Creando valor para los archivos de vídeo. Cómo recodificar y adaptar los vídeos de forma masiva utilizando herramientas open source

Creating value for video archives. How to massively recode and adapt videos using open-source tools

Carlos Turró, Luis Morcillo, Jaime Busquets

Resumen

La constante renovación de tecnologías de video digital siempre ha provocado una continua necesidad de migración y adaptación a nuevos formatos. Recientemente, se han producido diversos avances tecnológicos que han posibilitado la difusión de video digital más allá de los convencionales estándares 'broadcasting'.

Entre el video de alta definición a través de internet y los videos para dispositivos móviles, existe un amplio espectro de nuevos formatos y soportes digitales. Es por lo tanto necesario desarrollar mecanismos para automatizar la edición, post-producción y re-codificación de los contenidos para que se ajusten a dichos formatos.

La Universidad Politécnica de Valencia expone en el siguiente artículo las herramientas utilizadas y el proceso desarrollado en el 'Área de Información y Comunicaciones' para automatizar la edición y conversión de su repositorio de video digital a distintos formatos.

La constante renovación de tecnologías de video digital siempre ha provocado una continua necesidad de migración y adaptación a nuevos formatos

Summary

The continuous update of digital video technologies has always needed the migration and adaptation to new formats. Recently, there have been several technological advances that enable the delivery of digital video beyond the conventional broadcasting standards.

Between high-definition video trough the Internet and videos for mobile devices, there is a wide range of new formats and Medias. It is therefore necessary to develop mechanisms to automate the video-editing, the post-production and re-encoding process in order to conform the content to those new formats.

In the following article the Universidad Politécnica de Valencia outlines the tools used and the process developed in the 'Área de Sistemas de Información y Comunicaciones' that make possible to automate the editing and conversion of its digital video repository to several formats.

Es necesario desarrollar mecanismos para automatizar la edición, post-producción y re-codificación de los contenidos para que se ajusten a dichos formatos

Keywords: digital video, automatic editing, encoding, Avisynth

1. Introducción

Actualmente es muy frecuente que cualquier organismo público o privado (juzgados, hospitales, academias, universidades,...) cuente con una base de datos con su material audiovisual en formato digital para su distribución en internet. Así es el caso de la UPV que actualmente cuenta con distintos servicios con contenidos en diversos formatos:

TABLA 1. Contenidos de video digital UPV (Junio del 2008) AVIP

Servicio	Nº videos	formato
Polimedia	1889	Flash &Wmv (streaming)
Politube	433	Mp4 (progressive download)
UPVRTV	2392	wmv (streamin & p. download)
Total	**4715**	

En caso de querer únicamente re-codificar los videos, son muchas las herramientas tanto libres como propietarias que permiten automatizar la codificación a distintos formatos de una forma sencilla y automática.

En otros casos, es necesario aplicar postproducción sobre cada uno de los videos, por lo que una simple re-codificación no es suficiente. Ejemplos de estas situaciones pueden ser: añadir una logo corporativo, escalar el tamaño de los videos, aplicar técnicas de picture-in-picture, eliminar entrelazados de vídeo, etcétera.

Este proceso, dado la cantidad de contenidos, hace muy costosa la postproducción manual y aunque algunos de los editores de video no lineales permiten cierto grado de automatización y procesado por lotes, no llegan a ser tan versátiles como procesos de edición de video automática basados en secuencias de comandos.

Avisynth funciona como un editor de video no lineal, sin interfaz, controlado totalmente por scripts

Avisynth (1) es una herramienta open-source de edición y posproducción de vídeo que resulta especialmente adaptada para estas labores; además al estar basada en comandos permite obtener toda la potencia requerida para las operaciones anteriormente mencionadas. A continuación haremos una somera presentación de las potencialidades de Avisynth, veremos su aplicación a los casos antes comentados y finalmente veremos los resultados del uso de esta herramienta en la Universidad Politécnica de Valencia.

1.1. Avisynth

Avisynth funciona como un editor de video no lineal, sin interfaz, controlado totalmente por scripts. Funciona como un intermediario entre un video y un programa receptor, que puede ser un reproductor, software de edición de video, codificador, ... Para ello se comunica con cualquier programa utilizando un "falso" video, que al abrirse ejecuta los comandos definidos por Avisynth. Avisynth actúa de forma transparente al receptor, el cual, trata la imagen resultante como un video totalmente normal.

Funciona como un intermediario entre un video y un programa receptor

Veamos un ejemplo sencillo: Tenemos un vídeo estándar, denominado video.avi . Para verlo, la operación que realizamos es abrirlo con el media player. Utilizando Avisynth haremos la siguiente operación:

Crearemos un fichero ejemplo.avs con el siguiente contenido:

DirectShowSource("ejemplo.avi")

Y abrimos desde el media-player el fichero *ejemplo.avs.*
Ahora, cada vez que el media-player solicita un frame del vídeo, visynth abre un frame del contenido, de forma que el vídeo se reproduce de la misma forma. Ya estamos en condiciones de hacer uso de la potencia de Avisynth. Pongamos algunos ejemplos:

1) Para invertir un vídeo de izquierda a derecha, editamos ejemplo.avs y dejamos

DirectShowSource("ejemplo.avi")
FlipHorizontal()

2) Podemos crear de esta misma forma una imagen doble:

v=DirectShowSource("ejemplo.avi")
return StackHorizontal(v,FlipHorizontal(v))

3) En el siguiente ejemplo, añadimos un logo logo.png a un vídeo, con una transparencia del 50%. Ya que estamos, le añadimos un efecto de entrada y salida suave con los comandos fade-in y fade-out:

video=DirectShowSource("ejemplo2.avi")
video=fadein(video,25)
video=fadeout(video,25)
logo=ImageSource("logo.png",0,video.framecount, video.framerate)
video=Overlay(video,logo,video.width-logo.width, video.height-logo.height,0.5)return video

Se puede añadir un logo logo.png a un vídeo, con una transparencia del 50%

4) Creamos un picture-in-picture con dos vídeos video1.avi y video2.avi

video1=DirectShowSource("video1.avi")
video2=DirectShowSource("video2.avi")
return Overlay(video1,video2,video1.width-video2.width, video1.height-video2.height)

Reducimos un vídeo en tamaño y en frames por segundo, para codificarlo para móviles

5) Reducimos un vídeo en tamaño y en frames por segundo, para codificarlo para móviles.

video=DirectShowSource("ejemplo3.avi")
video2=ConvertFPS(video,15)
video3=LanzclosResize(video2,320,240) return video3

2. Ejecutando una recodificación masiva con Avisynth para H.264. Poniendo manos a la obra

A la hora de hacer una re-codificación masiva con Avisynth hay que tener en cuenta dos elementos: primero, como se ha visto en los ejemplos anteriores, el nombre del vídeo va embebido en el fichero de entrada, lo que es una restricción del sistema, con lo que tenemos que generar el archivo de entrada al vuelo. Segundo, tendremos que llamar al archivo avisynth con la utilidad de codificación que se desee, por ejemplo, utilizando ffmpeg -i input -o output, podríamos hacer la siguiente utilidad.bat

echo video=DirectShowSource("%1") >temp.avs
echo video2=ConvertFPS(video,15) >>temp.avs
echo video3=LanzclosResize(video2,320,240) >> temp.avs
echo return video3 >> temp.avs
ffmpeg -i temp.avs -o video_convertido.avi
del /y temp.avs

Vamos a explicar cómo codificamos el formato Mpeg4 /h264

El proceso que se describirá más adelante funciona para cualquier formato de destino, pero aprovechamos la ocasión para exponer el códec que hemos elegido basándonos en nuestra experiencia a lo largo de estos últimos años.

En nuestro caso concreto vamos a explicar cómo codificamos el formato Mpeg4 /h264. Hemos elegido este formato, por ser compatible con una amplia gama de dispositivos móviles (IPod incluidos) y por sus buenos resultados de compresión respecto a sus predecesores. Además, es compatible con distintos media-players y recientemente ha aparecido una versión de Adobe Flash Player 9.0.115 (2) que soporta dicho códec. Esta nueva versión permite su difusión por Internet incluso por streaming con Flash Media Server 3 y posiciona a dicho formato como posible líder en la eterna batalla de codecs para video por internet.

La versión permite su difusión por Internet incluso por streaming con Flash Media Server 3 y posiciona a dicho formato como posible líder en la eterna batalla de codecs para video por internet

Para realizar el proceso y dicha codificación por secuencias de comandos utilizamos las siguientes herramientas libres:

X264: Es una biblioteca libre y gratuita para codificar videos desarrollada por Videolan llamada x.264. (3)

FFMPEG: es una herramienta libre de línea de comandos para convertir un videos o audio de un formato a otro, utilizaremos esta para codificar el audio a formato AAC. (4)

MP4BOX: un multiplexor de Mp4, puede importar videos en mpeg-4 y audio a un "container" MP4. El archivo resultante es compatible con la norma "ISO compliant MPEG-4 streams".

AVISYNTH: Nuestra librería favorita.

Y con esto, la parte de codificación para H.264 compatible con Flash queda como sigue:

```
ffmpeg -i "script.avs" -ab 100 -vn "output.m4a"
```

Audio:

```
x264.exe --pass 1 --bitrate 1000 --stats "script.stats" --level 3 --no-cabac --
subme 1 --partitions none --vbv-bufsize 1000 --vbv-maxrate 10000 --me dia --threads
auto --thread-input --progress --no-psnr --no-ssim --output NUL "script.avs"
```

Video primera pasada

```
x264.exe --pass 2 --bitrate 1000 --stats "script.stats" --level 3 --no-cabac --
subme 6 --partitions p8x8,b8x8,i4x4 --vbv-bufsize 1000 --vbv-maxrate 10000 --
threads auto --thread-input --progress --no-psnr --no-ssim --output "script.264"
"script.avs"
```

Video segunda pasada

```
mp4box.exe -add "script.264" -add "script.m4a" -fps 25  -new "script.mp4"
```

Multiplexar

2.1. Resultados prácticos

Como se ha mencionado en la introducción, este desarrollo lo hemos realizado para actualizar los contenidos de vídeo de diversos formatos (WMV, FLV) a H.264, maquetar los vídeos tipo picture-in-picture y ejecutar algunas operaciones de post-procesado, fundamentalmente añadir los logos, realizar chroma-key y ajustar los niveles de vídeo para nuestros 4715 vídeos (5), que equivalen a unos 15 días de contenido.

Este desarrollo lo hemos realizado para actualizar los contenidos de vídeo de diversos formatos, maquetar los vídeos tipo picture-in-picture y ejecutar algunas operaciones de post-procesado

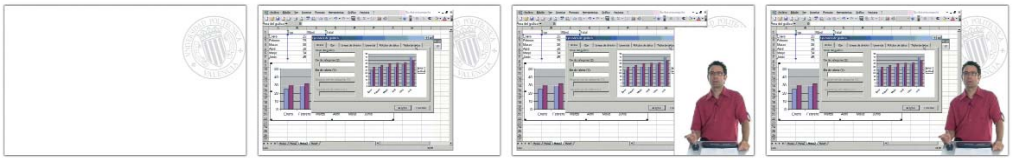
3. Diseño modular y Red UNED

Cada aula AVIP puede definirse como un módulo dentro de la red global. Así, se pueden seguir añadiendo aulas AVIP al sistema, controlando únicamente de la carga de trabajo de la MCU - que es escalable únicamente añadiendo nuevos dispositivos al backbone del nodo de conmutación de videoconferencia.

Cada aula AVIP puede definirse como un módulo dentro de la red global

FIGURA 1. VIDEOS DE ORIGEN Y RESULTADO

FIGURA 2. PROCESO DE INTEGRACIÓN



Avisynth en sí mismo es bastante rápido

Aquí hay que notar que Avisynth en sí mismo es bastante rápido, en nuestras pruebas hemos llevado a cabo procesos complejos y ha aguantado perfectamente el ritmo de 25 frames por segundo en un PC aceptable. Sin embargo el codificador x264 es extremadamente lento, obteniendo velocidades de 0.5 a 0.25 veces la velocidad de vídeo. Por ello, para hacer la re-codificación masiva hemos paralelizado el proceso sobre un cluster de máquinas, aprovechando que la única tarea a realizar es dividir el script de codificación.

Los resultados de la codificación han sido completamente exitosos y ya disponemos de nuestros contenidos actualizados al nuevo códec H.264 compatible con los nuevos reproductores de vídeo.

4. Bibliografía

[1] AviSynth. http://www.avisynth.org.

[2] Hassoun, David. Exploring Flash Player support for high-definition H.264 video. [Online] http://www.adobe.com/devnet/flashplayer/articles/hd_video_flash_player.html.

[3] x264. http://www.videolan.org/developers/x264.html.

[4] FFmpeg. [Online] http://ffmpeg.mplayerhq.hu/.

[5] Polimedia. http://polimedia.upv.es/polimedia.v2/.

Para hacer la re-codificación masiva hemos paralelizado el proceso sobre un cluster de máquinas, aprovechando que la única tarea a realizar es dividir el script de codificación

Carlos Turró
turro@asic.upv.es

Luis Morcillo
luimormu@asic.upv.es

Jaime Busquets
busquets@asic.upv.es

Universidad Politécnica de Valencia