Memoria Señales

Adrián Lara Velasco

2023-11-22

Contents

Compresión de video utilizando Wavelets 1
Separar el video en frames 1
Aplicación de la transformada para cada frame 2
Volver a unir los frames para obtener el video comprimido 3

Compresión de video utilizando Wavelets

La compresión de video utilizando wavelets sigue la evolución natural de nuestro proyecto donde hemos iniciado aplicando esta transformada para una imagen y ahora lo hacemos para múltiples imagenes ya que un video no es más que una sucesión de imagenes o frames consecutivos uno detras de otro. Por este motivo se nos ocurrió la forma más intuitiva de comprimir un video mediante wavelets que se resume en lo siguiente:

1. Separar el video en frames
2. Aplicación de la transformada para cada frame
3. Volver a unir los frames para obtener el video comprimido

A continuación nos dedicaremos a explicar cada paso del proceso de forma detallada.

Separar el video en frames

Para ilustrar todo el proceso utilizamos el archivo llamado `baile.mp4`, aunque se podría utilizar cualquier archivo de video de tipo **.mp4**. El códec de video del video es **H264**, *verémos que es importante despúes*, el peso del archivo es de **1.17 MB**. Nuestro objetivo es tratar de comprimir el video lo máximo posible tratando de comprometer la calidad lo mínimo posible.

Mediante el paquete open-cv de Python, podemos dividir el video en frames, en este caso, el video se convierte en 344 frames de peso **7,45 MB**, esto podría parecer poco intuitivo, lo lógico es pensar que si dividimos un video en frames, la suma de estos debería ser igual al peso del video, sin embargo, a los videos se les aplican códecs que comprimen el tamaño de los frames sin apenas influir en la calidad, por lo tanto, al obtener los frames individuales, se descomprimen y pesan más que el video. Esto nos indica que cuando queramos comparar el video original con el que se le ha aplicado la transformación, deben estar en el mismo códec, en este caso **H264**.

Los frames resultantes se almacenan en una carpeta en el directorio `./data/frames_video_original`. El tipo de archivo de las imagenes es **JPEG**.

Aplicación de la transformada para cada frame

Para aplicar la transformada, como para otras partes del proyecto, se utiliza el paquete **pywavelets** entre otros. En primer lugar creamos una nueva carpeta para almacenar los que serán los frames transformados, la ruta es `./data/frames_modificados`, luego de forma iterativa vamos aplicando el siguiente proceso a cada frame:

1. Aplicamos la transformada wavelet, en este caso debauchies, *db4 en python*, y cuantos niveles de descomposición en aproximación y detalles queremos, para el ejemplo hemos utilizado 3. No ha sido necesario utilizar los frames en escala de grises dado que pywavelets opera correctamente con imagenes a color.
2. Se obtienen los coeficientes de aproximación y los de detalles. De estos, solo nos quedamos con el último coeficiente de aproximación y con los coeficientes de detalle en el mismo nivel y por encima de los del coeficiente de aproximación.
3. Creamos un threshold adaptativo que consiste en utilizar un percentil como límite, todo valor por debajo de ese percentil se le asigna el valor 0, mientras que aquellos valores por encima del umbral no sufren cambios. Estos son los valores de los coeficientes que mayor información aportan sobre la imagen. Le aplicamos el umbral a los coeficientes de detalle que están en el mismo nivel que los de aproximación mientras que el resto de detalles no se les aplica dicho threshold.
4. Realizamos la transformada wavelet inversa con los nuevos coeficientes de aproximación y detalle que hemos obtenido, reconstruyendo así la imagen ya transformada, aplicando la wavelet debauchies.
5. Guardamos los frames dentro de la carpeta directorio.

Figure 1: Primer frame del video

Figure 2: Primer frame del video reconstruido

Volver a unir los frames para obtener el video comprimido

Una vez tenemos los frames transformados, realizamos el proceso de reconversión a video. Establecemos el directorio de salida del que será el video resultante, `./data/video_salida.mp4`, hay que establecer el número de frames por segundo (fps) del video, el estándar son 30 fps. La librería **open-cv** de Python, no permite utilizar el códec **H264**, por lo que inicialmente, el video se codifica con el tipo **XVID** y luego mediante el uso de la libreria **moviepy**, lo recodificamos a **H264** de nuevo. El resultado final se encuentra en `./data/video_convertido.mp4`, el peso del nuevo video es de **655 KB** por lo que al final se aprecia una reducción significativa del tamaño del archivo. En cuanto a la pérdida de calidad, apenas se perciben cambios visuales entre los videos.