Grado en Ingeniería de Tecnologías de Telecomunicación
Fundamentos de Sonido e Imagen
Bloque 2: Fundamentos de Imagen

Práctica 2 - Modelado de imagen y video

Objetivos

El objetivo de la práctica consiste en profundizar en los conceptos fundamentales sobre las señales utilizadas para la representación de imagen y video con Matlab. Para ello se programarán las rutinas necesarias para obtener información de una secuencia de video y visualizarla con diferentes modificaciones. Por otra parte, se trabajará sobre las propiedades espectrales de la señal temporal generada a partir de dicha secuencia de video. En una segunda parte se explorarán los efectos del muestreo y la cuantificación en las imágenes digitales y se llevará a cabo una toma de contacto con técnicas básicas de procesado de imagen.

1. Lectura de archivos de video y visualización

En esta parte de la práctica se pretende familiarizarnos con las herramientas que proporciona MATLAB para manipular videos. Para ello realice lo siguiente:

1. Utilice la orden `mmfileinfo` para obtener información sobre el video de ejemplo proporcionado en el campus virtual "testVideo.mpg". Proporcione detalles sobre la información obtenida.

2. Cree un objeto de tipo `VideoReader` para leer el video de ejemplo. Para ello basta con teclear `v=VideoReader('nombre_de_archivo')`. Examine el objeto creado y presente en pantalla las propiedades del objeto que le permitan conocer: 1) El número de bits por píxel; 2) La duración del video; 3) El *frame rate*; 4) El número de *frames*; ¿Qué otras propiedades del video puede obtener del objeto?

3. Para visualizar el video, debe crear una estructura de tipo película y leer los frames de video de forma secuencial. Dicha estructura consistirá en una secuencia temporal en la que almacenaremos los *frames* en un campo y el mapa de colores en otro. El siguiente fragmento de código le permitirá crear la mencionada estructura y visualizar el video, además de comprobar la duración y el número de frames.

```
%% Lectura y reproducción de un video en formato mpeg
% Precarga la estructura de datos.
vidObj = VideoReader('testVideo.mpg');
s=struct('cdata', zeros(vidObj.Height, vidObj.Width,...
    3, 'uint8'),'colormap', []);
numFrames = 0;
Duration=vidObj.CurrentTime;
while hasFrame(vidObj)
numFrames = numFrames + 1;
s(numFrames).cdata = readFrame(vidObj);
Duration=vidObj.CurrentTime;
end
% Visualización del video
```

```
13          implay(s,vidObj.FrameRate)
14          Duration
15          numFrames
```

4. Modifique el código anterior para:

   a) Visualizar la secuencia de video a cámara lenta y rápida.

   b) Visualizar la misma secuencia en sentido inverso.

   c) Visualizar la misma secuencia con los *frames* rotados 90°. La orden `imrotate` le puede resultar de utilidad.

2. Espectro unidimensional de la señal de videofrecuencia

En este ejercicio realizaremos un experimento sencillo para visualizar la forma del espectro de la señal de videofrecuencia. Para ello, utilizaremos $n = 10$ *frames* del video adquirido en el primer ejercicio.

1. Comience utilizando la secuencia sin animación (cada *frame* corresponderá a la primera imagen de la secuencia, sin movimiento). Genere un vector fila a partir de la lectura línea a línea de los n frames de la secuencia de video. Este vector representará la señal de videofrecuencia. Para realizar esta lectura secuencial puede hacer uso del comando `im2col`.

2. Para la frecuencia de imagen del video de partida, y teniendo en cuenta el tamaño de los *frames*, calcule la frecuencia de línea y la frecuencia de muestreo (f_s) correspondiente (1 muestra=1 píxel).

3. Represente la señal frente al tiempo utilizando un vector de tiempos obtenido a partir de esta frecuencia de muestreo.

4. Obtenga el espectro de la señal mediante el comando `fft` y visualice su valor absoluto en el intervalo $[-f_s/2, f_s/2)$. Ajuste el rango de visualización para comprobar qué ocurre en múltiplos enteros de la frecuencia de línea. ¿Coincide la posición de las líneas espectrales con la esperada?

5. Repita el experimento con la secuencia animada y explique las diferencias encontradas. Haga lo mismo para $n = 200$ *frames*.

3. Muestreo y cuantificación

En este ejercicio se analizarán los efectos del muestreo y la cuantificación en imágenes digitales. Comenzaremos con una imagen en escala de grises para después comprobar cómo afectan estas operaciones a los diferentes canales de una imagen en color.

1. Cargue la imagen *cameraman.tif* y simule un submuestreo de la imagen con diferentes tasas. Evalúe la calidad visual frente al tamaño de la imagen. ¿A partir de qué tasa de submuestreo la calidad se ve afectada de forma significativa?. Para simular el submuestreo puede seleccionar simplemente los píxeles correspondientes o promediar localmente simulando el efecto del sensor de adquisición. Para realizar esta última operación, la orden `blockproc` puede resultarle útil.

2. Repita el ejercicio anterior variando ahora los niveles de cuantificación de la imagen.

3. Cargue la imagen *peppers.png*, y pásela a formato YUV (puede utilizar el comando `rgb2ycbcr`). Una vez allí submuestree el espacio de color, dejando información de luminancia. Compruebe las variaciones en la percepción de la imagen. Debe tener en cuenta que al submuestrear sólo algunos de los canales, deberá cambiar el tamaño de dichos canales para representar la imagen. El comando `imresize` con sus diferentes opciones de interpolación le puede resultar útil. Además, tenga en cuenta que debe volver al espacio origen para la visualización. Repita el ejercicio submuestreando la luminancia y dejando inalterado el color.

4. De manera análoga, trabaje con la imagen en formato RGB submuestreando uno de los canales. Compruebe cuál es el más invariante a submuestreo (en términos de percepción).

4. Filtrado de imágenes

1. A partir de una imagen `I` en escala de grises calcule su media local mediante la convolución (en el dominio espacial) con una máscara 5x5:

```
h = ones([5,5])/25;
If = filter2(h,I);
```

2. Degrade la imagen utilizando la orden `imnoise` con ruido "salt & pepper" de densidad 0.05. Vuelva a filtrarla y observe el resultado.

3. Utilice la orden `imgaussfilt` para filtrar la imagen degradada con una máscara gaussiana con `sigma` igual a 2. Compare los resultados. Varíe el valor de `sigma` hasta conseguir prestaciones similares.

4. Utilice `medfilt2` para aplicar un filtro de mediana a la imagen degradada en los dos apartados anteriores del mismo tamaño que la máscara inicial. Compare los resultados.

5. Repita los tres ejercicios anteriores para ruido gaussiano de potencia 0.05.

6. Utilice la orden `imsharpen` para realzar los bordes de la imagen inicial. Fije los parámetros "Amount" a 1.2 y "Radius" a 1.5. Explique para qué sirven estos parámetros.

7. Compare el resultado anterior con el de filtrar con una máscara paso alto de tamaño 7×7. Para visualizar correctamente la imagen, debe añadir a la imagen filtrada paso alto, la imagen original.

5. Realce de contraste. Ecualización de histograma

En este ejercicio vamos a aplicar un procesado sencillo para ecualizar el histograma de la imagen utilizando funciones específicas de MATLAB. De forma adicional, realizaremos la transformación matemática equivalente desde 0 para conseguir el mismo efecto.

1. Comience cargando la imagen "pout.tif' y represéntela conjuntamente junto con su histograma. Comente la distribución del mismo. Puede utilizar la orden `imhist` para extraer dicho histograma. Utilice 64 bins.

2. Aplique una ecualización de histograma mediante la función `histeq`. Represente la imagen ecualizada junto a su histograma y compare con el caso anterior.

3. Obtenga la curva de transformación de la ecualización y represéntela en una figura aparte. La ayuda de las funciones que acaba de utilizar le puede resultar útil.

4. Realice la misma transformación sin utilizar la orden `histeq`. Para ello:

   a) Obtenga la salida de `imhist` para 256 bins y normalícela al tamaño de la imagen.

   b) Utilice `cumsum` para aproximar la función de distribución tras haber normalizado.

   c) Transforme la salida de `cumsum` a un mapa de niveles de gris.

   d) Convierta la imagen original a una imagen indexada de 256 niveles.

   e) Represente la imagen indexada con el nuevo mapa transformado.

5. Represente la nueva transformación en la misma gráfica que la representada anteriormente.

6. Modulación de histograma. Aplique una transformación a la imagen de manera que el histograma de la imagen de salida sea el correspondiente a una variable aleatoria $\mathbf{X}$ cuya función de densidad sea $f_{\mathbf{X}}(x) = 2x$ para $0 \leq x \leq 1$, y nula fuera de ese intervalo.