancia, distancia reportada por la subfirma de p_{nd}.
wmd	Parámetro $windowMultiplierDefault$.
n_b	Parámetro $numberOfBands$.
hm_b	Parámetro $howManyBits$.
hm_m	Parámetro $howManyMaps$.
b_v	Parámetro $bitVariations$.
s	segundos.
ms	milisegundos.

Capítulo 1

Introducción

El procesamiento digital de señales en conjunto con técnicas de búsqueda y recuperación de información, han adquirido mucha importancia en la actualidad debido a la gran cantidad de problemas en los que resulta conveniente su utilización.

Uno de dichos problemas es el Seguimiento de Audio (AF, del Inglés *Audio Following*) el cual consiste básicamente en la alineación de una interpretación en vivo realizada por un músico con una interpretación utilizada como referencia. Un problema similar al AF conocido como Seguimiento de Partitura (SF, del Inglés *Score Following*) ha sido abordado de numerosas maneras en el pasado [Dannenberg, 1984], [Puckette y Lippe, 1992], [Cano *et al.*, 1999], [Orio y Déchelle, 2001], [Orio y Schwarz, 2001], [Dixon, 2005], [Cont, 2006], [Jordanus, 2007], [Chou *et al.*, 2012] con buenos resultados.

En este capítulo se presenta una explicación de los problemas AF y SF, una descripción de los diferentes enfoques que se han utilizado para abordarlos previamente y, por último, el esquema propuesto durante este trabajo.

1.1. Planteamiento del Problema

En esta sección se brinda una explicación de dos variantes utilizadas para abordar el problema de seguimiento de audio, las diferencias y similitudes entre ellos, sus aplicaciones y retos. Así mismo, se describe el esquema de seguimiento planteado para este trabajo.

1.1.1. Seguimiento de Partitura

SF es la sincronización en tiempo real de un modelo computacional, como puede ser un HMM [Rabiner, 1989], con un músico que interpreta una partitura musical determinada previamente. El principio detrás del *SF* es bastante simple: introducir una partitura musical a una computadora de alguna forma, y después tocar dicha partitura utilizando la interpretación como una entrada en tiempo real para la computadora, ya sea mediante una interfaz MIDI (Interfaz Digital para Instrumentos Musicales, *Musical Instrument Digital Interface*) o con un micrófono realizando algún tipo de análisis acústico. La meta del sistema es entonces el mapear la secuencia entrante de audio con la representación almacenada de la partitura y determinar en tiempo real la posición del intérprete, usualmente utilizando un enfoque de nota a nota. En la figura 1.1 se aprecia el funcionamiento de SF, se toma un segmento de audio de la interpretación musical y se busca la posición correspondiente en la partitura.

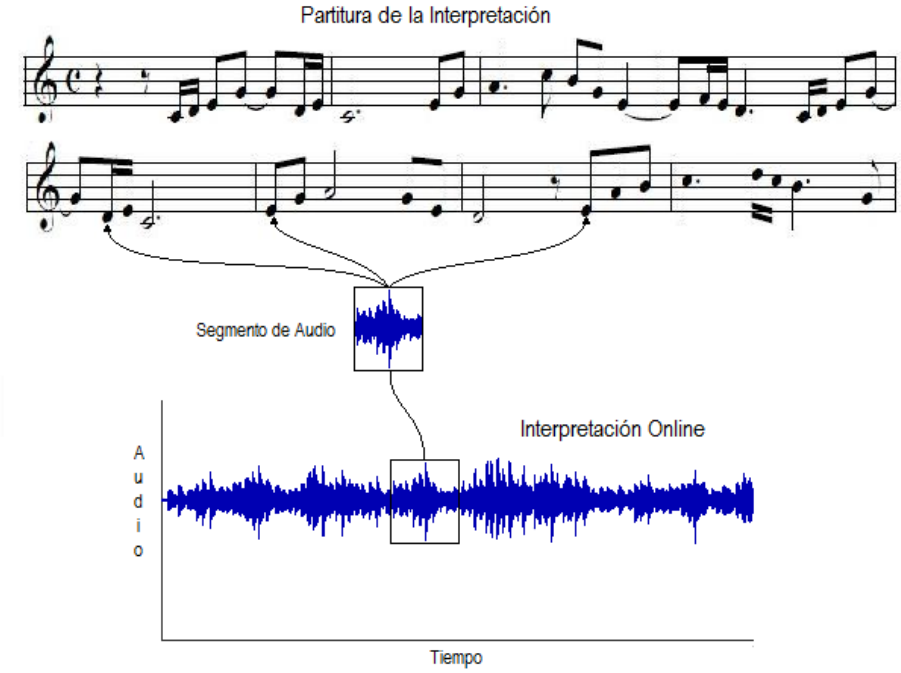


Figura 1.1: Ejemplo de SF.

El *SF* tiene muchas aplicaciones potenciales: se puede utilizar para complemen-

tar la experiencia de los asistentes a un concierto (aplicando efectos de sonido en ciertas secciones, aumentos o disminuciones en el volumen, efectos visuales, pirotecnia, etc.), en contextos de enseñanza/aprendizaje dando retroalimentación a los maestros y estudiantes y sistemas interactivos, entre otros. En resumen, ejecutando eventos electrónicos en puntos precisos de la interpretación.

Uno de los usos más comunes del SF es con el propósito de proveer acompañamiento automatizado para un intérprete en vivo. El acompañamiento en tiempo real resuelve la mayoría de los problemas de sincronización que son inherentes en el acompañamiento grabado. Sin embargo, surgen entonces tres subproblemas: detectar y procesar las entradas producidas por el intérprete, alinear dichas entradas contra la partitura adecuada y generar la información del ritmo necesaria para controlar la generación del acompañamiento.

También es de esperarse que la interpretación en vivo contenga errores que pueden afectar la alineación con la partitura esperada. Estos errores pueden ser de dos tipos:

1. **Accidentales:** notas tocadas incorrectamente, notas extra, notas no interpretadas o saltadas, un punto de inicio equivocado, ritmo equivocado, etc.

2. **Provocados:** cambios intencionales, ya sea en el ritmo o en la propia interpretación (saltos, notas adicionales), realizados por el intérprete.

A lo largo de los últimos 30 años se han desarrollado sistemas de acompañamiento automático, todos los cuales utilizan alguna forma de SF [Dannenberg, 1984], [Dannenberg y Grubb, 1989], [Desain *et al.*, 1997], [Dixon, 2005], [vanKasteren, 2006], [Jordanus, 2007],[Cont, 2010], [Chou *et al.*, 2012].

1.1.2. Alineación de Audio y Seguimiento de Audio

La Alineación de Audio (AA, del Inglés *Audio Alignment*) y el Seguimiento de Audio (AF, del Inglés *Audio Following*) son problemas hermanos del SF. Ambos consisten en la alineación de dos interpretaciones musicales entre sí en lugar de alinear una interpretación con su partitura como se hace en SF. En general, el AA es el proceso de obtener puntos correspondientes en el tiempo entre dos interpretaciones iguales o con contenidos similares.

En *AA* se conocen de antemano ambas interpretaciones en su totalidad y la alineación se realiza de manera offline. Por otro lado, en *AF* se conoce una de las interpretaciones en su totalidad y se utiliza como referencia, también llamada interpretación base; mientras que la segunda interpretación, llamada interpretación online, se obtiene paulatinamente. La alineación entre ambas interpretaciones se realiza usualmente en tiempo real conforme se recibe como entrada los datos de la segunda interpretación. En resumen, la diferencia fundamental entre *AA* y *AF* consiste en que en *AA* no se tienen las restricciones adicionales de tiempo real y conocimiento parcial de una de las interpretaciones que usualmente se manejan en el *AF*. Los resultados de la alineación offline generado por los sistemas de *AA* se pueden utilizar como un medio de entrenamiento o comparación de los sistemas en tiempo real *AF* para validar su desempeño.

Los sistemas *AF* comparten la mayoría de las aplicaciones y problemas mencionados en la Sección 1.1.1 para los sistemas de *SF* con la diferencia evidente de que la alineación se realiza con una interpretación de referencia en lugar de con la partitura. En la figura 1.2 se observa el funcionamiento del AF, se toma un segmento de audio de la interpretación online y se busca su posición correspondiente en la interpretación de referencia.

1.1.3. Esquema de Seguimiento Propuesto

En este trabajo se plantea la implementación de un sistema de *AF*. Utilizando un par de interpretaciones de una misma pieza musical, se utilizará una de ellas como interpretación de referencia y la otra como interpretación online. Se obtendrá un segmento de la interpretación online y se buscará su posición correspondiente dentro de la interpretación de referencia.

Para poder encontrar la posición correspondiente del segmento de audio de la interpretación online dentro de la interpretación de referencia, es necesario comparar de manera secuencial dicho segmento con todos los segmentos de audio equivalentes de la interpretación base como se muestra en la Figura 1.3. A este esquema se le conoce como búsqueda secuencial.

Para facilitar las comparaciones entre segmentos de audio, usualmente se realiza algún tipo de extracción de características de la señal de audio de manera que ésta pueda ser

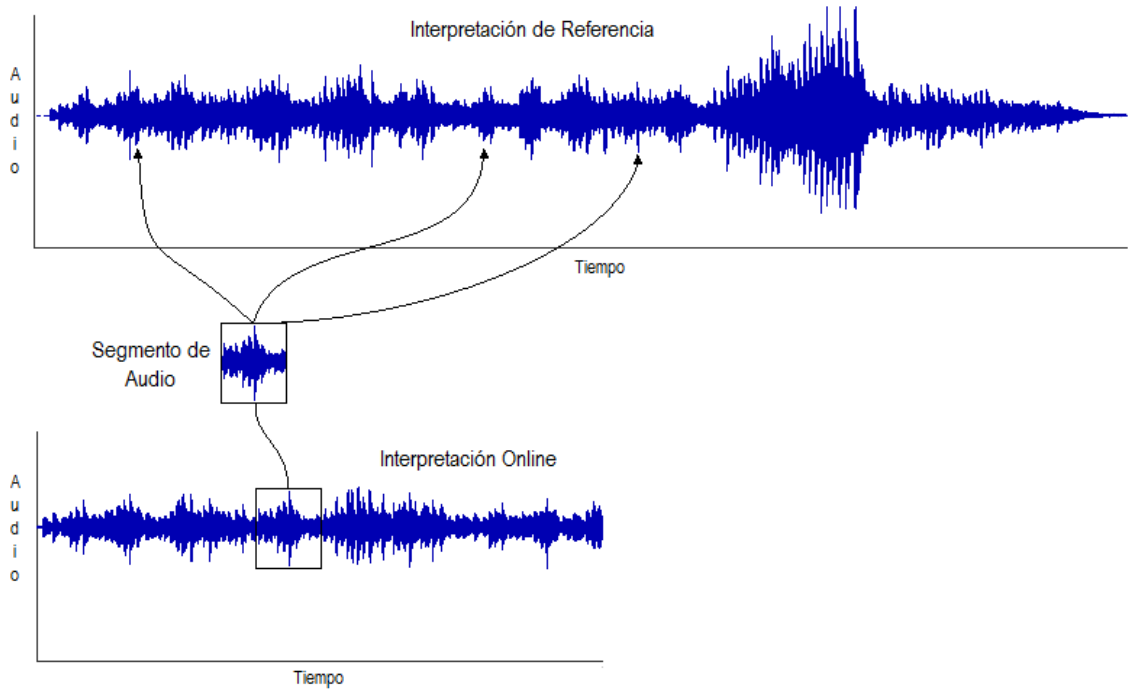


Figura 1.2: Ejemplo de AF.

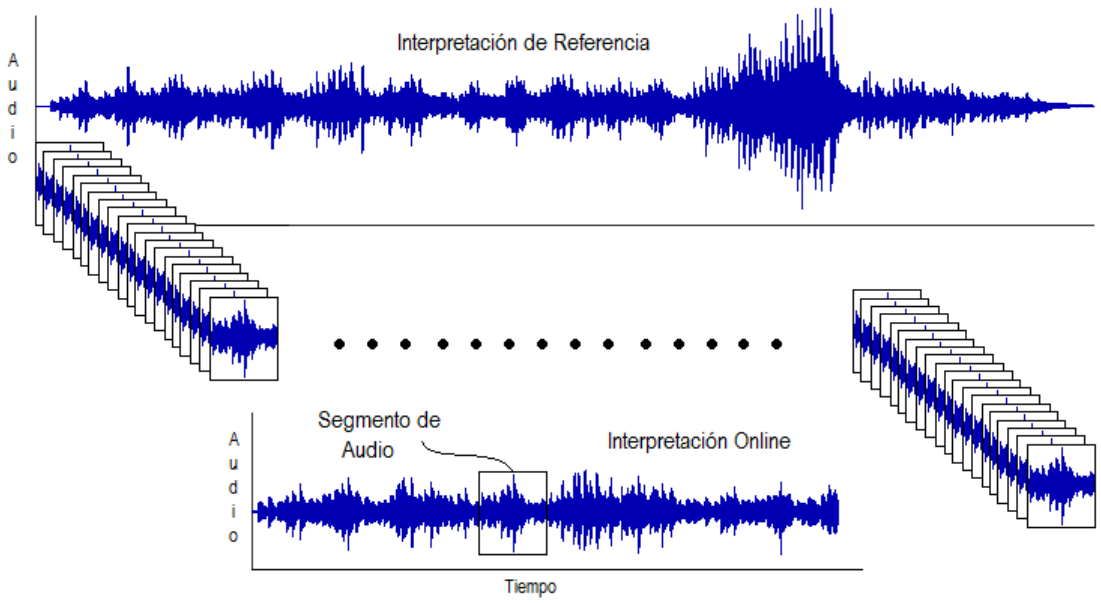


Figura 1.3: Ejemplo de búsqueda secuencial.

representada de forma adecuada por los vectores de características extraídos. A esta representación se le conoce como Firma o Huella de Audio (AFP, del Inglés *Audio Fingerprint*). Diferentes AFPs utilizan distintas características de extracción como pueden ser el volumen, tono, ritmo, nota, energía [Pohlmann, 1995], entropía [Shannon y Weaver, 1949], llanura espectral [Johnston, 1988], etc [Wold *et al.*, 1996][Kirchhoff y Lerch, 2011]. De esta manera en lugar de comparar los segmentos de audio directamente, se comparan segmentos de las AFP obtenidas a los cuales se les conoce como subfirmas de audio como se presenta en la figura 1.4.

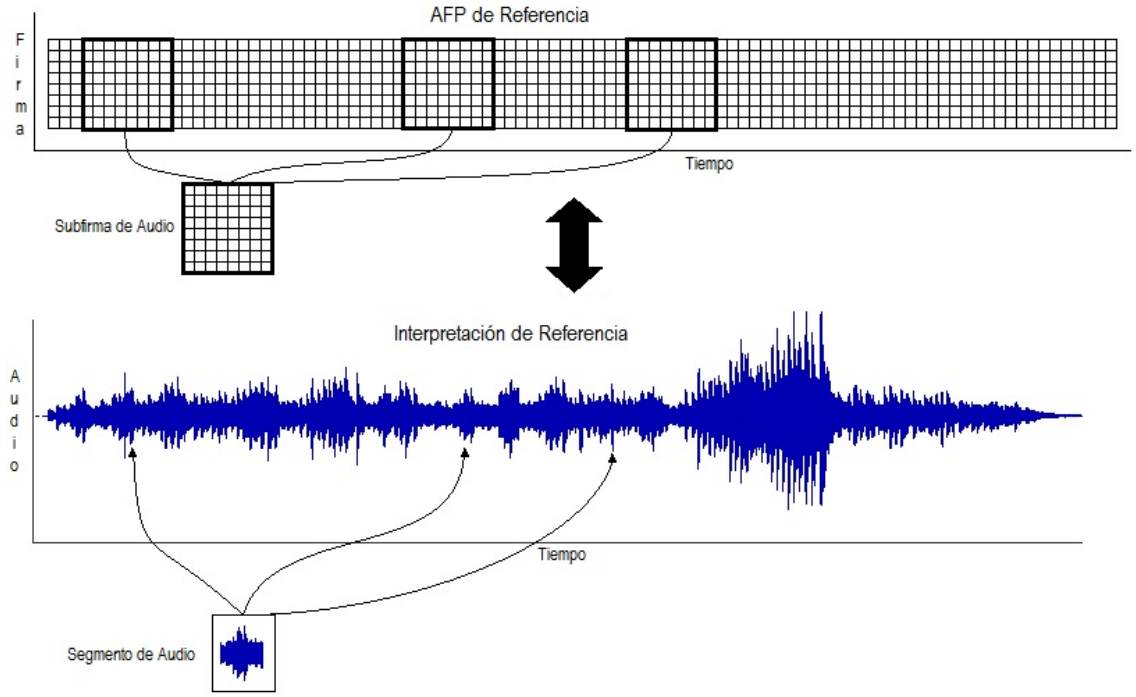


Figura 1.4: Conversión de audio a AFP.

Dependiendo de la AFP utilizada la comparación entre subfirmas se puede realizar utilizando diferentes funciones de distancia. Por ejemplo, si la AFP entrega vectores de características compuestos por números reales se puede utilizar la distancia Euclidiana [Deza y Deza, 2009] o la distancia Manhattan [Krause, 1986]. En cambio si la AFP utiliza algún tipo de codificación de los vectores, por ejemplo a cadenas de caracteres, se pueden utilizar las distancias de Hamming [Hamming, 1950] o Levenshtein [Levenshtein, 1966].

En particular, para el presente trabajo se utiliza la AFP publicada en [Camarena y Chávez, 2006], la cual utiliza como característica de extracción la entropía de la señal y realiza una codificación binaria de los vectores de características por lo cual para realizar las comparaciones entre subfirmas se utilizaron las distancias de Hamming, Levenshtein y LCS (Subsecuencia común más larga, del Inglés *Longest Common Subsequence*) [Hirschber, 1975].

Es claro que al utilizar la búsqueda secuencial descrita anteriormente se realiza una gran cantidad de comparaciones entre subfirmas, muchas de ellas innecesarias puesto que se realizan entre subfirmas muy diferentes. Para evitar el uso de la búsqueda secuencial y reducir el número de comparaciones entre subfirmas efectuadas, se propone la utilización de un índice de proximidad. La AFP de la interpretación de referencia es indexada utilizando el índice de proximidad y cuando se busca encontrar las posibles posiciones de una subfirma de la interpretación online se realiza una consulta al índice, la cual entregará como respuesta únicamente las posiciones de las subfirmas de la interpretación de referencia que sean más similares a la subfirma de consulta. Lo anterior se ejemplifica en la Figura 1.5

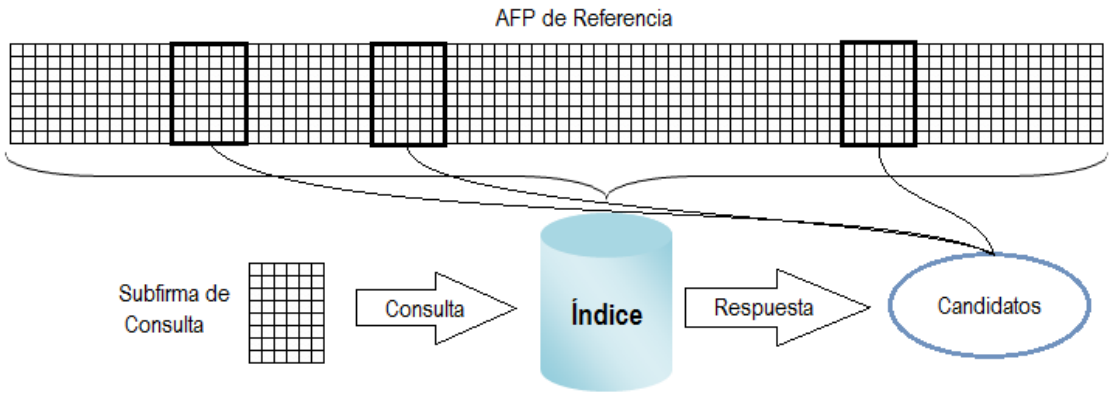


Figura 1.5: Consulta de una subfirma en el índice.

Como en el caso de las AFPs, también existen distintos índices de proximidad que se pueden utilizar. Para este trabajo, debido a las características de los vectores entregados por la AFP elegida, se decidió utilizar el índice de proximidad basado en *Hashing* Sensible a la Localidad (LSH, del Inglés *Locality Sensitive Hashing*) presentado por [Gionis *et al.*,

1999].

Puesto que el esquema utilizado por el presente trabajo es distinto al utilizado por la mayoría de los trabajos relacionados previamente (no se utiliza un enfoque de nota-a-nota) es difícil efectuar una comparación, por ello, para validar el desempeño del seguidor implementado se propone la utilización de un esquema novedoso el cual consiste en la creación de firmas de audio sintéticas. Estas firmas de audio sintéticas son modificadas en puntos específicos conocidos, de manera que se conoce de antemano el comportamiento esperado del seguimiento y es posible comparar el seguimiento obtenido con el esperado. Adicionalmente, se obtuvieron manualmente las mediciones de los momentos de ejecución de las notas (*note onset*) de algunas de las interpretaciones utilizadas durante los experimentos y de esta manera se puede conocer la alineación real de dichas interpretaciones y compararla con la alineación obtenida por el seguidor.

Al enfoque presentado en este trabajo, en el cual se utiliza un índice de proximidad para realizar la búsqueda se le denominó Seguimiento de Audio mediante un Índice de Proximidad (IAF, del Inglés $IndexAudioFollowing$) para diferenciarlo de otras versiones de AF.

1.2. Antecedentes y Estado del Arte

El problema de SF no es algo nuevo, fue presentado por primera vez en la Conferencia Internacional de Música por Computadora ($ICMC$, $International\ Computer\ Music\ Conference$) de 1984 por Barry Vercoe y Roger Dannenburg de manera independiente. En un inicio, el objetivo fue poder proveer a los intérpretes con un sistema capaz de realizar acompañamiento automático.

En [Vercoe, 1984] se describe el área de investigación por primera vez de la siguiente manera: "...entender la dinámica de una interpretación en conjunto en vivo lo suficientemente bien como para poder reemplazar a cualquier miembro del grupo con un intérprete sintético (por ejemplo, un modelo de computadora) de manera que el resto de los integrantes no puedan notar la diferencia...". En la implementación de Vercoe, se emplearon interpretaciones del flautista profesional Larry Beauregard y se propuso la utilización del

tono (*pitch*) como característica de extracción. Vercoe menciona que para obtener y detectar una secuencia de eventos tocados por un flautista profesional es necesario realizar la detección del tono a una velocidad tal que es casi imposible para los métodos acústicos. Por ello plantea además la utilización de dos fuentes adicionales de datos: la partitura musical y el tiempo de ejecución de las notas (*onset*). Para ello se utilizó una flauta a la cual se le instalaron sensores ópticos en las teclas con el fin de obtener las mediciones tiempo de ejecución de las notas.

En [Dannenberg, 1984] se presenta un algoritmo eficiente de programación dinámica para encontrar la mejor concordancia entre una interpretación y su partitura. Este algoritmo compara la secuencia de eventos generados por el intérprete con la secuencia de eventos esperados. Para encontrar la mejor concordancia, se calcula una matriz de enteros donde cada fila corresponde a un evento en la partitura y cada columna corresponde a un evento detectado en la interpretación. Cada vez que se detecta un nuevo evento dentro de la interpretación se calcula una nueva columna. El entero calculado para la fila r y la columna c responde a la pregunta: Si nos encontráramos en el evento r-ésimo evento de la partitura y el c-ésimo evento de la interpretación, ¿cuál sería la longitud de la concordancia entre los eventos esperados y los eventos obtenidos hasta este momento? De la misma manera que el trabajo presentado en [Vercoe, 1984], el trabajo de Dannenberg es orientado a la generación de un acompañamiento automático en tiempo real. Una de las limitaciones del algoritmo es que únicamente trata con música monofónica y un sólo instrumento.

Posteriormente en [Dannenberg y Grubb, 1989] y [Grubb y Dannenberg, 1998] se describió un nuevo enfoque que utiliza funciones de densidad de probabilidad. Dada una determinada posición previa en la partitura, el algoritmo de seguimiento continuamente estima la distancia desde dicha posición. Entonces se utilizan las observaciones más recientes del tono, picos espectrales, cambios en la amplitud, etc. para ubicar la nueva posición del intérprete.

Los algoritmos de Vercoe y Dannenberg fueron extendidos para poder tratar con música polifónica e instrumentos múltiples en [Desain *et al.*, 1997].

En [Puckette y Lippe, 1992] se presenta un resumen del algoritmo usado hasta esa fecha en el Instituto de Investigación y Coordinación Acústica Musical (*IRCAM*, *Institut*

de Recherche et Coordination Acoustique/Musique) para realizar acompañamiento automático. Dicho algoritmo utiliza un enfoque de nota por nota y utiliza como característica de extracción el tono (*pitch*) [Gerhard, 2003] de la interpretación. Utiliza punteros a la "nota actual" y una lista de notas anteriores llamadas "notas saltadas" (*skipped notes*). Cuando se toca una nota, el algoritmo intenta alinearla con alguna nota en la base de datos; la nota alineada debe tener el mismo tono que la interpretada en vivo (aunque se aceptan notas con diferencia de una octava). Primero se busca una coincidencia dentro de la lista de notas saltadas, si no se encuentra ahí, la búsqueda se continúa a partir de la nota actual. Cuando se encuentra una coincidencia, se actualiza el puntero hacia la nota actual y se agregan las nuevas notas saltadas a la lista. Una vez que la última nota se ha alineado, el algoritmo termina.

En [Cano *et al.*, 1999] se propone por primera vez utilizar un modelado estocástico del problema para lidiar con los problemas de incertidumbre asociados con el uso de la voz (en la cual las señales obtenidas no asemejan para nada a una secuencia de tonos discretos en intervalos determinados) y los posibles errores cometidos por los intérpretes instrumentales. Se propone la extracción de seis diferentes características de la señal: la energía, el cambio de la energía entre un marco y otro, los cruces por cero, la frecuencia fundamental [Gerhard, 2003], el cambio de la frecuencia fundamental entre un marco y otro y el error de la frecuencia fundamental. Estas características se utilizan para crear una secuencia de observaciones la cual será introducida a un HMM y se utiliza el algoritmo de Viterbi [Rabiner, 1989] para encontrar la secuencia de estados más probable producida por el modelo. Se utilizan tres HMMs de izquierda a derecha (*left-to-rigth*) para modelar la presencia de una nota, la ausencia de nota y los silencios. El modelo utilizado para las notas cuenta con tres estados que representan el comportamiento habitual de una nota: ejecución (*attack*, *onset*), mantenimiento (*sustain*, *hold*) y fin (*release*). El modelo para los silencios únicamente cuenta con un estado puesto que los silencios no cuentan con una estructura. El modelo de ausencia de nota cuenta también con tres estados y pretende representar todos los sonidos dentro de la interpretación que no son considerados como notas, éstos pueden ser ruidos o, en el caso de la voz, las aspiraciones, los sonidos fricativos o plosivos.

En [Raphael, 1999] se trata con un subproblema del seguimiento de audio: la

segmentación de las señales acústicas. También hace uso de HMMs como modelo. Dada la partitura de una pieza de música monofónica y la grabación de una interpretación de dicha pieza, se segmenta la señal en regiones contiguas correspondientes a las notas y los descansos con el fin de poder estimar la posición de la interpretación dentro de la partitura y proveer acompañamiento automático.

En [Orio y Déchelle, 2001] se propone un esquema similar al utilizado en [Cano *et al.*, 1999] pero utiliza un HMM de dos niveles: el nivel inferior compara las características de la señal de entrada con las esperadas; el nivel superior se utiliza para modelar la interpretación como una secuencia de eventos musicales tomando en consideración los posibles errores que puede cometer el intérprete. El HMM del nivel superior utilizado para modelar la interpretación tiene dos tipos de estados: estados normales llamados *n-states* y estados fantasma *g-states*. Los estados normales corresponden a eventos interpretados correctamente mientras que los estados fantasma corresponden a los posibles errores locales entre los eventos esperados en la partitura y los obtenidos en la interpretación. Cada evento es representado por un *n-state* y un *g-state* paralelo. La topología del HMM es de izquierda a derecha debido a la precedencia temporal de los eventos. Cada *n-state* está conectado hacia los subsecuentes estados n y g. Las transiciones de los *g-states* toman en cuenta tres posibles tipos de errores: notas equivocadas, notas extra y notas saltadas. El HMM del nivel inferior se utiliza para modelar la señal de entrada. Cada estado del nivel superior se compone por un conjunto de estados del nivel inferior. Estos estados toman en cuenta, por cada evento, las características relacionadas con la ejecución, mantenimiento y posible silencio al final. Adicionalmente se presenta un algoritmo alternativo a Viterbi para determinar la secuencia de estados más probable, el cual maximiza la probabilidad de alineaciones locales en lugar de maximizar la alineación global.

En [Orio y Schwarz, 2001] se expone una metodología para realizar la alineación offline de interpretaciones musicales monofónicas y polifónicas. Se basa en el uso del Doblado Dinámico del Tiempo (DTW, del Inglés *Dynamic Time Warping*) [Rabiner y Juang, 1993] para calcular la distancia entre los marcos de la interpretación y secciones de la partitura utilizando los picos espectrales de la señal.

En [Orio *et al.*, 2003] se presenta una descripción del estado del arte hasta esa

fecha incluyendo el sistema de seguimiento de audio utilizado en el $IRCAM$, el cual, para ese entonces, ya se basaba en HMM.

En [Hu *et al.*, 2003a] y [Hu *et al.*, 2003b] se propone un esquema para buscar y alinear grabaciones de audio polifónico con las representaciones simbólicas de sus partituras guardadas en archivos MIDI. El método se basa en el uso de *Chromagramas* [Fujishima, 1999] y DTW. Los chromagramas son secuencias de vectores de *chromas*. Los vectores chromas son vectores de 12 elementos donde cada elemento representa la energía espectral correspondiente a un tono (por ejemplo, Do, Do#, Re, Re#, Mi, etc.). Se convierten los archivos MIDI a audio utilizando un sintetizador y posteriormente se convierten ambos archivos de audio a vectores de chromas. Estas secuencias se alinean utilizando DTW. Para validar su esquema de alineación proponen alinear interpretaciones musicales con archivos MIDI que no les corresponden y observar gráficamente como el camino óptimo obtenido por DTW no es una diagonal sino que tiene un comportamiento errático. Adicionalmente, introducen cambios artificiales en los archivos MIDI de manera que las alineaciones obtenidas se ven afectadas por dichos cambios.

En [Dixon, 2005] se presenta una alternativa para realizar el seguimiento de una interpretación en vivo utilizando un método conocido como *Online Time Warping* (OTW). El DTW no resulta adecuado para aplicaciones en línea debido a que tiene la limitación de requerir conocimiento por completo de ambas series de tiempo para poder calcular su alineación. *OTW* realiza alineaciones incrementales de dos series de tiempo mientras que una de ellas se recibe en tiempo real. Dicho algoritmo se aplica a la alineación de señales de audio para poder dar seguimiento a interpretaciones musicales de longitudes arbitrarias. Esta implementación utiliza una representación espectral de los datos del audio. El resultado del trabajo presentado por Dixon se compiló en un *toolkit* para la alineación de grabaciones de audio llamado MATCH [Dixon y Widmer, 2005]. Este *toolkit* puede ser conseguido en línea en [Dixon, 2010] y se desarrolló un *plug-in* para su uso en conjunto con *Sonic Visualizer* [Cannam *et al.*, 2010].

En [Cont, 2006] se utiliza un HMM Jerárquico (*HHMM*, *Hierarchichal Hidden Markov Model* [Fine *et al.*, 1998]) de dos jerarquías: el primer nivel consiste en estados musicales de alto nivel correspondientes a las partes atómicas de la partitura: notas, acordes,

y descansos. El segundo nivel contiene un modelado temporal de acuerdo al tiempo de ocurrencia de cada evento de la partitura. Para la representación de los datos de audio se utiliza un algoritmo de análisis de tonos múltiples [de Cheveigné, 2006].

En [vanKasteren, 2006] se plantea la solución de un problema alterno pero similar: el seguimiento en tiempo real del ritmo de la interpretación. Inicialmente proponen el uso del filtro de Kalman para calcular un estimación óptima del ritmo usando las predicciones generadas por el modelo y las mediciones tomadas de una interpretación. Posteriormente, el usar un filtro de partículas les permite efectuar el seguimiento del ritmo de una interpretación de la cual se desconoce la partitura. Puesto que para calcular el ritmo de una interpretación es necesario saber o estimar la frecuencia de ocurrencia de las notas, el trabajo presentado por Kasteren se puede considerar también como seguimiento de audio.

En [Raphael, 2006] y [Cont, 2010] se proponen esquemas que estiman continuamente el ritmo (*tempo*) de la interpretación y lo toman en cuenta para efectuar la alineación. Lo anterior, puesto que en muchas interpretaciones el ritmo no es constante sino que cambia con el tiempo. Incluso si el ritmo se considera constante para cierta pieza musical, los intérpretes humanos son incapaces de mantenerlo y siempre existen variaciones. Además, estos esquemas basados en el ritmo se basan en la idea de que la percepción de la estructura musical es una actividad continua en la cual las expectativas futuras del oyente pueden ser tan importantes como los eventos musicales en sí mismos. En [Raphael, 2006] se presenta un sistema de alineación offline de audio polifónico compuesto por dos etapas consecutivas para estimar la posición y el ritmo. La primera etapa se compone por un HMM derivado de la partitura de la interpretación. Este modelo se encarga de estimar la posición. La segunda etapa utiliza una red bayesiana [Jordan, 1998] para calcular el ritmo a lo largo de la interpretación. Por el contrario, en [Cont, 2010] se centran en el procesamiento online en tiempo real y las estimaciones de ritmo y posición se realizan en paralelo.

En [Jordanus, 2007] se utilizó un esquema de seguimiento para el acompañamiento automático basado en el trabajo de [Orio y Déchelle, 2001] usando HMMs de dos niveles para representar los eventos esperados y las posibles desviaciones/errores que pueden tener lugar. Únicamente permite 13 posibles observaciones correspondientes a las 12 notas de la escala cromática más los silencios. Utiliza también el enfoque de nota a nota y para la

extracción de características de la señal se utiliza el tono. Una cuestión notable es que, además del tono de la nota, toma en cuenta el volumen de la misma para tratar de que el acompañamiento se toque con un volumen relativo al utilizado por el intérprete.

En [Kirchhoff y Lerch, 2011] se presenta un estudio y evaluación de las características de extracción utilizadas en los esquemas de AA y SF con el fin de determinar cuáles entregan los mejores resultados. Se analiza una gran cantidad de características de la señal: la energía, valor máximo, volumen, espectro, llanura espectral, Coeficientes Cepstrales de Frecuencia de Mel (MFCC, del Inglés $Mel\ Frequency\ Cepstral\ Coefficients$) [Davis y Mermelstein, 1980], chromas, tiempos de ejecución, entre otros. En los resultados describen que la elección de las características de extracción debe depender de las interpretaciones de audio con las que se trabaje, sin embargo se menciona que, si bien se pueden obtener alineaciones precisas utilizando únicamente una característica de extracción, el utilizar múltiples características usualmente arroja mejores resultados. Más aún, el aplicar diferentes pesos a cada una de las características usadas puede en algunos casos mejorar los resultados.

Por último en [Chou *et al.*, 2012] se presenta un sistema de seguimiento musical en tiempo real que puede alinear una interpretación con su correspondiente partitura. Este trabajo describe la implementación de una aplicación desarrollada por los autores en la cual los usuarios utilizan un teclado conectado mediante una interfaz MIDI para interpretar una pieza musical cuya partitura se ha cargado previamente en el sistema mediante un archivo MIDI y una imagen. Así, cuando el usuario interpreta una nota en el teclado, se lee la entrada MIDI, se compara con el archivo MIDI de la partitura, se estima la posición del usuario dentro de la partitura y se despliega en la imagen de la partitura un indicador que muestra al usuario su posición. Con este enfoque no se requiere procesamiento de la señal de audio puesto que es convertida a entradas MIDI directamente. Se utiliza el algoritmo de programación dinámica presentado en [Dannenberg, 1984] pero extiende dicho algoritmo para poder procesar música polifónica: agrupando y clasificando las notas polifónicas en notas iniciales y notas seguidoras. Adicionalmente, toma en consideración los hábitos del intérprete y las circunstancias, como el repetir partes de la partitura o tocar la nota incorrecta.

Como se mencionó en la sección 1.1.3, para el presente trabajo se utiliza un enfoque

que no ha sido considerado hasta la fecha para abordar el problema de AF y AA. Se utiliza la entropía de la señal [Camarena y Chávez, 2006] como característica de extracción y se utiliza un índice de proximidad para estimar la posición del intérprete. Estas diferencias hacen que una comparación directa con alguno de los esquemas previamente implementados sea complicada. Por esta razón, se propone una técnica de validación similar a la utilizada en [Hu *et al.*, 2003b] en donde se modificaron artificialmente los archivos MIDI utilizados para evaluar la alineación obtenida. Se crearán firmas de audio modificadas artificialmente en puntos específicos conocidos de manera que se conozca de antemano la alineación esperada entre las interpretaciones y se pueda comparar con la alineación obtenida. En conjunto con la técnica de validación mencionada, se obtuvieron manualmente las mediciones de los momentos de ejecución de las notas (*note onset*) de algunas de las interpretaciones utilizadas durante los experimentos y de esta manera se puede conocer la alineación real de dichas interpretaciones y compararla con la alineación obtenida por el seguidor. El conjunto de interpretaciones utilizadas para efectuar los experimentos de este trabajo es el mismo conjunto utilizado en algunos trabajos previos como [Goebl, 2001], [Dixon, 2005] y [Kirchhoff y Lerch, 2011].

1.3. Objetivos y Aporte de la Tesis

1.3.1. Objetivo general

Implementar un esquema para realizar seguimiento de audio utilizando una huella de audio basada en la entropía de la señal y un índice de búsqueda aproximada basado en *Locality Sensitive Hashing*.

1.3.2. Objetivos particulares

- Obtener las huellas de audio de varias interpretaciones utilizando una firma basada en la entropía.
- Diseñar un esquema de seguimiento mediante la búsqueda secuencial de subfirmas dentro de las firmas extraídas previamente.

- Implementar el índice LSH.
- Incrementar la calidad del desempeño del índice LSH usando múltiples tablas Hash.
- Implementar el esquema *Index Audio Following* o Seguimiento de Audio mediante Índices.

1.3.3. Aporte

Este trabajo de tesis tiene como aporte el presentar un enfoque distinto para abordar el problema del seguimiento de audio puesto que no se había considerado anteriormente el uso de índices de proximidad para tratar de resolver dicho problema. Adicionalmente, tampoco se había considerado la entropía de la señal como una característica de extracción en las señales de audio cuando se ha abordado el AF previamente. De esta forma, en este trabajo se proponen por primera vez los algoritmos y parámetros necesarios para realizar AF mediante el uso de índices de proximidad. Así como un esquema de validación novedoso basado en la creación de firmas de audio sintéticas modificadas artificialmente.

1.4. Descripción de Capítulos

Este capítulo introduce al lector al problema del seguimiento de audio y se provee una apreciación global de sus antecedentes. Así mismo se describen objetivos del presente trabajo y sus contenidos. El resto del documento guarda la estructura presentada a continuación.

En el Capítulo 2 se incluyen los aspectos teóricos de las herramientas utilizadas en la implementación del proyecto: la huella de audio utilizada para la extracción de características de la señal de audio y el índice de proximidad basado en *Locality Sensitive Hashing* (LSH).

En el Capítulo 3 se presenta una descripción a detalle del esquema de seguimiento de audio propuesto al cual se denominó *Seguimiento de Audio mediante Índices* o IAF (del Inglés, *Index Audio Following*), las decisiones tomadas para su diseño e implementación y los parámetros utilizados.

El Capítulo 4 expone los experimentos planteados con el fin de validar el funcionamiento del seguidor y los resultados obtenidos al realizar las pruebas de seguimiento; primero mediante el uso de una búsqueda secuencial y posteriormente con el índice LSH implementado, acompañados de una discusión sobre su significado y desempeño.

Por último en el Capítulo 5 se proporcionan las conclusiones alcanzadas al término del trabajo y se plantean algunos de los posibles trabajos futuros que pueden surgir de la presente investigación.

Capítulo 2

Marco Teórico

En este capítulo se presentan los conceptos teóricos de las herramientas utilizadas para el proyecto. En principio se discutirá la AFP utilizada para la extracción de características de la señal de audio por lo que se brinda una explicación de algunos conceptos como: entropía y la escala de Bark [Zwicker, 1961]. Posteriormente, se discute el índice de proximidad utilizado en este trabajo el cual se basa en LSH [Gionis *et al.*, 1999].

2.1. Extracción de Características

Generalmente, el primer paso en la identificación de señales de audio consiste en la extracción de las características de dicha señal de manera que éstas representen al audio de manera adecuada. A esta representación generalmente se le conoce como Huella de Audio o Firma de Audio (AFP, del Inglés *Audio Fingerprint*). Existen diferentes características acústicas que se pueden extraer y analizar como: volumen, nota, tono, ritmo o compás, etc[Wold *et al.*, 1996][Kirchhoff y Lerch, 2011]. Se han diseñado diversas huellas de audio, cada una de las cuales tiene sus características particulares con ventajas y desventajas, pero todas siguen un proceso similar. En la Figura 2.1 se describe el proceso de extracción de características.

El primer paso usualmente es el preprocesamiento, durante el cual, se realizan operaciones como la conversión de la señal a un formato mono aural efectuando el promedio de los canales de la señal original. En seguida se divide la señal en Marcos (*Frames*),

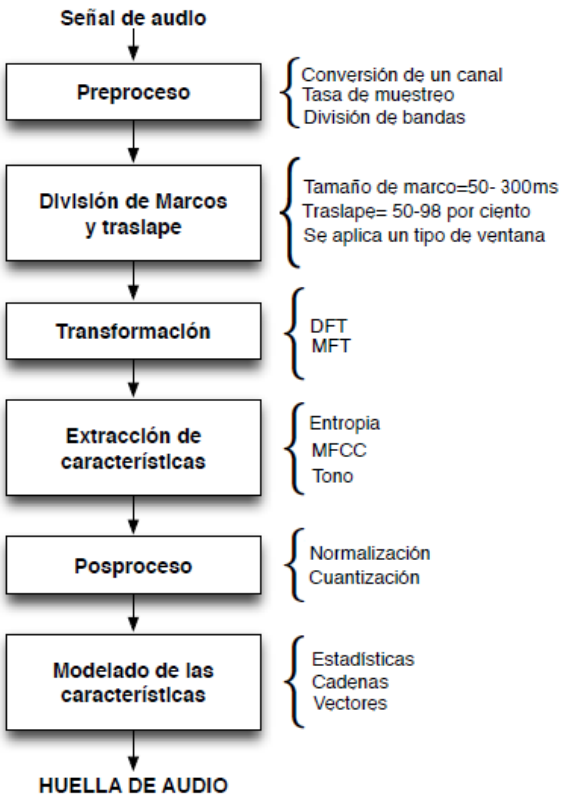


Figura 2.1: Proceso de extracción de características.

pequeños segmentos de la señal (usualmente de entre 50 a 300 ms) y que generalmente tienen un traslape (50 % - 98 %) entre ellos. Posteriormente se aplica algún tipo de transformación a la señal: la transformada de Fourier, para cambiar al domino de la frecuencia, es utilizada comúnmente pero se puede utilizar otro tipo de transformación (ej. transformada coseno) dependiendo de la huella que se requiera.

Una vez que se tiene la señal en el dominio deseado, se realiza la extracción de características como pueden ser los espectrogramas, los coeficientes cepstrales de frecuencia de Mel (*Mel Frecuency Cepstral Coefficients, MFCC*) [Davis y Mermelstein, 1980], los coeficientes de predicción lineal (*Linear Prediction Coefficients, LPC*) [O'Shaughnessy, 1988], la entropía de la señal [Shannon y Weaver, 1949], etc.

Opcionalmente se puede realizar un postprocesamiento de la señal, el cual puede consistir en la cuantización vectorial, normalización o alguna operación adicional sobre los

vectores de características. Y por último, cada firma elige una representación para sus características las cuales pueden ser vectores, cadenas, valores únicos, etc.

Como se mencionó anteriormente, cada huella de audio tiene sus fortalezas y debilidades particulares, lo cual conlleva a que su desempeño dependa en gran medida de las características del problema en cuestión. Para el presente trabajo se decidió utilizar la AFP publicada en [Camarena y Chávez, 2006]. Esta firma se basa en la idea de que es la información de la señal lo que en realidad percibe el cerebro humano y es por ello que realiza el cálculo de la entropía de la señal por cada banda crítica de Bark [Zwicker, 1961] como se explica a continuación.

Como se describe en [Camarena y Chávez, 2006], el contenido de información $I(p_i)$ de un valor v_i depende únicamente de su probabilidad de ocurrir $p_i = P(v_i)$. Mientras menos probable sea un valor de aparecer, mayor será la información que provea cuando sí ocurra. De esta manera, el contenido de información deberá ser una función monotónicamente decreciente de la probabilidad como en 2.1 [Shannon y Weaver, 1949].

$$I(p_i) = ln(1/(p_i)) = -ln(p_i) \tag{2.1}$$

La entropía H es el contenido de información esperado en una secuencia, es decir, la sumatoria de todos los contenidos de información multiplicados por su probabilidad de ocurrir 2.2.

$$H(x) = E[I(p)] = \sum_{i=1}^{n} p_i I(p) = -\sum_{i=1}^{n} p_i ln(p_i) \tag{2.2}$$

Para calcular la entropía de una señal se requiere una estimación de su Función de Densidad de Probabilidad (PDF, del Inglés *Probability Density Function*). El método utilizado para realizar esta estimación se basa en histogramas, se avanza el marco de audio únicamente una muestra a la vez, lo cual permite que los histogramas sean fácilmente actualizables: cada vez que se obtiene una nueva muestra de audio basta con efectuar el incremento adecuado en la posición correspondiente del histograma y realizar el decremento de la muestra que sale del marco.

La escala de Bark [Zwicker, 1961] es una escala acústica propuesta por Eberhard

Zwicker en 1961, la cual divide al espectro de frecuencia en veinticuatro bandas que coinciden con las primeras veinticuatro bandas críticas del oído [Zwicker y Fastl, 1990]. La conversión de frecuencia f a Barks se realiza como se indica en 2.3

$$Bark = 13arctan(0.00076f) + 3.5arctan((f/7500)^2) \tag{2.3}$$

Así pues, la AFP utilizada primeramente convierte la señal al dominio de la frecuencia utilizando la transformada de Fourier y calcula la entropía de cada marco de la señal por cada banda crítica de Bark obteniendo entonces vectores de veinticuatro valores de entropía. Por último, se realiza una codificación de los vectores de entropía obtenidos a ceros y unos. Si la entropía de la señal aumentó para una determinada banda respecto al marco anterior se utiliza un uno y si bajó o se mantuvo igual se utiliza un cero. En la Figura 2.2 podemos observar el resultado de la codificación utilizada por la huella.

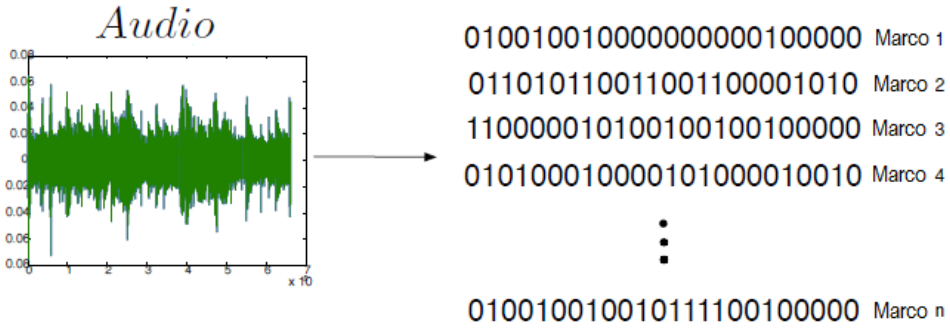


Figura 2.2: Ejemplo de la codificación de la huella de audio.

Debido a las características de esta huella, es altamente compatible para ser utilizada con índices basados en Hashing Sensible a la Localidad (*Locality Sensitive Hashing, LSH*) y se ha utilizado en aplicaciones de recuperación de información como se presentó en [Santoyo y Chávez, 2012] con un desempeño adecuado.

2.2. Índice de Proximidad LSH

Una manera de clasificar los índices de búsqueda es a partir del tipo de respuesta que garantizan:

1. Los índices con respuesta exacta garantizan que dentro de los resultados siempre se encontrará al elemento más cercano a la búsqueda.

2. Los índices con respuesta aproximada no garantizan que se entregue al vecino más cercano, pero sí resultados muy cercanos a la consulta. Estos métodos producen un conjunto de elementos más cercanos.

Ambos tipos tienen sus ventajas y desventajas. La mayoría de los métodos de búsqueda exacta sufren del fenómeno conocido como la *maldición de la dimensionalidad* como se describe en [Bellman, 1961]. Este fenómeno básicamente consiste en lo siguiente: conforme la dimensionalidad de los vectores aumenta, el volumen del espacio se incrementa muy rápidamente por lo cual los datos disponibles se vuelven dispersos. Dicha dispersidad provoca que para poder realizar cualquier tipo de análisis significativo, la cantidad de datos necesarios aumente exponencialmente. Por otro lado, los métodos que devuelven resultados aproximados intercambian la exactitud por velocidad de respuesta o el espacio de memoria necesario.

La calidad de los resultados entregados por una búsqueda realizada en un índice se determina por la proximidad de los elementos obtenidos respecto al elemento de consulta. Si se entregan los vecinos más cercanos al elemento de consulta, los resultados se consideran de buena calidad. Si por otra parte, los elementos obtenidos por la búsqueda son muy diferentes al elemento de consulta, se considera que los resultados son de mala calidad.

El problema de la búsqueda aproximada es de gran importancia en gran variedad de aplicaciones como pueden ser la compresión y minería de datos, recuperación de información, reconocimiento de patrones, estadística y análisis de datos entre muchos otros. En muchas de estas aplicaciones, no es necesaria la respuesta exacta para una búsqueda; determinar una respuesta aproximada suele ser suficiente.

El *Locality Sensitive Hashing* fue presentado en [Gionis *et al.*, 1999]. Es una técnica de búsqueda por proximidad muy rápida que provee garantías probabilistas sobre la calidad de los resultados. La idea es aplicar una función hash a los diferentes elementos de la base de datos de manera que se garantice que la probabilidad de que dos elementos obtengan el mismo hash sea mayor para elementos similares que para aquellos que son diferentes.

Usualmente las características de los vectores son representadas como puntos en $\mathbb{R}^b$, donde b es el número de coordenadas del objeto vector y una distancia métrica es utilizada para medir la similitud entre dos objetos.

Usualmente, los índices LSH se implementan como tablas hash de manera que exista una alta probabilidad que los objetos que sean cercanos respecto a alguna función de distancia d se encuentren dentro de la misma cubeta. A cada tabla hash utilizada para implementar el índice se le conoce como **instancia**. Existe un límite en la calidad de resultados que puede ser garantizada usando una única instancia LSH. Para aumentar la calidad de los resultados la solución más simple es utilizar varias instancias. No obstante, lo anterior aumenta los requerimientos de memoria. Por ello se han desarrollado esquemas de compresión para índices LSH como los presentados en [Santoyo y Chávez, 2012] y [Santoyo *et al.*, 2012]. En este trabajo no fue necesario utilizar ningún esquema de compresión del índice LSH, puesto que no se trabaja con bases de datos de millones de elementos, de hecho el caso con mayor número de elementos con el que se trabajó fue de 13050 vectores (Ballade #14).

2.2.1. Preprocesamiento del Índice LSH

Gionis propone que para explicar cómo se define la función hash usada en LSH resulta muy adecuado utilizar una representación unaria de los vectores. Como se explicó en la sección 2.1, la AFP utilizada en este trabajo utiliza una codificación a ceros y unos por lo cual se puede explicar la función hash fácilmente. La función hash se define en [Gionis *et al.*, 1999] y [Andoni y Indyk, 2008] como sigue: sea P el conjunto de todos los vectores y b la dimensionalidad de dichos vectores. Para un entero l, se eligen l subconjuntos $I_1, ..., I_l$ de entre $\{1, ..., b\}$. Sea $p_{|I}$ la proyección del vector p en el conjunto de coordenadas I. Se calcula $p_{|I}$ seleccionando las posiciones de las coordenadas según I y concatenando los bits en estas posiciones. Se representa la cubeta $g_j(p) = p_{|I_j}$. Para el preprocesamiento, se almacena cada $p \in P$ en la cubeta $g_j(p)$ para $j = 1, ..., l$. Es decir, l determina el número de instancias (tablas hash) a utilizar. Para cada instancia se elige un subconjunto I_l de tamaño n, donde $n <= b$, de posiciones de entre $\{1, ..., b\}$. Se toman los bits de p de las posiciones indicadas por I_l y se concatenan para formar el hash de p. Por último se almacena p en la cubeta

indicada por el hash obtenido.

En la Figura 2.3 se muestra un ejemplo del procedimiento descrito en el párrafo anterior. Se determinó usar dos instancias $l = 2$ y utilizar tres posiciones para generar las funciones hash $n = 3$ de las cinco disponibles $b = 5$. Los conjuntos I_1 e I_2 se determinan al azar. Los bits a utilizar para la primera instancia l_1 son $I_1 = \{1, 3, 5\}$; mientras que para la segunda instancia l_2 son $I_2 = \{3, 4, 5\}$. El preprocesamiento se aplica para cada uno de los vectores de la firma a indexar, sin embargo para fines de este ejemplo se tomó el vector de la posición 5 (00111) como muestra. Para calcular el hash de la instancia l_1 se toman los bits de las posiciones 1, 3 y 5 por lo cual el hash obtenido es 011, entonces se agrega la posición del vector a la lista de posiciones de la cubeta 011 de la primera tabla hash. Así mismo, para calcular el hash de la instancia l_2 se toman los bits 3, 4 y 5 por lo cual el hash que se obtiene es 111, entonces se agrega la posición 5 a la lista de posiciones de la cubeta 111 de la segunda tabla. Nótese que en este trabajo no se almacena todo el vector p dentro de la cubeta $g_j(p)$ sino simplemente la posición del vector p.

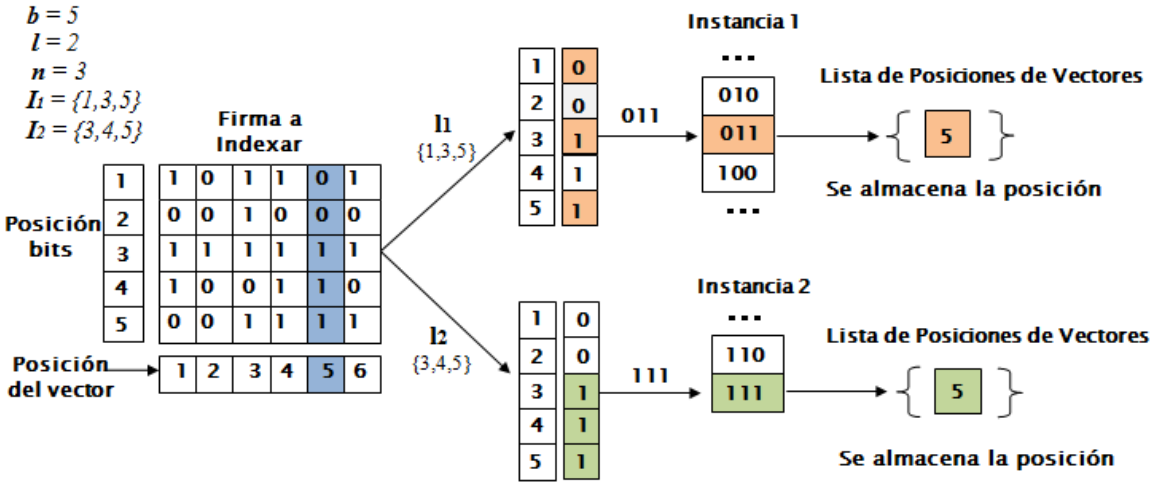


Figura 2.3: Proceso de indexado de la firma de audio.

El Algoritmo 1 [Andoni y Indyk, 2008] presenta el proceso de preprocesamiento descrito previamente. El algoritmo toma como entrada el conjunto de vectores P, el número de instancias (tablas hash) a utilizar l y el número de bits a utilizar para generar el hash n; y devuelve como salida el conjunto de tablas hash T pobladas con las posiciones de los

vectores en las cubetas correspondientes. Para cada instancia se determina una función hash aleatoria que en este caso serán los subconjuntos $I_1, ..., I_l$ (líneas 2-3). Posteriormente, por cada instancia, para todos los vectores $p \in P$ se calcula su función hash y se almacena la posición del vector en la cubeta correspondiente (líneas 4-6). Por último se regresa el conjunto de tablas hash T.

Algoritmo 1: Algoritmo de preprocesamiento del índice LSH

Entrada: P - conjunto de vectores, l - número de instancias, n - número de bits para generar el hash

Salida : T - conjunto de tablas hash.

```
1 s ← | P |;
2 for i ← 1 to l do
3     Inicializar la tabla hash T_i generando una función hash aleatoria g_i();
4 for i ← 1 to l do
5     for j ← 1 to s do
6         Almacenar cada punto p_j en la cubeta g_i(p_j) de la tabla hash T_i
7 Regresar T;
```

2.2.2. Consulta del Índice LSH

En [Gionis *et al.*, 1999] se describe la manera de realizar una consulta al índice LSH de la siguiente manera: para procesar una consulta q, se realiza la búsqueda en todas las cubetas $g_1(q), ..., g_l(q)$ hasta que se encuentran $c * l$ objetos donde c es el número de objetos encontrados por cubeta. Sean $p_1, ..., p_{cl}$ los objetos encontrados en la consulta. Para la búsqueda aproximada de vecinos más cercanos KNN (del Inglés, *K-Nearest Neighbors*) se regresan los K objetos más cercanos a q; en general, es posible regresar menos puntos si el número de puntos encontrados fue menor a K.

Para el presente trabajo, al consultar el índice LSH no se obtienen directamente los vectores (objetos) puesto que éstos nunca fueron almacenados en el índice para empezar. En su lugar, se obtienen las posiciones que ocupan los vectores dentro de la AFP de referencia.

En la figura 2.4 se presenta un ejemplo del proceso de consulta del índice LSH. Para este ejemplo se utilizan tres instancias $l = 3$, y se utilizan tres bits $n = 3$ de los cinco disponibles $b = 5$ para crear los hash. Los conjuntos I_1, I_2, I_3 quedaron de la siguiente manera: $I_1 = \{1, 2, 3\}$, $I_2 = \{2, 3, 4\}$ e $I_3 = \{2, 3, 5\}$. El vector de consulta q es el vector 10101. Para cada instancia l_j se obtiene el hash del vector de consulta q a partir de las posiciones indicadas en I_j. De esta manera los hashes que se obtienen son: $g_1(q) = 101$, $g_2(q) = 010$ y $g_3(q) = 011$. Se revisan las cubetas correspondientes a los hashes obtenidos en cada instancia y se obtienen las posiciones $\{1, 8, 19, 24\}$ en l_1, $\{3, 14\}$ en l_2 y $\{5, 10, 21\}$ en l_3 por lo que la lista final de posiciones obtenidas por la consulta es $\{1, 3, 5, 8, 10, 14, 19, 21, 24\}$.

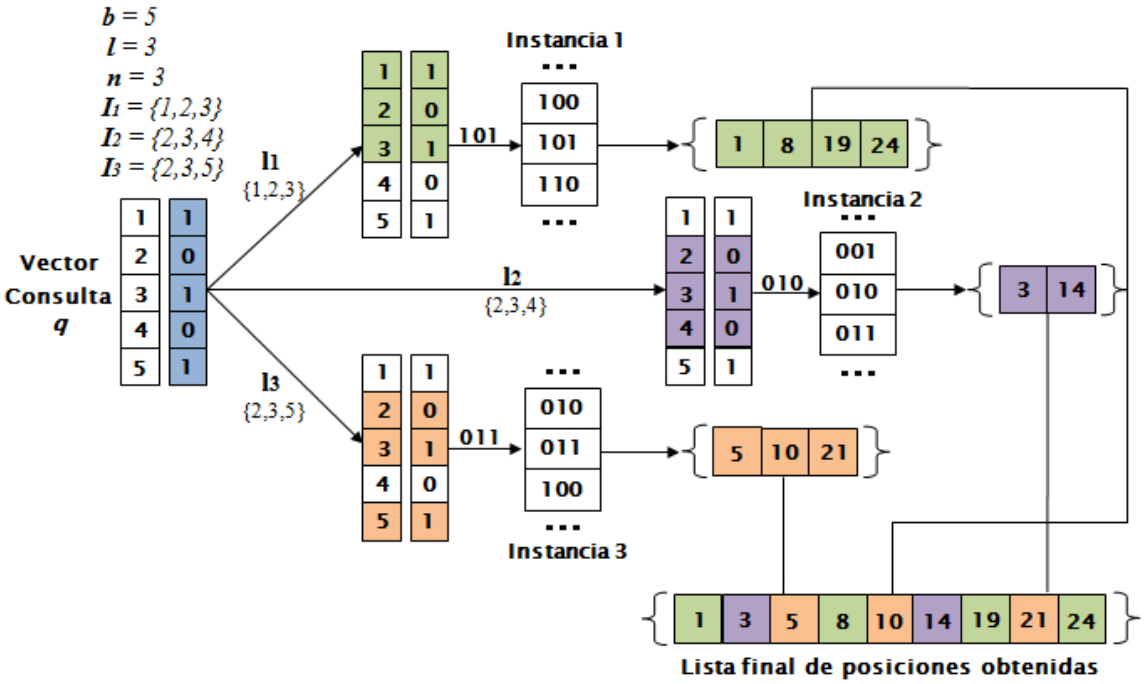


Figura 2.4: Proceso de consulta del Índice LSH.

El algoritmo 2 [Andoni y Indyk, 2008] presenta el proceso de consulta descrito previamente. Se inicia una lista vacía C que contendrá a todas las posibles posiciones candidato (línea 1). Para cada instancia l_j, se calcula el hash del vector consulta q y se busca en la cubeta correspondiente de la tabla T_j. Cada posición encontrada en la cubeta se almacena en C (líneas 2-4). Al terminar de procesar todas las instancias se regresa C (línea 5).

Algoritmo 2: Algoritmo de consulta del índice LSH

Entrada: q - vector de consulta, T - conjunto de tablas hash, l - número de tablas hash, I - conjunto de posiciones utilizadas para determinar el hash

Salida : C - lista de posiciones de los candidatos

```
1 C ← {};
2 for j ← 1 to l do
3     Calcular g_j(q) utilizando los bits indicados en I_j;
4     C ← C ⋃ {c|c es una posición encontrada en la cubeta g_j(q) de la tabla T_j};
5 Regresar C;
```

El principio detrás del método esta basado en que la probabilidad de colisión de dos puntos p y q esta relacionada con la distancia entre ellos. Específicamente, entre más grande es la distancia entre dos objetos, menor es la probabilidad de colisión. En resumen la calidad de los resultados entregados por el índice LSH dependerá de 2 factores principalmente:

- La cantidad de instancias LSH que se utilicen: usar más instancias mejora la calidad de los resultados pero aumenta también los requerimientos de memoria.

- La cantidad de bits n que se utilicen para crear el hash: un valor de n bajo creará instancias con cubetas muy pobladas y, por lo tanto, se obtendrán más candidatos; por otro lado un valor de n alto creará instancias con cubetas poco pobladas y se obtendrán menos candidatos.

La elección de la cantidad de mapas y el número de bits a utilizar dependerá en particular de los requerimientos y recursos de la aplicación para la cual se utilice el índice LSH.

2.3. Resumen del Capítulo

En este capítulo se presentaron los fundamentos teóricos de funcionamiento de la AFP basada en la entropía de la señal y del índice de proximidad basado en LSH que se utilizan en este trabajo. Se describió el proceso general que sigue una AFP y se brindó una explicación de los conceptos necesarios para entender el funcionamiento de la AFP utilizada. Para el índice LSH se presentó la idea detrás de su origen e implementación, así como la explicación de sus procedimientos de preprocesamiento y consulta. Una vez que han sido introducidos estos conceptos se puede proceder a la explicación sobre el diseño y la implementación del seguidor de audio presentado en este trabajo. Éste será el enfoque del próximo capítulo donde se brindará una descripción a detalle.

Capítulo 3

Diseño e Implementación

En este capítulo se discuten algunas de las decisiones más importantes de diseño e implementación del seguidor y se brinda una descripción del funcionamiento general del mismo. Se introducen además los parámetros propuestos y se explica su propósito y la relación entre ellos.

3.1. Definición de Conceptos

Para poder tratar con el diseño del seguidor primero es necesario definir algunos términos que serán utilizados frecuentemente a través de este documento:

- **Interpretación:** un intento realizado por un músico específico para tocar una pieza de música en un momento dado.
- **Intérprete:** el músico que toca la pieza musical.
- **Interpretación de referencia:** la interpretación que fue procesada de manera offline y que se utiliza como referencia para alinear a otra interpretación.
- **Interpretación online:** la interpretación que se procesa paulatinamente conforme se reciben sus datos y se alinea, usualmente en tiempo real, con la interpretación de referencia.

- **Ritmo:** también llamado *tempo*, es el reloj musical dinámico utilizado por los intérpretes. Usualmente se refiere a la velocidad o la frecuencia con que un intérprete toca las notas de una interpretación.

- **Partitura:** es la música de forma escrita que un intérprete lee cuando toca una pieza musical.

- **Firma de audio:** es la totalidad de la huella de audio de una interpretación.

- **Subfirma de audio:** es un segmento de la totalidad de la huella de audio de una interpretación. Para este trabajo en particular se refiere a una matriz que equivale a medio segundo de audio.

3.2. Diseño General

En esta sección se describe la idea general detrás del diseño del seguidor. Como se menciona en la Sección 1.1, el Seguimiento de Partitura (SF) se refiere a la alineación en tiempo real entre una interpretación musical y su correspondiente partitura, la cual ha sido introducida a una computadora mediante alguna representación simbólica (como pueden ser los estados de un HMM).

El presente trabajo adopta un enfoque diferente a la mayoría de los mencionados en la Sección 1.2. No se utiliza programación dinámica (DP), funciones de densidad de probabilidad (PDFs), filtro de Kalman, ni HMMs. Adicionalmente, el seguimiento no se realiza nota a nota, es decir, la alineación no se efectúa contra la partitura de la interpretación. En su lugar se realiza Seguimiento de Audio (AF) que básicamente consiste en alinear una interpretación online con una interpretación de referencia que fue procesada (extracción de características e indexamiento) previamente.

El esquema de seguimiento propuesto se compone de los siguientes pasos:

- Preprocesamiento

  1. Obtención de la AFP de la interpretación de referencia.

2. Indexamiento en el índice de proximidad LSH de la AFP de audio obtenida en el paso anterior.

- Seguimiento

1. Obtención de una subfirma de la interpretación online.
2. Obtención de posiciones candidato a partir de la consulta en el índice de los vectores de la subfirma.
3. Obtención de la lista de vecinos más cercanos mediante la comparación entre la subfirma de la interpretación online y las subfirmas obtenidas a partir de las posiciones candidato.
4. Elección de la siguiente posición en el seguimiento utilizando la lista de vecinos más cercanos obtenida en el paso anterior basándose en alguno de los modos implementados.
5. Repetir los pasos 1 - 4 hasta que termine la interpretación online.

En las subsecciones siguientes se detallan cada uno de los pasos que componen el esquema de seguimiento propuesto.

3.2.1. Obtención de la AFP de la interpretación de referencia

Como se explicó en la Sección 2.1, la huella de audio utilizada entrega como resultado secuencias de ceros y unos correspondientes al aumento o disminución de la entropía por cada marco en cada una de las veinticuatro bandas críticas de Bark. En este trabajo únicamente se utilizan las primeras diecisiete debido a que las interpretaciones utilizadas no tienen componentes de frecuencia en las bandas dieciocho a veinticuatro. Así, una interpretación será representada por una matriz de ceros y unos cuyas dimensiones dependerán de los parámetros utilizados por la firma.

Se utilizó un tamaño de marco de $\approx 93ms$ (4 KB para muestras tomadas a 44100 Hz) con un traslape de $\approx 81ms$ (3584 bytes), es decir, se produce un nuevo vector cada $\approx 12ms$ (512 bytes). Así, una interpretación de 1 minuto ($60s$) se convierte en una firma de $\approx$ [17 X 5000] ($60s/0.012s = 5000$).

3.2.2. Indexamiento de la AFP

Cuando se ha extraído la huella de audio de la interpretación de referencia se procede a indexarla. Los conceptos generales de funcionamiento del índice LSH utilizado se describen en la Sección 2.2. En la Figura 2.3 se presentó un ejemplo del mismo. En resumen:

- Se determina cuantas tablas hash (instancias, l), se utilizarán.
- Se determina cuantos bits (n) de los diecisiete disponibles se utilizarán para crear los hashes.
- Para cada instancia se determinan al azar los conjuntos de posiciones de los bits $(I_1, ..., I_l)$ que serán utilizados para crear los hashes.
- Para cada vector binario de la huella se calcula la función hash para cada instancia y se almacena la posición del vector en la cubeta adecuada.

En la Tabla 3.1 se observa un ejemplo de la función utilizando $l = 2$ y $n = 3$ para el vector muestra $v = 10100110011101100$. Se define $I_1 = \{4, 8, 11\}$ para l_1 e $I_2 = \{1, 10, 17\}$ para l_2 respectivamente. Así, para calcular la función hash de la instancia l_1 se toman los bits de las posiciones 4, 8 y 11 del vector v, se concatenan y como resultado se obtiene $h_1 = 001$. De la misma forma, para la instancia l_2 se toman los bits de las posiciones 1,10 y 17 del vector v, se concatenan y como resultado se obtiene el hash $h_2 = 110$.

Vector Binario	1 0 1 0 0 1 1 0 0 1 1 1 0 1 1 0 0	
Posición bit	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	
Instancia	Posiciones Utilizadas I	Hash Obtenido(Cubeta)
l_1	$I_1 = \{4, 8, 11\}$	$h_1 = 001$
l_2	$I_2 = \{1, 10, 17\}$	$h_2 = 110$

Tabla 3.1: Ejemplo función hash.

3.2.3. Obtención de las subfirmas de la interpretación online

Una vez que la huella de audio de la interpretación de referencia se ha indexado completamente se ha terminado con el preprocesamiento y se puede comenzar a realizar

el seguimiento. Para realizar la alineación entre la interpretación de referencia y la interpretación online, se determinó utilizar segmentos de medio segundo ($0.5s$) de audio de la interpretación online. Inicialmente se planteó utilizar segmentos de un segundo ($1s$) de audio, sin embargo se consideró que, gracias a que el traslape entre marcos es bastante grande, se produce una firma bastante "fina" por lo cual se podía realizar un seguimiento más refinado. Adicionalmente, se consideró que utilizar segmentos de audio de tamaño menor a medio segundo, si bien brindarían una alineación de mayor calidad, no sería adecuado puesto que no habría tiempo suficiente para poder realizar la alineación en tiempo real. Así pues, el tamaño de las subfirmas de audio utilizadas será de $\approx$ [17 X 43] ($0.5s/0.116s = 43$).

3.2.4. Obtención de posiciones candidato

Cuando se cuenta con una subfirma de medio segundo de audio de la interpretación online, el siguiente paso será determinar la posición donde correspondería dicha subfirma en la firma de la interpretación de referencia. Para ello se realiza una consulta al índice. La respuesta a dicha consulta arrojará las posibles posiciones donde podría ocurrir una correspondencia. A estas posiciones se les llaman posiciones candidato.

En el Algoritmo 3 se describe el procedimiento para la obtención de las posiciones candidato. Se recibe como entrada la subfirma a buscar s_f, el conjunto de tablas hash T y los conjuntos de bits utilizado para obtener los hashes en cada instancia I. Se inicia con una lista de candidatos vacía (línea 1). Para lidiar con la proximidad, el posible ruido y aumentar la robustez de la búsqueda, por cada vector en la subfirma se crean variaciones de uno, dos o hasta tres bits, como se menciona en [Haitsma y Kalker, 2002] y [Haitsma *et al.*, 2001] (líneas 2-3). Por cada vector de variación creado se calcula su función hash para cada instancia y se obtienen las posiciones almacenadas en las cubetas correspondientes (líneas 4-7). Las variaciones ayudan a aumentar la robustez de la consulta puesto que las variaciones creadas a partir del vector de consulta original q producirán hashes distintos y, por lo tanto, se obtendrán posiciones candidato de cubetas que no se habrían revisado si únicamente se utiliza el vector q. Estas posiciones son agregadas a la lista de candidatos si es que no se encuentran ya en la lista (líneas 8-10). Por último se regresa la lista con las posiciones candidato (línea 11).

Algoritmo 3: Algoritmo para la obtención de las posiciones candidato.

Entrada: s_f - subfirma $0.5s$ a buscar en el índice , T - conjunto de tablas hash, I - conjuntos de bits utilizados en cada instancia l

Salida : *candidatos* - lista de posiciones candidato

```
1  candidatos ← {};
2  for vector ∈ s_f do
3      variaciones ← crearVariaciones(vector);
4      for var ∈ variaciones do
5          for t ∈ T do
6              h ← hash(var, I_t);
7              cubeta ← t.obtener(h);
8              for posicion ∈ cubeta do
9                  if !candidatos.contiene(posicion) then
10                     candidatos.agregar(posicion);
11 Regresar candidatos;
```

Cuando ha terminado este paso se cuenta con todas las posibles posiciones consideradas como candidatos para la alineación.

3.2.5. Obtención de los K vecinos más cercanos, KNN

El siguiente paso en el seguimiento consiste en producir una lista de vecinos más cercanos de entre todas las posiciones candidato que se obtuvieron en el paso anterior. Para ello se comparará la subfirma de la interpretación online con cada una de las subfirmas de la interpretación de referencia conseguidas a partir de las posiciones candidato. Como se explicó en la Sección 3.2.3 el tamaño de las subfirmas utilizadas para la alineación es de [17 X 43] equivalentes a medio segundo de audio. Para obtener una subfirma de la interpretación de referencia a fin de compararla con la subfirma de la interpretación online se toma la posición candidato y se obtiene una matriz del tamaño adecuado a partir de ella. En la figura 3.1 se ejemplifica lo anterior. Suponga que el tamaño de las subfirmas

para el ejemplo es de [5 X 6] y se tiene una lista de posiciones candidato compuesta por las posiciones $\{21, 35, 50\}$. Si se toma la posición candidato 21 se obtendrá una subfirma que va desde dicha posición hasta la posición 26 de la AFP de referencia. La subfirma obtenida será posteriormente comparada con la subfirma de consulta.

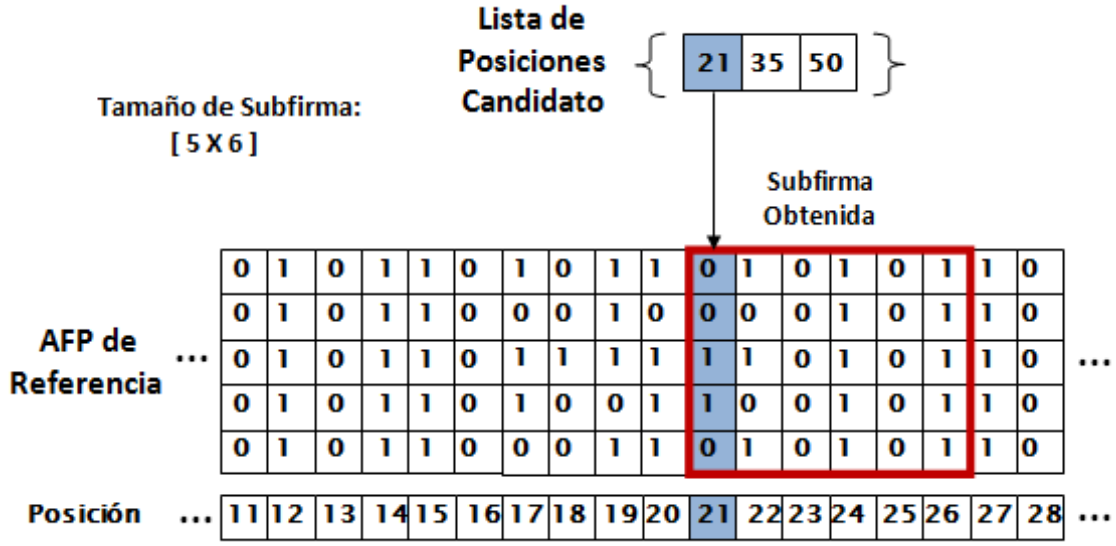


Figura 3.1: Obtención de una subfirma de la AFP de referencia a partir de una posición candidato.

Cuando el vector a partir del cual se obtuvo la posición candidato no es el primer vector en la subfirma de consulta, entonces se considera un desplazamiento (offset) al momento de obtener la subfirma de la AFP de referencia. Para el mismo ejemplo de la figura anterior 3.1, si se obtuvo la posición candidato $p_c = 21$ a partir del segundo vector (offset = 1) de la subfirma de consulta, al obtener la subfirma de la AFP de referencia se utilizará la posición 21 como la segunda posición, es decir, la subfirma se obtendrá desde la posición 20 hasta la posición 25 de la AFP de referencia. Lo anterior para aumentar la probabilidad de que las subfirmas de consulta y la de referencia sean lo más parecidas posible.

Para la comparación de subfirmas se utiliza una de tres funciones de distancia implementadas. Es altamente improbable que se encuentre una subfirma dentro de la interpretación de referencia que sea exactamente igual ($distancia = 0$) a la subfirma que se busca de la interpretación online debido a que se trata de dos interpretaciones distintas.

Por la razón anterior, y para aumentar la robustez del seguimiento, no se considera únicamente la posición cuya subfirma entregue la distancia más pequeña respecto a la subfirma buscada. En cambio, se obtiene una lista de k vecinos más cercanos knn integrada por las k posiciones que tengan las distancias más pequeñas a la subfirma buscada. Alguna de estas posiciones será elegida como la siguiente posición. El Algoritmo 4 describe el proceso para obtener dicha lista.

Algoritmo 4: Algoritmo para la obtención de KNN.

Entrada: s_f - subfirma de consulta, *candidatos* - lista de posiciones candidato

Salida : knn - lista con las k posiciones candidato más cercanas a s_f, $distanciasKnn$ - lista con las distancias obtenidas por las subfirmas de las posiciones de knn

```
1  knn ← {};
2  distanciasKnn ← iniciar la lista con valores grandes;
3  indicesRevisados ← {};
4  for c ∈ candidatos do
5      if !indicesRevisados.contiene(c) then
6          indicesRevisados.agregar(c);
7          subfirmaCandidato ← obtenerSubfirma(c);
8          distancia ← distanciaEntreSubfirmas(s_f, subfirmaCandidato);
9          distanciaMin = distanciasKnn.obtenerUltimoElemento();
10         if distancia < distanciaMin then
11             knn.agrega(c);
12             distanciasKnn.agrega(distancia);
13             ordenar(distanciasKNN);
14 Regresar knn, distanciasKnn;
```

Se recibe como entrada la subfirma de consulta s_f y la lista de posiciones candidato *candidatos*. Como salida se entregarán dos listas al terminar: la lista knn que contendrá las

k posiciones candidato que resultaron más cercanas a s_f y la lista $distanciasKnn$ que contendrá las distancias obtenidas por las subfirmas de las posiciones correspondientes contenidas en knn. Se inicia una lista vacía para knn y la lista $distanciasKnn$ se inicia con k valores grandes (por ejemplo 9999999) (líneas 1-2). Se inicia además una lista vacía para almacenar las posiciones candidato que ya han sido revisadas y así evitar realizar la misma comparación entre subfirmas más de una vez (línea 3). Para cada posición en la lista de posiciones candidato, si no se ha revisado anteriormente, se obtiene su subfirma como se describió anteriormente (lineas 4-7). Posteriormente, la subfirma de referencia obtenida es comparada con la subfirma de consulta usando alguna de las distancias implementadas. Para saber si forma parte de los k vecinos más cercanos, la lista $distanciasKnn$ se mantiene siempre ordenada de manera ascendente. Así, sabemos que si la distancia $distancia$ calculada por la comparación entre las subfirmas s_f y $subfirmaCandidato$ (línea 8) es menor a la última distancia $distanciaMin$ almacenada en $distanciasKnn$ (línea 9), entonces cumple los requisitos, y la posición candidato y su correspondiente distancia calculada se incluyen en las listas knn y $distanciasKnn$ (líneas 10-13). Lo anterior se repite hasta que se han revisado todos los candidatos. Por último, se regresan las listas knn y $distanciasKnn$.

3.2.6. Proceso completo de consulta del índice

En la Figura 3.2 se ejemplifica el procedimiento completo de búsqueda de candidatos en el índice descrito en las secciones previas. Se tiene la subfirma de la interpretación online llamada subfirma de consulta de tamaño [5 X 6]. Se toma cada uno de sus vectores como se indicó en el Algoritmo 3 de la Sección 3.2.4, en la figura de ejemplo se tomó el vector de la posición 5. Se crean variaciones de uno, dos o tres bits, en el ejemplo por cuestiones de espacio se crearon variaciones de un bit únicamente. Se toma cada variación y se calcula su hash para cada tabla, en el ejemplo se utiliza una instancia donde $I = \{1, 3, 4\}$ para calcular la función hash. El hash obtenido para el vector '00111' es $h = 011$. Se busca en la cubeta correspondiente de la tabla hash y se revisan las posiciones candidato que contiene. En el ejemplo, la primera posición de la cubeta es la posición 25, entonces se debería obtener una subfirma de la AFP de referencia que comience en el índice 25 como se indica en la figura 3.1. Sin embargo, como el vector a partir del cual se crearon las variaciones se encontraba

en la posición 5 de la subfirma de consulta, entonces existe un desplazamiento (offset) de 4 $(5 - 1 = 4)$ posiciones. Por lo tanto la subfirma se obtiene a partir de la posición 21 $(25 - 24 = 21)$ y hasta la posición 26 de la AFP de referencia.

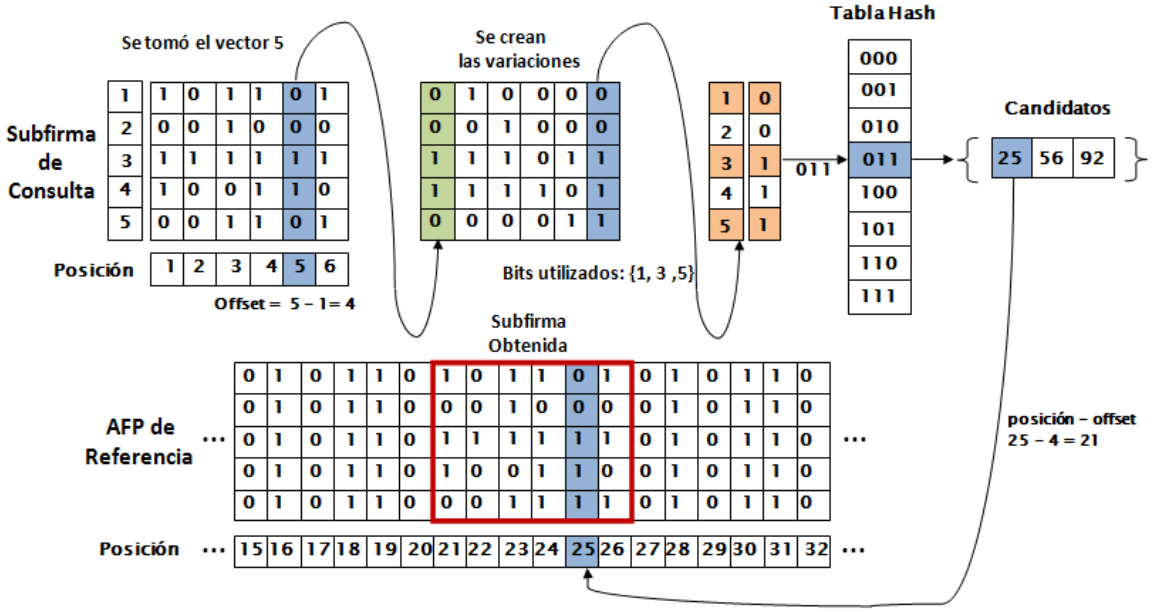


Figura 3.2: Proceso de obtención de un candidato.

3.2.7. Elección de la siguiente posición

El último paso en el algoritmo de seguimiento consiste en la elección de la que será considerada como la siguiente posición. Como se mencionó en la sección anterior, una de las posiciones incluidas en la lista de k vecinos será elegida como posición siguiente, a menos que el algoritmo decida no avanzar y permanecer en la posición anterior.

Inicialmente se asume la posición inicial $p_s = 0$ también se indica que la posición previa $p_p = 0$ puesto que no existe ninguna posición previa. A partir de ahí, debido a la precedencia temporal de los eventos, se elegirá siempre una posición siguiente que sea superior o igual a la posición anterior $p_n \geq p_p$. Es decir, asumimos que el intérprete no "regresará" dentro de la interpretación, sino que siempre "avanza" o se "mantiene" en una posición. La elección de la siguiente posición se realiza en base a los modos implementados

que se describen posteriormente de manera extensa.

El proceso se repite hasta que la interpretación online termina y se han alineado todas sus subfirmas con la interpretación de referencia con lo cual se da por terminado el seguimiento.

3.3. Distancias

En esta sección se describen los 3 tipos de distancias implementadas para la comparación de subfirmas que se necesita durante el seguimiento.

3.3.1. Distancia de Hamming

La distancia de Hamming es una distancia para comparación de cadenas introducida por Richard Hamming en [Hamming, 1950]. Consiste simplemente en comparar caracter por caracter ambas cadenas y registrar en cuantas posiciones dichos caracteres son diferentes. Para cadenas binarias a y b es equivalente a la operación XOR(a,b). Por ejemplo:

1. La distancia de Hamming entre $a = 10011$ y $b = 10101$ es de 2.
2. La distancia de Hamming entre $a = 00101$ y $b = 10000$ es de 3.
3. La distancia de Hamming entre $a = 11001$ y $b = 11000$ es de 1.

Las ventajas de la distancia de Hamming consisten en que es sumamente sencilla de implementar y muy rápida de evaluar. Entre sus desventajas se encuentra la necesidad de que ambas cadenas sean de la misma longitud. Para comparar cadenas de diferentes longitudes, donde no sólo se consideran substituciones sino también inserciones y borrados, se necesita una métrica más robusta como la distancia de Levenshtein.

3.3.2. Distancia de Levenshtein

La distancia de Levenshtein o distancia de edición, es una distancia para la comparación de cadenas introducida por Vladimir Levenshtein en [Levenshtein, 1966]. Informalmente, consiste en el número mínimo de ediciones de un carácter (substitución, inserción,

borrado) requeridas para convertir una cadena en otra. Matemáticamente, la distancia entre dos cadenas a y b está dada por $lev_{a,b}(|a|,|b|)$ donde

$$lev_{a,b} = \begin{cases} max(i,j) \text{ if } min(i,j) = 0, \\ min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \text{ de otra manera} \\ lev_{a,b}(i-1,j-1) + 1_{a_i \neq b_i} \end{cases} \end{cases}$$

$a_i \neq b_i$ indica que se suma uno si a_i y b_i son diferentes, en caso de que sean iguales se suma cero. Algunos ejemplos de la distancia de Levenshtein:

1. La distancia de Levenshtein entre $a = 10101$ y $b = 111$ es de 2.
2. La distancia de Levenshtein entre $a = 1001$ y $b = 1000$ es de 1.
3. La distancia de Levenshtein entre $a = 001$ y $b = 01$ es de 1.

La distancia de Levenshtein es ampliamente utilizada en diversas aplicaciones de comparación de cadenas como los correctores de ortografía. Sin embargo, la mayor desventaja de esta distancia es su costo de cálculo, el cual es aproximadamente proporcional al producto de la longitud de ambas cadenas.

3.3.3. Subsecuencia Común Más Larga, LCS

Esta distancia consiste en encontrar la subsecuencia común más larga entre dos o más secuencias de caracteres. Se denomina LCS por sus siglas en Inglés (*Longest Common Subsequence*) y no debe confundirse con el problema de la subcadena más larga (*Longest Common Substring*). Para un número constante de secuencias (dos para el presente trabajo) el problema es soluble en tiempo polinomial utilizando programación dinámica. Matemáticamente la distancia entre dos secuencias $A = (a_1, a_2, ..., a_m)$ y $B = (b_1, b_2, ..., b_n)$

$$LCS(X_i, Y_j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ LCS(X_{i-1}, Y_{j-1}) + 1 & \text{if } x_i = y_j \\ max(LCS(X_i, Y_{j-1}), LCS(X_{i-1}, Y_j)) & \text{if } x_i \neq y_j \end{cases}$$

Ejemplos de LCS:

1. El valor LCS entre $a = 10011$ y $b = 10101$ es 4 (LCS = '1001').

2. El valor LCS entre $a = 001010$ y $b = 10000$ es 4 (LCS = '0000').

3. El valor LCS entre $a = 101$ y $b = 010$ es 2 (LCS = '10').

Esta métrica únicamente permite inserciones y borrados, no incluye sustituciones. Para convertir la métrica LCS a una distancia, se propone la diferencia de longitud entre la LCS obtenida y la longitud de la secuencia más larga:

$$distLCS(sec_1, sec_2) = max(longitud(sec_1), longitud(sec_2)) - LCS(sec_1, sec_2)$$

Así para los ejemplos presentados las distancias serían: 1, 2 y 1 respectivamente.

3.3.4. Comparación entre las distancias

Como es de esperarse, se obtienen valores de distancia diferentes al utilizar las diversas distancias implementadas, como se muestra en el ejemplo siguiente:

1. La distancia de Hamming entre $a = 11010101110$ y $b = 01101000110$ es 6.

2. La distancia de Levenshtein entre $a = 11010101110$ y $b = 01101000110$ es 3.

3. La distancia LCS entre $a = 11010101110$ y $b = 01101000110$ es 2 (LCS = '110100110').

Estas diferencias pueden tener un impacto muy significativo en el comportamiento del seguimiento. Lo anterior debido a que la lista de k vecinos más cercanos obtenida puede ser sumamente distinta dependiendo de la distancia de comparación utilizada. Las comparaciones del comportamiento del seguimiento con las diversas distancias se discutirán en el Capítulo 4.

3.4. Modos de Seguimiento

En esta sección se presentan las cuatro formas diferentes que se implementaron para elegir la siguiente posición a partir de la lista de k vecinos más cercanos. A estas formas se les denominó "modos":

- Modo 1: posición más cercana en tiempo.
- Modo 2: posición más cercana en distancia.
- Modo 3: combinación de los Modos 1 y 2.
- Modo 4: posición más cercana a la predicción.

La función de distancia empleada para calcular las distancias entre subfirmas es independiente del modo utilizado para elegir la siguiente posición, por lo cual con cualquier modo se puede utilizar cualquiera de las distancias descritas en la sección anterior.

3.4.1. Modo 1: posición más cercana en tiempo

La manera más sencilla de elegir la siguiente posición en el seguimiento consiste en simplemente elegir la posición más cercana en tiempo (mayor) a la posición previa. De la lista de k posiciones candidato entregada por la consulta al índice se descartan aquellos que sean menores o iguales a la última posición conocida. Se obtiene la diferencia de los restantes respecto a la posición previa y se elige como siguiente posición aquella que tenga la menor diferencia. El algoritmo 5 describe el funcionamiento de este modo.

Se recibe como entrada la posición previa en el seguimiento p_p y la lista de k posiciones más cercanas knn. Como salida se entrega la siguiente posición en el seguimiento p_n. Se inicializa la d_{min} la cual nos servirá para almacenar la diferencia mínima en tiempo desde cualquiera de las posiciones candidato a la p_p (línea 1). La siguiente posición p_n se inicia con el mismo valor que p_p por si no se encuentra ningún candidato p_c que cumpla con las condiciones (ser mayor a p_p, entonces el algoritmo se mantendrá en la misma posición (línea 2). Para cada candidato p_c en knn, si es mayor a p_p se obtiene su diferencia en tiempo d_t y si es menor a d_{min} se actualizan los valores de d_{min} y p_n (líneas3-8). Cuando se termina de revisar todos los candidatos se regresa p_n (línea 9). Como se puede apreciar, el algoritmo del Modo 1 ni siquiera toma en cuenta la lista de distancias $distanciasKnn$, lo cual puede ocasionar problemas, como se verá a continuación.

El siguiente ejemplo se ideó para describir el funcionamiento del Modo 1 y además resaltar algunas de sus posibles carencias. Los valores de posición y distancia utilizados fue-

Algoritmo 5: Algoritmo para elección de la siguiente posición del Modo 1.

Entrada: p_p - la posición previa del seguimiento, *knn* - lista de k posiciones candidato

Salida : p_n - siguiente posición en el seguimiento

```
1 d_min ← 9999999 ;
2 p_n ← p_p;
3 for p_c ∈ knn do
4     if p_c >= p_p then
5         d_t ← (p_c − p_p);
6         if d_t < d_min then
7             d_min ← d_t;
8             p_n ← p_c;
9 Regresar p_n;
```

ron elegidos con ese propósito y no son el resultado de un experimento práctico. Considérese que se tiene como posición previa $p_p = 100$ y las listas *knn* y *distanciasKnn* entregadas son las mostradas en la Tabla 3.2. Para este modo se elegiría la posición 105 como siguiente posición debido a que es mayor a 100 y tiene la menor diferencia, $(100 - 105 = 5)$. La posición 97 sería descartada ya que es menor a 100.

knn	*distanciasKnn*
posición: 110	distancia: 430
posición: 120	distancia: 432
posición: 97	distancia: 435
posición: 105	distancia: 520
posición: 200	distancia: 521

Tabla 3.2: Ejemplo de listas *knn* y *distanciasKnn*.

Se puede apreciar los problemas que podría presentar el Modo 1. Para que el seguimiento sea lo más adecuado posible, sería deseable que la siguiente posición se eligiera entre aquellas que se encuentran más cercanas en distancia a la subfirma de audio que estamos buscando. Es claro que la posición 110 o incluso 120 serían mejores candidatos

para ser designados como siguiente posición puesto que reportan valores de distancia mucho menores. Sin embargo, la heurística del Modo 1 los descarta y en su lugar elige la posición 105, la cual tiene una distancia muy superior a las posiciones 110 o 120.

Este modo favorece los "pasos pequeños", lo cual podría ser no deseable en algunos casos ya que el seguimiento podría quedarse "estancado" y no avanzar de la manera adecuada.

3.4.2. Modo 2: posición más cercana en distancia

Una segunda política para la elección de la siguiente posición consiste en elegir la posición dentro de la lista de vecinos más cercanos que tenga la menor distancia al segmento de audio que estamos buscando (cumpliendo que debe ser una posición "superior" a la anterior). El algoritmo 6 describe el funcionamiento de este modo.

Algoritmo 6: Algoritmo para elección de la siguiente posición del Modo 2.

Entrada: p_p - la posición previa del seguimiento, knn - lista de k posiciones candidato, $distanciasKnn$ - lista con las distancias obtenidas a partir de las subfirmas de las posiciones candidato de knn

Salida : p_n - siguiente posición en el seguimiento

```
1 d_min ← 9999999 ;
2 p_n ← p_p;
3 for p_c ∈ knn do
4     if p_c >= p_p then
5         d_d ← distanciasKnn.obtenerDistancia(p_c);
6         if d_d < d_min then
7             d_min ← d_d;
8             p_n ← p_c;
9 Regresar p_n;
```

Se recibe como entrada la posición previa en el seguimiento p_p, la lista de k posiciones más cercanas knn y la lista las distancias obtenidas a partir de las subfirmas de

las posiciones candidato $distanciasKnn$. Como salida se entrega la siguiente posición en el seguimiento p_n. Se inicializa la d_{min} la cual nos servirá para almacenar la diferencia mínima en distancia desde cualquiera de las posiciones candidato a la p_p (línea 1). La siguiente posición p_n se inicia con el mismo valor que p_p por si no se encuentra ningún candidato p_c que cumpla con las condiciones (ser mayor a p_p, entonces el algoritmo se mantendrá en la misma posición (línea 2). Para cada posición candidato p_c en knn, si es mayor a p_p, se obtiene su distancia correspondiente d_d de la lista de distancias $distanciasKnn$ (líneas 3-5). Si d_d es menor a d_{min} entonces se actualizan los valores de d_{min} y p_n (líneas 6-8). Cuando se termina de revisar todos los candidatos se regresa p_n (línea 9).

Para el mismo ejemplo utilizado en la Sección 3.4.1, la posición elegida sería la 110 puesto que es superior a la posición anterior $p_p = 100$ y es la que reporta la menor distancia.

Como se puede apreciar, el algoritmo del Modo 2 no toma la cercanía en tiempo de las posiciones candidato. Es por ello que para esta política también existen problemas potenciales. Por ejemplo, debido a la naturaleza cíclica de la música es altamente probable que varias secciones de una interpretación/canción se parezcan mucho entre sí. Los coros de las canciones son un muy buen ejemplo. De esta manera, la posición elegida por la política del Modo 2 podría ser una posición que se encuentre muy lejana a la posición anterior, lo cual podría ser no deseable en ciertas circunstancias.

El siguiente ejemplo se ideó para describir el funcionamiento del Modo 2 y además resaltar algunas de sus posibles carencias. Los valores de posición y distancia utilizados fueron elegidos con ese propósito y no son el resultado de un experimento práctico. Considere que la posición previa es $p_p = 100$ y la listas knn y $distanciasKnn$ entregadas son las mostradas en la Tabla 3.3.

La posición elegida como siguiente posición sería la posición 300 ya que reporta una distancia de 101, sin embargo nuestra posición anterior era la posición 100 por lo cual existe una diferencia de 200 posiciones entre la posición anterior y la siguiente. Ésto podría ser deseable, por ejemplo, si justo en esa posición la interpretación que se está siguiendo tiene una parte faltante respecto a la interpretación de referencia que se encuentra indexada. Como si el intérprete se saltara algunas notas.

knn	$distanciasKnn$
posición: 300	distancia: 101
posición: 120	distancia: 103
posición: 480	distancia: 104
posición: 99	distancia: 110
posición: 101	distancia: 112

Tabla 3.3: Ejemplo 2 de listas knn y $distanciasKnn$.

Pero para el ejemplo presente, es claro que la siguiente posición debería ser la posición 120 que se encuentra a sólo 20 posiciones de la posición anterior y la distancia entre sus subfirmas es únicamente de 2 puntos más que la posición 300.

3.4.3. Modo 3: combinación de los Modos 1 y 2

Es claro a partir de las secciones anteriores, que sería deseable encontrar un equilibrio entre elegir la posición más cercana a a la posición anterior en tiempo o la más cercana en distancia. El Modo 3 pretende resolver este problema.

Llamemos p_p a la posición previa, se calcula la siguiente posición en tiempo p_{nt} (sección 3.4.1) y en distancia p_{nd} (sección 3.4.2). Así como sus respectivas distancias d_t y d_d. Como el Modo 1 se basa en la cercanía en tiempo, es posible que la distancia reportada por la subfirma de la posición obtenida con este modo sea mucho mayor a la distancia reportada por la posición del Modo 2. Para lidiar con ésto, se utilizará un valor umbral que sirve para distinguir si la posición reportada por el Modo 1 tiene una distancia mucho mayor a la reportada por el Modo 2. Se aplican los siguientes criterios:

- Si la diferencia entre d_t y d_d es mayor al umbral ($d_t - d_d > umbral$), asignamos p_{nd} como siguiente posición.
- Si la diferencia entre d_t y d_d es menor o igual al umbral ($d_t - d_d \leq umbral$), significa que ambas posiciones deberán ser tomadas en cuenta:
  - Si p_{nd} se encuentra a más del doble de distancia en tiempo de p_p que p_{nd} ($(p_{nt} - pp) * 2 < (p_{nd} - pp)$), entonces se elige como siguiente posición p_{nt}.
  - De lo contrario, se elige como siguiente posición p_{nd}.

Este modo combina los dos anteriores pues se favorece la elección del candidato más parecido siempre y cuando no se encuentre muy lejano de la posición anterior. Lo anterior se describe en el Algoritmo 7.

Algoritmo 7: Algoritmo para elección de la siguiente posición del Modo 3.

Entrada: p_p - posición previa en el seguimiento, p_{nt} - posición siguiente elegida por cercanía en tiempo, p_{nd} - posición siguiente elegida por cercanía en distancia, d_t - distancia de p_{nt}, d_d - distancia de p_{nd}, *umbral* - valor umbral determinado previamente

Salida : p_n - siguiente posición en el seguimiento

```
1 p_n ← p_p.;
2 if d_t − d_d > umbral then
3 |  p_n ← p_nd;
4 else
5 |  if (p_nt − p_p) ∗ 2 < (p_nd − p_p) then
6 |  |  p_n ← p_nt;
7 |  else
8 |  |  p_n ← p_nd;
9 Regresar p_n;
```

Se toma como entrada la posición previa en el seguimiento p_p, la posición siguiente elegida por cercanía en tiempo p_{nt}, la posición siguiente elegida por cercanía en distancia p_{nd} y sus respectivas distancias d_t y d_d. Además el valor umbral que sirve para determinar si se tomarán en cuenta tanto p_{nt} como p_{nd}. Como salida se entregará la siguiente posición del seguimiento p_n. La siguiente posición p_n se inicia con el mismo valor que p_p (línea 1). Si la diferencia entre d_t y d_d es mayor al *umbral* entonces se asigna a p_n la posición p_{nd} (líneas 2-3). Si no fue mayor al *umbral* entonces se toman en cuenta tanto p_{nt} como p_{nd}. Si p_{nd} se encuentra a más del doble de distancia en tiempo de p_p que p_{nt} entonces se elige como siguiente posición a p_{nt} (líneas 5-6), de otra manera, se elige como siguiente posición a p_{nd}. Por último se regresa p_n.

3.4.4. Modo 4: posición más cercana a la predicción

Se implementó una cuarta política que se basa en la idea de que los cambios muy bruscos son poco comunes, salvo en casos extremos, y que se puede realizar una "predicción" de cuál será la siguiente posición a partir de las posiciones anteriores.

Para ésto se almacena no solamente la posición previa p_p, sino también la anterior a la previa p_{pp} y se realiza una extrapolación lineal a partir de ambas, para obtener la predicción de la siguiente posición p_{pred}. Posteriormente se elige la posición dentro de la lista de vecinos más cercanos que se encuentre más cerca p_{pred}. La extrapolación se efectúa mediante la fórmula siguiente:

$$y_k = y_{k-2} + \frac{x_k - x_{k-2}}{x_{k-1} - x_{k-2}} * (y_{k-1} * y_{k-2}) \tag{3.1}$$

Por ejemplo, se toman las listas knn y $distanciasKnn$ mostradas en la Tabla 3.3 y si se tiene un valor $p_{pred} = 110$ se elegiría como siguiente posición la posición 101 puesto que $abs(101 - 110) = 9$. Si $p_{pred} = 111$, entonces se elegiría la posición 120 puesto que $abs(120 - 111) = 9$.

3.5. Parámetros

En esta sección se describen los parámetros importantes utilizados en la presente implementación de seguimiento de audio, así como su papel e impacto en el comportamiento del mismo.

La sección se divide en dos subsecciones: una de ellas dedicada a los parámetros que afectan el algoritmo de seguimiento en general y otra dedicada a los parámetros que afectan el comportamiento del índice LSH.

3.5.1. Parámetros Generales

Estos parámetros afectan el funcionamiento del algoritmo de seguimiento sin importar si la búsqueda de posibles candidatos se realiza de forma secuencial o utilizando el índice LSH.

Parámetro k: El parámetro k se refiere al tamaño de la lista de vecinos más cercanos que se utilizará para elegir de entre ellos a la siguiente posición, tiene un impacto importante en el seguimiento puesto que limita las posibles opciones a elegir.

- Si se elige un valor k muy pequeño se corre el riesgo de dejar fuera posibles candidatos viables. Podría ser que la lista se llene con candidatos que no son viables por diversas razones: se encuentran "atrás" o demasiado alejados de la posición previa y el algoritmo podría optar por "mantenerse" en la posición previa cuando debería de "avanzar". Una ventaja de tener un valor pequeño es que la búsqueda tarda un poco menos.

- Si se elige un valor de k muy grande se corre el riesgo de incluir entre los posibles candidatos a posiciones cuyas subfirmas en realidad no son similares a la subfirma buscada. El algoritmo podría decidir "avanzar" cuando en realidad debería "mantenerse". Una ventaja de aumentar el valor de k es que aumenta la posibilidad de encontrar una posición que cumpla las condiciones para ser elegida como posición siguiente.

El parámetro k, como muchos otros, se puede optimizar para cada caso particular de manera que el seguimiento se realice de la mejor manera posible. Sin embargo, gracias a los experimentos realizados (algunos de los cuales se pueden encontrar en el Capítulo 4) se propone utilizar un valor de k entre 10 y 50, dependiendo de la longitud de las interpretaciones.

Parámetro n_b: Este parámetro se refiere a cuántas de las veinticuatro bandas críticas de Bark que contiene la huella de audio, se utilizarán para realizar las comparaciones entre matrices. Lo anterior toma particular importancia cuando las canciones o interpretaciones utilizadas en el seguimiento no contienen componentes de frecuencia en todas las bandas críticas.

- Si se utiliza un número muy grande de bandas críticas (mayor a la cantidad de bandas con contenido útil) se puede afectar el resultado de la comparación entre las matrices. Adicionalmente, el tiempo de comparación aumenta puesto que las matrices son de mayor tamaño.

- Igualmente si se utiliza un número demasiado pequeño de bandas críticas, se corre el riesgo de no tomar en cuenta información importante que podría ser determinante en la elección correcta de la posición siguiente.

Lo recomendable entonces es utilizar el número mínimo de bandas críticas de Bark que alcancen a cubrir el espectro de frecuencia encontrado en las interpretaciones utilizadas.

Para los experimentos de este trabajo se utilizaron interpretaciones de piezas clásicas tocadas en piano. Estas interpretaciones consisten de veintidós pianistas distintos tocando dos extractos de piezas en piano de Frédéric Chopin (Étude en E Major, Op. 10, no. 3, barras 1-21; y Ballade Op. 38, barras 1-45). Un piano de 88 teclas contiene ocho octavas por lo que su rango de frecuencia va de los $27.5Hz$ (A0) hasta los $4186.01Hz$ (C8). Debido a ésto, se utilizaron únicamente las primeras **diecisiete** bandas críticas de Bark las cuales comprenden un rango de $20Hz$ - $3700Hz$ con lo cual se cubre casi a totalidad el espectro de frecuencia del piano. Quedando fuera únicamente las últimas dos teclas del piano B7($3951.07Hz$) y C8($4186.01Hz$), lo cual no afecta el resultado de los experimentos realizados puesto que las interpretaciones utilizadas no se utilizan esas dos teclas. Las partituras se pueden consultar en el anexo D.

Parámetro windowSize: El parámetro $windowSize$ se refiere al tamaño que tendrá la subfirma de consulta. Como se explicó en la Sección 3.2.3, se usarán subfirmas que equivalen a medio segundo de audio por lo cual para marcos con un traslape de $11.6ms$, este parámetro tendrá un valor de $windowSize = 43$ y las subfirmas serán de dimensiones [17 X 43]

Si se varía el parámetro $windowSize$, simplemente se cambia el tiempo al cual equivale la subfirma a buscar. Por ejemplo, si el valor se reduce a $windowSize = 22$, se tratará con subfirmas equivalentes a $0.25s$; si se aumenta el valor a $windowSize = 86$ se tratará con subfirmas equivalentes a $1s$.

Estas variaciones tienen sus respectivas ventajas y desventajas:

- Si se trata con subfirmas de $0.25s$, el seguimiento podrá ser mucho más fino, en el sentido de que se podrá tener una alineación cada cuarto de segundo. Sin embargo, un

cuarto de segundo podría ser tiempo insuficiente para realizar todo el procesamiento necesario para realizar la alineación: es necesario capturar el audio, obtener la AFP, una vez que se tiene una subfirma completa se buscan los posibles candidatos, se obtiene la lista de k vecinos y se elige la siguiente posición de acuerdo al modo activo (véase Sección 3.4). Para las aplicaciones en tiempo real, todo lo anterior se realiza mientras paralelamente se continúa capturando el audio para la siguiente subfirma a buscar. Si se utiliza una distancia como Levenshtein, la cual es costosa de calcular, las comparaciones entre subfirmas requieren mucho más tiempo y podría ser que no se cumplan los requerimientos de tiempo real.

- Si se trata con subfirmas mayores a $1s$, el tiempo puede dejar de ser problema. Sin embargo el seguimiento ya no sería tan preciso lo cual puede ser muy perjudicial para ciertas aplicaciones.

Parámetro windowMultiplier: Este parámetro se utiliza como una manera de hacer valer la decisión de diseño que indica que la siguiente posición no puede estar muy alejada de la posición anterior. Es decir, limita los posibles "saltos" que puede dar el seguimiento.

Se utiliza en conjunto con el parámetro $windowSize$ de la siguiente manera: se establece un límite inferior para el parámetro, por ejemplo $windowMultiplierDefault = 3$ o wmd por conveniencia. Si $windowSize = 43$ $(0.5s)$ como se describió en la sección anterior, eso implica que la posición siguiente no podrá estar a más de 129 posiciones de distancia de la posición anterior $(43 * 3 = 129)$. En otras palabras, no podrá estar a más de 1.5 segundos. Si la posición elegida de entre los posibles candidatos no cumple con ésta condición (por ejemplo: $p_p = 100$ y $p_{np} = 300$) entonces el algoritmo de seguimiento determinará que es más conveniente mantenerse en la posición actual $p_n = p_p$.

Lo anterior puede ocasionar comportamientos no deseados: conforme avanza el audio de la interpretación "online" es más probable que si no se avanzó en el paso anterior los candidatos a siguiente posición se encuentren más alejados de p_p, lo cual haría que la condición

$$p_n \leq p_p + (windowSize * windowMultiplier) \tag{3.2}$$

no se cumpla jamás.

El Algoritmo 8 muestra la solución propuesta para dicha situación. Se recibe como entrada la posición previa p_p, la posible siguiente posición p_{np} obtenida por cualquiera de los modos explicados previamente, el parámetro *windowSize* que indica el tamaño de las subfirmas utilizadas, el parámetro *windowMultiplier* que indica cuantas "ventanas" adelante podrá estar la siguiente posición p_n y *wmd* que determina el límite inferior de *windowMultiplier*. Si se cumple la condición de tolerancia 3.2, se asigna como siguiente posición la posición elegida por el modo activo $p_n = p_{np}$ (líneas 1-2), además, el parámetro *windowMultiplier* se reduce siempre y cuando esté por encima del mínimo *wmd* (líneas 3-4). Si no se cumple la condición de tolerancia, entonces se asigna como siguiente posición la posición previa (no se avanza) $p_n = p_p$ y se aumenta el valor de *windowMultiplier* para aumentar el rango de posiciones permitido en la siguiente iteración (líneas 5-7).

Algoritmo 8: Algoritmo de modificación del parámetro *windowMultiplier*

Entrada: p_p - la posición previa, p_{np} - la probable posición siguiente, *windowSize* - el tamaño de la subfirma, *windowMultiplier* - determina el tamaño de los saltos, *wmd* - límite inferior del parámetro *windowSize*

Salida : p_n - siguiente posición en el seguimiento

```
1 if p_np ≤ p_p + (windowSize * windowMultiplier) then
2     p_n ← p_np;
3     if windowMultiplier > wmd then
4         windowMultiplier − −;
5 else
6     p_n ← p_p;
7     windowMultiplier + +;
```

En resumen, si se avanza en el seguimiento se disminuye el valor de *windowMultiplier* siempre y cuando sea mayor al valor mínimo. Si no se avanza en el seguimiento, se aumenta el valor de *windowMultiplier* con el fin de permitir que el seguimiento se reponga del

posible retraso.

El límite inferior del parámetro *windowSize*, llamado *wmd*, indica cuál será la dimensión esperada de los posibles "saltos" en la interpretación. Si se le asigna un valor muy grande podría ocasionar que el seguimiento no sea adecuado gracias a la posibilidad de elegir posiciones muy lejanas a la posición previa dependiendo del modo de selección utilizado (véase la sección 3.4).

Parámetro distance: El parámetro *distance* simplemente indica cuál de las tres funciones de distancia descritas en la Sección 3.3, se utiliza para realizar la comparación de las matrices. Se puede utilizar cualquiera de las tres lo cual ocasionará variaciones en la lista de k vecinos obtenida.

1. Distancia de Levenshtein.
2. Distancia de Hamming.
3. Distancia LCS.

Parámetro mode: El parámetro *mode* únicamente indica cuál de los cuatro modos descritos en la sección 3.4 se utiliza para realizar la elección de la posición siguiente.

1. Posición más cercana en tiempo.
2. Posición más cercana en distancia.
3. Combinación de los Modos 1 y 2.
4. Posición más cercana a la predicción.

3.5.2. Parámetros del índice LSH

Estos parámetros únicamente afectan el funcionamiento del índice LSH, por lo cual no modifican directamente el comportamiento del algoritmo de seguimiento. Sin embargo, sí pueden tener un impacto importante puesto que determinan cuáles serán las posiciones a ser revisadas para posteriormente poder ser consideradas como posibles candidatos.

Los primeros dos parámetros que se describirán enseguida, $howManyMaps$ y $howManyBits$, determinan la manera en que se creará el índice LSH. El parámetro $bitVariations$ se utiliza cuando se va a realizar una búsqueda en el índice.

Parámetro howManyMaps (h_{mm}): El parámetro h_{mm} determina cuantas instancias utilizará el índice LSH. Aumentar o disminuir este parámetro tiene consecuencias bastante obvias:

- Si se aumenta el número de instancias utilizadas, aumentará la respuesta del índice puesto que se obtendrán más cubetas por cada vector binario analizado (una cubeta por cada mapa). Aumentará el tiempo de procesamiento por la misma razón.
- Si se disminuye el número de mapas el efecto es contrario, disminuye el tiempo de procesamiento pero también disminuye la cantidad de posibles posiciones candidato.

Parámetro howManyBits (h_{mb}): El parámetro h_{mb} determina cuántos bits de los diecisiete disponibles se utilizarán para obtener la función hash de los vectores binarios. Este parámetro es sumamente importante puesto que determina que tan "densamente poblados" estarán las cubetas de cada instancia.

Para ejemplificar lo anterior recordemos la siguiente situación: una interpretación de 1 minuto se convierte en aproximadamente 5000 vectores binarios (considerando marcos con un traslape de $11.6ms$). Consideremos los siguientes dos ejemplos:

1. Si se utilizan 14 bits, existirán $16,384$ posibles cubetas. Si los hashes se distribuyeran uniformemente, significaría que aproximadamente $11,000$ de las $16,000$ cubetas estarían vacías y $5,000$ tendrían un elemento únicamente. Como cada vector de la firma produce un hash único por cada instancia, ésto implicaría que por cada vector únicamente se revisarían $1 * h_{mm}$ posiciones lo cual podría ser insuficiente para tener una respuesta adecuada.
2. Si se utilizan 6 bits, existirán únicamente 64 posibles cubetas en cada tabla. Si suponemos de nuevo que los hashes se distribuyen uniformemente, cada cubeta tendría ≈ 78 elementos. Lo cual implica que por cada vector se revisarían $78 * h_{mm}$, lo cual

podría ocasionar que el índice haga más comparaciones que usando búsqueda secuencial (dependiendo de la cantidad de instancias y la longitud de la interpretación).

Adicionalmente, debido a que el conjunto de bits a utilizar por cada instancia se elige aleatoriamente con cada ejecución, dos ejecuciones distintas con los mismos parámetros pueden obtener resultados distintos.

Parámetro bitVariations (b_v): El parámetro b_v determina cuantas variaciones de cada vector binario se crearán con el fin de aumentar el número de cubetas revisadas por instancia. Las posibles opciones son uno, dos y tres, las cuales indican si se crearán todas las posibles variaciones de uno, dos o tres bits respectivamente.

Este parámetro se incluyó con el fin de compensar que es poco probable que se encuentre un vector binario en la subfirma de la interpretación online, tal cual como se encuentra en la firma de la interpretación base debido al posible ruido o cualquier otro tipo de degradación. Así, si un vector de la interpretación base es similar, más no igual a otro de la interpretación online (tienen uno, dos o tres bits de diferencia), también se considerará dicho vector como posible candidato.

Por ejemplo, si se crean todas las posibles variaciones de 1 bit para un vector binario de 17 bits se generarán 17 vectores alternativos mas el original. Si usamos 6 bits para generar el hash de un vector, 11 de los 17 vectores alternativos generarán el mismo hash que el vector original. Sin embargo, 6 de ellos generarán un hash diferente, lo cual ocasiona que se revisen 6 cubetas adicionales por cada instancia. Siguiendo con el ejemplo de la sección anterior, el número total de candidatos a revisar sería $78 * h_{mm} * 6$.

Consecuentemente, crear todas las posibles variaciones de dos o tres bits aumentará exponencialmente la cantidad de cubetas obtenidas y, por lo tanto, la cantidad de candidatos revisados.

3.5.3. Interacción entre Parámetros

Debería resultar obvio entonces que todos los parámetros mencionados en las dos secciones anteriores interactúan entre sí, ocasionando cambios en el comportamiento del

algoritmo de seguimiento. También resulta natural el buscar optimizar dichos parámetros con el fin de que el seguimiento se comporte de la manera más adecuada posible.

Algunas de estas interacciones son muy claras, por ejemplo: si aumentar el número de instancias h_{mm} aumenta la cantidad de cubetas obtenidas, entonces se puede aumentar también el número de bits que se utiliza para calcular el hash h_{mb}. Ésto disminuye la "densidad poblacional" de cada cubeta con lo cual se mantiene el equilibrio en la cantidad de candidatos finales revisados.

En realidad, los tres parámetros LSH (h_{mm}, h_{mb} y b_v) deben de calibrarse de tal manera que se visite la cantidad suficiente de candidatos para obtener un comportamiento equivalente al de la búsqueda secuencial, pero haciendo un número considerablemente menor de comparaciones, de manera que se justifique la utilización del índice.

Como nota adicional sobre estos tres parámetros, es importante mencionar también que en este trabajo se están combinando dos técnicas diferentes para lidiar con la proximidad: el índice LSH con múltiples instancias [Gionis *et al.*, 1999] y las variaciones de bits [Haitsma y Kalker, 2002]. Anteriormente, se han utilizado por separado. Sin embargo, nunca habían sido utilizadas en conjunto por lo que el presente trabajo es el primero en proponer este esquema.

Otras interacciones podrían no ser tan obvias, por ejemplo: si se permite la posibilidad de hacer saltos grandes con *wmd*, entonces se debe de reducir la cantidad de vecinos k ya que si no, se corre el riesgo de saltar a una posición muy lejana que no es tan parecida a la matriz que se busca.

En la Tabla A.1 del Apéndice A, se presenta un resumen del efecto de aumentar o disminuir cada parámetro y, en caso de ser necesario, la manera de equilibrar dicho efecto cambiando otro.

3.6. Resumen de Capítulo

En este capítulo se presentó una descripción del esquema propuesto en este trabajo para realizar el seguimiento de audio. Se inició con la definición de algunos términos comunes y posteriormente se expuso a detalle cada uno de los pasos propuestos en conjunto con

sus respectivos algoritmos y ejemplos ilustrativos de funcionamiento. Finalmente, se definieron los parámetros utilizados y la manera en que afectan el comportamiento del seguidor. Habiendo discutido los detalles de diseño e implementación del seguidor, en el siguiente capítulo se presentan algunos de los resultados de los experimentos realizados, así como los valores recomendados, obtenidos a través de la experimentación, para los parámetros presentados durante este capítulo.

Capítulo 4

Experimentos y Resultados

En este capítulo se presentan los resultados obtenidos a partir de los experimentos realizados. Para realizar dichos experimentos se utilizaron las grabaciones utilizadas en trabajos similares [Dixon, 2005] y [Kirchhoff y Lerch, 2011]. Estas grabaciones consisten de veintidós pianistas distintos tocando dos extractos de piezas en piano de Frédéric Chopin (Étude en E Major, Op. 10, no. 3, barras 1-21; y Ballade Op. 38, barras 1-45, Stereo, 44.1kHz). Los pianistas eran estudiantes o profesores en la Universidad de Música de Vienna. Las partituras de ambas interpretaciones se pueden encontrar en el Apéndice D. Todas las grabaciones se pueden obtener en línea en [Goebl, 2013]. Las interpretaciones del Étude tienen una duración de entre 70.1 a 94.4 segundos, lo cual se transforma en AFP que contienen de entre 6038 a 8130 vectores aproximadamente; así mismo, las interpretaciones de la Ballade tienen una duración de entre 112.2 y 151.5 segundos, sus AFPs contienen entre 9664 y 13049 vectores. Se puede apreciar que existen diferencias significativas entre las interpretaciones aún cuando es la misma pieza musical interpretada por pianistas profesionales.

Este capítulo se divide en tres subsecciones. En la primera de ellas, se presenta el esquema de validación planteado para evaluar el desempeño del seguidor implementado. En dicha subsección se utilizan AFPs sintéticas para efectuar la alineación, de manera que se pueda comparar la alineación obtenida con la alineación esperada. Durante esta subsección no se hace uso del índice LSH, se utiliza una búsqueda secuencial, puesto que solamente se

quiere validar el comportamiento del algoritmo de seguimiento. Como resultado de la subsección de validación, se obtienen los valores recomendados para los parámetros propuestos en la Sección 3.5.1. En la segunda subsección, se utiliza nuevamente una búsqueda secuencial pero ya no se utilizan AFPs sintéticas sino que se realiza la alineación entre AFPs de dos interpretaciones diferentes, haciendo uso de los valores para los parámetros obtenidos durante la validación. Por último, en la tercera subsección se introduce el uso del índice LSH con el fin de disminuir la cantidad efectuada de comparaciones entre subfirmas. El propósito de esta última subsección es demostrar que se puede obtener una alineación equivalente a la obtenida utilizando búsqueda secuencial, pero realizando un número considerablemente menor de comparaciones entre subfirmas.

Las figuras presentadas en las tres siguientes subsecciones muestran el resultado de alineación obtenido entre una interpretación tomada como referencia con una interpretación online. La Figura 4.1 presenta un ejemplo de como se crean estas gráficas. Como se presenta en dicha figura, las gráficas de alineación son el resultado de mapear en el eje de las abcisas los momentos donde ocurren los eventos de la interpretación online con los momentos en que ocurren los eventos equivalentes en la interpretación de referencia en el eje de las ordenadas. Así, una gráfica de alineación que resulte en una línea a 45 grados implica que en ambas interpretaciones ocurren los mismos eventos en el mismo instante.

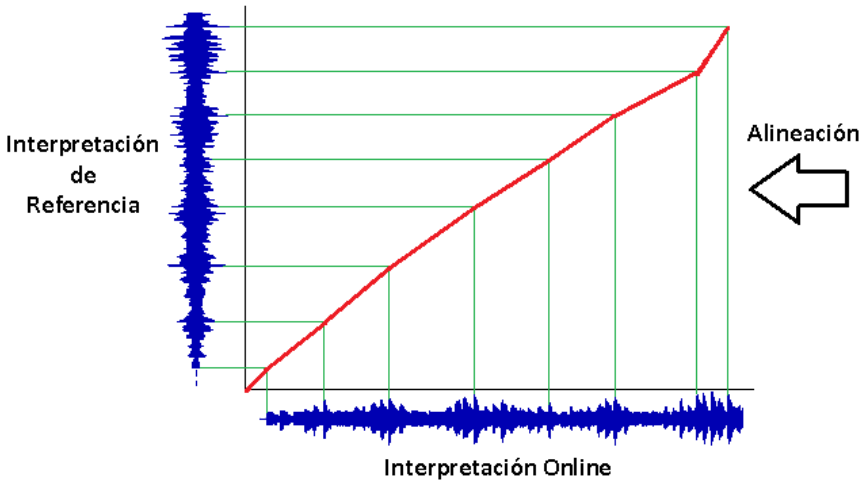


Figura 4.1: Ejemplo de gráfica de alineación.

Como nota adicional, por cuestiones de edición, las figuras mostradas a lo largo de este capítulo han sido reducidas en su tamaño lo cual puede dificultar su interpretación. Es por ello que todas las figuras de este capítulo han sido incluidas en su tamaño original en el Apéndice B de manera que se puedan consultar en caso de ser necesario.

4.1. Validación

Los sistemas de SF usualmente se evalúan registrando la cantidad de notas que se alinearon correctamente con la partitura de la interpretación. Para el presente trabajo, al no utilizar un seguimiento por nota, no contamos con una alineación base de comparación para evaluar el desempeño del seguidor. Esta diferencia en el enfoque utilizado, provoca que el esquema de seguimiento presentado en este trabajo sea difícil de comparar con los esquemas de seguimiento de partitura.

Por la razón anterior fue necesario idear un esquema propio de validación. La técnica de validación propuesta es similar a la utilizada en [Hu *et al.*, 2003b], en donde se modificaron artificialmente los archivos MIDI utilizados para evaluar la alineación obtenida. En nuestro caso, se crearon firmas de audio sintéticas. Las AFP sintéticas son AFPs modificadas artificialmente en puntos específicos conocidos, para simular las posibles diferencias que podrían ocurrir entre diversas interpretaciones. De esta manera, se conoce de antemano la alineación esperada entre las interpretaciones y se puede comparar con la alineación obtenida. Se hicieron tres tipos de modificaciones a las firmas:

1. Se replicaron algunas filas para simular situaciones en las que el intérprete sostiene alguna nota o acorde por un tiempo determinado.

2. Se removieron algunas filas para simular situaciones en las que el intérprete se "salta" parte de la partitura.

3. Se introdujo ruido en la AFP sintética de manera que sus vectores no coincidieran exactamente con los de la AFP que la originó.

A continuación se presentan algunos de los resultados de validación obtenidos. Los resultados mostrados fueron seleccionados específicamente para ejemplificar de manera

evidente las diferencias en el comportamiento del algoritmo de seguimiento que resultan de la variación de los parámetros k, *mode*, *wmd* y *distance*. Los parámetros n_b y $windowSize$ tienen sus valores fijos en $n_b = 17$ y $windowSize = 43$ respectivamente, como se expuso en la Sección 3.5. En cada caso se tomo una de las grabaciones con las que se cuenta y se realizaron modificaciones aleatorias a su firma de audio. En cada figura se indica la grabación utilizada, las modificaciones realizadas a la huella, el resultado esperado y el resultado obtenido por el esquema de seguimiento implementado.

Una nota importante es que, para los resultados de esta sección mostrados a continuación, se utilizó una búsqueda de manera secuencial, es decir, no se utilizó el índice LSH. Lo anterior debido a que la finalidad de esta sección es validar el funcionamiento únicamente del esquema de seguimiento y no el probar el desempeño del índice. A final de cuentas, la utilización del índice LSH simplemente tiene como finalidad que se realice un menor número de comparaciones entre subfirmas que el número de comparaciones realizado al utilizar la búsqueda secuencial. Así, al no utilizar el índice LSH se eliminan los posibles problemas de desempeño ocasionados por el mismo. Es por ello que para esta sección se descartan los parámetros que determinan la cantidad de bits por vector a utilizar para calcular la función hash h_{mb}, el número de variaciones por vector que se crean b_v y la cantidad de mapas hash a utilizar h_{mm} puesto que dichos parámetros controlan el comportamiento del índice LSH. Los resultados al utilizar el índice LSH se pueden observar en la Sección 4.3.

En las subsecciones siguientes, se pretende mostrar cuales son los resultados de variar uno de los parámetros a la vez mientras el resto de ellos se mantienen fijos. Lo anterior con el fin de determinar los valores o rangos de valores óptimos para cada parámetro y posteriormente utilizar los valores obtenidos en las Secciones 4.2 y 4.3 donde ya no se utilizan AFPs sintéticas sino AFPs de dos interpretaciones distintas.

4.1.1. Variación del modo de seguimiento, *mode*

El parámetro *mode* determina la manera en la que se elige la siguiente posición de entre la lista de candidatos posibles. Los cuatro modos implementados fueron expuestos a profundidad en la Sección 3.4. En las Figuras 4.2(a), 4.2(b), 4.2(c) y 4.2(d) se presenta el comportamiento de cada uno de los diferentes modos implementados. Se le recuerda al

lector que todas las figuras pueden ser consultadas en su tamaño original en el Apéndice B.

Se utilizó la grabación del *Étude* #16 la cual tiene una duración de 1 minuto con 11 segundos y su firma tiene 6126 vectores. El vector 900 se replicó 150 veces, el vector 1720 130 veces, los vectores 2750 a 3850, 3400 a 3550 y 4650 a 4780 fueron eliminados y, por último el vector 5300 se replicó 100 veces.

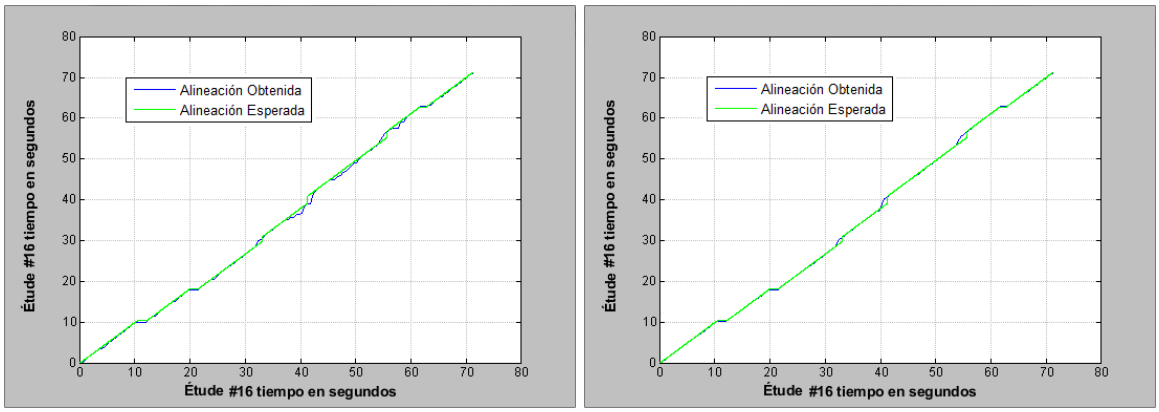
En esta sección se pretende demostrar los cambios ocurridos en el algoritmo de seguimiento que son resultado de cambiar el modo de elección de la siguiente posición p_n, por lo tanto, los parámetros k, wmd y $distance$ tendrán valores fijos. Sin embargo, se realizó una extensa experimentación con diferentes valores de k, wmd y $distance$. Los valores presentados a continuación se eligieron porque brindan resultados que resultan adecuados para observar de manera clara las ventajas y desventajas de los distintos modos implementados. los parámetros se utilizaron con los siguientes valores: $k = 20$, $wmd = 4$, $distance = Levenshtein$.

El Modo 1, ejemplificado en la Figura 4.2(a), elige el candidato más cercano en tiempo por lo que favorece pasos pequeños, lo anterior se puede observar en dicha figura donde el ejemplo más claro se presenta al rededor de los segundos $37 - 42$. El algoritmo elige posiciones demasiado cercanas por lo cual se aleja del resultado esperado.

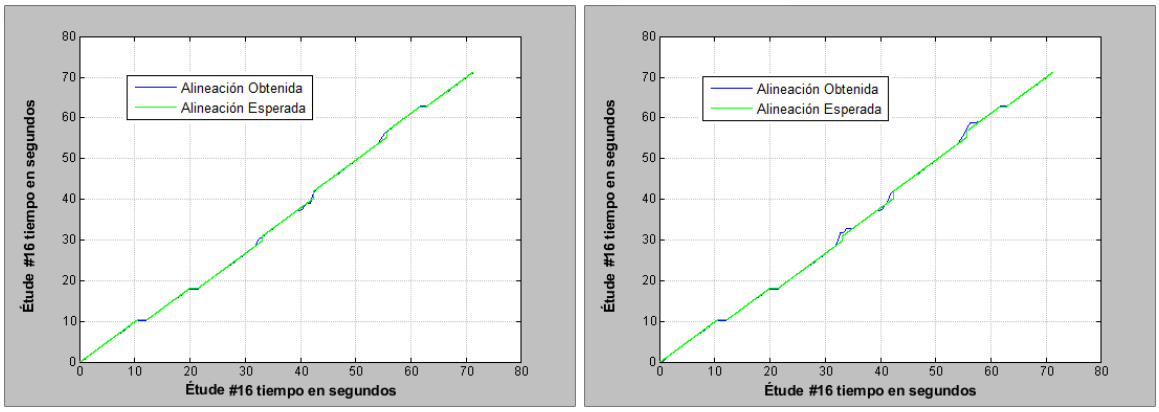
El Modo 2, ejemplificado en la Figura 4.2(b), elige al candidato más cercano en distancia, ésto puede tener resultados no deseados como se puede corroborar en la misma figura alrededor de los segundos 40 y 54, donde el algoritmo elige un candidato muy parecido pero lejano y se aleja del resultado esperado.

El Modo 3, ejemplificado en la Figura 4.2(c), es una combinación de los Modos 1 y 2. Elige al candidato considerado como más adecuado entre el más cercano en tiempo y el más cercano en distancia. Se observa en la Figura 4.2(c) que este modo tiene un comportamiento superior a los Modos 1 y 2.

El Modo 4, ejemplificado en la Figura 4.2(d), calcula una extrapolación lineal a partir de las últimas dos posiciones elegidas y elige al candidato más cercano a la extrapolación calculada. Se observa en dicha figura que este modo tiene un comportamiento adecuado pero no superior al Modo 3. En particular se observa que alrededor de los segundos 54-57 hay una separación considerable de la alineación esperada.



(a) **(b)**



(c) **(d)**

Figura 4.2: Comportamiento del seguimiento con los diferentes modos. (a).- Modo 1. (b).- Modo 2. (c).- Modo 3. (d).- Modo 4.

En conclusión, se determinó que el Modo 3 es el que presenta el comportamiento más adecuado y será el modo de selección utilizado en los experimentos posteriores.

4.1.2. Variación del cálculo de distancia, *distance*

El parámetro *distance* determina cuál de las tres distancias implementadas se utiliza para calcular la distancia entre las subfirmas. Las distancias que se utilizaron fueron Levenshtein, Hamming y LCS (véase la Sección 3.3).

Para los experimentos mostrados a continuación, se utiliza la misma grabación que en la sección anterior con las mismas modificaciones y con todos los mismos valores en todos los parámetros, a excepción del parámetro *mode*, el cual se utiliza con un valor de tres ($mode = 3$), puesto que se observó en la sección previa que dicho modo es el que presenta un mejor comportamiento.

En las Figuras 4.3(a), 4.3(b), 4.3(c) se muestra cómo al utilizar funciones de distancia diferentes para efectuar los cálculos de distancia, el comportamiento del algoritmo cambia puesto que las listas de candidatos obtenidas son diferentes.

En la Figura 4.3(a) se observa el resultado de utilizar la distancia de Hamming. La alineación obtenida es buena, salvo alrededor de los segundos 39-46, donde existe una diferencia considerable con la alineación esperada.

En la Figura 4.3(b) se observa el resultado de utilizar la distancia LCS. La alineación obtenida muestra el mismo problema que al utilizar la distancia de Hamming, aunque es un poco menor la diferencia con la alineación esperada.

Por último, en la Figura 4.3(c) se observa el resultado de utilizar la distancia de Levenshtein. Se puede apreciar que la alineación obtenida es casi igual a la alineación esperada, por lo que podemos concluir que el utilizar la distancia de Levenshtein nos brinda un mejor comportamiento. Por ello se utilizará la distancia de Levenshtein en el resto de los experimentos.

4.1.3. Variación del tamaño de la lista de candidatos, k

El parámetro k determina la cantidad de candidatos que se obtendrán para de entre ellos elegir la siguiente posición. Es evidente que si se tienen muy pocos candidatos

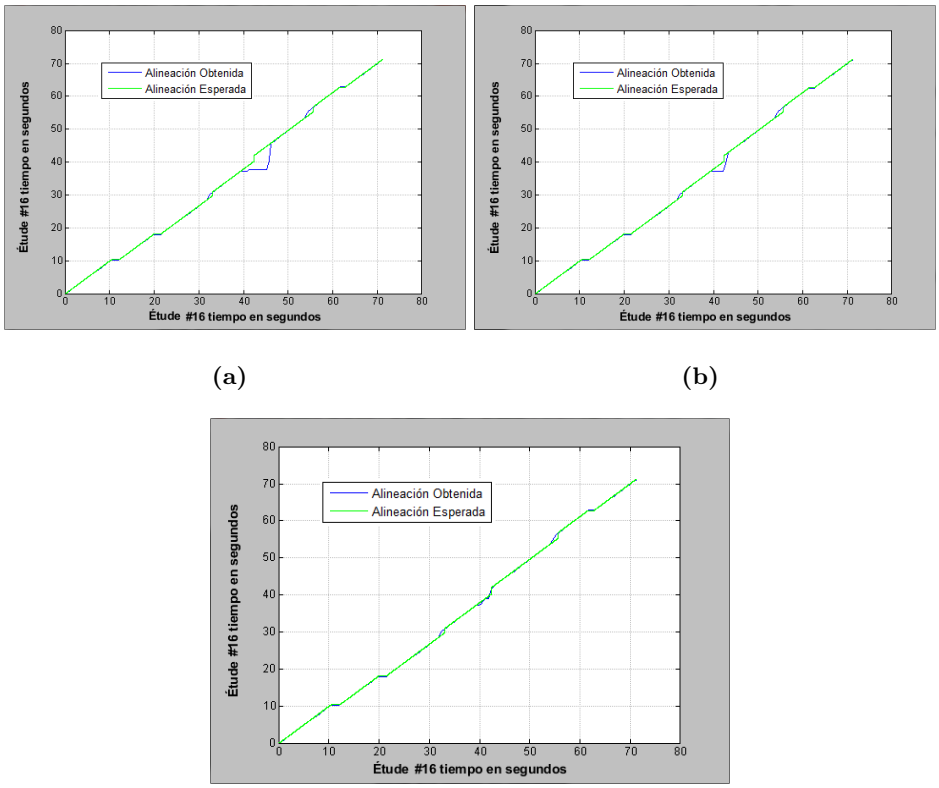


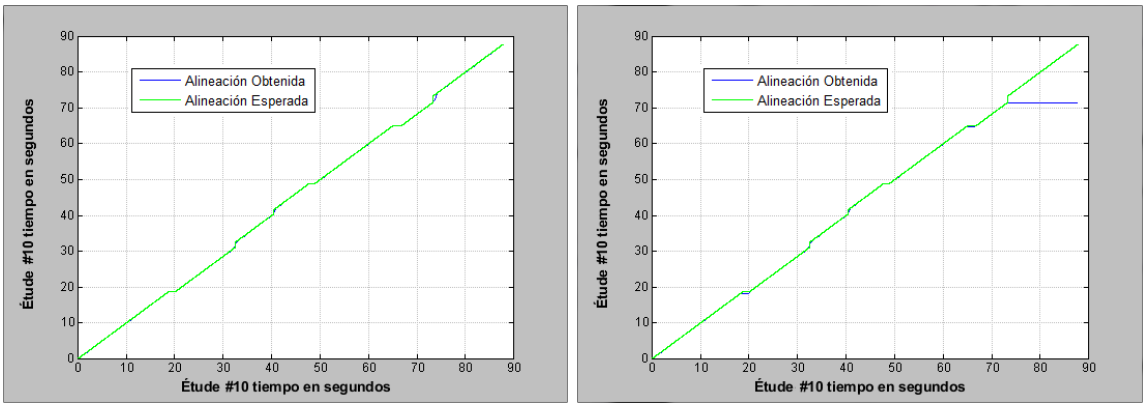
Figura 4.3: Comportamiento del seguimiento con las diferentes distancias (a).- Hamming. (b).- LCS. (c).- Levenshtein.

para elegir, será altamente probable que no se encuentre uno entre ellos que cumpla los requisitos del seguimiento (véase la Sección 3.4). Por otro lado si se tienen demasiados candidatos la situación se revierte, es muy probable que se encuentre un candidato que cumpla los requisitos, sin embargo, dicho candidato podría resultar ser inadecuado debido a que podría ser muy diferente a la subfirma de consulta.

Para ejemplificar ambos casos se utilizan dos ejemplos distintos. Para el primer caso (k bajo) se utilizó el *Étude* #10 que tiene una duración de 1 minuto y 27 segundos, por lo cual su AFP tiene 7554 vectores. El vector 1600 se replicó 120 veces, los vectores 2800-2920 y 3500-3600 se eliminaron, los vectores 4100 y 5600 se replicaron 100 y 150 veces respectivamente y, por último, los vectores 6300-6450 se eliminaron. Se utilizaron los

siguientes valores para el resto de los parámetros: $wmd = 4$, $distance = Levenshtein$, $mode = 3$.

En la Figura 4.4(a) se presenta el comportamiento del seguimiento con un valor de k adecuado, en este caso, $k = 20$. Se observa en dicha figura que la alineación obtenida y la alineación esperada son prácticamente iguales. Así mismo, en la Figura 4.4(b) se presenta el comportamiento del seguimiento con un valor de k demasiado bajo, en este caso, $k = 10$. Se puede observar en esta figura que al final de la interpretación, alrededor del segundo 73, el algoritmo es incapaz de encontrar un candidato adecuado para la siguiente posición por lo cual se estanca y permanece en la misma posición hasta el final.



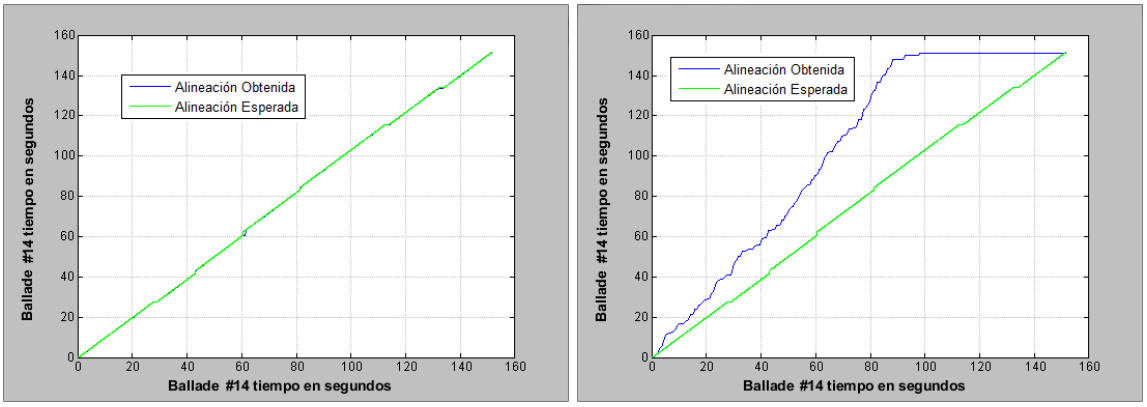
(a) **(b)**

Figura 4.4: Primer ejemplo del comportamiento del seguimiento con diferentes valores de k (a).- Valor adecuado. (b).- Valor bajo.

Para el segundo caso (k alto) se utilizó la *Ballade* #14 con una duración de 2 minutos con 31 segundos y AFP de 13040 vectores. El vector 2400 se replicó 100 veces, los vectores 3700-3820, 5200-5350 y 7000-7100 se eliminaron y, por último, los vectores 9700 y 11400 se replicaron 120 y 150 veces respectivamente. Se utilizaron los siguientes valores para el resto de los parámetros: $wmd = 4$, $distance = Levenshtein$, $mode = 3$.

En la Figura 4.5(a) se presenta el comportamiento del seguimiento con un valor de k adecuado, en este caso, $k = 35$. Se observa que la alineación obtenida y la alineación esperada son prácticamente iguales. De la misma forma, en la Figura 4.5(b) se presenta el

comportamiento del seguimiento con un valor de k alto, en este caso, $k = 100$. Se puede observar en la figura que al tener tantos candidatos, el algoritmo siempre encuentra alguno que cumple con los requisitos y avanza rápidamente llegando al final de la interpretación antes de lo adecuado. Como se mencionó en las Secciones 3.4 y 3.5, este efecto puede mitigarse con el parámetro *wmd* pero podría traer otros efectos negativos en el comportamiento.



(a) **(b)**

Figura 4.5: Segundo ejemplo del comportamiento del seguimiento con diferentes valores de k (a).- Valor adecuado. (b).- Valor alto.

En los experimentos realizados con las grabaciones mencionadas, se encontró que generalmente se puede elegir un valor de k adecuado en el rango de $\approx [15-45]$, dependiendo del tamaño de la interpretación. Para interpretaciones más largas se requiere un valor k más grande. Aunque valores menores o mayores a este rango podrían tener un funcionamiento adecuado, dependiendo del caso en particular.

4.1.4. Variación del tamaño de los saltos, *wmd*

El parámetro *wmd* establece un límite para la elección de la siguiente posición. En la Sección 3.5.1 se brinda una explicación a detalle del funcionamiento de este parámetro. En resumen, determina qué tan lejos (en tiempo) puede estar un candidato para que pueda ser considerado como siguiente posición.

Para los experimentos siguientes se utilizó el *Étude* #5 con duración de 1 minuto

con 10 segundos y cuya AFP tiene 6027 vectores. El vector 1000 se replicó 100 veces y el vector 2450 se replicó 150 veces. A su vez, se eliminaron los vectores 3800-3900 y 5100-5250. Se utilizaron los siguientes valores para el resto de los parámetros: $k = 20$, $distance = Levenshtein$, $mode = 3$.

En la Figura 4.7(a) se utiliza un valor *wmd* adecuado. En particular se utilizó $wmd = 4$. Como se puede observar en la figura, el valor utilizado permite una alineación adecuada de la interpretación. Ambos "saltos" creados artificialmente se solventan de manera adecuada.

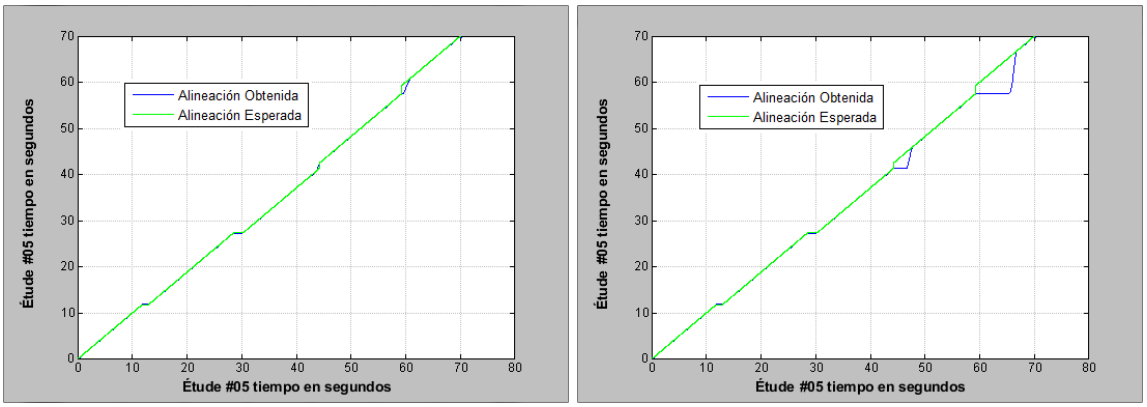
En la Figura 4.6(b) se observa el resultado de utilizar un valor *wmd* demasiado bajo. En este caso en particular se utilizó un valor de $wmd = 2$. El seguimiento resulta adecuado hasta el punto donde aparece el primer "salto". Puesto que el "salto" generado es de 100 posiciones (se borraron los vectores 3800-3900) y $wmd = 2$, el algoritmo no encuentra un candidato que cumpla con los requisitos:

$$windowMultiplier * windowSize = 2 * 43 = 86 < 100$$

Debido a ésto, se mantiene en la misma posición hasta que el valor de $windowMultiplier$ ha aumentado lo suficiente para encontrar un candidato adecuado. Ésto ocurre hasta el segundo 47 aproximadamente. Lo mismo sucede, de manera aún más evidente, con el segundo salto de 150 posiciones (se borró 5100-5250), el cual se recupera hasta el segundo 66 aproximadamente.

En la Figura 4.7(b) se observa el resultado de utilizar un valor de *wmd* demasiado alto. En este caso en particular se utilizó $wmd = 9$, lo cual permite que la siguiente posición pueda ser elegida 387 posiciones adelante de la posición anterior ($\approx 4.5s$). Adicionalmente se aumentó el valor de k de 20 a 40 para proveer al algoritmo con más posibles candidatos y hacer el efecto más evidente.

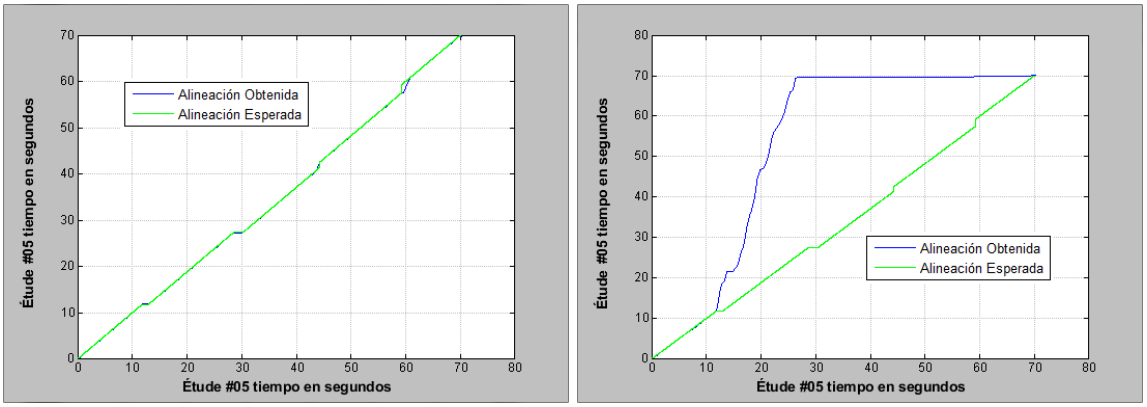
Se observa en la Figura 4.7(b) que en cuanto se llega a la primera modificación (vector 1000 replicada 100 veces) el algoritmo, en lugar de permanecer en el mismo índice, como se esperaría, encuentra un candidato muy alejado en tiempo de la posición anterior que sin embargo cumple los requisitos para ser elegido como posición siguiente debido al valor tan alto de *wmd*. La situación anterior se repite a lo largo del seguimiento, lo cual



(a) **(b)**

Figura 4.6: Primer ejemplo del comportamiento del seguimiento con diferentes valores de wmd (a).- Valor adecuado. (b).- Valor bajo.

evita que éste se recupere y el resultado es evidentemente pésimo.



(a) **(b)**

Figura 4.7: Segundo ejemplo del comportamiento del seguimiento con diferentes valores de wmd (a).- Valor adecuado. (b).- Valor alto.

El parámetro *wmd* es entonces sumamente importante y una elección adecuada del mismo puede ser la diferencia entre un seguimiento apropiado y uno muy malo. Elegir un valor demasiado bajo puede ocasionar que algunos de los saltos en la interpretación no sean manejados de manera adecuada; y elegir un valor demasiado alto (en particular en

conjunto con un valor de k alto), ocasiona que el algoritmo avance en lugar de mantenerse donde debería.

En los experimentos realizados a lo largo de esta sección, se conocen de antemano las modificaciones realizadas a las huellas de audio y, por lo tanto, se puede elegir siempre un valor para *wmd* que sea adecuado. Este no será el caso cuando se comparen dos interpretaciones diferentes, así que se deberá elegir un valor *wmd* basándose en conocimientos empíricos u otra información que sea relevante, como puede ser la duración de ambas interpretaciones o las habilidades de los intérpretes.

Para el presente trabajo, y debido a que las grabaciones utilizadas fueron realizadas por pianistas profesionales, se puede estimar que las interpretaciones son lo suficientemente competentes y no existirán saltos mayores a tres segundos ($wmd <= 6$). Por lo tanto, se utilizará usualmente un valor de *wmd* en el rango $\{3, .., , 6\}$.

Para un recordatorio del funcionamiento e interacción entre los parámetros *wmd*, $windowMultiplier$ y $windowSize$, véase la Sección 3.5.1.

4.1.5. Introducción de ruido

Por último, en esta sección se plantea la introducción de ruido para afectar los vectores de la AFP sintética utilizada, con el fin de que éstos no coincidan con los vectores de la AFP original y probar comportamiento del seguimiento bajo estas condiciones.

El ruido se implementó de la siguiente manera: se establece un porcentaje de ruido que será introducido a la AFP sintética y se modifican aleatoriamente la cantidad de bits de la AFP indicados por dicho porcentaje. Por ejemplo, si se tiene una AFP de dimensiones [5 X 6] y se establece un porcentaje de ruido de 10 %, entonces se modificarán tres bits de manera aleatoria ($5 * 6 = 30 * 0.1 = 3$), si se establece un porcentaje de ruido del 50 % se modificarán aleatoriamente quince bits ($5 * 6 = 30 * 0.5 = 15$), etc.

Para estos experimentos se utilizó el *Étude* #1 con una duración de 1 minuto y 26 segundos y cuya AFP contiene 7406 vectores. Los vectores 1000 y 2150 se replicaron 100 y 130 respectivamente; los vectores 3100-3200, 4200-4330 y 5500-5620 se eliminaron; y por último, el vector 6700 se replicó 120 veces. El resto de los parámetros se utilizaron con los siguientes valores $k = 30$, $wmd = 4$, $mode = 3$ y $distance = Levenshtein$. En la figura 4.8

se observa la alineación obtenida antes de aplicar un porcentaje de ruido a la AFP sintética de referencia. Se puede observar en dicha figura que la alineación obtenida y la esperada son prácticamente iguales.

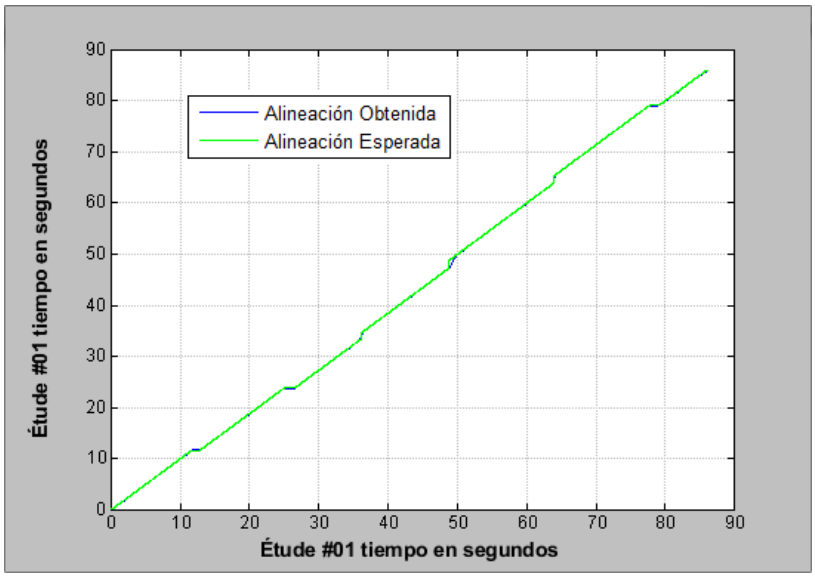
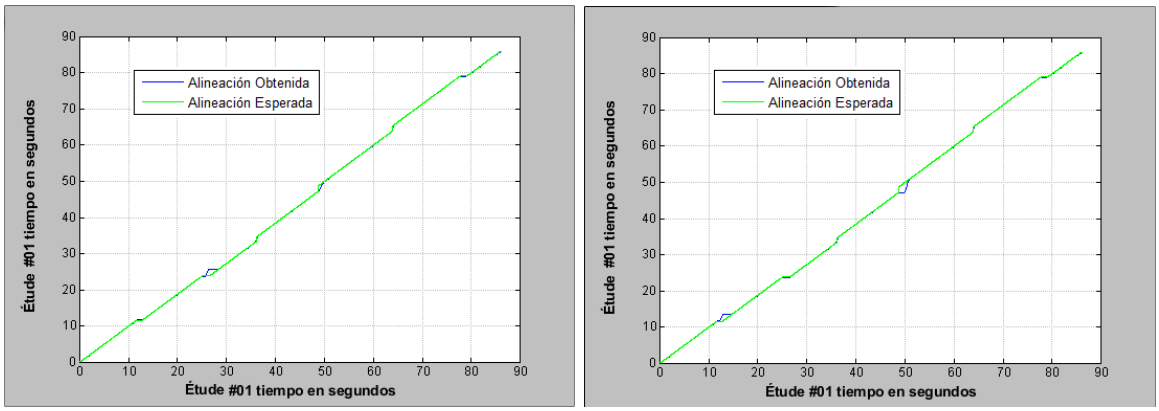
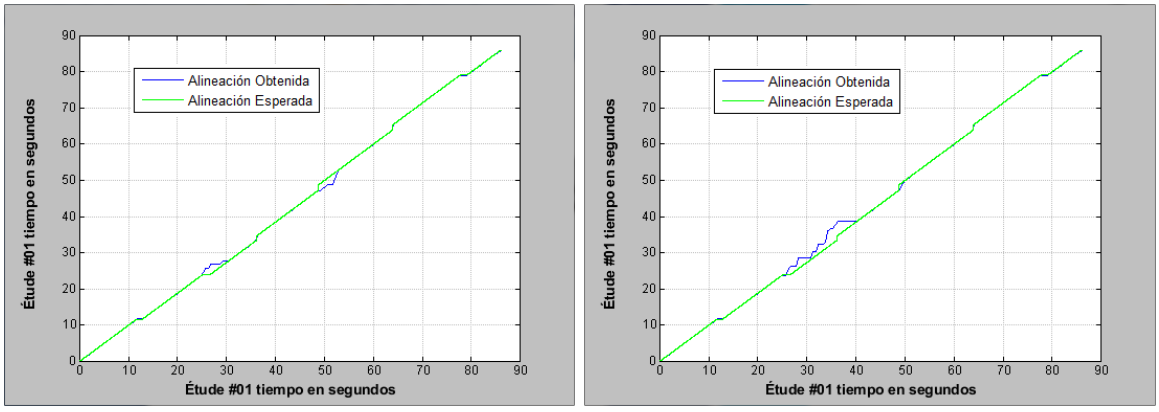


Figura 4.8: Alineación sin ruido.

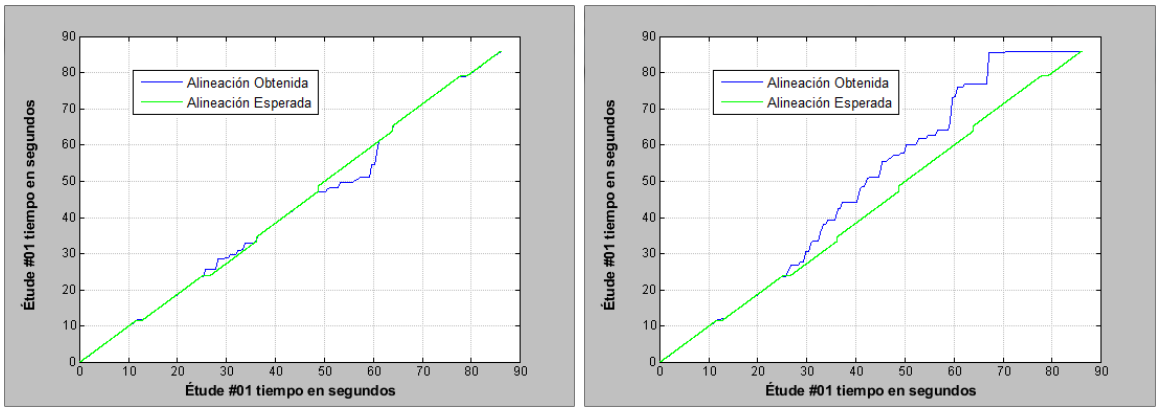
En las figuras 4.9(a)-4.9(f) siguientes, se muestran los resultados de alineación al ir incrementando gradualmente el porcentaje de ruido desde 10 % hasta 50 %. Se puede observar cómo la alineación obtenida se va degradando, conforme aumenta el porcentaje de ruido introducido. Sin embargo, también se puede observar que el esquema de seguimiento propuesto mantiene un comportamiento aceptable, hasta con un 25 % de ruido en la AFP sintética. Es importante mencionar que se realizó una vasta experimentación con el ruido puesto que, como las modificaciones son aleatorias, dos ejecuciones distintas generan modificaciones distintas y, por lo tanto, resultados distintos. Se registraron ejecuciones donde con tan sólo el 10 % de ruido, la alineación obtenida resultaba ser completamente inadecuada; mientras que en otras ejecuciones con un porcentaje de ruido tan alto como del 70 % la alineación obtenida era casi igual a la alineación esperada. Los resultados mostrados se eligieron porque, a juicio del autor, reflejan el comportamiento habitual del seguimiento para los porcentajes indicados.



(a) **(b)**



(c) **(d)**



(e) **(f)**

Figura 4.9: Comportamiento del seguimiento con diferentes porcentajes de ruido (a).- 10 %. (b).- 20 %. (c).- 25 %. (c).-30 %. (e).-35 %. (f).-50 %.

4.1.6. Resumen de la validación

En base a los resultados obtenidos en los experimentos se concluye que el algoritmo de seguimiento propuesto tiene un comportamiento adecuado, supeditado a la elección adecuada de sus parámetros.

En la Tabla 4.1 se presenta un resumen de los valores recomendados para los parámetros del algoritmo.

Parámetro	Valores Recomendados
k	$\{n \mid n \in N, 15 \leq n \leq 45\}$
distance	Levenshtein, LCS
mode	3, 4
wmd	$\{n \mid n \in N, 3 \leq n \leq 6\}$

Tabla 4.1: Valores recomendados para los parámetros de seguimiento

4.2. Resultados: Búsqueda Secuencial

En esta sección se presentan los resultados obtenidos al realizar la alineación de dos interpretaciones distintas, utilizando el algoritmo de seguimiento propuesto con búsqueda secuencial, cuya validación se presentó en la sección anterior. Estos resultados se tomarán como referencia en la Sección 4.3, en la cual se evalúa el desempeño del índice LSH.

De todos los experimentos realizados, se eligieron los ejemplos descritos en las subsecciones 4.2.1 a 4.2.7, debido a que en ellos se utilizan grabaciones de diferentes duraciones y características. Por tanto, ayudan a visualizar el comportamiento del algoritmo en diversas posibles situaciones que se pueden presentar.

Para cada ejemplo, se mencionan las grabaciones que fueron utilizadas y el número de comparaciones entre subfirmas que se realizan por cada subfirma de consulta. Se incluyen dos gráficas por cada ejemplo, puesto que para cada par de grabaciones utilizado se realizaron dos experimentos: primero utilizando una de las interpretaciones como interpretación de referencia y posteriormente utilizando la otra.

Para continuar con la evaluación del algoritmo de seguimiento, se tomaron manualmente las medidas de los tiempos de ejecución (*onset*) de las notas de cuatro de las

interpretaciones: *Étude* #01, *Étude* #02, *Étude* #07 y *Étude* #22. Con estas mediciones se puede crear una gráfica de alineación real de dichas interpretaciones y de esta manera comparar la alineación real con la alineación obtenida por el seguidor. Las mediciones mencionadas se pueden consultar en el Apéndice C. Los primeros seis ejemplos de esta sección muestran el resultado de alinear estas cuatro interpretaciones entre sí. Así, para estos primeros seis ejemplos 4.2.1 a 4.2.6 se incluye, además de los resultados obtenidos, la gráfica que representa la alineación real.

En todos los casos, los valores de k y wmd utilizados se tomaron de los rangos recomendados en la Sección 4.1.6. Las distancias de Levenshtein y LCS, así como el Modo 3 para elección de la posición siguiente, fueron utilizados puesto que mostraron los mejores resultados durante la validación y los experimentos realizados.

Todas las formas de onda de las grabaciones usadas, así como sus duraciones exactas, se pueden consultar en el Apéndice E con el fin de que el lector pueda constatar por sí mismo las características de cada interpretación, que se mencionan en los ejemplos presentados.

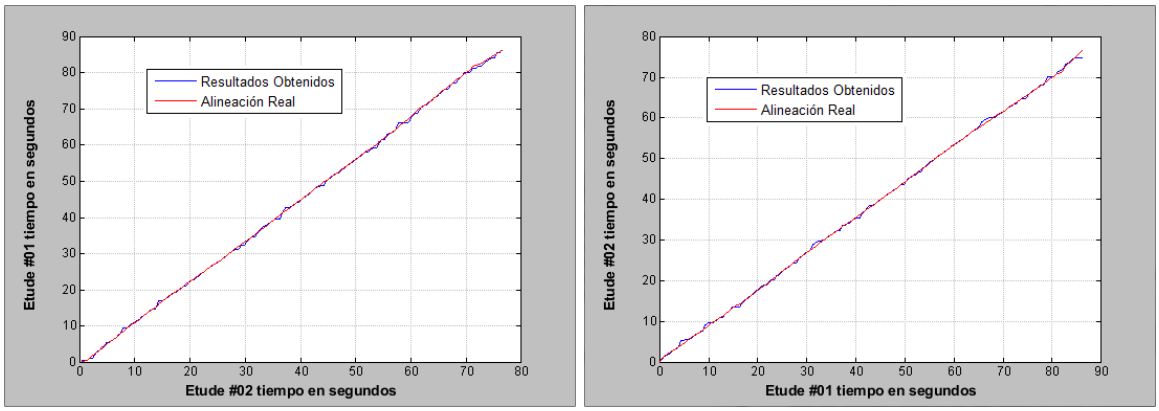
4.2.1. Ejemplo 1

Para este ejemplo se utilizaron las grabaciones #1 (E.1) y #2 (E.2) del Étude las cuales tienen duraciones muy diferentes. El Étude #1 tiene una duración de 1 minuto y 26 segundos, mientras que el Étude #2 tiene una duración de 1 minuto y 16 segundos.

Esta diferencia de 10 segundos entre las grabaciones nos indica que el intérprete del Étude #2 tiene un ritmo mucho más rápido que su contraparte del Étude #1. Las interpretaciones terminan aproximadamente en las marcas $1:24$ y $1:15$ para el Étude #1 y #2 respectivamente. Además, existe un silencio ligeramente más largo al inicio del Étude #2.

En la Figura 4.10(a), donde se utiliza el Étude #1 como referencia, debido al tamaño de la AFP del Étude #1 se realizan 7363 comparaciones por cada subfirma buscada. En la Figura 4.10(b) se observa el resultado de realizar la alineación de manera inversa (utilizando el Étude #2 como referencia). El resultado es el esperado, ya que la alineación tiene un comportamiento opuesto al caso anterior. Para este caso, se realizan 6535 comparaciones

por subfirma.



(a) **(b)**

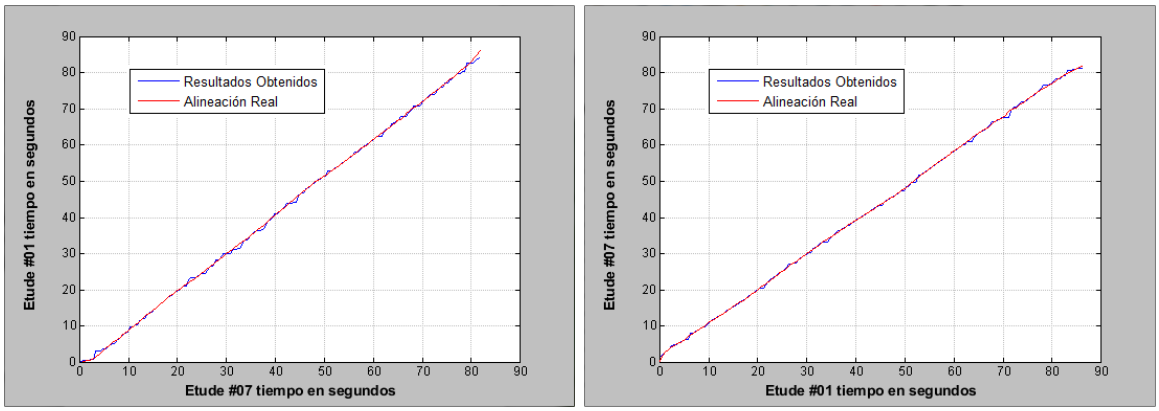
Figura 4.10: Alineación de los Études #01 y #02 (a).- Étude #01 utilizado como referencia. (b).- Étude #02 utilizado como referencia.

En ambos casos se puede observar que el seguimiento obtenido (línea azul), es sumamente parecido a la alineación real (línea roja) y se alcanza una diferencia de entre 9 y 10 segundos, como era de suponerse debido a las características de ambas interpretaciones. Los parámetros utilizados para obtener estos resultados fueron $k = 40$, $wmd = 4$ y $mode = 3$ para ambos casos. Se utilizó la distancia LCS para el primer caso y la de Levenshtein para el segundo.

4.2.2. Ejemplo 2

Para este ejemplo se utilizaron las grabaciones #1 (E.1) y #7 (E.4) del Étude, las cuales tienen duraciones de 1 minuto y 26 segundos y 1 minuto y 21 segundos, respectivamente. Existe un silencio ligeramente más largo al inicio del Étude #7.

En la figura 4.11(a) se observa el resultado del seguimiento obtenido cuando se utilizó el Étude #1 como referencia. Por el tamaño de la AFP del Étude #1 se realizan 7363 comparaciones por subfirma de consulta. Por otro lado, la figura 4.11(b) muestra la alineación obtenida al utilizar el Étude #7 como referencia. En este caso se realizan 6998 comparaciones por cada subfirma de consulta.



(a) **(b)**

Figura 4.11: Alineación de los Études #01 y #07 (a).- Étude #01 utilizado como referencia. (b).- Étude #07 utilizado como referencia.

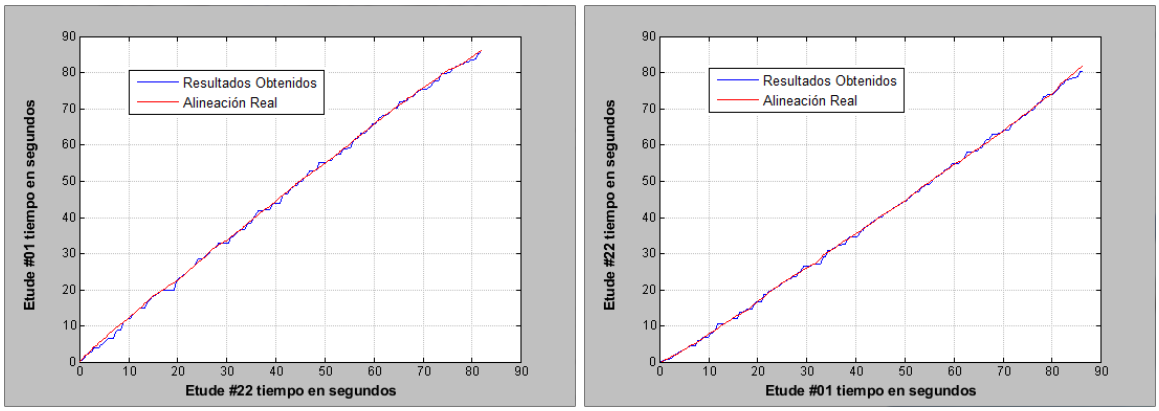
Nuevamente se tienen excelentes resultados, pues en ambos casos la alineación obtenida (línea azul) es prácticamente igual a la alineación real (línea roja), entre estas dos interpretaciones. Los parámetros utilizados para obtener estos resultados fueron $k = 45$, $wmd = 5$, $mode = 3$ y $distance = LCS$ para el primer caso y $k = 40$, $wmd = 4$, $mode = 3$ y $distance = Levenshtein$ en el segundo caso.

4.2.3. Ejemplo 3

Para este ejemplo se utilizaron las grabaciones #1 (E.1) y #22 (E.8) del Étude, las cuales tienen duraciones de 1 minuto y 26 segundos y 1 minuto y 21 segundos respectivamente.

En la figura 4.12(a) se observa el comportamiento demostrado por el seguimiento cuando se utilizó el Étude #1 como referencia. Por el tamaño de la AFP del Étude #1 se realizan 7363 comparaciones por subfirma de consulta. En la figura 4.12(b) se muestra la alineación obtenida al utilizar el Étude #22 como referencia. En este caso se realizan 6989 comparaciones por cada subfirma de consulta.

En estos casos también se obtuvieron buenos resultados, pues las alineaciones obtenidas son muy parecidas a las alineaciones reales entre ambas interpretaciones. Sin



(a) **(b)**

Figura 4.12: Alineación de los Études #01 y #22 (a).- Étude #01 utilizado como referencia. (b).- Étude #22 utilizado como referencia.

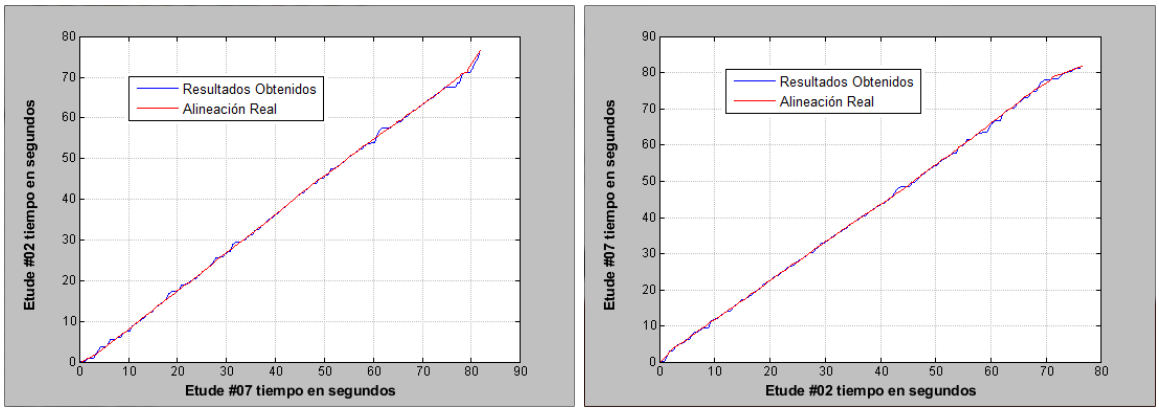
embargo, en el primer caso, utilizando el Étude #1 como referencia, se puede observar que al inicio (segundos 5-8) existe una diferencia entre la alineación obtenida y la real. De igual manera, en el mismo caso, alrededor del segundo 17 el algoritmo de seguimiento se mantuvo en la misma posición y se recuperó hasta el segundo 20 aproximadamente. Los parámetros utilizados para obtener estos resultados fueron $k = 40$, $wmd = 4$ y $mode = 3$ en ambos casos. Para las distancias, se utilizó LCS en el primer caso y Levenshtein en el segundo.

4.2.4. Ejemplo 4

Para este ejemplo se utilizaron los Études #2 (E.1) y #07 (E.4). El Étude #2 tiene una duración de 1 minuto y 16 segundos, mientras que el Étude #07 tiene una duración de 1 minuto y 21 segundos respectivamente.

En la figura 4.13(a) se observa el resultado del seguimiento obtenido cuando se utilizó el Étude #2 como referencia. Por el tamaño de la AFP del Étude #2 se realizan 6535 comparaciones por subfirma de consulta. Por otro lado, en la figura 4.13(b) se muestra la alineación obtenida al utilizar el Étude #7 como referencia. En este caso se realizan 6998 comparaciones por cada subfirma de consulta.

Los resultados obtenidos para la alineación entre estas dos interpretaciones, tam-



(a) **(b)**

Figura 4.13: Alineación de los Études #02 y #07 (a).- Étude #02 utilizado como referencia. (b).- Étude #07 utilizado como referencia.

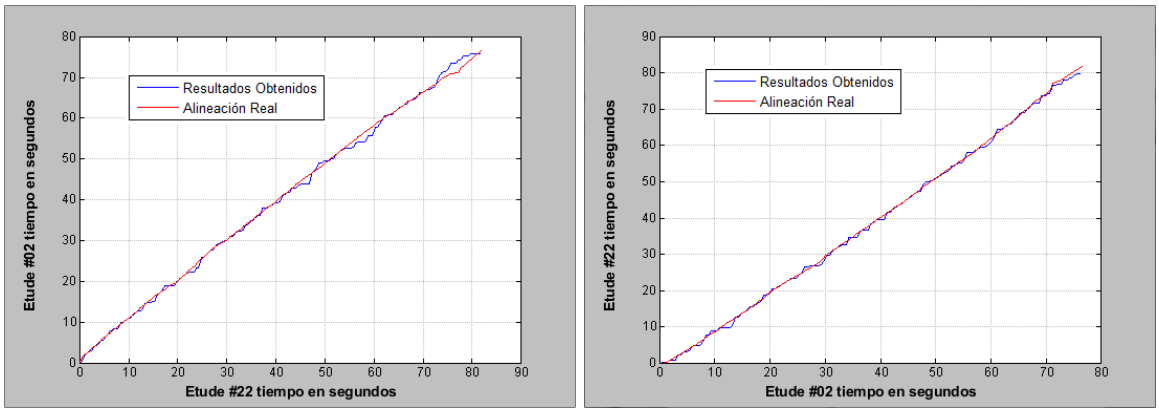
bién fueron muy buenos. Para el primer caso, cuando se usa el Étude #2 como referencia, existen algunas ligeras diferencias entre la alineación obtenida (línea azul) y la alineación real (línea roja), especialmente al final de la interpretación alrededor del segundo 75, donde se produce una diferencia de aproximadamente tres segundos entre ambas. Sin embargo en el segundo caso, cuando se utiliza el Étude #7 como referencia, la alineación obtenida es prácticamente igual a la alineación real. Los parámetros utilizados para obtener estos resultados fueron $k = 40$, $wmd = 4$ y $mode = 3$ en ambos casos. Para las distancias, se utilizó LCS en el primer caso y Levenshtein en el segundo.

4.2.5. Ejemplo 5

Para este ejemplo se utilizaron las grabaciones #2 y #22 del Étude. Con duraciones de 1 minuto con 16 segundos y 1 minuto con 21 segundos respectivamente. El Étude #2 tiene un ligero silencio al inicio de la grabación, mientras que el número 22 empieza casi inmediatamente.

En la figura 4.14(a) se puede apreciar el resultado de alineación obtenida al utilizar el Étude #2 como referencia. En este caso, existe un salto al inicio de la alineación debido al silencio inicial del Étude #22. Por el tamaño de la AFP del Étude #2 se realizan

6535 comparaciones por subfirma de consulta. El caso contrario, utilizar el Étude #22 de referencia, se presenta en la figura 4.14(b). Se aprecia que al inicio se produce un efecto inverso al de la figura anterior. En lugar de existir un salto, el algoritmo de seguimiento se mantiene en la posición inicial hasta que termina el silencio inicial del Étude #2. Se realizan 6989 comparaciones por cada subfirma de consulta.



(a) **(b)**

Figura 4.14: Alineación de los Études #2 y #22 (a).- Étude #2 utilizado como referencia. (b).- Étude #22 utilizado como referencia.

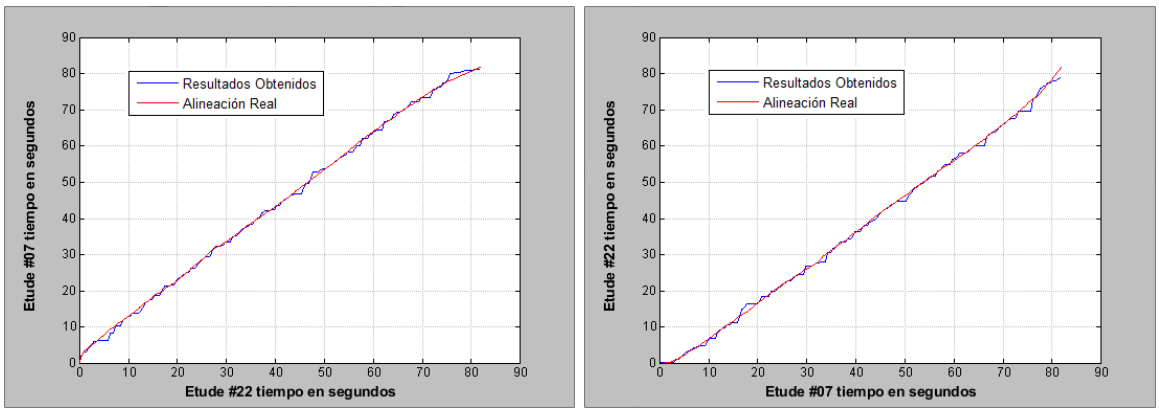
4.2.6. Ejemplo 6

El siguiente ejemplo es un caso bastante particular. Se utilizaron las grabaciones del Étude #7 (E.4) y #22 (E.8). Aparentemente, ambas interpretaciones deberían ser muy similares, puesto que sus grabaciones tienen duraciones casi exactamente iguales (1 minuto y 26 segundos) y su alineación debería ser casi perfecta.

Sin embargo, el Étude #7 tiene la particularidad de tener un silencio bastante largo al inicio (entre 2 y 2.5 segundos aproximadamente) y de terminar la interpretación casi simultáneamente a la grabación, es decir, casi no tiene silencio al final. A su vez, el Étude #22 comienza la interpretación de manera casi inmediata, pero contiene un silencio de entre 2 y 2.5 segundos al final de la grabación. Estas características se compensan entre sí, ocasionando que la alineación entre ambas interpretaciones sea casi perfecta pero con un

desplazamiento de 2-2.5 segundos entre ellas.

En la Figura 4.15(a) se presenta el comportamiento de la alineación usando el Étude #7 como referencia. Se aprecia que, como el Étude #22 no tiene el silencio inicial, existe un salto al principio de la alineación. De igual forma, en la Figura 4.15(b) se observa el comportamiento usando el Étude #22 como referencia. En este caso, el algoritmo se mantiene en la misma posición durante el silencio inicial del Étude #7 y comienza a avanzar una vez que comienza la interpretación. Debido al tamaño de las firmas, se realizan 6998 y 6989 comparaciones en cada caso respectivamente, por cada subfirma de consulta.



(a) **(b)**

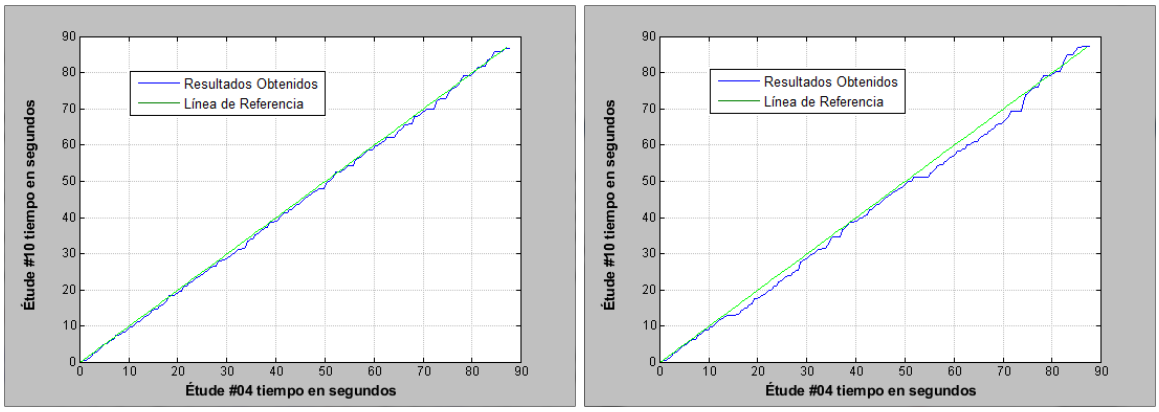
Figura 4.15: Alineación de los Études #7 y #22 (a).- Étude #7 utilizado como referencia. (b).- Étude #22 utilizado como referencia.

4.2.7. Ejemplo 7

Como último ejemplo, se presenta un experimento que se realizó para probar el comportamiento del seguimiento implementado, utilizando dos interpretaciones diferentes pero donde una de ellas ha sido modificada artificialmente, como se realizó en la Sección 4.1. Para este ejemplo se utilizaron grabaciones que tienen aproximadamente la misma duración y cuyos intérpretes tienen ritmos similares, por lo cual su alineación debería ser casi exacta. Las grabaciones utilizadas fueron los Études #4 (E.3) y #10(E.5), ambas con una duración de 1 minuto y 27 segundos.

La Figura 4.16(a) presenta el resultado de la alineación utilizando el Étude #10 como referencia. Al no contar con los tiempos de ejecución de las notas como en los ejemplos anteriores, en la gráfica se presenta una línea de referencia (línea verde) para ayudar al lector a percibir que la alineación entre ambas interpretaciones es casi perfecta. Es decir, ocurren los mismos eventos al mismo tiempo, en una interpretación que en la otra. Estos resultados de alineación fueron obtenidos antes de realizar modificaciones a la AFP.

Posteriormente, se modificó la AFP del Étude #4 de la siguiente manera: se replicó 150 veces el vector 1200, se borraron los vectores 2450-2600, se replicó 200 veces el vector 4500, se borraron los vectores 6300-6500 y, por último, se aplicó un 15 % de ruido. En la Figura 4.16(b) se puede observar el resultado. El seguimiento reacciona de la manera esperada ante las modificaciones: se mantiene en la misma posición cuando se llega al segundo 14 (vector $\approx$ 1200 del Étude #4; existe un salto al llegar aproximadamente al segundo 29(vector $\approx$ 2500) y nuevamente se mantiene en posición al llegar al segundo 52 (vector $\approx$ 4500).



(a) **(b)**

Figura 4.16: Alineación de los Études #4 y #10 utilizando el Étude #10 como referencia. (a).- Alineación previa modificación de la AFP. (b).- Alineación con la AFP del Étude #4 modificada artificialmente.

El comportamiento presentado al llegar al segundo 73 (vector $\approx$ 6300) se explica de la siguiente manera: se definió un valor para $wmd = 4$, lo cual equivale a 2 segundos ($4 * 0.5s$) o 172 ($4 * 43$) posiciones. El salto del vector 6300 es de 200 posiciones, por lo tanto,

excede el salto máximo. El índice elegido como siguiente posición, se encuentra por tanto demasiado lejos de la última posición elegida. El algoritmo entonces decide mantenerse en la misma posición y se incrementa el valor de *windowMultiplier* (sección 3.5.1). Lo anterior continúa hasta que se puede eligir una posición siguiente adecuada (alrededor del segundo 74.5, vector $\approx$ 6400).

4.2.8. Comparación de varias interpretaciones contra una sola referencia

Por último, a continuación se incluye la Figura 4.17 donde se presenta en una sola gráfica el resultado de alinear varias interpretaciones online contra una sola interpretación de referencia. Se utilizó la AFP del Étude #1 como AFP de referencia y se alinearon con ella las AFPs de los Études #2, #4, #7, #10, #11, #20 y #22, todas con duraciones distintas.

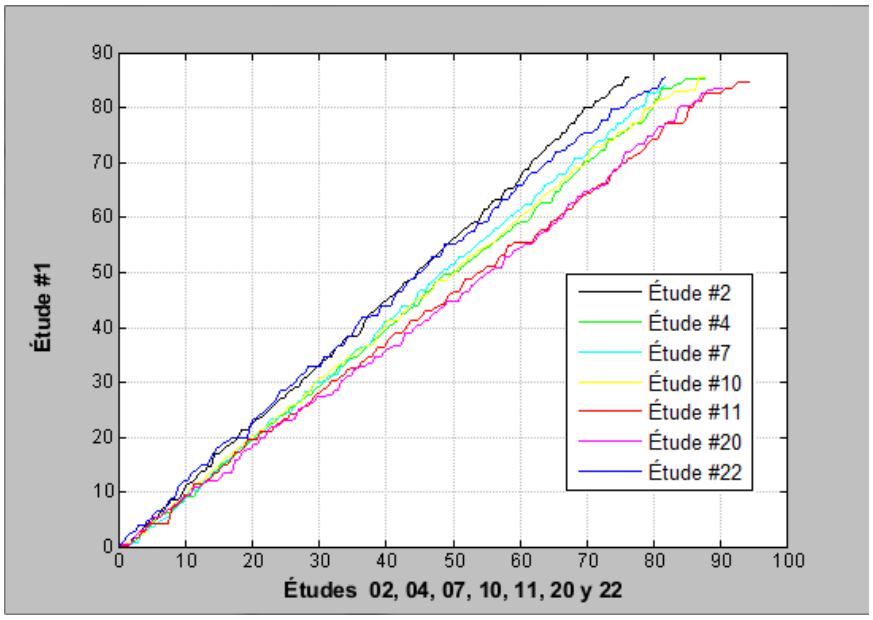


Figura 4.17: Varias interpretaciones online contra una sola referencia.

4.3. Resultados: Índice LSH

Esta sección se reserva para valorar el funcionamiento del índice LSH y justificar su utilización. Como se menciona en la Secciones 2.2 y 3.5.2, el funcionamiento del índice LSH depende principalmente de 2 cosas: la cantidad de instancias/mapas utilizados y la cantidad de bits que se utilizan para obtener la función hash.

Adicionalmente, debido a la manera en que se obtienen los índices candidato en la presente implementación (véase la Sección 3.1), la cantidad de variaciones creadas por cada vector de la matriz binaria a buscar, es otro factor a considerar.

En resumen, son tres los parámetros que determinarán el funcionamiento del índice LSH implementado:

1. $howManyMaps, h_{mm}$
2. $howManyBits, h_{mb}$
3. $bitVariations, b_v$

El efecto de aumentar o disminuir el valor de dichos parámetros se explica con mayor precisión en la Sección 3.5.2. Un resumen breve se encuentra en la Tabla 4.2.

Parámetro	Aumentar	Disminuir
h_{mm}	Aumenta la cantidad de candidatos	Disminuye la cantidad de candidatos
h_{mb}	Disminuye la cantidad de candidatos	Aumenta la cantidad de candidatos
b_v	Aumenta en gran medida la cantidad de candidatos	Disminuye en gran medida la cantidad de candidatos

Tabla 4.2: Resumen efectos de los parámetros LSH

Como ya se ha mencionado antes, el motivo de la utilización del índice LSH, en lugar de utilizar la búsqueda secuencial, consiste simplemente en obtener resultados equivalentes a ésta última, pero realizando un número considerablemente menor de comparaciones.

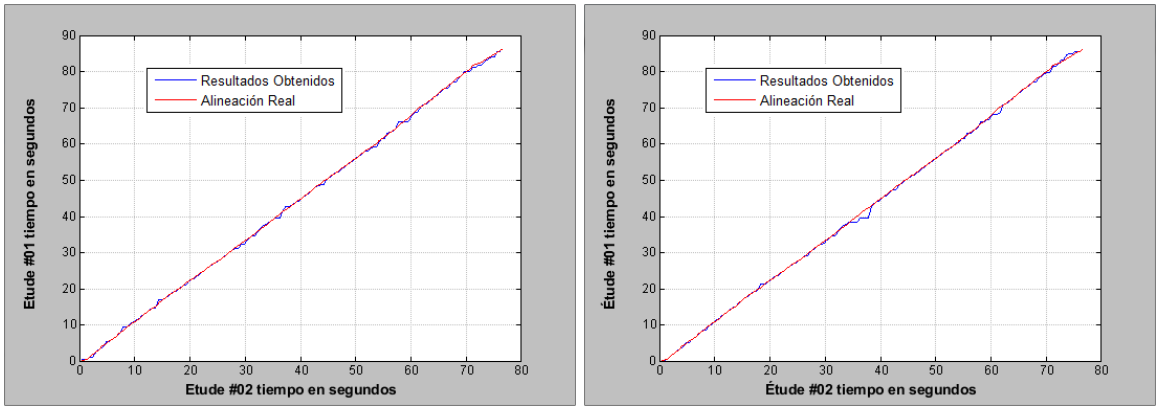
Para justificar el uso del índice, a continuación se muestran los resultados obtenidos en los diversos ejemplos que fueron explicados en la Sección 4.2, pero utilizando el índice

LSH en lugar de la búsqueda secuencial. Las Figuras 4.18 - 4.31 presentan las gráficas de alineación que resultaron al utilizar el índice LSH, lado a lado con el resultado correspondiente que se obtuvo usando búsqueda secuencial 4.2. Lo anterior con el fin de que puedan ser comparados visualmente. No se han incluido comentarios para las figuras mencionadas, pues basta con que el lector las compare y observe que la alineación resultante, al utilizar el índice LSH, es muy similar a la obtenida mediante búsqueda secuencial.

En la Tabla 4.3 se presenta un resumen de los valores de los parámetros h_{mm}, h_{mb} y b_v utilizados en cada ejemplo. Además se incluye la cantidad de comparaciones por subfirma de consulta que se realizan utilizando búsqueda secuencial, el número de comparaciones por subfirma efectuadas con el índice LSH y el porcentaje de reducción resultante.

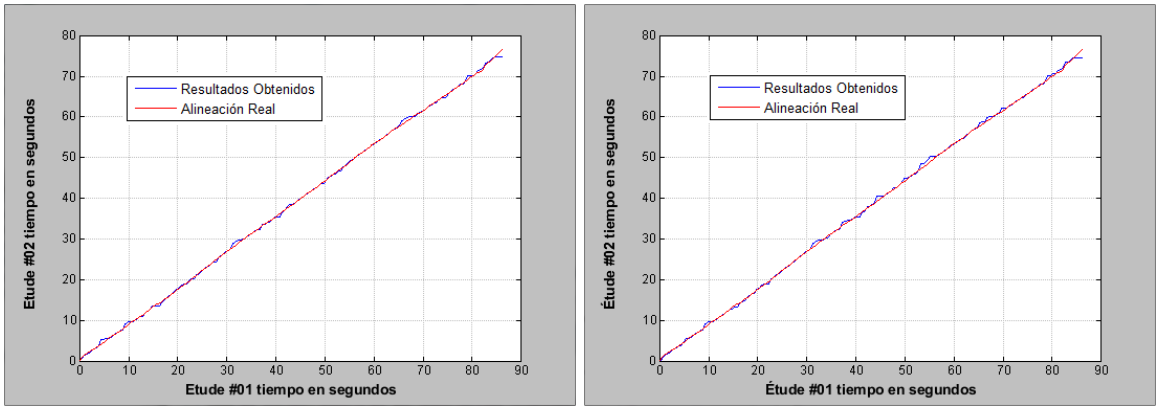
Número Ejemplo	**Figura**	**Secuencial**	**LSH**	**Reducción**	h_{mm}	h_{mb}	b_v
Ej.1 Caso (a)	4.18	7363	2016.4971	72.6 %	90	14	1
Ej.1 Caso (b)	4.19	6535	2261.1111	65.3 %	90	14	1
Ej.2 Caso (a)	4.20	7363	1138.7976	84.5 %	90	15	1
Ej.2 Caso (b)	4.21	6998	1212.0487	82.6 %	90	15	1
Ej.3 Caso (a)	4.22	7363	2465.1560	66.5 %	30	13	1
Ej.3 Caso (b)	4.23	6989	1994.0060	71.4 %	100	15	1
Ej.4 Caso (a)	4.24	6535	1136.1830	82.6 %	80	15	1
Ej.4 Caso (b)	4.25	6998	1043.8841	85 %	80	15	1
Ej.5 Caso (a)	4.26	6535	1101.5816	83.1 %	80	15	1
Ej.5 Caso (b)	4.27	6989	1047.1585	85 %	80	15	1
Ej.6 Caso (a)	4.28	6998	2777.0975	60 %	40	13	1
Ej.6 Caso (b)	4.29	6989	1141.4878	83.6 %	100	15	1
Ej.7 AFP sin modificar	4.30	7511	1197.5852	84 %	100	15	1
Ej.7 AFP modificada	4.31	7511	2438.5000	67.5 %	100	14	1

Tabla 4.3: Resultados con el Índice LSH.



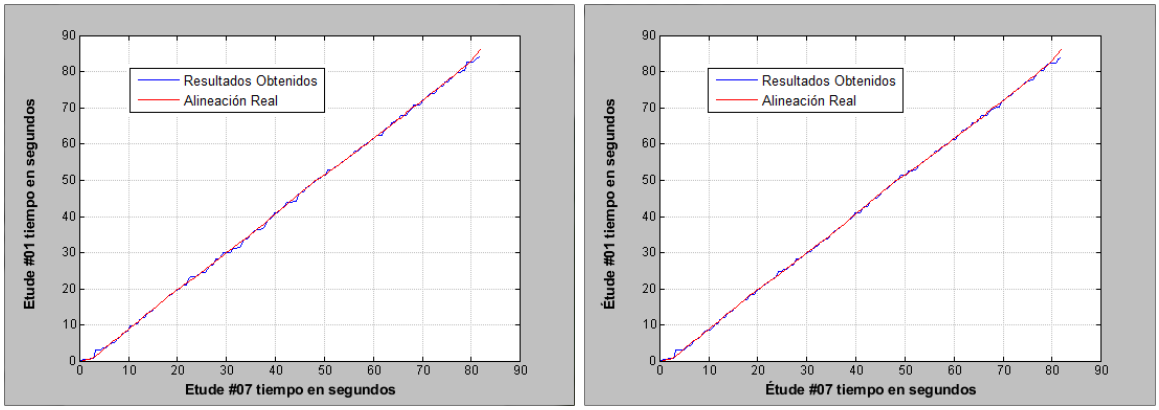
(a) **(b)**

Figura 4.18: Alineación de los Études #1 y #2 utilizando #1 como referencia (a).- Búsqueda Secuencial. (b).- Índice LSH.



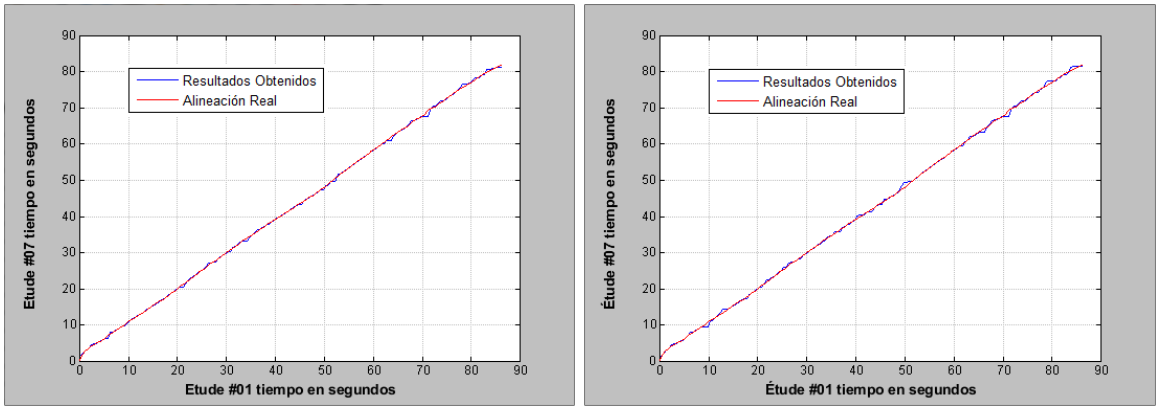
(a) **(b)**

Figura 4.19: Alineación de los Études #1 y #2 utilizando #2 como referencia (a).- Búsqueda Secuencial. (b).- Índice LSH.



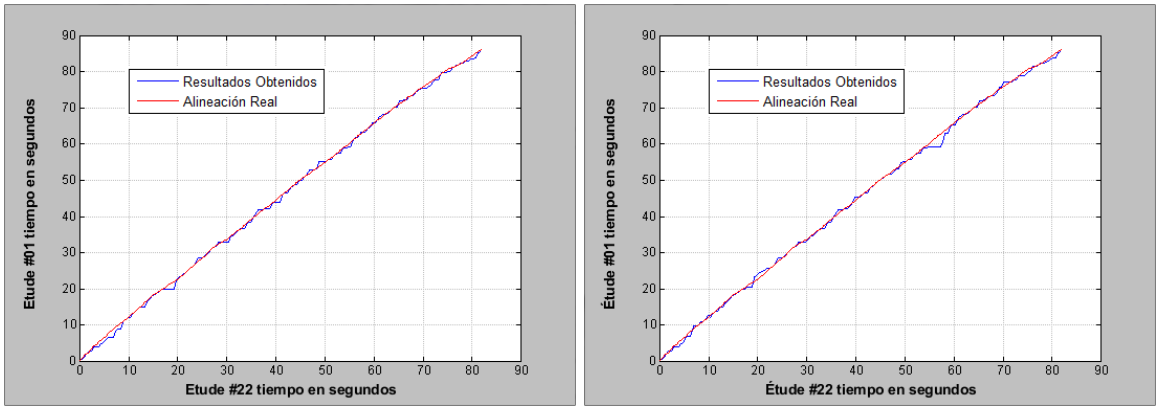
(a) **(b)**

Figura 4.20: Alineación de los Études #1 y #7 utilizando #1 como referencia (a).- Búsqueda Secuencial. (b).- Índice LSH.



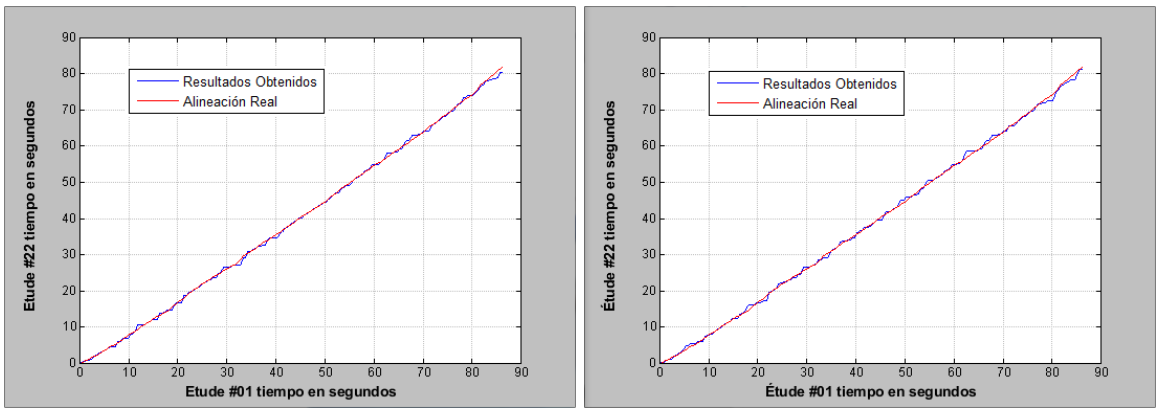
(a) **(b)**

Figura 4.21: Alineación de los Études #1 y #7 utilizando #7 como referencia (a).- Búsqueda Secuencial. (b).- Índice LSH.



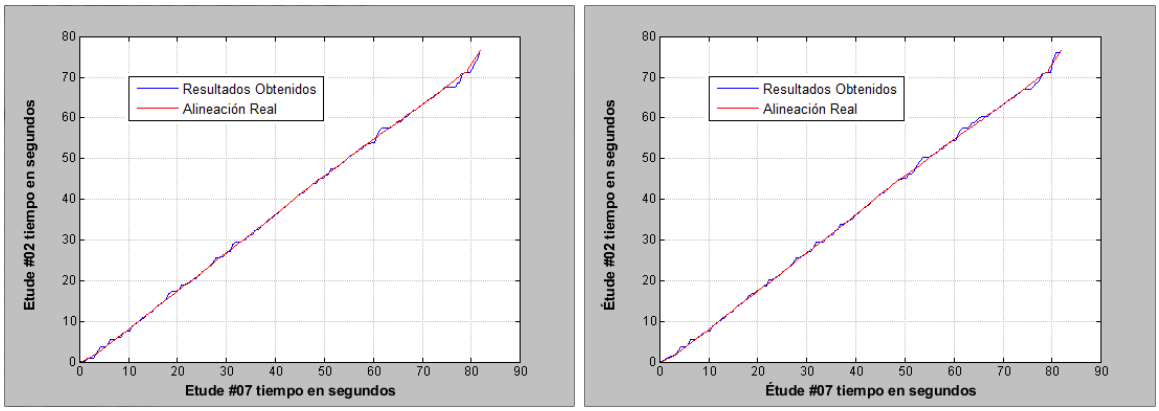
(a) **(b)**

Figura 4.22: Alineación de los Études #1 y #22 utilizando #1 como referencia (a).- Búsqueda Secuencial. (b).- Índice LSH.



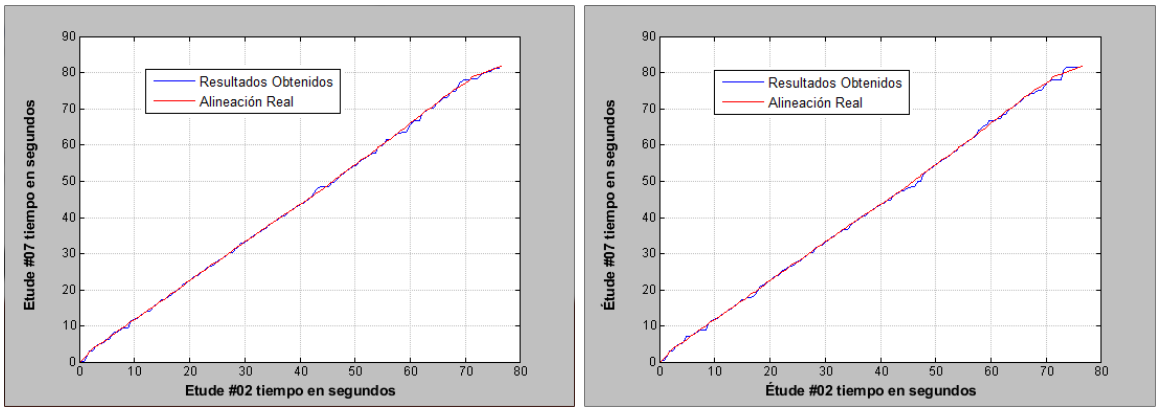
(a) **(b)**

Figura 4.23: Alineación de los Études #1 y #22 utilizando #22 como referencia (a).- Búsqueda Secuencial. (b).- Índice LSH.



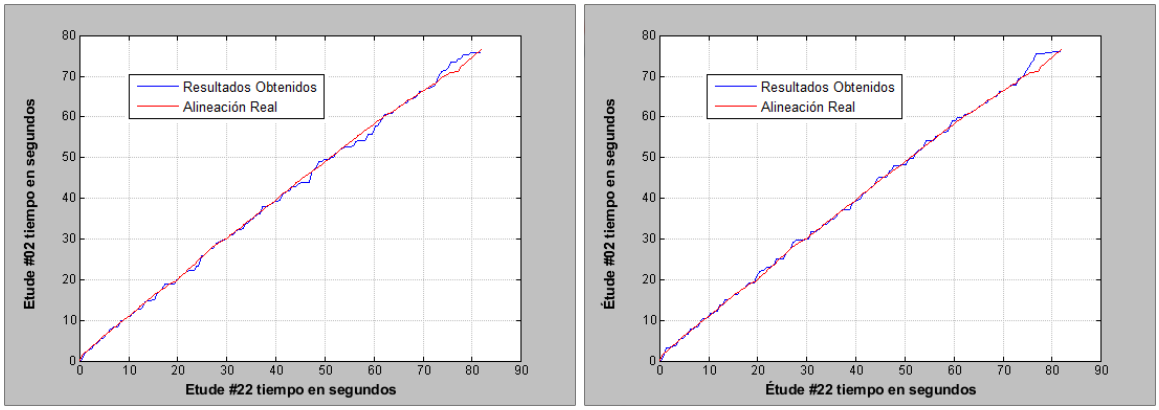
(a) **(b)**

Figura 4.24: Alineación de los Études #2 y #7 utilizando #2 como referencia (a).- Búsqueda Secuencial. (b).- Índice LSH.



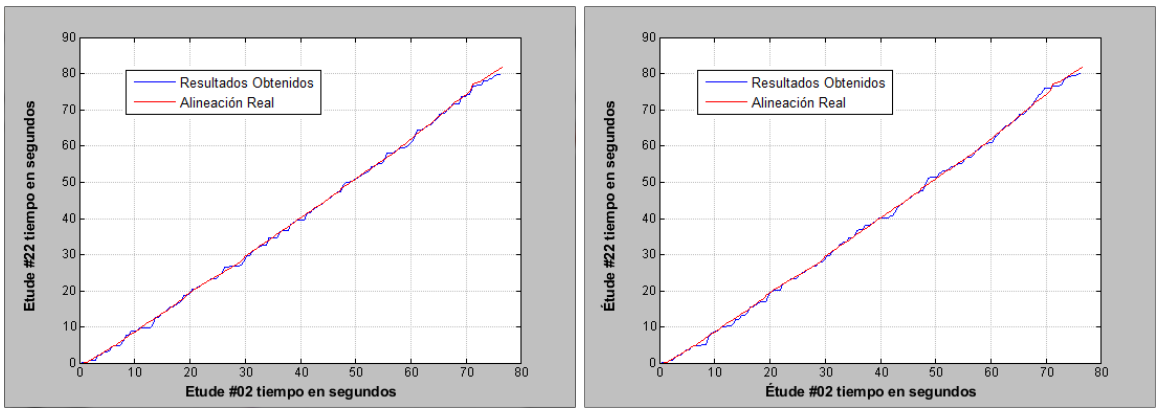
(a) **(b)**

Figura 4.25: Alineación de los Études #2 y #7 utilizando #7 como referencia (a).- Búsqueda Secuencial. (b).- Índice LSH.



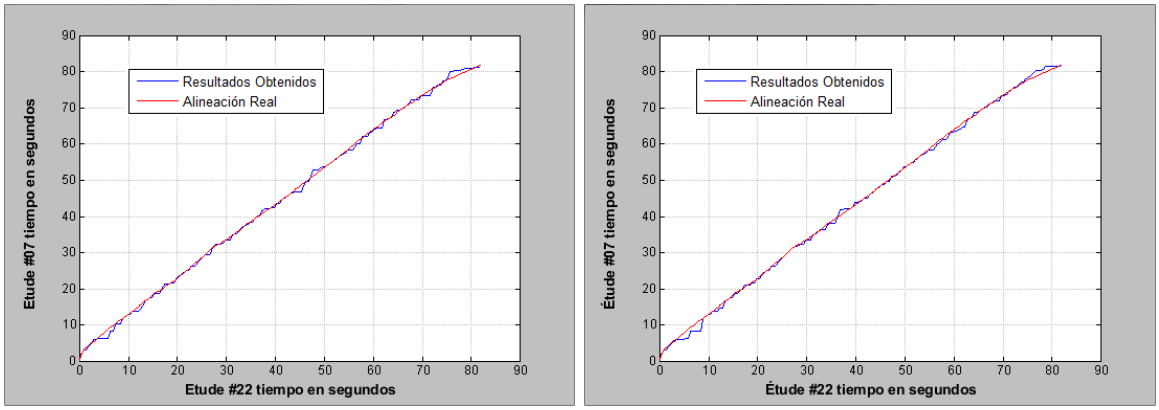
(a) **(b)**

Figura 4.26: Alineación de los Études #2 y #22 utilizando #2 como referencia (a).- Búsqueda Secuencial. (b).- Índice LSH.



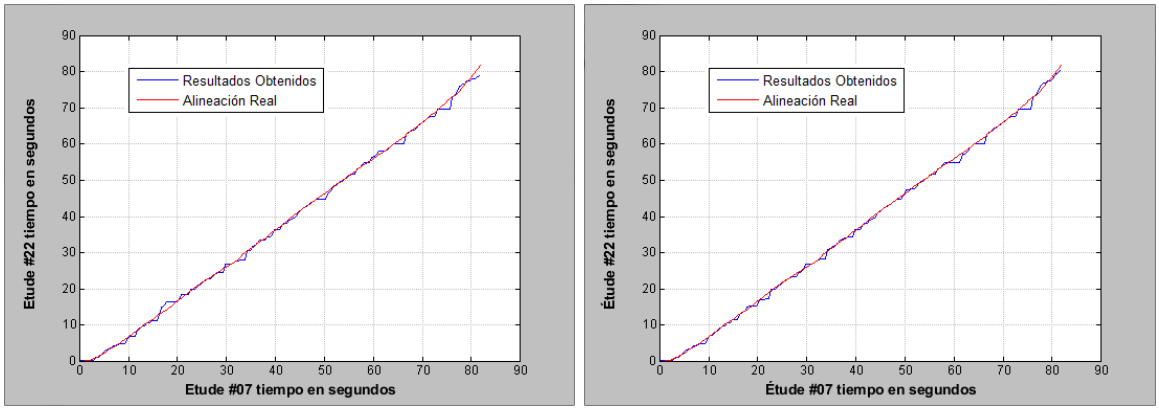
(a) **(b)**

Figura 4.27: Alineación de los Études #2 y #22 utilizando #22 como referencia (a).- Búsqueda Secuencial. (b).- Índice LSH.



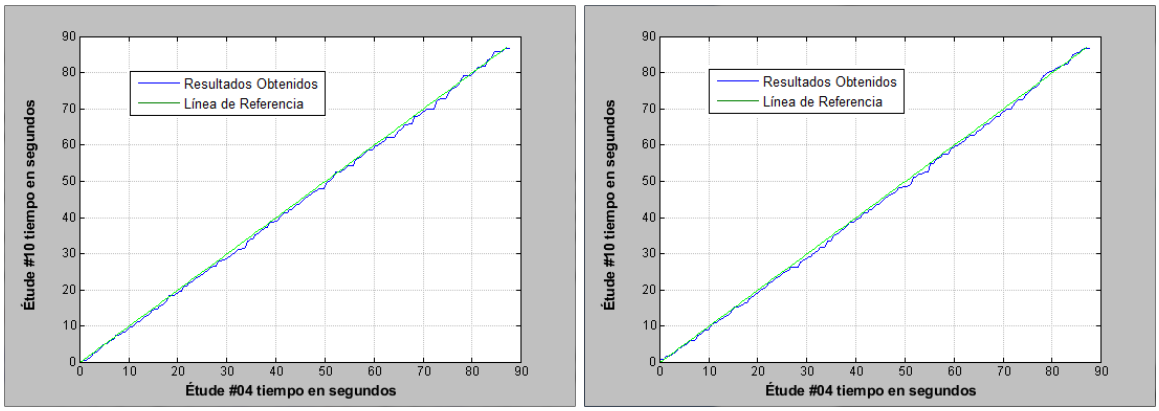
(a) **(b)**

Figura 4.28: Alineación de los Études #7 y #22 utilizando #7 como referencia (a).- Búsqueda Secuencial. (b).- Índice LSH.



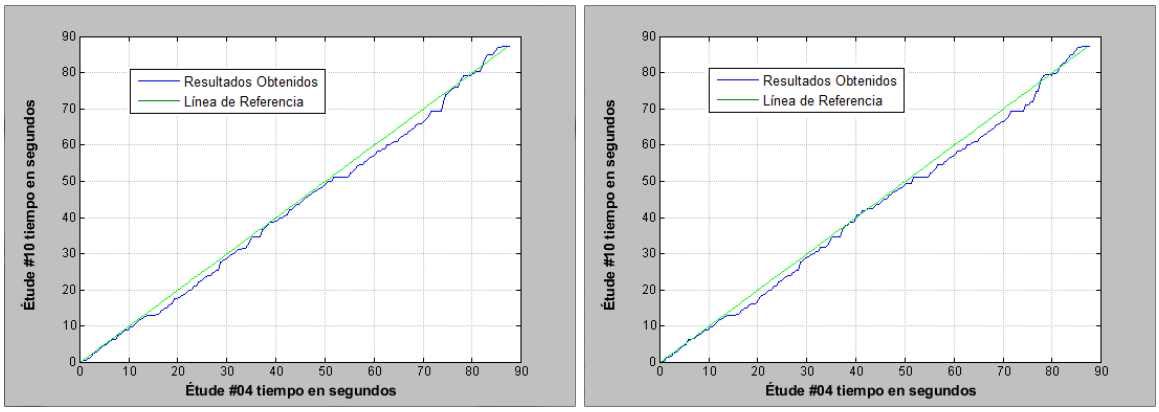
(a) **(b)**

Figura 4.29: Alineación de los Études #7 y #22 utilizando #22 como referencia (a).- Búsqueda Secuencial. (b).- Índice LSH.



(a) **(b)**

Figura 4.30: Alineación de los Études #10 y #4 utilizando #10 como referencia (a).- Búsqueda Secuencial. (b).- Índice LSH.



(a) **(b)**

Figura 4.31: Alineación de los Études #10 y #4 utilizando #10 como referencia y modificando la AFP del Étude #4 artificialmente (a).- Búsqueda Secuencial. (b).- Índice LSH.

4.4. Notas Finales

4.4.1. Notas Resultados LSH

- Todos los ejemplos presentados en la Sección 4.3 fueron calibrados para que los resultados fueran lo más parecido posible a los obtenidos con búsqueda secuencial pero tratando, a su vez, de reducir la cantidad de comparaciones entre subfirmas realizadas con el fin de justificar el uso del índice. El número de comparaciones necesarias puede ser reducido aún más si se sacrifica un poco la similitud entre ellos.

- Como las posiciones de los bits a utilizar por cada instancia se generan aleatoriamente cada que se ejecuta el algoritmo, dos ejecuciones con los mismos parámetros pueden reportar resultados ligeramente distintos.

- La importancia de utilizar el índice aumenta considerablemente conforme más largas son las interpretaciones. En los ejemplos presentados, se observó una reducción de entre 60 % y 85 % en la cantidad de comparaciones entre subfirmas efectuadas, aunque este porcentaje puede ser mayor para interpretaciones con una duración mayor.

- Una reducción del 60 % podría parecer poco para justificar la creación del índice, no obstante, como se mencionó anteriormente, se utilizan las distancias de Levenshtein y LCS, las cuales son costosas de calcular. Por lo tanto, una reducción de incluso el 60 % en el número de comparaciones entre subfirmas conlleva un efecto dramático en tiempo de ejecución, lo cual podría permitir que el algoritmo pueda ser utilizado en tiempo real.

- La cantidad de mapas hash utilizada en los experimentos mostrados en la Sección 4.3 podría parecer excesiva, especialmente considerando los requerimientos de memoria. Sin embargo no es una cuestión relevante en nuestro problema por las siguientes razones:

  – En el índice LSH creado, únicamente se indexarán los vectores de características de una de las interpretaciones a la vez y las interpretaciones utilizadas en los experimentos tienen de entre 6000 y 13000 vectores de características únicamente.

- El número de bits utilizado para crear el hash en los experimentos fue grande (13, 14 y hasta 15 bits de los 17 disponibles), lo cual parece implicar que los mapas serán bastante grandes con 2^{13} (8192), 2^{14} (16384) o 2^{15} (32768) posibles cubetas. No obstante, incluso si se tomara una interpretación cuya AFP contenga 13000 vectores, solamente si todos los hash creados fueran diferentes se crearían 13000 cubos, cada uno con un elemento. Ese jamás es el caso. Para la interpretación más larga utilizada, la Ballade #14, con una duración de 2 minutos y 31 segundos y cuya AFP tiene 13050 líneas; utilizando 14 bits los mapas tienen 8300 cubetas en promedio. Para un Étude como el #4 con una AFP de 7548 vectores, el número de cubetas es de 4500 únicamente.
- Utilizar un número alto de bits para crear el hash ocasiona además que los cubos tengan pocos elementos. Lo anterior tiene como consecuencia que se revisen pocos candidatos por mapa.
- De esta manera el número de bits utilizado, en conjunto con la cantidad de mapas, permite obtener los resultados reportados.
- La opción opuesta sería utilizar pocos bits para la creación de los hashes. De esa manera los cubos estarían más poblados y se utilizarían menos mapas para contrarrestar el número de comparaciones. Esta opción tiene un desempeño menos deseable, debido a que la selección de las bandas que se utilizarán para crear el hash se hace al azar y ésto podría provocar que bandas con información importante no se tomen en cuenta.

4.4.2. Notas Adicionales

Como se mencionó en la Sección 3.2.4, el crear variaciones de uno, dos o tres bits de cada vector de la subfirma, tiene como propósito revisar cubetas adicionales a las que se habrían revisado utilizando únicamente el vector original. Lo anterior con el fin de que la obtención de candidatos sea lo más robusta posible. Sin embargo, ésto tiene consecuencias importantes en el funcionamiento del seguidor. Utilizar variaciones de tres bits puede generar una cantidad de vectores muy grande: para los vectores de diecisiete bits utilizados, si se

crean variaciones de un bit se generan diecisiete (17) vectores adicionales; con variaciones de dos bits se generan ciento cincuenta y tres (153) vectores adicionales y con variaciones de tres bits se generan ochocientos treinta y tres (833) vectores adicionales. Lo anterior puede afectar de manera significativa; recuerde que por cada vector se generará un hash por cada instancia y se obtendrán las posiciones candidato almacenadas en la cubeta correspondiente al hash obtenido. Estas posiciones candidato serán revisadas una a una obteniendo su subfirma correspondiente y comparando la subfirma obtenida con la subfirma de consulta, mediante alguna función de distancia. Este aumento en el número de comparaciones entre subfirmas realizadas, si bien incrementa la calidad del seguimiento, también aumenta el tiempo de ejecución del algoritmo y restringe su uso para aplicaciones en tiempo real.

Se intentó acelerar el algoritmo, limitando la revisión de posibles candidatos a aquellos que fueran mayores o iguales a la posición previa. Para la búsqueda secuencial se comenzaba la comparación entre matrices a partir de la posición indicada como anterior y para el índice LSH únicamente se revisaban candidatos superiores a la posición anterior. Este enfoque tuvo resultados contraproducentes, puesto que las listas de vecinos más cercanos se llenaban con candidatos que no necesariamente tenían las matrices más parecidas, ya que los verdaderos vecinos más cercanos habían sido descartados por encontrarse por detrás de la posición anterior. Por ejemplo, si en determinado momento el algoritmo no encuentra un candidato dentro de la lista de vecinos más cercanos que cumpla con las restricciones, deberá optar por mantenerse en la misma posición. Si la lista de vecinos más cercanos está llena de candidatos que se encuentran por detrás de la posición anterior, ninguno de ellos cumple las restricciones y el algoritmo tomará dicha decisión. Si se evita revisar candidatos por detrás de la posición anterior, el caso presentado jamás ocurrirá. En su lugar, la lista se llenará de otros candidatos que no deberían entrar a la lista. Ésto provocaría que el algoritmo avanzara, cuando en realidad debería mantenerse.

4.5. Resumen de Capítulo

En este capítulo se presentó el esquema de validación diseñado para evaluar el desempeño del seguidor. La validación se basó en la utilización de AFPs modificadas arti-

ficialmente en puntos específicos, de manera que se puede comprobar si el seguimiento se comporta de la manera esperada. Posteriormente, se presentaron los resultados obtenidos por el seguidor utilizando una búsqueda secuencial para la obtención de los posibles candidatos. Se obtuvieron manualmente mediciones sobre los tiempos de ejecución de las notas de algunas interpretaciones y, con estas mediciones, se crearon gráficas de alineación real entre ellas. Se observó que al utilizar búsqueda secuencial el seguidor tiene un comportamiento adecuado en los diferentes ejemplos presentados, puesto que la alineación obtenida siempre resultó muy similar a la alineación real. Por último, se expusieron los resultados obtenidos al utilizar el índice LSH en lugar de la búsqueda secuencial. Se corroboró que mediante la utilización del índice, se reduce significativamente (60 %-85 % véase la Tabla 4.3) la cantidad de comparaciones entre subfirmas realizadas, manteniendo un seguimiento equivalente al obtenido con la búsqueda secuencial.

Habiendo presentado los resultados, el enfoque del siguiente capítulo será el discutir las conclusiones generales y particulares a las que se arribó y los posibles trabajos futuros que pueden surgir del presente proyecto.

Capítulo 5

Conclusiones

5.1. Conclusiones Generales

El seguimiento de audio o alineación de interpretaciones es un problema que ha sido abordado de muchas maneras diferentes. La aportación del presente trabajo de Tesis consiste en presentar un esquema novedoso que no había sido considerado previamente. El esquema propuesto utiliza la huella de audio, basada en la entropía presentada en [Camarena y Chávez, 2006] y el índice de búsqueda aproximada, basado en Hashing Sensible a la Localidad (LSH), presentado originalmente en [Gionis *et al.*, 1999].

Se trabajó con grabaciones de veintidós diferentes interpretaciones en piano de dos piezas de Frédéric Chopin, las cuales han sido utilizadas en trabajos previos de índole similar al presente, como [Dixon, 2005] y Kirchhoff y Lerch [2011].

Para realizar la alineación de dos interpretaciones, se obtuvo la firma de cada una y se utilizó una de ellas como referencia, para buscar la posición correspondiente de una secuencia de subfirmas equivalentes a 0.5 segundos de audio, de la otra interpretación.

Se validó el funcionamiento del algoritmo de seguimiento con un método de validación novedoso, similar al utilizado en [Hu *et al.*, 2003b]. Los experimentos de validación fueron realizados utilizando firmas de audio que fueron alteradas artificialmente en puntos específicos, de manera que se tuviera una alineación esperada que pudiera ser comparada con la alineación obtenida.

Una vez que se validó el desempeño del seguidor, se prosiguió a efectuar experimentos de alineación entre dos interpretaciones diferentes. Estos experimentos se realizaron de dos maneras distintas: primero utilizando una búsqueda secuencial para encontrar las posiciones candidato y, posteriormente, utilizando el índice LSH. Al utilizar búsqueda secuencial se obtuvieron resultados satisfactorios de alineación, puesto que éstos coinciden con las características y particularidades de las interpretaciones comparadas, así como con las alineaciones reales entre dichas interpretaciones que fueron obtenidas manualmente. Con la utilización del índice LSH, se logró mantener un resultado de alineación equivalente al obtenido con búsqueda secuencial reduciendo además, de manera considerable, la cantidad de comparaciones entre subfirmas efectuadas.

Por las razones anteriores, se concluye que se cumplió con el objetivo planteado en el presente trabajo, puesto que se implementó exitosamente el *Index Audio Following* (IAF), un esquema capaz de realizar la alineación entre dos interpretaciones distintas de la misma pieza musical, mediante el uso de un índice de proximidad.

5.2. Conclusiones Específicas

1. Es evidente, a partir de los resultados de los ejemplos presentados durante la sección 4.2, que el algoritmo de seguimiento implementado con búsqueda secuencial muestra un comportamiento adecuado ante las características de las grabaciones utilizadas. Adicionalmente, comparado con las alineaciones reales entre las interpretaciones, el seguidor presenta un muy buen desempeño.

2. Con la implementación y uso del índice LSH, se logró una gran disminución en la cantidad de cálculos de comparación de subfirmas respecto a la búsqueda secuencial. En los experimentos presentados se tienen reducciones de entre el 60 % y el 85 % y este porcentaje puede ser mayor en interpretaciones con mayor duración. Esta disminución en el número de cálculos de comparación de subfirmas, tiene un impacto muy significativo en el tiempo de ejecución del algoritmo.

3. La implementación de AF expuesta, puede ser utilizada en aplicaciones que requieran

de la alineación de interpretaciones musicales, como podría ser un sistema de partitura automática para los intérpretes. No obstante, es todavía necesario realizar más experimentos, para determinar la exactitud y viabilidad de la presente implementación en escenarios reales.

5.3. Trabajos Futuros

Como se mencionó en la sección anterior, aún existen diversas áreas de experimentación y expansión del trabajo que se presenta. A continuación, se enumeran algunos de los posibles trabajos a futuro:

1. Al ser el presente trabajo uno de los primeros en adoptar el esquema de búsqueda utilizando índices para realizar seguimiento de audio, las pruebas se realizaron con grabaciones en las que únicamente existe un instrumento (piano) y que fueron efectuadas en un ambiente muy controlado, sin ruidos y otros tipos de degradaciones. Trabajos posteriores podrían incorporar el uso de interpretaciones que incluyan:

   - Instrumentos diferentes al piano.
   - Varios instrumentos a la vez y/o voz.
   - Grabaciones con degradaciones como ruido, ecualizaciones, cambios en el volumen, etc.
   - Música en vivo.

2. En este trabajo se hizo uso de una huella de audio basada en la entropía y un índice basado en LSH. Otros esquemas podrían optar por utilizar otras huellas de audio que muestren mayor consistencia cuando se trate con interpretaciones diferentes. Adicionalmente, se utilizó el índice LSH, debido a que la firma utilizada realiza una codificación de los vectores de características a ceros y unos. Si se cambia la AFP utilizada, probablemente sea conveniente utilizar un índice diferente.

3. Una característica muy evidente del índice LSH es que es altamente paralelizable. Una extensión al trabajo presentado, consistiría en paralelizar el proceso de búsqueda en

el índice, de manera que cada una de las instancias creadas se procese en paralelo. Ésto disminuiría el tiempo de ejecución del algoritmo en general, mejorando así su capacidad de desempeñarse en tiempo real.

4. Implementar una interfaz para pruebas en tiempo real, que obtenga el audio del ambiente y utilice el algoritmo presentado para realizar la alineación.

5. La implementación de AF presentada en este trabajo se concibió para que pudiera ser utilizada en aplicaciones de tiempo real. Sin embargo, si la ejecución en tiempo real no es un objetivo, se podría utilizar una ligera variación de este trabajo para realizar la alineación de interpretaciones de manera offline. Durante algunos experimentos realizados se comprobó que, modificando los parámetros de la AFP para la extracción de características, de manera que ésta tenga mayor definición, los resultados de alineación mejoraban considerablemente. Utilizando marcos con un traslape de aproximadamente 3 ms, se obtiene una firma de audio lo suficientemente definida como para realizar una alineación muy precisa. Desafortunadamente, utilizar un traslape de 3 ms, simplemente no es viable en un contexto de tiempo real.

Apéndice A

Tabla de Interacción de Parámetros

En este apéndice se presenta una tabla de interacción entre los parámetros descritos en la Sección 3.5. En ella se incluye un resumen del efecto de aumentar y disminuir estos parámetros, así como la manera de contrarrestar dichos efectos.

Parámetro	Acciones		Compensación
k	+	Aumenta la posibilidad de encontrar un candidato que cumpla las condiciones. Se pueden utilizar posiciones cuyas matrices no se parecen a la buscada	Disminuir $windowMultiplier$
	−	Disminuye la posibilidad de encontrar un candidato adecuado. Las posiciones candidato tienen matrices muy parecidas a la buscada	Aumentar $windowMultiplier$
nB	+	Aumenta el tiempo de comparación entre matrices. Si se utilizan bandas sin contenido útil se puede afectar el resultado de la comparación.	Utilizar una distancia que no tome en cuenta inserciones y borrados como Hamming.
	−	Disminuye el tiempo de comparación. Si se utilizan menos bandas de las necesarias no se toma en cuenta información importante	Usar una distancia con inserciones y borrados como Levenshtein.
$windowSize$	+	Disminuye la "fineza" del seguimiento. Aumenta el tiempo disponible para realizar el procesamiento	
	−	Aumenta la "fineza" del seguimiento. Disminuye el tiempo disponible para realizar el procesamiento	
$windowMultiplier$	+	Permite "saltos" más grandes. Poco probable que el algoritmo decida "mantenerse".	Disminuir k. Utilizar una política de elección que tome en cuenta la cercanía en tiempo.
	−	Permite "saltos" pequeños. Si existen secciones faltantes el seguimiento podría ser inadecuado.	Aumentar k para aumentar las posibilidades de avanzar.
$howManyMaps$	+	Aumenta la cantidad de *buckets* obtenidos por vector.	Aumentar hmb y/o disminuir bV.
	−	Disminuye la cantidad de *buckets* obtenidos por vector	Aumentar bV y/o disminuir hmb.
$howManyBits$	+	Disminuye la "densidad" de los *buckets* y ∴ la cantidad de posiciones a revisar.	Aumentar $howManyMaps$ y/o $bitVariations$.
	−	Aumenta la "densidad" de los *buckets* y ∴ la cantidad de posiciones a revisar.	Disminuir $howManyMaps$ y/o $bitVariations$.
$bitVariations$	+	Aumenta las variaciones creadas por cada vector y ∴ la cantidad de *buckets* obtenidos.	Aumentar hmb y/o disminuir hmm.
	−	Disminuye las variaciones creadas por cada vector y ∴ la cantidad de *buckets* obtenidos.	Aumentar hmm y/o disminuir hmb.

Tabla A.1: Interacción entre parámetros.

Apéndice B

Figuras: Resultados de Alineación

En este apéndice se incluyen todas las figuras de los experimentos presentados durante el Capítulo 4, en su tamaño original. Lo anterior, debido a que en el Capítulo 4 las figuras fueron reducidas de tamaño por cuestiones de edición. Consultar dichas figuras en su tamaño original, puede ayudar al lector a realizar una mejor interpretación de los resultados presentados.

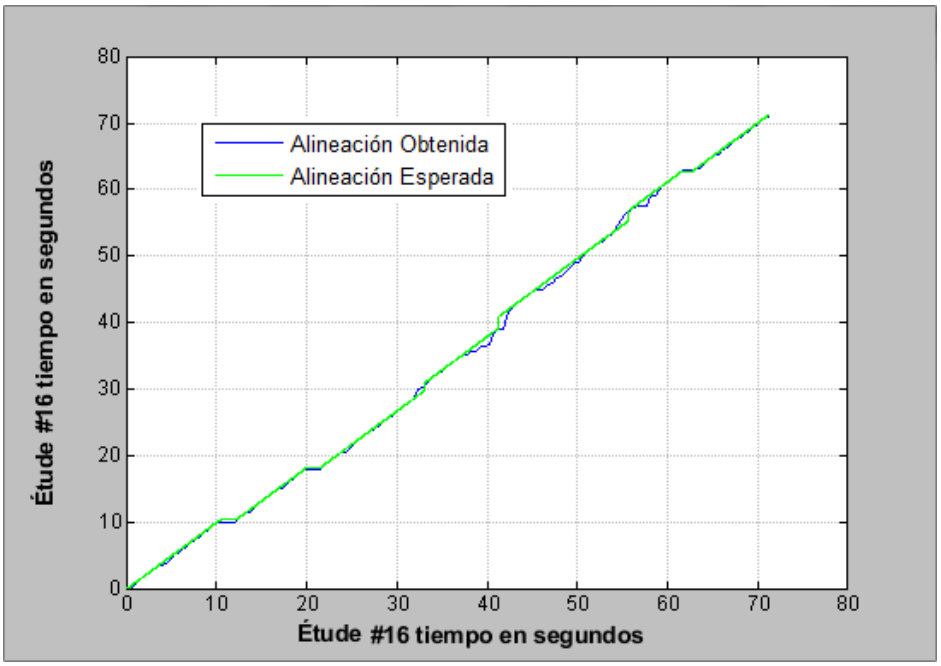


Figura B.1: Comportamiento del seguimiento con el Modo 1.

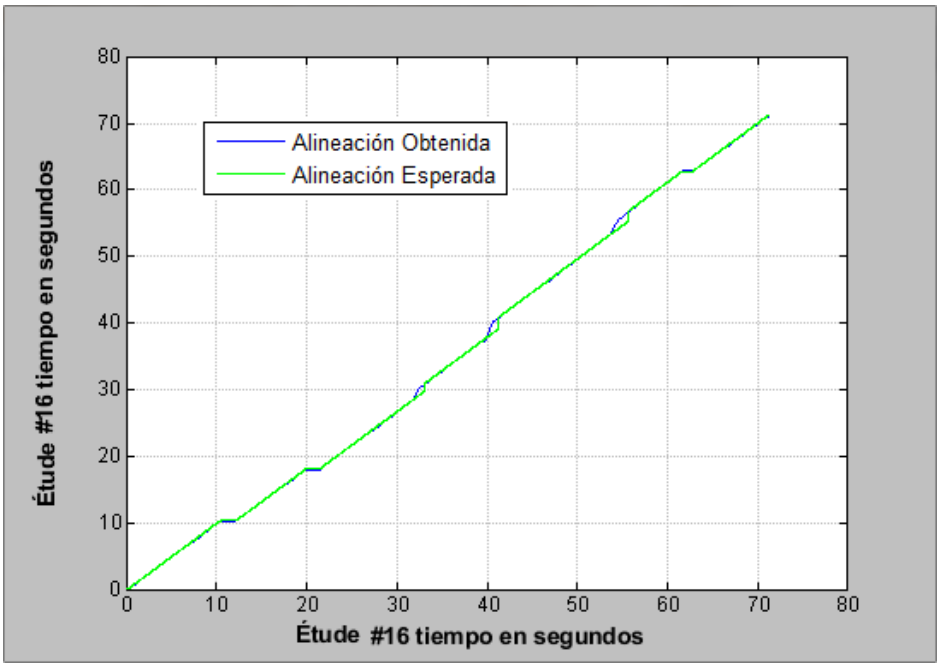


Figura B.2: Comportamiento del seguimiento con el Modo 2.

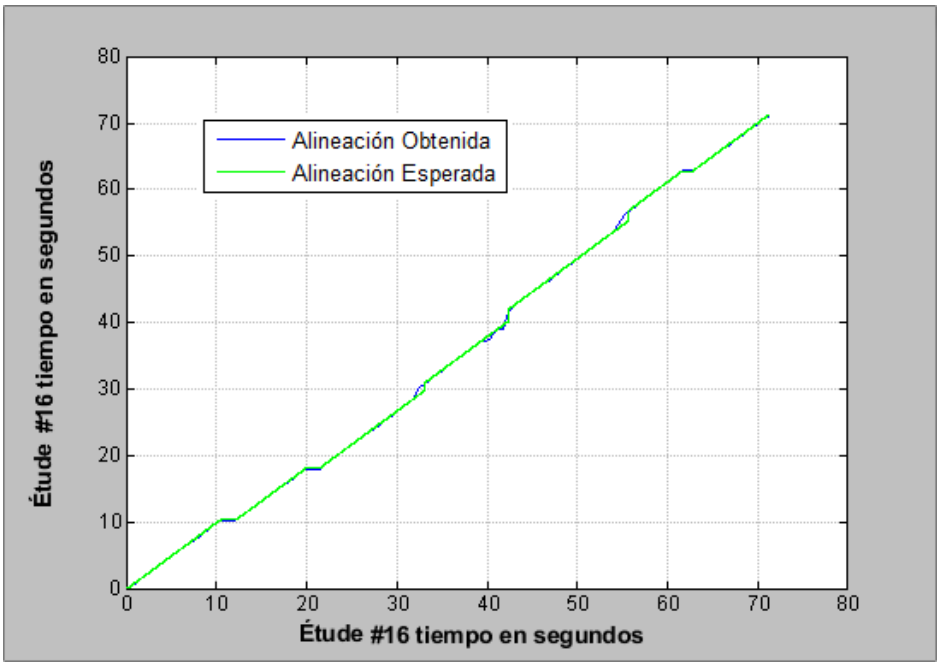


Figura B.3: Comportamiento del seguimiento con el Modo 3.

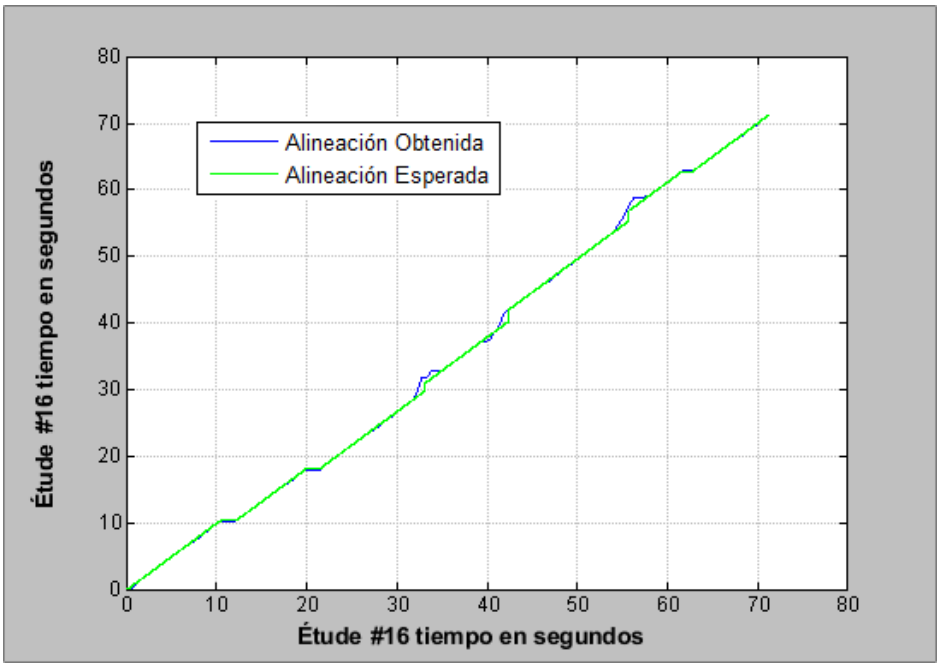


Figura B.4: Comportamiento del seguimiento con el Modo 4.

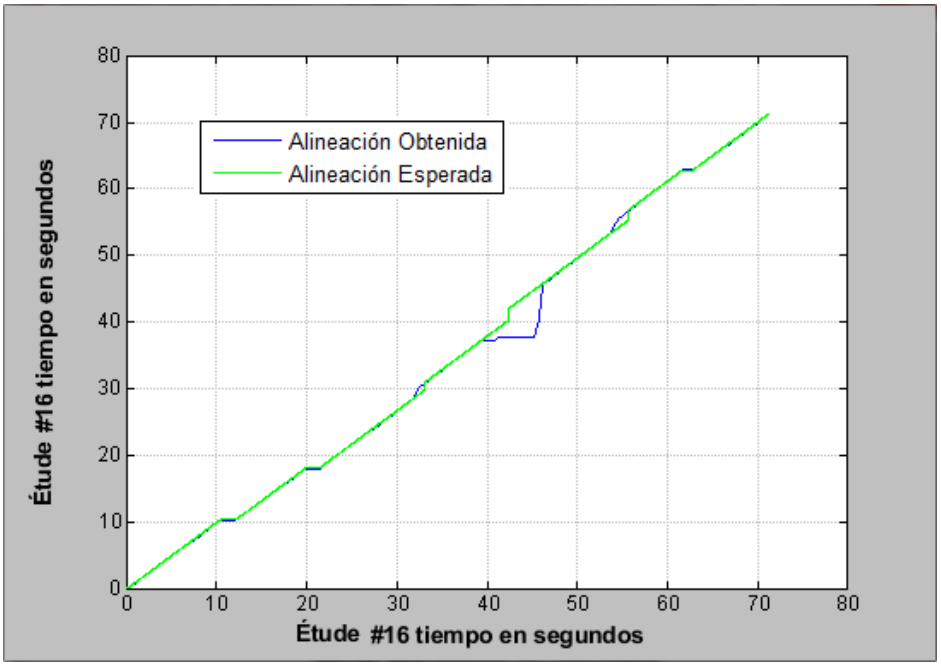


Figura B.5: Comportamiento del seguimiento la distancia de Hamming.

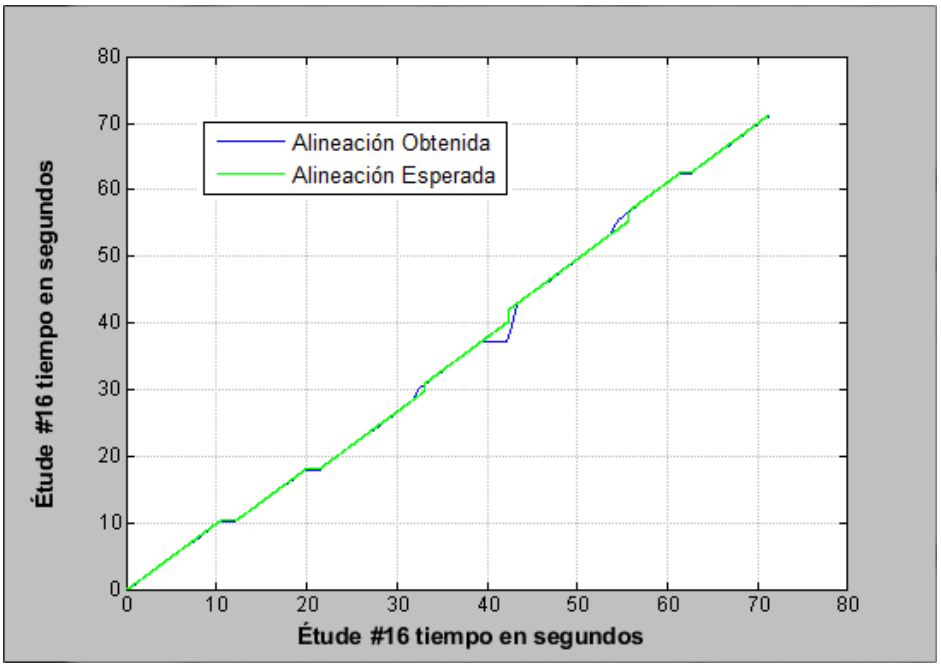


Figura B.6: Comportamiento del seguimiento la distancia LCS.

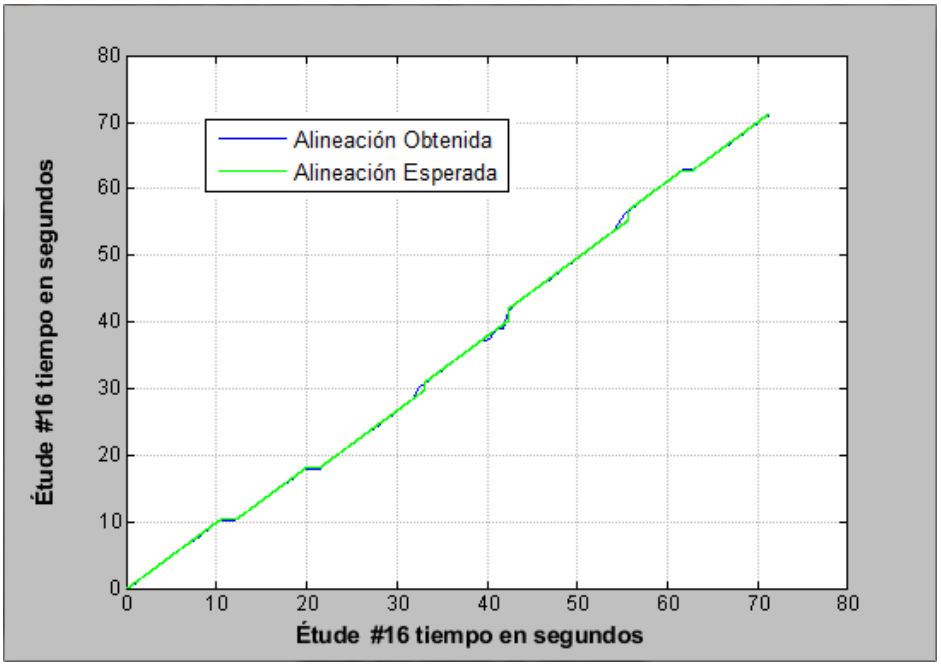


Figura B.7: Comportamiento del seguimiento la distancia de Levenshtein.

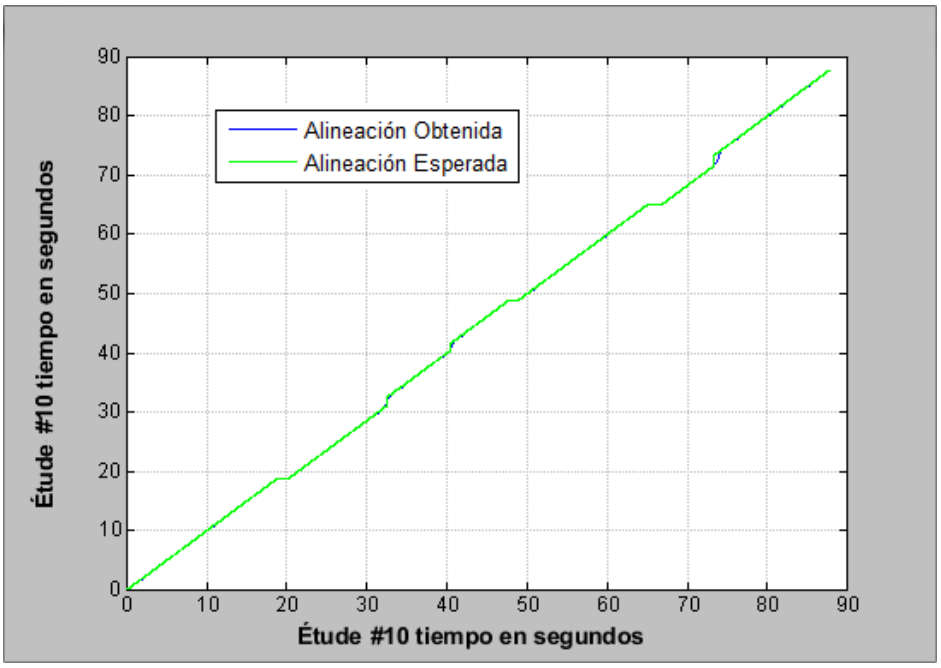


Figura B.8: Comportamiento del seguimiento con un valor k adecuado.

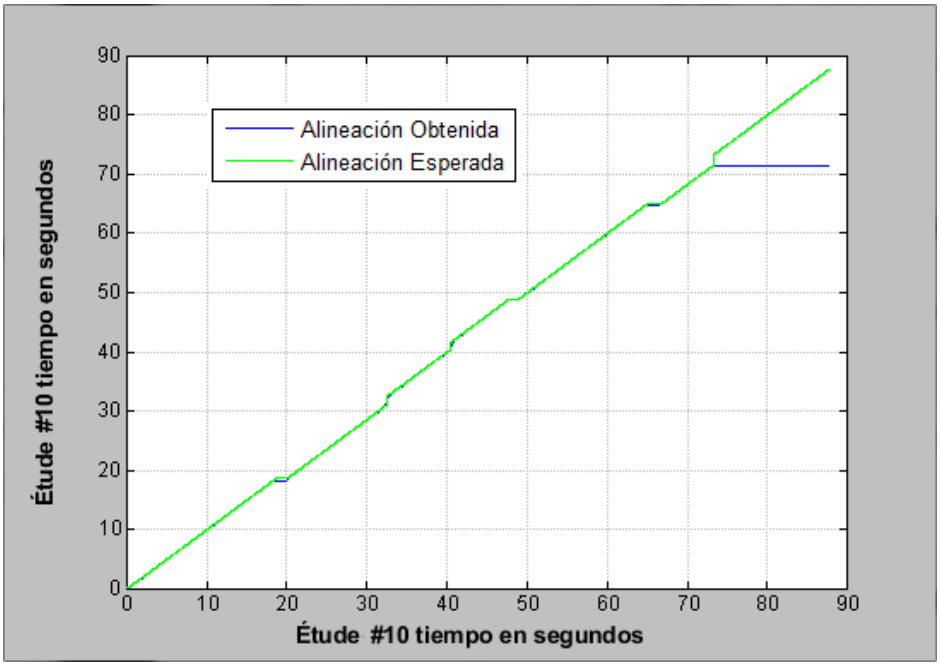


Figura B.9: Comportamiento del seguimiento con un valor k bajo.

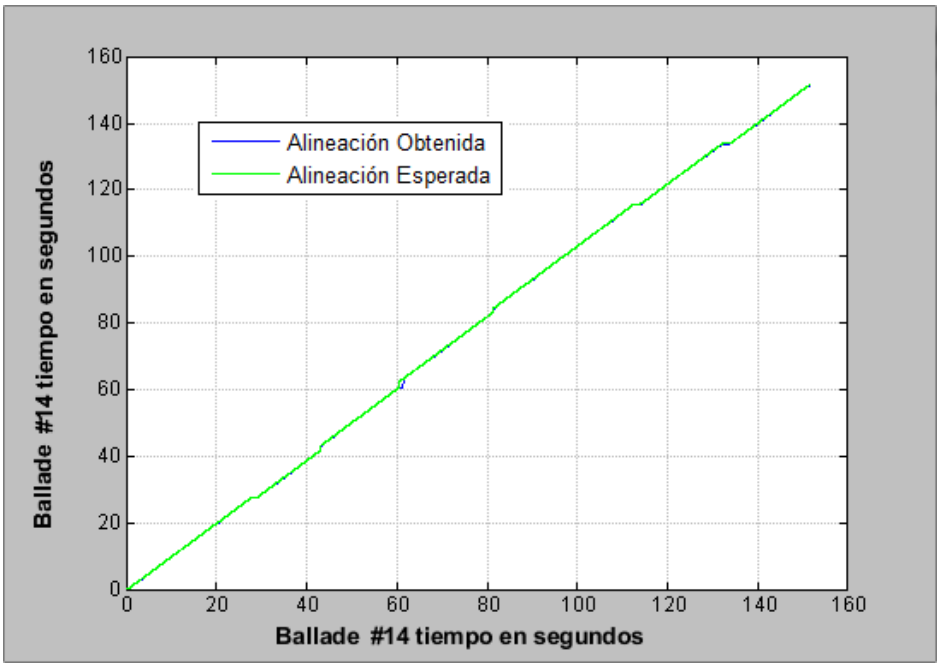


Figura B.10: Comportamiento del seguimiento con un valor k adecuado.

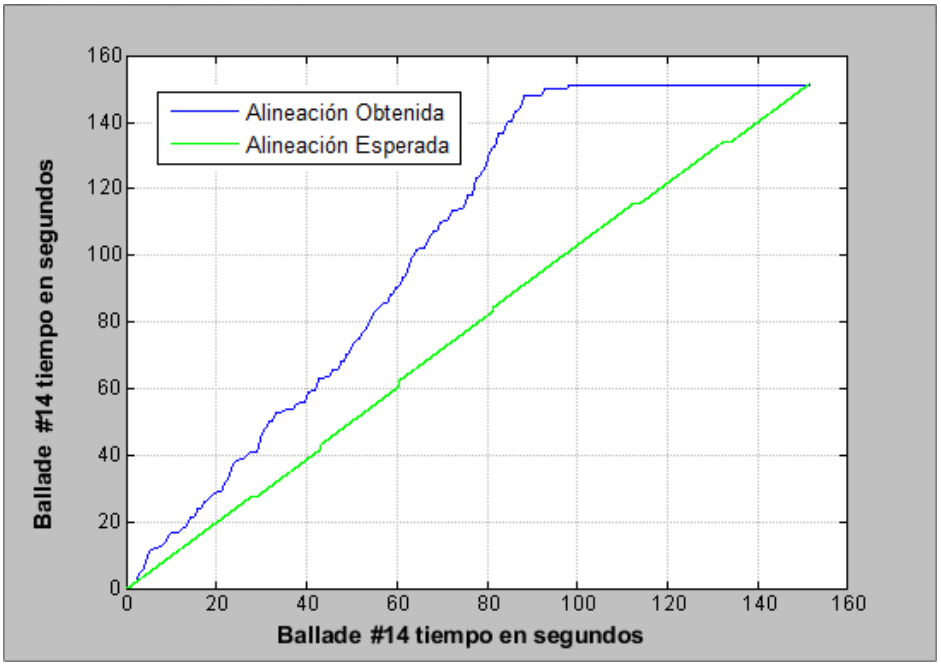


Figura B.11: Comportamiento del seguimiento con un valor k alto.

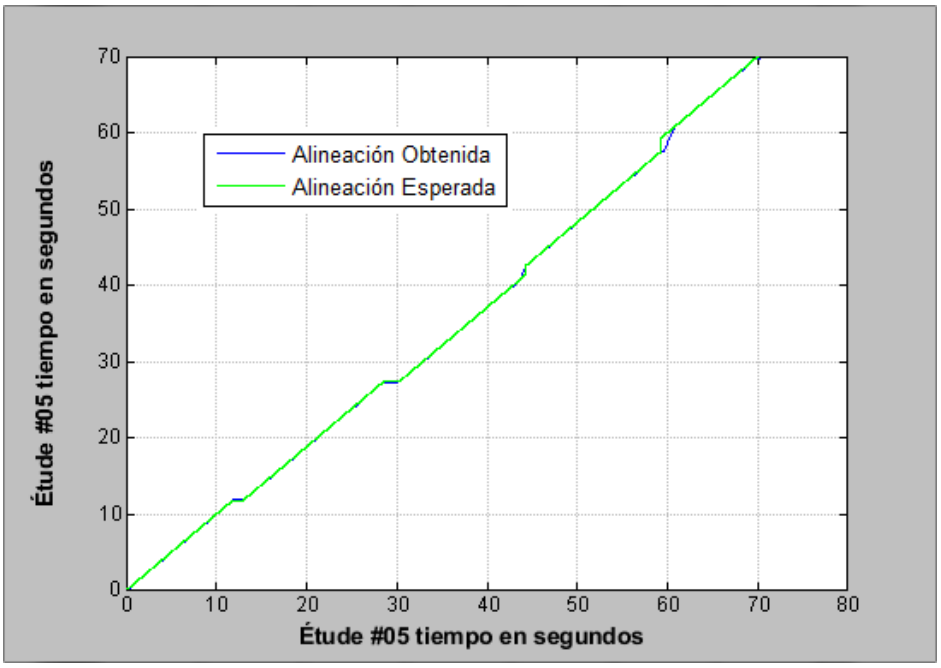


Figura B.12: Comportamiento del seguimiento con un valor *wmd* adecuado.

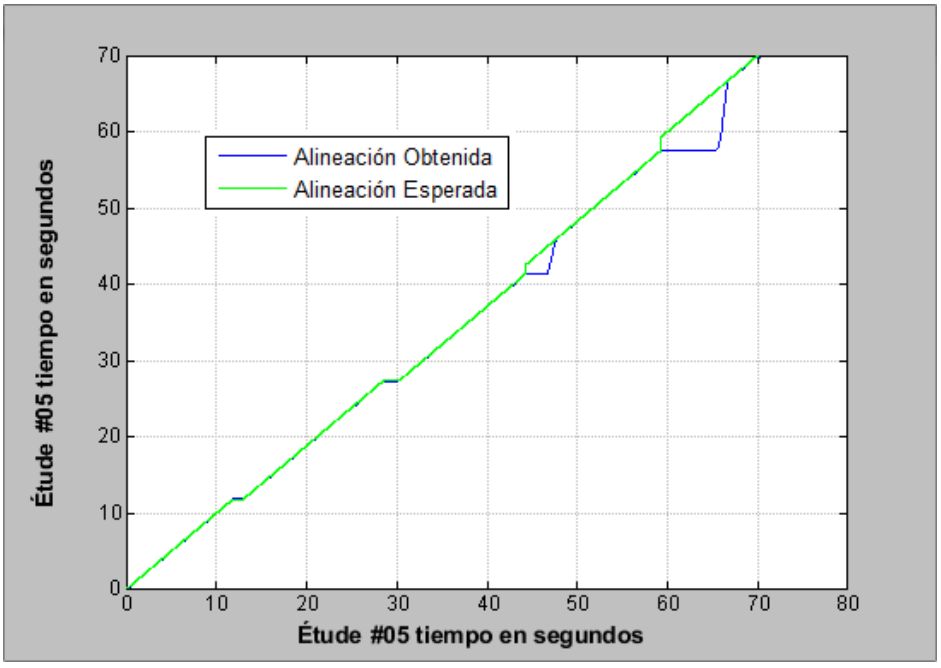


Figura B.13: Comportamiento del seguimiento con un valor *wmd* bajo.

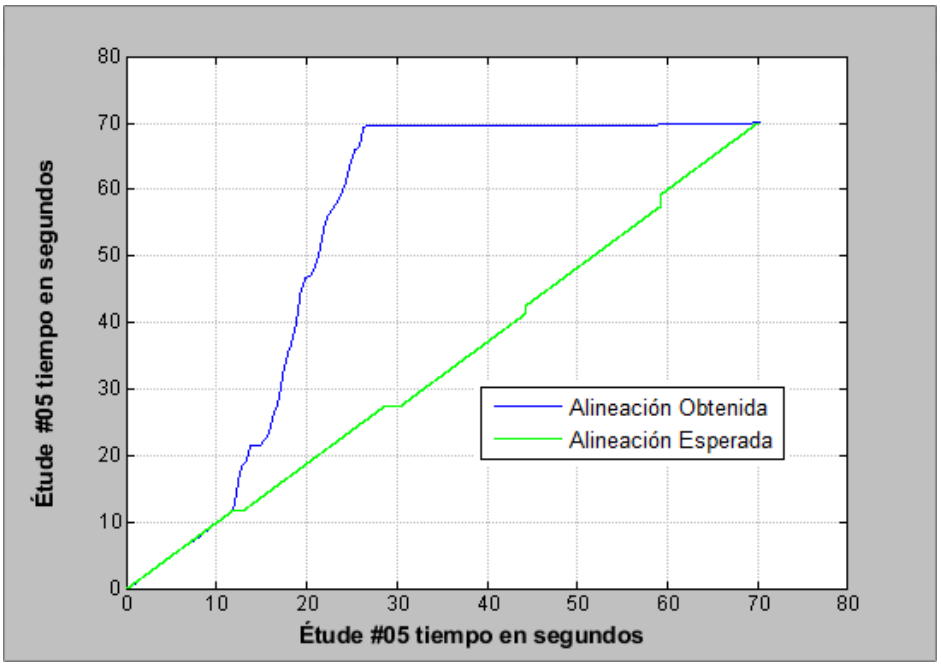


Figura B.14: Comportamiento del seguimiento con un valor *wmd* alto.

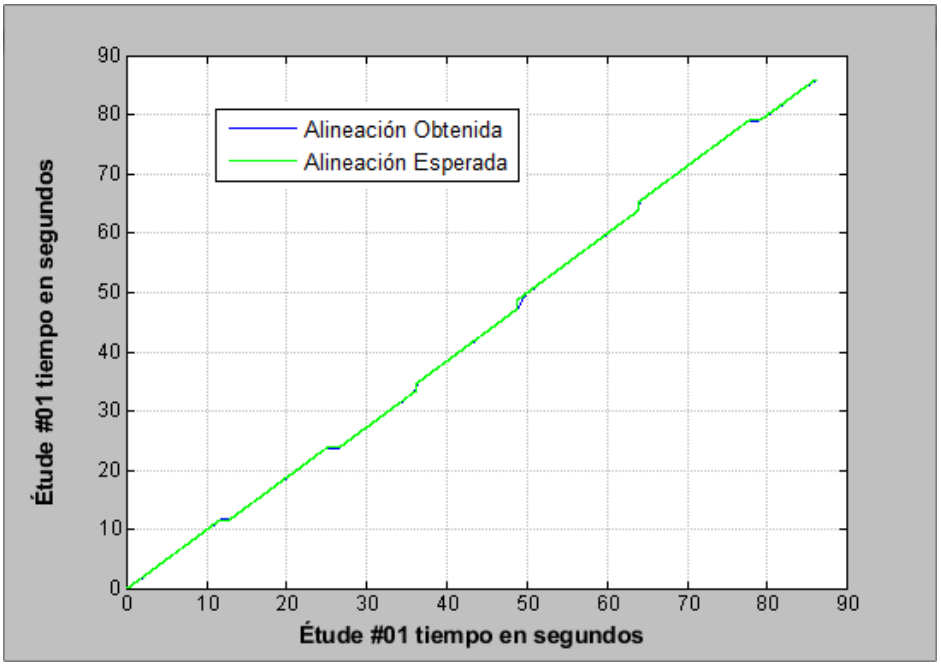


Figura B.15: Alineación sin ruido.

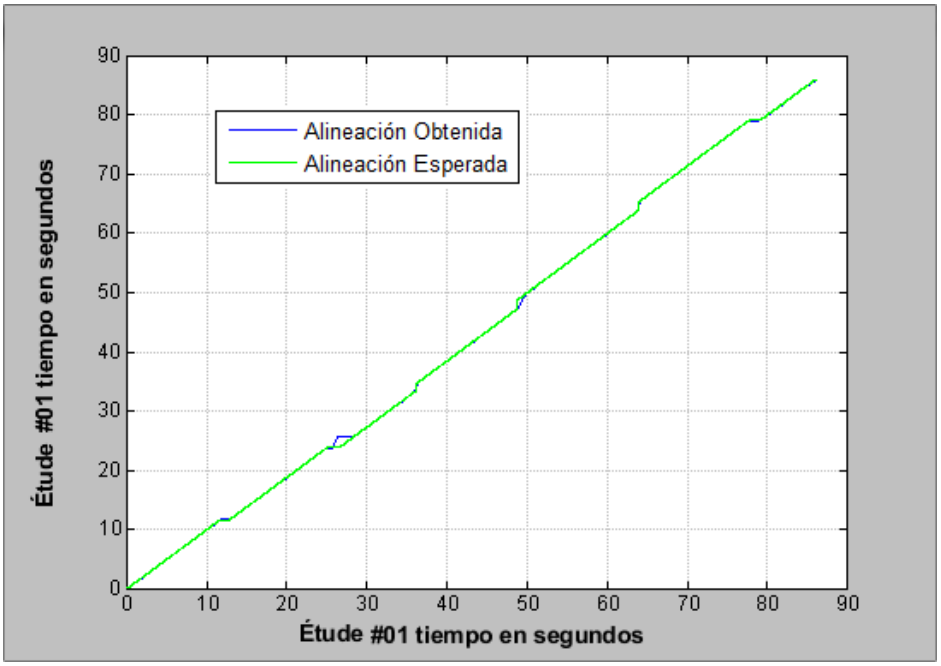


Figura B.16: Comportamiento del seguimiento con 10 % de ruido.

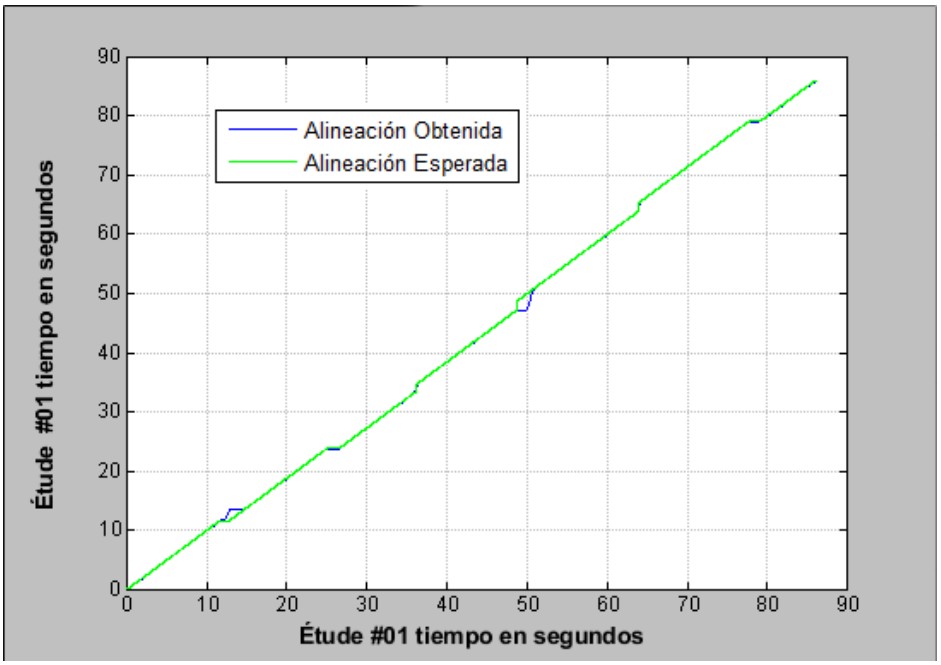


Figura B.17: Comportamiento del seguimiento con 20 % de ruido.

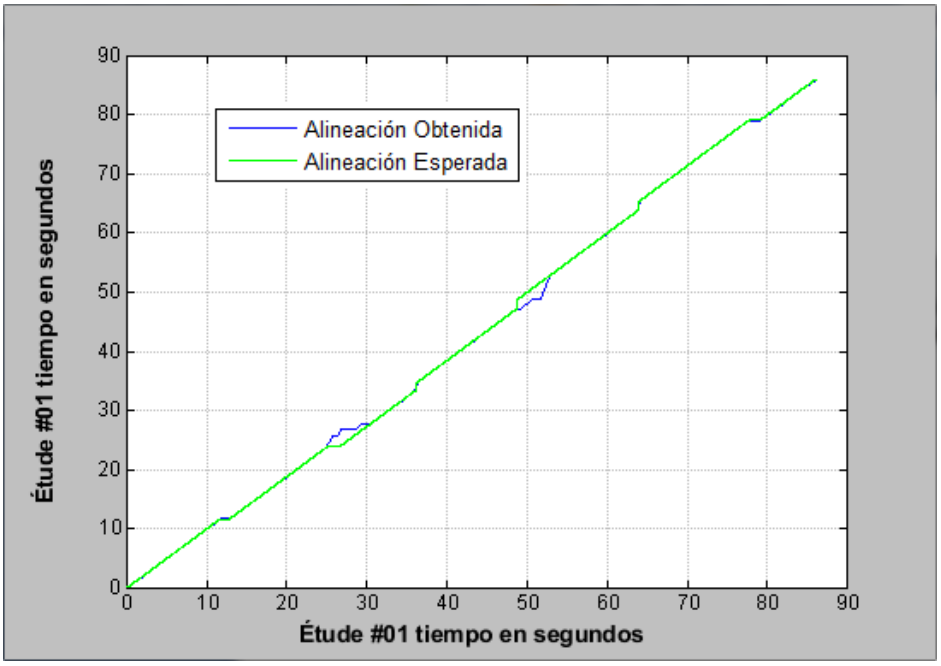


Figura B.18: Comportamiento del seguimiento con 25 % de ruido.

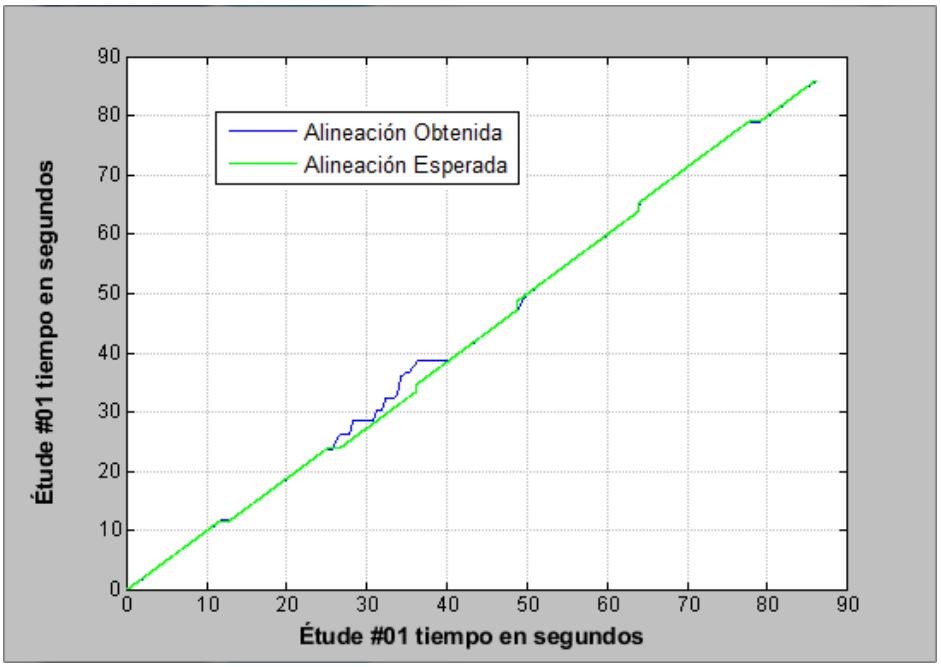


Figura B.19: Comportamiento del seguimiento con 30 % de ruido.

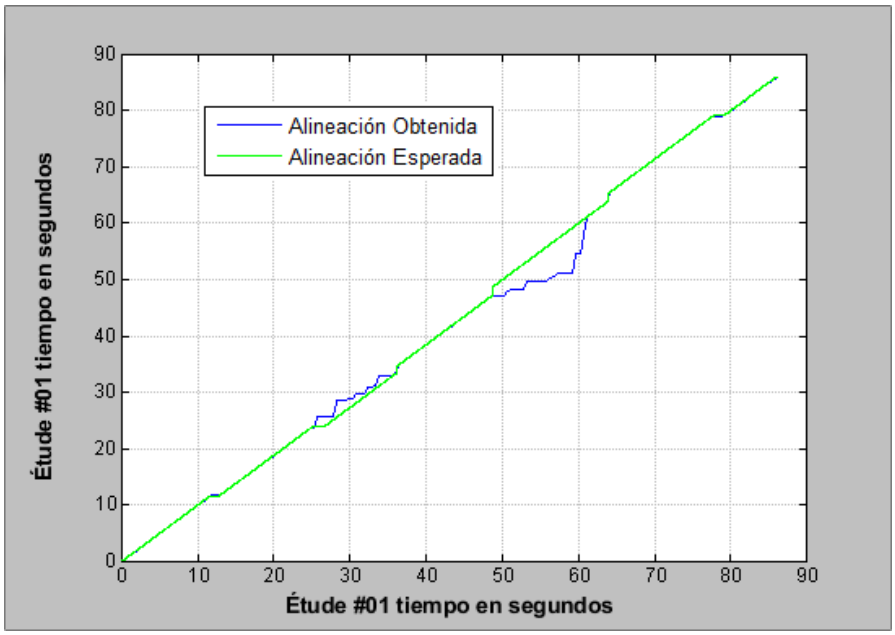


Figura B.20: Comportamiento del seguimiento con 35 % de ruido.

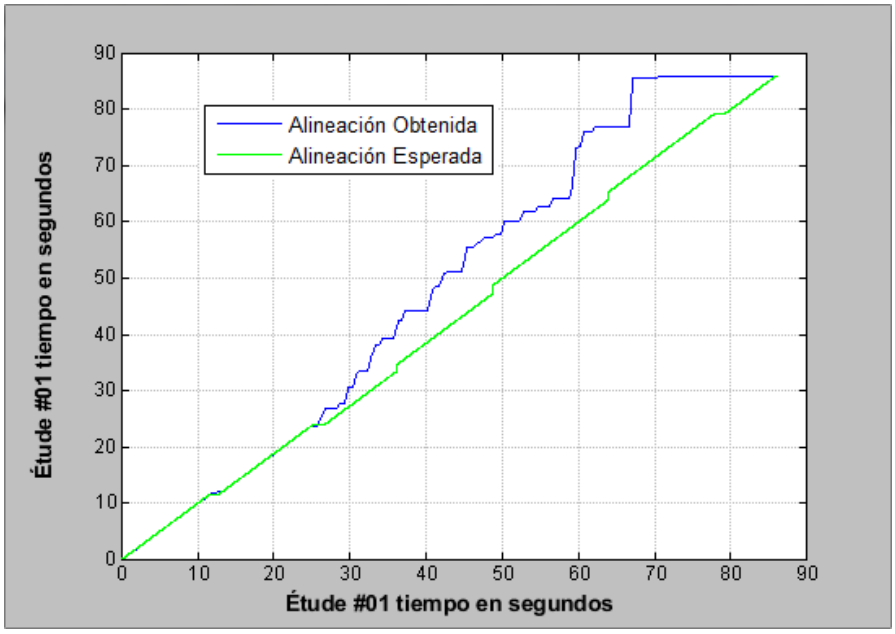


Figura B.21: Comportamiento del seguimiento con 50 % de ruido.

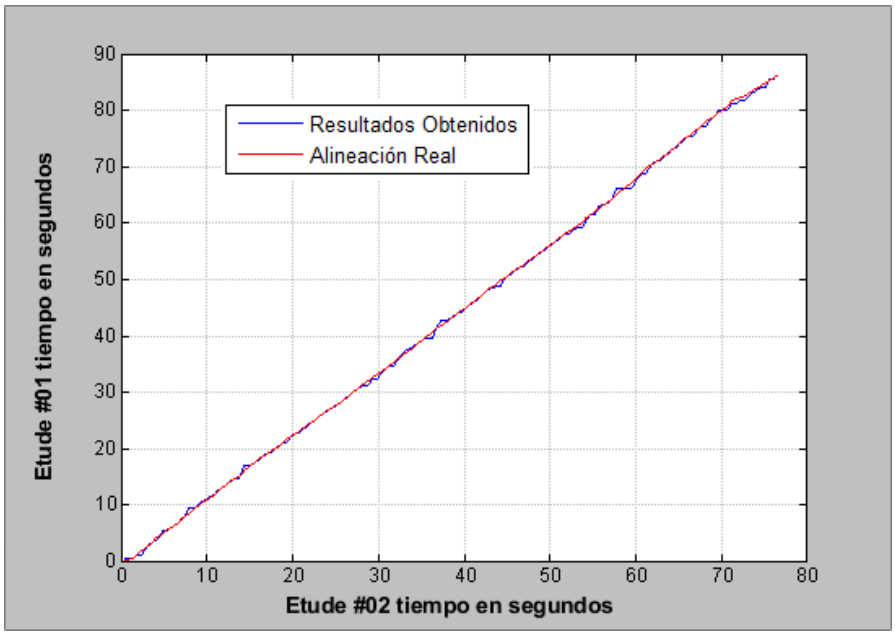


Figura B.22: Alineación de los Études #1 y #2 usando el #1 como referencia.

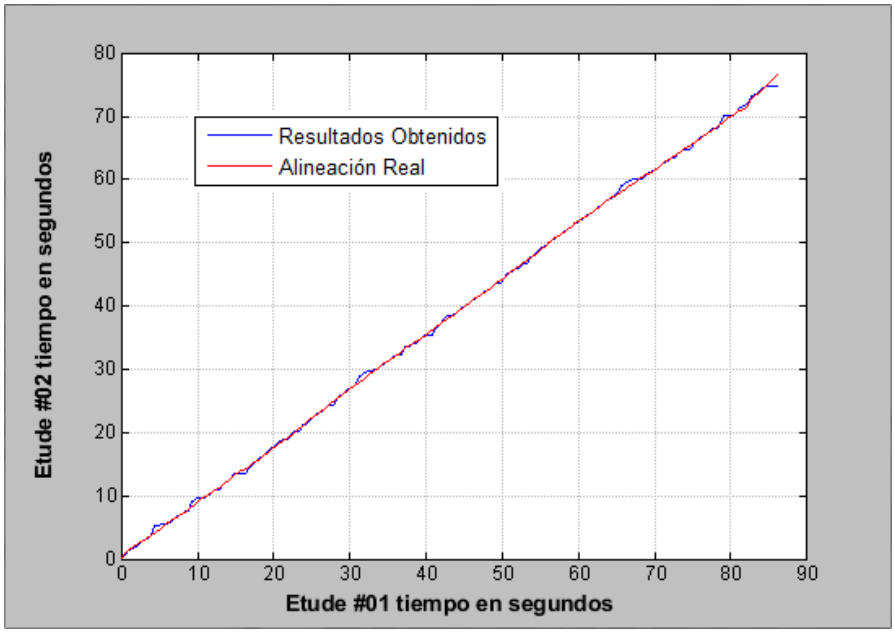


Figura B.23: Alineación de los Études #1 y #2 usando el #2 como referencia.

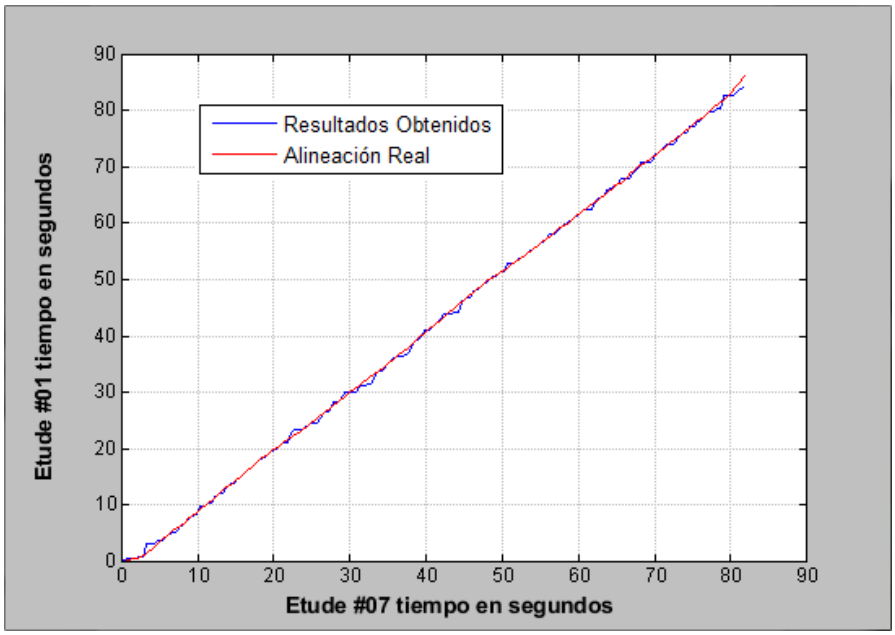


Figura B.24: Alineación de los Études #1 y #7 usando el #1 como referencia.

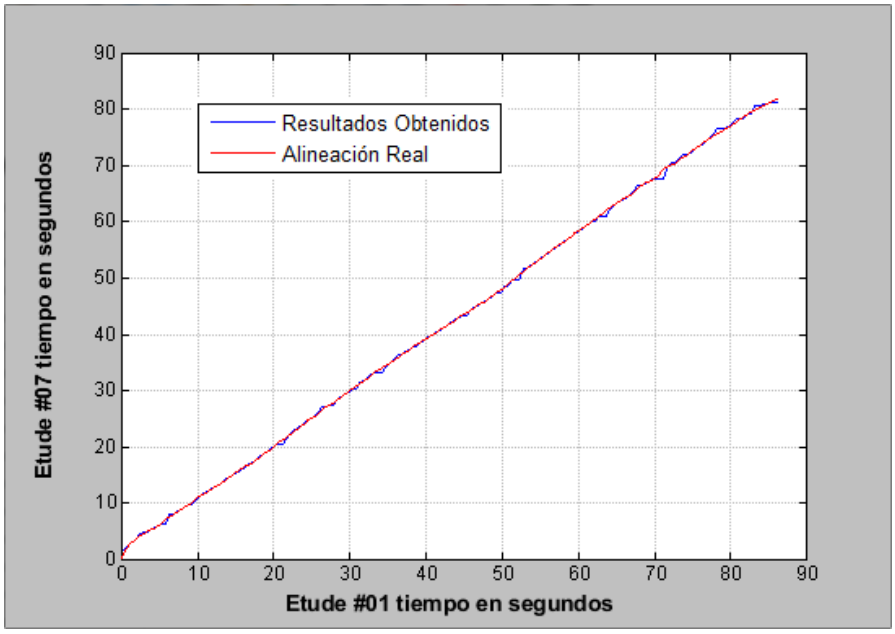


Figura B.25: Alineación de los Études #1 y #7 usando el #7 como referencia.

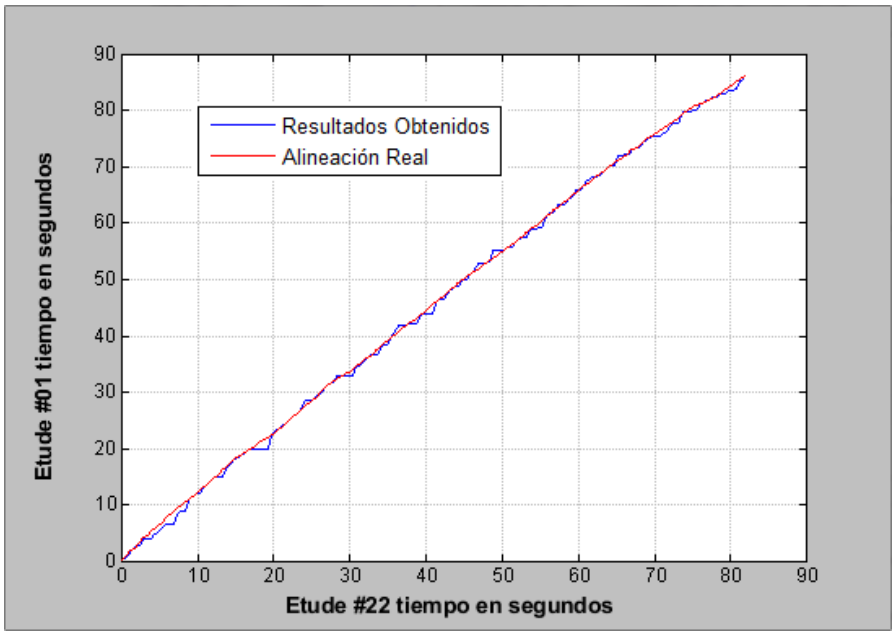


Figura B.26: Alineación de los Études #1 y #22 usando el #1 como referencia.

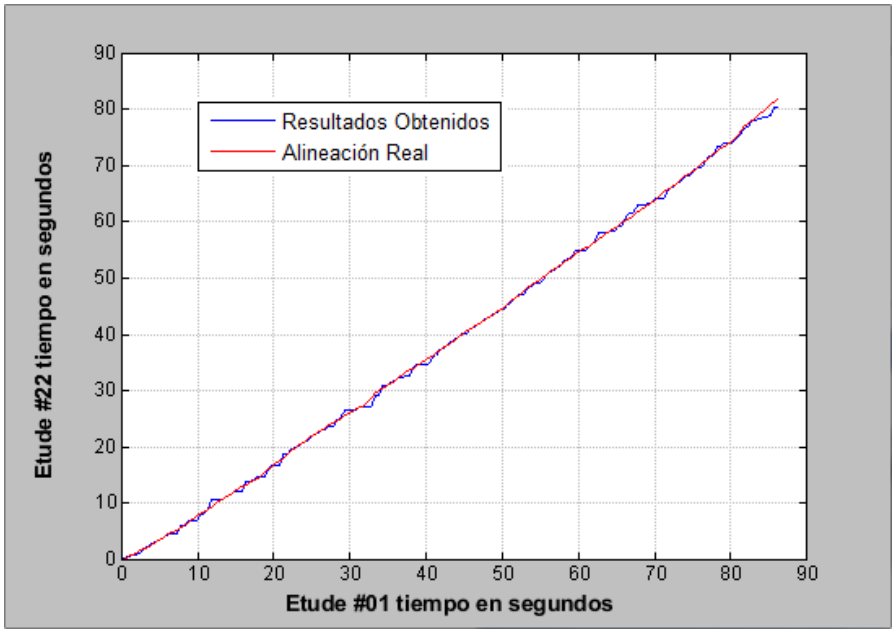


Figura B.27: Alineación de los Études #1 y #22 usando el #22 como referencia.

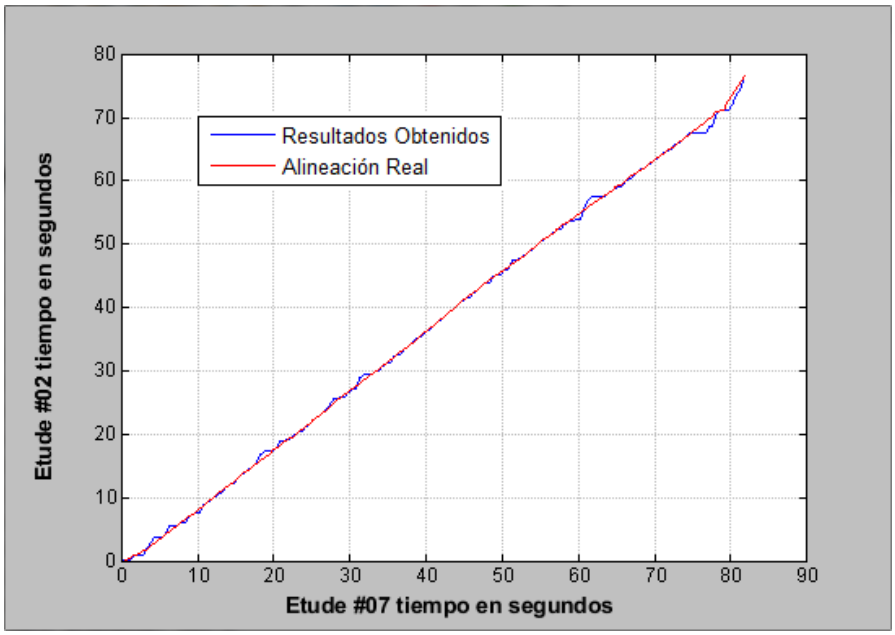


Figura B.28: Alineación de los Études #2 y #7 usando el #2 como referencia.

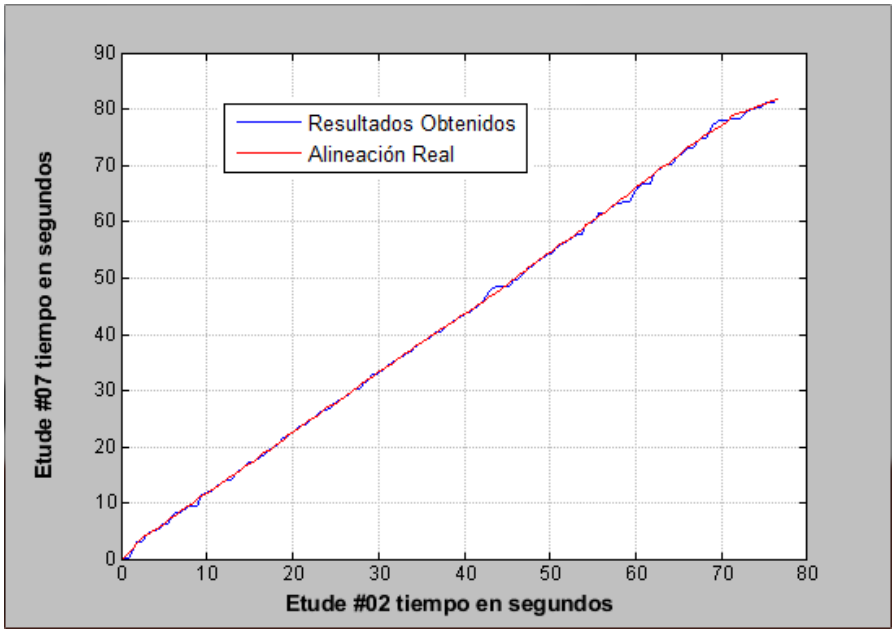


Figura B.29: Alineación de los Études #2 y #7 usando el #7 como referencia.

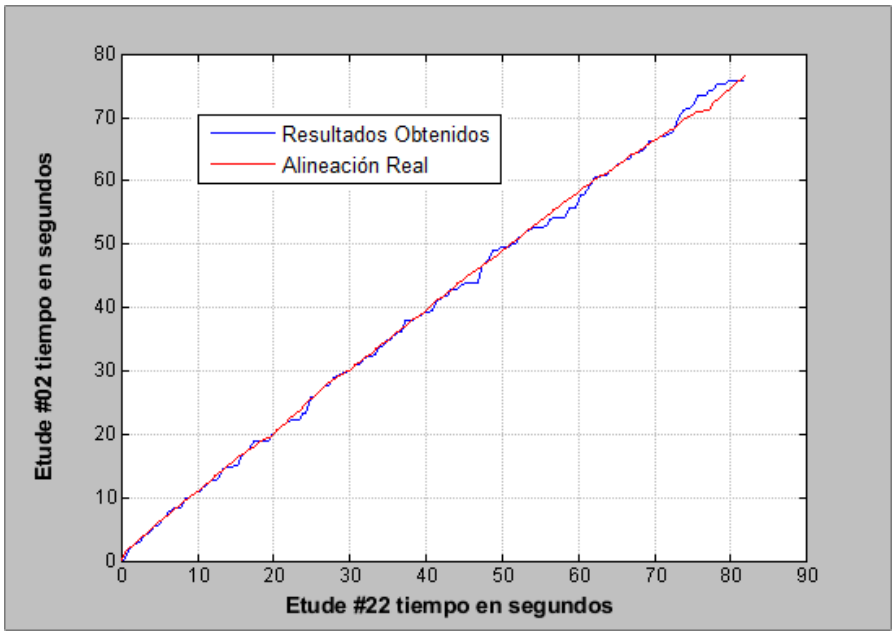


Figura B.30: Alineación de los Études #2 y #22 usando el #2 como referencia.

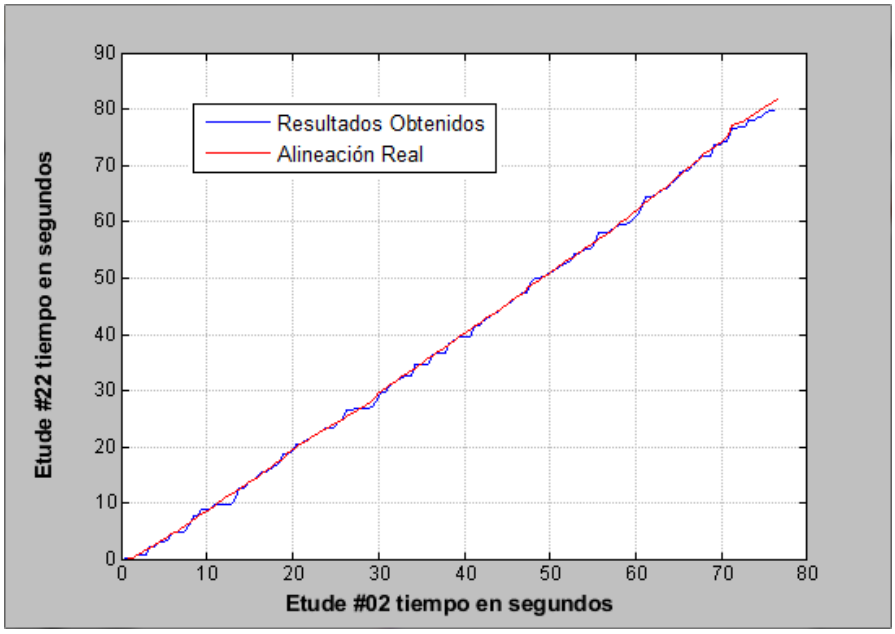


Figura B.31: Alineación de los Études #2 y #22 usando el #22 como referencia.

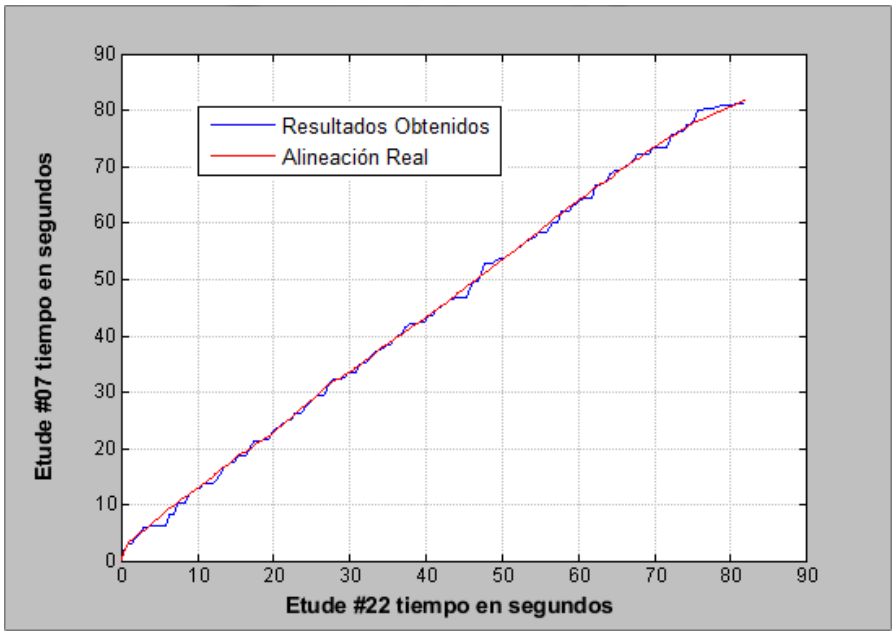


Figura B.32: Alineación de los Études #7 y #22 usando el #7 como referencia.

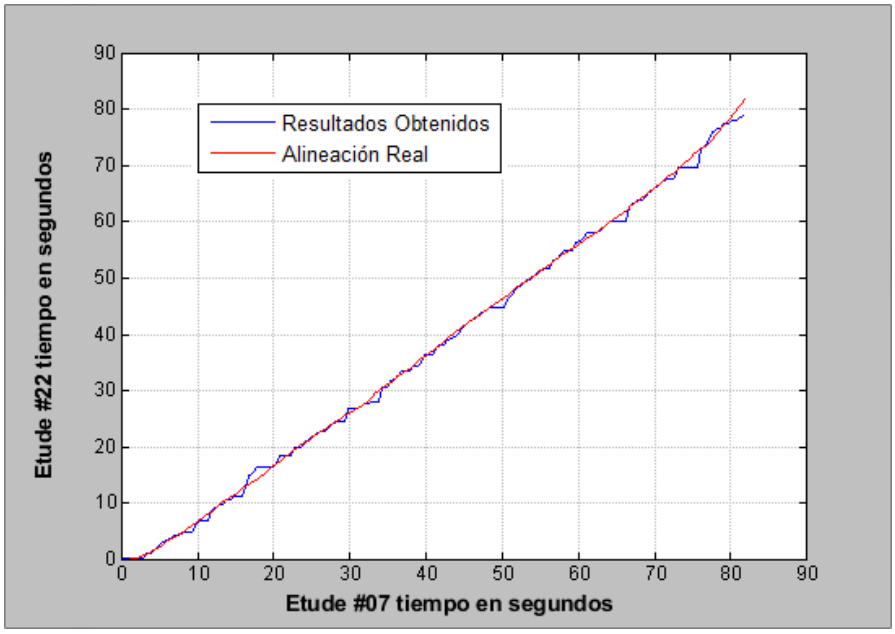


Figura B.33: Alineación de los Études #7 y #22 usando el #22 como referencia.

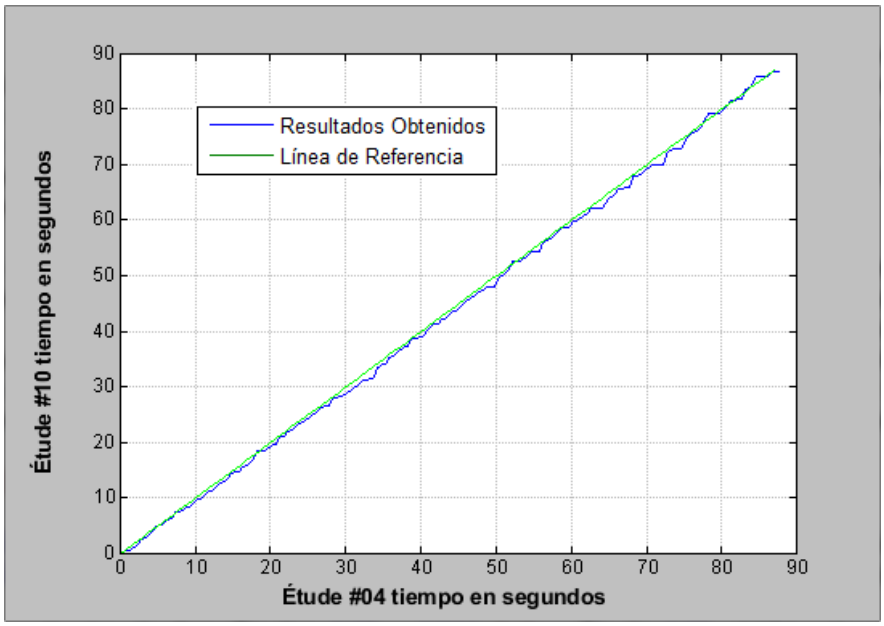


Figura B.34: Alineación de los Études #4 y #10 usando el #10 como referencia.

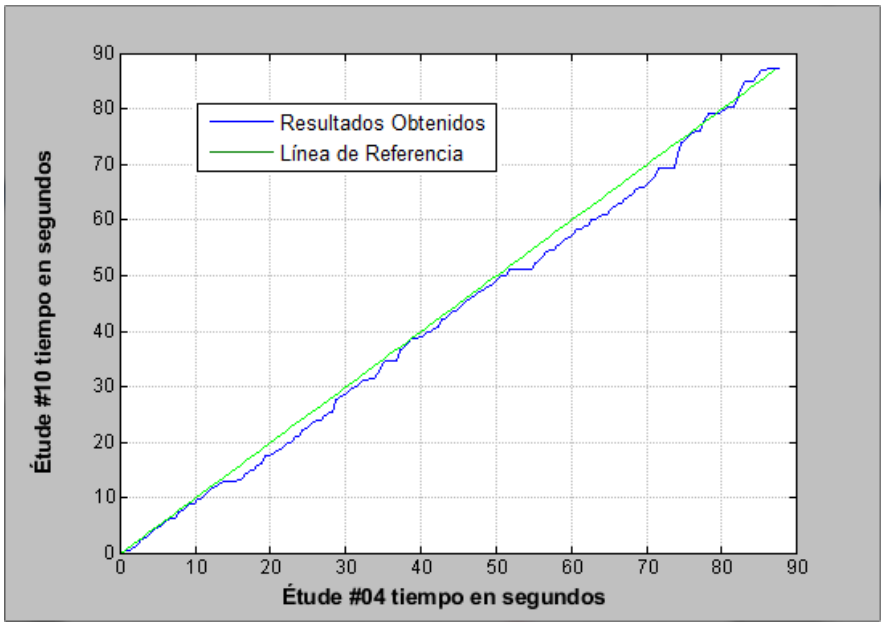


Figura B.35: Alineación de los Études #4 y #10 usando el #10 como referencia y modificando artificialmente la AFP del Étude #4.

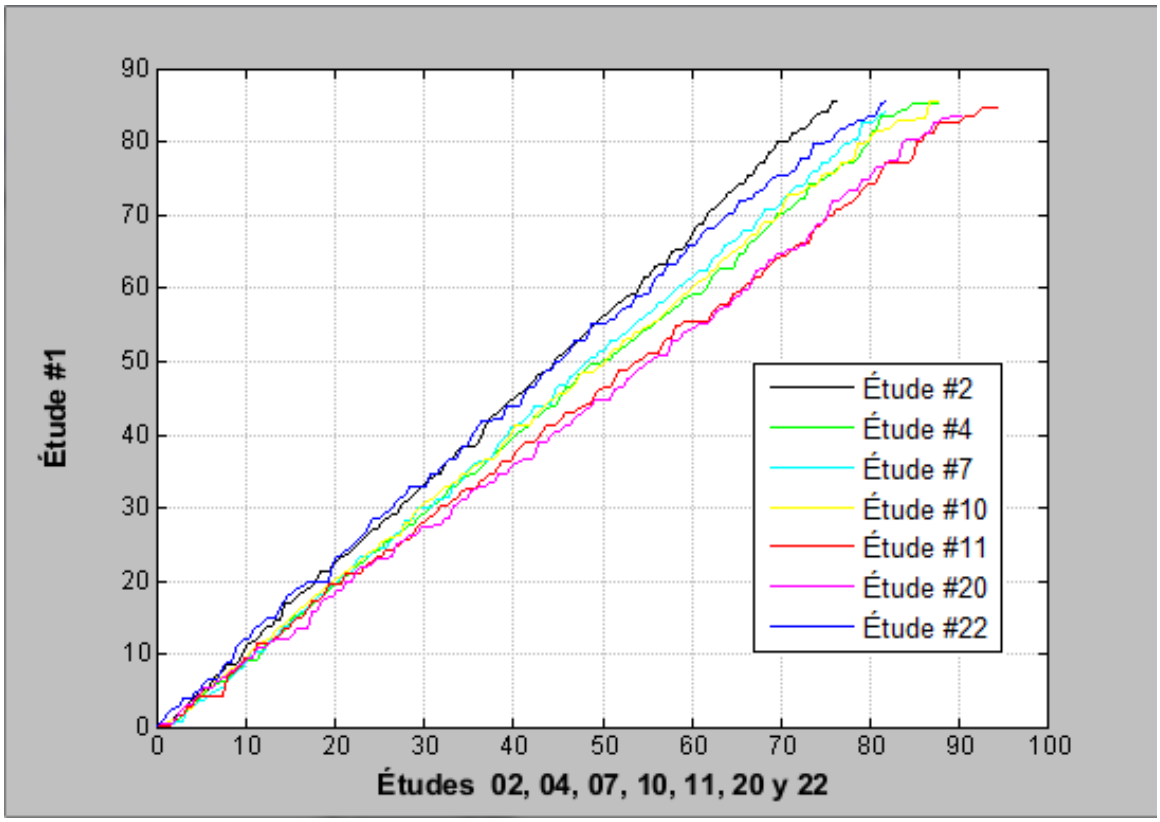


Figura B.36: Varias interpretaciones online contra una sola referencia.

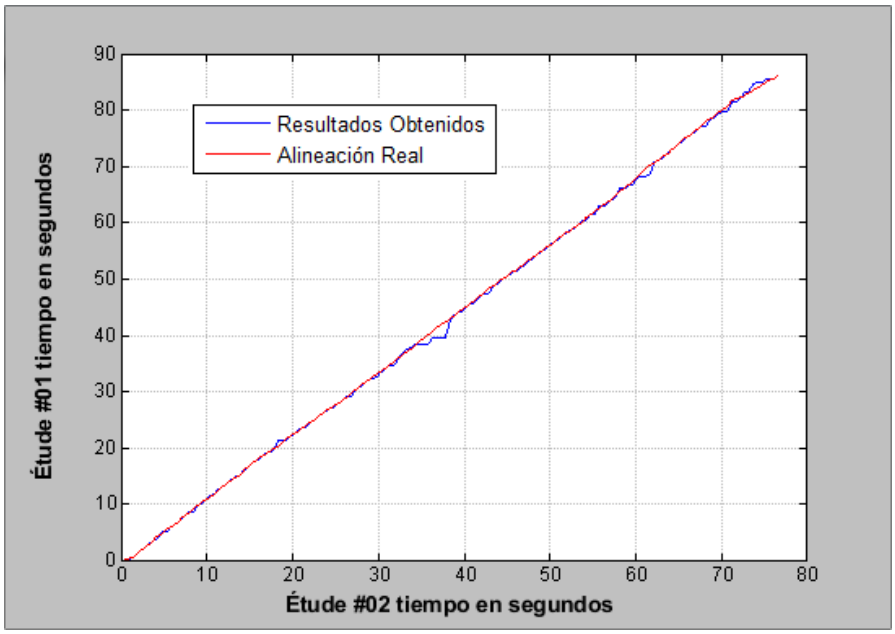


Figura B.37: Alineación de los Études #1 y #2 mediante el índice LSH usando el #1 como referencia.

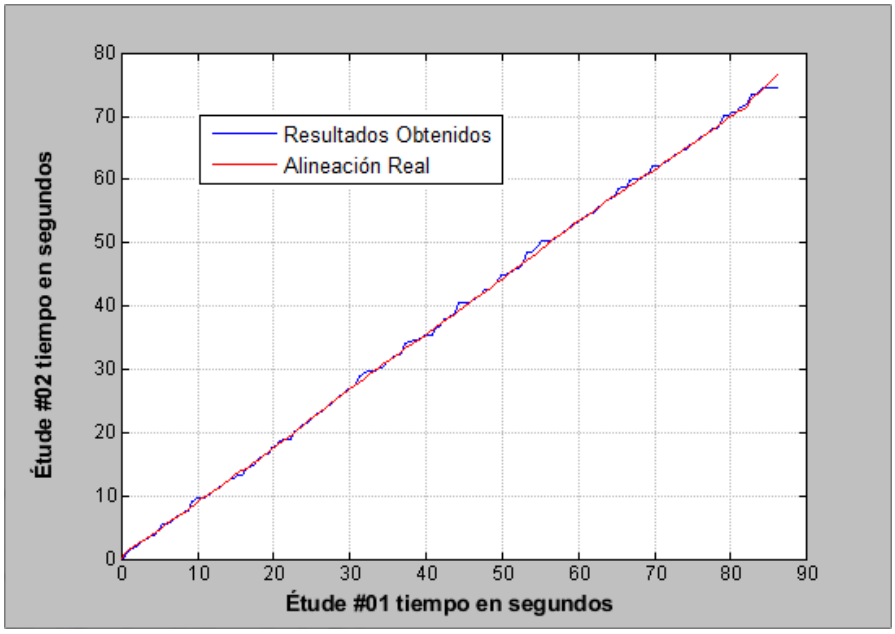


Figura B.38: Alineación de los Études #1 y #2 mediante el índice LSH usando el #2 como referencia.

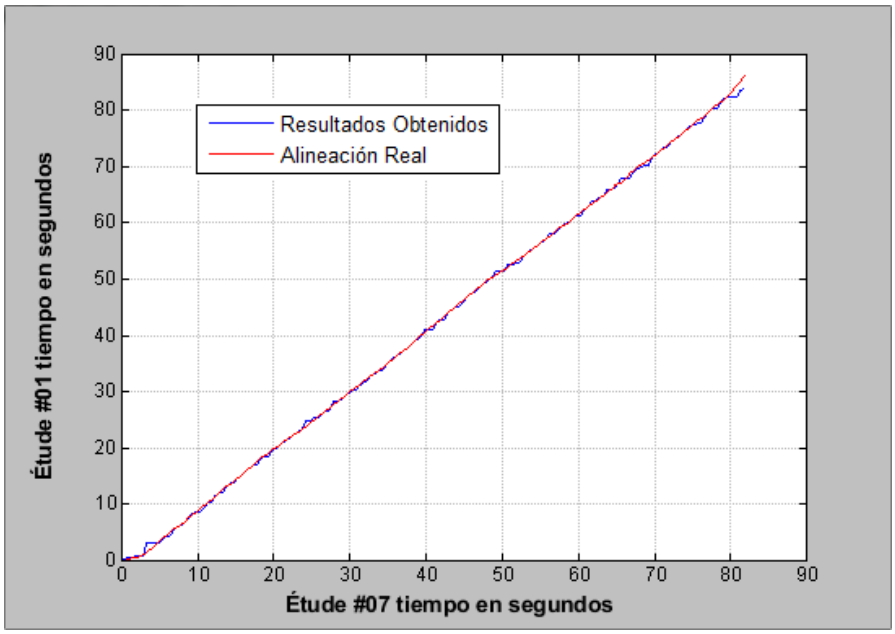


Figura B.39: Alineación de los Études #1 y #7 mediante el índice LSH usando el #1 como referencia.

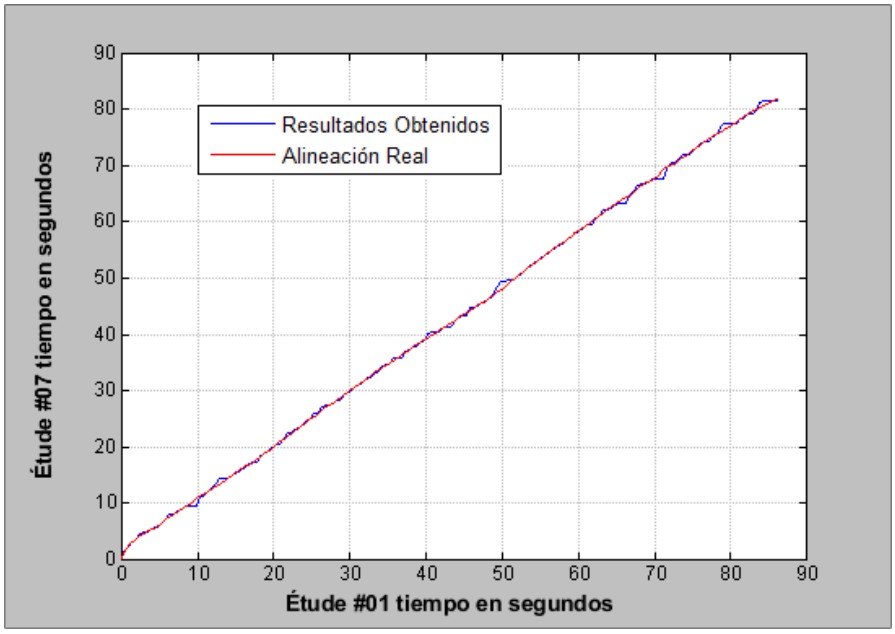


Figura B.40: Alineación de los Études #1 y #7 mediante el índice LSH usando el #7 como referencia.

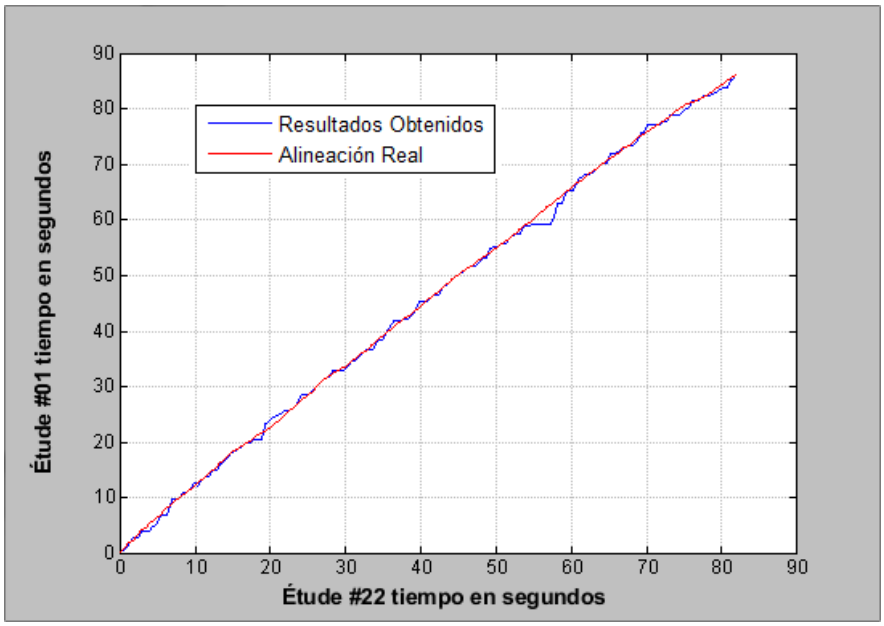


Figura B.41: Alineación de los Études #1 y #22 mediante el índice LSH usando el #1 como referencia.

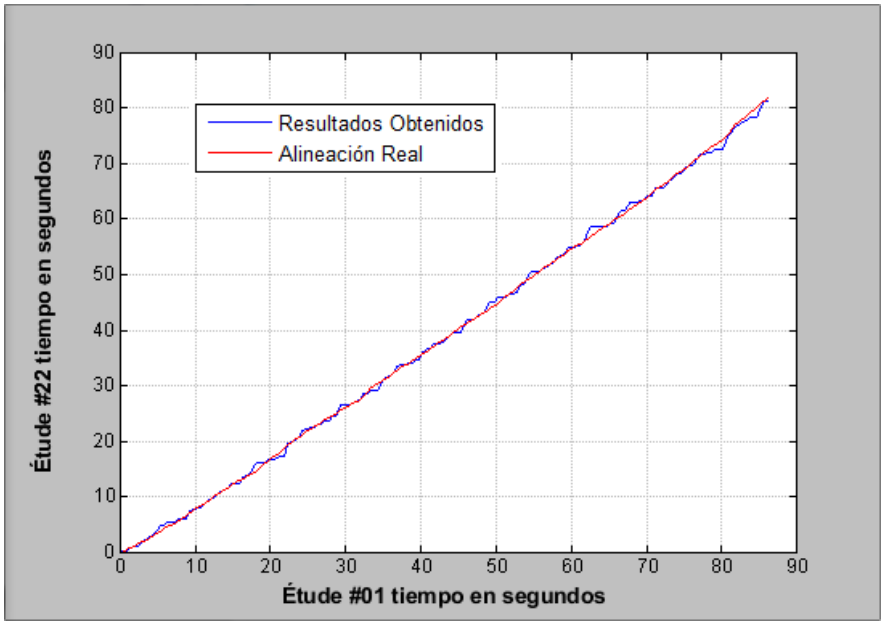


Figura B.42: Alineación de los Études #1 y #22 mediante el índice LSH usando el #22 como referencia.

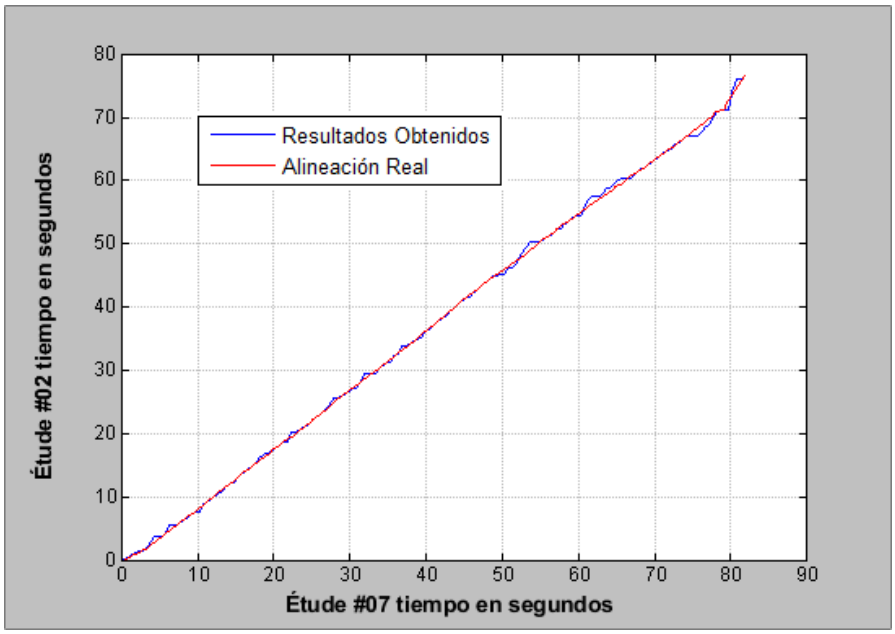


Figura B.43: Alineación de los Études #2 y #7 mediante el índice LSH usando el #2 como referencia.

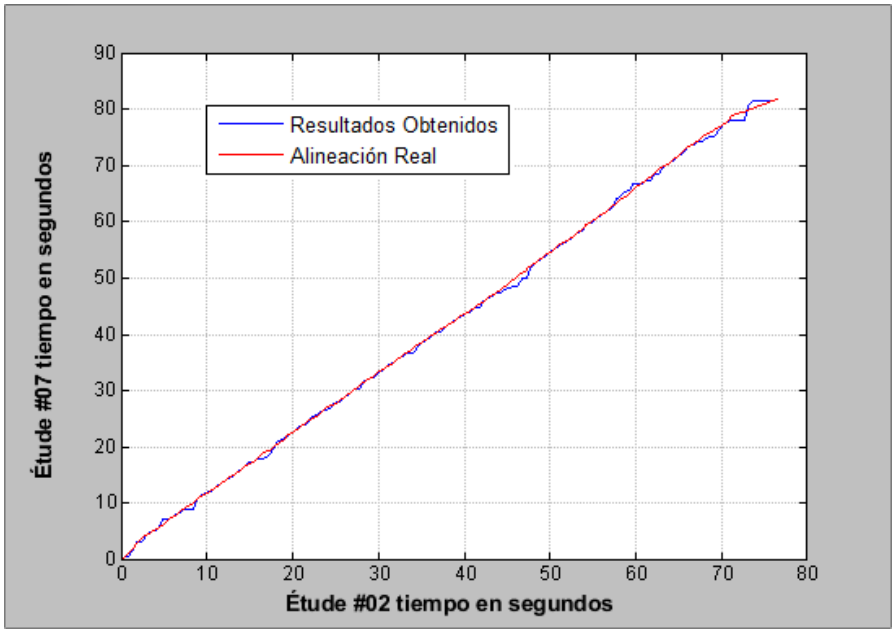


Figura B.44: Alineación de los Études #2 y #7 mediante el índice LSH usando el #7 como referencia.

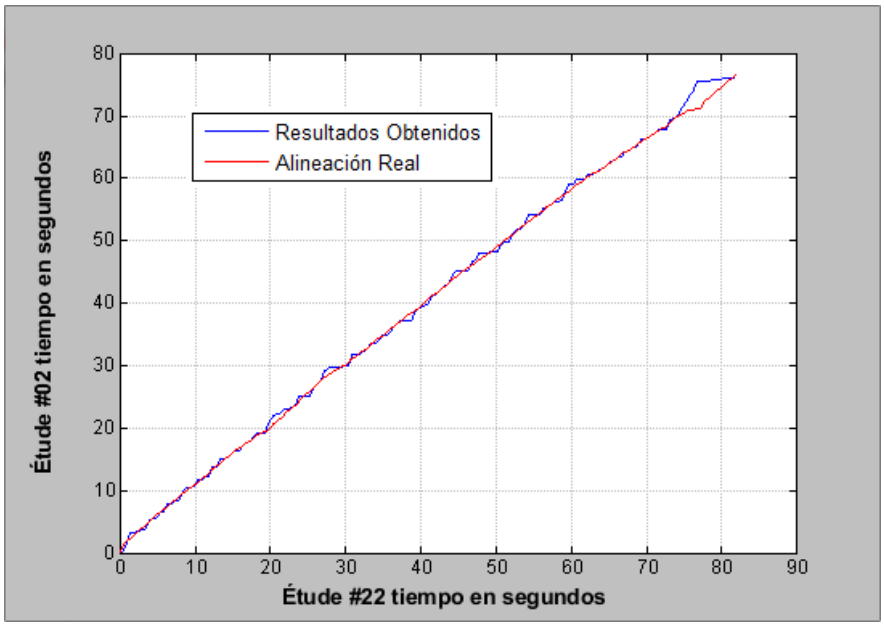


Figura B.45: Alineación de los Études #2 y #22 mediante el índice LSH usando el #2 como referencia.

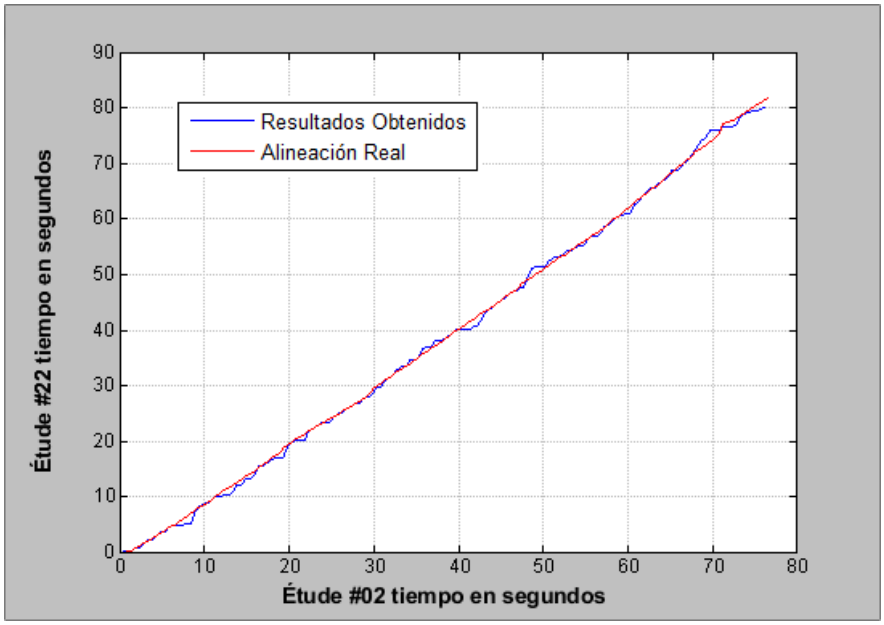


Figura B.46: Alineación de los Études #2 y #22 mediante el índice LSH usando el #22 como referencia.

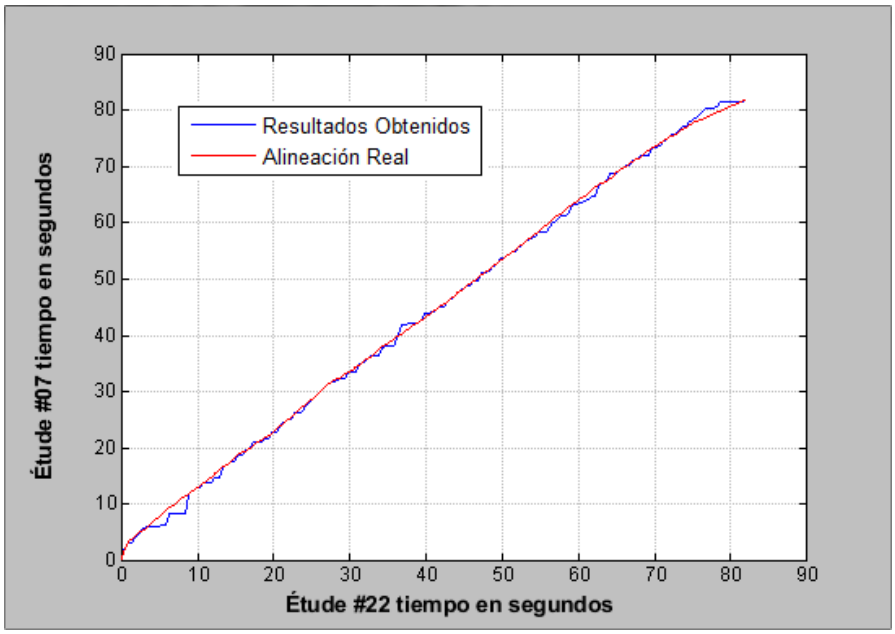


Figura B.47: Alineación de los Études #7 y #22 mediante el índice LSH usando el #7 como referencia.

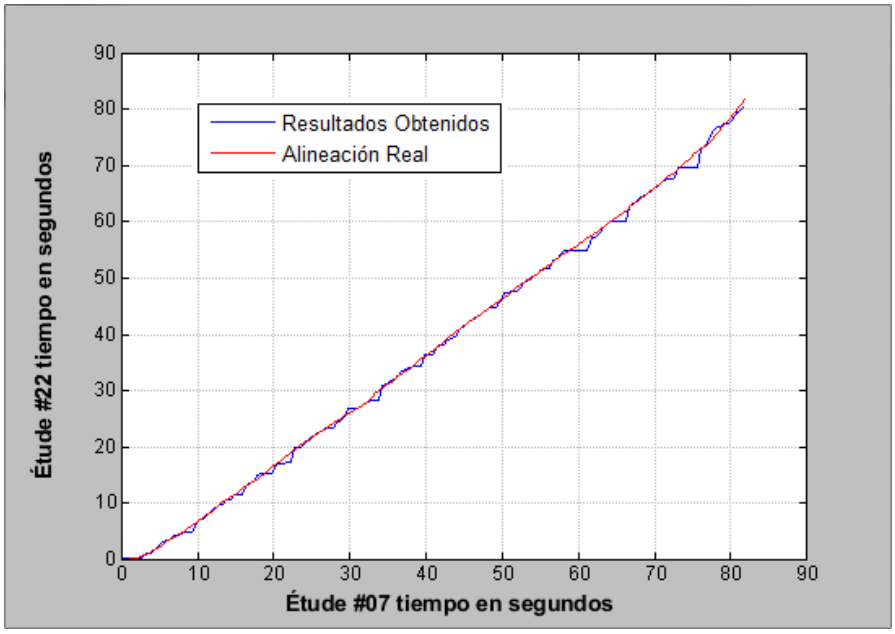


Figura B.48: Alineación de los Études #7 y #22 mediante el índice LSH usando el #22 como referencia.

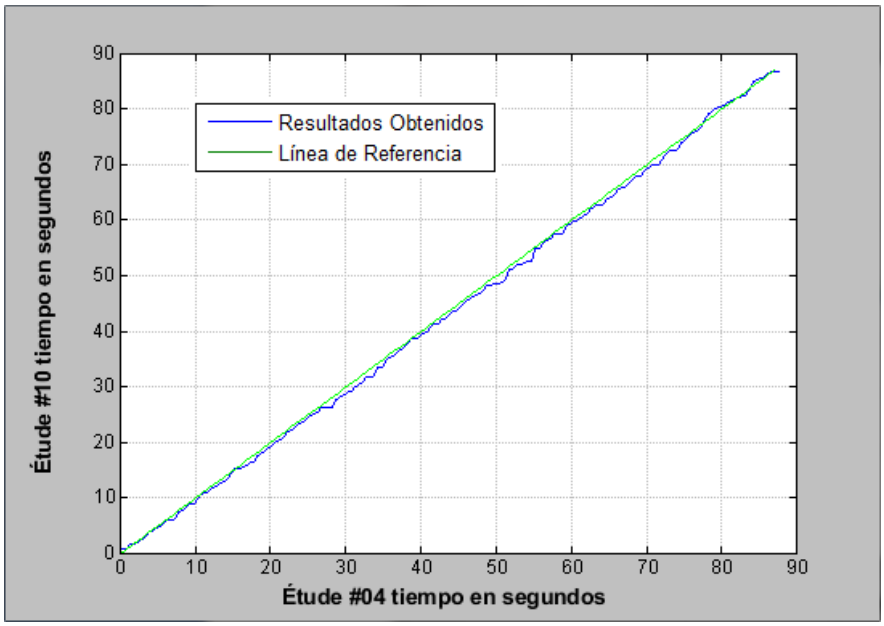


Figura B.49: Alineación de los Études #4 y #10 mediante el índice LSH usando el #10 como referencia, AFP #4 sin modificar.

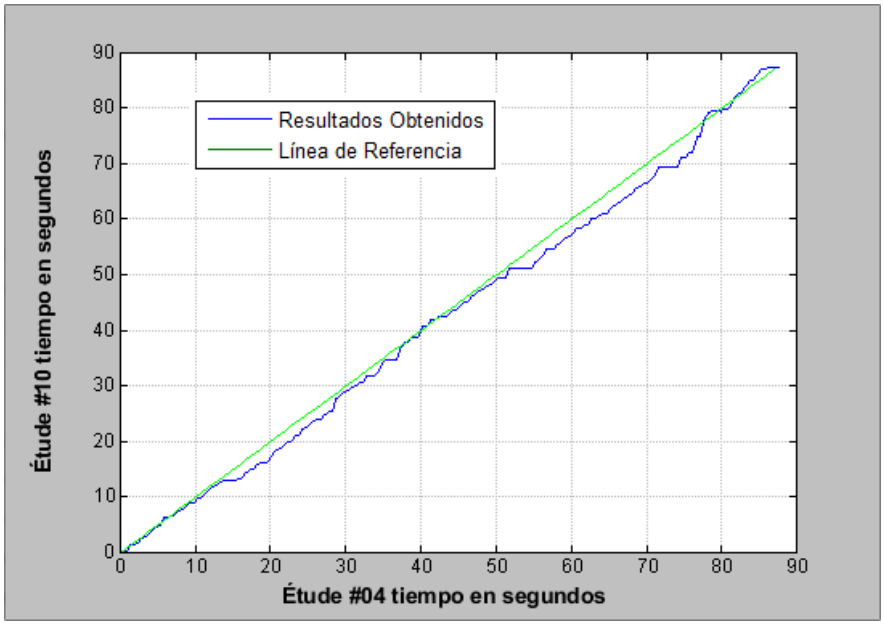


Figura B.50: Alineación de los Études #4 y #10 mediante el índice LSH usando el #10 como referencia, AFP #4 modificada artificialmente

Apéndice C

Mediciones

En este apéndice se incluye una tabla con las mediciones obtenidas de manera manual, sobre los tiempos de ejecución en segundos de las notas de los Études #1, #2, #7 y #22. Estas mediciones se utilizaron durante las Secciones 4.2 y 4.3 para crear las gráficas de alineación real entre estas interpretaciones.

Evento	Étude #1	Étude #2	Étude #7	Étude #22
0	0.0	0.0	0.0	0.0
1	0.4542	1.1321	1.8118	0.2259
2	1.1681	1.7195	3.0163	0.7704
3	1.9144	2.2642	3.7164	1.2273
4	2.4336	2.7234	4.2208	1.664
5	2.9636	3.14	4.612	2.0133
6	3.5153	3.6206	5.065	2.4344
7	4.2183	4.1012	5.59	2.8864
8	4.6834	4.5177	5.94	3.3795
9	5.1702	4.977	6.4548	3.7595
10	5.6893	5.3721	6.8975	4.0882
11	6.1328	5.81	7.4843	4.4991
12	6.5871	6.3013	7.9166	4.91
13	7.0522	6.6431	8.3387	5.3312
14	7.6579	7.113	8.9152	5.8242
15	8.3393	7.5616	9.4609	6.3378

Tabla C.1: Mediciones parte 1.

Evento	Étude #1	Étude #2	Étude #7	Étude #22
16	8.8044	7.9781	9.8521	6.759
17	9.2479	8.4053	10.2536	7.1699
18	9.7022	8.8219	10.6756	7.55
19	10.1997	9.2491	11.1595	7.9814
20	10.6216	9.6336	11.5404	8.382
21	11.0975	10.0501	11.9419	8.7723
22	11.6491	10.5093	12.4963	9.2859
23	12.2548	11.0113	12.9714	9.8714
24	12.7091	11.3958	13.3832	10.2926
25	13.2067	11.7482	13.7435	10.6932
26	13.7042	12.2609	14.2582	11.1349
27	14.2018	12.6668	14.7215	11.5561
28	14.6777	13.1046	15.1333	11.9259
29	15.0887	13.5212	15.5451	12.3265
30	15.6944	14.0231	16.0804	12.8914
31	16.4083	14.461	16.6569	13.3948
32	16.8951	14.8669	17.0584	13.8057
33	17.3493	15.3161	17.4599	14.1652
34	17.8253	15.7524	17.9129	14.5863
35	18.3553	16.1592	18.4894	15.0178
36	18.8095	16.5864	18.9835	15.5108
37	19.2855	17.0563	19.4468	16.045
38	19.8155	17.6438	20.0027	16.7127
39	20.3455	18.103	20.5895	17.2982
40	20.7889	18.5623	21.0425	17.6885
41	21.373	18.9468	21.4852	18.2021
42	21.9571	19.4594	22.1852	19.1985
43	22.4438	19.908	22.6691	19.7224
44	22.811	20.2912	23.0603	20.0614
45	23.2551	20.6876	23.5132	20.4312
46	23.6877	21.1255	23.9559	20.8112
47	24.0771	21.51	24.3677	21.1708
48	24.4881	21.9159	24.7486	21.52
49	24.8991	22.3217	25.1295	21.8385
50	25.3751	22.7489	25.5722	22.2391
51	25.97	23.2082	26.1281	22.691
52	26.3918	23.582	26.5502	23.1533
53	26.8028	24.0092	27.0238	23.5231
54	27.2463	24.4685	27.3635	23.8518
55	27.6357	24.8743	27.7444	24.1805
56	27.9926	25.2268	28.1047	24.4785

Tabla C.2: Mediciones parte 2.

Evento	Étude #1	Étude #2	Étude #7	Étude #22
57	28.4469	25.5792	28.5165	24.8482
58	28.912	25.9744	28.918	25.2077
59	29.3014	26.3909	29.4019	25.5878
60	29.68	26.7327	29.7622	25.9165
61	30.2208	27.1812	30.2563	26.2863
62	30.8806	27.7366	30.8946	26.7793
63	31.5187	28.2065	31.4917	27.2724
64	32.0055	28.7192	31.955	27.8065
65	32.5463	29.3493	32.5006	28.4023
66	33.39	30.1076	33.4065	29.7377
67	34.0389	30.631	33.9521	30.2821
68	34.5257	31.0475	34.4051	30.6725
69	34.98	31.4234	34.786	31.0936
70	35.4991	31.8592	35.2184	31.638
71	36.1265	32.3185	35.8258	32.1825
72	36.5267	32.7243	36.217	32.6139
73	37.0675	33.1515	36.6494	33.0453
74	37.4893	33.6214	37.1332	33.4357
75	37.9761	33.9525	37.6274	33.8465
76	38.4304	34.337	38.0186	34.2473
77	38.9063	34.7749	38.4201	34.6683
78	39.4687	35.2662	38.8834	35.1819
79	40.1177	35.7148	39.3672	35.7469
80	40.5936	36.11	39.7687	36.1578
81	41.0155	36.5158	40.2011	36.5584
82	41.4914	36.9217	40.6026	37.0001
83	41.9565	37.3382	40.9938	37.4007
84	42.4	37.744	41.4056	37.8116
85	42.8651	38.1606	41.7968	38.2533
86	43.4059	38.6198	42.2703	38.8182
87	44.0008	39.1111	42.8365	39.4346
88	44.4659	39.5063	43.2792	39.8763
89	44.8877	39.8908	43.6087	40.2666
90	45.4069	40.4141	44.041	40.6878
91	45.9153	40.852	44.4528	41.1295
92	46.3371	41.2685	44.9161	41.5198
93	46.813	41.6744	45.3073	41.9204
94	47.4079	42.1229	45.8117	42.4546
95	48.0028	42.5822	46.3265	43.0093
96	48.4679	43.0094	46.7486	43.3893
97	48.9438	43.4366	47.1912	43.7694

Tabla C.3: Mediciones parte 3.

Evento	Étude #1	Étude #2	Étude #7	Étude #22
98	49.3765	43.8852	47.6236	44.1392
99	49.874	44.2697	48.0972	44.5706
100	50.3716	44.7289	48.5707	45.0431
101	50.7934	45.1455	49.1678	45.5876
102	51.3775	45.6688	49.7443	46.2142
103	51.8642	46.1494	50.3723	46.7586
104	52.3077	46.6193	50.7841	47.19
105	52.8485	47.0359	51.2474	47.6728
106	53.4434	47.5699	51.9989	48.4329
107	53.9734	48.0398	52.5136	49.0082
108	54.3953	48.4136	52.946	49.4191
109	54.8495	48.8622	53.3475	49.8094
110	55.2822	49.2894	53.8005	50.21
111	55.704	49.6312	54.2324	50.6414
112	56.0826	50.0263	54.6138	51.001
113	56.5044	50.4108	55.005	51.3913
114	56.9371	50.7846	55.4477	51.8638
115	57.4455	51.1798	55.9418	52.3363
116	57.8673	51.5643	56.333	52.7164
117	58.2567	51.9274	56.7551	53.1068
118	58.6785	52.3333	57.1463	53.5074
119	59.122	52.7285	57.5684	53.908
120	59.5114	53.0916	57.9596	54.2675
121	59.944	53.444	58.392	54.6168
122	60.5065	53.8499	58.9067	55.1098
123	61.0365	54.2237	59.4523	55.5823
124	61.4583	54.6295	59.895	55.9316
125	61.891	55.014	60.2862	56.3117
126	62.3344	55.3985	60.698	56.7123
127	62.7887	55.8151	61.1613	57.1437
128	63.2214	56.2423	61.6142	57.5957
129	63.7514	56.7336	62.0775	58.0682
130	64.5193	57.3744	62.8702	58.7667
131	65.2873	57.9297	63.6011	59.5063
132	65.8389	58.303	64.0438	59.9993
133	66.3257	58.6987	64.435	60.3178
134	66.7475	59.1046	64.8262	60.7697
135	67.2018	59.4677	65.4027	61.232
136	67.7102	59.8736	65.9792	61.7251
137	68.2404	60.2794	66.4116	62.2181
138	68.9432	60.7707	66.9366	63.0091

Tabla C.4: Mediciones parte 4.

Evento	Étude #1	Étude #2	Étude #7	Étude #22
139	69.603	61.1872	67.4823	63.574
140	70.133	61.6144	67.8838	63.9952
141	70.5224	61.9882	68.2853	64.4266
142	70.9767	62.4582	68.9441	64.93
143	71.4202	62.8747	69.5412	65.5052
144	71.8744	63.2592	69.9118	65.9366
145	72.3828	63.7185	70.3236	66.368
146	73.021	64.135	70.8384	66.9638
147	73.5618	64.5836	71.3737	67.6829
148	73.9944	64.9574	71.7649	68.104
149	74.492	65.3632	72.1973	68.5663
150	75.0436	65.7797	72.7532	69.1107
151	75.5844	66.1749	73.2885	69.6859
152	76.0604	66.5808	73.7621	70.1687
153	76.493	67.0293	74.1533	70.6823
154	77.0879	67.5313	74.7401	71.463
155	77.7369	67.9906	75.3063	72.141
156	78.2453	68.4287	75.7181	72.6135
157	78.7212	68.845	76.1505	73.0655
158	79.2512	69.3149	76.6343	73.4866
159	79.7163	69.7742	77.0255	73.9797
160	80.2895	70.2441	77.4991	74.6576
161	80.9602	70.917	78.0344	75.3151
162	81.9228	71.2908	79.0021	77.0716
163	82.1824	71.6325	79.208	77.4106
164	82.4744	72.4122	79.4963	77.7598
165	86.0792	76.4662	81.8435	81.7429

Tabla C.5: Mediciones parte 5.

Apéndice D

Partituras

Este apéndice contiene las partituras de las interpretaciones utilizadas durante los experimentos mostrados en el Capítulo 4. Se incluyen con el fin de que el lector pueda consultarlas para corroborar las afirmaciones que se hacen respecto a ellas.

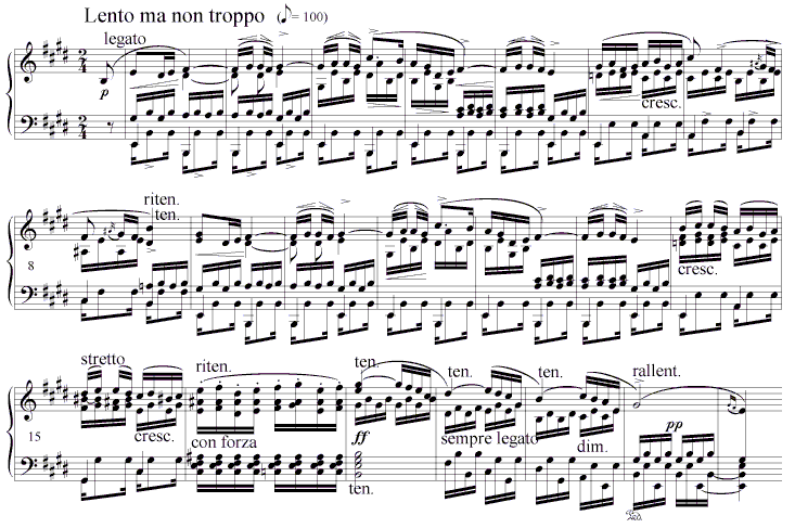


Figura D.1: Etude en E Major, Op. 10, no. 3, barras 1-21, Frédéric Chopin.

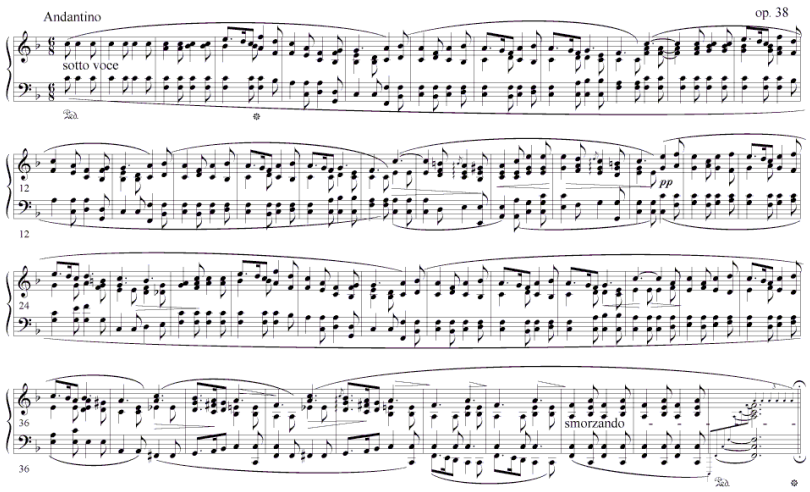


Figura D.2: Ballade Op. 38, barras 1-45, Frédéric Chopin.

Apéndice E

Formas de Onda

En este apéndice se incluyen las formas de onda de las grabaciones utilizadas en los ejemplos presentados en las secciones 4.2 y 4.3. Se incluyen con el propósito de que el lector pueda corroborar las características de las que se hace mención en dichas secciones. Todas las grabaciones pueden ser obtenidas en línea en Goebl [2013].

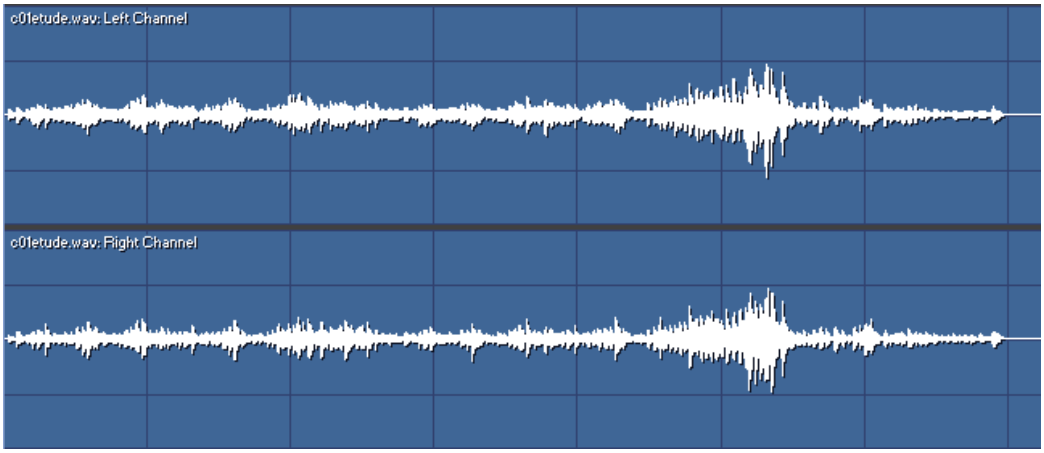


Figura E.1: Forma de onda Etude #1, duración 1:26:07

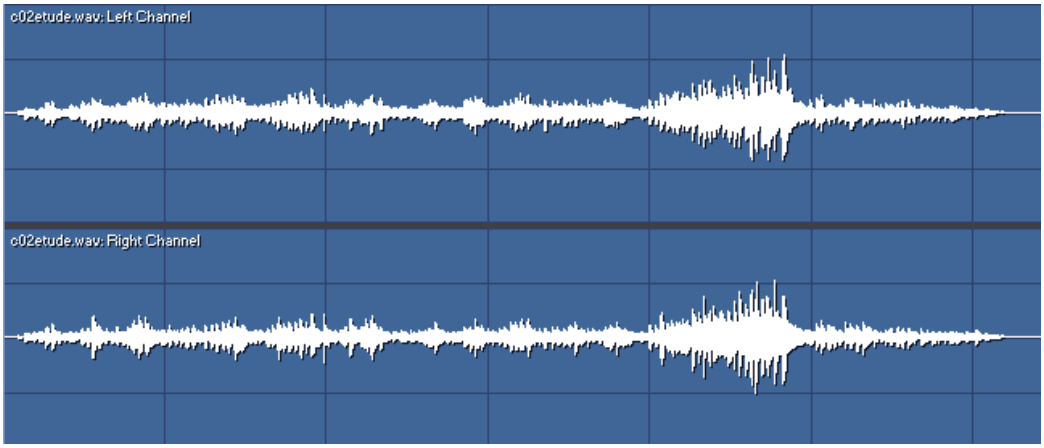


Figura E.2: Forma de onda Etude #2, duración 1:16:46

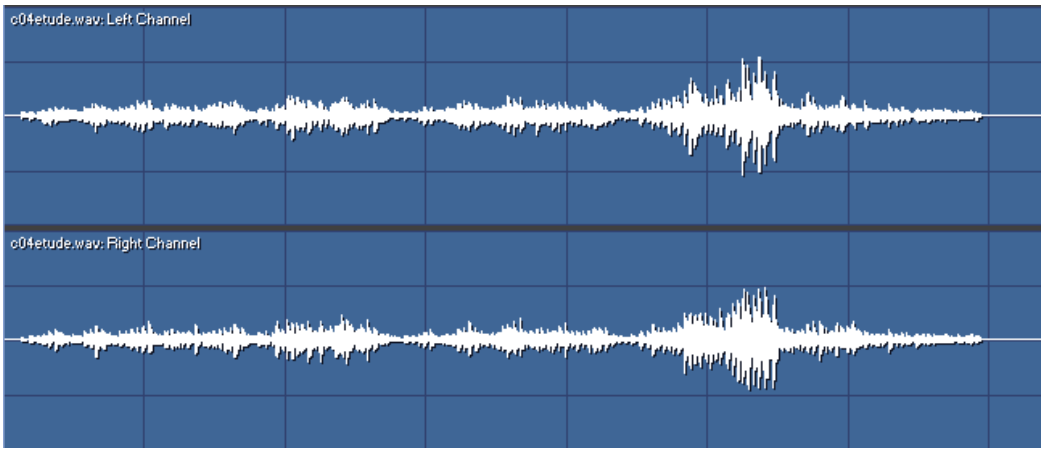


Figura E.3: Forma de onda Etude #4, duración 1:27:72

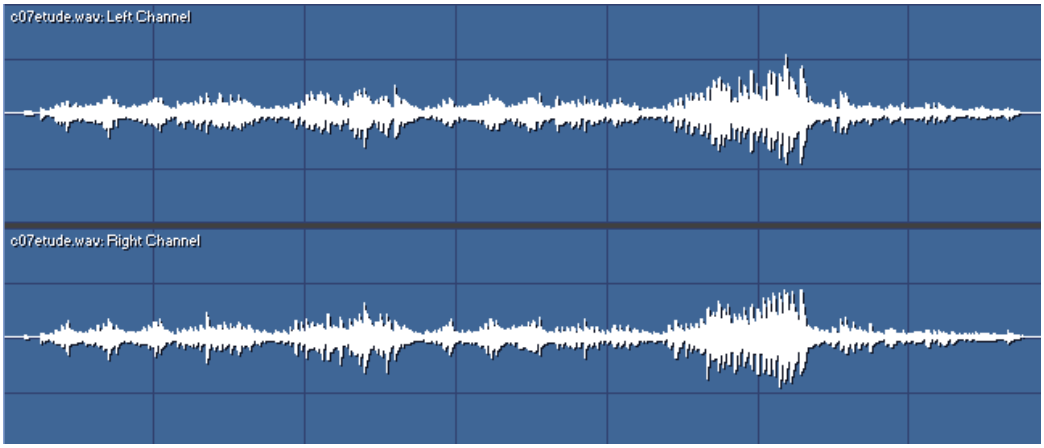


Figura E.4: Forma de onda Etude #7, duración 1:21:84

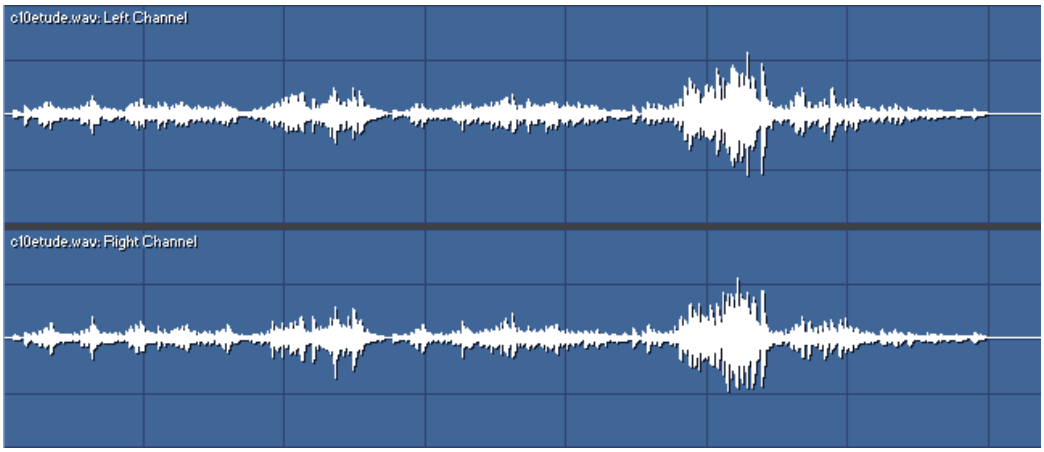


Figura E.5: Forma de onda Etude #10, duración 1:27:80

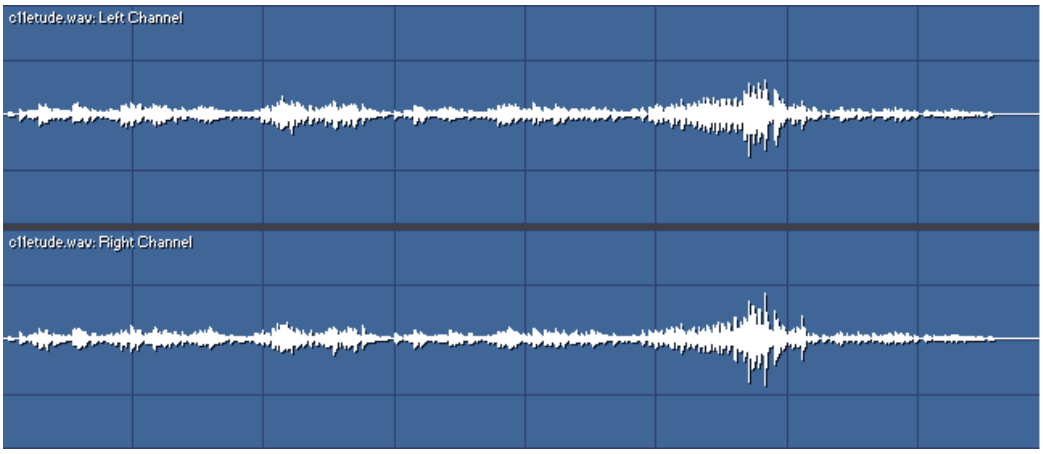


Figura E.6: Forma de onda Etude #11, duración 1:34:38

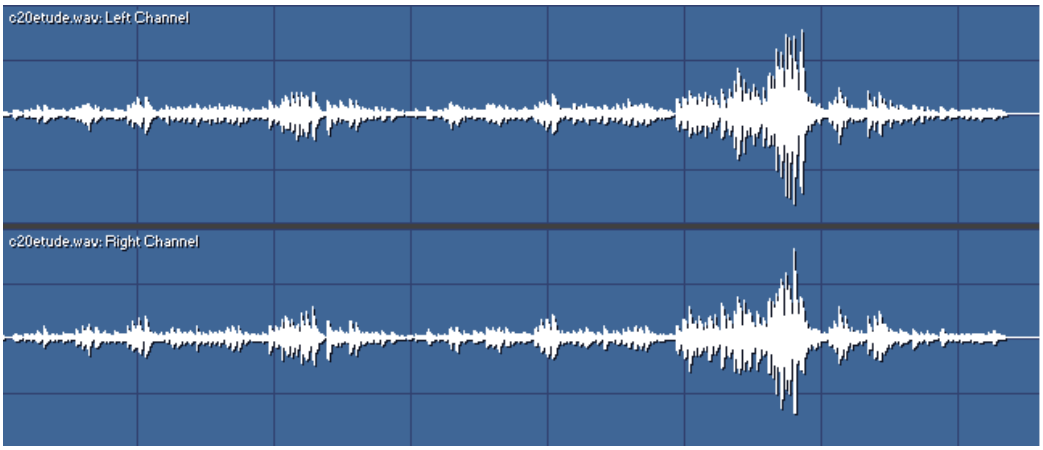


Figura E.7: Forma de onda Etude #20, duración 1:30:36

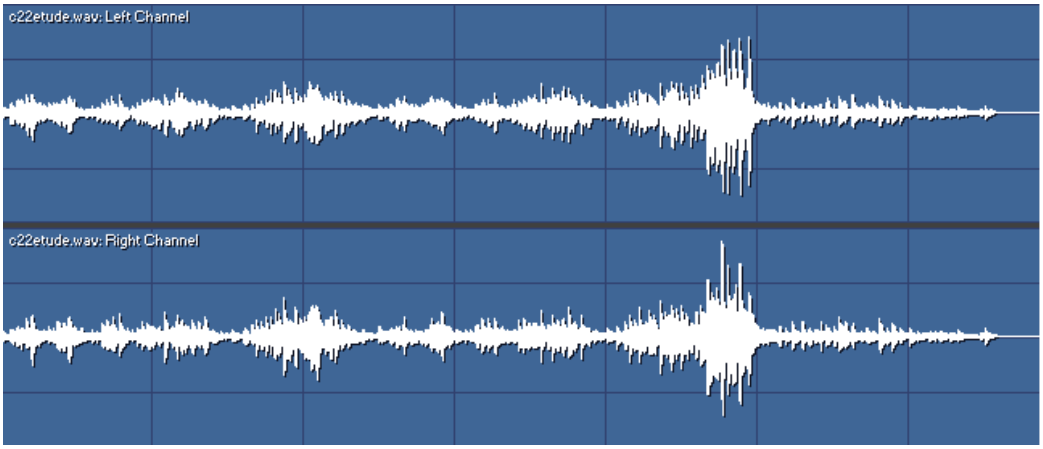


Figura E.8: Forma de onda Etude #22, duración 1:21:74

Referencias

Andoni, A., y Indyk, P. (2008). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications ACM*, *51*, 117–122.

Bellman, R. (1961). *Adaptive control processes - A guided tour.*. Princeton University Press.

Camarena, A., y Chávez, E. (2006). A robust entropy-based audio-fingerprint. *IEEE International Conference on Multimedia and Expo*, *1*, 1729–1731.

Cannam, C., Landone, C., y Sandler, M. (2010). Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files. *Proceedings of the ACM Multimedia 2010 International Conference*, (pp. 1467–1468).

Cano, P., Loscos, A., y Bonada, J. (1999). Score-performance matching using hmms. *Proceedings of International Computer Music Conference (ICMC)*.

Chou, T., Chen, W., Wang, S., Chang, K., y Chen, H. (2012). Real-time polyphonic score following system. *IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, (pp. 205–210).

Cont, A. (2006). Realtime audio to score alignment for polyphonic music instruments using sparse non-negative constraints and hierarchical hmms. *IEEE International Conference on Acoustics and Speech Signal Processing (ICASSP)*.

Cont, A. (2010). A coupled duration-focused architecture for realtime music to score alignment. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, *32(6)*, 974–987.

Dannenberg, R. (1984). An on-line algorithm for real-time accompaniment. *Proceedings of the International Computer Music Conference (ICMC)*, (pp. 193–198).

Dannenberg, R., y Grubb, L. (1989). Real-time scheduling and computer accompaniment. *MIT Press Series in System Development Foundation Benchmark*, (pp. 225–261).

Davis, S. B., y Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *28*, 357–366.

de Cheveigné, A. (2006). Múltiple f0 estimation. *Computational Auditory Scene Analysis: Principles, Algorithms and Applications*.

Desain, P., Honing, H., y Heijink, H. (1997). Robust score-performance matching: Taking advantage of structural information. *Proceedings of International Computer Music Conference (ICMC)*.

Deza, E., y Deza, M. M. (2009). *Encyclopedia of Distances*. Springer.

Dixon, S. (2005). Live tracking of musical performances using on-line time warping. *8th International Conference on Digital Audio Effects*.

Dixon, S. (2010). Simon dixon's personal page. Online. Http://www.eecs.qmul.ac.uk/ simond/match/.

Dixon, S., y Widmer, G. (2005). Match: A music alignment tool chest. *6th International Conference on Music Information Retrieval*, (pp. 492–497).

Fine, S., Singer, Y., y Tishby, N. (1998). The hierarchical hidden markov model : Analysis and applications. *Machine Learning*, *32*(1), 41–62.

Fujishima, T. (1999). Realtime chord recognition of musical sound: A system using common lisp music. *Proceedings of the International Computer Music Conference*.

Gerhard, D. (2003). *Pitch extraction and fundamental frequency: History and current techniques*. University of Regina.

Gionis, A., Indyk, P., y Motwani, R. (1999). Similarity search in high dimensions via hashing. *Proceedings of the 25th International Conference on Very Large Databases*, *1*, 518–529.

Goebl, W. (2001). Melody lead in piano performance: Expressive device or artifact? *Journal of the Acoustical Society of America*, *110*(1), 563–572.

Goebl, W. (2013). Werner goebl's personal page. Online. Http://iwk.mdw.ac.at/goebl/mp3.html.

Grubb, L., y Dannenberg, R. (1998). Enhanced vocal performance tracking using multiple information sources. *Proceedings of ICMC*, (pp. 37–44).

Haitsma, S., y Kalker, T. (2002). A highly robust audio fingerprintig system. *ISMIR*.

Haitsma, S., Kalker, T., y Oostveen, J. (2001). Robust audio hashing for content identification. *Content Based Multimedia Indexing*.

Hamming, R. (1950). Error detecting and error correcting codes. *Bell System Technical Journal*, *29*, 147–160.

Hirschber, D. S. (1975). A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*, *18*, 341–343.

Hu, N., Dannenberg, R., y Tzanetakis, G. (2003a). Polyphonic audio matching and alignment for music retrieval. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*.

Hu, N., Dannenberg, R., y Tzanetakis, G. (2003b). Polyphonic audio matching for score following and intelligent audio editors. *Proceedings of the ICMC*.

Johnston, J. D. (1988). Transform coding of audio signals using perceptual noise criteria. *IEEE Journal on Selected Areas in Communications*, *6*(2), 314–332.

Jordan, M. I. (1998). *Learning in Graphical Models*. MIT Press.

Jordanus, A. (2007). *Score Following: An Articially Intelligent Musical Accompanist*. Tesis de Maestría, University of Edinburgh.

Kirchhoff, H., y Lerch, A. (2011). Evaluation of features for audio-to-audio alignment. *Journal of New Music Research*, *40*(1), 27–41.

Krause, E. F. (1986). *Taxicab Geometry: An Adventure in Non-Euclidean Geometry*. Courier Dover Publications.

Levenshtein, V. (1966). Binary codes capable of correction deletions, insertions and reversals. *Soviet Physics Doklady*, *10*, 707–710.

Orio, N., y Déchelle, F. (2001). Score following using spectral analysis and hidden markov models. *Proceedings of International Computer Music Conference (ICMC)*.

Orio, N., Lemouton, S., y Schwarz, D. (2003). Score following: State of the art and new developments. *Proceedings of the Conference on New Interfaces for Musical Expression (NIME)*.

Orio, N., y Schwarz, D. (2001). Alignment of monophonic and polyphonic music to a score. *Proceedings of International Computer Music Conference (ICMC)*.

O'Shaughnessy, D. (1988). Linear predictive coding. *IEEE Potentials*, *7*(1), 29–32.

Pohlmann, K. (1995). *Principles of Digital Audio*. McGraw-Hill.

Puckette, M., y Lippe, C. (1992). Score following in practice. *Proceedings of International Computer Music Conference (ICMC)*.

Rabiner, L. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*, 257–286.

Rabiner, L. R., y Juang, B. H. (1993). *Fundamentals of Speech Recognition*. Prentice Hall.

Raphael, C. (1999). Automatic segmentation of acoustic musical signals using hidden markov models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21(4)*, 360–370.

Raphael, C. (2006). Aligning music audio with symbolic scores using a hybrid graphical model. *Machine Learning*, *65*(2-3), 389–409.

Santoyo, F., y Chávez, E. (2012). *Índices Comprimidos para Búsqueda por Proximidad en Series de Tiempo*. Tesis de Maestría, Universidad Michoacana de San Nicolás de Hidalgo.

Santoyo, F., Chávez, E., y Tellez, E. (2012). Compressing locality-sensitive hashing tables.

Shannon, C., y Weaver, W. (1949). *The Mathematical Theory of Communication*. University of Illinois Press.

vanKasteren, T. (2006). *Realtime Tempo Tracking using Kalman Filtering*. Tesis de Maestría, University of Amsterdam.

Vercoe, B. (1984). The synthetic performer in the context of live musical performance. *Proceedings of the International Computer Music Conference (ICMC)*.

Wold, E., Blum, T., Keislar, D., y Wheaton, J. (1996). Content-based classification, search and retrieval of audio. *IEEE Multimedia*, *3*, 27–36.

Zwicker, E. (1961). Subdivision of the audible frequency range into critical bands. *The Journal of the Acoustic Society of America*, *33*(2), 248–248.

Zwicker, E., y Fastl, H. (1990). *Psychoacoustics, Facts and Models*. Springer.