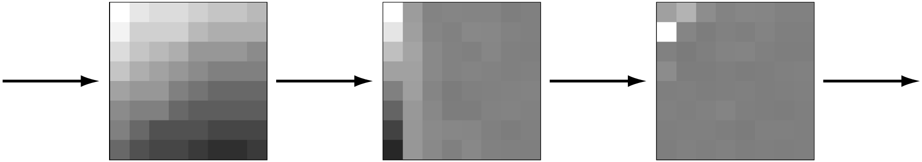
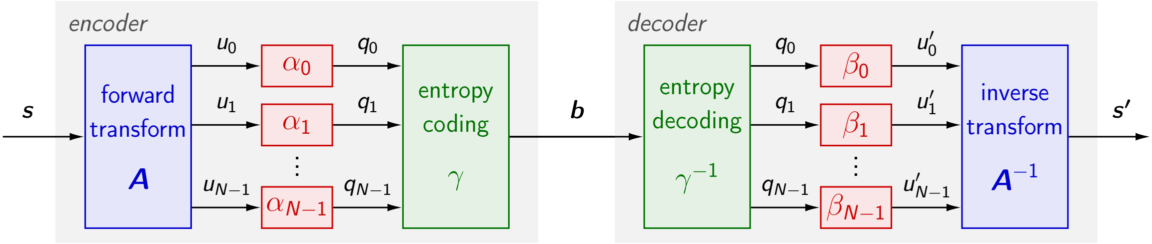
Transform Coding in Practice



Last Lectures: Basic Concept Transform Coding

- Transform reduces linear dependencies (correlation) between samples before scalar quantization
- For correlated sources: Scalar quantization in transform domain is more efficient



Encoder (block-wise)

- ➡ Forward transform: $\boldsymbol{u} = \boldsymbol{A} \cdot \boldsymbol{s}$
- ➡ Scalar quantization: $q_k = \alpha_k(u_k)$
- ➡ Entropy coding: $\boldsymbol{b} = \gamma(\{q_k\})$

Decoder (block-wise)

- ➡ Entropy decoding: $\{q_k\} = \gamma^{-1}(\boldsymbol{b})$
- ➡ Inverse quantization: $u'_k = \beta_k(q_k)$
- ➡ Inverse transform: $\boldsymbol{s'} = \boldsymbol{A}^{-1} \cdot \boldsymbol{u'}$

Last Lectures: Orthogonal Block Transforms

- Transform matrix has property: $\boldsymbol{A}^{-1} = \boldsymbol{A}^{\mathrm{T}}$ (special case of unitary matrix: $\boldsymbol{A}^{-1} = (\boldsymbol{A}^{*})^{\mathrm{T}}$)

$$\boldsymbol{A} = \begin{bmatrix} \text{---} & \boldsymbol{b}_0 & \text{---} \\ \text{---} & \boldsymbol{b}_1 & \text{---} \\ \text{---} & \boldsymbol{b}_2 & \text{---} \\ & \vdots & \\ \text{---} & \boldsymbol{b}_{N-1} & \text{---} \end{bmatrix} \qquad \boldsymbol{A}^{-1} = \boldsymbol{A}^{\mathrm{T}} = \begin{bmatrix} | & | & | & & | \\ \boldsymbol{b}_0 & \boldsymbol{b}_1 & \boldsymbol{b}_2 & \cdots & \boldsymbol{b}_{N-1} \\ | & | & | & & | \end{bmatrix}$$

➡ Basis vectors $\boldsymbol{b}_k$ (rows of $\boldsymbol{A}$, columns of $\boldsymbol{A}^{-1} = \boldsymbol{A}^{\mathrm{T}}$) form an orthonormal basis

➡ Geometric interpretation: Rotation (and potential reflection) in N-dimensional signal space

Why Orthogonal Transforms ?

- Same MSE distortion in sample and transform space: $||\boldsymbol{u}' - \boldsymbol{u}||_2^2 = ||\boldsymbol{s}' - \boldsymbol{s}||_2^2$

➡ **Minimum MSE in signal space can be achieved by minimization of MSE for each individual transform coefficient**

Last Lectures: Bit Allocation and High-Rate Approximations

Bit Allocation of Transform Coefficients

- Optimal bit allocation: Pareto condition

$$\frac{\partial}{\partial R_k} D_k(R_k) = -\lambda = \text{const} \qquad \Longrightarrow \qquad \text{high rates:} \quad D_k(R_k) = \text{const}$$

High-Rate Approximation

- High-rate distortion rate function for transform coding with optimal bit allocation

$$D(R) = \tilde{\varepsilon}^2 \cdot \tilde{\sigma}^2 \cdot 2^{-2R} \qquad \text{with} \qquad \tilde{\varepsilon}^2 = \Big(\prod_k \varepsilon_k^2\Big)^{\frac{1}{N}}, \quad \tilde{\sigma}^2 = \Big(\prod_k \sigma_k^2\Big)^{\frac{1}{N}}$$

- High-rate transform coding gain G_T and energy compaction measure G_{EC}

$$G_T = \frac{D_{SQ}(R)}{D_{TC}(R)} = \frac{\varepsilon_S^2 \cdot \sigma_S^2}{\tilde{\varepsilon}^2 \cdot \tilde{\sigma}^2}, \qquad G_{EC} = \frac{\sigma_S^2}{\tilde{\sigma}^2} = \frac{\frac{1}{N}\sum_{k=0}^{N-1} \sigma_k^2}{\sqrt[N]{\prod_{k=0}^{N-1} \sigma_k^2}}$$

Last Lectures: Karhunen Loève Transform (KLT)

- Design criterion: Orthogonal transform $\boldsymbol{A}$ that yields uncorrelated transform coefficients

$$\boldsymbol{C}_{UU} = \boldsymbol{A} \cdot \boldsymbol{C}_{SS} \cdot \boldsymbol{A}^{\mathrm{T}} = \begin{bmatrix} \sigma_0^2 & 0 & \cdots & 0 \\ 0 & \sigma_1^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{N-1}^2 \end{bmatrix} \quad \Longrightarrow \quad \boldsymbol{C}_{SS} \cdot \boldsymbol{b}_k = \sigma_k^2 \cdot \boldsymbol{b}_k$$

➡ Eigenvector equation for all basis vectors $\boldsymbol{b}_k$ (rows of transform matrix $\boldsymbol{A}$)

➡ Rows of KLT matrix $\boldsymbol{A}$ are the unit-norm eigenvectors of $\boldsymbol{C}_{SS}$

➡ Transform coefficient variances σ_k^2 are the eigenvalues of $\boldsymbol{C}_{SS}$

$$\boldsymbol{A} = \begin{bmatrix} — & \boldsymbol{b}_0 & — \\ — & \boldsymbol{b}_1 & — \\ & \vdots & \\ — & \boldsymbol{b}_{N-1} & — \end{bmatrix} \qquad \boldsymbol{C}_{UU} = \begin{bmatrix} \sigma_0^2 & 0 & \cdots & 0 \\ 0 & \sigma_1^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{N-1}^2 \end{bmatrix}$$

Last Lectures: Maximum Energy Compaction and Optimality

High-Rate Approximation for KLT and Gauss-Markov

- High-rate operational distortion-rate function

$$D_N(R) = \varepsilon^2 \cdot \sigma_S^2 \cdot (1 - \varrho^2)^{\frac{N-1}{N}} \cdot 2^{-2R}$$

➡ High-rate transform coding gain: Increases with transform size N

$$G_T^N = G_{EC}^N = (1 - \varrho^2)^{\frac{1-N}{N}} \qquad \Longrightarrow \qquad G_T^\infty = \frac{1}{1 - \varrho^2}$$

➡ For $N \to \infty$, gap to fundamental lower bound reduces to space-filling gain (1.53 dB)

On Optimality of KLT

- KLT yields uncorrelated transform coefficients and maximizes energy compaction G_{EC}

➡ KLT is the optimal transform for stationary Gaussian sources

- Other sources: Optimal transform is hard to find (iterative algorithm)

Transform Selection in Practice

Optimal Unitary Transform

- Stationary Gaussian sources: KLT
- General sources: Not straightforward to determine (typically KLT close to optimal)

➡ Signal dependent (may change due to signal instationarities)

Adaptive Transform Selection

- Determine transform in encoder, include transform specification in bitstream

➡ Increased side information may lead to sub-optimal overall coding efficiency

➡ Simple variant: Switched transforms (e.g., in H.266/VVC)

Signal-Independent Transforms

- Choose transform that provides good performance for variety of signals

➡ Not optimal, but often close to optimal for typical signal

➡ Most often used design in practice

Walsh-Hadamard Transform

- For transform sizes N that are positive integer powers of 2

$$\boldsymbol{A}_N = \frac{1}{\sqrt{2}} \begin{bmatrix} \boldsymbol{A}_{N/2} & \boldsymbol{A}_{N/2} \\ \boldsymbol{A}_{N/2} & -\boldsymbol{A}_{N/2} \end{bmatrix} \qquad \text{with} \qquad \boldsymbol{A}_1 = \begin{bmatrix} 1 \end{bmatrix}.$$

- Examples: Transform matrices for $N = 2$, $N = 4$, and $N = 8$

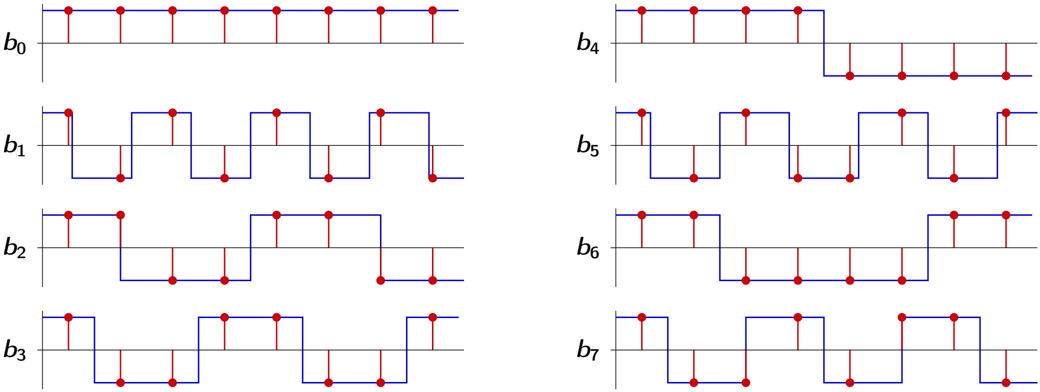
$$\boldsymbol{A}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\boldsymbol{A}_4 = \frac{1}{\sqrt{4}} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

$$\boldsymbol{A}_8 = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

➡ Very simple orthogonal transform (only additions, subtractions, and final scaling)

Basis Functions of the WHT (Example for $N = 8$)



Media coding: Walsh-Hadamard transform with strong quantization

➡ Piece-wise constant basis vectors yield subjectively disturbing artifacts

Discrete Version of the Fourier Transform

The Fourier Transform

- Fundamental transform used in mathematics, physics, signal processing, communications, ...
- Integral transform representing signal as integral of frequency components
- Forward and inverse transform are given by

$$X(f) = \mathcal{F}\{x(t)\} = \int_{-\infty}^{\infty} x(t) \cdot e^{-2\pi i f t} \, \mathrm{d}t \quad \Longleftrightarrow \quad x(t) = \mathcal{F}^{-1}\{x(t)\} = \int_{-\infty}^{\infty} X(f) \cdot e^{2\pi i f t} \, \mathrm{d}f$$

➡ Basis functions are complex exponentials $b_f(t) = e^{2\pi i f t}$

Discrete Version of the Fourier Transform

- Fourier transform for finite discrete signals
- Could also be useful for coding of discrete signals
- Can be derived using sampling and windowing

Important Properties of the Fourier Transform

- Linearity: $$\mathcal{F}\Big\{ a \cdot h(t) + b \cdot g(t) \Big\} \;=\; a \cdot H(f) + b \cdot G(f)$$
- Scaling: $$\mathcal{F}\Big\{ h(a \cdot t) \Big\} \;=\; \frac{1}{|a|} \cdot H\left(\frac{f}{a}\right)$$
- Translation: $$\mathcal{F}\Big\{ h(t - t_0) \Big\} \;=\; e^{-2\pi \mathrm{i} t_0 f} \cdot H(f)$$
- Modulation: $$\mathcal{F}\Big\{ e^{2\pi \mathrm{i} t f_0} \cdot h(t) \Big\} \;=\; H(f - f_0)$$
- Duality: $$\mathcal{F}\Big\{ H(t) \Big\} \;=\; h(-f)$$
- Convolution: $$\mathcal{F}\Big\{ h(t) * g(t) \Big\} = \mathcal{F}\left\{ \int_{-\infty}^{\infty} g(\tau)\, h(t-\tau)\, \mathrm{d}\tau \right\} \;=\; H(f) \cdot G(f)$$
- Multiplication: $$\mathcal{F}\Big\{ h(t) \cdot g(t) \Big\} \;=\; H(f) * G(f)$$

The Dirac Delta Function

Dirac Delta Function

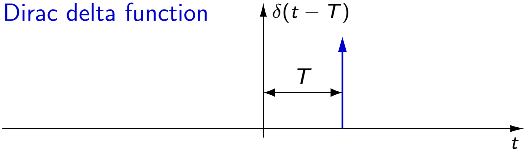
- Not a function in traditional sense ➜ **Dirac delta distribution**
- Can be thought of function with the following properties

$$\delta(x) = \begin{cases} +\infty & : \quad x = 0 \\ 0 & : \quad x \neq 0 \end{cases} \qquad \text{and} \qquad \int_{-\infty}^{\infty} \delta(x)\,\mathrm{d}x = 1$$

Important Properties

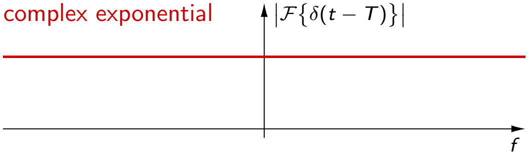
- Sifting:
$$\int_{-\infty}^{\infty} h(t)\,\delta(t-t_0)\,\mathrm{d}t \;=\; h(t_0)$$
- Convolution:
$$h(t) * \delta(t-t_0) \;=\; \int_{-\infty}^{\infty} h(\tau)\,\delta(t-t_0-\tau)\,\mathrm{d}\tau = h(t-t_0)$$
- Sampling:
$$\int_{-\infty}^{\infty} h(t)\left(\sum_{k=-\infty}^{\infty} \delta(t-k\cdot t_0)\right)\mathrm{d}t \;=\; \sum_{k=-\infty}^{\infty} h(k\cdot t_0)$$

Selected Fourier Transform Pairs

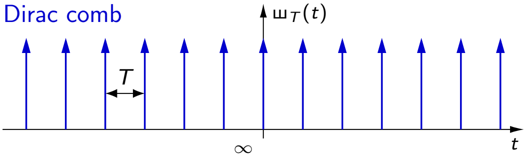


$$x(t) = \delta(t - T)$$

○—●

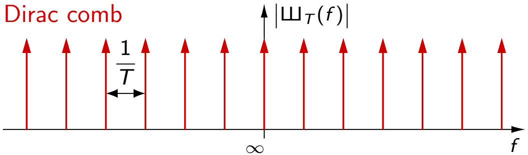


$$X(f) = e^{-2\pi \mathrm{i} f T} = \cos(2\pi f T) + \mathrm{i}\sin(2\pi f T)$$



$$\text{ш}_T(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT)$$

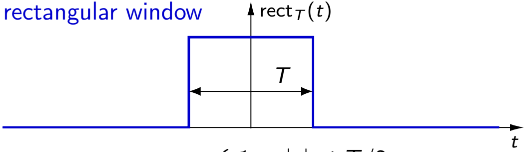
○—●



$$\text{Ш}_T(f) = \sum_{k=-\infty}^{\infty} \delta(f - k/T)$$

Selected Fourier Transform Pairs

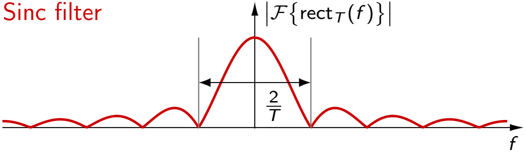
rectangular window



$$\text{rect}_T(t) = \begin{cases} 1 & : \; |t| \le T/2 \\ 0 & : \; |t| > T/2 \end{cases}$$

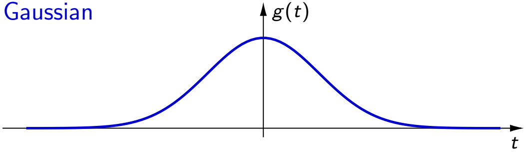
○—●

Sinc filter



$$\mathcal{F}\{\text{rect}_T(f)\} = \frac{1}{\pi f}\sin(\pi f T) = T\,\text{sinc}(fT)$$

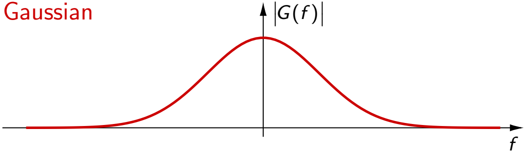
Gaussian



$$g(t) = e^{-\pi \cdot t^2} \quad \text{with} \quad \sigma_t^2 = \frac{1}{2\pi}$$

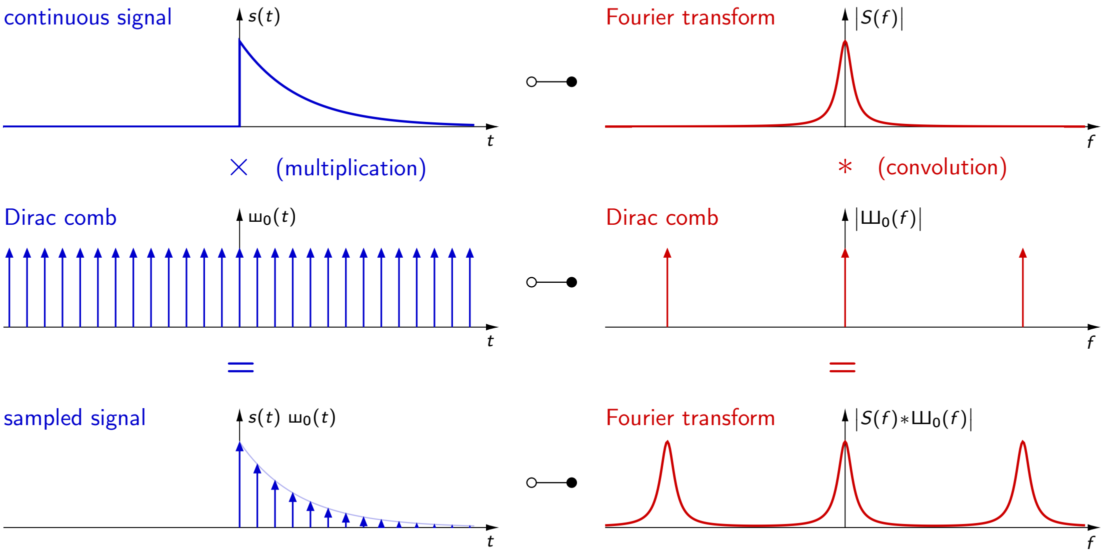
○—●

Gaussian

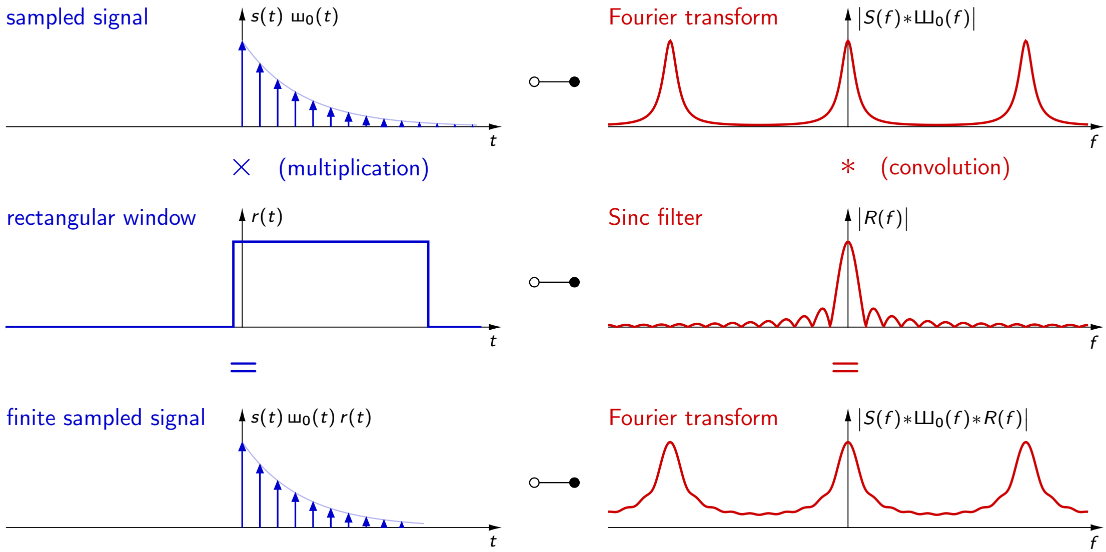


$$G(f) = e^{-\pi \cdot f^2} = g(f)$$

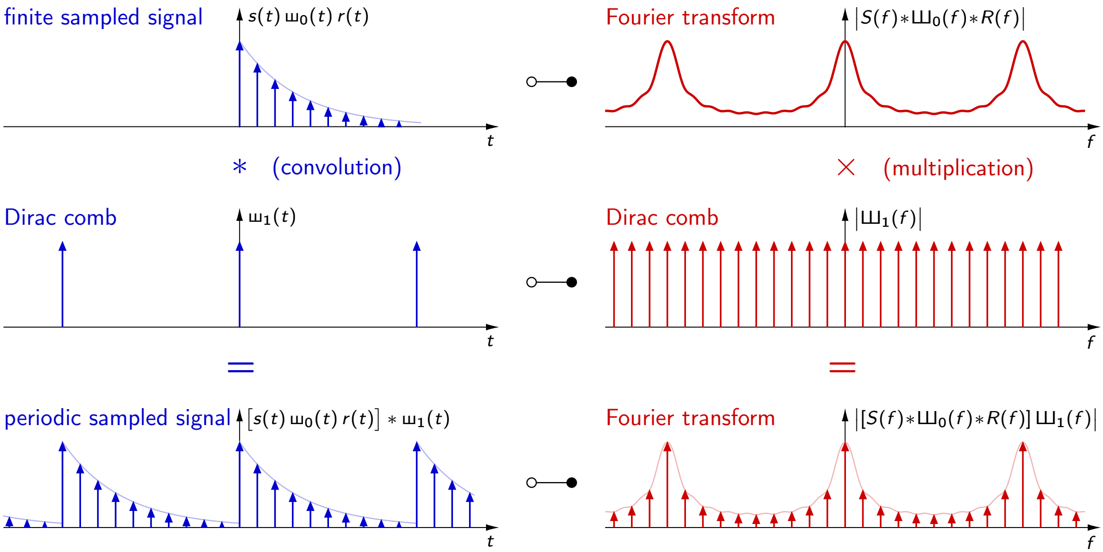
Derivation of Discrete Fourier Transform: (1) Sampling of Signal



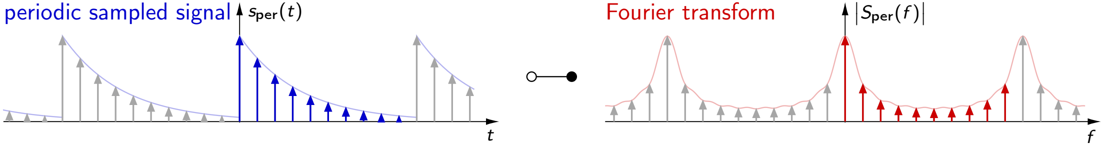
Derivation of Discrete Fourier Transform: (2) Time Restriction



Derivation of Discrete Fourier Transform: (3) Sampling of Spectrum



The Discrete Fourier Transform



➡ N samples are represented by N complex Fourier coefficients

Discrete Fourier Transform

- Forward and inverse transform are given by

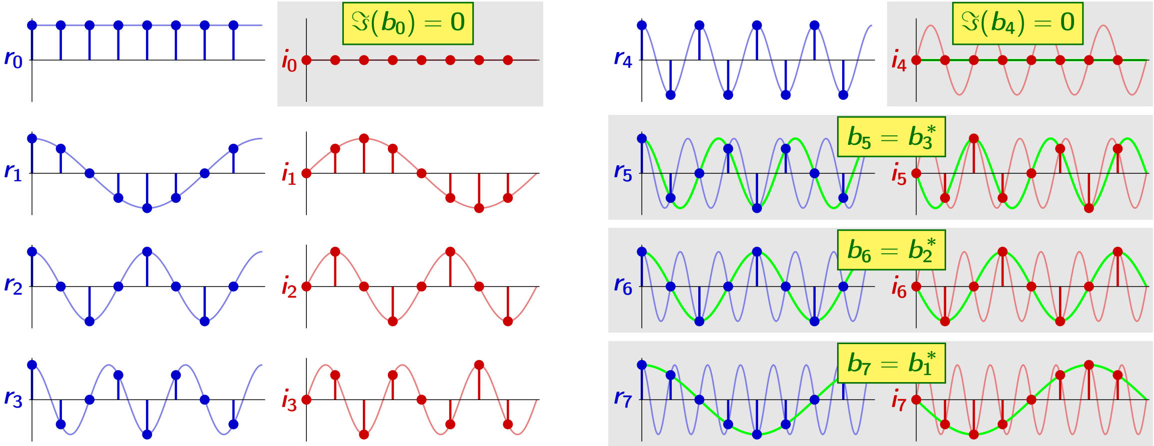
$$u[k] = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} s[n] \cdot e^{-\mathrm{i}\frac{2\pi kn}{N}} \qquad \text{and} \qquad s[n] = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} u[k] \cdot e^{\mathrm{i}\frac{2\pi kn}{N}}$$

➡ Unitary transform that produces complex transform coefficients

➡ Basis vectors are sampled complex exponentials

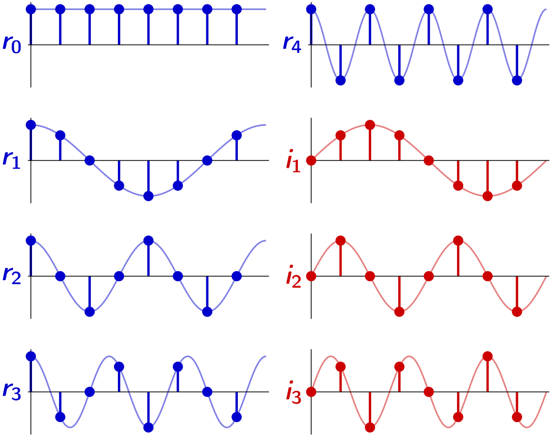
Complex Basis Functions of the DFT (Example for $N = 8$)

$$b_k[n] = \frac{1}{\sqrt{N}}\, e^{\mathrm{i}\,\frac{2\pi k}{N} n} = \frac{1}{\sqrt{N}} \cos\left(\frac{2\pi k}{N} n\right) + \mathrm{i}\cdot\frac{1}{\sqrt{N}} \sin\left(\frac{2\pi k}{N} n\right) = r_k[n] + \mathrm{i}\cdot i_k[n]$$



Complex Basis Functions of the DFT (Example for $N = 8$)

$$\boldsymbol{b_k}[n] = \frac{1}{\sqrt{N}}\, e^{\mathrm{i}\,\frac{2\pi k}{N} n} = \frac{1}{\sqrt{N}} \cos\left(\frac{2\pi k}{N} n\right) + \mathrm{i}\cdot \frac{1}{\sqrt{N}} \sin\left(\frac{2\pi k}{N} n\right) = \boldsymbol{r_k}[n] + \mathrm{i}\cdot \boldsymbol{i_k}[n]$$



DFT for Real Signals

- Symmetry of complex coefficients

$$u[k] = u^*[N-k]$$

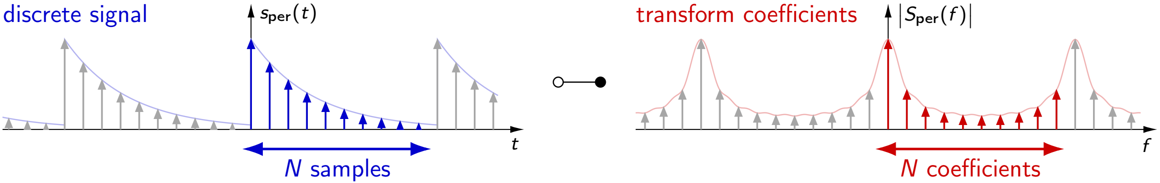
- Vanishing imaginary parts

$$k \in \left\{0, \tfrac{N}{2}\right\}: \quad \Im\{u[k]\} = 0$$

➡ ***N* real samples are mapped to *N* real coefficients**

- Fast algorithm:
  Fast Fourier transform (FFT)

Disadvantage of DFT for Transform Coding



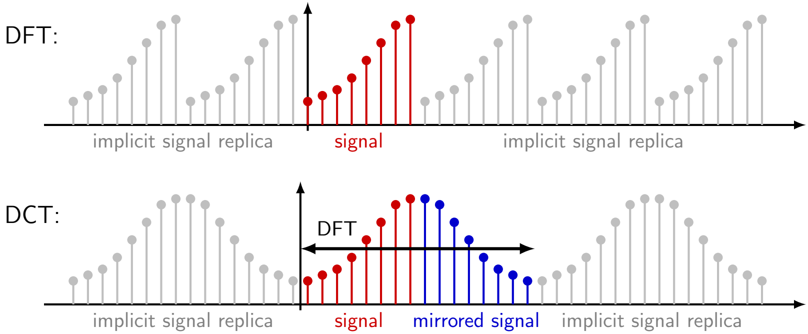
➡ Sampling of frequency spectrum causes **implicit periodic signal extension**

➡ Often: Large differences between left and right signal boundary

➡ Large difference reduces rate of convergence of Fourier series

➡ **Strong quantization yields significant high-frequency artefacts**

Overcome DFT Disadvantage: Discrete Cosine Transform



Idea of Discrete Cosine Transform (DCT)

- Introduce mirror symmetry (different possibilities)
- Apply DFT of approximately double size (or four times the size)

➡ **No discontinuities in periodic signal extension**

➡ Ensure symmetry around zero: **Only cosine terms**

Discrete Trigonometric Transforms (DTTs)

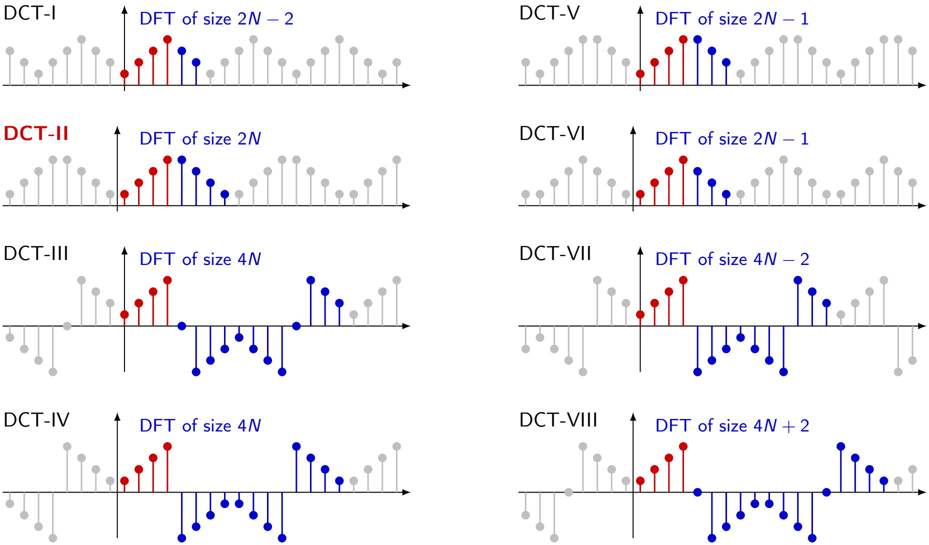
Discrete Cosine Transforms (DCTs)

- Introduce mirror symmetry around zero and apply DFT of larger size
  - ➡ Imaginary sine terms get eliminated
  - ➡ Only cosine terms remain
- 8 possibilities: DCT-I to DCT-VIII
  - 2 cases for left side: Whole sample or half-sample symmetry
  - 4 cases for right side: Whole sample or half-sample symmetry or anti-symmetry
- Most relevant case: **DCT-II** (half-sample symmetry at both sides)

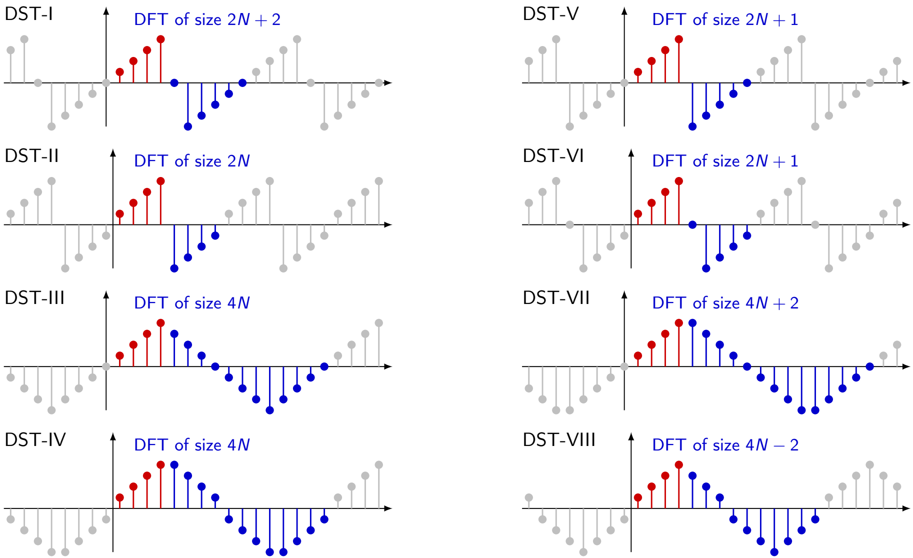
Discrete Sine Transforms (DSTs)

- Introduce anti-symmetry around zero and apply DFT of larger size
  - ➡ Real cosine terms get eliminated
  - ➡ Only imaginary sine terms remain
- Similarly as for DCT: 8 possibilities (DST-I to DST-VIII)

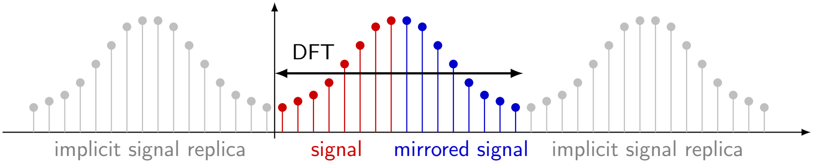
The Discrete Cosine Transform (DCT) Family



The Discrete Sine Transform (DST) Family



Derivation of the Discrete Cosine Transform of Type II (DCT-II)



Signal for applying the DFT

- Given: Discrete signal $s[n]$ of size N (i.e., $0 \le n < N$)
- Mirror signal with sample repetition at both sides (size $2N$)

$$s^m[n] = \begin{cases} s[n] & : \quad 0 \le n < N \\ s[2N-n-1] & : \quad N \le n < 2N \end{cases}$$

- Ensure symmetry around zero by adding half-sample shift

$$s^+[n] = s^m[n-1/2] = \begin{cases} s[n-1/2] & : \quad 0 \le n < N \\ s[2N-n-3/2] & : \quad N \le n < 2N \end{cases}$$

➡ **Apply DFT of size $2N$ to new signal $s^+[n]$**

Derivation of the Discrete Cosine Transform of Type II (DCT-II)

$$s^+[n] = \begin{cases} s[n-1/2] & : \quad 0 \leq n < N \\ s[2N-n-3/2] & : \quad N \leq n < 2N \end{cases}$$

➡ **DFT of size $2N$:**

$$u^+[k] = \frac{1}{\sqrt{(2N)}} \sum_{n=0}^{(2N)-1} s^+[n] \cdot e^{-\mathrm{i}\,\frac{2\pi kn}{(2N)}}$$

(s^+ only known at half-sample positions $\rightarrow$ use $m = n - 1/2$)

$$= \frac{1}{\sqrt{2N}} \sum_{m=0}^{2N-1} s^+\left[m+\frac{1}{2}\right] \cdot e^{-\mathrm{i}\,\frac{\pi k}{N}\left(m+\frac{1}{2}\right)}$$

$$= \frac{1}{\sqrt{2N}} \left(\sum_{n=0}^{N-1} s[n] \cdot e^{-\mathrm{i}\,\frac{\pi k}{N}\left(n+\frac{1}{2}\right)} + \sum_{m=N}^{2N-1} s[2N-m-1] \cdot e^{-\mathrm{i}\,\frac{\pi k}{N}\left(m+\frac{1}{2}\right)} \right)$$

$\downarrow n=2N-m-1$

$$= \frac{1}{\sqrt{2N}} \left(\sum_{n=0}^{N-1} s[n] \cdot e^{-\mathrm{i}\,\frac{\pi k}{N}\left(n+\frac{1}{2}\right)} + \sum_{n=0}^{N-1} s[n] \cdot e^{-\mathrm{i}\,\frac{\pi k}{N}\left(2N-n-\frac{1}{2}\right)} \right)$$

Derivation of the Discrete Cosine Transform of Type II (DCT-II)

- Continue derivation

$$u^+[k] = \frac{1}{\sqrt{2N}} \left(\sum_{n=0}^{N-1} s[n] \cdot e^{-\mathrm{i}\frac{\pi k}{N}\left(n+\frac{1}{2}\right)} + \sum_{n=0}^{N-1} s[n] \cdot e^{-\mathrm{i}\frac{\pi k}{N}\left(2N-n-\frac{1}{2}\right)} \right)$$

$$= \frac{1}{\sqrt{2N}} \left(\sum_{n=0}^{N-1} s[n] \cdot e^{-\mathrm{i}\frac{\pi k}{N}\left(n+\frac{1}{2}\right)} + \sum_{n=0}^{N-1} s[n] \cdot \underbrace{e^{-\mathrm{i}2\pi k}}_{1} \cdot e^{\mathrm{i}\frac{\pi k}{N}\left(n+\frac{1}{2}\right)} \right)$$

$$= \frac{1}{\sqrt{2N}} \sum_{n=0}^{N-1} s[n] \cdot \underbrace{\left(e^{-\mathrm{i}\frac{\pi k}{N}\left(n+\frac{1}{2}\right)} + e^{\mathrm{i}\frac{\pi k}{N}\left(n+\frac{1}{2}\right)} \right)}_{2\cos\left(\frac{\pi k}{N}\left(n+\frac{1}{2}\right)\right)}$$

➡ DFT of extended signal

$$u^+[k] = \sqrt{\frac{2}{N}} \cdot \sum_{n=0}^{N-1} s[n] \cdot \cos\left(\frac{\pi}{N}\, k \left(n + \frac{1}{2} \right) \right)$$

Derivation of the Discrete Cosine Transform of Type II (DCT-II)

- DFT of extended signal ($2N$ real samples) has $2N$ real transform coefficients

$$k = 0, \ldots, 2N-1: \qquad u^{+}[k] = \sqrt{\frac{2}{N}} \cdot \sum_{n=0}^{N-1} s[n] \cdot \cos\left(\frac{\pi}{N} \, k \left(n + \frac{1}{2} \right) \right)$$

1. Signal $s[n]$ is completely described by first N transform coefficients

$$k = 0, \ldots, N-1: \qquad u^{+}[k] = \sqrt{\frac{2}{N}} \cdot \sum_{n=0}^{N-1} s[n] \cdot \cos\left(\frac{\pi}{N} \, k \left(n + \frac{1}{2} \right) \right)$$

2. Basis functions of derived transform are orthogonal to each other, but don't have the same norm

➡ Introduce factors α_k so that transform matrix becomes orthogonal

$$k = 0, \ldots, N-1: \qquad u[k] = \alpha_k \cdot \sum_{n=0}^{N-1} s[n] \cdot \cos\left(\frac{\pi}{N} \, k \left(n + \frac{1}{2} \right) \right)$$

Discrete Cosine Transform of Type II (DCT-II)

Specification of DCT-II

- Forward transform (DCT-II) and inverse transform (IDCT-II) are given by

$$u[k] = \alpha_k \sum_{n=0}^{N-1} s[n] \cdot \cos\left(\frac{\pi}{N} \, k \left(n + \frac{1}{2} \right) \right) \quad \text{and} \quad s[n] = \sum_{k=0}^{N-1} \alpha_k \cdot u[k] \cdot \cos\left(\frac{\pi}{N} \, k \left(n + \frac{1}{2} \right) \right)$$

  with scaling factors

$$\alpha_k = \begin{cases} \sqrt{1/N} & : \quad k = 0 \\ \sqrt{2/N} & : \quad k \neq 0 \end{cases}$$

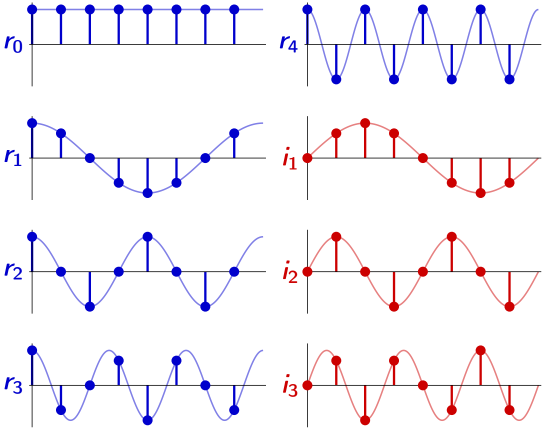
- The orthogonal transform matrix $\boldsymbol{A} = \{a_{kn}\}$ has the elements

$$a_{kn} = \alpha_k \cdot \cos\left(\frac{\pi}{N} \, k \left(n + \frac{1}{2} \right) \right)$$

Comparions of DFT and DCT-II Basis Functions (Example for $N = 8$)

DFT: $\boldsymbol{b_k}[n] = \frac{1}{\sqrt{N}}\, e^{\mathrm{i}\frac{2\pi k}{N} n} = \boldsymbol{r_k}[n] + \mathrm{i} \cdot \boldsymbol{i_k}[n]$

DCT-II: $\boldsymbol{b_k}[n] = \alpha_k \cdot \cos\left(\frac{\pi}{N} k \left(n + \frac{1}{2} \right) \right)$



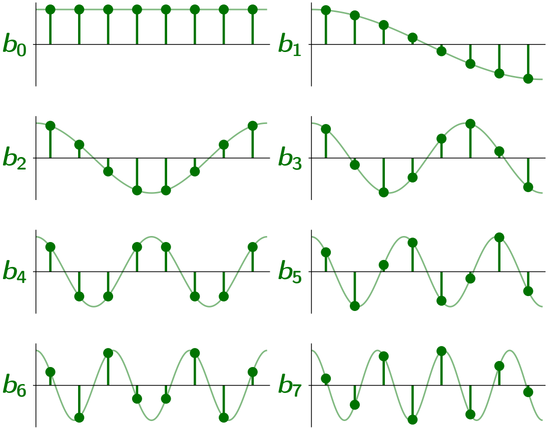


Image & Video Coding: 2D Transforms

Separable Transforms

- Successive 1D transforms of rows and columns of image block

➡ Separable forward and inverse transforms

$$\boxed{\boldsymbol{u} = \boldsymbol{A} \cdot \boldsymbol{s} \cdot \boldsymbol{B}^{\mathrm{T}}} \quad \text{and} \quad \boxed{\boldsymbol{s} = \boldsymbol{A}^{\mathrm{T}} \cdot \boldsymbol{u} \cdot \boldsymbol{B}}$$

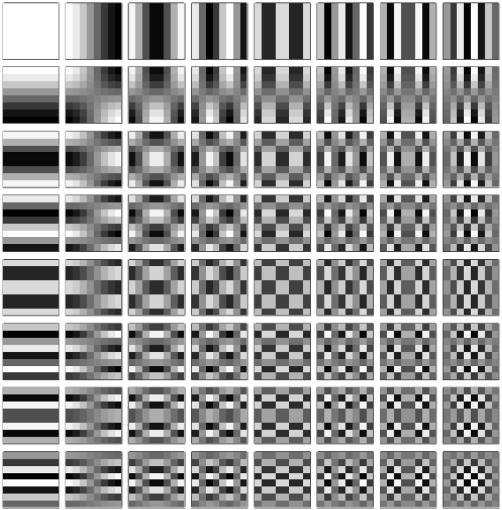
with $\boldsymbol{s}$ — $N \times M$ block of image samples
$\boldsymbol{A}$ — $N \times N$ transform matrix (typically DCT-II)
$\boldsymbol{B}$ — $M \times M$ transform matrix (typically DCT-II)
$\boldsymbol{u}$ — $N \times M$ block of transform coefficients

Great practical importance:

- Two matrix multiplications of size $N \times N$ instead of one multiplication of a vector of size $1 \times N^2$ with a matrix of size $N^2 \times N^2$

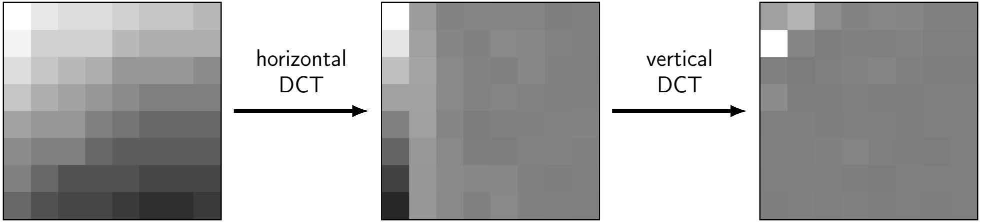
➡ Complexity reduction from $\mathcal{O}(N^4)$ to $\mathcal{O}(N^3)$ [also fast algorithms for DCT-II]

Example: Basis Images of Separable 8×8 DCT-II



Example: Separable DCT-II for 8×8 Image Block

Forward transform for 8×8 block of samples: $\boldsymbol{u} = \boldsymbol{A} \cdot \boldsymbol{s} \cdot \boldsymbol{A}^{\mathrm{T}}$



original block　　after 2d DCT

Example calculation of 2d DCT-II:

1. Horizontal DCT of input block: $\boldsymbol{u}^* = \boldsymbol{s} \cdot \boldsymbol{A}^{\mathrm{T}}$
2. Vertical DCT of intermediate result: $\boldsymbol{u} = \boldsymbol{A} \cdot \boldsymbol{u}^* = \boldsymbol{A} \cdot \boldsymbol{s} \cdot \boldsymbol{A}^{\mathrm{T}}$

Practical Importance of DCT-II

Justification for usage of DCT-II

- Represents signal as weighted sum of frequency components
- Similar to KLT for highly correlated sources ($\varrho \rightarrow 1$)
- Independent of source characteristics
- Fast algorithms for computing forward and inverse transform

DCT-II of size 8×8 is used in

- Image coding standard: JPEG
- Video coding standards: H.261, H.262/MPEG-2, H.263, MPEG-4 Visual

Integer approximation of DCT-II is used in

- Video coding standard H.264/AVC (4×4 and 8×8)
- Video coding standard H.265/HEVC (4×4, 8×8, 16×16, 32×32)
- New standardization project H.266/VVC (from 4×4 to 64×64, including non-square blocks)

Transform Coding in Practice

Orthogonal Transform

- Typically: DCT-II or integer approximation thereof (separable transform for blocks)
- Potential extension in H.266/VVC:
  - Switched transform of DCT/DST families (DCT-II, DST-VII, ...)
  - Non-separable transforms

Scalar Quantization

- Uniform reconstruction quantizers (or very similar designs)
- Bit allocation by using **same quantization step size** for all coefficients
- Usage of advanced quantization algorithms in encoder
- May use quantization weighting matrices for perceptual optimization

Entropy Coding of Quantization Indexes

- Zig-zag scan (or similar scan) for 2D transforms
- Simple: Run-level coding, run-level-last coding, or similar approach
- Better coding efficiency: Adaptive arithmetic coding

Bit Allocation in Practice (for Uniform Reconstruction Quantizers)

- Remember: Optimal bit allocation: Pareto condition

$$\frac{\partial D_k(R_k)}{\partial R_k} = \text{const}$$

- Pareto condition for high rates

$$D_k = \varepsilon_k^2 \cdot \sigma_k^2 \cdot 2^{-2R_k} \qquad \Longrightarrow \qquad D_k(R_k) = \text{const}$$

- High rate distortion approximation for URQs

$$D_k = \frac{1}{12}\Delta_k^2$$

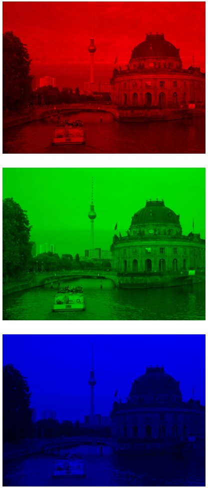
➡ Quantization step sizes for optimal bit allocation at high rates

$$D_k = \frac{1}{12}\Delta_k^2 = \text{const} \qquad \Longrightarrow \qquad \Delta_k = \text{const} = \Delta$$

➡ **In practice, (nearly) optimal bit allocation is typically achieved by using the same quantization step size Δ for all transform coefficients**

Color Transform for Image & Video Coding

RGB



Color Transform for Compression

- Many versions (also depends on RGB color space)
- ➜ **Example**: RGB → YCbCr transform used in JPEG

$$\begin{bmatrix} Y \\ Cb - 128 \\ Cr - 128 \end{bmatrix} = \begin{bmatrix} 0.2990 & 0.5870 & 0.1140 \\ -0.1687 & -0.3313 & 0.5000 \\ 0.5000 & -0.4187 & -0.0813 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.4020 \\ 1 & -0.3441 & -0.7141 \\ 1 & 1.7720 & 0 \end{bmatrix} \cdot \begin{bmatrix} Y \\ Cb - 128 \\ Cr - 128 \end{bmatrix}$$

Energy Compaction for Example Image

$\sigma_R^2 = 3862.28$ $\sigma_Y^2 = 3099.67$

$\sigma_G^2 = 4250.44$ ➜ $\sigma_{Cb}^2 = 83.94$

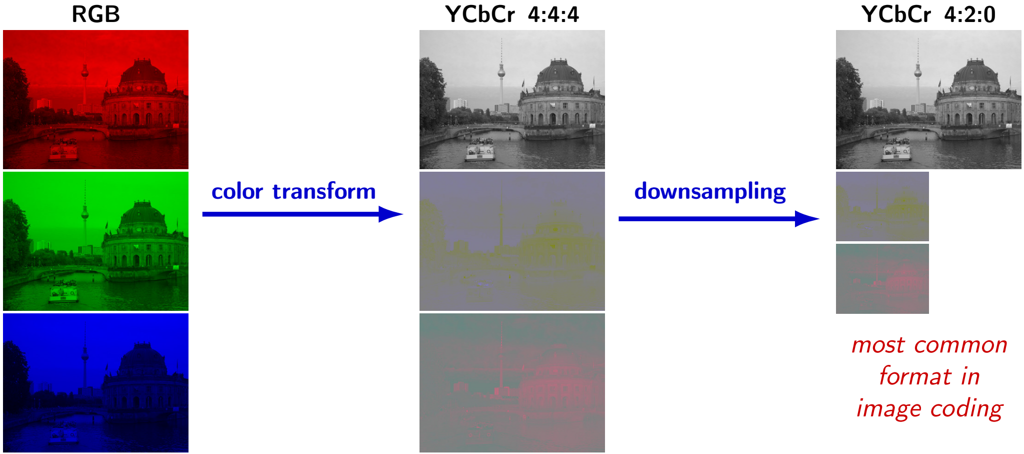
$\sigma_B^2 = 5869.39$ $\sigma_{Cr}^2 = 70.10$

YCbCr

The YCbCr Chroma Sampling Format

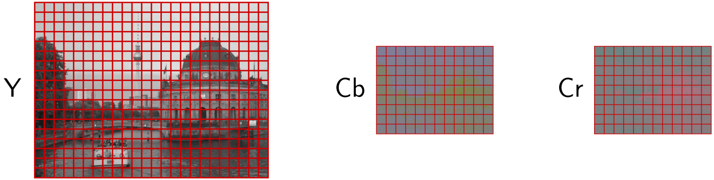
- Human being are less sensitive to color differences (at same luminance)

→ In most applications: Color difference components are downsampled

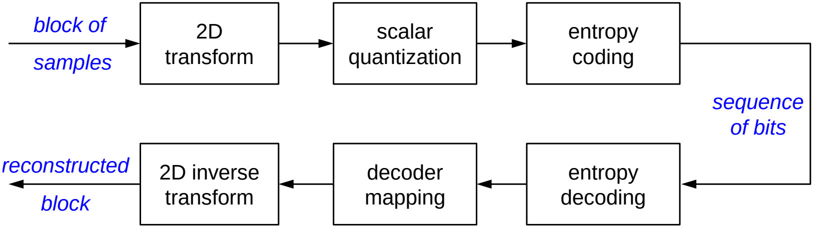


The Image Compression Standard JPEG

- Partition color components (Y, Cb, Cr) into blocks of 8×8 samples

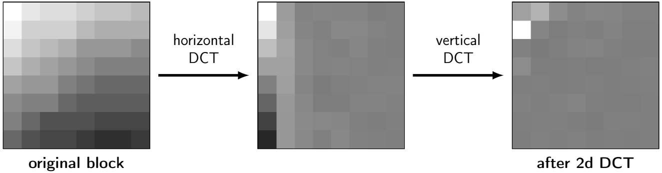


- Transform coding of 8×8 blocks of samples

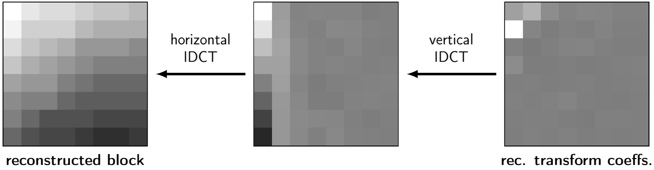


JPEG: Transform of Sample Blocks

- Separable DCT-II of size 8×8 (fast implementation possible)
- Forward transform (in encoder)

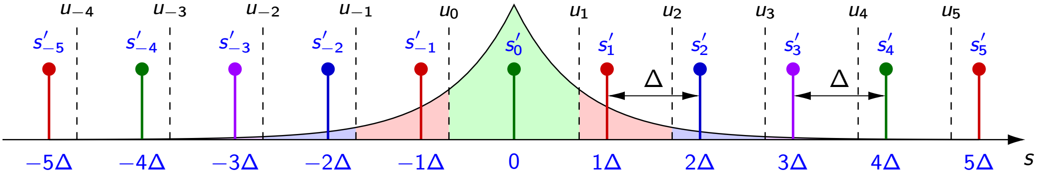


- Inverse transform (in decoder)



➡ Effect of transform: Compaction of signal energy (for typical blocks)

JPEG: Quantization



Uniform Reconstruction Quantizers

- Equally spaced reconstruction levels (indicated by step size Δ)
- Simple decoder mapping

$$t' = \Delta \cdot q$$

- Simplest (but not best) encoder:

$$q = \text{round}(t/\Delta)$$

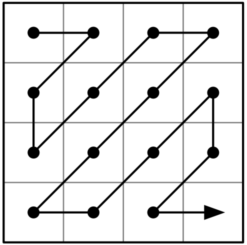
- Better encoders use Lagrangian optimization (minimization of $D + \lambda R$)

➡ **Quantization step size Δ determines tradeoff between quality and bit rate**

JPEG: Entropy Coding

0.242	0.108	0.053	0.009
0.105	0.053	0.022	0.002
0.046	0.017	0.006	0.001
0.009	0.002	0.001	0.000

probabilities $P(q_k \neq 0)$



zig-zag scan (JPEG)

1 Scanning of Quantization indexes

- Convert matrix of quantization indexes into sequence
- Traverse quantization indexes from low to high frequency positions
- JPEG: Zig-zag scan

JPEG: Entropy Coding

2 Entropy Coding of Sequences of Quantization Indexes

- Often long sequences of zeros (in particular at end of sequence)

➡ Entropy coding should exploit this property

JPEG: Run-Level Coding (V2V code)

- Map sequence a symbols (transform coefficients) into (run,level) pairs, including a special end-of-block (eob) symbol

  level : value of next non-zero symbol
  run : number of zero symbols that precede next non-zero symbol
  eob : all following symbols are equal to zero (end-of-block)

➡ Assign codewords to (run,level) pairs (including eob symbol)

- **Example**: 64 symbols: 5 3 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 ...
  (run,level) pairs: (0,5) (0,3) (3,1) (1,1) (2,1) (eob)

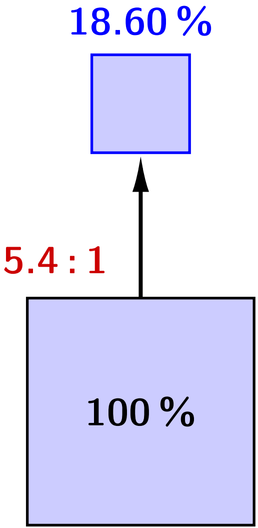
JPEG Compression Example

Original Image (960 × 720 image points, RGB: 2 MByte)

100 %

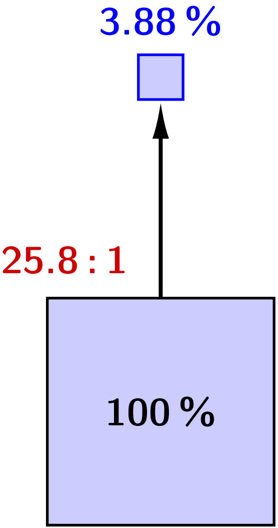
JPEG Compression Example

Lossy Compressed: JPEG (Quality 94)



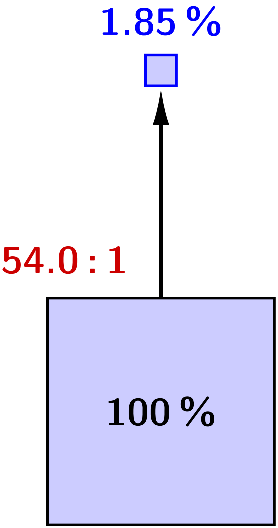
JPEG Compression Example

Lossy Compressed: JPEG (Quality 66)



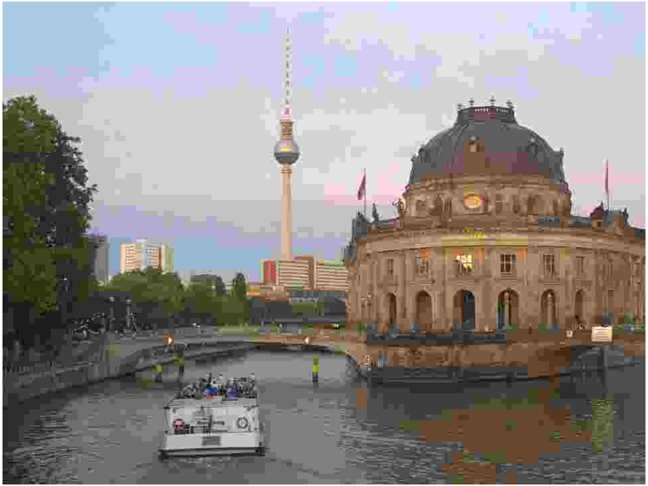
JPEG Compression Example

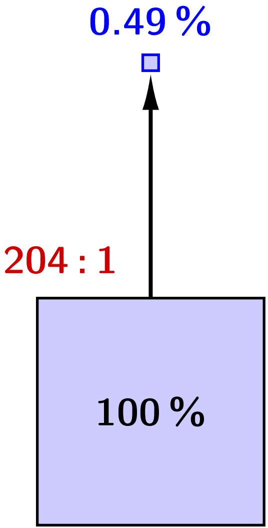
Lossy Compressed: JPEG (Quality 27)



JPEG Compression Example

Lossy Compressed: JPEG (Quality 6)





Audio Compression Example: MPEG-2 Advanced Audio Coding (AAC)

Main Component: Transform Coding of Sample Blocks

- Transform: Modified DCT for overlapping blocks
- Quantization: Scalar quantization with psycho-acoustic model
- Entropy Coding: Variant of Huffman coding

Linear Transform

- Audio signal is coded based on overlapping blocks of samples
- Transform: Modified discrete cosine transform (MDCT)

Quantization of Transform Coefficients

- Scalar quantization of transform coefficients (spectral coefficients)
- Utilization of psycho-acoustic models by noise shaping

Entropy Coding of Quantization Indexes

- Grouping and interleaving
- Huffman coding for tuples of n quantization indexes (n is variable)

Modified Discrete Cosine Transform (MDCT)

Forward Transform (MDCT)

- The forward transform maps $2N$ samples to N transform coefficients

$$u[k] = \frac{1}{\sqrt{N}} \sum_{n=0}^{2N-1} s[n] \cdot \cos\left(\frac{\pi}{N} \left(n + \frac{N+1}{2} \right) \left(k + \frac{1}{2} \right) \right)$$

Inverse Transform (IMDCT)

- The inverse transform maps N transform coefficients to $2N$ samples

$$x[n] = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} u[k] \cdot \cos\left(\frac{\pi}{N} \left(n + \frac{N+1}{2} \right) \left(k + \frac{1}{2} \right) \right)$$

Perfect Reconstruction

- Neighboring blocks of samples $s[n]$ overlap by 50% (at each side)
- Perfect reconstruction of $s[n]$ is achieved by adding the inverse transformed blocks $x[n]$

➡ Property of time-domain aliasing cancellation

Summary of Lecture

Signal-Independent Transforms

- Walsh-Hadamard Transform (WHT): Perceptual disturbing artefacts
- Discrete Fourier Transform (DFT): Problem due to implicit periodic signal extension
- Discrete Trigonometric Transforms: Family of Sine and Cosine transforms

Discrete Cosine Transform of Type II (DCT-II)

- DFT of mirrored signal with half-sample symmetry at both sides
- Reduced blocking artifacts compared to DFT
- Good approximation of KLT for highly-correlated signals

Transform Coding in Practice

- Color transforms in image and video coding: RGB to YCbCr conversion
- JPEG image compression: 2D DCT-II + URQ + Run-level coding
- AAC audio compression: MDCT for overlapped blocks + scalar quantization + Huffman coding

Exercise 1: Correlation of Transform Coefficients

Given is a zero-mean AR(1) sources with a variance σ^2 and a correlation coefficient $\varrho = 0.9$

Consider transform coding of blocks of 2 samples using the transform

$$\begin{bmatrix} u_{k,0} \\ u_{k,1} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_{2k} \\ s_{2k+1} \end{bmatrix},$$

where k represents the index of the transform block

- Determine the following variances and covariances of the transform coefficients (inside a block and between neighbouring blocks):

$$\begin{aligned} \mathrm{E}\{\, U_{k,0}^2 \,\} &= ? & \qquad \mathrm{E}\{\, U_{k,0}\, U_{k+1,0} \,\} &= ? \\ \mathrm{E}\{\, U_{k,1}^2 \,\} &= ? & \qquad \mathrm{E}\{\, U_{k,1}\, U_{k+1,1} \,\} &= ? \\ \mathrm{E}\{\, U_{k,0}\, U_{k,1} \,\} &= ? & \qquad \mathrm{E}\{\, U_{k,0}\, U_{k+1,1} \,\} &= ? \end{aligned}$$

- Is it worth to exploit the correlations between the transform coefficients of neighboring block (e.g., for typical correlation factors of $\varrho \approx 0.9$) ?

Exercise 2: First Version of Lossy Image Codec (Implementation)

Implement a first lossy image codec for PPM images:

1. Use the source code of last weeks exercise as basis (see KVV)
2. Add some variant of entropy coding for the quantization indexes, for example:
   - Simple Rice coding or Exp-Golomb coding (see lossless codec example in KVV)
   - Adaptive binary arithmetic coding using a unary binarization (see lossless coding example in KVV)
   - ...
3. Implement an encoder that converts a PPM image into a bitstream file
4. Implement a corresponding decoder that converts a bitstream file into a PPM image
5. Test your encoder with some example images and multiple quantization step sizes
6. (Optional) Try to improve your codec by using the YCbCr color format
   - Implement an RBG-to-YCbCr transform before the actual encoding
   - Implement the inverse YCbCr-to-RGB transform after the actual decoding
   - Possible extension: Sub-sampling of chroma components