Práctica Tema 5

Manejo de vídeos en MATLAB. Detección y estimación de movimiento.

Curso 2019-2020

En la primera parte de esta práctica se pretende familiarizar al alumno con la manipulación básica de vídeos en MATLAB, para ello se trabajará con el archivo '`shopping_center.mpg`' que acompaña al material de esta práctica. En la segunda parte, se pretende explorar algoritmos de estimación de movimiento basada en bloques, para ello se utilizarán los archivos 'foreman69.Y' y 'formeman72.Y' y las funciones 'ebma.m', 'hbma.m' y 'PhaseCorrelation.m'.

I. Manejo básico de vídeo en MATLAB

La función `VideoReader` toma como parámetro el nombre de un vídeo y devuelve la información sobre ese vídeo. Lea la información sobre '`shopping_center.mpg`' y sálvela en una variable.

```
file_name = 'shopping_center.mpg';
my_movie= VideoReader(file_name);
```

1. ¿Qué información proporciona la función `VideoReader` sobre este vídeo?

La función `VideoReader` nos permite cargar un archivo de vídeo en el *workspace* de MATLAB. La información sobre cada *frame* se puede guardar en dos campos: `cdata`, que son los datos de la imagen y `colormap`, que guarda el mapa de colores de `cdata`, si la imagen es truecolor, este campo aparece en blanco:

```
s_movie =
struct('cdata',zeros(my_movie.Height,my_movie.Width,3,'uint8'),'colo
rmap',[]);

k = 1;
while my_movie.hasFrame
      s_movie(k).cdata = readFrame(my_movie);
      k = k+1;
 end
```

2. ¿Cuál es el tamaño de la estructura `s_movie`? ¿Por qué tiene ese tamaño?

Inspeccione el primer frame the la estructura `s_movie`.

```
s_movie(1);
```

Visualice el primer frame como una imagen.

```
imshow(s_movie(1).cdata);
```

También se pueden cargar *frames* individuales especificando los números de los *frames* como segundo parámetro:

```
frame_nums = [5 20];
my_movie2 = read(my_movie, frame_nums);
```

3. ¿Cuál es el tamaño de my_movie2?

Se pueden visualizar todos los *frames* simultáneamente con la función `montage`.

```
%Preasignar array de 4D que contendrá todos los frames

image_data=uint8(zeros(my_movie.Height,my_movie.Width,3,...
file_info.NumFrames));

for k = 1:file_info.NumFrames
image_data(:,:,:,k) = s_movie(k).cdata;

end
montage(image_data)
```

4. Explique cómo se almacenan los datos de cada frame en la variable `image_data`.

Reproduzca el vídeo (`s_movie`) con la función `implay` utilizando los parámetros por defecto.

5. ¿Cuál es la tasa de *frame* por defecto?

Reproduzca el vídeo 5 veces con una tasa de 30 fps (utilice una sola instrucción).

6. ¿Qué instrucción ha utilizado?

Para realizar operaciones sobre *frames* individuales, se puede utilizar la función `frame2im`, que convierte un frame a una imagen. Después se pueden utilizar las herramientas de procesado de imagen. Convierta el frame 10 a imagen.

```
old_img = frame2im(s_movie(10));
```

Desefoque la imagen con un filtro de media

```
fn = fspecial('average',7);
new_img = imfilter(old_img, fn);
```

Represente ambas imágenes en una figura.
Utilizando otro *frame* obtenga el negativo de la imagen y represente el resultado.

```
image_neg = imadjust(old_img2, [0 1], [1 0]);
```

A continuación convierta las imágenes que ha procesado a *frames* y guárdelas en la estructura de vídeo.

```
my_movie_new = my_movie;
new_frame10 = im2frame(new_img);
my_movie_new(10) = new_frame10;
```

Guarde también el *frame* negativo en la misma estructura (`my_movie_new`).

Utilice de nuevo la función `movie` para reproducir el nuevo vídeo.

7. ¿Cuál de las dos es más perceptible durante la reproducción?

A partir de una matriz de imágenes, se puede construir una estructura de vídeo con la función `immovie`. Cree una estructura de vídeo a partir de una matriz de imágenes negativas del vídeo original.

```
my_imgs = uint8(zeros(my_movie, my_movie.Width,3, ...
my_movie.NumFrames));
for i = 1:my_movie.NumFrames
...
...
end
```

8. Complete las líneas de código necesarias dentro del bucle. ¿Cuáles son?

Ahora construya una estructura de vídeo con `immovie` y reprodúzcala.

Utilice la función `VideoWriter` para convertir la estructura de vídeo que ha creado a un vídeo AVI. Compruebe que se ha guardado en el directorio en el que se encuentra.

9. ¿Qué parámetros le ha pasado a la función `VideoWriter`?

II. Estimación de movimiento basada en bloques

II.1 EBMA

En primer lugar va a trabajar con un algoritmo de estimación de movimiento basado en bloques exhaustivo, *Exhaustive Block Matching Algorithm* (EBMA).

Realice la siguiente definición de variables:

```
anchorName = 'foreman69.Y';
targetName = 'foreman72.Y';
frameHeight = 352;
frameWidth = 288;
blockSize = [16,16];
```

Lea los *frames* de datos referencia y objetivo:

```
fid = fopen(anchorName,'r');
anchorFrame = fread(fid,[frameHeight,frameWidth]);
anchorFrame = anchorFrame';
fclose(fid);
fid = fopen(targetName,'r');
targetFrame = fread(fid,[frameHeight,frameWidth]);
targetFrame = targetFrame';
fclose(fid);
```

Represente los *frames* referencia y objetivo (para representarlos debe convertirlos primero al tipo de datos uint8).

Ejecute la función `ebma.m`. Esta función recibe como parámetros los *frames* de referencia y objetivo, así como el tamaño de los bloques. Devuelve el *frame* estimado y los vectores de movimiento.
La variable `time_ebma` guarda el tiempo que se ha tardado en procesar los *frames.*

```
tic
[predictedFrame, mv_d, mv_o] = ...
ebma(targetFrame, anchorFrame, blockSize);
time_ebma = toc
```

Represente los vectores de movimiento superpuestos al *frame* referencia.

```
figure, imshow(uint8(anchorFrame))
hold on
quiver(mv_o(1,:),mv_o(2,:),mv_d(1,:),mv_d(2,:)), ...
title('Vectores de movimiento EBMA');
hold off
```

Represente el *frame* estimado.

Calcule el error cuadrático medio (MSE) de estimación.

```
errorFrame = imabsdiff(anchorFrame, predictedFrame_Full);
MSE_ebma = mean(mean((errorFrame.^2)));
```

Vuelva a obtener los *frames* estimados y los MSE para tamaños de bloque de 8x8 píxeles y 32x32 píxeles.

1. ¿Cuál es la relación entre el tamaño de los bloques y el número de vectores de movimiento obtenidos?
2. ¿Cuál es la relación entre el tamaño de los bloques y el tiempo de procesado?
3. ¿Cuál es la relación entre el tamaño de los bloques y el MSE?

II.2 HBMA

A continuación va a trabajar con un algoritmo de estimación de movimiento basado en bloques de resolución múltiple, *Hierarchical Block Matching Algorithm* (HBMA).

Ejecute la función `hbma.m`. Esta función recibe como parámetros los *frames* de referencia y objetivo, el tamaño de los bloques y el número de niveles de búsqueda. Devuelve el *frame* estimado y los vectores de movimiento.

```
blockSize = [16,16];
%numero de niveles
L = 3;

tic
[predictedFrame,mv_d,mv_o]=hbma(targetFrame,anchorFrame,blockSize,L);
time_ebma = toc
```

Represente los vectores de movimiento superpuestos al *frame* referencia.

Represente el *frame* estimado.

Calcule el error cuadrático medio (MSE) de estimación.

1. Compare los vectores de movimiento obtenidos con EBMA y HBMA (para el mismo tamaño de bloques).
2. Compare la calidad de los *frames* estimados con EBMA y HBMA (para el mismo tamaño de bloques).

II.3 Phase Correlation Method

Por último explorará una técnica de estimación de movimiento en el dominio espectral, *Phase Correlation Method.*

Realice las siguientes deficiniciones:

```
frame(:,:,1) = anchorFrame;
frame(:,:,2) = targetFrame;
```

Ejecute el script `PhaseCorrelation.m` que genera el *frame* estimado, la representación de la función de correlación de fase y la representación de los vectores de movimiento.

1. Compare la calidad del *frame* estimado con este método con la calidad de los *frames* estimados con los métodos EBMA y HBMA.
2. ¿Qué representan los picos en la función de correlación de fase? ¿Por qué hay picos de mayor amplitud en el centro?