PRÁCTICA SOBRE TECNOLOGÍAS MULTIMEDIA
RECUERDEN: Esta práctica es obligatoria e INDIVIDUAL.

1. Introducción

Esta práctica tiene como objetivo observar el principal mecanismo de compresión con pérdidas del formato JPEG, esto es la utilización de la transformada discreta coseno (DCT) cuantizada conforme a la prestaciones requeridas. Las explicaciones siguientes se basan en el paquete Octave, aunque no serían necesarios demasiados cambios para desarrollar la práctica en Matlab 4.2 bajo Windows.

2. Objetivo

El resultado que se espera al finalizar la práctica es la construcción de un script que pueda simular la forma en que se representa la imagen en JPEG tras la cuantización de la DCT, así como la reconstrucción de la misma a partir de los datos almacenados. Con este script se pretende estudiar el efecto de utilizar diferentes factores de compresión en la calidad de la imagen, así como en los datos cuantizados.

3. Información preliminar sobre la carga y representación de las imágenes en OCTAVE.

En la práctica vamos a partir de una imagen primaria en formato RGB, donde los niveles de intensidad se establecen usando un rango entre 0 y 255. Cada una de las partes de la imagen (rojo, verde y azul), constituye por tanto una matriz de valores numéricos de la que se dispone. Las imágenes se pueden descargar del ftp de la asignatura, en el directorio denominado jpeg, y son archivos .mat. Al cargarlos en Octave se añadirán al espacio de trabajo las tres matrices que constituyen el formato RGB.

Nota importante: Si el fichero con la imagen es iris.mat, la orden en Octave para cargarlo es:
load –mat-binary iris.mat

Existen, en Octave diferentes funciones de visualización de imágenes, sin embargo se recomienda usar el siguiente procedimiento. Supongamos que R, G y B son las matrices que conforman la imagen. Primero obtenemos la imagen en el formato estándar de Octave, que es un esquema truecolor y por tanto requiere de un mapa de colores:

```
[X,map]=rgb2ind(R/255,G/255,B/255);
```

Rgb2ind devuelve la imagen en X y el mapa de colores en map. Nótese que es necesario dividir las matrices por 255 para normalizarlas entre 0 y 1. Para visualizar la imagen, lo primero es establecer el mapa de colores con:

```
colormap(map);
```

y a continuación visualizar la imagen con:

```
imagesc(X,f)
```

La función imagesc representa la imagen con un zoom dado por f. Si f= 1 la imagen se ve a su tamaño normal, si f= 2 se ve al doble, etc...

4. Información sobre el estándar JPEG

Recordemos que las primeras fases del estándar JPEG consisten en:

1. Para cada matriz R, G y B calcularemos la DCT en bloques de 8x8 píxeles. Si la matriz no tuviera un número de filas o columnas divisible por 8 completaremos la matriz repitiendo el número de filas y columnas que fuese necesario hasta que sea divisible por 8.
2. Si el bloque de 8x8 píxeles es la matriz x, la transformación se puede realizar mediante la operación: xt= T * x * T', donde T es la matriz asociada a la DCT de 8x8. Esta matriz, se puede descargar también del ftp de la asignatura y se llama dctmtx8.mat. También se debe cargar en Octave con la instrucción load –mat-binary dctmtx8.mat.

3. Una vez todos los bloques de 8x8 de cada una de las imágenes han sido transformados, pasaremos a simular el proceso de cuantización. La cuantización uniforme de la magnitud variable x, lo asimilaremos a la operación:

   xc=round(x/alfa)

   donde alfa es cierto número. Es fácil de entender que si alfa=1, el proceso de cuantización se produce por redondeo al entero más cercano, mientras que a medida que alfa es mayor el número de niveles de cuantización decrece.

   En el estándar JPEG se pretende utilizar de forma más eficiente posible el número de niveles de cuantización por lo que se asigna un divisor base diferente para cada coeficiente de la DCT. Esto nos da 3 matrices de 64 divisores precalculadas experimentalmente para cada color R, G, y B. Estas matrices de divisores base se pueden obtener a partir de las funciones quantR.m , quantG.m y quantB.m, que también se pueden descargar del ftp de las asignatura. Estas matrices de divisores no se aplican directamente sobre los bloque de 8x8 resultantes de la DCT, sino que se modifican en función del factor de compresión requerido.

   Es decir, si xt es un bloque transformado, el correspondiente bloque cuantizado es:

   xtc=round(xt./mask)

   donde la matriz de 8x8 mask se obtiene a partir de quantR, quantG o quantB según sea el caso a partir de la siguiente expresión:

   sea fac el factor de compresión entre (0<fac<=100, 100 indica codificación sin pérdidas), entonces:

   si fac<50

      escala= 5000/fac

   en otro caso,

      escala = 200 – fac * 2

   y finalmente

     mask = (quant * escala + 50)/100, siendo quant la matriz de divisores base quantR, quantG o quantB que corresponda.

4. El proceso de reconstrucción de la señal a partir de los datos cuantizados es justamente el inverso. Primero obtenemos los coeficientes de la DCT para cada bloque de 8*8. Para esto hacemos:

   xtr= mask.* xtc

5. Ahora hacemos la transformada coseno inversa de cada bloque. Esto se puede hacer con la misma matriz T que se usó en la transformación directa, esta vez:

   xr= T' * xtr * T

   Esta matriz reconstruida debe redondearse y controlar aquellos valores que sean menores que 0 (se reasignan a 0) y mayores que 255 (se reasignan a 255).

5. Pasos recomendados para construir el script de procesamiento.

1º. Construir una función que complete filas y columnas en otra matriz hasta que tanto el número de filas y columnas sea divisible por 8.
2º Construir una función que calcule la DCT (T*x*T') por bloques de 8x8.

3º. Modificar la función anterior para obtener otra que calcule la DCT inversa (T'*x*T) en bloques de 8x8.
4º. Modificar la función anterior para obtener otra que calcule round(x./mask) en bloques de 8x8, siendo mask otra matriz de 8x8.
5º. Modificar la función anterior para obtener otra que calcule (mask.*x) en bloques de 8x8, siendo mask otra matriz de 8x8.
6º. Escribir una función que calcule la matriz de divisores (máscara) en función del factor de compresión requerido (ver explicación sobre la cuantización más arriba).
7º. Escribir una función que busque en una matriz aquellos elementos mayores de 255 y los sustituya por 255 y los elementos menores que 0 y los sustituya por 0.
8º. Construir un script, donde se cargue el fichero .mat con la imagen que se desea utilizar y que realice todo el proceso. Para ir depurando poco a poco se recomienda usar inicialmente un factor de compresión de 100 con lo que la imagen original y la reconstruida tras haber sido codificada con JPEG deberán ser cuasi idénticas.

6. **Trabajo de análisis a realizar con el script de procesamiento JPEG.**

Aplicar el proceso de codificación – decodificación sobre diferentes imágenes (archivos .mat del directorio) y observar la diferencia de calidad en la imagen al utilizar diferentes factores de compresión.