Práctica 9

Mejora y restauración de vídeo.

Filtrado *interframe*

Curso 2019-2020

El objetivo de esta práctica es familiarizar al alumno con la utilización del filtrado *interframe* para la mejora y restauración de vídeos en MATLAB, para ello se trabajará con el archivo `rhinos.avi` y las funciones `addnoise.m` y `tempNoiseFilter.m` que acompañan al material de esta práctica.

I. Mejora y Restauración de vídeo. Filtrado de promediado temporal.

Lea el archivo `mthr_dotr.yuv` utilizando la función `readYUV`. Utilice el comando *hasFrame* para extraer los *frames* de la secuencia de vídeo. Vamos a realizar el filtrado en el espacio de color RGB.

1. ¿Cuántos *frames* tiene la estructura `mov`?

Reproduzca el vídeo.

2. ¿Tiene buena calidad? ¿se aprecia ruido?

Va a contaminar la secuencia `mov` con ruido para comprobar después el funcionamiento dos tipos de filtrado *interframe* sencillos. Para apreciar la degradación que se va a provocar, extraiga el primer *frame* de la secuencia `mov` y cree dos nuevas imágenes, una contaminada con ruido tipo "sal y pimienta" y otra con ruido de tipo gaussiano, utilice para ello la función `imnoise`.

Represente las tres imágenes, la original, la contaminada con ruido "sal y pimienta" y la contaminada con ruido gaussiano.

La función `addnoise.m` aplica ruido a los *frames* que se le indiquen como parámetro y devuelve una nueva estructura de vídeo contaminada. Utilice esta función para obtener dos nuevas estructuras de vídeo, una contaminada con ruido de tipo gaussiano y otra contaminada con ruido de tipo sal y pimienta (en ambos casos introduzca ruido en todos los *frames* de la secuencia `mov`). Reproduzca las secuencias resultantes.

La función `tempNoiseFilter` realiza el filtrado temporal de la secuencia de vídeo que recibe como parámetro, promediando el número de *frames* que se le indiquen. El promediado temporal se realiza asignando el mismo peso a todos los *frames.*
Filtre las secuencias contaminadas en el paso anterior utilizando 3 *frames* en el promediado. Reproduzca las secuencias resultantes.

3. ¿Por qué en los dos primeros *frames* no se ha eliminado nada de ruido? ¿qué se le ocurre que se podría hacer para filtrar los primeros *frames* que quedan sin filtrar en el filtrado temporal?
4. ¿Se ha conseguido eliminar el ruido en el resto de *frames* de la secuencia?

Aumente a 6 el número de *frames* que se utilizan para realizar el promediado temporal. Ahora serán 5 los *frames* que se quedan sin filtrar al inicio de la secuencia.

5. ¿Cuál es el efecto de esta modificación en el filtrado del **resto** de la secuencia?
6. ¿Cuál es la diferencia, que provoca el aumento del número de *frames* en el promediado, entre las zonas con y sin movimiento de la secuencia?
   Para observar estas diferencias abra diferentes figuras para reproducir los vídeos contaminado, filtrado con 3 *frames* de promediado y filtrado con 6 *frames* de promediado. Una vez detenidos los vídeos puede comparar el último frame de cada uno de los tres casos. Hágalo para el ruido gaussiano y para el ruido sal y pimienta. Incluya como figuras en la memoria estos *frames* (6 *frames* en total).

II. Mejora y Restauración de vídeo. Filtrado de orden estadístico.

Modifique ahora la rutina `tempNoiseFilter` para que en lugar de realizar un filtrado de promediado temporal de los datos, realice un filtrado de orden estadístico simple como el que describe la siguiente ecuación,

$$\hat{f}(n,k) = median(g(n,k-1), g(n,k), g(n,k+1))$$

donde n representa la posición de un pixel, k representa el número de frame, g representa la secuencia de *frames* corruptos.

Nota: en este caso el número de *frames* que recibirá como parámetro `tempNoiseFilter` será siempre 3.

Pruebe este filtrado con los dos tipos de ruido.

1. ¿Qué ocurre en este caso con el último *frame* en las secuencias filtradas?¿por qué?
2. Compare la calidad de los resultados utilizando este filtrado y el filtrado de promediado temporal con 3 *frames*.