Video Compression - Cours 3

Nicolas Boutry[1]

↪ nicolas.boutry@lrde.epita.fr

[1] Laboratoire de Recherche et Développement de l'EPITA (LRDE), France

Novembre 2019

Outline

Outline

Complexity in the exhaustive case

- Assume we decompose an image of domain Λ into N blocks of size $S \times S$,
- We have then $|\Lambda| = N \times S \times S$ (no overlap),
- Even using an image of size 1024×1024, assuming each block is of size 32×32, we have 1024 blocks,
- We can then assume that we have $N >> 1$,
- We want to compute the complexity of various representations assuming that the range is $R > 0$,

- In the EBMA case,
  - We have to compute 1 MV by block,
  - It means 2 scalar parameters by block,
  - Since blocks are separated, we have to compute $(2R+1)^2$ energies by block,
  - Each (local) energy needs a number of S^2 operations,
  - We have then a complexity of $(2R+1)^2 * S^2$ operations by block,
  - Finally, we have a number of $(2R+1)^2 * S^2 * N = (2R+1)^2 \times |\Lambda|$,
  - Remark that this complexity does not depend on the number of blocks.
  - However the different final blocks can overlap of be disjoint.

- In the DBMA (nodal) case,
  - We have to compute 4 MVs by block,
  - It means 8 scalar parameters by block,
  - Since blocks are separated, we have to compute $(2R+1)^8$ energies by block,
  - Each (local) energy needs a number near to S^2 operations,
  - We have then a complexity of $(2R+1)^8 * S^2$ operations by bloc,
  - Finally, we have a number of $(2R+1)^8 * S^2 * N = (2R+1)^8 \times |\Lambda|$,
  - This representation is more flexible than the EBMA case, but costs much more in matter of computation.

- In the mesh-based case,
  - We have to compute about 1 MV by block (think N is big!),
  - It means 2 scalar parameters by block,
  - Blocks are not separated anymore!
  - We have then $2N$ parameters to estimate,
  - We have $(2R + 1)$ possibility by parameter,
  - Then we have $(2R + 1)^{2N}$ estimations of (global) energies to make!
  - Exhaustive computation is then impossible!
  - Also, note that, this time, the complexity depends on the number of blocks!

Multi-resolution motion estimation

Introduction I

- Various ME approaches can be reduced (see below) to solving an error minimization problem.
- Major difficulties:
  - many local minima in the gradient-descent case,
  - it is not easy to reach the global minimum except if we are close to the initial solution,
  - the amount of computation involved in the minimization process is very high.
- A good alternative is then the multi-resolution approach!
- It searches the solution for an optimization problem in successively finer resolutions.
- By first searching the solution in coarse resolution, one can usually obtain a solution that is close to the true motion.
- The total number of searches is reduced (small neighborhoods in each scale).

General formulation I

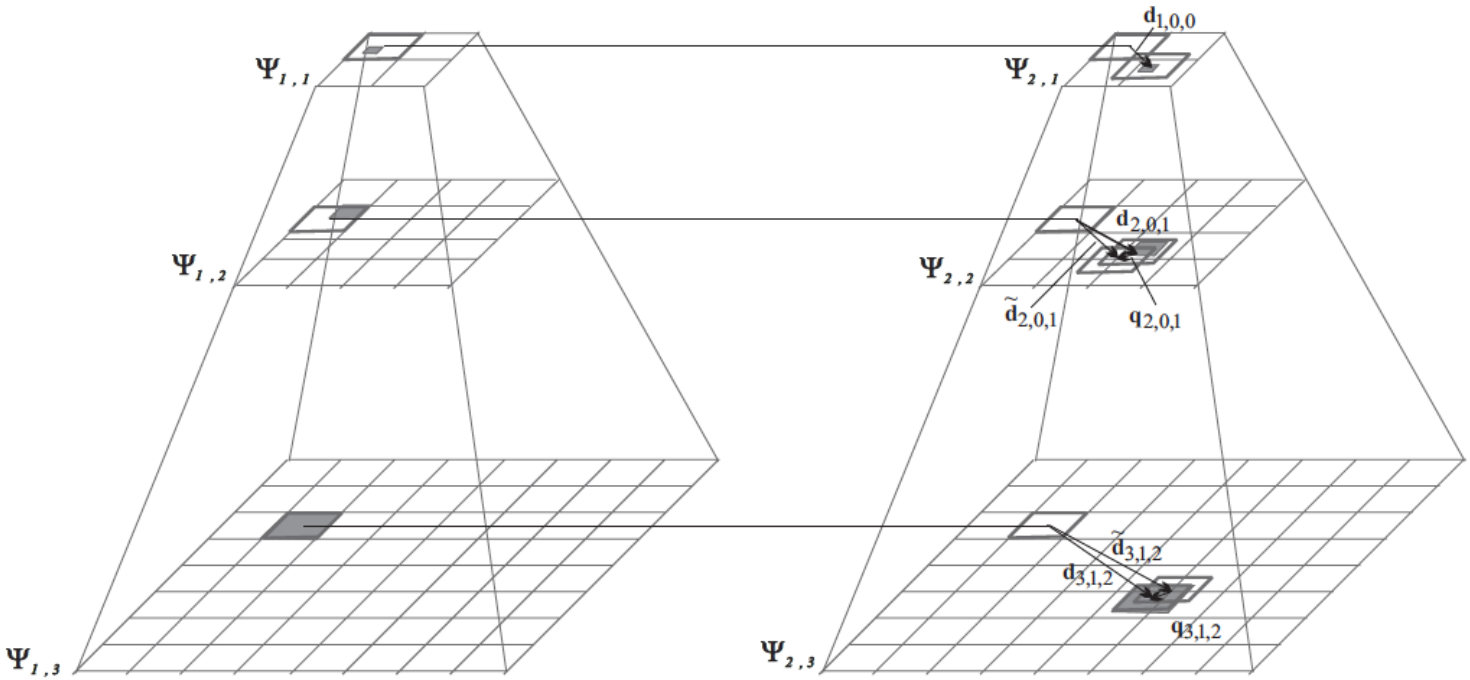

Figure 6.19. Illustration of the Hierarchical Block Matching Algorithm.

Figure: A multi-resolution approach (HBMA)

General formulation II

- In these pyramidal representations, the images at the top are the coarsest levels, and the images at the bottom are the original images.
- The motion field starts from the top coarsest level and progresses to the next finer level repeatedly.
- At each new finer resolution level, the motion field is interpolated and then refined.
- Assume that the number of levels is L, with the L-th level being the original image.
- Let the l-th level images of the reference and current frames be represented $\forall x \in \Lambda_l, \forall t \in \{1,2\}$, by:

$$\psi_{t,l}(x),$$

where Λ_l is the set of pixels at level l.

- Denote the total motion field obtained from levels 1 to $l-1$ by $d_{l-1}(x)$.

General formulation III

- At the l-th level, we first interpolate $d_{l-1}(x)$ to the resolution of level l, to produce an initial motion estimate:

$$\tilde{d}_l(x) = \mathcal{U}(d_{l-1}(x))$$

where $\mathcal{U}$ represents the interpolation.

- We then determine the update $q_l(x)$ at this level so that the error:

$$\sum_{x \in \Lambda_l} \left| \psi_{2,l}\left(x + \tilde{d}_l(x) + q_l(x)\right) - \psi_{1,l}(x) \right|^p$$

is minimized.

- The new motion field obtained after this step is:

$$d_l(x) = q_j(x) + \tilde{d}_l(x)$$

General formulation IV

- Upon completion of successive refinements, the total motion at the finest resolution is:

$$d(x) = q_L(x) + \mathcal{U}(q_{L-1}(x) + \mathcal{U}(q_{L-2}(x) + \mathcal{U}(...(q_1(x) + \mathcal{U}(q_0(x) + d_0(x))))))$$

- The initial conditions for the above procedure is $d_0(x) = 0$.
- The benefits of the multi-resolution are two folds:
  - First, the minimization problem at a coarser resolution is less ill-posed than at a finer resolution,
  - The solution obtained at a coarser level is closer to the true solution at that level,
  - At the end, we converge then more easily to the global minimum.
  - Second, the estimation at each resolution level can be confined to a smaller range,
  - Then the total number of searches is smaller than in the other methods.
- The use of multi-resolution representation for image processing was first introduced by Burt and Adelson (Laplacian Pyramid).

Hierarchical block matching algorithm (HBMA) I

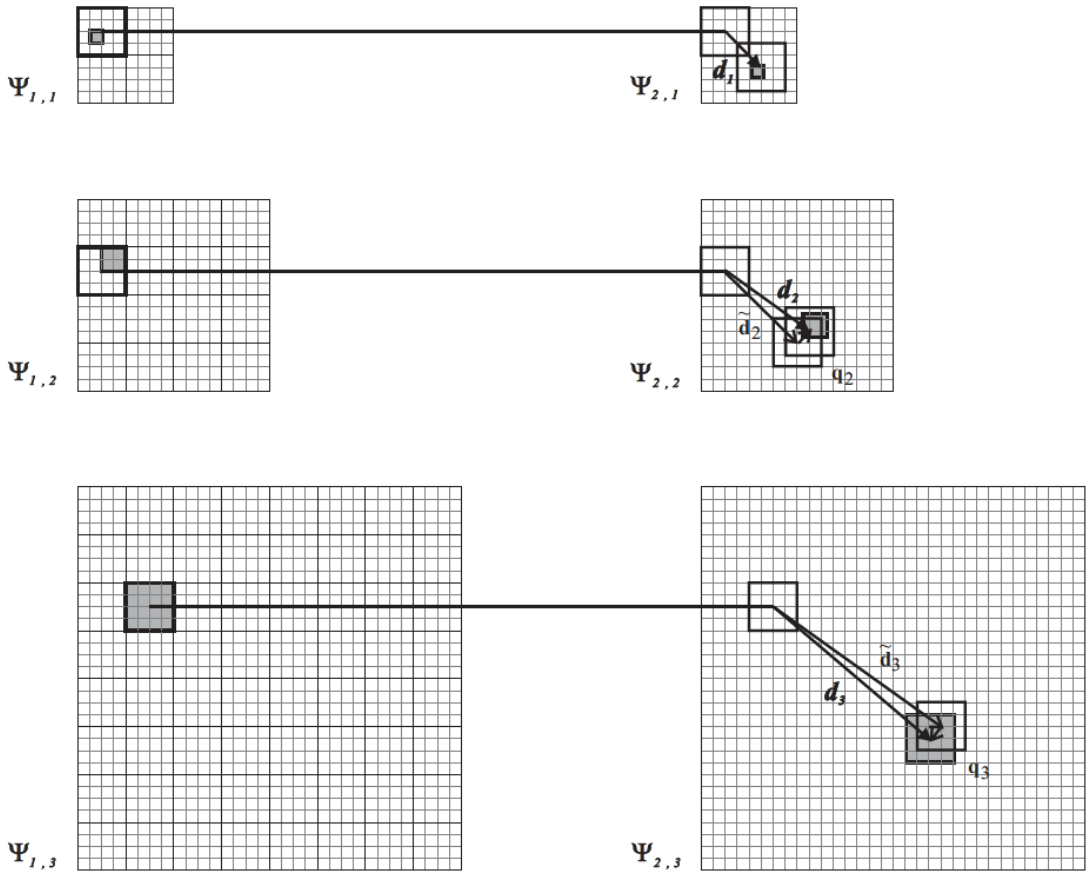

Figure 6.20. An example of using 3-level HBMA for block motion estimation. See Example 6.3.

Hierarchical block matching algorithm (HBMA) II

- Here, we introduce a hierarchical block-matching algorithm (HBMA), which is a special case of the multi-resolution approach presented before.
- We assume that the same block size is used at different levels.
- The number of blocks is then reduced by half in each dimension.
- Let the MV for block (m, n) at level l be denoted by $d_{l,m,n}$.
- Starting from level 1, we first find the MVs for all blocks in this level.
- At each new level $l > 1$, for each block, its initial MV $\tilde{d}_{l,m,n}$ is interpolated from a corresponding block in level $l - 1$ by:

$$\tilde{d}_{l,m,n} = \mathcal{U}(d_{l-1,\lfloor m/2 \rfloor,\lfloor n/2 \rfloor}) = 2\, d_{l-1,\lfloor m/2 \rfloor,\lfloor n/2 \rfloor}$$

- Then a correction vector $q_{l,m,n}$ is searched, yielding the final estimated MV:

$$d_{l,m,n} = \tilde{d}_{l,m,n} + q_{l,m,n}$$

Hierarchical block matching algorithm (HBMA) III

- Note that using a block width N at level l corresponds to a block width of $2^{l-1}N$ at the full resolution.
- By using the same block size, search range and stepsize at different levels, we actually use in the beginning of the search:
  - a larger block size,
  - a larger search range,
  - a larger stepsize
- The complexity using HBMA is about:

$$4^{-(L-2)}|\Lambda|R^2$$

when the search range is R in the finest resolution, and the pyramid has L levels.

Hierarchical block matching algorithm (HBMA) IV

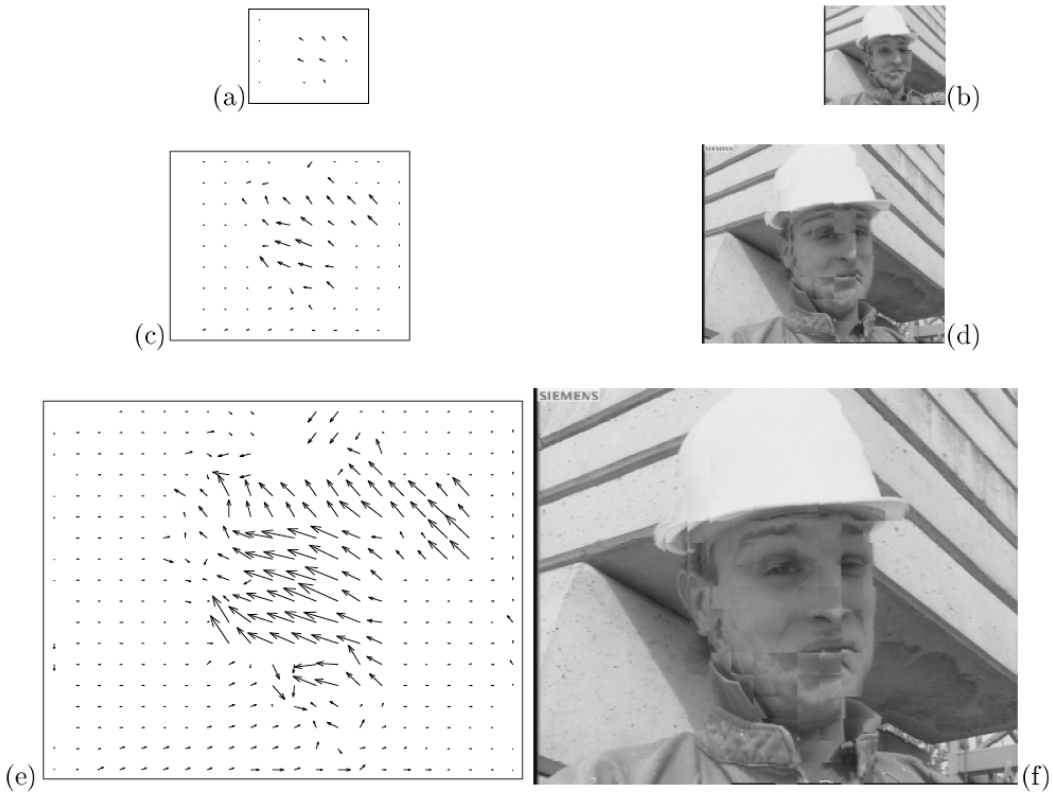

Figure 6.21. Example motion estimation results by HBMA for the two images shown in Fig. 6.8: (a-b) the motion field and predicted image at level 1; (c-d) the motion field and predicted image at level 2; (e-f) the motion field and predicted image at the final level (PSNR=29.32); A three-level HBMA algorithm is used. The block size is 16×16 at all levels. The search range is 4 at all levels with integer-pel accuracy. The result in the final level is further refined by a half-pel accuracy search in the range of ± 1.

Hierarchical block matching algorithm (HBMA) V

- We can see that the multi-resolution approach indeed yields a smoother motion field than EBMA.
- Visual observation also reveals that this motion field represents more the underlying motion between the two image frames.
- This is true in spite of the fact that the EBMA yields a higher PSNR (but the PSNR is not always a good measure!).
- At the end, when EBMA will have required $4.3E + 8$ operations, the HBMA will have required $1.1E + 7$ operations.
- There are many variants to implementation of HBMA as the quad-tree BMA.
- In this case, all the processing is done on the original image resolution.

Computation of the derivative of the energy in the node-based case

- Since we use the node-based motion model, the motion parameters for any block are the nodal MVs, i.e.,

$$\boldsymbol{a} = [d_k; k \in \mathcal{K}],$$

with $\mathcal{K} = \{1, 2, \ldots, K\}$.

- We are going to use the gradient-based method to minimize the following energy:

$$E(\boldsymbol{a}) = \sum_{\boldsymbol{x} \in B} \left(\psi_2(w(\boldsymbol{x}; \boldsymbol{a})) - \psi_1(\boldsymbol{x})\right)^2$$

- The new position $w(\boldsymbol{x}; \boldsymbol{a})$ of each $\boldsymbol{x}$ is predicted w.r.t. the motions d_k of the nodes of each quadrangle B so that:

$$w(\boldsymbol{x}; \boldsymbol{a}) = \boldsymbol{x} + \sum_{k \in \mathcal{K}} \phi_k(\boldsymbol{x}) d_k$$

- We define the motion parameter as:

$$\boldsymbol{a} = [\boldsymbol{a}_x^T, \boldsymbol{a}_y^T]^T$$

,
with $\boldsymbol{a}_x = [d_{x,1}, d_{x,2}, \ldots, d_{x,K}]^T$ and $\boldsymbol{a}_y = [d_{y,1}, d_{y,2}, \ldots, d_{y,K}]^T$, then $\boldsymbol{a}$ is a vector made of $2K$ scalars:

$$\boldsymbol{a} = [d_{x,1}, d_{x,2}, \ldots, d_{x,K}, d_{y,1}, d_{y,2}, \ldots, d_{y,K}]^T,$$

- In other words, for $i \in [1, K]$, $a_x^i = d_{x,i}$, and $a_y^i = d_{y,i}$,
- To begin the computation of the derivative of the energy, we separate it into two vectors $\frac{\partial E}{\partial \boldsymbol{a}_x}$ and $\frac{\partial E}{\partial \boldsymbol{a}_y}$,
- So, we have for the moment:

$$\frac{\partial E}{\partial \boldsymbol{a}} = \left[\frac{\partial E}{\partial \boldsymbol{a}_x}; \frac{\partial E}{\partial \boldsymbol{a}_y}\right]^T$$

- We want to compute each of these two derivatives, let us begin with $\frac{\partial E}{\partial \boldsymbol{a}_x}$:

- Let i be an integer in $[1, K]$,

$$\frac{\partial E}{\partial a_x^i} = \frac{\partial}{\partial a_x^i}\left(\sum_{\boldsymbol{x}\in B}(\psi_2(w(\boldsymbol{x};\boldsymbol{a})) - \psi_1(\boldsymbol{x}))^2\right),$$

- Since the sum on $\boldsymbol{x}$ is finite, we obtain:

$$\frac{\partial E}{\partial a_x^i} = \sum_{\boldsymbol{x}\in B}\frac{\partial}{\partial a_x^i}\left((\psi_2(w(\boldsymbol{x};\boldsymbol{a})) - \psi_1(\boldsymbol{x}))^2\right),$$

- Now we can derive:

$$\frac{\partial E}{\partial a_x^i} = \sum_{\boldsymbol{x}\in B} 2\,(\psi_2(w(\boldsymbol{x};\boldsymbol{a})) - \psi_1(\boldsymbol{x}))\,\frac{\partial}{\partial a_x^i}(\psi_2(w(\boldsymbol{x};\boldsymbol{a})) - \psi_1(\boldsymbol{x})),$$

- Now we can define the error image:

$$e(\boldsymbol{x};\boldsymbol{a}) = \psi_2(w(\boldsymbol{x};\boldsymbol{a})) - \psi_1(\boldsymbol{x}),$$

- We obtain then:

$$\frac{\partial E}{\partial a_x^i} = 2 \sum_{\boldsymbol{x} \in B} e(\boldsymbol{x}; \boldsymbol{a}) \frac{\partial}{\partial a_x^i} (\psi_2(w(\boldsymbol{x}; \boldsymbol{a})) - \psi_1(\boldsymbol{x})),$$

- Since $\psi_1(\boldsymbol{x})$ does not depend on $\boldsymbol{a}$:

$$\frac{\partial E}{\partial a_x^i} = 2 \sum_{\boldsymbol{x} \in B} e(\boldsymbol{x}; \boldsymbol{a}) \frac{\partial}{\partial a_x^i} (\psi_2(w(\boldsymbol{x}; \boldsymbol{a}))),$$

- Let us denote w_x and w_y this way:

$$w(\boldsymbol{x}; \boldsymbol{a}) = [w_x(\boldsymbol{x}; \boldsymbol{a}), w_y(\boldsymbol{x}; \boldsymbol{a})]^T,$$

with

$$w_x(\boldsymbol{x}; \boldsymbol{a}) = x + \sum_{k \in \mathcal{K}} \phi_k(\boldsymbol{x}) d_{x,k}$$

and

$$w_y(\boldsymbol{x}; \boldsymbol{a}) = y + \sum_{k \in \mathcal{K}} \phi_k(\boldsymbol{x}) d_{y,k}$$

- Then,

$$\frac{\partial}{\partial a_x^i}(\psi_2(w(\boldsymbol{x};\boldsymbol{a}))) = \frac{\partial}{\partial a_x^i}(\psi_2(w_x(\boldsymbol{x};\boldsymbol{a}), w_y(\boldsymbol{x};\boldsymbol{a}))), \tag{1}$$

$$= \frac{\partial \psi_2}{\partial x}(w(\boldsymbol{x};\boldsymbol{a}))\frac{\partial(w_x(\boldsymbol{x};\boldsymbol{a}))}{\partial a_x^i} + \frac{\partial \psi_2}{\partial y}(w(\boldsymbol{x};\boldsymbol{a}))\frac{\partial(w_y(\boldsymbol{x};\boldsymbol{a}))}{\partial a_x^i} \tag{2}$$

- We can easily observe that:

$$\frac{\partial(w_x(\boldsymbol{x};\boldsymbol{a}))}{\partial a_x^i} = \frac{\partial(x + \sum_{k\in\mathcal{K}} \phi_k(\boldsymbol{x}) d_{x,k})}{\partial a_x^i} = \phi_i(\boldsymbol{x})$$

and

$$\frac{\partial(w_y(\boldsymbol{x};\boldsymbol{a}))}{\partial a_x^i} = \frac{\partial(y + \sum_{k\in\mathcal{K}} \phi_k(\boldsymbol{x}) d_{y,k})}{\partial a_x^i} = 0$$

so

$$\frac{\partial}{\partial a_x^i}(\psi_2(w(\boldsymbol{x};\boldsymbol{a}))) = \frac{\partial \psi_2}{\partial x}(w(\boldsymbol{x};\boldsymbol{a}))\phi_i(\boldsymbol{x}),$$

- We define:

$$\boldsymbol{\phi}(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x})), \ldots, \phi_K(\boldsymbol{x})]^T,$$

- Then we finally obtain:

$$\frac{\partial E}{\partial \boldsymbol{a}_x}(\boldsymbol{a}) = 2 \sum_{\boldsymbol{x} \in B} e(\boldsymbol{x}; \boldsymbol{a}) \frac{\partial \psi_2(w(\boldsymbol{x}; \boldsymbol{a}))}{\partial x} \boldsymbol{\phi}(\boldsymbol{x}),$$

  and by symmetry we also obtain:

$$\frac{\partial E}{\partial \boldsymbol{a}_y}(\boldsymbol{a}) = 2 \sum_{\boldsymbol{x} \in B} e(\boldsymbol{x}; \boldsymbol{a}) \frac{\partial \psi_2(w(\boldsymbol{x}; \boldsymbol{a}))}{\partial y} \boldsymbol{\phi}(\boldsymbol{x}),$$

- Done :)

Outline

Waveform-based video coding: a brief introduction I

We are going to talk about:

- Karhunen-Loeve Transform (KLT)
- DCT-based image/video coding
- correlation between neighboring pixels (smooth changes)
- correlation between consecutive frames of a video sequence (motion of physical objects in the scene)
- coding techniques using spatial/temporal prediction

Motivation behind the block-based transform coding I

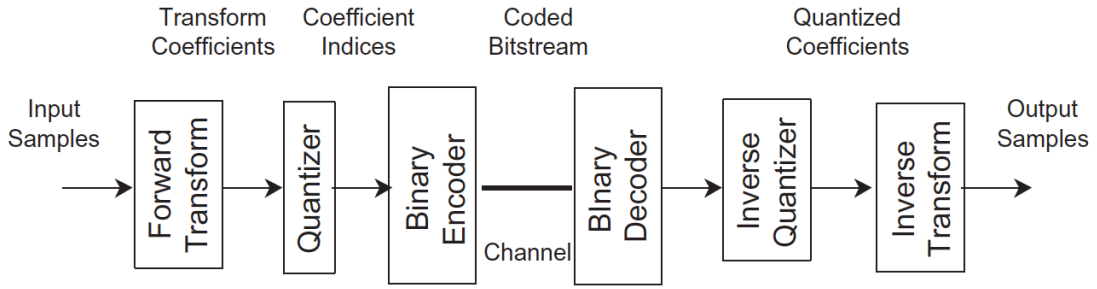

Figure 9.1. Block Diagrams for the Encoder and Decoder of a Block Transform Coding System.

- Transform coding has been proven to be specially effective for compression of still images and video frames,
- A transform can fully exploit the spatial correlation among pixels.
- But to reduce computational complexity, block-based transform coding is more often used (DCT).

Block-based transform coding [REMINDER]

Overview of the block-based transform coding I

- Block-based transform coding schemes (encoding):
  - (1) we divide an image into non-overlapping blocks.
  - (2) we transform each block in a set of coefficients.
  - (3) each coefficient is quantized separately using a scalar quantizer.
  - (4) the quantized coefficients indices are finally converted into binary bits using variable length coding (VLC).
- Block-based transform coding schemes (decoding): the image block is then recovered from the quantized coefficients through an inverse transform.
- The forward transform process can be seen as a projection onto selected basis functions (linear combination of basic patterns also called transform basis functions).
- The inverse transform reconstructs the initial function/image using (a part of) the coefficients computed before.

Overview of the block-based transform coding II

- The performance of a transform coder depends on the basis functions used.
- A good transform should compact the energy of the original pixel block into as few coefficients as possible,
- Usually coefficients correspond to decorrelated parts of the signal,
- The best transform according to these criteria is the KLT (PCA),but the Discrete Cosine Transform (DCT) is faster, independent of the signal, and is a good approximation of the KLT.
- In almost transform-based image coders, the DCT is used.
- One way to exploit the correlation among adjacent pixels is to quantize a block of pixels together using vector quantization (VQ)
- It replaces each image block by one of the ”typical” block patterns that is closest to the original block.

Overview of the block-based transform coding III

- Problem of the VQ: the more we want block size to be high (to compress more), the higher is the complexity of the research of the optimal typical blocks.
- Transform coding is one way to implement a constrained vector quantizer without using an exhaustive search.
- Specifically, the codewords in this quantizer are those representable by linear combinations of basis vectors with quantized coefficients:

$$u(x) = \sum_i w_i \, v_i(x),$$

- Interpretation of the coefficients v_i: each transform coefficient represents the contribution from one basis vector.
- A special class of linear transform is the unitary transform, in which the basis vectors are orthonormal to each other:

$$\|v_i\|_2 = 1, \forall i$$

Overview of the block-based transform coding IV

- To minimize the approximation error for a particular signal vector, we simply choose the K coefficients v_i that have the largest absolute values,
- We can then reconstruct the signal using only these coefficients with the weighted sum of basis functions
- It is equivalent to consider the others coefficients as equal to zero.
- It works in the same way in 1D and 2D when we have separability of the orthonormal transform basis functions:

$$v_i(x, y) = v_i^x(x) \times v^y(y)$$

- Typically, the complexity in the separable case is $O(N^3)$ when in the non-separable case, it is equal to $O(N^4)$ (squarred images of size $N \times N$).

Overview of the block-based transform coding V

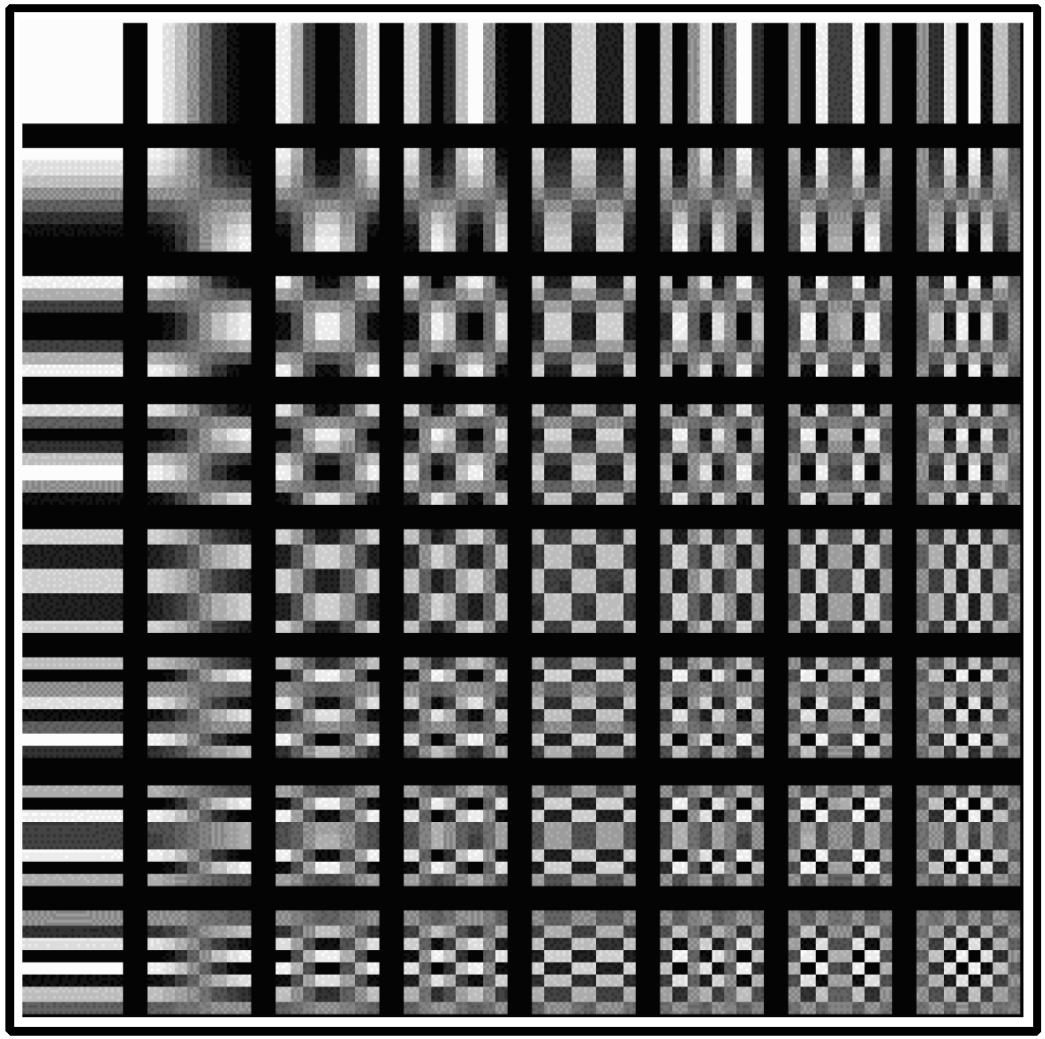

Figure 9.3. Basis images of 8×8 DCT transform.

Overview of the block-based transform coding VI

- Interpretation of the DCT: each DCT coefficient specifies the contribution of a sinusoidal pattern at a particular frequency of the actual signal.

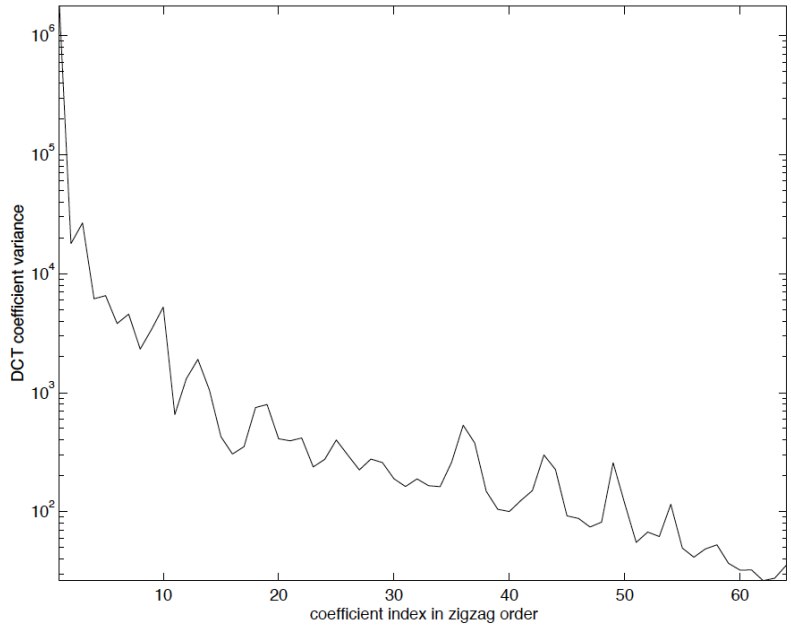

Figure 9.4. Energy distribution of the 8×8 DCT coefficients of the test image "flower".

Overview of the block-based transform coding VII

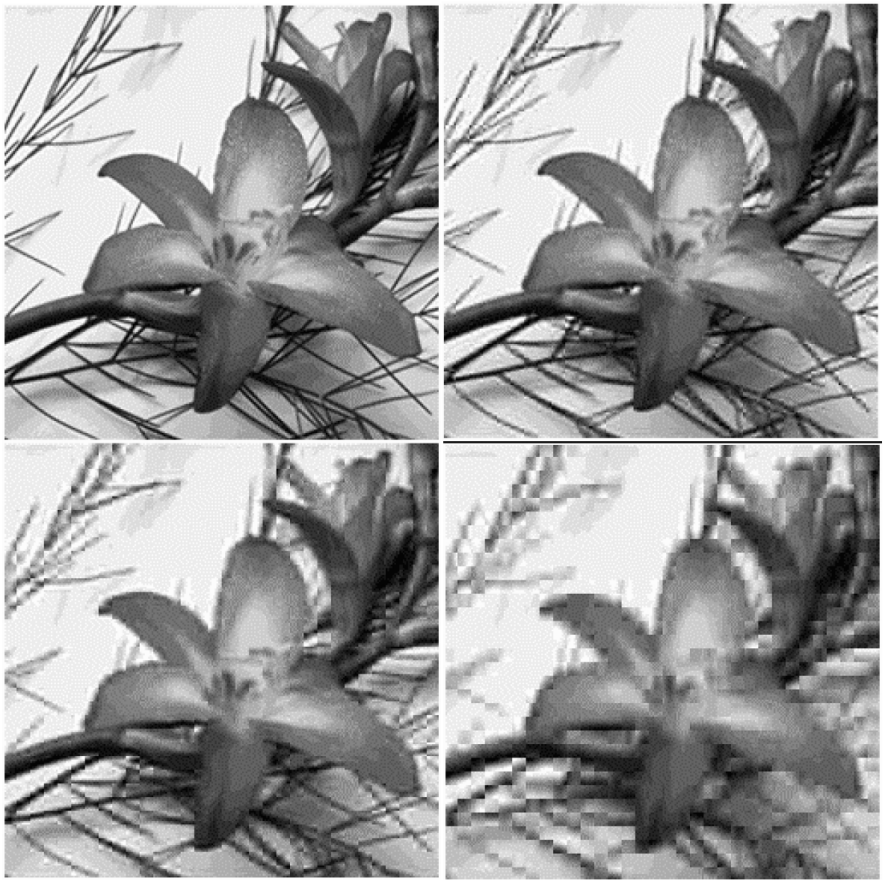

Figure 9.5. Reconstructed images with different numbers of DCT coefficients. The top left is the original image with 64 coefficients per block, the top right with 16 coefficients, the bottom left with 8 coefficients, and the bottom right with 4 coefficients. The DCT coefficients are arranged in the zigzag order.

Overview of the block-based transform coding VIII

- The reason that DCT is well suited for image coding is that an image block can often be represented with a few low frequency DCT coefficients
- This is because the intensity values in an image are usually varying smoothly and very high frequency components only exists near edges.
- Rate-Distortio: the goal is to derive the relation between the average distortion per sample in the vector and average bit rate per sample.
- Optimal bit allocation: Given the desired average bit rate R, the question is how should we allocate the total bit R N among the N coefficients so that the error D is minimized.
- This problem is equivalent to minimize J defined as:

$$J := N\,R + \lambda D,$$

with λ satisfying some constraints.

Overview of the block-based transform coding IX

- This approach (very complex from a theoretical point of view!) proves that a coefficient with a larger variance should be given more bits.
- Furthermore, the optimal allocation is such that all the coefficients have the same quantization error.
- Fundamental result: the KLT minimizes the coding distortion (Gaussian source or not),
- Another property of the KLT is that it yields the minimal approximation error with $K < N$ coefficients among all unitary transforms.
- In general, the DCT is not as efficient as the KLT, but the KLT is only efficient for stationnary sources (with a known covariance matrix).
- In reality, the source may be spatially/temporally varying, and one needs to constantly update the covariance matrix, which is computationally very high demanding,

Overview of the block-based transform coding X

- Furthermore, there exist no fast algorithms for the KLT derived from an arbitrary covariance matrix.
- Therefore, for practical applications, it is desirable to employ transforms that are signal-independent.
- It has been shown that the DCT is very close to the KLT for the covariance matrices of common image signals.
- Therefore, the DCT has been used in place of the KLT for image coding.
- Remark: we could imagine that we do VQ on the coefficient matrices, however it has been observed in practice that it is not efficient (since the transform coefficients are in general decorrelated).

Predictive coding

Overview of predictive coding I

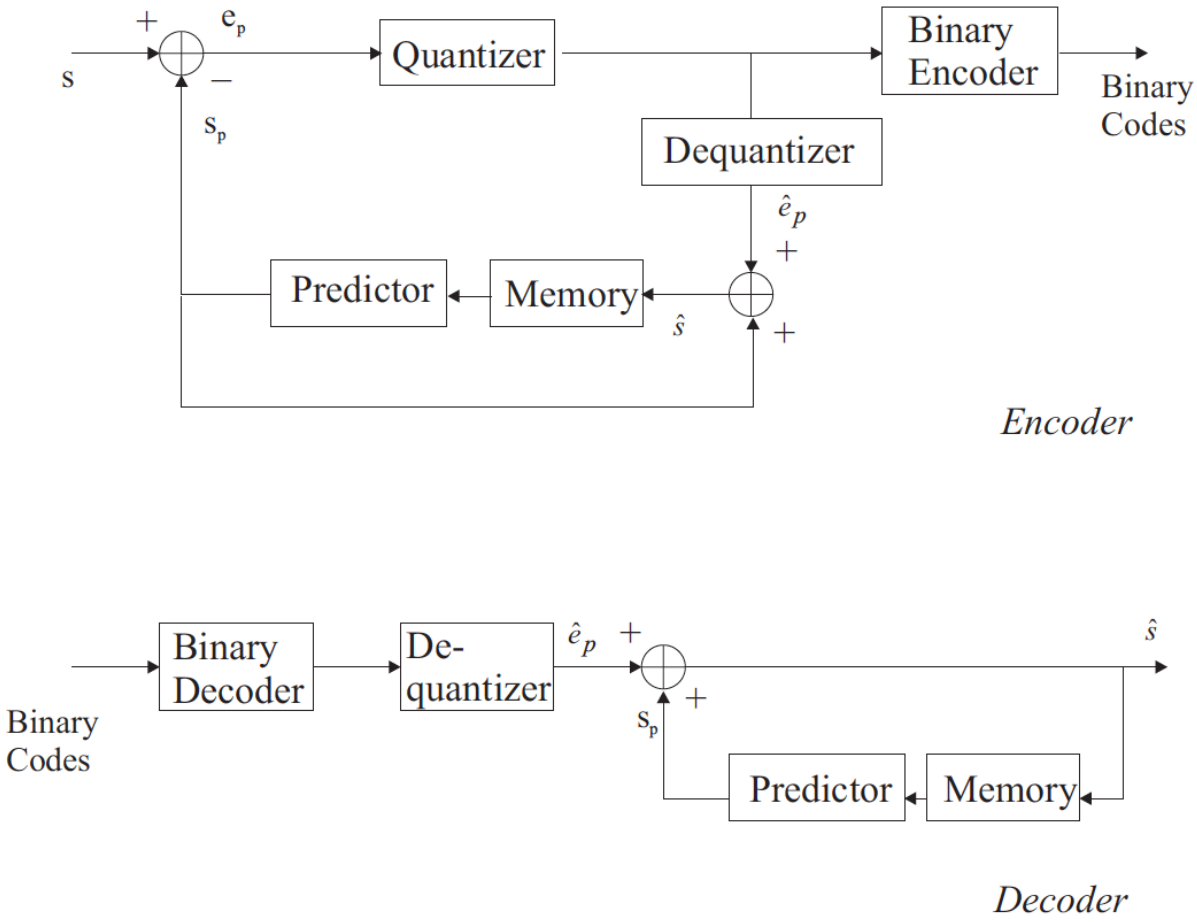

Figure 9.11. Encoder and Decoder Block Diagrams of a Lossy Predictive Coding System

Overview of predictive coding II

- In addition to transform coding, predictive coding is another important technique for image and video coding.
- In fact, temporal predictive coding using motion compensated prediction is the key of the success of modern video coding standards.
- The encoder must repeat the same process as the decoder to reproduce reconstructed samples; this is called closed-loop prediction.
- This kind of coder is generally referred to as differential pulse coded modulation (DPCM).

Overview of predictive coding III

- Error analysis:
  - s original sample value,
  - s_p predicted sample value,
  - $e_p = s - s_p$ the original prediction error,
  - $\hat{e}_p$ the quantized prediction error,
  - $e_q = e_p - \hat{e}_p$ the quantization error for e,
  - the reconstruction $\hat{s}$ for s is:

$$\begin{aligned} \hat{s} &= s_p + \hat{e}_p && (3)\\ &= s_p + e_p - e_q && (4)\\ &= s_p + s - s_p - e_q && (5)\\ &= s - e_q, && (6) \end{aligned}$$

Overview of predictive coding IV

- Therefore, the error between the original and the reconstructed sample value is:

$$s - \hat{s} = e_q,$$

  exactly the same as the quantization error for the prediction error,

- Thus, the distortion in a lossy predictive coder is completely dependent on the quantizer for the prediction error, for a fixed predictor.

Motion compensated temporal prediction (unidirectional) I

- Uni-directional temporal prediction: we predict a pixel value in the current frame from its corresponding pixel in a previous frame.
- Let $\psi(x,t)$ represent the pixel value in frame t at pixel x, and let t^- denote the previous frame time. Then, the prediction process is described by:

$$\psi_p(x,t) = \psi(x,t_-),$$

this is known as linear temporal prediction.

- Note: such type of prediction is effective only if the underlying scene is stationary.
- In a real-world video, the objects in the scene as well as the camera are usually moving.
- In this case, motion-compensated prediction (MCP) is more appropriate, which uses:

$$\psi_p(x,t) = \psi(x+d(x),t_-),$$

where $d(x)$ represent the motion vector (MV) of pixel x from time t to time t_-.

Motion compensated temporal prediction (unidirectional) II

- Recall: frame $\psi(x,t)$ is the current frame, and frame $\psi(x,t_-)$ is the reference frame, and $\psi_p(x,t)$ is the predicted frame.
- Remember: the reference frame must be coded and reconstructed before the current frame.
- Theoretically, using pixels from more than one previous frame can improve prediction accuracy.

Motion compensated temporal prediction (bidirectional) I

- In bidirectional temporal prediction, a pixel in a current frame is predicted from a pixel in a previous frame t_- as well as a pixel in a following frame t_+.
- The predicted value at frame t is described by:

$$\psi_p(x,t) = a^- \, \psi(x + d^-(x), t_-) + a^+ \, \psi(x + d^+(x), t^+),$$

  where $d^-(x)$ and $d^+(x)$ represent the MV at x from t to t^- and that from t to t^+.
- Typically, we call:
  - the prediction of the current frame from a previous ($t_- < t$) reference frame forward motion compensation,
  - the prediction of the current frame from a future ($t^+ > t$) reference frame backward motion compensation.

Motion compensated temporal prediction (bidirectional) II

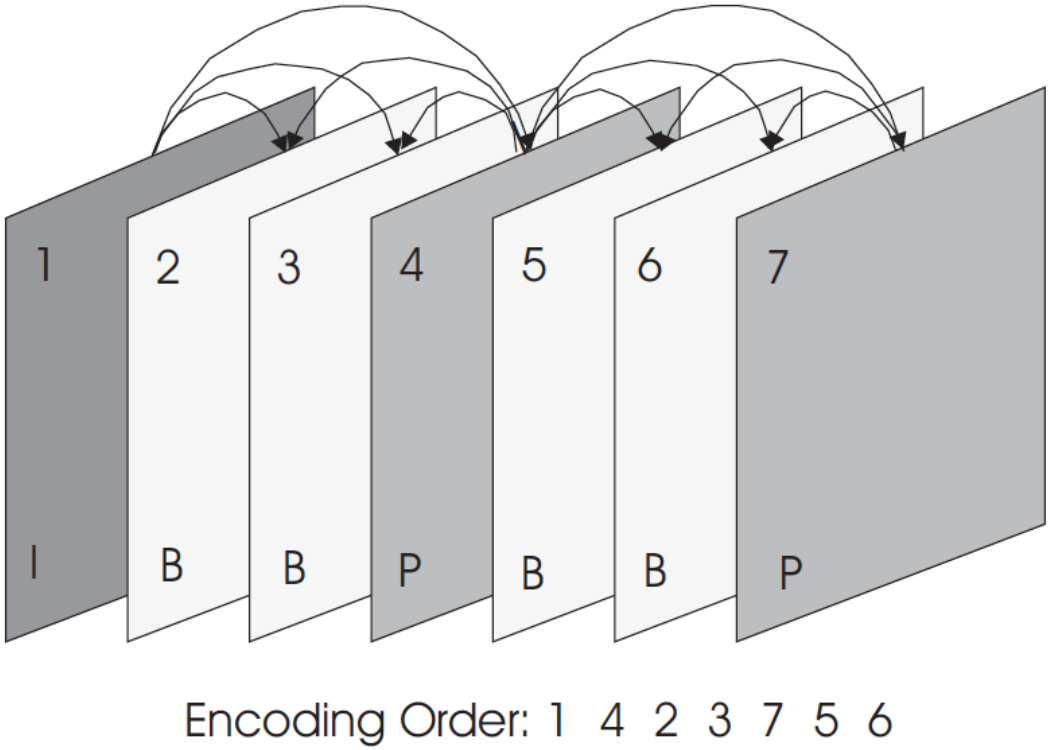

Figure 9.12. Video coding using both uni-directional and bi-directional temporal prediction. The arrows indicate the reference frames used for predicting a coded frame. Frames labeled I, P, and B are coded without prediction, with uni-directional prediction, and with bi-directional prediction, respectively.

Motion compensated temporal prediction (bidirectional) III

- Note that the use of bi-directional prediction needs the coding of frames in an order that is different from the original temporal order.
- Bi-directional prediction implies encoding delay and is typically not used in real-time applications (video phone or video conferencing).
- The MPEG standard series, targeted mainly for video distribution, employ both uni- and bi-directional prediction.

Outline

Introduction I

- Scalability is the capability of recovering physically meaningful image or video information from partial compressed bitstreams,
- Example: we would that an user with high bandwidth connection can download the entire bitstream to view the full quality video, while the user with a low-bandwidth connection will only download a subset of the stream, and see a low quality video.
- Scalable coders can have coarse granularity (two or three layers), or fine granularity,
- In the extreme case of fine granularity, the bit stream can be truncated at any point.
  - The more bits are retained, the better will be the reconstructed image quality.
  - We call such type of bitstream embedded.

Introduction II

- Scalable coding is typically accomplished by providing multiple versions of a video either in terms of:
  - amplitude resolutions (called quality scalability or SNR scalability),
  - temporal resolutions (temporal scalability),
  - frequency resolution (frequency scalability),
  - a combination of these options.
- When scalable contents can be accessed at object level, we call this object-based scalability (MPEG4).
- Simulcast simply codes the same video with different resolutions,
- This method is simple but not efficient: it encodes several times the same information.

Quality scalability I

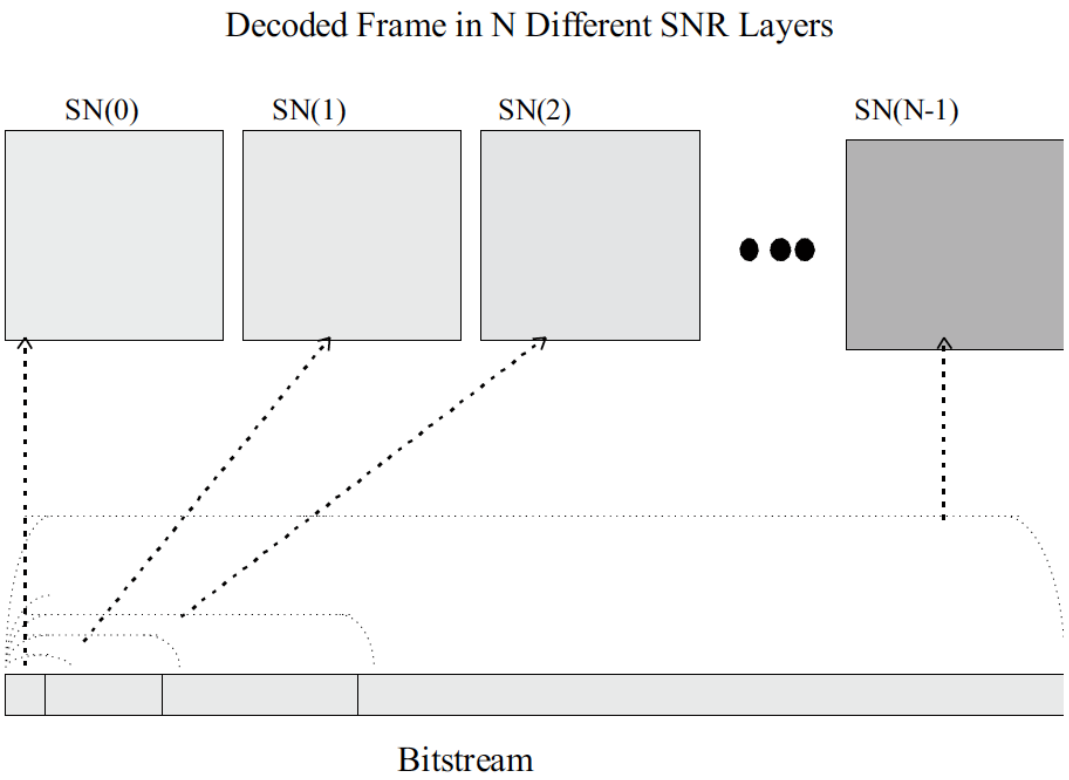

Figure: Quality scalability

Quality scalability II

- Decoding the first layer (also called base layer) provides a low quality version of the reconstructed image.
- Further decoding the remaining layers (also called enhancement layers) results in a quality increase of the reconstructed image up to the highest quality.
- The first layer is obtained by applying a coarse quantizer to the original image or in a transform (e.g., DCT) domain.
- The second layer contains the quantized difference between the original image and that reconstructed from the first layer,
- This quantizer that is finer than that used to produce the first layer.
- And we continue this way using increasingly finer quantizers.

Quality scalability III

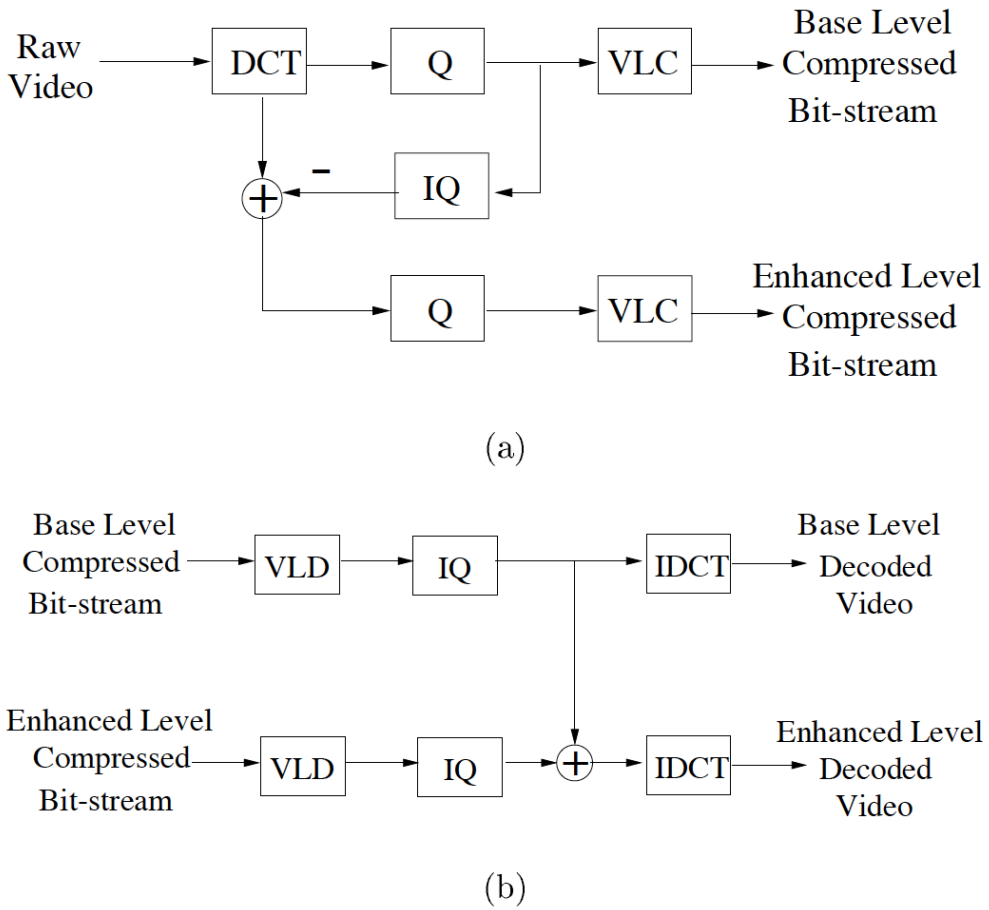

Figure 11.3. A two level quality-scalable codec. (a) encoder; (b) decoder.

Spatial scalability I

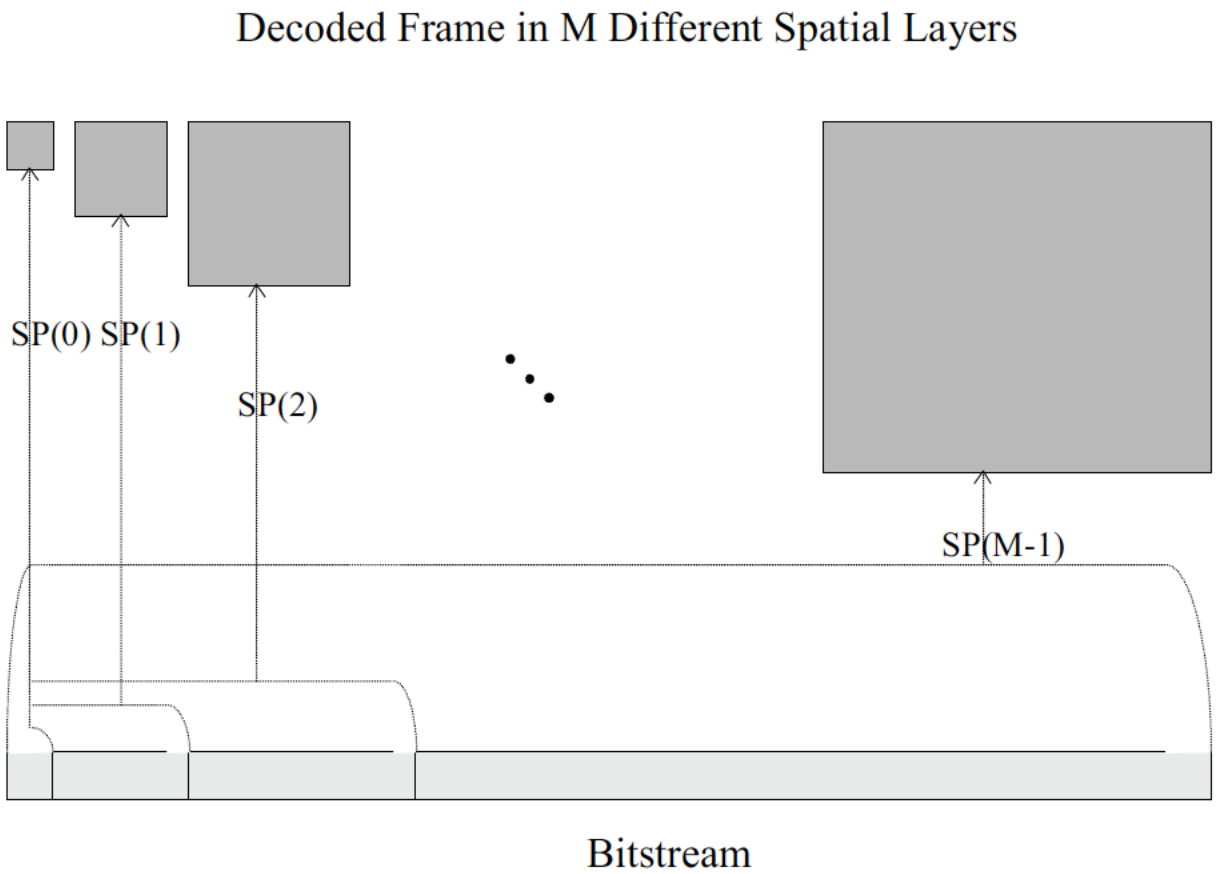

Figure: Spatial scalability

Spatial scalability II

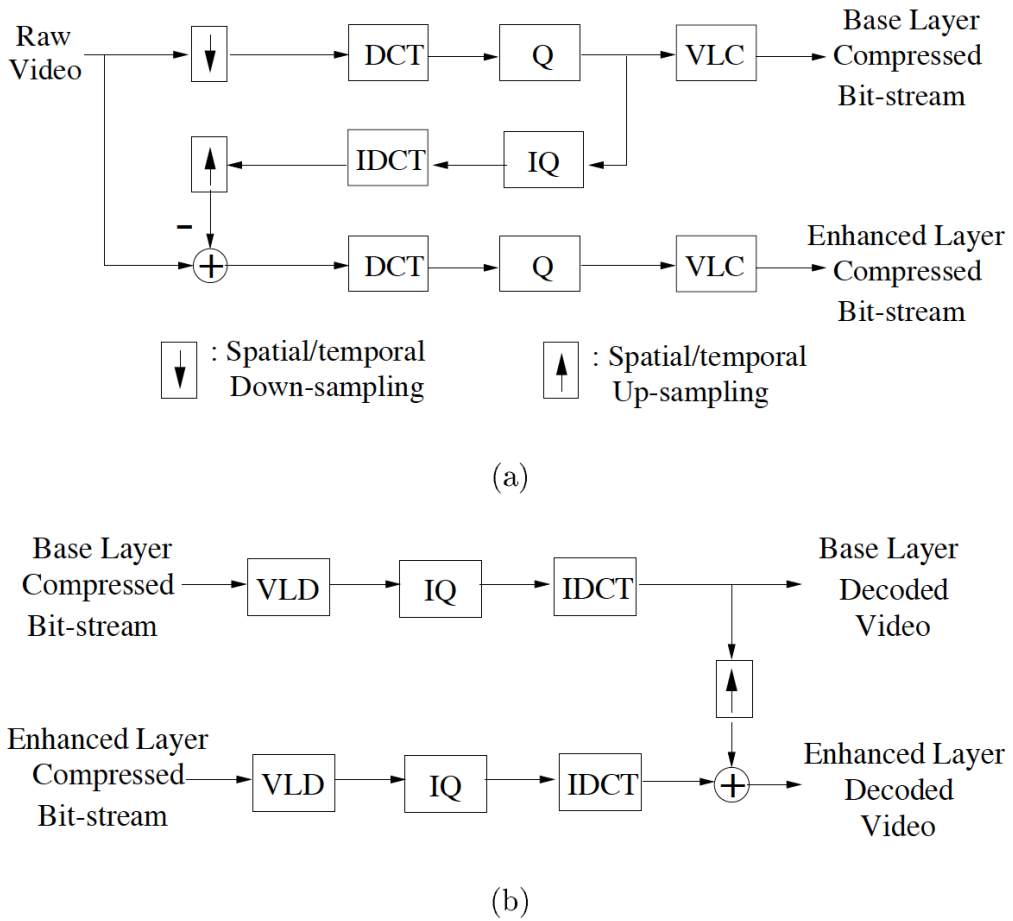

Figure 11.6. A two level spatially/temporally scalable codec. (a) encoder; (b) decoder.

Spatial scalability III

- Spatial scalability is defined as representing the same video in different spatial resolutions or sizes.
- By decoding the first layer, the user can display a preview version of the decoded image at a lower resolution.
- Decoding the second layer results in a larger reconstructed image.
- By progressively decoding the additional layers, the viewer can increase the spatial resolution of the image up to the full resolution of the original image.
- To produce such a layered bit stream, we must compute a multi-resolution decomposition of the original image.

Temporal scalability I

- Temporal scalability is defined as representing the same video in different temporal resolutions or frame rates.
- Temporal scalability enables different frame rates for different layers of the contents.
- The block diagram of temporally scalable codec is the same as that of spatially scalable codec.
- The simplest temporal down-sampling is by frame skipping.
- Temporal up-sampling can be accomplished by frame copying.
- Note that the reasoning is different in space and in time due to the different perceptions!

Frequency scalability I

- We include different frequency components in each layer.
- The base layer contains low frequency components,
- The other layers contain increasingly higher frequency components.
- This way, the base layer will provide a blurred version of the image, and the addition of enhancement layers will yield increasingly sharper images.
- Whole-frame transforms: subband decompositions, wavelet transforms.
- Block-based transforms: block DCT.

Combination of basic schemes I

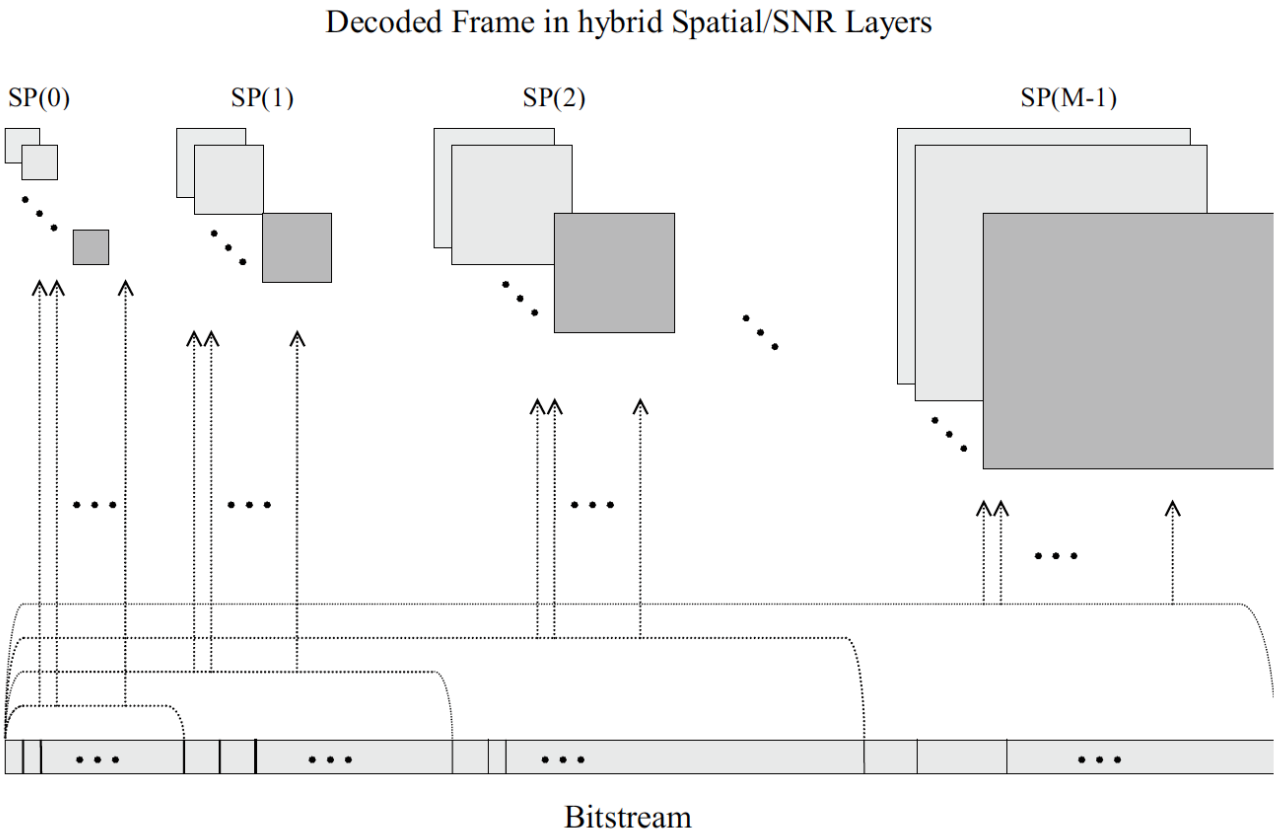

Figure 11.7. NxM layers of combined spatial/quality scalability. From [20, Fig. 3].

Combination of basic schemes II

- Quality, spatial, temporal and frequency scalabilities are basic scalable mechanisms.
- They can be combined to reach finer granularity.
- For example:

1. First we improve the image quality at a given spatial resolution,
2. Then we refine until the best quality is achieved at this spatial resolution,
3. Then we increase the spatial resolution to a higher level ... (and so on!)

Fine granularity scalability (FGS) I

- Fine granularity scalability refers to a coding method by which the rate as well as quality increment at a much smaller step.
- When a bitstream can provide continuously improving video quality with every additional bit, the underlying coding method is called embedded coding.
- Obviously, FGS and embedded coding can adapt to bandwidth variations in real networks more effectively.
- In practice, a base layer is first produced to provide a low but guaranteed level of quality,
- Then an enhancement layer may be generated to provide improvement in fine granularity.
- Example: MPEG-4.

Object-based scalability I

- Object temporal scalability:
- The frame rate of the object is higher that that of the remaining area.

Object-based scalability II

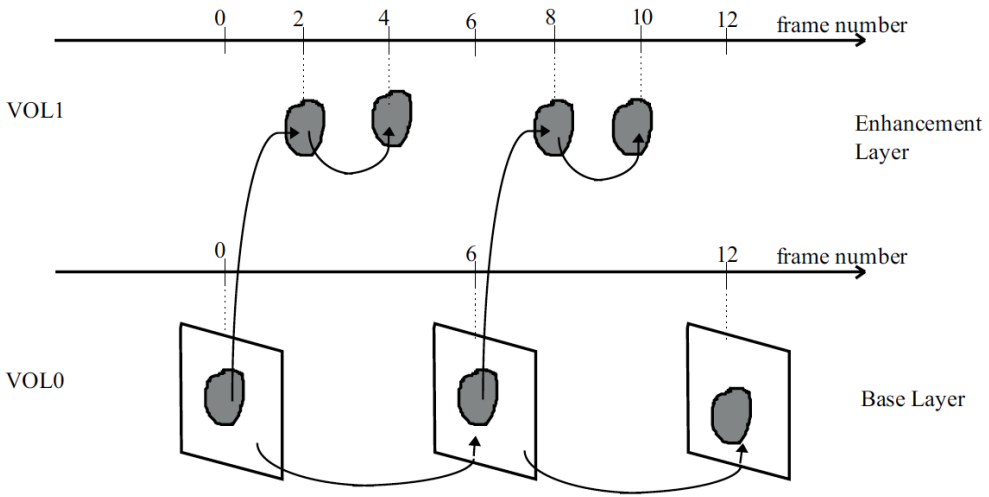

Figure 11.9. Enhancement structure of Type 1 with P-VOPs (Courtsey of MPEG4)

Object-based scalability III

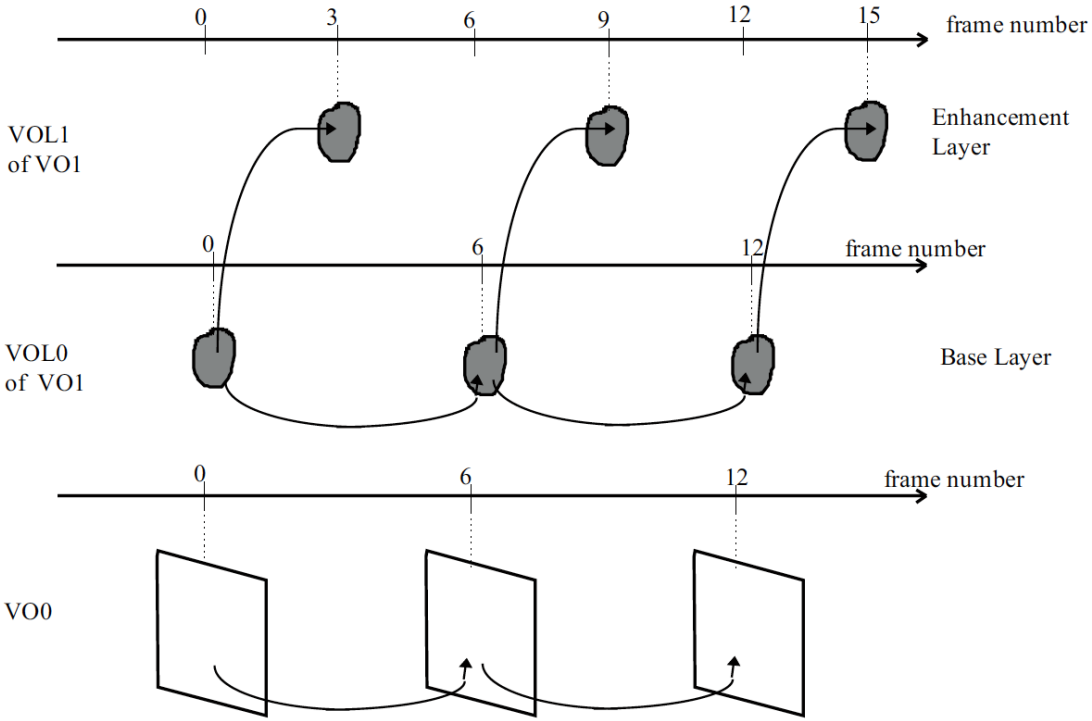

Figure 11.11. Enhancement structure of Type 2 (Courtesy of MPEG4)

Wavelet transform based coding (REMINDER) I

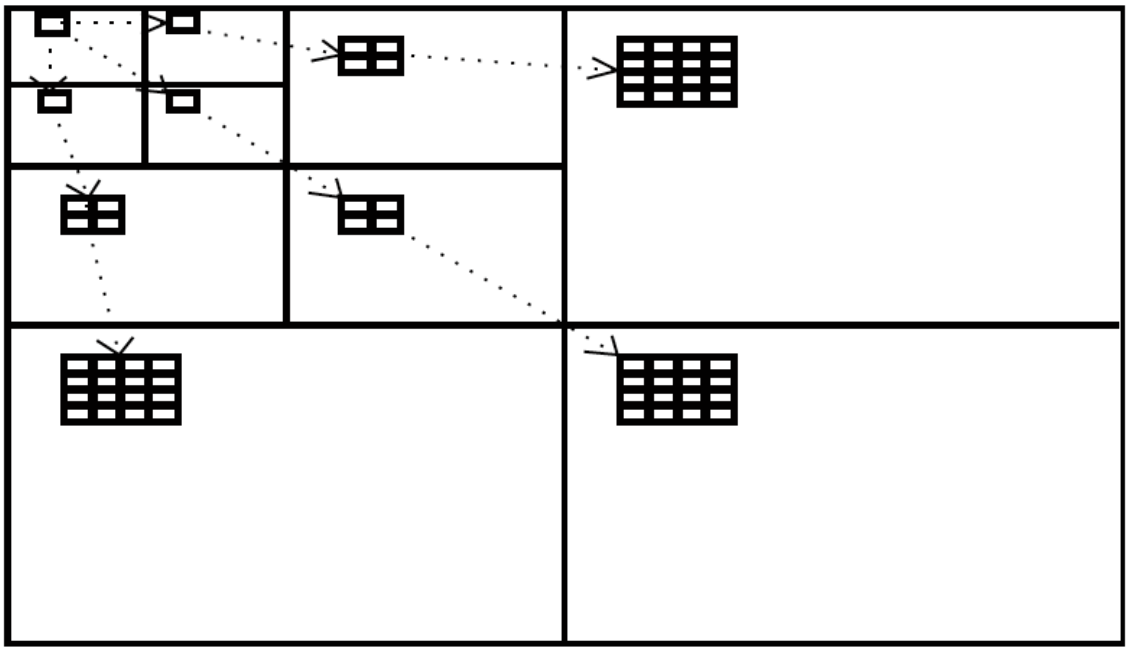

Figure 11.12. The parent-child relationship of wavelet coefficients. From [20, Fig. 4].

Wavelet transform based coding (REMINDER) II

- The discrete wavelet transform (DWT) provides a mutliresolution/multifrequency expression of a signal with localization in both time and frequency.
- The multiresolution/multifrequency decomposition offered by the wavelet transform lends itself easily to a scalable bit stream.
- Like DCT-based approach, wavelet transform based coding for images consists of threes steps:
  - (1) wavelet transform,
  - (2) quantization,
  - (3) entropy coding.
- The results in matter of compression are relatively the same as the MPEG-4's DCT-based coder (in terms of PSNR).
- At the end, it has been shown that the optimization of the whole framework (coding, ...) is more important that optimizing the transform itself.

Outline

Overview of a video coding system I

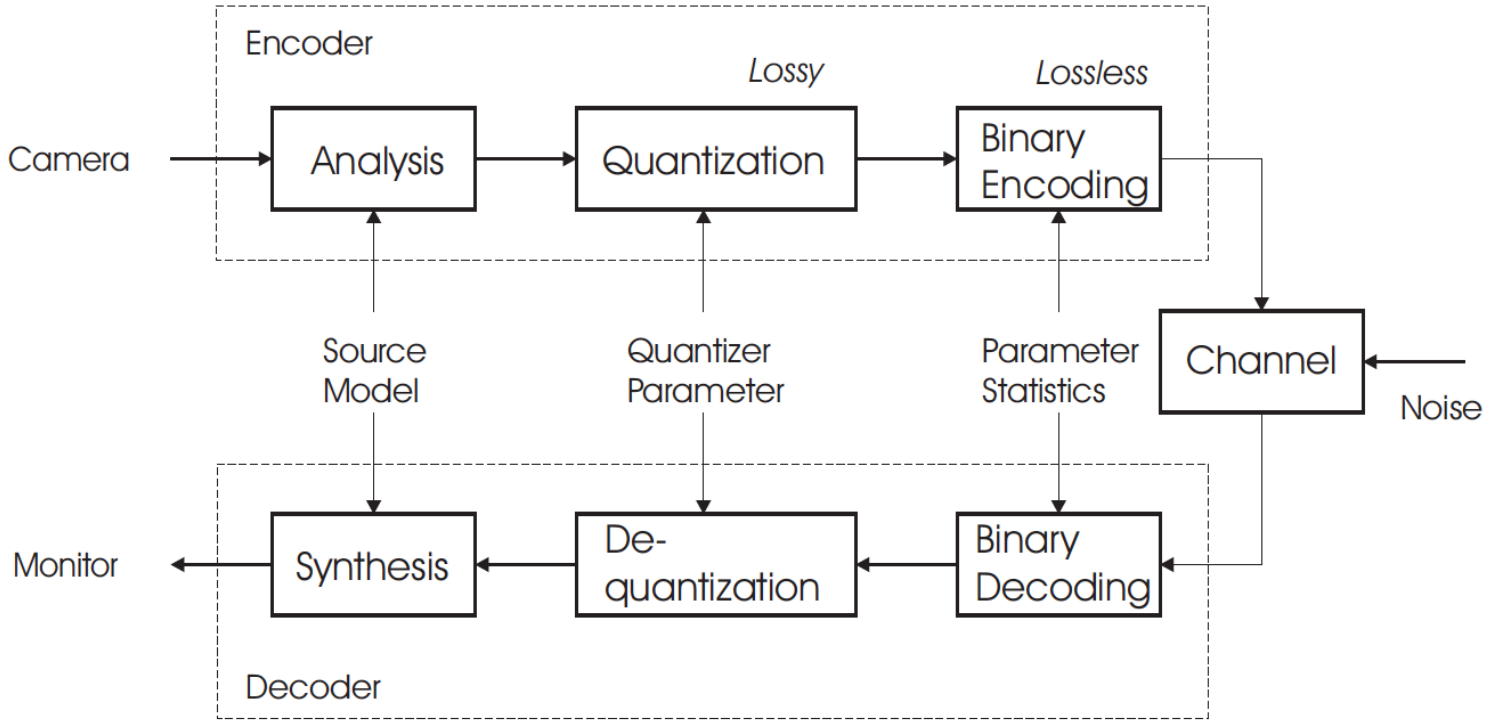

Figure 8.1. Overview of a video coding system.

References


Indrajit Chakrabarti, Kota Naga Srinivasarao Batta, S. K. C. (2015).
Motion Estimation for Video Coding.
Springer.


Jie Chen, Ut-Va Koc, K. R. L. (2002).
Design of Digital Video Coding Systems.
Signal Processing and Communication Series, Marcel Dekker, Inc.


Yao Wang, Jorn Ostermann, Y.-Q. Z. (2002).
Video Processing and Communications.
Prentice Hall.