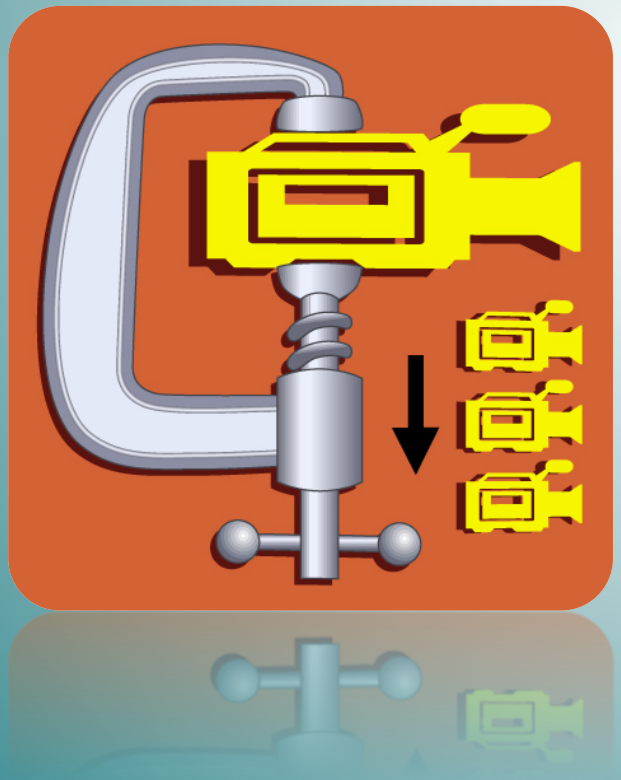
Compresión de Vídeo

Tema 2.3. Estimación de movimiento y compensación: conceptos básicos y algoritmos



Juan A. Michell Martín
Gustavo A. Ruiz Robredo

Departamento de Electrónica y Computadores

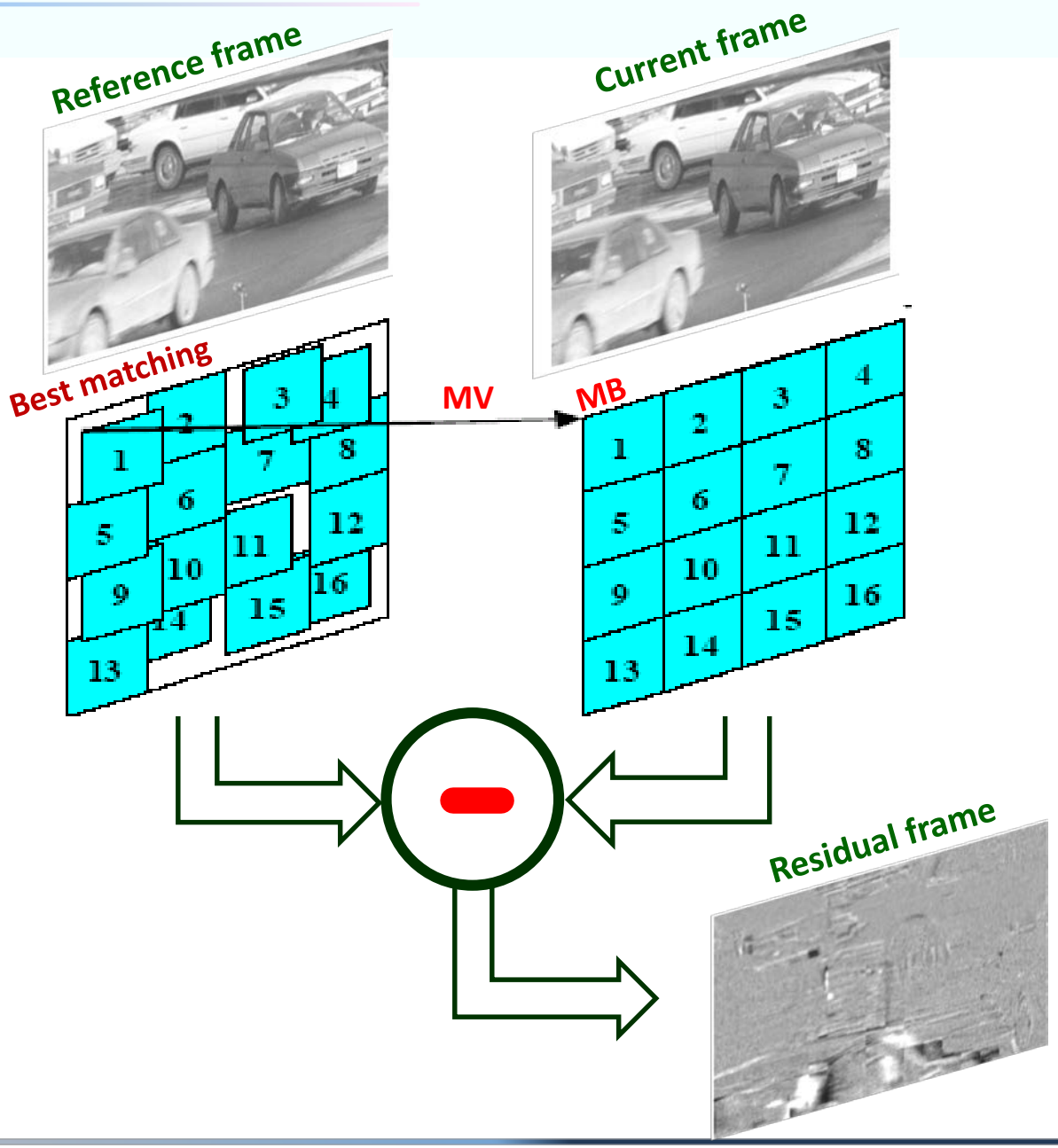
INTRODUCCIÓN

- La estimación de movimiento y compensación es una parte muy importante del proceso de codificación de vídeo y se utiliza en los estándares y códecs más populares.
- Permite reducir de una manera eficiente la redundancia temporal en secuencias de video.
- Constituye el 70% del coste computacional del proceso de codificación del video.
- Básicamente, este proceso consiste en 3 pasos:
  1. Para cada uno de los macrobloques (MB) de una imagen actual encontrar su mejor emparejamiento (*match*) de otros MB pertenecientes a una o varias imágenes de referencia. Genera como resultado el vector de movimiento MV={mv_x, mv_y}.
  2. El mejor *match* encontrado constituye la predicción de ese MB. Ambos MB se restan para obtener el MB residual, proceso conocido como compensación de movimiento.
  3. Finalmente, ese MB residual es codificado y comprimido junto con su MV.

Video

Estimación de Movimiento

Compensación



Estimación de Movimiento

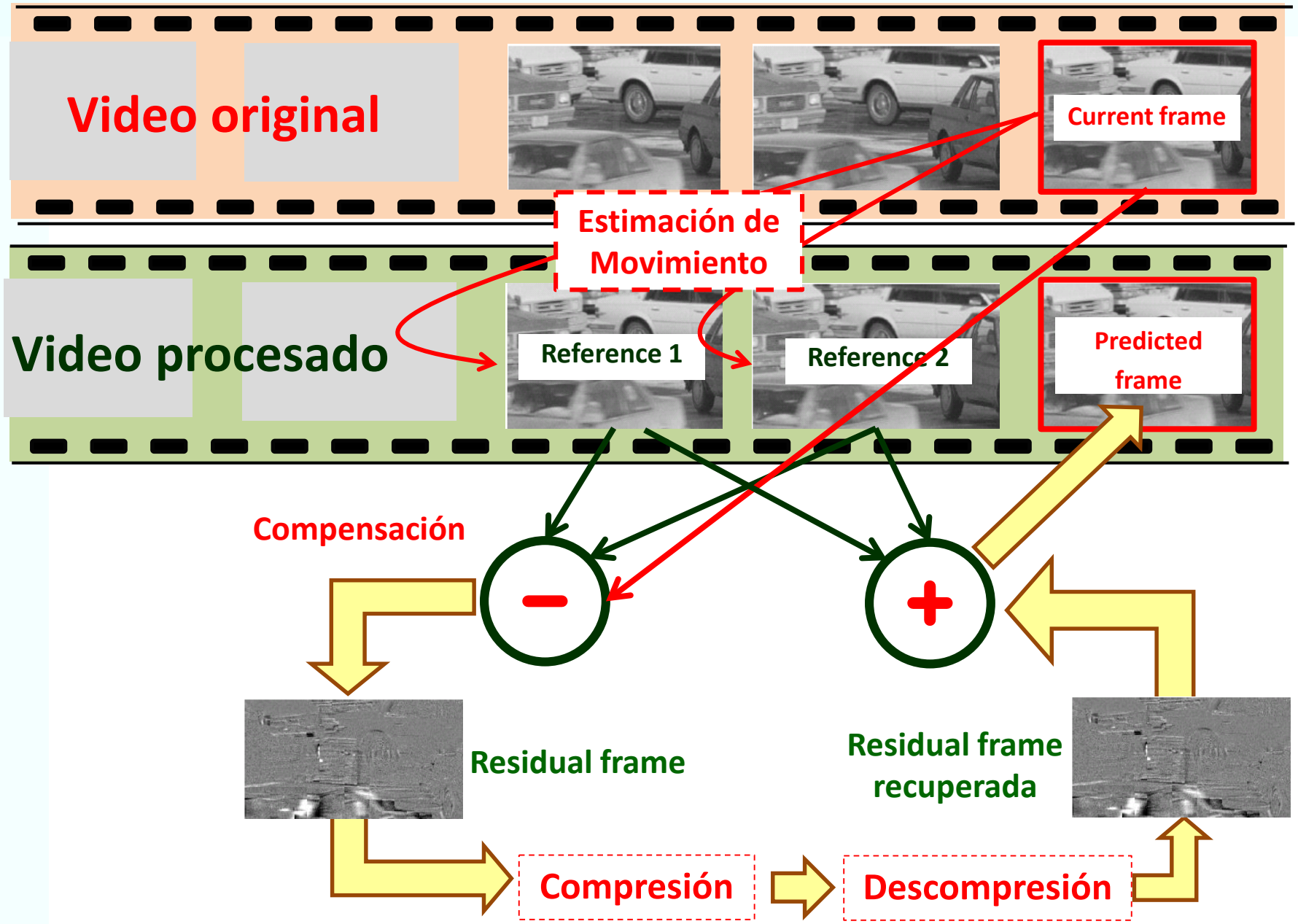
Reference frame

Current frame

Imágenes diferencia

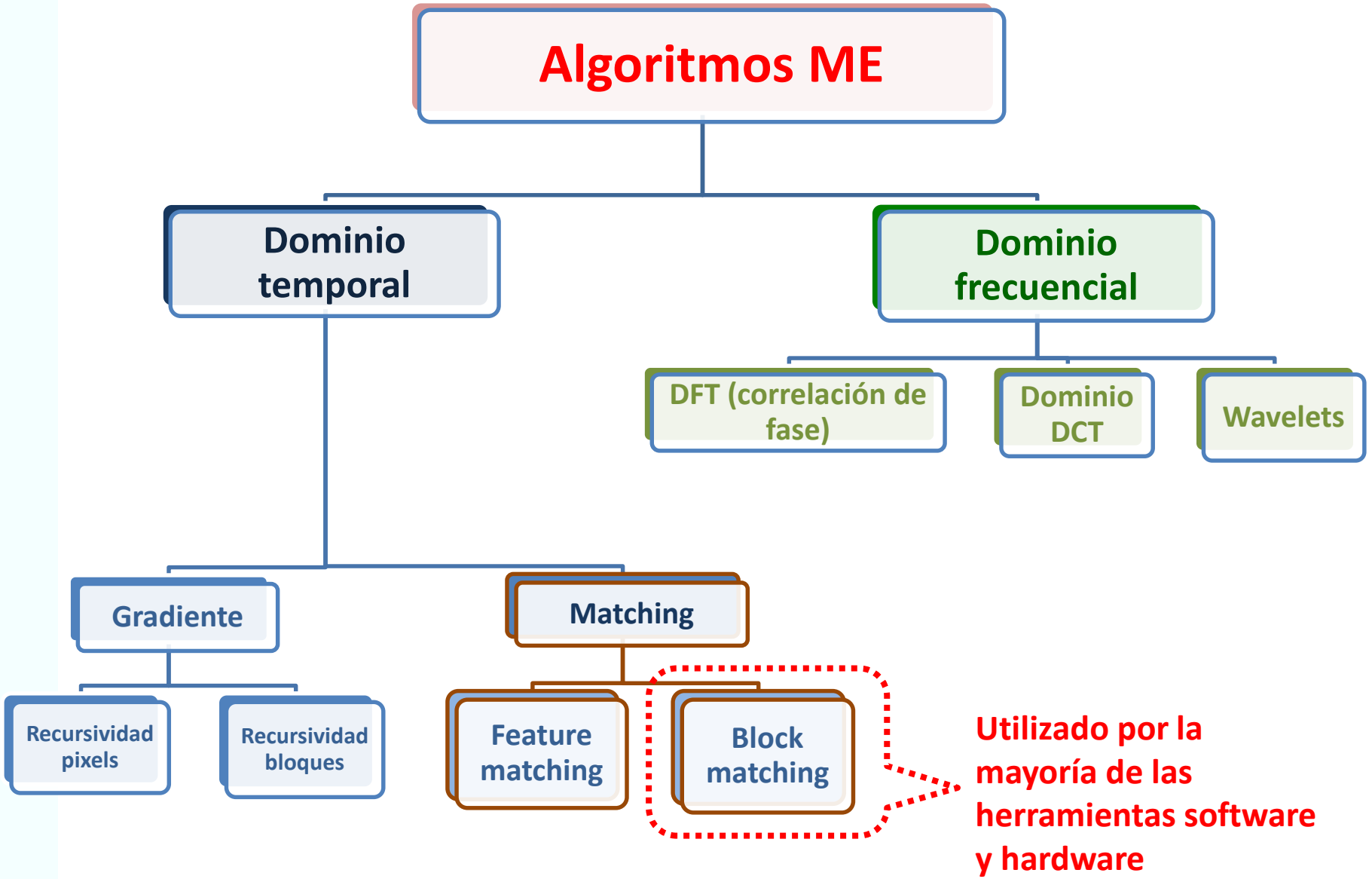
SIN estimación de movimiento

CON estimación de movimiento


Video original
Current frame
Estimación de Movimiento
Video procesado
Reference 1
Reference 2
Predicted frame
Compensación
Residual frame
Residual frame recuperada
Compresión
Descompresión

ALGORITMOS DE ESTIMACIÓN DE MOVIMIENTO

- **Clasificación de los algoritmos de ME:**
  - **Algoritmos dominio de la frecuencia: Se aplican sobre los coeficientes de transformadas DFT, DCT o wavelets.**
  - **Algoritmos de dominio del tiempo: Estos forman parte de algoritmos de coincidencia (*matching*) y algoritmos basados en gradiente.**
    - **Algoritmos basados en sus características: Búsqueda de coincidencias con la meta-información/datos del bloque actual con la del bloque en el marco de referencia.**
    - **Algoritmos basados en su coincidencia (*block matching*): Búsqueda de la máxima coincidencia entre todos o algunos de píxeles en el bloque actual con el bloque en el área de búsqueda de marco de referencia basado en alguna función de coste.**


Algoritmos ME
Dominio temporal
Dominio frecuencial
DFT (correlación de fase)
Dominio DCT
Wavelets
Gradiente
Matching
Recursividad pixels
Recursividad bloques
Feature matching
Block matching
Utilizado por la mayoría de las herramientas software y hardware

- **La complejidad computacional de los algoritmos de estimación de movimiento puede ser determinada por tres factores:**
  - **El algoritmo de búsqueda (*search algorithm*). Decide la complejidad general de cálculo y la precisión de la estimación de movimiento.**
  - **Área de búsqueda (*search area*). Zona de búsqueda para encontrar la mejor combinación.**
  - **Función de coste: Métrica que indica el mejor resultado de coincidencia (*match*).**

FUNCIONES COSTE

- La estimación de movimiento basado en el algoritmo de *block-matching* calcula el vector de movimiento (mv_x,mv_y) óptimo a partir de una función coste. Esta función indica el nivel de coincidencia entre dos bloques I_C e I_R.
- Las diferentes funciones coste propuestas varían en términos de complejidad y eficiencia.

SAD (*Sum of Absolute Differences*)

El SAD es la implementación práctica del MAE eliminar la división por MN.

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

$$(mv_x, mv_y) = \min_{dx,dy}\{SAD(dx,dy)\}$$

MSE (*Mean Square Error*)

El MSE es uno de los más populares, a pesar de tener alta complejidad computacional, por ser muy próximo a las características de percepción del ojo humano.

$$MSE(d_x, d_y) = \frac{1}{M \cdot N} \sum_{i=1}^{N} \sum_{j=1}^{M} \left[I_C(i,j) - I_R(i + d_x, j + d_y) \right]^2$$

$$(mv_x, mv_y) = \min_{dx,dy} \{MSE(dx, dy)\}$$

MAE (*Mean Absolute Error*)

El MAE (o también conocido como como MAD) es muy popular por su sencilla implementación en VLSI.

$$MAE(d_x, d_y) = \frac{1}{M \cdot N} \sum_{i=1}^{N} \sum_{j=1}^{M} \left| I_C(i,j) - I_R(i + d_x, j + d_y) \right|$$

$$(mv_x, mv_y) = \min_{dx,dy} \{MAE(dx, dy)\}$$

CCF (*cross correlation function*)

$$CCF(d_x, d_y) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} I_C(i,j) I_R(i+d_x, j+d_y)}{\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{M} I_C^2(i,j)} \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{M} I_R^2(i+d_x, j+d_y)}}$$

$$(mv_x, mv_y) = \max_{dx,dy}\{CCF(dx,dy)\}$$

Su alto coste de cómputo los hace impracticable en aplicaciones de codificación de video en tiempo real.

PDC (*Pixel Difference Classification*)

$$PDC(d_x, d_y) = \sum_{i=1}^{N}\sum_{j=1}^{M} T(d_x, d_y)$$

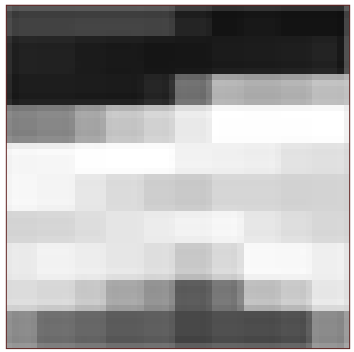
$$T(d_x, d_y) = \begin{cases} 1 & \text{si } \left| I_C(i,j) - I_R(i+d_x, j+d_y) \right| < \text{Threshold} \\ 0 & \text{otherwise} \end{cases}$$

$$(mv_x, mv_y) = \max_{dx,dy}\{PDC(dx,dy)\}$$

Es sencillo, pero existe dificultad para establecer el valor adecuado de Theshold.

Ejemplo: Uso del SAD en estimación de movimiento

Search area



$I_R(i,j)=$

$$\begin{bmatrix} 75 & 77 & 76 & 75 & 74 & 73 & 72 & 76 & 83 & 82 \\ 84 & 83 & 83 & 84 & 84 & 81 & 54 & 35 & 43 & 38 \\ 56 & 55 & 54 & 45 & 36 & 34 & 39 & 46 & 47 & 49 \\ 44 & 45 & 45 & 43 & 42 & 49 & 113 & 170 & 163 & 168 \\ 132 & 132 & 136 & 160 & 187 & 196 & 217 & 246 & 240 & 235 \\ 226 & 226 & 227 & 236 & 251 & 246 & 224 & 222 & 220 & 211 \\ 227 & 228 & 225 & 214 & 204 & 193 & 189 & 200 & 200 & 196 \\ 202 & 199 & 201 & 207 & 214 & 219 & 224 & 228 & 215 & 207 \\ 204 & 218 & 225 & 221 & 215 & 207 & 188 & 203 & 230 & 229 \\ 213 & 203 & 199 & 181 & 154 & 133 & 86 & 112 & 172 & 186 \end{bmatrix}$$

Current MB

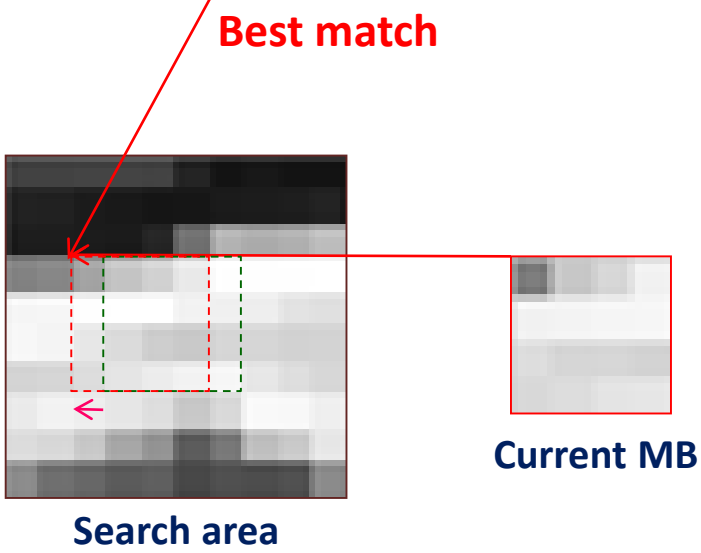
$I_C(i,j)=$

$$\begin{bmatrix} 39 & 39 & 75 & 103 \\ 127 & 127 & 190 & 203 \\ 226 & 230 & 232 & 233 \\ 213 & 209 & 203 & 201 \end{bmatrix}$$

$$SAD(0,0)=|75-39|+|77-39|+|76-75|+|75-103|+$$
$$+|84-127|+|83-127|+|83-190|+|84-203|+$$
$$+|56-226|+|55-230|+|54-232|+|45-233|+$$
$$+|44-213|+|45-209|+|45-203|+|43-201|=1776$$

$$SAD(1,0)=|77-39|+|76-39|+|75-75|+|74-103|+$$
$$+|83-127|+|83-127|+|84-190|+|84-203|+$$
$$+|55-226|+|54-230|+|45-232|+|36-233|+$$
$$+|45-213|+|45-209|+|43-203|+|42-201|=1799$$

$$SAD(i,j)=\begin{bmatrix} 1776 & 1799 & 1818 & 1794 & 1710 & 1608 & 1528 \\ 1563 & 1530 & 1487 & 1410 & 1313 & 1171 & 1001 \\ 1032 & 1020 & 973 & 806 & 592 & 392 & 294 \\ 288 & 235 & 214 & 238 & 417 & 584 & 734 \\ 630 & 745 & 864 & 985 & 1080 & 1107 & 1096 \\ 1078 & 1061 & 1035 & 970 & 891 & 888 & 906 \\ 883 & 920 & 986 & 1123 & 1217 & 1181 & 1097 \end{bmatrix}$$



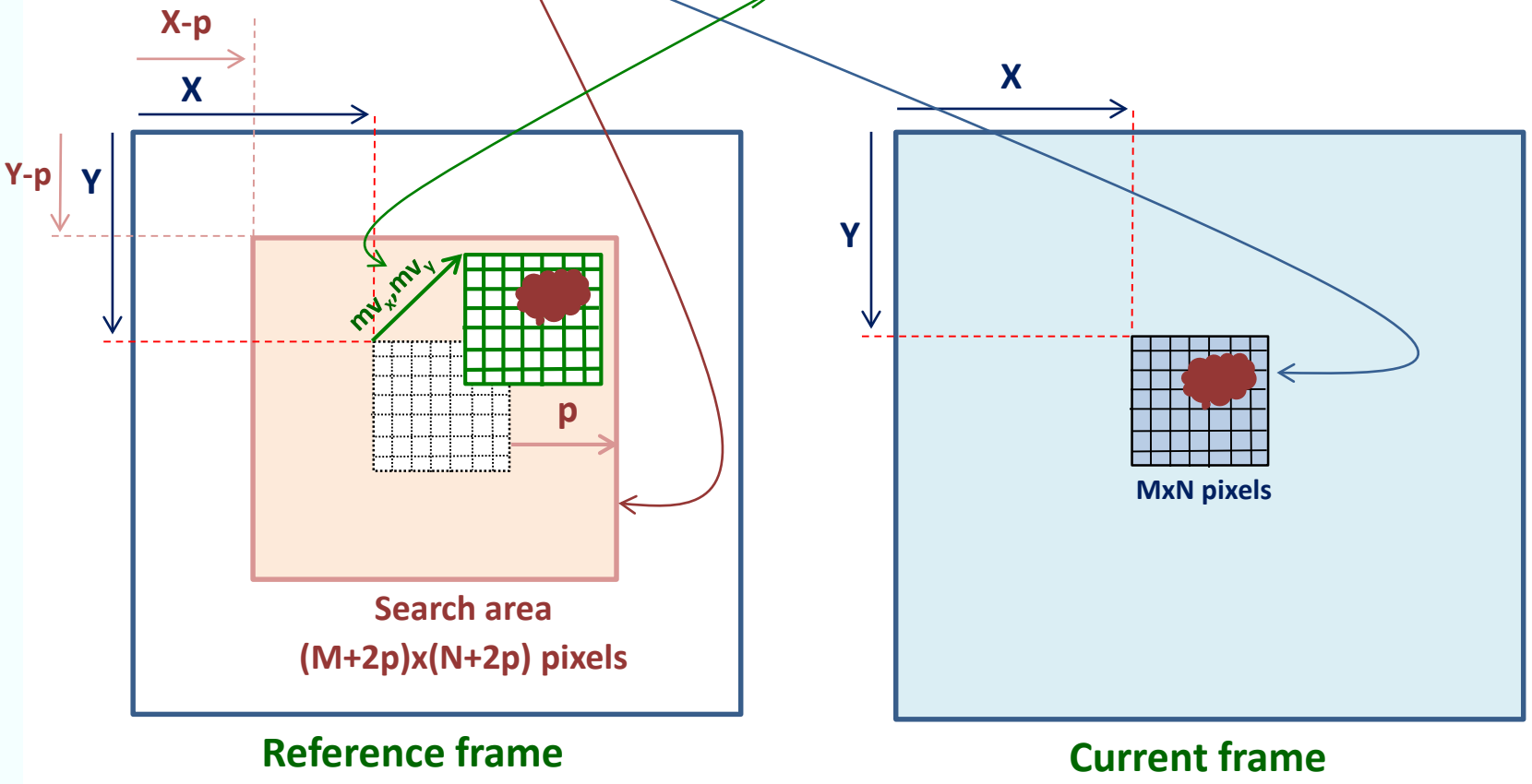
PRINCIPALES ALGORITMOS DE BÚSQUEDA

- Los algoritmos de estimación de movimiento (*motion estimation* o ME) buscan:
  - Reducir la complejidad computacional.
  - Mejorar la calidad visual en términos de representación de movimiento.
  - Proporcionar altos valores de compresión.

Búsqueda completa (*full-search*)

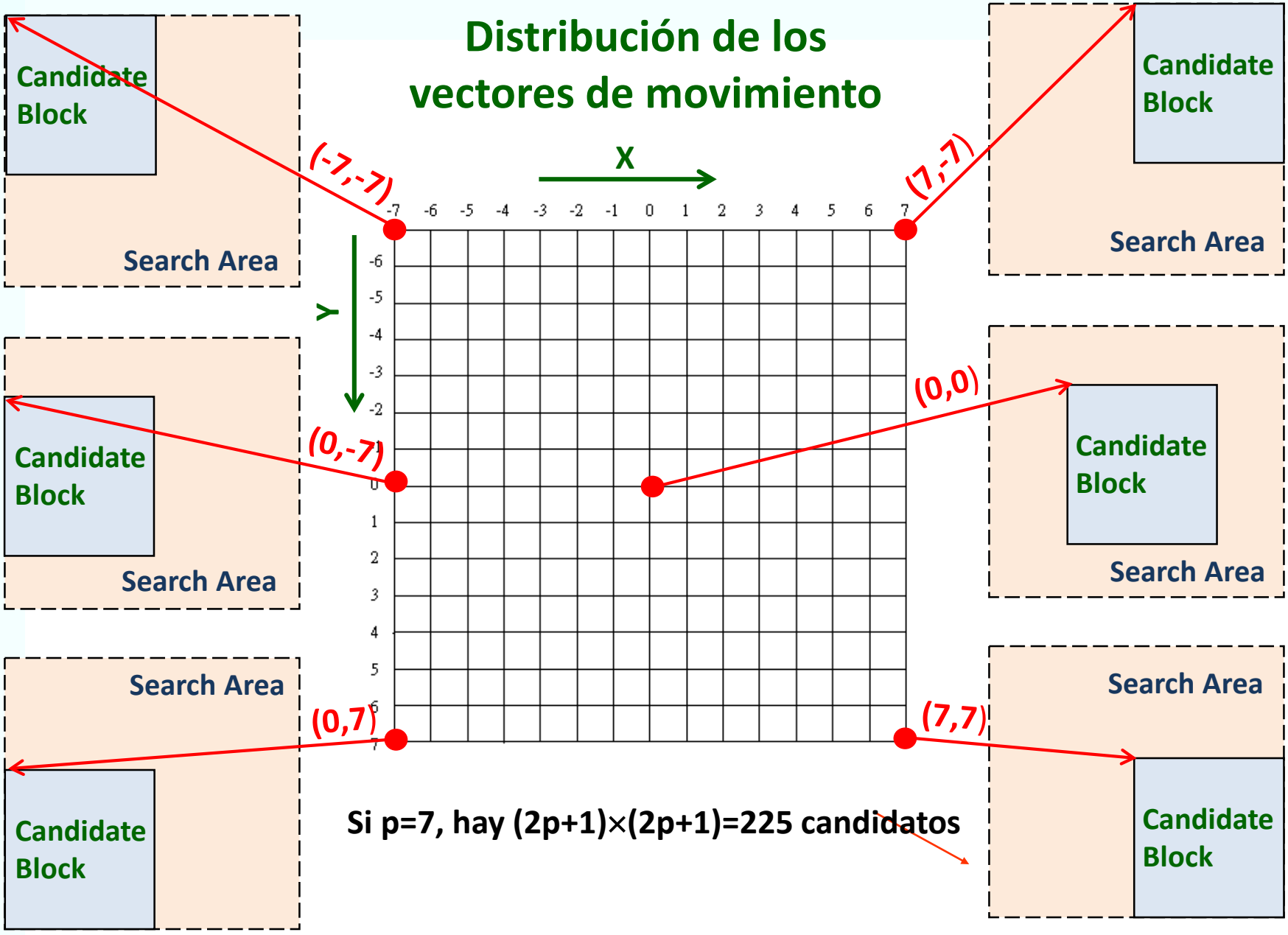
- Algoritmo de fuerza bruta. El bloque actual (*current block*) es comparado (*matched*) con cada uno de los bloques localizados en la *frame* de referencia (*reference frame*) hasta encontrar la mejor aproximación definida por la función coste.
- Es un algoritmo simple y muy preciso.
- No es eficiente porque exige un alto coste computacional.

- El bloque de MxN pixels en el *current frame* recorre exhaustivamente el área de búsqueda (*search area*) en el *reference frame*.
- La función coste proporciona la posición de la mejor coincidencia con su correspondiente vector de movimiento (mv_x, mv_y).



Número de búsquedas realizadas entre el bloque del *current frame* y el área de búsqueda (*search area*): (2p+1)(2p+1)

Distribución de los vectores de movimiento



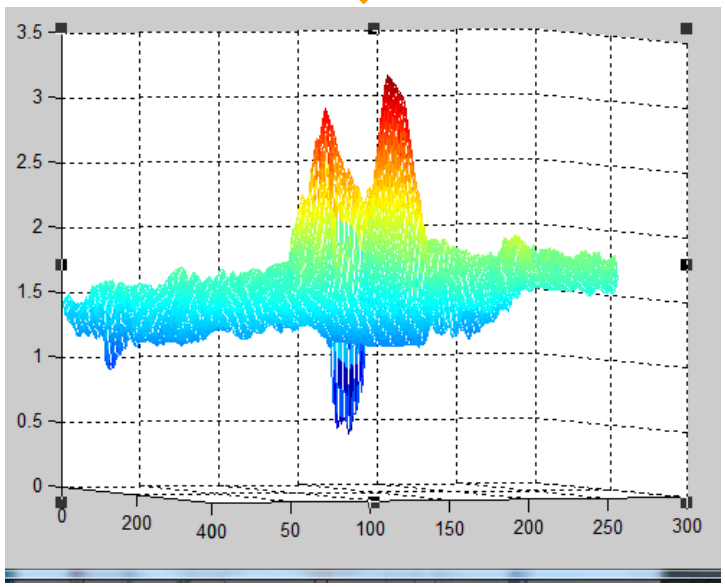
Si p=7, hay (2p+1)×(2p+1)=225 candidatos

Caso práctico 1: Fichero *CasoPracticoCalculoSADImagen.m*

Frame t-1

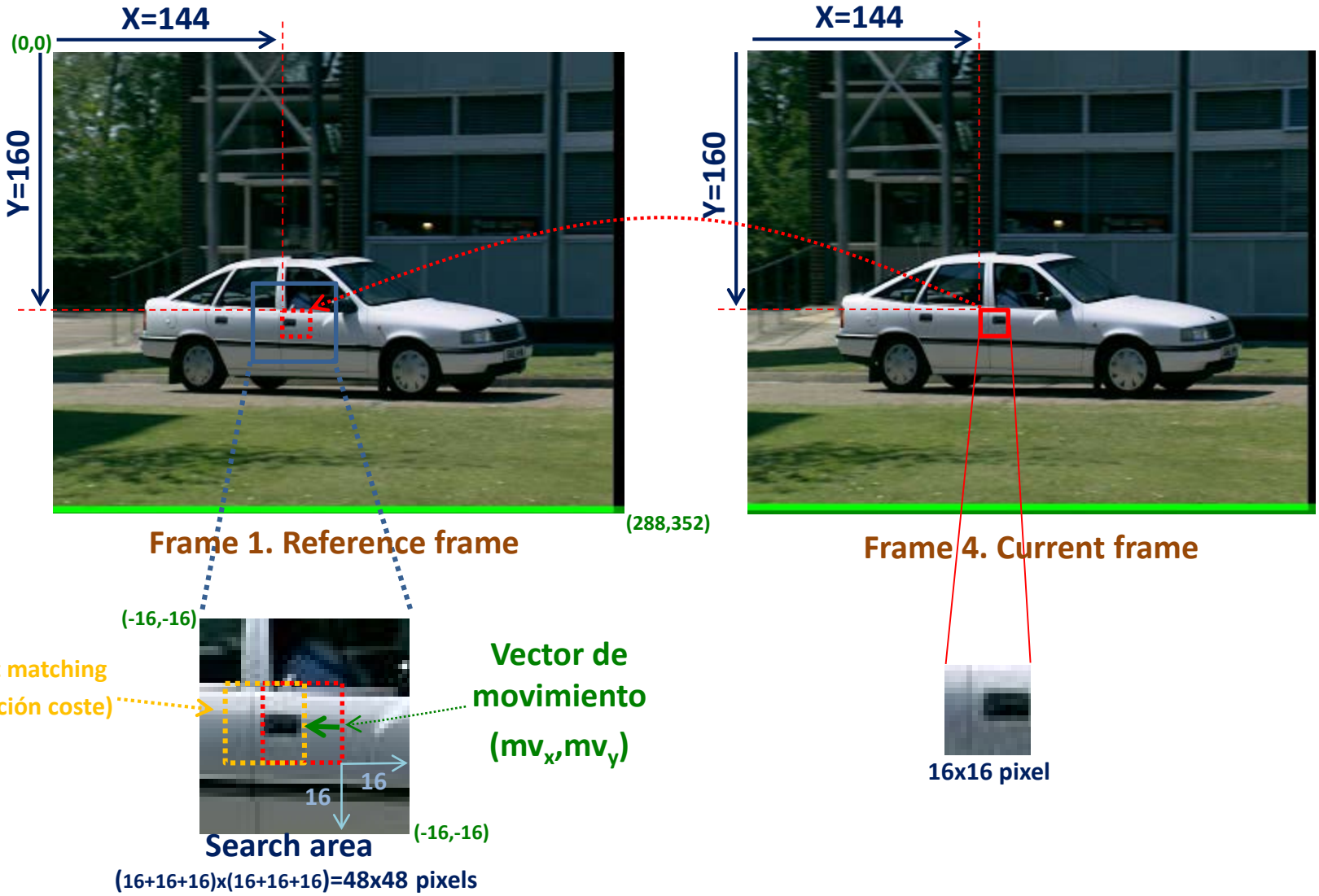
Frame t

Bloque 16x16



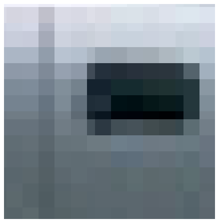
3D SAD

Caso práctico 2: Fichero *CasoPracticoEstimaciónMvtoBloque.m*



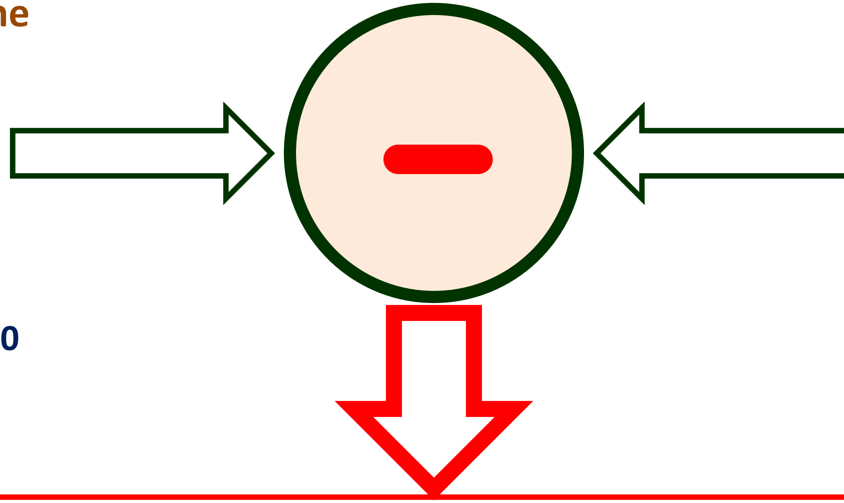
➢ Compensación

Frame 1.
Reference frame

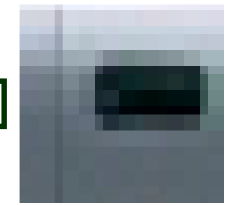


Best match

(mv_x=-9, mv_y=0



Frame 4.
Current frame



Current block

Y=

0	0	0	0	0	0	1	1	4	8	9	12	14	16	24	32
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0
0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	3	0	2	0	0	0	0	0	0	0	0	0	0	0
1	0	3	0	1	0	0	0	0	0	3	0	0	2	0	3
1	2	0	0	4	3	0	1	4	0	1	2	1	4	0	1

U=

1	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0
1	1	2	0	2	1	2	2
0	2	1	2	1	1	2	4
0	0	0	0	0	0	1	2
0	0	0	0	1	0	0	2
0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0

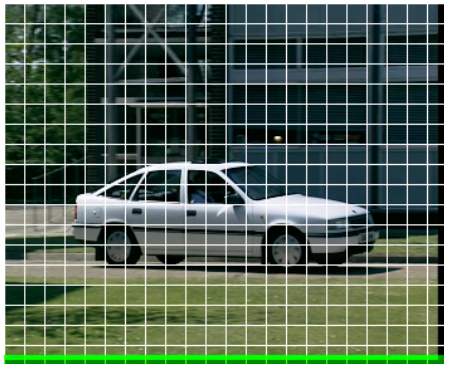
V=

1	0	2	1	0	0	0	0
3	2	3	2	1	2	1	1
0	0	1	0	4	2	0	2
0	0	0	1	2	1	0	5
1	0	0	1	3	0	0	4
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	1

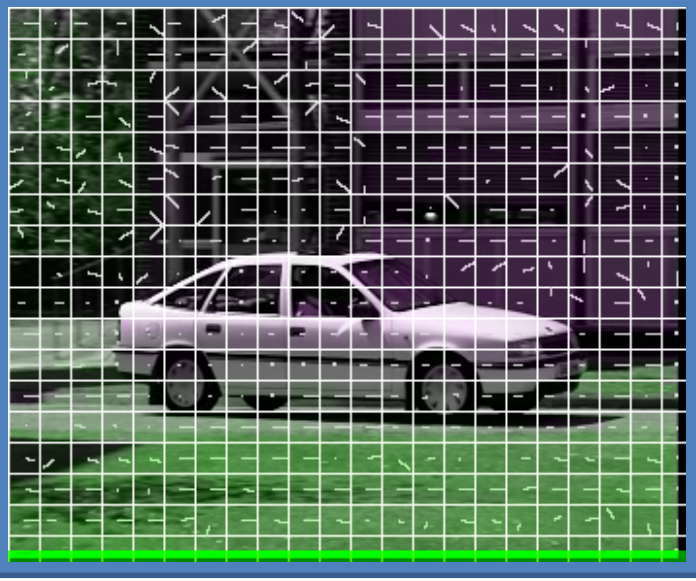
Caso práctico 3: Fichero *CasoPracticoEstimaciónMvto.m*

Extended Reference Frame

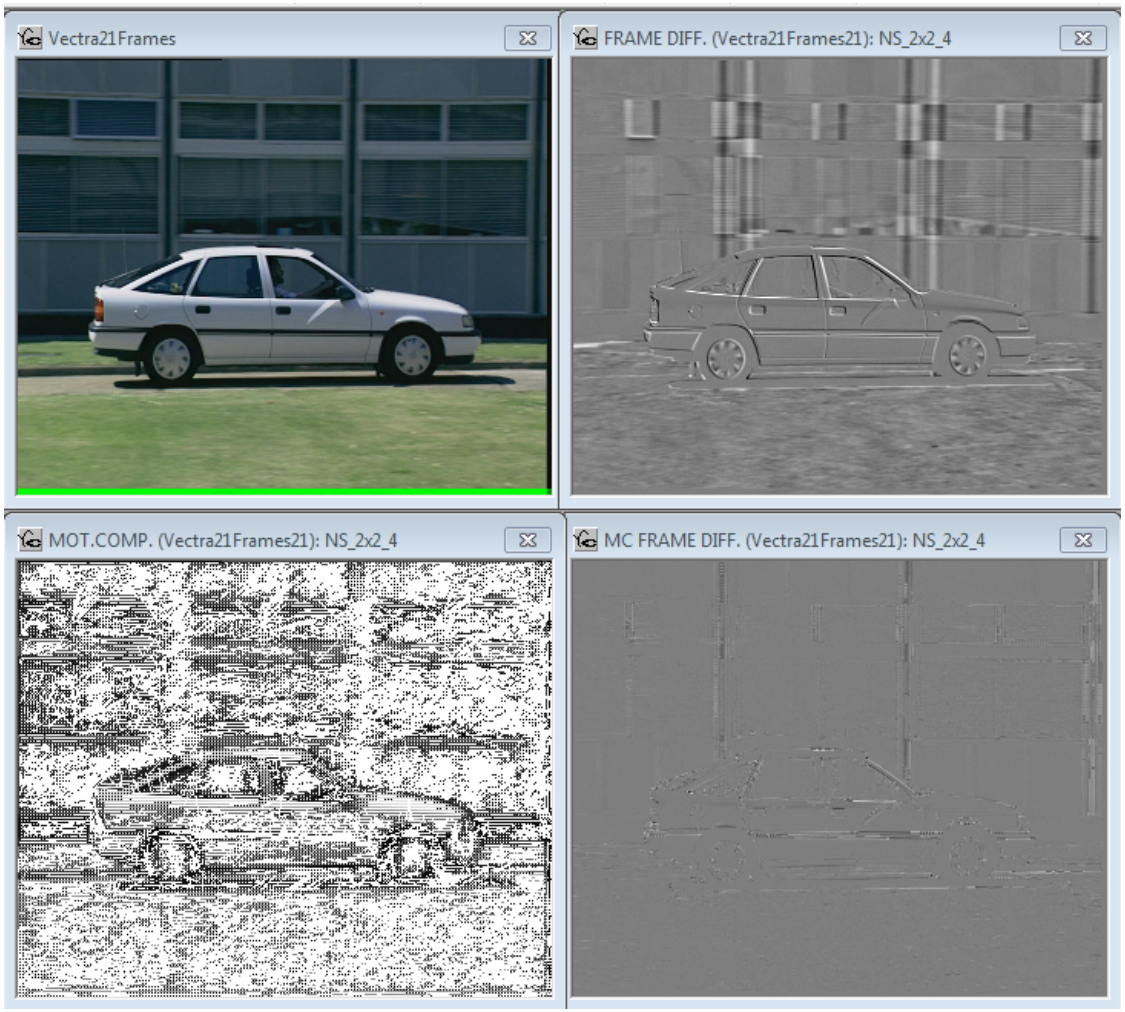
CurrentFrame



ESTIMACIÓN DE MOVIMIENTO

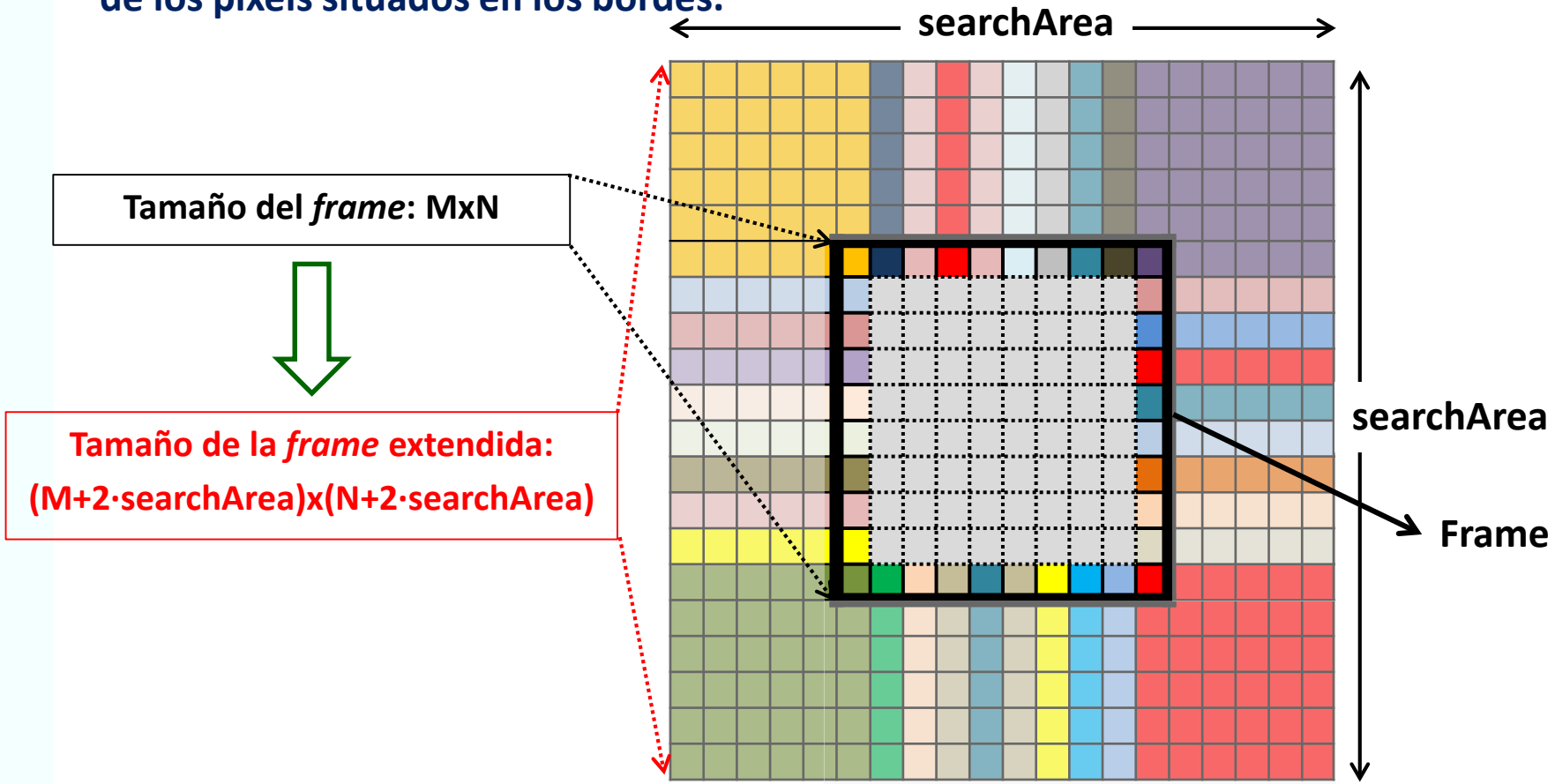


VCDemo tool: ME



Extensión del *frame*

- La región de búsqueda (*search area*) sobrepasa las dimensiones de la *frame* en sus bordes.
- Por ello, es necesario extender fuera de las dimensiones de la *frame* el valor de los pixels situados en los bordes.

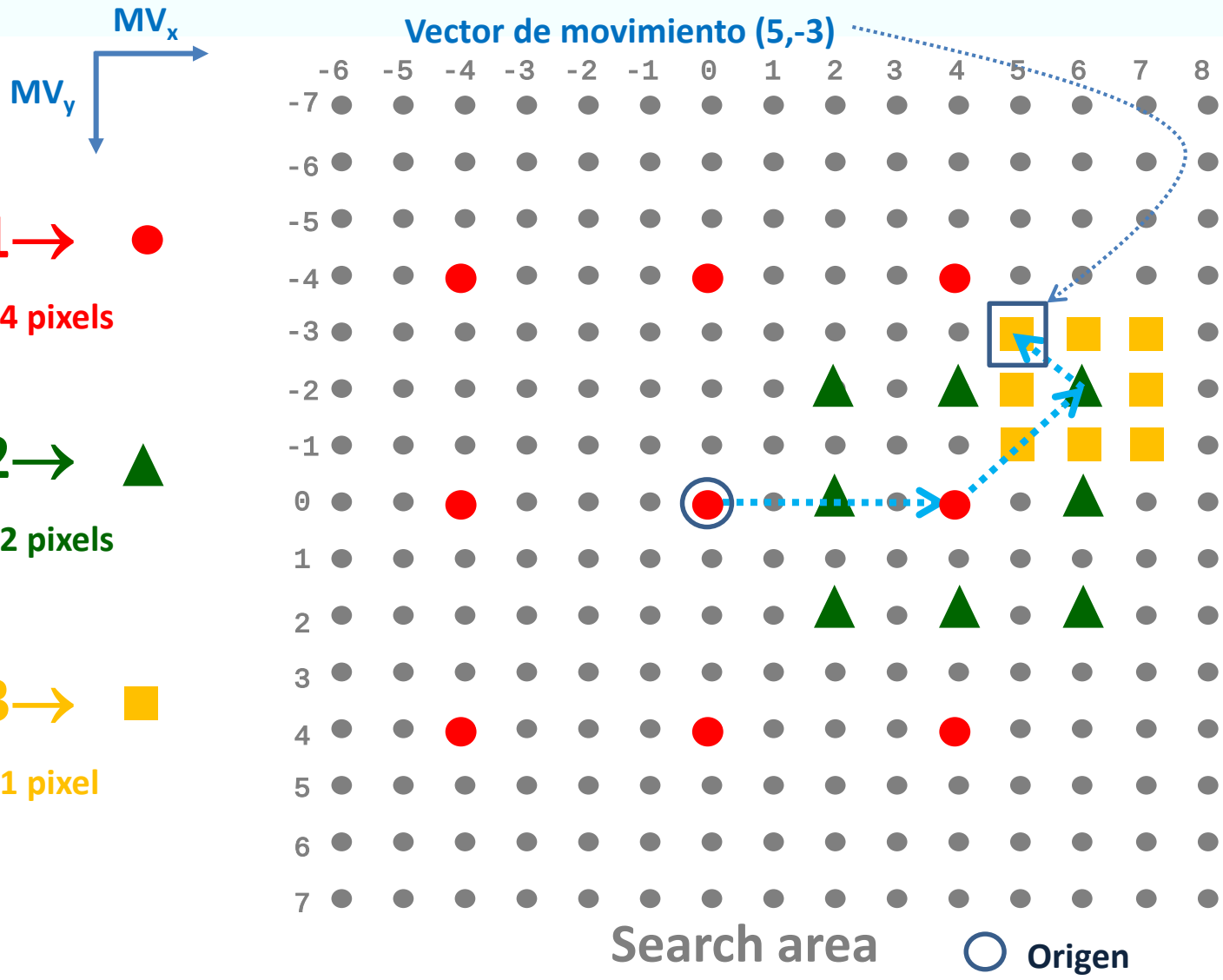


Algoritmos de ME rápidos (*Fast Motion Estimation*)

- Estos algoritmos rápidos han surgido por la necesidad de encontrar los requerimientos de bajo consumo de potencia de la mayoría de los equipos de tiempo real y reducir la complejidad de su cómputo.
- Se basan principalmente en reducir el número de puntos del área de búsqueda. Esta reducción en cómputo implica un coste en la calidad visual. Peligro en encontrar mínimos locales.
- Clasificación de los algoritmos rápidos:
  - Sub-muestreo del área de búsqueda:
    a) Area de búsqueda fija: TSS, OTS, NTSS, 4SS, OSA, Búsqueda de cruce, 2DLOG.
    b) Búsqueda de área adaptativa: Búsqueda basada en la zona o en espiral
  - Jerarquía/Multiresolución: Submuestreo, HPDS, Gaussian, pixel reducido.
  - Decimación pixel: 2:1, 4:1, 16:1.
  - Predicción: Temporal, espacial, lineal y no-lineal.
  - Coincidencia (*matching*) de las características: IPM, SEA, BBM, BFM, BPM.

Búsqueda tres pasos (*Three Step Search* o TSS)

- El algoritmo TSS emplea patrones rectangulares de búsqueda con diferentes tamaños.
- Fue introducido por Koga et al. en 1981.
- Se hizo muy popular por su sencillez y robustez con un rendimiento óptimo cercano al algoritmo de búsqueda completa.
- Obtiene los mejores vectores de movimiento en un patrón de búsqueda grueso a fino.
- Su mayor inconveniente es que se vuelve ineficaz para la estimación de movimiento pequeño.



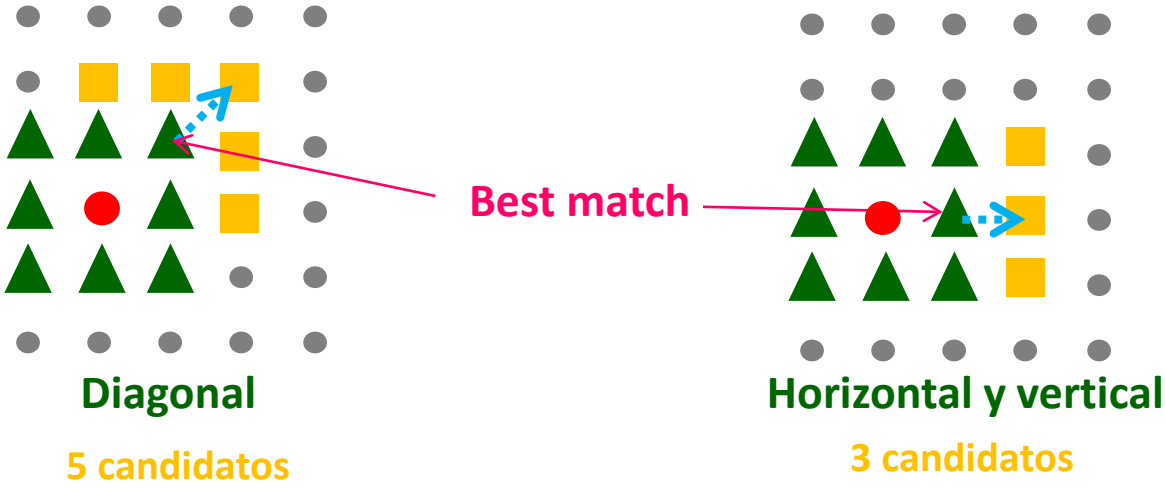
- **Nivel 1→** ●
  - Distancia 4 pixels
- **Nivel 2→** ▲
  - Distancia 2 pixels
- **Nivel 3→** ■
  - Distancia 1 pixel

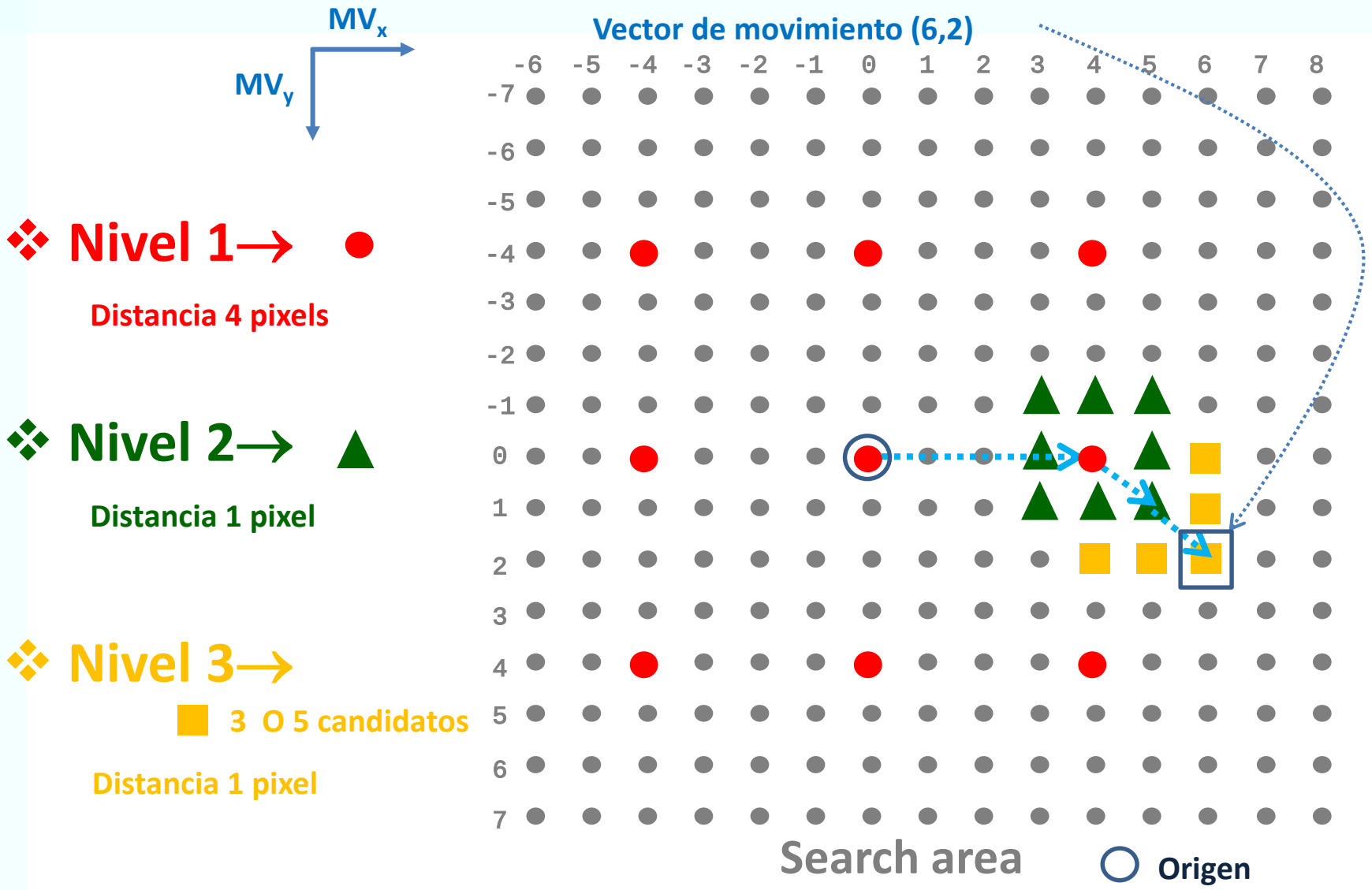
Número total de operaciones *block matching* = 9 + 8 + 8 = 25

❑ Búsqueda nueva de tres pasos (*New Three Step Search* o NTSS)

- ➢ Mejora algunos aspectos del TSS.

- ➢ Movimiento final de 1 pixel

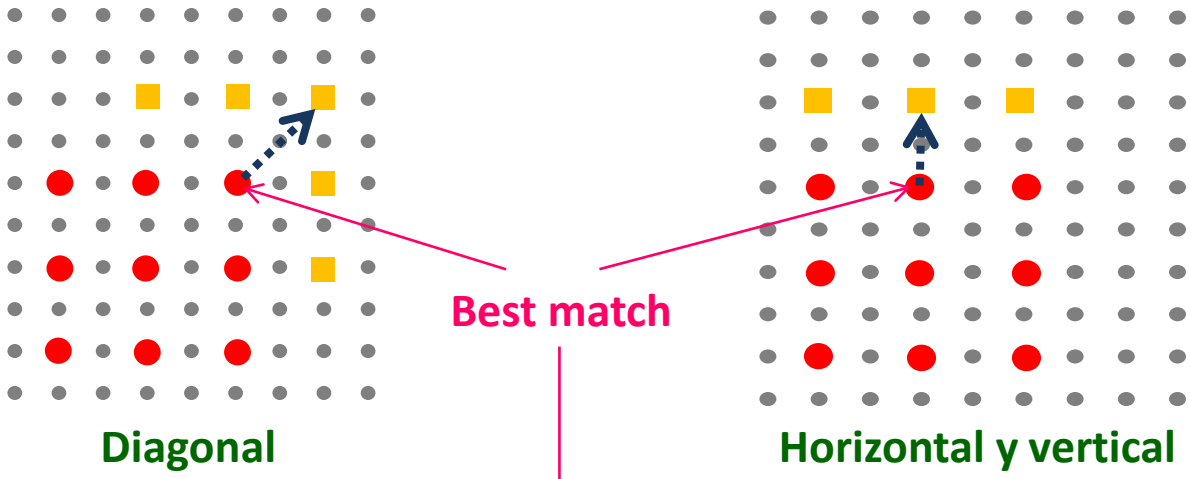




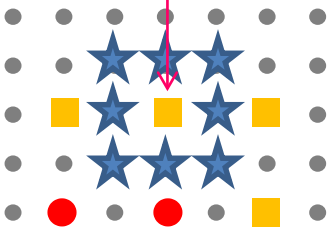
Número total de operaciones *block matching* = 17 a 33

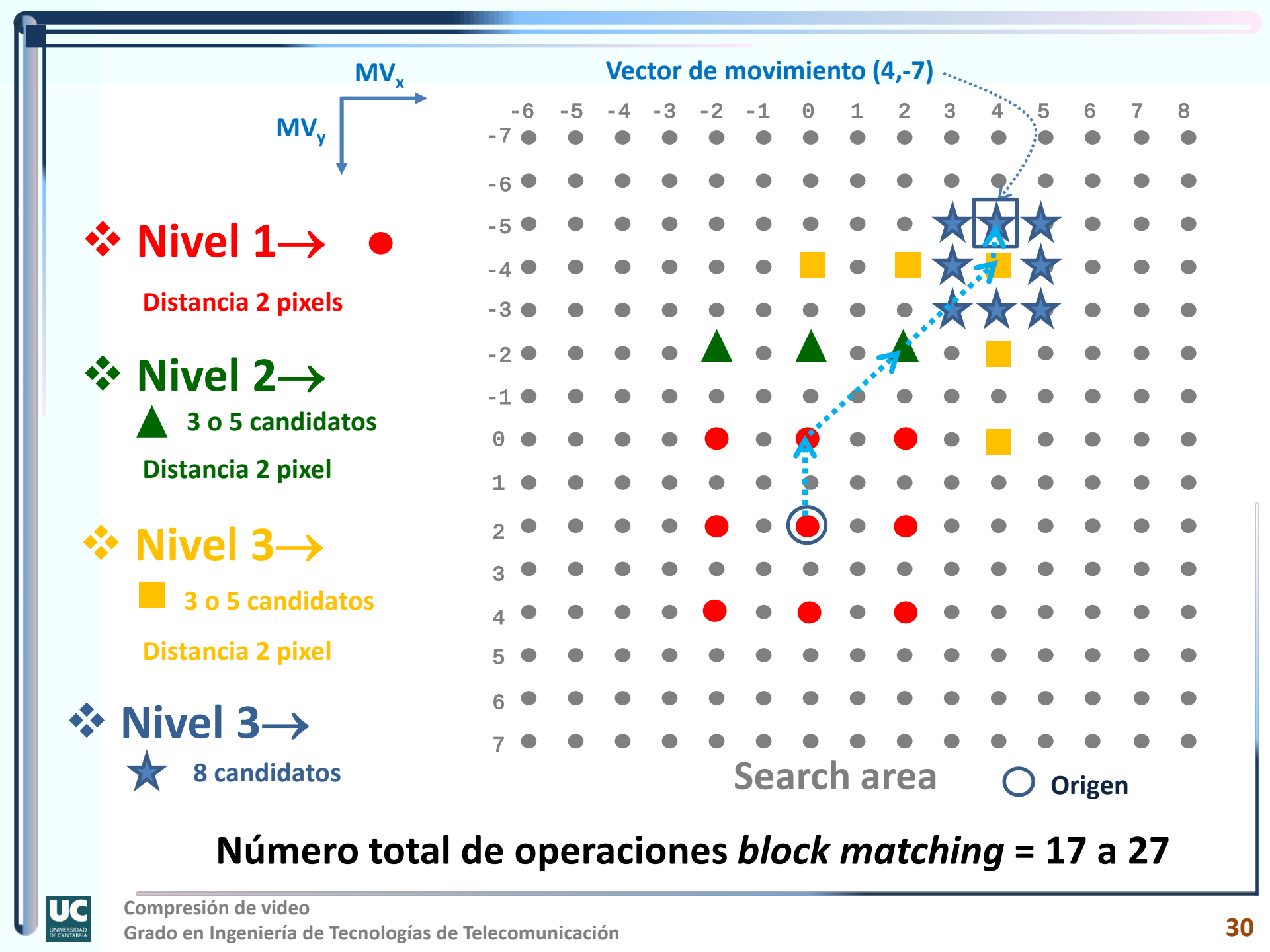
Búsqueda de cuatropasos (*Four Step Search* o 4SS)

- Su complejidad computacional es menor que la del NTSS, manteniendo los niveles de calidad de imagen.
- Es robusto frente a movimientos complejos de cámara como son el zoom o rápido movimiento.
- Movimientos de dos pixels



- Movimiento final de 1 pixel





- **Nivel 1→**
  - Distancia 2 pixels
- **Nivel 2→**
  - 3 o 5 candidatos
  - Distancia 2 pixel
- **Nivel 3→**
  - 3 o 5 candidatos
  - Distancia 2 pixel
- **Nivel 3→**
  - 8 candidatos

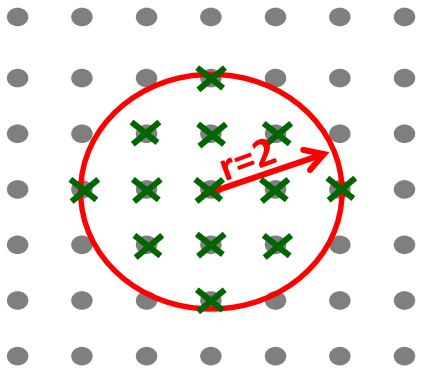
Número total de operaciones *block matching* = 17 a 27

❑ Búsqueda en diamante (*Diamond search or DS*)

- **Experimento hecho con diferentes videos sobre el porcentaje de vectores de movimiento localizados en un radio de diferentes valores de pixels.**

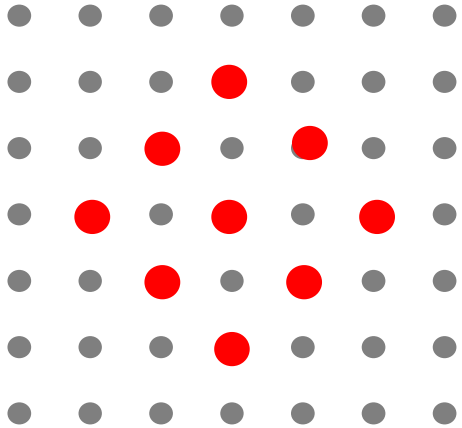
radium (pel)	Tennis	Football	Susie	Salesman
0	0.2622	0.6196	0.0938	0.6562
1	0.3751	0.7297	0.3592	0.9452
2	0.5276	0.7983	0.5950	0.9609
3	0.7178	0.8641	0.7622	0.9741
4	0.8402	0.9042	0.8225	0.9795
5	0.8930	0.9329	0.8779	0.9853
6	0.9200	0.9483	0.9038	0.9957
7	0.9599	0.9658	0.9365	0.9975

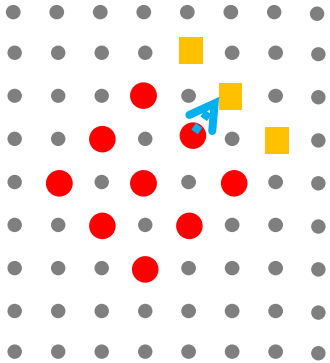
- **Esta table demuestra que el 52% (Tennis) y 96% (Salesman) de los vectores de movimiento están alrededor de 2 pixels. El DS se basa en esta característica.**



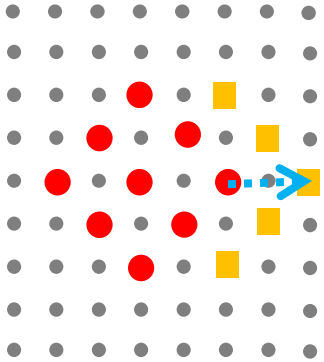
➢ Tipos de desplazamiento:

- **LDSP *o Large Diamond Search Pattern*:**



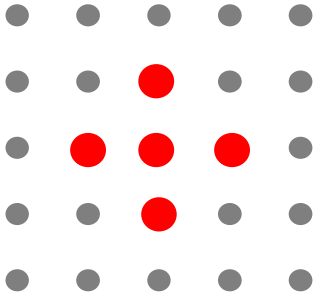


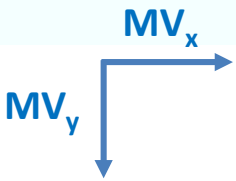
Diagonal



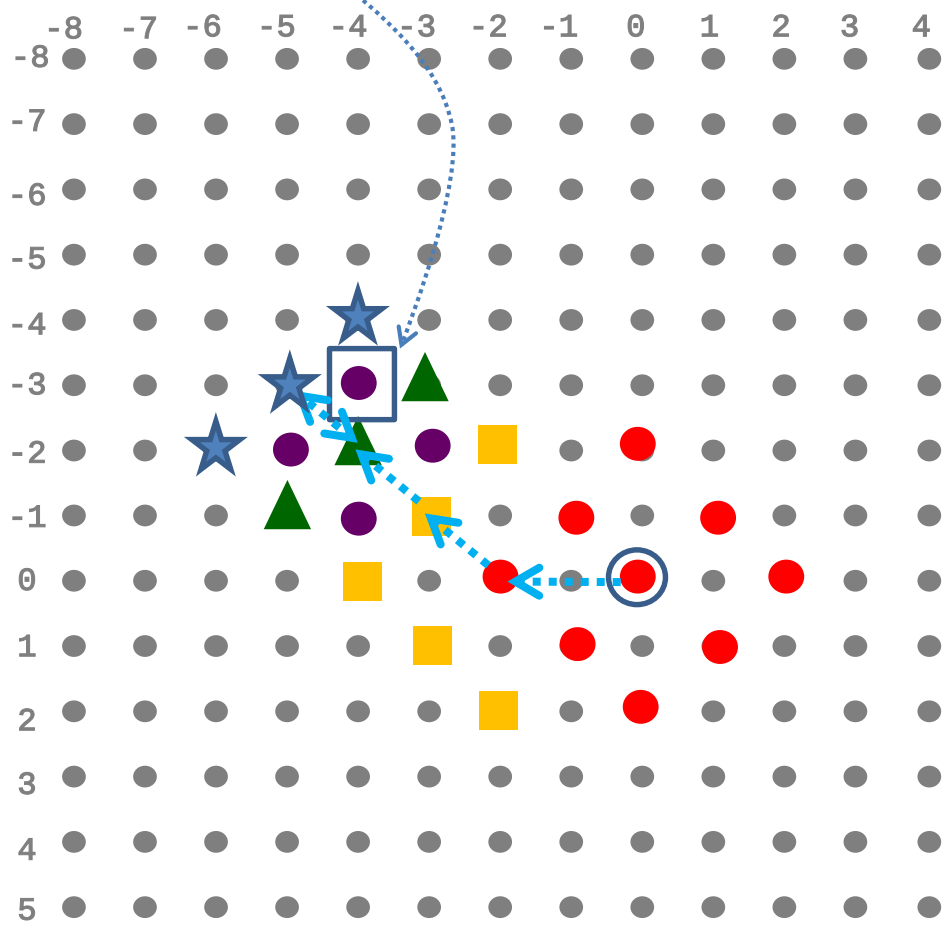
Horizontal y vertical

- **SPSP o *Small Diamond Search Pattern*:**





Vector de movimiento (-4,-3)



- LDSP 1→ ●
- LDSP 2→ ■
- LDSP 3→ ▲
- LDSP 4→ ★
- SDSP → ●

No hay límite en el número de pasos

Conclusiones

- Resultados de la simulación muestran que el algoritmo de búsqueda completa le ofrece la mejor calidad (PSNR más alto) a costa de una alta complejidad computacional. Estimación de movimiento constituye alrededor del 60% de la carga total de cálculo en los codificadores. Por ello, el algoritmo de búsqueda completa no es la técnica más factible para ser usado en la estimación de movimiento para aplicaciones en tiempo real.

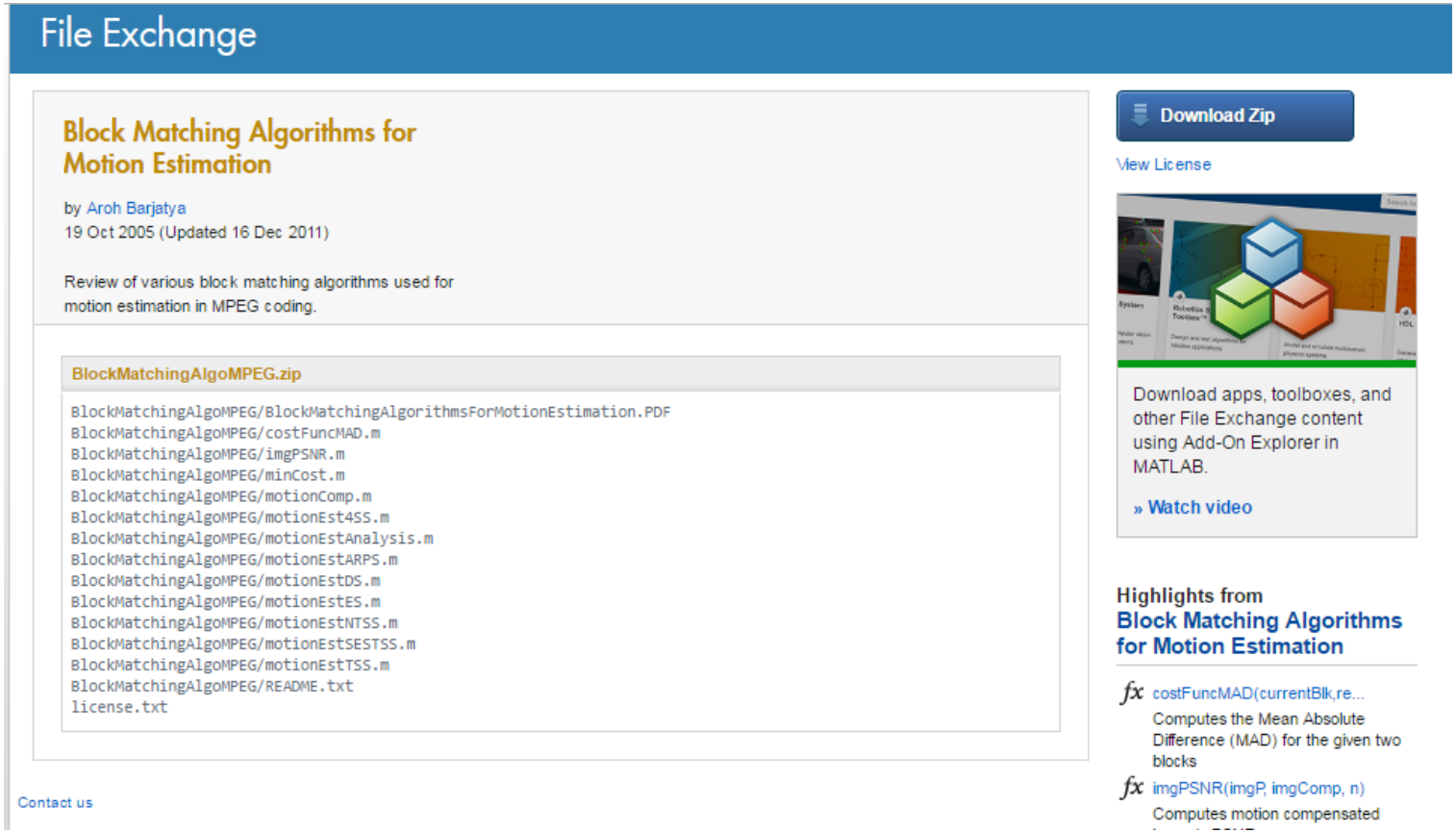
- Algoritmo de TSS ofrece un buen rendimiento, pero no siempre es capaz de detectar pequeños movimientos.

- NTSS es bueno para movimientos lentos, pero su complejidad computacional es mayor que la de TSS para el caso de movimientos rápidos.

- 4SS tiene un pequeño primer paso y es bueno para detectar movimientos lentos. Su complejidad computacional es comparable a la de TSS y menor que la de NTSS.
- DS es un solución intermedia que se comporta bien tanto computacionalmente como en términos de calidad.

Caso práctico : Fichero *AnalisisMotionComp.m*

https://es.mathworks.com/matlabcentral/fileexchange/8761-block-matching-algorithms-for-motion-estimation/all_files



File Exchange

Block Matching Algorithms for Motion Estimation

by Aroh Barjatya
19 Oct 2005 (Updated 16 Dec 2011)

Review of various block matching algorithms used for motion estimation in MPEG coding.

BlockMatchingAlgoMPEG.zip

```
BlockMatchingAlgoMPEG/BlockMatchingAlgorithmsForMotionEstimation.PDF
BlockMatchingAlgoMPEG/costFuncMAD.m
BlockMatchingAlgoMPEG/imgPSNR.m
BlockMatchingAlgoMPEG/minCost.m
BlockMatchingAlgoMPEG/motionComp.m
BlockMatchingAlgoMPEG/motionEst4SS.m
BlockMatchingAlgoMPEG/motionEstAnalysis.m
BlockMatchingAlgoMPEG/motionEstARPS.m
BlockMatchingAlgoMPEG/motionEstDS.m
BlockMatchingAlgoMPEG/motionEstES.m
BlockMatchingAlgoMPEG/motionEstNTSS.m
BlockMatchingAlgoMPEG/motionEstSESTSS.m
BlockMatchingAlgoMPEG/motionEstTSS.m
BlockMatchingAlgoMPEG/README.txt
license.txt
```

Contact us

Download Zip

View License

Download apps, toolboxes, and other File Exchange content using Add-On Explorer in MATLAB.

» Watch video

Highlights from
Block Matching Algorithms for Motion Estimation

fx costFuncMAD(currentBlk,re...
Computes the Mean Absolute Difference (MAD) for the given two blocks

fx imgPSNR(imgP, imgComp, n)
Computes motion compensated