Producción y análisis de música

Autores: Alejandro Magnorsky, Andrés Mata Suárez, Mariano Merchante

Instituto Tecnológico de Buenos Aires

Resumen

Palabras clave

Archivo de sonido; Partitura; Transformada de Fourier; Frecuencia.

I. Introducción

II. Desarrollo

A. Construcción de una partitura a partir de un archivo de sonido

El objetivo es la creación de un programa que, dado un archivo de sonido (.wav), genere una partitura. El código se divide en dos archivos o funciones, que son las que se detallan en la figuras 1 y 2.

Una señal física, como lo es la onda del sonido, puede ser representada mediante una función del tiempo continua. Así, un tono puro produce una onda proporcional a:

$$x(t) = \sin(f_0 * t) \tag{1}$$

La ecuación (1) es la representación en el tiempo del tono puro, pero también existe una representación en la frecuencia. Las canciones están formadas por sucesiones y superposiciones de funciones de la forma de la ecuación (1). En particular, si se considera el caso de tonos puros consecutivos, obteniendo la representación en frecuencia cada cierto intervalo de tiempo, se pueden conseguir todas las notas musicales que componen dicha música. Cabe mencionar que se trabaja con una discretización de la función continua que modela la música.

Para obtener la representación de la música en frecuencia, es decir, el espectro de frecuencias, se utiliza la Transformada Discreta de Fourier para cada uno de los intervalos de la función original. La Transformada Discreta de Fourier se define como:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi \frac{k}{N} n} \tag{2}$$

donde x_n es la función periódica que representa el tono puro y N es su período.

Para comenzar, la función de la figura 1 recibe la ruta del archivo de sonido que se desea procesar. Utilizando la función de Octave, `wavread`, se obtiene el vector que representa la onda del sonido, x, junto a la frecuencia de muestro de dicha señal, f_s. Considerando intervalos de tiempo de $30ms$, se divide a x en vectores de $\frac{30}{1000} f_s$ elementos, ya que esa es la cantidad de muestreos del sonido que se realizan a lo largo de $30ms$. Para cada uno de dichos vectores, se calcula la Transformada Discreta de Fourier usando la función que provee Octave, `fft`, la cual usa el algoritmo de la Transformada Rápida de Fourier. La ventaja de esto es que tiene $O(n \log n)$ en lugar de $O(n^2)$.

Cada vector X que devuelve la función `fft`, pasa a ser la entrada para otra función llamada `fftshift`. El efecto de `fftshift` es mover los valores de X para que queden centrados en torno a la frecuencia 0. Finalmente, considerando los valores en que la frecuencia es positiva, es decir, la segunda mitad del vector X, se averigüa en qué valor de la frecuencia X se maximiza. Ese valor es la frecuencia fundamental del intervalo considerado, que se guarda en el vector de frecuencias que devuelve la función en cuestión.

Listing 1: Implementación de la generación de la secuencia de frecuencias de un archivo de sonido.

```
function frequencies = getFrequencies(wavFile)

% x es la onda del sonido
% fs es la frecuencia de muestreo (cantidad de muestreos por segundo).
% Es decir, un segundo equivale a fs muestras de x
% bits es la cantidad de bits de cada muestra
[x, fs, bits] = wavread(wavFile);

interval = floor(fs*30/1000);
quant = floor(length(x)/interval);

for k=1:quant
        lower = (k-1)*interval+1;
        upper = k*interval;

        X = fft(x(lower:upper));
        X = fftshift(X);
        X = X(floor(length(X)/2)+1:length(X));
        f = 0:fs/interval:fs/2-1;

        [number, pos] = max(X);
        frequencies(k) = f(pos);
endfor
plot(frequencies)
endfunction
```

Listing 2: Implementación de la escritura de una partitura dada una secuencia de frecuencias.

```
function partiture = writePartiture(frequencies)
        for i=1:length(frequencies)
                partiture(i,:) = frequencyToNote(frequencies(i));
        endfor
endfunction
```

Utilizando el vector de frecuencias generado a partir de la función de la figura 1, la función de la figura 2 genera un vector de caracteres que representan los tripletes de cada una de las frecuencias.

III. Resultados

IV. Conclusiones

Referencias

Mathews, John H., Fink, Kurtis D., “Numerical Methods Using MATLAB”, Prentice Hall, 1999