山下研の常識

画像符号化

山下幸彦

大学院理工学研究科
東京工業大学

平成２２年１２月２１日
(平成２３年１０月２０日修正)

発表内容 (1/2)

1. **はじめに**
   - 画像の情報量
   - 非可逆符号化と可逆符号化
2. **画像符号化の基盤技術**
   - エントロピー符号化
     - ハフマン符号化
     - 算術符号化
       - レンジコーダ
       - Q-coder
   - ランレングス符号化
   - 予測符号化
   - 変換符号化

発表内容 (1/2)

4. **標準的画像符号化法**
   - JPEG
   - MPEG 2
   - MPEG 4 AVC
5. **離散コサイン変換を用いない画像符号化**
   - ウェーブレット変換
   - JPEG 2000
   - サブバンド変換
   - サブバンド画像符号化
     - Generalized variable length lapped transform
     - サブバンド動画像符号化
6. まとめ

画像符号化の重要性

画像の情報量

- 画像の情報量は大きい。
  SDTV (Standard Television) の例：
  - 色　　：3 (R, G, B)
  - 明るさ：8 [bit]
  - 横　　：720 [画素]
  - 縦　　：480 [画素]
  - 時間　：30 [frame/s]

$$3 \times 8 \times 720 \times 480 \times 30 \simeq \mathbf{248.8}[\text{Mbit/s}]$$

  HDTV (High Definition Television)

$$3 \times 8 \times 1920 \times 1080 \times 30 \simeq \mathbf{1.48}[\text{Gbit/s}]$$

- 地上波テレビ：18 [Mbit/s] 程度
  $\Rightarrow$ **データ量を圧縮する必要がある。**

非可逆符号化 vs. 可逆符号化

非可逆符号化 (lossy)

- 原画像が復号画像と異なる。
- その差は人間の目には分かりにくいようにする。
- ブロック歪み，リンギング(モスキート雑音)
- データ圧縮率に優れる。
- 一般に用いられる画像符号は非可逆符号化

可逆符号化 (無歪み符号化) (lossless)

- 原画像が復号画像が全く同じ。
- 圧縮率は1/2程度
- 医用画像などに用いられる(データの保存が法律で必要)。

今回は，非可逆符号化を扱う。

エントロピー符号化

- **符号化**：アルファベット (整数) 列を $0, 1$ の系列に直すこと。
- **符号**：1つのアルファベットに割り当てられた $0, 1$ の系列

  例：

  符号：$0 \to 00,\ 1 \to 01,\ 2 \to 1001$

  アルファベット列：$2011 \to 1001000101$
- **エントロピー**：平均情報量 (符号の出現確率で決まる量)
- 平均符号長はエントロピー以下にはできない。
- エントロピー符号化：平均符号長をエントロピーに近づける符号化
- **原則：**

  頻繁に現れるアルファベットに短い符号

  あまり現れないアルファベットに長い符号を割り当てる。
- **コンパクト符号**：平均符号長が最小になる符号

ハフマン(Huffman) 符号化

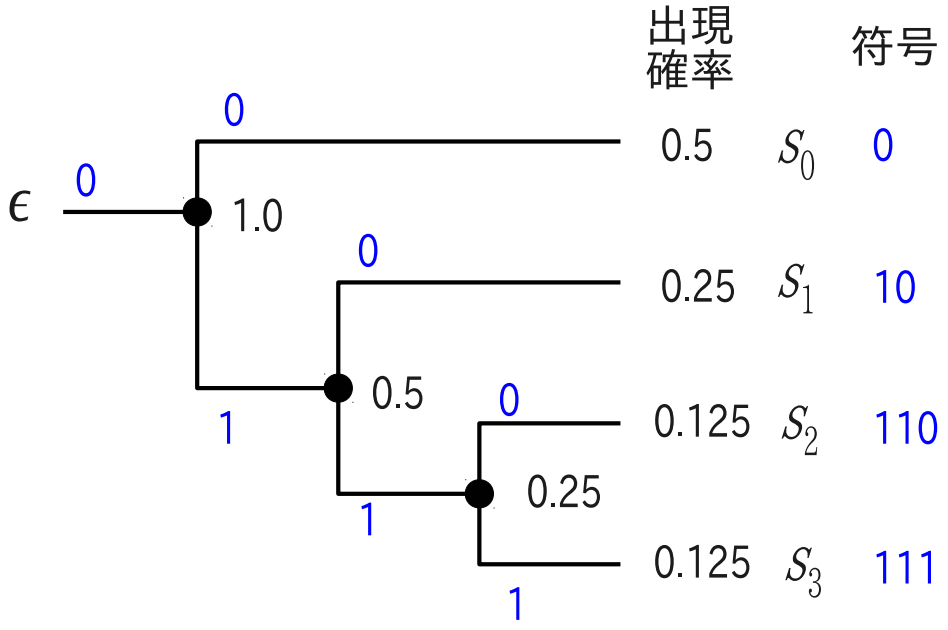
- アルファベットの出現確率が与えられたときに，**コンパクト符号を作ることができる。**
- **符号化テーブル**を使って符号を決める。
- 画像符号化では，本当に確率を使って符号を決めるのではなく，**符号化テーブルを使って符号を決める方法一般に使われている。**
- 具体的な符号化テーブルの例は，JPEGで説明する。

ハフマン符号の符号化構成法

平均符号長：

$0.5 \times 1 + 0.25 \times 2 + 0.125 \times 3$

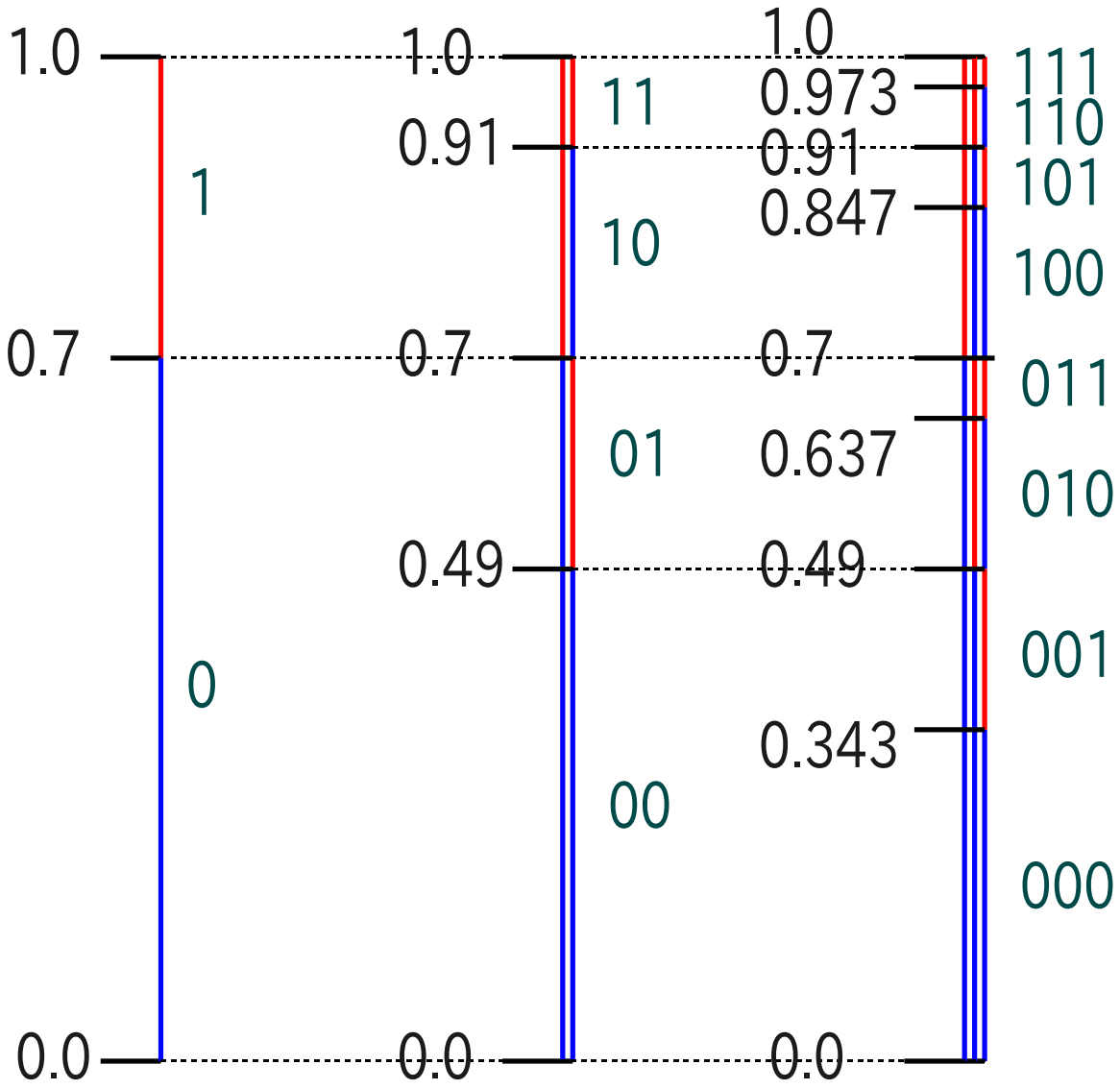
$+0.125 \times 3 = 1.75$



算術符号化

- MITのP. Eliasによって提案
- IBMのG. G. Langdon, J. J. Rissanenが完成(特許)
- JPEG2000のMQ coderに使われている。
- 符号化アルゴリズム (原理)：
  1. 初期の対象区間：$[0, 1)$
  2. 現在の対象区間を，**シンボルの確率に比例させた長さの区間**に分割
  3. 符号化するシンボルに対応する区間を対象区間とする。
  4. 2へもどる
- 区間に含まれる1つの小数でその区間を表す。
- シンボル列の出現確率が高い方が，長い区間になる。
  $\Rightarrow$ 一般には長い区間には桁数の短い小数が含まれる。
  $\Rightarrow$ アルファベット列の出現確率が高いほど短い符号になる。
- 復号は小数が含まれる区間を，アルファベットの順番で調べる。

算術符号化の原理



- 0が出る確率：0.7
- 1が出る確率：0.3

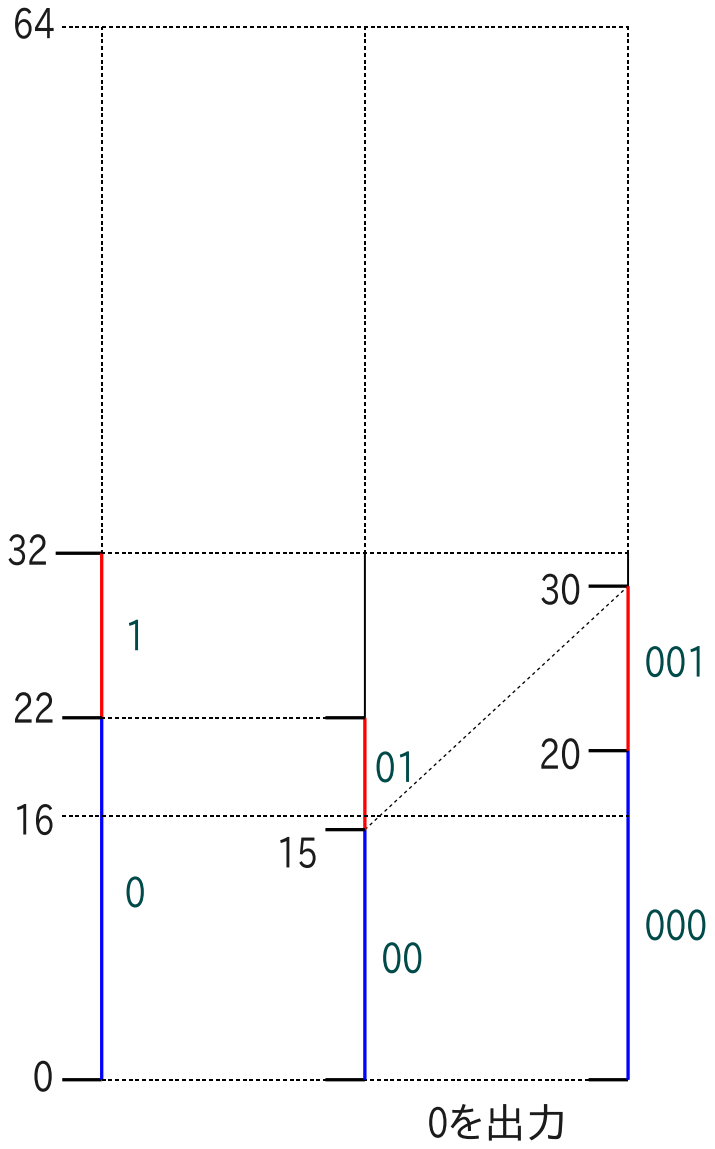
- 000を表す：0.0 (0 bit)
- 010を表す：0.5 (1, 1 bit)
- 100を表す：
  0.75 (11, 2 bit)
- 110を表す：
  0.9375 (1111, 4 bit)

算術符号化の例

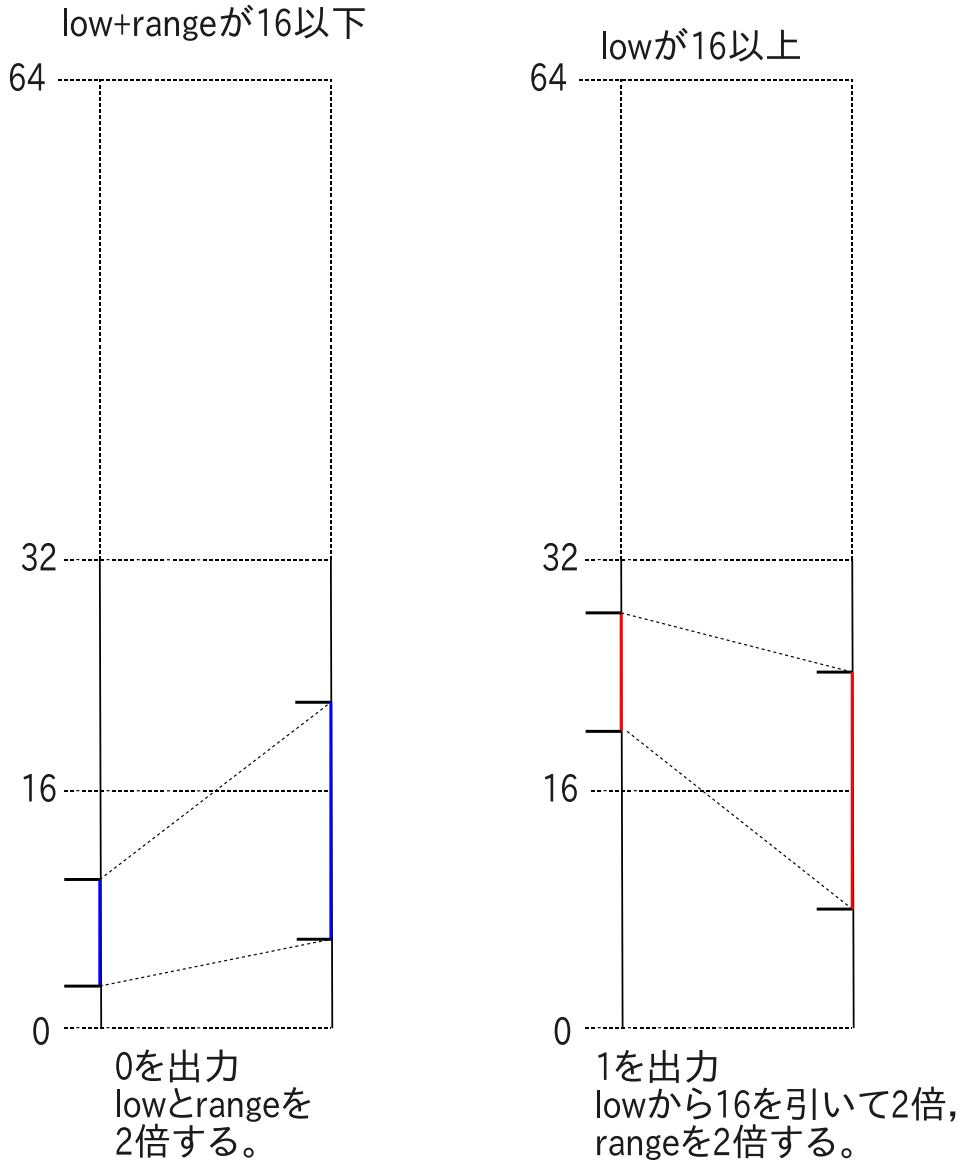
- Dmitry Subbotinのレンジコーダ (の1 bit版)
  - 全体区間を**下端 (low)** と**区間範囲 (range)** で表現
  - 符号化：**最後アルファベットで分割した後のlowの値**を送る。
  - アルゴリズムが簡単である。特許がない。
- 処理の基本の区間を[0, 31]の 5 bit (処理には6 bit必要) として説明
  初期状態では，0は0.0, 32が1.0を表している。
- 今回の説明：**アルファベット列の終了は実装しない。**
  - 符号化したものより長いアルファベットが復号される可能性がある。
  - 符号化したアルファベット列と復号されたアルファベット列は，符号化したところまでは一致する。
  - 通常は，入力アルファベットとして終了記号を加える。または，アルファベット列から終了が分かる。

(学習用の Java のソースコードが，`http://www.ide.titech.ac.jp/~yamasita/yylab/index.html` から取得できる。)

基本

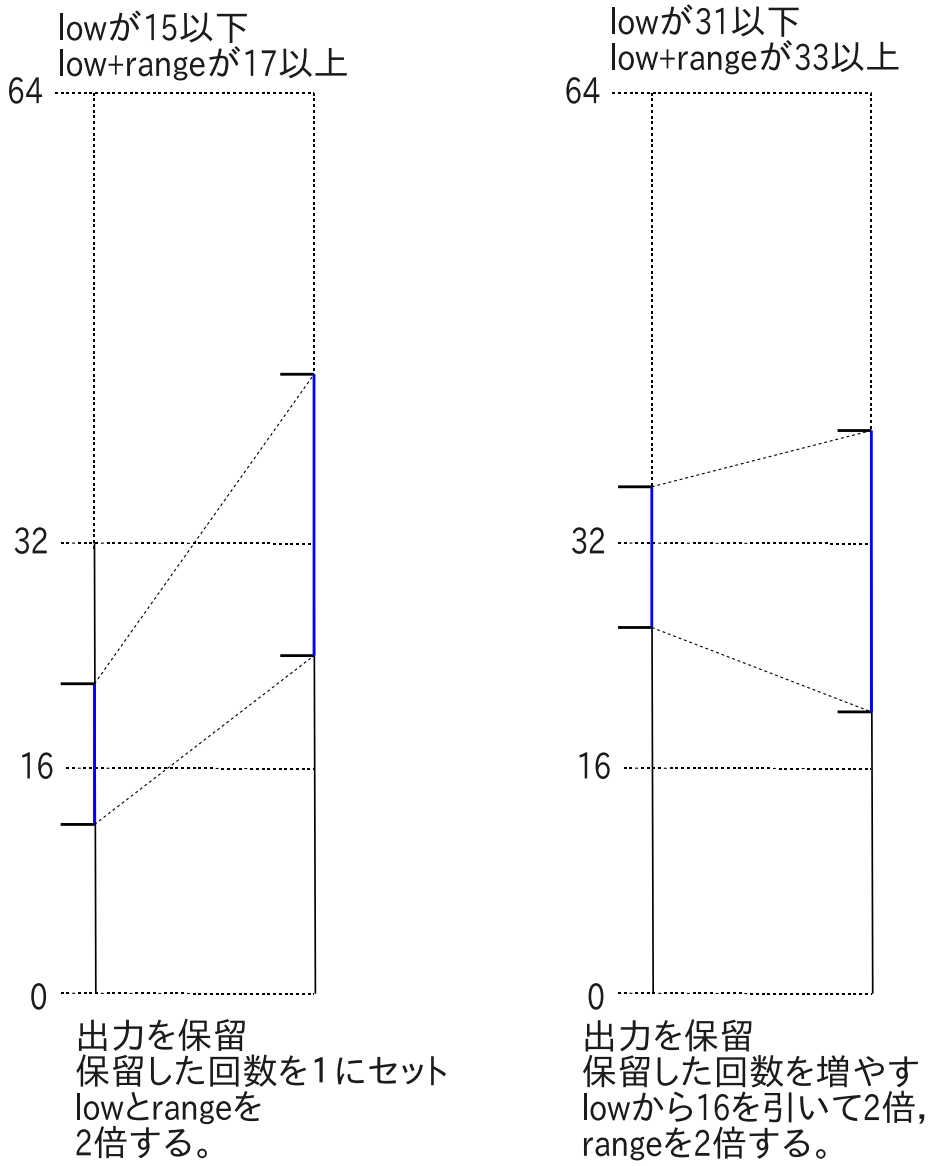


区間拡大の基本 (保留がないとき)



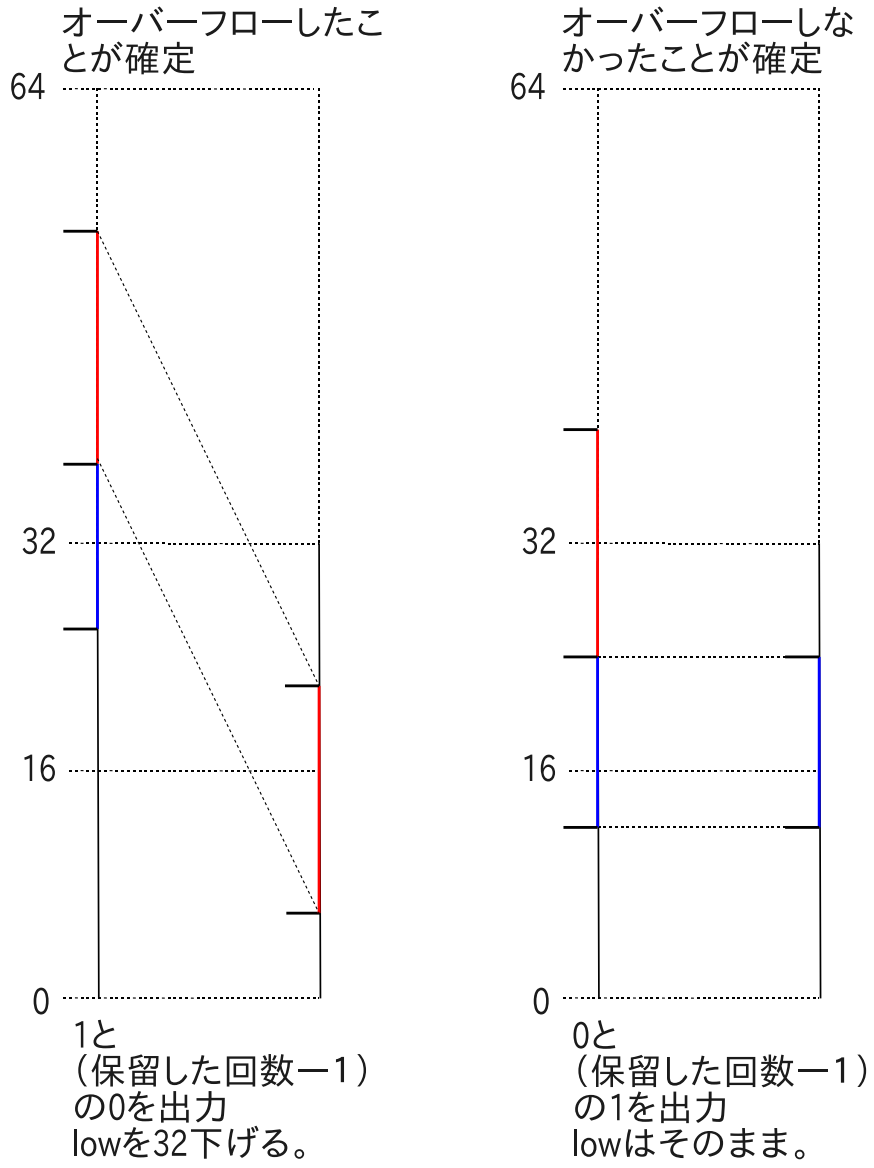
- rangeが15以下になると，区間を拡大する。
- low+rangeが15以下の場合：
  それ以降の分割によらず，最上位ビットが0と決まる。
  0を出力し，区間を2倍に拡大する。
- lowが16以上の場合：
  1を出力し，lowから16を引いて(最上位ビットを0にする)，区間を2倍に拡大する。
- 上記操作は，左ビットシフト

区間拡大の基本（保留）



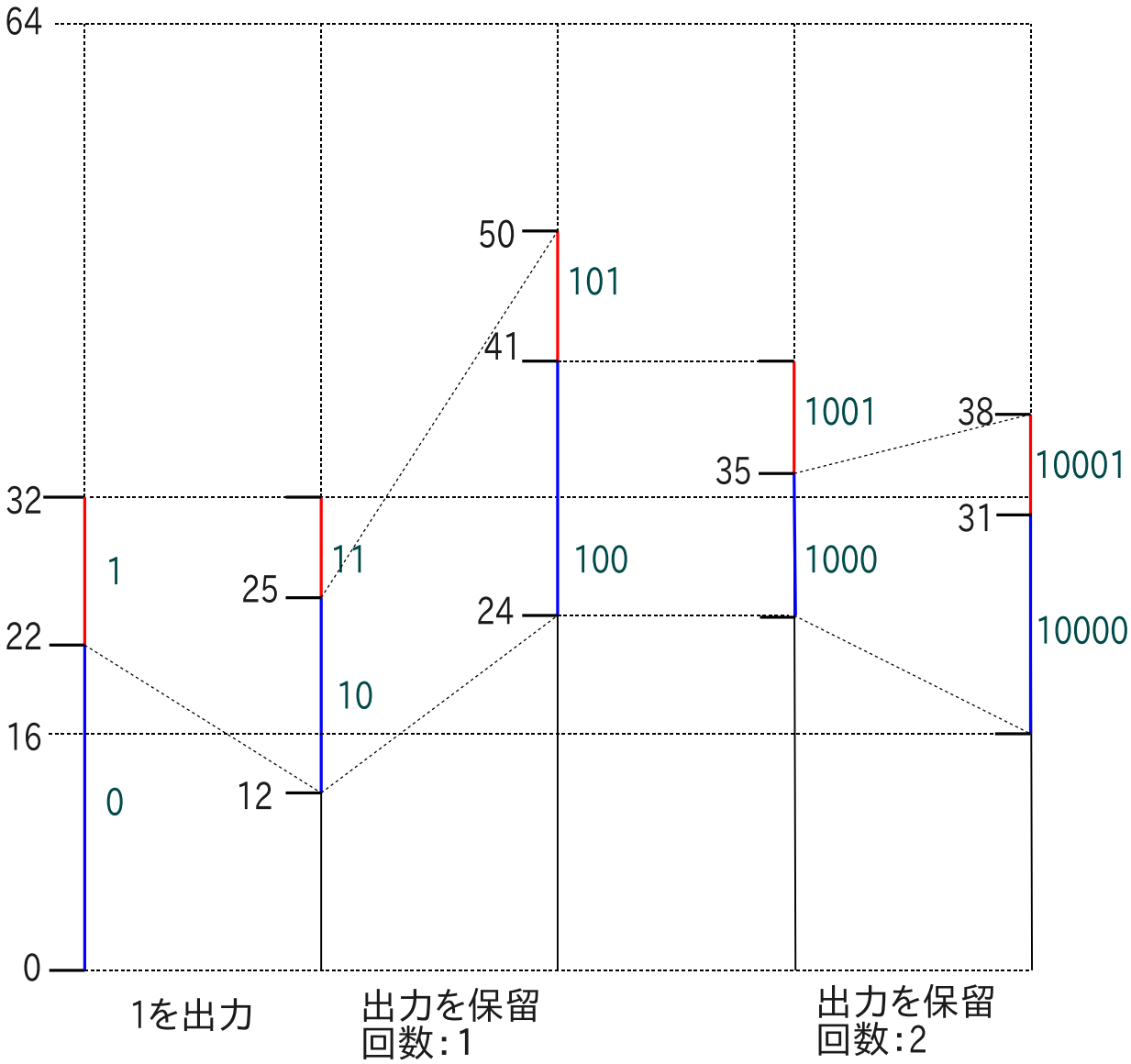
- lowが15以下，low+rangeが16以上の時は，最上位ビットが決まらないため，**保留**とする。
  出力はしない。
- その最上位ビットを0と仮定して処理を続ける。
- 保留中の拡大時に，lowが31以下，low+rangeが32以上の時は，先の最上bitが確定しないので，再度保留する。
- **連続して保留した回数を数える。**

保留の解決

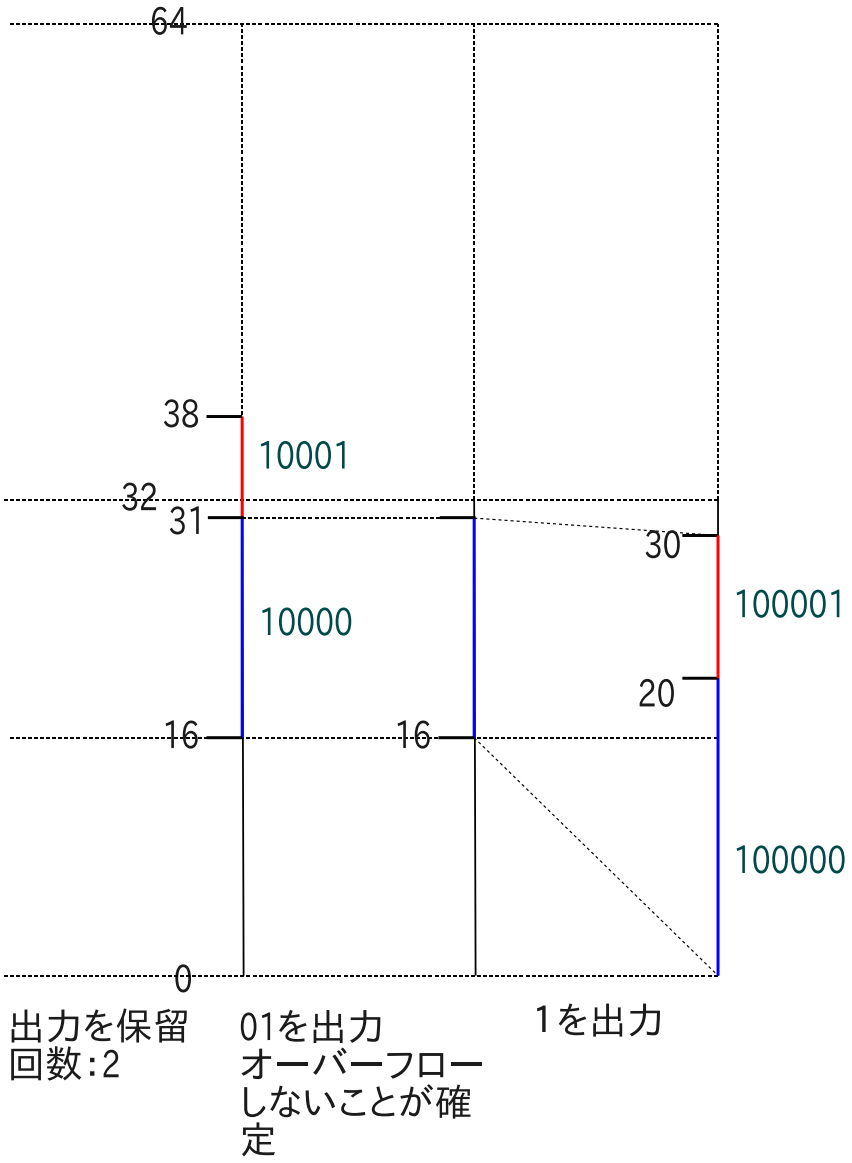


- 区間の長さに関わらず，
- 保留中に，lowが32以上の場合，保留を始めた最上位ビットが1であったことが確定する。
  1と(保留した回数−1個)の0を出力し，lowから32引く。
- 保留中に，low+rangeが31以下の場合，保留を始めた最上位ビットが0であったことが確定する。
  0と(保留した回数−1個)の1を出力する。
- 保留状態を解除する。

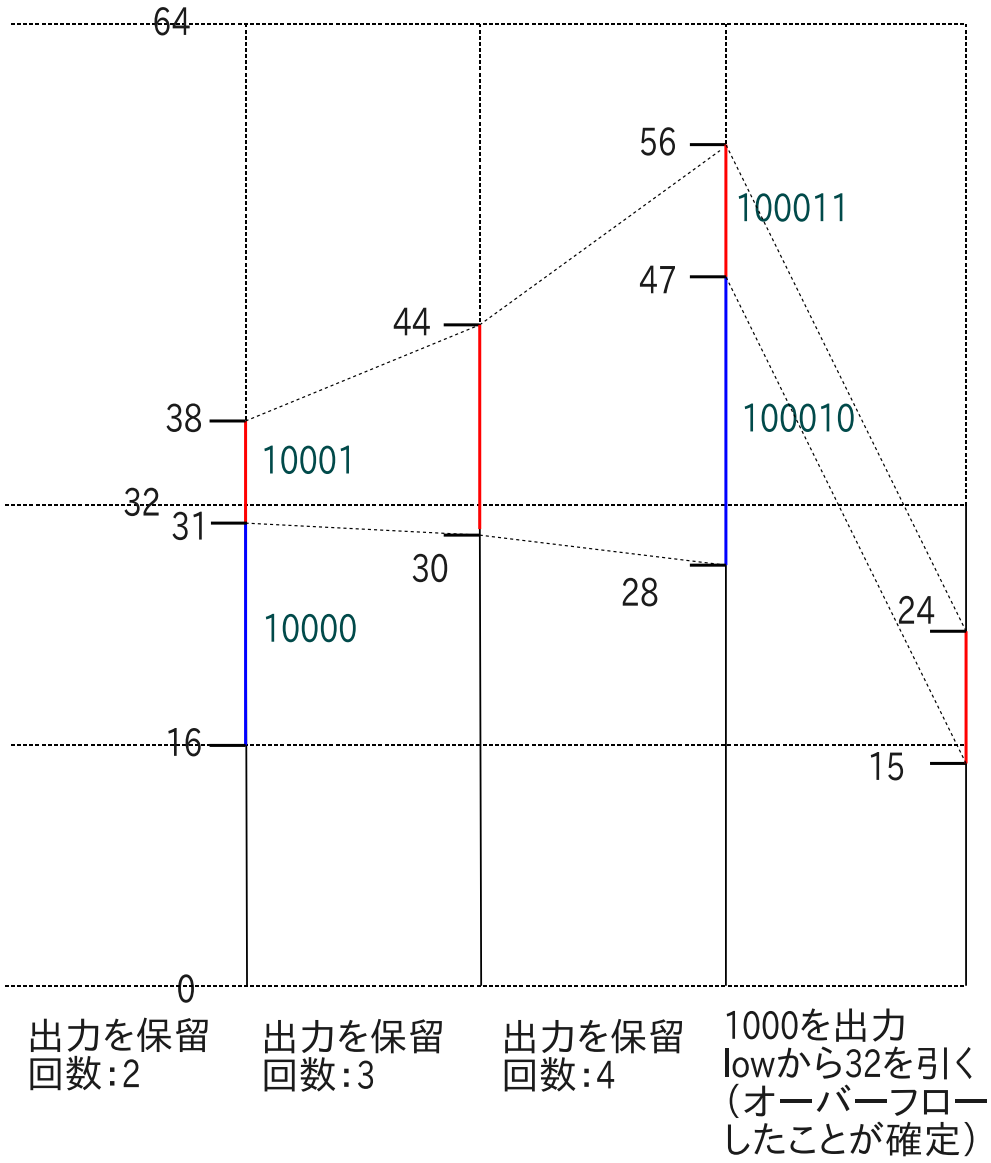
保留の例


64
50
101
41
1001
38
10001
35
32
31
1
11
100
1000
25
24
10000
22
10
16
0
12
0
1を出力
出力を保留
回数:1
出力を保留
回数:2

保留解決の例(オーバーフローなし)



保留解決の例(オーバーフローあり)



Q-coder

- 通常の算術符号化では，アルファベットを1つ処理するごとに乗算の計算が1回必要。
  ⇒ **処理として重い。**(少なくとも2000年ごろは)
- Q-coderは，乗算をしないbinary (入力が0 or 1)の算術符号化
- JPEG 2000のMQ-coderはQ-coderを元にしている。
- 確率の和を，1.0でなく，0.75以上1.5未満で考える。
- 下側を必ず出現確率が少ない入力アルファベット(0 or 1)とする。
- 処理中に，上側区間が小さくなる場合は，入れ替えを行う。
- 記号：
  Q_e : 下側区間長(下側アルファベットの出現確率に対応する整数)
  A : 全体区間長 (整数，レンジコーダのrangeに相当)
  C : 区間の下側位置(レンジコーダの**low**に相当)
  最後アルファベットで分割した後のこの値を送る)

Q-coder

- **全体区間長に関わらず，下側のアルファベットのための区間長はQ_eとする。**上側のアルファベットのための区間長は $A - Q_e$ となる。
  下側区間：$[C,\ C + Q_e)$
  上側区間：$[C + Q_e,\ C + A)$
- **誤差**：
  Q_eに乗算を用いないため，下側アルファベットの出現確率に対して，Q_eによる区間幅は，$1/1.5 \sim 1/0.75$倍の誤差を含む。
- Q-coderは，基本的には16 bitで演算する**(以下，16進数で表記)**。
- Aは，8000 (0.75)からFFFF (1.5未満の最大数)の範囲となる。
- Cも，オーバーフローしないときはFFFF以下の値になる。
- Aの値が8000未満になったら，**再正規化する(拡大する)**。
  Aを2倍
  Cの最上位ビットを送り，左ビットシフトする。
- 再正規化時に生じる1 bitの出力は，1 byte (8 bit)にまとめて出力する。

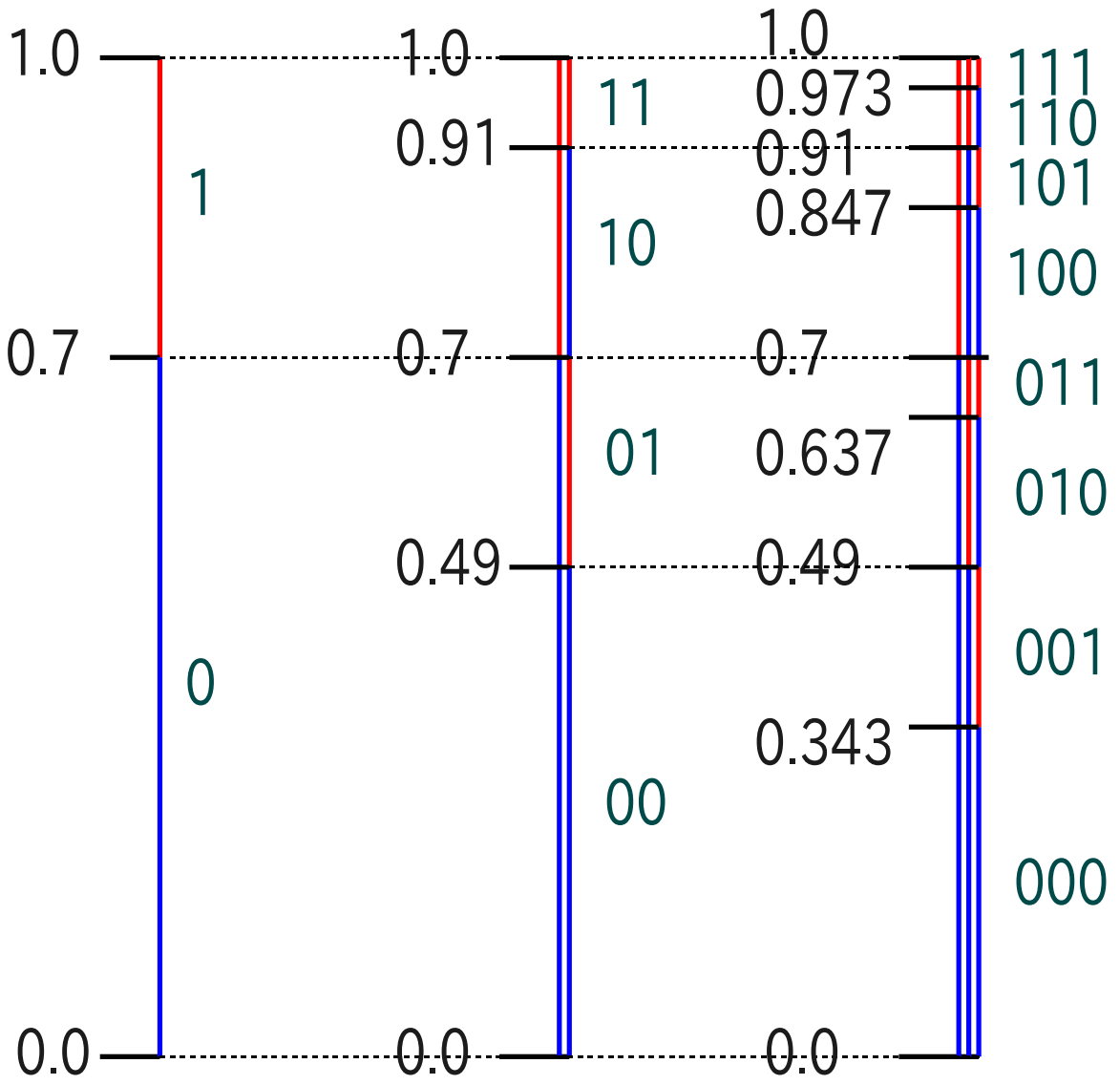
Q-coderのオーバーフローの処理

- 再正規化時にオーバーフローが生じる可能性があるときでも，レンジコーダのような保留処理を行わず，オーバーフローしないものとして処理を続ける。
- **オーバーフロー**が生じたときは，出力のために蓄積している1 byteのデータに1を加算することによって行う。
- 1 byteを越えるオーバーフローに対しては，その byte に対してオーバーフローしたとのフラグを立てる。
- 一通り符号化が終わった後に，出力の最後の byte から調べて行き，オーバーフローしている byte がある場合は，その直前のbyteに対して1を加算する処理を行う。
- 加算された byteがオーバーフローした場合は，その前 byte に加算することを繰り返す。

算術符号化による適応符号化

- 符号化するアルファベット列に適応して区間幅を変更する。
  0が続くようならば，0に対する区間幅を長くする。
- 算術符号化の符号列は，区間に含まれる小数を表している。
- **復号時には，復号するアルファベットまでの区間が分かっていれば，復号できる。**
- **既に送ったアルファベットに依存して区間幅を変えても，**区間幅を変更するアルゴリズムを符号化器，復号器で共有していれば，復号器でも変更した区間幅が構成できる。
- 1 bit ごとの適応的な符号化器の作成が可能
- この原理が JPEG2000 の MQ-coder で使われている。
- 符号化するビットの**コンテキスト (文脈)** が，すでに送ったデーターから分かる場合は，**コンテクストごとに異なる確率**を使うことができる。

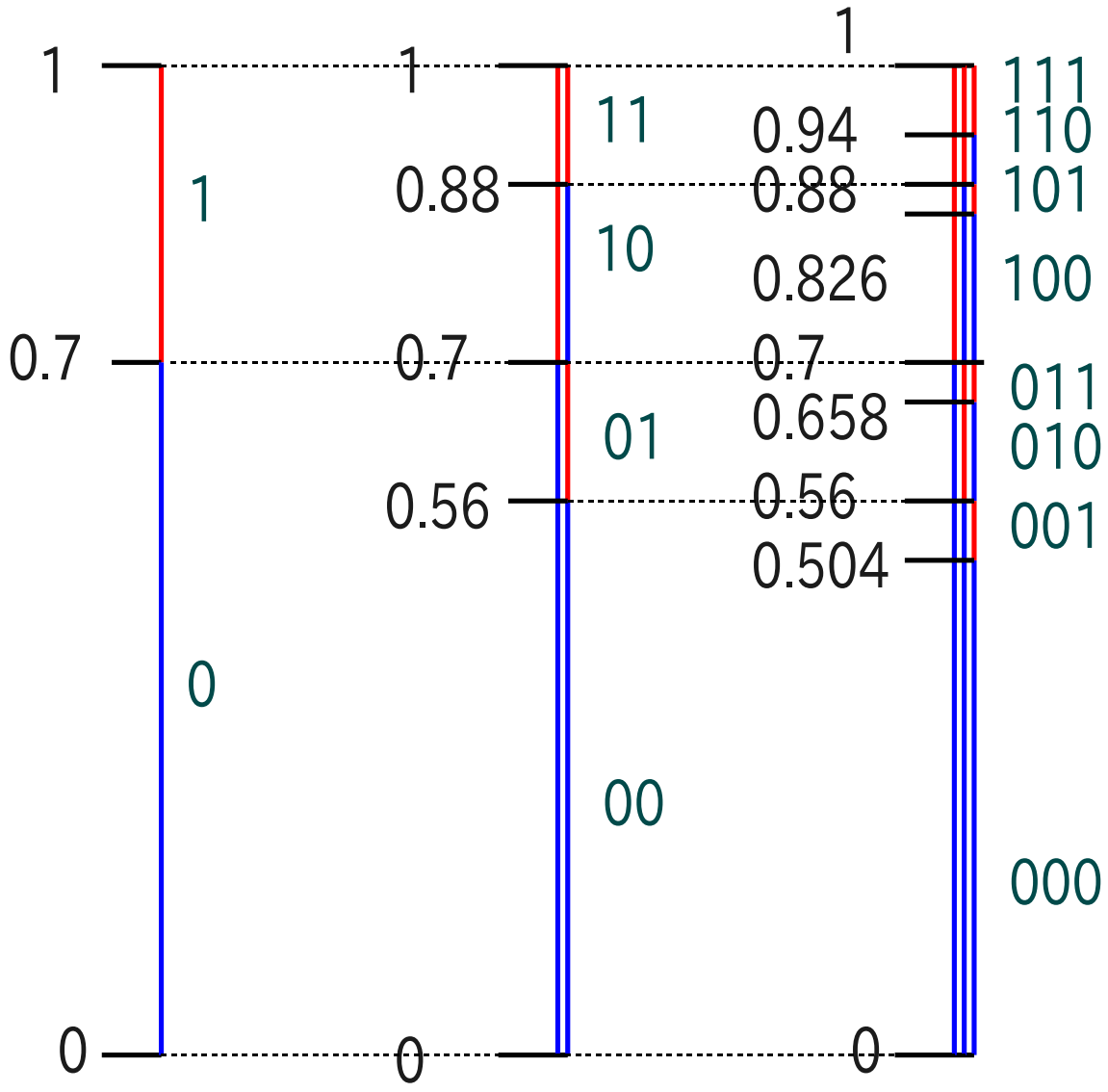
算術符号化による適応符号化 (非適応符号化の例)



- p_0：0が出る確率
- p_1：1が出る確率
- 固定

$$p_0 = 0.7$$
$$p_1 = 0.3$$

算術符号化による適応符号化



- p_0：0が出る確率
- p_1：1が出る確
- 初期

$$p_0 = 0.7$$

$$p_1 = 0.3$$

- 直前の符号が0

$$p_0 \to p_0 + 0.1$$

$$p_1 \to p_1 - 0.1$$

- 直前の符号が1

$$p_0 \to p_0 - 0.1$$

$$p_1 \to p_1 + 0.1$$

- p_0 or p_1の値が0.1以下の場合は，値を変えない。

ランレングス (run lenght) 符号

- 画像符号化では，符号化したいアルファベットに0が続くことが多い。
  zero run
- 連続している0の個数(**ゼロランレングス**)を情報とする。
  $0 \to 1,\ 00 \to 2,\ 000 \to 3,\ 0000 \to 4, \cdots$
- 連続して0が続く回数の出現確率は，その値の種類に依存する。
  $\Rightarrow$ 0が続く値もエントロピー符号化する。
- 残りがすべて0のときのための，専用の符号を割り当てる。
- 具体的な例は，JPEGで説明する。

量子化

- 画像の画素値やその変換係数は一般には**連続量**
- 符号化するために，整数値に変換する。
- 線形量子化

  Q：量子化係数

  u：入力連続値

  v：量子化された値

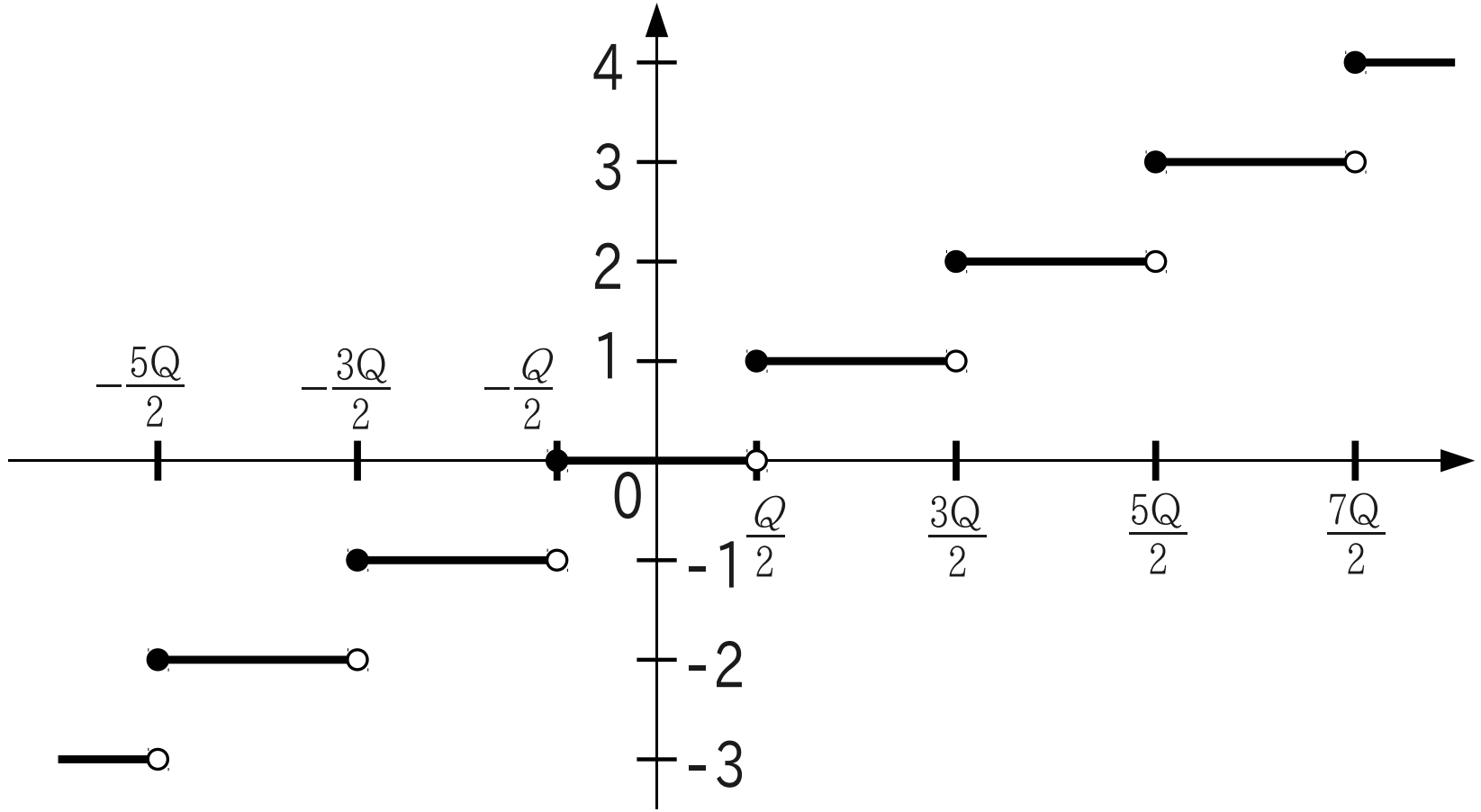
$$v = \mathrm{floor}\left(\frac{u}{Q} + 0.5\right)$$

- 逆量子化：

  $u_O = Qv$：出力連続値
- u と u_O には，$\pm\frac{1}{2}Q$ の誤差が生じる。
- デッドゾーン (0にする範囲を広くする。)

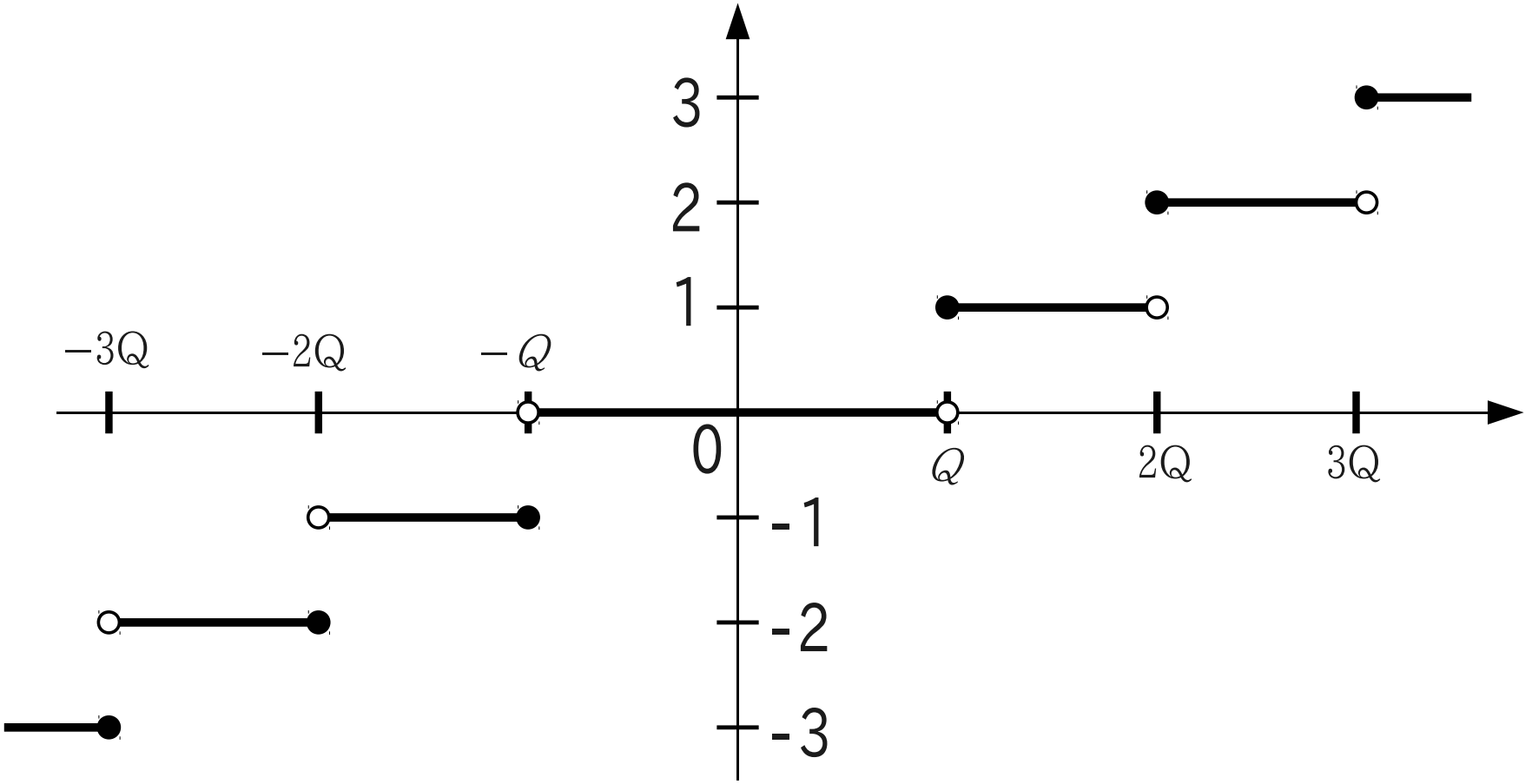
  0を符号化する効率が良いためである。

線形量子化



- Qが大きいと，送信する整数が小さくなり誤差が大きくなる。

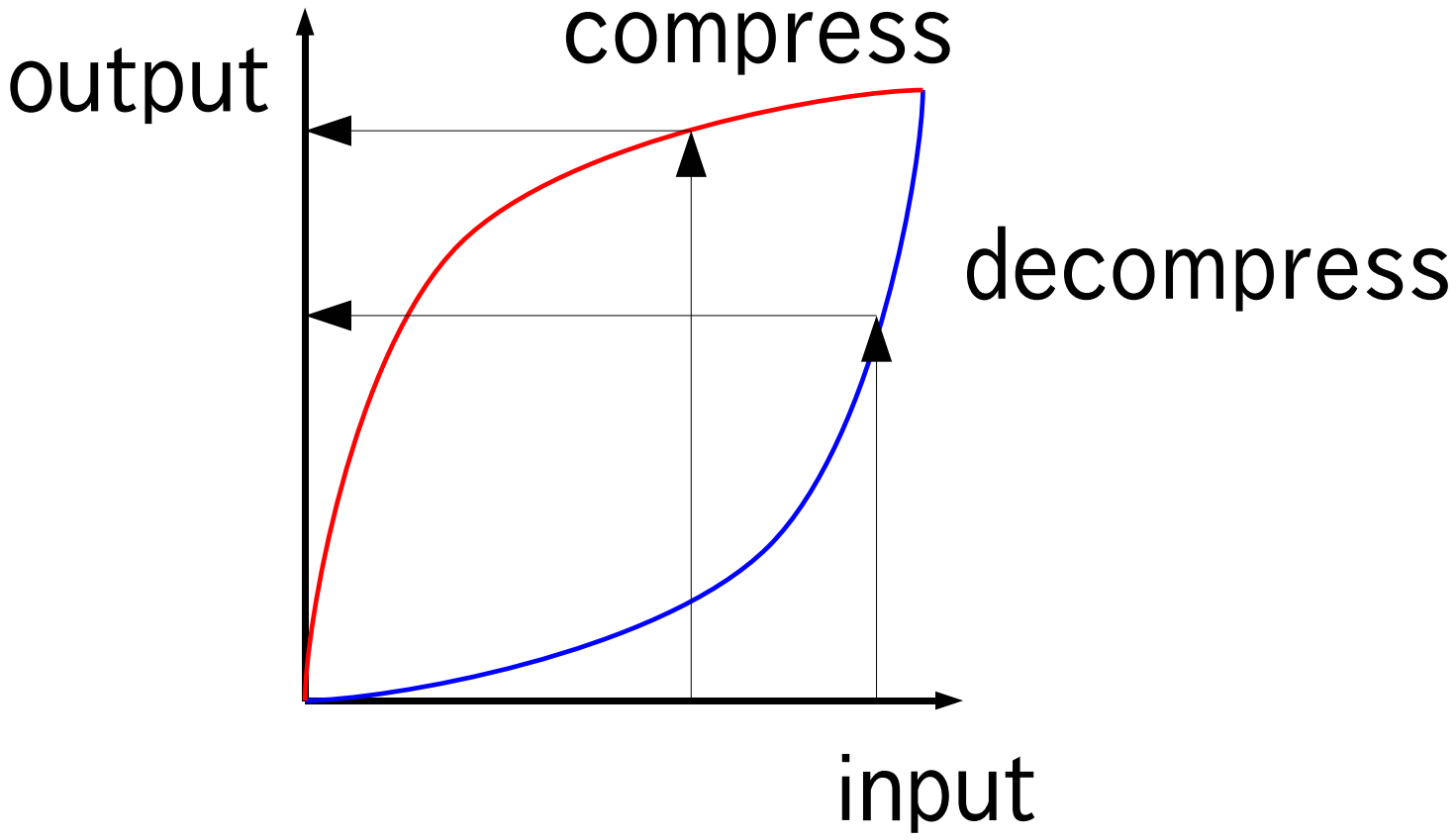
線形量子化 (デッドゾーンあり)



- デッドゾーン がある線形量子化 (JPEG 2000)

非線形量子化

- 音が大きいところでは誤差が分かりにくいため，量子化係数を大きくする。
- 画像符号化ではあまり使わない。



予測符号化

- **差分符号化** (最も簡単な例)
  - 直前の値を予測値とする。
  - $x_1, x_2, x_3, \cdots$ を送りたい信号列とする。
  - x_{n+1} と x_n の差は小さいことが多い。
  - $x_{n+1} - x_n$ を送る。
    値が小さいため，少ないbit数で送ることができる。

n	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
入力値	120	130	134	137	140	144	138	142	134
差分	120	10	4	3	3	4	-6	4	-8

局所復号器 (ローカルデコーダ) 1/2

- 予測符号化で量子化を行う場合は，注意が必要
- 不完全な情報を送っている。
- 先の例で量子化係数10で量子化する場合：

n	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
入力値	120	130	134	137	140	144	138	142	134
差分	120	10	4	3	3	4	-6	4	-8
差分の量子化	120	10	0	0	0	0	-10	0	-10
復号結果	120	130	130	130	130	130	120	120	110

- 誤差が累積してしまう。

局所復号器 (ローカルデコーダ) 2/2

- **符号器側で復号結果を再構成し**その差分を量子化して送る。

$$\text{差分} = \text{入力値} - \text{局所復号器出力}$$

$$\text{送信データ} = \text{差分を量子化したもの}$$

n	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
入力値	120	130	134	137	140	144	138	142	134
局所復号結果との差分	120	10	4	7	0	4	-2	2	-6
差分の量子化結果	120	10	0	10	0	0	0	0	-10
局所復号結果	120	130	130	140	140	140	140	140	130
復号結果	120	130	130	140	140	140	140	140	130

変換符号化 (1/4)

- ベクトル値を符号化するときに，各要素を符号化するのではなく，線形変換(直交変換)してから，係数を符号化する。
- s_i $(0 \leq i \leq N)$：N次元ベクトル$\boldsymbol{s}$の第i要素の値
- σ_i：s_iの標準偏差 (分散の平方根)
- s_iは連続値であるので，その値を完全に送ることはできない。
- s_iを符号化するとき，精度を一定とすると，その値を送るための平均符号長の下限は以下のようになる。

$$\log_2 \sigma_i + \text{const} \quad [\text{bit}]$$

- 従って，ベクトルの平均符号長の下限は以下のようになる。

$$\sum_{i=1}^{N} \log_2 \sigma_i + \text{const} \quad [\text{bit}]$$

変換符号化 (2/4)

- 画像を直交行列変換U することを考える。

$$\boldsymbol{t} = U\boldsymbol{s}$$

  Uは等長変換である。

$$\|U\boldsymbol{s}\| = \|\boldsymbol{s}\|$$

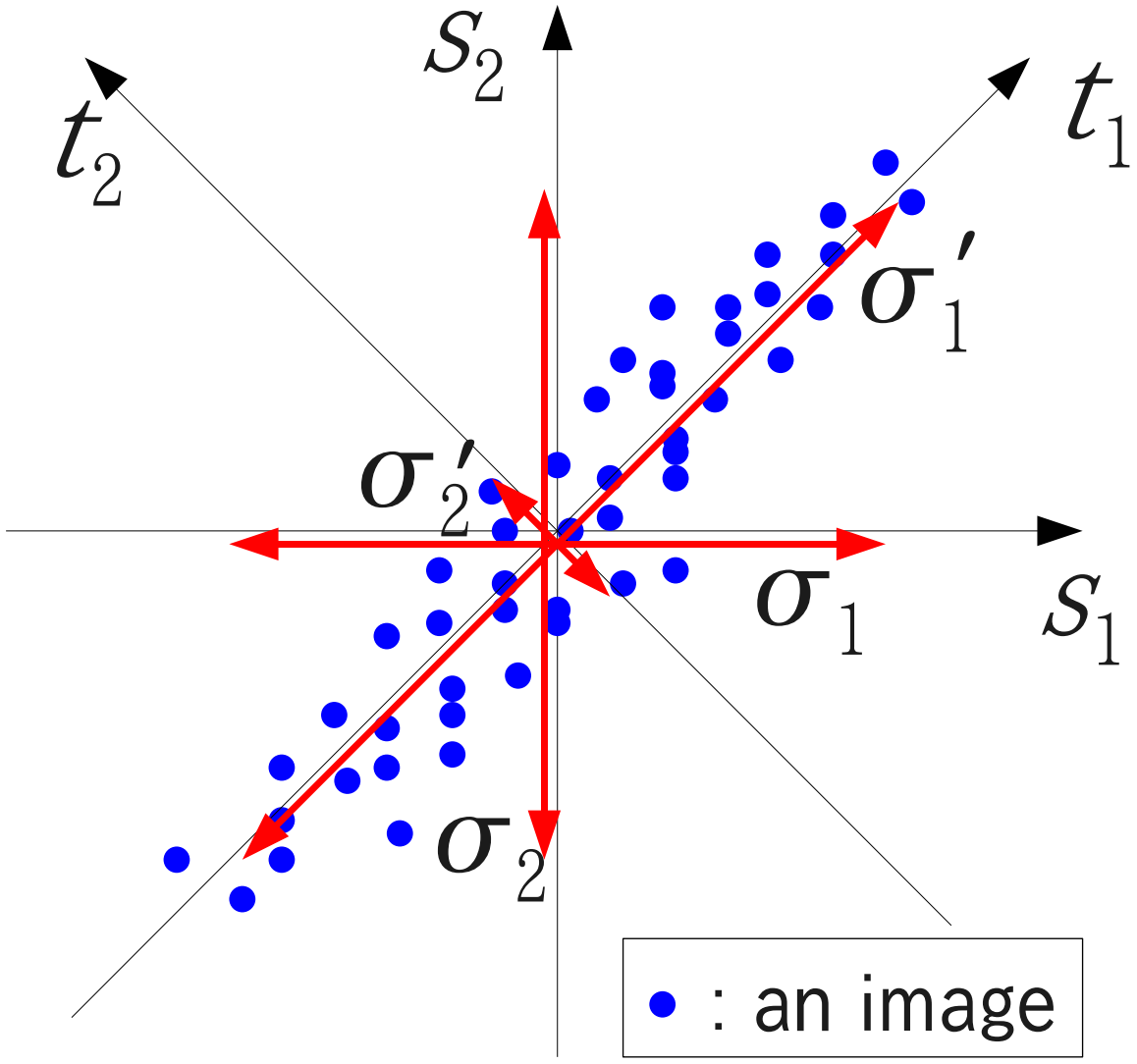
  - 座標値の精度と画像の精度の関係は$\boldsymbol{s}$と$U\boldsymbol{s}$で変わらない。
  - σ_i'：t_iの標準偏差 (座標値の標準偏差は変化する。)

- s_iを画素iにおける画素値と考える。
- 画像の統計的性質は移動不変であるから，画素値の分散σ_iの値は全て等しい。

$$\sigma_1 = \sigma_2 = \sigma_3 = \cdots = \sigma_N$$

- しかし，σ_i' は等しいとは限らない。

変換符号化 (3/4)



$\sigma_1 = \sigma_2$

$\sigma'_1 \neq \sigma'_2$

変換符号化 (4/4)

- 例：

  $\sigma_1 = \sigma_2 = \cdots = \sigma_{11} = \sqrt{10}$

  $\sigma'_1 = 10, \sigma'_2 = \sigma'_3 = \cdots = \sigma'_{11} = 1$

  分散の合計は同じ。

  $\sigma_1^2 = \sigma_2^2 + \cdots + \sigma_{11}^2 = 110$

  $(\sigma'_1)^2 + (\sigma'_2)^2 + (\sigma'_3)^2 + \cdots + (\sigma'_{11})^2 = 110$

- 平均符号長の下限の差は以下のようになる。

$$\left[11 \log_2 \sqrt{10}\right] - \left[\log_2 10 + 10 \log_2 1\right] \simeq 14.9[\text{bit}]$$

- 信号の分散(パワー)を少ない成分に集めると，平均符号長が短くなる。
- このための最適な変換は，**Karhunen-Loève transform (KLT)**.

Karhunen-Loève transform (KL変換)

- $\boldsymbol{X}$：信号を表すN次元の確率変数のベクトル
- $\{\boldsymbol{\phi}_n\}_{n=1}^N$：正規直交基底
- P_K：$\phi_1, \ldots, \phi_K$ で張られる部分空間への正射影行列$(K \leq N)$
- 任意のKに対して次式を最小化するように$\{\boldsymbol{\phi}_n\}_{n=1}^N$を選ぶ。

$$E_X \| P_K \boldsymbol{X} - \boldsymbol{X} \|^2$$

低い次元の部分空間で，元の信号を近似する。

- 初めの方の変換係数に，エネルギーが集約されている。
- 信号$\boldsymbol{x}$に対する，Karhunen-Loève transformの変換係数は，
  $\langle \boldsymbol{x}, \boldsymbol{\phi}_1 \rangle, \langle \boldsymbol{x}, \boldsymbol{\phi}_2 \rangle, \ldots, \langle \boldsymbol{x}, \boldsymbol{\phi}_N \rangle$ となる。
- しかし，直接このKL変換を使うことは難しい。

  確率分布が分からない。計算量が多い。
- KL変換の代わりに，離散コサイン変換を用いることが多い。

画像符号化の標準規格

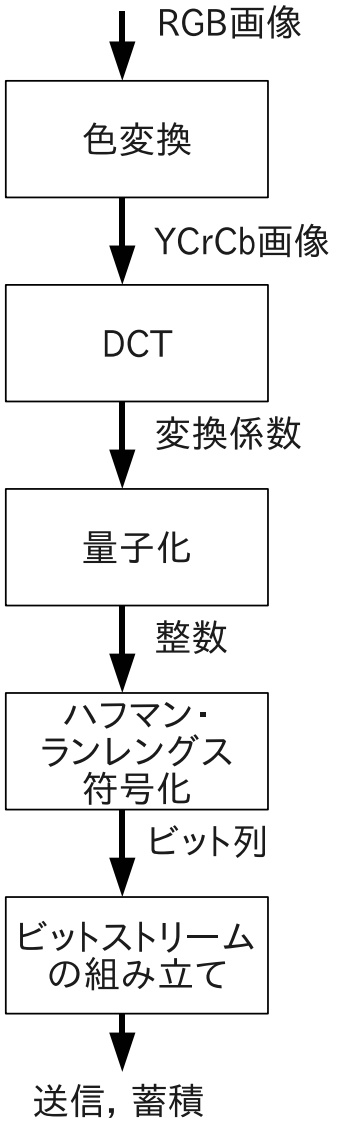
JPEG

- Joint Picture Experts Group
  - ISO (International Organization for Standardization，国際標準化機構)
  - ITU-T (International Telecommunication Union，国際電気通信連合)-(Telecommunication Standardization Sector，電気通信標準化部門)
- **静止画像**のための規格
  パソコン，ディジタルカメラ

MPEG

- Moving Picture Experts Group
- **動画像**のための規格
  ビデオCD，DVD，ディジタル放送
- ITUの，H.261 ～ H.264 とほぼ同様

静止画像符号化方式の標準 (JPEG)



- **色変換**:
  RGB空間から，YC_rC_b空間へ変換
- **離散コサイン変換 (DCT)**
- **量子化**：DCT係数の量子化
- **ハフマン・ランレングス符号化**：
  量子化された係数の符号化
- **ビットストームの組み立て**：
  各種符号列(画像のサイズ，テーブル，係数)を1つのストリームにする。

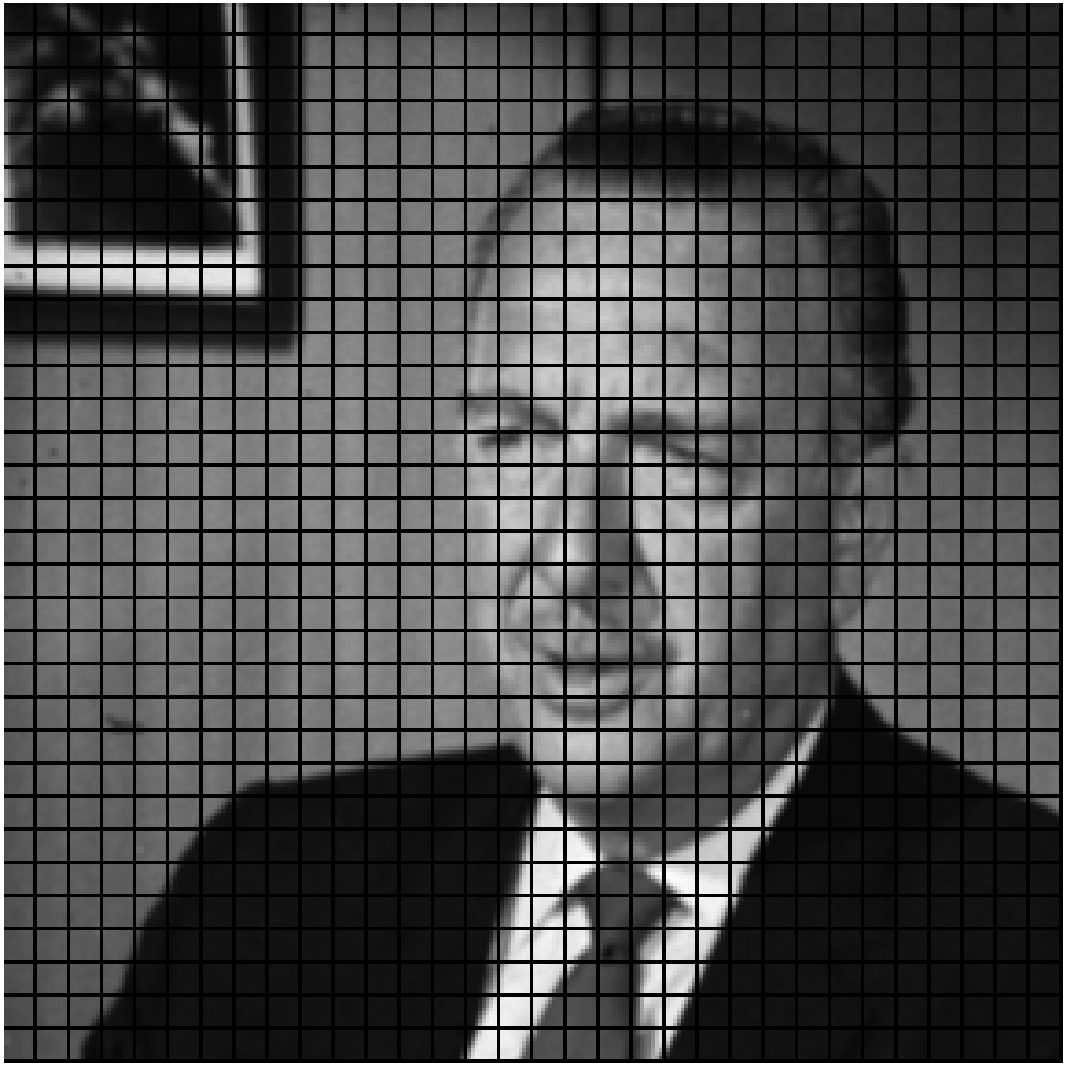
色変換

- 人間は色に関する分解能が明度に関する分解能よりも低い。
  - **錐体**：明るさを感じる。密度が高い (1.2〜1.4億個)。
  - **杆体**：色を感じる。密度が低い。
    (500〜600万個，赤60%，緑30%，青10%)
- 明るさの成分と色の成分を分けて処理する。
- **RGB–YCbCr変換**

$$
\begin{aligned}
Y &= 0.29900R + 0.58700G + 0.11400B \\
Cb &= -0.16874R - 0.33126G + 0.50000B + 128 \\
Cr &= 0.50000R - 0.41869G - 0.08131B + 128
\end{aligned}
$$

- **色に関するデータ (CbとCr) は間引く (1/2 × 1/2)。**

ブロック分割



縦8×横8画素のブロックに分割する

以降，基本的にはそれぞれのブロックを**独立に処理**する。

DCT

- 信号のエネルギーを少数の成分に集める。
- DCT基底(正規直交基底)：

$$f^n_{\mathrm{DCT}}(m) = C_n \frac{1}{\sqrt{N}} \cos\left\{\frac{\pi}{N} m \left(n + \frac{1}{2}\right)\right\},$$

where

$$C_n = \begin{cases} 1 & (n = 0) \\ \sqrt{2} & (n \neq 0) \end{cases}.$$

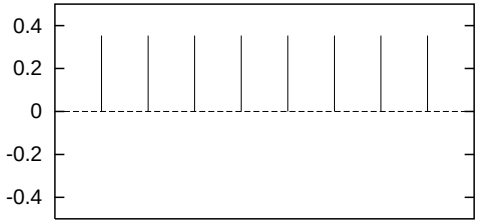
- **DCT係数 = 信号と$\boldsymbol{f}^n_{\mathrm{DCT}}$の内積**
- DCTを表す $A = [\boldsymbol{f}^0\ \boldsymbol{f}^1\ \ldots\ \boldsymbol{f}^{N-1}]^T$ は，直交行列
- 入力画像を表すベクトルを$\boldsymbol{x}$とすれば，変換係数$\boldsymbol{y}$は次のようになる。

$$\boldsymbol{y} = A\boldsymbol{x}$$

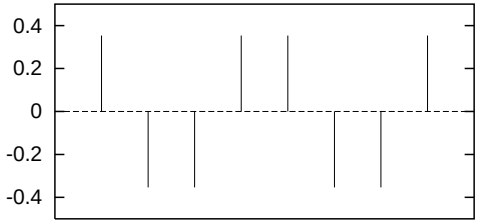
- 高速計算法が存在する。

DCT基底

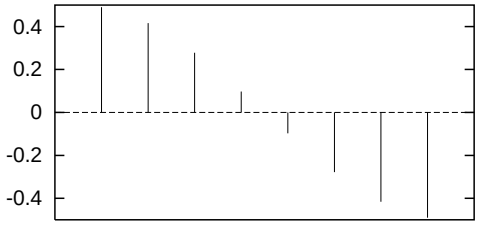
$n = 0$



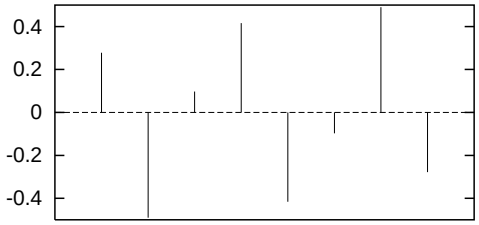
$n = 4$



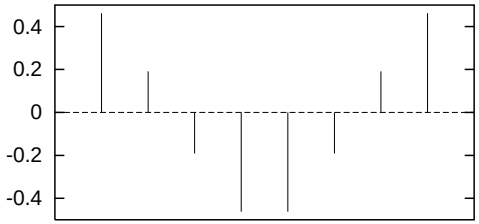
$n = 1$



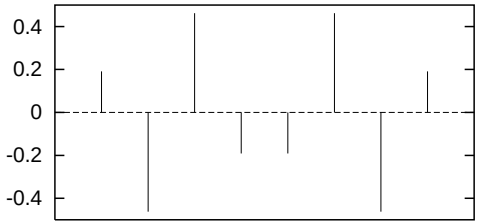
$n = 5$



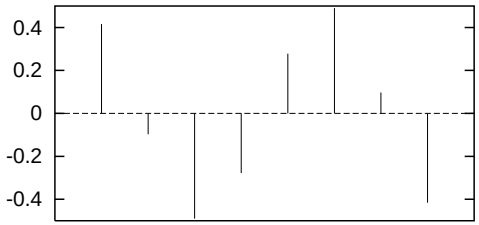
$n = 2$



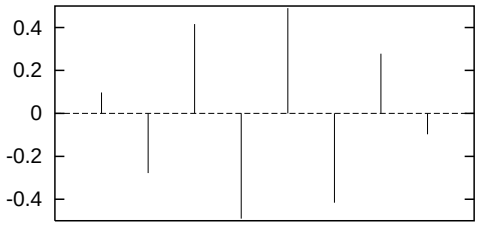
$n = 6$



$n = 3$

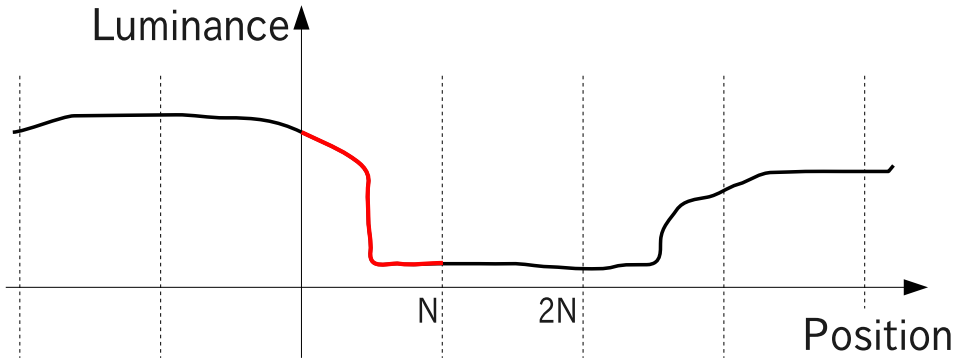


$n = 7$

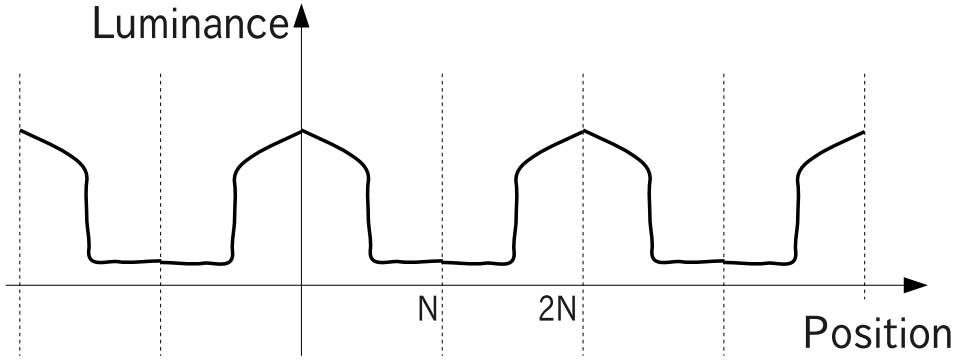


DCTとDFT (1/2)

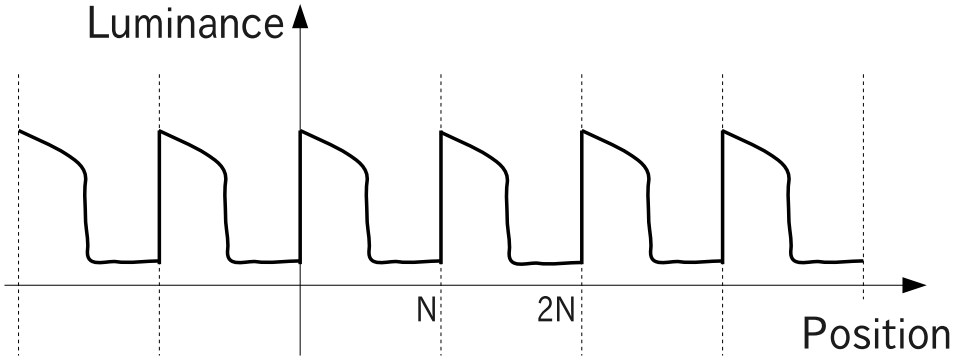
原信号



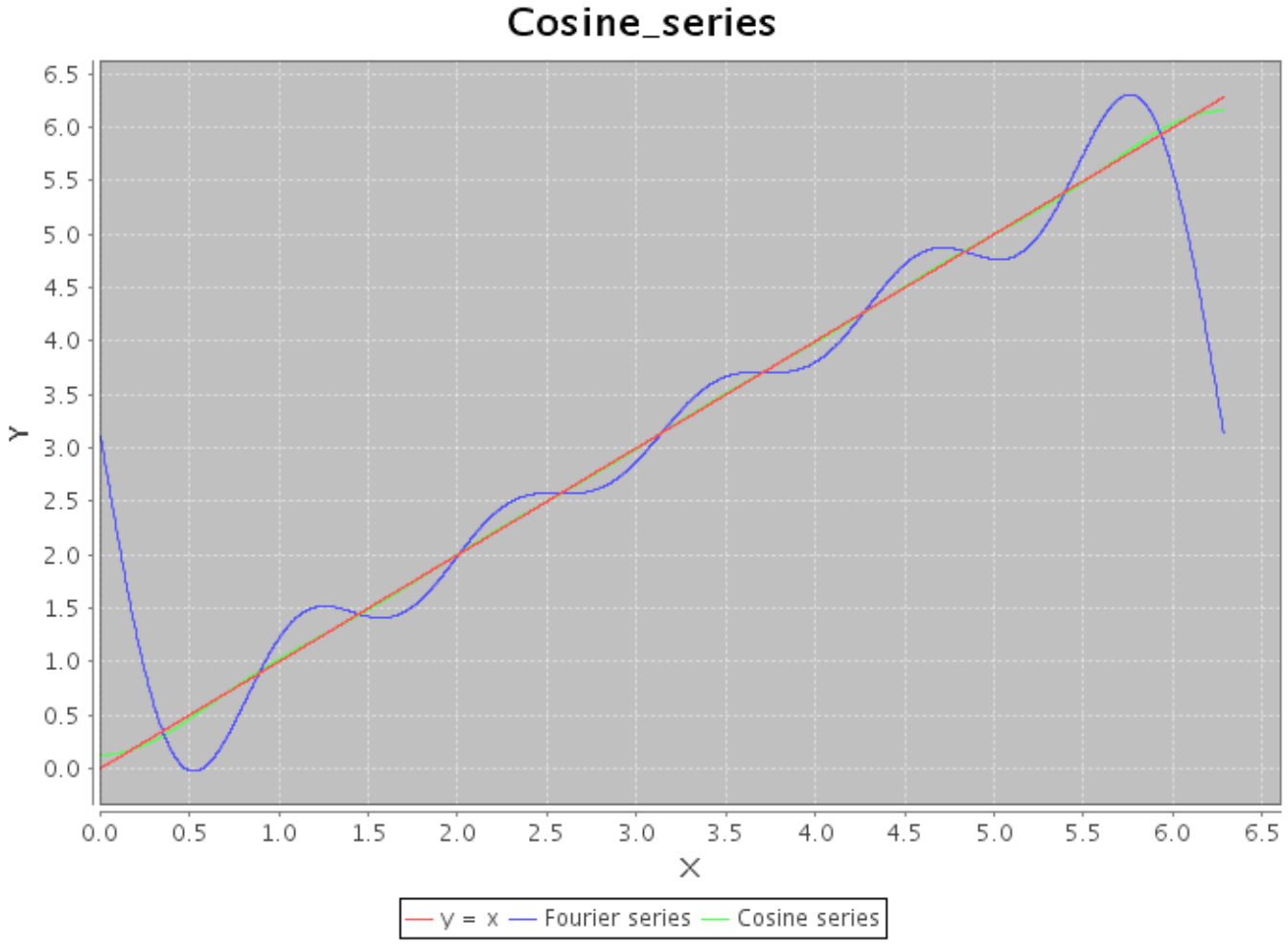
DCT



DFT



DCTとDFT (2/2)



- $y = x$を，DCT, DFT共にはじめの5項で近似

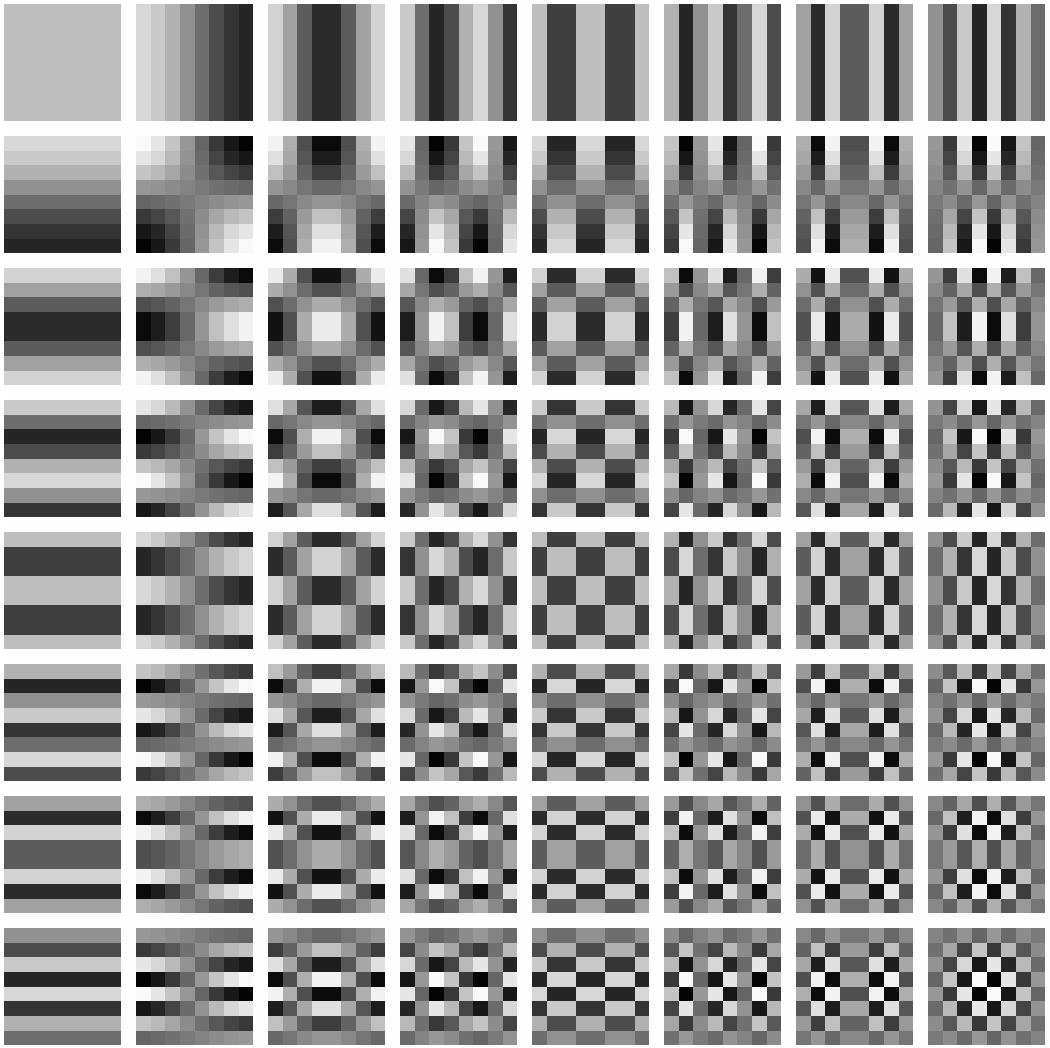
DCTとKLT

- X_i，X_j:第i成分と第j成分を表す確率変数
- X_iとX_jの相関

$$E_{X_i,X_j}X_iX_j$$

- この相関が，$\rho(<1)$を定数として，$|i-j|$の指数関数$\rho^{|i-j|}$で与えられるとき，$\rho \to 1$の極限で，KLTはDCTで与えられる
- 現実的にも画素間の相関は高い。
- **DCTを用いる問題**：**ブロック歪み (Block distortion)**
  - 変換ブロック境界に見える不連続性
  - 視覚的に目立つ。
  - 復号画像に対して画像処理(エッジ抽出など)をするときに誤差が生じる。

2次元DCT



ブロック歪み

原画像

JPEG

量子化

- **線形量子化**を用いる。
- DCT係数の周波数成分によって，異なる量子化係数を使う。
  高周波成分の誤差は低周波成分に比べて目立たない。
- YとCbCrには異なる量子化係数を使う。
- **量子化テーブル**：各係数に対する基準となる量子化係数のテーブル
- 量子化テーブルの値に定数を乗算して，実際の量子化係数とする。
- 標準の量子化テーブルの値は決められているが，JPEGデータの中に独自の量子化テーブルを埋め込むことが可能

量子化テーブル (輝度成分, Y)

V ＼ H	0	1	2	3	4	5	6	7
0	16	11	10	16	124	140	151	161
1	12	12	14	19	126	158	160	155
2	14	13	16	24	140	157	169	156
3	14	17	22	29	151	187	180	162
4	18	22	37	56	168	109	103	177
5	24	35	55	64	181	104	113	192
6	49	64	78	87	103	121	120	101
7	72	92	95	98	112	100	103	199

量子化テーブル (色差成分, CbCr)

V ＼ H	0	1	2	3	4	5	6	7
0	17	18	24	47	99	99	99	99
1	18	21	26	66	99	99	99	99
2	24	26	56	99	99	99	99	99
3	47	66	99	99	99	99	99	99
4	99	99	99	99	99	99	99	99
5	99	99	99	99	99	99	99	99
6	99	99	99	99	99	99	99	99
7	99	99	99	99	99	99	99	99

画像 (1ブロック)

V ＼ H	0	1	2	3	4	5	6	7
0	123	71	82	129	180	171	171	186
1	128	100	110	140	164	161	176	195
2	148	139	151	162	170	179	181	194
3	173	177	182	184	187	194	191	192
4	183	184	188	189	183	189	179	177
5	187	183	186	183	175	177	181	172
6	183	182	181	183	172	176	183	170
7	173	170	165	164	158	163	164	156

DCT変換係数

V ＼ H	0	1	2	3	4	5	6	7
0	1343.80	-75.40	1.22	20.43	22.00	10.22	4.90	1.92
1	-83.81	-112.16	1.56	23.44	40.38	-5.45	7.17	8.55
2	-97.00	-42.23	8.17	22.50	24.42	3.17	-3.14	6.69
3	-54.82	-2.59	-5.68	13.94	10.55	2.80	-0.70	4.09
4	0.75	2.12	-8.46	12.25	2.00	-5.34	2.42	-0.59
5	21.78	-3.74	-4.44	4.79	-1.96	0.32	-0.85	0.86
6	-8.79	1.67	-0.39	3.42	0.16	-0.69	3.57	1.04
7	-0.78	0.54	-4.23	1.48	0.38	0.09	2.75	-2.61

DCT変換係数 (量子化後)

DCT V ＼ H	0	1	2	3	4	5	6	7
0	84	-7	9	1	0	0	0	0
1	-7	-9	0	1	0	0	0	0
2	-70	-3	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	1	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

DC係数のハフマン符号化

- **対象**：DC係数$(0, 0)$を量子化した整数
- **差分符号化**：直前のブロックの値との差を送る。
- 出現確率に基づく本当のハフマン符号化ではない。
- その整数を表すためのbit長が**カテゴリ**になる。
- **カテゴリに符号を割り振る。**
- 1つの整数は，カテゴリの符号と値(+極性)を表すもののbit列となる。
- 標準のハフマンコードブック(符号化テーブル)が決まっている。
  次ページ：DC係数差分 (輝度成分) のハフマンコードブック
- JPEGデータの中に独自のコードブックを埋め込むことが可能
  カテゴリーは変えることはできない。

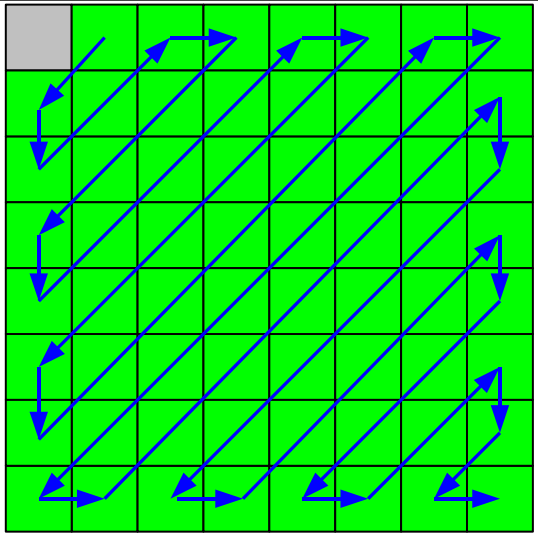
DC係数差分 (輝度成分) のハフマンコードブック

量子化値	カテゴリー	符号	付加ビット
0	0	00	なし
-1, 1	1	010	0, 1
-3, -2, 2, 3,	2	011	00, 01, 10, 11
-7, ..., -4, 4, ..., 7	3	100	000, ..., 111
-15, ..., -8, 8, ..., 15	4	101	0000, ..., 1111
-31, ..., -16, 16, ..., 31	5	110	00000, ..., 11111
-63, ..., -32, 32, ..., 63	6	1110	000000, ..., 111111
-127, ..., -64, 64, ..., 127	7	11110	0000000, ..., 1111111
-255, ..., -128, 128, ..., 255	8	111110	00000000, ..., 11111111
-511, ..., -256, 256, ..., 511	9	1111110	000000000, ..., 111111111
-1023, ..., -512, 512, ..., 1023	10	11111110	0000000000, ..., 1111111111
-2047, ..., -1024, 1024, ..., 2047	11	111111110	00000000000, ..., 11111111111

例: $0 \rightarrow \mathbf{00}$, $10 \rightarrow \mathbf{1011010}$, $-27 \rightarrow \mathbf{11000100}$

AC係数のランレングスハフマン符号化

- **対象**：AC係数($(0, 0)$以外)を量子化した整数
- **ジグザグスキャン**：右図
- 0が多く続く
  $\Rightarrow$ ランレングス符号化



- ジグザグスキャン中に0でない値が現れたときに,
  - それまで続いてきた0の数
  - 自身の値のbit数(サイズ)

  に対して1つの符号を与える(2次元ハフマン符号)。
- その符号の後に自身の値(+極性)を表す付加ビットを出力する。
- 走査の残りが全て0の場合は，そのための符号を出力して，そのブロックに対する出力を終了する。

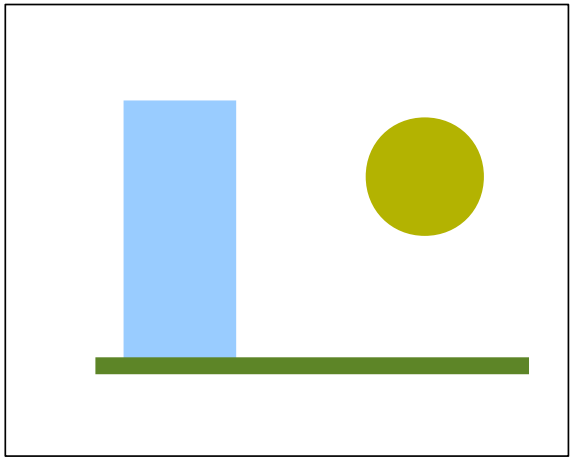
動画像符号化 (MPEG)

- **動画像**：時系列上の複数の画像
- **MPEG = 動き補償予測 + JPEG**
- **動き補償予測**
  - 時間的に近い画像間の類似性が高い。⇒ **予測符号化**
  - **対象画像** (target image)：これから符号化する画像
  - **参照画像** (reference image)：対象画像を符号化する前に送信済みで，対象画像を符号化するときに利用できる画像
  - **予測画像**：参照画像から生成する対象画像を予測した画像
  - 参照画像そのものを予測画像とすると，画像の内の動きによる誤差が無視できない。
  - 動きを補償して予測画像を生成する。
- 基本的には復号のための規格(符号化器の設計には自由度がある)

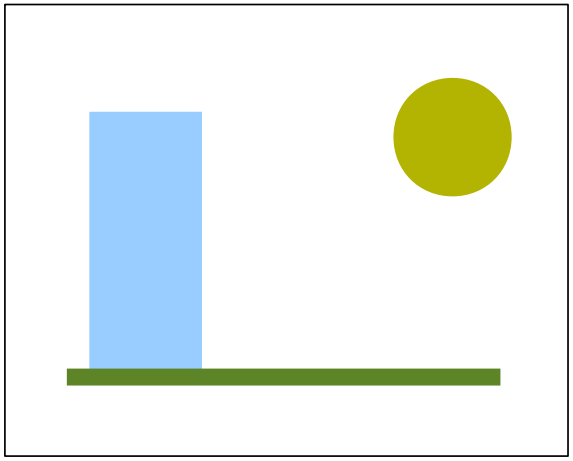
動き補償予測

- 動き補償予測画像の生成法
  - 対象画像をブロックで分割する (MPEG 2では，16×16画素)。
  - 参照画像の中から上記の各ブロックの画像と似ている領域(ブロックと同じ大きさ)を探す。
  - **動きベクトル**：ブロックの座標と似ている部分の座標の差
- 符号器側では，参照画像は復号済みなので，**動きベクトルさえあれば，予測画像を生成できる。**
- 動きベクトルはハフマン符号化によって送る。
- 動きベクトル：1画像，1/2画素単位

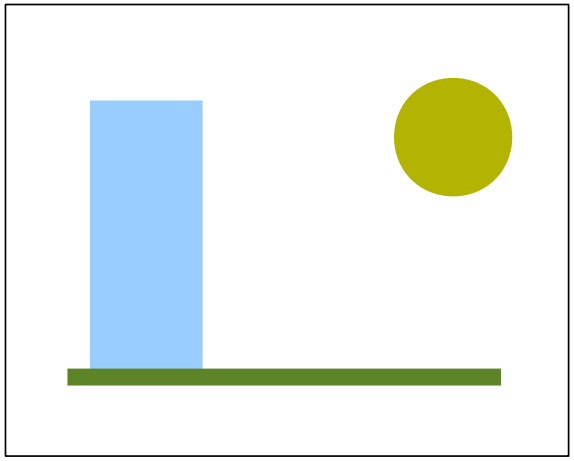
動き補償予測



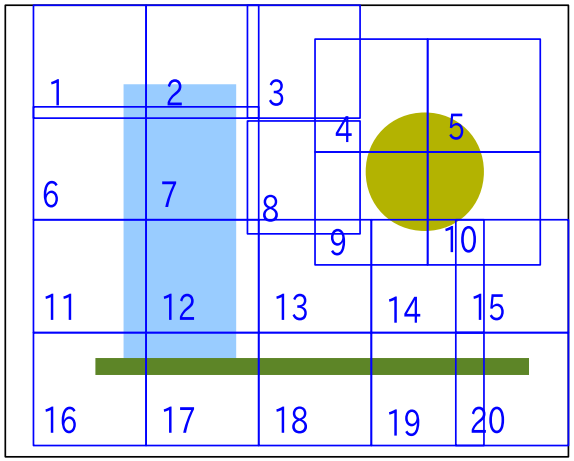
参照画像



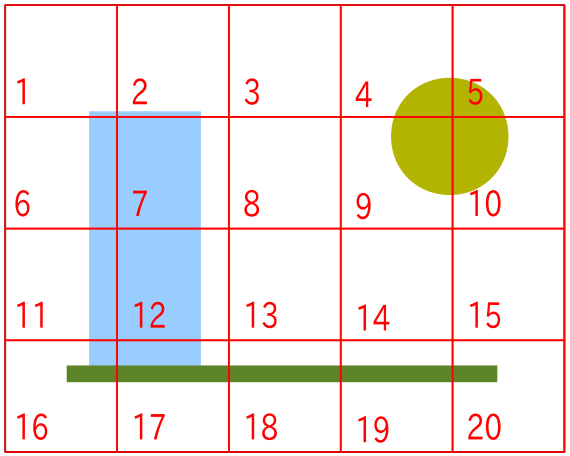
対象画像



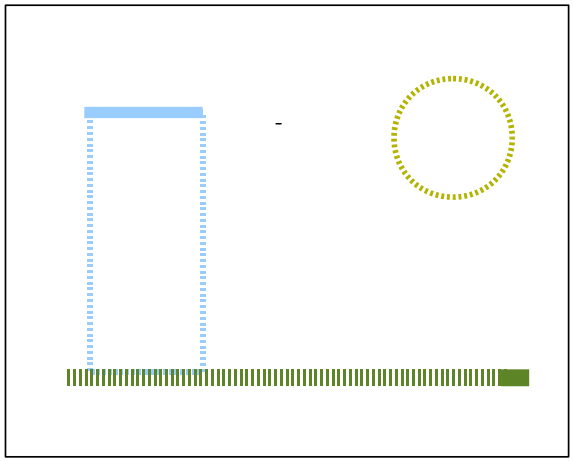
予測画像



参照画像

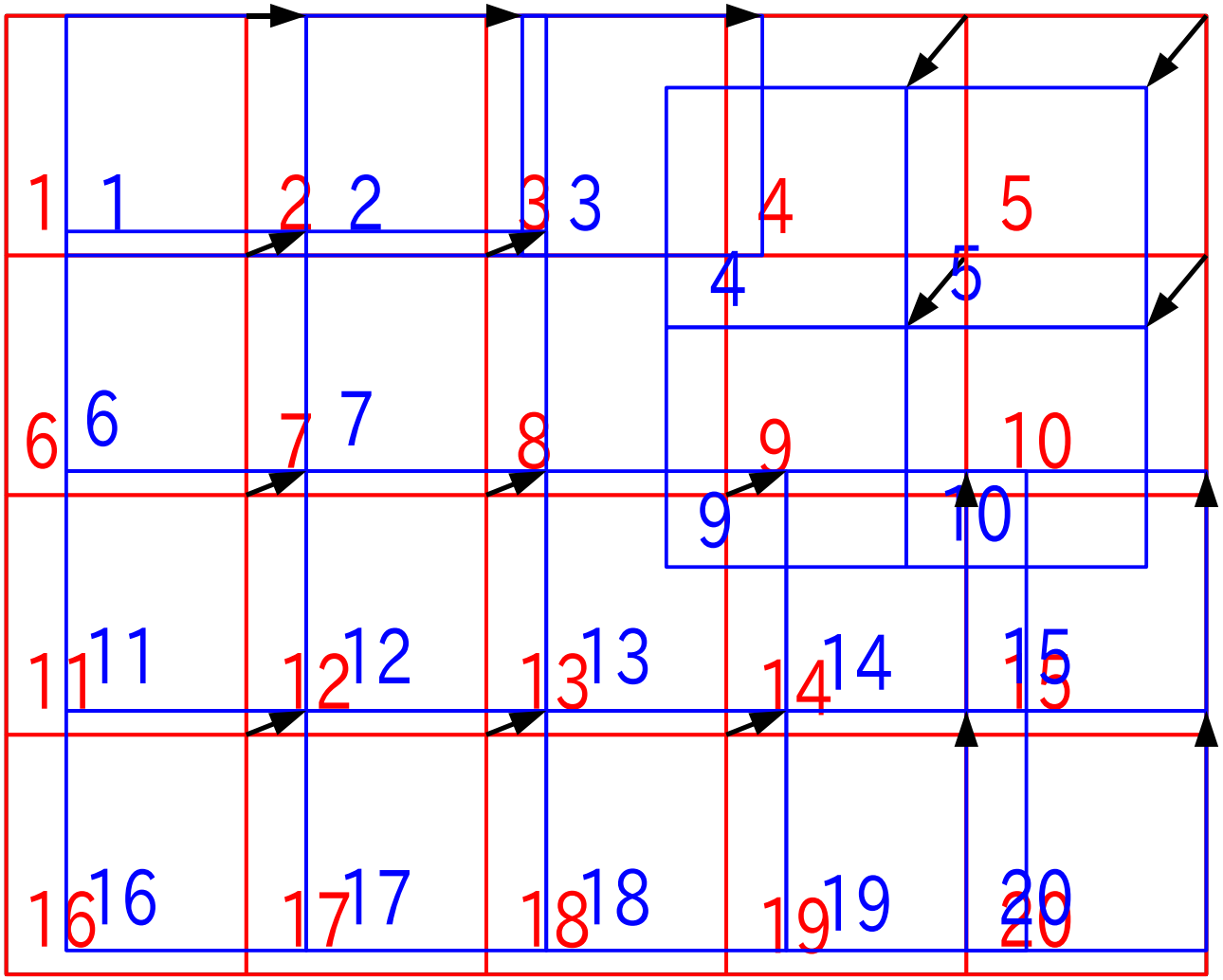


対象画像



差分画像

動きベクトル


1 1
2 2
3 3
4
5
4
5
6 6
7 7
8
9
10
9
10
11 11
12 12
13 13
14 14
15
15
16 16
17 17
18 18
19 19
20
20

動き補償予測 (実画像)

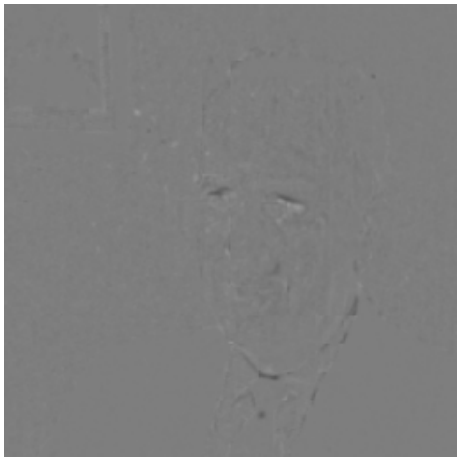
参照画像

対象画像

対象画像 – 参照画像

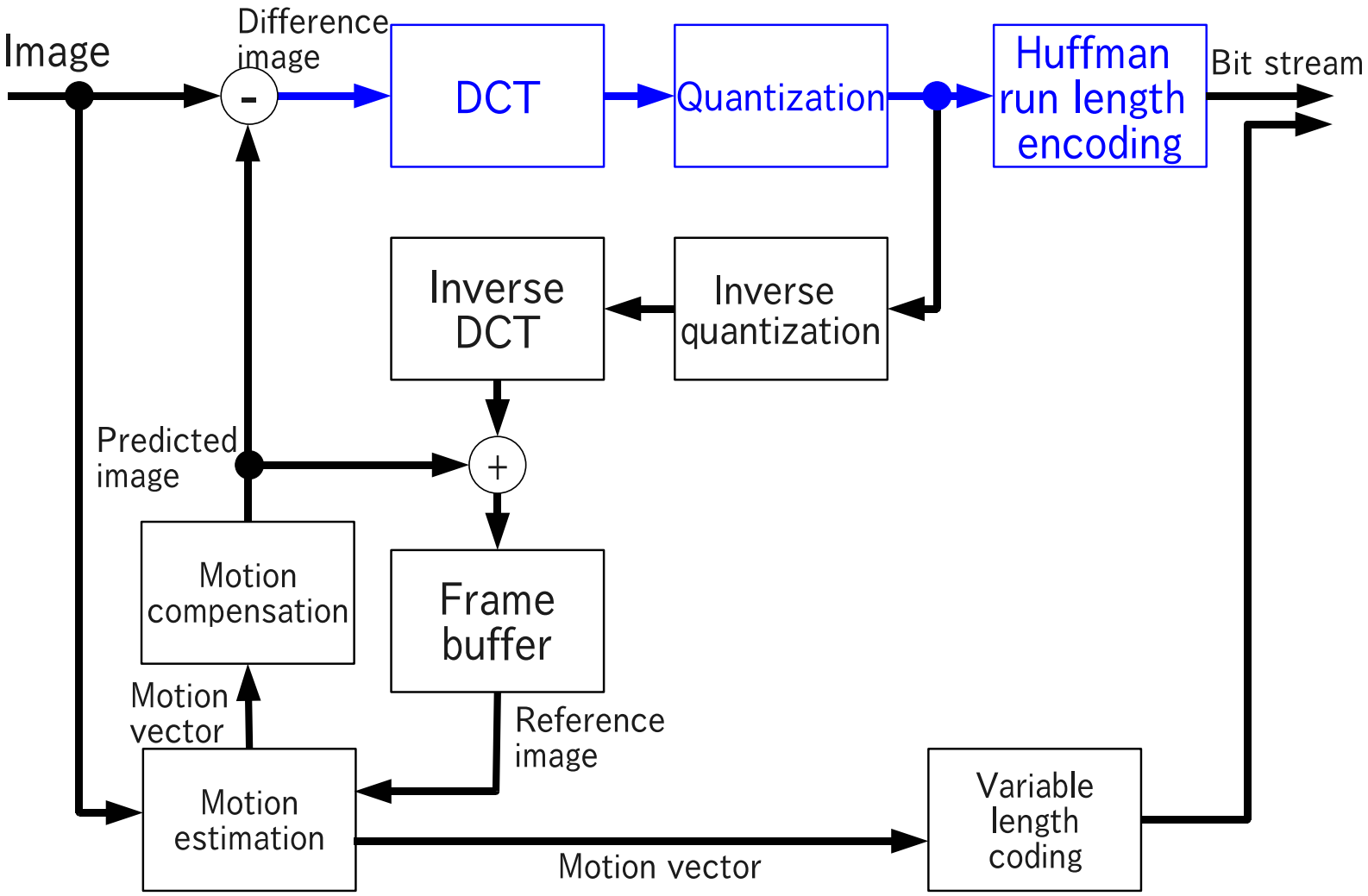
予測画像

対象画像 (with MV)

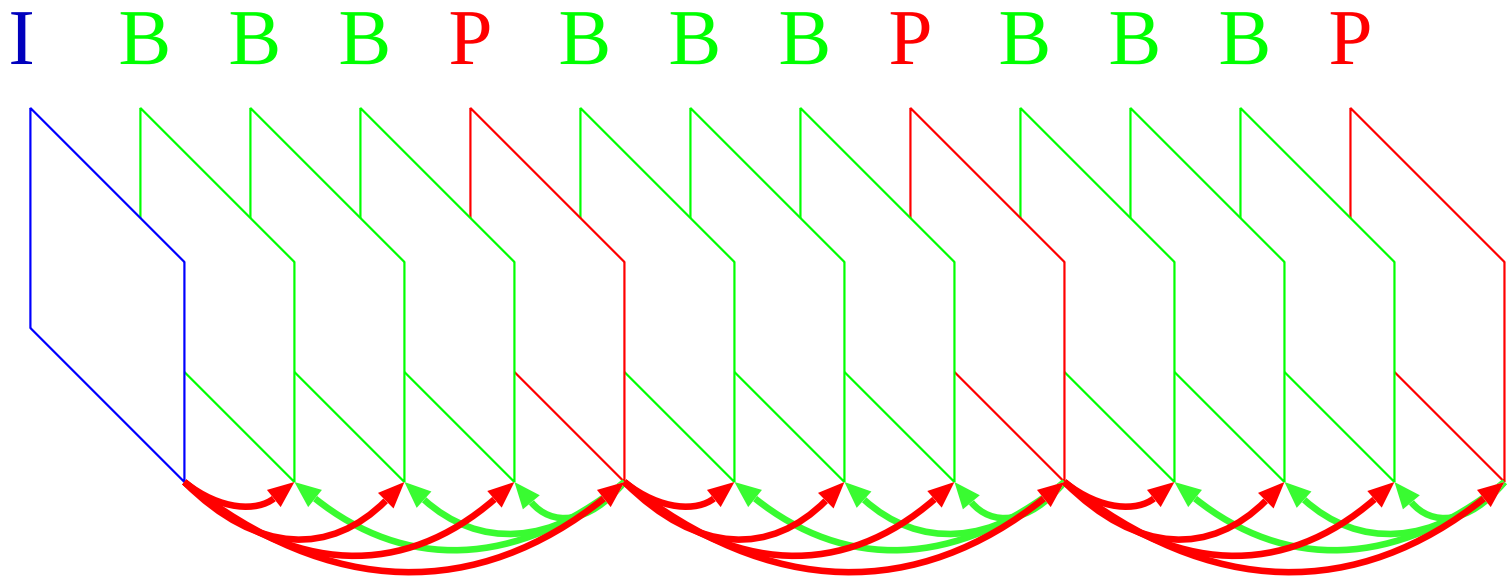


対象画像 – 予測画像

MPEGエンコーダのブロック図



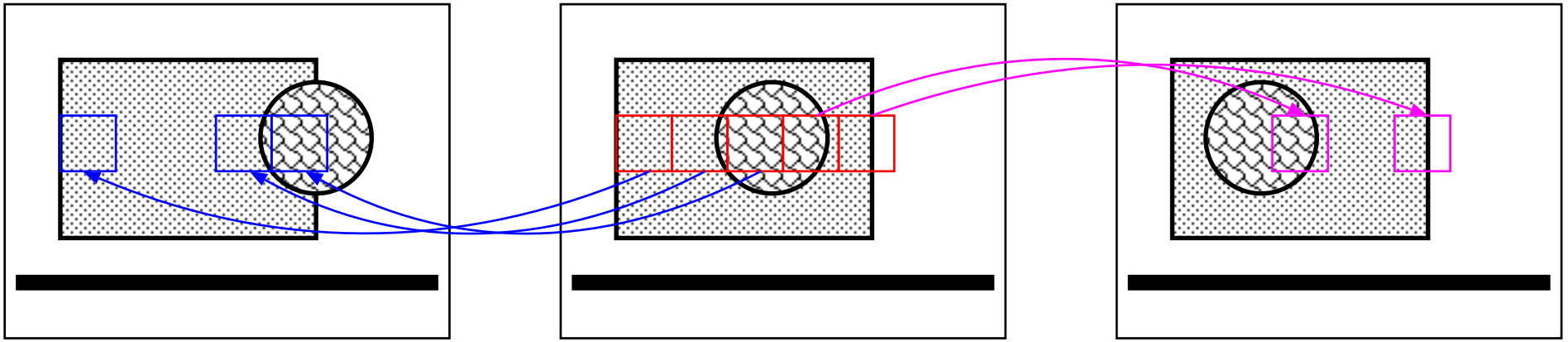
I, P, Bピクチャ (フレーム)



- Iピクチャ：動き予測補償を行わない.
- Pピクチャ：前のIまたはPピクチャで動き予測補償
- Bピクチャ：前後のIまたはPピクチャで動き予測補償

  前のフレームでは隠れていた部分も補償できる.

Bピクチャ (フレーム)



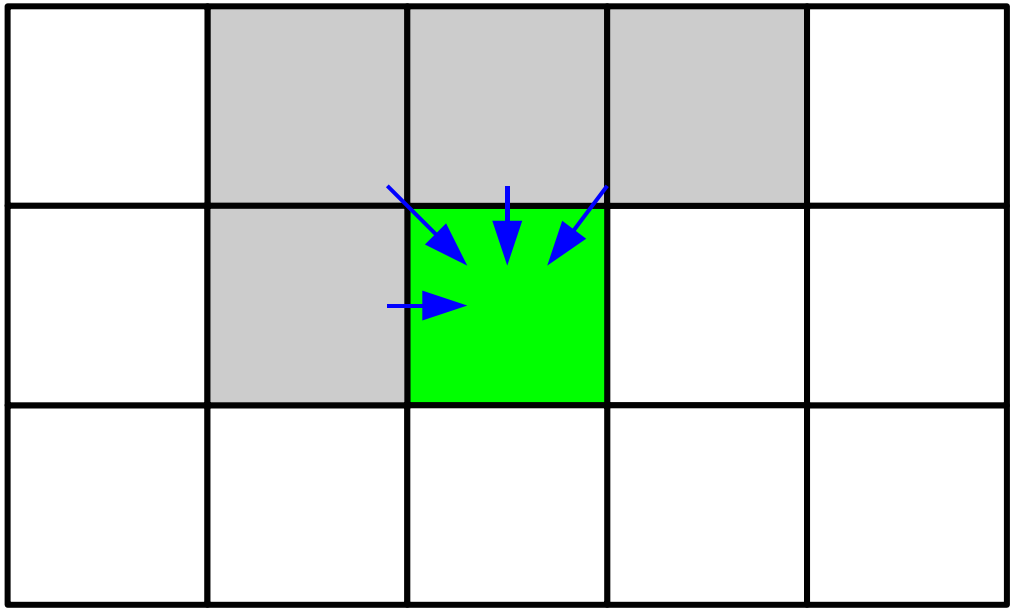
- 中央のフレームの予測画像を生成する。
- 時間的に後のフレームを使うことによって,
  前のフレームでは**隠れていた部分**の情報を使うことができる.

MPEG4 AVC(Advance Video Coding) H.264

- MPEG 1 (H.261)：ビデオCD (1.5Mbit/s)
- MPEG 2 (H.262)：DVD，地上ディジタル放送
- MPEG 4 (H.263)：携帯電話
- MPEG 4 AVC (H.264)：Blu-ray，ワンセグ，ビデオカメラ
  - **MPEG2に比べて，同じ画質でデータ量を半分にできる。**
    (できない場合もある。)
  - 遅れて地上ディジタル放送を始める国は，MPEG 4 AVC 方式を用いている(日本より綺麗な画像を見ている)。
  - MPEG 4 HVC (High-Performance Video Coding) (H.265)
    が策定中。
    ただ，しらばくはMPEG 4 AVCが最新機器の主流

H.264のIピクチャの処理

- DCTを使わず，フレーム内予測を利用する。
- 画像に変化があるところ：4×4のブロック
- 画像に変化がないところ：16×16のブロック
- **予測方向**を与える。
  (4×4場合8種類，16×16のブロックの場合4種類)



H.264の動き補償予測

- ブロックサイズが可変

  16×16，16×8，8×16，8×8，8×4，4×8，4×4
- 1/4画素精度の動きベクトル
- 動きベクトルの予測
- 複数(2つ以上の)画像を参照画像とすることが可能
- 参照画像(復号画像)の生成にデブロッキング(ブロック歪み除去)フィルタを用いることができる。
- デブロッキングフィルタは標準化されている，参照画像を生成する予測ループ内に入れる。
- 予測画像にはブロック歪みが残る。**(後述)**

H.264の差分画像の処理

- 4×4のDCT
  - 8×8の方が符号化効率が良い場合も多い。
- 整数係数DCT
  - 16bit演算で規定されている。
  - 符号化器のローカルデコーダと復号器間の齟齬の問題を解決
  - 差分量子化の場合，齟齬が蓄積していく可能性があった。
- 変換係数の符号化のために，2種類のエントロピー符号化が規定されている。
  - ハフマン符号化
  - 算術符号化

H.264のエントロピー符号化

- 指数ゴロム符号
  - 基本的なシンタックス要素に利用する。
  - 整数の符号化において，1の前に続く0の数でその整数の桁数を表す。
- CAVLC (Context-Adaptive Variable Length Coding)
  (コンテキストについては，JPEG 2000で説明)
  - DCT係数の符号化に利用する。
  - ジグザグスキャン
  - 0ランの情報，最後の絶対値1のランの情報を加えるが，値とランの情報は分けて格納する。
- CABAC (Context-Adaptive Binary Arithmetic Coding)
  - 整数値をbinaryに変換
  - Q-coder系 (64種類のQ_e)

多重解像度解析

- $\phi(t)$: **スケーリング関数** (解像度 0)
- $\phi(2^{-i}t - n)$ は，幅を $\phi(t)$ を幅を 2^i 倍に伸ばして，t 軸方向に $2^i n$ だけ平行移動した関数。
- i が小さいほど細かい。
- V_i : $\phi(2^{-i}t - n)$ $(n = 0, \pm 1, \pm 2, \ldots)$ で張られる部分空間
- $\phi(2^{-i}t - n)$ は，V_i の基底関数
- V_i は i が小さいほど解像度が高い。
- V_{i+1} は V_i に含まれる。
  解像度の低い空間は，解像度の高い空間に含まれる。
- $V_i = V_{i+1} + W_{i+1}$ となる空間が存在する。
  W_i : $\psi(2^i t - n)$ $(n = 0, \pm 1, \pm 2, \ldots)$ で張られる部分空間
- $\psi(t)$: **ウェーブレット関数**

離散双直交ウェーブレット変換

- $\tilde{\phi}(t)$ を $\phi(t)$ の双直交基底，$\tilde{\psi}(t)$ を $\psi(t)$ の双直交基底とする。

$$\begin{aligned}
\langle \phi(t-k), \tilde{\phi}(t-l) \rangle &= \delta_{k-l} \\
\langle \psi(t-k), \tilde{\psi}(t-l) \rangle &= \delta_{k-l} \\
\langle \phi(t-k), \tilde{\psi}(t-l) \rangle &= 0 \\
\langle \psi(t-k), \tilde{\phi}(t-l) \rangle &= 0
\end{aligned}$$

- Dilation equation

$$\phi(t) = \sum_{m=-\infty}^{\infty} h(m)\phi(2t-m) \tag{1}$$

$$\tilde{\phi}(t) = \sum_{m=-\infty}^{\infty} \tilde{h}(m)\tilde{\phi}(2t-m) \tag{2}$$

$$\psi(t) = \sum_{m=-\infty}^{\infty} g(m)\phi(2t-m) \tag{3}$$

$$\tilde{\psi}(t) = \sum_{m=-\infty}^{\infty} \tilde{g}(m)\tilde{\phi}(2t-m) \tag{4}$$

- V_{-1}の任意の信号$f_{-1}(t)$を考える。

$$f_{-1}(t) = \sum_{k=-\infty}^{\infty} a(-1,k)\phi(2t-k) \tag{5}$$

- $V_{-1} = V_0 + W_0$より，$f_0(t) \in V_0$と$g_0(t) \in W_0$が存在して，

$$f_{-1}(t) = f_0(t) + g_0(t)$$

となる。

- それぞれ，以下のように展開できる。

$$f_0(t) = \sum_{n=-\infty}^{\infty} a(0,n)\phi(t-n)$$

$$g_0(t) = \sum_{n=-\infty}^{\infty} b(0,n)\psi(t-n)$$

- 双直交基を使うと，$\langle g_0(t), \tilde{\phi}(t-n)\rangle\rangle = 0$, $\langle f_0(t), \tilde{\psi}(t-n)\rangle\rangle = 0$ より，

$$\begin{aligned} a(0,n) &= \langle f_0(t), \tilde{\phi}(t-n)\rangle = \langle f_0(t) + g_0(t), \tilde{\phi}(t-n)\rangle \\ &= \langle f_{-1}(t), \tilde{\phi}(t-n)\rangle \end{aligned}$$

および

$$\begin{aligned} b(0,n) &= \langle g_0(t), \tilde{\psi}(t-n)\rangle = \langle f_0(t) + g_0(t), \tilde{\psi}(t-n)\rangle \\ &= \langle f_{-1}(t), \tilde{\psi}(t-n)\rangle \end{aligned}$$

が成立する。

従って，式 (5)，(2)，(4) より，

$$
\begin{aligned}
a(0,n) &= \langle f_{-1}(t), \tilde{\phi}(t-n) \rangle \\
&= \left\langle f_{-1}(t), \sum_{m=-\infty}^{\infty} \tilde{h}(m)\tilde{\phi}(2t-2n-m) \right\rangle \\
&= \left\langle \sum_{k=-\infty}^{\infty} a(-1,k)\phi(2t-k), \sum_{m=-\infty}^{\infty} \tilde{h}(m)\tilde{\phi}(2t-2n-m) \right\rangle \\
&= \sum_{m=-\infty}^{\infty} a(-1,2n+m)\tilde{h}(m) \\
&= \sum_{m=-\infty}^{\infty} a(-1,m)\tilde{h}(m-2n)
\end{aligned}
$$

および

$$
\begin{aligned}
b(0,n) &= \langle f_{-1}(t), \tilde{\psi}(t-n) \rangle \\
&= \left\langle f_{-1}(t), \sum_{m=-\infty}^{\infty} \tilde{g}(m)\tilde{\phi}(2t-2n-m) \right\rangle \\
&= \left\langle \sum_{k=-\infty}^{\infty} a(-1,k)\phi(2t-k), \sum_{m=-\infty}^{\infty} \tilde{g}(m)\tilde{\phi}(2t-2n-m) \right\rangle \\
&= \sum_{m=-\infty}^{\infty} a(-1,2n+m)\tilde{g}(m) \\
&= \sum_{m=-\infty}^{\infty} a(-1,m)\tilde{g}(m-2n)
\end{aligned}
$$

が成立する。

- この式は，$a(-1,n)$から$a(0,n)$，$b(0,n)$へ信号を分解していることを表している。

- $a(-1,n)$ と $\tilde{h}(n)$ または $\tilde{g}(n)$ との畳み込み和は,

$$c(n) = \sum_{m=-\infty}^{\infty} a(-1,m)\tilde{h}(m-n)$$

$$d(n) = \sum_{m=-\infty}^{\infty} a(-1,m)\tilde{g}(m-n)$$

  となる。$a(0,n)$, $b(0,n)$ は，$c(n)$ と $d(n)$ の偶数点で構成されている。
  $\Rightarrow a(-1,n)$ に対して，$\tilde{h}(n)$，$\tilde{g}(n)$ というフィルタをかけてから，
  ダウンサンプリング (間引き) をしたものに等しい。
- 画像符号化では，$a(-1,n)$ を画素値と考え，$a(0,n)$ をその低周波分，$b(0,n)$ をその高周波分と考える。
- 低周波分に信号のパワーが集まっているため，変換符号化の原理でデータ量を圧縮できる。
- **$a(0,n)$ と $b(0,n)$ から，$a(-1,n)$ を再構成できる** (次ページ)。

- 逆に，$f_{-1}(t) = f_0(t) + g_0(t)$ より，次式が成立

$$
\begin{aligned}
f_{-1}(t) &= \sum_{k=-\infty}^{\infty} a(-1,k)\phi(2t-k) \\
&= \sum_{n=-\infty}^{\infty} a(0,n)\phi(t-n) + \sum_{n=-\infty}^{\infty} b(0,n)\psi(t-n) \\
&= \sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} a(0,n)h(m)\phi(2t-2n-m) \\
&\quad + \sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} b(0,n)g(m)\phi(2t-2n-m) \\
&= \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} a(0,n)h(m-2n)\phi(2t-m) \\
&\quad + \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} b(0,n)g(m-2n)\phi(2t-m)
\end{aligned}
$$

- 従って，次式が成立する。

$$
\begin{aligned}
a(-1,k) &= \sum_{n=-\infty}^{\infty} a(0,n)h(k-2n) + \sum_{n=-\infty}^{\infty} b(0,n)g(k-2n) \\
&= \sum_{n=-\infty}^{\infty} a_u(0,n)h(k-n) + \sum_{n=-\infty}^{\infty} b_u(0,n)g(k-n)
\end{aligned}
$$

- ここで，$a_u(0,n)$ と $b_u(0,n)$ は，$a(0,n)$ と $b(0,n)$ を**アップサンプリング**(値と次の値の間に0を挿入する処理)をしたものである。

$$
a_u(0,n) = \begin{cases} a(0,n/2) & (n:\text{even}) \\ 0 & (n:\text{odd}) \end{cases}
$$

$$
b_u(0,n) = \begin{cases} b(0,n/2) & (n:\text{even}) \\ 0 & (n:\text{odd}) \end{cases}
$$

離散双直交ウェーブレット変換のフィルタ特性

- 双直交関係式に，diliation equationを代入すると，次式が成立する。

$$\sum_{m=-\infty}^{\infty} h(m+2n)\tilde{h}(m) = \delta(n)/2$$

$$\sum_{m=-\infty}^{\infty} g(m+2n)\tilde{g}(m) = \delta(n)/2$$

$$\sum_{m=-\infty}^{\infty} h(m+2n)\tilde{g}(m) = 0$$

$$\sum_{m=-\infty}^{\infty} g(m+2n)\tilde{h}(m) = 0$$

離散双直交ウェーブレット変換のフィルタ特性

- $H(\omega)$，$\tilde{H}(\omega)$，$G(\omega)$，$\tilde{G}(\omega)$を，$h(n)$，$\tilde{h}(n)$，$g(n)$，$\tilde{g}(n)$を離散フーリエ変換したものとする。

$$\text{離散フーリエ変換：} \quad F(\omega) = \sum_{n=-\infty}^{\infty} f(n)e^{-i\omega n}$$

- 先の関係を離散フーリエ変換すると，次式が成立する。

$$\begin{aligned}
H(\omega)^*\tilde{H}(\omega) + H(\omega+\pi)^*\tilde{H}(\omega+\pi) &= 1 \\
G(\omega)^*\tilde{G}(\omega) + G(\omega+\pi)^*\tilde{G}(\omega+\pi) &= 1 \\
H(\omega)^*\tilde{G}(\omega) + H(\omega+\pi)^*\tilde{G}(\omega+\pi) &= 0 \\
G(\omega)^*\tilde{H}(\omega) + G(\omega+\pi)^*\tilde{H}(\omega+\pi) &= 0
\end{aligned}$$

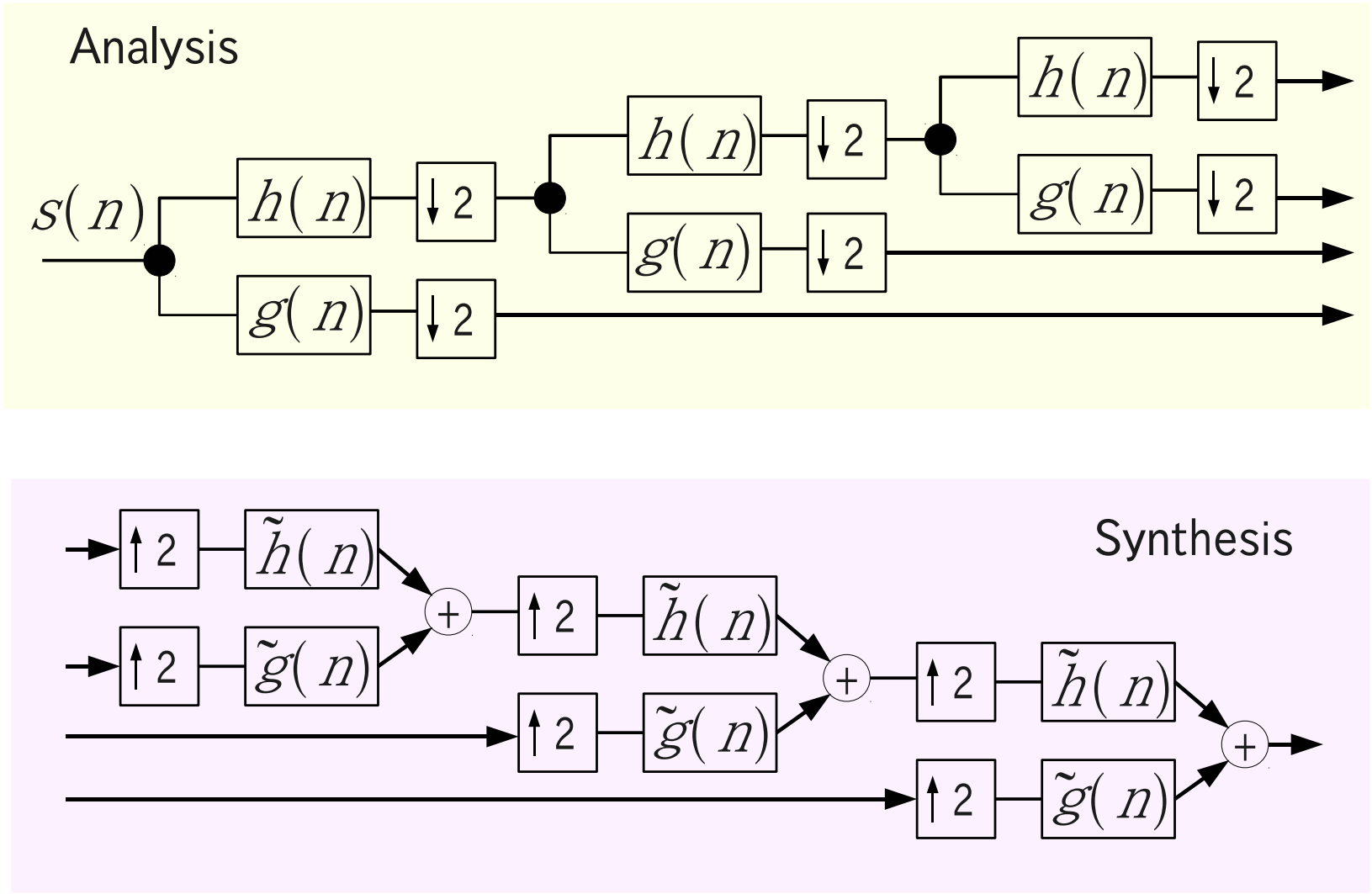
- ウェーブレット変換係数は，信号に対してこのような周波数特性をもつフィルタを施したものと考えることができる。
- 一般に，$h(n)$はローパスフィルタ，$g(n)$はハイパスフィルタになる。

ウェーブレットの例

- Haarウェーブレット (1と-1で構成されている)
- CDF97 (Cohen, Daubechies, Feauveaの9-7)フィルタ
  - 線形位相 (位相変化が周波数に比例する。)
    フィルタ係数が対称
    画像処理では必要な性質
    (音声処理では，人間の聴覚は位相を認識できないので不要)
  - 画像符号化で利用されることが多い。

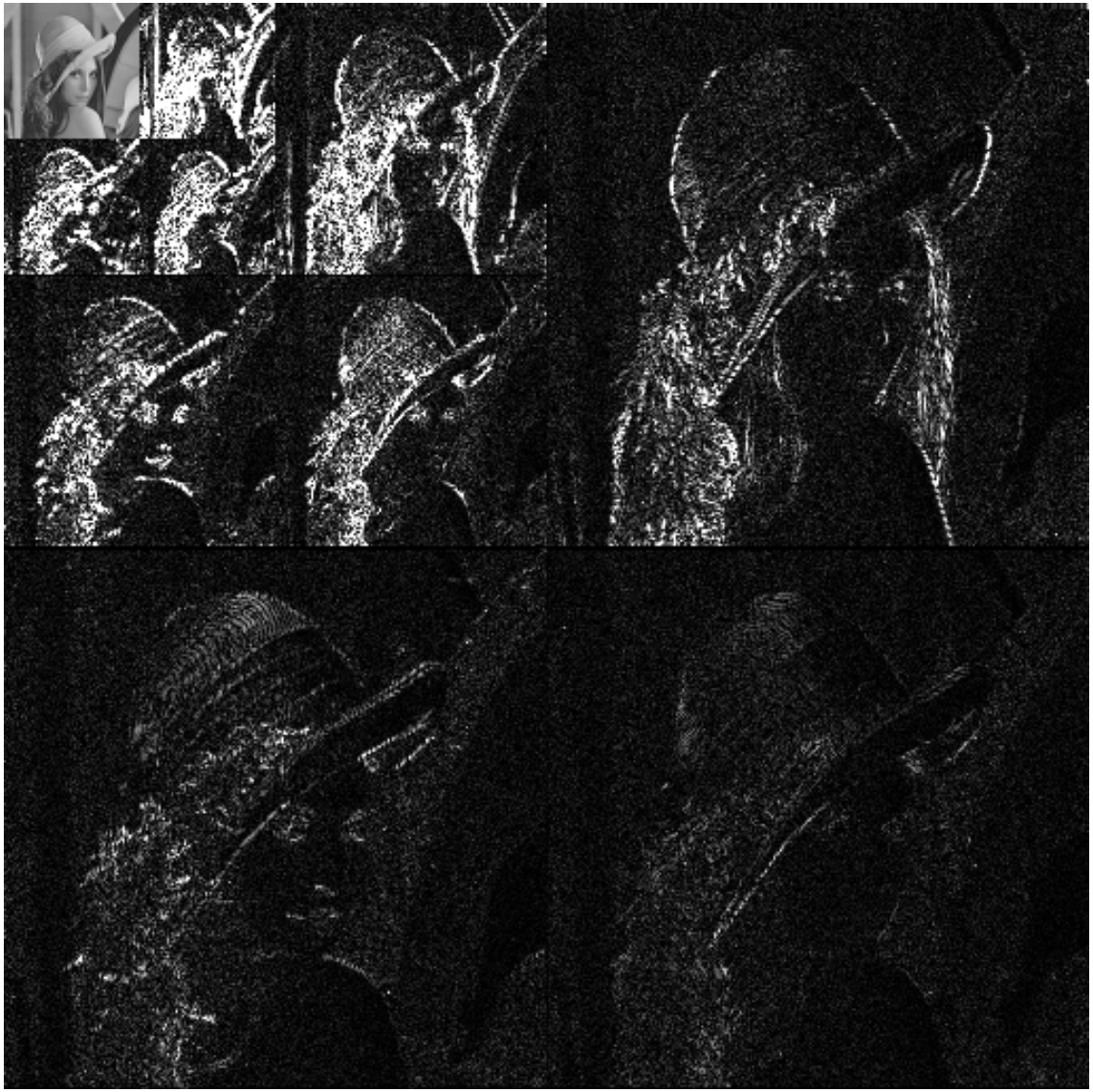
n	$h(n)$	$g(n)$
0	0.6029490182363579	1.115087052456994
±1	0.2668641184428723	-0.5912717631142470
±2	-0.07822326652898785	-0.05754352622849957
±3	-0.01686411844287495	0.09127176311424948
±4	0.02674875741080976	

ウェーブレットフィルタバンクの例



低周波の信号は繰り返し低周波と高周波に分解する (上図は3段の例)。

ウェーブレット変換



JPEG2000 (ウェーブレットを用いる符号化方式)

- JPEGの**約2倍**の効率もつ符号化方式として標準化された。
  特に低ビットレート時の符号化性能が良い。
- 符号化手順：
  - **色変換**
  - **離散ウェーブレット変換**
  - **量子化** (デッドゾーン＋線形量子化)
  - **EBCOT** (Embedded Block Coding with Optimal Truncation)
  - **MQ-coder** (算術符号化器)
- レイト歪み最適化 (Rate-distortion optimizer)
- 性能は良いが普及しなかった。
  - ストレージ，通信が進歩してJPEGで満足
  - (平均2乗誤差は小さいか) ボケている感じがする。
  - (サブマリン) 特許の問題

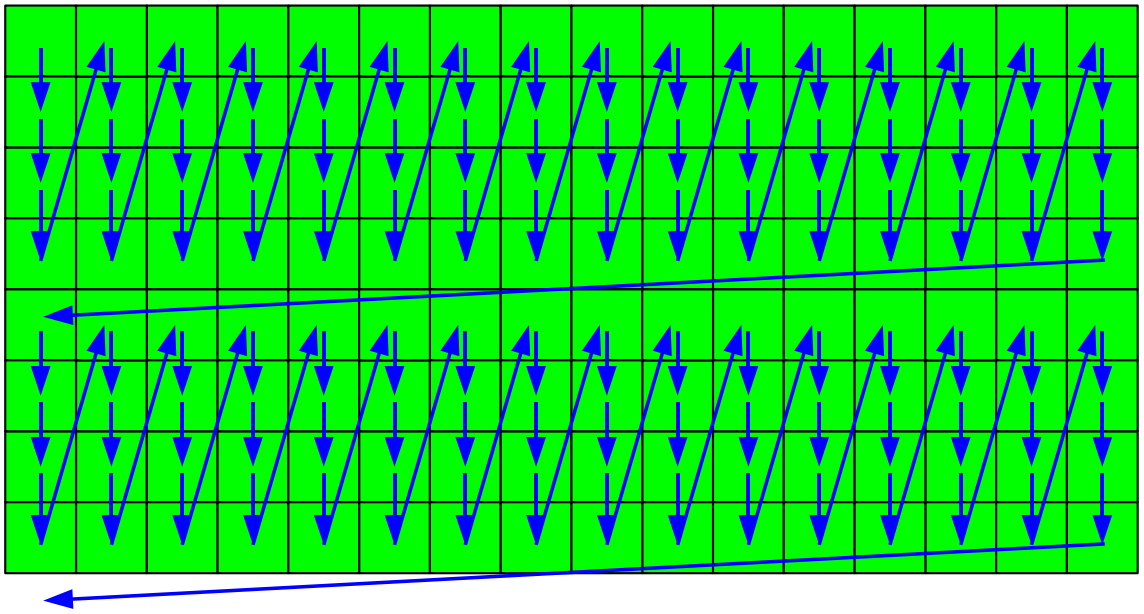
ビットプレーン処理

量子化値	5	3	12	8	7	0	31	21
4 bit	0	0	0	0	0	0	1	1
3 bit	0	0	1	1	0	0	1	0
2 bit	1	0	1	0	1	0	1	1
1 bit	0	1	0	0	1	0	1	0
0 bit	1	1	0	0	1	0	1	1

- ビットプレーン：2進数表現における1つの桁の0/1の2次元配列
- 上位bitから処理(符号化)してゆく。
- **周波数が低いバンドで0の場合は，画像上で対応する位置の高いバンドでも0の可能性が高い。**
- **ある値の近傍で，上位ビットが1になっているものがあり，自分の上位ビットはすべて0である場合，そのビットが1になる可能性が高い。**
- **これらの情報を，エントロピー符号化の確率に反映する。**

EBCOT (Embedded Block Coding with Optimal Truncation)

- **Embedded coding**：符号列を途中で打ちきっても，精度は落ちるが復号可能な符号(ビットプレーン)
- ビットプレーンをコードブロック(例えば，64×64)に分ける。
- 量子化係数を全て符号化するのではなく，コードブロックごとに打ち切り行う。
- コードブロック内をスキャンする順番(処理の容易さを考えて決定)。



EBCOTの符号化パス

- 3つの符号化パス
  - Significance propagation：有意である係数の周囲にある有意でない係数の符号化
  - Magnitude refinement：すでに有意である係数の符号化
  - Cleanup：それ以外の符号化
- EBCOTはビットと共に，その**コンテキスト**をMQ-coderに送る。
  コンテキストは既に送られた情報から分かるもの。
  - Significance propagationに含まれるビットで，近傍にある有意の係数の数。
  - 極性(±)のビットで，周囲の有意である係数の極性の状況
    有意になったときに極性も送る。
  - Cleanupで1列(4係数)がすべて0かどうか。

MQ-coder

- Q-coder系
  - 乗算が不要
  - ハードウエア(論理回路)での実装が可能
  - 確率0.75が8000 (16進数)に相当する。
- Q_eの値を46通りに決めている。(Q_e，$A - Q_e$)
- 符号化するビットに応じて，Q_eの値の状態を遷移させる。
- Q_eの値の状態は，コンテキストごとに保持している。
- Q-coderであるため，確率の和が0.75から1.5という誤差が生じる。
  - **実際上の問題**は生じていない。
  - この確率の精度を上げたい場合は，Aの値に応じて数通りのQ_eを用意すれば良い。

ウェーブレット変換を使った動画像符号化

- **Motion JPEG 2000**
  - 基本的には，JPEG 2000で順次符号化していく。
  - ビットストリームが決まっている。
- **Dirac** (BBCが開発，NHK技研が参加)
- ウェーブレット変換：CDF97
  - 重複動き補償予測

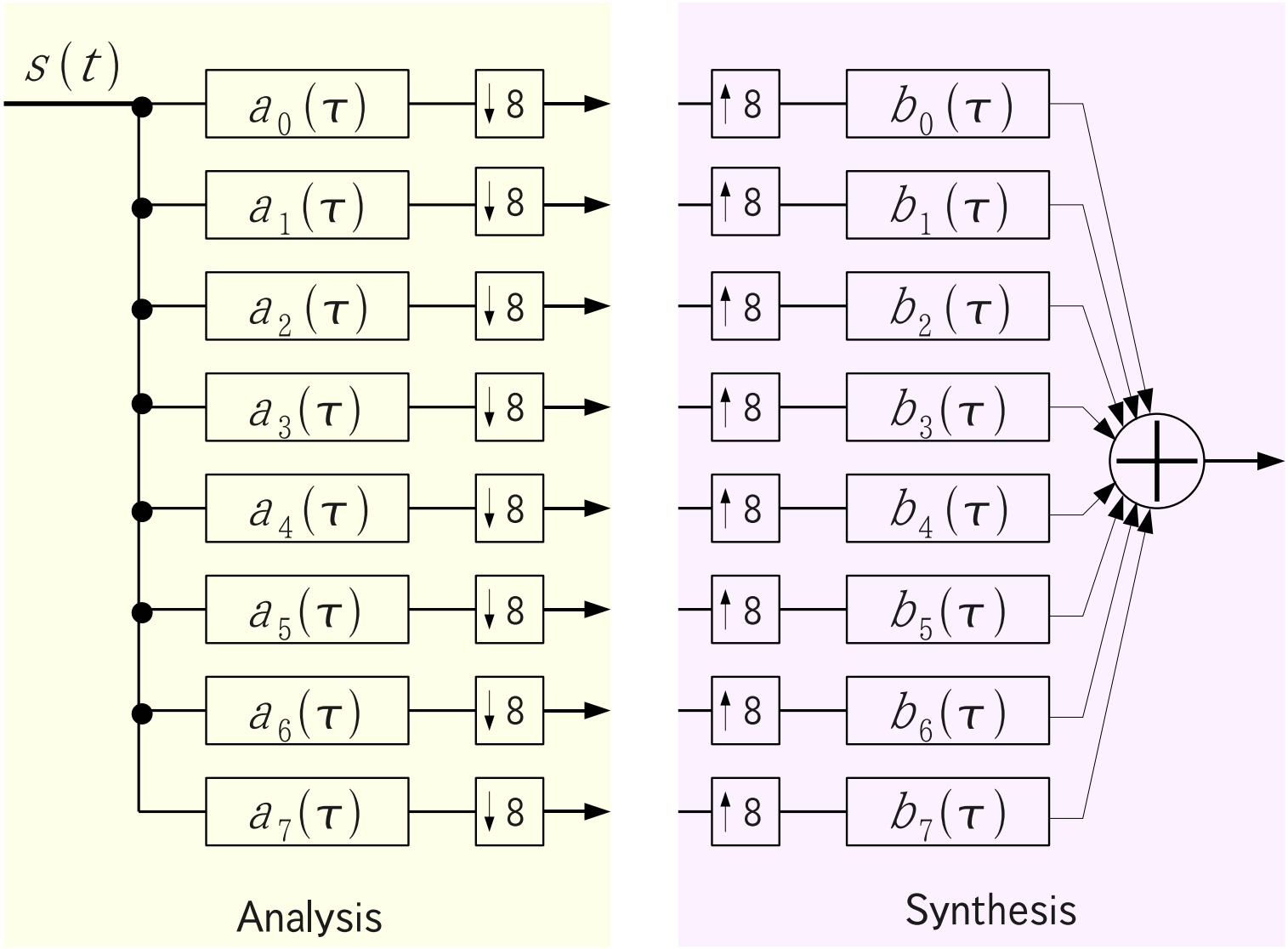
    対象画像を動き補償予測するためのブロックが隣接するブロックとオーバーラップしている。

    オーバーラップ部は，それぞれの結果の平均を値とする。
  - 整数係の数算術符号化 (計算量は多い)
  - レートディストーションオプティマイザー
  - 同じビットレートでのPSNRではH.264には及ばない。
  - ffmpeg (オープンソフト) に実装され公開されている。

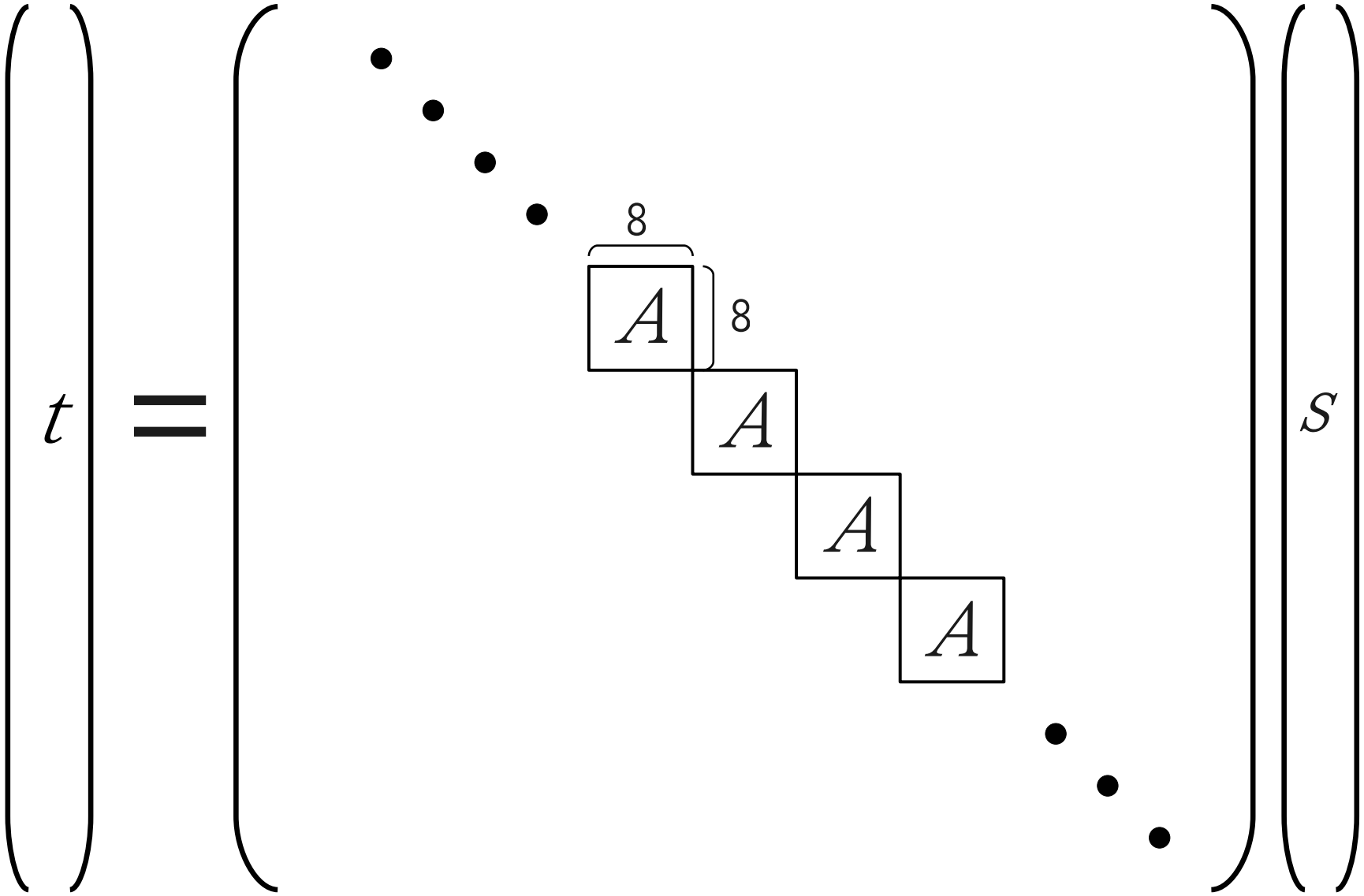
サブバンド変換

- ウェーブレットより1回で分解する周波数帯域が多い。
- **解析** (Analysis, Decomposition)
  1. 信号に対して複数の解析フィルタで変換し，変換係数を得る。
  2. (フィルタ or 基底関数) **タップ数**：フィルターの長さ
  3. それぞれの変換係数をダウンサンプリング(間引き)する。
  4. **デシメーションファクタ**：信号を取り出す割合
     (通常フィルタの数と一致)
- **合成** (Synthesis)
  1. それぞれのアップサンプリングする。
     係数列に (デシメーションファクタ-1) 個の0を挿入する。
  2. それぞれの合成フィルタで変換して，結果を足し合わせる。
- フィルターが重複(オーバーラップ)している。
  ◯ ブロック歪みが減少する。× リンギングが拡散する。

サブバンド変換 (フィルタバンク)

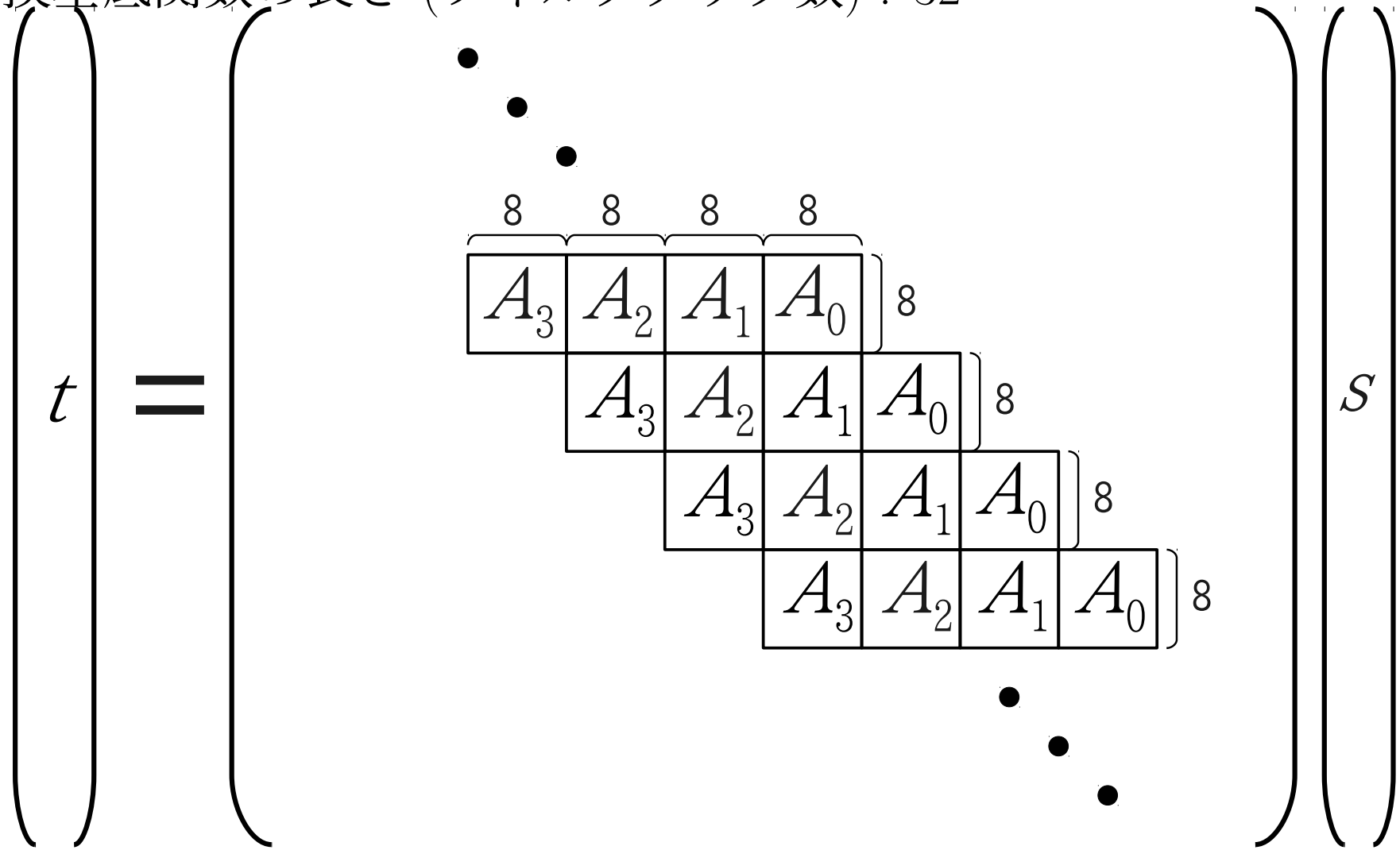


DCTの行列表示



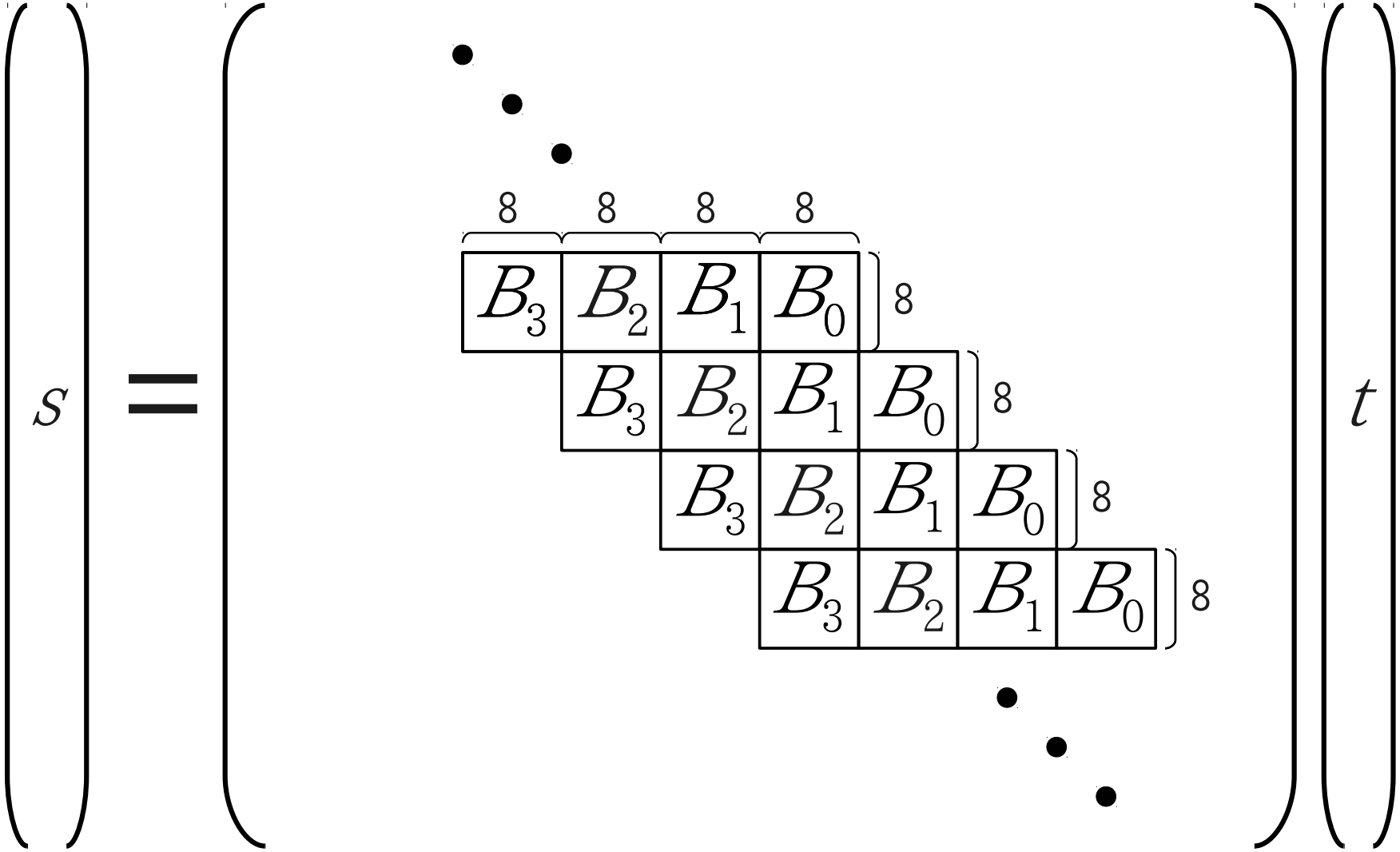
サブバンド変換の行列表示 (解析)

- 変換基底関数の長さ (フィルタタップ数) : 32



サブバンド変換の行列表示 (合成)

- 変換基底関数の長さ (フィルタタップ数) : 32



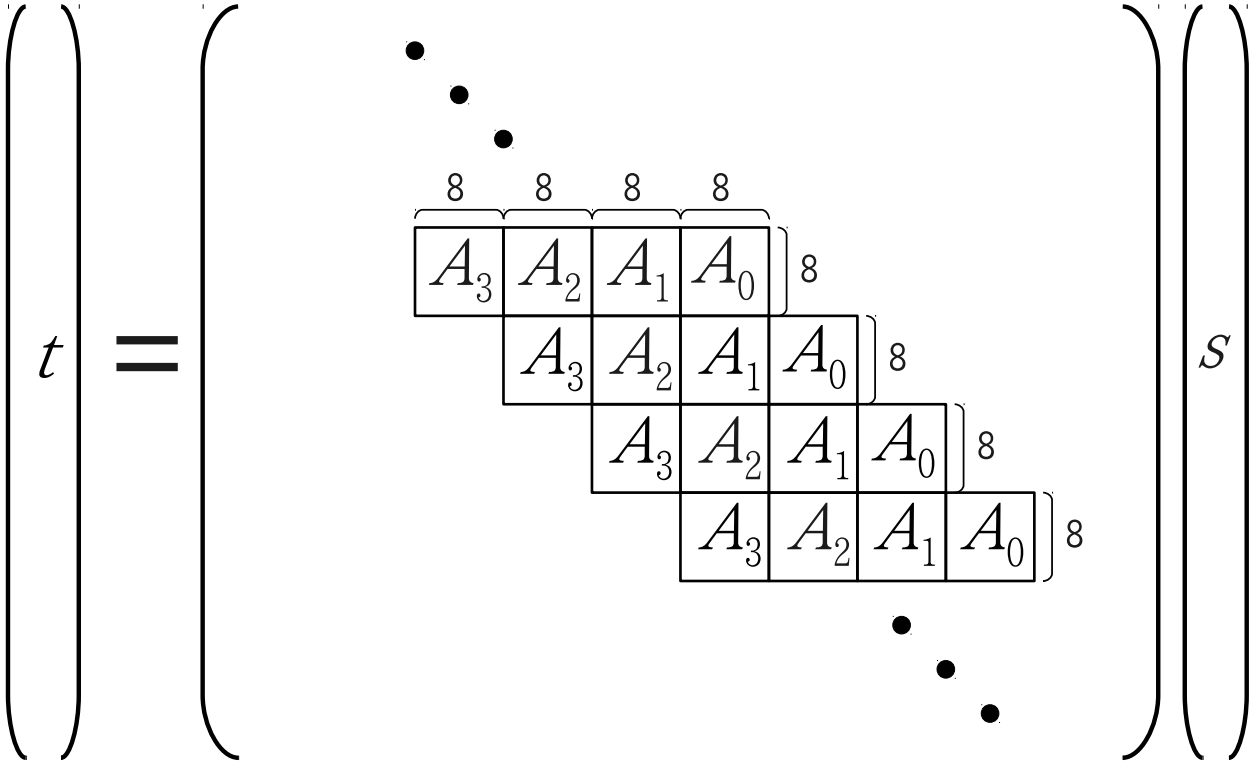
完全再構成 (Perfect reconstruction, PR)

2つの行列の積が単位行列になる。

$$
\begin{aligned}
B_0A_0 &= O \\
B_1A_0 + B_0A_1 &= O \\
B_2A_0 + B_1A_1 + B_0A_2 &= O \\
B_3A_0 + B_2A_1 + B_1A_2 + B_0A_3 &= I
\end{aligned}
\qquad
\begin{aligned}
B_3A_1 + B_2A_2 + B_1A_3 &= O \\
B_3A_2 + B_2A_3 &= O \\
B_3A_3 &= O
\end{aligned}
$$

$$
\begin{pmatrix}
\ddots & & & & & \\
 & B_3 & B_2 & B_1 & B_0 & \\
 & & B_3 & B_2 & B_1 & B_0 \\
 & & & B_3 & B_2 & B_1 & B_0 \\
 & & & & B_3 & B_2 & B_1 & B_0 \\
 & & & & & \ddots
\end{pmatrix}
\begin{pmatrix}
\ddots & & & & & \\
 & A_3 & A_2 & A_1 & A_0 & \\
 & & A_3 & A_2 & A_1 & A_0 \\
 & & & A_3 & A_2 & A_1 & A_0 \\
 & & & & A_3 & A_2 & A_1 & A_0 \\
 & & & & & \ddots
\end{pmatrix}
=
\begin{pmatrix}
\ddots & & & & & \\
 & I & & & & \\
 & & I & & & \\
 & & & I & & \\
 & & & & I & \\
 & & & & & \ddots
\end{pmatrix}
$$

ポリフェーズ表現



を,

$$\boldsymbol{A}(z) = A_0 + A_1 z^{-1} + A_1 z^{-2} + A_1 z^{-3}$$

と表す。

完全再構成 (ポリフェーズ表現)

解析：

$$\boldsymbol{A}(z) = \sum_{k=0}^{N} A_k z^{-k}$$

合成：

$$\boldsymbol{B}(z) = \sum_{k=0}^{N} B_k z^{-k}$$

Reversal matrix J

$$\begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdot & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

D : 次の条件を満たす対称行列

$$D_{ii} = \begin{cases} +1 & \text{フィルタ関数が対称} \\ -1 & \text{フィルタ関数が反対称} \end{cases}$$

完全再構成条件

$$\boldsymbol{B}(z)\boldsymbol{A}(z) = z^{-N} I$$

線形位相条件

$$\boldsymbol{A}(z) = z^{-N} D \boldsymbol{A}(z^{-1}) J$$

A^T : A の転置行列

パラコンジュゲートの定義

$$\tilde{\boldsymbol{A}}(z) = \sum_{k=0}^{N} A_{N-k}^{T} z^{-k}$$

線形位相完全再構成フィルタバンク (LPPRFB)

- $\boldsymbol{A}(z)$が**パラユニタリー**であるとは，次式を満たすこと。

$$\tilde{\boldsymbol{A}}(z)\boldsymbol{A}(z) = z^{-N}I$$

  - $\boldsymbol{A}(z)$と$\tilde{\boldsymbol{A}}(z)$でサブバンド変換とその逆変換になっている。
  - 重複直交変換

- $\tilde{\boldsymbol{A}}(z)$を$N$次の解析側のLPPRFBとする。このとき，

$$\boldsymbol{A}(z) = \boldsymbol{G}(z)\boldsymbol{F}(z)$$

  と**分解できる**。ここで，

  - $\boldsymbol{F}(z)$は$N-1$次のLPPRFB
  - $\boldsymbol{G}(z)$は可逆な1次のFB(フィルタバンク)で，次式を満たす。

$$\boldsymbol{G}(z) = D\boldsymbol{G}(z^{-1})D \tag{6}$$

LPPRFBの構成

- $\boldsymbol{A}(z)$は，次のように分解できる。

$$\boldsymbol{A}(z) = \boldsymbol{G}_N(z)\boldsymbol{G}_{N-1}(z)\cdots\boldsymbol{G}_1(z)A_0$$

  - $\boldsymbol{G}_1(z), \boldsymbol{G}_2(z), \ldots, \boldsymbol{G}_N(z)$は，式(6)を満たす可逆な1次のFB
  - A_0は可逆な行列で，次式を満たす。

$$A_0 = DA_0J$$

- 重複双直交変換
- 式(6)を満たす$\boldsymbol{G}_i(z)$は次のように書くことができる。

$$\boldsymbol{G}_i(z) = \begin{pmatrix} U_i & O \\ O & V_i \end{pmatrix} \begin{pmatrix} I & I \\ I & -I \end{pmatrix} \begin{pmatrix} I & O \\ O & z^{-1}I \end{pmatrix} \begin{pmatrix} I & I \\ I & -I \end{pmatrix}$$

U_i，V_iは可逆行列

LPPRFBの構成

- A_0は次のように書くことができる。

$$\boldsymbol{A}_0 = \begin{pmatrix} U_0 & O \\ O & V_0 \end{pmatrix} \begin{pmatrix} I & J \\ J & -I \end{pmatrix}$$

  U_0，V_0は可逆行列

- 合成フィルタは次のように書くことができる。

$$\boldsymbol{A}(z) = A_0^{-1} z^{-1} \boldsymbol{G}_1(z)^{-1} z^{-1} \boldsymbol{G}_2(z)^{-1} \cdots z^{-1} \boldsymbol{G}_N(z)^{-1}$$

  ここで,

$$\boldsymbol{A}_0^1 = \frac{1}{2} \begin{pmatrix} I & J \\ J & -I \end{pmatrix} \begin{pmatrix} U_0^{-1} & O \\ O & V_0^{-1} \end{pmatrix}$$

$$z^{-1}\boldsymbol{G}_i(z)^{-1} = \frac{1}{4} \begin{pmatrix} I & I \\ I & -I \end{pmatrix} \begin{pmatrix} z^{-1}I & O \\ O & I \end{pmatrix} \begin{pmatrix} I & I \\ I & -I \end{pmatrix} \begin{pmatrix} U_i^{-1} & O \\ O & V_i^{-1} \end{pmatrix}$$

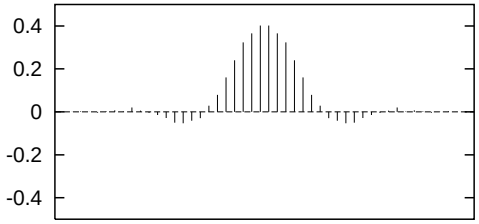
LPPRFBの設計

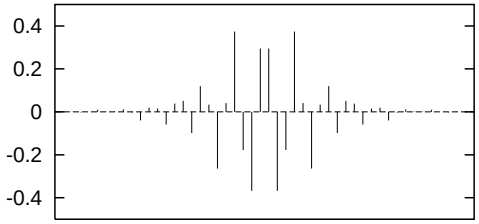
- A_0はDCTを使って構成できる。
- 可逆行列$U_1, V_1, U_2, V_2, \ldots, U_N, V_N$を決めるだけ。
- これらが可逆行列ならば，LPPRFBになる。
- 行列をSVDを使った表現でパラメタライズして，(非線形)最適化によってパラメータを求める。
- **最適化条件**
  - Stop band：
    周波数帯域をフィルタ数で分け，自分の帯域以外は通さない。
  - エントロピー最小化：
    信号の相関を仮定して$\rho^{|i-j|}$，変換係数の分散の対数和を最小化する(変換符号化)。
  - No DC leakage：
    最低周波数のバンドだけで，DC成分は完全に再構成できる。

画像符号化に関するサブバンド変換の文献

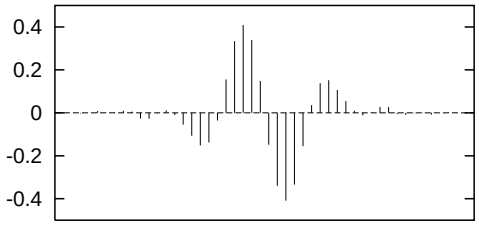
- **Wavelet** : Daubechies, 1988 [1], 1992 [2].
- Lapped orthogonal transform (**LOT**) : Malvar et al., 1989 [3].
- Generalized LOT (**GenLOT**) : Queiroz et al., 1996 [4].
- Generalized lapped transform (**GLT**) : Chan, 1995. [5]
  Lapped biorthogonal transform (**LBT**) : Malvar et al., 1998 [6].
- Generalized LBT (**GLBT**) : Tran et al., 2000 [7].
- Variable length LOT (**VLLOT**) : Tran et al., 1999 [8].
  Generalized unequal length LOT (**GulLOT**) : Nagai et al., 2000 [9].
- **VLGLBT** : Tran et al., 1998 [10].
- Adaptive LT (**ALT**) : Tanaka and Yamashita, 2002 [11].
- **GVLLT** : Tanaka, Hirasawa, and Yamashita, 2006 [12].

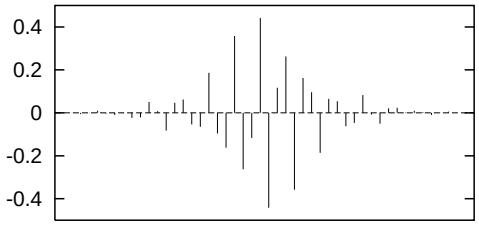
GenLOT (48 taps)

$n = 0$ 

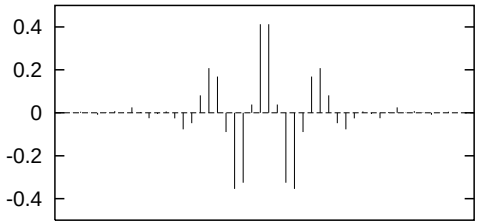


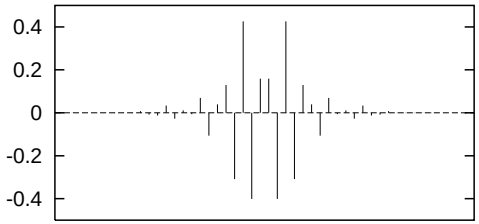
$n = 4$

$n = 1$ 

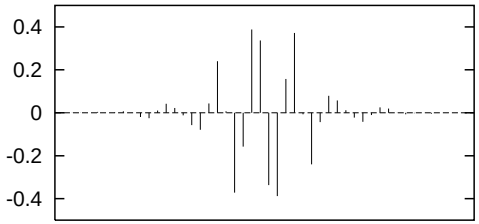


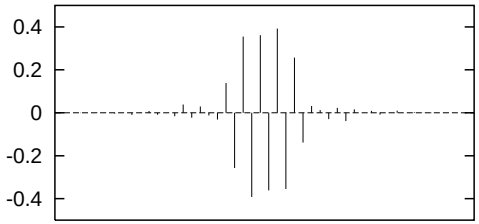
$n = 5$

$n = 2$ 



$n = 6$

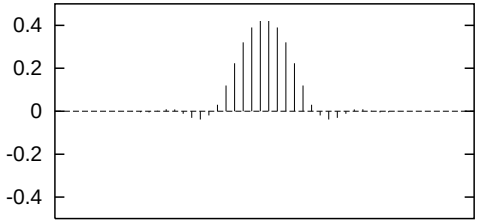
$n = 3$ 

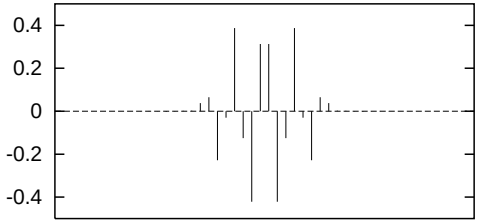


$n = 7$

GulLOT (48-16 taps)

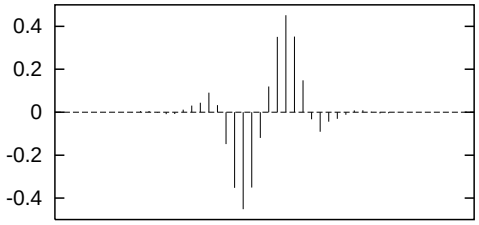
$n = 0$

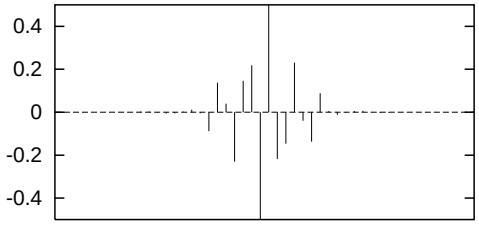




$n = 4$

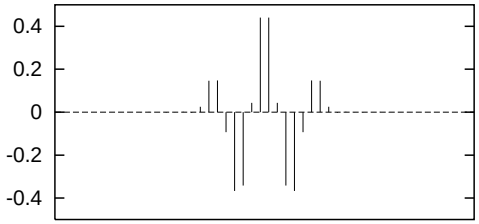
$n = 1$

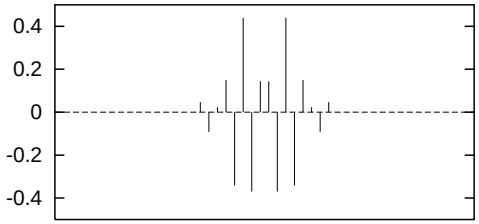




$n = 5$

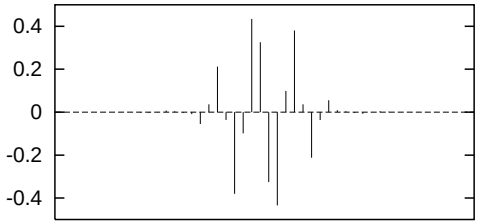
$n = 2$

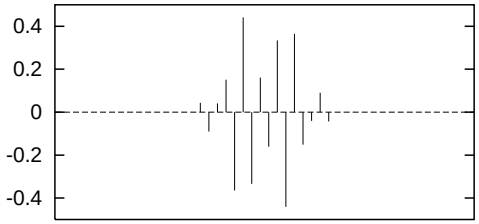




$n = 6$

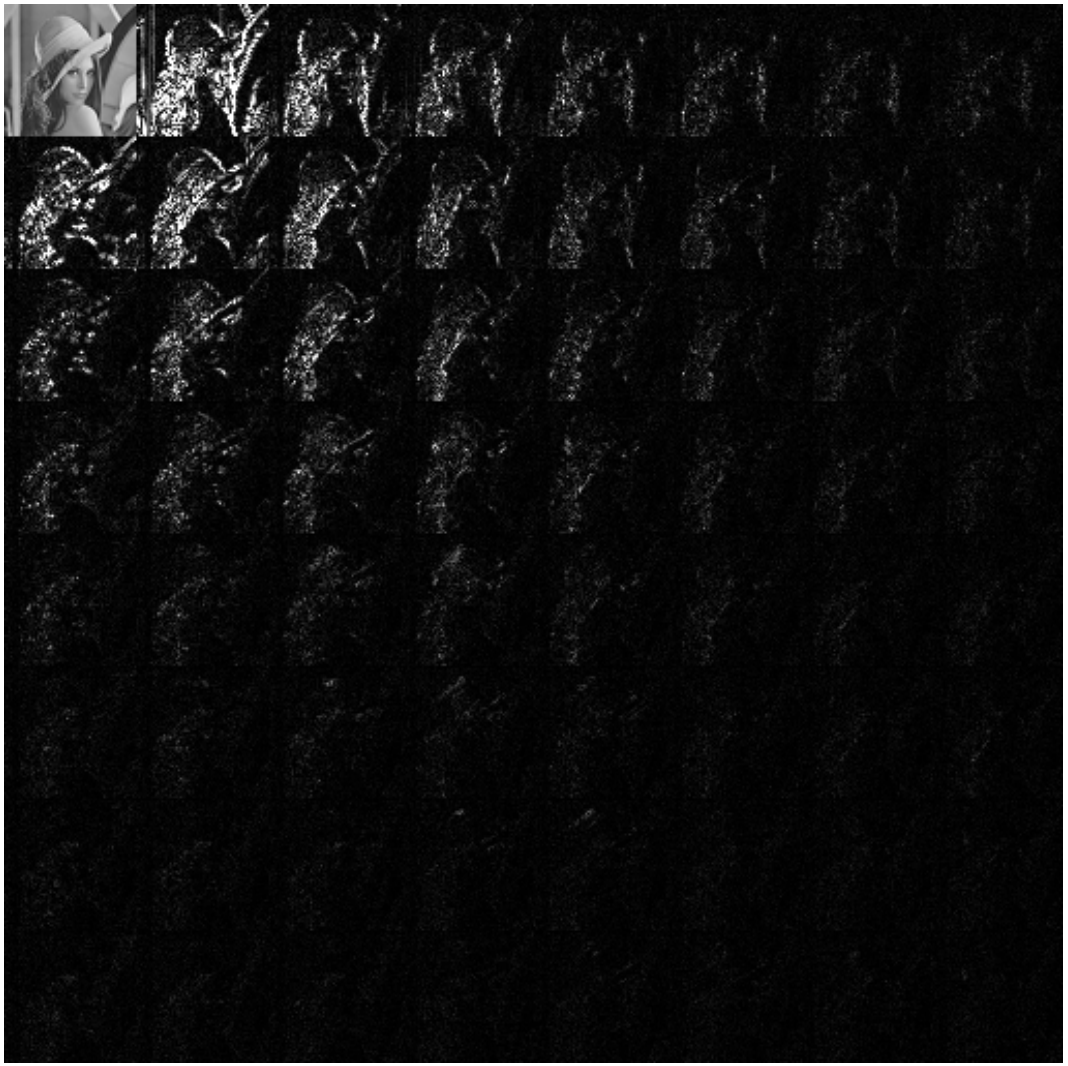
$n = 3$





$n = 7$

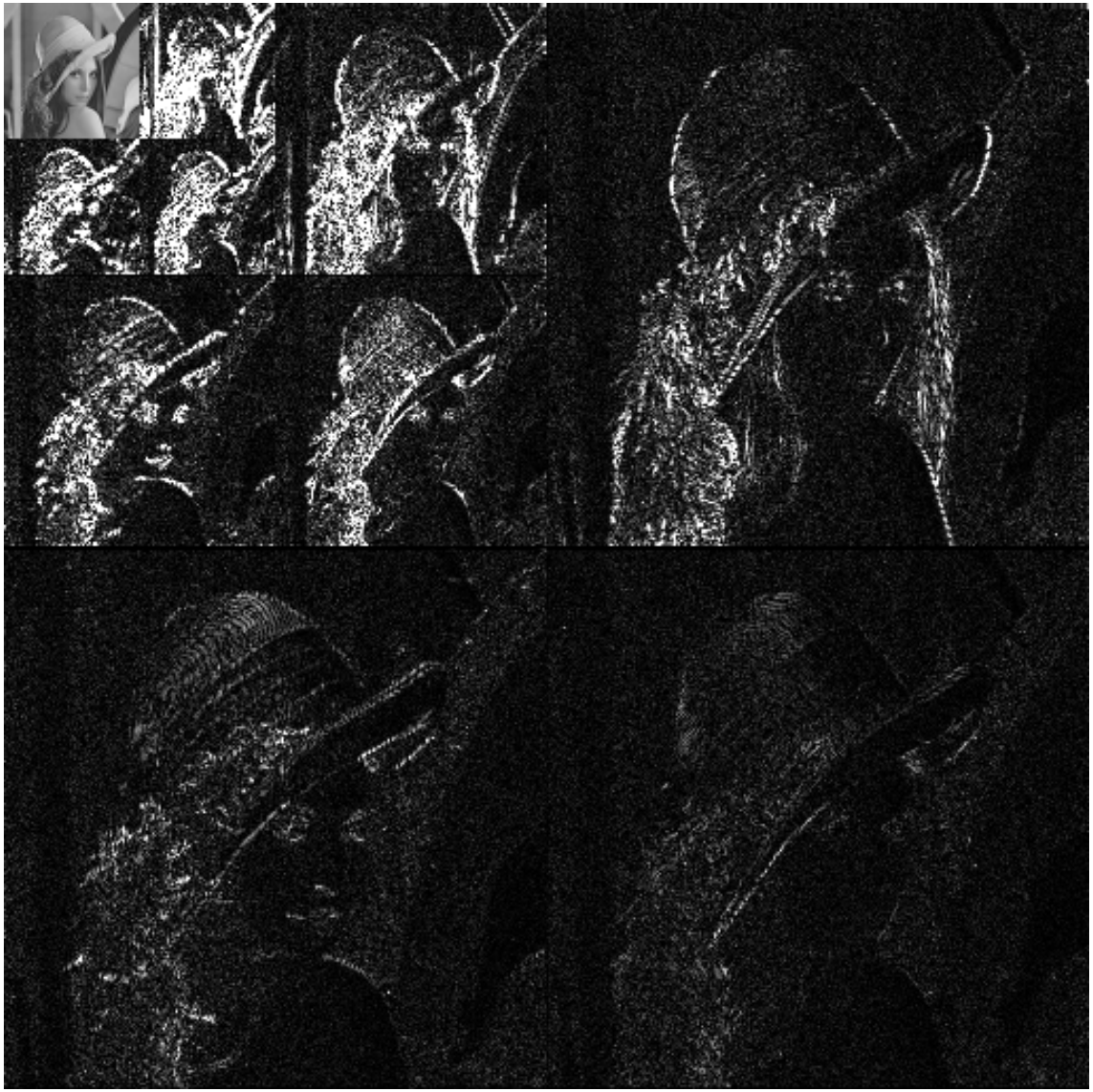
サブバンド変換の例



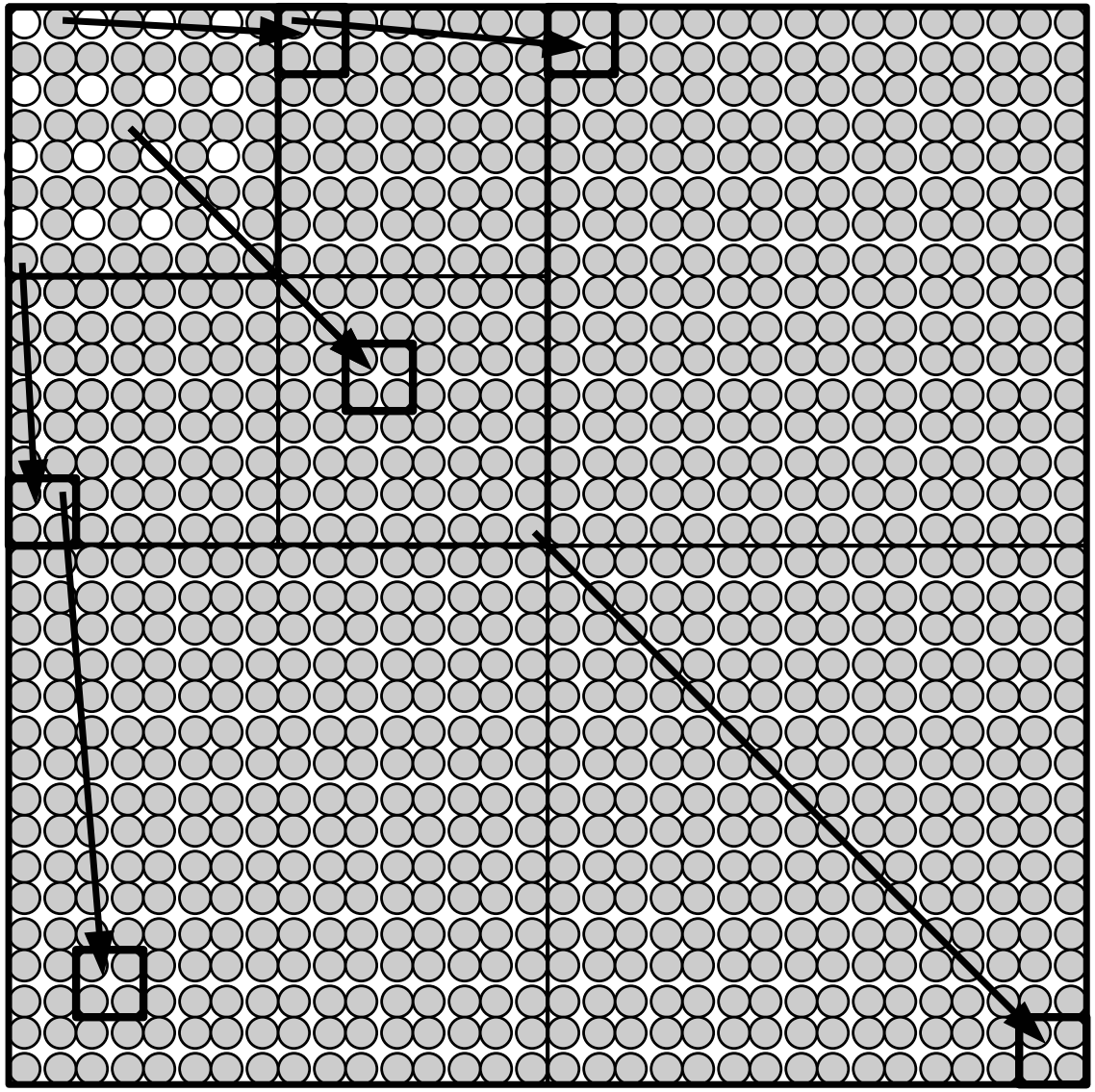
サブバンド変換係数の符号化

- **変換係数に空間的な冗長性が残されている**
  - 空間的に近い変換係数は類似した値
  - 低周波成分の変換係数が小さい場合は，対応する点の高周波成分の変換係数も小さい。
    ⇒ ビットプレーンの走査で**低い周波数で打ち切ることが可能**.
- 木構造を生成する。
  - Embedded zero tree (**EZW**) : Shapiro, 1993 [13].
  - Set Partitioning in Hierarchical Trees (**SPIHT**) : Said et al., 1996 [14].
- コンテキストモデリングによる予測
  - Embedded Block Coding with Optimal Truncation (**EBCOT**) : JPEG 2000

ウェーブレット変換



係数の木構造

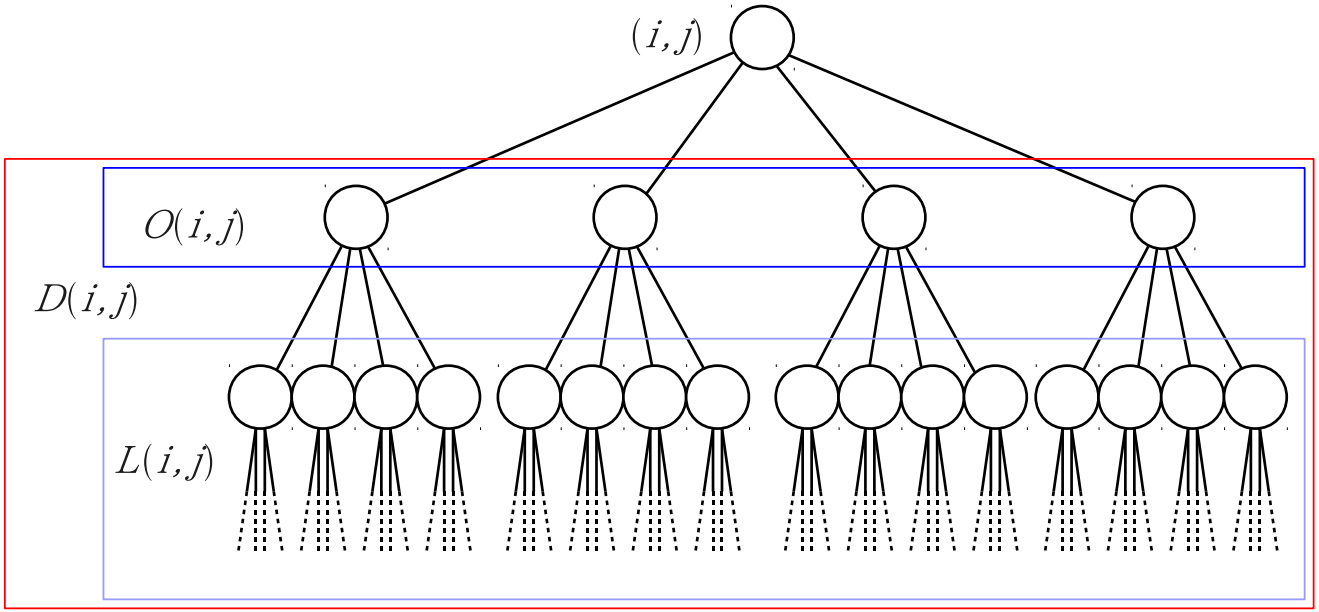


ノード(i,j)の処理で利用する集合

- $\mathcal{O}(i,j)$：ノード(i,j)の子供
- $\mathcal{D}(i,j)$：ノード(i,j)の子孫
- $\mathcal{L}(i,j) = \mathcal{D}(i,j) - \mathcal{O}(i,j)$.
- 木の根のノードの集合

処理全体で利用する集合

- LSP : 有意なノード(i,j)の集合
- LIP : 有意でないノード(i,j)の集合
- LSP : 有意でない集合を表す(i,j)の集合
  - type A : $\mathcal{D}(i,j)$を表す
  - type B : $\mathcal{L}(i,j)$を表す
- $c_{i,j}$: 符号化する係数



- $S_n(i,j)$：ノード(i,j)の2^nに対する有意性

$$S_n(i,j) = \begin{cases} 1 & (|c_{k,l}| \geq 2^n) \\ 0 & (\text{else}) \end{cases}$$

- $S_n(\mathcal{A})$：ノードの集合$\mathcal{A}$の2^nに対する有意性

$$S_n(\mathcal{A}) = \begin{cases} 1 & (\{\max_{(k,l)\in\mathcal{A}} |c_{k,l}|\} \geq 2^n) \\ 0 & (\text{else}) \end{cases}$$

SPIHTのアルゴリズム

1. **初期化**

   - $n = \left\lfloor \log_2 \left(\max_{(i,j)}\{|c_{i,j}|\} \right) \right\rfloor$
   - LSPを空集合にする。
   - LIPを$\mathcal{H}$のすべのノードとする。
   - LISを$\mathcal{H}$で子孫を持つすべのノードとする (type A)。

2. **Sorting Pass**

   (a) LIPのすべての(i,j)に対して次の処理を実行

      i. $S_n(i,j)$を出力

      ii. $S_n(i,j) = 1$ならば，(i,j)をLSPに加え，$c_{i,j}$の極性を出力

(b) LISのすべての(i,j)に対して次の処理を実行

i. その要素がtype Aならば,

- $S_n(\mathcal{D}(i,j))$を出力する。
- $S_n(\mathcal{D}(i,j))=1$ならば，すべての$(k,l)\in\mathcal{O}(i,j)$に対して次の処理を実行する。
  - $S_n(k,l)$を出力する。
  - $S_n(k,l)=1$ならば，(k,l)をLSPに加え，$c_{k,l}$の極性を出力
  - $S_n(k,l)=0$ならば，(k,l)をLIPに加える。
  - $\mathcal{L}(i,j)\neq\phi$ ならば，(i,j)をLISにtype Bノードとして加える。
  - (i,j)をLISから取り除く。

ii. その要素がtype Bならば,

- $S_n(\mathcal{L}(i,j))$を出力する。
- もし$S_n(\mathcal{L}(i,j))=1$ならば，次の処理を実行する。
  - すべての$(k,l)\in\mathcal{O}(i,j)$を，type AとしてLISに加える。
  - (i,j)をLISから取り除く。

3. **Refinement Pass**:
   このSorting passの前にあったLSPのノード(i, j)に対して，$|c_{i,j}|$の$n$bit目を出力する。
4. **Quantization-Step Update**: $n \leftarrow n - 1$とする。

SPIHT の特徴

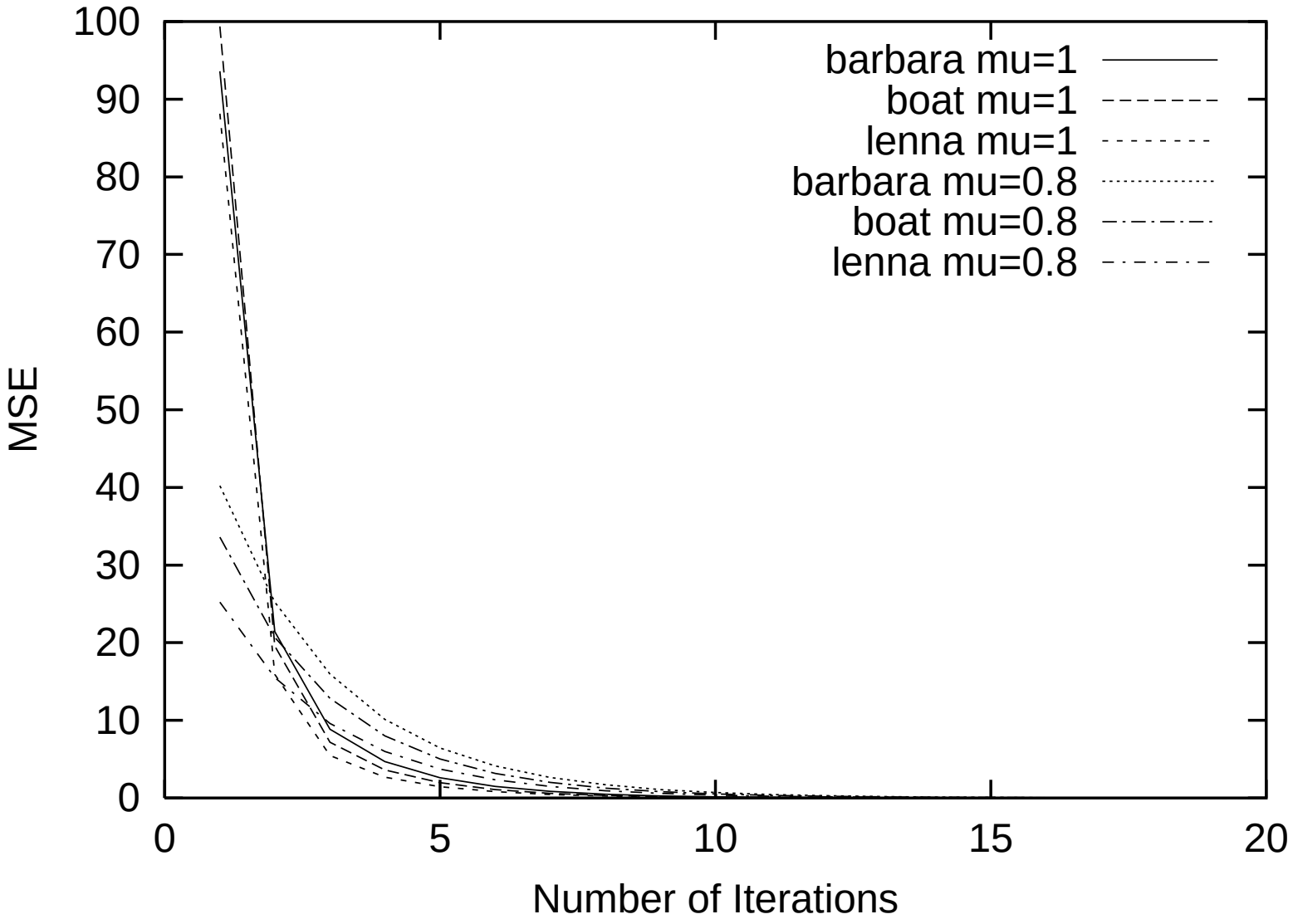
- アルゴリズムが簡単
- ハフマン符号化や算術符号化を使わなくても高い符号化性能
  (EBCOTには少し劣るがそれほどでもない。)

Generalized variable length lapped transform

- 低い周波数帯のフィルタには**長い**基底関数
  ⇐ ブロック歪みを抑制する。
- 高い周波数帯のフィルタには**短い**関数
  ⇐ リンギングの拡散を抑える。
- 直交(双直交)の条件は強く，フィルタの形が自由に選べない。
  ブロック境界でエッジを持つ関数になる。
  ⇒ **復号画像は合成フィルタの関数の線形和になる。**
  合成フィルタの関数は重要
- 解析フィルタは繰り返しで実現される。

実験結果 (原画像)

実験結果 (繰り返し回数)


100
90
80
70
60
50
40
30
20
10
0
MSE
0
5
10
15
20
Number of Iterations
barbara mu=1
boat mu=1
lenna mu=1
barbara mu=0.8
boat mu=0.8
lenna mu=0.8

PSNR と bit rate

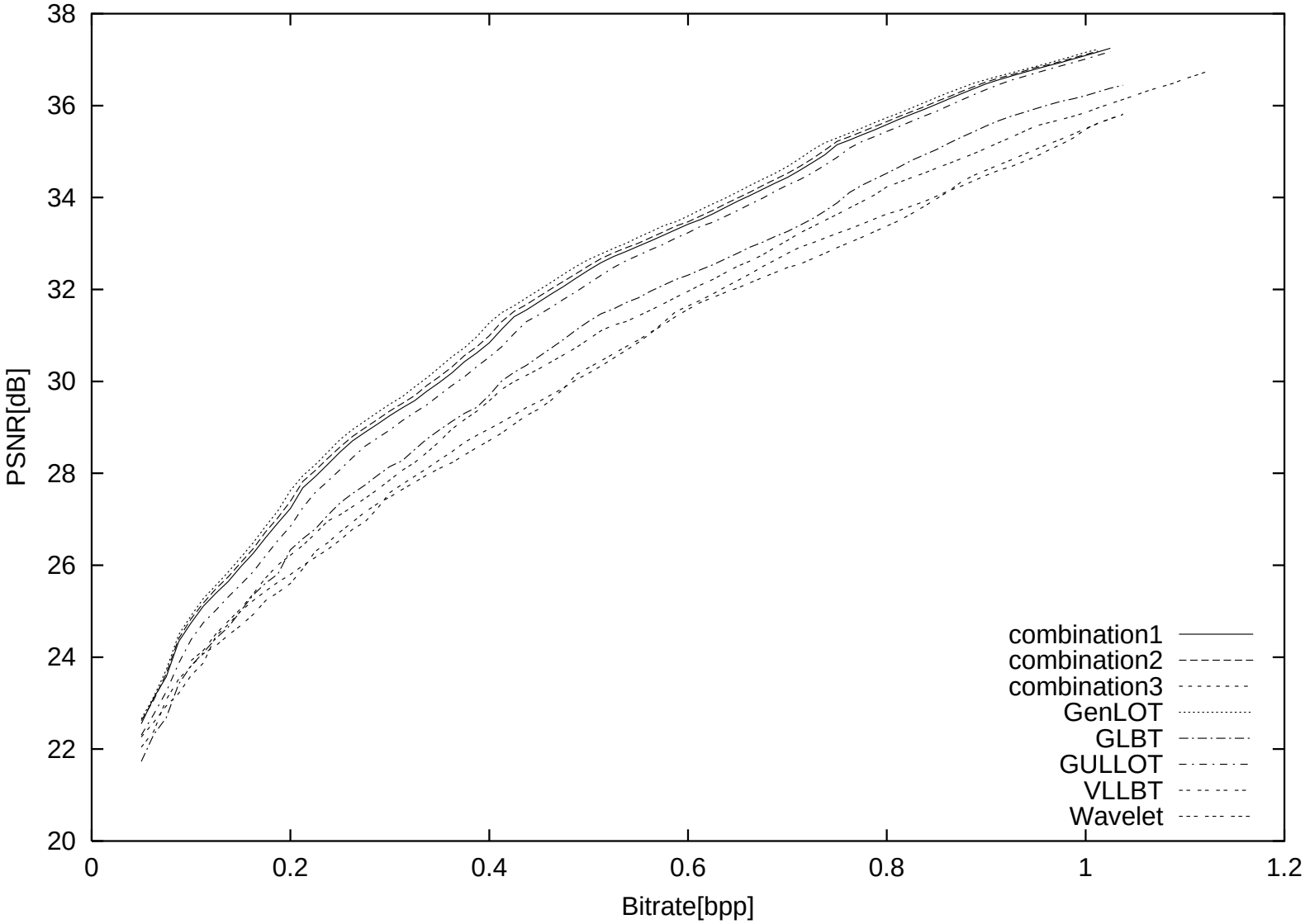
- PSNR (Peak Signal to Noise Ratio)

$$\mathrm{PSNR} = -10 \log_{10} \frac{\text{原画像と復号画像の画素値に関する平均2乗誤差}}{(\text{画素値の最大値})^2}$$

- ビットレート

$$\text{bit rate} = \frac{\text{画像を表すためのデータ量 (bit)}}{\text{画像のピクセル数}}$$

実験結果 (PSNR vs. bit rate)



実験結果 (復号画像)

GenLOT

GVLLT

実験結果 (拡大画像)

GenLOT

GVLLT

サブバンド動画像符号化

目的：**ブロック歪みの少ない**動画像符号化

ブロック歪みの原因と対策：

1. **ブロック変換**

   ⇒ サブバンド変換(重複変換)の適用

2. **ブロック単位の動き保証予測**

   ⇒ 適応デブロッキングフィルタの適用

◯ 参照画像を切り貼りして予測画像を作成すときにブロック歪みが生じる。

ブロック歪みが生じる前の参照画像を使うことができる。

⇒ ブロック境界のエッジが本当のエッジかブロック歪みかわかる。

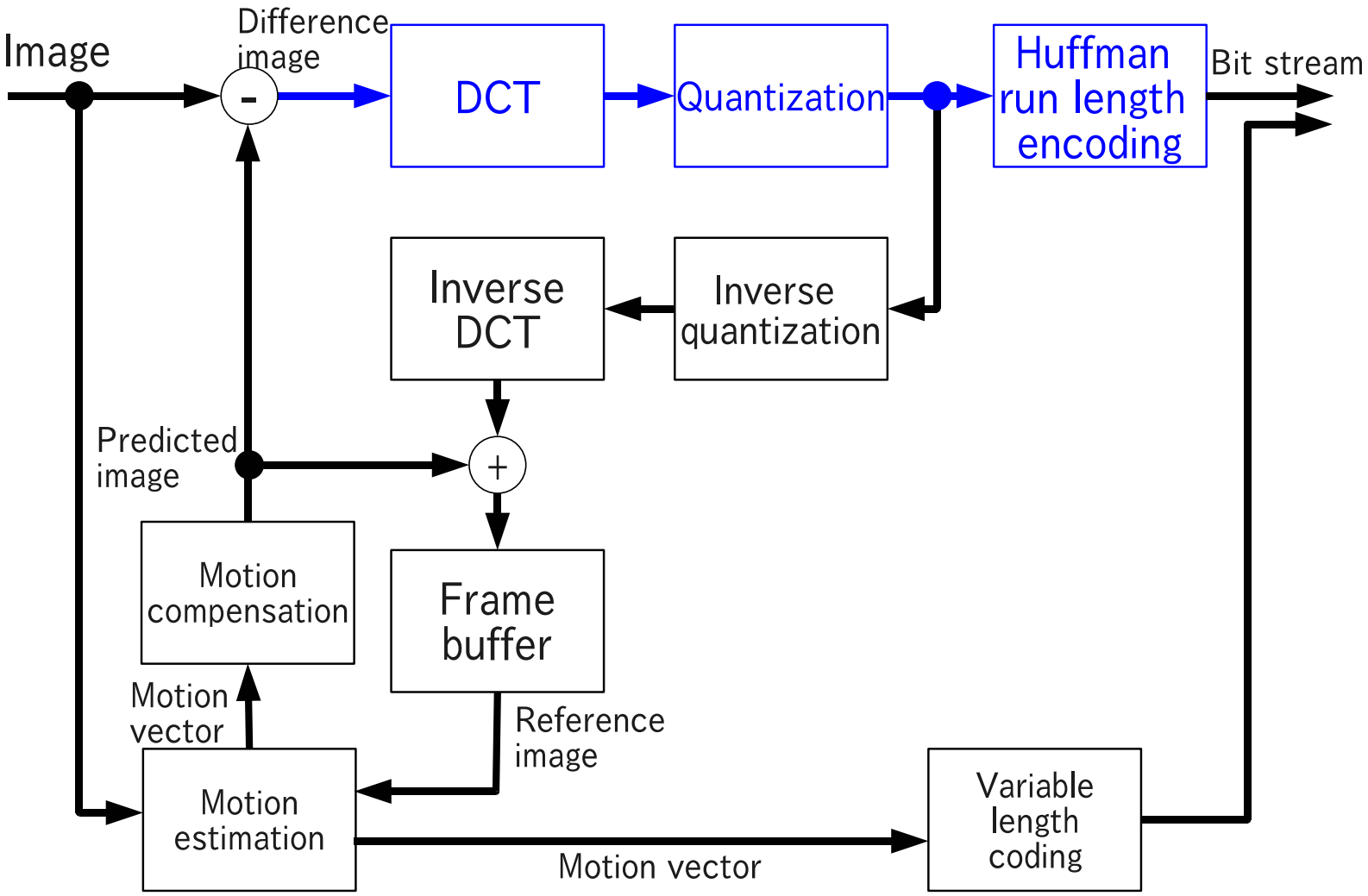
⇒ 適応デブロッキングフィルタが効率的に働く。

× **通常のブロック変換を使う方法では**

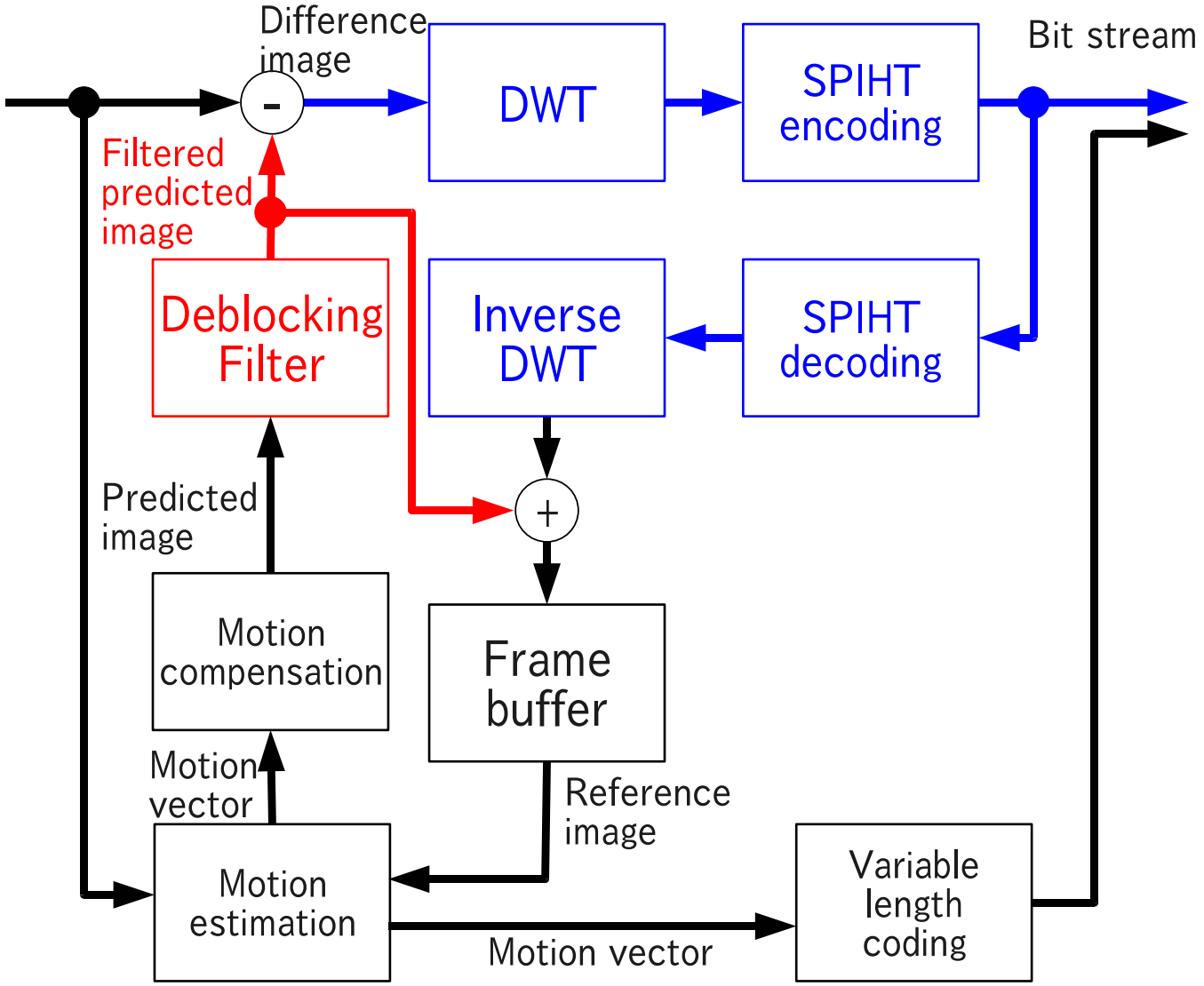
参照画像に既にブロック変換から生じるブロック歪みが生じている。

⇒ 画像のエッジを検出することが難しい。

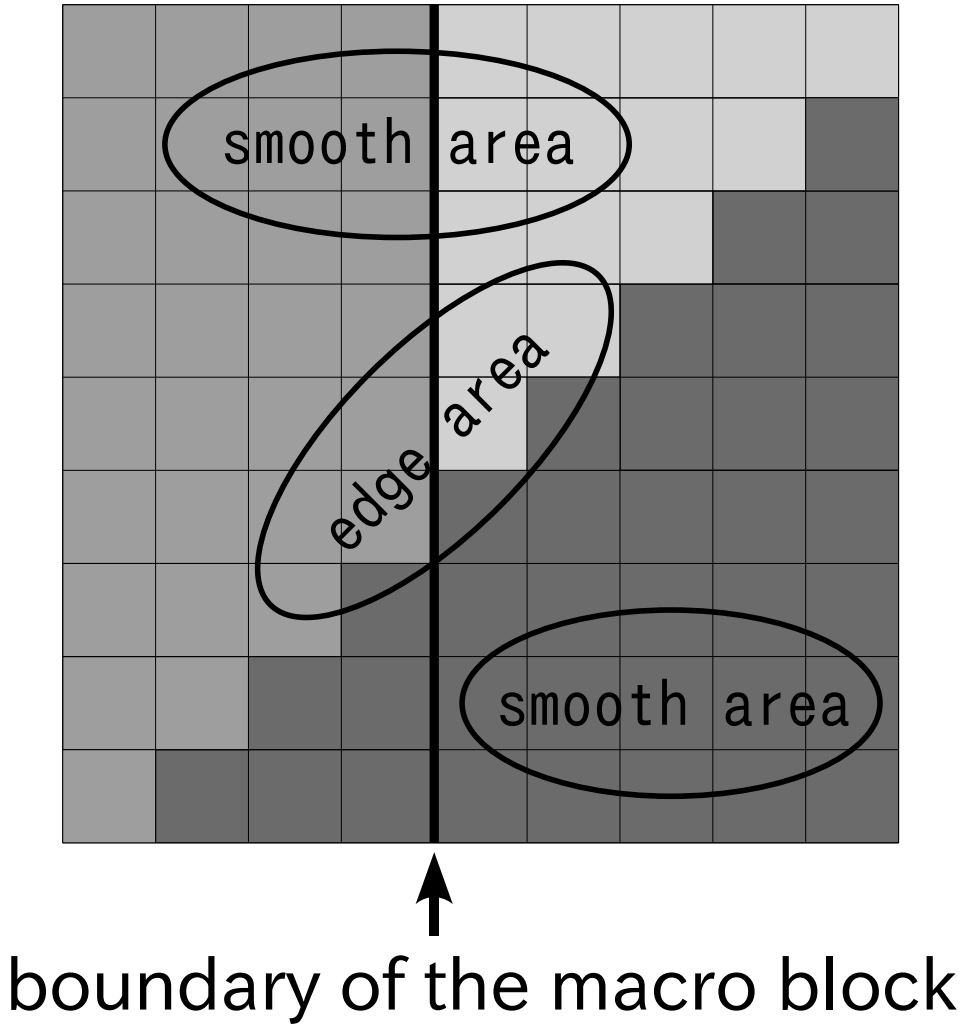
MPEGエンコーダのブロック図



提案手法のエンコーダのブロック図

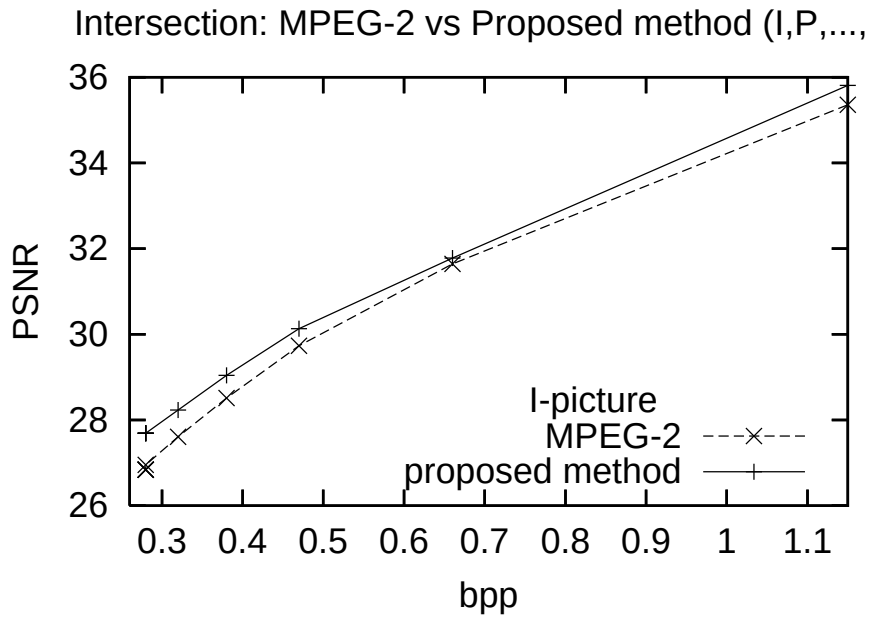


エッジ適応デブロッキングフィルタ

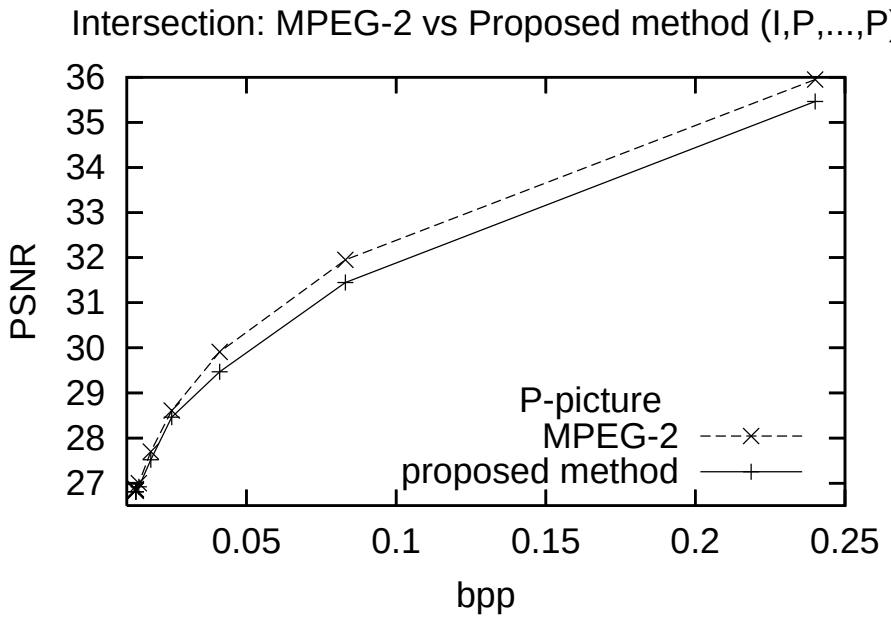


Original image (Intersection)

実験結果 (PSNR vs. bit rate)



I-Frame



P-Frame

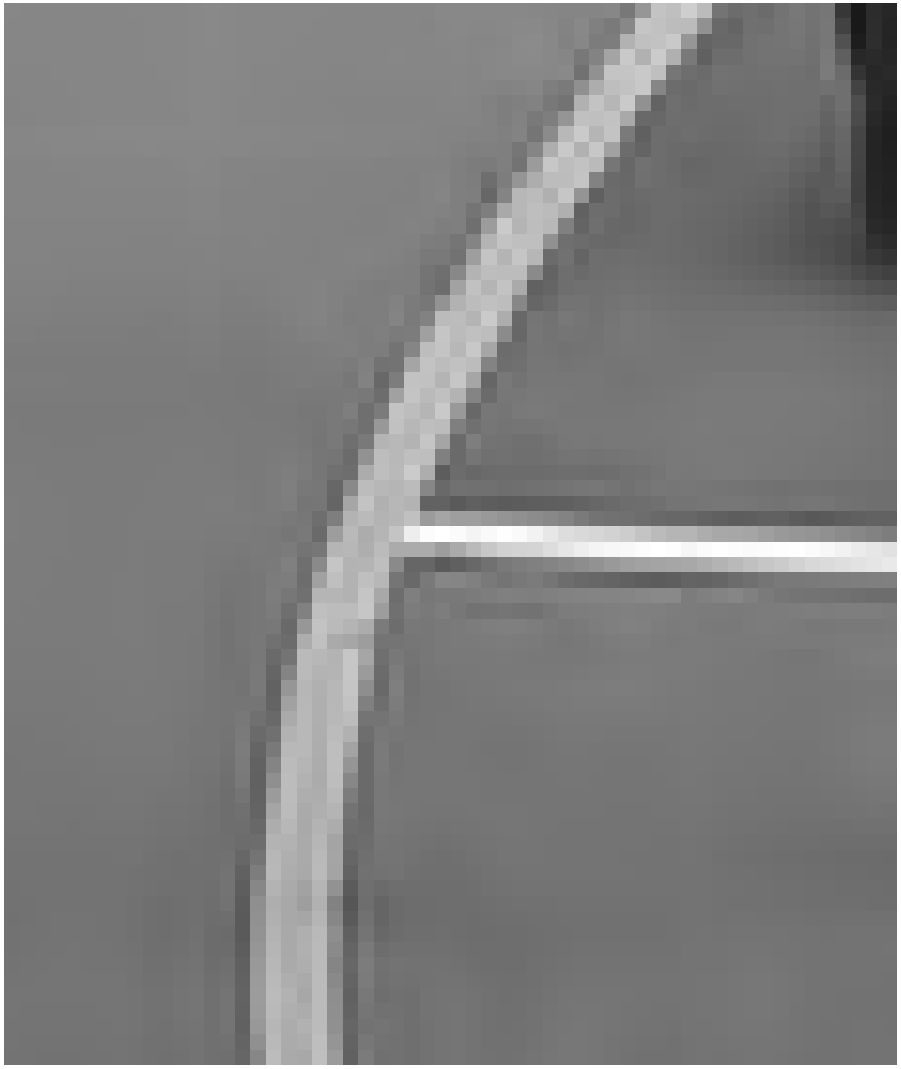
実験結果

MPEG2 (ffmpeg)

Proposed

実験結果 (拡大画像)

MPEG2 (ffmpeg)



Proposed

まとめ

- **画像符号化の基盤技術**
  - エントロピー符号化
  - 予測符号化
  - 変換符号化 (DCT，KL変換)
- **標準的画像符号化法**
  - JPEG
  - MPEG, MPEG 4 AVC (H.264)
- **離散コサイン変換を用いない画像符号化**
  - ウェーブレット変換
  - サブバンド変換 (重複変換)
- 画像符号化は情報技術の集大成
- マルチメディア社会のなかで高い重要性

参考文献

[1] I. Daubechies, “Orthonormal bases of compactly supported wavelets,” Communications on Pure and Applied Mathematics, vol.41, no.7 pp.909–996, Oct. 1988.

[2] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, “Image coding using wavelet transform,” IEEE Transactions on Image Processing, vol.1, no.2, pp.205–220, Apr 1992.

[3] H.S. Malvar and D.H. Staelin, “The LOT: transform coding without blocking effects,” IEEE Transactions on Acoustics, Speech and Signal Processing, vol.37, no.4, pp.553-0559, Apr 1989.

[4] R.L. de Queiroz, T.Q. Nguyen, and K.R. Rao, “The GenLOT: generalized linear-phase lapped orthogonal transform, ” IEEE Transactions on Signal Processing, vol.44, no.3 pp.497–507, Mar 1996.

[5] S.C. Chan, “The generalized lapped transform (GLT) for subband coding applications,” Proceedings of 1995 IEEE International Conference on Proceedings of the Acoustics, Speech, and Signal Processing, vol.2, pp.1508–1511, 9–12 May 1995.

[6] H.S. Malvar, “Biorthogonal and nonuniform lapped transforms for transform coding with reduced blocking and ringing artifacts,” IEEE Transactions on Signal Processing, vol.46, no.4, pp.1043-1053, Apr 1998.

[7] T.D. Tran, R.L. de Queiroz, T.Q. Nguyen, “Linear-phase perfect reconstruction filter bank: lattice structure, design, and application in image coding,” IEEE Transactions on Signal Processing vol.48, no.1, pp.133–147, Jan 2000.

[8] T.D. Tran, M. Ikehara, and T.Q. Nguyen, “Linear phase paraunitary filter bank with filters of different lengths and its application in image compression,” IEEE Transactions on Signal Processing, vol.47, no.10, pp.2730–2744, Oct. 1999.

[9] T. Nagai, M. Ikehara, M. Kaneko, and A. Kurematsu, “Generalized unequal length lapped orthogonal transform for subband image coding,” IEEE Transactions on Signal Processing, vol.48, no.12, pp.3365–3378, Dec. 2000.

[10] T.D. Tran, R.L. de Queiroz, and T.Q. Nguyen, “The variable-length generalized lapped biorthogonal transform,” Proceedings of 1998 IEEE International Conference on Image Processing, vol.3, pp.697–701, 4–7 Oct. 1998.

[11] T. Tanaka and Y. Yamashita, “An adaptive lapped biorthogonal transform and its application in orientation adaptive image coding,” Signal Processing, vol.82, no.11, pp.1633–1647, Nov. 2002.

[12] T. Tanaka, Y. Hirasawa, and Y. Yamashita, “Variable-length lapped transform with combination of multiple synthesis filter banks for image coding,” IEEE Transactions Image Processing, vol.15, no.1, pp.81–88, Jan. 2006.

[13] J.M. Shapiro, “Embedded image coding using zerotrees of wavelet coefficients,” IEEE Transactions on Signal Processing, vol.41, no.12 pp.3445–3462, Dec. 1993.

[14] A. Said and W.A. Pearlman, “A new fast and efficient image codec based on set partitioning in hierarchical trees,” IEEE Transactions on Circuits and Systems for Video Tech., vol.6, pp.243–250, June 1996.

[15] Nasharuddin Zainal, Toshihisa Tanaka and Yukihiko Yamashita, “Moving picture coding by lapped transform and edge adaptive deblocking filter with zero pruning SPIHT,” IEICE Trans. on Information and Systems, vol. E93-D, no. 6, pp.1608–1617, June, 2010.

[16] 小野定康, 鈴木純司, “わかりやすい JPEG2000 の技術,” オーム社, 2003.

[17] 大久保榮, 角野眞也, 菊池義浩, 鈴木輝彦, “H.264/AVC 教科書,” インプレス, 2004.

[18] Telecommunication standarization sector of ITU, “Advanced video coding for generic audiovisual services,” ITU, 2007.