Trabajo Práctico 2: Análisis Armónico

1. Escala Musical

Existe una variedad de escalas musicales que tienen diferentes orígenes históricos. Una búsqueda breve en Internet le dará al alumno interesado una introducción sobre el tema. En este utilizaremos la escala conocida como de *Temperamento Igual*.

El temperamento igual divide la escala de frecuencias musicales en fragmentos compuestos por doce niveles diferentes que se muestran en la Tabla 1. El símbolos ♯ (se lee *sostenido* en castellano, *sharp* en inglés) y ♭ (se lee *bemol* en castellano, *flat* en inglés) representan los denominados *semitonos*. Cada grupo de doce notas implica duplicar la frecuencia fundamental de la escala (es decir, pasar de una *octava* a la siguiente) y, en el temperamento igual, la relación de frecuencias entre notas sucesivas es siempre la misma: $2^{\frac{1}{12}}$. Es decir que, si la frecuencia de un *do* es 262 Hz, entonces:

- la frecuencia del *do sostenido* que le sigue es $262 \times 2^{\frac{1}{12}} \approx 277{,}6$ Hz;
- la frecuencia del *re* que le sigue es $262 \times 2^{\frac{2}{12}} \approx 294{,}1$ Hz;
- ...
- la frecuencia del *do* de la octava superior es $262 \times 2^{\frac{12}{12}} = 262 \times 2 \approx 524$ Hz.

Cuadro 1: Notas Musicales

Notación Latina	Denominación literal o inglesa
do	C
do♯	C♯
re	D
mi♭	E♭
mi	E
fa	F
fa♯	F♯
sol	G
la♭	A♭
la	A
si♭	B♭
si	B

1. Escriba un programa en el lenguaje de su preferencia (Octave es recomendado) que traduzca un archivo de sonido en una "partitura". Es decir:
   - El programa debe aceptar un archivo de sonido (*.wav* o *.raw*), teniendo como parámetro la frecuencia de muestreo y la cantidad de bits de cada muestra.

- El programa debe analizar cada fragmento de 30 ms aproximadamente (puede variar de manera de lograr un número de muestras igual a una potencia de 2). Mediante la FFT sobre ese fragmento, se debe detectar la nota musical fundamental.
- La nota detectada debe ser traducida a un triplete *letra-alteración-número*:
  - la *letra* corresponde a la nota: A, B, C, D, E, F o G;
  - la *alteración* puede ser "b", "s" o "-" correspondiente a ♭, ♯ o *sin alteración* respectivamente;
  - la *octava* es un número entero que identifica... bueno, eso, la octava; tómese como referencia la *cuarta octava* donde *A-4* identifica un *la* (ver Tabla 1) que tiene una frecuencia fundamental de 440 Hz.
- La salida del programa debe ser una lista de tripletes, la "partitura".

2. Escriba un programa en el lenguaje de su preferencia (Octave es recomendado) que traduzca una lista de tripletes como la producida por el programa anterior en archivo de sonido.

3. Experimentación:

   a) Pruebe el funcionamiento de sus programas con los archivos de música que se encuentran en el sitio de la materia en IOL.

   b) Pruebe el funcionamiento de sus programas con otro archivo de música de su elección.

   c) Discuta los resultados obtenidos.

4. **Opcional - ¡Nota Extra!** ¿Se le ocurre alguna manera de mejorar la calidad de la música producida por sus programas? Explique e implemente las mejoras propuestas.

2. Marcando un número telefónico

Los teléfonos fijan un sistema de tonos para marcar conocido como *Dual Tone Multifrequency* (DTMF). Esencialmente, todos los números del teclado del teléfono están representados por dos tonos o frecuencias, tal como se muestra en la Tabla 2. Las frecuencias fueron elegidas para evitar armónicos: ninguna frecuencia es múltiplo de otra, ni la diferencia ni la suma de cualquier par de frecuencias es igual a otra frecuencia en el conjunto.

Cuadro 2: DTMF

Frecuencias	1209 Hz	1336 Hz	1477 Hz
697 Hz	**1**	**2**	**3**
770 Hz	**4**	**5**	**6**
852 Hz	**7**	**8**	**9**
941 Hz	*****	**0**	#

1. **Codificación DTMF:**

   a) Escriba un programa que, dada una secuencia de números o caracteres especiales * y # produzca el par de tonos correspondientes. Utilice una frecuencia de muestreo de 8 kHz, una duración de tonos de aproximadamente medio segundo y una distancia temporal entre tonos de aproximadamente 0.1 segundos.

   b) Pruebe su programa realizando una llamada telefónica.

2. **Decodificación DTMF:**

   a) *Filtros.* Para poder decodificar una señal DTMF, usaremos un banco de filtros pasabanda. Por "banco de filtros" se entiende a un conjunto de filtros que actúan "en paralelo" sobre la señal. La respuesta al impulso de un filtro pasabanda sencillo viene dada por:

   $$h(n) = \frac{2}{N} \cos\left(\frac{1\pi f_0 n}{f_s}\right), \quad 0 \leq n < N,$$

   donde N es la longitud del filtro, f_s la frecuencia de muestreo y f_0 la frecuencia central de la banda de paso. El ancho de banda del filtro (qué tanto alrededor de f_0 deja pasar) está determinado por la longitud N: a mayor N, menor ancho de banda. Por otro lado, cuanto mayor es la longitud del filtro N, más tiempo es necesario "escuchar" la entrada.

   1) Escriba una función que, dados los valores de N, f_0 y f_s devuelva la respuesta al impulso del filtro pasabanda correspondiente.
      - Grafique la respuesta al impulso para $N = 64$, $f_0 = 770$ Hz y $f_s = 8000$ Hz. En Octave o Matlab, puede utilizar la función `stem`.
      - Repita el ejercicio anterior para $f_0 = 1336$ Hz.

   2) Escriba una función que, dadaos los valores N, f_0 y f_s devuelva la respuesta en frecuencia del filtro pasabanda correspondiente.

- Determine la respuesta en frecuencia del filtro para $N = 64$, $f_0 = 770$ Hz y $f_s = 8000$ Hz.
  - Grafique el módulo de la respuesta en frecuencia.
  - ¿Cuánto vale el módulo de la respuesta en frecuencia para $f = 770$ Hz.
  - Determine el ancho de banda del filtro por la diferencia entre aquellas frecuencias en las cuales el módulo de la respuesta en frecuencia cae a $1\sqrt{2}$.
- Repita el ejercicio anterior con $N = 128$.

Nota: En Octave, puede usar la función `freqz` para determinar la respuesta en frecuencia de un filtro discreto. Dicha función recibe como parámetro un vector de frecuencias angulares (i.e., en rad/s). Utilice un vector de 0 a π con al menos 256 elementos. Luego mapee este vector de frecuencias angulares al espacio de frecuencias original mediante una transformación adecuada (dividiendo por $2\pi/f_s$).

b) **Decisión:** Implemente un programa que, dadas las muestras correspondientes a una tecla presionada, determine la tecla correspondiente. El programa debe proceder de la siguiente manera:

- Filtrar las muestras de la entrada con cada filtro pasabandas.
- Calcular valor medio del cuadrado de la salida de cada filtro como una manera de evaluar qué tan importante es la esa componente del espector de la señal.
- Decidir qué par de tonos son los más importantes a partir de los valores calculados en el paso anterior.

1) Pruebe la eficacia de su programa, pasándole los datos producidos por el generador DTMF que usted mismo implementó. ¿Qué longitud de la respuesta al impulso de los filtros pasabandas resulta más conveniente?
2) **¡Nota Extra!** Extienda su programa para que, recibidas las muestras de una secuencia cualquiera de teclas presionadas, pueda decodificarla correctamente.
3) Pruebe su programa con muestras grabadas de un teléfono fijo.

3. Procesamiento de imágenes I ++

Este ejercicio tiene como objetivo analizar espectralmente una imagen y utilizar la transformada discreta de Fourier para realizar *filtrados espaciales*.

La Trasnformada Discreta de Fourier de una secuencia bidimensional $x_{n,m}$ de $N \times N$ (imagen), es:

$$X_{l,k} = \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} x_{n,m} e^{-i\frac{2\pi}{N}(nl+mk)},$$

siendo la Transformada Inversa Discreta:

$$x_{n,m} = \frac{1}{N^2} \sum_{l=0}^{N-1} \sum_{k=0}^{N-1} X_{l,k} e^{+i\frac{2\pi}{N}(nl+mk)}.$$

1. Implementar un programa que compute la TDF 2D.
2. El archivo `saturno` contiene una matriz de 400×400 pixeles y corresponde a niveles de intensidad luminosa comprendidos entre 0 y 255 (todos enteros). Para visualizar esta imagen en escala de grises, es necesario establecer un mapa de color de 255 niveles. Por ejemplo en MATLAB, se puede leer y visualizar así:

   ```
   x=load('saturno');
   colormap(gray(255));
   image(x');
   ```

   visualizando la Figura 1 que muestra una imagen del planeta Saturno, capturada por la misión Voyager.
3. Computar la Trasnformada discreta de Fourier de la imagen original. Armar las imágenes de 400×400 pixeles correspondientes a la amplitud y la fase. Dichas imágenes deben verse como se muestra en la Figura 2 (Tener en cuenta de mapear los valores de amplitud y fase al intervalo entero $[0, 255]$).
4. Computar la Transformada inversa para reconstruir la imagen original de 400×400 pixeles.
5. Considerar el efecto que producen los siguientes filtros $H_{k,l}$ de 400×400 pixeles en el dominio de las frecuencias (espaciales):
   - $$H_{k,l} = \begin{cases} 0 & \text{si } 0 \geq k \geq 400, 190 \geq l \geq 210, \\ 0 & \text{si } 0 \geq l \geq 400, 190 \geq k \geq 210, \\ 1 & \text{en todo otro caso.} \end{cases}$$
   - El filtro Gaussiano $H_{k,l} = \exp\left(-0{,}1(k^2 + l^2)\right)$.
   - El damero
     $$H_{k,l} = \begin{cases} 0 & \text{si } l + k \text{ es par,} \\ 1 & \text{si } l + k \text{ es impar.} \end{cases}$$

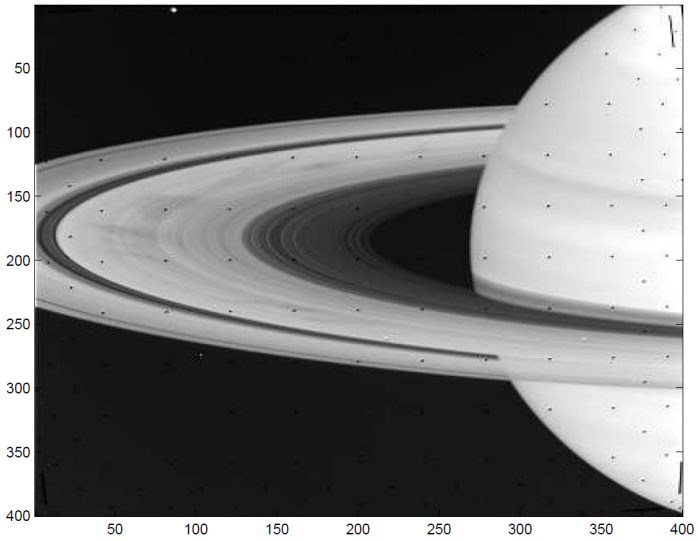

Figura 1: Imagen original

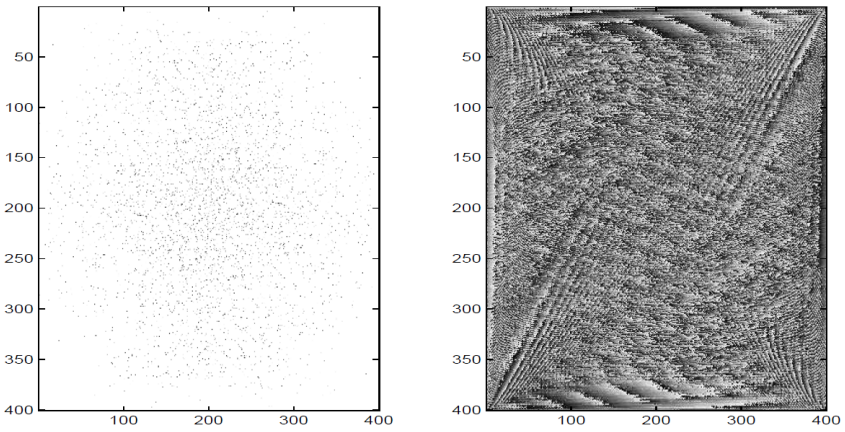

Figura 2: Transformada de Fourier de la imagen original. Izquierda: Amplitud, Derecha: Fase

4. Procesamiento de imágenes II

Fuente básica: Wikipedia.

El estándar de compresión JPEG original (es decir, no el JPEG2000) permite reducir el tamaño en disco de una imagen mediante (por lo menos) dos técnicas de compresión con pérdida:

1. Chroma subsampling: se reduce la resolución espacial de las componentes de color de la imagen.
2. DCT Quantization: se "truncan" las componentes de alta frecuencia de la figura.

En este ejercicio, nos concentraremos en la segunda técnica.

1. El estándar JPEG no trabaja con la imagen en el espacio de color RGB, sino en la codificación equivalente $Y'C_BC_R$, donde Y' representa el brillo de cada pixel (denominado *luma*) y C_B, C_R llevan información de color (la *crominancia*). En el estándar JPEG se utiliza el siguiente "cambio de coordenadas"[1]:

$$\begin{pmatrix} Y' \\ C_B \\ C_R \end{pmatrix} = \begin{pmatrix} +0{,}2990 & +0{,}5870 & +0{,}1140 \\ -0{,}1687 & -0{,}3313 & +0{,}5000 \\ +0{,}5000 & -0{,}4187 & -0{,}0813 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix}$$

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} +1{,}00000 & +0{,}00000 & +1{,}40200 \\ +1{,}00000 & -0{,}34414 & -0{,}71414 \\ +1{,}00000 & +1{,}77200 & +0{,}00000 \end{pmatrix} \begin{pmatrix} Y' \\ C_B - 128 \\ C_R - 128 \end{pmatrix}$$

   a) Demuestre que las matrices en las transformaciones son una la inversa de la otra.

   b) Implemente programas que pasen de una imagen RGB a $Y'C_BC_R$ y al revés.

2. El estándar JPEG divide la imagen en bloques no-solapados de 8×8. Por cada bloque, hay tres matrices, una para Y', otra para C_B y otra para C_R. En caso que de que no haya un número entero de bloques, se debe realizar alguna manganeta para "completar" la imagen. Una posibilidad (no la más elaborada) es repetir las últimas columnas o filas (repetir el patrón del borde). Implemente un programa que divida la imagen en bloques de 8×8 y otro programa que pueda reconstruir la imagen a partir de dichos bloques.

3. Los elementos de las tres matrices resultantes de cada bloque se encuentran en $[0, 255]$. A cada elemento se lo centra restándole 128 y luego cada matriz de 8×8 se procesa aplicándole la Transformada Discreta del Coseno Tipo II (TDC). Esta es una variación de la TDF, pero que sólo utiliza cosenos:

$$X_{l,k} = \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} \sqrt{\frac{\alpha(k)}{N}} \sqrt{\frac{\alpha(l)}{N}} x_{n,m} \cos\left(\frac{\pi}{N}\left(n + \frac{1}{2}l\right)\right) \cos\left(\frac{\pi}{N}\left(m + \frac{1}{2}k\right)\right),$$

[1] Asumimos RGB de 8 bits.

donde $\alpha(k) = 1$ si $k = 0$ y $\alpha(k) = 2$ para todo otro k. En nuestro caso particular, $N = 8$. La transformada inversa (ITDC) es:

$$x_{n,m} = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} \sqrt{\frac{\alpha(k)}{N}} \sqrt{\frac{\alpha(l)}{N}} X_{l,k} \cos\left(\frac{\pi}{N}\left(n + \frac{1}{2}l\right)\right) \cos\left(\frac{\pi}{N}\left(m + \frac{1}{2}k\right)\right),$$

Implemente programas que calculen la TDC y la ITDC de bloques de 8×8. **!Nota Extra!** si se implementa para cualquier N una Transformada Rápida Coseno.

4. El ojo humano es bueno para distinguir pequeñas diferencias de brillo sobre regiones relativamente grandes, pero no puede reconocer fácilmente cambios bruscos de brillo. Por ello es que se realiza una cuantización particular de la transformada discreta coseno de matriz Y' correspondiente a cada bloque. Sea $\mathbf{A}$ la matriz correspondiente a la TDC de la luma de un bloque, $\mathbf{Q}_Y$ la matriz de cuantización correspondiente al brillo y $\mathbf{B}$ la matriz resultante de la cuantización. Luego, se la cuantización se produce de la siguiente forma:

$$\mathbf{B}(k,l) = \text{round}\left(\frac{\mathbf{A}(k,l)}{\mathbf{Q}_Y(k,l)}\right).$$

Un procedimiento similar se utiliza con las matrices correspondientes a la crominancia.

La matriz de cuantización para el brillo especificada por el estándar JPEG original es:

$$\mathbf{Q}_Y = \begin{pmatrix} 16 & 11 & 10 & 16 & 124 & 140 & 151 & 161 \\ 12 & 12 & 14 & 19 & 126 & 158 & 160 & 155 \\ 14 & 13 & 16 & 24 & 140 & 157 & 169 & 156 \\ 14 & 17 & 22 & 29 & 151 & 187 & 180 & 162 \\ 18 & 22 & 37 & 56 & 168 & 109 & 103 & 177 \\ 24 & 35 & 55 & 64 & 181 & 104 & 113 & 192 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 199 \end{pmatrix}$$

La matriz de cuantización para la crominancia es:

$$\mathbf{Q}_C \begin{pmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{pmatrix}$$

Implemente programas que permitan la cuantización y "de-cuantización" de matrices de luma y crominancia.

5. Utilizando los programas anteriores realice las siguientes operaciones sobre la imagen `lenacolor`:

a) Procese la imagen de manera de generar las matrices cuantizadas de la Transformada Discreta Coseno de Y, C_B y C_R para cada bloque de 8×8. ¿Cuál es el porcentaje de elementos de las matrices que queda igual a cero? ¿Cuáles es el menor valor observado? ¿Cuál es el mayor valor?

b) ¿Puede estimar el tamaño en bytes de la imagen "comprimida"?

c) Recupere la imagen luego de la cuantización. Muestre la imagen original junto a la imagen recuperada. ¿Nota alguna diferencia? Muestre la imagen "diferencia" (no se olvide de trasladar la diferencia al intervalo $[0, 255]$).

5. Procesamiento de audio con Fourier

Fuentes:

- **Digital Signal Processing using Matlab and Wavelets** - Michael Weeks
- **Digital Signal Related Website** - http://www.dsprelated.com/dspbooks/
- **Stanford Publications** - https://ccrma.stanford.edu/˜ jos/pubs.html

En la actualidad están apareciendo muchas aplicaciones novedosas que hacen uso del análisis de audio. Por ejemplo un juego de autos que fue revolucionario en el ámbito de los juegos online donde el usuario cargaba una canción de su gusto y el juego armaba una pista de tal forma de que los obstáculos y la velocidad del vehículo aumentaban conforme a la rítmica de la canción cargada. Otro caso famoso es el de **Shazaam**, una aplicación para celulares que utiliza la captura del micrófono para determinar que canción está sonando de fondo. Claramente el desarrollo de estas aplicaciones se encuentra en un ámbito que compete a la ingeniería y más puntualmente a la informática. La cantidad de cosas que se pueden hacer mediante el análisis de audio son incontables: identificador de razas de perros mediante sus ladridos, iluminación automatizada de una banda en vivo captando los distintos golpes de batería, etc.

1. **Ecualizador**

   Hacer uso de la transformada de Fourier para simular el comportamiento de un ecualizador y mostrar la variación, en magnitud, de las frecuencias mediante un gráfico (función plot) en Matlab. Utilizar la captura del micrófono como señal de entrada a intervalos pequeños de tiempo y luego, procesar utilizando la Transformada Rápida de Fourier para obtener las frecuencias. Se recomienda utilizar la función `fftshift` para centrar en cero las frecuencias resultantes y además, expresar el resultado en decibeles utilizando como referencia el umbral de audición humano.

   Se recomienda la utilización de los parámetros:

   - **Frecuencia de muestreo**: $44100Hz$
   - **Precisión de muestreo**: $16bits$
   - **Umbral de audición**: $20\mu Pa$
   - **Canales**: Audio mono (un solo canal).

   Funciones que necesitaría:

   - `fft`: para aplicar a las muestras la transformada rápida de Fourier y obtener las frecuencias.
   - `audiorecorder`: para crear un objeto de audio con los parámetros dados.
   - `recordblocking`: captura la señal de micrófono durante un cierto tiempo dado un objeto de audio
   - `getaudiodata`: obtiene las muestras capturadas por el objeto de audio.

- `pause`: pausa el procesamiento durante un intervalo de tiempo. Necesaria si estamos en un ciclo capturando y ploteando, donde luego del plot necesitamos dar tiempo al sistema a que dibuje en pantalla antes de volver a procesar nuevamente.

2. **Filtrado de frecuencias**

   Elaborar 2 filtros para modificar un archivo de audio:

   a) **Filtro pasa altos**: Recortar las frecuencias mayores a 1500Hz.

   b) **Filtro pasa bajos**: Recortar las frecuencias menores a 500Hz.

   Compruebe empíricamente ambos resultados. Grafique los espectros de frecuencia para la señal de entrada y para las señales de salida, usando `subplot` y `fftshift`. Compare los resultados. Combine ambos filtros y vuelva a graficar. Exprese los gráficos en Hz y dB.

   Funciones que necesitaría:

   - `wavread`: Para leer las muestras del archivo de audio.
   - `fft`: Para aplicar la transformada rápida de Fourier.
   - `fir1`: Para crear el filtro deseado.
   - `filter`: Para aplicar el filtro creado con `fir1` y aplicarlo a las frecuencias resultantes.
   - `wavwrite`: Para guardar los resultados.

3. **Efecto karaoke**

   Teniendo en cuenta lo realizado en los ejercicios anteriores, realice un script en Matlab que dada una canción en formato wav recorte las frecuencias medias más comunes para la voz humana (85 a 255Hz) y emule el efecto karaoke. Para ello utilice las mismas funciones del ejercicio 2 sólo que esta vez cree un filtro pasabanda. Grafique la señal original y la señal de salida y observe el comportamiento del filtro. Haga la prueba con distintas canciones y evalúe en cuales funciona mejor.

4. **Extensiones**

   Considere lo hecho en los ejercicios anteriores y piense como implementaría:

   a) Filtrado de ruido. La señal de ruido maneja frecuencias altas.

   b) Realce de instrumentos. Recuerde que instrumento maneja frecuencias particulares.

   c) Categorizador de cantantes. Considerar el rango de frecuencias para un Soprano, Mezzo-soprano, Contralto, Tenor, Barítono, Bajo.

   d) Afinador de instrumentos.