Ingeniería. Revista de la Universidad de Costa Rica
ISSN: 1409-2441
marcela.quiros@ucr.ac.cr
Universidad de Costa Rica
Costa Rica

Fonseca Solís, Juan M.
RECONOCIMIENTO AUTOMÁTICO DE LA ALTURA MUSICAL EN LA INTERFAZ DE UN JUEGO DE COMPUTADORA
Ingeniería. Revista de la Universidad de Costa Rica, vol. 25, núm. 1, enero-julio, 2015, pp. 21-32
Universidad de Costa Rica
Ciudad Universitaria Rodrigo Facio, Costa Rica

Ingeniería 25 (1): 21-32, ISSN: 2215-2652; 2015. San José, Costa Rica

RECONOCIMIENTO AUTOMÁTICO DE LA ALTURA MUSICAL EN LA INTERFAZ DE UN JUEGO DE COMPUTADORA

Juan M. Fonseca Solís

Resumen

El reconocimiento automático de la altura musical permite crear software capaz de identificar las notas tocadas por un instrumento musical. En este artículo se explica cómo es posible incorporar un algoritmo de reconocimiento de altura a un proyecto de software, para ello empleamos la implementación en C del algoritmo SWIPEP. El proyecto escogido es un juego de memoria en el cual el usuario escucha una secuencia de notas y las toca a la computadora usando una flauta dulce soprano. Adicionalmente se explican los conceptos básicos para entender el fenómeno acústico involucrado. El artículo va dirigido a todos aquellos estudiantes con conocimientos básicos en programación que quieran añadir procesamiento de sonido a sus proyectos.

Palabras clave: Interacción humano-computador, recuperación de información musical, procesamiento digital del sonido, SWIPEP, flauta dulce soprano.

Abstract

Automatic pitch recognition provides mechanisms for recognizing notes played by musical instruments. This paper explains how to embedd a pitch recognition algorithm in a project using the C implementation of SWIPEP. The chosen project is a memory game in which user has to listen to a sequence of notes and play them back to the computer using a soprano recorder flute. Additionally we explain the basic concepts for understanding the acoustic phenomena involved. This paper was written for all students with basic programming knowledge who want to incorporate sound processing to their projects.

Keywords: Human-computer interaction, music information retrieval, sound digital processing, SWIPEP, soprano recorder flute.

Recibido: 31 de Agosto 2014 **Aprobado:** 27 de Noviembre 2014

1. INTRODUCCIÓN

El reconocimiento automático de la altura musical provee a la computadora la capacidad de interpretar las ondas acústicas y extraer de ellas información valiosa como las notas tocadas por un instrumento musical.

Aplicaciones como Songs2see de Songquito (2014) actualmente son capaces de aprovechar ésta capacidad para lograr que el usuario aprenda a interpretar partituras de música usando un micrófono, una computadora y un instrumento propio. Esto abre enormes oportunidades en la creación de tutores automatizados de música que mejoren la experiencia de aprendizaje del estudiante.

Para el campo de la interacción humano-computador y la recuperación de información musical la estimación de altura es un gran logro, pues ha permitido encontrar nuevas

formas de usar una computadora sin depender exclusivamente del teclado y del ratón.

Autores como (ISHII: 2004), con su metáfora de las "Botellas Musicales" ya habían explorado el uso de elementos físicos para recuperar melodías en la computadora. Esta vez el sonido viene a sustituir a las botellas y se convierte en una nueva interfaz invisible y transparente para recuperar información musical. De hecho servicios como Midomi® ya permiten al usuario recuperar el nombre de una canción escuchada usando el método del tarareo.[1]

En el curso de Procesamiento de Sonido CI2813 del programa de Pregrado de la Escuela de Ciencias de la Computación e Informática, la estimación de la altura es precisamente uno de los temas principales. Por ello se estudian algoritmos como SWIPEP, Producto Armónico del Espectro de (Schroeder: 1968), Sumatoria de Subarmónicos de (Hermes: 1988), Radio de Armónicos y Subarmónicos de (Sun: 2000) y Colador Armónico de (Duifhuis, Willems, Sluyter:1982).

El objetivo principal de este documento es demostrar que se puede incorporar SWIPEP a un proyecto de software. Para ello se rescata uno de los proyectos finales del curso CI2813 que consiste en un juego que evalúa la memoria del usuario. La dinámica consiste en que el usuario memorice una secuencia de notas y las toque a la computadora usando una flauta dulce soprano.

El documento se organiza de esta manera: en la sección dos, se presenta el marco teórico con los conceptos básicos utilizados en el artículo. En la sección tres, se describen los requerimientos del juego. En la sección cuatro, se describe la metodología seguida en la selección de parámetros del algoritmo SWIPEP. En la sección cinco, se analizan los resultados obtenidos. Posteriormente, en la sección seis, se exponen mejoras que se pueden realizar a futuro y finalmente, en la sección siete, se presentan las conclusiones obtenidas del experimento.

2. MARCO TEÓRICO

En esta sección se describen los conceptos básicos utilizados en el artículo.

2.1. Altura musical y conceptos relacionados

A continuación se explican los términos básicos que se deben manejar para procesar sonido: oscilograma, frecuencia fundamental, transformada de Fourier, espectro de la señal,

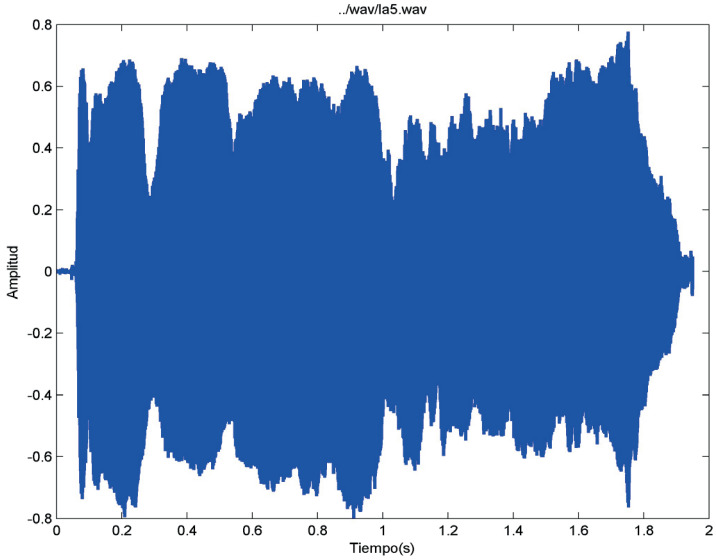

Figura 1. Oscilograma de la grabación de la nota La5 con duración aproximada de 2s tocada por una flauta dulce soprano.

tono puro, tono complejo, frecuencia de muestreo, octava y algoritmo SWIPEP.

Oscilograma

Un oscilograma es un gráfico que permite visualizar la amplitud de una señal de sonido con forme ésta avanza en el tiempo. El oscilograma muestra en el eje de las abscisas, el tiempo de la señal, y en el eje de las ordenadas, la amplitud de la onda.

La amplitud es una medida de la energía presente en la señal y puede ser expresada en términos de sonios o watts por metro cuadrado, dependiendo de si se quiere medir la intensidad o la sonoridad. Por conveniencia la amplitud suele expresarse en decibeles, los cuales no son una nueva unidad de medida sino una forma de expresar linealmente una cantidad distribuida logarítmicamente. En la Figura 1 se puede apreciar el oscilograma para la nota La5.

Período y frecuencia fundamental

Período

Una onda está formada por los valores que capta el micrófono. En presencia de sonido, el diafragma de un micrófono se expande o se contrae dependiendo de la cantidad de energía percibida del entorno. Este movimiento se debe al empuje y retroceso de las ondas sonoras, que trasladan el diafragma lejos de su posición habitual.[2] Por eso la señal escuchada oscila en un rango de valores negativos y positivos formando picos y valles.

El período de una señal se define como el tiempo necesario que debe transcurrir para que se repita el mismo patrón de picos valles. El mismo se crea solamente en señales periódicas, como la voz humana o las notas tocadas por un instrumento musical y puede observarse al realizar un acercamiento del oscilograma de

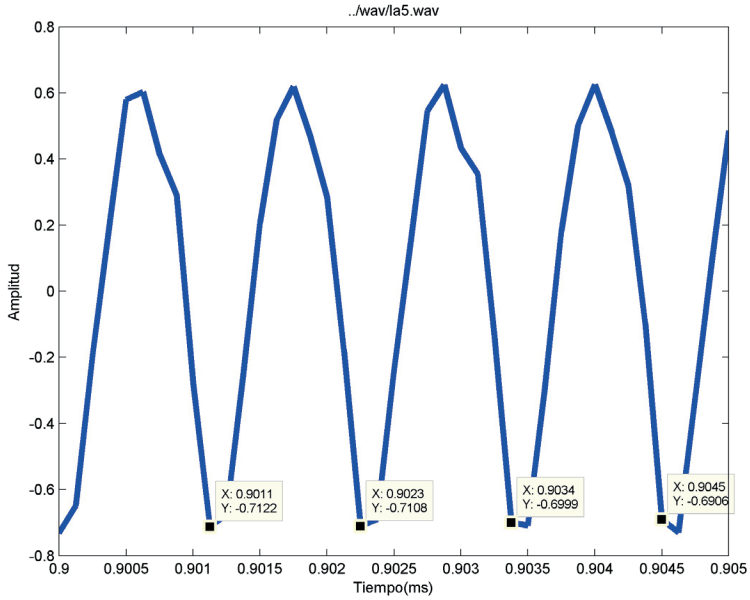

Figura 2. Acercamiento del oscilograma de la nota La5 en el intervalo 0.900 segundos a 0.905 segundos. Realizando la resta de los puntos adyacentes se puede notar que el período es cercano a 2,27 ms, lo que equivale a 880 Hz.

una nota musical. Este concepto queda mejor explicado en la Figura 2.[3]

En algunas señales el período puede distinguirse a simple vista, usando el oscilograma, pero en otras es más difícil hacerlo. Por eso se emplean métodos como la función de la autocorrelación (ecuación 1) o la función de la magnitud promedio de la diferencia, también llamada FMPD (ecuación 2).[4] El primer método encuentra el período como los picos de la gráfica, el segundo lo encuentra como los valles o depresiones.

Las funciones empleadas en ambos casos pueden apreciarse a continuación:

$$r(t) = \int_{-\infty}^{\infty} s(\tau)s(\tau - t)\mathrm{d}\tau \tag{1}$$

$$f(t) = \int_{-\infty}^{\infty} |s(\tau) - s(\tau - t)|\,\mathrm{d}\tau \tag{2}$$

Frecuencia fundamental

La frecuencia fundamental se define como el inverso del período de la señal y suele expresarse en términos de Hertz (veces por segundo) o en radianes por segundo. Este es precisamente el concepto que nos interesa comprender y que es posible estudiar usando el espectro de frecuencias.

Transformada de Fourier y el espectro de frecuencias

La transformada de Fourier es una herramienta matemática que permite expresar una señal en términos de la magnitud de la energía que cada frecuencia aporta a la onda.

La transformada es en el fondo un cambio de base que permite expresar una señal en términos de una suma de senoidales o exponenciales complejas. De la transformada de Fourier también es posible conocer los desfases de los senoidales de cada frecuencia.[5]

La fórmula de la transformada de Fourier *S(f)* para una señal continua *s(t)* se calcula utilizando la la ecuación 2.3:

$$\hat{s}(f) = \int_{-\infty}^{\infty} s(t)e^{i2\pi ft}\mathrm{d}t \tag{3}$$

Al gráfico que resulta de aplicar la transformada de Fourier a la señal y tomar su valor absoluto se le denomina espectro y es el instrumento que nos permite visualizar la frecuencia fundamental y sus armónicas.[6]

Según (Akant, Tech, Pande, Limaye: 2010) la razón por la cual se usa el espectro para estudiar la altura y no la autocorrelación o la FMPD es que cuando la señal analizada no es totalmente periódica (contiene ruido) éstas dos funciones suelen presentar varios problemas para encontrar la octava correcta del tono reconocido.

Tonos puros

Un tono puro es un tono creado usando un senoidal. El espectro del mismo se compone de la frecuencia fundamental y carece de armónicas. Un ejemplo del espectro de un tono puro, con impulsos ubicados en la fundamental, puede visualizarse en la Figura 3.[7]

Tonos complejos

Los tonos complejos son tonos que están formados por la suma de varios senoidales. Se crean sumando varios tonos puros y su espectro contiene la frecuencia fundamental y las armónicas respectivas. Un ejemplo de un tono complejo puede visualizarse en la Figura 4.

Definición de la altura musical

La altura musical o *pitch* (en inglés) es una característica del sonido que permite ordenar los tonos encontrados como notas en una escala musical. Un algoritmo de estimación de altura encuentra la frecuencia fundamental característica del tono y sus armónicas, tal que se maximice la cantidad de energía extraída del espectro.[8]

A menos que se tenga un tono puro, la frecuencia fundamental correcta no se puede

apreciar a simple vista porque no siempre corresponde al primer pico más alto e inclusive puede estar ausente.[9]

La altura musical separa los sonidos en graves y agudos, no debe confundirse con la intensidad del sonido que más bien divide los sonidos en fuertes y débiles.[11]

Frecuencia de muestreo

Cuando se realiza una grabación digital en la computadora se toman varias muestras por segundo del audio capturado por el micrófono.

La señal digital registrada siempre tiene menor calidad que la que se escucha en el mundo

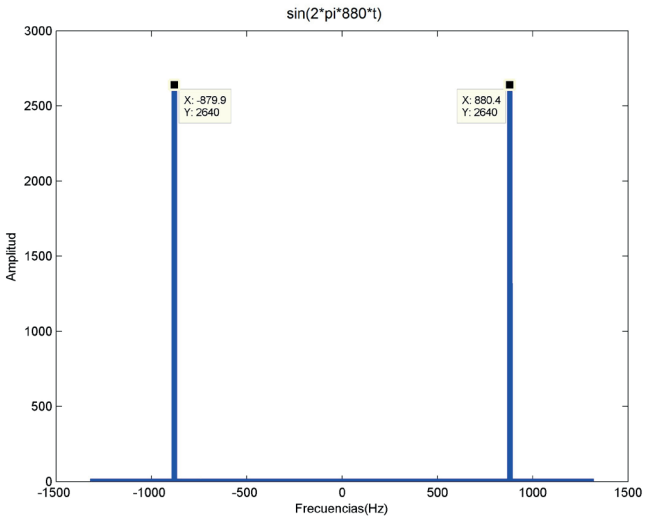

Figura 3. Espectro calculado para una nota La5 que fue sintetizada usando la ecuación sen(2π880t). Se pueden apreciar dos impulsos en ±880Hz. El espectro es simétrico respecto el eje y.

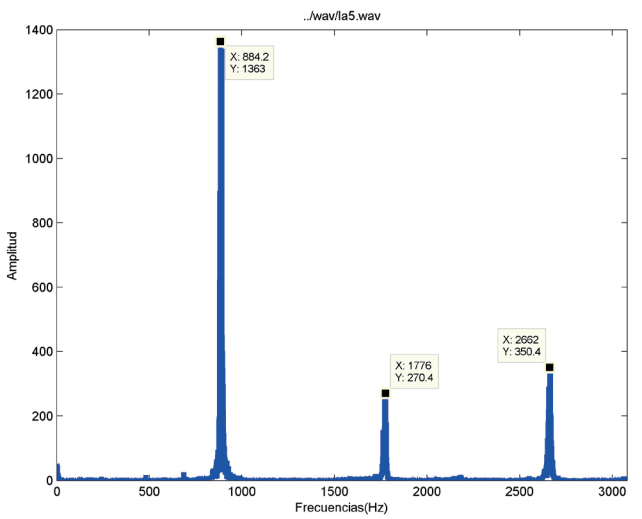

Figura 4. Espectro calculado para una grabación de la nota La5 tocada por flauta dulce soprano. La magnitud de cada armónica determina el timbre del instrumento. Hay más armónicas pero aquí se tomaron solamente las dos primeras, se consideró solo el eje x positivo.

real. Esto se debe a que se está usando un medio digital en lugar de uno analógico.

Como regla general se sabe que entre mayor sea la frecuencia de muestreo, mejor será la aproximación a la señal real. El Teorema de Nyquist sugiere que para capturar sin pérdida la información de un sonido con frecuencia máxima F, se debe muestrear a por lo menos 2F, o sea el doble de la frecuencia máxima representable. Por ejemplo si se quiere grabar una voz femenina, que ocurre en el rango de 250Hz a 4kHz, se debe muestrear a por lo menos el doble de la frecuencia máxima o sea 8kHz.

La frecuencia de muestreo también determina el rango de frecuencias visibles en el espectro. Si se realiza una grabación con una tasa de muestreo de 8000 Hz, el rango visible en el espectro será de -4000Hz a 4000 Hz.

Octava musical

Una octava musical se define como un intervalo compuesto por 12 notas (8 tonos y 8 semitonos). Los tonos son las notas do-re-mi-fa-sol-la-si, y los semitonos son los respectivos bemoles y sostenidos. Una forma de entenderlo mejor es visualizando el teclado de un piano, las teclas blancas son los tonos y las teclas negras son los semitonos. Cada nota tiene una frecuencia fundamental particular, por ejemplo la frecuencia de La4 es 440Hz y de La5 es 880Hz.

SWIPEP

SWIPEP es un algoritmo de estimación automática de la altura creado por Camacho (2007) en el lenguaje Matlab®.[11]

En términos generales, el algoritmo SWIPEP funciona creando una onda diente de sierra, a la que se le denomina el kernel, y multiplicándola por el espectro. El resultado es un espectro con energía filtrada en la fundamental y sus armónicas.[12]

Para encontrar la altura correcta el algoritmo SWIPEP crea un kernel diferente por cada frecuencia fundamental candidata. Aquella frecuencia candidata que logre maximizar la energía encontrada es elegida como la altura correcta.

El poder de SWIPEP reside en que toma en cuenta ciertas propiedades de la altura musical como la difusión de los armónicos, el decaimiento natural del espectro y el uso de armónicas primas.

Camacho (2007) ha demostrado que SWIPEP presenta mejor rendimiento que otros algoritmos importantes de estimación de altura como por ejemplo YIN (Cheveigné, Kawahara: 2002)

En la Figura 5 se muestra el análisis creado por SWIPEP para una grabación de la nota La5 tocada con flauta dulce soprano.

Invocando al algoritmo SWIPEP

Cuando se utiliza SWIPEP para analizar una señal se obtiene un arreglo de números que representan las estimaciones de altura musical cada cierto tiempo (por defecto cada 10 ms). El algoritmo no analiza cada punto de la señal porque resultaría caro computacionalmente y se sabe que la nota tocada varía poco en intervalos pequeños.

Al usar SWIPEP los sonidos que no tienen altura se interpretan convenientemente como un valor cercano a cero, lo que se pueden distinguir los momentos de silencio o de ruido en la grabación.

Para invocar el algoritmo SWIPEP es necesario definir los parámetros descritos a continuación:

- Señal a procesar, un arreglo de amplitudes (dB)
- Frecuencia de muestreo de la grabación (Hz)
- Frecuencia mínima y máxima (Hz). De este rango se eligen los candidatos de la altura.
- Saltos en el tiempo (segundos)
- Saltos en el dominio de las frecuencias ($R \in [0,1]$). El número de frecuencia a buscar por octava
- Saltos en la escala ERB.[13]
- Puntaje mínimo que debe tener un candidato para ser considerado (rango de $-\infty$ a $+\infty$)

Al ejecutar el algoritmo se obtienen los siguientes tres resultados:

- Alturas ganadoras (encontradas en cada salto de tiempo)
- Eje de los saltos en el tiempo (es el eje X de las alturas ganadoras)

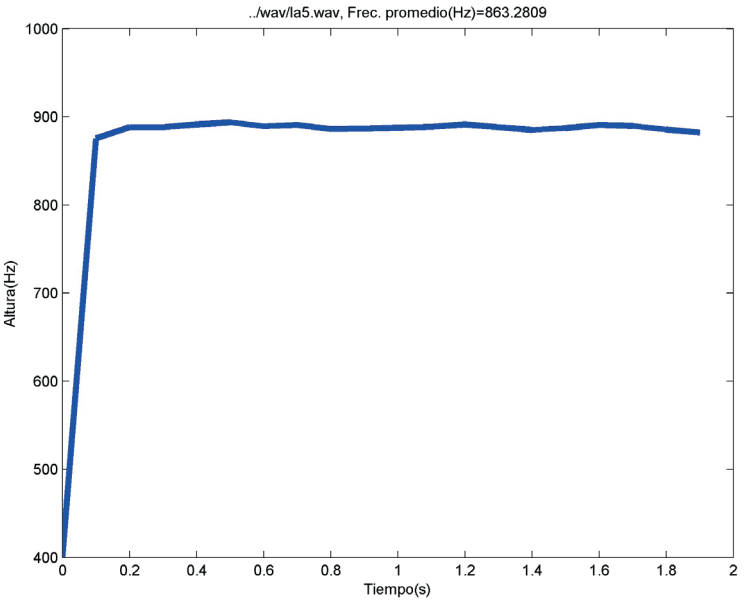

Figura 5. Análisis realizado por SWIPEP para la grabación de la nota La5 tocada por una flauta dulce soprano. El promedio de altura está en 863.28 Hz, cercano a 880 Hz que es la altura real.

- Matriz de puntajes (para cada candidato, describe el puntaje obtenido por cada altura candidata en cada salto de tiempo)

3. DESCRIPCIÓN DEL JUEGO DE MEMORIA

En esta sección se describen los requerimientos de la aplicación a desarrollar y las estructuras básicas del diseño del programa.

3.1. Enunciado de la aplicación

El proyecto elegido consiste en un juego de memoria capaz de procesar las notas que toca el jugador usando su flauta dulce soprano.

El programa creado cumple con ciertos requerimientos, los más importantes son descritos a continuación:

- El sistema debe crear secuencias aleatorias de notas musicales y debe reproducirlas al jugador
- El sistema debe indicar al jugador cuándo empezar a tocar
- El sistema debe grabar las notas que el usuario reproduzca con su flauta dulce soprano
- El sistema debe identificar la altura de las notas grabadas y luego indicar al jugador cuál fue el puntaje obtenido
- El juego debe ofrecer tres modos de dificultad: fácil, intermedio y difícil
- El modo fácil incluye solamente tonos y permite que el jugador toque las notas en octavas diferentes a las reproducidas
- El modo intermedio incluye solamente tonos y no permite que el jugador toque notas en octavas que no sean las reproducidas
- El modo difícil incluye tonos y semitonos, y no permite que el jugador toque notas en octavas que no sean las reproducidas
- El jugador sube de nivel cuando es capaz de reproducir la secuencia de notas correcta. Entonces la próxima secuencia tendrá una nota adicional

3.2. Estructura de la aplicación

Para facilitar la construcción del proyecto, aumentar la cohesión y disminuir el acoplamiento

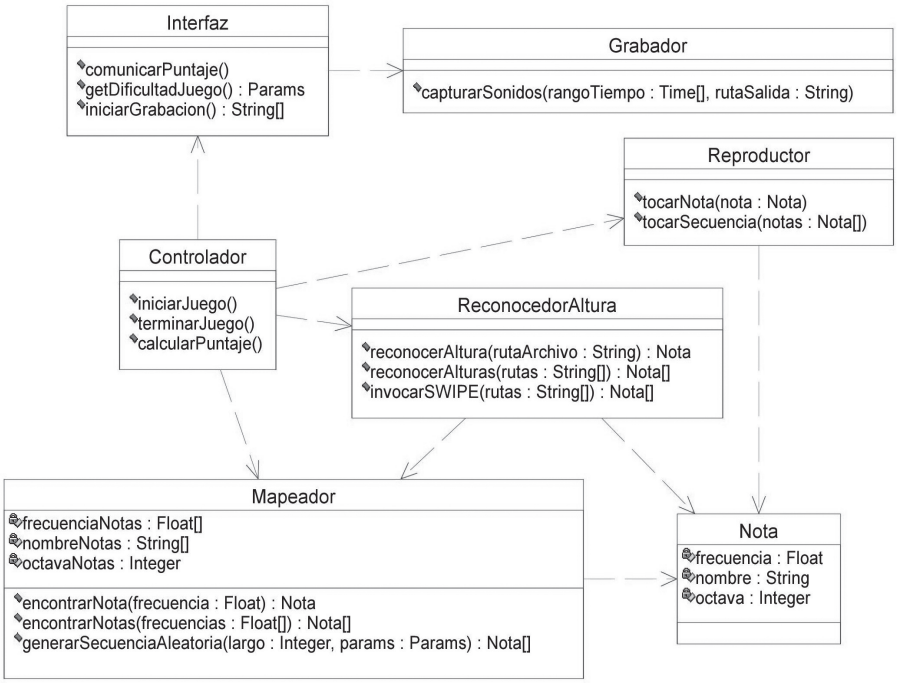

Figura 6. Descripción de la estructura de la aplicación

se empleó programación orientada a objetos y un patrón de diseño modelo-vista-controlador.

Como se puede observar en la Figura 6, el sistema se organizó de modo que la lógica del juego estuviera encapsulada en el módulo Controlador.

La capacidad de entrada y salida de datos se puso a cargo de un módulo Interfaz, que a su vez se apoya en los objetos auxiliares Reproductor y Grabador, que tienen la tarea de reproducir la secuencia de notas al usuario y grabar los sonidos que éste emita con su flauta.

El uso del algoritmo SWIPEP es administrado por un componente llamado ReconocedorAltura encargado de recibir una secuencia de Archivo de imagenes de sonido y devolver una serie de objetos tipo Nota.

El objeto Mapeador se encarga de determinar las frecuencias y octavas correspondientes a las notas. Para ello emplea una tabla con filas de la forma <frecuencia, nombre nota, octava>. El Mapeador también es el encargado de crear la secuencia de notas que se reproducirá al usuario con base en el modo de dificultad del juego.

Los objetos tipo Nota son contenedores del conocimiento del objeto Mapeador que permiten al componente Controlador saber qué calificación poner a la secuencia tocada por el jugador en todo momento, sin necesidad de realizar llamados extra. El flujo principal del juego puede observarse en la Figura 7.

4. METODOLOGÍA

En esta sección explicamos la forma de acoplar el algoritmo SWIPEP al proyecto, así como las particularidades a considerar en el reconocimiento de las notas de la flauta dulce soprano.

4.1. Implementación en C del algoritmo SWIPEP

Kyle Gorman (Gorman: 2011) publicó una implementación en C del algoritmo SWIPEP. Su código es una alternativa de código libre a la implementación original realizada por Camacho (2007) en Matlab®.

Para este proyecto se utiliza la implementación de Gorman para analizar las grabaciones obtenidas del usuario y calcular la altura musical de las notas tocadas. Al usar éste código obtuvimos varias ventajas como la disminución del tiempo de procesamiento de la altura y la independencia sobre el uso de licencias de Matlab® para correr el algoritmo.

La única desventaja encontrada es que al ser una implementación en C no es portable como Java y, adicionalmente, es necesario instalar previamente las librerías: LIBSND y la Fast Fourier Transform West (FFTW).[14]

4.2. Notas tocadas con flauta dulce soprano

El programa fue diseñado para analizar las notas ejecutadas por una flauta dulce soprano. Este instrumento se puede conseguir fácilmente, pues se emplea en casi todos los colegios y escuelas del país en los cursos de educación musical.

Según Rochester (Rochester: 2014) la tesitura de la flauta dulce soprano cuenta con 37 notas diferentes que van desde Do5 hasta Re7, o sea tres octavas y un tono extra. Eso quiere decir, según Wolfe (2014), que el rango de frecuencias que abarca va de los 523.25 Hz hasta los 2349.3 Hz.

Aunque el algoritmo SWIPEP no tiene problemas para analizar un rango mayor, nos limitamos a este intervalo porque sólo nos interesa analizar los sonidos producidos por el instrumento escogido.

4.3. Frecuencia de muestreo

La frecuencia de muestreo escogida fue 8000 Hz ya que se quiso incluir por lo menos la tercera armónica de la frecuencia máxima reproducible. Esto significa que la tercera armónica de Re7 ocurre en 7047.9 Hz, entonces la frecuencia de muestreo más cercana por arriba es 8000 Hz.

4.4. Invocando al algoritmo SWIPE

Para que la implementación en C de SWIPEP analizara la señal de audio recibida del usuario se escogieron los parámetros descritos a continuación:

- Señal a procesar (obtenida de las grabaciones)
- Frecuencia de muestreo de la grabación (8000 Hz a 16 bits)
- Frecuencia mínima y máxima (500 Hz a 2500 Hz)
- Saltos en el tiempo (100 ms)
- Saltos en el dominio de las frecuencias (1/48, 48 frecuencias por octava)
- Saltos en la escala ERB (0.1 Erb)
- Puntaje mínimo ($-\infty$)

5. RESULTADOS

A continuación se analizan los resultados obtenidos luego de usar la implementación de Kyle Gorman (2011) para el algoritmo SWIPEP con los parámetros escogidos en la sección anterior.

5.1. Reconocimiento de las notas tocadas con flauta dulce soprano

Debido a la robustez del algoritmo SWIPEP se pudo reconocer exitosamente la altura musical de todas las notas de la flauta dulce soprano, tocadas por el jugador en los experimentos realizados. Se logró adicionalmente en un tiempo menor que si se hubiera usado la implementación en Matlab®.

Con el equipo empleado en el experimento, se alcanzó medir que la implementación en C permite analizar grabaciones de más de 100 segundos. Contrario a la implementación en Matlab® que lanzaba un error de *Out of Memory* para grabaciones con esta duración.[15]

En la Figura 8 se puede ver el rendimiento de la implementación de Gorman comparado con la implementación en Matlab®. Se puede apreciar que el primero se ejecuta con un orden de duración casi lineal, mientras que el segundo tiene un orden de duración logarítmico con constantes de proporcionalidad muy altas.

Para que los tonos fueran reconocidos con mayor facilidad el algoritmo SWIPEP necesitó que la duración de las notas tocadas fuera de

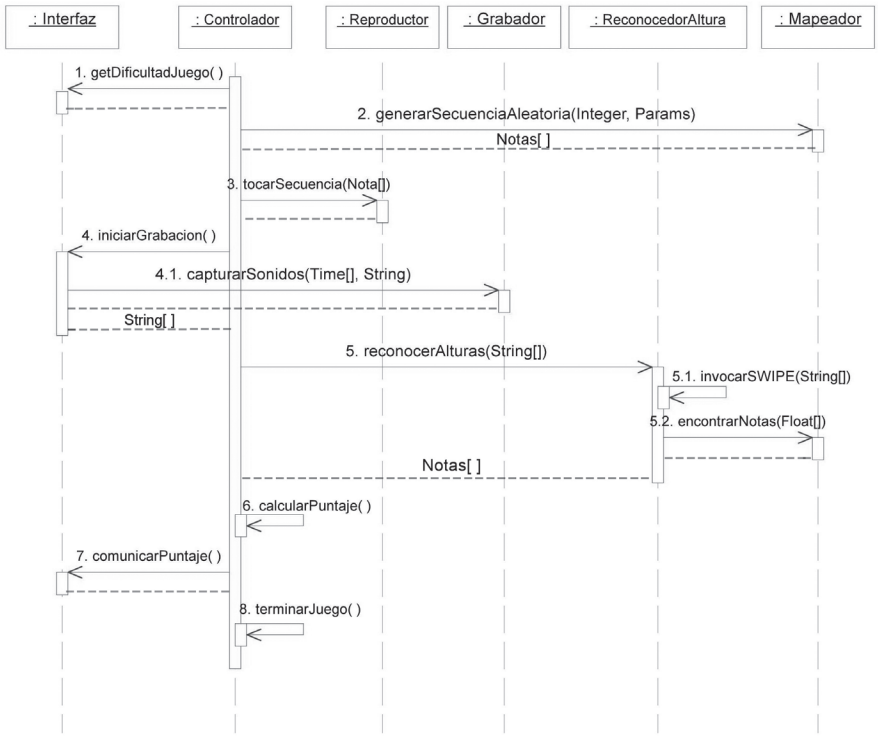

Figura 7. Diagrama de secuencia para explicar el flujo principal del programa.

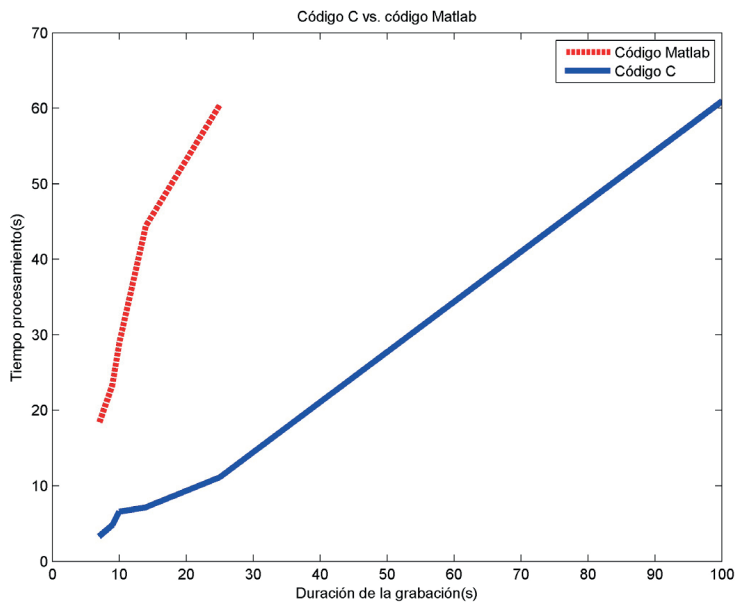

Figura 8. Comparación entre el rendimiento de la implementación original de SWIPEP y la implementación de Gorman. Se puede apreciar que la implementación en C es mejor en términos de tiempo requerido.

al menos 0.8 segundos, y que el tiempo entre grabaciones fuera de 10 ms.[16]

6. TRABAJO FUTURO

Los parámetros del algoritmo SWIPEP fueron ajustados para funcionar solo con la flauta dulce soprano, pero podrían ser ajustados fácilmente para reconocer sonidos de otros instrumentos o bien tonos de la voz humana.

El componente Mapeador podría mejorarse para que en lugar de usar una tabla con tuplas <frecuencia, nombre nota, octava>, pueda usar una fórmula que relacione las frecuencias de las notas con el nombre y la octava que les corresponde.

Al diseñar el sistema con un patrón modelo-vista-controlador se pudo encapsular la lógica del reconocimiento de altura en un solo componente, por lo que a futuro se podrían agregar fácilmente otros algoritmos de estimación de altura como YIN o autocorrelación.

En el futuro también podría ser necesario buscar algoritmos alternativos a SWIPEP para identificar sonidos armónicos, pues actualmente no se pueden procesar acordes musicales producidos por instrumentos más complejos de cuerdas simples y percutidas.

7. CONCLUSIONES

Como explicamos a lo largo de este documento, la utilización de algoritmos de estimación de altura proveen al software la capacidad de procesar el sonido producido por los instrumentos musicales.

Con la aplicación de SWIPEP en el caso de estudio, se logró que el proyecto contara con la capacidad de reconocer los sonidos de flauta dulce soprano tocados por el jugador.

Los parámetros del algoritmo fueron ajustados para adaptarse a la tesitura de éste instrumento y se mostró que para hacerlo, es necesario tener un conocimiento básico en la teoría de procesamiento de señales.

Se descubrió adicionalmente que la implementación de Kyle Gorman (2011) es más rápida en términos de tiempo comparada con su análoga implementada en Matlab®.

A criterio del autor, los algoritmos de estimación de altura son un excelente campo de estudio para contribuir al desarrollo de las aplicaciones relacionadas con la interacción humano-computador y la recuperación de la información musical, y es necesario más inversión e investigación para alcanzar su verdadero potencial e inclusión en la vida cotidiana de los usuarios.

8. CITAS Y NOTAS

1. Se refiere con “transparente” a que la interfaz es familiar para el usuario y por eso sabe inmediatamente cómo usarla. Con “invisible” nos referimos a que no se puede palpar.
2. En el caso particular de un micrófono de diafragma.
3. Denominados movimientos de rarefacción y compresión.
4. Un ejemplo de una señal no periódica es el ruido blanco.
5. Average Magnitude Difference Function o AMDF, por sus siglas en inglés.
6. Cambiando los desfases se pueden agregar retrasos en la señal.
7. Una armónica se define como un múltiplo de la frecuencia fundamental. Por ejemplo para una fundamental en 880Hz, las armónicas serían: 1760 Hz, 2640 Hz, 3520 Hz, y así sucesivamente.
8. Un impulso es una función capaz de extraer el valor de otra función cuando su propio argumento se hace cero.
9. Aquí se refiere solamente al reconocimiento de sonidos melódicos (mono-tonos) y no a sonidos armónicos (poli-tonos).
10. En cuyo caso estarían presentes solo sus armónicas.
11. La intensidad es lo que se conoce como el volumen, se debilita conforme aumenta la distancia y tiene que ver con otro concepto llamado poder sonoro.
12. El código de SWIPEP en Matlab® puede encontrarse en el sitio: http://www.cise.ufl.edu/~acamacho/publications/swipep.m. Matlab® es propiedad de MathWorks.

13. Su espectro es un tren de impulsos que decae inversamente proporcional a la frecuencia (-6 db/octava).
14. La escala ERB (equivalent rectangular bandwidth) distribuye las frecuencias del espectro logarítmicamente, de manera como una persona las percibiría. Para ello se utiliza la distribución de frecuencias: ERB (f)= 6.4 $\log_2$ • (1+ (f/230))
15. Para más información de cómo usar las librerías leer las referencias de De Castro (1999) y Frigo, Johnson (1997)
16. Descripción del equipo usado: Intel(R) Celeron(R) CPU B820 @ 1.70GHz, 2 GB of RAM, Microsoft Windows 7 Starter, Netbeans IDE 7.2 y Matlab® 6.
17. El tiempo de silencio entre notas es opcional, se pueden realizar varias grabaciones en lugar de una sola para evitar depender de los silencios.

BIBLIOGRAFÍA

Akant, K., Tech, M., Pande, R. y Limaye, S. (2010). AccurateMonophonic Pitch Tracking Algorithmfor QBH and MicrotoneResearch. ThePacificJournal of Science and Technology, 11 (2). Disponible en http://www.akamaiuniversity.us/PJST11_2_342.pdf

Camacho, A. (2007). Swipe: A Sawtooth Waveform Inspired Pitch Estimatorfor Speech and Music. (Tesis doctoral, Universidad de Florida). Disponible en http://www.cise.ufl.edu/~acamacho/publications/dissertation.pdf

Castro, E. (1999). LibsndfileAppStore. Disponible en http://www.mega-nerd.com/libsndfile/

Cheveigné, A. y Kawahara, H. (2002). YIN, a fundamental frequencyestimatorforspeech and music. The Journal of the Acoustical Society of America. Disponible en http://audition.ens.fr/adc/pdf/2002_JASA_YIN.pdf

Duifhuis, H., Willems, L. y Sluyter, R. (1982). Measurement of pitch in speech: Animplementation of Goldstein'stheory of pitch perception. AcousticalSociety of America 71 (6), pp 1568-1580.

Frigo, M. y Johnson, S. (1997). Fastest Fourier Transform in the West. Página web. Disponible en http://www.fftw.org/

Gorman, K. (2011). SWIPE0 pitch extractor software. Center forspokenlanguageunderstanding, OregonHealth and ScienceUniversity. Disponible en http://www.ling.upenn.edu/~kgorman/c/swipe/

Hermes, D. (1988). Measurement of pitch bysubharmonicsummation. Journal of theAcousticalSociety of America 83, pp 257-264

Hiroshi Ishii. (2004). Bottles: A Transparent Interface as a Tribute to Mark Weiser. IEICE transactionsoninformation and systems, 87 (6), pp. 1299-1311. Recuperado el 22 de agosto de 2014 en http://ci.nii.ac.jp/naid/110003214002/en

Rochester Chapter of the American Recorder-Society. (2014). RecorderFrequencyRanges. Disponible en http://www.rochars.org/recorder_freqency_ranges

Schroeder, M. (1968). PeriodHistogram and ProductSpectrum. New Methodsfor Fundamental-FrequencyMeasurement. TheJournal of theAcousticalSociety of America, 43 (4), pp. 829.

Songquito UG 2014 (haftungsbeschrnkt). (20015). Songs2See - Learn to Play by Playing. Disponible en http://www.songs2see.com/

Sun, X. (2000). A Pitch DeterminationAlgorithm-BasedonSubharmonic-to-Harmonic Ratio. The 6th International Conference of Spoken-LanguageProcessing. (pp. 676-679)

Wolfe, J. (2014). Note names, MIDI numbers and frequencies. University New South Wales, MusicAcousticsLaboratory. Disponible en http://www.phys.unsw.edu.au/jw/notes.html

SOBRE EL AUTOR

Juan M. Fonseca Solís: Egresado de la Escuela de las Ciencias de Computación e Informática de la Universidad de Costa Rica. Correo electrónico: juan.fonsecasolis@ucr.ac.cr .