



# TV: TeleVisión – Plan 2010

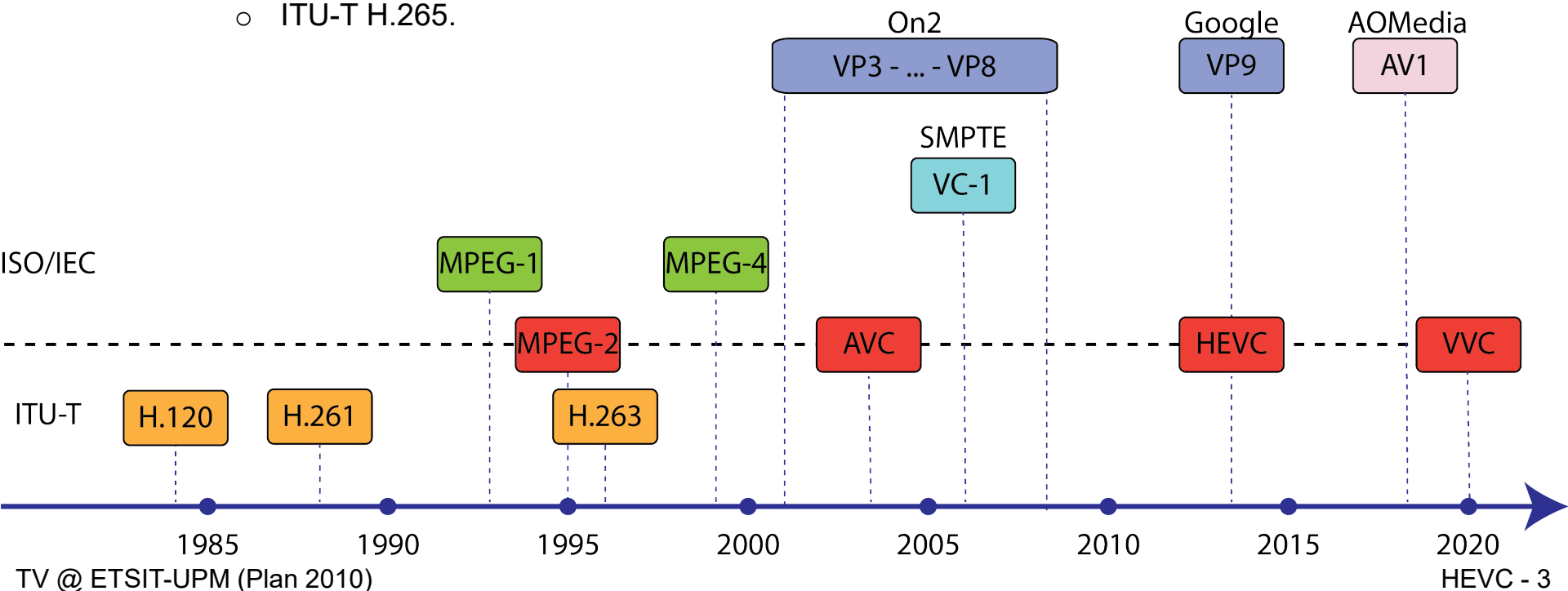
## HEVC

# Contenido

1. Introducción
  1. Objetivos
  2. Principales características
2. Arquitectura
3. Estructura de datos
4. Predicción
  1. Intra-cuadro
  2. Inter-cuadro
  3. Ejemplo: *Elecard HEVC analyzer*
5. Transformación
6. Cuantificación
7. Filtro de reconstrucción (*deblocking*)
8. *Sample Adaptive Offset (SAO)*
9. Codificación

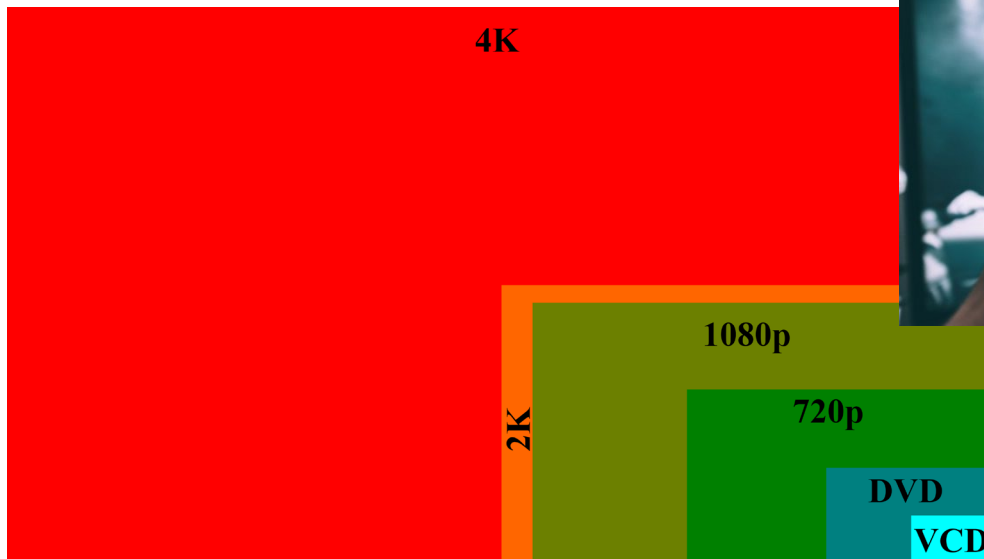
# 1 – Introducción (1)

- HEVC (*High Efficiency Video Coding*):
  - Nuevo estándar con la capacidad de comprimir mucho más que cualquiera de los anteriores.
  - Creado por el *Joint Collaborative Team on Video Coding* (JCT-VC).
  - En desarrollo desde enero de 2010 → Estándar desde abril de 2013.
  - Denominaciones:
    - ISO/IEC 23008-2 (MPEG-H Parte 2).
    - ITU-T H.265.



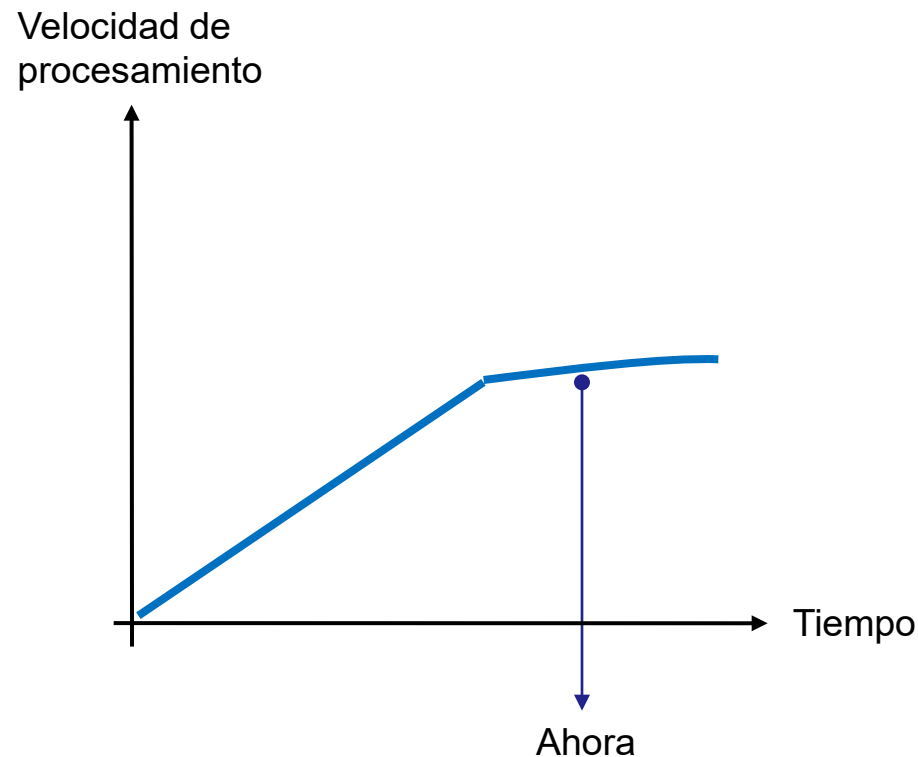
# 1 – Introducción (2)

- ¿Por qué es necesario?
  - Nuevas necesidades comerciales y tecnológicas:
    - Resoluciones cada vez mayores → Ultra HD (4K) y 8K.
    - Demanda de mayor calidad de imagen (más bits y más imágenes por segundo).
    - Nuevos segmentos del mercado (por ejemplo, *healthcare*, o videojuegos).



# 1 – Introducción (3)

- ¿Por qué es necesario?
  - Las mejoras computacionales de los procesadores no siguen el curso esperado.
  - Es necesario paralelizar (procesadores con múltiples núcleos) → Se debe crear un estándar con algoritmos orientados a la paralelización.

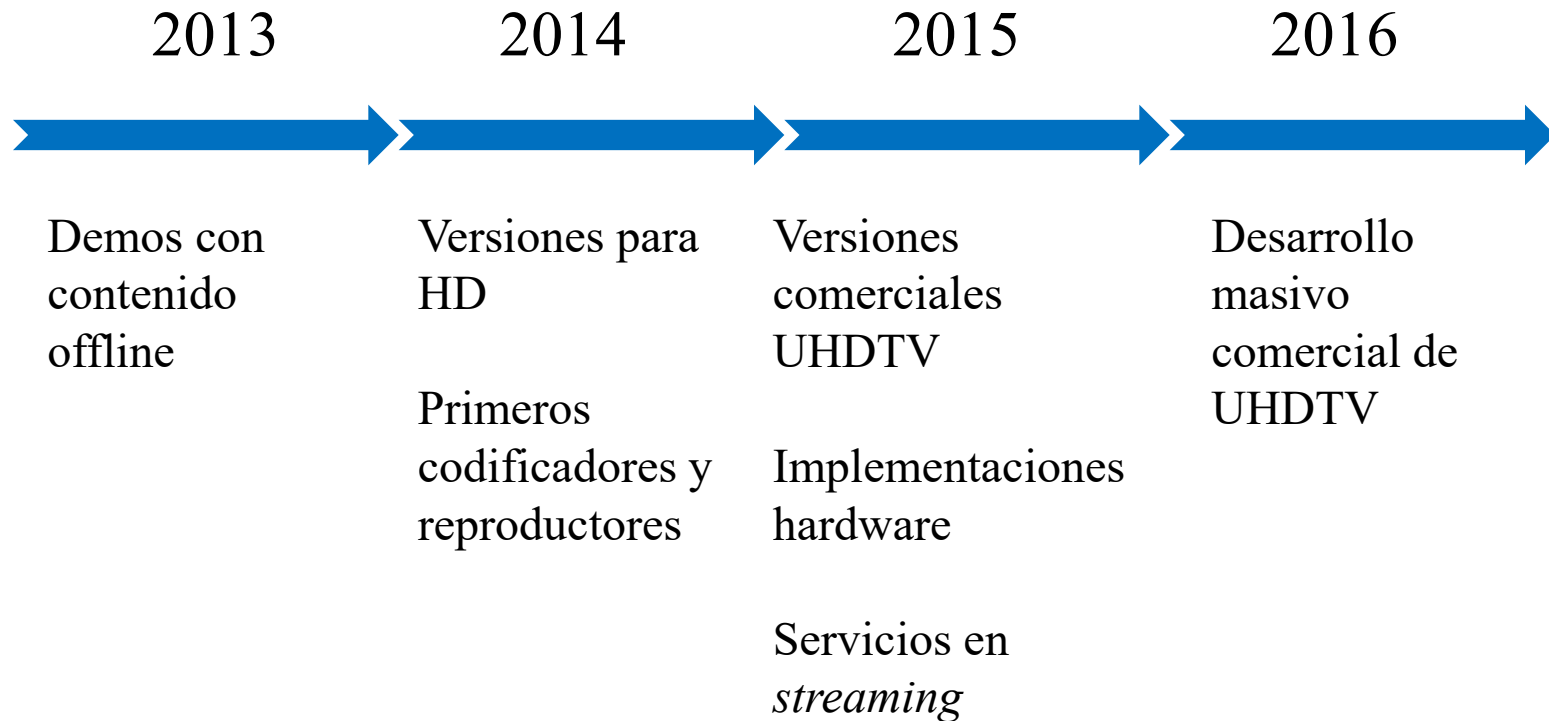


# 1 – Introducción (4)

- ¿Por qué es necesario?
  - Las nuevas tecnologías de transmisión (por ejemplo, DVB o LTE) requieren la transmisión de señales audiovisuales de muy distinto tipo:
    - Escalabilidad → Resolución, fps, tasa binaria...
    - También se ha de tener en cuenta el limitado uso del espectro → Se necesita comprimir.
  - La demanda de vídeo transmitido crece muy rápido:
    - Se estimó que en 2018 se transmitirían aproximadamente un millón de minutos de vídeo por segundo.

# 1 – Introducción (5)

- Industrialización:



# 1.1 – Objetivos (1)

- Principales:
  - Ser una referencia para la siguiente generación de dispositivos de captura y visualización de vídeo.
  - Mejorar la eficiencia de codificación → Menor tasa binaria:
    - ¿Cuánto? → Como mínimo a **la mitad que AVC**.
    - Problema → **Mayor complejidad computacional** (50% - 400% más).
  - Mejorar la resolución:
    - *Ultra High Definition* TV (UHD-TV) → 8K (8192 x 4320 píxeles).
  - Mejorar los métodos de procesamiento paralelo.

## 1.1 – Objetivos (2)

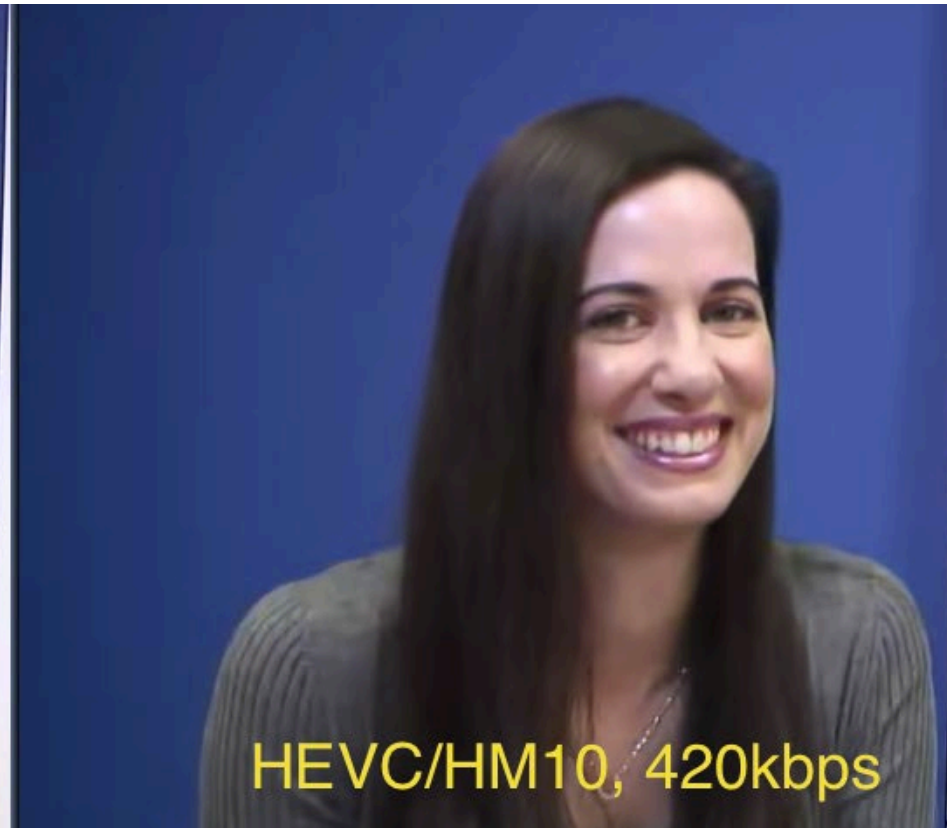
- Otros:
  - Permitir SVC (*Scalable Video Coding*).
  - Mejorar la calidad de la imagen:
    - Permitiendo cuantificar con 12 bits.
    - Con esquemas de crominancia 4:2:2 y 4:4:4.

## 1.2 – Principales características (1)

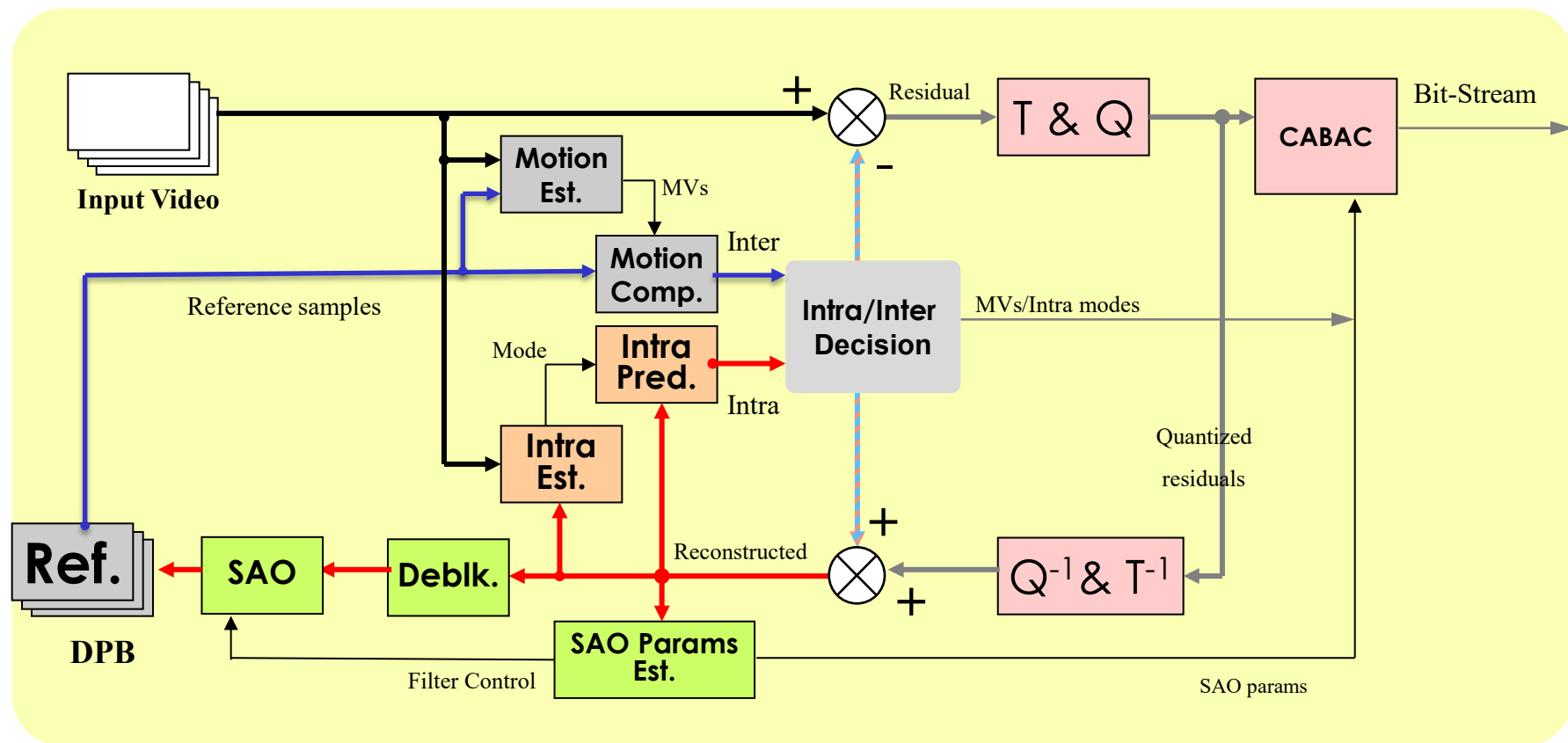
- Sigue la misma estructura que los estándares previos:
  - Codificación asimétrica → Los descodificadores son mucho más sencillos que los codificadores.
  - Codificación híbrida predictivo-transformacional.
- Propone numerosas mejoras:
  - Particionado más flexible (ya no se usa la estructura típica de macrobloques y bloques).
  - Mayor flexibilidad en los modos de predicción intra.
  - Mayor número de posibles tamaños de bloques sobre los que aplicar la transformación.
  - Interpolación más sofisticada.
  - Filtro de *deblocking* más sofisticado.
  - Algoritmos de predicción más sofisticados (partiendo de una partición refinada) y señalización de modos más eficiente.
  - Características para adaptar los módulos al procesamiento en paralelo.

## 1.2 – Principales características (2)

- Comparación con AVC:

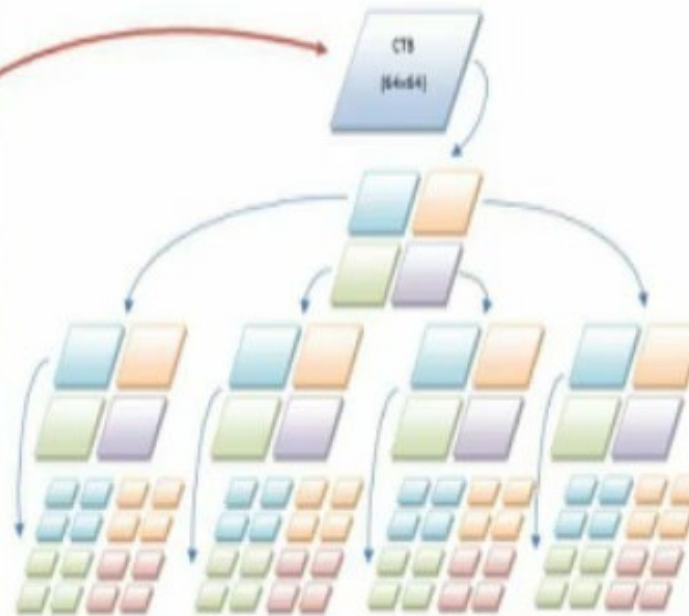
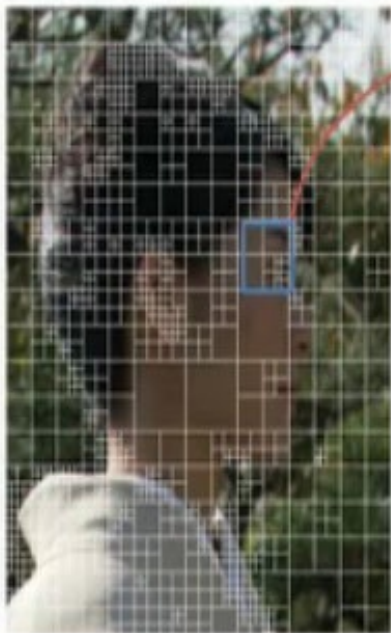


## 2 - Arquitectura



## 3 – Estructura de datos (1)

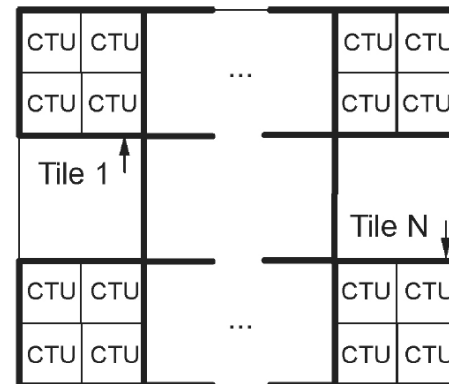
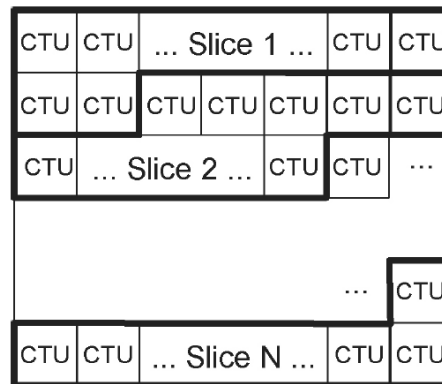
- HEVC permite un particionado muy flexible de los datos.
  - Por las mismas razones que en estándares previos → Propagación de errores, sincronización, etc.
  - Para aprovechar el procesamiento en paralelo.



- Se sigue una estructura jerárquica y adaptativa, cuyo principal elemento es el **Coding Tree Unit (CTU)** → Son equivalentes a los macrobloques en estándares previos.

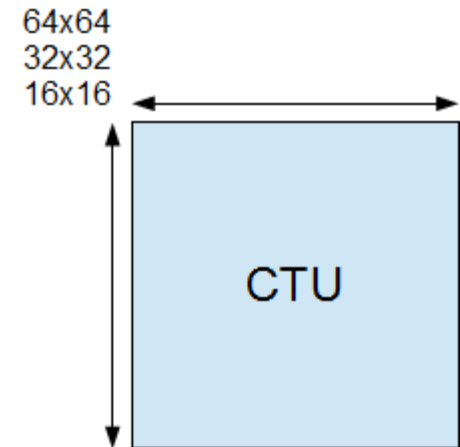
## 3 – Estructura de datos (2)

- **Tiras (*slices*):**
  - Su objetivo principal es la resincronización en caso de pérdida de datos en recepción.
  - Si son independientes entre sí se pueden procesar en paralelo.
  - Pueden contener un número variable de CTUs (el número dependerá de la tasa binaria objetivo).
- **Azulejos o baldosas (*tiles*):**
  - No existen en estándares previos.
  - Son independientes → Procesamiento en paralelo.
  - Número constante de CTUs → Para optimizar el procesamiento en paralelo.



## 3 – Estructura de datos (3)

- CTU (*Coding Tree Unit*):
  - Unidad básica de procesamiento → Equivalente a los macrobloques en estándares previos.
  - Son cuadradas y de 3 posibles tamaños → 16x16, 32x32 o 64x64 píxeles:
    - Permiten dividir mejor las imágenes (mucha flexibilidad).
    - Los macrobloques (16x16) se quedaban pequeños para cubrir imágenes muy grandes (HD o UHD).
  - Son la raíz de una estructura en árbol que da lugar a:
    - Unidades de codificación (*coding units*, CUs).
    - Unidades de predicción (*prediction units*, PUs).
    - Unidades de transformación (*transform units*, TUs).



## 3 – Estructura de datos (4)

- CTU (*Coding Tree Unit*):

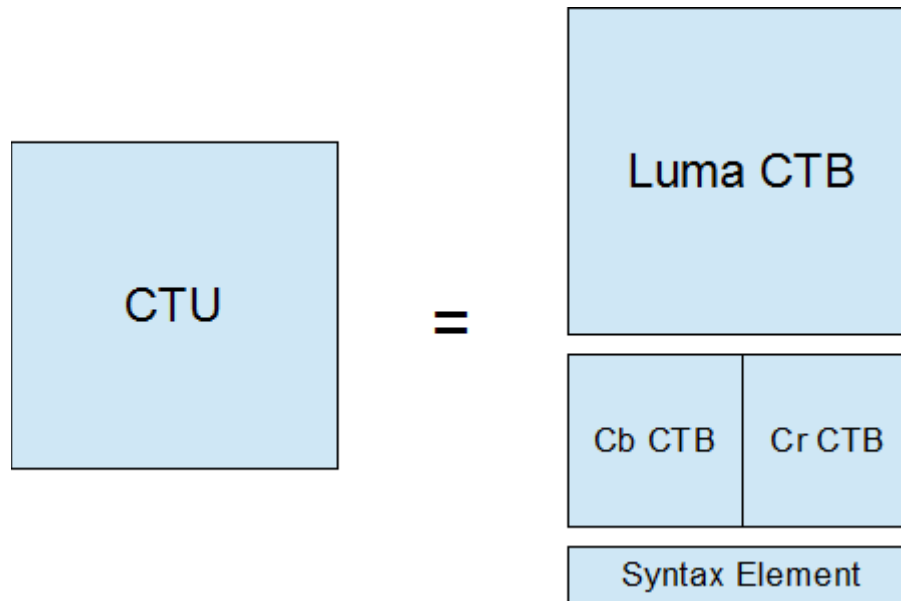


CTU

*Slice*

## 3 – Estructura de datos (5)

- CTB (*Coding Tree Block*):
  - Una CTU puede estar formada por:
    - Una única CTB → Sólo luminancia.
    - Tres CTBs → Luminancia y crominancias.



- Los CTBs de luminancia tienen el mismo tamaño que los correspondientes CTUs.

- En función del esquema de muestreo de crominancias, el tamaño de los CTBs será:

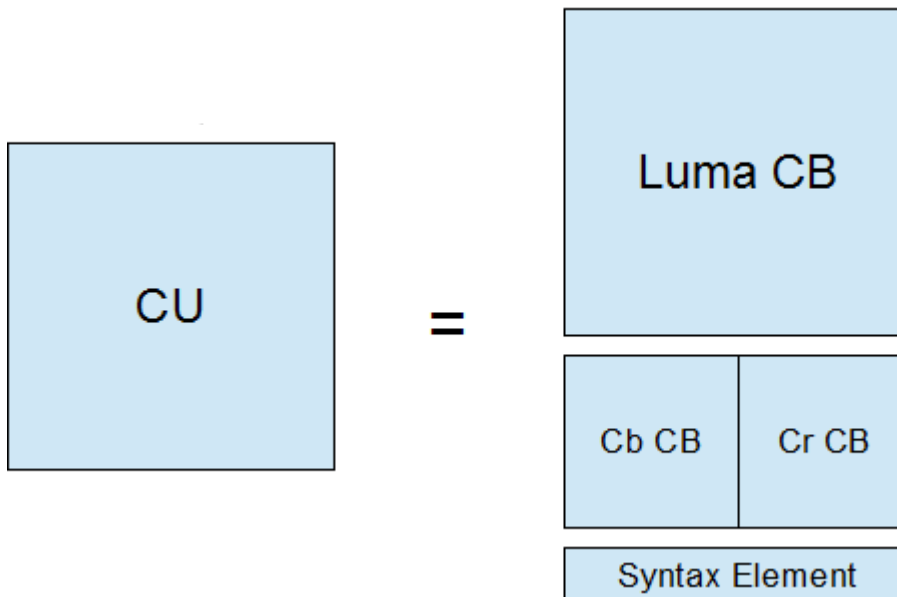
- 4:4:4 → Igual que el de la CTU.

- 4:2:2 → La mitad de columnas que la CTU

- 4:2:0 → Un cuarto de la CTU.

## 3 – Estructura de datos (6)

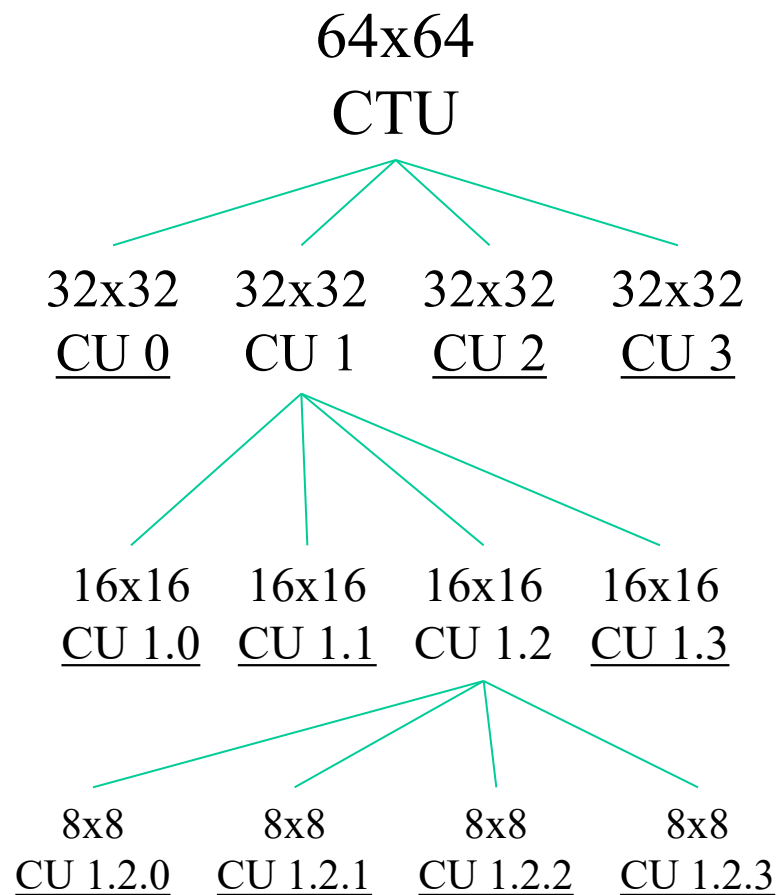
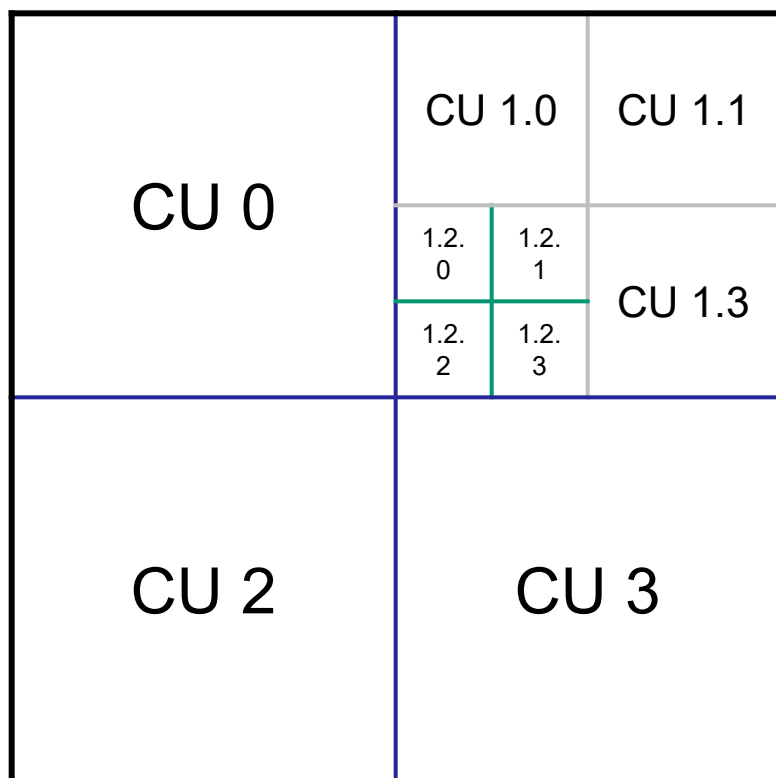
- CU (*Coding Unit*):
  - Determina si la codificación es intra o inter.
  - Su tamaño máximo es el de la CTU.
  - Su tamaño mínimo es de 8x8 píxeles.
  - Se puede dividir una única vez para formar PUs.



- Al igual que antes, una CU puede dividirse en:
  - Un CB.
  - Tres CBs.

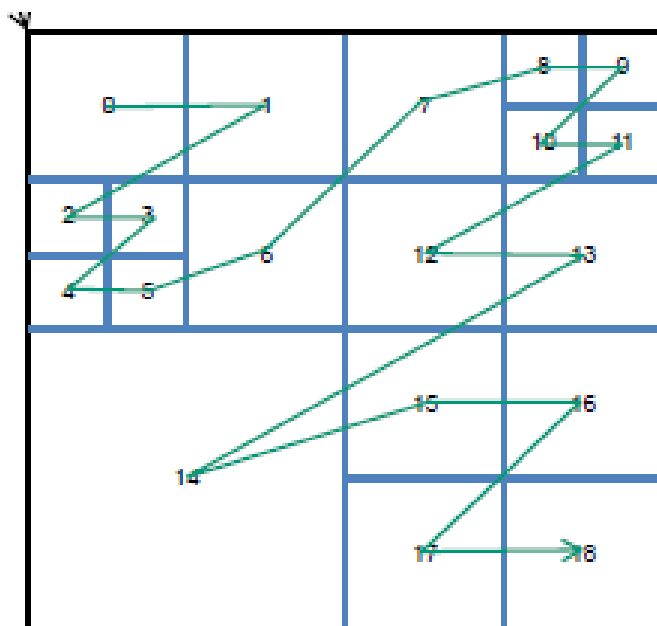
## 3 – Estructura de datos (7)

- Ejemplo de división de una CTU en CUs:

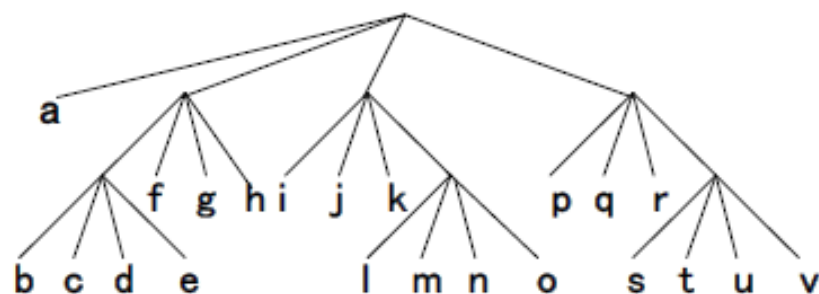


# 3 – Estructura de datos (8)

- Escaneado de las CUs:



a		b	c	f	
		d	e		
		g		h	
i	j	p	q		
k	l	m	r	s	t
	n	o		u	v



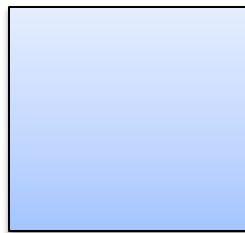
## 3 – Estructura de datos (9)

- PU (*Prediction Unit*):
  - Se decide el modo de predicción a aplicar.
  - Su tamaño máximo es el de la CU.
  - Su tamaño mínimo es de 4x4 píxeles.
  - Al igual que las estructuras anteriores, consta de 1 o 3 PBs (*Prediction Blocks*).
- En el caso de predicción tipo intra:
  - Sólo se permiten las divisiones simétricas.
- En el caso de predicción tipo inter:
  - Las divisiones pueden ser simétricas o asimétricas.

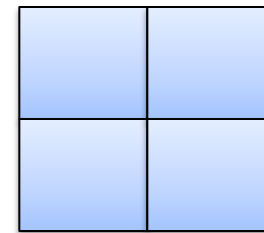
## 3 – Estructura de datos (10)

- CU - intra:
  - Si la CU no ha llegado a su tamaño mínimo → La PU será idéntica a la CU.
  - Si la CU ha llegado a 8x8 píxeles → Se podrá decidir entre:
    - No dividirla.
    - Dividirla en 4 PUs de idéntico tamaño.

CU de tamaño:  $2N \times 2N$



1 PU ( $2N \times 2N$ ) por CU

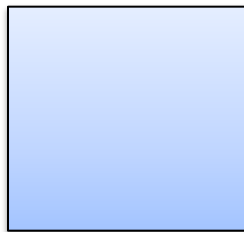


4 PUs ( $N \times N$ ) por CU

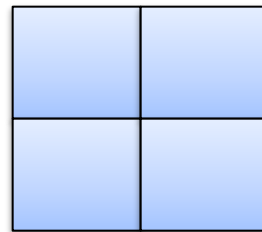
## 3 – Estructura de datos (11)

- CU - inter:
  - Puede dar lugar a 1, 2 o 4 PUs.
  - Las divisiones pueden ser simétricas o asimétricas.
  - La división en 4 PUs sólo es posible si la CU es de 8x8 píxeles.
  - Hay 4 posibles divisiones simétricas:

CU de tamaño:  $2N \times 2N$



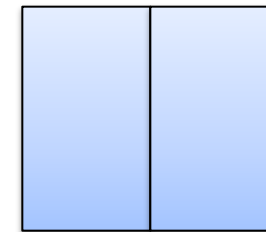
1 PU ( $2N \times 2N$ )



4 PUs ( $N \times N$ )



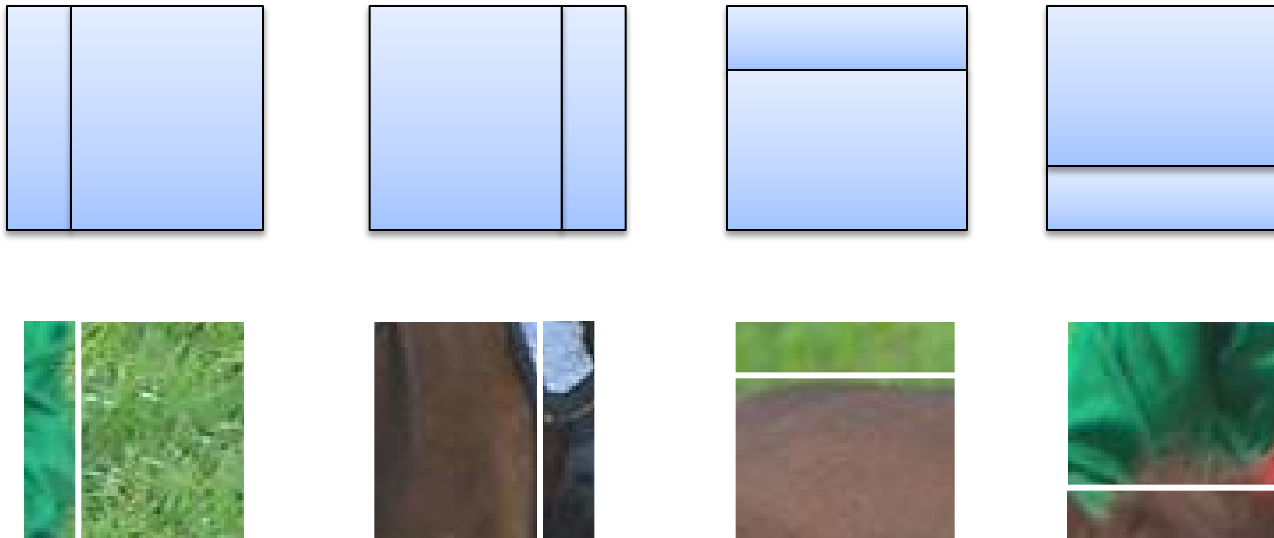
2 PUs ( $2N \times N$ )



2 PUs ( $N \times 2N$ )

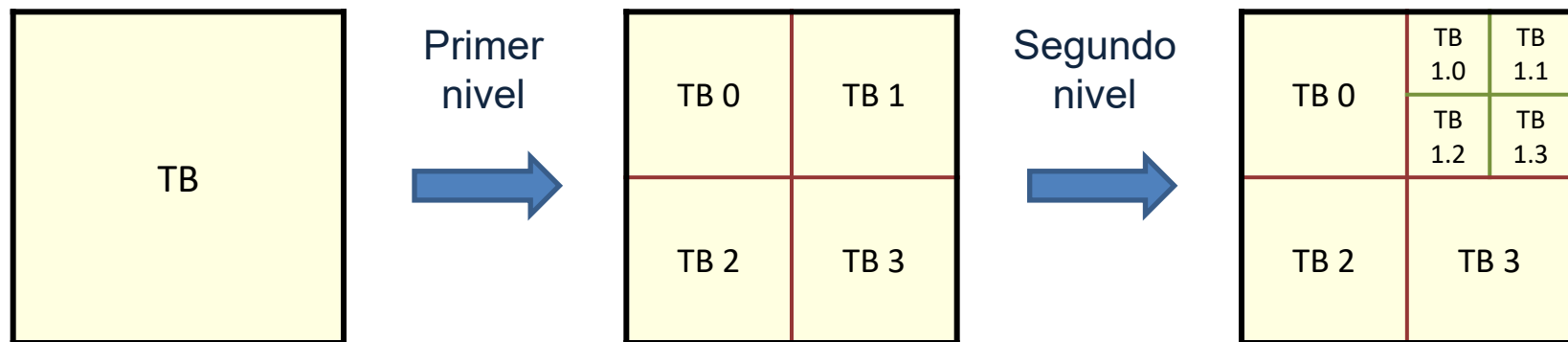
## 3 – Estructura de datos (12)

- CU - inter:
  - Hay 4 posibles divisiones asimétricas:
    - Sólo se permiten cuando la CU es de  $32 \times 32$  o superior.
    - Una PU tendrá un lado de tamaño  $N$  y la otra PU lo tendrá de tamaño  $N/4$ .

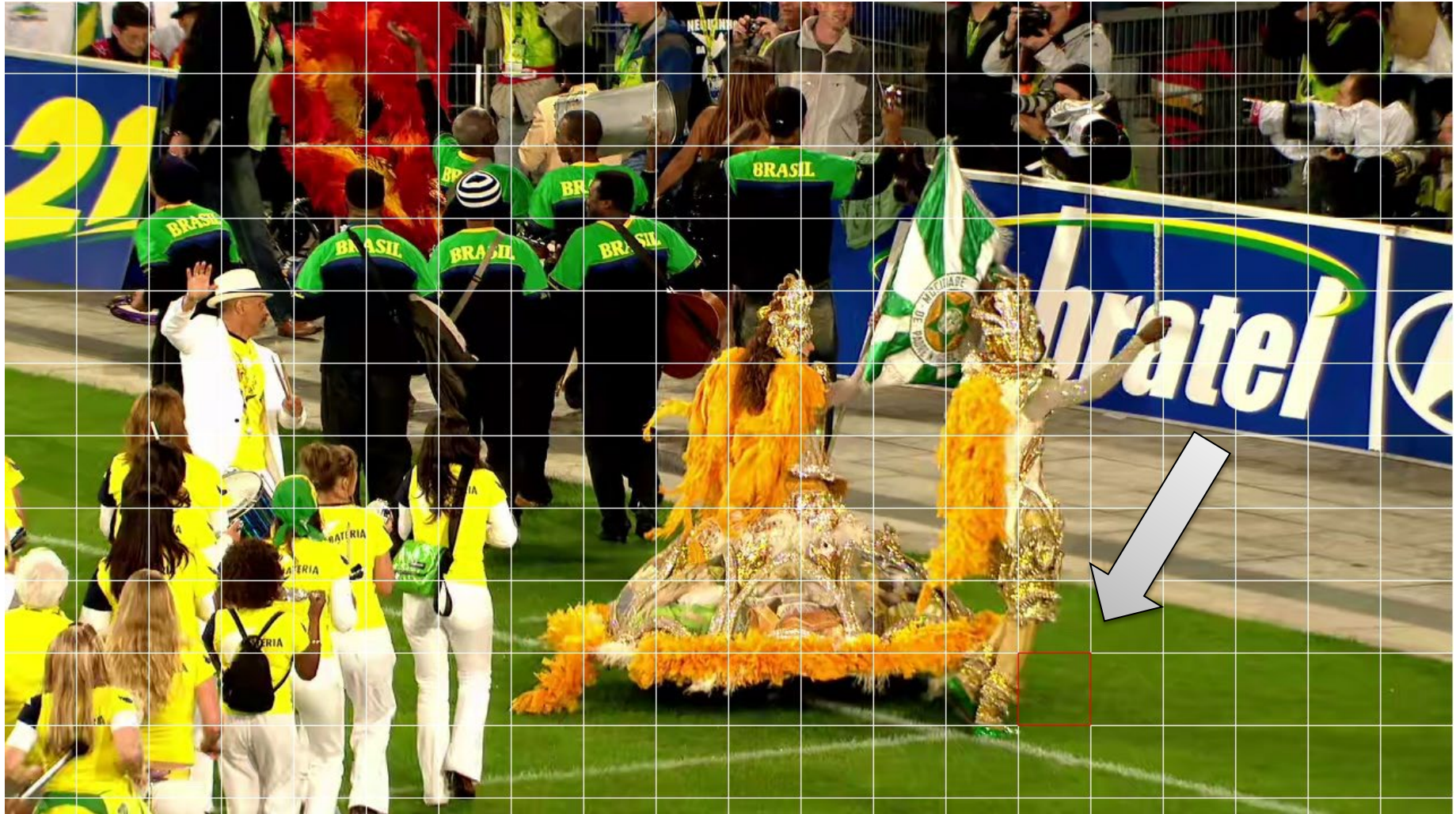


### 3 – Estructura de datos (13)

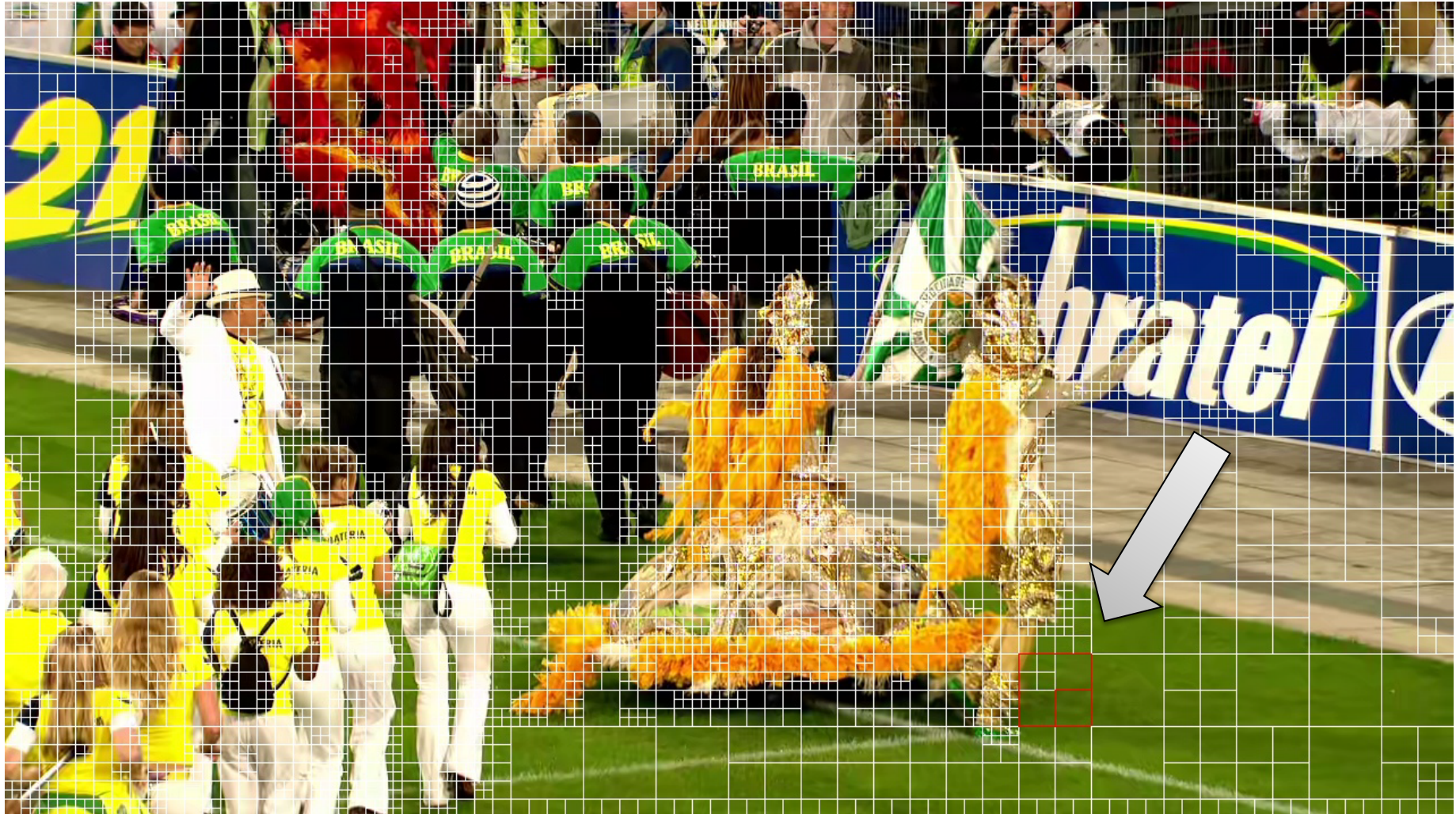
- TU (*Transform Unit*):
  - Es la unidad básica para la transformación y la cuantificación.
  - Su raíz es una CU.
  - Puede ser idéntica a una CU o dividirse en unidades de menor tamaño.
  - Sus tamaños pueden ser: 4x4, 8x8, 16x16 y 32x32.
  - Al igual que las estructuras anteriores, consta de 1 o 3 TBs (*Transform Blocks*).



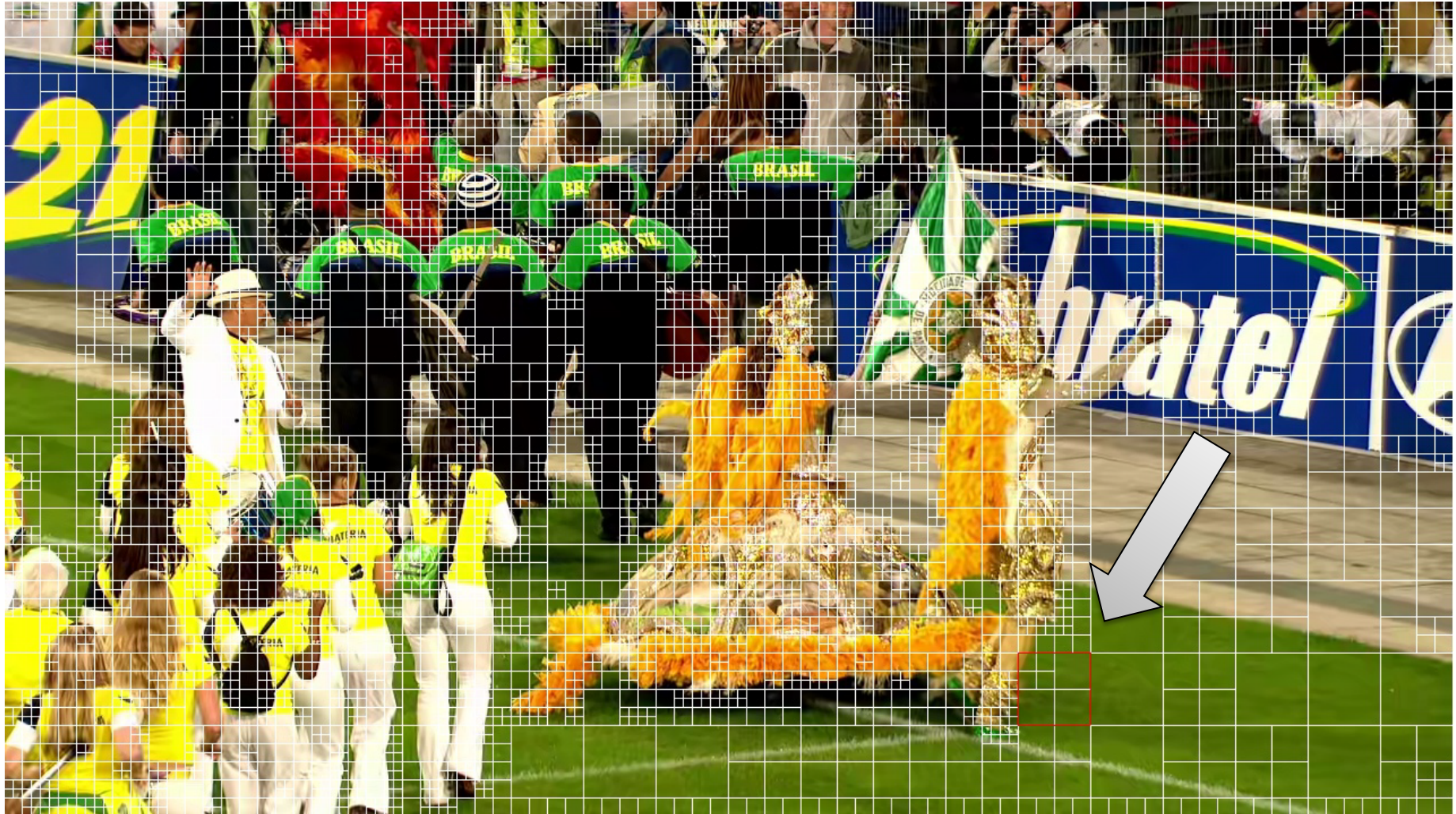
### 3 – Estructura de datos (14)



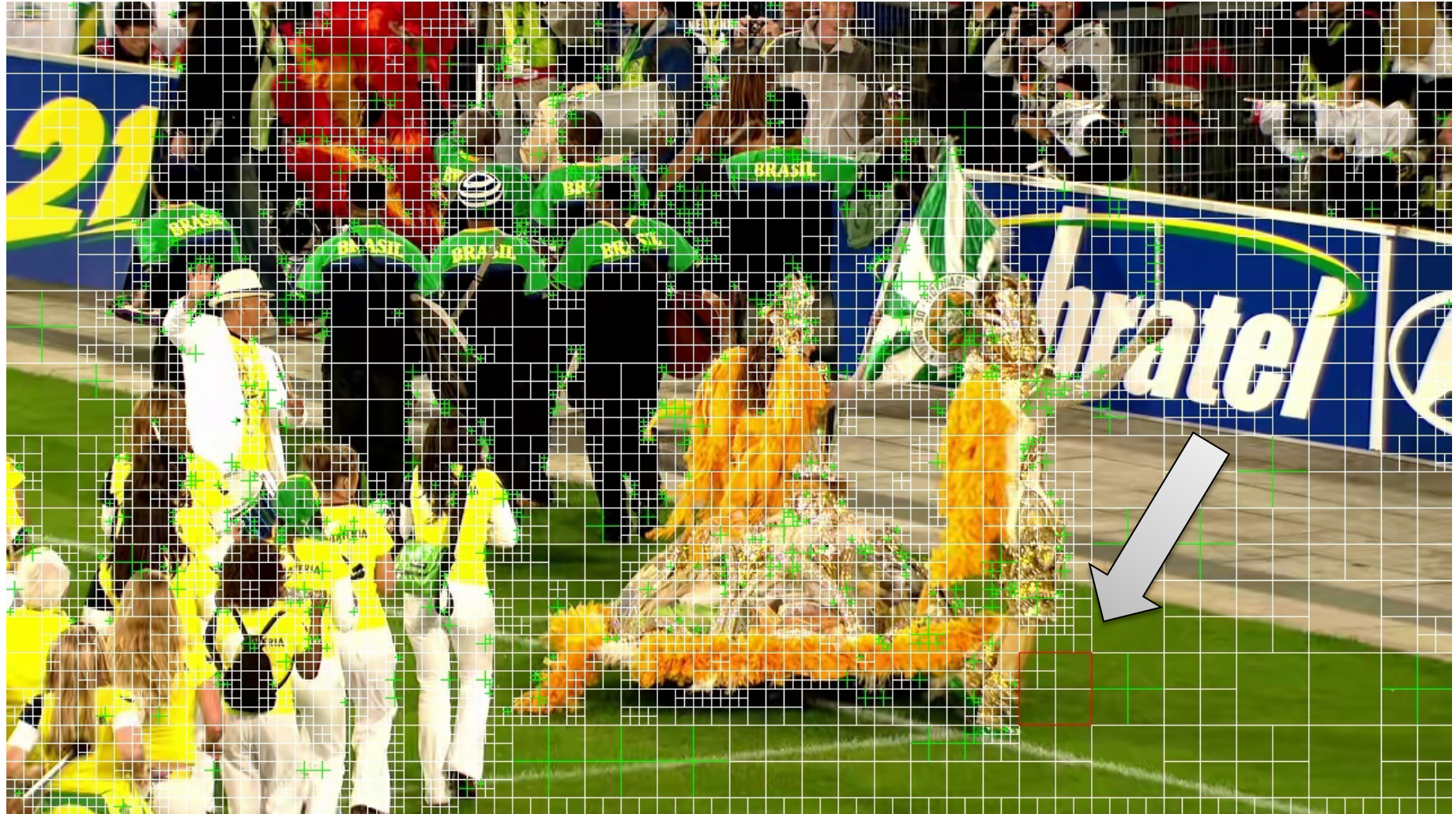
### 3 – Estructura de datos (15)



### 3 – Estructura de datos (16)



### 3 – Estructura de datos (17)

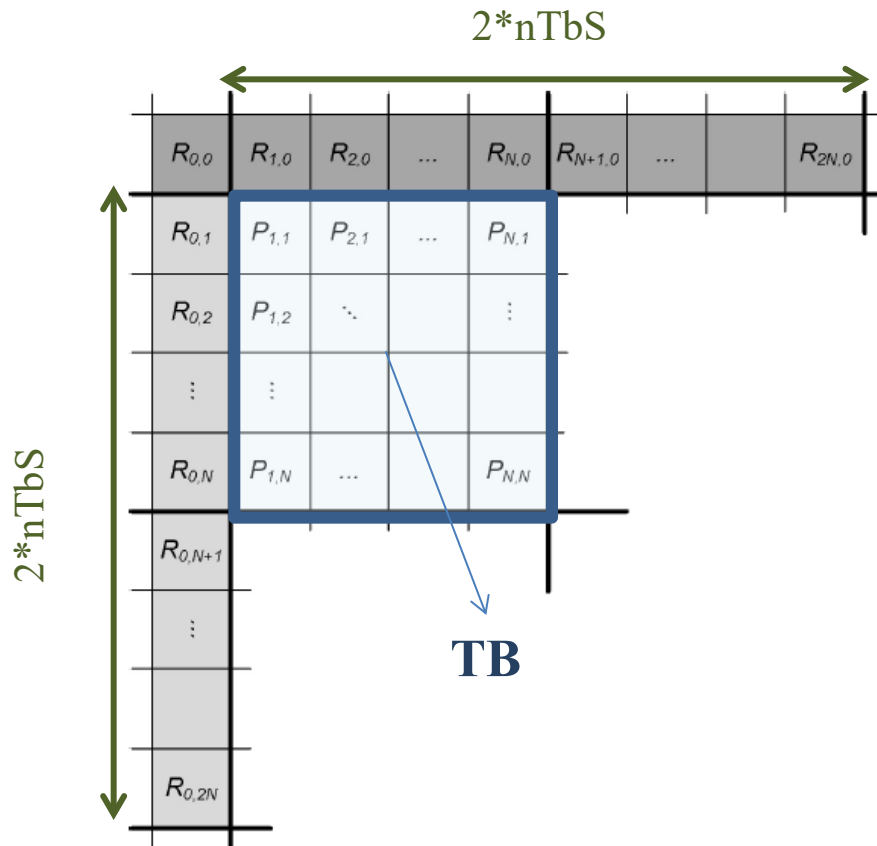




## 4.1 – Predicción intra-cuadro (2)

- Se aplica a CUs completas.
- Aunque el modo de predicción se decide para cada PU, la predicción se lleva a cabo de forma independiente para cada TU ( $TU \leq PU$ ).
- Hay 35 posibles modos de predicción:
  - Modo 0 → *Planar*.
  - Modo 1 → DC.
  - Modos 2 a 34 → Direccionales (en AVC hay sólo 8).
- Se selecciona el modo que minimiza el error entre la TU predicha y la TU original.
- Para la crominancia sólo hay 5 modos.

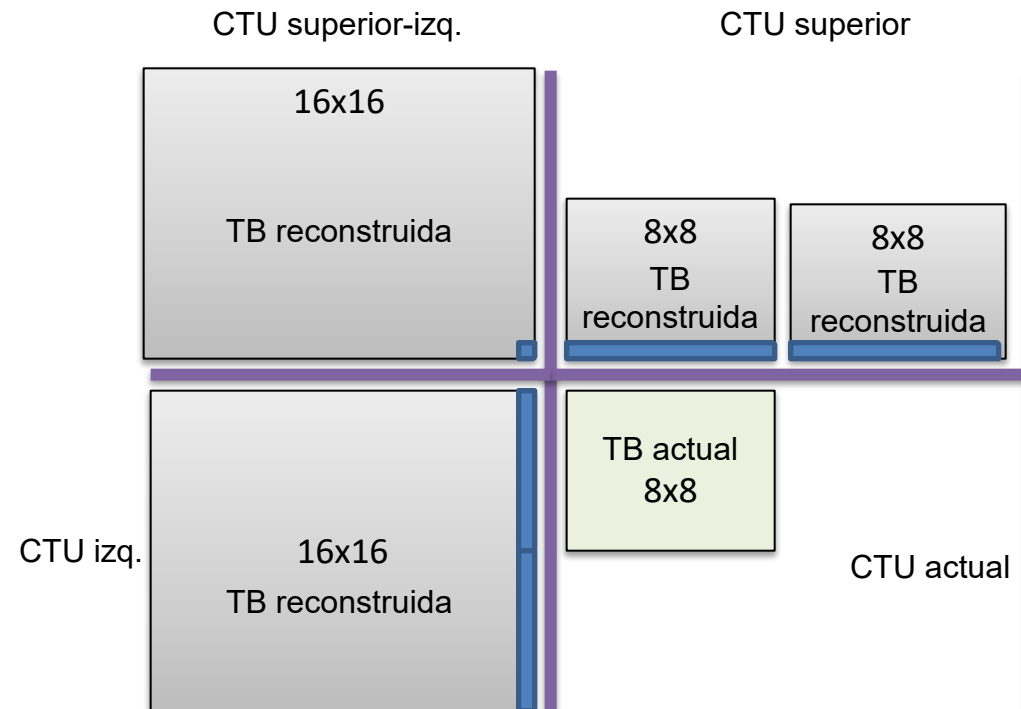
## 4.1 – Predicción intra-cuadro (3)



Cada TU se predice a partir de los píxeles más próximos de las TUs lindantes (descodificadas)  
 → Similar a lo que se hace en AVC.

TB Size (nTbS): 4, 8, 16 or 32

## 4.1 – Predicción intra-cuadro (4)



- A diferencia de AVC, se tienen en cuenta datos a la izquierda y por debajo del TB actual.
- Esto es porque que en HEVC, debido a la gran variedad de tamaños de bloque, la existencia de estos datos es más frecuente.

## 4.1 – Predicción intra-cuadro (5)

- ¿Qué ocurre cuando no existen muestras de referencia?:
  - Se extrapolan los datos.
  - Esto puede suceder en los bordes de la imagen, en los bordes de una tira (slice) o cuando los datos de referencia todavía no han sido codificados (*z-order*).

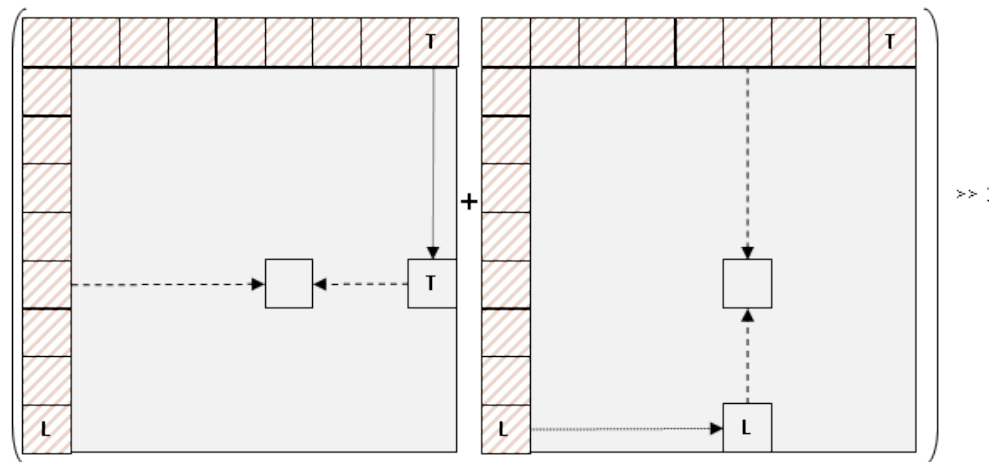
Caso	Sup.	Sup.- der.	Izq.	Izq.- abajo
1	X	X	X	X
2	X	X		
3		X		
4			X	X
5				X
6		X		X



## 4.1 – Predicción intra-cuadro (6)

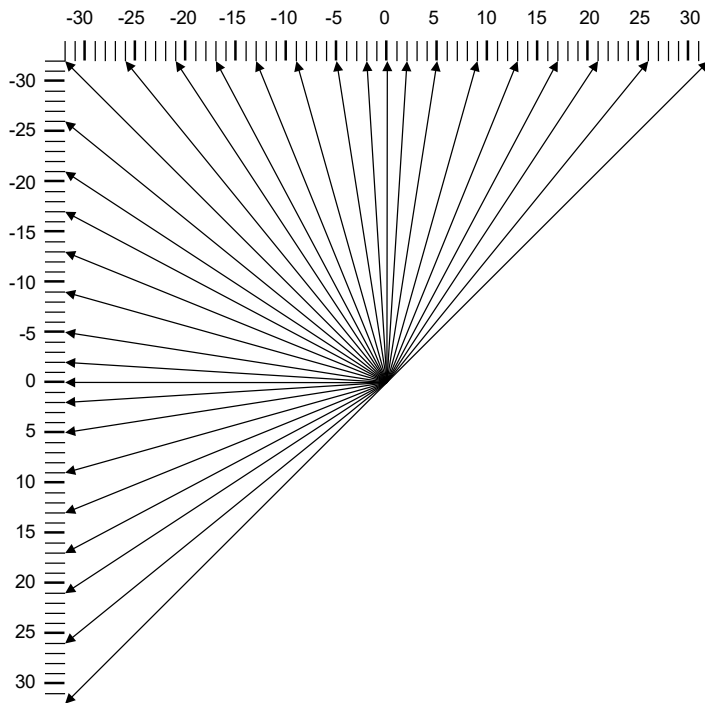
- Modo 0 (*planar*):
  - Es útil en regiones con variaciones suaves.
  - Consiste en una predicción bilineal → Resultado de la media entre dos interpolaciones lineales.

$$\begin{aligned}
 P_{x,y}^V &= (N - y) \cdot R_{x,0} + y \cdot R_{0,N+1} \\
 P_{x,y}^H &= (N - x) \cdot R_{0,y} + x \cdot R_{N+1,0} \\
 P_{x,y} &= (P_{x,y}^V + P_{x,y}^H + N) \gg (\log_2(N) + 1)
 \end{aligned}$$



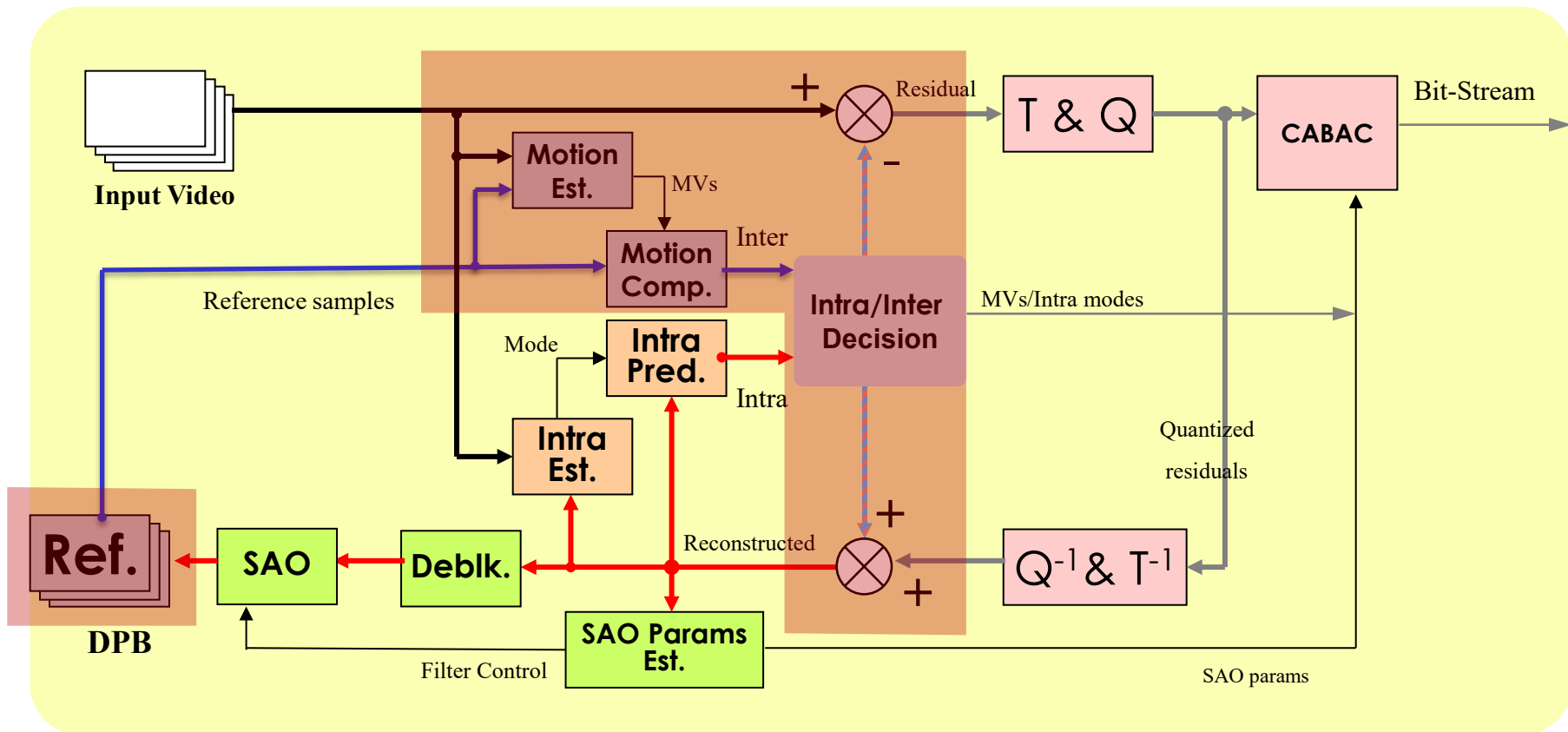
## 4.1 – Predicción intra-cuadro (7)

- Modos 2-34 (direccionales):
  - El descodificador debe dar soporte a 132 posibles combinaciones:
    - 4 posibles tamaños de bloque: 4x4, 8x8, 16x16 y 32x32.
    - 33 modos (direcciones).



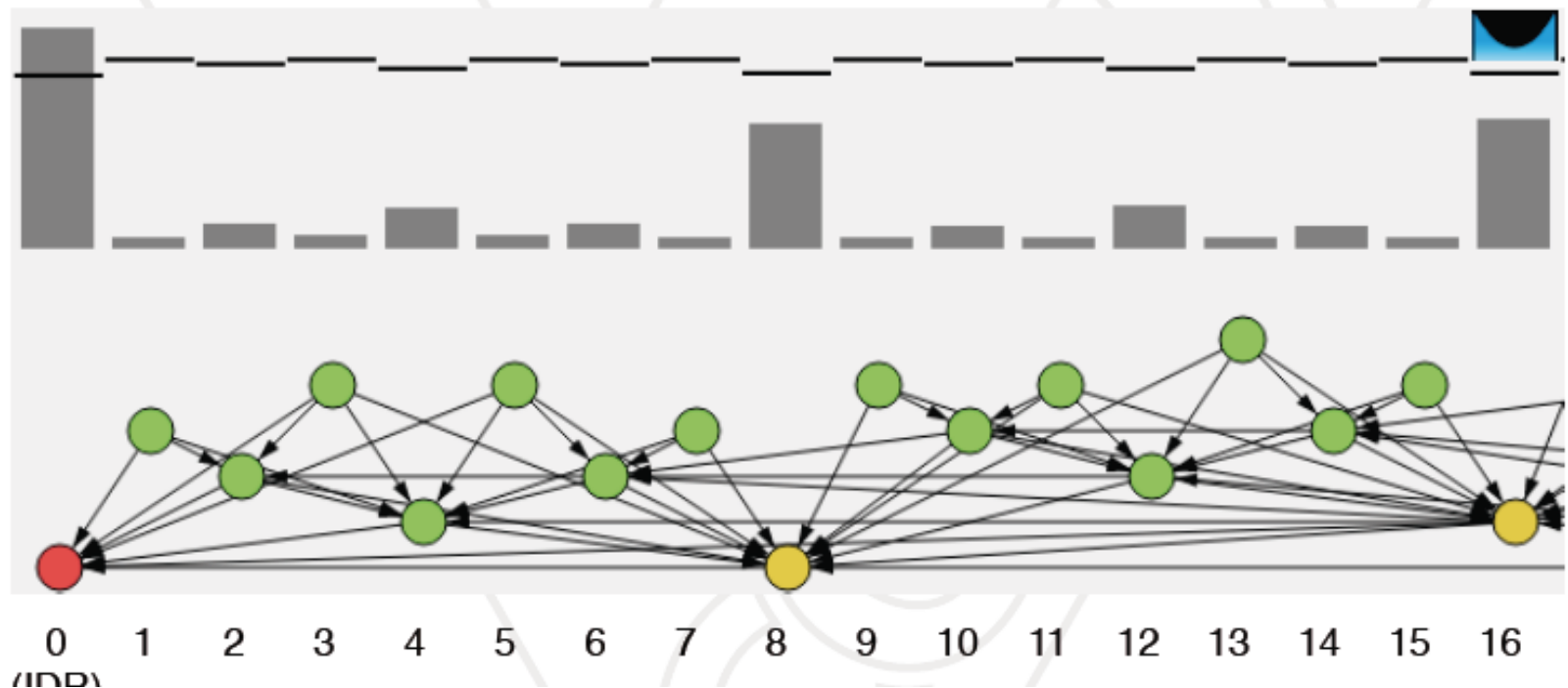
- Las direcciones consideradas están adaptadas a las características de la visión humana:
  - Los patrones más típicos y que mejor distinguimos son los horizontales y verticales.
  - Menor densidad de modos según nos alejamos de estos patrones.

# 4.1 – Predicción inter-cuadro (1)



## 4.2 – Predicción inter-cuadro (2)

- La primera imagen se codifica en modo intra.
- Las siguientes pueden codificarse en modo intra y/o inter:
  - Tomando como referencia una, dos o más imágenes previamente codificadas.
- Se aplica sobre cada PU.



## 4.2 – Predicción inter-cuadro (3)

- Principales diferencias con estándares previos:
  - Estimación de movimiento con precisión de hasta:
    - 1/4 de píxel para la luminancia.
    - 1/8 de píxel para la crominancia.
  - Filtros de interpolación en posiciones fraccionarias:
    - Luminancia → Separable, FIR de 7-8 muestras.
    - Crominancias → Separable, FIR de 4 muestras.
    - Mejor precisión que AVC → Filtro más largo y con menor redondeo.
  - Dos modos de predicción de vectores de movimiento:
    - AMVP (*Advanced Motion Vector Prediction*).
    - *Merge mode*.

# 4.3 – Ejemplo: Elecard HEVC analyzer (1)

The screenshot displays the Elecard HEVCAnalyzer v.0.101 interface. The main window shows a video frame with a grid overlay representing the HEVC coding structure. The interface is divided into several panels:

- Top Panel:** A timeline and bar chart showing video metrics over time.
- Left Panel (picture):** A table of video parameters:
 

name	value
resolution	1280 x 720
size (bytes)	
cu	3197
prediction	1.169
transform	1.852
qp	
min / max	33 / 33
pixels	
count	921.600
intra	74.304 (8,06%)
inter	288.832 (31,34%)
skip	558.464 (60,60%)
mv	
mv[0].x (max/min)	75 / -200
mv[0].y (max/min)	235 / -207
mv[1].x (max/min)	77 / -46
mv[1].y (max/min)	48 / -72
distribution (bits)	
total	27584
split_cu_flag	866
cu_skip_flag	921
prediction	
intra_prediction	1074
pred_mode_flag	496
part_mode	1000
inter_prediction	7268
transform	
rqt_root_cbf	347
split_transform_flag	567
cbf_cb	222
- Center Panel:** The main video frame with a grid overlay. A 'ctu presenter' window on the right shows a zoomed-in view of a Coding Tree Unit (CTU) with red lines indicating its structure.
- Right Panel (cu):** A table of Coding Unit (CU) parameters:
 

name	value
location	416x224
slice idx	0
tile idx	0
size	
ctu	1009
prediction	342
transform	651
coded unit	
type	PART_Nx2N
dimensions	16x16
depth	2
size	100
prediction	19
transform	81
transform unit	
dimensions	8x8
qp	33
prediction unit	
dimensions	8x16
- Bottom Panel:** A hex viewer showing the raw stream data in hexadecimal and ASCII format.

# 4.3 – Ejemplo: Elecard HEVC analyzer (2)

The screenshot displays the Elecard HEVC Analyzer v.0.101 interface. At the top, a toolbar includes navigation and analysis tools. Below it, a 'picture' panel on the left lists various video parameters:

name	value
resolution	1280 x 720
size (bytes)	
cu	17.944
prediction	1.904
transform	14.016
qp	
min / max	32 / 32
pixels	
count	921.600
intra	921.600 (100.00)
distribution (bits)	
total	149256
split_cu_flag	2072
cu_skip_flag	0
prediction	
intra_prediction	29038
pred_mode_flag	0
part_mode	2379
inter_prediction	0
transform	
rqf_root_cbf	0
split_transform_flag	2054
cbf_cb	2763
cbf_cr	2219
cbf_luma	7908
transform	97191

The central area shows a video frame with a red grid overlay, indicating the current picture being analyzed. To the right, a 'ctu presenter' shows a zoomed-in view of a Coding Tree Unit (CTU). At the bottom, a 'hex viewer' displays the raw stream data in hexadecimal and ASCII format.

## 4.3 – Ejemplo: *Elecard HEVC analyzer* (3)

The screenshot displays the Elecard HEVC Analyzer v.0.101 interface. The main window shows a video frame with overlaid analysis data. The interface is divided into several panels:

- Top Panel:** A timeline view showing the video stream with various markers and a blue line representing a metric.
- Left Panel (picture):** A table of video parameters and their values.
- Center Panel:** A video frame showing a woman on the left and a woman on the right, with a grid overlay on the woman on the left.
- Right Panel (ctu presenter):** A small grid overlay on a portion of the video frame.
- Bottom Right Panel (cu):** A table of CU (Coding Unit) parameters.
- Bottom Panel (hex viewer):** A hex viewer showing the raw video data in hexadecimal and ASCII format.

**picture parameters table:**

name	value
resolution	1280 x 720
size (bytes)	
cu	17.944
prediction	1.904
transform	14.016
qp	
min / max	32 / 32
pixels	
count	921.600
intra	921.600 (100.00)
distribution (bits)	
total	149256
split_cu_flag	2072
cu_skip_flag	0
prediction	
intra_prediction	29038
pred_mode_flag	0
part_mode	2379
inter_prediction	0
transform	
rq_t_root_cbf	0
split_transform_flag	2054
cbf_cb	2763
cbf_cr	2219
cbf_luma	7908
transform	97191

**cu parameters table:**

name	value
location	416x224
slice_idx	0
tile_idx	0
size	
ctu	1505
prediction	119
transform	1230
coded unit	
type	PART_2Nx2N
dimensions	16x16
depth	2
size	86
prediction	8
transform	78
transform unit	
dimensions	16x16
qp	32
prediction unit	
dimensions	16x16

**hex viewer output:**

```
0x00000001 00 00 01 40 01 0C 01 FF FF 00 80 00 00 03 00 00 ...@...y$.....
0x00000011 03 00 00 03 00 00 03 00 00 94 90 24 00 00 00 01 .....$.
0x00000021 42 01 01 00 80 00 00 03 00 00 03 00 00 03 00 00 B.....
```

# 4.3 – Ejemplo: Elecard HEVC analyzer (4)

The screenshot displays the Elecard HEVCAnalyzer v.0.101 interface. At the top, a bar chart shows the video's frame rate and other metrics. The main window is divided into several panels:

- picture**: A table of video parameters.
 

name	value
resolution	1280 x 720
size (bytes)	
cu	17.944
prediction	1.904
transform	14.016
qp	
min / max	32 / 32
pixels	
count	921.600
intra	921.600 (100.00)
distribution (bits)	
total	149256
split_cu_flag	2072
cu_skip_flag	0
prediction	
intra_prediction	29038
pred_mode_flag	0
part_mode	2379
inter_prediction	0
transform	
rqt_root_cbf	0
split_transform_flag	2054
cbf_cb	2763
cbf_cr	2219
cbf_luma	7908
transform	97191
- decoded**: A large central window showing a grayscale image of a person's face with a grid overlay, representing the decoded video frame.
- ctu presenter**: A small window showing a zoomed-in view of a Coding Tree Unit (CTU) with a grid overlay.
- cu**: A table of Coding Unit (CU) parameters.
 

name	value
location	416x224
slice idx	0
tile idx	0
size	
ctu	1505
prediction	119
transform	1230
coded unit	
type	PART_2Nx2N
dimensions	16x16
depth	2
size	86
prediction	8
transform	78
transform unit	
dimensions	16x16
qp	32
prediction unit	
dimensions	16x16
- hex viewer**: A window at the bottom showing the raw video data in hexadecimal format.

# 4.3 – Ejemplo: Elecard HEVC analyzer (5)

The screenshot displays the Elecard HEVC Analyzer v.0.101 interface. The main window shows a video frame with a grid overlay on a woman's face, indicating a coding unit (CU). The interface is divided into several panels:

- Top Panel:** A chart showing metrics over time, with a blue line and green bars.
- Left Panel (picture):** A table of video parameters:
 

name	value	%
resolution	1280 x 720	
size (bytes)		
cu	17.944	
prediction	1.904	
transform	14.016	
qp		
min / max	32 / 32	
pixels		
count	921.600	
intra	921.600 (100.00)	
distribution (bits)		
total	149256	
split_cu_flag	2072	
cu_skip_flag	0	
prediction		
intra_prediction	29038	
pred_mode_flag	0	
part_mode	2379	
inter_prediction	0	
transform		
rqt_root_cbf	0	
split_transform_flag	2054	
cbf_cb	2763	
cbf_cr	2219	
cbf_luma	7908	
transform	97191	
- Center Panel:** A video frame showing a woman and a man in a meeting. A grid is overlaid on the woman's face, and a 'ctu presenter' window shows a zoomed-in view of the grid.
- Right Panel (ctu presenter):** A table of CU parameters:
 

name	value
location	416x224
slice idx	0
tile idx	0
size	
ctu	1505
prediction	119
transform	1230
coded unit	
type	PART_2Nx2N
dimensions	16x16
depth	2
size	86
prediction	8
transform	78
transform unit	
dimensions	16x16
qp	32
prediction unit	
dimensions	16x16
- Bottom Panel (hex viewer):** A hex viewer showing the raw stream data in hexadecimal and ASCII format.
 

```

0x00000001 00 00 01 40 01 0C 01 FF FF 00 80 00 00 03 00 00 ...@...yy.....
0x00000011 03 00 00 03 00 00 03 00 00 94 90 24 00 00 00 01 .....$.
0x00000021 42 01 01 00 80 00 00 03 00 00 03 00 00 03 00 00 B.....
      
```

# 4.3 – Ejemplo: Elecard HEVC analyzer (6)

The screenshot displays the Elecard HEVC Analyzer interface. At the top, a bar chart shows metrics over time, with a red circle highlighting a specific data point. Below this, a central window shows a video frame with a grid overlay, representing the HEVC coding structure. To the left, a 'picture' table lists various parameters:

name	value
resolution	1280 x 720
size (bytes)	
cu	3.197
prediction	1.169
transform	1.852
qp	
min / max	33 / 33
pixels	
count	921.600
intra	74.304 (8,06%)
inter	288.832 (31,34%)
skip	558.464 (60,60%)
mv	
mv[0].x (max/min)	75 / -200
mv[0].y (max/min)	235 / -207
mv[1].x (max/min)	77 / -46
mv[1].y (max/min)	48 / -72
distribution (bits)	
total	27584
split_cu_flag	866
cu_skip_flag	921
prediction	
intra_prediction	1074
pred_mode_flag	496
part_mode	1000
inter_prediction	7268
transform	
rqt_root_cbf	347
split_transform_flag	567
cbf_cb	222

To the right, a 'ctu tree' table shows the hierarchical structure of the current coding tree unit:

name	value
transform	2
coding_quadtree	416, 208, 16
coding_quadtree	416, 208, 8
prediction	2
transform	5
coding_quadtree	424, 208, 8
prediction	2
transform	2
coding_quadtree	416, 216, 8
prediction	2
transform	12
coding_quadtree	424, 216, 8
prediction	3
transform	2
coding_quadtree	432, 208, 16
coding_quadtree	432, 208, 8
prediction	2
transform	23
coding_quadtree	440, 208, 8

At the bottom, a hex viewer shows the raw data in hexadecimal and ASCII format:

```

hex viewer
Display: hexadecimal | Text: ANSI | Columns: Auto
0x00005A65 00 00 01 02 01 E2 10 F2 13 63 B0 F9 88 97 59 31 .....&.ó.c"ú..Y1
0x00005A75 A8 AA 08 14 A8 B5 D1 8A D1 1E D4 D2 E6 D4 17 20 *...µñ.N.Óo&.
0x00005A85 F6 0A FC 65 68 D5 DF E7 F9 4F E3 50 DC A3 97 ß.1ehô&.UN&P&E.
hex viewer | stream viewer
ready Strm 9 Disp 16 Type B Size 3448 Offset 0x00005A65
  
```

# 4.3 – Ejemplo: Elecard HEVC analyzer (7)

The screenshot displays the Elecard HEVC Analyzer v.0.101 interface. The main window shows a video stream with a grid overlay, indicating the current frame being analyzed. The video content shows a person in a video conference. The interface is divided into several panels:

- Timeline:** Shows the video stream with a blue line representing the video data and red vertical bars representing keyframes or I-frames.
- Picture Properties:** A table listing various video parameters:

name	value
resolution	1280 x 720
size (bytes)	
cu	3.197
prediction	1.169
transform	1.852
qp	
min / max	33 / 33
pixels	
count	921.600
intra	74.304 (8,06%)
inter	288.832 (31,34%)
skip	558.464 (60,60%)
mv	
mv[0].x (max/min)	75 / -200
mv[0].y (max/min)	235 / -207
mv[1].x (max/min)	77 / -46
mv[1].y (max/min)	48 / -72
distribution (bits)	
total	27584
split_cu_flag	866
cu_skip_flag	921
prediction	
intra_prediction	1074
pred_mode_flag	496
part_mode	1000
inter_prediction	7268
transform	
rqt_root_cbf	347
split_transform_flag	567
cbf_cb	222

- ctu presenter:** Shows a small thumbnail of the current frame with a grid overlay.
- ctu tree:** A tree structure showing the coding tree for the current frame:

name	value
transform	2
coding_quadtree	416, 208, 16
coding_quadtree	416, 208, 8
prediction	2
transform	5
coding_quadtree	424, 208, 8
prediction	2
transform	2
coding_quadtree	416, 216, 8
prediction	2
transform	12
coding_quadtree	424, 216, 8
prediction	3
transform	2
coding_quadtree	432, 208, 16
coding_quadtree	432, 208, 8
prediction	2
transform	23
coding_quadtree	440, 208, 8

- hex viewer:** Shows the raw video data in hexadecimal and ASCII format.

# 4.3 – Ejemplo: Elecard HEVC analyzer (8)

The screenshot displays the Elecard HEVC Analyzer v.0.101 interface. The main window shows a video frame with overlaid analysis data. The interface is divided into several panels:

- Top Panel:** A timeline chart showing various metrics across the video stream.
- Left Panel (picture):** A table of video parameters:
 

name	value
resolution	1280 x 720
size (bytes)	
cu	3.197
prediction	1.169
transform	1.852
qp	
min / max	33 / 33
pixels	
count	921.600
intra	74.304 (8,06%)
inter	288.832 (31,34%)
skip	558.464 (60,60%)
mv	
mv[0].x (max/min)	75 / -200
mv[0].y (max/min)	235 / -207
mv[1].x (max/min)	77 / -46
mv[1].y (max/min)	48 / -72
distribution (bits)	
total	27584
split_cu_flag	866
cu_skip_flag	921
prediction	
intra_prediction	1074
pred_mode_flag	496
part_mode	1000
inter_prediction	7268
transform	
qrt_root_cbf	347
split_transform_flag	567
cbf_cb	222
- Center Panel:** A video frame showing two women in a meeting. A small grid overlay is visible on the left side of the frame.
- Right Panel (ctu presenter):** A small thumbnail of the video frame with a grid overlay.
- Bottom Right Panel (ctu tree):** A table showing the CTU tree structure:
 

name	value
transform	2
coding_quadtree	416, 208, 16
coding_quadtree	416, 208, 8
prediction	2
transform	5
coding_quadtree	424, 208, 8
prediction	2
transform	2
coding_quadtree	416, 216, 8
prediction	2
transform	12
coding_quadtree	424, 216, 8
prediction	3
transform	2
coding_quadtree	432, 208, 16
coding_quadtree	432, 208, 8
prediction	2
transform	23
coding_quadtree	440, 208, 8
- Bottom Panel (hex viewer):** A hex viewer showing the raw stream data in hexadecimal and ASCII format.
 

```

      0x00005A65 00 00 01 02 01 E2 10 F2 13 63 B0 F9 88 97 59 31 .....&.ó.c'á..Y1
      0x00005A75 A8 AA 08 14 A8 B5 D1 8A D1 1E D4 D2 E6 D4 17 20 *...µñ.N.Ó&æ.
      0x00005A85 F6 0A FC 65 68 D5 DF E7 F9 4F E3 50 DC A3 97 ß.1eh&æ.ü&æP&æ.
      
```

# 4.3 – Ejemplo: *Elecard HEVC analyzer* (10)

The screenshot displays the Elecard HEVC Analyzer v.0.101 interface. At the top, there is a menu bar and a toolbar. Below the toolbar is a 'picture' panel with a table of video parameters:

name	value
resolution	1280 x 720
size (bytes)	
cu	3.197
prediction	1.169
transform	1.852
qp	
min / max	33 / 33
pixels	
count	921.600
intra	74.304 (8,06%)
inter	288.832 (31,34%)
skip	558.464 (60,60%)
mv	
mv[0].x (max/min)	75 / -200
mv[0].y (max/min)	235 / -207
mv[1].x (max/min)	77 / -46
mv[1].y (max/min)	48 / -72
distribution (bits)	
total	27584
split_cu_flag	866
cu_skip_flag	921
prediction	
intra_prediction	1074
pred_mode_flag	496
part_mode	1000
inter_prediction	7268
transform	
qrt_root_cbf	347
split_transform_flag	567
cbf_cb	222

The central part of the interface shows a 'picture' window with tabs for 'decoded', 'predicted', 'unfiltered', 'residual', 'reference', and 'difference(none/decoded)'. The 'residual' tab is selected, showing a dark image with a small inset of a grid. To the right, there is a 'ctu presenter' window showing a grid of red lines on a blue background, and a 'ctu tree' window showing a list of CTU parameters:

name	value
transform	2
coding_quadtree	416, 208, 16
coding_quadtree	416, 208, 8
prediction	2
transform	5
coding_quadtree	424, 208, 8
prediction	2
transform	2
coding_quadtree	416, 216, 8
prediction	2
transform	12
coding_quadtree	424, 216, 8
prediction	3
transform	2
coding_quadtree	432, 208, 16
coding_quadtree	432, 208, 8
prediction	2
transform	23
coding_quadtree	440, 208, 8

At the bottom, there is a 'hex viewer' window showing hexadecimal data:

```

0x00005A65 00 00 01 02 01 E2 10 F2 13 63 B0 F9 88 97 59 31 .....â.ô.c"â..Y1
0x00005A75 A8 AA 08 14 A8 B5 D1 8A D1 1E D4 D2 E6 D4 17 20 *..µâ.N.Ôoæð.
0x00005A85 F6 0A FC 65 68 D5 DF E7 F9 4F E3 50 DC A3 97 ã.1ehðæ.üWâRð.
  
```

The status bar at the bottom indicates 'ready', 'Strm 9', 'Disp 16', 'Type B', 'Size 3448', and 'Offset 0x00005A65'.

# 4.3 – Ejemplo: Elecard HEVC analyzer (9)

The screenshot displays the Elecard HEVC Analyzer v.0.101 interface. At the top, there is a menu bar and a toolbar. Below the toolbar is a 'picture' panel with a table of video parameters:

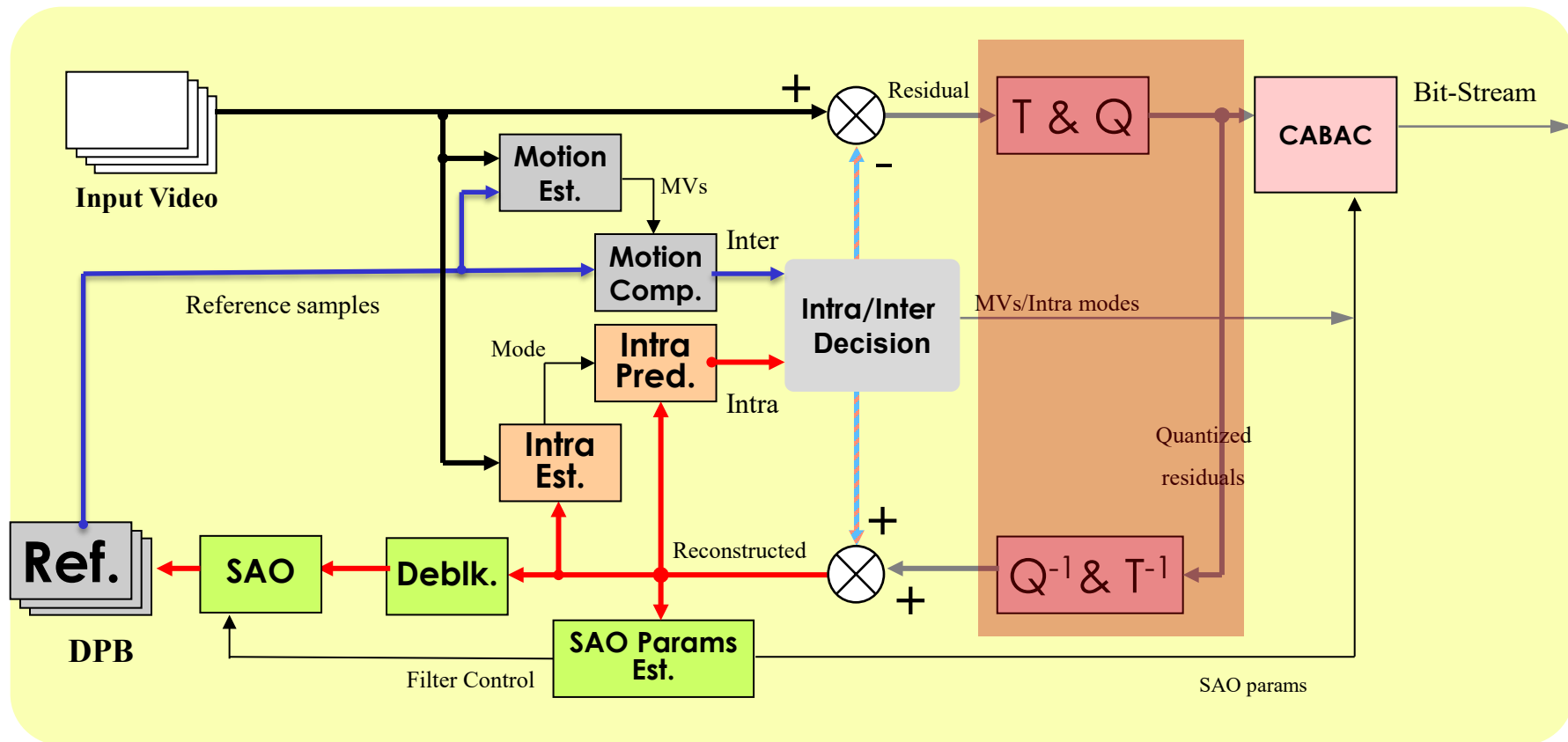
name	value
resolution	1280 x 720
size (bytes)	
cu	3.197
prediction	1.169
transform	1.852
qp	
min / max	33 / 33
pixels	
count	921.600
intra	74.304 (8,06%)
inter	288.832 (31,34%)
skip	558.464 (60,60%)
mv	
mv[0].x (max/min)	75 / -200
mv[0].y (max/min)	235 / -207
mv[1].x (max/min)	77 / -46
mv[1].y (max/min)	48 / -72
distribution (bits)	
total	27584
split_cu_flag	866
cu_skip_flag	921
prediction	
intra_prediction	1074
pred_mode_flag	496
part_mode	1000
inter_prediction	7268
transform	
qrt_root_cbf	347
split_transform_flag	567
cbf_cb	222

The central part of the interface is a video player showing a scene with two women in a meeting. To the right of the video player is a 'ctu presenter' panel showing a grid of red lines representing the Coding Tree Unit (CTU) structure. Below the video player is a 'ctu tree' panel with a table of CTU statistics:

name	value
transform	2
coding_quadtree	416, 208, 16
coding_quadtree	416, 208, 8
prediction	2
transform	5
coding_quadtree	424, 208, 8
prediction	2
transform	2
coding_quadtree	416, 216, 8
prediction	2
transform	12
coding_quadtree	424, 216, 8
prediction	3
transform	2
coding_quadtree	432, 208, 16
coding_quadtree	432, 208, 8
prediction	2
transform	23
coding_quadtree	440, 208, 8

At the bottom of the interface is a 'hex viewer' panel showing the raw data in hexadecimal and ASCII format. The status bar at the very bottom indicates 'ready', 'Strm 9', 'Disp 16', 'Type B', 'Size 3448', and 'Offset 0x00005A65'.

# 5 – Transformación (1)



## 5 – Transformación (2)

- Principales características:
  - Tanto la transformación como la cuantificación (y el escalado) se aplican a nivel de TB (partiendo de imágenes de error).
  - Cada tamaño de TB (4x4, 8x8, 16x16 y 32x32) requiere una transformación distinta.
  - Se aplica una transformación principal a todos los TB (*core transform*) y una adicional sólo a los TB de luminancia codificados en modo intra y de tamaño 4x4.

## 5 – Transformación (3)

- Transformación básica (*core transform*):
  - A cada TB se le aplica una transformada 2-D → Concatenando una transformación 1-D horizontal y otra 1-D vertical.
  - Los elementos de la matriz de transformación se han obtenido de aproximaciones de valores escalados de la DCT.
    - Similar a lo que se hace en AVC → Para mejorar la eficiencia computacional.
  - Por simplicidad, se utiliza una única matriz de 32x32 muestras.
    - Para TB de menor tamaño se submuestra.
  - HEVC (al igual que estándares previos) sólo define la matriz inversa.
    - Para que no haya pérdidas en la transformación, la matriz de transformación se diseña para que, concatenada con la inversa, no produzca pérdidas → Es la que vamos a analizar en las siguientes diapositivas.

# 5 – Transformación (4)

## Matriz de 32x32

transMatrixCol0to15 =

```

{64 64 64 64 64 64 64 64 64 64 64 64 64 64 64}
{90 90 88 85 82 78 73 67 61 54 46 38 31 22 13 4}
{90 87 80 70 57 43 25 9 -9 -25 -43 -57 -70 -80 -87 -90}
{90 82 67 46 22 -4 -31 -54 -73 -85 -90 -88 -78 -61 -38 -13}
{89 75 50 18 -18 -50 -75 -89 -89 -75 -50 -18 18 50 75 89}
{88 67 31 -13 -54 -82 -90 -78 -46 -4 38 73 90 85 61 22}
{87 57 9 -43 -80 -90 -70 -25 25 70 90 80 43 -9 -57 -87}
{85 46 -13 -67 -90 -73 -22 38 82 88 54 -4 -61 -90 -78 -31}
{83 36 -36 -83 -83 -36 36 83 83 36 -36 -83 -83 -36 36 83}
{82 22 -54 -90 -61 13 78 85 31 -46 -90 -67 4 73 88 38}
{80 9 -70 -87 -25 57 90 43 -43 -90 -57 25 87 70 -9 -80}
{78 -4 -82 -73 13 85 67 -22 -88 -61 31 90 54 -38 -90 -46}
{75 -18 -89 -50 50 89 18 -75 -75 18 89 50 -50 -89 -18 75}
{73 -31 -90 -22 78 67 -38 -90 -13 82 61 -46 -88 -4 85 54}
{70 -43 -87 9 90 25 -80 -57 57 80 -25 -90 -9 87 43 -70}
{67 -54 -78 38 85 -22 -90 4 90 13 -88 -31 82 46 -73 -61}
{64 -64 -64 64 64 -64 -64 64 64 -64 -64 64 64 -64 -64 64}
{61 -73 -46 82 31 -88 -13 90 -4 -90 22 85 -38 -78 54 67}
{57 -80 -25 90 -9 -87 43 70 -70 -43 87 9 -90 25 80 -57}
{54 -85 -4 88 -46 -61 82 13 -90 38 67 -78 -22 90 -31 -73}
{50 -89 18 75 -75 -18 89 -50 -50 89 -18 -75 75 18 -89 50}
{46 -90 38 54 -90 31 61 -88 22 67 -85 13 73 -82 4 78}
{43 -90 57 25 -87 70 9 -80 80 -9 -70 87 -25 -57 90 -43}
{38 -88 73 -4 -67 90 -46 -31 85 -78 13 61 -90 54 22 -82}
{36 -83 83 -36 -36 83 -83 36 36 -83 83 -36 -36 83 36}
{31 -78 90 -61 4 54 -88 82 -38 -22 73 -90 67 -13 -46 85}
{25 -70 90 -80 43 9 -57 87 -87 57 -9 -43 80 -90 70 -25}
{22 -61 85 -90 73 -38 -4 46 -78 90 -82 54 -13 -31 67 -88}
{18 -50 75 -89 89 -75 50 -18 -18 50 -75 89 -89 75 -50 18}
{13 -38 61 -78 88 -90 85 -73 54 -31 4 22 -46 67 -82 90}
{9 -25 43 -57 70 -80 87 -90 90 -87 80 -70 57 -43 25 -9}
{4 -13 22 -31 38 -46 54 -61 67 -73 78 -82 85 -88 90 -90}

```

transMatrixCol16to31 =

```

{64 64 64 64 64 64 64 64 64 64 64 64 64 64 64}
{-4 -13 -22 -31 -38 -46 -54 -61 -67 -73 -78 -82 -85 -88 -90 -90}
{-90 -87 -80 -70 -57 -43 -25 -9 9 25 43 57 70 80 87 90}
{13 38 61 78 88 90 85 73 54 31 4 -22 -46 -67 -82 -90}
{89 75 50 18 -18 -50 -75 -89 -89 -75 -50 -18 18 50 75 89}
{-22 -61 -85 -90 -73 -38 4 46 78 90 82 54 13 -31 -67 -88}
{-87 -57 -9 43 80 90 70 25 -25 -70 -90 -80 -43 9 57 87}
{31 78 90 61 4 -54 -88 -82 -38 22 73 90 67 13 -46 -85}
{83 36 -36 -83 -83 -36 36 83 83 36 -36 -83 -83 -36 36 83}
{-38 -88 -73 -4 67 90 46 -31 -85 -78 -13 61 90 54 -22 -82}
{-80 -9 70 87 25 -57 -90 -43 43 90 57 -25 -87 -70 9 80}
{46 90 38 -54 -90 -31 61 88 22 -67 -85 -13 73 82 4 -78}
{75 -18 -89 -50 50 89 18 -75 -75 18 89 50 -50 -89 -18 75}
{-54 -85 4 88 46 -61 -82 13 90 38 -67 -78 22 90 31 -73}
{-70 43 87 -9 -90 -25 80 57 -57 -80 25 90 9 -87 -43 70}
{61 73 -46 -82 31 88 -13 -90 -4 90 22 -85 -38 78 54 -67}
{64 -64 -64 64 64 -64 -64 64 64 -64 -64 64 64 -64 -64 64}
{-67 -54 78 38 -85 -22 90 4 -90 13 88 -31 -82 46 73 -61}
{-57 80 25 -90 9 87 -43 -70 70 43 -87 -9 90 -25 -80 57}
{73 31 -90 22 78 -67 -38 90 -13 -82 61 46 -88 4 85 -54}
{50 -89 18 75 -75 -18 89 -50 -50 89 -18 -75 75 18 -89 50}
{-78 -4 82 -73 -13 85 -67 -22 88 -61 -31 90 -54 -38 90 -46}
{-43 90 -57 -25 87 -70 -9 80 -80 9 70 -87 25 57 -90 43}
{82 -22 -54 90 -61 -13 78 -85 31 46 -90 67 4 -73 88 -38}
{36 -83 83 -36 -36 83 -83 36 36 -83 83 -36 -36 83 -83 36}
{-85 46 13 -67 90 -73 22 38 -82 88 -54 -4 61 -90 78 -31}
{-25 70 -90 80 -43 -9 57 -87 87 -57 9 43 -80 90 -70 25}
{88 -67 31 13 -54 82 -90 78 -46 4 38 -73 90 -85 61 -22}
{18 -50 75 -89 89 -75 50 -18 -18 50 -75 89 -89 75 -50 18}
{-90 82 -67 46 -22 -4 31 -54 73 -85 90 -88 78 -61 38 -13}
{-9 25 -43 57 -70 80 -87 90 -90 87 -80 70 -57 43 -25 9}
{90 -90 88 -85 82 -78 73 -67 61 -54 46 -38 31 -22 13 -4}

```

# 5 – Transformación (5)

## Matriz de 16x16

transMatrixCol0to15 =

```

{64 64 64 64 64 64 64 64 64 64 64 64 64 64 64}
{90 90 88 85 82 78 73 67 61 54 46 38 31 22 13 4}
{90 87 80 70 57 43 25 9 -9 -25 -43 -57 -70 -80 -87 -90}
{90 82 67 46 22 -4 -31 -54 -73 -85 -90 -88 -78 -61 -38 -13}
{89 75 50 18 -18 -50 -75 -89 -89 -75 -50 -18 18 50 75 89}
{88 67 31 -13 -54 -82 -90 -78 -46 -4 38 73 90 85 61 22}
{87 57 9 -43 -80 -90 -70 -25 25 70 90 80 43 -9 -57 -87}
{85 46 -13 -67 -90 -73 -22 38 82 88 54 -4 -61 -90 -78 -31}
{83 36 -36 -83 -83 -36 36 83 83 36 -36 -83 -83 -36 36 83}
{82 22 -54 -90 -61 13 78 85 31 -46 -90 -67 4 73 88 38}
{80 9 -70 -87 -25 57 90 43 -43 -90 -57 25 87 70 -9 -80}
{78 -4 -82 -73 13 85 67 -22 -88 -61 31 90 54 -38 -90 -46}
{75 -18 -89 -50 50 89 18 -75 -75 18 89 50 -50 -89 -18 75}
{73 -31 -90 -22 78 67 -38 -90 -13 82 61 -46 -88 -4 85 54}
{70 -43 -87 9 90 25 -80 -57 57 80 -25 -90 -9 87 43 -70}
{67 -54 -78 38 85 -22 -90 4 90 13 -88 -31 82 46 -73 -61}
{64 -64 -64 64 64 -64 -64 64 64 -64 -64 64 64 -64 -64 64}
{61 -73 -46 82 31 -88 -13 90 -4 -90 22 85 -38 -78 54 67}
{57 -80 -25 90 -9 -87 43 70 -70 -43 87 9 -90 25 80 -57}
{54 -85 -4 88 -46 -61 82 13 -90 38 67 -78 -22 90 -31 -73}
{50 -89 18 75 -75 -18 89 -50 -50 89 -18 -75 75 18 -89 50}
{46 -90 38 54 -90 31 61 -88 22 67 -85 13 73 -82 4 78}
{43 -90 57 25 -87 70 9 -80 80 -9 -70 87 -25 -57 90 -43}
{38 -88 73 -4 -67 90 -46 -31 85 -78 13 61 -90 54 22 -82}
{36 -83 83 -36 -36 83 -83 36 36 -83 83 -36 -36 83 -83 36}
{31 -78 90 -61 4 54 -88 82 -38 -22 73 -90 67 -13 -46 85}
{25 -70 90 -80 43 9 -57 87 -87 57 -9 -43 80 -90 70 -25}
{22 -61 85 -90 73 -38 -4 46 -78 90 -82 54 -13 -31 67 -88}
{18 -50 75 -89 89 -75 50 -18 -18 50 -75 89 -89 75 -50 18}
{13 -38 61 -78 88 -90 85 -73 54 -31 4 22 -46 67 -82 90}
{9 -25 43 -57 70 -80 87 -90 90 -87 80 -70 57 -43 25 -9}
{4 -13 22 -31 38 -46 54 -61 67 -73 78 -82 85 -88 90 -90}

```

transMatrixCol16to31 =

```

{64 64 64 64 64 64 64 64 64 64 64 64 64 64 64}
{-4 -13 -22 -31 -38 -46 -54 -61 -67 -73 -78 -82 -85 -88 -90 -90}
{-90 -87 -80 -70 -57 -43 -25 -9 9 25 43 57 70 80 87 90}
{13 38 61 78 88 90 85 73 54 31 4 -22 -46 -67 -82 -90}
{89 75 50 18 -18 -50 -75 -89 -89 -75 -50 -18 18 50 75 89}
{-22 -61 -85 -90 -73 -38 4 46 78 90 82 54 13 -31 -67 -88}
{-87 -57 -9 43 80 90 70 25 -25 -70 -90 -80 -43 9 57 87}
{31 78 90 61 4 -54 -88 -82 -38 22 73 90 67 13 -46 -85}
{83 36 -36 -83 -83 -36 36 83 83 36 -36 -83 -83 -36 36 83}
{-38 -88 -73 -4 67 90 46 -31 -85 -78 -13 61 90 54 -22 -82}
{-80 -9 70 87 25 -57 -90 -43 43 90 57 -25 -87 -70 9 80}
{46 90 38 -54 -90 -31 61 88 22 -67 -85 -13 73 82 4 -78}
{75 -18 -89 -50 50 89 18 -75 -75 18 89 50 -50 -89 -18 75}
{-54 -85 4 88 46 -61 -82 13 90 38 -67 -78 22 90 31 -73}
{-70 43 87 -9 -90 -25 80 57 -57 -80 25 90 9 -87 -43 70}
{61 73 -46 -82 31 88 -13 -90 -4 90 22 -85 -38 78 54 -67}
{64 -64 -64 64 64 -64 -64 64 64 -64 -64 64 64 -64 -64 64}
{-67 -54 78 38 -85 -22 90 4 -90 13 88 -31 -82 46 73 -61}
{-57 80 25 -90 9 87 -43 -70 70 43 -87 -9 90 -25 -80 57}
{73 31 -90 22 78 -67 -38 90 -13 -82 61 46 -88 4 85 -54}
{50 -89 18 75 -75 -18 89 -50 -50 89 -18 -75 75 18 -89 50}
{-78 -4 82 -73 -13 85 -67 -22 88 -61 -31 90 -54 -38 90 -46}
{-43 90 -57 -25 87 -70 -9 80 -80 9 70 -87 25 57 -90 43}
{82 -22 -54 90 -61 -13 78 -85 31 46 -90 67 4 -73 88 -38}
{36 -83 83 -36 -36 83 -83 36 36 -83 83 -36 -36 83 -83 36}
{-85 46 13 -67 90 -73 22 38 -82 88 -54 -4 61 -90 78 -31}
{-25 70 -90 80 -43 -9 57 -87 87 -57 9 43 -80 90 -70 25}
{88 -67 31 13 -54 82 -90 78 -46 4 38 -73 90 -85 61 -22}
{18 -50 75 -89 89 -75 50 -18 -18 50 -75 89 -89 75 -50 18}
{-90 82 -67 46 -22 -4 31 -54 73 -85 90 -88 78 -61 38 -13}
{-9 25 -43 57 -70 80 -87 90 -90 87 -80 70 -57 43 -25 9}
{90 -90 88 -85 82 -78 73 -67 61 -54 46 -38 31 -22 13 -4}

```

# 5 – Transformación (6)

## Matriz de 8x8

transMatrixCol0to15 =

```

{ 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 }
{ 90 90 88 85 82 78 73 67 61 54 46 38 31 22 13 4 }
{ 90 87 80 70 57 43 25 9 -9 -25 -43 -57 -70 -80 -87 -90 }
{ 90 82 67 46 22 -4 -31 -54 -73 -85 -90 -88 -78 -61 -38 -13 }
{ 89 75 50 18 -18 -50 -75 -89 -89 -75 -50 -18 18 50 75 89 }
{ 88 67 31 -13 -54 -82 -90 -78 -46 -4 38 73 90 85 61 22 }
{ 87 57 9 -43 -80 -90 -70 -25 25 70 90 80 43 -9 -57 -87 }
{ 85 46 -13 -67 -90 -73 -22 38 82 88 54 -4 -61 -90 -78 -31 }
{ 83 36 -36 -83 -83 -36 36 83 83 36 -36 -83 -83 -36 36 83 }
{ 82 22 -54 -90 -61 13 78 85 31 -46 -90 -67 4 73 88 38 }
{ 80 9 -70 -87 -25 57 90 43 -43 -90 -57 25 87 70 -9 -80 }
{ 78 -4 -82 -73 13 85 67 -22 -88 -61 31 90 54 -38 -90 -46 }
{ 75 -18 -89 -50 50 89 18 -75 -75 18 89 50 -50 -89 -18 75 }
{ 73 -31 -90 -22 78 67 -38 -90 -13 82 61 -46 -88 -4 85 54 }
{ 70 -43 -87 9 90 25 -80 -57 57 80 -25 -90 -9 87 43 -70 }
{ 67 -54 -78 38 85 -22 -90 4 90 13 -88 -31 82 46 -73 -61 }
{ 64 -64 -64 64 64 -64 -64 64 64 -64 -64 64 64 -64 -64 64 }
{ 61 -73 -46 82 31 -88 -13 90 -4 -90 22 85 -38 -78 54 67 }
{ 57 -80 -25 90 -9 -87 43 70 -70 -43 87 9 -90 25 80 -57 }
{ 54 -85 -4 88 -46 -61 82 13 -90 38 67 -78 -22 90 -31 -73 }
{ 50 -89 18 75 -75 -18 89 -50 -50 89 -18 -75 75 18 -89 50 }
{ 46 -90 38 54 -90 31 61 -88 22 67 -85 13 73 -82 4 78 }
{ 43 -90 57 25 -87 70 9 -80 80 -9 -70 87 -25 -57 90 -43 }
{ 38 -88 73 -4 -67 90 -46 -31 85 -78 13 61 -90 54 22 -82 }
{ 36 -83 83 -36 -36 83 -83 36 36 -83 83 -36 -36 83 -36 }
{ 31 -78 90 -61 4 54 -88 82 -38 -22 73 -90 67 -13 -46 85 }
{ 25 -70 90 -80 43 9 -57 87 -87 57 -9 -43 80 -90 70 -25 }
{ 22 -61 85 -90 73 -38 -4 46 -78 90 -82 54 -13 -31 67 -88 }
{ 18 -50 75 -89 89 -75 50 -18 -18 50 -75 89 -89 75 -50 18 }
{ 13 -38 61 -78 88 -90 85 -73 54 -31 4 22 -46 67 -82 90 }
{ 9 -25 43 -57 70 -80 87 -90 90 -87 80 -70 57 -43 25 -9 }
{ 4 -13 22 -31 38 -46 54 -61 67 -73 78 -82 85 -88 90 -90 }

```

transMatrixCol16to31 =

```

{ 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 }
{ -4 -13 -22 -31 -38 -46 -54 -61 -67 -73 -78 -82 -85 -88 -90 -90 }
{ -90 -87 -80 -70 -57 -43 -25 -9 9 25 43 57 70 80 87 90 }
{ 13 38 61 78 88 90 85 73 54 31 4 -22 -46 -67 -82 -90 }
{ 89 75 50 18 -18 -50 -75 -89 -89 -75 -50 -18 18 50 75 89 }
{ -22 -61 -85 -90 -73 -38 4 46 78 90 82 54 13 -31 -67 -88 }
{ -87 -57 -9 43 80 90 70 25 -25 -70 -90 -80 -43 9 57 87 }
{ 31 78 90 61 4 -54 -88 -82 -38 22 73 90 67 13 -46 -85 }
{ 83 36 -36 -83 -83 -36 36 83 83 36 -36 -83 -83 -36 36 83 }
{ -38 -88 -73 -4 67 90 46 -31 -85 -78 -13 61 90 54 -22 -82 }
{ -80 -9 70 87 25 -57 -90 -43 43 90 57 -25 -87 -70 9 80 }
{ 46 90 38 -54 -90 -31 61 88 22 -67 -85 -13 73 82 4 -78 }
{ 75 -18 -89 -50 50 89 18 -75 -75 18 89 50 -50 -89 -18 75 }
{ -54 -85 4 88 46 -61 -82 13 90 38 -67 -78 22 90 31 -73 }
{ -70 43 87 -9 -90 -25 80 57 -57 -80 25 90 9 -87 -43 70 }
{ 61 73 -46 -82 31 88 -13 -90 -4 90 22 -85 -38 78 54 -67 }
{ 64 -64 -64 64 64 -64 -64 64 64 -64 -64 64 64 -64 -64 64 }
{ -67 -54 78 38 -85 -22 90 4 -90 13 88 -31 -82 46 73 -61 }
{ -57 80 25 -90 9 87 -43 -70 70 43 -87 -9 90 -25 -80 57 }
{ 73 31 -90 22 78 -67 -38 90 -13 -82 61 46 -88 4 85 -54 }
{ 50 -89 18 75 -75 -18 89 -50 -50 89 -18 -75 75 18 -89 50 }
{ -78 -4 82 -73 -13 85 -67 -22 88 -61 -31 90 -54 -38 90 -46 }
{ -43 90 -57 -25 87 -70 -9 80 -80 9 70 -87 25 57 -90 43 }
{ 82 -22 -54 90 -61 -13 78 -85 31 46 -90 67 4 -73 88 -38 }
{ 36 -83 83 -36 -36 83 -83 36 36 -83 83 -36 -36 83 -36 }
{ -85 46 13 -67 90 -73 22 38 -82 88 -54 -4 61 -90 78 -31 }
{ -25 70 -90 80 -43 -9 57 -87 87 -57 9 43 -80 90 -70 25 }
{ 88 -67 31 13 -54 82 -90 78 -46 4 38 -73 90 -85 61 -22 }
{ 18 -50 75 -89 89 -75 50 -18 -18 50 -75 89 -89 75 -50 18 }
{ -90 82 -67 46 -22 -4 31 -54 73 -85 90 -88 78 -61 38 -13 }
{ -9 25 -43 57 -70 80 -87 90 -90 87 -80 70 -57 43 -25 9 }
{ 90 -90 88 -85 82 -78 73 -67 61 -54 46 -38 31 -22 13 -4 }

```

# 5 – Transformación (7)

## Matriz de 4x4

transMatrixCol0to15 =

```

{ 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 }
{ 90 90 88 85 82 78 73 67 61 54 46 38 31 22 13 4 }
{ 90 87 80 70 57 43 25 9 -9 -25 -43 -57 -70 -80 -87 -90 }
{ 90 82 67 46 22 -4 -31 -54 -73 -85 -90 -88 -78 -61 -38 -13 }
{ 89 75 50 18 -18 -50 -75 -89 -89 -75 -50 -18 18 50 75 89 }
{ 88 67 31 -13 -54 -82 -90 -78 -46 -4 38 73 90 85 61 22 }
{ 87 57 9 -43 -80 -90 -70 -25 25 70 90 80 43 -9 -57 -87 }
{ 85 46 -13 -67 -90 -73 -22 38 82 88 54 -4 -61 -90 -78 -31 }
{ 83 36 -36 -83 -83 -36 36 83 83 36 -36 -83 -83 -36 36 83 }
{ 82 22 -54 -90 -61 13 78 85 31 -46 -90 -67 4 73 88 38 }
{ 80 9 -70 -87 -25 57 90 43 -43 -90 -57 25 87 70 -9 -80 }
{ 78 -4 -82 -73 13 85 67 -22 -88 -61 31 90 54 -38 -90 -46 }
{ 75 -18 -89 -50 50 89 18 -75 -75 18 89 50 -50 -89 -18 75 }
{ 73 -31 -90 -22 78 67 -38 -90 -13 82 61 -46 -88 -4 85 54 }
{ 70 -43 -87 9 90 25 -80 -57 57 80 -25 -90 -9 87 43 -70 }
{ 67 -54 -78 38 85 -22 -90 4 90 13 -88 -31 82 46 -73 -61 }
{ 64 -64 -64 64 64 -64 -64 64 64 -64 -64 64 64 -64 -64 64 }
{ 61 -73 -46 82 31 -88 -13 90 -4 -90 22 85 -38 -78 54 67 }
{ 57 -80 -25 90 -9 -87 43 70 -70 -43 87 9 -90 25 80 -57 }
{ 54 -85 -4 88 -46 -61 82 13 -90 38 67 -78 -22 90 -31 -73 }
{ 50 -89 18 75 -75 -18 89 -50 -50 89 -18 -75 75 18 -89 50 }
{ 46 -90 38 54 -90 31 61 -88 22 67 -85 13 73 -82 4 78 }
{ 43 -90 57 25 -87 70 9 -80 80 -9 -70 87 -25 -57 90 -43 }
{ 38 -88 73 -4 -67 90 -46 -31 85 -78 13 61 -90 54 22 -82 }
{ 36 -83 83 -36 -36 83 -83 36 36 -83 83 -36 -36 83 -36 }
{ 31 -78 90 -61 4 54 -88 82 -38 -22 73 -90 67 -13 -46 85 }
{ 25 -70 90 -80 43 9 -57 87 -87 57 -9 -43 80 -90 70 -25 }
{ 22 -61 85 -90 73 -38 -4 46 -78 90 -82 54 -13 -31 67 -88 }
{ 18 -50 75 -89 89 -75 50 -18 -18 50 -75 89 -89 75 -50 18 }
{ 13 -38 61 -78 88 -90 85 -73 54 -31 4 22 -46 67 -82 90 }
{ 9 -25 43 -57 70 -80 87 -90 90 -87 80 -70 57 -43 25 -9 }
{ 4 -13 22 -31 38 -46 54 -61 67 -73 78 -82 85 -88 90 -90 }

```

transMatrixCol16to31 =

```

{ 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 }
{ -4 -13 -22 -31 -38 -46 -54 -61 -67 -73 -78 -82 -85 -88 -90 -90 }
{ -90 -87 -80 -70 -57 -43 -25 -9 9 25 43 57 70 80 87 90 }
{ 13 38 61 78 88 90 85 73 54 31 4 -22 -46 -67 -82 -90 }
{ 89 75 50 18 -18 -50 -75 -89 -89 -75 -50 -18 18 50 75 89 }
{ -22 -61 -85 -90 -73 -38 4 46 78 90 82 54 13 -31 -67 -88 }
{ -87 -57 -9 43 80 90 70 25 -25 -70 -90 -80 -43 9 57 87 }
{ 31 78 90 61 4 -54 -88 -82 -38 22 73 90 67 13 -46 -85 }
{ 83 36 -36 -83 -83 -36 36 83 83 36 -36 -83 -83 -36 36 83 }
{ -38 -88 -73 -4 67 90 46 -31 -85 -78 -13 61 90 54 -22 -82 }
{ -80 -9 70 87 25 -57 -90 -43 43 90 57 -25 -87 -70 9 80 }
{ 46 90 38 -54 -90 -31 61 88 22 -67 -85 -13 73 82 4 -78 }
{ 75 -18 -89 -50 50 89 18 -75 -75 18 89 50 -50 -89 -18 75 }
{ -54 -85 4 88 46 -61 -82 13 90 38 -67 -78 22 90 31 -73 }
{ -70 43 87 -9 -90 -25 80 57 -57 -80 25 90 9 -87 -43 70 }
{ 61 73 -46 -82 31 88 -13 -90 -4 90 22 -85 -38 78 54 -67 }
{ 64 -64 -64 64 64 -64 -64 64 64 -64 -64 64 64 -64 -64 64 }
{ -67 -54 78 38 -85 -22 90 4 -90 13 88 -31 -82 46 73 -61 }
{ -57 80 25 -90 9 87 -43 -70 70 43 -87 -9 90 -25 -80 57 }
{ 73 31 -90 22 78 -67 -38 90 -13 -82 61 46 -88 4 85 -54 }
{ 50 -89 18 75 -75 -18 89 -50 -50 89 -18 -75 75 18 -89 50 }
{ -78 -4 82 -73 -13 85 -67 -22 88 -61 -31 90 -54 -38 90 -46 }
{ -43 90 -57 -25 87 -70 -9 80 -80 9 70 -87 25 57 -90 43 }
{ 82 -22 -54 90 -61 -13 78 -85 31 46 -90 67 4 -73 88 -38 }
{ 36 -83 83 -36 -36 83 -83 36 36 -83 83 -36 -36 83 -36 }
{ -85 46 13 -67 90 -73 22 38 -82 88 -54 -4 61 -90 78 -31 }
{ -25 70 -90 80 -43 -9 57 -87 87 -57 9 43 -80 90 -70 25 }
{ 88 -67 31 13 -54 82 -90 78 -46 4 38 -73 90 -85 61 -22 }
{ 18 -50 75 -89 89 -75 50 -18 -18 50 -75 89 -89 75 -50 18 }
{ -90 82 -67 46 -22 -4 31 -54 73 -85 90 -88 78 -61 38 -13 }
{ -9 25 -43 57 -70 80 -87 90 -90 87 -80 70 -57 43 -25 9 }
{ 90 -90 88 -85 82 -78 73 -67 61 -54 46 -38 31 -22 13 -4 }

```

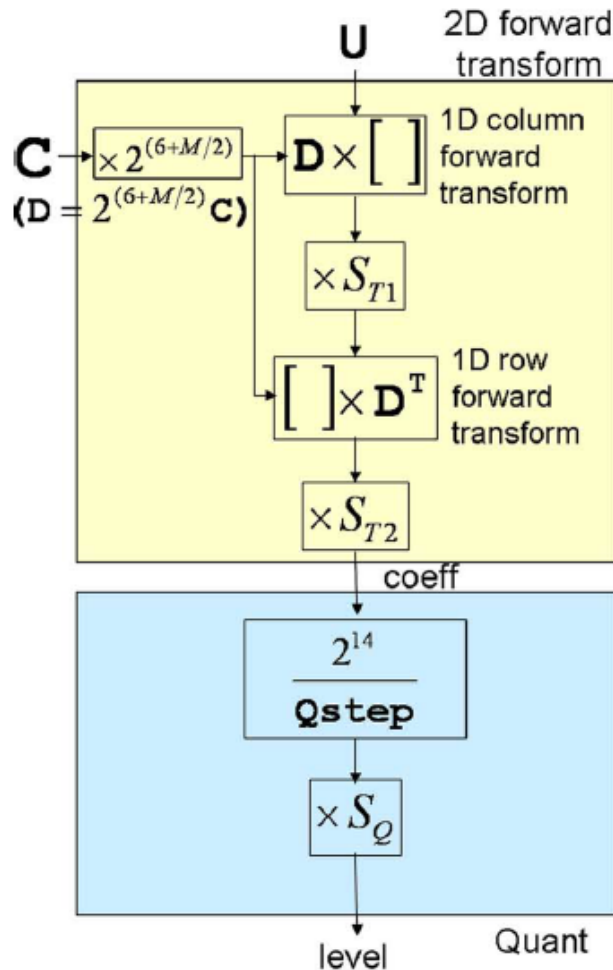
## 5 – Transformación (8)

- Transformación adicional 4x4:
  - Se aplica a TB de tamaño 4x4 (luminancia en modo intra).
  - Es una transformación derivada de la DST (*Discrete Sine Transform*).
  - Sólo a bloques de luminancia codificados en modo intra.
    - Son bloques que, en principio, requerirán mayor compresión.
    - Consigue reducir aproximadamente un 1% la tasa binaria de estos bloques.
    - Se ha comprobado que en otros bloques no vale la pena aplicarla.
  - La matriz utilizada es la siguiente:

$$\begin{bmatrix} 29 & 55 & 74 & 84 \\ 74 & 74 & 0 & -74 \\ 84 & -29 & -74 & 55 \\ 55 & -84 & 74 & -29 \end{bmatrix}$$

## 5 – Transformación (9)

- Escalado:



- Dado que las matrices de transformación utilizadas están escaladas con respecto a la DCT, para mantener la ortonormalidad es necesario re-escalarlas.
- En este caso, el software de referencia de HEVC sí que especifica los valores que han de usarse en el codificador.

## 6 – Cuantificación (1)

- Se utiliza un cuantificador uniforme controlado por el parámetro  $QP$ :
  - Permite determinar el valor del escalón de cuantificación ( $Q_{STEP}$ ).
  - Su valor se predice a partir del  $QP$  de los datos descodificados.
  - El número de valores posibles de  $QP$  depende de los bits que se utilicen:

BitDepth	8	9	10	11	12	13	14
Nº QP values	52	58	64	70	76	82	88
Range	0..51	0..57	0..63	0..69	0..77	0..81	0..87

## 6 – Cuantificación (2)

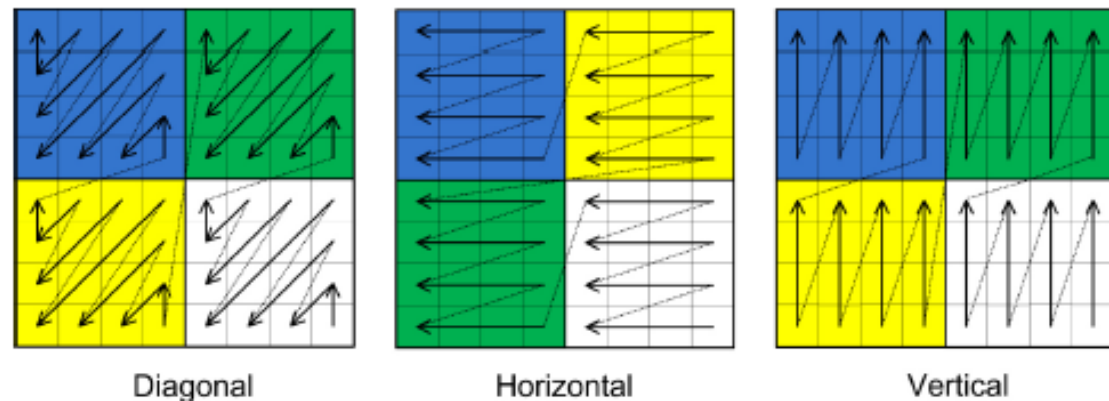
- Si se incrementa en 1 el valor de  $QP \rightarrow$  El escalón de cuantificación aumenta aproximadamente un 12% ( $2^{1/6}$ ).
- Por lo tanto:
  - Si se incrementa en 6  $\rightarrow$  El escalón se duplica.
  - Si se añade un bit  $\rightarrow$  6 valores más de  $QP \rightarrow$  El doble de intervalos.
- La relación entre  $QP$  y  $Q_{STEP}$  se puede expresar como:

$$Q_{STEP}(QP) = \left(2^{1/6}\right)^{QP-4}$$

## 6 – Cuantificación (3)

- Cada TB se divide en sub-bloques de 4x4.
- Los coeficientes se recorren en orden inverso (abajo-derecha → arriba-izquierda).
- Un mapa de significancia indica qué coeficientes no son nulos.
- Se empieza a procesar el primer coeficiente no nulo encontrado y se sigue hasta llegar al coeficiente DC.

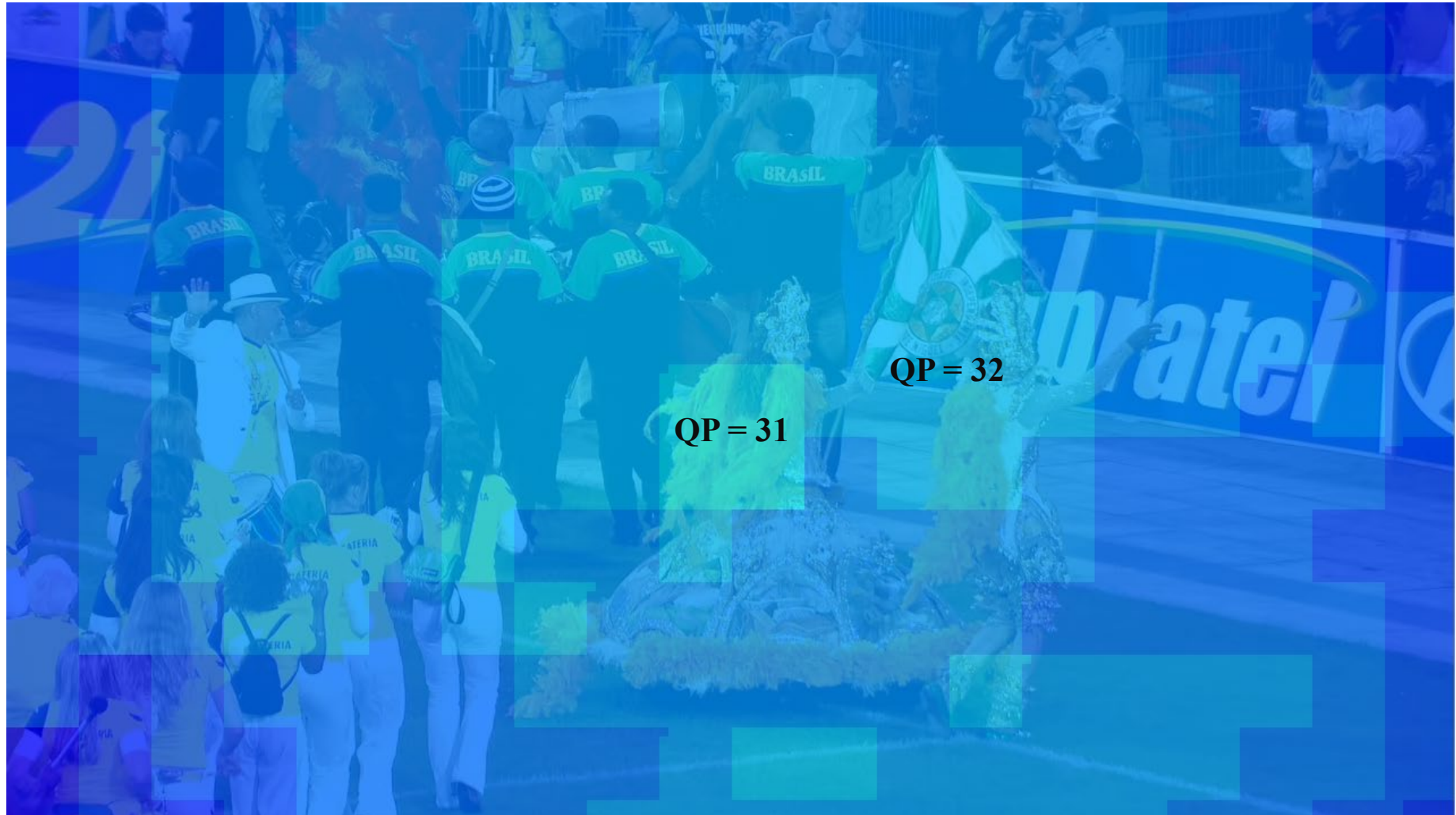
**Existen 3 posibles modos para recorrer los bloques**



TU Size	Prediction Type	Scanning Order
4x4, 8x8	Intra (mode-dependent)	Vertical, Horizontal
All	Intra (mode-dependent), Inter	4x4 Sub-diagonal

## 6 – Cuantificación (3)

- Ejemplo:

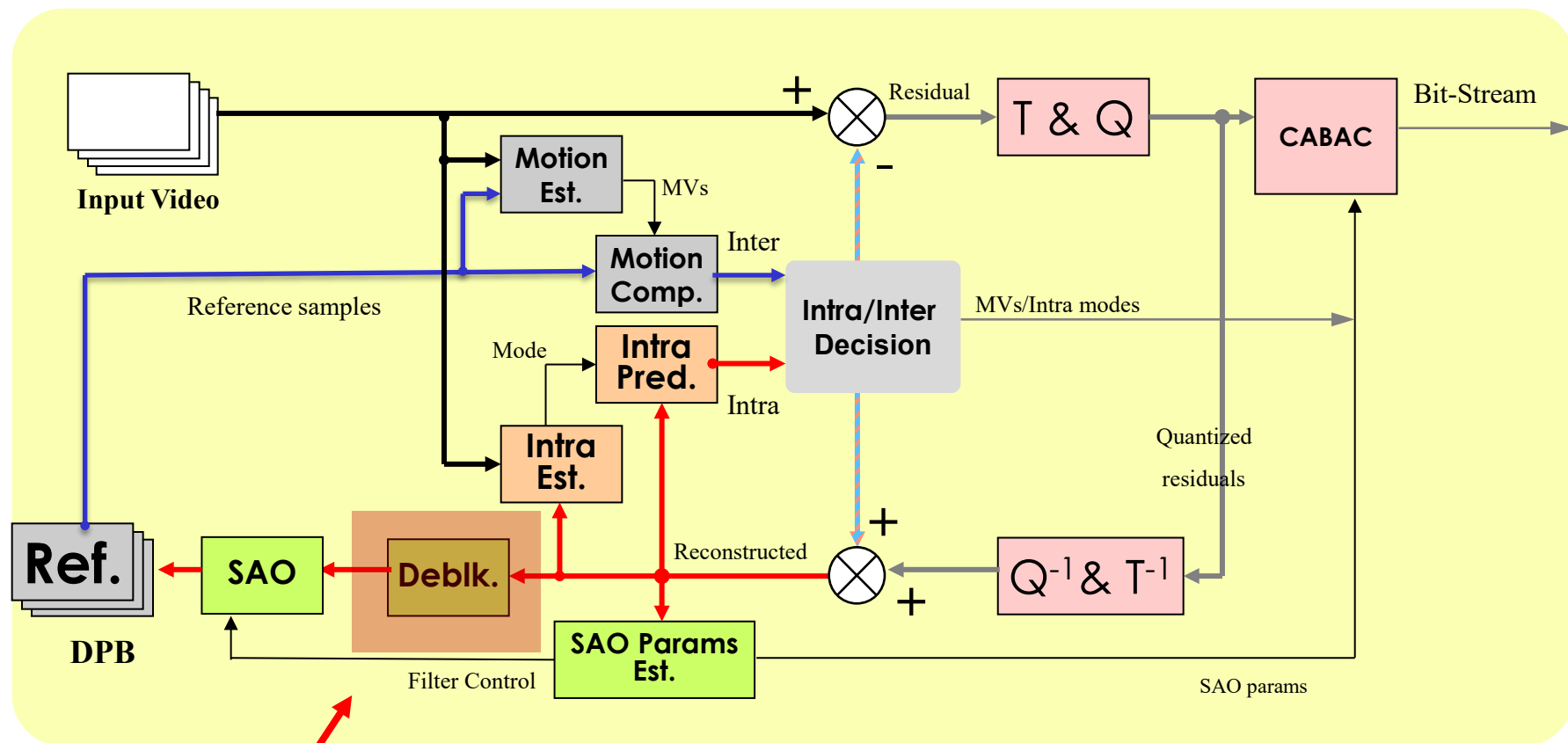


## 6 – Cuantificación (4)

- Ejemplo:

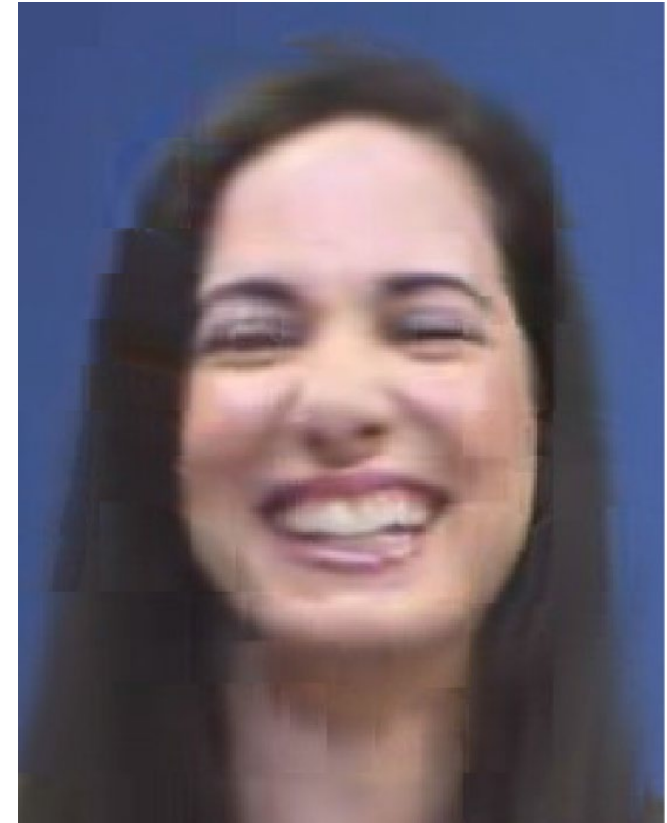


# 7 – Filtro de reconstrucción (1)



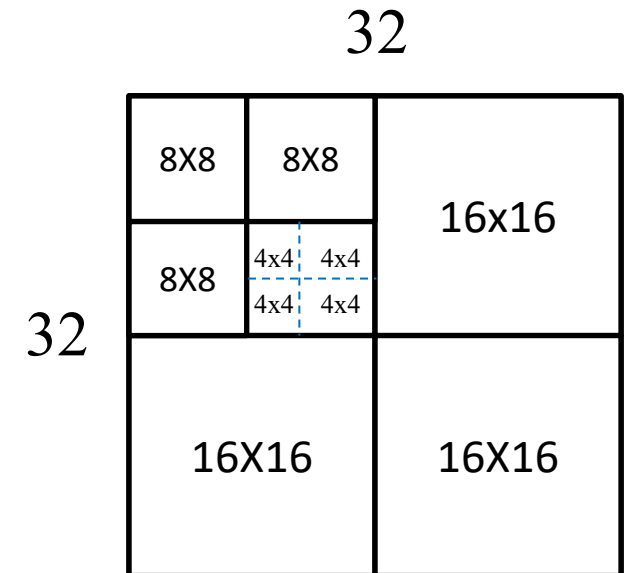
## 7 – Filtro de reconstrucción (2)

- Su objetivo es reducir la visibilidad entre fronteras de bloques.
- En comparación con el de AVC:
  - Menor complejidad computacional.
  - Mejor capacidad para su implementación en arquitecturas de procesamiento en paralelo.
  - Calidad similar.



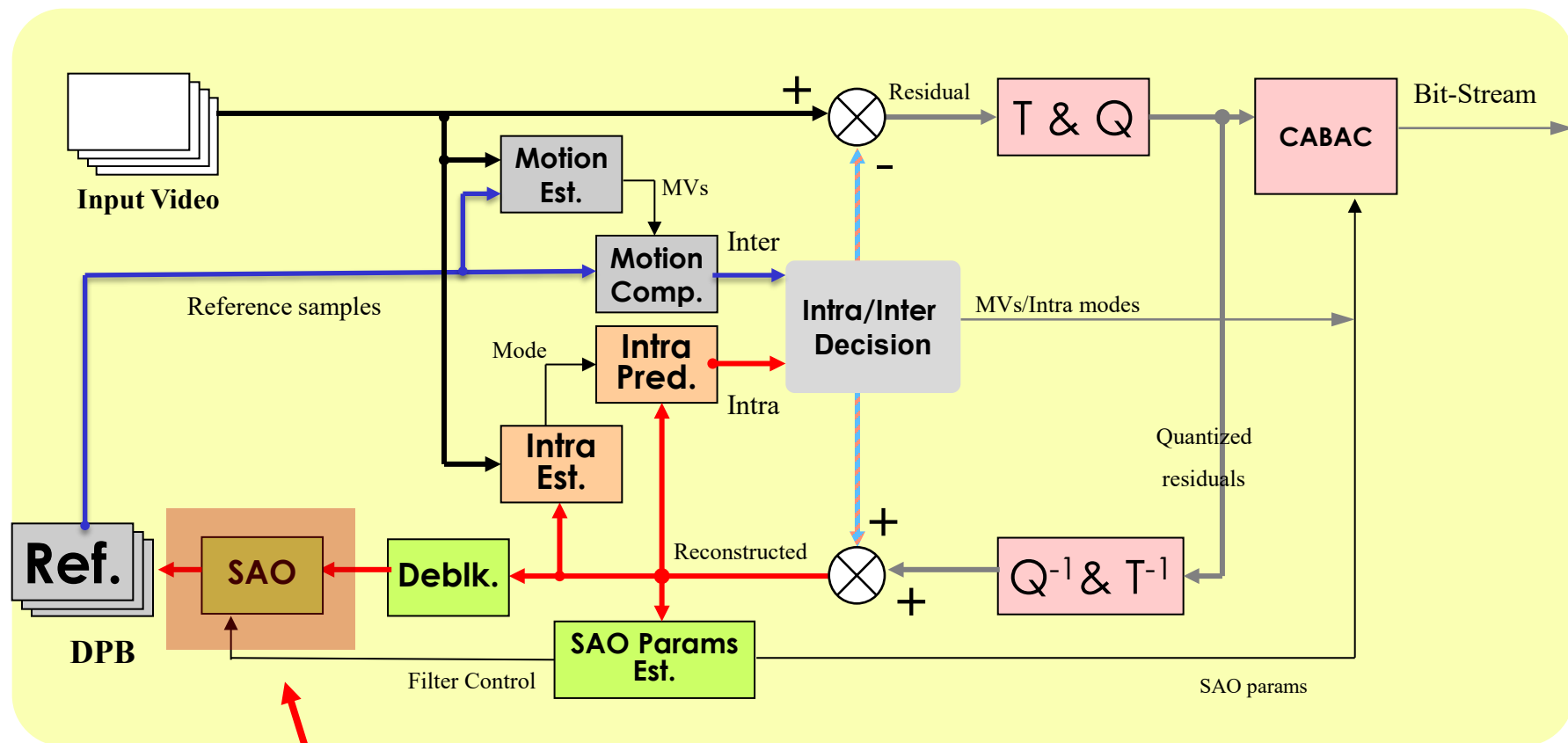
## 7 – Filtro de reconstrucción (2)

- El tipo de filtrado aplicado se basa en el análisis de los coeficientes descodificados a cada lado de las fronteras analizadas.
- El filtrado se aplica en todas las fronteras de PBs y TBs, con las siguientes excepciones:
  - Las fronteras de la imagen.
  - Las fronteras de una *slice* (cuando se desactiva la opción de filtrar a nivel de secuencia).
  - Las fronteras de una *tile* (idem).
- El tamaño mínimo de bloque filtrado es de 8x8 (en AVC era de 4x4).



Deblocked  
 Non-deblocked

## 8 – Sample Adaptive Offset - SAO (1)



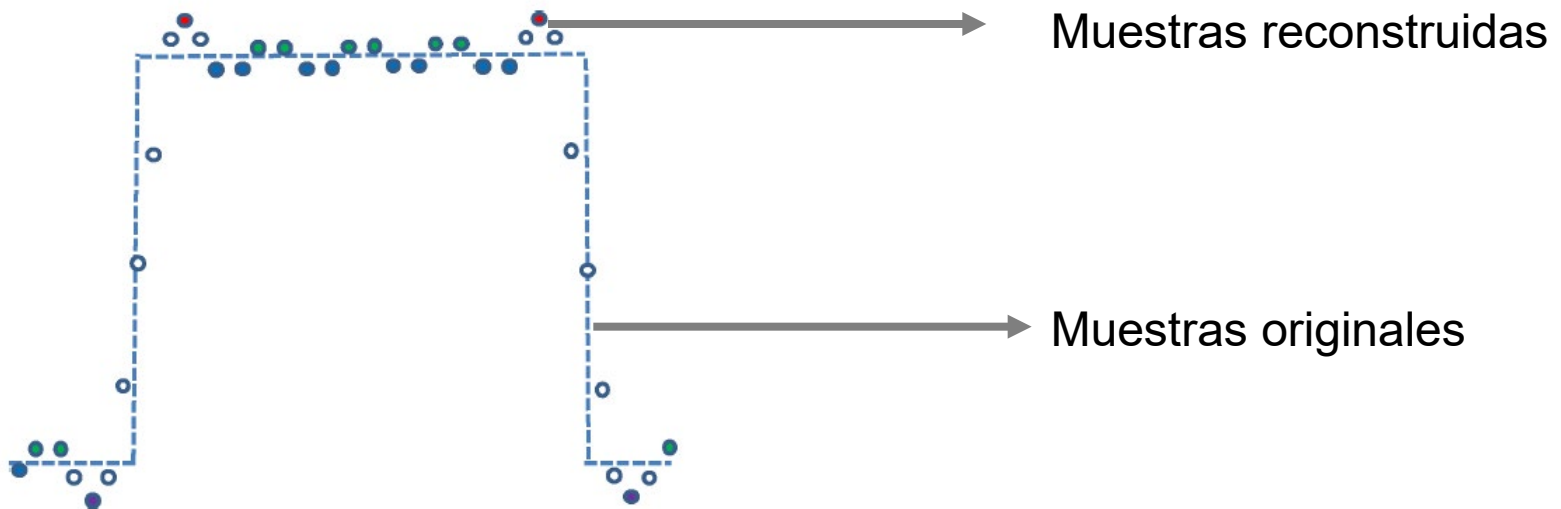
## 8 – *Sample Adaptive Offset - SAO* (2)

- Su objetivo es reducir la presencia de “artefactos” visuales que el filtro de reconstrucción no es capaz de eliminar:
  - Debidos a cuantificaciones demasiado groseras de algunos coeficientes transformados.
  - Efectos no sesados debidos al uso de la interpolación.
- Corrige la amplitud de las muestras descodificadas:
  - Aplica *offsets* almacenados en una tabla (*lookup table*) que se envían en el *bitstream*.
- Cuando mayor es el tamaño de las TUs, más importante es su USO.

## 8 – Sample Adaptive Offset - SAO (3)

- Ejemplo:

- La cuantificación hace que las muestras reconstruidas difieran de las originales.
- El error de cuantificación no está distribuido uniformemente a lo largo de los píxeles (es mayor en los bordes).

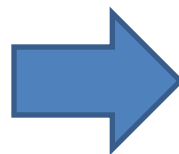


## 8 – *Sample Adaptive Offset* - SAO (4)

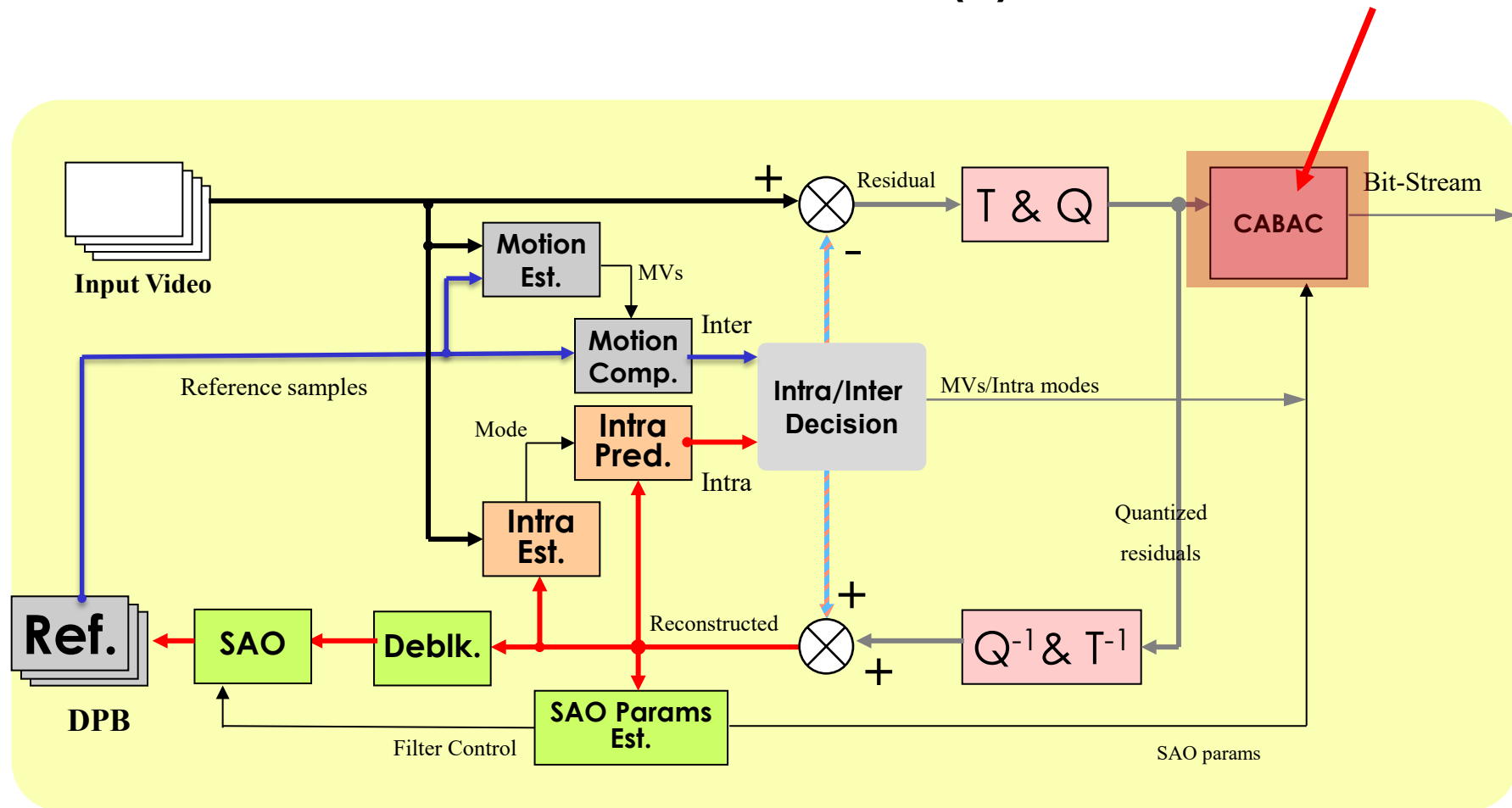
SAO desactivado (QP = 32 )



SAO activado (QP = 32 )



# 9 – Codificación (1)



## 9 – Codificación (2)

- HEVC propone el uso de un único método de codificación entrópica:
  - CABAC → *Context-Adaptive Binary Arithmetic Coding*.
  - En AVC se permitía el uso de CABAC y CAVLC.
- HEVC-CABAC consta de 3 etapas:
  - 1) Binarización → Mapear los datos a símbolos binarios (*bins*, *codewords*).
  - 2) Modelado del contexto → Estimar la probabilidad de los *bins*.
  - 3) Codificación aritmética → Compresión de los *bins* a partir de las probabilidades estimadas (usa la misma aritmética que AVC).