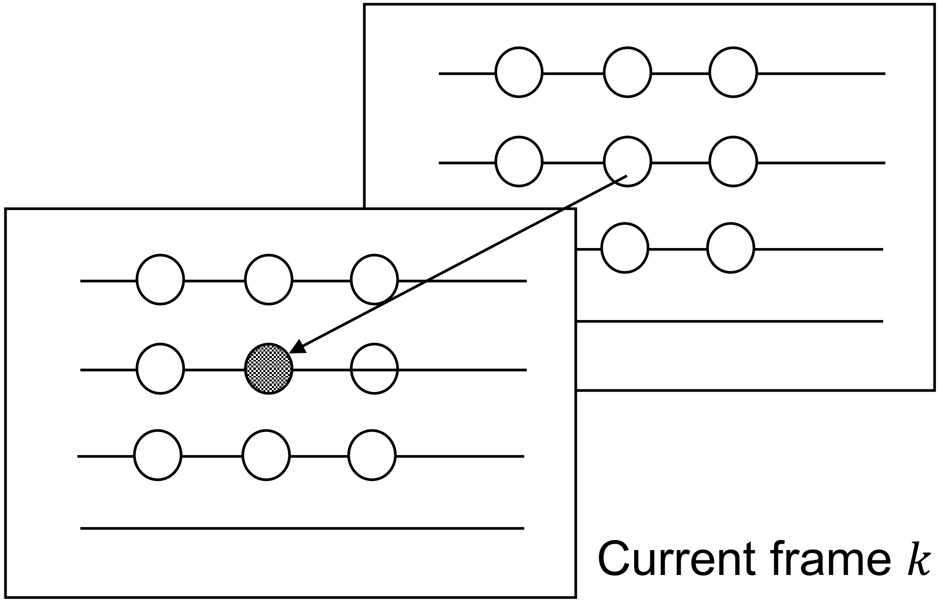
11 Interframe Coding

11.1 Interframe Prediction

Interframe coding exploits high similarity of temporally successive frames in a video sequence
⇨ Predict current frame k from previous frame $k-1$



Previous frame $k-1$

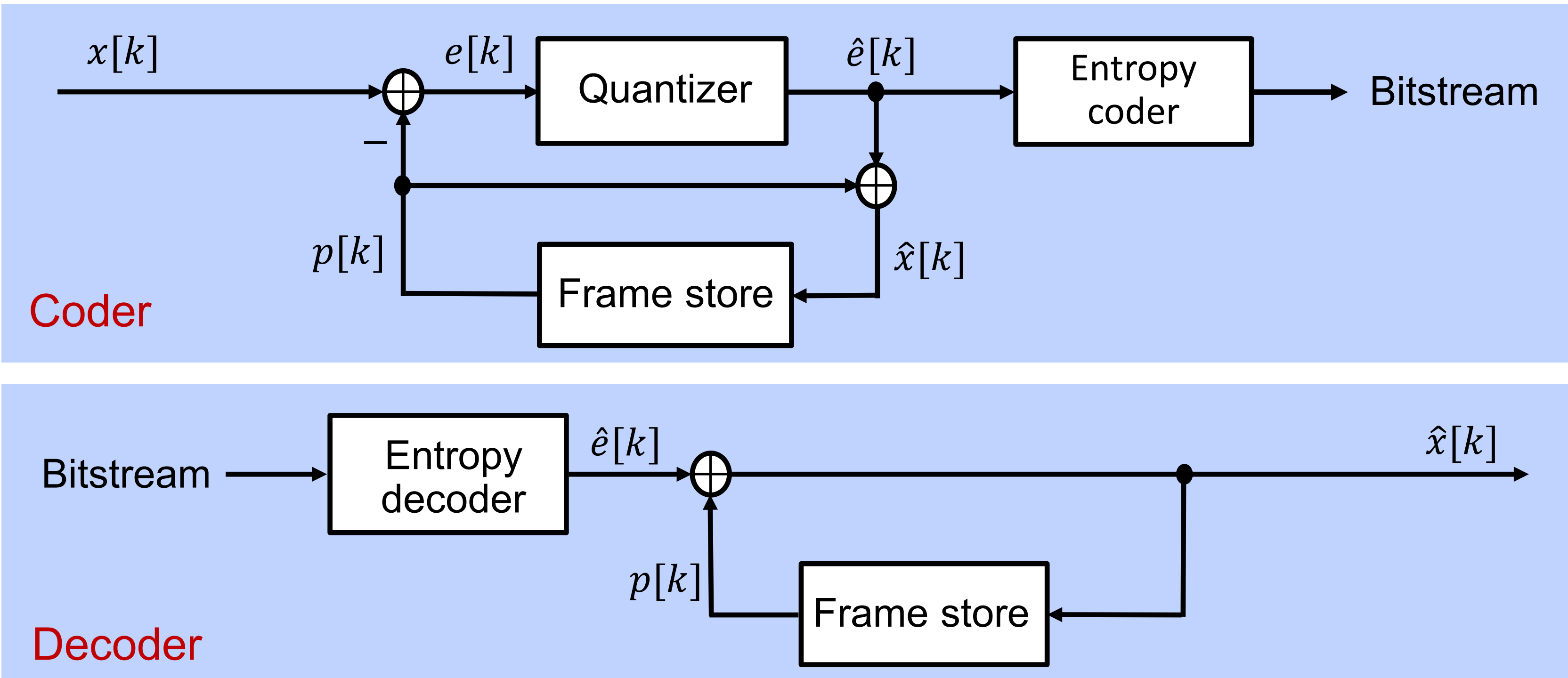
Current frame k

Important methods for interframe coding

- Adaptive intra / interframe coding
- Conditional replenishment
- Motion compensated prediction

Interframe DPCM

Idea: use same DPCM block diagram as in predictive coding, but apply prediction in temporal (instead of spatial) direction using a frame store



Temporal DPCM: each pixel is predicted by reconstructed pixel at same spatial location in previous frame:

$$p[m,n,k] = \hat{x}[m,n,k-1]$$

Adaptive Intra- / Interframe Prediction

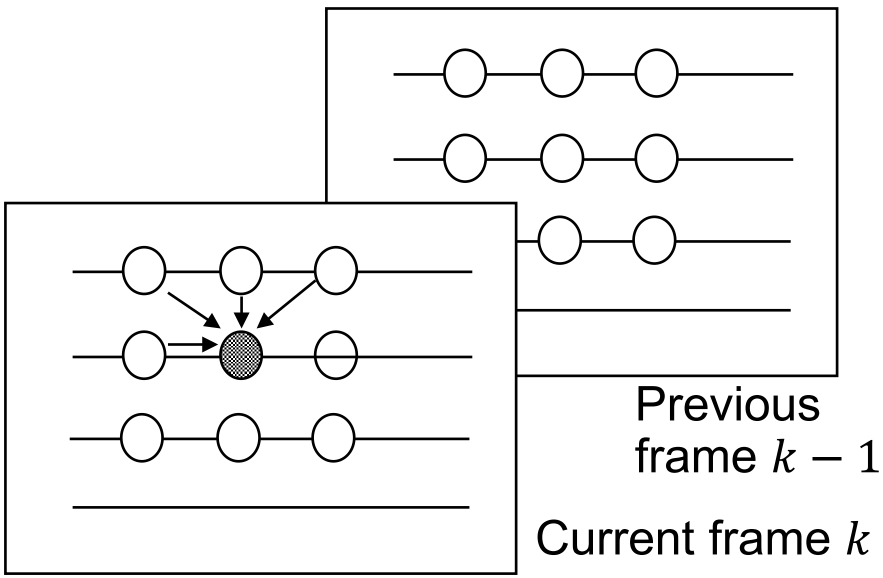
Switch predictor between two possible prediction modes depending on image content:

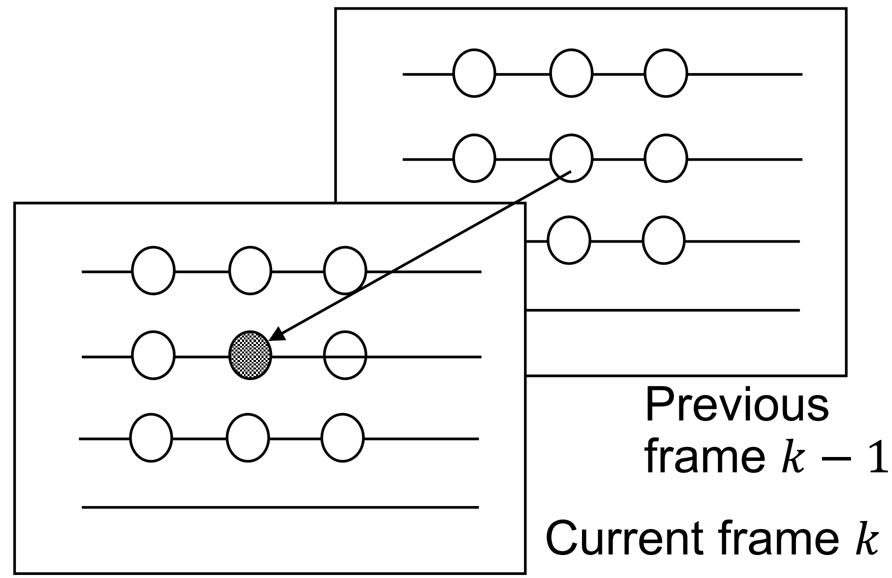
Intraframe prediction

- for moving or changed areas

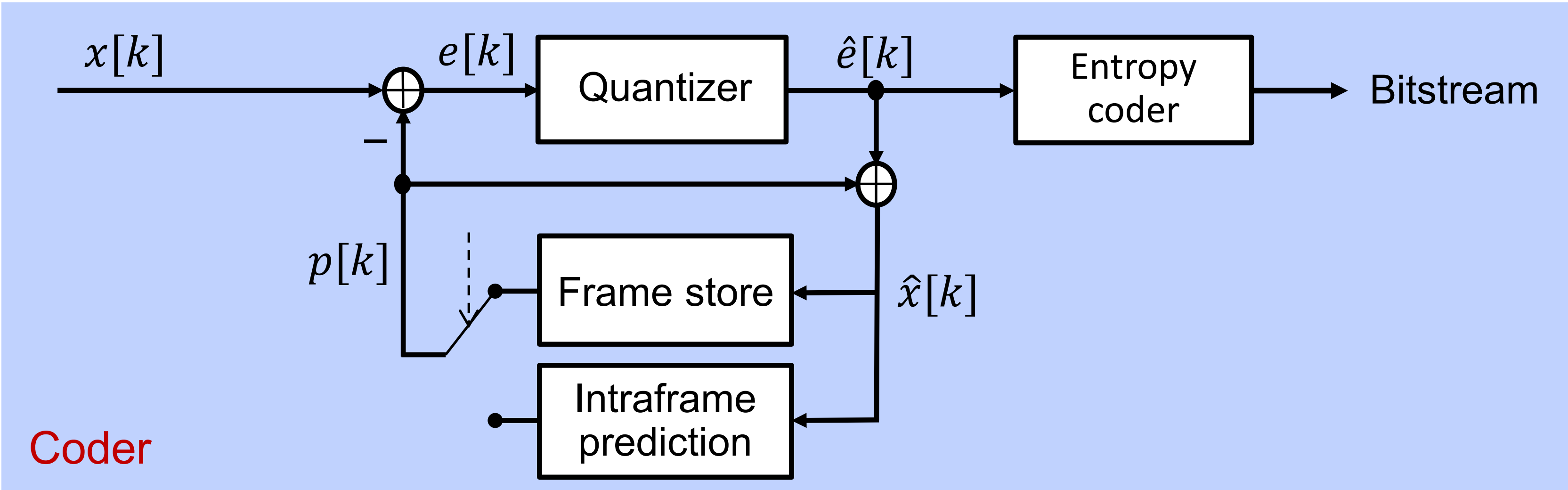
Interframe prediction

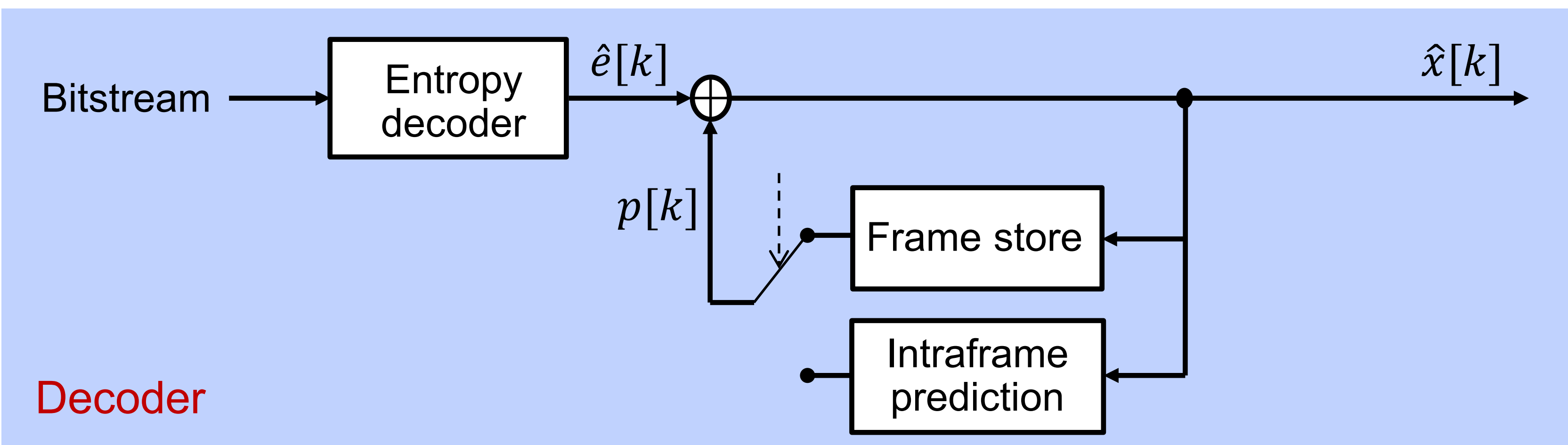
- for still picture areas





Adaptive Intra- / Interframe Prediction (Cont.)





Feed-Forward and Feed-Backward Adaptation

Signaling of switching information necessary in adaptive intra- / interframe prediction

Feed-forward adaptation: switching information is calculated at encoder using image data $x[m,n,k]$

- Advantage: decision takes place on original image data
- Disadvantage: switching state has to be transmitted as side information

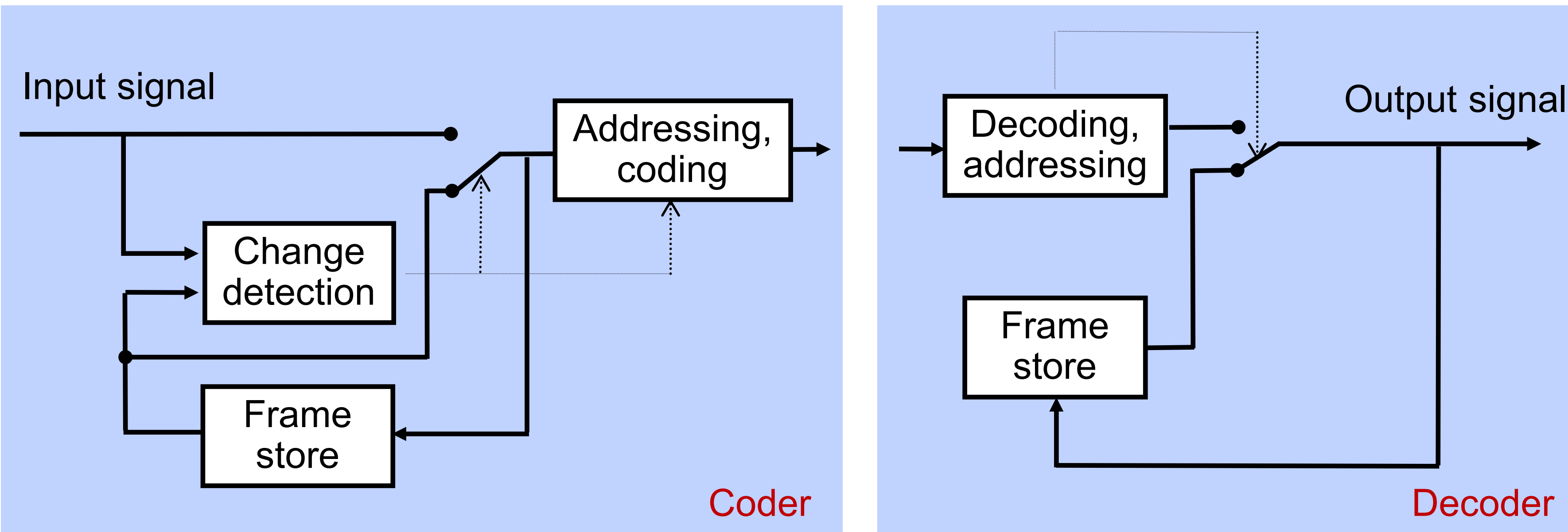
Feed-backward adaptation: switching information is calculated at encoder and decoder using only decoded image data $\hat{x}[m,n,k]$ available at both sides

- Advantage: switching state can be calculated at decoder, no need for transmission as side information
- Disadvantage: decision less reliable since based on quantized and coded image information

Conditional Replenishment Coding

Simple implementation of adaptive intra- / interframe prediction using feed-forward adaptation

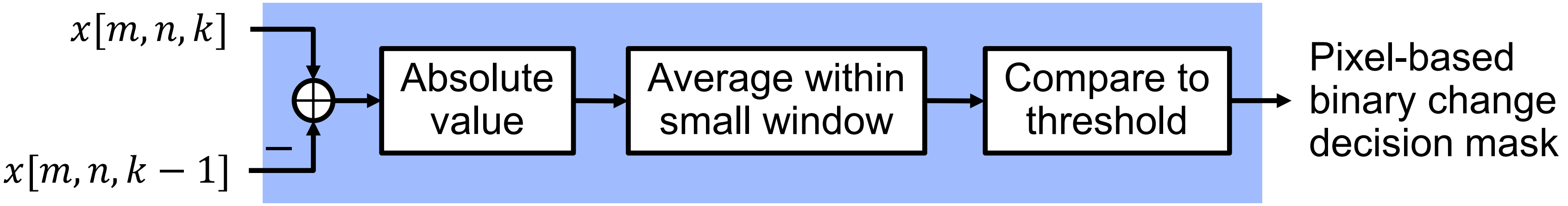
- Still areas: repeat from frame store
- Moving areas: zero prediction, transmit spatial address and code waveform



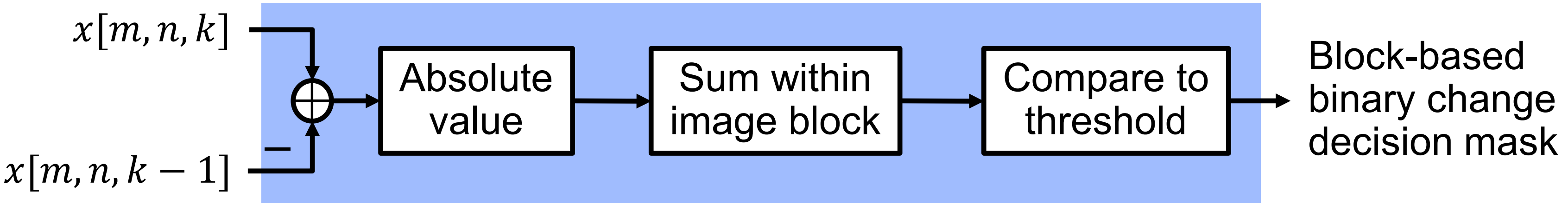
⇨ Used in ITU-T H.120 Standard, released 1984

Change Detection in Video

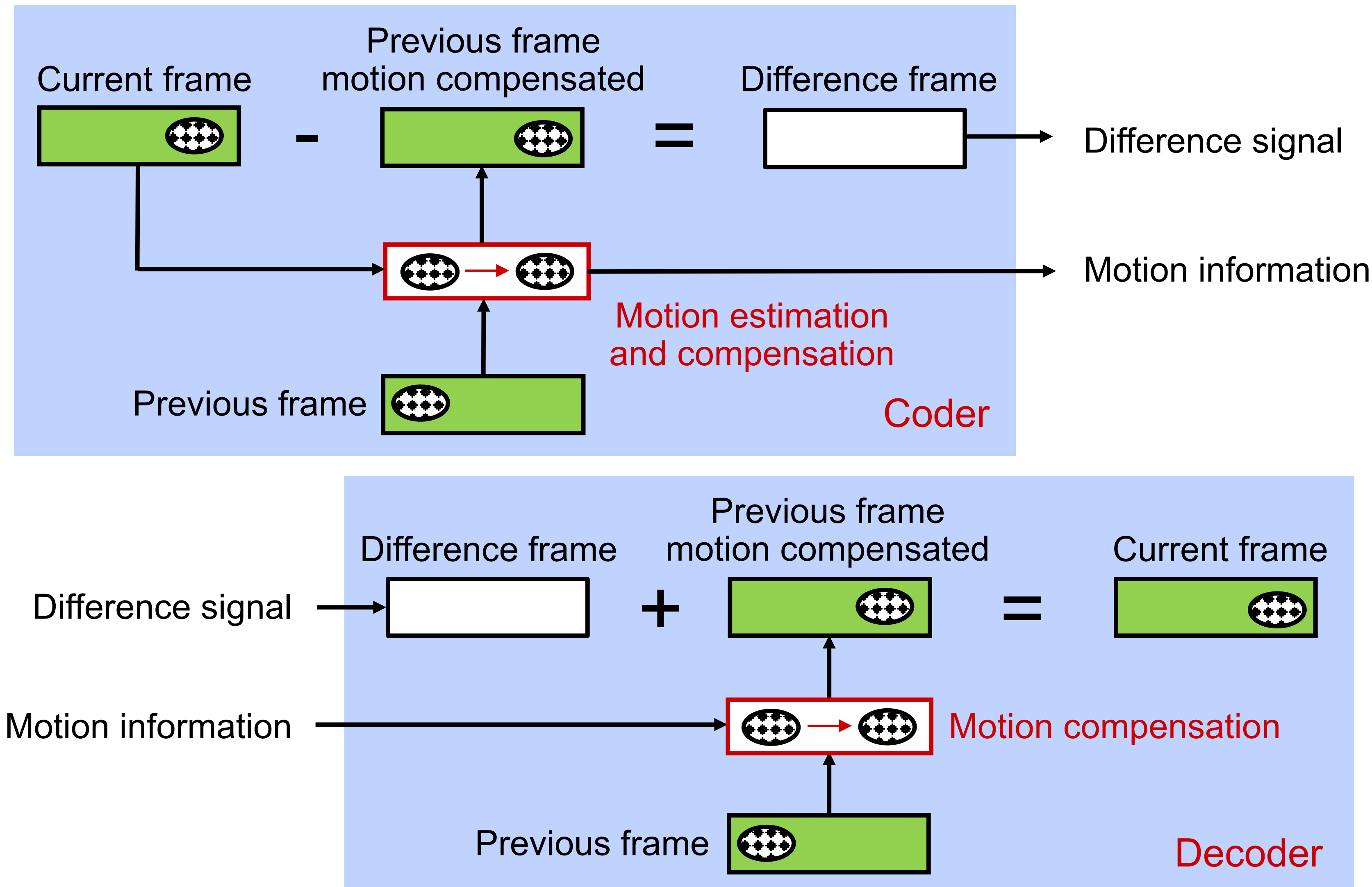
Pixel-oriented change detection by thresholding difference image over a small $n{\times}n$ averaging window (typ. $n = 3$ or 5)



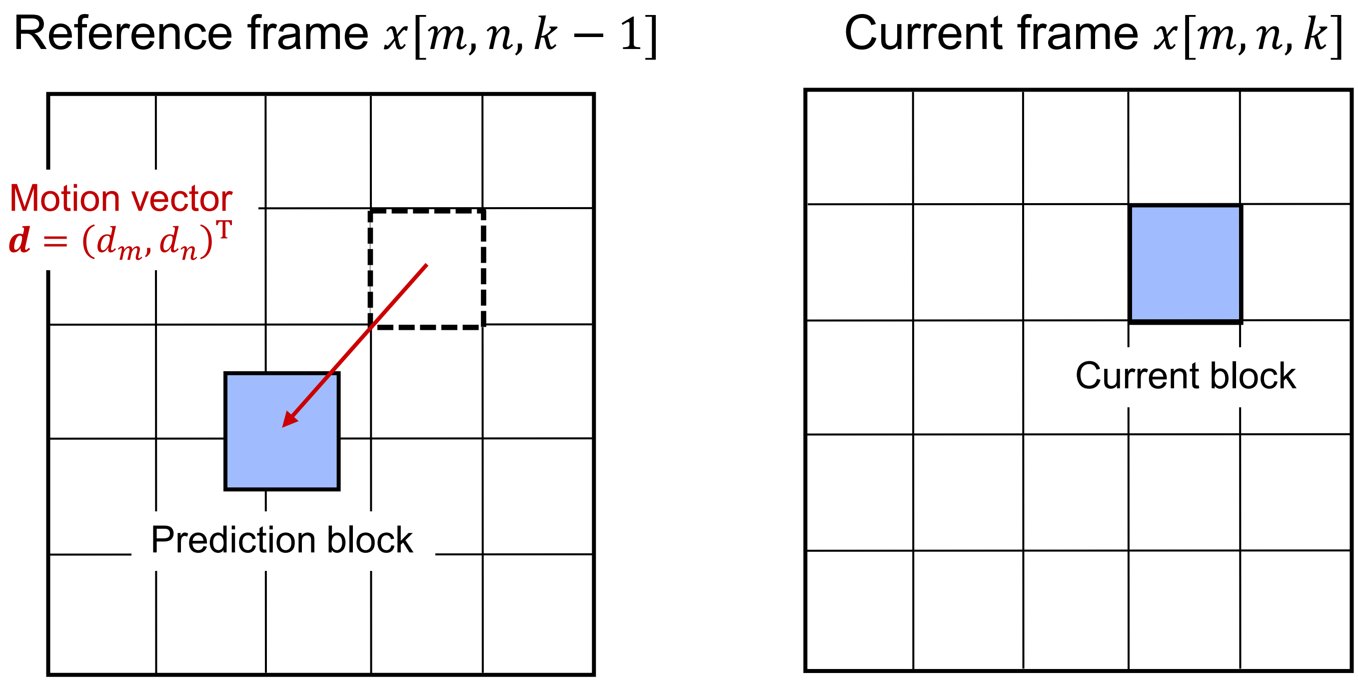
Block-oriented change detection by thresholding sum of differences for $N{\times}N$ blocks (typ. $N = 8$ or 16)



11.2 Motion Compensated Prediction



Block-Based Motion Compensation



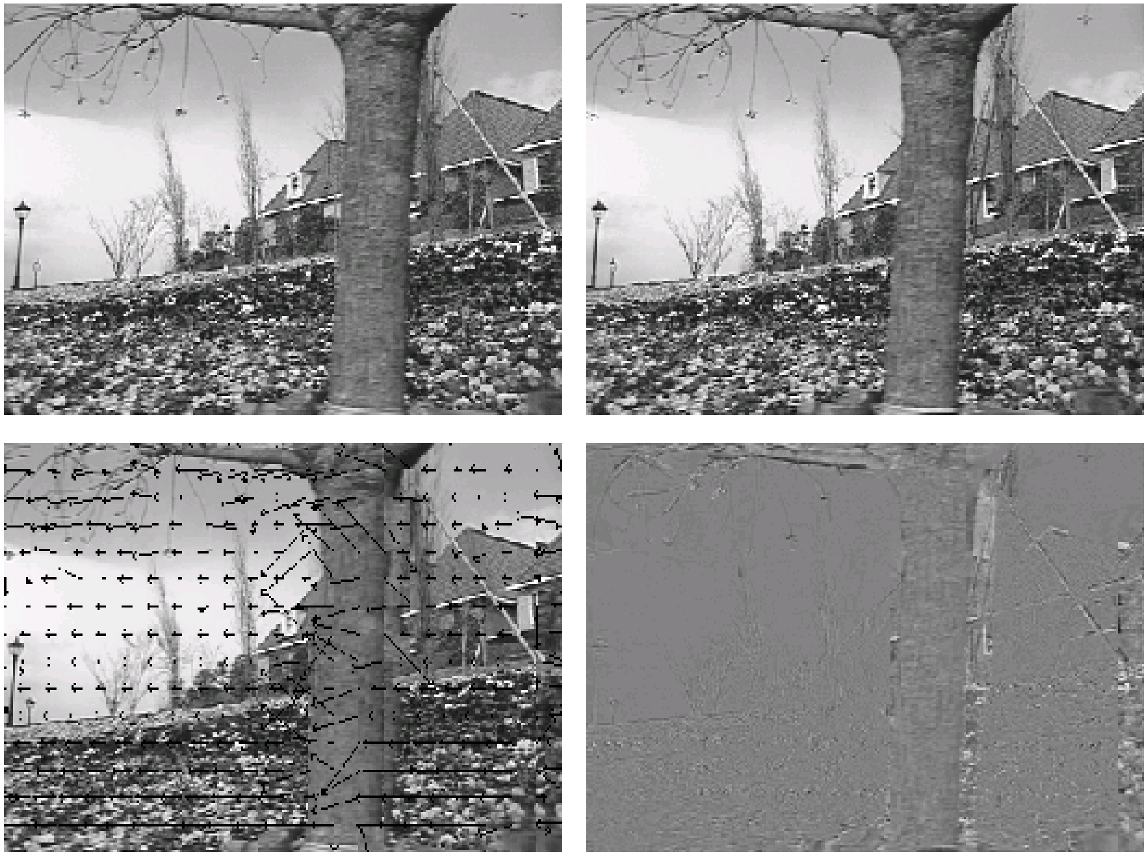
Assumption: motion can be approximately described by translation of small image blocks

- Motion vector $\boldsymbol{d} = (d_m, d_n)^{\mathrm{T}}$ gives the relative position of the prediction block in the reference image to the current block position

Prediction given by $\hat{x}[m,n,k] = x[m + d_m, n + d_n, k-1]$

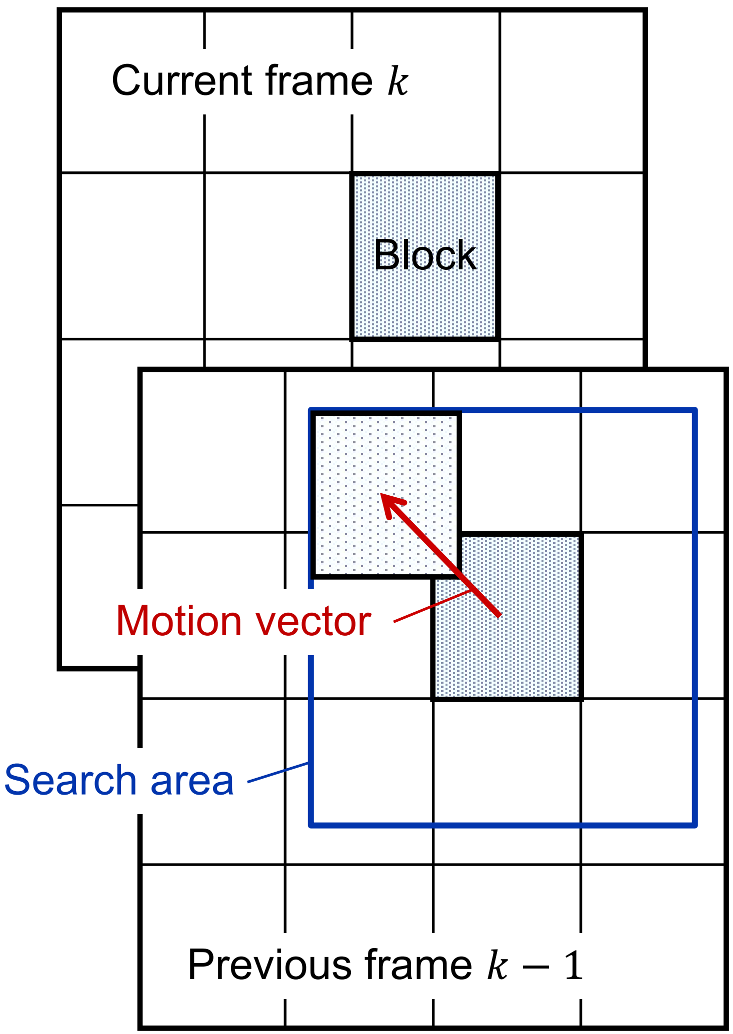
Example for Block-Based Motion Compensation

frame 1 (previous) | frame 2 (current)



frame 2 with displacement vectors

difference between motion compensated prediction and current frame



Block matching: simple method to estimate motion, widely used in video compression

Principle

- Subdivide current frame into blocks
- Estimate one motion vector for each block
- Find best match minimizing an error measure by testing all possible search positions within search area

Complexity proportional to number of search positions n_s

- $M \times N$ image, block size $B \times B$, search range +/- $d_{\max}$

$$n_s = (2d_{\max} + 1)^2 \left\lceil \frac{M}{B} \right\rceil \left\lceil \frac{N}{B} \right\rceil$$

⇨ fast methods required

Matching Criteria for Block Matching

Sum of squared differences (SSD)

- Sum up squared differences between current image and previous image for block of size $N \times N$

$$\mathrm{SSD}(d_m, d_n) = \sum_{m,n \in \mathrm{Block}} (x[m,n,k] - x[m+d_m, n+d_n, k-1])^2$$

- Corresponds to mean square error
- Requires multiplications, usually avoided

Sum of absolute differences (SAD)

- Sum up absolute differences between current and previous image for block of size $N \times N$

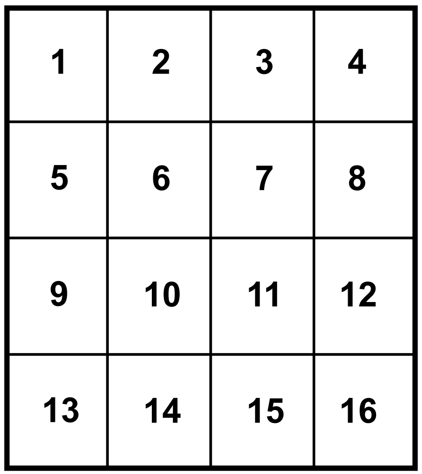
$$\mathrm{SAD}(d_m, d_n) = \sum_{m,n \in \mathrm{Block}} |x[m,n,k] - x[m+d_m, n+d_n, k-1]|$$

- Computationally less expensive, similar performance than SSD

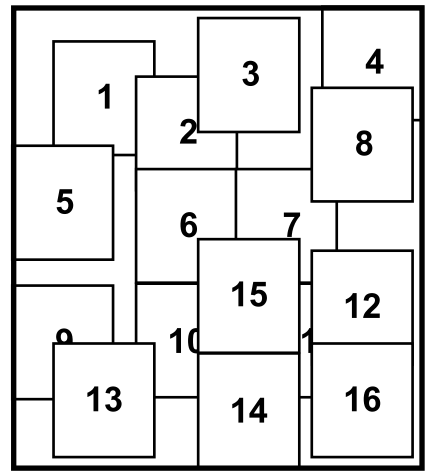
Block Overlap in Reference Frame

Displacement estimation takes place for every block in the current frame using a regular grid decomposition, e.g. into 8×8 or 16×16 blocks

⇨ Blocks used for prediction in reference image may overlap and must not cover whole image



Blocks in current frame
$x[m,n,k]$



Possible overlap of blocks
in reference frame $x[m,n,k-1]$

Demo 11 „Block Matching“

Full Search Block Matching

Principle

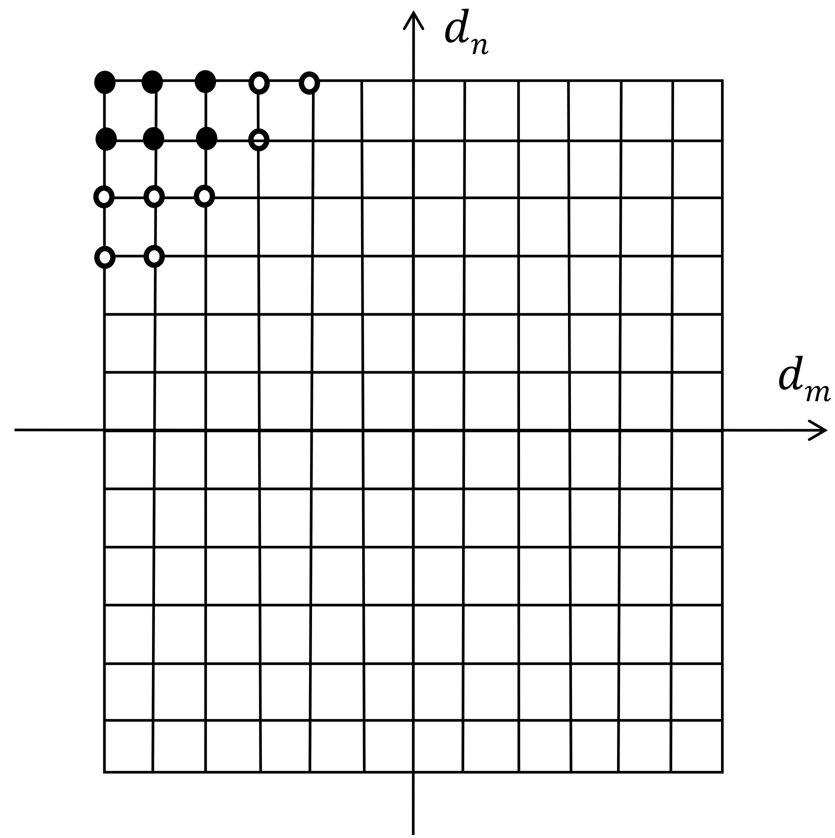
- Define maximum search range
- Calculate error measure for all possible displacement vectors within search range

Advantages

- Regular, fully parallelizable
- Well suited for hardware implementation

Disadvantage

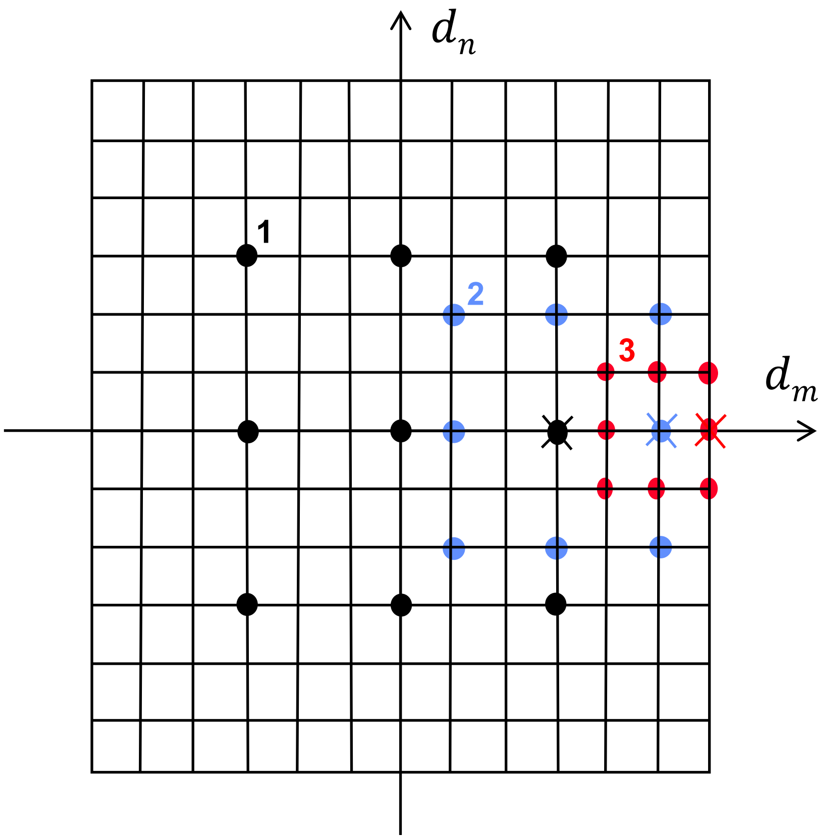
- Computationally expensive



Three Step Search

Principle

- Compare error measure values for nine neighboring points (“1”) with distance of three pixels
- Compare error measure values for nine neighboring point (“2”) around previous best match using distance of two pixels
- Repeat with distance of one pixel (“3”)



Complexity for search range +/- 6 pixels

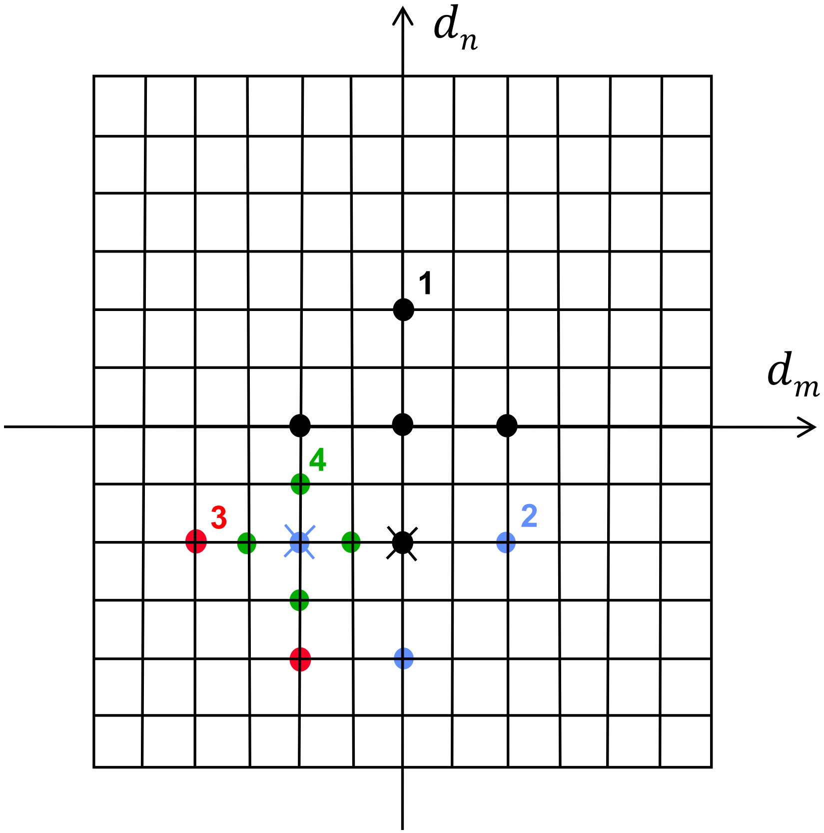
- $2\times8+9$ = 25 search positions
- Full search requires 169 positions

⇨ Extendible to larger search areas

2D Logarithmic Search

Principle

- Define maximum search range
- Compare error measure values for five hexagonally neighboring points
- If best match is not center position (“1”, “2”) shift search diamond to position of best match
- Reduce size of diamond if best match is center position (“3”) or arrives at boundary of search range

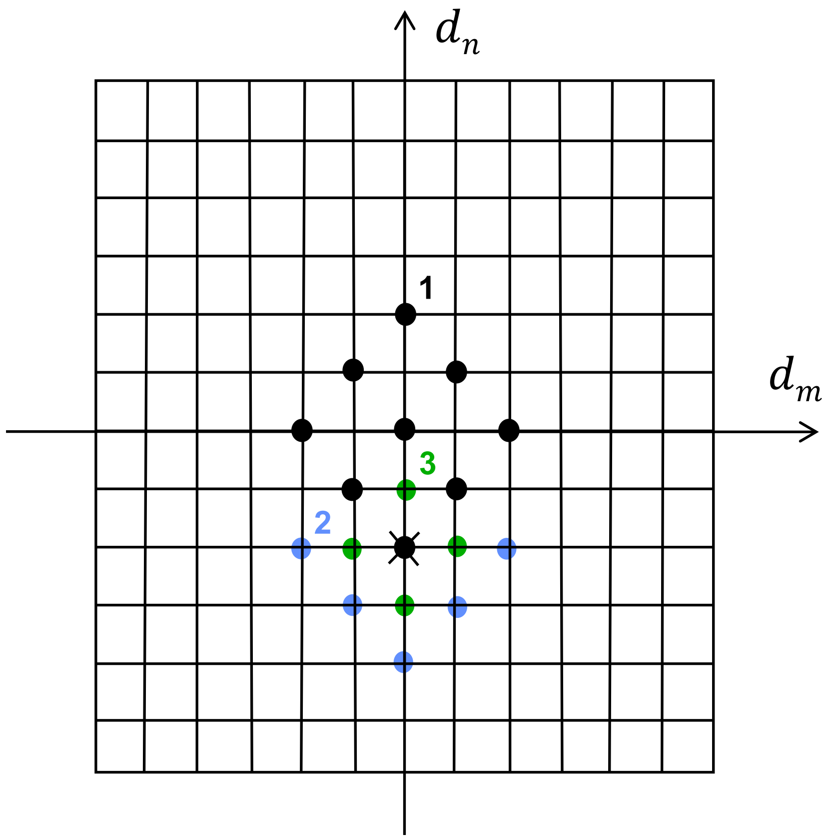


Advantage: automatically adapts search range by following error gradient

Diamond Search

Principle

- Define maximum search range
- Compare error measure values for all nine diamond samples (black, “1”)
- If best match not at center position, shift large diamond to position of best match (“2”, blue)
- If best match is at center position of large diamond or touches border of search range, proceed with smaller diamond (“3”, green)



Advantage: extends 2D logarithmic search by diagonal shifts

Predictive Search and Early Termination

Predictive search to speed-up motion vector estimation

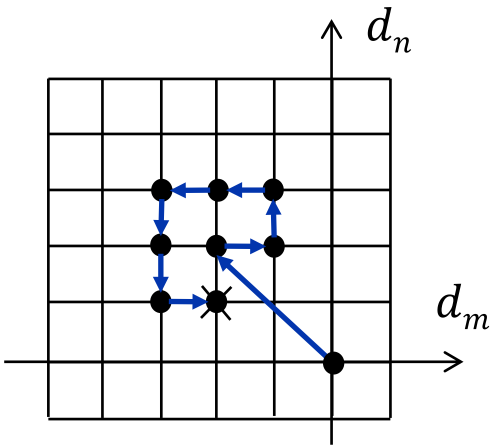
- Set starting point for motion search using suitable prediction, e.g. median of already estimated motion vectors in causal neighborhood
- Additionally test zero motion vector as starting point

Early termination of search algorithm

- Stop summation to calculate error measure (SSD or SAD) if current sum already exceeds error value of previous best match
- Stop search if current match is good enough, e.g. error measure (SSD or SAD) smaller than predefined threshold

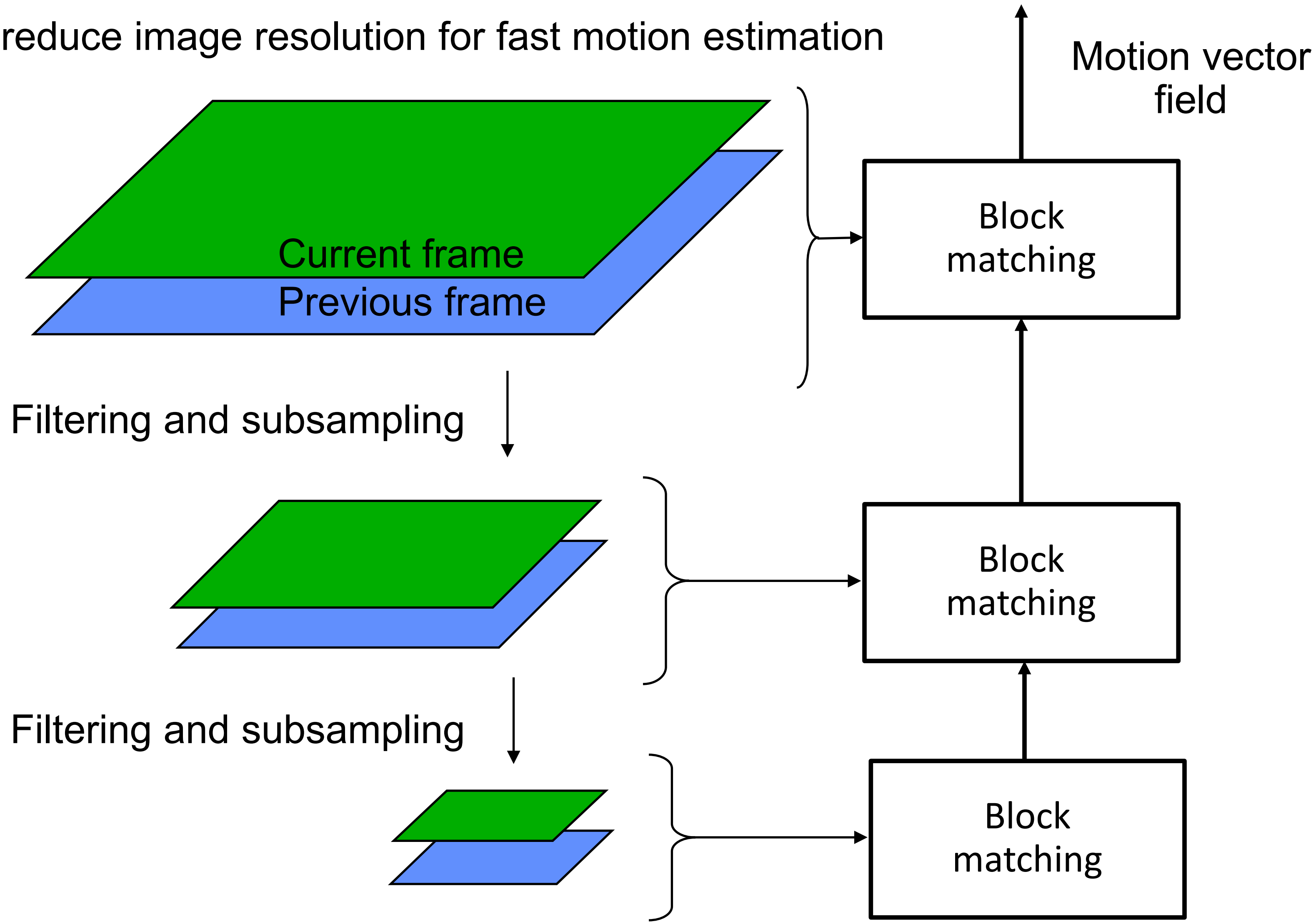
Example: spiral search

- Start with good prediction
- Test further search positions around stating point in spiral fashion as long as error measure decreases



Hierarchical Block Matching

Idea: reduce image resolution for fast motion estimation



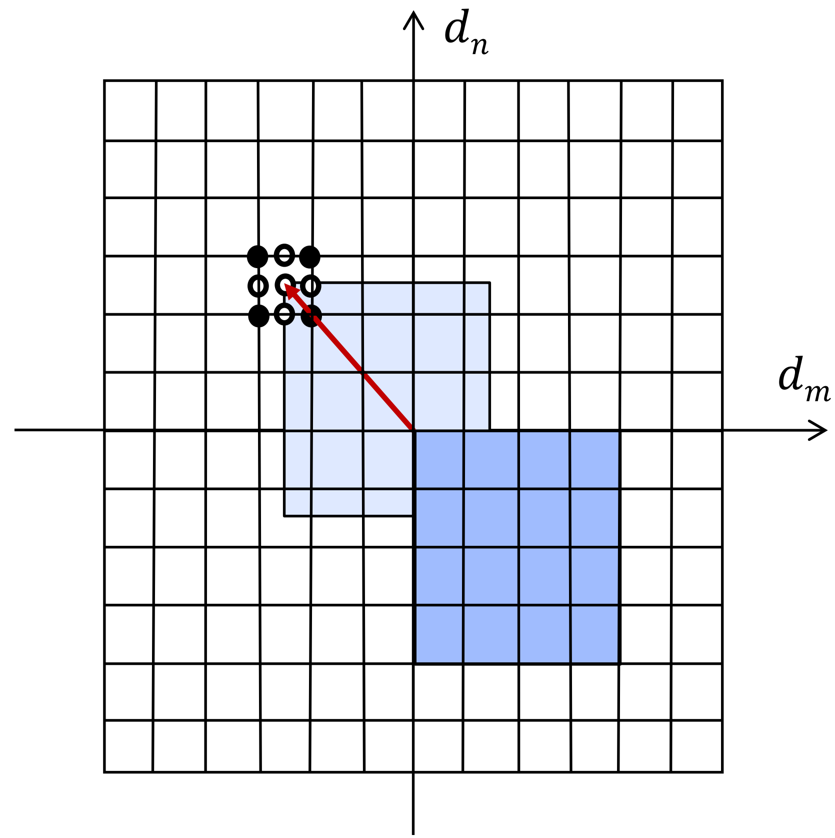
Sub-Pixel Accuracy

Prediction accuracy can be improved by using fractional pixel motion vectors

- Interpolate pixel raster of reference image to desired sub-pixel accuracy
- Extend motion vector search to sub-pixel positions
- Typical motion vector resolution: half-pixel or quarter-pixel accuracy

Full-pixel position ●

Half-pixel position ○

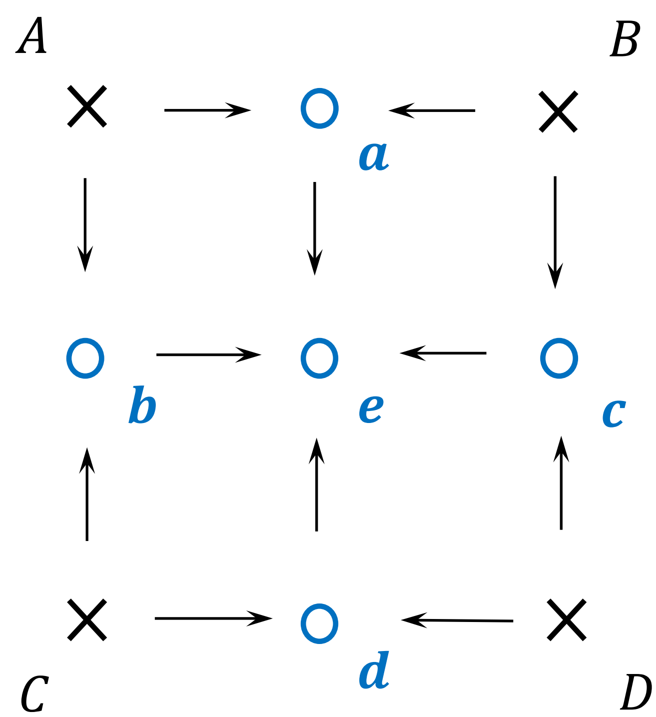


Example: Half-pixel resolution

Bilinear Interpolation

Sub-pixel motion vectors require interpolation of previous image $k-1$

⇨ Typical choice: 2D bilinear interpolation, especially by half-pixel accuracy



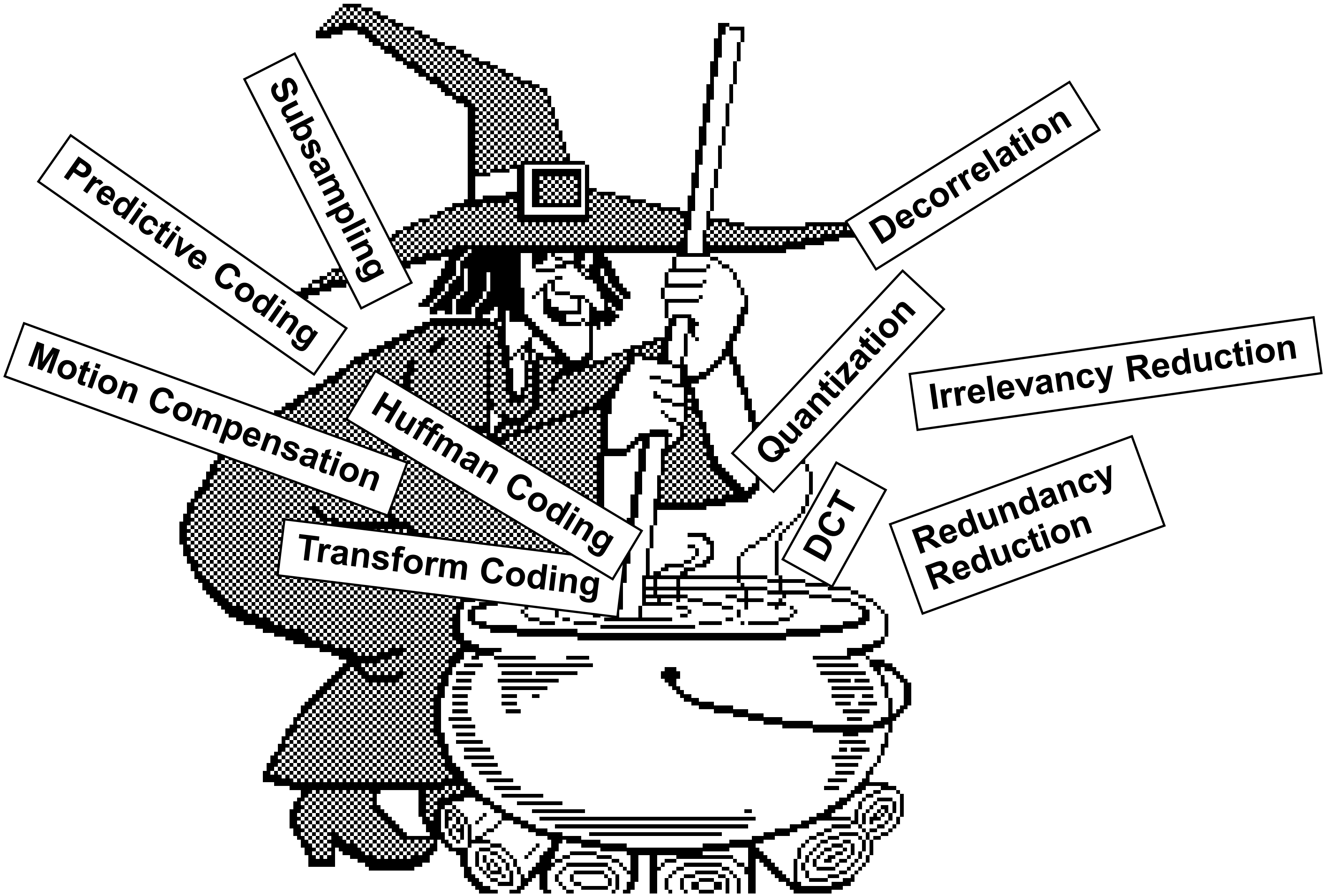
× Integer pixel position

○ Half-pixel position

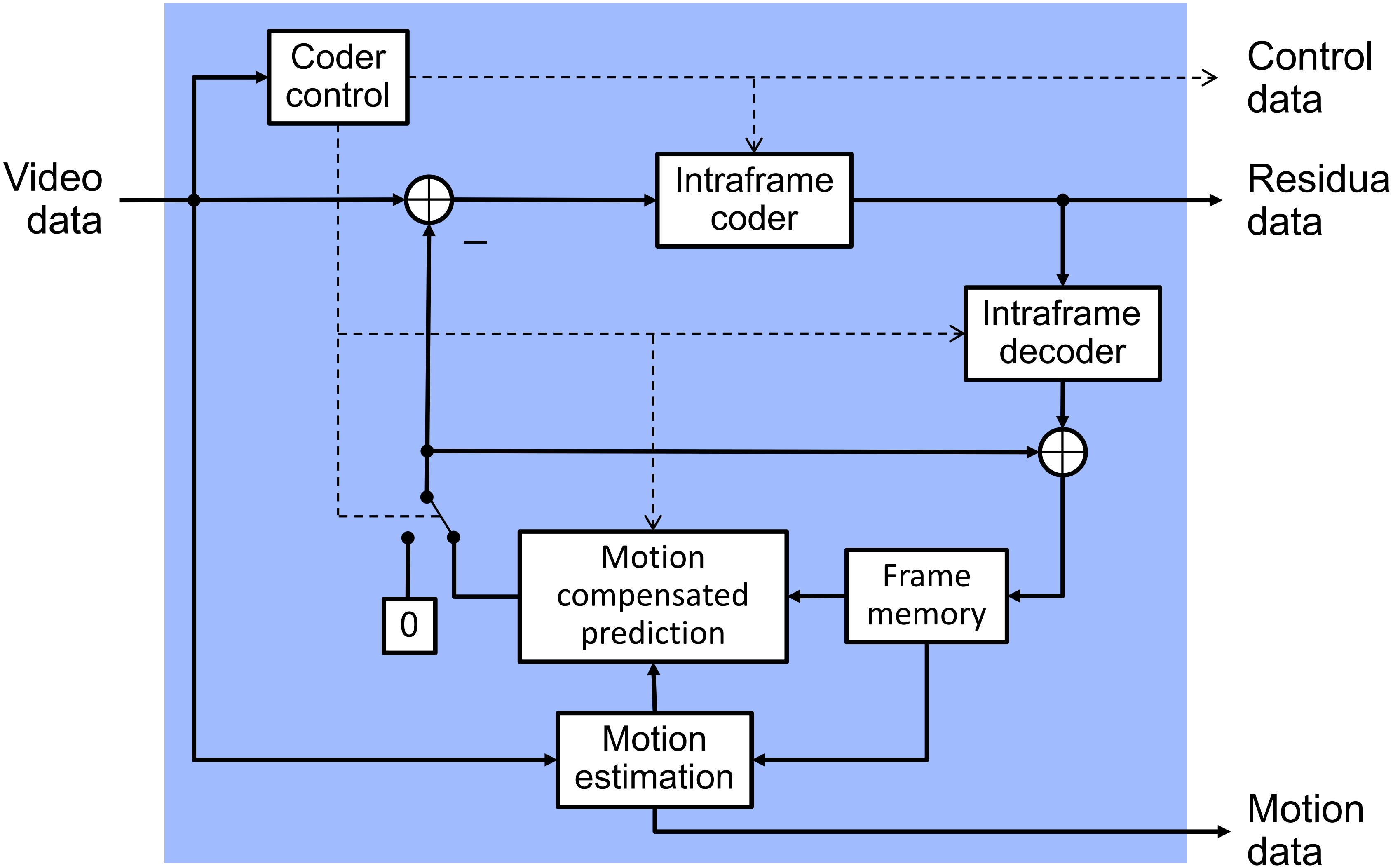
$$
\begin{aligned}
a &= (A+B+1) >> 1\\
b &= (A+C+1) >> 1\\
c &= (B+D+1) >> 1\\
d &= (C+D+1) >> 1\\
e &= (A+B+C+D+2) >> 2
\end{aligned}
$$

Can be generalized to any sub-pixel position, e.g. quarter-pixel resolution

11.4 Magic Coding Recipe



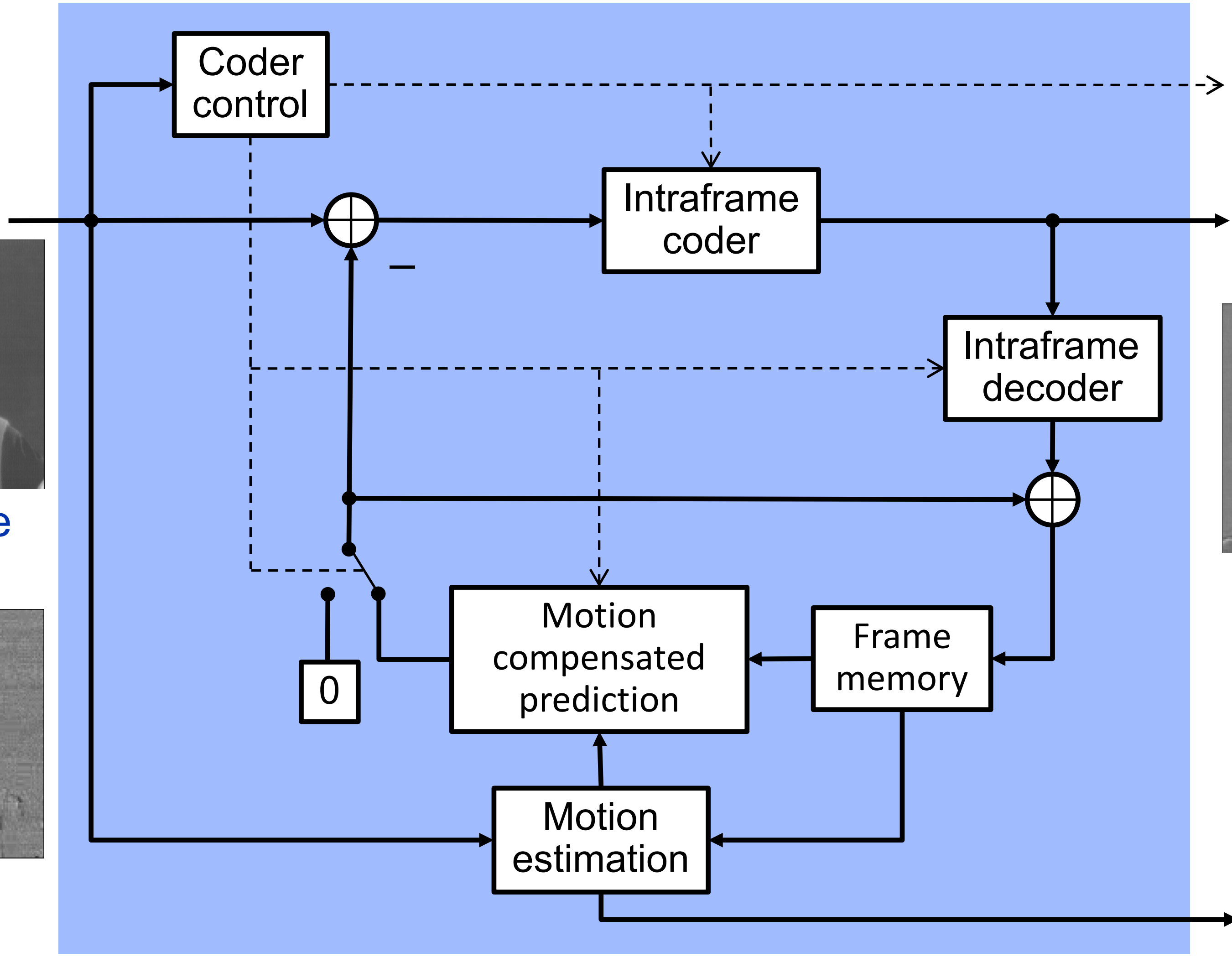
Motion Compensated Hybrid Coder



Motion Compensated Hybrid Coder (Cont.)

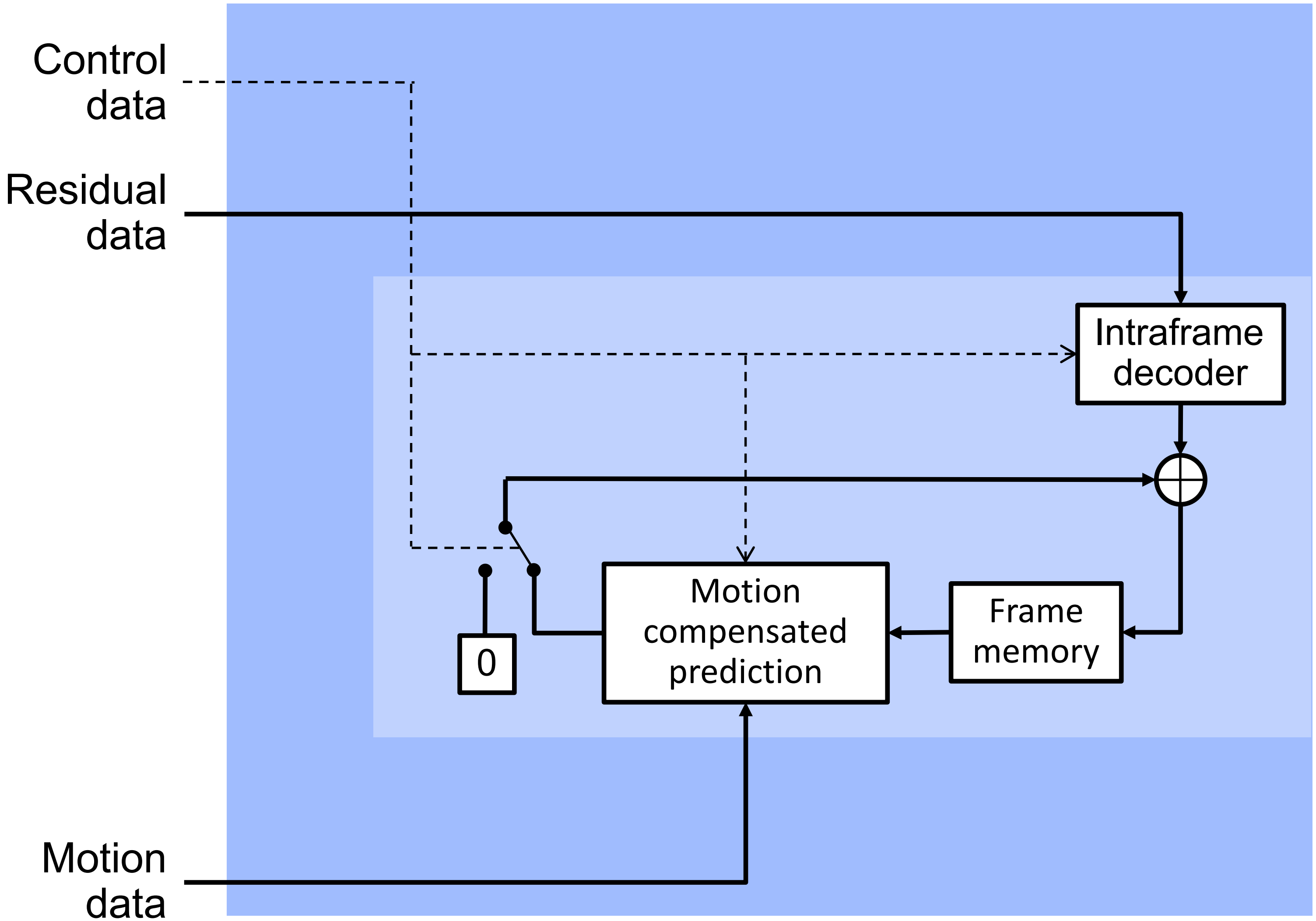
Input image

Prediction error



Coding error

Motion Compensated Hybrid Decoder



History of Motion Compensated Coding

Conditional replenishment
- *H.120 [1984]*

Frame difference coding
- *H.120 Version 2 [1988]*

Motion compensation with full-pixel accuracy
- *H.261 [1991]*

Half-pixel accurate motion compensation
- *MPEG-1 [1993], MPEG-2/H.262 [1994]*

Variable block-size motion compensation
- *H.263 [1996], MPEG-4 [1999]*

Motion compensation with quarter-pixel accuracy
- *H.264/AVC [2003], H.265/HEVC [2013], H.266/VVC [2020]*

Interframe Coding - Summary

- Interframe coding exploits similarity of consecutive images
- Motion is taken into account by block-based displacement compensation
- Block matching finds best match by searching candidate positions
- Sub-pixel accuracy in motion compensation improves prediction
- Speed up of block matching by fast search methods
- Hybrid coding combines motion compensation and prediction error coding