Estimación de Movimiento (Marzo 2012)

Iván López Espejo, Jonathan Prados Garzón

Documento que trata de la estimación de movimiento en imágenes.

I. Introducción

Las variaciones temporales de intensidad luminosa a lo largo de una secuencia de imágenes se deben fundamentalmente al movimiento de los objetos 3D (rotación, traslación, etc) en el mundo real proyectados sobre el plano imagen.

La fuerte correlación existente en la dirección de movimiento tiene múltiples aplicaciones en el campo del procesamiento de la imagen como, por ejemplo:

- *Compresión de vídeo*: Muy importante dentro de la familia de estándares MPEG, H.261 o H.263. Se trata de eliminar la redundancia temporal en términos de intensidad luminosa entre frames temporalmente adyacentes.
- *Procesamiento de vídeo*: Algunas aplicaciones concretas son segmentación o mitigación y supresión de ruido.
- *Visión por computador*: Estimación del movimiento de la cámara y recuperación de la estructura de la escena.

La estimación de movimiento entre cada par de frames se puede llevar a cabo sobre diferentes soportes de modelo de movimiento, siendo usualmente: imagen completa, píxel a píxel, por bloques o por regiones. El movimiento píxel a píxel se representa por un conjunto de parámetros (normalmente, desplazamiento, velocidad y aceleración), siendo computacionalmente muy costoso. La estimación de movimiento recursiva píxel a píxel tiene aplicaciones en compresión de vídeo. En el movimiento por bloques la imagen se divide en bloques rectangulares no solapados de tamaño fijo. Se presupone en este contexto que sólo se producen traslaciones, siendo el modelo por excelencia empleado en compresión de vídeo, principalmente por su simpleza de implementación sobre hardware. No obstante, presenta problemas ante movimientos más generales como una sencilla transformación afín que incluya escalado y/o rotación. Por último, en el soporte de regiones se hace uso de particiones más generales, segmentándose la imagen y agrupándose los píxels de la misma superficie (siendo esta la filosofía de MPEG-4). La idea es agrupar todos los objetos que sufren el mismo movimiento, realizándose conjuntamente la segmentación y la estimación.

Los modelos de movimiento se pueden agrupar en dos grandes categorías: modelos paramétricos y no paramétricos. Dentro de los paramétricos se pueden llevar a cabo estimaciones en perspectiva o afines, principalmente, mientras que dentro de los modelos no paramétricos se pueden encontrar métodos de estimación de flujo óptico, métodos por bloques, recursivos y bayesianos. En este texto únicamente se mencionan los tres últimos, haciendo hincapié en los métodos por bloques debido a su predominancia dentro de los estándares de compresión de vídeo.

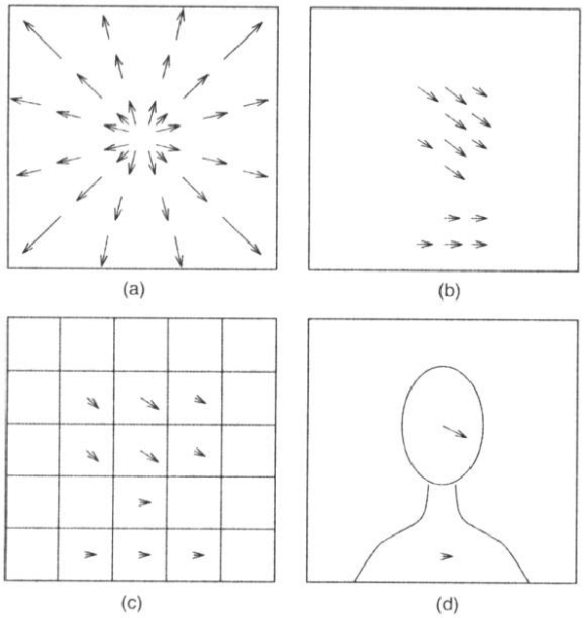

Fig. 1. Soportes de modelos de movimiento: (a) imagen completa, (b) píxel a píxel, (c) por bloques y (d) por regiones.

II. Métodos por bloques

La hipótesis sobre la que se sustentan estos métodos es que un bloque de tamaño *LxL* en la imagen *k-1* centrado en el punto (x_0, y_0) coincide con un bloque del mismo tamaño en la imagen *k*. Esto expresado matemáticamente quedaría

$$f(x, y, k) = f(x + d_1, y + d_2, k - 1) \quad \forall\, (x, y) \in B.$$

Dicho de otro modo, cada uno de los bloques que componen cada una de las imágenes o frames de la secuencia de vídeo posee un homólogo, esto es, un bloque del mismo tamaño que representa la misma porción de la escena en otra imagen o frame consecutivo de la secuencia de vídeo.

El procedimiento que se sigue en este tipo de métodos de estimación de movimiento es dividir el frame actual de la secuencia de vídeo en una matriz de macro bloques y luego buscar este mismo bloque en alguno de los frames del vídeo consecutivos. Los tamaños de bloque que suelen considerarse típicamente se encuentran en torno a 16x16 píxels. La zona de

búsqueda queda determinada por un parámetro *p* denominado parámetro de búsqueda. El significado de este parámetro queda ilustrado en la siguiente figura.

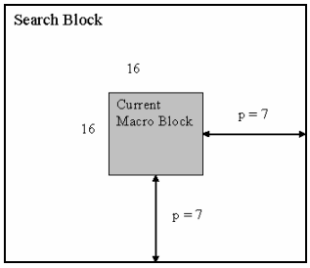

Fig. 2. Zona de búsqueda dentro del frame objetivo determinada por un tamaño de bloque 16x16 y un parámetro de búsqueda *p=7*.

Ahora bien, para determinar cuál es el bloque del frame objetivo que mejor casa con el bloque del frame ancla es necesario emplear una función de coste. Para este problema, las dos funciones de coste más empleadas suelen ser:

$$MAD = \frac{1}{L^2}\sum_{i=0}^{L-1}\sum_{j=0}^{L-1}\left|C_{ij} - R_{ij}\right|,$$

$$MSE = \frac{1}{L^2}\sum_{i=0}^{L-1}\sum_{j=0}^{L-1}(C_{ij} - R_{ij})^2,$$

donde C_{ij} y R_{ij} son los píxels que están siendo comparados en el bloque ancla y en el bloque objetivo, respectivamente.

Una vez introducidas las técnicas de estimación de movimiento por bloques, pasaremos a presentar algunas de las variantes más conocidas dentro de este conjunto.

Búsqueda exhaustiva (ES o Full Search)

Constituye el algoritmo más simple conceptualmente hablando. Este algoritmo calcula la función de coste para todas las posibles localizaciones dentro de la ventana de búsqueda. Este algoritmo encontrará el homólogo que proporciona la mejor PSNR de entre todos los posibles bloques de la imagen objetivo dentro de la ventana de búsqueda fijada. El único inconveniente es que es muy costoso en términos computacionales. El problema anterior es principalmente el que motiva la definición de nuevos algoritmos de búsqueda que logren la misma PSNR que este método pero realizando el menor número de cálculos posibles.

Búsqueda en tres pasos (Three Step Search o TSS)

Este procedimiento se basa en la hipótesis de que la superficie de error dentro del área de búsqueda (o de la función coste aplicada dentro de la ventana de búsqueda) es unimodal. De este modo, sólo existirá un único mínimo local que se corresponderá con el mínimo global. El procedimiento queda ilustrado en la figura 3. Se comienza con un tamaño de paso *S=4* para un parámetro de búsqueda típico de 7. Entonces se evalúa la función de coste en 8 localizaciones situadas en torno al píxel central del bloque, además de en el propio píxel central. De las 8 localizaciones mencionadas, 4 se encontrarán a una distancia raíz de dos veces el tamaño del paso *S* y 4 a distancia *S* (ver la figura). Una vez se ha evaluado la función de coste en estos 9 puntos, se comprueba cuál de estos da un valor mínimo, se toma este nuevo píxel como central, se reduce el tamaño del paso *S* a la mitad y se repite el procedimiento. Este proceso continuará hasta que el valor de *S* sea igual a 1 que, para nuestro caso, será en la siguiente iteración (de ahí el nombre del algoritmo).

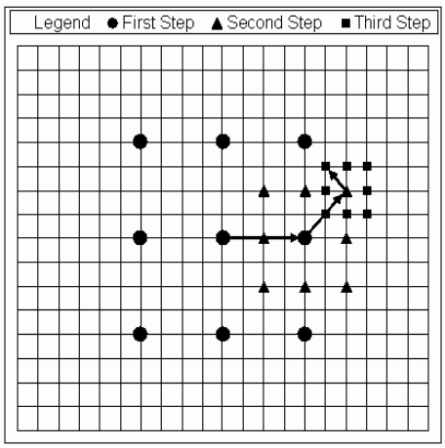

Fig. 3. Procedimiento de búsqueda por bloques TSS. El vector de movimiento hallado es (5, -3).

Es interesante notar que para el caso de la figura 3 se calcula la función de coste para 25 macro bloques, mientras que, si usásemos el método de búsqueda exhaustiva, se hubiese calculado para 255 bloques. Por tanto, se ha reducido la computación necesaria en un factor 9. La desventaja de este método con respecto al full search es que existe el riesgo de que la función error no sea unimodal y que, por tanto, no se encuentre el mínimo global. En este caso la PSNR se verá degradada.

Método nuevo de búsqueda en tres pasos (New Three Step Search o NTSS)

Fue el primer algoritmo de búsqueda rápido para estimación de movimiento ampliamente aceptado. Se usaba en las implementaciones de estándares como MPEG-1 y H.261.

El procedimiento seguido por este método queda ilustrado en la figura 4. En primer lugar se evalúa la función de coste en 17 puntos, donde uno de ellos constituye el píxel central del bloque considerado. En total, 8 de estos puntos se encuentran a una distancia *S=1* (considerando 8 vecinos) y, los otros 8 puntos, a una distancia *S=4*. Si el punto de menor valor para la función de coste es el píxel central, entonces el algoritmo finalizaría. Si es uno de los 8 puntos que se encuentran a distancia *S=1*, entonces trasladamos el centro a este nuevo punto y evaluamos la función de coste en sus 8 vecinos. Dependiendo del punto en cuestión, tendremos que realizar 3 o 5 cálculos de la función de coste adicionales. De estos

nuevos puntos, aquel que nos dé un valor menor será el elegido. Por otro lado, si en el primer paso el punto para el cual obtenemos un valor mínimo de la función de coste se encuentra a *S=4*, entonces procederemos de la misma forma que para el método TSS.

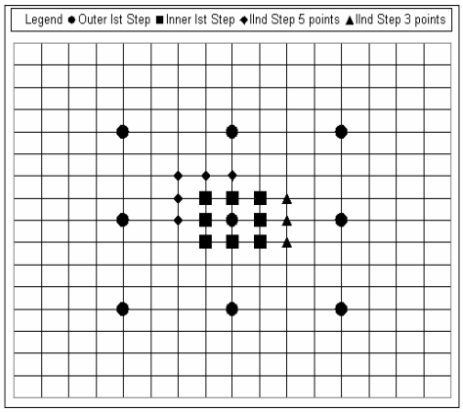

Fig. 4. Procedimiento del método nuevo de búsqueda en tres pasos (NTSS).

Por último, es importante notar que en el mejor de los casos este método requerirá de tan sólo 17 evaluaciones de la función de coste, mientras que en el peor de los casos (en el primer paso se obtiene un valor de la función de coste mínima a una distancia *S=4* del punto central), serán necesarias 33 evaluaciones de la función de coste.

Búsqueda simple y eficiente (Simple and Efficient Search o SES)

Este método constituye una extensión del método TSS y explota aún más el hecho de suponer que la superficie del error es unimodal.

El algoritmo comienza calculando la función de coste para los puntos A, B y C representados en la figura 5(a). El punto A constituiría el píxel central y los puntos B y C se encontrarían a una distancia *S=4* del punto central A. Una vez evaluada la función de coste en estos puntos, se selecciona el patrón de búsqueda en función de las siguientes reglas:

Si MAD(A) ≥ MAD(B) y MAD(A) ≥ MAD(C), seleccionar el patrón de la figura 5(b).
Si MAD(A) ≥ MAD(B) y MAD(A) ≤ MAD(C), seleccionar el patrón de la figura 5(c).
Si MAD(A) < MAD(B) y MAD(A) < MAD(C), seleccionar el patrón de la figura 5(d).
Si MAD(A) < MAD(B) y MAD(A) ≥ MAD(C), seleccionar el patrón de la figura 5(e).

Una vez seleccionado el patrón de búsqueda, se evalúa la función de coste en los nuevos puntos que proceda y se selecciona aquel que proporcione un valor mínimo. Posteriormente, se aplica el mismo procedimiento expuesto en el punto seleccionado. El algoritmo concluirá cuando *S=1* (para nuestro caso será en tres iteraciones dado que *S* se divide por dos en cada iteración).

A pesar de que este algoritmo ahorra muchos cálculos comparado con el TSS, éste no fue ampliamente usado debido principalmente a dos motivos:

- En la práctica, las superficies de error no suelen ser estrictamente unimodales y, por lo tanto, la PSNR que suele obtenerse es bastante pobre si la comparamos con la obtenida con el método TSS.
- Se había publicado ya otro algoritmo, conocido como búsqueda en cuatro pasos, que presentaba menos requisitos de computación y una mejora significante en la PSNR obtenida con respecto al método TSS.

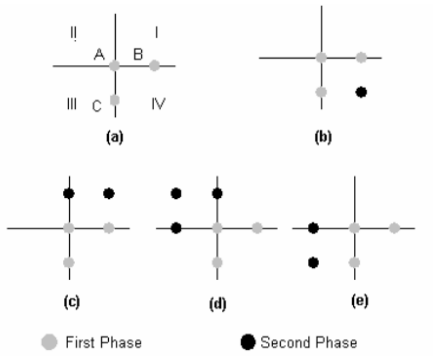

Fig. 5. Patrón de búsqueda seguido en la primera fase del algoritmo de búsqueda simple y eficiente (a). Patrón de búsqueda seguido en la segunda fase del algoritmo de búsqueda simple y eficiente en función del cuadrante seleccionado en la primera fase ((b), (c), (d) o (e)).

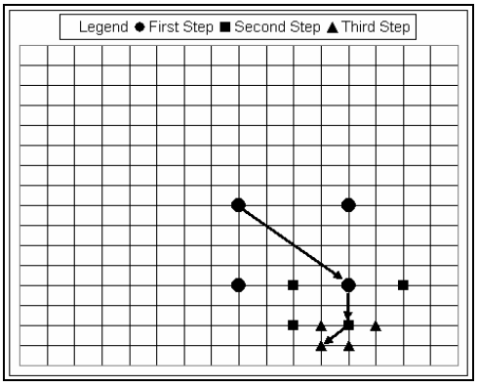

Fig. 6. Ejemplo de procedimiento SES. El vector de movimiento es (3, 7) para este ejemplo.

Búsqueda en cuatro pasos (Four Step Search o 4SS)

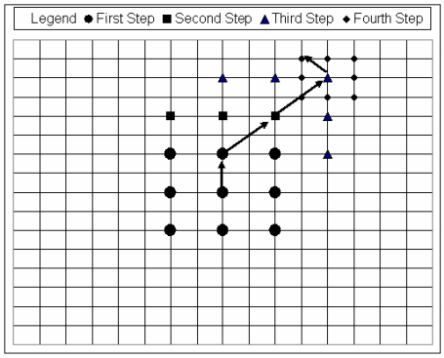

Fig. 7. Procedimiento de búsqueda en cuatro pasos (4SS). El vector de movimiento para este ejemplo es (3, -7).

Este procedimiento, como su propio nombre indica, se realiza en cuatro pasos. El método es totalmente independiente del

parámetro de búsqueda *p* que se utilice.

En primer lugar se comienza con un patrón de búsqueda de 9 puntos, incluido el píxel central. Los 8 puntos restantes de los 9 comentados constituyen los 8 vecinos a una distancia *S=2* del píxel central. Se evalúa en cada uno de estos puntos la función de coste que se esté empleando usando un tamaño de ventana de 5x5. El punto donde se obtenga un valor mínimo se selecciona. En caso de ser este el punto central, nos vamos directamente al cuarto paso. En caso de ser un punto distinto al central, centrándonos en ese punto repetimos el primer procedimiento. De este modo repetiríamos el mismo método 3 veces, salvo que en alguno de los pasos hayamos obtenido como valor mínimo el asociado al punto central y hayamos saltado directamente al cuarto paso. El cuarto paso es similar pero, ahora, se reduce la distancia al píxel central a *S=1* y el tamaño de la ventana a 3x3.

Es importante notar que este algoritmo evalúa la función de coste 17 veces en el mejor de los casos y 27 veces en el peor de los casos.

Búsqueda en diamante (Diamond Search o DS)

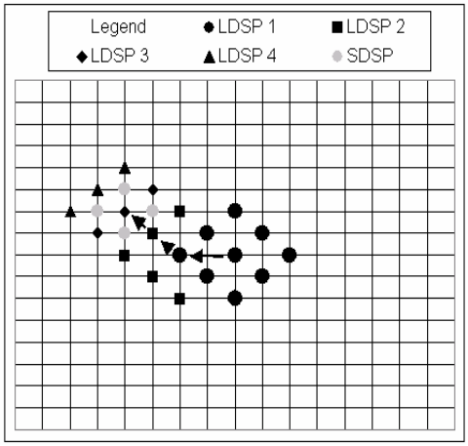

Fig. 8. Procedimiento de búsqueda en diamante. En este ejemplo se determina el vector de movimiento con valor (-4, -2) en cinco pasos.

Este algoritmo es muy similar al algoritmo FSS, sólo que en lugar de usar un patrón de búsqueda en forma de cuadrado usa un patrón en forma de rombo y, además, no limita el número de pasos que puede dar el algoritmo. Para todos los pasos distintos al último se usa un patrón de búsqueda más grande: es el conocido como LDSP (Large Diamond Search Pattern). Para el último paso de la búsqueda se usa un patrón de diamante más pequeño conocido como SDSP (Small Diamond Search Pattern). El criterio de parada se da cuando se llega al último paso, siendo necesario para ello que en un paso intermedio se obtenga un valor mínimo de la función de coste para el punto central.

Este algoritmo ofrece unos valores para la PSNR muy cercanos a los proporcionados por el método ES mientras que el coste computacional es significativamente inferior.

Búsqueda en patrón de cruz adaptativo (Adaptive Rood Pattern Search o ARPS)

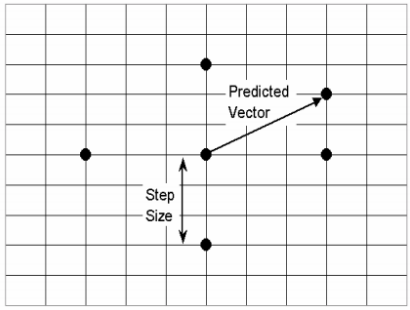

Fig. 9. Patrón en cruz adaptativo. El vector de movimiento obtenido es (3, -2) y el tamaño del paso de *S=Max(|3|, |-2|)=3*.

Este algoritmo se aprovecha del hecho de que el movimiento general de un frame es normalmente coherente, esto es, que si los macro bloques que hay alrededor del macro bloque actual (del que queremos estimar su vector de movimiento) se han movido en una dirección en particular, entonces habrá una alta probabilidad de que el macro bloque actual también se haya movido en esa dirección.

El algoritmo usa el vector de movimiento del macro bloque situado inmediatamente a la izquierda del macro bloque actual para predecir el vector de movimiento. Una vez consultado el vector de movimiento del macro bloque de la izquierda, se evalúa la función de coste en un patrón en cruz en torno al punto central (ver figura 9). Los 4 puntos que no son el punto central se encontrarán a una distancia de este de *S=Max(|X|,|Y|)*, siendo *X* e *Y* las coordenadas del vector de movimiento del macro bloque situado inmediatamente a la izquierda. En este patrón que se ha fijado se evalúa la función de coste para cada uno de los puntos y se seleccionará aquel punto que proporcione un menor valor. Tras este paso se aplica iterativamente el patrón SDSP visto en el algoritmo DS y no se concluye hasta que la función de coste dé valor mínimo para un píxel central del patrón.

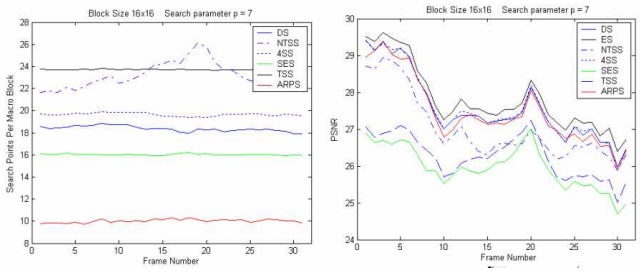

Fig. 10. Comparativa en términos de PSNR (figura de la izquierda) y coste computacional (figura de la derecha) de cada uno de los 7 métodos de estimación de movimiento por bloques expuestos.

En cuanto a una comparativa de los métodos por bloques presentados, el mejor resultado se concluye que es para el método ARPS, dado que ofrece unos valores de PSNR muy cercanos al método ES a la vez que obtiene una reducción muy notable en la cantidad de cómputo necesaria con respecto a los demás métodos.

III. Métodos recursivos

La filosofía de este método es la de realizar estimaciones

para el desplazamiento del tipo predicción-corrección:

$$d(\boldsymbol{x}, t, \Delta t) = d^i(\boldsymbol{x}, t, \Delta t) + u(\boldsymbol{x}, t, \Delta t),$$

donde $d(\boldsymbol{x}, t, \Delta t)$ es el desplazamiento estimado, $d^i(\boldsymbol{x}, t, \Delta t)$ es el desplazamiento de predicción y $u(\boldsymbol{x}, t, \Delta t)$ es el error de predicción respecto de la estimación. La estimación del desplazamiento o actualización de la predicción se basa en la minimización del gradiente de desplazamiento de diferencia de frame (DFD) en ese píxel, el cual se expresa de la forma,

$$DFD = I(\boldsymbol{x} + d(\boldsymbol{x}, t, \Delta t), t + \Delta t) - I(\boldsymbol{x}, t),$$

donde $I(\boldsymbol{x}, t)$ es la intensidad luminosa del frame en el píxel $\boldsymbol{x}$ en el instante t y $I(\boldsymbol{x} + d(\boldsymbol{x}, t, \Delta t), t + \Delta t)$ representa la intensidad luminosa de este mismo píxel pero desplazado en el frame que acontece en el instante $t + \Delta t$.

Además, como norma general, es interesante notar que la estimación encontrada en el paso previo se toma como predicción en el siguiente por la hipótesis de variación suave del desplazamiento para un Δt lo suficientemente pequeño, lo que refuerza el concepto de recursividad, pudiendo expresarse el desplazamiento para este tipo de métodos en los siguientes términos:

$$d(\boldsymbol{x}, t, \Delta t) = d(\boldsymbol{x}, t - \Delta t, \Delta t) + u(\boldsymbol{x}, t, \Delta t).$$

Por ello, a la hora de codificar la posición de un determinado píxel a través de una secuencia de frames dado el primero, puede ser suficiente con transmitir al decodificador la primera estimación de desplazamiento más los sucesivos errores de predicción.

Si desarrollamos en serie de potencias de Taylor el primer monomio de la DFD, la DFD se puede expresar, haciendo uso de los primeros términos del desarrollo, como

$$DFD = \frac{\partial I(\boldsymbol{x}, t)}{\partial x_1} d_1(\boldsymbol{x}) + \frac{\partial I(\boldsymbol{x}, t)}{\partial x_2} d_2(\boldsymbol{x}) + \Delta t \frac{\partial I(\boldsymbol{x}, t)}{\partial t}.$$

Como vemos, esta expresión recuerda a la estimación del flujo óptico (piénsese en el método de Lucas-Kanade). Por tanto, se trata ahora de estimar el vector desplazamiento a partir de la anterior aproximación haciendo uso de alguna de las siguientes técnicas:

- Empleando una estrategia de correspondencia entre bloques.
- Llevando a cabo una optimización por descenso de gradiente (recursividad).
- Imponer $DFD = 0$ y, para $t = 1$ y considerando un bloque de píxels, estimar el desplazamiento.

Sin entrar en su definición ni formulación, algunos algoritmos recursivos son:

- Algoritmo de Netravali-Robbins
- Algoritmo de Walker-Rao
- Estimación de Wiener (extensión del algoritmo Netravali-Robbins)

IV. Métodos bayesianos

En este tipo de métodos se lleva a cabo una estimación basada en MAP (máximum a posteriori). Se hace uso de dos funciones densidad de probabilidad:

- Una *pdf* condicional de la intensidad de la imagen observada dado el campo de movimiento.
- Una *pdf* a priori sobre los vectores de movimiento (modelo de movimiento).

Se trata, por tanto, de calcular la siguiente expresión en términos de la Regla de Bayes, donde la contribución de la distribución de probabilidad a priori sobre la imagen observada puede eliminarse ya que resulta constante en el proceso de estimación:

$$(\hat{d}_1, \hat{d}_2) = \underset{d_1, d_2}{\operatorname{argmax}}\, p(d_1, d_2 | g_k, g_{k-1}) \propto$$
$$\propto \underset{d_1, d_2}{\operatorname{argmax}}\, p(g_k | d_1, d_2, g_{k-1}) p(d_1, d_2 | g_{k-1}).$$

En la anterior igualdad, d_1 y d_2 son las componentes del vector desplazamiento que deseamos estimar, donde g_{k-1} y g_k son, respectivamente, los frames anterior y siguiente entre los que deseamos estimar el campo de movimiento. Estas variables siguen un modelo de imagen con ruido aditivo, de la forma,

$$g_k = I_k + n_k,$$

donde I_k es el frame *k-ésimo* libre de ruido y n_k es su componente de ruido aditivo asociada.

Por último, notar que la distribución a priori sobre los vectores de movimiento del anterior modelado bayesiano viene dada por el campo de movimiento del frame g_{k-1}.

V. Compresión inter-frame

La idea en la compresión inter-frame es expresar un frame en términos de uno o más frames vecinos en la secuencia de vídeo. Se trata, por tanto, de emplear predicción para aprovechar la redundancia temporal entre frames adyacentes en términos temporales, pudiendo conseguir así un alto factor de compresión.

El frame que deseamos comprimir (inter-frame) es fragmentado en macro bloques. Con algún tipo de algoritmo de block matching se lleva a cabo una estimación de movimiento hacia atrás (backward) sobre la imagen de referencia, obteniéndose en caso positivo un vector de movimiento con el que codificar el bloque del inter-frame. Además, lo normal es que no exista un matching con un error (obtenido como la diferencia entre el macro bloque en el inter-frame y el macro bloque con el que se ha establecido matching en la imagen de referencia anterior) cero, por lo que se calcula la diferencia entre los dos macro bloques (error de predicción) y esta es transmitida junto con el vector de movimiento al

decodificador. La imagen de la figura 11 ejemplifica el proceso de compresión inter-frame.

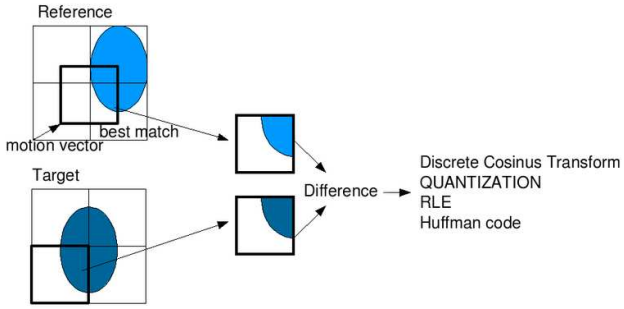

Fig. 11. Compresión inter-frame.

En el mejor de los casos conseguiremos un factor de compresión adecuado, pero también podemos tener situaciones desfavorables en las que el macro bloque buscado no se encuentre en la imagen de referencia. En este caso, el error de predicción puede ser excesivamente alto lo que, unido a la necesidad de transmitir este al codificador junto con el vector de movimiento, puede provocar el envío de una mayor cantidad de datos respecto de no aplicarse ninguna técnica de compresión. Usualmente, en esta situación, el codificador hace una excepción y transmite el macrobloque sin codificar.

Otro aspecto crítico es el llevar a cabo sucesivas codificaciones de inter-frames sobre frames de referencia estimados, lo que puede provocar una propagación del error inadmisible con el paso de un tiempo suficiente. Este problema y el de la sincronización es el que trata de solucionar la estandarización de la estructura de lo que se conoce como el GoF (Group of Frames), el cual define qué tipo de frame (original, comprimido, etc) se va intercalando en la secuencia de transmisión.

VI. Referencias

[1] Estimación y compensación de movimiento, *Profesor Agustín Cisa*. Noviembre de 2003.
[2] Estimación de movimiento, *J. M. Sotoca*. Junio de 2002.
[3] Inter frame, *Wikipedia*.
[4] Block Matching Algorithms For Motion Estimation, *Aroh Barjatya*, Student Member, IEEE. 2004.
[5] Transparencias de estimación de movimiento por similitud de bloques para Máster en Sistemas Multimedia, *Javier Mateos*.
[6] http://www.mathworks.com/matlabcentral/fileexchange/8761