Reunión Anual de la Unión Matemática Argentina - Bahía Blanca, 2016

Arierl Salort

Guía Práctica GNU OCTAVE
Compresión de sonidos usando la DCT

Dado un vector $x = (x_1, \ldots, x_N)$, su transformada de coseno discreta o DCT es el vector dado por $X = (X_1, \ldots, X_N)$, donde

$$X_k = w_k \sum_{n=1}^{N} x_n \cos(\tfrac{\pi}{2N}(2n-1)(k-1)), \quad k = 1, 2, \ldots, N$$

siendo

$$w_k = \begin{cases} \frac{1}{\sqrt{N}}, & k = 1 \\ \sqrt{\frac{2}{N}}, & k = 2, \ldots, N \end{cases}$$

La transformada de coseno discreta inversa o IDCT del vector $X = (X_1, \ldots, X_N)$ devuelve el vector $x = (x_1, \ldots, x_N)$ dado por

$$x_n = \sum_{k=1}^{N} w_k X_k \cos(\tfrac{\pi}{2N}(2n-1)(k-1)), \quad k = 1, 2, \ldots, N$$

La onda sonora que representa el audio es una función continua definida en el intervalo finito [0,T]. El sonido analógico se lo digitaliza tomando cierta cantidad de muestras por segundo, y almacenando estos samples con cierta precisión.

Por ejemplo, en un **CD de audio** tenemos:
Tasa de muestro (o sample rate): 44100 (i.e., se toman 44100 muestras por segundo, o 44.1 kHz)
Calidad de sonido (o Bit Depth): 16 bits (i.e., cada muestra se almacena en una variable de 16 bits)
Esto nos da un **bitrate** de 44100*16*2/1024^2 = 1.34 \text Mbit/s si el sonido es estereo. En otras palabras, el sonido almacenado ocupará 1.34/8*60=10.1 megabytes/seg.

En el siguiente código de Matlab u Octave haremos uso de los paquetes "control" y "signal".

Comenzamos leyendo el sonido wav de prueba. En el vector **x** almacenamos la muestra, en **f** la tasa de muestreo, y el **b** la calidad de sonido.

```
[x,f,b] = wavread('gnr.wav');
```

En este caso, el sonido fue grabado con **f**=16000 y **b**=16 y su duración es de **T**=9.2 segundos, lo que da **x**=147000, un sonido bastante aceptable.

Guardamos en X la transformada de coseno discreta de la muestra x

```
X = dct(x);
```

Ahora podemos analizar el sonido es el dominio de frecuencias. Primero contamos cuantos coeficientes de X son nulos.

```
NoNulosAntes=sum(X(:)!=0);
```

Ordenamos los coeficientes para saber cuales tienen más peso

```
[XX,ind] = sort(abs(X),'descend');
```

y nos quedamos con aquellos coeficientes que den, por ejemplo, el **Q**=90 por ciento de la energía total

```
i = 1;
while norm(X(ind(1:need)))/norm(X)<Q
   i++
end;
```

Las frecuencias restantes las eliminamos (las ponemos en 0)

```
X(ind(i+1:end)) = 0;
```

O sea, el sonido comprimido tiene i coeficientes no nulos

```
NoNulosDespues=i;
```

Ahora reconstruimos el sonido que contiene menos frecuencias usando la transformada de coseno discreta inversa

```
X(ind(i+1:end)) = 0;
```

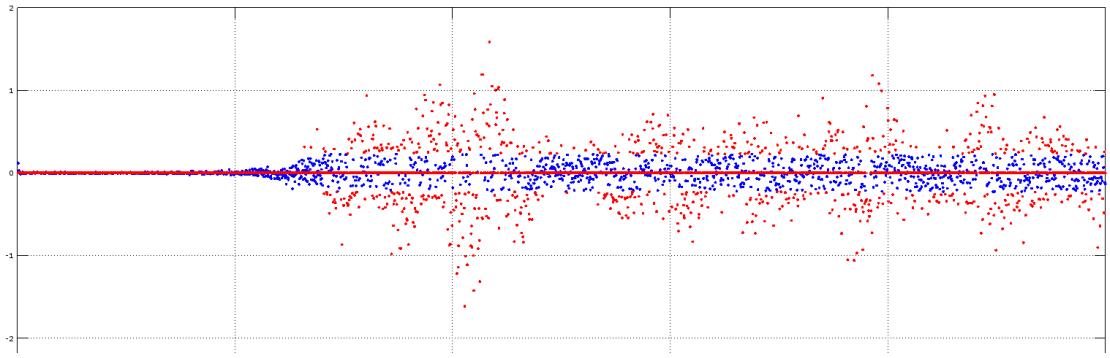
es decir, el sonido comprimido tiene i coeficientes no nulos

```
xx = idct(X);
```

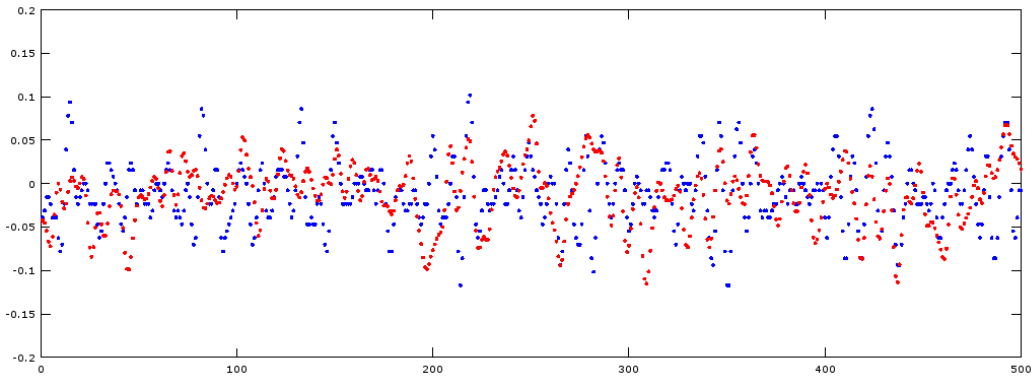
Finalmente empaquetamos los datos en un nuevo wav

```
wavwrite(xx,f,b, 'gnrComprimido.wav');
```

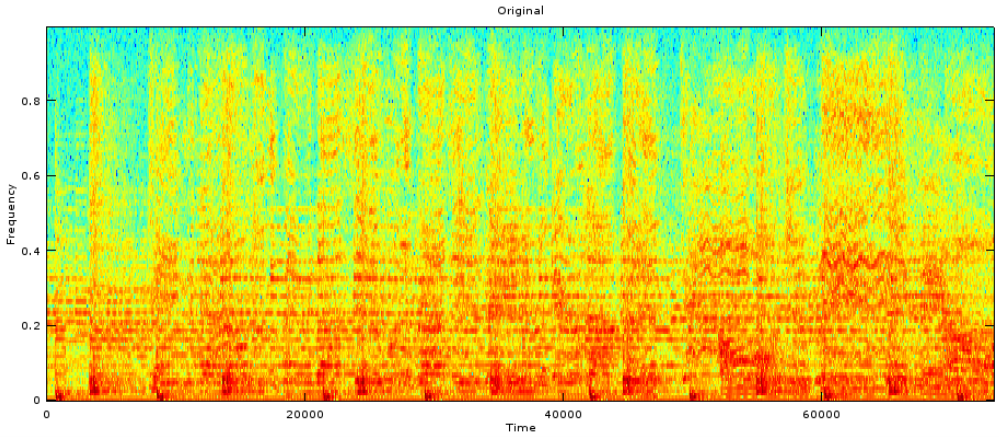
Resultados



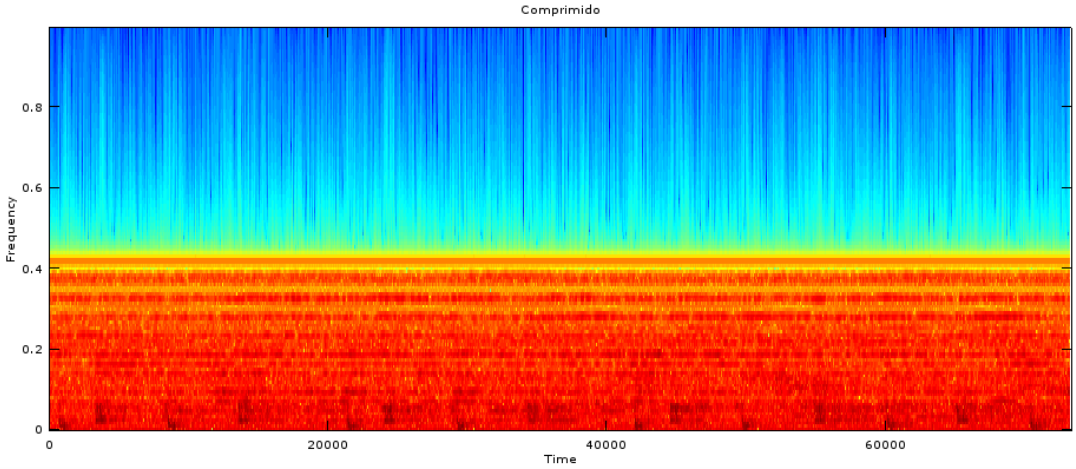
Tomamos **Q=0.8**. Aquí se ven los primeros 2500 coeficientes del vector X, en color rojo se ven aquellos necesarios para alcanzar en 80% de la energía. Todos los puntos azules son descartados.



Con Q=0.8. En azul vemos los valores del sonido original x. En rojo vemos los valores reconstruídos a partir de los coeficientes que bastan para alcanzar el 80% de la energía original.



Histograma del sonido original. Los rojos corresponden a coeficientes altos, bajando con naranja, amarillo, verde, celeste, y azul significa coeficientes nulos.



Histograma del sonido comprimido. Los rojos corresponden a coeficientes altos, bajando con naranja, amarillo, verde, celeste, y azul significa coeficientes nulos.