S3-Python & Video part II

Ejercicio 1

Primero he extraído 10 segundos del vídeo original BBB.mp4 y he creado un nuevo vídeo con el que trabajaré, llamado BBB_short.mp4. Con esto me ahorraré bastante tiempo de computación.

El vídeo original está en 1080p. Ahora lo pasaré a 720p, 480p, 360x240 y 160x120 como hice en el seminario 2 y lo guardaré en la carpeta *videos_ex1*.

Todo el código que he hecho hasta ahora lo comentaré para que no se ejecute cada vez.

El vídeo original tiene el códec H.264.

Voy a explicar brevemente algunas de las características de los cuatro códecs:

- **Códec VP8:** El sucesor de VP8, VP9, proporciona una mejor calidad de vídeo con una tasa de bits inferior.
- **Códec VP9:** libvpx-vp9 puede ahorrar alrededor de un 20-50% de Bitrate en comparación con libx264 (el codificador H.264 por defecto), manteniendo la misma calidad visual.
- **Códec AV1:** Dependiendo del caso de uso, AV1 puede lograr una eficiencia de compresión un 30% mayor que VP9, y un 50% mayor que H.264
- **Códec H265:** H.265 puede ofrecer un ahorro de entre el 25 y el 50% de la tasa de bits en comparación con el vídeo H.264 codificado con libx264, manteniendo la misma calidad visual. Estas ganancias serán más pronunciadas en resoluciones de 1080p y superiores.

He usado ffmpeg para crear cuatro vídeos con distinto códec (VP8, VP9, AV1 y H.265) para cada una de las cuatro resoluciones. Usando el códec VP8 he tenido que guardar los vídeos en formato webm en vez de mp4 para evitar errores.

Después de cada conversión, he usado ffprove para imprimir la información del nuevo vídeo y así saber información como el Bitrate.

Voy a hacer la comparación con los distintos códecs con el vídeo en 480p (BBB_480.mp4):

- El vídeo original tiene un "bit_rate": "1207125". Su tamaño es de 1'44 MB.

- **Códec VP8:** Con el Bitrate por defecto ("bit_rate": "541308") obtengo un fichero con tamaño 663 KB, pero con una calidad bastante mala. Obtengo un mejor resultado con un "bit_rate": "1205478", pero el tamaño es de 1'44 MB, como el vídeo original.

  En general, si queremos conseguir la misma calidad que el vídeo original usando este códec, el vídeo resultante será más pesado.

- **Códec VP9:** Con el Bitrate por defecto ("bit_rate": "1302069") obtengo un fichero con tamaño 1'55 MB con una calidad como la del vídeo original.

  He probado con un Bitrate de 2 Mbps, pero ocupa casi el doble y no se nota la mejora.

  En todas las resoluciones he obtenido un fichero con tamaño un poco mayor al del vídeo original, usando el Bitrate por defecto. Probablemente, si probara con Bitrates menores podría obtener un vídeo menos pesado con una visualización decente.

- **Códec AV1:** Por defecto obtengo un "bit_rate": "1057345" y un vídeo de 1'26 MB con la misma calidad que el vídeo original. Este códec da muy buen resultado ya que obtenemos la misma calidad con menos tamaño de fichero.

  El problema de este códec es que tarda mucho en ejecutarse. Para los vídeos de 720p y 480p tardó bastantes minutos

- **Códec H265:** He probado con los valores por defecto y he obtenido un "bit_rate": "687857" y un fichero con calidad parecida al vídeo original y 842 KB de tamaño.

  Con mayor resolución, por ejemplo, con el vídeo en 720p, obtengo la misma calidad de visualización por menos de la mitad del tamaño. Por lo tanto, según los resultados que he obtenido, este códec es el que mejor funciona.

Los cuatro vídeos resultantes para cada resolución son bastantes parecidos. Esto lo he hecho a propósito, cambiando los Bitrates en cada caso que fuera necesario para obtener una visualización parecida a la del vídeo original.

He juntado los cuatro vídeos de cada resolución y los he guardado como BBB_720_collage.mp4.

La diferencia entre los vídeos no se ve tan bien con el collage porque los vídeos se vuelven más pequeños y no se pueden apreciar tan bien las diferencias. Por eso he hecho la comparación anterior viendo los vídeos en tamaño original.

Ejercicio 2

Usando tkinter he creado un programa para convertir un vídeo introducido por el usuario a cualquier de los cuatro códecs del ejercicio anterior.

Para ello, he creado una ventana con dos controles para que el usuario introduzca la dirección del vídeo de entrada y la del vídeo de salida.

Después he creado los cuatro botones para los códecs y sus cuatro funciones correspondientes para llevar a cabo la conversión.

Por ejemplo, si queremos convertir un vídeo a VP9 podemos introducir en la etiqueta *'Insert video path to convert:'* la dirección Videos_ex1/480p/BBB_480.mp4, y en la etiqueta *'Insert output video path:'* la dirección Videos_ex2/480p/BBB_480_vp9.mp4. Entonces, si le damos al botón *'Convert to VP9'* se guardará la conversión en VP9 en la carpeta Videos_ex2, que he creado para no mezclar los vídeos con los del ejercicio anterior.