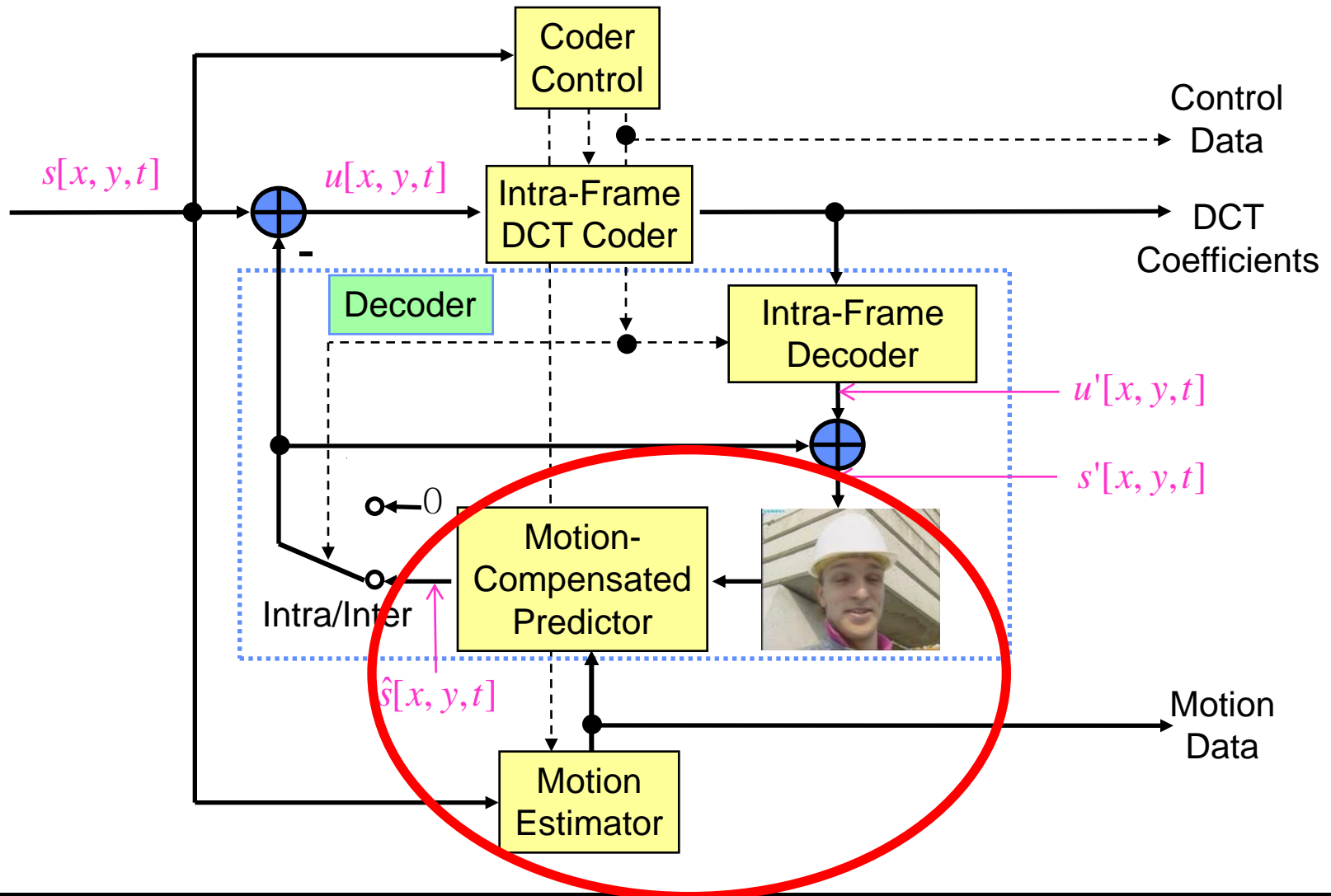
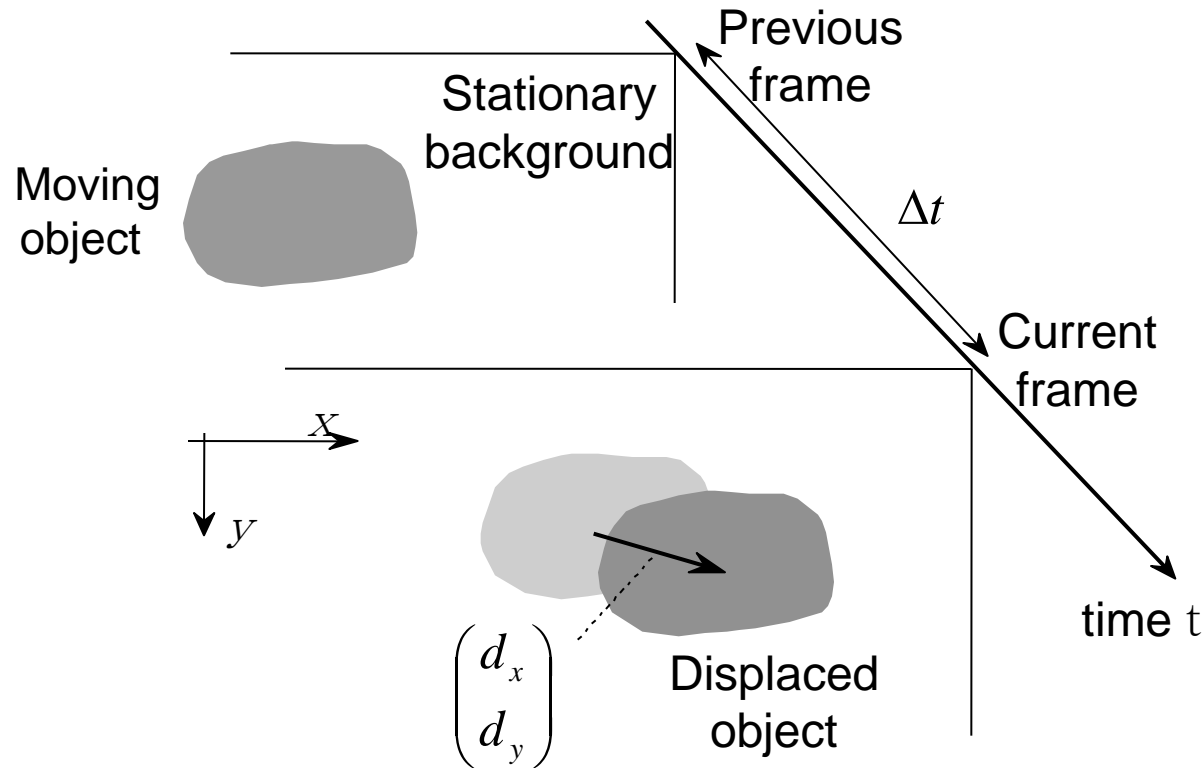

Motion Estimation for Video Coding

- Motion-Compensated Prediction
- Bit Allocation
- Motion Models
- Motion Estimation
- Efficiency of Motion Compensation Techniques

Hybrid Video Encoder



Motion-Compensated Prediction



Prediction for the luminance signal $s[x, y, t]$ within the moving object:

$$\hat{s}[x, y, t] = s'(x - d_x, y - d_y, t - \Delta t)$$

Motion-Compensated Prediction: Example

Frame 1 $s[x,y,t-1]$ (previous)



Frame 2 $s[x,y,t]$ (current)

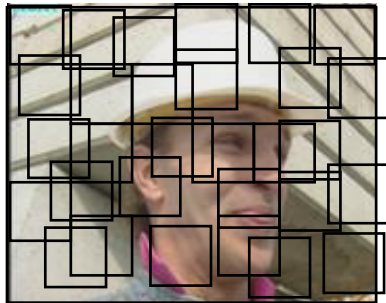


Partition of frame 2 into blocks (schematic)

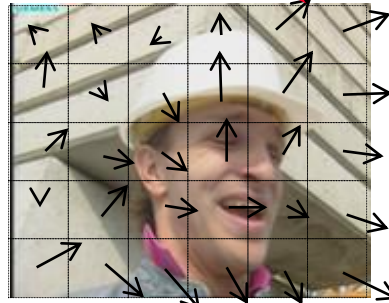


Size of Blocks

Accuracy of Motion Vectors



Referenced blocks in frame 1



Frame 2 with displacement vectors



Difference between motion-compensated prediction and current frame $u[x,y,t]$

Motion Models

- Motion in 3-D space corresponds to displacements in the image plane
- Motion compensation in the image plane is conducted to provide a prediction signal for efficient video compression
- Efficient motion-compensated prediction often uses side information to transmit the displacements
- Displacements must be efficiently represented for video compression
- Motion models relate 3-D motion to displacements assuming reasonable restrictions of the motion and objects in the 3-D world

Motion Model

$$d_x = x' - x = f_x(\mathbf{a}, x, y), \quad d_y = y' - y = f_y(\mathbf{b}, x, y)$$

x, y : location in previous image

x', y' : location in current image

\mathbf{a}, \mathbf{b} : vector of motion coefficients

d_x, d_y : displacements

Representation of Video Signal

Decoded video signal is given as

$$s'[x, y, t] = \hat{s}[x, y, t] + u'[x, y, t]$$

Motion-compensated
prediction signal

$$\hat{s}[x, y, t] = s'(x - d_x, y - d_y, t - \Delta t)$$

$$d_x = \sum_{i=0}^{N-1} a_i \cdot \phi_i(x, y), \quad d_y = \sum_{i=0}^{N-1} b_i \cdot \phi_i(x, y)$$

Transmitted motion parameters

$$R_m = f(\mathbf{a}, \mathbf{b}), \quad \mathbf{a} = (a_0, \dots), \quad \mathbf{b} = (b_0, \dots)$$

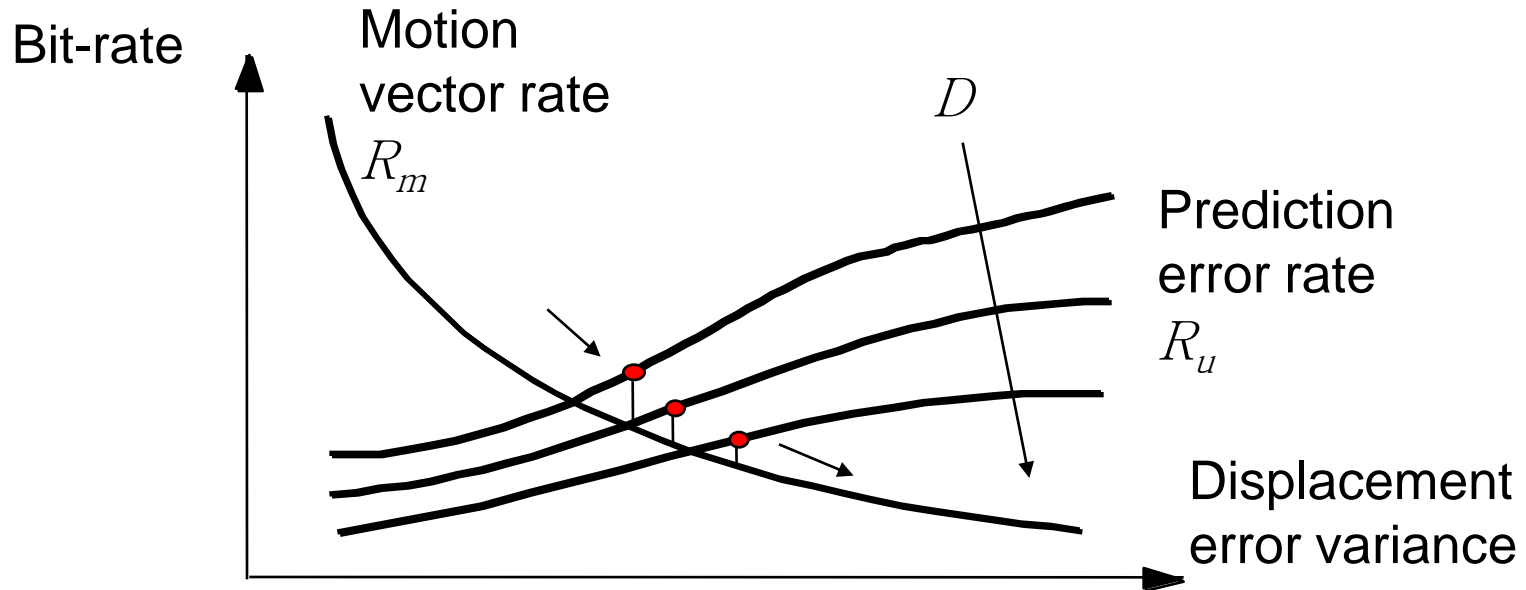
Prediction residual signal

$$u'(x, y) = \sum_{j=0}^{M-1} c_j \cdot \phi_j(x, y)$$

Transmitted residual
parameters

$$R_u = f(\mathbf{c}), \quad \mathbf{c} = (c_0, \dots)$$

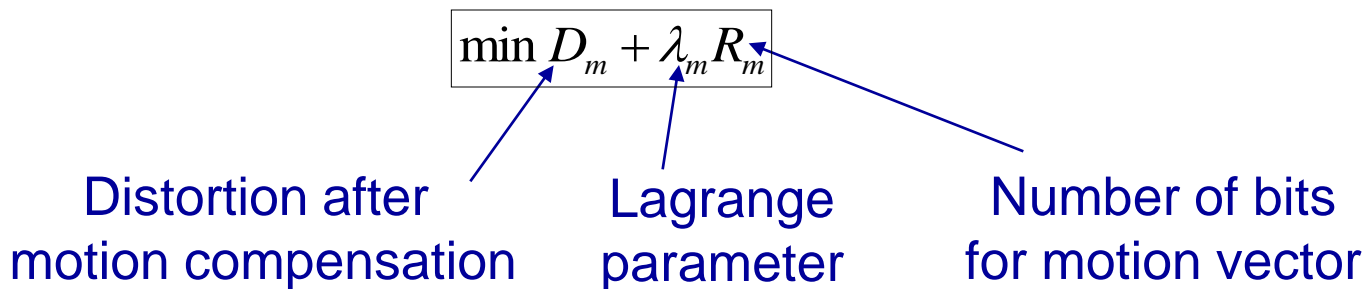
Rate-Constrained Motion Estimation



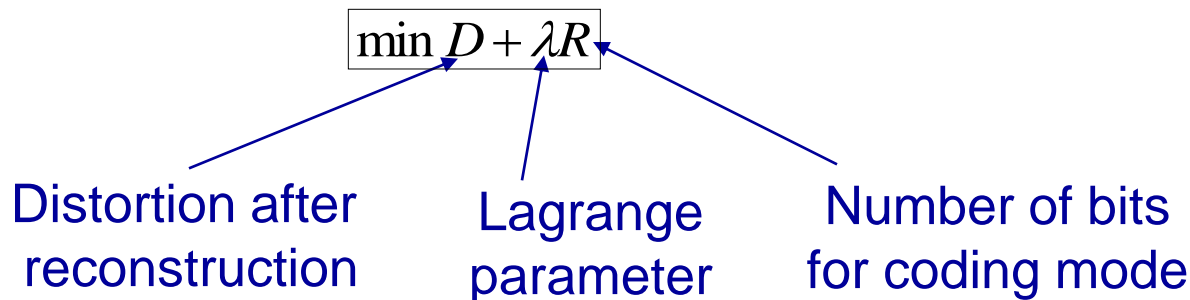
- Optimum trade-off: $\frac{dD}{dR_m} = \frac{dD}{dR_u}, \quad R = R_m + R_u$
- Displacement error variance can be influenced via
 - Block-size, quantization of motion parameters
 - Choice of motion model

Lagrangian Optimization in Video Coding

- A number of interactions are often neglected
 - Temporal dependency due to DPCM loop
 - Spatial dependency of coding decisions
 - Conditional entropy coding
- *Rate-Constrained Motion Estimation* [Sullivan, Baker 1991]:



- *Rate-Constrained Mode Decision* [Wiegand, et al. 1996]:

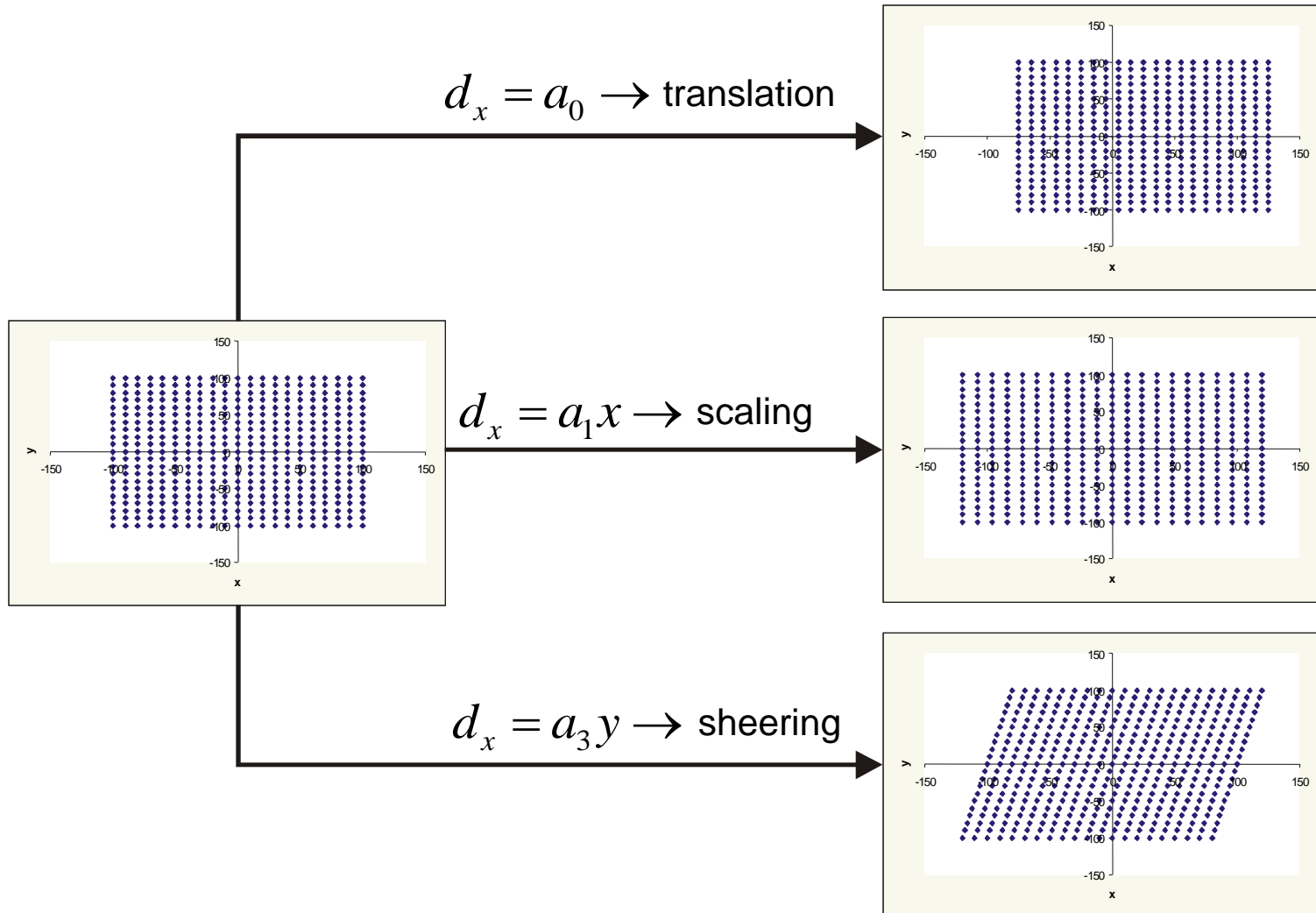


Motion Models

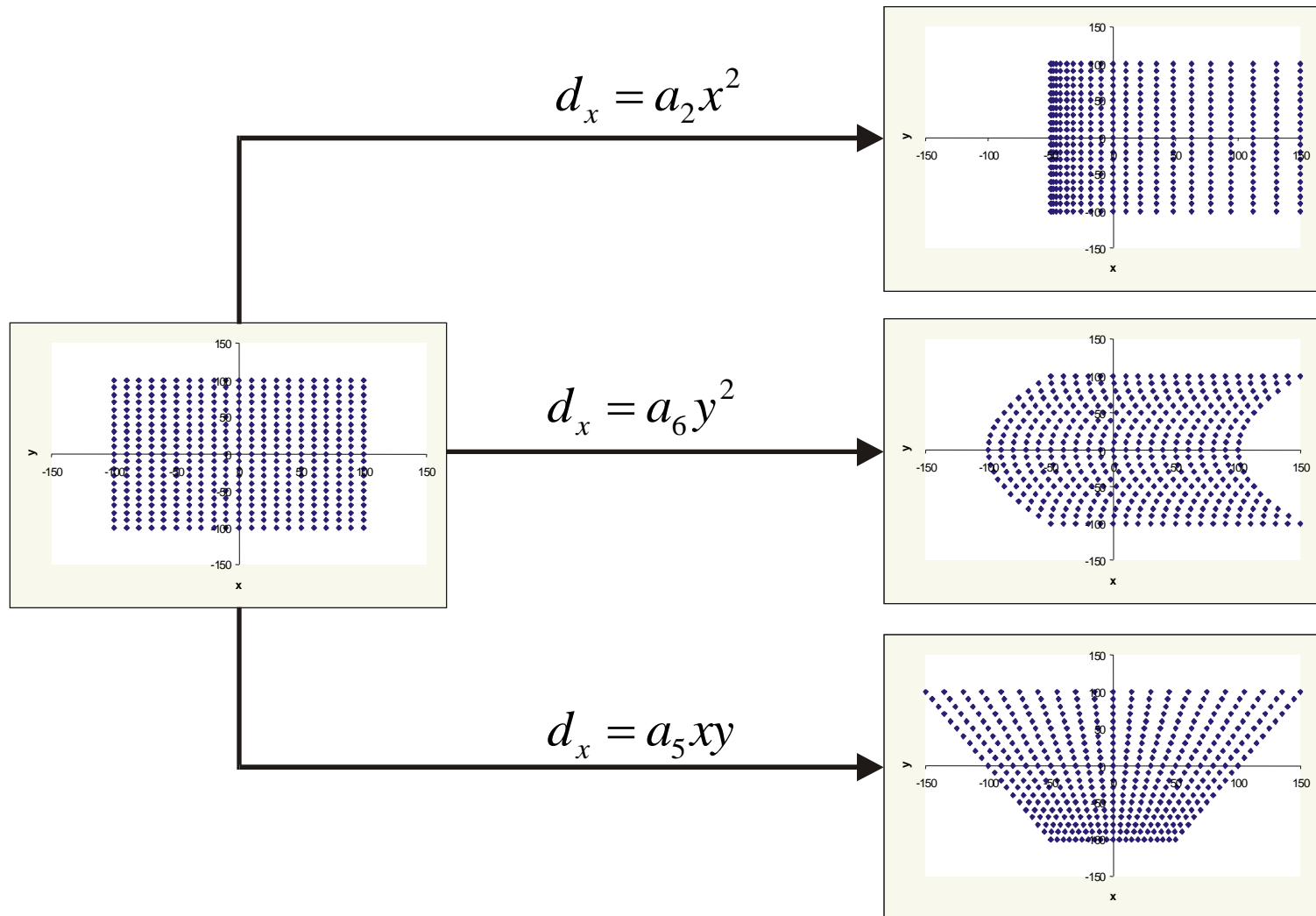
$$d_x = \sum_{i=0}^{N-1} a_i \cdot \varphi_i(x, y), \quad d_y = \sum_{i=0}^{N-1} b_i \cdot \varphi_i(x, y)$$

- Translational motion model $d_x = a_0, d_y = b_0$
- 4-Parameter motion model: translation, zoom (isotropic Scaling), rotation in image plane
$$d_x = a_0 + a_1 x + a_2 y$$
$$d_y = b_0 + a_2 x + a_1 y$$
- Affine motion model:
$$d_x = a_0 + a_1 x + a_2 y$$
$$d_y = b_0 + b_1 x + b_2 y$$
- Parabolic motion model
$$d_x = a_0 + a_1 x + a_3 y + a_2 x^2 + a_6 y^2 + a_5 xy$$
$$d_y = b_0 + b_1 x + b_3 y + b_2 x^2 + b_6 y^2 + b_5 xy$$

Impact of the Affine Parameters



Impact of the Parabolic Parameters



Differential Motion Estimation

- Assume small displacements d_x, d_y :

$$u(x, y, t) = s(x, y, t) - \hat{s}(x, y, t, d_x, d_y)$$
$$\approx s(x, y, t) - s'(x, y, t - \Delta t) - \frac{\partial s'(x, y, t - \Delta t)}{\partial x} \cdot d_x - \frac{\partial s'(x, y, t - \Delta t)}{\partial y} \cdot d_y$$

Displace frame difference

Horizontal and vertical
gradient of image signal S

- Aperture problem: several observations required
- Inaccurate for displacements > 0.5 pel
→ multigrid methods, iteration
- Minimize

$$\min \sum_{y=1}^{By} \sum_{x=1}^{Bx} u^2(x, y, t)$$

Gradient-Based Affine Refinement

- Displacement vector field is represented as

$$x' = a_1 + a_2x + a_3y$$

- Combination

$$y' = b_1 + b_2x + b_3y$$

$$u(x, y, t) \approx s(x, y, t) - s'(x, y, t - \Delta t) + \frac{\partial s'}{\partial x} (a_1 + a_2x + a_3y) + \frac{\partial s'}{\partial y} (b_1 + b_2x + b_3y)$$

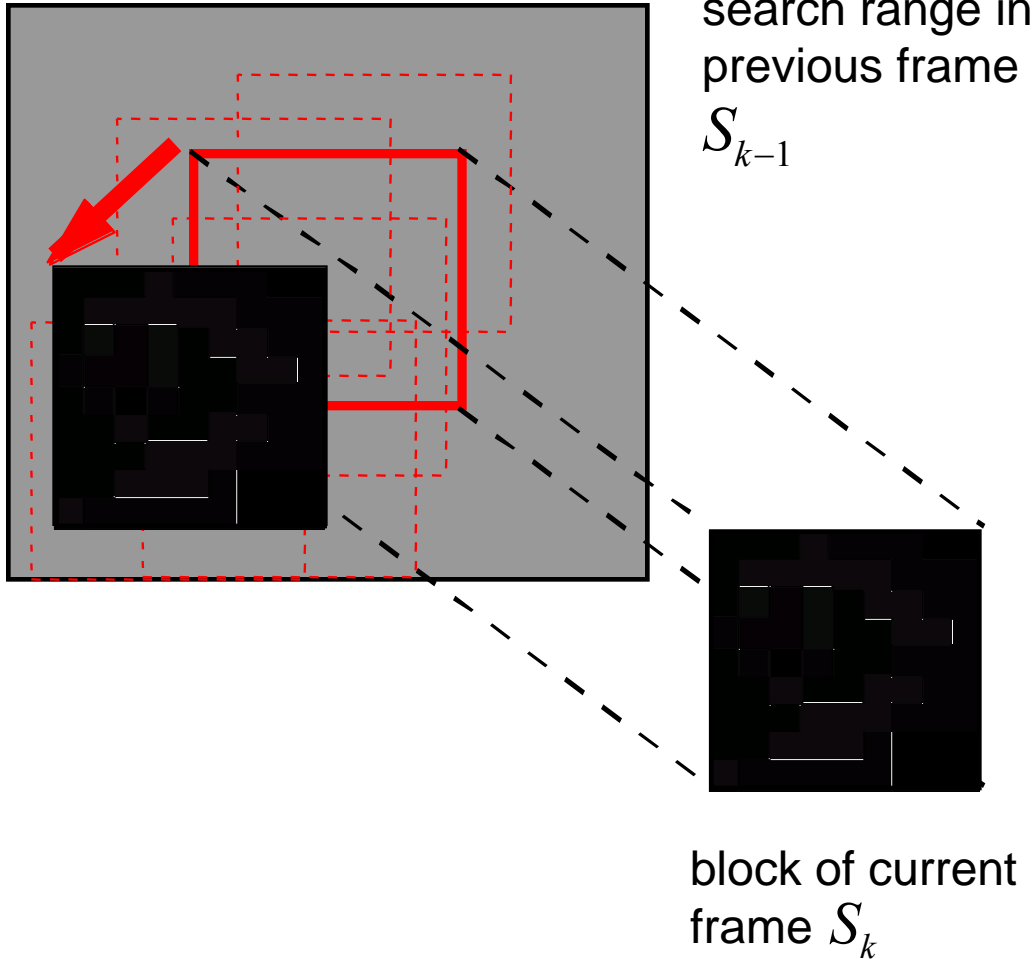
yields a system of linear equations:

$$u = s - s' - \left(\frac{\partial s'}{\partial x}, \frac{\partial s'}{\partial x} x, \frac{\partial s'}{\partial x} y, \frac{\partial s'}{\partial y}, \frac{\partial s'}{\partial y} x, \frac{\partial s'}{\partial y} y, \right) \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

- System can be solved using, e.g., pseudo-inverse, by minimizing

$$\arg \min_{a_1, a_2, a_3, b_1, b_2, b_3} \sum_{y=1}^{By} \sum_{x=1}^{Bx} u^2(x, y, t)$$

Block-matching Algorithm



- Subdivide current frame into blocks.
- Find one displacement vector for each block.
- Within a search range, find a “best match” that minimizes an error measure.
- Intelligent search strategies can reduce computation.

Block-matching Algorithm

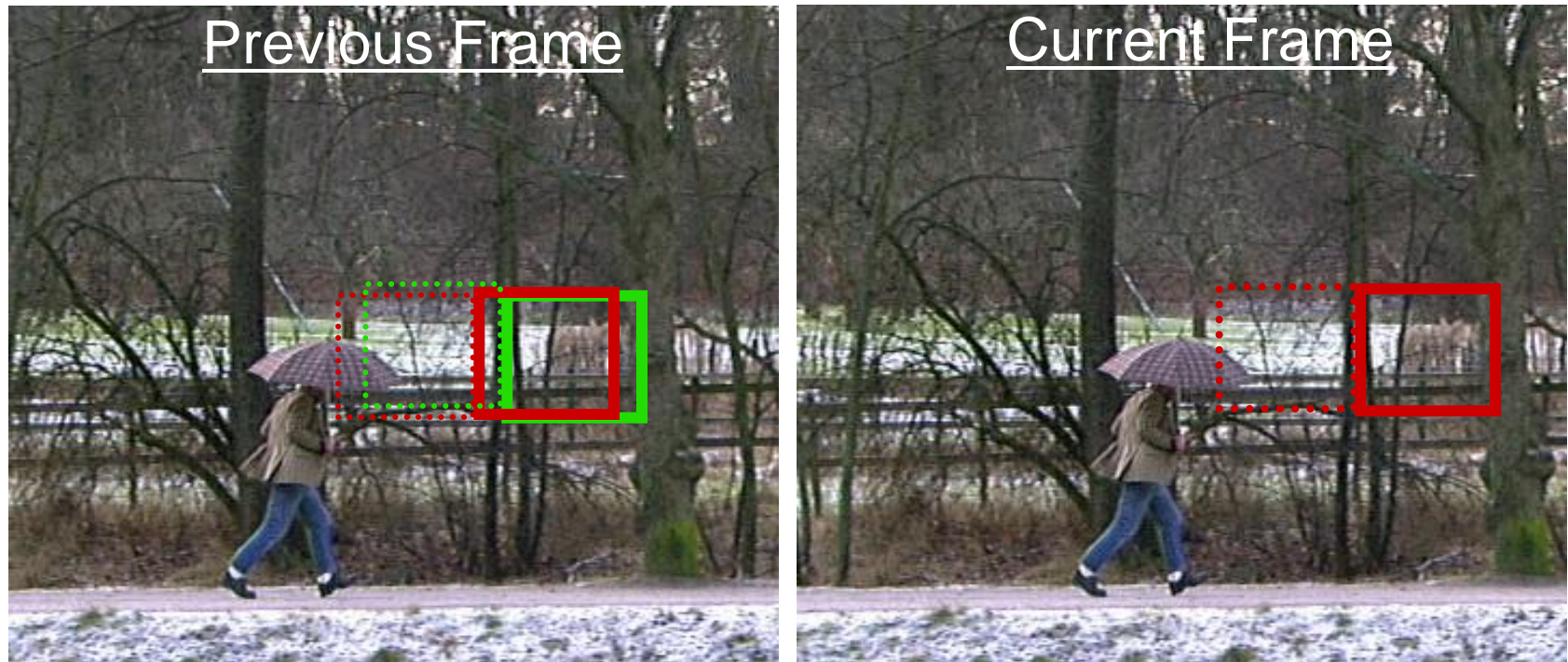


Measurement window is compared with a shifted block of pixels in the other image, to determine the best match



Block of pixels is selected as a measurement window

Block-matching Algorithm



. . . process repeated for another block.

Error Measures for Block-matching

- Mean squared error (sum of squared errors)

$$SSD(d_x, d_y) = \sum_{y=1}^{By} \sum_{x=1}^{Bx} [s(x, y, t) - s'(x - d_x, y - d_y, t - \Delta t)]^2$$

- Sum of absolute differences

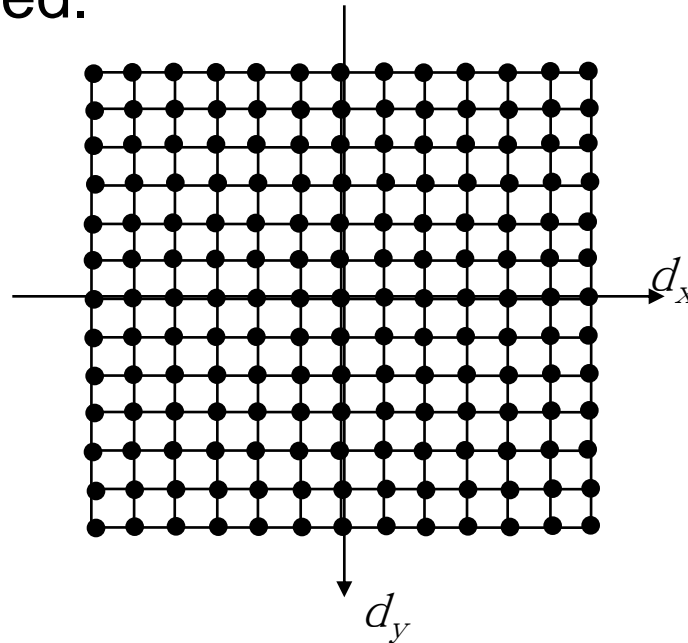
$$SAD(d_x, d_y) = \sum_{y=1}^{By} \sum_{x=1}^{Bx} |s(x, y, t) - s'(x - d_x, y - d_y, t - \Delta t)|$$

- Approximately same performance
→ SAD less complex for some architectures

Block-matching: Search Strategies

Full search

- All possible displacements within the search range are compared.



- Computationally expensive
- Highly regular, parallelizable

Speedup of Block-matching

Complexity of block-matching:

evaluation of complex error measure for many candidates

Reduce complexity

- Approximations
- Early terminations
- Exclude candidates

Reduce number

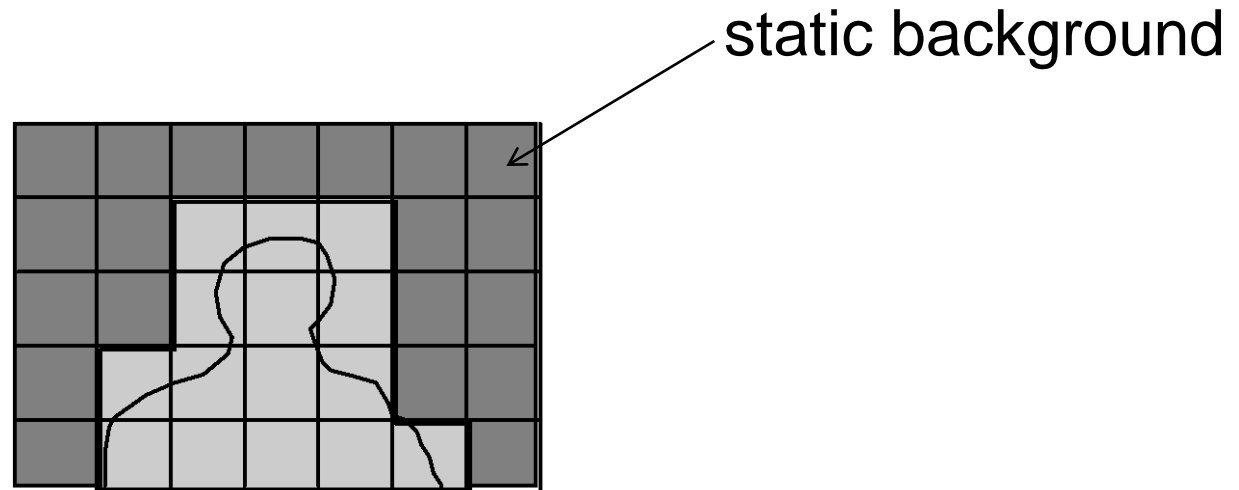
- Cover likely search areas
- Unequal steps between searched candidates

Combine both approaches:

Choose starting point and search order that maximizes likelihood for efficient approximations, early terminations and excluding of candidates

Approximations

- Stop search, if match is “good enough”
(SSD , $SAD < \text{threshold}$ or $J=D+\lambda R$ is small enough)
- Practical method in video conferencing for static background: test zero-vector first and stop search if match is good enough



Early Termination

- Partial distortion measure: $D_K(d_x, d_y)$, $K = N \dots B_y$

$$D_K(d_x, d_y) = \sum_{y=1}^K \sum_{x=1}^{B_x} [s(x, y, t) - s'(x - d_x, y - d_y, t - \Delta t)]^p$$


- Previously computed minimum: J_{\min}

- Early termination without loss:

$$\text{Stop, if } D_K(d_x, d_y) + \lambda_m \cdot R(d_x, d_y) \geq J_{\min}$$

- Early termination with possible loss, but higher speedup:

$$\text{Stop, if } D_K(d_x, d_y) + \lambda_m \cdot R(d_x, d_y) \geq \alpha(K) \cdot J_{\min}$$


e.g., $\alpha(K) = (K / B_y)^{1/2}$

Block Comparison Speed-Ups

- Triangle inequality for samples in block B (here SAD)

$$\sum_B |S_k - S_{k-1}| \geq \sum_B |S_k| - |S_{k-1}|$$

- Strategy:

1. Compute partial sums for blocks in current and previous frame
2. Compare blocks based on partial sums
3. Omit full block comparison, if partial sums indicate worse error measure than previous best result

- Block partitioning $B = \bigcup_n B_n, \quad \bigcap_n B = \emptyset$

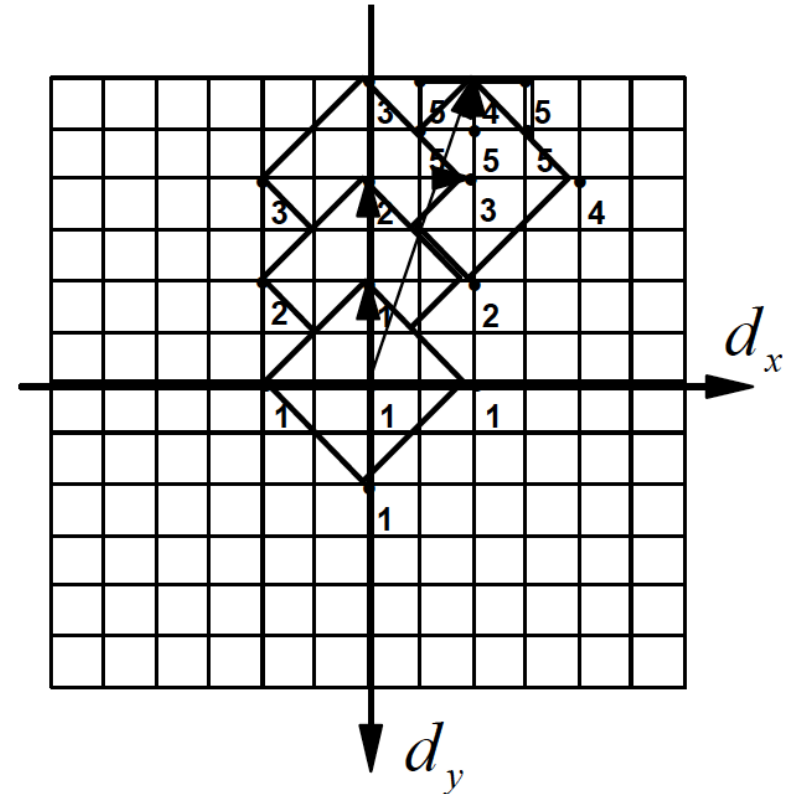
$$\sum_B |S_k - S_{k-1}| \geq \sum_n \sum_{B_n} |S_k - S_{k-1}|$$

- Choose blocks B_n to be nested (4x4, 8x8, 16x16, ...) – efficient for modern variable block size video codecs

Blockmatching: Search Order I

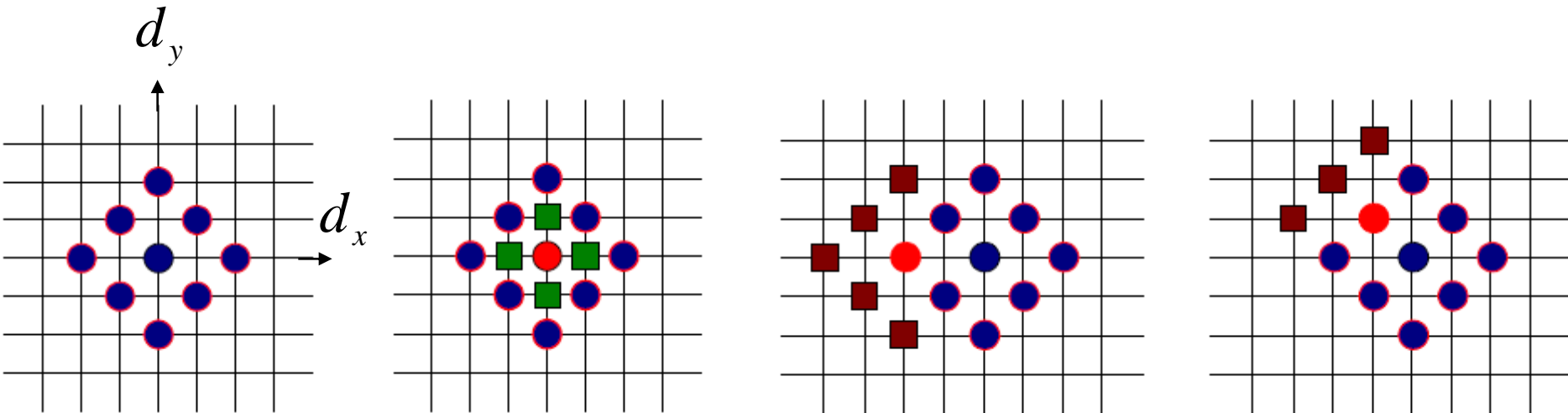
2D logarithmic search [Jain + Jain, 1981]

- Iterative comparison of error measure values at 5 neighboring points
- Logarithmic refinement of the search pattern if
 - best match is in the center of the 5-point pattern
 - center of search pattern touches the border of the search range



Blockmatching: Search Order II

Diamond search [Li, Zeng, Liou, 1994] [Zhu, Ma, 1997]

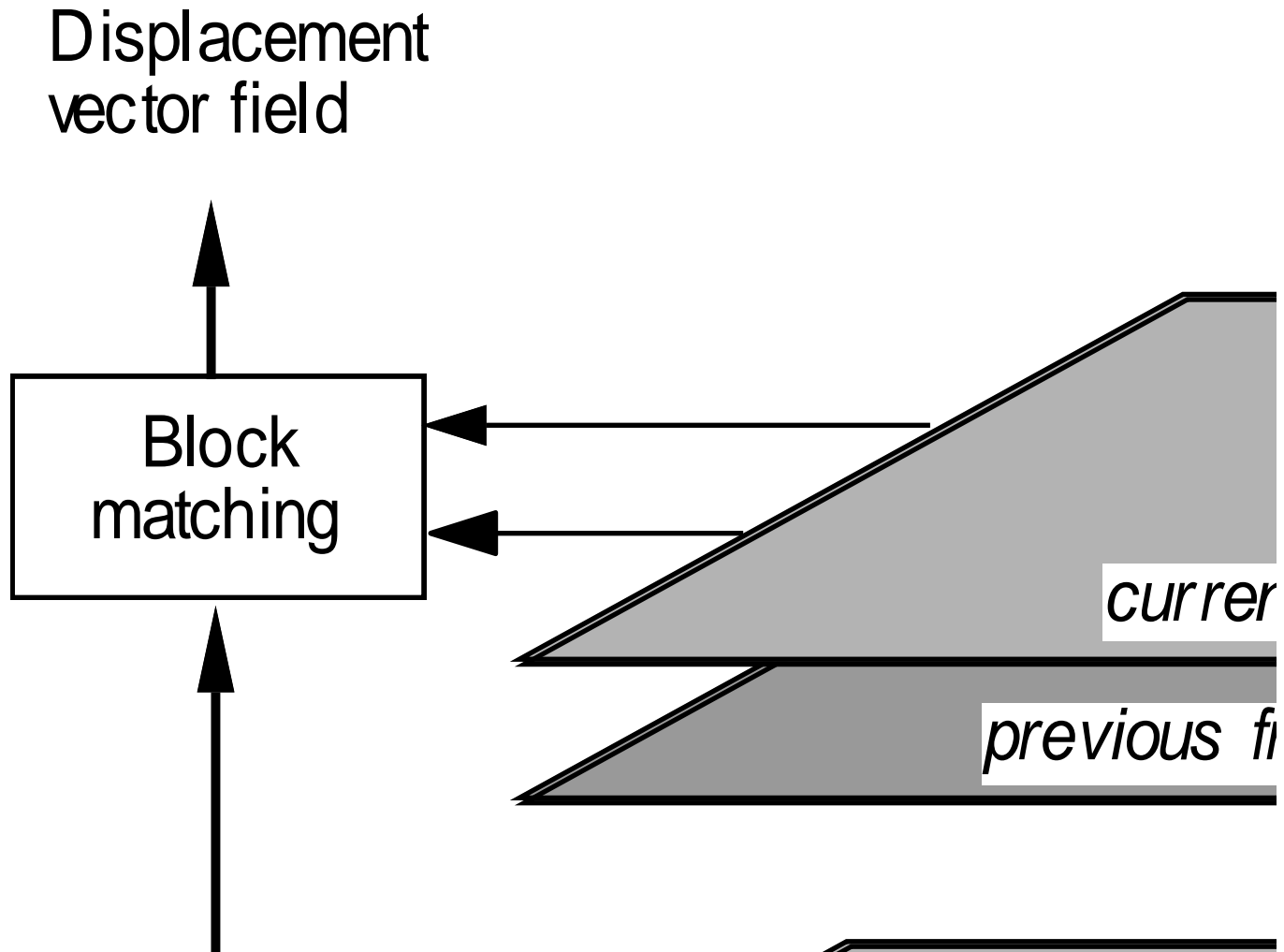


Start with
large diamond
pattern at
(0,0)

If best match
lies in the center
of large diamond,
proceed with
small diamond

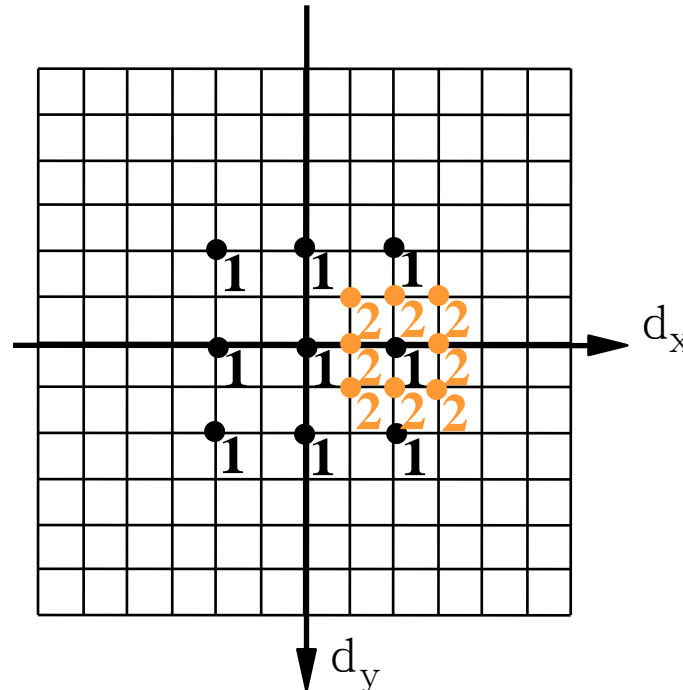
If best match does not lie
in the center of large
diamond, center large
diamond pattern at new
best match

Hierarchical blockmatching



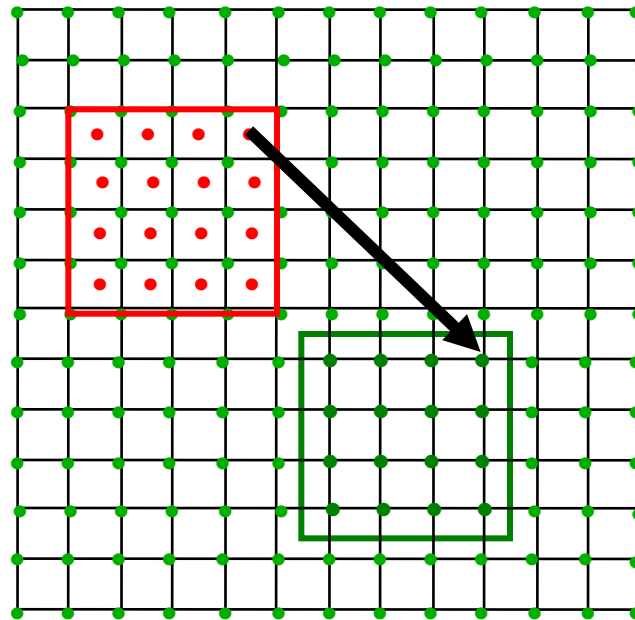
Sub-pel Translational Motion

- Motion vectors are often not restricted to only point into the integer-pel grid of the reference frame
- Typical sub-pel accuracies: half-pel and quarter-pel
- Sub-pel positions are often estimated by „refinement“



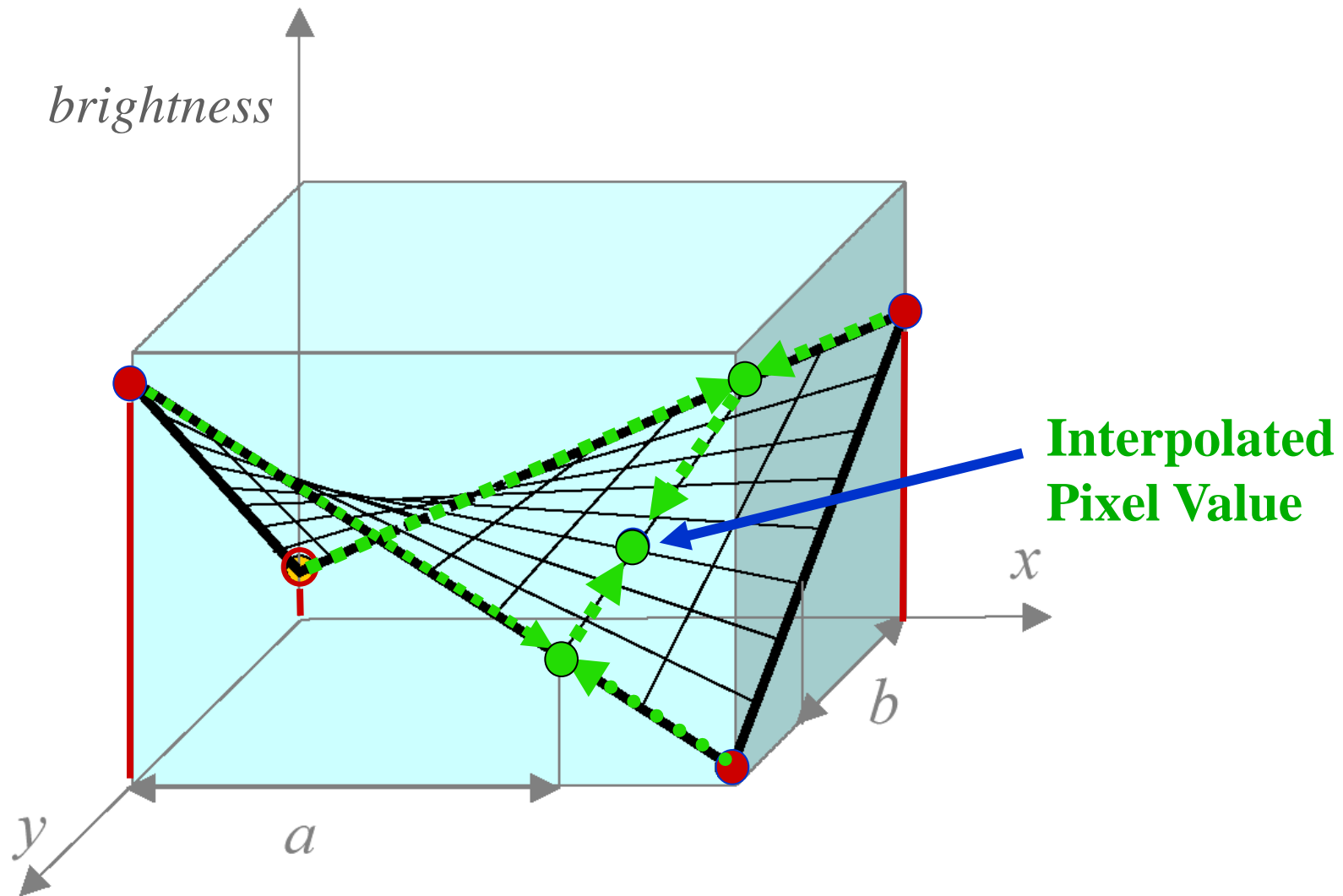
Sub-pel Motion Compensation

- Sub-pel positions are obtained via interpolation
- Example: half-pixel accurate displacements



$$\begin{pmatrix} d_x \\ d_y \end{pmatrix} = \begin{pmatrix} 4.5 \\ 4.5 \end{pmatrix}$$

Bi-linear Interpolation



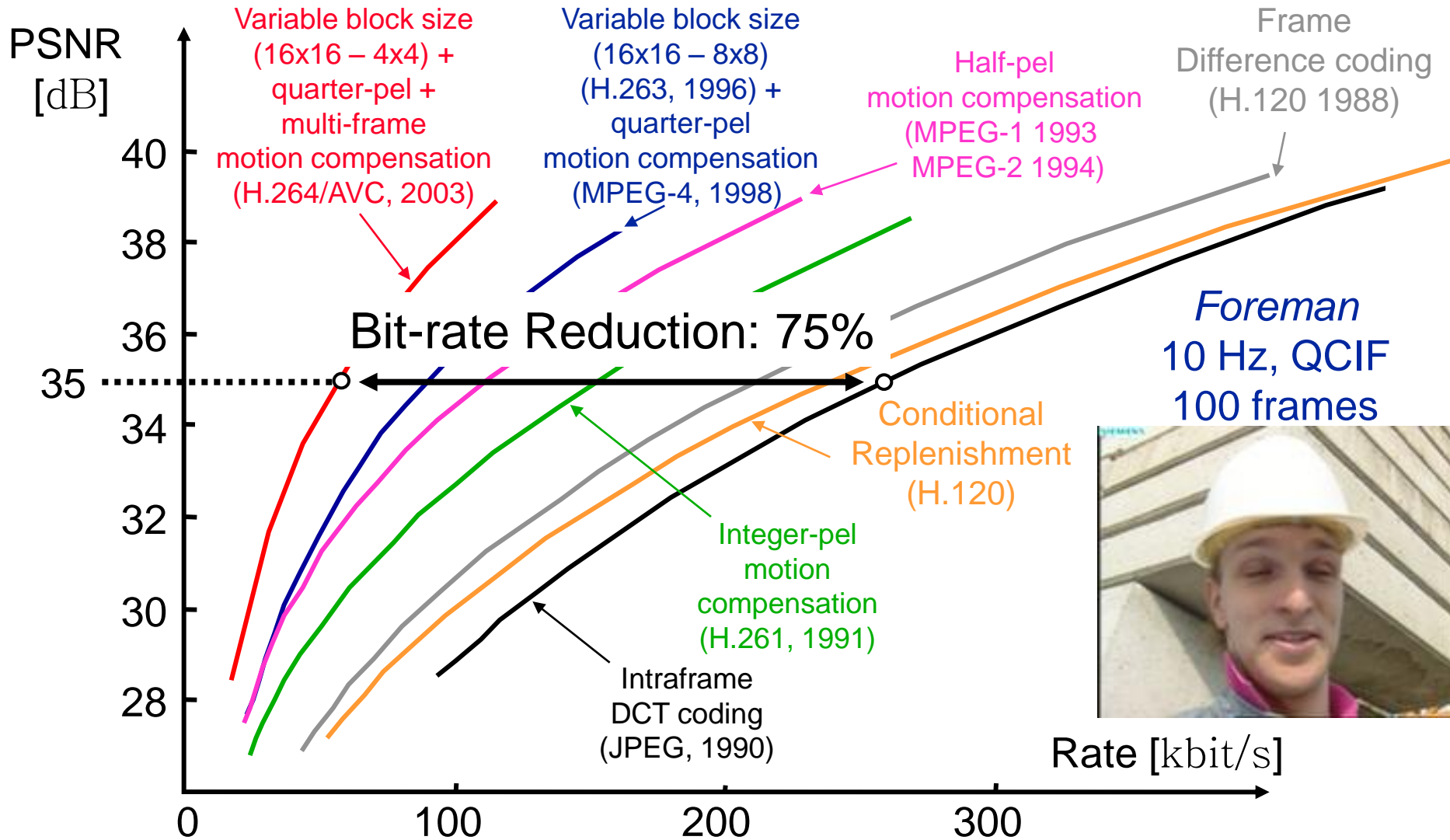
History of Motion Compensation

- *Intraframe coding*: only spatial correlation exploited
 - ➔ DCT [Ahmed, Natarajan, Rao 1974], JPEG [1992]
- *Conditional replenishment*
 - ➔ H.120 [1984] (*DPCM, scalar quantization*)
- *Frame difference coding*
 - ➔ H.120 Version 2 [1988]
- *Motion compensation: integer-pel accurate displacements*
 - ➔ H.261 [1991]
- *Half-pel accurate motion compensation*
 - ➔ MPEG-1 [1993], MPEG-2/H.262 [1994]
- *Variable block-size (16x16 & 8x8) motion compensation*
 - ➔ H.263 [1996], MPEG-4 [1999]
- *Variable block-size (16x16 – 4x4) and multi-frame motion compensation*
 - ➔ H.264/MPEG-4 AVC [2003]

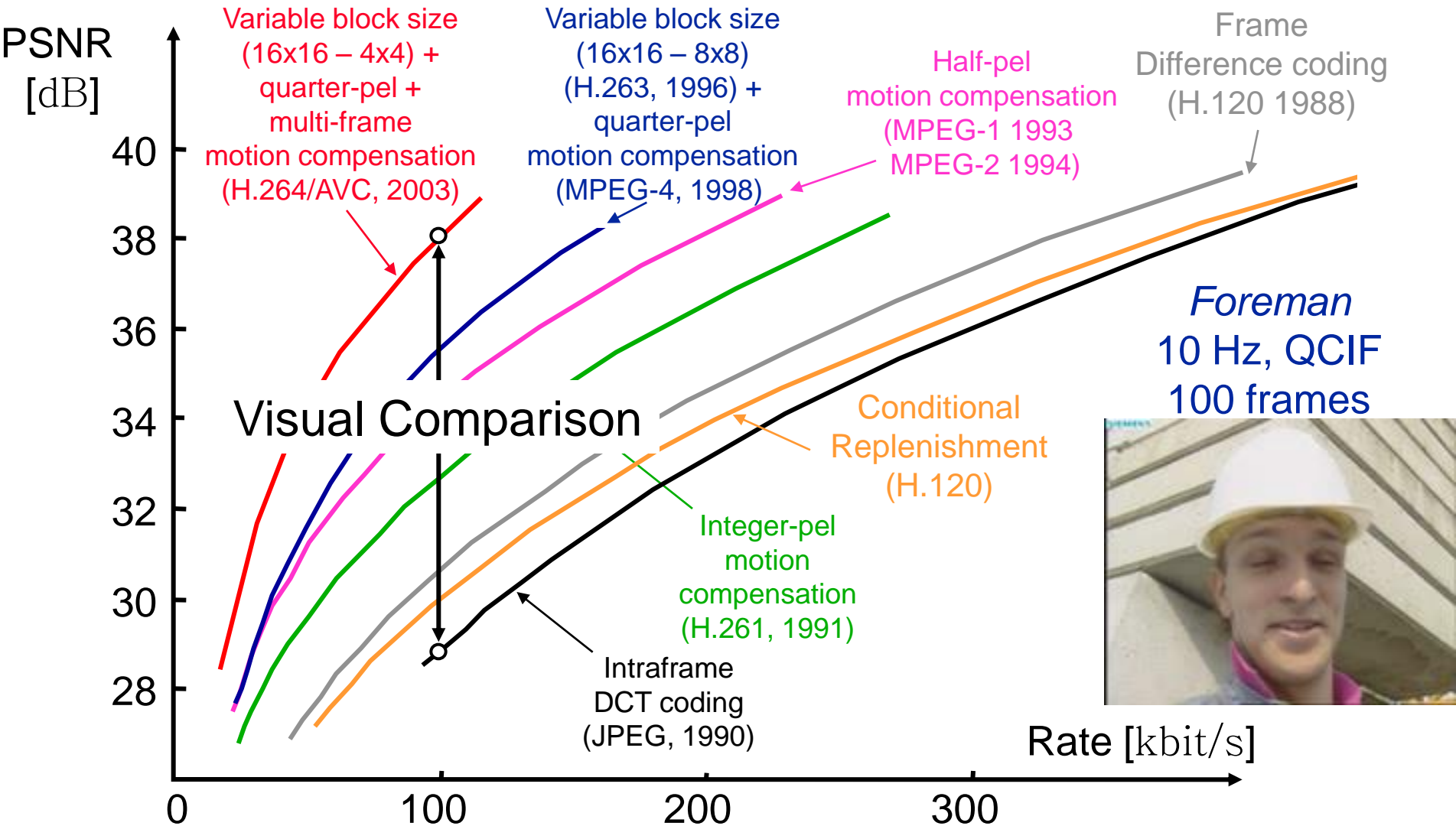
Complexity
increase



Milestones in Video Coding



Milestones in Video Coding



Visual Comparison

Foreman, QCIF, 10 Hz, 100 kbit/s

JPEG

H.264/AVC



Summary

- Video coding as a hybrid of motion compensation and prediction residual coding
- Motion models can represent various kinds of motions
- Lagrangian bit-allocation rules specify constant slope allocation to motion coefficients and prediction error
- In practice: affine or 8-parameter model for camera motion, translational model for small blocks
- Differential methods calculate displacement from spatial and temporal differences in the image signal -
- Block matching computes error measure for candidate displacements and finds best match
- Speed up block matching by fast search methods, approximations, early terminations and clever application of triangle inequality
- Hybrid video coding has been drastically improved by enhanced motion compensation capabilities