Práctica 1
Teoría de la Información. Codificación de imagen

Resumen de la práctica

Esta práctica tiene como objetivo comprobar el funcionamiento típico de un codificador con pérdidas. En concreto nos centraremos en el caso de la codificación de imagen y en la práctica siguiente haremos algo similar con un archivo de audio.

1 Introducción

JPEG se enmarca dentro de lo que se conocen como codificadores con pérdidas. Esto quiere decir que la imagen original nunca se va a poder volver a reconstruir a partir de la imagen en formato JPEG, ya que hay partes de la información que se van a eliminar. Esta información se elimina de forma que su impacto visual sobre la imagen sea el menor posible, de forma que, si todo va bien, la imagen JPEG y la imagen original sean prácticamente idénticas.

El algoritmo de JPEG lo que hace es dividir la imagen de entrada en bloques 8x8 píxeles. A continuación, se convierte al dominio transformado cada uno de estos bloques y se cuantifica con un cuantificador diseñado según reglas de percepción de imagen.

2 Codificación de un fichero de imagen

Este apartado tiene como idea complementar lo ya hecho anteriormente con un ejemplo de aplicación real como es la codificación de imágenes. En concreto vamos a pensar en una versión muy simplificada de un codificador JPEG.

Para conseguir todo esto, el algoritmo JPEG hace lo siguiente con la imagen (vamos a pensar por simplicidad que tenemos una imagen en escala de grises):

1. Antes de nada, convertimos los valores de cada pixel de la imagen de formato *entero sin signo* a *entero con signo*. Para un caso como el que vamos a considerar, en el que la imagen viene codificada con 8 bits por muestra, esto implica simplemente restarle $2^7 = 128$ a cada muestra, de forma que los valores pasan de $[0, 255]$ a $[-128, 127]$.

2. Ahora procesamos la imagen en bloques de 8x8 píxeles.

3. Se calcula la Transformada del Coseno (DCT) para cada bloque. La DCT es una variante de la transformada de Fourier en la que las funciones base en lugar de ser exponenciales complejas son cosenos. Esto tiene algunas limitaciones que no vienen al caso, pero también tiene una gran ventaja, y es que los valores de la DCT son reales, por lo que reducimos a la mitad la información a codificar con respecto a una DFT normal.

4. Se cuantifica el bloque transformado utilizando una matriz de cuantificación. El estándar propone una, aunque luego cada fabricante puede utilizar una propia. Si llamamos Q a la matriz de cuantificación y X al bloque obtenido tras la DCT, el proceso es simplemente:

$$X_q = Round\left(\frac{X}{Q}\right) \tag{1}$$

5. Se aplica un proceso de codificación Huffman a la señal cuantificada

Una vez en el decodificador, se siguen los pasos inversos para cada bloque:

1. Hacemos la cuantificación inversa:

$$X_R = X_q \cdot Q \tag{2}$$

2. Calculamos la transformada inversa del coseno (IDCT).

En `https://github.com/enriquealexandre/ComunicacionesDigitales`, podéis encontrar un par de funciones (jpegCod.m y jpegDec.m) que realizan los pasos anteriores (salvo la codificación Huffman), así como una imagen para hacer pruebas. El código utiliza la matriz de cuantificación estándar propuesta en el estándar JPEG, pero se puede jugar aumentando los valores para conseguir una imagen con menor calidad.

Todo esto es lo que tenéis implementado en las funciones *jpegCod.m* y *jpegDec.m*. Os propongo que hagáis lo siguiente:

1. Calcular cuántos bits serían necesarios para codificar la imagen JPEG a la salida del codificador utilizando un número de bits fijo por pixel.
2. Generar un código Huffman adecuado para la imagen.
3. Calcular la ganancia de codificación con ese esquema.
4. Compara el resultado con el que se obtendría utilizando un código Huffman directamente sobre la imagen en bruto, sin trabajar por bloques y sin la transformada DCT ni la cuantificación.

Por si queréis investigar un poco más, en realidad el proceso es algo más complejo, ya que se codifica de forma separada el primer pixel de cada bloque de 8x8 (que tiene un valor muy superior al resto de píxeles del bloque)[1]. El valor de estos píxeles iniciales de cada bloque se codifica de forma diferencial con el del anterior bloque, ahorrando así algún bit extra, y el resto de los píxeles sí que se codifican utilizando un código Huffman, ordenando los valores siguiendo un patron en diagonal.

3 ¿Qué entregar?

- Código de las funciones generadas

[1] Si accedéis desde la UAH, o con VPN, podéis echarle un ojo a este tutorial para haceros una idea: `https://ieeexplore.ieee.org/document/125072`