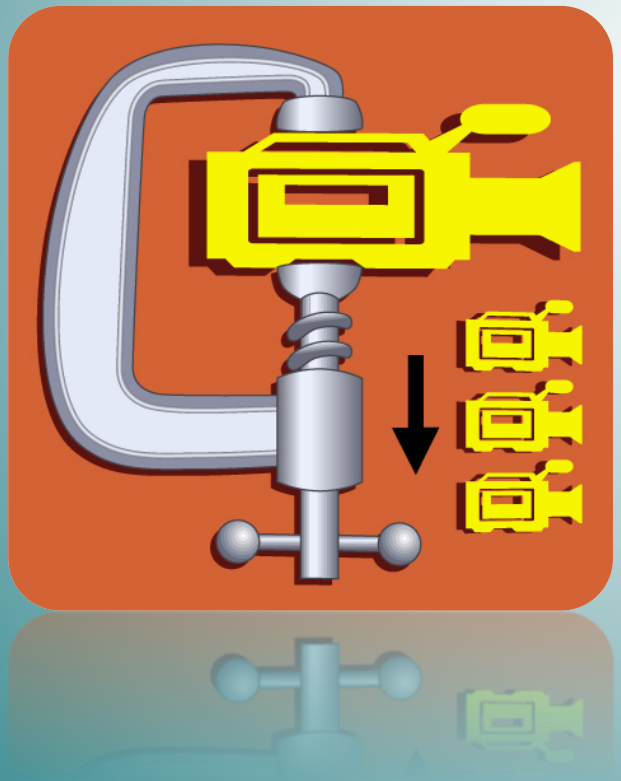
Compresión de Vídeo

Tema 2.4. Interpredicción en el H.264



Juan A. Michell Martín
Gustavo A. Ruiz Robredo

Departamento de Electrónica y Computadores

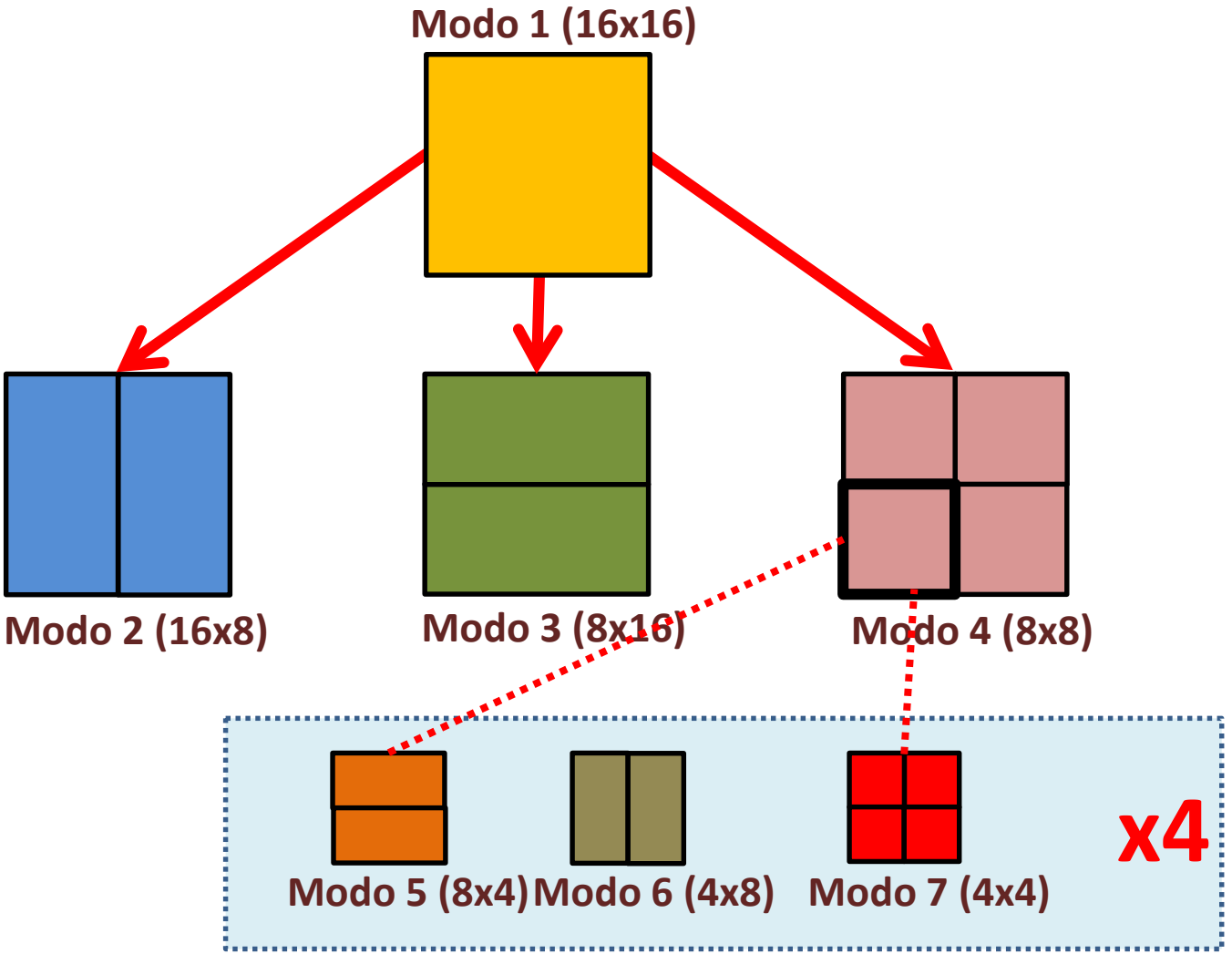
INTRODUCCIÓN

- La interpredicción o predicción *Inter* crea un modelo de predicción a partir de una o más *frames* previamente codificadas usando un técnica conocida como *variable block size motion estimation* o VBSME.

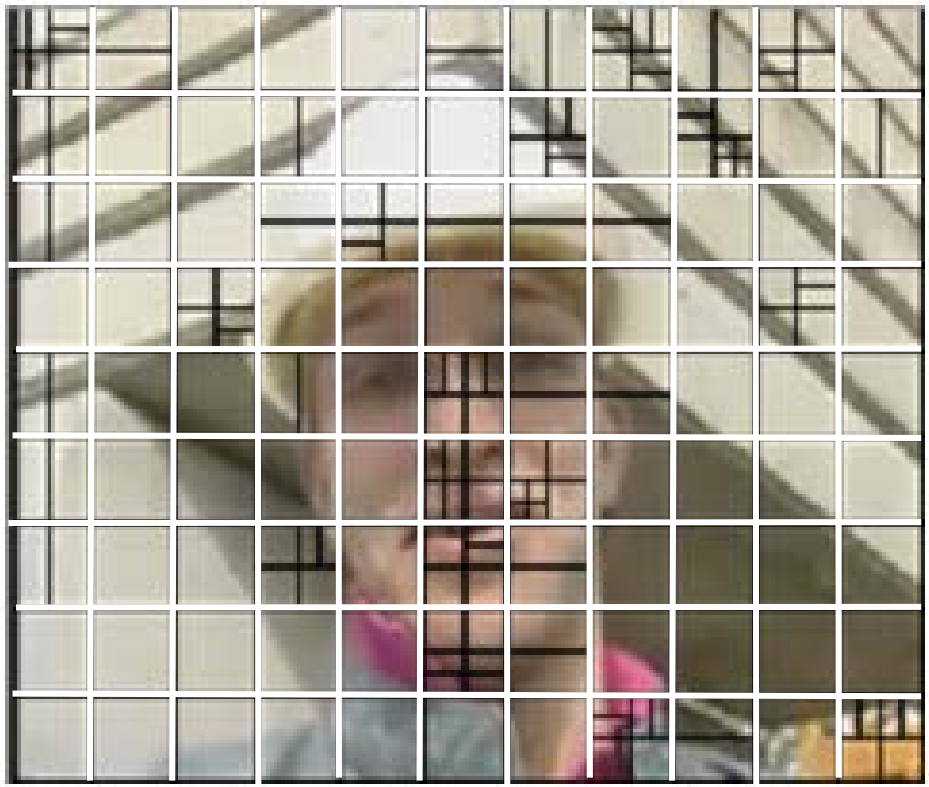
- A diferencia de los anteriores estándares, el H.264 incluye un rango de bloques de diferente tamaño que varían desde 16x16 hasta 4x4 siguiendo una estructura jerárquica:
  - Modo 1: 16x16
  - Modo 2: 16x8
  - Modo 3: 8x16
  - Modo 4: 8x8
  - Modo 5: 8x4
  - Modo 6: 4x8
  - Modo 7: 4x4

- La extrapolación de las muestras luma permite realizar estimaciones de los vectores de movimiento con resoluciones de ½ y ¼ pixeles.

Partición de los subbloques



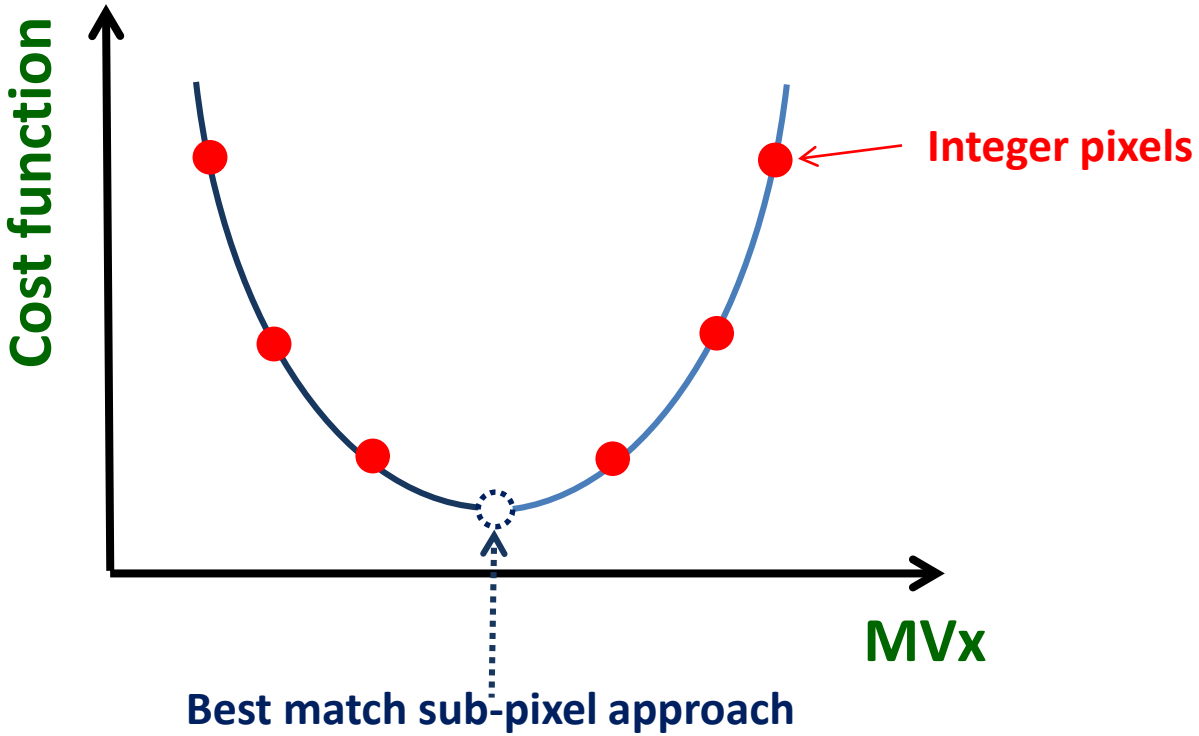
- **Para cada MB 16x16, se evalúan todos los posibles modos:**
  - **Bloques de mayor tamaño son convenientes para zonas con movimientos homogéneos .**
  - **Bloques de menor tamaño son útiles en zonas detalladas o con movimientos rápidos.**



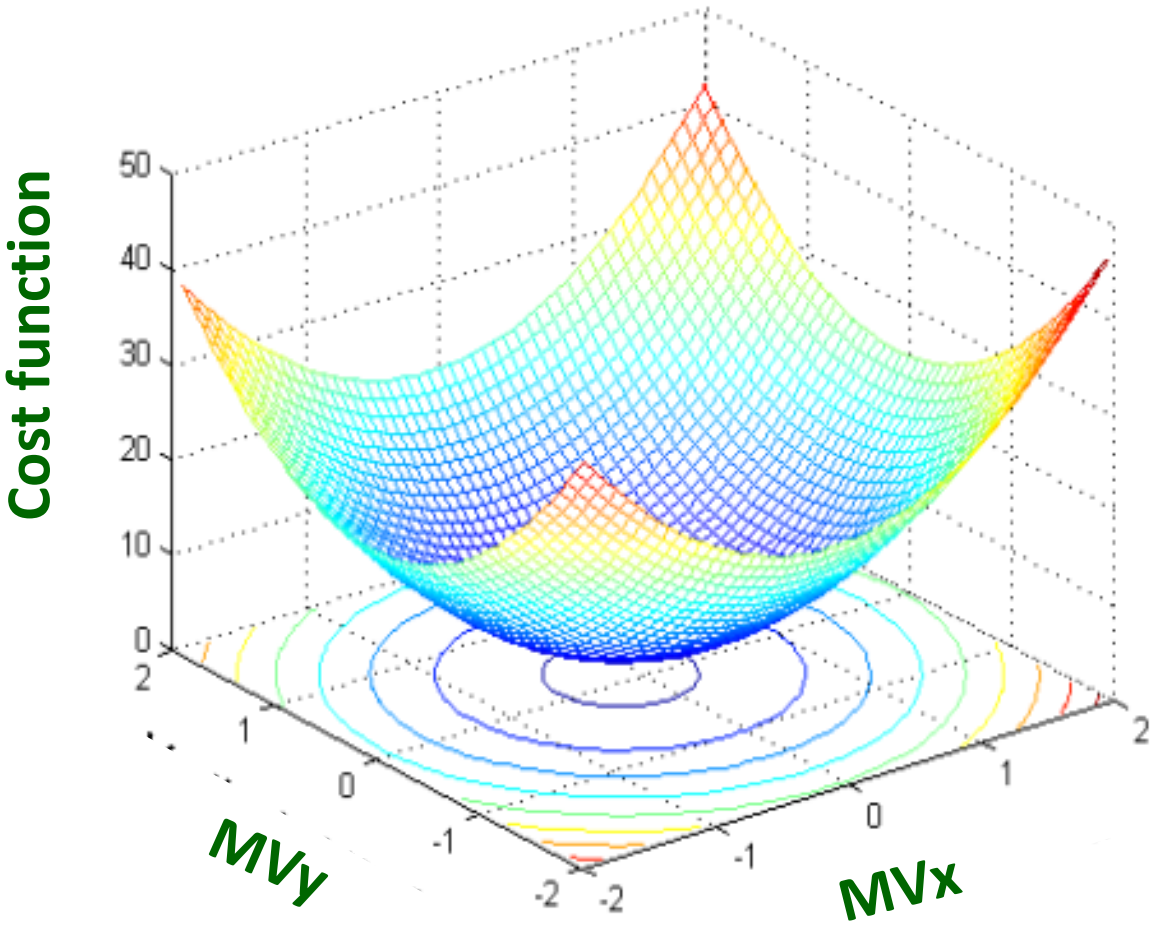
Efecto del tamaño del bloque

Pequeño Original Grande

INTERPOLACIÓN FRACCIONAL



Interpolación 2-D



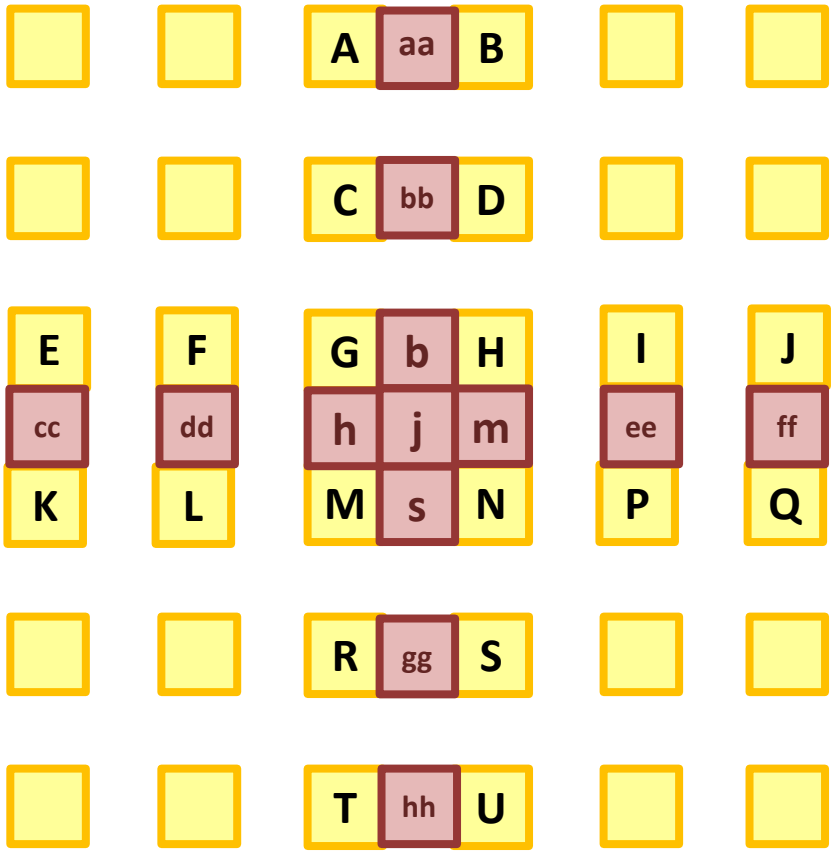
INTERPOLACIÓN FRACCIONAL EN EL H.264

- En el H.264, los vectores de movimiento pueden tener resolución de ½ y ¼-pixel.

- Las muestras de posiciones sub-pixel no existen en la *frame* de referencia. Es necesario crearlos usando una interpolación basada en muestras cercanas y utilizando filtros de 6-tap tipo Wiener y filtros Bilineal.

- La interpolación sub-pixel se realiza solamente alrededor del mejor *match* de cada sub-bloque obtenido sobre muestras enteras.

Interpolación luma 1/2-pixel



Entera

½ pixel

```
b=(E-5F+20G+20H-5I+J+1<<4)>>5
h=(A-5C+20G+20M-5R+T+1<<4)>>5
m=(B-5D+20H+20N-5S+U+1<<4)>>5
s=(K-5L+20M+20N-5P-q+1<<4)>>5
j=(aa-5bb+20b+20s-5gg+hh+1<<4)>>5
j=(cc-5dd+20h+20m-5ee+ff+1<<4)>>5
```

FinalDraftH.264, pag. 63

Interpolación luma 1/4-pixel

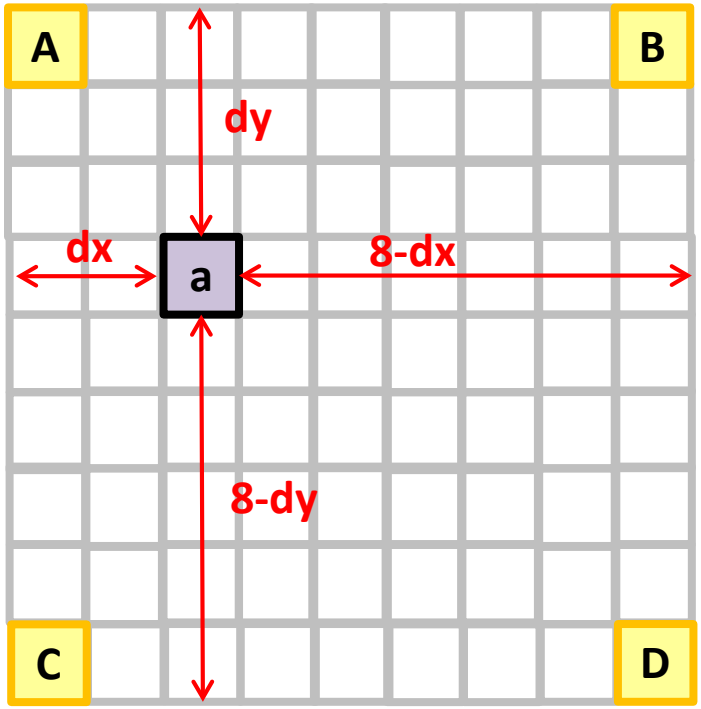
Entera

1/2 pixel

1/4 pixel

```
e=(A+a+1)>>1
f=(a+B+1)>>1
g=(A+b+1)>>1
h=(a+b+1)>>1
.....
```

Interpolación de la croma

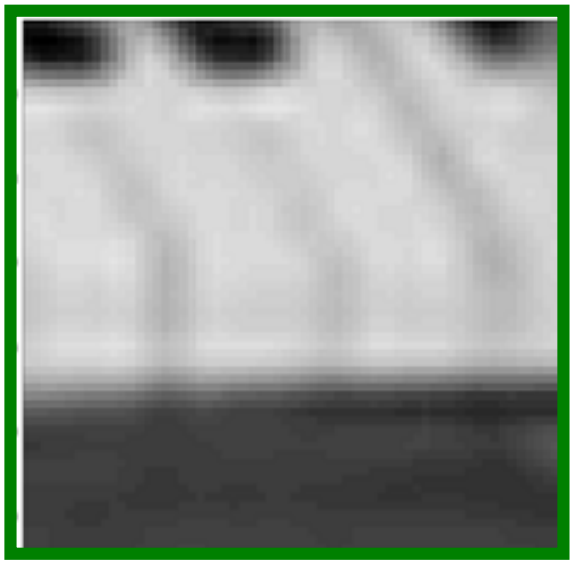


$$a = \left[(8 - dx)(8 - dy) \cdot A + dx \cdot (8 - dy) \cdot B + (8 - dx) \cdot dy \cdot C + dx \cdot dy \cdot D + 1 << 5 \right] >> 6$$

FinalDraftH.264, pag. 65

Entera
(Original)

1/2 pixel

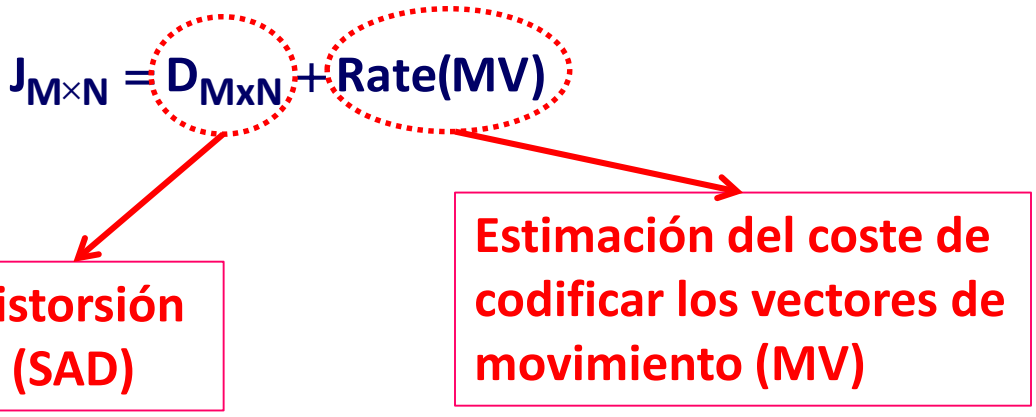


1/4 pixel

COSTE LAGRANGIANO

- EL H.264 utiliza la función coste Lagrangiano J para encontrar la mejor coincidencia (*match*) durante el proceso de estimación de movimiento.

- Para un bloque de dimensiones MxN, se selecciona el vector de movimiento MV que minimiza el lagrangiano J_{MxN} definido como



Cálculo de la distorsión o D_{MxN}

- Para calcular la distorsión, el H.264 utiliza una de las dos funciones coste posibles que se aplica a dos bloques de imágenes I_C e I_R :

1. Sin Hadamard. Se utiliza el SAD o *Sum of Absolute Differences*

$$SAD(i,j) = \sum_{i=1}^{N}\sum_{j=1}^{M} \left| I_C(i,j) - I_R(i,j) \right|$$

2. Con Hadamard. La matriz de Hadamard es definida como

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \qquad H_{2^n} = \begin{bmatrix} H_{2^{n-1}} & H_{2^{n-1}} \\ H_{2^{n-1}} & -H_{2^{n-1}} \end{bmatrix}$$

El cálculo de la función coste basada en la transformada de Hadamard se aplica a bloques de 4x4 y se realiza en tres pasos:

1.- Se calcula la transformada 2D de Hadamard de la diferencia

$$DIFHAD_4 = H_4 \cdot \left(I_C - I_R \right) \cdot H_4$$

2.- Se calcula la suma total del valor absoluto de los coeficientes

$$SADHacc = \sum_{i=1}^{4} \sum_{j=1}^{4} \left| DIFHAD(i,j) \right|$$

3.- Se realiza una operación de redondeo

$$SADH = \left(SADHacc + 1 \right) >> 1$$

Rate de un vector de movimiento MV

- El **Rate** de un vector de movimiento $MV=(MV_x,MV_y)$ se calcula como

$$Rate(MV) = \lambda_{motion} \cdot \left[bits\left(\left| MV_x - MVpred_x \right| \right) + bits\left(\left| MV_y - MVpred_y \right| \right) \right]$$

  - El factor lagrangiano $\lambda_{motion} = \sqrt{0.85 \cdot 2^{QP/3}}$

  - **bits** es una función tabulada que estima el coste d codificación del vector de movimiento. Tiene dos componentes:
    - $MV=(MV_x,MV_y) \rightarrow$ candidato
    - $MVpred=(MVpred_x,MVpred_y) \rightarrow$ predicción

Tabla de valores de λ_{motion}

QP	λ_{motion}	QP	λ_{motion}	QP	λ_{motion}	QP	λ_{motion}
0	1	13	1	26	5	39	23
1	1	14	1	27	6	40	25
2	1	15	1	28	6	41	29
3	1	16	2	29	7	42	32
4	1	17	2	30	8	43	36
5	1	18	2	31	9	44	40
6	1	19	2	32	10	45	45
7	1	20	3	33	11	46	51
8	1	21	3	34	13	47	57
9	1	22	3	35	14	48	64
10	1	23	4	36	16	49	72
11	1	24	4	37	18	50	81
12	1	25	4	38	20	51	91

Tabla de valores de bits(α)

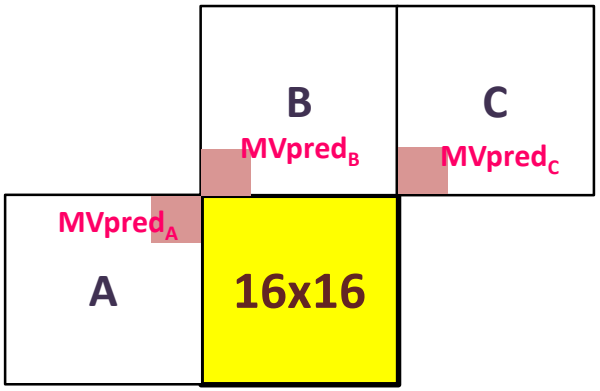
bits(α)							
$\alpha=0$	$\alpha=1$	$2\leq\alpha\leq3$	$4\leq\alpha\leq7$	$8\leq\alpha\leq15$	$16\leq\alpha\leq31$	$32\leq\alpha\leq63$	...
1	3	5	7	9	11	13	...

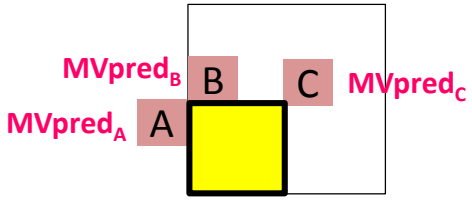
Predicción de los vectores de movimiento MV

- Los vectores de movimiento están altamente relacionados con el movimiento de bloques vecinos. Para ello, el $MVpred=(MVpred_x,MVpred_y)$ predice el MV de partida de un bloque analizando los MV de bloques próximos.

- El MVpred tiene dos componentes

$$MVpred=MVpred_{16x16}+MVpred_{block}$$



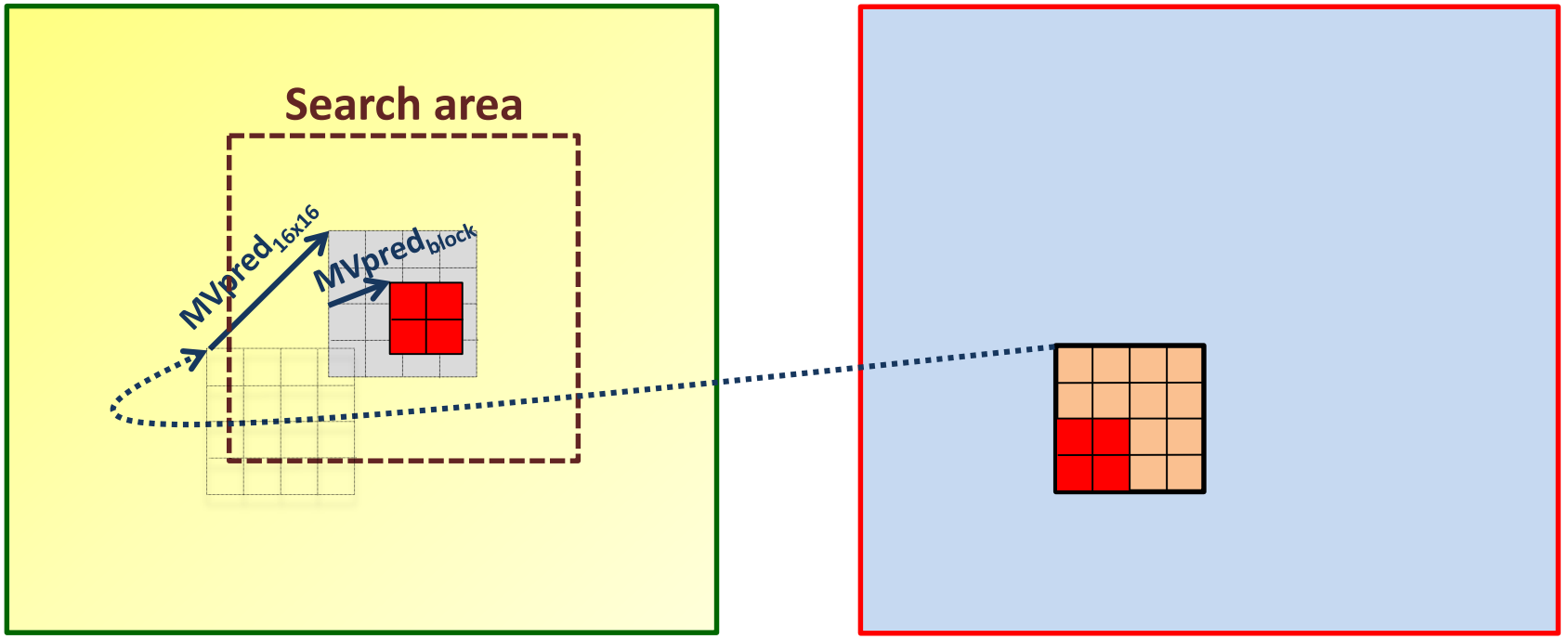


$$MVpred_{block}=median(MVpred_A,MVpred_B,MVpred_C)$$

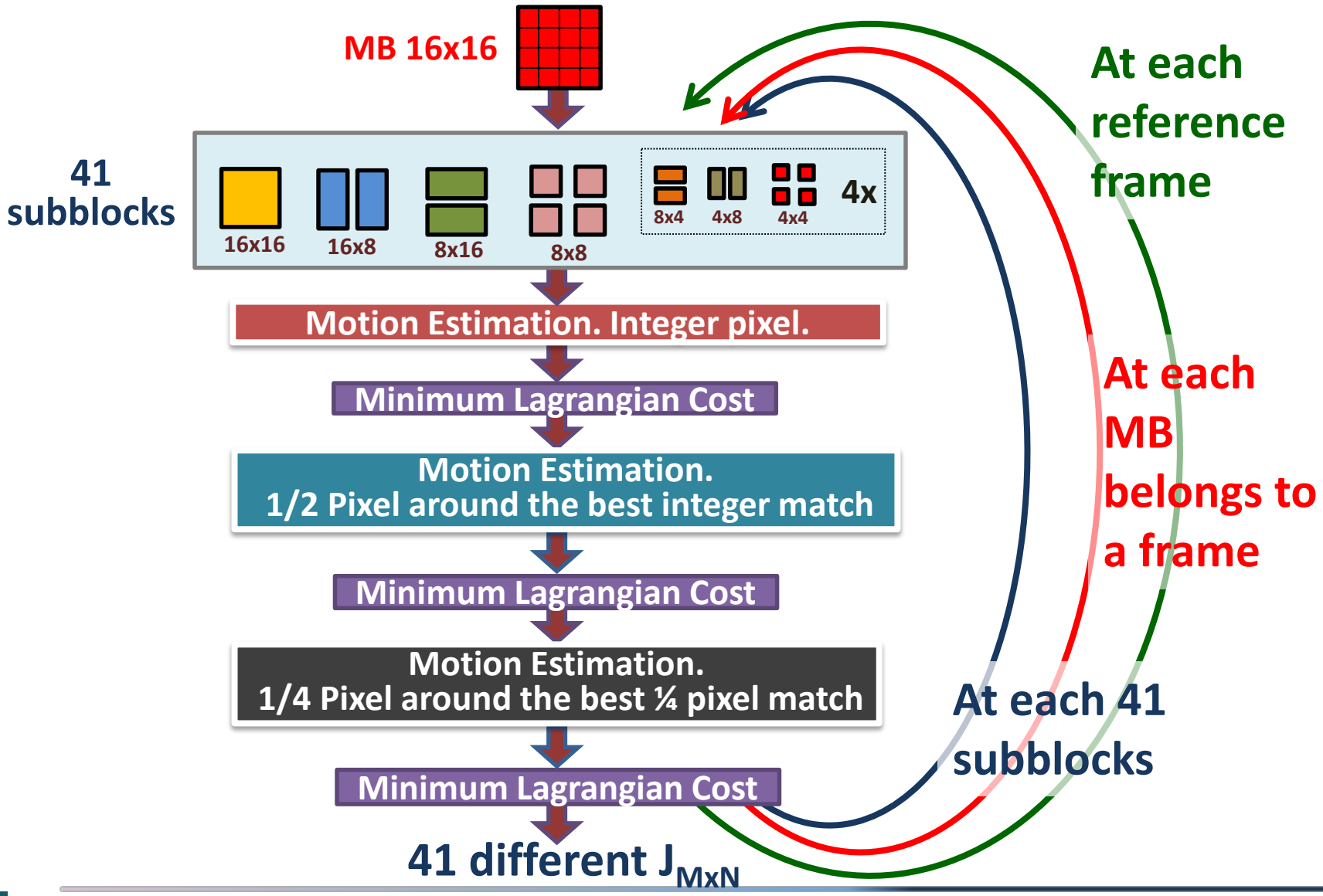
$$MVpred_{16x16}=median(MVpred_A,MVpred_B,MVpred_C)$$

La función median selecciona de los tres valores aquel que se encuentra en el medio.

➢ Efecto del MVpred en el proceso de estimación de movimiento.



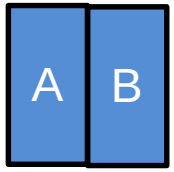
ALGORITMO DE INTERPREDICCIÓN

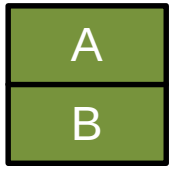


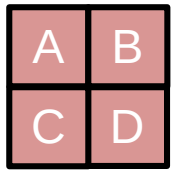
Decisión del mejor modo

8x8

J^{A}_{8x8}	$J^{A,A}_{8x4} + J^{0,B}_{8x4}$	$J^{A,A}_{4x8} + J^{A,B}_{4x8}$	$J^{A,A}_{4x4} + J^{A,B}_{4x4} + J^{A,C}_{4x4} + J^{A,D}_{4x4}$	Mínimo?	JA
J^{B}_{8x8}	$J^{B,A}_{8x4} + J^{B,B}_{8x4}$	$J^{B,A}_{4x8} + J^{B,B}_{4x8}$	$J^{B,A}_{4x4} + J^{B,B}_{4x4} + J^{B,C}_{4x4} + J^{B,D}_{4x4}$	Mínimo?	JB
J^{C}_{8x8}	$J^{C,A}_{8x4} + J^{C,B}_{8x4}$	$J^{C,A}_{4x8} + J^{C,B}_{4x8}$	$J^{C,A}_{4x4} + J^{C,B}_{4x4} + J^{C,C}_{4x4} + J^{C,D}_{4x4}$	Mínimo?	JC
J^{D}_{8x8}	$J^{D,A}_{8x4} + J^{D,B}_{8x4}$	$J^{D,A}_{4x8} + J^{D,B}_{4x8}$	$J^{D,A}_{4x4} + J^{D,B}_{4x4} + J^{D,C}_{4x4} + J^{D,D}_{4x4}$	Mínimo?	JD

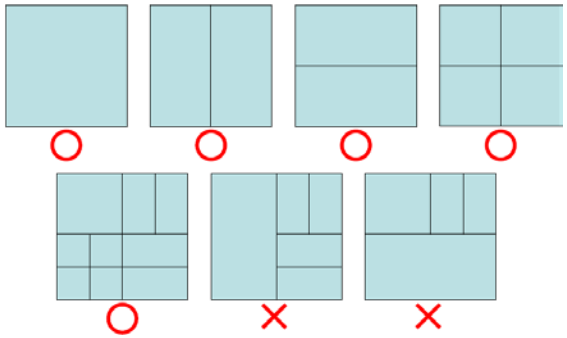


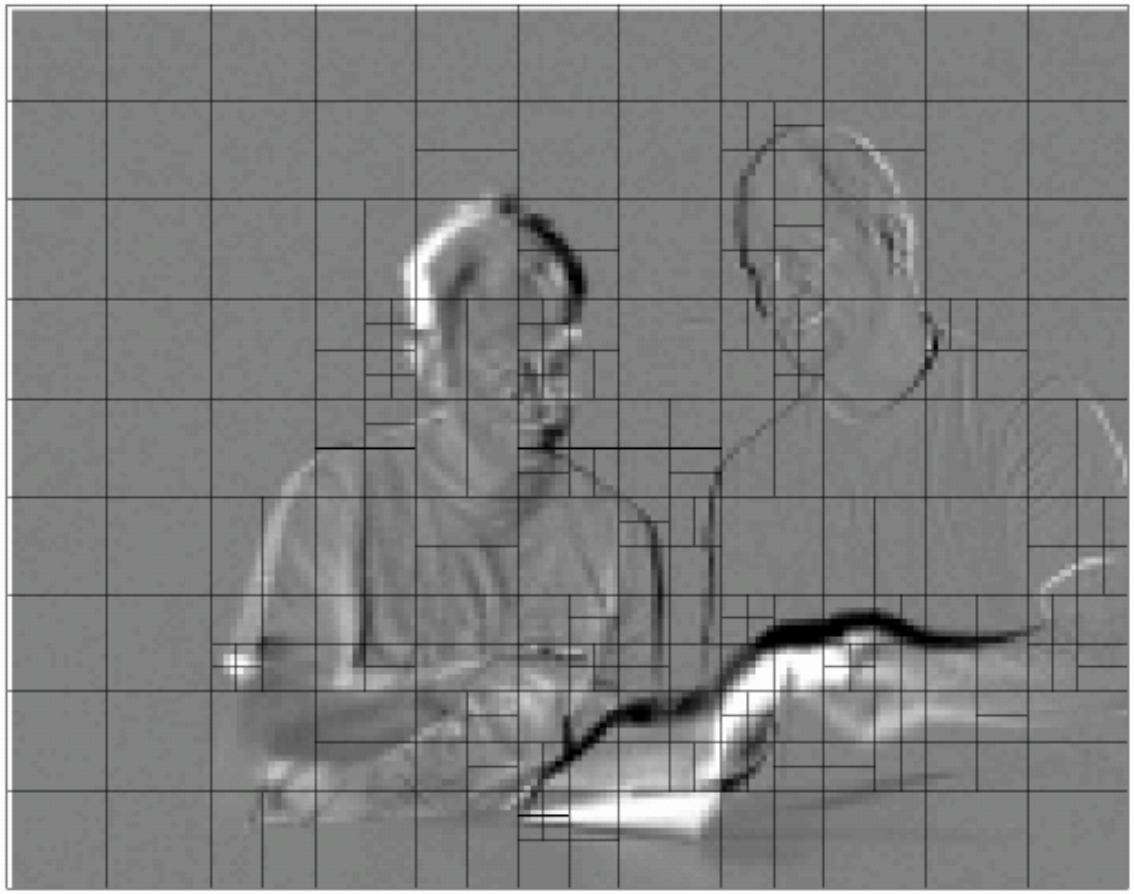




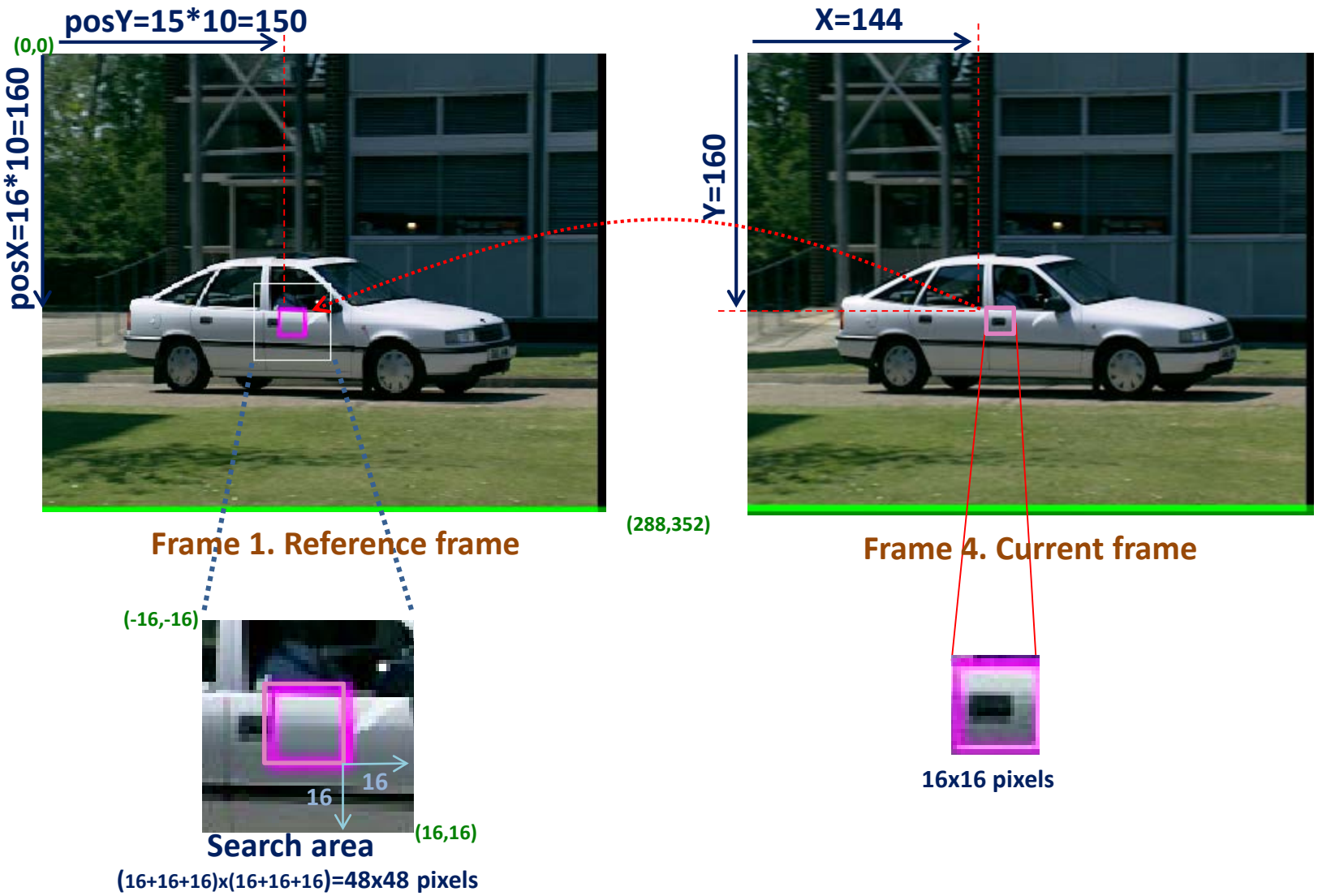
J_{16x16}	$J^{A}_{16x8} + J^{B}_{16x8}$	$J^{A}_{8x16} + J^{B}_{8x16}$	$JA + JB + JC + JD$

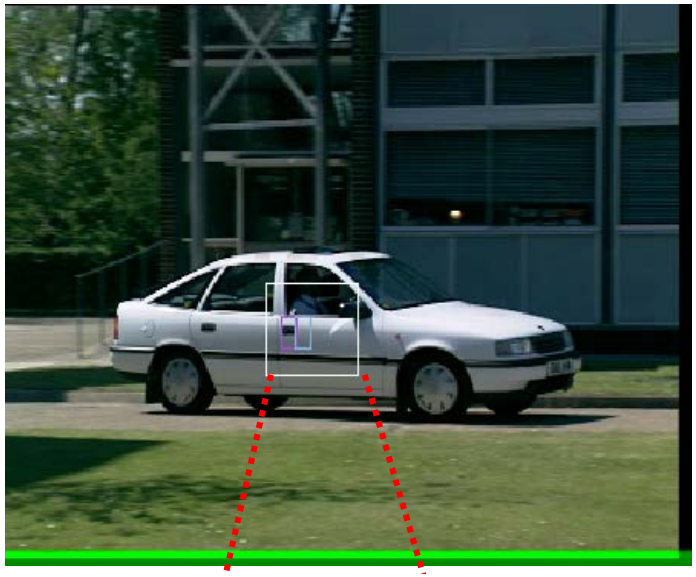
Selección del mejor modo: Valor mínimo del coste total



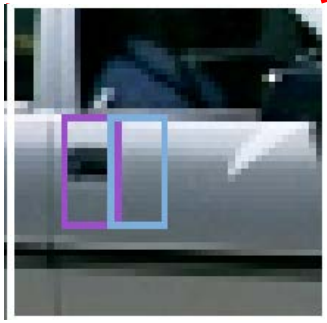


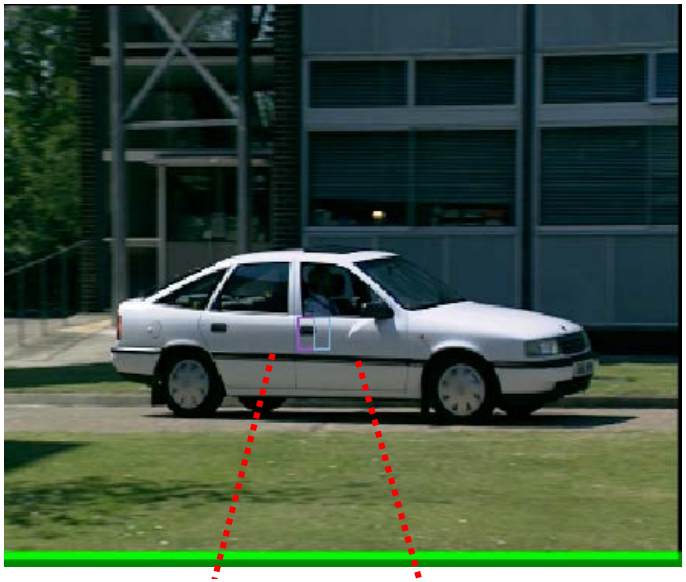
Caso práctico : Fichero *CasoPracticoEstimaciónMovimientoH264.m*



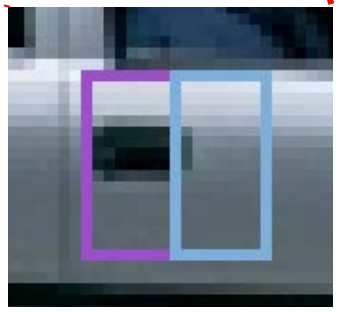


Frame 1. Reference frame. Partición MB





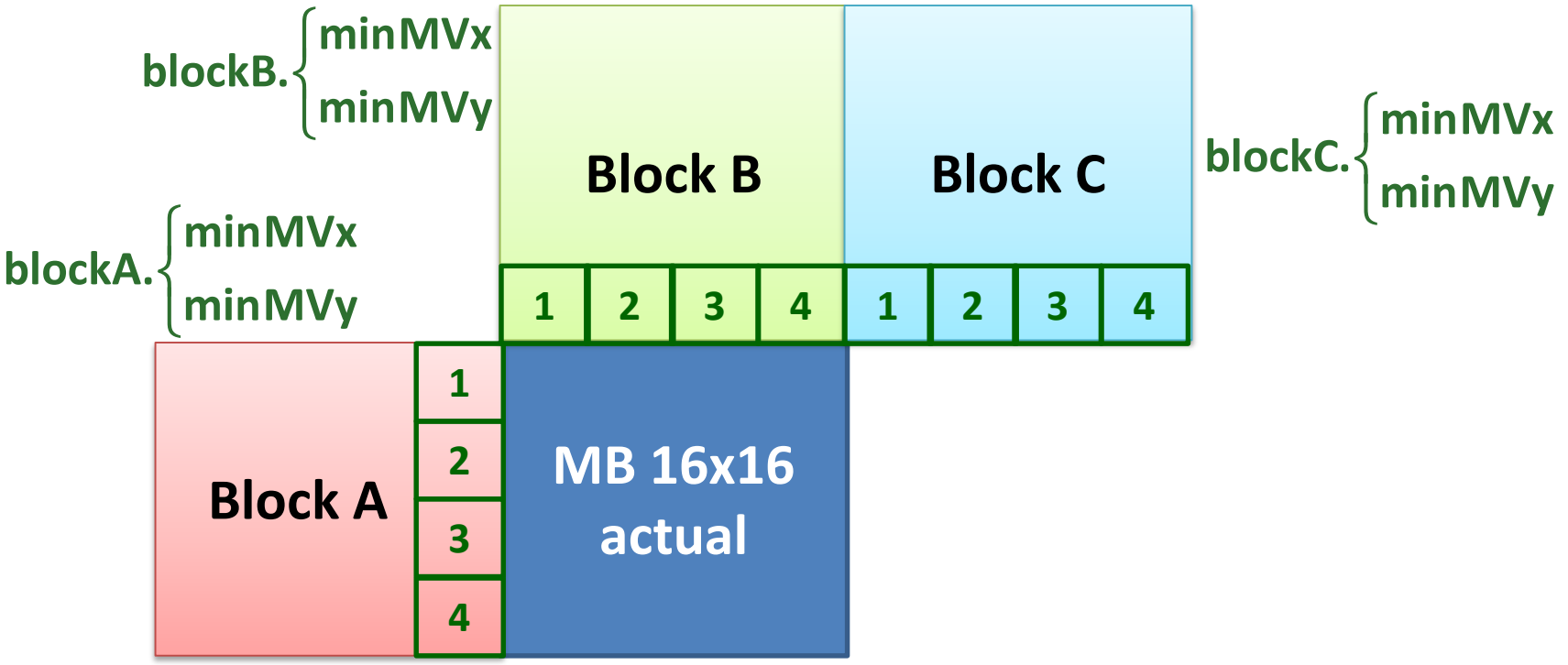
Frame 4. Current frame. Partición MB



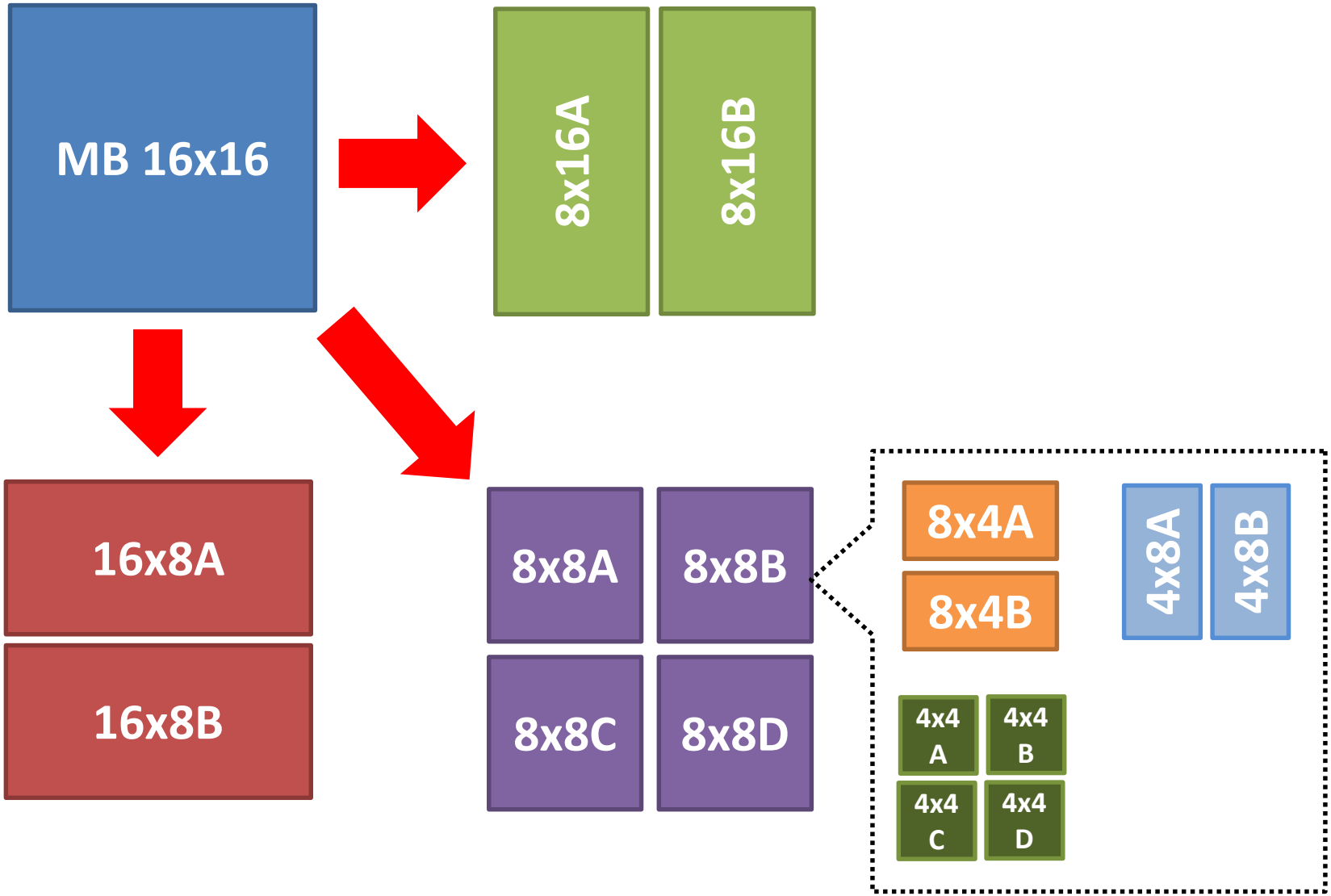
- **Parámetros**
  - **QP (=30)**
  - **Precision (=2)**
    - **2 → Entero**
    - **1 → Entero y ½ pixel**
    - **0 → Entero, ½ pixel y ¼ pixel**
  - **PosX (=15*5) y PosY (=15*5) . Posición del MB 16x16 en la imagen actual. Debe ser múltiplo de 16.**
  - **searchArea (=8). Número de pixels que rodean al MB 16x16.**
  - **Selección de los distintos macrobloques a través de las variables:**
    - **InterSearch16x16 (=1)**
    - **InterSearch16x8 (=1)**
    - **InterSearch8x16 (=1)**

- InterSearch8x8 (=1)
- InterSearch8x4 (=1)
- InterSearch4x8 (=1)
- InterSearch4x4 (=1)

o **Vectores de predicción de los MB 16x16 contiguos:**



➢ Diferentes particiones del MB 16x16 en blocks



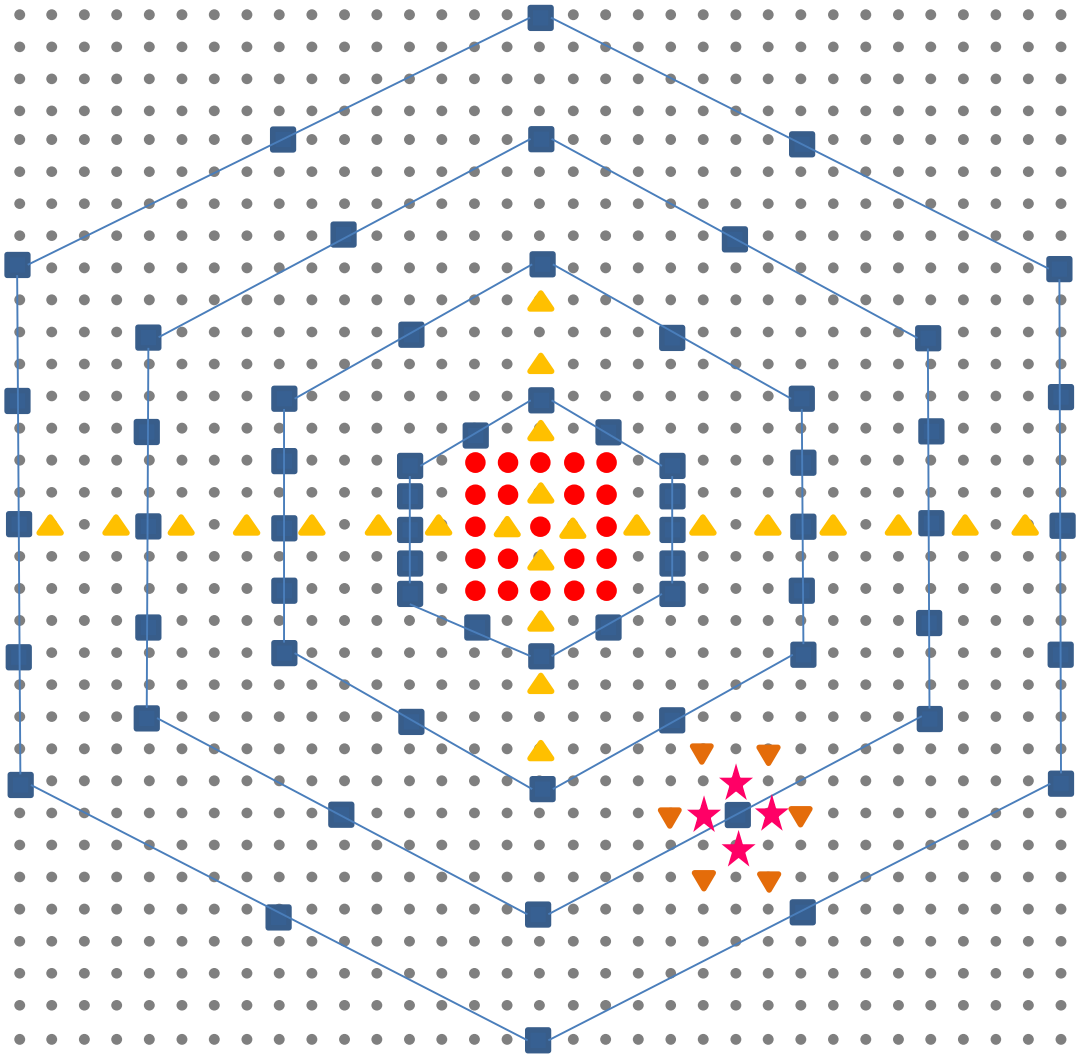
cmd) EncodeH264MSWin.exe –f configuracion.cfg

```
.....
SearchRange          = 16  # Max search range

DisableSubpelME      =  0  # Disable Subpixel Motion Estimation (0=off/default, 1=on)

QPISlice       = 30  # Quant. param for I Slices (0-51)
QPPSlice       = 30  # Quant. param for P Slices (0-51)

UseHadamard     =  1  # Hadamard transform (0=not used, 1=used for all subpel positions, 2=use
only for qpel)

InterSearch16x16     =  1  # Inter block search 16x16 (0=disable, 1=enable)
InterSearch16x8      =  1  # Inter block search 16x8  (0=disable, 1=enable)
InterSearch8x16      =  1  # Inter block search  8x16 (0=disable, 1=enable)
InterSearch8x8       =  1  # Inter block search  8x8  (0=disable, 1=enable)
InterSearch8x4       =  1  # Inter block search  8x4  (0=disable, 1=enable)
InterSearch4x8       =  1  # Inter block search  4x8  (0=disable, 1=enable)
InterSearch4x4       =  1  # Inter block search  4x4  (0=disable, 1=enable)
.....
```

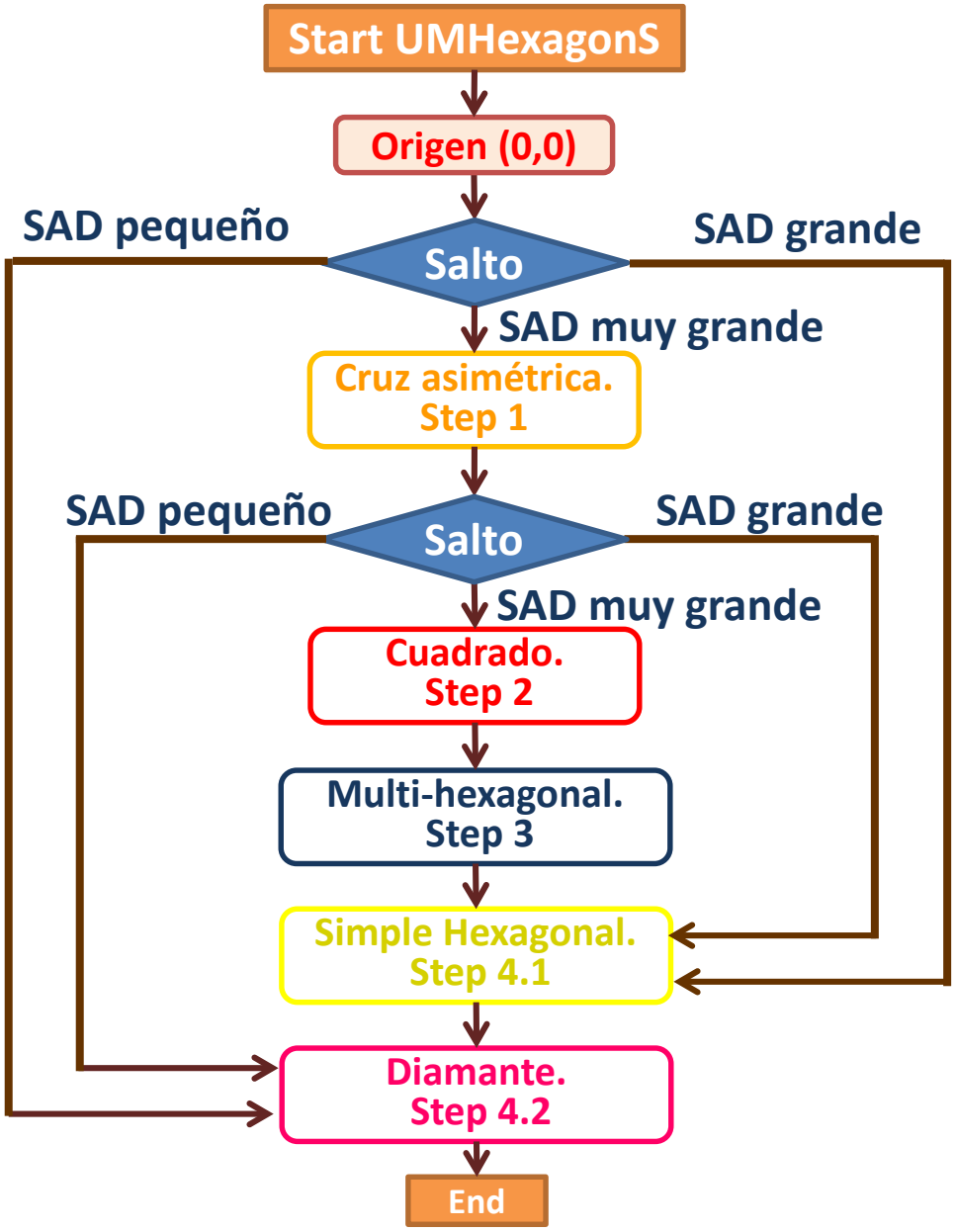
ALGORITMOS DE ESTIMACIÓN DE MOVIMIENTO EN EL H.264

- **Algoritmos de estimación de movimiento usados por el H.264 para seleccionar cada uno de los candidatos MV=(MV_x, MV_y):**
  - ***Full search***
  - **Algoritmo rápido UMHexagonS**
  - **EPZS o *Enhanced Predictive Zonal Search***

Algoritmo rápido UMHexagonS

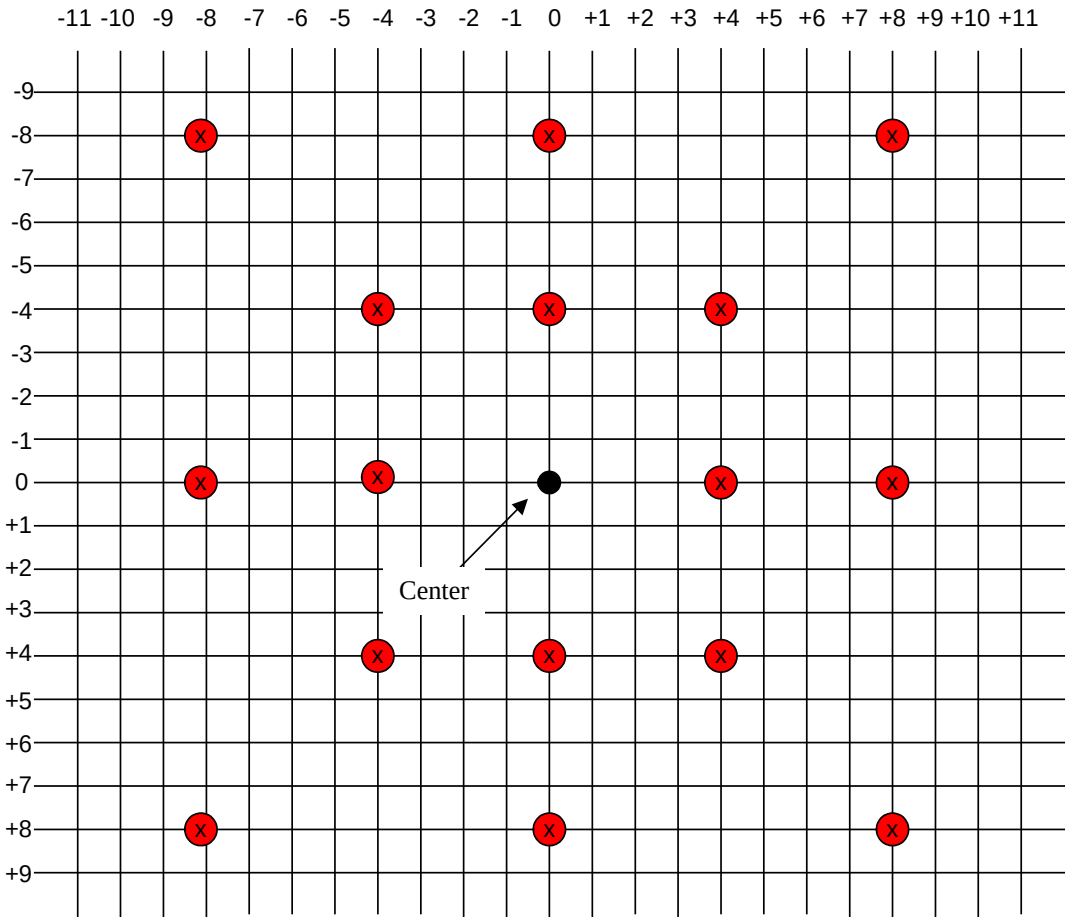


Step 1 ▲ Step 2 ● Step 3 ■ Step 4.1 ▼ Step 4.2 ★


Start UMHexagonS
Origen (0,0)
SAD pequeño
Salto
SAD grande
SAD muy grande
Cruz asimétrica.
Step 1
SAD pequeño
Salto
SAD grande
SAD muy grande
Cuadrado.
Step 2
Multi-hexagonal.
Step 3
Simple Hexagonal.
Step 4.1
Diamante.
Step 4.2
End

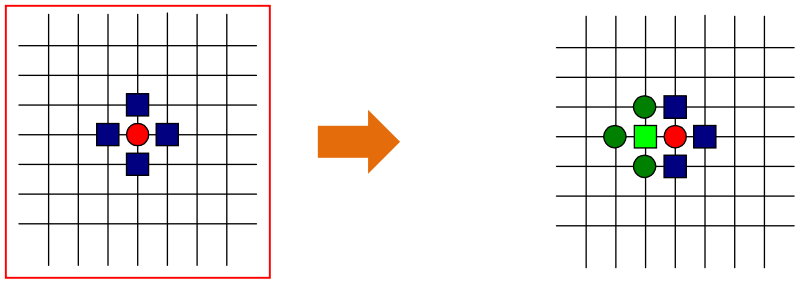
Algoritmo rápido Enhanced Predictive Zonal Search (EPZS)

Utiliza múltiples predicciones explotando la correlación, tanto espacial como temporal, que existe en la secuencia.

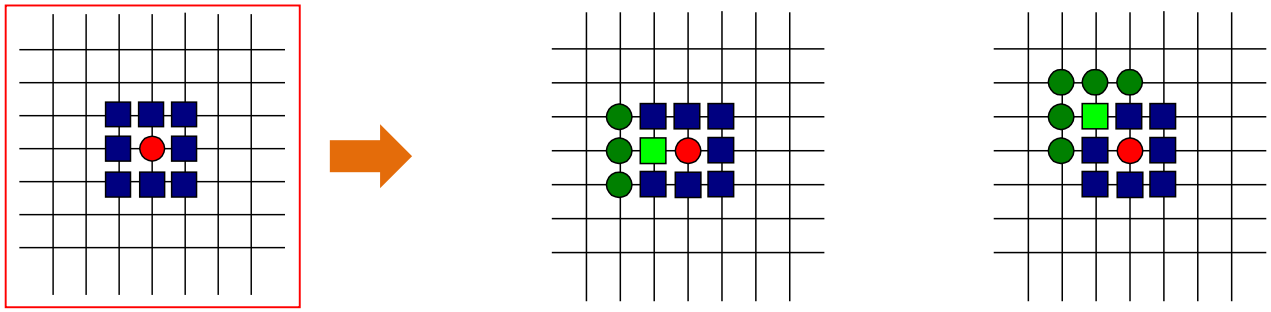


Diferentes refinamientos del EPZS

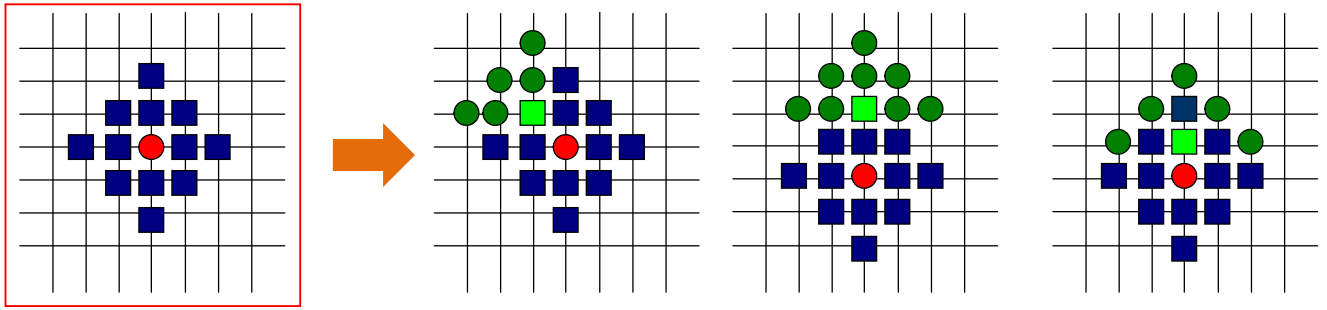
Diamante pequeño



Cuadrado/circular



Diamante extendido



cmd) EncodeH264MSWin.exe –f configuracion.cfg

.....

UseFME = 0 # Use fast motion estimation (0=disable, 1=enable)

.....

```
14683860    504952    496528
16152240    504952    496528
-------------------------------------------------------------------------------
Freq. for encoded bitstream       : 20
Hadamard transform                : Not used
Image format                      : 352x288
Error robustness                  : Off
Search range                      : 16
Total number of references        : 1
References for P slices           : 1
Total encoding time for the seq.  : 39.986 sec (0.75 fps)
Total ME time for sequence        : 32.531 sec
Sequence type                     : IPPP (QP: I 15, P 15)
Entropy coding method             : CAVLC
Profile/Level IDC                 : (66,30)
Search range restrictions         : none
RD-optimized mode decision        : not used
Data Partitioning Mode            : 1 partition
Output File Format                : H.264 Bit Stream File Format
Residue Color Transform           : not used
------------------ Average data all frames ------------------------------------
SNR Y(dB)                         : 45.24
SNR U(dB)                         : 46.00
SNR V(dB)                         : 46.01
Total bits                        : 881054 (I 1001480, P 7808896, NVB 168)
Bit rate (kbit/s)  @ 20.00 Hz     : 5873.70
Bits to avoid Startcode Emulation : 16
Bits for parameter sets           : 168
-------------------------------------------------------------------------------
```