HEPPIE: un algoritmo simple para la estimación de tono, en un amplio espectro, con exactitud y costo computacional ajustables

Gerardo A. Laguna-Sánchez, Diana V. Ortiz-Martínez y Victor M. Pichardo-Infante
Universidad Autónoma Metropolitana
CDMX, México
g.laguna@correo.ler.uam.mx

***Resumen*—El algoritmo denominado HEPPIE (por *HElpful and Practical PItch Estimation*) permite la estimación de la frecuencia fundamental de señales sonoras, con un mínimo de parámetros y operando en el espectro audible que abarca de los 50 Hz a los 5 kHz, con una resolución ajustable que determina el costo computacional. Si algoritmo se configura para operar con una resolución de 1 Hz, reduce significativamente su costo computacional y se convierte en una alternativa tan competitiva como el popular algoritmo YIN, pero con una mayor robustez ante el ruido AWGN. El algoritmo ha sido diseñado para para su implementación en sistemas con recursos limitados, como lo son los sistemas embebidos, el Internet de las Cosas. El algoritmo opera a partir de principios simples bien conocidos y aspira a ser una alternativa de fácil implementación y operación: en esencia, se calcula el periodograma y se procesa mediante algunos filtros FIR simples. Se comparó el desempeño del algoritmo, en cuanto a exactitud y costo computacional, contra diversos algoritmos bien conocidos de estimación de tono, incluyendo algunos muy recientes en el estado del arte, y la posición obtenida resultó ser muy competitiva.**

***Palabras clave*—Frecuencia fundamental, Estimación de tono, Detección de tono, Notas musicales**

I. Introducción

Las notas sonoras producida por instrumentos musicales se conforman por la superposición de un cierto número de frecuencias armónicas correspondientes a una frecuencia fundamental característica. Las frecuencias armónicas presentan valores que son múltiplos enteros de su frecuencia fundamental. Dado que la intensidad de cada una de las frecuencias armónicas de una señal depende de las condiciones y características físicas del instrumento que la produce, el sonido resultante se distingue por un timbre específico. En este contexto, la estimación de la frecuencia fundamental de una señal audible resulta de gran interés, ya sea para la afinación de instrumentos musicales o para el procesamiento y análisis del audio.

El algoritmo HEPPIE (por *HElpful and Practical PItch Estimation*) fue presentado recientemente [1] como una alternativa de fácil comprensión, implementación y uso, con un costo computacional que depende directamente de la resolución requerida. A mayor resolución, mayores recursos computacionales son requeridos y mayor el tiempo consumido; a menor resolución, menores son los recursos computacionales requeridos y menor el tiempo consumido. El algoritmo fue implementado en Matlab y en este trabajo se presentan los resultados de la comparación de desempeño contra 8 algoritmos no supervisados de referencia que, o bien, se encuentran disponibles dentro del módulo de procesamiento de audio de Matlab [2], o sus códigos Matlab se encuentran disponible en repositorios de código abierto.

Nuestra exposición considera las siguientes secciones: La sección II, refiere muy brevemente el trabajo relacionado y enlista los algoritmos empleados en la comparación de desempeño. La sección III, explica el algoritmo HEPPIE. La sección IV, presenta la metodología para la comparación de desempeño y discute los resultados obtenidos. Finalmente, en la sección V, compartimos nuestras conclusiones.

II. Trabajo relacionado y algoritmos empleados en la comparación de desempeño

En general, dentro de la literatura especializada, la estimación de la frecuencia fundamental se asocia con el proceso conocido como estimación/detección de tono (*pitch*). Una primera clasificación de los algoritmos de estimación/detección de tono es la que parte del dominio del procesamiento de la señal, a saber, 1) el dominio del tiempo, 2) el dominio de la frecuencia o 3) una combinación de ambos dominios, como en [3]. Otro enfoque, más amplio, es el que parte de la naturaleza supervisada o no supervisada del algoritmo, es decir, si el algoritmo requiere o no de un entrenamiento previo con datos de ejemplo, como en [4]. Finalmente, algunos autores, por ejemplo [5], también hacen énfasis en el hecho de que existen propuestas paramétricas, donde el objetivo es ajustar los parámetros del modelo hasta que éste reproduzca al comportamiento observado con un error mínimo. Las propuestas paramétricas llegan a ser significativamente más exactas, pero también son computacionalmente muy costosas. En este contexto,

podemos decir que algoritmo HEPPIE es una técnica no paramétrica, no supervisada y que opera en el dominio de la frecuencia.

Para comparar el desempeño del algoritmo HEPPIE, en principio, se contemplaron los 13 algoritmos de referencia que aparecen en la tabla I, incluyendo tres algoritmos rápidos bien conocidos: el algoritmo *Normalized Correlation Function* (NCF) [6], el algoritmo *Cepstrum* (CEP) [7] y el algoritmo YIN [8]. Esta lista tambien incluye un algoritmo paramétrico del estado del arte, el conocido como *Fast Nonlinear Least Squares* (FNLS) [4], [5] que teóricamente garantiza el mínimo error en entornos adicionados con ruido blanco. La lista tambien incluye propuestas de desempeño intermedio, incluyendo los algoritmos *Robust Algorithm for Pitch Tracking* (RAPT) [9] y YAAPT (*Yet Another Algorithm for Pitch Tracking*) [10], además de 7 algoritmos emparentados con el método *Harmonic Summation* (HS), con resultados que se aproximan al criterio *nonlinear least squares* pero con un menor costo computacional. Los algoritmos afines al tipo HS incluyen al *Log-Harmonic Summation* (LHS) [11], el algoritmo *Summation of Residual Harmonics* (SRH) [12], el algoritmo *Subharmonic-to-Harmonic Ratio* (SHR) [13], el algoritmo *Pitch Estimation Filter* (PEFAC/PEF) [14], el algoritmo *Sawtooth Waveform Inspired Pitch Estimator* (SWIPE) [15] y una variante del algoritmo SRH (SRHv) [16].

Sin embargo, debido a que no todos los algoritmos de la tabla I pudieron ajustarse para operar, con un error de estimación aceptable, en intervalo espectral de interés, que abarca de 50 Hz a los 5kHz, la lista final de algoritmos de referencia quedó constituida por el algoritmo FNLS, que provee del mínimo error teórico aunque con un costo computacional significativo; dos algoritmos que pueden considerarse, estrictamente, del tipo HS, los algoritmos LHS y SRH; dos algoritmos que, en sentido amplio, pueden considerarse como del tipo HS [16], los algoritmos PEF y SWIPE; y tres algoritmos soportados por autocorrelaciones en el dominio del tiempo, con tiempos de ejecución mínimos, los algoritmos NCF, RAPT y el muy popular algoritmo YIN. Ahora, debemos aclarar que el hecho de que un algoritmo se haya excluido de nuestra lista final no necesariamente implica que éste no se pueda desarrollar, con base en sus principios de operación, para funcionar bien en el intervalo de las frecuencias de interés, simplemente significa que los autores no encontraron la forma de configurarlo, con una misma configuración de sus parámetros, para que operara con un error aceptable en el referido intervalo del espectro.

Debido a que una de las métricas de desempeño es la exactitud en la estimación del valor de la frecuencia fundamental, se empleó una base de datos con señales sintéticas construidas con componentes espectrales de valores específicos, a partir de los patrones espectrales de los tonos de algunos instrumentos musicales reales disponibles dentro de la base de datos TinySOL [24]. La selección de los instrumentos y tonos de referencia se realizó a partir del requerimiento de contar con las frecuencias de los tonos, sin incluir semitonos, de notas musicales que abarcaran el mayor intervalo posible dentro de la escala temperada. En particular, el conjunto de señales sintéticas comienza en la frecuencia de la nota A1 (55Hz) y termina en la frecuencia de la nota C8 (4186Hz), resultando en un total de 45 patrones espectrales representativos, incluyendo las notas producidas por el acordeón, la tuba bajo, el clarinete, el violín o la viola.

III. Operación del algoritmo HEPPIE

La idea básica del algoritmo HEPPIE consiste en obtener el periodograma de la señal sonora, para proceder con el procesamiento del perfil espectral en potencia y determinar los picos espectrales más significativos, correspondientes a las frecuencias de los armónicos, y concluir con la determinación de la distancia mínima entre estos picos espectrales que, en principio, correspondería con la frecuencia fundamental.

El sustento teórico para partir del periodograma es bien conocido, ya que corresponde al cálculo de la autocorrelación en el dominio de la frecuencia, lo que implica la minimización de los efectos del ruido gaussiano blanco y aditivo (AWGN, por sus siglas en inglés) [25]. En resumen, sea la autocorrelación de una señal muestreada, representada por la secuencia $x(n)$,

$$r(m) = E[x^*(n)x(n+m)] \tag{1}$$

con $E[]$ el operador de valor esperado y x^* el conjugado de x. Para secuencias finitas de tamaño N, un estimador sesgado de $r(m)$ sería:

$$\widetilde{r}(m) = \frac{1}{N}\sum_{n=0}^{N-1-m} x^*(n)x(n+m) \tag{2}$$

para $0 \leq m < N$. El periodograma resulta de aplicar la transformada de Fourier discreta (DFT, por sus siglas en inglés) a 2, lo que resulta en

$$S(k) = \frac{1}{N}X^*(k)X(k) = \frac{1}{N}|X(k)|^2 \tag{3}$$

donde $X(k)$ es la DFT de $x(n)$ y se puede calcular, con un costo computacional mínimo, mediante el empleo del algoritmo de la transformada rápida de Fourier (FFT, por sus siglas en inglés).

La codificación del algoritmo propuesto resultó en la función $heppie()$, que devuelve la estimación de la frecuencia fundamental, $f0$, a partir de cinco parámetros: la secuencia de entrada, $s(n)$; la frecuencia de muestreo Fs, en Hz; el tamaño de la ventana de procesamiento $wsize$, en segundos; el paso para cada estimación hop, en segundos; y la mínima resolución requerida $mres$, en Hz. En el diagrama a bloques de la figura 1, se puede

TABLA I
Algoritmos de referencia

Acrónimo	Algoritmo	Dominio	Repositorio
CEP	Cepstrum	Frecuencia	[2]
FNLS	Fast Nonlinear Least Squares (modelo paramétrico)	Frecuencia	[17]
LHS	Log-Harmonic Summation	Frecuencia	[2]
NCF	Normalized Correlation Function	Tiempo	[2]
PEF	Pitch Estimation Filter without Amplitude Compression	Frecuencia	[2]
PEFAC	Pitch Estimation Filter with Amplitude Compression	Frecuencia	[18]
RAPT	Robust Algorithm for Pitch Tracking	Tiempo	[18]
SHR	Subharmonic-to-Harmonic Ratio	Frecuencia	[19]
SRH	Summation of Residual Harmonics	Frecuencia	[2]
SRHv	SRH Variant	Frecuencia	[20]
SWIPE	Sawtooth Waveform Inspired Pitch Estimator	Frecuencia	[21]
YAAPT	Yet Another Algorithm for Pitch Tracking	Híbrido	[22]
YIN	YIN	Tiempo	[23]

apreciar que lo primero que se hace es calcular el periodograma. Aunque el periodograma se calcula a partir de la información dentro de la ventana de procesamiento en turno, el número de muestras de la secuencia sobre la que realmente se calcula el periodograma se determina automáticamente, al interior de la función $heppie()$, y depende de la frecuencia de muestreo y de la mínima resolución requerida. De hecho, de ser necesario, se recurre a la técnica de *zero padding* para alcanzar la resolución requerida por el usuario. Específicamente, el tamaño de la secuencia sobre la que se debe calcular el periodograma está dada por

$$N = \frac{Fs}{mres} \tag{4}$$

Después del cálculo del periodograma, a fin de eliminar las irregularidades de los picos significativos en el perfil espectral, en ciertas circunstancias conviene realizar un procesamiento mínimo. En estos casos se emplea un filtro FIR pasa bajos (LP) de 17 coeficientes, con una frecuencia de corte, correspondiente a 0.1 de la frecuencia de Nyquist, a fin de suavizar el perfil de los picos de las componentes armónicas. A continuación del cálculo del periodograma, inicia la etapa donde se determinan cuáles son picos significativos y cuáles no. Para ello, el flujo del procesamiento continua con un filtro FIR diferenciador, es decir un filtro con coeficientes $b = [1 - 1]$, para contar con una estimación de la derivada del perfil espectral y poder detectar los picos significativos mediante la ubicación de los puntos con pendiente cero y, más específicamente, de los tránsitos de pendiente positiva a pendiente negativa, pasando por cero.

Así, la salida del filtro diferenciador permite, por una parte, estimar la longitud de la ondoleta que caracteriza el cambio de pendiente, de valor positivo a valor negativo, para el caso del pico de mayor potencia en el perfil espectral. Esta ondoleta típicamente tiene la forma aproximada de un ciclo senoidal y se pueden detectar fácilmente mediante una correlación a través de un *matched filter* cuya respuesta al impulso, h, es precisamente la ondoleta de derivación caracterizada pero rotada sobre su eje vertical. Para esto, se determina la longitud de la ondoleta de derivación y la salida del filtro diferenciador se alimenta al *matched filter*. Entonces, a la salida del *matched filter*, se obtiene una secuencia con picos de correlación que señalan la posición de las componentes espectrales correspondientes a las frecuencias armónicas.

A continuación del *matched filter*, aparece el bloque que detecta los picos significativos, es decir aquellos picos de correlación que superan el umbral correspondiente al 0.1 de la magnitud del máximo valor de correlación. Finalmente, en el último bloque funcional, se realiza la estimación de la frecuencia fundamental, para lo que se recurre al criterio de la mínima distancia entre los picos de correlación, incluyendo, por defecto, a un pico de correlación en la frecuencia cero. Para conocer los detalles específicos de la implementación y operación práctica del algoritmo, en la función $heppie()$, el lector puede acceder al código fuente en Matlab, que se encuentra disponible en [26].

IV. Metodología y métricas para la comparación de desempeño

Durante la comparación de desempeño, se procuró emplear una misma configuración para todas las pruebas y para todo el intervalo [50Hz-5kHz]. Aunque todos los algoritmos probados permiten configurar del intervalo de frecuencias, no todos soportan los valores límite del intervalo de interés. Así mismo, la mayoría de los algoritmos probados, excepto el FNLS, pudieron configurarse para trabajar con ventanas de procesamiento y pasos (*hops*) de 0.1 segundos. En el caso la función $heppie()$, aunque no se requiere especificar el intervalo de frecuencias, si es necesario especificar la resolución de la estimación. A fin de comparar el desempeño del algoritmo HEPPIE en el escenario que le demanda mayor costo computacional, se le configuró la resolución requerida en 0.1Hz.

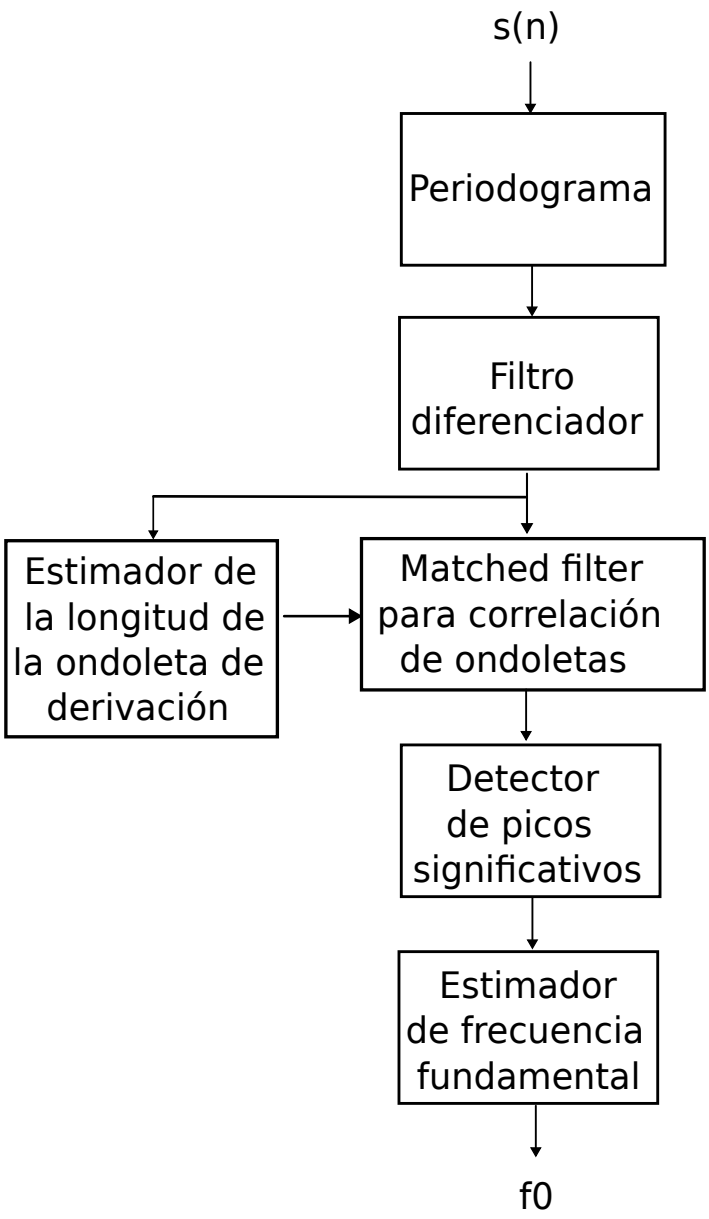

Fig. 1. Diagrama a bloques del algoritmo HEPPIE para estimación de tono.

Para comparar cuantitativamente el desempeño de todos los algoritmos, nos hemos concentrado en dos métricas: 1) el tiempo consumido, como métrica del costo computacional, y 2) la media del error absoluto, como métrica de la exactitud de las estimaciones. Específicamente, como métrica del tiempo consumido se empleó la siguiente expresión:

$$C = \frac{T}{N_{f0} \times wsize} \tag{5}$$

con T el tiempo de ejecución de la corrida, en segundos; N_{f0} el número de estimaciones realizadas y $wsize$ la duración de la ventana de procesamiento, en segundos. El valor de C es adimensional y nos indica el tiempo consumido, por cada estimación, como una fracción de la duración de la ventana de procesamiento. El valor de C es una medida relativa del consumo de tiempo y, por lo tanto, puede expresarse como un porcentaje. El registro de la métrica C obtenidas por cada algoritmo probado, ejecutándose en una computadora soportada por un procesador Intel Core i5 @1.6GHz, se presenta en la tabla II.

TABLA II
Tiempos de ejecución relativos

Algoritmo	C [%]
FNLS	1868.7
LHS	0.68
NCF	0.17
PEF	0.70
RAPT	1.81
SRH	2.16
SWIPE	15.45
YIN	0.56
HEPPIE	9.56

Por otra parte, la media del error absoluto (MAE, por sus siglas en inglés), en Hz, no es más que el promedio del valor absoluto de las diferencias entre los valores estimados, $f0_i$, y el valor real de la frecuencia fundamental, f_t, ambas en Hz:

$$MAE = \frac{1}{N_{f0}} \sum_{i=1}^{N_{f0}} |f0_i - f_t| \tag{6}$$

con N_{f0} el número de las estimaciones procesadas.

Respecto de las señales de entrada para los algoritmos, se empleó una base de datos con señales sintéticas que, específicamente, incluye frecuencias fundamentales puras y notas con hasta 11 armónicos. A su vez, cada una de estas señales, puras o harmónicas, se proporcionan sin adición de ruido o con ruido AWGN, ya sea con SNR=10 o SNR=1, resultando en 6 colecciones de señales, cada una con 45 señales, tantas como frecuencias, todas con duración de 1 segundo y muestreadas a 16ksps. En la tabla III, se resume la información relevante de los 6 tipos de señales contenidas en la base de datos.

TABLA III
Colecciones con señales sintéticas

Arreglo	Descripción
sol_pure_16ksps	señales de frecuencias puras
sol_p10snr_16ksps	señales de frecuencias puras y AWGN con SNR=10
sol_p1snr_16ksps	señales de frecuencias puras y AWGN con SNR=1
sol_harm_16ksps	señales con hasta 11 harmónicos
sol_h10snr_16ksps	señales con hasta 11 harmónicos y AWGN con SNR=10
sol_h1snr_16ksps	señales con hasta 11 harmónicos y AWGN con SNR=1
list_freq	lista de frecuencias

En particular, a fin de calcular la MAE, se emplearon las siguientes colecciones de señales:

- sol_pure_16ksps, denotada con la etiqueta *pure*.
- sol_harm_16ksps, denotada con la etiqueta *harm*.
- sol_h10snr_16ksps, denotada con la etiqueta *h10snr*.
- sol_h1snr_16ksps, denotada con la etiqueta *h1snr*.

Dada la configuración de los algoritmos, el cálculo de la MAE consideró un mínimo de 9×45 estimaciones de tono para cada uno de los tipos de señales empleadas. El registro de las MAE obtenidas, con su correspondiente desviación estándar (D.E.), y el valor MAE acumulado, se presenta en la tabla IV.

A partir los resultados de la tabla II, respecto de los tiempos de ejecución y, por ende, en cuanto al costo computacional, podemos ubicar el desempeño del algoritmo HEPPIE en una posición media, aunque cercana a los mejores tiempos de ejecución, sobre todo, tomando en cuenta que el algoritmo se operó con una la resolución de 0.1 Hz, lo que le implica mayor costo computacional. Si las pruebas se repiten con una resolución de 10 Hz, se puede confirmar que el algoritmo HEPPIE logra alcanzar un tiempo de ejecución en el orden de magnitud del registrado por algoritmo YIN.

Respecto de la exactitud de los algoritmos, a partir de los resultados registrados en la tabla IV, es importante mencionar que los algoritmos LHS, PEF y SRH no pudieron configurarse para abarcar la totalidad del intervalo, ya que sus límites superiores quedaron acotados, debido a la frecuencia de muestreo empleada (16ksps), en 3198Hz, 3999Hz y 3198Hz, respectivamente, por lo que no es de sorprender el relativamente pobre desempeño obtenido. De estos algoritmos, PEF fue el menos afectado por la restricción en su límite superior (3999Hz), ya que este valor es muy cercano a la frecuencia fundamental sintetizada más alta, correspondiente a 4186Hz. Por lo demás, cabe resaltar el excelente desempeño del algoritmo FNLS, ya que teóricamente es el algoritmo con el menor error, incluso en presencia de ruido AWGN. Se puede observar que el registro MAE para el algoritmo FNLS ronda el valor de 0.4Hz, para las señales sin ruido o con poco ruido (SNR=10), y el valor de 5Hz para una señal con ruido significativo (SNR=1). Otro caso digno de mención es el desempeño del algoritmo YIN que, para las señales sin ruido, o con poco ruido (SNR=10), su registro MAE ronda el valor de 2Hz. Desafortunadamente, en presencia de ruido significativo (SNR=1) su desempeño cae notoriamente hasta alcanzar niveles de error del orden de 100Hz.

Respecto del algoritmo HEPPIE, queda clara su posición competitiva ya que muestra un registro de MAE con un orden de magnitud que ronda los 10Hz, para la mayoría de las señales, sean sin ruido o ruidosas. Esto muestra la robustez del algoritmo propuesto ante el ruido. Adicionalmente, es de resaltar que, en el caso de las señales con frecuencias fundamentales puras y sin ruido, el valor MAE para el algoritmo HEPPIE es del mismo orden de magnitud que el algoritmo mejor posicionado, es decir el FNLS, que es un algoritmo del último estado del arte.

Para resumir en forma gráfica los resultados obtenidos, en la figura 2, se presenta una conveniente figura de mérito que permite apreciar el desempeño global para cada uno de los algoritmos probados, al considerar las dos métricas empleadas. Son notorias las posiciones de los algoritmos FNLS y NCF. El algoritmo FNLS aparece con el menor error acumulado, aunque con el mayor tiempo consumido. Por el contrario, el algoritmo NCF se posiciona con el menor tiempo consumido, aunque con el mayor error acumulado. Los demás algoritmos también quedan claramente posicionados, según su desempeño en cada una de las dos métricas, pero es evidente que la bondad de un algoritmo mejora cuando su posición relativa se aproxima al cruce de los ejes en la figura 2. En este sentido, es clara la ventaja relativa de dos algoritmos: el algoritmo YIN y el algoritmo HEPPIE, con la única diferencia de que el segundo está más cercana a la diagonal imaginaria que pasa por el origen y divide a la figura de mérito por la mitad. Esta mayor proximidad relativa del algoritmo HEPPIE a la referida diagonal, en comparación con el algoritmo YIN, significa que, aunque ambos algoritmos tienen posiciones globales similares, el algoritmo HEPPIE guarda un mejor equilibrio entre exactitud y costo computacional.

Para facilitar la reproducción los resultados reportados en este trabajo, en el repositorio [26] se deja, a disposición de los interesados, el material necesario, tanto el código Matlab de la función $heppie()$ como la base de datos con la colección de señales sintéticas empleadas en las pruebas.

V. Conclusiones

Se ha presentado el algoritmo HEPPIE, como una alternativa competitiva para la estimación de la frecuencia fundamental de una señal audible, que opera con una atractiva relación exactitud/consumo de tiempo, en un amplio intervalo de frecuencias, de 50Hz a 5kHz, y que requiere únicamente cinco parámetros: la secuencia de entrada, la frecuencia de muestreo, el tamaño de la ventana de procesamiento, paso para cada estimación y la mínima resolución requerida. El algoritmo HEPPIE opera fundamentalmente en el dominio de la frecuencia, específicamente a partir del cálculo del periodograma, mediante una FFT, y su procesamiento mediante un filtro diferenciador seguido de un *matched filter*, a fin de detectar los picos espectrales significativos y la distancia entre estos.

El desempeño del algoritmo HEPPIE fue comparado contra reconocidos algoritmos de referencia y del último estado del arte, empleando, para ello, una colección de señales sintéticas, construidas a partir de los perfiles espectrales de instrumentos musicales reales. A partir de los registros de las métricas de desempeño, la media del error absoluto (MAE) y el consumo relativo de tiempo,

TABLA IV
Valores de MAE [Hz]

Algoritmo	h10snr MAE	h10snr D.E.	h1snr MAE	h1snr D.E.	harm MAE	harm D.E.	pure MAE	pure D.E.	MAE acumulada
FNLS	0.4	0.1	5.3	1.6	0.4	0.1	0.3	0.0	6.34
LHS	247.2	129.1	212.0	117.2	274.2	105.5	647.4	130.3	1380.89
NCF	877.1	133.4	883.1	654.3	961.7	0.0	1858.2	0.0	4580.08
PEF	135.5	105.3	275.8	201.6	20.1	0.2	9.3	0.2	440.67
RAPT	384.7	105.0	868.8	321.1	322.1	0.1	256.5	5.6	1832.13
SRH	335.6	170.8	294.5	186.8	345.1	22.1	1050.1	130.7	2025.35
SWIPE	50.4	82.0	409.8	329.8	19.2	5.7	9.2	4.9	488.58
YIN	2.7	0.3	152.5	58.7	2.7	0.0	2.1	0.0	159.93
HEPPIE	11.2	8.4	17.1	24.5	9.3	8.8	0.1	0.0	37.70

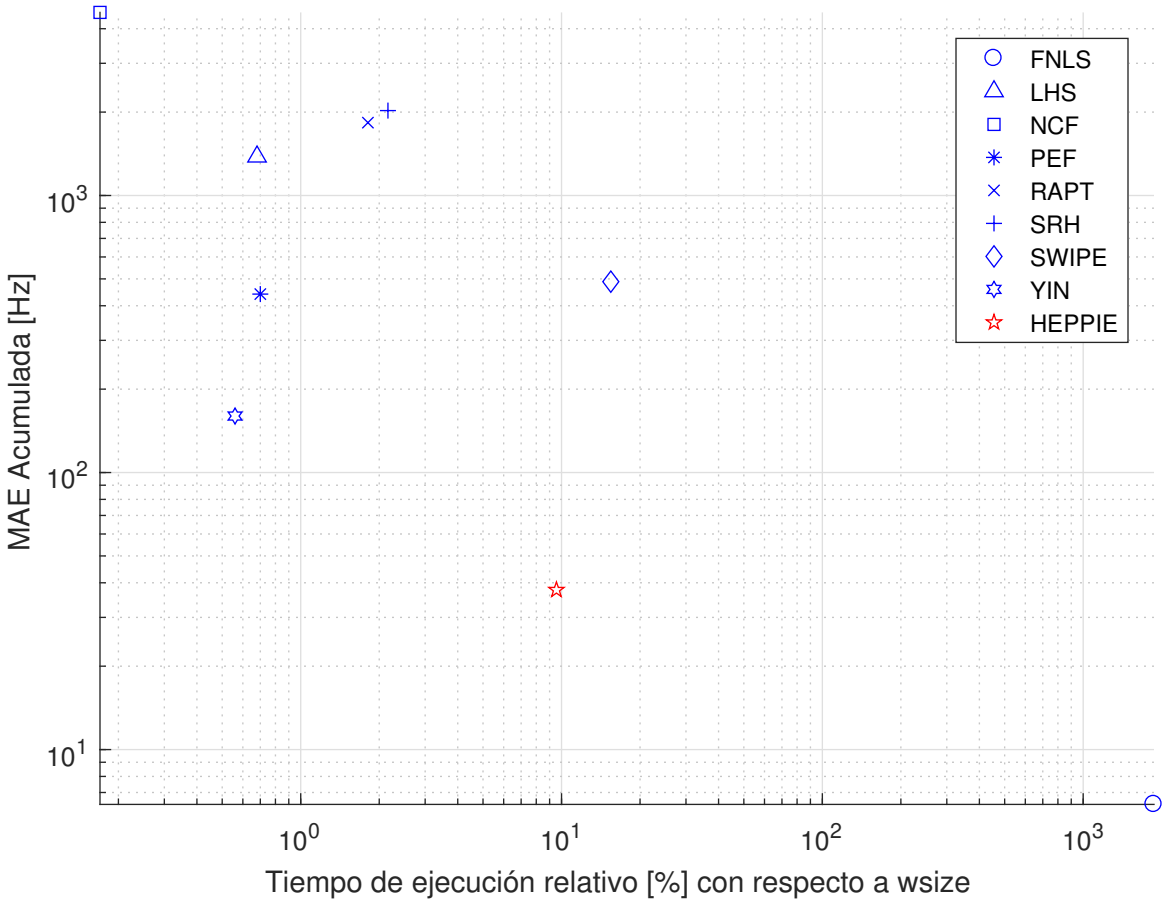

Fig. 2. Figura de mérito para el desempeño de los algoritmos probados.

se pudo comprobar que el desempeño del algoritmo HEPPIE es competitivo y que su posición global, dentro de la figura de mérito expuesta, se encuentra muy cerca del algoritmo YIN, aunque con mucho mejor tolerancia al ruido AWGN significativo.

En las pruebas realizadas, el algoritmo HEPPIE fue configurado intencionalmente para operar demandando el máximo de recursos computacionales y, a pesar de ello, su posición global en la figura de mérito resultó ser muy cercana a los algoritmos con la mejor combinación de exactitud/costo computacional. En el caso de que el algoritmo HEPPIE se empleara en aplicaciones que no requieren de una gran resolución, digamos de 10Hz, como podría ser el caso de la detección de tono en la voz humana, su consumo de tiempo se reduce significativamente hasta alcanzar un orden de magnitud similar al tiempo consumido por el algoritmo YIN, aunque con mucho mayor robustez ante el ruido.

Por todo lo anterior, podemos concluir que el algoritmo HEPPIE es una atractiva alternativa, para la estimación de la frecuencia fundamental, que permite al usuario determinar la relación exactitud/costo computacional que más le convenga.

Referencias

[1] G. Laguna-Sanchez *et al.*, "Simple and practical algorithm for wide spectrum pitch estimation with an acceptable compromise between accuracy and computational cost," in *IV Conferencia Internacional sobre Comunicación y Tecnologías Aplicadas 2024 (ICOMTA'24)*, 2024.

[2] MathWorks, "pitch: Estimate fundamental frequency of audio signal." The MathWorks Repository, 2024. https://la.mathworks.com/help/audio/ref/pitch.htmll.

[3] L. Rabiner *et al.*, "A comparative performance study of several pitch detection algorithms," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, no. 5, pp. 399–418, 1976. doi: 10.1109/TASSP.1976.1162846.

[4] L. Shi *et al.*, "Robust bayesian pitch tracking based on the harmonic model," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 11, pp. 1737–1751, 2019. doi: 10.1109/TASLP.2019.2930917.

[5] J. Nielsen *et al.*, "Fast fundamental frequency estimation: Making a statistically efficient estimator computationally efficient," *Signal Processing*, vol. 135, pp. 188–197, 2017. https://doi.org/10.1016/j.sigpro.2017.01.011.

[6] B. Atal, "Automatic speaker recognition based on pitch contours," *Journal of the Acoustical Society of America*, vol. 52, no. 6B, pp. 1687–1697, 1972. https://doi.org/10.1121/1.1913303.

[7] M. Noll, "Cepstrum pitch determination," *Journal of the Acoustical Society of America*, vol. 31, no. 2, pp. 293–309, 1967. https://doi.org/10.1121/1.1910339.

[8] A. Cheveigne and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002. DOI:10.1121/1.1458024.

[9] D. Talkin, *Speech Coding & Synthesis*, ch. A Robust Algorithm for Pitch Tracking (RAPT). Elsevier, 1995. W.B. Kleijn and K.K. Paliwal eds.

[10] S. Zahorian and H. Hu, "A spectral/temporal method for robust fundamental frequency tracking," *Journal of the Acoustical Society of America*, vol. 123, no. 6, pp. 4559–4571, 2008. https://doi.org/10.1121/1.2916590.

[11] D. Hermes, "Measurement of pitch by subharmonic summation," *Journal of the Acoustical Society of America*, vol. 83, no. 1, pp. 257–264, 1988. https://doi.org/10.1121/1.396427.

[12] T. Drugman and A. Alwan, "Joint robust voicing detection and pitch estimation based on residual harmonics," in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pp. 1973–1976, 2011. .

[13] X. Sun, "Pitch determination and voice quality analysis using subharmonic-to-harmonic ratio," in *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. I–333–I–336, 2002. doi: 10.1109/ICASSP.2002.5743722.

[14] S. Gonzalez and M. Brookes, "A pitch estimation filter robust to high levels of noise (PEFAC)," in *19th European Signal Processing Conference. Barcelona*, pp. 451–455, 2011. .

[15] A. Camacho and J. Harris, "A sawtooth waveform inspired pitch estimator for speech and music," *Journal of the Acoustical Society of America*, vol. 124, no. 3, pp. 1638–1652, 2008. https://doi.org/10.1121/1.2951592.

[16] D. Wang *et al.*, "A robust and low computational cost pitch estimation method," *Sensors*, vol. 22, pp. 1–21, 2022. https://doi.org/10.3390/s22166026.

[17] L. Shi, "Bayesian pitch tracking using harmonic model." GitHub Repository, 2019. https://github.com/LimingShi/Bayesian-Pitch-Tracking-Using-Harmonic-model.

[18] M. Brookes, "VOICEBOX: Speech processing toolbox for MATLAB." Imperial College Repository, 2013. http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html.

[19] X. Sun, "Pitch determination algorithm." MATLAB Central File Exchange, 2016. https://la.mathworks.com/matlabcentral/fileexchange/1230-pitch-determination-algorithm.

[20] D. Wang, "Srh variant." GitHub Repository, 2022. https://github.com/deshengwang001/SRH_Variant.

[21] K. Gorman, "Swipe' pitch estimator." GitHub Repository, 2012. https://github.com/kylebgorman/swipe.

[22] A. Zahorian, "YAAPT pitch tracking MATLAB function." Repository of Binghamton University, 2016. http://www.ws.binghamton.edu/zahorian/yaapt.htm.

[23] A. Cheveigne, "YIN, a fundamental frequency estimator for audio signals." Repository of Ecole normale superieure. http://audition.ens.fr/adc/.

[24] C. Cella *et al.*, "TinySOL: an audio dataset of isolated musical notes." Zenodo Repository, 2020. https://zenodo.org/records/3685367.

[25] L. Jackson, *Digital Filters and Signal Processing: with MATLAB exercises*. Kluwer Academic Publishers, 1995.

[26] G. Laguna-Sanchez, "HEPPIE, a simple and practical pith estimation algorithm." GitHub Repository, 2024. https://github.com/galaguna/HEPPIE_pitch_estimation_algorithm.