

## White paper: Desarrollo de un formato de video

Autor: Ramiro A. Gómez

Web: [www.peiper.com.ar](http://www.peiper.com.ar)

### ¿Qué es un formato de video?

Este mes decidimos tratar un tema muy importante del mundo del multimedia: los videos. No nos enfocaremos en los distintos formatos que hay disponibles, en vez de eso vamos a explicar las distintas técnicas que existen para lograr un formato de video de gran calidad.

Por otra parte, si estás pensando en implementar un formato de video o de imagen este white paper te será de gran utilidad.

Un formato de video es un tipo de archivo que contiene secuencias de imágenes que se muestran de forma fluida (24, 30 o más cuadros por segundo), es decir, video.



Ejemplo de captura de video DivX.

### Requisitos para lograr un buen formato

Un buen formato de video no es sencillo de lograr. De hecho, los formatos más populares fueron desarrollados por expertos mundiales en la materia, organismos de estandarización e investigadores. Lograr contener toda la secuencia de video en un diminuto archivo no es una tarea fácil, ya que no se trata sólo de comprimir las secuencias de bytes que representan los colores. Hay todo un conjunto de técnicas que permiten aumentar los ratios de compresión drásticamente.

Un buen formato de video debe:

- tener buena calidad de imagen y sonido
- ocupar poco espacio de almacenamiento
- ser liviano para ejecutar

- ser portable (que se pueda visualizar en distintas plataformas)
- ser lo más simple posible

Debemos aceptar que lo que presentamos arriba es muy difícil de lograr, y es por eso que los videos introducen cierta pérdida de datos. Para lograr mejores resultados los expertos optan por perder los datos que menos captan los sentidos humanos, como ser colores demasiado similares, sonidos con muy altas o muy bajas frecuencias, etc.

### **Representación del color**

El color se puede representar de muchas maneras. Una de ellas es la representación RGB.

RGB son las siglas de los 3 colores usados (Red, Green, Blue). Cada componente de color rojo, verde y azul se representa por una secuencia de 8 bits, o lo que es lo mismo, 1 byte. Con 8 bits podemos lograr 256 combinaciones para cada color.

Los tres componentes suman 3 bytes o 24 bits. Y todas las combinaciones posibles que se pueden realizar son  $256^3=16.777.216$ . Esa es la cantidad de colores que podemos formar con 24 bits.

Otra representación del color muy útil es HSV (Hue, Saturation,



Value). Es muy útil porque maneja los colores de manera más intuitiva. Se compone de tono (hue), saturación y valor. El tono indica el color puro elegido en un arco de color cuyos componentes puros son verde, amarillo, rojo, magenta, azul y cian. La saturación nos indica qué tan puro es el color y el valor es la claridad.

Para desarrollar un formato de video se debe elegir la representación más conveniente, es decir, la que sea más fiel al original y permita la mejor compresión.

En muchos formatos se utiliza HSV y se comprime más la crominancia (H, tono de color) que la luminancia (V, cantidad de luz). Es así porque el ojo humano percibe más las variaciones de luminancia que de tono.

Al descartar ciertos datos de los componentes de color estamos perdiendo información que tal vez no podremos recuperar. Es por eso que, en general, los formatos de video son con pérdida de información. Perdemos información y calidad pero ganamos en compresión. La habilidad del desarrollador está en encontrar la mejor configuración que no pierda demasiada calidad pero que logre una aceptable compresión.

### **Representación del audio**

Al igual que las imágenes, el audio se representa con bits.

Los sonidos son ondas, y las propiedades de las ondas son frecuencia o velocidad de muestreo, amplitud y fase. Queremos decir que son vibraciones del aire o medio (líquidos, sólidos o gases) que se desplazan por el medio a una velocidad aproximada de 300 metros/seg. (en el aire). De acuerdo a como vibre el medio nuestros oídos perciben estas vibraciones y nuestro cerebro las interpreta. En el vacío no hay sonido.

Al ser ondas, el sonido es analógico. Por esta razón debemos digitalizarlo si lo queremos representar con bits. El sonido digital tiene 2 propiedades muy importantes:

- la tasa de muestreo: se mide en Hertz (Hz.) o ciclos por segundo y es la cantidad de muestras del sonido que se toman por segundo.
- el bitrate: es la tasa de bits/segundo que tiene cada muestras de sonido

Para incluirlo en los videos, el audio se comprime usando distintas técnicas basadas en la entropía y redundancia de los datos (ver nuestro white paper de [Compresión](#)). Otra técnica que se usa comunmente es la eliminación de los sonidos que no percibe el oído humano (de muy alta o muy baja frecuencia). El oído humano percibe los sonidos en el rango de frecuencias 20 a 20.000 Hz.

### **Representación de los subtítulos**

Muchas veces un video está en otro idioma y es necesario contar con la traducción, que se muestra como texto en la pantalla. Podemos optar por 2 opciones:

1. No separar los subtítulos del video y tomarlo todo como imágenes.

2. Disponer de texto dentro del archivo que se agregue automáticamente cuando se visualiza el video.

La primera opción tiene algunas desventajas:

- No podemos separar los subtítulos de los videos y se debe tener un archivo de video completo por idioma.
- Pueden producirse pérdidas de calidad en el texto porque está codificado de la misma manera que el video.

Las ventajas de la segunda opción son lo opuesto a las desventajas de la primera.

- No necesitamos tener un mismo video varias veces con distintos idiomas, podríamos tener el archivo de video por un lado y los subtítulos en otros archivos. El visualizador se encarga de mostrarlos juntos de la manera correcta.
- No se producen pérdidas de calidad en los subtítulos.
- Se puede comprimir el texto con algoritmos de compresión de texto, disminuyendo el tamaño del video.

### **Otros elementos que pueden contener los videos**

Los formatos de video de última generación son mucho más que secuencias de imágenes y audio. Además de estos se pueden incluir gráficos vectoriales, sprites, texto, efectos especiales, escenas 3D interactivas, etc.

### **Submuestreo**

No es necesario en algunos casos almacenar toda la información del video. Una técnica muy útil pero que produce cierta pérdida de calidad es el submuestreo.

El submuestreo consiste en almacenar sólo el un parámetro del color cada cierto intervalo. Lo más común es usarlo con esquemas de color HSV (YUV). Los valores de crominancia y de saturación de cada píxel no los almacenamos todos, en vez de eso tomamos el color cada 2 o 4 píxeles (o n píxeles) o un promedio del segmento de píxeles. Los valores de luminancia los almacenamos todos.

Como el ojo humano es más sensible a los cambios de intensidad que de color, este proceso nos asegura mínima pérdida de calidad.

Por eso, cuando se dice un submuestre 4:4:4 significa que se toman todas las muestras HSV de cada píxel. Si tenemos 4:2:2 sólo se toman completos los valores de luminancia y cada 2 píxeles los restantes.

Por último, un submuestreo 4:2:0 indica que no se toman muestras de saturación, cada 2 píxeles se toma un muestra de crominancia y se toma la luminancia de todos los píxeles.

### **Redundancia espacial y temporal**

Los videos, así como las imágenes, presentan redundancias. Los datos tienen una relación muy estrecha, sobretodo los datos de los píxeles adyacentes de un mismo cuadro.

La redundancia espacial significa que los píxeles cercanos a otro en el mismo cuadro posiblemente tengan un color similar. Si el video muestra una hoja verde, muchos de los píxeles adyacentes tendrán tonos verdes parecidos, si se muestra el cielo muchos tendrán tonos celestes. No queremos decir que los colores adyacentes sean exactamente iguales al color del pixel, pero tienen cierta relación.

La redundancia espacial se tiene en cuenta a la hora de comprimir los datos porque, como vimos en nuestro white paper de "[Compresión](#)", ésta se sustenta en los datos redundantes.

La redundancia temporal se refiere a la similitud o relación que existe entre píxeles de un cuadro y otros antes o después de éste. Demos un ejemplo para entenderlo: Un video a 24 cuadros/segundo de 1 hora de duración NO mostrará imágenes totalmente distintas en cada cuadro durante toda la hora. En muchas partes del video los objetos se moverán lentamente de un cuadro a otro. Este movimiento se puede almacenar con valores numéricos que indican el desplazamiento (estimación de movimiento) o con predicciones que luego se corrigen.

Existen 3 tipos de cuadros que se clasifican de acuerdo a la predicción que se realiza:

- Cuadros exactos: se almacenan exactamente y no dependen de otros para reconstruirlos. Sirven de base para realizar predicciones de los cuadros anteriores o siguientes.
- Cuadros interpolados: se reconstruyen a través de interpolaciones entre dos cuadros exactos.
- Cuadros extrapolados: se reconstruyen a través de extrapolaciones de otros cuadros.

De esta manera, en una secuencia de video que presenta movimientos suaves sólo harán falta unos pocos cuadros exactos y los demás se reconstruyen a través de técnicas de interpolación y extrapolación (ver "Cálculo numérico").

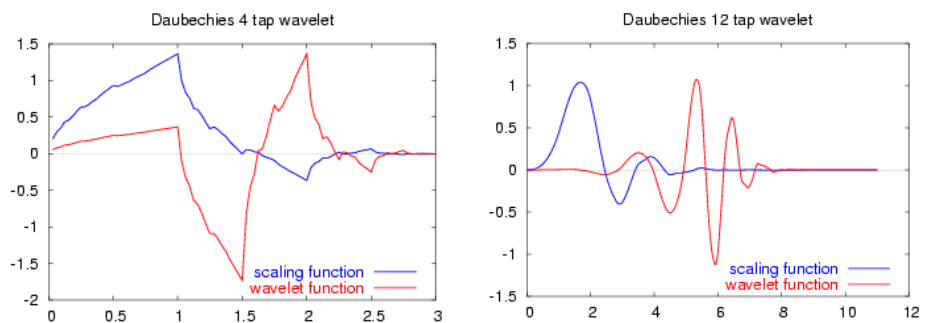
### **Transformaciones basadas en ondas y wavelets**

Se ha descubierto hace tiempo que una secuencia de datos se puede representar aproximadamente con ondas superpuestas. Cada una de estas ondas tienen distintas amplitudes, fases y frecuencias. Sumándolas o restándolas de una manera correcta obtenemos una onda final que se ajusta aproximadamente a los datos que deseamos

representar.

El algoritmo más conocido para realizar este proceso se conoce como DCT (discrete cosine transformation, transformación discreta del coseno). Se sustenta en utilizar ondas de la función coseno para representar un conjunto de datos numéricos. Estos datos son los colores de la imagen, que se pueden elegir de cualquier manera. En general se usan grillas de 8x8 o 4x4 píxeles. Este proceso produce ciertos "artifacts" en las imágenes reconstruidas, el color de los bloques adyacentes difiere en cierta medida, lo que permite distinguirlos cuando el cuadro está muy comprimido.

La DCT está basada en una disciplina matemática llamada Análisis de Fourier.



Extendiendo y mejorando la idea de ondas, hace un tiempo se está incursionando en un nuevo tipo de ondas no uniformes ni simétricas llamadas wavelets. El concepto de wavelets está desarrollado matemáticamente, lo que nos indica que puede resultar intrincado comprenderlo si no tenemos una sólida formación matemática y afinidad por la simbología. Pero la idea no es difícil.

Se trata de representar una secuencia de datos a través de ondas complejas, asimétricas o simétricas, no uniformes. Estas ondas se pueden combinar de la mejor manera para representar nuestra secuencia de datos, se las puede estirar o comprimir, agrandar, achicar, etc. Para realizar estas operaciones se parte de una onda básica llamada "wavelet madre". Hay muchos tipos de wavelets madre, algunos con cambios abruptos y otros con suaves curvas.

Ejemplo: si disponemos de una secuencia de datos

$$\{ 20, 30, 35, 40, 35, 32, 25, 20 \}$$

la podemos representar aproximadamente con una parábola invertida que en el punto  $X = 0$ ,  $f(x) = 20$ ; en  $X = 1$ ,  $f(x) = 30$ , etc.

Pero no sólo podemos utilizar parábolas. Hay muchas wavelets madres que sirven como referencia para aproximar una secuencia numérica.

## Cuantificación

El proceso de cuantificación es quizás el que produce mayores pérdidas de información, lo que es equivalente a perder calidad en el video. Sin embargo, permite comprimir más las secuencias de imágenes.

Existe toda una formulación matemática sobre este apartado, pero siguiendo con la filosofía de Peiper, lo explicaremos lo más sencillo posible.

Se trata de dividir cada muestra de color por un número mayor que 1. O en otra forma, multiplicar cada dato por un número flotante entre 0 y 1. Como obtenemos un número decimal y es muy costoso almacenarlo, lo redondeamos al entero más próximo.

Si usamos la división, cuanto mayor sea el número por el que dividimos más comprimiremos la imagen, pero al restaurar el valor se perderá más calidad.

Ejemplo: Supongamos que tenemos los valores de luminancia

100, 131, 156, 203

Dividimos cada uno de estos números por 16 ( o cualquier otro número)

$$100/16 = 6,25 \Rightarrow 6$$

$$131/16 = 8,19 \Rightarrow 8$$

$$156/16 = 9,75 \Rightarrow 10$$

$$203/16 = 12,69 \Rightarrow 13$$

Con este proceso obtenemos luego los datos:

6, 8, 10, 13

que son números más pequeños y por eso más "compresibles".

A la hora de obtener la imagen original se hace el proceso inverso, se multiplica cada uno por 16.

$$6 * 16 = 96 \text{ (error: } 100 - 96 = 4)$$

$$8 * 16 = 128 \text{ (error: } 131 - 128 = 3)$$

$$10 * 16 = 160 \text{ (error: } 160 - 156 = 4)$$

$$13 * 16 = 208 \text{ (error: } 208 - 203 = 5)$$

Como ven este proceso produce ciertos errores, que generalmente se llaman ruidos de cuantificación.

Cuanto más grande sea el número por el que dividimos los datos más podremos comprimir, pero mayores errores obtendremos.

Este proceso también elimina los cambios de frecuencia muy rápidos en un rango pequeño. Por ejemplo, si tenemos los datos 100, 98, 101, 99, 101, 97, luego de cuantificarlos se podrán regenerar al mismo número. Si los dividimos por 16 obtenemos:

6, 6, 6, 6, 6, 6.

Una alternativa novedosa es la cuantificación adaptativa. Esta usa diferentes matrices de cuantificación para cada situación. Añade complejidad al desarrollo del formato pero logra mejor calidad y/o compresión.

### **Comprimiendo los datos**

Una vez que se tienen los datos procesados es común aplicarles compresión sin pérdida. Los datos están procesados luego de representarlos como redundancias, aplicarles DCT o wavelets, transformar los colores a otros sistemas de representación, etc.

La compresión en esta etapa debe ser sin pérdida porque algún dato mal regenerado afectará las regeneraciones posteriores. Como éstos representan aspectos de desplazamiento espacial, temporal, o predicción de valores; si se reconstruyen mal luego de la descompresión la reconstrucción de la imagen posiblemente tendrá severos errores.

Se pueden aplicar métodos de compresión de Huffman, compresión aritmética, RLE, con diccionarios, técnicas predictivas, etc. (tal como vimos en nuestro white paper sobre "[Compresión](#)").

### **Post-procesamiento**

Una vez que procesamos el video para almacenarlo en el formato propiamente dicho podemos decir que ya concluyó nuestro trabajo.

Pero si lo que queremos es lograr la mejor calidad de imagen, tendremos que realizar post-procesamiento.

El post-procesamiento consiste en aplicar distintos filtros para mejorar el contraste, el brillo, la saturación, las tonalidades, los posibles artifacts que se generen por realizar las transformaciones, etc.

Debemos destacar que algunas de estas tareas ya las realizan las placas de video más avanzadas, con tecnologías como Avivo o PureVideo.

Una opción para lograr correcciones de posibles pérdidas de continuidad de las formas de las imágenes es utilizar curvas que se ajusten a un conjunto de datos mediante interpolación. Algunos tipos de curvas muy útiles y versátiles son los SPLINES, NURBS y curvas Bézier.



Ejemplo (qué ejemplo!) de un cuadro de video DivX

### **Caso de estudio 1: el formato de video DivX**

Cuenta la leyenda que el códec de video DivX nació de la mano de un hacker que hurtó de Microsoft una especificación de un códec para videoconferencias con grandes ratios de compresión y bajo una modificación hecha por Microsoft del estándar MPEG-4 parte 2. Así, elaboró un códec con alta calidad de imagen y muy buena compresión.

Tiempo después surgió una alternativa de código abierto llamada Xvid.



Estos códecs usan muchas de las técnicas más actuales de codificación de video, entre ellas muchas de las que vimos párrafos arriba.

El formato DivX tiene soporte para:

- ✓ menús interactivos
- ✓ subtítulos en varios idiomas
- ✓ múltiples pistas de audio

Al estar basado en el estándar MPEG-4 parte 2, utiliza DCT (transformada discreta del coseno), matrices de cuantización, tiene en cuenta la redundancia especial (intraframe) y temporal (interframe), compensación de movimiento, etc.

El sonido se almacena en MP3.

### **Caso de estudio 2: el formato de video RA2**

RA2 es un formato de video en desarrollo a cargo del redactor de este white paper. Su filosofía es simplicidad en la implementación y

preservación de las formas gráficas. Utiliza compresión con la técnica Deflator, y posee una mínima pérdida de color que es configurable. También permite almacenar los videos sin pérdidas de color.

La imagen de cada cuadro se barre de una manera óptima para lograr que los datos con redundancia espacial estén próximos. Esto permite obtener mejores ratios de compresión.

Utiliza interpolación de colores para resoluciones grandes y un esquema de color RGB.

Está desarrollado en el lenguaje Java SE especialmente para el software de simulación física Partion 3D. Consiste de tan solo una clase y es muy sencillo de utilizar.

No es el formato más eficiente ni compite con los sofisticados estándares actuales, pero su uso es muy simple y su implementación harto sencilla.

**Autor: Ramix (Ramiro A. Gómez)**



<http://www.peiper.com.ar>

Noticias y white papers