Universidad Nacional de Tres de Febrero
Ingeniería de Sonido

Procesamiento Digital de Señales

Trabajo práctico final
Phase Vocoder

Profesor: Ing. Ignacio Mieza

Martina Cribari - Jonathan Freidkes

12 de Diciembre de 2016

ANÁLISIS Y DESARROLLO DE UN PHASE VOCODER

MARTINA CRIBARI [1] Y JONATHAN FREIDKES [2]

[1] Universidad Nacional de Tres de Febrero, Ingeniería de Sonido, Procesamiento digital de señales, Buenos Aires, Argentina.
martinacribari@hotmail.com
[2] Universidad Nacional de Tres de Febrero, Ingeniería de Sonido, Procesamiento digital de señales, Buenos Aires, Argentina.
jonifreidkes@gmail.com

RESUMEN *– El presente trabajo abarca el análisis de funcionamiento y del código de programación del phase vocoder. Se busca la implementación del mismo y el estudio de diversos algoritmos involucrados en el funcionamiento de este sistema que procesa señales de audio en tiempo y frecuencia a través de la transformada de Fourier para tiempos cortos (STFT). Se estudian distintos métodos, desde el planteado inicialmente por Flanagan y Golden en 1966, las propuestas de Röbel y Portnoff, entre otros, hasta llegar a los desarrollos de Laroche y Dolson en 1999. Se intenta también desarrollar un código propio adaptando otros implementados para analizar su eficiencia y procesamiento.*

ABSTRACT *– This paper analyzes the phase vocoder operation and programming code. In order to arrive at its implementation, it is necessary to analyze several algorithms involved in this kind of system that process time and frequency of audio signals through Short Time Fourier Transform (STFT). The methods studied begin with the first paper presented by Flanagan and Golden in 1966, the additions made by Röbel and Portnoff, and the developments made by Laroche and Dolson in 1999. Besides new programming code is developed by the authors of this paper through other pieces founded in order to analyze its efficiency and processing.*

1. INTRODUCCIÓN

El phase vocoder, cuya primera presentación data de 1966, es un sistema que procesa el tiempo y la frecuencia de señales de audio. Puede ser visto como una técnica en la que señales de voz son representadas por su fase y espectro en tiempos cortos. Fue diseñado para economizar ancho de banda en transmisiones y para ser un medio para la compresión y expansión de tiempo de las señales [1]. Es también una solución de alta calidad para modificación de escala de tiempo y de tono. Se puede implementar como combinación de escalado de tiempo y conversión de frecuencia de muestreo o como un rastreo de picos de señal y modificación de su posición para luego ser sumado al espectro original [2]. Muchas veces también se utiliza este procesamiento para efectos sonoros buscados en el procesamiento de señales de audio en forma artística.

El algoritmo presentado originalmente comprende la extracción de módulo y fase de la transformada de Fourier de la señal modulada en amplitud y fase con una función cosenoidal. A través de esto se logra sintetizar una señal de salida a través de la señal portadora alterada por las magnitudes de la señal modulante. Se llega al multiplexado de señal con expansión o compresión de tiempo, que luego de pasar por la línea de transmisión puede ser vuelta a su forma original. Este algoritmo ha sido analizado y modificado por varios autores, con distintas propuestas para su aplicación más efectiva y sencilla, con menos requerimientos en el procesamiento. En el sistema se implementa una transformada de Fourier de tiempo corto (STFT) sobre la señal real en el dominio del tiempo que está multiplicada por una ventana. Se obtienen recuadros de espectro superpuestos con mínimo efecto sobre las bandas adyacentes. El retardo temporal entre cuadros tomados de la señal se denomina “hop”. Mediante la transformada inversa (ISTFT) en cada recuadro y la acumulación de recuadros se logra llegar a la señal de salida. Basta con conocer el módulo de cada recuadro tomado, mientras que la información de fase es necesaria para una perfecta recuperación de la señal sin modificaciones, como podría ser para el caso de uso en telecomunicaciones. La información de fase además permite la evaluación de espectro instantáneo [3].

Se presentan también en este trabajo los fundamentos teóricos que rigen los principios de este sistema, los resultados obtenidos con los códigos utilizados y varias conclusiones respecto de este sistema de codificación de señales de voz.

2. FUNDAMENTOS TEÓRICOS

El phase vocoder en general presenta tres etapas de implementación, de las cuales este trabajo utiliza las dos que son fundamentales.

En primer lugar, a partir de la señal a procesar, se efectúa la STFT (transformada de Fourier de tiempo corto) para obtener información de tiempo, frecuencia y amplitud de la señal. Se denomina etapa de análisis.

La segunda etapa, aquí no incluida, plantea que debido a las discontinuidades de fase que se producen

cuando se aplica una ventana periódica a la señal de entrada, se debe realizar una corrección de fase con un corrimiento determinado. Este procedimiento es conveniente utilizarlo en los casos en que se desee una reconstrucción perfecta de la señal y cuando en esta misma etapa se implemente algún tipo de efecto digital (chorus, delay, armonizadores, etc.).

La tercera etapa (síntesis) consiste en aplicar la transformada inversa de Fourier para tiempo cortos (ISTFT). La clave de esta parte es el método OLA (Overlap Add) en el que se anti-transforma el espectro de la señal de a tramos (en función del hop y la longitud de la ventana elegida) y se superponen los cuadros (f*rames*) en el dominio temporal.

A continuación, en la Figura 1 se puede ver un diagrama en bloque básico de lo detallado anteriormente:

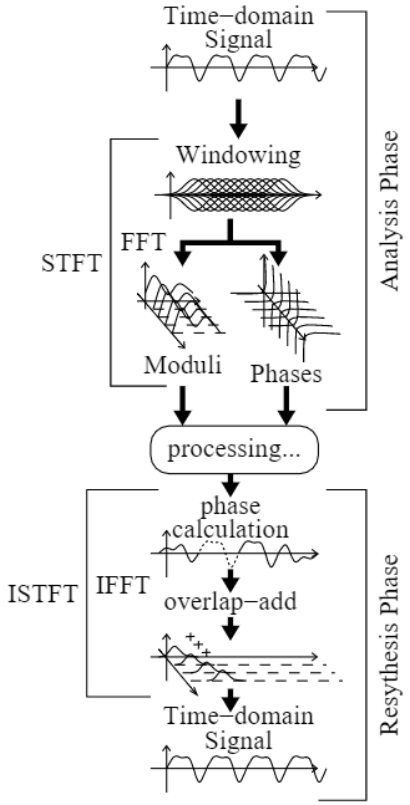

Figura 1: Etapas del Phase Vocoder. Fuente: De Götzen A, Bernardini N, Arfib D. "Traditional (?) implementations of a phase-vocoder: the tricks of the trade"

Time-domain signal: Señal en dominio temporal
Windowing: Ventaneo
FFT: Fast Fourier Transform, Transformada rápida de Fourier.
Moduli: Módulo de señal
Phases: Fase de señal
Processing: Procesamiento
Phase calculation: Cálculos de fase
Overlap-add: Superposición y adición (OLA)
STFT: Short Time Fourier Transform, Transformada de Fourier para tiempos cortos.
ISTFT: Inverse Short Time Fourier Transform, Transformada inversa de Fourier para tiempos cortos.
Analysis Phase: Etapa de análisis.
Resynthesis Phase: Etapa de síntesis.

Para la comprensión de este proceso es necesario considerar cierta teoría del procesamiento de señales que se describe a continuación.

2.1 TRANSFORMADA DE FOURIER

La transformada de Fourier representa uno de los fundamentos más importantes en el análisis de señales de audio. Es una herramienta matemática que permite convertir una señal de dominio temporal a su dominio frecuencial. Es reversible sin pérdidas, con lo que una señal puede ser procesada en una de las dos magnitudes y luego transformada a la otra. Es aplicable en señales periódicas que cumplan las condiciones de Dirichlet: absolutamente integrable y debe tener un número finito de mínimos y máximos y un número finito de discontinuidades en cualquier intervalo finito. Tiene propiedades de linealidad, cambio de escala, traslación, derivación, integración, convolución, entre otras.

Se ve en las ecuaciones 1 y 2 la síntesis y análisis para el caso de la transformada discreta (DFT, Discrete Fourier Transform por sus siglas en inglés), válidas de 0 hasta la anteúltima muestra (N-1).

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]\, W_N^{-kn} \quad (1)$$

$$X[k] = \sum_{n=0}^{N-1} x[n]\, W_N^{-kn} \quad (2)$$

Donde x[n] es la señal discreta en tiempo, X[k] la transformada, N el número de muestras y W_N^{-kn} es el factor de giro.

En este caso se utiliza la STFT (Short Time Fourier Transform, según sus siglas en inglés) que implica tiempo discreto y es usada para determinar el contenido en frecuencia y de fase en secciones locales de una señal y sus variaciones a través del tiempo. Implica la multiplicación de la señal con una ventana temporal, normalmente Hanning. Se define la STFT como:

$$STFT\,\{x[n]\} = X\,(m,\omega) = \sum_{n=-\infty}^{\infty} x\,[n]\, w[n-m]\, e^{-j\omega n} \quad (3)$$

Donde x[n] es la señal discreta en tiempo, w[n] es la ventana, m se supone discreta y ω continua. Mediante la transformada rápida de Fourier (FFT por sus siglas en inglés) se logra ω discreta y cuantizada.

La inversa de la transformación, ISTFT, está dada por:

$$x(t)w(t-\tau) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\tau,\omega)\, e^{j\omega t}\, d\omega \quad (4)$$

2.2 VENTANAS

Las ventanas son funciones temporales matemáticas que se multiplican con las funciones a analizar en procesamiento de señales. Limitan en tiempo el comportamiento de la señal y dependiendo de su tipo, traerán diversas consecuencias en la transformación de la señal. Por propiedades de transformada de Fourier, al multiplicar en tiempo, se producirá una convolución en frecuencia, con lo que la

respuesta en frecuencia de la ventana puede afectar a la señal. Se muestra en la Tabla 1 la comparación de características relevantes de las mismas en distintos tipos.

Ventana	Amplitud relativa de lóbulo lateral [dB]	Ancho aproximado de lóbulo principal
Rectangular	-13	4π/(M+1)
Bartlett	-25	8π/M
Hanning	-31	8π/M
Hamming	-41	8π/M
Blackman	-57	12π/M

Tabla 1: Varias ventanas y características.

Existe también el método de Kaiser que tiene parámetros variables en función de la atenuación buscada y el número de muestras.

2.3 FRECUENCIA DE MUESTREO

La frecuencia de muestreo de un sistema implica la cantidad de muestras de amplitud por segundo tomadas a la hora de discretizar una señal de tiempo continuo. De acuerdo al teorema de Nyquist, en condiciones de filtro de restauración ideal y ancho de banda de señal limitado, la frecuencia de muestreo debería ser mayor o igual al doble de la máxima frecuencia de la señal a discretizar. En el procesamiento digital de señales es de vital importancia, ya que puede determinar alteraciones frecuenciales de las componentes de una señal en caso de no respetarse la misma frecuencia en todo el encadenado de procesos.

2.4 HOP

Durante el proceo de STFT se multiplica la señal en tiempo con una ventana y se obtienen cuadros de espectro (*frames*). El tiempo de retardo en el que cada cuadro es tomado de la señal se denomina *hop*.

3. TECNOLOGÍA INVOLUCRADA

El desarrollo, implementación, compilación y prueba del código generado fue dado íntegramente en el software Matlab. El soporte de hardware fue en dos computadoras de distintas características:

- Apple Macbook Pro 13.3" 2010. Procesador Intel Core 2 Duo 2.4 GHz, RAM 8 GB 1067 MHz DDR3 con SSD, OS X 10.9.5. Matlab versión 2014 b.
- Computadora de escritorio. Procesador AMD FX-4100 Quad-Core 3.6 GHz, RAM 8 GB 1600 MHz, Windows 10 64 bit. Matlab versión 2012 b.

3.1 TIEMPOS DE PROCESAMIENTO

Los tiempos de procesamiento (en segundos) se especifican para las dos computadoras utilizadas y para las señales analizadas (voz y guitarra) como se muestra en la Tabla 2.

	Apple Macbook Pro	AMD FX-4100
Voz	3.9366	0.85273
Guitarra	3.9413	0.87588

Tabla 2: Tiempos de procesamiento para distintas señales y hardware.

4. RESULTADOS Y DISCUSIONES

Se trabajó a modo de prueba y error variando los principales parámetros del procesamiento, tales como el hop, la frecuencia de muestreo y la longitud de la ventana con el objetivo de modificar el pitch y mantener la longitud original de la señal. Para ello se utilizaron dos señales diferentes: por un lado una voz grabada por un locutor de 10 segundos de duración y por otro, una pista de guitarra acústica de la misma duración.

La Figura 2 muestra las formas de onda de una señal de voz (arriba, en azul) y la que se obtuvo luego de realizar las modificaciones (abajo, en rojo) que se describen a continuación:

En el primer caso, el hop de síntesis es el doble del hop de análisis y se duplica la frecuencia de muestreo.

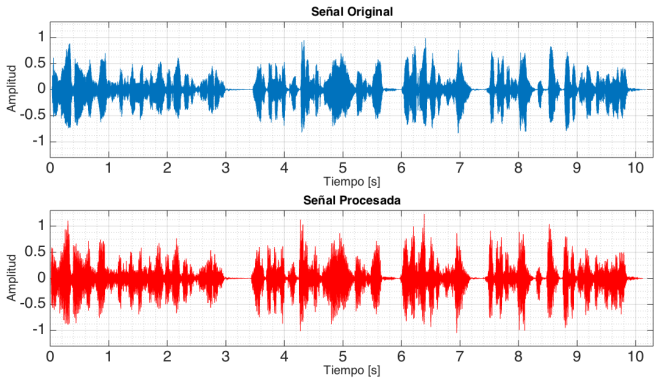

Figura 2: Señal de voz de entrada y salida del phase vocoder (pitch superior).

En el segundo caso, el hop de síntesis es la mitad del hop de análisis y la frecuencia de muestreo es la mitad, se muestra el resultado en la Figura 3.

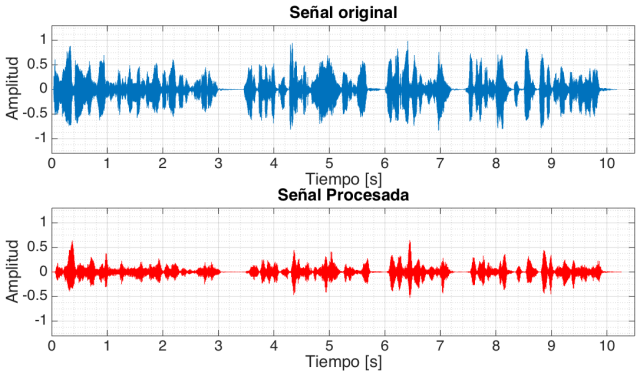

Figura 3: Señal de voz a la entrada y salida del phase vocoder (pitch inferior).

Se puede ver que sucede lo mismo cuando se introduce una pista de guitarra como muestran las Figuras 4 y 5. Musicalmente suena disonante, no solamente se modifica el pitch. En una voz, esto no es tan fácilmente perceptible.

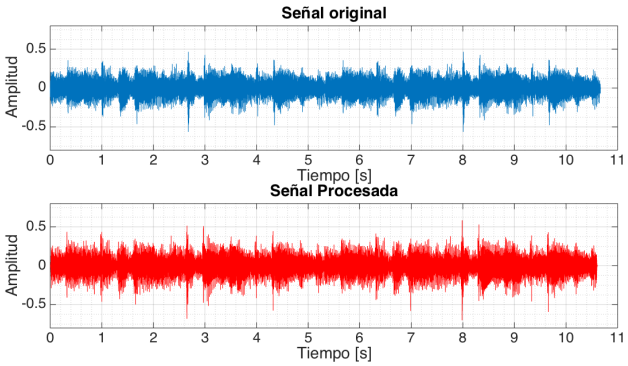

Figura 4: Señal de guitarra a la entrada y salida del phase vocoder (pitch superior).

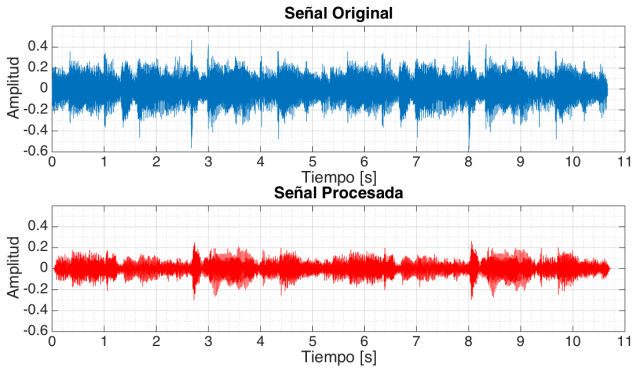

Figura 5: Señal de guitarra a la entrada y salida del phase vocoder (pitch inferior).

Una ventana de un gran de número de muestras resulta en una muy buena definición en frecuencia, no así en tiempo. En cambio, una ventana reducida favorece la resolución en tiempo. Por eso también se modifica la longitud de la ventana de la STFT. En el caso del código de este trabajo, cuando se la agranda, la señal se "corta" y solo quedan algunos picos en diferentes intervalos de tiempo. La señal a la salida presenta graves inconvenientes de pérdida de información.

Se prueba también distintos tipos de ventanas para cada señal, llegando a notar una sutil diferencia entre la utilizada normalmente, Hamming, y la ventana de Blackman, donde se ve una leve disminución de la forma de onda en la Figura 7 respecto de la 6. Eso mismo se refleja más claramente para el espectro mostrado en la Figura 9 respecto de la Figura 8, donde se ve un leve aumento de nivel en baja frecuencia, pero auditivamente es casi imperceptible.

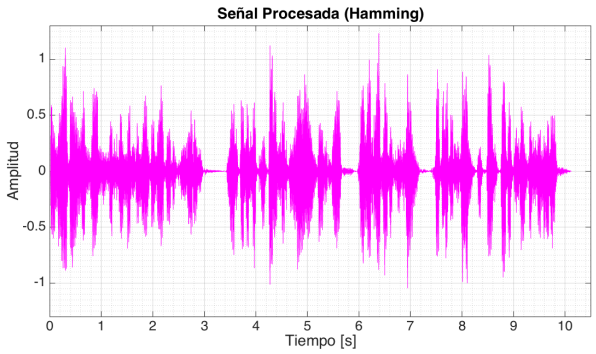

Figura 6: Señal de voz procesada con ventana Hamming.

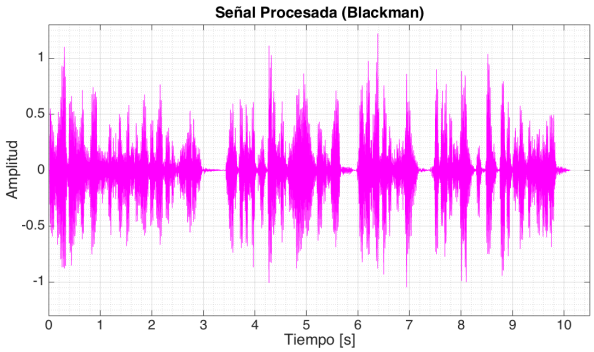

Figura 7: Señal de voz procesada con ventana Blackman.

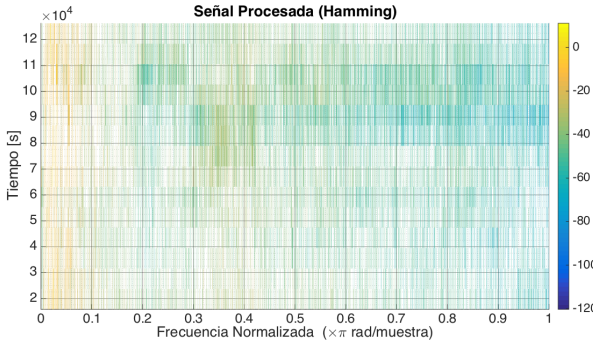

Figura 8: Espectro de señal de voz procesada con ventana Hamming.

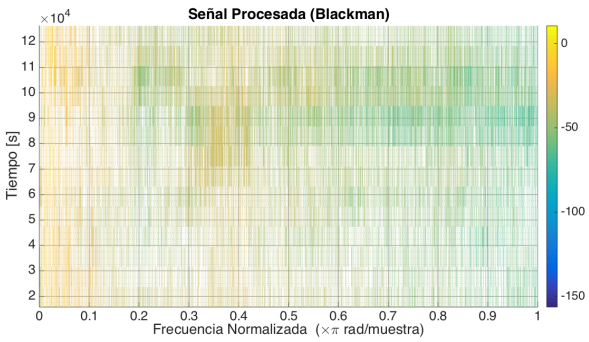

Figura 9: Espectro de señal de voz procesada con ventana Blackman.

Se prueba lo mismo para la señal de guitarra, obteniendo resultado resultados análogos, mostrados en las Figuras 10 a 13.

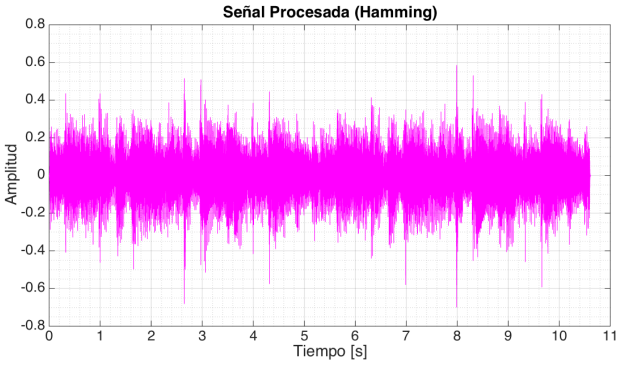

Figura 10: Señal de guitarra procesada con ventana Hamming.

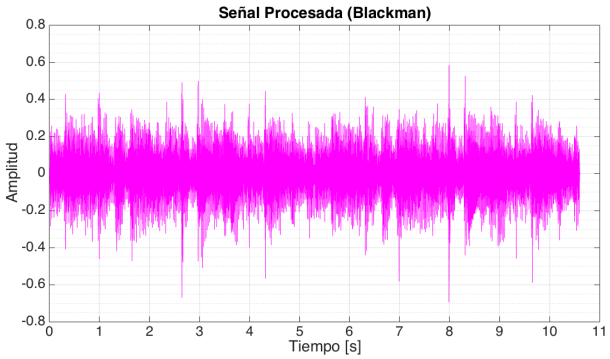

Figura 11: Señal de guitarra procesada con ventana Blackman.

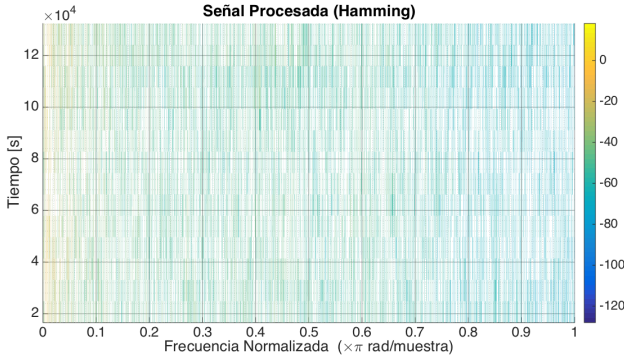

Figura 12: Espectro de señal de guitarra procesada con ventana Hamming.

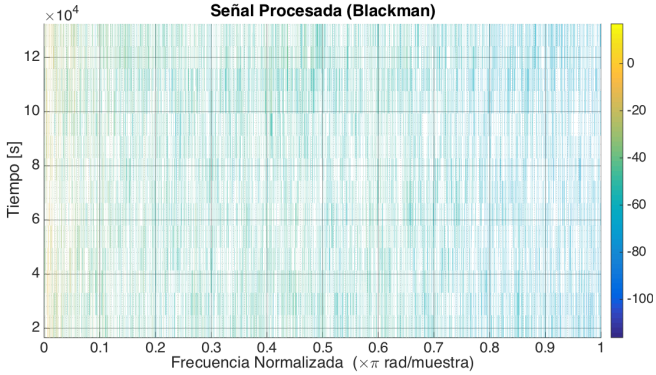

Figura 13: Espectro de señal de guitarra procesada con ventana Blackman.

5. CONCLUSIONES

Cuando el hop de la etapa de síntesis es la mitad del hop de análisis, la pista se acorta a la mitad; es decir se obtienen la mitad de muestras.

Al duplicar el hop en la última etapa, la pista se "estira" y se percibe un sonido "dilatado".

Modificando la frecuencia de muestreo y el hop en simultáneo, comienza a producirse el cambio de tono. Para que se mantenga la duración de la señal, esta frecuencia se duplica o se reduce a la mitad según lo haga el hop de la etapa de síntesis.

Observando las Figuras 1 y 2, se puede concluir que cuando se eleva el tono (pitch) y el sonido es más agudo, la forma de onda resultante es muy similar a la original. Podría decirse que hay un pequeño corrimiento entre ambas señales, pero esta diferencia es sutil en relación a lo que sucede cuando el pitch es inferior.

Al trabajar con la mitad de hop y frecuencia de muestreo, la señal se reduce de forma global en concepto de amplitud. Esto puede ser solucionado con una normalización en la salida del phase vocoder. Sin embargo, se pierde información sobre todo en los picos y se puede ver claramente en la diferencia de área abarcada por cada señal.

El cambio de tipo de ventana en la STFT no parece afectar el procesamiento, sino en forma muy sutil, como se refleja en el espectro mostrado en las Figuras 8, 9, 12 y 13.

Se intentó también desarrollar un código propio mediante el método propuesto por Laroche y Dolson [2]

que implica modificar los picos hallados en la señal para facilitar el procesamiento del phase vocoder. El mismo no resultó como era esperado, pero se anexa al trabajo de todos modos.

Asimismo se presenta el anexo del código utilizado para el análisis modificado a partir de otro phase vocoder encontrado [5] [6].

6. REFERENCIAS

[1] Flanagan J. L, Golden R. M. "*Phase Vocoder*". The Bell System Technical Journal, vol. 45, pp. 1493-1509. Estados Unidos, Noviembre 1966.

[2] Laroche J, Dolson M. "New phase-vocoder techniques for pitch.shifting, harmonizing and other exotic effects"1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, New York, Oct. 17-20, 1999

[3] De Götzen A, Bernardini N, Arfib D. "Traditional (?) implementations of a phase-vocoder: the tricks of the trade" COST G-6 Conference on Digital Audio Effects (DAFX-00). Verona, Italia, Diciembre 2000

[4] Apuntes de la cátedra, Procesamiento Digital de Señales, Ingeniería de Sonido, UNTreF. Buenos Aires, Argentina, Noviembre 2016.

[5] MathWorks, File Exchange, *"Short-Time Fourier Transformation (STFT) with Matlab Implementation".* https://www.mathworks.com/matlabcentral/fileexchange/45197-short-time-fourier-transformation--stft--with-matlab-implementation/content/stft.m

[6] Electrical Engineering in Columbia University, New York, USA. *"A Phase Vocoder in Matlab". http://www.ee.columbia.edu/ln/rosa/matlab/pvoc/*

7. ANEXOS

7.1. CÓDIGO IMPLEMENTADO

```
%PHASE VOCODER

clear all
clc

tstart=tic;

% x0 es la senal de entrada

      %x0=audioread('Strummed Acoustic.wav');
      x0=audioread('voz2.wav');
      % x representa como vector columna a x0
      x=x0(:,1);

      %frecuencia de muestreo
      fs=44100;
      % vector tiempo
      t=(0:1/fs:((length(x)-1)/fs));

      %grafico de la senal de entrada
      subplot(2,1,1)
      plot(t,x)
      xlabel('Tiempo [s]')
      ylabel('Amplitud')
      grid on
      hold on

      %implementacion de la STFT


      % wlen es la longitud de la ventana
      wlen=2^12;

      % h es la longitud del hop
      h=wlen/4;

      % nfft es la cantidad de muestras para calcular la FFT
      nfft=2^12;

      % f es el vector frecuencia en Hz
      % t es el vector tiempo en segundos
      % stft es la matriz resultante (tiempo en las columnas, frecuencias en las
      % filas)

      % xlen es la longitud de la senal
      xlen = length(x);

      % win es la ventana de Hamming periodica
      win = hamming(wlen, 'periodic');

      % otras ventanas
      %win = hann(wlen, 'periodic');
      %win = blackman(wlen, 'periodic');

      % dimensiones de la matriz stft
      rown = ceil((1+nfft)/2);          % total de filas
      coln = 1+fix((xlen-wlen)/h);      % total de columnas
      stft = zeros(rown, coln);         % dimensiones de la matriz stft


      % valor inicial de los indices
      indx = 0;
      col = 1;

      while indx + wlen <= xlen
          % ventaneo
          xw = x(indx+1:indx+wlen).*win;

          % FFT
          X = fft(xw, nfft);

          % se actulizan las columnas de la stft
          stft(:, col) = X(1:rown);

          % se actualizan los indices de la matriz stft
          indx = indx + h;
          col = col + 1;
      end

      % calculo de los vectores de tiempo y frecuencia
      t = (wlen/2:h:wlen/2+(coln-1)*h)/fs;
      f = (0:rown-1)*fs/nfft;

      stft;

      %_____________________________________________________


```

```
89.      % Interpolacion de fases en
   caso de reconstruccion perfecta
90.
91.      % b=stft;
92.
93.      % [rows,cols] = size(b);
94.
95.      % N = 2*(rows-1);
96.
97.      % matriz de salida
98.      % c = zeros(rows,
   length(t));
99.
100.     % % corrimiento de fase
   esperado
101.     % dphi = zeros(1,N/2+1);
102.     % dphi(2:(1 + N/2)) =
   (2*pi*h)./(N./(1:(N/2)));
103.
104.     % Acumulador de fase
105.     % Se establece la fase del
   primer cuadro para reconstruccion
   perfecta
106.
107.     % ph = angle(b(:,1));
108.
109.     % % Para evitar problemas se
   agrega una columna de ceros a la
   matriz b
110.
111.     % b = [b,zeros(rows,1)];
112.
113.     % ocol = 1;
114.     % for tt = t
115.     %   % Para las primeras dos
   columnas de b
116.     %   bcols = b(:,floor(tt)+[1
   2]);
117.     %   tf = tt - floor(tt);
118.     %   bmag = (1-
   tf)*abs(bcols(:,1)) +
   tf*(abs(bcols(:,2)));
119.     %   % se calcula el avance
   de fase
120.     %   dp = angle(bcols(:,2)) -
   angle(bcols(:,1)) - dphi';
121.     %   % Se reduce alrango que
   va de -pi a pi
122.     %   dp = dp - 2 * pi *
   round(dp/(2*pi));
123.     %   % Se guarda la
   informacion de la columna
124.     %   c(:,ocol) = bmag .*
   exp(j*ph);
125.     %   % Se acumula la fase y
   se repite el procedimiento para el
   cuadro siguiente
126.     %   ph = ph + dphi' + dp;
127.     %   ocol = ocol+1;
128.     % end
129.
130.
131.
132.     %_____________________________
   _____________________________
133.
134.     % ISTFT
135.
136.     %nuevo hop
137.     %h=wlen/8; % vale la mitad
   que el hop anterior
138.     h=wlen/2; %el doble de h de
   analisis
139.     %h=wlen/4; %mismo que el
   anterior
140.
141.     % estimacion de la longitud
   de la senal
142.     coln = size(stft, 2);
143.     xlen = nfft + (coln-1)*h;
144.     x = zeros(1, xlen);
145.
146.     % ventana de hamming
   periodica
147.     win = hamming(nfft,
   'periodic');
148.
149.     %  ISTFT y OLA (Overlapp
   add)
150.     if rem(nfft, 2)
   % nfft impar excluye el punto de
   Nyquist
151.         for b = 0:h:(h*(coln-1))
152.             % se extraen las
   FFTs
153.             X = stft(:, 1 +
   b/h);
154.             X = [X; conj(X(end:-
   1:2))];
155.
156.             % se hace la FFT
   inversa
157.             xprim =
   real(ifft(X));
158.
159.             % se superponen las
   IFFT
160.             x((b+1):(b+nfft)) =
   x((b+1):(b+nfft)) + (xprim.*win)';
161.         end
162.     else
   % nfft par incluye el punto de
   Nyquist
163.         for b = 0:h:(h*(coln-1))
164.             % se extraen las
   FFTs
165.             X = stft(:, 1+b/h);
166.             X = [X; conj(X(end-
   1:-1:2))];
167.
168.             % se hace la FFT
   inversa
```

```
            xprim =
real(ifft(X));

            % se superponen las
IFFT
            x((b+1):(b+nfft)) =
x((b+1):(b+nfft)) + (xprim.*win)';
        end
    end

    W0 = sum(win.^2);
    x = x.*h/W0;
% se escala la superposicion de
senales

    % se calcula el vector de
tiempo de la senal de salida
    fs1=fs*2;
% frecuencia de muestreo en la
reconstruccion
    actxlen = length(x);
% longitud de la senal a la salida
    t = (0:actxlen-1)/fs1;
% vector de tiempo

    %grafico de la senal a la
salida
    subplot (2,1,2)
    plot(t,x,'r')
    xlabel('Tiempo [s]')
    ylabel('Amplitud')
    grid on
    hold off

    filename= 'tiempo2.wav';
    audiowrite(filename,x,fs1)

    telapsed=toc(tstart);
    str=['El tiempo de
procesamiento es :
',num2str(telapsed)];
    disp(str)
```

7.2 CÓDIGO DE DETECCIÓN DE PICOS

```
clear all
clc

% inicio de los índices
col = 1;
% se obtienen los picos para cada
columna de la matriz stft
while col <= coln;

    for n=1:coln;
    values=abs(stft0(:,n));

for k=1:(length(f)-4)
    if values(2+k)>values(1+k)&&
values(2+k)>values(k) &&
values(2+k)>values(3+k) &&
values(2+k)>values(4+k);
        picos(k)=values(2+k);
    end
end

picos=[picos zeros(1,length(f)-
length(picos))];
P=find(picos);
% Se hace un corrimiento de 2 muestras
Pn=P+2;

for r=1:length(P);
values(P(r))=values(Pn(r));
end
    end

    % se actualize la matriz stft
procesada
    stft2(:, col) = values(1:rown);

    % se actualizan los índices
    col = col + 1;
end

% Se obtiene stft2 que es la matriz
resultante de desplazar los picos en
frecuencia
stft2;
```