INSTITUTO TECNOLÓGICO
DE
BUENOS AIRES

Ingeniería Electrónica

(22.47) - Procesamiento de Voz

TRABAJO PRÁCTICO 1

Vocoder de aplicación musical en tiempo real.

Grupo:
Díaz, Ian Cruz
Rodrigez Turco, Martin
Sebastián

Legajos:
57515
56629

Contenido

1. Introducción

Para el siguiente proyecto se busco implementar un **Vocoder** de aplicación musical. Un vocoder es un sistema que analiza el sonido de una persona hablando, y mediante diferentes algoritmos realiza cambios en los pulsos glotales sin alterar el filtro articulatorio, y de esta manera logra sintetizar un sonido artificial de la señal de voz de la persona que este hablando. Junto con esto, y eligiendo frecuencias fundamentales de los pulsos glotales acordemente, se pueden generar melodías y acordes para utilizar en ambientes musicales.

2. Enfoque LPC

Para poder realizar este sistema se decidió utilizar el enfoque *Linear Prediction Coefficients* (LPC), que se detallará brevemente a continuación.

2.1. Modelo de producción de voz

Si recordamos del modelo de producción de voz, podemos simplificar el modelo en los bloques de la Figura 2.1.

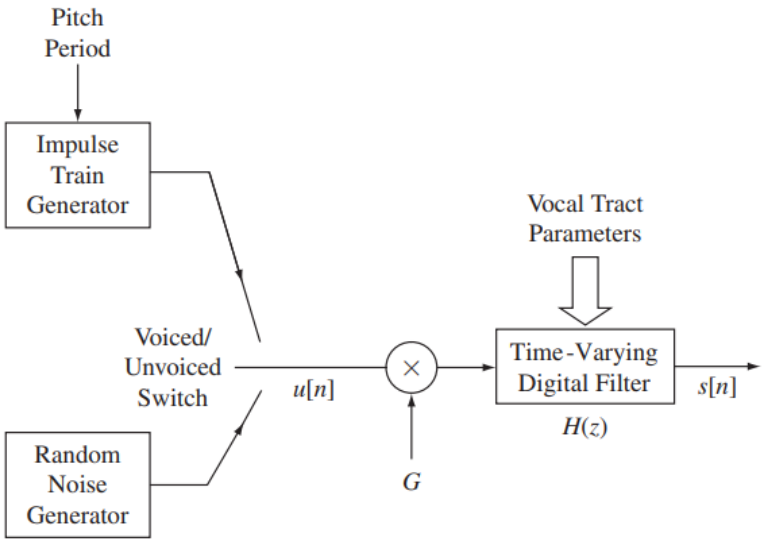


FIGURA 2.1: Diagrama en bloques del modelo de producción de voz

Realizando algunos cálculos [2] [3], se llega a la conclusión que el filtro error de predicción estima la inversa del filtro articulatorio, por lo tanto, simplemente estimando el filtro error de predicción podemos utilizar el filtro articulatorio obtenido de una porción de audio para sintetizar unos pulsos glotales artificiales.

2.2. Implementación

Para la implementación de este sistema, se realizó el método de la autocorrelación para estimar los coeficientes del filtro de error de producción y una ventana de *hann* para realizar la sintetización[1]. Esto se puede ver en el código a continuación[2]

[1]Recordemos que para sintetizar una señal correctamente debemos cumplir con el requisito de que la suma de las ventanas en el tiempo debe ser constante, y por lo tanto, con las ventanas de hann, el overlap debe ser del 50 %.

[2]Este código fue basado en [1], donde ese código fue implementado para poder sintetizar la voz con pulsos artificiales, pero intentando modificar lo menos posible la voz de salida respecto de la entrada,

```
def vocode(signal, fs, f_custom, block_len, overlap, order, prev_block
    , p_coverage=0.01, unvoiced2zeros=True, glotal_type="triang"):

  out = zeros(len(signal))
  out[:len(prev_block)] = prev_block
  prev_delta = 0
  T_samples = int((fs / f_custom))

  for block, idx in block_process(signal, fs, block_len, overlap):
    gain_correction = (1 - overlap) * 2  # *2 due to hann window
    block *= hann(len(block)) * gain_correction

    rxx = correlate(block, block, mode='full')
    rxx = rxx[len(rxx) // 2:]
    period_samples, is_voiced = fundamental_period_estimate(rxx, fs)
    a = -solve_toeplitz(rxx[:order], rxx[1:order + 1])
    a = np.concatenate(([1], a))
    error_power = rms(lfilter(a, (1,), block))
    if is_voiced:
      vocoded, new_delta = pitch_maker(len(block), T_samples,
   prev_delta, overlap=overlap)
      prev_delta = new_delta
      impulse_response = glotales[glotal_type](len(block), p_coverage=
   p_coverage)
      vocoded = np.convolve(vocoded, impulse_response, mode="same")
    else:
      if unvoiced2zeros:
        vocoded = np.zeros(len(block))
      else:
        vocoded = randn(len(block)) / 2

    vocoded = lfilter((error_power,), a, vocoded)
    vocoded *= hann(len(block))
    out[idx:idx + len(block)] += deemphasis(vocoded)
  return out
```

Listing 1: Vocoder

En este código se puede ver que, lo primero que se realiza es dividir la señal de entrada en bloques de *block_len* milisegundos, con un overlap de *overlap* % [3], luego, se trabajará por bloques. Como se puede ver, en cada bloque, lo primero que se hace es obtener la autocorrelación $R_{xx}(\tau)$ para $\tau \geq 0$, y se llama a la función *fundamental_period_estimate*, esta función definida por el creador del código original se utilizaba

por ejemplo para utilizarla en codificación de voz.

[3]En este caso, el overlap siempre debería ser del 50 % ya que solo se utiliza ventanas de *hann*

para obtener el período de los pulsos glotales así como para saber si la señal es sonora o sorda. Si ahondamos mas profundo en la función y con lo que nos interesa (saber si la señal es sonora o no), podemos ver que el método que se realiza para determinar si la señal es o no sonora, simplemente se busca cual es el máximo de la autocorrelación, y si esta supera un cierto threshold, se determina que es sonora, esto se puede ver en el Listing 2. Una vez hecho esto, se determinan los coeficientes del filtro error de producción con *solve_toeplitz*, y se estima la potencia de error para hacer el control automático de ganancia (AGC) al final de la síntesis. Finalmente, cuando la señal de entrada es determinada como sonora, se realizan los pulsos glotales deseados y se los filtran por el filtro $H(z) = \frac{G}{A(z)}$, siendo A(z) el filtro error de predicción y $G^2 = e(n)$.

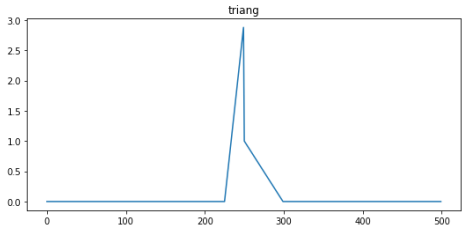
```
def fundamental_period_estimate(rxx, fs):
  """
  Calculates the fundamental frequency of an auto correlation array.
  rxx   the auto correlation array.
  fs    the sample rate in hertz.
  """
  f_low, f_high = 50, 250
  f_low_idx = round(fs / f_low)
  f_high_idx = round(fs / f_high)
  period_idx = argmax(rxx[f_high_idx:f_low_idx]) + f_high_idx
  is_voiced = max(rxx) > 0.20
  return (period_idx, is_voiced)
```

Listing 2: Fundamental Period Estimate

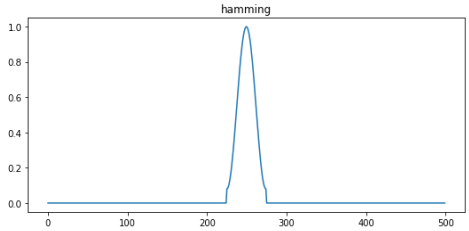
2.3. Pulsos glotales

Para la implementación de pulsos glotales sintéticos, se utilizaron 4 formas de pulsos distintas, ellas son:

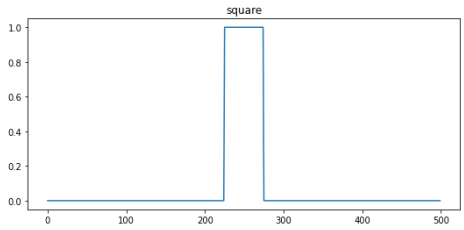
- **Triangular**: Cuenta con una amplia personalización donde se es capaz de ajustar la pendiente de crecimiento, la de decrecimiento, la altura, el ancho, el final de la primera rampa, y el comienzo de la segunda rampa.
- **Hamming**: Es una ventana de hamming centrada, con ancho ajustable y amplitud ajustable.
- **Cuadrada**: Es un pulso cuadrado con personalización del ancho y la altura del mismo
- **Exponencial**: Un pulso centrado en 0 donde la parte negativa es una exponencial creciente y la parte positiva una exponencial decreciente.



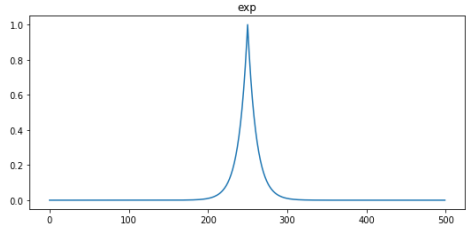
(A) Pulso glotal Triangular



(B) Pulso glotal Hamming



(C) Pulso glotal Cuadrada



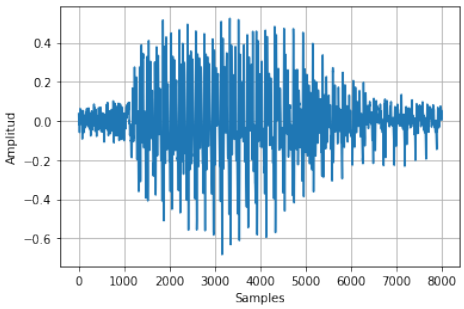
(D) Pulso glotal Exponencial

FIGURA 2.2: Pulsos glotales

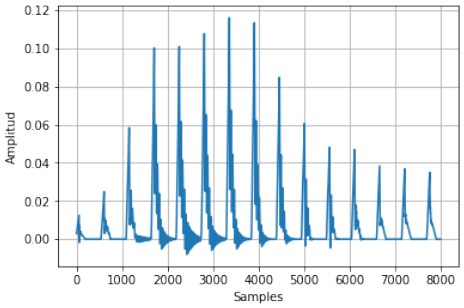
Ejemplos de estos pulsos se pueden ver en la figura 2.2. Por otro lado, si sintetizamos una señal de voz de ejemplo con estos pulsos glotales, con los siguientes parámetros:

- Frecuencia = 500 (Hz)
- Overlap = 50 %
- Bloque = 32 (ms)

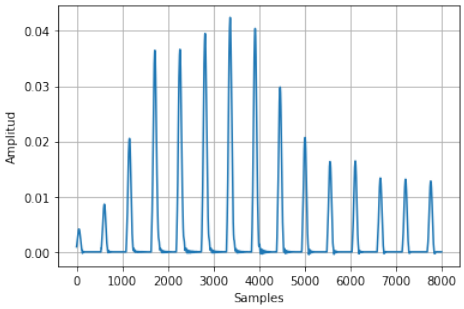
Obtenemos los resultados que se pueden observar en la Figura 2.3.



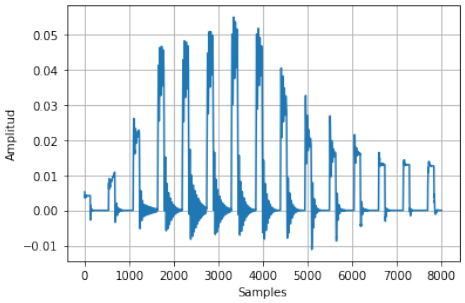
(A) Señal original



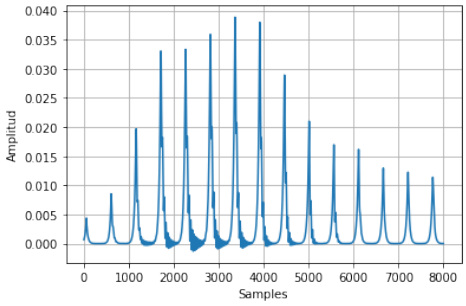
(B) Síntesis con triangular



(C) Síntesis con Hamming



(D) Síntesis con Cuadrada



(E) Síntesis con Exponencial

FIGURA 2.3: Síntesis de señales con diferentes pulsos glotales

Referencias

[1] Bastian Bechtold. *Pocoder*. URL: https://github.com/bastibe/pocoder.

[2] Thomas F. Quatieri. *Discrete Time Speech Signal Proccesing: Principles and practice*. 1.ª ed. Prentice-Hall, Inc, 2002. ISBN: 0-13-242942-X.

[3] Lawrence R. Rabiner y Ronald W. Schafer. *Introduction to Digital Speech Processing*. 1.ª ed. now Publishers Inc., 2007. ISBN: 978-1-60198-070-0.