Estimación y Compensación del Movimiento

Jose Antonio Lorencio Abril

12/21

Contents

1 Introducción

Sabemos que un vídeo es una sucesión ordenada de imágenes, que evoluciona suficientemente rápido como para que nuestro cerebro no perciba la estaticidad de lo que vemos, parece realmente una imagen en movimiento. Según [2] el ojo humano es capaz de detectar hasta 75 FPS, es decir, hasta 75 imágenes en un segundo. Aunque la ilusión de movimiento puede conseguirse con una tasa menor de imágenes, el problema sigue siendo el mismo: debemos de guardar 75 imágenes por cada segundo de vídeo. Si quisiéramos guardar una película de una hora, hecha con imágenes de calidad HD (720p), lo que equivale a algo más de 900.000 px, necesitaríamos

$$1\ h \times 3600\ s/h \times 75\ imágenes/s \times 900000\ px/imagen \times 3\ B/px \sim 2.43 \cdot 10^{11} B = 243 GB$$

Esta cantidad es demasiado grande. Supongamos ahora, no obstante, que conseguimos comprimir cada una de las imágenes de la película hasta un 25% de su tamaño original. Esto se traduciría en un tamaño final de 60.75 GB, lo cual sigue siendo una cantidad demasiado elevada, y esto suponiendo la gran compresión que hemos supuesto.
Así, en la búsqueda de una solución para este problema surge la brillante idea de DPCM, **Differential Pulse-Code Modulation**, que se basa en que imágenes sucesivas de un vídeo serán, generalmente, muy similares, puesto que el movimiento se entiende como algo continuo, que obliga a que la sucesión de imágenes que lo emula deba evolucionar lentamente, con cambios pequeños entre imágenes consecutivas. De esta observación nace la idea de, en lugar de guardar cada una de las imágenes del vídeo, guardar únicamente una imagen, y a partir de ella calcular la diferencia entre la segunda imagen y esta primera. Esta diferencia es lo que almacenamos y transmitimos, y a partir de la primera imagen y la diferencia, reconstruimos la segunda imagen. Esto se hace con todas las imágenes del vídeo, y se consigue una tasa mucho más grande de compresión, con tamaños mucho más asequibles. El proceso no es tal cual así, ya que en el proceso se va acumulando error, pero hay técnicas para solventar este problema, con la misma idea básica. Hoy en día, una película con las características anteriores puede ocupar unos 3.6 GB.

2 DPCM

Vamos ahora a explicar detenidamente cómo funciona esta técnica. El desarrollo seguido viene de los apuntes de la asignatura [3] y de [1].
Consideremos una señal $x(t)$ muestreada por $x(kT_s)$, donde T_s es el período de muestreo (diferencia temporal entre la toma de dos muestras consecutivas) y k es un entero que indica el índice de cada muestra. Si acordamos T_s de antemano, podemos escribir las muestras como $x(k)$.
Sea, ahora, $d(k)$ la diferencia entre la muestra actual y la anterior:

$$d(k) = x(k) - x(k-1)$$

Que podemos cuantizar en lugar de $x(k)$, obteniendo una señal cuantizada $d_q(k)$. Si la señal $x(t)$ es muestreada a una tasa suficientemente alta como para que se cumpla la hipótesis de parecido de las imágenes mencionada en la introducción, entonces el rango de valores de $d(k)$ será menor que el de $x(k)$, por lo que su compresión será mayor.
Tras transmitir $d_q(k)$, teóricamente podemos reconstruir la señal original invirtiendo la operación anterior, obteniendo una aproximación de $x(k)$ mediante:

$$\hat{x}(k) = \hat{x}(k-1) + d_q(k)$$

De forma que si $\hat{x}(k-1) \sim x(k-1)$ y $d_q(k) \sim d(k)$, entonces obtendremos una buena aproximación de $\hat{x}(k)$.
Como adelantábamos en la introducción, este proceso idílico no funciona en la vida real, pues presenta distintos problemas:

2.1 Acumulación de ruido de cuantización

Si en lugar de usar este método, transmitiésemos cada imagen por separado, siendo estas las que cuantizamos, cada una de ellas tendría un ruido de cuantización, obtenido únicamente del proceso de cuantización para cada imagen.
No obstante, al hacer el proceso descrito, notamos que los errores de cuantización existentes entre $d(k)$ y $d_q(k)$, se van acumulando en $\hat{x}$ conforme avanzamos en el proceso. Esto puede hacer que terminemos obteniendo una imagen final muy distinta de la que deseamos transmitir.

2.2 Solución

En lugar de cuantizar $d(k)$, definimos ahora

$$g(k) = x(k) - x_q(k-1)$$

y cuantizamos esta cantidad.
Aparentemente necesitaríamos cuantizar $x(k-1)$, que es justo lo que queremos evitar, pero en realidad no es necesario. Veamos el caso de las 3 primeras imágenes.

1. Tenemos $x(1), x(2), x(3)$ y las queremos transmitir. Calculamos $x_q(1)$, $g(2) = x(2) - x_q(1)$ y $g_k(2)$. Transmitimos $x_q(1)$ y $g_q(2)$.

2. Sea $g_q(2) = g(2) + q(2)$, de forma que

   $$g_q(2) = x(2) - x_q(1) + q(2)$$

   y entonces

   $$x_q(2) = g_q(2) + x_q(1) = x(2) - x_q(1) + q(2) + x_q(1) = x(2) + q(2)$$

De forma que podemos ahora obtener

$$g(3) = x(3) - x_q(2)$$

y transmitirlo.

3. Este proceso se repite hasta terminar. Para recuperar la imagen se hace de la misma forma que antes.

Esto evita la acumulación de error por ruido.

2.3 Errores de transmisión

De nuevo, en el escenario en que transmitimos cada imagen por separado, un error en la trasmisión de una imagen solo afectará a esta. Pero en DPCM, un error de este tipo haría que todas las imagenes tras la que sufre el error también sufriesen cambios.

2.4 Solución

El mejor método para evitar este problema es dividir la muestra en subconjuntos y cada cierto tiempo, enviar una imagen cuantizada directamente, reiniciando el proceso de DPCM. De esta forma, un error de transmisión solo afectará a las muestras siguientes a la del error, hasta que se reinicie el proceso, y no hasta el final de la transmisión.

3 Estimación y Compensación del Movimiento

Como se explica en [4], la **estimación del movimiento** trata de predecir donde estarán ciertos elementos de una imagen, a partir de las anteriores. Para ello, podemos ver el movimiento como un vector de desplazamiento de un objeto desde una posición inicial hasta una final. El flujo óptico es el campo de vectores del desplazamiento aparente $d = (d_1, d_2)$ que obtenemos a partir de la ecuación de restricción:

$$x\left(n_1, n_2, n\right) = x\left(n_1 - d_1, n_2 - d_2, n - 1\right)$$

Es decir, que si en la muestra $x\left(n-1\right)$, un pixel tiene posición (n_1, n_2), entonces el campo del desplazamiento aparente debe cumplir que en la muestra $x\left(n\right)$ ese píxel debe estar en la posición $(n_1 + d_1, n_2 + d_2)$. Esto no puede hacerse de forma exacta, pero si de forma aproximada, normalmente con MCO.
Por otro lado, el movimiento no puede ser determinado píxel a píxel, ya que hay dos componentes de movimiento por píxel, de forma que tenemos el doble de incógnitas que de ecuaciones. Un enfoque usad normalmente es asumir que el movimiento es constante en una pequeña región, la **apertura**. Si esta es demasiado grande, entonces perderemos detalle del movimiento y solo obtendremos una media de cómo se mueven los objetos en pantalla. Si es demasiado pequeña, entonces estaremos estimando muy poco. Así, la determinación de la apertura es un punto importante del proceso de estimación del movimiento.
La **compensación del movimiento** es la técnica que se vale de los vectores calculados gracias a la estimación del movimiento, para reconstruir una frame a partir de los anteriores.

3.1 Relación con la compresión de vídeo y DPCM

La relación es bastante sencilla de ver, si podemos estimar cómo será un frame a partir del anterior, entonces podemos transmitir menos datos para recomponer una misma información. Como vimos en clase, MPEG1 se basa en las ideas de DPCM, y además utiliza las técnicas de estimación y compensación del movimiento, por lo que constituyen un buen ejemplo de la relación existente entre todos estos conceptos.

- MPEG1 usa **frames I**, que son aquellos que se envía directamente su cuantización; **frames P**, que son predecidos utilizando los I; y **frames B**, que se predicen usando tanto los I como los P
- La disposición es de la forma

  $$I_i \quad B \quad B \quad P \quad B \quad B \quad P \quad B \quad B \quad I_{i+1}$$

  de manera que los P se predicen a partir de I_i y los B a partir de un I y un P o a partir de dos P, dependiendo de qué entre qué dos de estos tipos se encuentren.
- Por tanto, solo es necesario transmitir los frames I_i, los demás pueden ser reconstruidos mediante estimaciones.
- Si fuese importante que el error fuese muy pequeño, podría hacerse como en DPCM y enviar la diferencia entre P e I, reduciendo el error que podría darse en la predicción.

4 Técnicas de estimación y compensación del movimiento

4.1 Block Matching

Las imágenes se dividen en bloques. Es el movimiento conjunto de los píxeles de los bloques lo que se trata de estimar. Los bloques del frame actual se comparan con los bloques del frame de referencia, deslizando el actual a lo largo de una región concreta de píxeles del frame destino. Hay que seleccionar un criterio de semejanza, o distancia, que normalmente será MCO. De entre los candidatos entre los que elegir, se determina aquél que minimiza esta distancia, y se calcula el vector de desplazamiento como el vector diferencia entre ambos bloques. Obviamente, el resultado no es siempre exactamente el original, ya que hay presencia de ruido y la forma de los objetos también varía, no solo su posición. Por ejemplo los giros no son estimables mediante este tipo de técnicas.

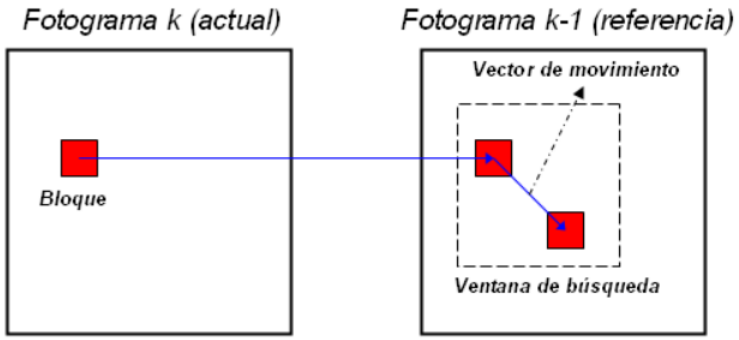

Figure 1: Esquema de Block Matching

4.2 Block Matching Jerárquico

Este método se basa en la utilización de núcleos de funciones para estimar los vectores de desplazamiento utilizando poca resolución espacial.
Se comienza creando una pirámide espacial, con los niveles de resolución fijados como una potencia de dos. Normalmente basta con tres o cuatro etapas. Comenzamos con el menor nivel de resolución, la cima de la pirámide y hacemos de esta forma block matching. Después, descendemos en la pirámide, aumentando la resolución en un factor de 2×2 en cada nivel. Doblamos el vector de desplazamiento del nivel anterior, para obtener la posición inicial de búsqueda en el nivel actual. Terminamos en el nivel más bajo de la pirámide, que contemplaría la máxima resolución.
Es decir, lo que se hace es aplicar Block Matching a varios niveles, estimando el movimiento en cada uno de ellos, como unas muñecas matrioskas.

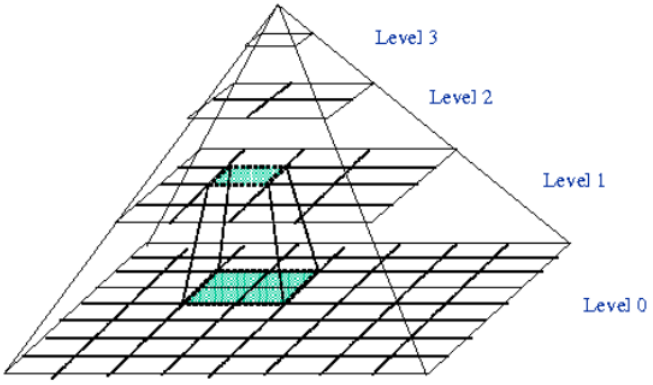

Figure 2: Esquema de Block Matching Jerárquica

4.3 Compensación de movimiento mediante Overlapped Block

La idea es efectuar block-matching pero de forma que no se toman los bloques de forma disjunta, sino que se permite el solapamiento entre distintos bloques. Se efectúa, entonces, block maching en los distintos bloques, y el vector de desplazamiento de las zonas de solapamiento se determina mediante algún tipo de media.
No obstante, este enfoque sería demasiado costoso computacionalmente, por lo que, en realidad, lo que se hace es solapar los vectores de velocidad, sin solapar los propios bloques.
La estimación inicial puede mejorarse mediante la actualización iterada de las estimaciones de las velocidades de diferentes bloques, uno cada vez, usando los resultados de las estimaciones resultantes en el cálculo del error.
De esta forma se suaviza el campo de velocidad y se elimina la estructura de bloques creada artificialmente.

4.4 Estimación del movimiento Pel-Rescursive

Es un método iterativo que calcula recursivamente un vector de desplazamiento para cada pixel (pel) en el frame actual. Comenzamos con una estimación $d = (d_1, d_2)$, y usamos el método iterativo:

$$\hat{d}_1^{(k+1)} = \hat{d}_1^{(k)} - \varepsilon \frac{\partial \mathcal{E}}{\partial d_1}|_{d=\hat{d}^{(k)}}$$

$$\hat{d}_2^{(k+1)} = \hat{d}_2^{(k)} - \varepsilon \frac{\partial \mathcal{E}}{\partial d_2}|_{d=\hat{d}^{(k)}}$$

con valor inicial el proporcionado por el valor final del pixel anterior. O sea, se usa algo similar al método del gradiente para acercar una primera estimación del vector de distancia hasta el real. La derivada calculada es la del error, por lo que nos estamos desplazando en la dirección de mayor decrecimiento del error de estimación del desplazamiento.
Este método suele funcionar bien con unas pocas iteraciones cuando el movimiento es pequeño, pero le cuesta converger cuando se encuentra ante grandes movimientos.
Este enfoque puede extenderse al enfoque jerárquico, mediante la utilización de interpolación de frames de secuencias de imágenes.

4.5 Métodos de Flujo Óptico

Este método trata de estimar las derivadas del campo de desplazamiento, en lugar del propio campo. Lo hace mediante la resolución usando MCO de

$$v_x \frac{\partial f}{\partial x} + v_y \frac{\partial f}{\partial y} + \frac{\partial f}{\partial t} = 0$$

A partir de esta ecuación se utilizan conceptos de geometría diferencial y cáculo de variaciones para calcular un óptimo de la función del error en términos de v_x y v_y para cada t. Se obtienen ciertas ecuaciones que se aproximan por aproximaciones de primer orden para las distintas derivadas. Esto se resuelve iterativamente mediante el método de Gauss-Seidel, método numérico de aproximación funcional.

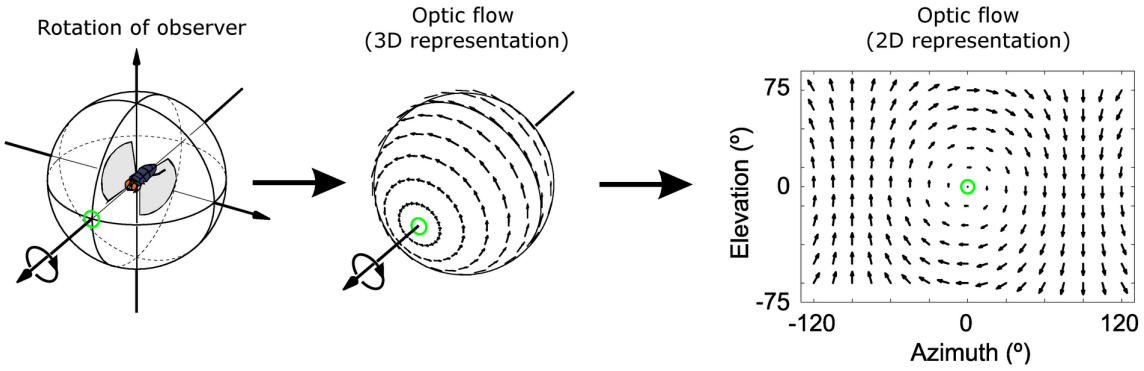

Figure 3: Representación del campo de flujo óptico

4.6 Métodos basados en mallas

Se establece una malla de puntos de velocidad denominados **puntos de control** en el frame destino y los correspondientes puntos en el frame de referencia. A diferencia de en el método de block matching, el movimiento no se considera constante entre los puntos de control, sino que estos puntos se utilizan para establecer un modelo de movimiento afín. Esto es, un modelo con 6 parámetros que representa rotaciones y traslaciones en el plano. Matricialmente

$$d = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} a_{13} \\ a_{23} \end{pmatrix}$$

Es decir, se busca la transformación afín de la imagen de referencia que mejor aproxima la imagen destino.

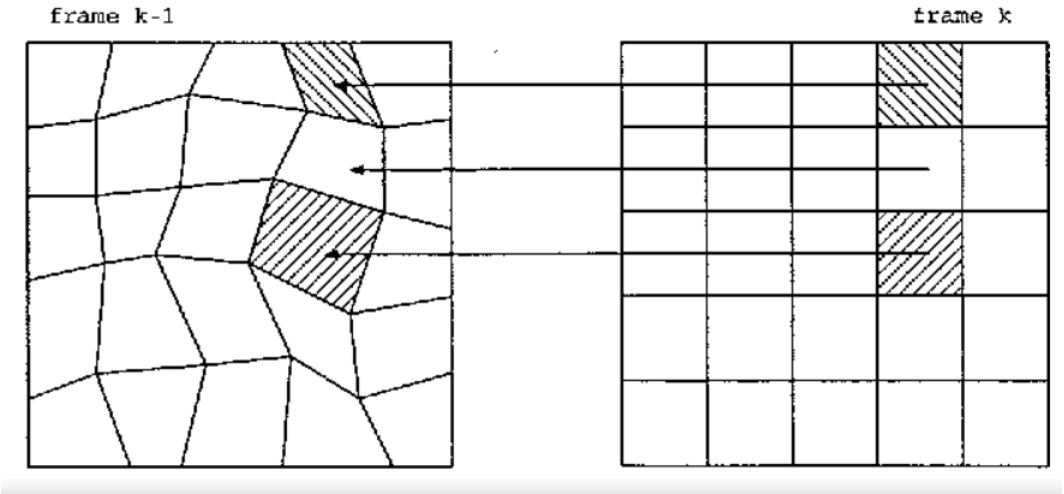

Figure 4: Esquema de la estimación basada en mallas

Como se observa en esta ilustración, los puntos de control no sufren todos el mismo movimiento, al contrario que en la estimación medianto block matching.

5 Conclusión

Hemos visto en el desarrollo de este documento la utilidad de las técnicas de estimación y compensación, que nos ofrecen un método para poder guardar vídeos de larga duración, una cuestión de alta importancia estos últimos años, en los que encontramos enormes mejoras en la calidad de imagen y, por tanto, en la cantidad de información necesaria para su codificación. Por otro lado, las plataformas de streaming son cada vez más populares y está claro que si no se dispusieran de este tipo de métodos para poder transmitir tales cantidades de información de video de forma eficiente no podrían llevar a cabo su labor tan cómodamente.
Por otro lado, vemos que hay no solo distintos métodos para hacer esto, sino que hay también diferentes enfoques, que atacan el problema desde diferentes perspectivas, y que proporcionan diferentes contextos de utilización de las técnicas, en las que unos enfoques son más efectivos que otros.

References

[1] Wajih Abu-Al-Saud. Sampling and pulse code mod. dpcm.

[2] HealthLine. How many fps can the human eye see? https://www.healthline.com/health/human-eye-fps.

[3] Luis Daniel Hernandez Molinero. Compresion de video.

[4] John W. Woods. *Multidimensional Signal, Image, and Video Processing and Coding*. ACADEMIC PR INC, June 2011.