# 画像符号化事始め
# ～ 前半：基礎の基礎 ～

平成 25 年 11 月 7 日（木） @ PCSJ2013

吉田 俊之（福井大学）



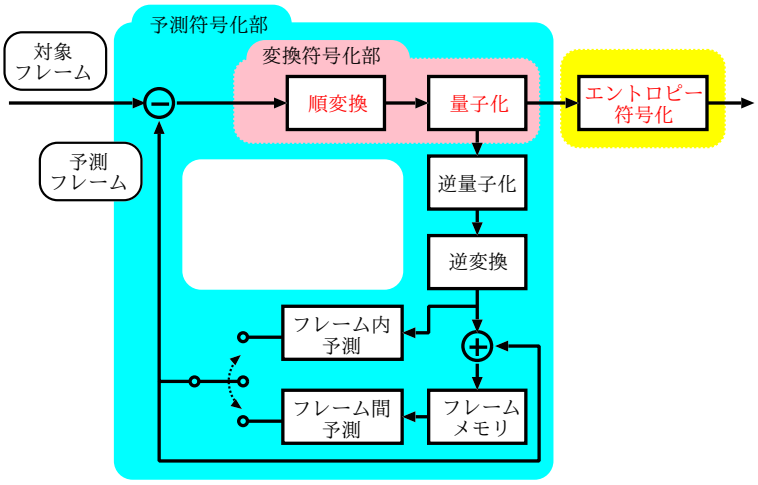
## 0. このチュートリアルについて

本チュートリアルは「事始め」として、これから研究を始めよう／進めようとしている皆さんに、この分野の「敷居」を少しでも低くし、多くの方を研究者としてお招きしたいという目的の下に企画された。符号化技術はしっかりした理論的基礎の上に複雑で繊細なアルゴリズム が乗っているという背景に鑑み、前半を「基礎の基礎」として吉田が、後半を「最先端動向」として清水が担当する。PCSJ での発表数が少しでも増えることを願ってお話を進めたい。

## ベースとする/させて頂く資料

- N.S. Jayant, P.Noll: “Digital Coding of Waveforms”, Prentice-Hall (1984)
- 酒井，吉田：“映像情報符号化”，オーム社 (2001 年).
- 吉田，鈴木，広明：“画像情報符号化”，コロナ社 (2008 年).
- 電子情報通信学会「知識ベース」2 群 5 編 $\beta$ 版サイト
  http://member.ieice-hbkb.org/portal/
- 講座 “基礎からの画像符号化”，映メ学会誌 (2013 年 1 月〜6 月).

## 前半「基礎の基礎」の目的 — 後半の理解のための準備






## 「前半：基礎の基礎」の内容

1. 画像と符号化 — 準備
2. 画像の性質と符号化 — 画素間相関と圧縮
3. エントロピー符号化
4. 量子化
5. 相関除去法 (1) — 予測による方法
6. 相関除去法 (2) — 変換による方法
7. 動画像の性質と符号化 — 時間方向の相関除去
8. 以上を組み合わせて

## お話の方針

- 個々のトピックは皆さんが専門家
- 全体が見渡せるように

「前半：基礎の基礎」の内容

1. 画像と符号化 — 準備
2. 画像の性質と符号化 — 画素間相関と圧縮
3. エントロピー符号化
4. 量子化
5. 相関除去法 (1) — 予測による方法
6. 相関除去法 (2) — 変換による方法
7. 動画像の性質と符号化 — 時間方向の相関除去
8. 以上を組み合わせて

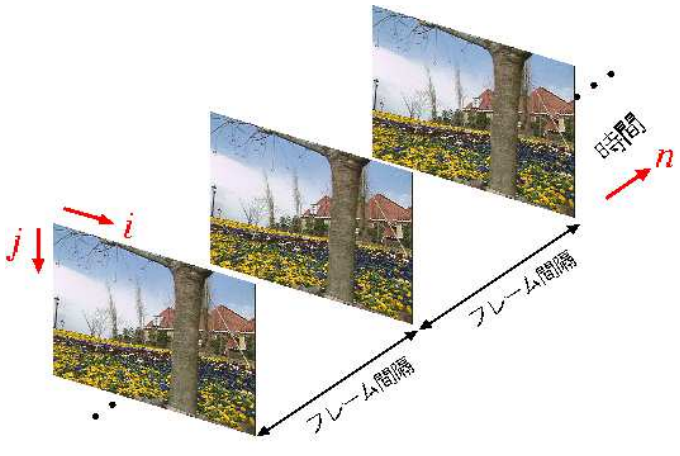


# 1. 画像と符号化 — 準備

- 静止画像 — （正しくサンプリングされた）画素の配列


モノクロ画像　　カラー画像＝「RGB 各 “モノクロ画像” の組み」

- 動画像 — 一定時間間隔でサンプルした静止画（フレーム）の列




- 当面，モノクロ動画像 $f_n(i,j)$ を対象

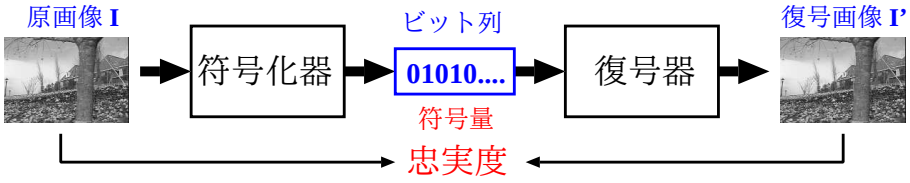
## 画像と符号化

- 対象画像を可能な限り少ないデータ量で忠実に表現すること

符号化 { 情報源符号化 (source coding) ··· 情報圧縮
通信路符号化 (channel coding) ··· 誤り訂正

- 厳密には，“画像 圧縮 符号化” が正しい
- 「原画像 I の情報量」 > 「符号化ビット列の符号量」 ⇒ 成功！

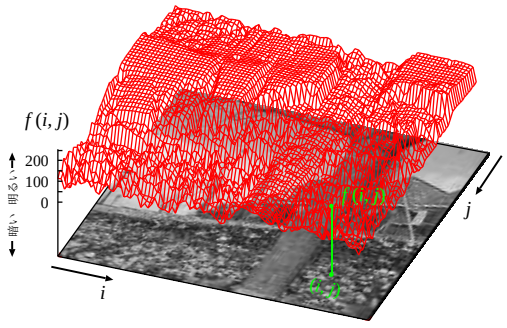

- I = I' ⇒ 可逆 (lossless) 符号化 ··· 「符号量」が小さければ良い
- I ≠ I' ⇒ 非可逆 (lossy) 符号化 ··· 「符号量」と共に「I〜I' 間の忠実度」を評価する必要



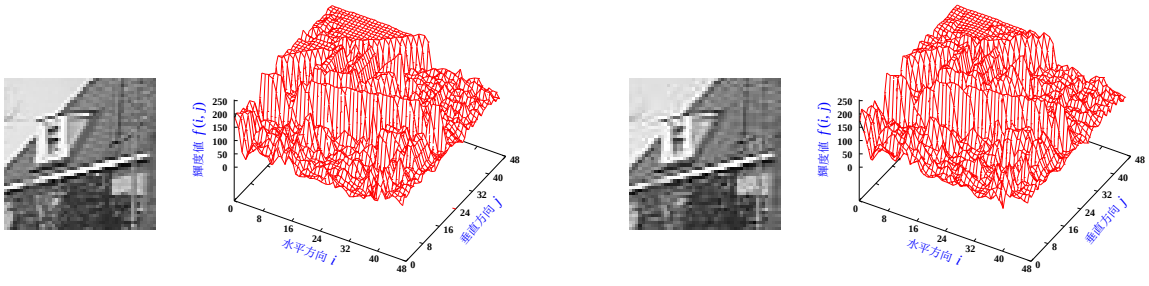



## 画像と波形，波形符号化

- 画像の各画素の輝度を波形として見ると ... ⇒ 2 次元波形

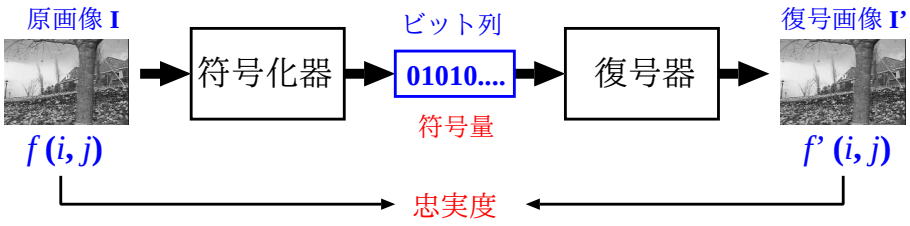



- 波形符号化 ··· 画像を「2 次元波形」と捉え，極力少ない符号量で表現




(a) 原画像　　(b) 復号画像

## 非可逆符号化の評価 (1) — 符号量と忠実度



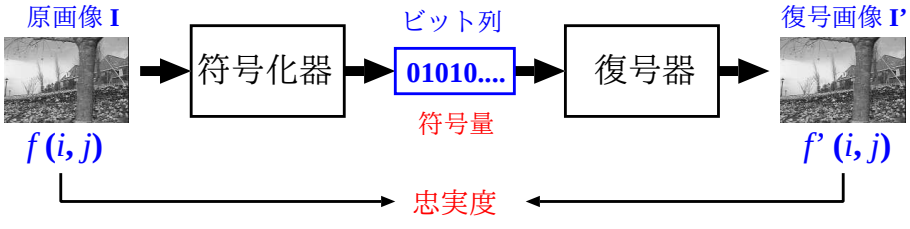

- 「符号量」の評価
  - $256 \times 256$ で 8kB，$3520 \times 2400$ で 1MB ⇒ 低符号量は？ ⇒ “規格化”
  - 静止画 ： 画素数で規格化 ⇒ **画素当たりの符号量 (BPP)** [bit/pixel]
  - 動画像 ： 時間で規格化 ⇒ **1 秒当りたの符号量 (BPS)** [bit/s]
- 「忠実度」の評価 ⇒ I と I' が いかに違わないか
  - 主観評価 ··· 人が実際に見て判断した評価値 ··· MOS 等
  - 客観評価 ··· I〜I' の違いを数式で数値化 ··· MSE, PSNR, SSIM 等

    ⇒ MSE (PSNR) が用いられている ⇐ 最小化が容易だから

以降,「違い」,「誤差」,「雑音の大きさ」はすべて MSE で計る



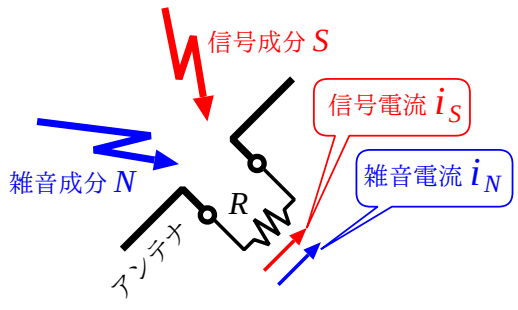
## 非可逆符号化の評価 (2) — MSE と PSNR




- **MSE：平均 2 乗誤差 (Mean Squared Error)** ⇐ 画像全体で平均

$$\mathrm{MSE} = \frac{1}{WH} \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} \{ \overbrace{f(i,j)}^{\text{原画像 I}} - \overbrace{f'(i,j)}^{\text{復号画像 I'}} \}^2$$

- MSE は誤差電力，雑音電力等と呼ばれる．電力 ?? ⇐ 電気工学的用語

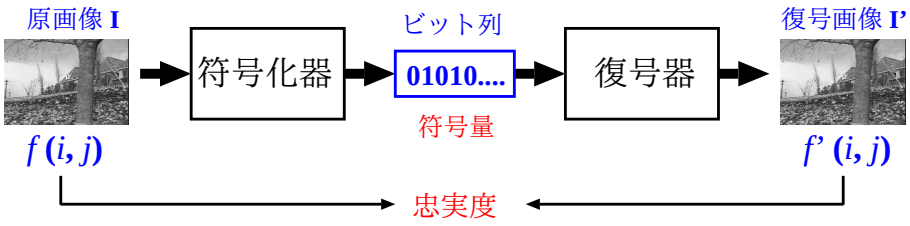



信号電力 ： $W_S = {i_S}^2 R$

雑音電力 ： $W_N = {i_N}^2 R$

信号対雑音比 ： $SNR = W_S \,/\, W_N$

## 非可逆符号化の評価 (3) — MSE と PSNR




- $SNR = W_S \,/\, W_N$ は大きな値 ⇒ log 取る [B] ⇒ 小さ過ぎ ⇒ 10 倍 [dB]

$$SNR = 10 \log \frac{\text{信号電力 } W_S}{\text{雑音（誤差）電力 } W_N} \ [\text{dB}] \quad \Rightarrow \quad \text{拝借.}$$

- 雑音（誤差成分）$W_N$ ： MSE（2 乗誤差成分），信号成分 $W_S$ は？

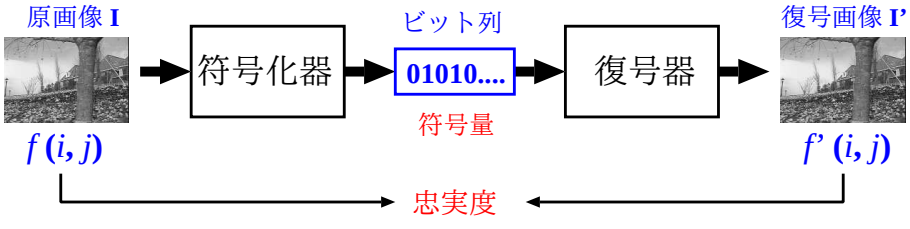
  ⇒ 8 ビット 256 階調で最大 (Peak) の成分 $W_S = 255^2$ ⇒ **PSNR**

$$PSNR = 10 \log \frac{255^2}{MSE} = 10 \log \frac{255^2}{\dfrac{1}{WH} \displaystyle\sum_{i=0}^{W-1} \sum_{j=0}^{H-1} \{(f(i,j) - f'(i,j)\}^2}$$

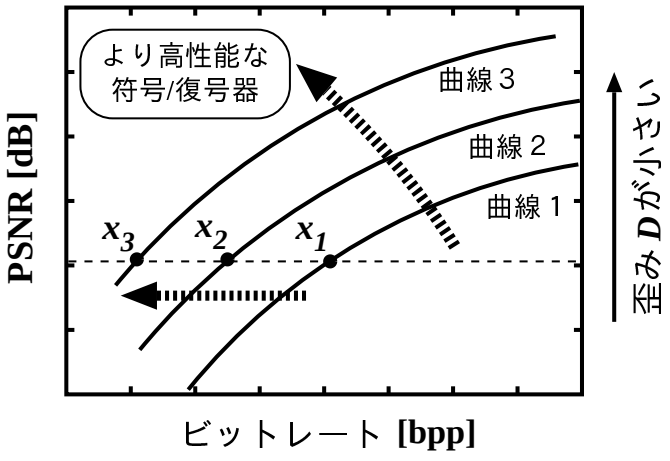
- **PSNR** と「平均 **2** 乗誤差 **MSE**」は同等



## 非可逆符号化の評価 (4)




- 符号量： **BPP**，**BPS** で計る ⇒ ビットレート $R$
- 忠実度： （ここでは）**MSE**，**PSNR** で計る ⇒ 歪み $D$




レート–歪み曲線

## 画像符号化法の理解に必要な基礎知識

- 線形代数（基底の概念，固有値と固有ベクトル程度）
- 確率統計（分散，平均程度）
- 信号処理（FIR フィルタ，周波数応答程度）
- 情報理論（エントロピー計算，情報源符号化定理程度）

⇓

**どれも初歩的な知識で十分**

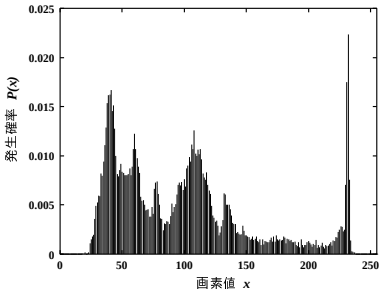


## 「前半：基礎の基礎」の内容

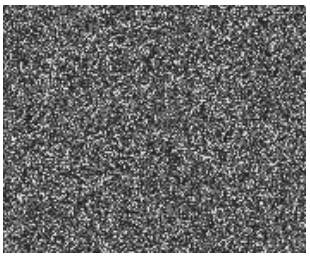
1. 画像と符号化 — 準備
2. 画像の性質と符号化 — 画素間相関のはなし
3. エントロピー符号化
4. 量子化
5. 相関除去法 (1) — 予測による方法
6. 相関除去法 (2) — 変換による方法
7. 動画像の性質と符号化 — 時間方向の相関除去
8. 以上を組み合わせて

# 2. 画像の性質と符号化 — 画素間相関のはなし

“Carphone” (8bit Y $176 \times 144$)




画素値の生起確率



“Carphone と同じ生起確率の疑似画像

「画素の配列」が「画像」として見えるためには，少なくとも

- 輝度値が急激に変化するエッジが存在
- 画像全体がエッジによってほぼ均一に見えるいくつかの領域 に分割

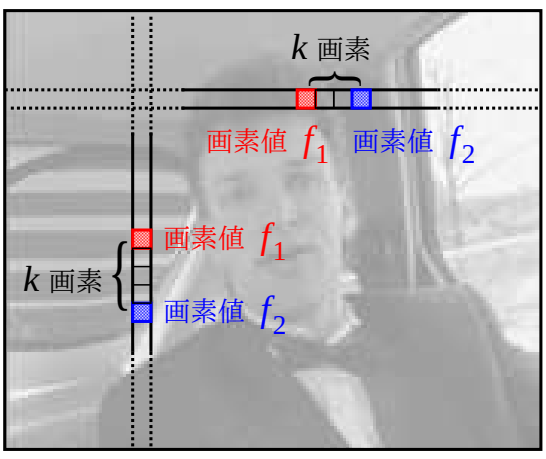
$\Downarrow$

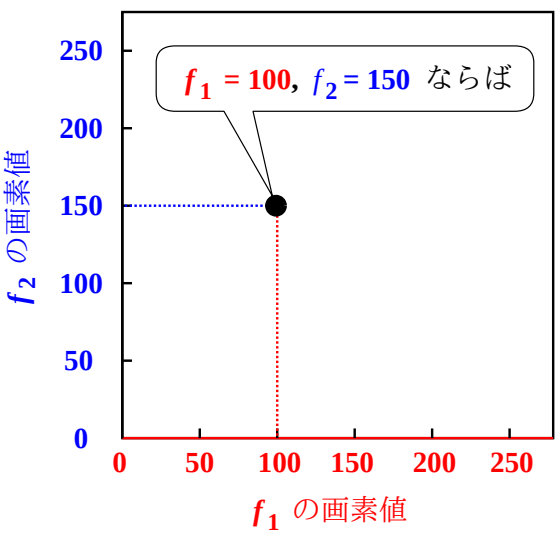
画素単体（1 次統計量） $\Rightarrow$ 近傍画素との関係（2 次統計量）を調べる必要



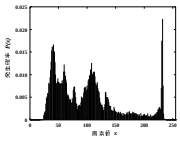
近傍画素との関係（2 次統計量）を調べるため

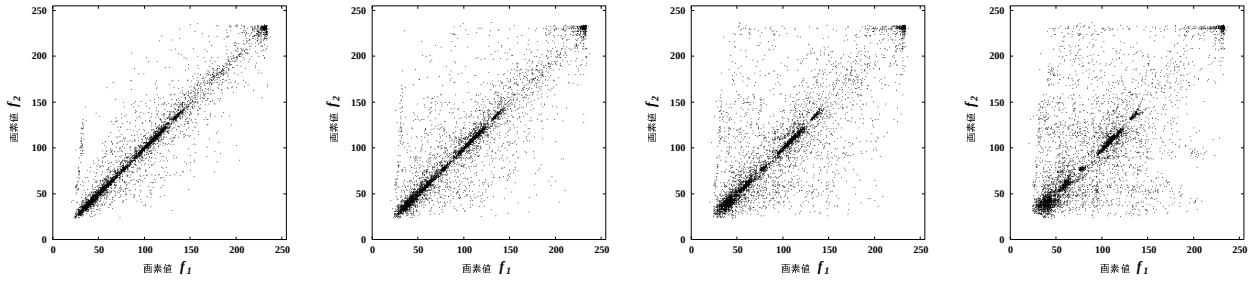
- 水平 or 垂直方向に「互いに $k$ 画素」離れた 2 つの画素を $f_1$ と $f_2$
- 全組 $(f_1, f_2)$ を 2 次元座標上にプロット（散布図）



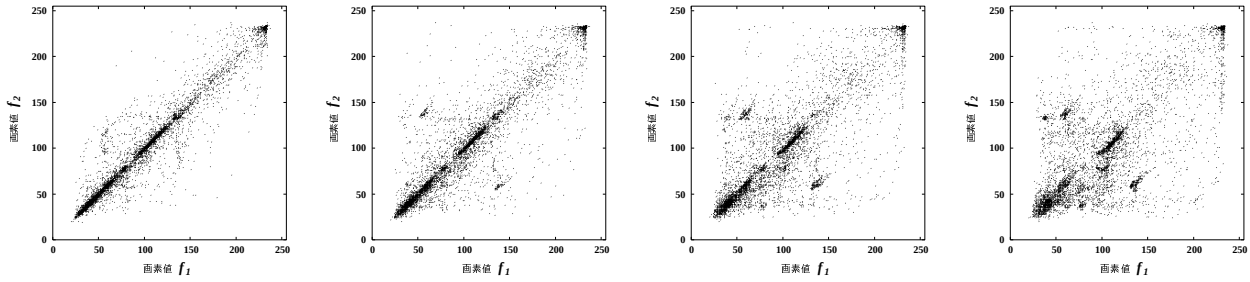



【画像 Carphoneの場合】





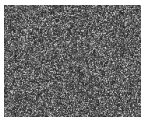

水平 $k=1$　　水平 $k=2$　　水平 $k=4$　　水平 $k=8$

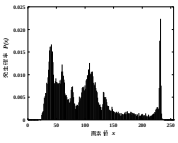


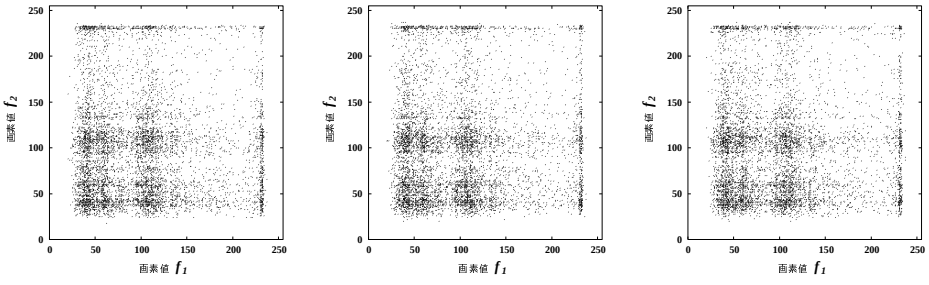

垂直 $k=1$　　垂直 $k=2$　　垂直 $k=4$　　垂直 $k=8$



【「疑似画像」の場合】

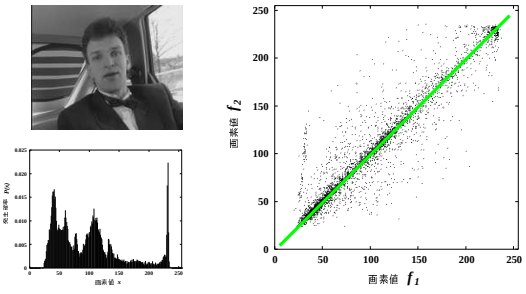







水平 $k=1$　　水平 $k=4$　　水平 $k=8$

画像 Carphone のような自然画像の場合



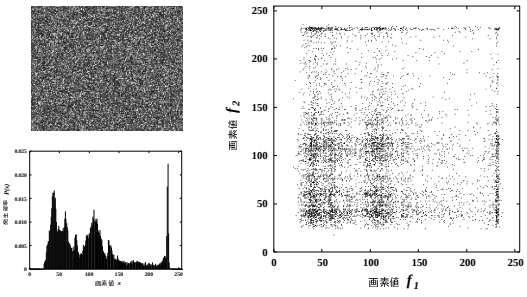

水平 $k = 1$

45° の直線 $f_1 = f_2$ に沿って分布

⇓

画素値 $f_1$ と $f_2$ は互いに近い！

「疑似画像」のような雑音の場合




水平 $k = 1$

全面に広がる

⇓

画素値 $f_1$ に $f_2$ 脈絡なし！



画像 Carphone のような自然画像の場合

(i) 水平垂直とも $k = 1$〜4 画素程度離れた 2 画素の値は近い（例外 : エッジ）
(ii) $k = 8$ 程度離れると，2 画素の値は近いとは言えない

「疑似画像」のような雑音の場合

(i) 隣接する 2 画素ですら値が近いとは言えない

⇓

「画素の配列」が「画像」として見えるのは...

「近傍画素の値が互いに近い」＝「画素間相関が高い」ため

⇓

自然画像が効率的に圧縮符号化できるのは ...

「画素間相関が高い」ため

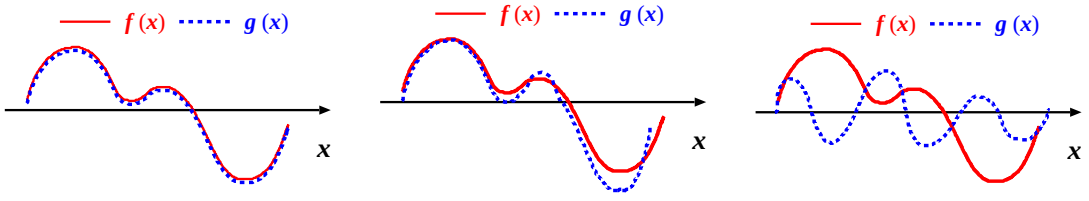
## 画素間相関の定量化 — 自己共分散関数

**【1 次元共分散】**

2 つの 1 次元信号（波形） $f(x)$ と $g(x)$ （$x$ は整数）の共分散

$$C_{fg} = \lim_{X\to\infty} \frac{1}{2X} \sum_{x=-X}^{X} (f(x)-\mu_f)(g(x)-\mu_g)$$

$$\equiv E_x\left[(f(x)-\mu_f)(g(x)-\mu_g)\right] \qquad (\mu_\cdot\text{: 波形 } \cdot \text{ の平均値})$$




(a) $C_{fg}$ 大　　(b) $C_{fg}$ 中　　(c) $C_{fg} \simeq 0$

⇓

共分散によって「波形 $f$ と $g$ の近さ（マッチ度）」を定量化できる



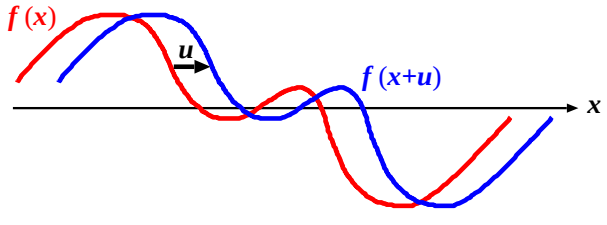
「近傍画素値との近さ」を測りたい...

⇒ $f(x)$ を $u$ だけずらした波形 $f(x+u)$ と $f(x)$ の共分散を考える

⇓

**自己共分散関数** $\cdots$ $g(x) = f(x+u)$ とおいた $C_{fg}$

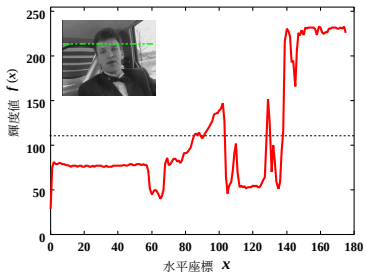
$$C_{\underbrace{ff}_{\text{自己}}}(\underbrace{u}_{\text{関数}}) = E_x[(f(x)-\mu_f)(f(x+u)-\mu_f)]$$



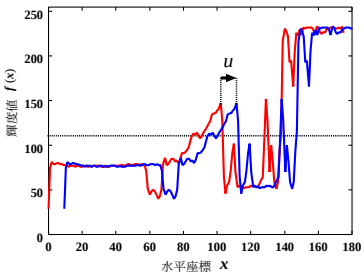

⇓

「$u$ だけ離れた値と平均的にどれだけ近いか」 ⇒ 画素間相関が定量化

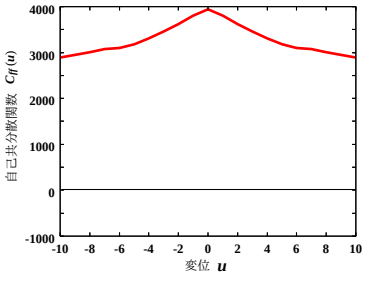
## 自己共分散関数の比較



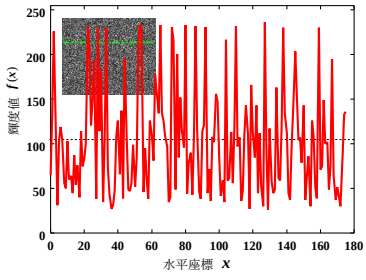

自然画像 40 ライン目



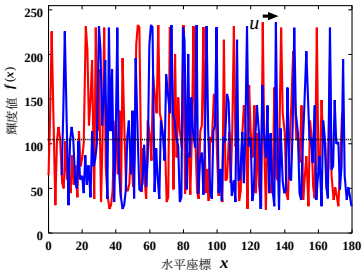

$u$ ずらす



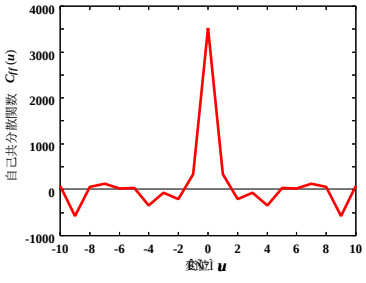

$C_{ff}(u)$




疑似画像 40 ライン目




$u$ ずらす




$C_{ff}(u)$

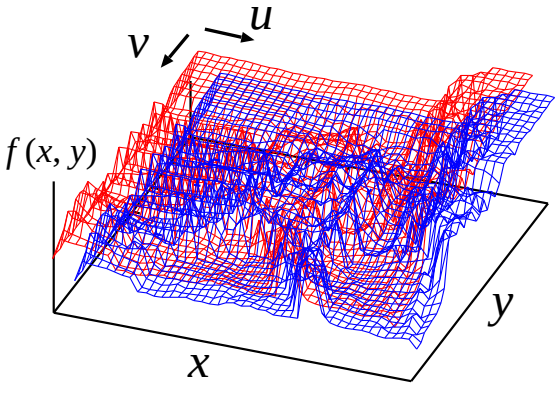
$\Downarrow$

画素間相関が高い自然画像 $\Rightarrow$ $C_{ff}(u)$ はゆっくり減少



## 画像に適用するため 2 次元に拡張 — 2 次元の自己共分散関数

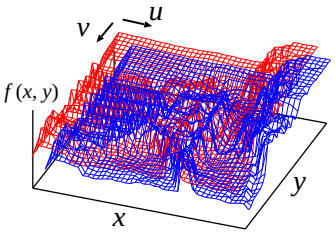
$$
\begin{aligned}
C_{ff}(u,v) &= \lim_{X,Y\to\infty} \frac{1}{4XY} \sum_{x=-X}^{X} \sum_{y=-Y}^{Y} (f(x,y)-\mu_f)(f(x+u,y+v)-\mu_g) \\
&= E_{xy}[(f(x,y)-\mu_f)(f(x+u,y+v)-\mu_f)]
\end{aligned}
$$

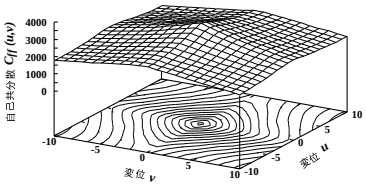

$(u,v)$ ずらして自己共分散関数を計算

## 2 次元画像での比較

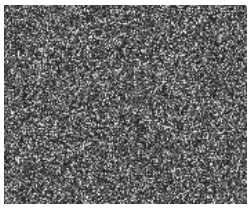
自然画像



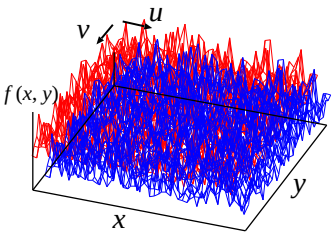

$(u, v)$ ずらす



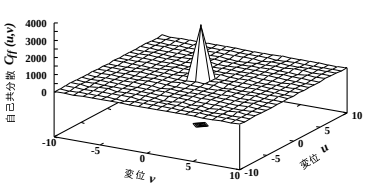

$C_{ff}(u, v)$



疑似画像




$(u, v)$ ずらす




$C_{ff}(u, v)$

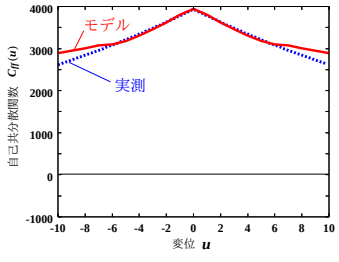
$\Downarrow$

画素間相関が高い自然画像 $\Rightarrow$ $C_{ff}(u, v)$ はゆっくり減少

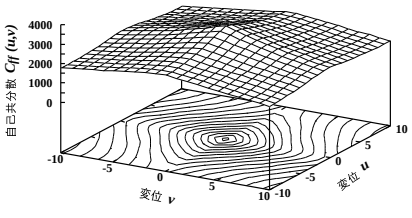


## 自然画像に対する自己共分散モデル

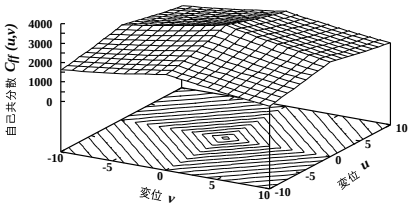
$$C_{ff}(u, v) = E_{xy}[(f(x, y) - \mu_f)(f(x + u, y + v) - \mu_f)]$$




1 次元 $C_{ff}(u)$




2 次元 $C_{ff}(u, v)$




2 次元自己共分散モデル

- 原点で最大．$C_{ff}(0) = \sigma_f^2$，$C_{ff}(0, 0) = \sigma_f^2$ （$f$ の分散）
- $\rho = C_{ff}(1)/\sigma_f^2$，$\rho_H = C_{ff}(1, 0)/\sigma_f^2$，$\rho_V = C_{ff}(0, 1)/\sigma_f^2$ とおくと，

  $$C_{ff}(u) \simeq \sigma_f^2 \, \rho^{|u|}, \qquad C_{ff}(u, v) \simeq \sigma_f^2 \, \rho_H^{|u|} \rho_V^{|v|}$$

  で良好に近似可能 $\Rightarrow$ 自己共分散モデル
- $\rho$：相関係数．自然画像 $\Rightarrow$ $\rho_H$，$\rho_V$ $\Rightarrow$ 0.8〜0.95 程度（1 に近い）

突然ですが ... □に入る漢字わかりますか？

| ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 康 | 忠 | 光 | 綱 | 吉 | 宣 | 継 | □ | 重 | 治 | 斉 | 慶 | 定 | 茂 | 喜 |

これはどうでしょうか？

| ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 郎 | 也 | 規 | 介 | 弘 | 司 | 悟 | □ | 吾 | 也 | 由 | 郎 | 志 | 智 | 裕 |

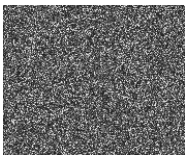


ちょっと強引ですが ...

| ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 康 | 忠 | 光 | 綱 | 吉 | 宣 | 継 | □ | 重 | 治 | 斉 | 慶 | 定 | 茂 | 喜 |

- 「自然画像」に対応 ⇒ 相関（脈絡）がある/高い
- 文字を省略しても影響小 ⇒ 情報欠損（誤差，雑音）小 ⇒ 圧縮できる

| ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 郎 | 也 | 規 | 介 | 弘 | 司 | 悟 | □ | 吾 | 也 | 由 | 郎 | 志 | 智 | 裕 |



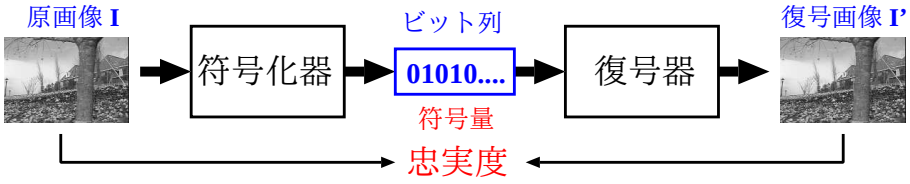
- 「疑似画像」に対応 ⇒ 相関（脈絡）全くない
- 文字を省略 ⇒ その分の情報（誤差，雑音）が完全欠損 ⇒ 圧縮できない

† 厳密には，生起確率の偏りがあれば，ちょっと圧縮できる

つまり ...

- 画素間相関がない疑似画像 ⇒ 圧縮できない（圧縮しただけの “祟り”）
- 画素間相関が高い自然画像 ⇒ 圧縮できる（圧縮しても “祟り” 小さい）

## 非可逆符号化のまとめ




- 自然画像の性質

  「画素間相関高い」＝「$C_{ff}(u,v)$ ゆっくり減少」＝「$\rho$ が 1 に近い」

- 圧縮符号化可能な理由：画素間相関高いため
- 画素間相関を完全に除去できれば $\Rightarrow$ 可逆符号化 $\Leftarrow$ 圧縮率低い
- 高い圧縮率が必要 $\Rightarrow$ 非可逆符号化 $\Leftarrow$ 「量子化」



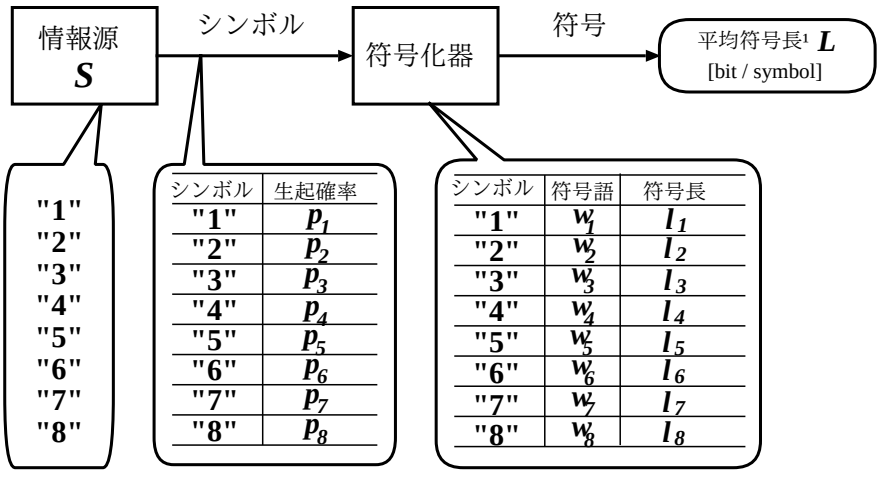
## 「前半：基礎の基礎」の内容

1. 画像と符号化 — 準備
2. 画像の性質と符号化 — 画素間相関と圧縮
3. エントロピー符号化
4. 量子化
5. 相関除去法 (1) — 予測による方法
6. 相関除去法 (2) — 変換による方法
7. 動画像の性質と符号化 — 時間方向の相関除去
8. 以上を組み合わせて

# 3. エントロピー符号化

## エントロピー符号化とは

- 符号化対象（数値，記号，文字 ...）に 2 進符号を割り当てる可逆符号化
- シンボル：符号化対象
- 情報源：シンボルを生成する源泉．統計的性質（生起確率等）のみに着目




- 符号化の効率：（可逆符号化のため）符号量のみで評価 ⇒ 平均符号長



## 平均符号長

- シンボル $N$ 個を符号化 ⇒ 全符号量 $R$ [bit] ⇒ 平均符号長 $L = N/R$
- 各シンボルの生起確率 $p_i$ と符号語長 $l_i$ が既知 ⇒ 平均符号長 $L = \sum_i p_i\, l_i$

### 平均符号長を小さくするには ...

生起確率（発生頻度）が高いシンボルに短い符号語を割り当てる

### 平均符号長の下限 — 情報源符号化定理

平均符号長 $L$ の下限はエントロピー

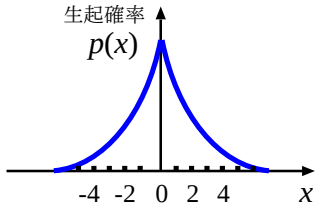
$$H(S) = -\sum_i p_i \log_2 p_i$$

で与えられる．平均符号長 $L$ が限りなく $H(S)$ に近い符号が存在する．

- エントロピー符号化と呼ばれるのはこのため

## 画像符号化とエントロピー符号化

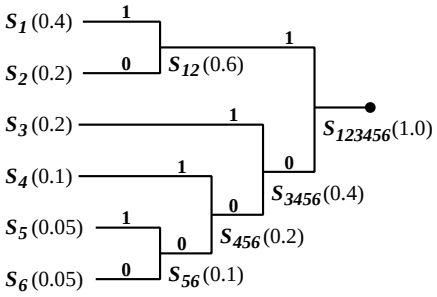
- 量子化器出力，動きベクトル，符号化モード等 ⇒ **発生頻度に偏り**



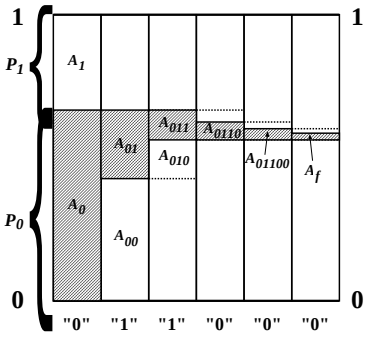

$$\begin{cases} \text{発生頻度：高 } 0,\ \pm 1 \text{ 等} & \Rightarrow \text{ 短い符号} \\ \text{発生頻度：小} & \Rightarrow \text{ 長い符号} \end{cases}$$

⇒ 平均符号長が減少 ⇒ 符号量減少

- 具体的な符号化手法の例




Huffman 符号化




算術符号化

- 実際には，直近に符号化した値に応じて符号語表/確率モデルを切替えるコンテクスト適応方式が用いられる



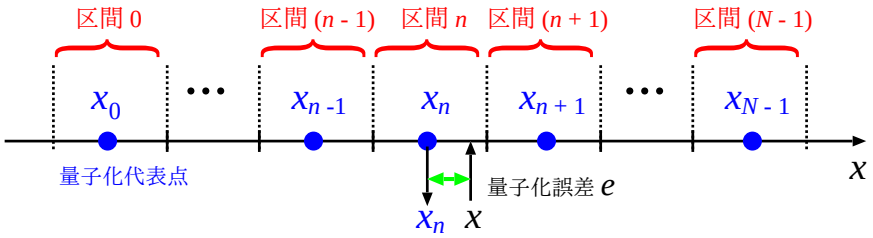
## 「前半：基礎の基礎」の内容

1. 画像と符号化 — 準備
2. 画像の性質と符号化 — 画素間相関と圧縮
3. エントロピー符号化
4. 量子化
5. 相関除去法 (1) — 予測による方法
6. 相関除去法 (2) — 変換による方法
7. 動画像の性質と符号化 — 時間方向の相関除去
8. 以上を組み合わせて

# 4. 量子化

## 量子化とは

- 数値精度を下げ，より少ないデータ量で表現する操作




量子化　$Q[x]$ ： $|x_n - x|$ が最小となる $n$ を出力
逆量子化 $Q^{-1}[x]$ ： $n$ に対応する $x_n$ を出力

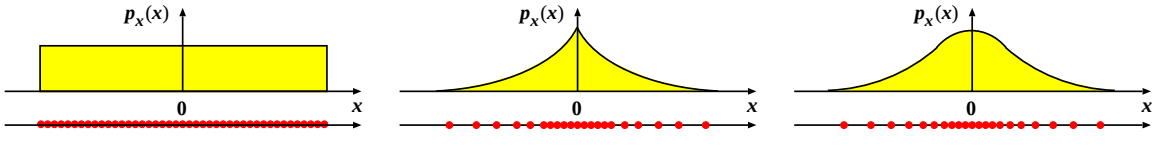
- $N = 2^L$ : $L$ ビット量子器 $\cdots$ 実数 $x$ を $L$ ビットで近似表現
- 区間 $n$ の区間幅 $\Delta_n$ : 量子化ステップ幅
- $\Delta_n$ が常に一定 $\Rightarrow$ 一様量子化器 $\longleftrightarrow$ 非一様量子化器
- 量子化–逆量子化 $Q^{-1}[Q[x]] = x_n \Rightarrow$ 量子化誤差 $e$ が発生

$$e = Q^{-1}[Q[x]] - x = x_n - x$$



## 量子化誤差の評価

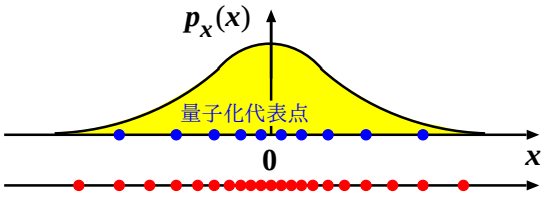
- 入力 $x$ で（いつも通り）2 乗誤差を平均 $\Rightarrow$ 分散 $\sigma_e^2$ に相当




$$\sigma_e^2 = \int_{-\infty}^{\infty} e^2 p_x(x)\,dx = \int_{-\infty}^{\infty} (x_n - x)^2 p_x(x)\,dx$$

## 量子化平均 2 乗誤差 $\sigma_e^2$ を小さくするには

- ビット数 $L$ を増やし，量子化代表点数 $N = 2^L$ を増やす $\Leftarrow$ 当たり前
- $L$ 一定下では？ $\Rightarrow$ $x$ の生起が多い（$p_x(x)$ 大きい）付近に多く割当て




- 非一様量子化に．$p_x(x)$ が既知 $\Rightarrow$ Max–Lloyed アルゴリズムで設計可
  $\Rightarrow$ 最適量子化器

### 最適量子化器の得失

【利点】

- 既知の $p_x(x)$，$L$ 一定下で量子化平均 2 乗誤差最小
- 量子化出力 $n$ の生起数がフラット $\Rightarrow$ エントロピー符号化不要

【欠点】

- 量子化処理の計算量大（各代表点との距離最小の $n$ 求める）
- $p_x(x)$ が必要 $\Leftarrow$ 対象依存．$p_x(x)$ が変化すると性能低下

$\Downarrow$

### 一様量子化の方が扱い易い

【利点】

- 量子化処理が簡単（量子化ステップ幅 $\Delta$ で割算）
- $p_x(x)$ 不要 $\Rightarrow$ 未知の対象画像についても対応可能

【欠点】

- 最適量子化器には劣る
- エントロピー符号化が必要



### $L$ ビット一様量子化器の量子化平均 2 乗誤差

- $L \geq 2$〜3 程度に大きい（高レート近似）
- 入力 $x$ の生起に極端な偏りはない（いろんな入力が入る）

$\Downarrow$

#### $L$ ビット一様量子化器の平均 2 乗誤差 — レート–歪み関数

$$\underbrace{\sigma_e^2}_{\text{誤差}} = \underbrace{\epsilon^2}_{\text{定数}} \cdot 2^{-2L} \cdot \underbrace{\sigma_x^2}_{\text{入力 } x \text{ の分散}}$$

$\Downarrow$

- $L$ が 1 ビット増える $\Rightarrow$ 誤差 $\sigma_e^2$ は 1/4 に
- $L$ 一定 $\Rightarrow$ 誤差 $\sigma_e^2$ は入力 $x$ の分散 $\sigma_x^2$ に比例
  $\Rightarrow$ $\sigma_x^2$ が小さいほど量子化誤差小 $\Rightarrow$ 画質，PSNR 高

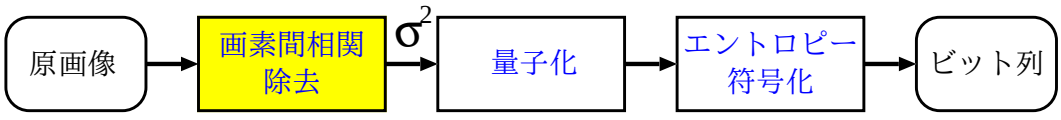
「前半：基礎の基礎」の内容

1. 画像と符号化 — 準備
2. 画像の性質と符号化 — 画素間相関と圧縮
3. エントロピー符号化
4. 量子化
5. 相関除去法 (1) — 予測による方法
6. 相関除去法 (2) — 変換による方法
7. 動画像の性質と符号化 — 時間方向の相関除去
8. 以上を組み合わせて



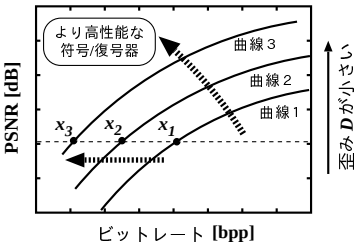
# 5. 相関除去法 (1) — 予測による方法

画像符号化器の一般的構成



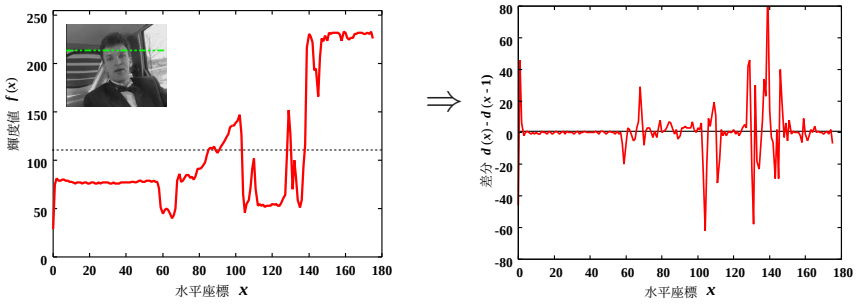

予測による方法 — 予測符号化，差分符号化

与えられた符号量で，原画像–復号画像間の PSNR を最大にするには？




- 画像 ⇒ 画素間相関が高い ⇒ 前後の画素値は互いに近い
- PSNR の低下 ⇒ 量子化誤差 ⇐ 「量子化入力の分散 $\sigma^2$」に比例
- PSNR の最大化 ⇒ $\sigma^2$ を小さくすれば良い（一般論）
- 予測符号化 ⇒ 予測値との差分を符号化
  ⇒ 画素間相関が除去され，$\sigma^2$ も小さく

## 予測符号化の原理 — 簡単な例 (1)




- 符号化：$f(x)$ に対し，左隣の画素値 $f(x-1)$ との差分を計算

  $d(x) = f(x) - f(x-1)$

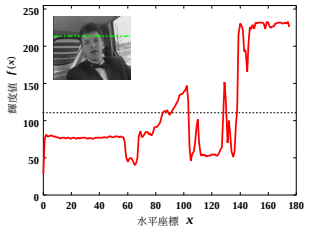
- 復号側：$d(x)$ を受取り，左隣の復号画素値に加える

  $f'(x) = f'(x-1) + d(x)$

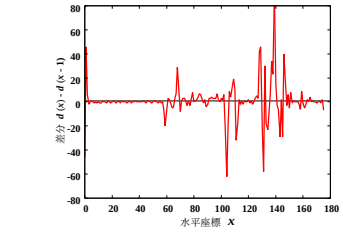
| | $x$ | $-1$ | 0 | 1 | 2 | 3 | 4 | 5 | $\cdots$ | 174 | 175 | 分散$\sigma^2$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 符号 | $f(x)$ | (0) | 29 | 75 | 81 | 79 | 79 | 80 | $\cdots$ | 233 | 226 | 3893.6 |
| | | | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | | ↓ | ↓ | |
| | $d(x)$ | | 29 | 46 | 6 | $-2$ | 0 | 1 | $\cdots$ | 2 | $-7$ | 198.3 |
| 復号 | | | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | | ↓ | ↓ | |
| | $f'(x)$ | (0) | 29 | 75 | 81 | 79 | 79 | 80 | $\cdots$ | 233 | 226 | |



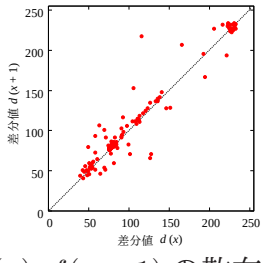
## 予測符号化の原理 — 簡単な例 (2)



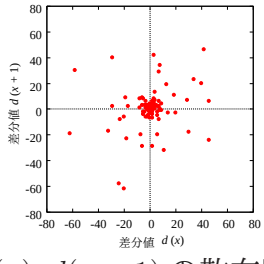

$f(x)$



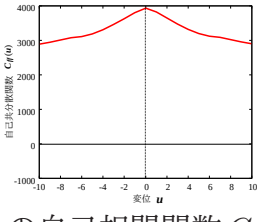

$d(x) = f(x) - f(x-1)$



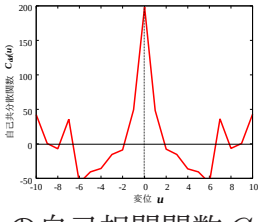

$f(x)$–$f(x+1)$ の散布図




$d(x)$–$d(x+1)$ の散布図




$f(x)$ の自己相関関数 $C_{ff}(u)$




$d(x)$ の自己相関関数 $C_{dd}(u)$

差分を取ると画素間相関が除去されている

## 差分 $d(x) = f(x) - f(x-1)$ の分散 $\sigma_d^2$ はどの程度低下する？

- $f(x)$ の分散を $\sigma_f^2$，相関係数を $\rho$，$f^\mu(x) = f(x) - \mu_f$ とおく
- $C_f f(u) = \sigma_f^2 \rho^{|u|}$ のモデルを利用

$$\begin{aligned}\sigma_d^2 &= E_x\left[(d(x)-\mu_d)^2\right] = E_x[\{\overbrace{(f(x+1)-\mu_f)}^{f^\mu(x+1)} - \overbrace{(f(x)-\mu_f)}^{f^\mu(x)}\}^2] \\ &= \underbrace{E_x[f^\mu(x+1)^2]}_{\sigma_f^2} + \underbrace{E_x[f^\mu(x)^2]}_{\sigma_f^2} - 2\underbrace{E_x[f^\mu(x+1)f^\mu(x)]}_{C_{ff}(1)=\sigma_f^2\rho} \\ &= \sigma_f^2 \cdot 2(1-\rho)\end{aligned}$$

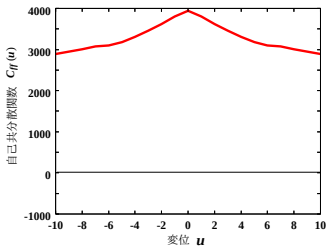
## 差分 $d(x) = f(x) - f(x-1)$ の画素間相関（相関係数 $\rho_d$ ）は？

- 同様に $C_{dd}(u)$ を計算し，相関係数 $\rho_d$ を求めると，

$$\rho_d = 2\rho - 1 - \rho^2$$

## $\rho = 0.96$ とすると

- $\sigma_d^2 = 0.08\sigma_f^2 \Rightarrow$ 1/10 程度に低下
- $\rho_d = 0.0016 \Rightarrow$ 相関はほぼ除去






## 予測符号化とは

**【符号化手順】**

1. 符号化対象 $f(x)$ に対する予測値 $p(x)$ を求める（先の例：$p(x) = f(x-1)$）
2. 差分 $d(x) = f(x) - p(x)$ を計算，量子化して符号を構成 $\Rightarrow$ 復号側へ

**【復号手順】**

1. 差分を逆量子化して $d'(x)$ とする（量子化誤差のため $d(x) \neq d'(x)$）
2. 符号側と同じ手順で予測値 $p(x)$ を求める
3. 予測値に加える：$f'(x) = d'(x) + p(x)$（量子化誤差のため $f(x) \neq f'(x)$）

## 先の例に問題が … 量子化を伴う非可逆符号化の場合

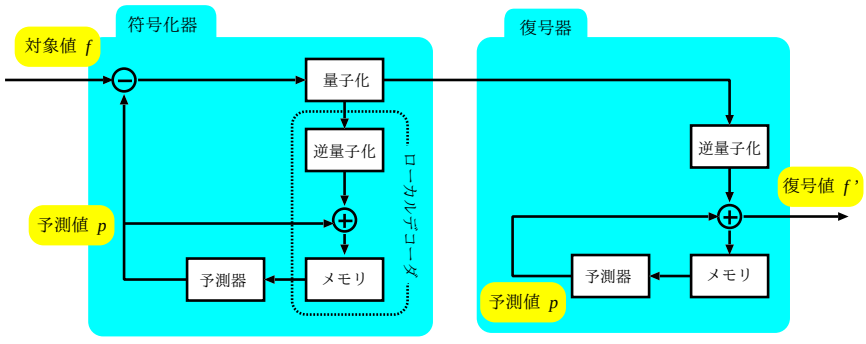
- 符号化側の予測値 $p(x)$：原画像 $f(x-1)$
- 復号側の予測値 $p(x)$：1 回前に復号した画素値 $f'(x-1)$ $(\neq f(x-1))$
- $p(x) = f'(x-1)$ を用いて復号すると，

$$f'(x) = \overbrace{d'(x)}^{\text{今回の量子化誤差含む}} + \overbrace{f'(x-1)}^{\text{1 回前の量子化誤差含む}}$$

- $f'(x+1), f'(x+2), \cdots$ には量子化誤差が蓄積し続ける $\Rightarrow$ ドリフト現象

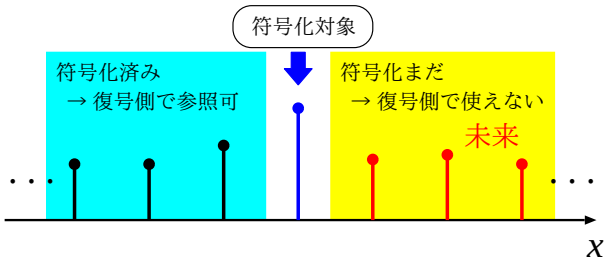
## 解決法

- 非可逆符号化 ⇒ 復号側では原画素値は得られない ⇒ 符号化側で対策
- 符号化側でも復号，復号側と同じ復号値を用いて予測 ⇒ 予測値同一に

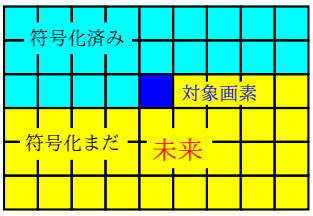



## もうひとつの注意点 — 復号側で作れない予測値はダメ

- 符号化まだ（未来）の画素 ⇒ 復号側で参照不可 ⇒ 予測に用いない




1 次元の場合




2 次元（ラスタスキャン順）の場合



## 予測符号化の効率を上げるには ...

### 予測器の性能を上げる ⇒ 差分 $d(x) = f(x) - p(x)$ を低減

- 線形予測 $\cdots$ $f(x-1)$，$f(x-2)$，$\cdots$，$f(x-N)$ の線形結合で予測

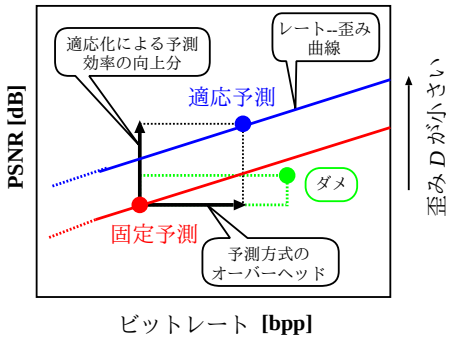
$$p(x) = a_1 f(x-1) + a_2 f(x-1) + \cdots a_N f(x-N) \quad (a_k\text{:予測係数})$$

予測誤差（平均 2 誤差） $E_x[(f(x) - p(x))^2]$ を最小に ⇒ 線形予測器

- 非線形予測も可能 $p(x) = \mathrm{med}[f(x-1), f(x-2), f(x-3)]$ など ...

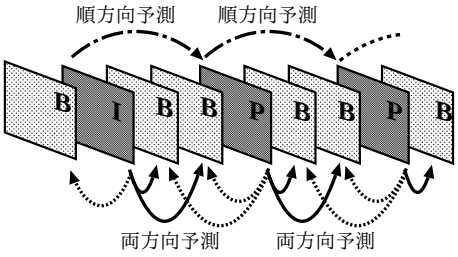
### 適応予測

- 画像は非定常 ⇒ より良い予測器を切替えて使う ⇒ 適応予測
- 用いる予測器/予測方式は，
  - 付加情報として復号側に伝送
    ⇒ 符号量増加（オーバヘッド）
  - 符号化済み画素から自動判定
    ⇒ オーバヘッドはない

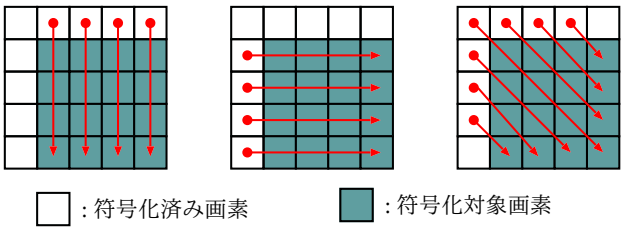


## 動画像符号化への応用 — 原理が単純で扱い易い

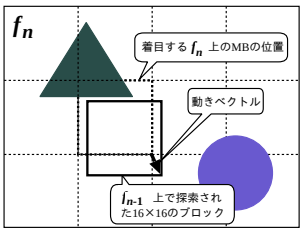
- フレーム間予測符号化（後述）




- フレーム内予測符号化




- 動きベクトルや符号化モードの符号化



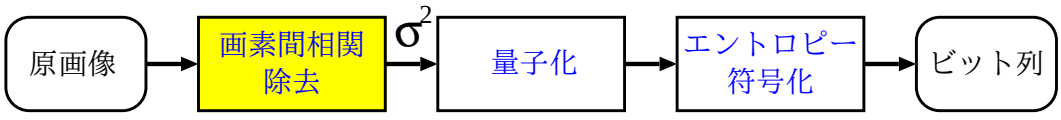



## 「前半：基礎の基礎」の内容

1. 画像と符号化 — 準備
2. 画像の性質と符号化 — 画素間相関と圧縮
3. エントロピー符号化
4. 量子化
5. 相関除去法 (1) — 予測による方法
6. 相関除去法 (2) — 変換による方法
7. 動画像の性質と符号化 — 時間方向の相関除去
8. 以上を組み合わせて

# 6. 相関除去法 (2) — 変換による方法

## 画像符号化器の一般的構成

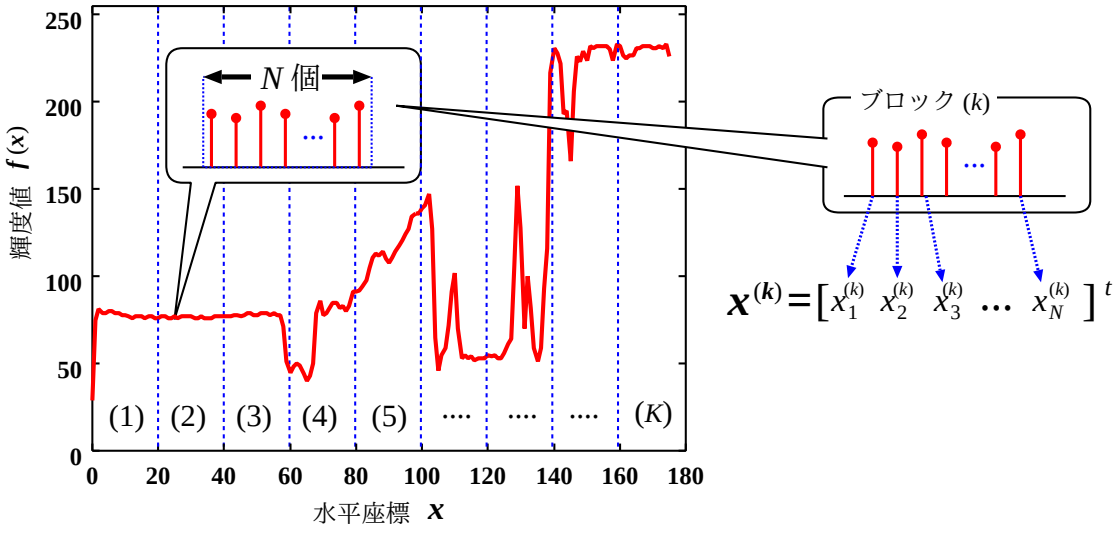



## 変換による方法 — 変換符号化とは

- 符号化対象 $f(x)$ を，互いに相関がないデータ $F(x)$ に変換 $\Rightarrow$ 無相関化
- $F(x)$ を量子化し，符号を構成
- ブロック単位，非ブロック単位の方法に大別 $\Leftarrow$ ここでは前者
- まずは 1 次元データで，後に 2 次元データ（画像）に拡張



## ブロック分割とブロック単位の無相関化




- 対象 $\Rightarrow$ $N$ 個のデータを含むブロックに分割 （$K$ ：ブロック数）
- $k$ 番目のブロック内の画素値を縦ベクトルに並べる

$$\boldsymbol{x}^{(k)} = \left[ \begin{array}{cccc} x_1^{(k)} & x_2^{(k)} & x_3^{(k)} & \cdots & x_N^{(k)} \end{array} \right]^t$$

## 自己共分散行列 (1)

- $\boldsymbol{x}^{(k)} = [x_1, x_2, \cdots, x_N^{(k)}]^t$ の各要素は互いに相関 $\Rightarrow$ 自己共分散で評価

$$c_{x_i x_j} = E_k\left[(x_i^{(k)} - \mu)(x_j^{(k)} - \mu)\right] \qquad (\mu：\text{平均値})$$

- $x_i^{(k)} - \mu$ を改めて $x_i^{(k)}$ と表す $\Rightarrow$ 簡潔に

$$c_{x_i x_j} = E_k\left[\underbrace{(x_i^{(k)} - \mu)}_{x_i^{(k)}}\underbrace{(x_j^{(k)} - \mu)}_{x_j^{(k)}}\right] = E_k\left[x_i^{(k)} x_j^{(k)}\right]$$

- $C_{x_i x_j}$ を $ij$ 要素とする行列 $\Rightarrow$ 共分散行列 $\boldsymbol{C_{xx}}$

$$\boldsymbol{C_{xx}} = \begin{bmatrix} E_k[x_1^{(k)}x_1^{(k)}] & E_k[x_1^{(k)}x_2^{(k)}] & \cdots & E_k[x_1^{(k)}x_N^{(k)}] \\ E_k[x_2^{(k)}x_1^{(k)}] & E_k[x_2^{(k)}x_2^{(k)}] & \cdots & E_k[x_2^{(k)}x_N^{(k)}] \\ \vdots & \vdots & \ddots & \vdots \\ E_k[x_N^{(k)}x_1^{(k)}] & E_k[x_N^{(k)}x_2^{(k)}] & \cdots & E_k[x_N^{(k)}x_N^{(k)}] \end{bmatrix}$$

- 平均操作 $E_k[\,\cdot\,]$ を行列の外に出し，さらに

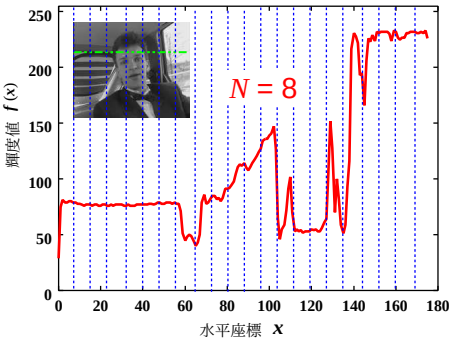


## 自己共分散行列 (2)

- ベクトル表記 $\boldsymbol{x}^{(k)} = \begin{bmatrix} x_1^{(k)} & x_2^{(k)} & x_3^{(k)} & \cdots & x_N^{(k)} \end{bmatrix}^t$ を用いると，

$$\boldsymbol{C_{xx}} = E_k\begin{bmatrix} x_1^{(k)}x_1^{(k)} & x_1^{(k)}x_2^{(k)} & \cdots & x_1^{(k)}x_N^{(k)} \\ x_2^{(k)}x_1^{(k)} & x_2^{(k)}x_2^{(k)} & \cdots & x_2^{(k)}x_N^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ x_N^{(k)}x_1^{(k)} & x_N^{(k)}x_2^{(k)} & \cdots & x_N^{(k)}x_N^{(k)} \end{bmatrix} = E_k\left[\underbrace{\left(\boldsymbol{x}^{(k)}\right)^t}_{\text{横ベクトル}} \cdot \underbrace{\left(\boldsymbol{x}^{(k)}\right)}_{\text{縦ベクトル}}\right]$$

## $\boldsymbol{C_{xx}}$ の計算例 — ブロックサイズ $N = 8$




$$\boldsymbol{C_{xx}} = \begin{bmatrix} 3181 & 2889 & 2729 & 2676 & 2583 & 2510 & 2475 & 2419 \\ 2889 & 2999 & 2940 & 2839 & 2743 & 2660 & 2605 & 2543 \\ 2729 & 2940 & 3117 & 3060 & 2931 & 2858 & 2791 & 2718 \\ 2676 & 2839 & 3060 & 3225 & 3159 & 3063 & 2993 & 2911 \\ 2583 & 2743 & 2931 & 3159 & 3309 & 3255 & 3153 & 3076 \\ 2510 & 2660 & 2858 & 3063 & 3255 & 3359 & 3277 & 3173 \\ 2475 & 2605 & 2791 & 2993 & 3153 & 3277 & 3369 & 3289 \\ 2419 & 2543 & 2718 & 2911 & 3076 & 3173 & 3289 & 3385 \end{bmatrix}$$

- $x_i$–$x_j$ 間の相関高い $\Leftarrow$ 非対角要素が大きい

## 自己共分散行列 (3)

- 我々の目的 $\Rightarrow$ 画素ベクトル $\boldsymbol{x}^{(k)}$ $\Rightarrow$ 無相関なデータ列 $\boldsymbol{y}^{(k)}$ に変換
  $\Rightarrow$ 変換できれば $\boldsymbol{C_{yy}}$ は対角行列

$$\boldsymbol{x}^{(k)} \qquad\qquad \Rightarrow \qquad\qquad \boldsymbol{y}^{(k)}$$

$$\boldsymbol{C_{xx}} = \begin{bmatrix} 3181 & 2889 & 2729 & 2676 & 2583 & 2510 & 2475 & 2419 \\ 2889 & 2999 & 2940 & 2839 & 2743 & 2660 & 2605 & 2543 \\ 2729 & 2940 & 3117 & 3060 & 2931 & 2858 & 2791 & 2718 \\ 2676 & 2839 & 3060 & 3225 & 3159 & 3063 & 2993 & 2911 \\ 2583 & 2743 & 2931 & 3159 & 3309 & 3255 & 3153 & 3076 \\ 2510 & 2660 & 2858 & 3063 & 3255 & 3359 & 3277 & 3173 \\ 2475 & 2605 & 2791 & 2993 & 3153 & 3277 & 3369 & 3289 \\ 2419 & 2543 & 2718 & 2911 & 3076 & 3173 & 3289 & 3385 \end{bmatrix} \Rightarrow \boldsymbol{C_{yy}} = \begin{bmatrix} \bullet & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \bullet & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \bullet & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \bullet & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \bullet & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \bullet & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \bullet \end{bmatrix}$$

- $\boldsymbol{C_{xx}} = E_k\left[\left(\boldsymbol{x}^{(k)}\right)^t \cdot \left(\boldsymbol{x}^{(k)}\right)\right]$ は対称行列 $\Rightarrow$ $\boldsymbol{C_{xx}}$ を対角化する

## 対称行列 $\boldsymbol{A}$ の対角化

- $\boldsymbol{A}$ の固有値は実数 $\lambda_1, \cdots, \lambda_N$.
- 固有ベクトル $\boldsymbol{l}_1, \boldsymbol{l}_2, \cdots, \boldsymbol{l}_N$ は互いに直交 $\boldsymbol{L}^t\boldsymbol{L} = \boldsymbol{I}$.
- 固有ベクトルを並べて $\boldsymbol{L} = [\boldsymbol{l}_1, \boldsymbol{l}_2, \cdots, \boldsymbol{v}_N]$ を作ると,

$$\boldsymbol{L}^t\boldsymbol{A}\boldsymbol{L} = \mathrm{diag}\begin{bmatrix} \lambda_1 & \cdots & \lambda_N \end{bmatrix}$$



## カルーネン–レーブ変換 (KLT)

- 共分散行列 $\boldsymbol{C_{xx}}$ の固有ベクトル $\boldsymbol{L} = [\boldsymbol{l}_1, \boldsymbol{l}_2, \cdots, \boldsymbol{v}_N]$ を用いた変換
- $|\boldsymbol{l}_i| = 1$ としておくと $\boldsymbol{L}^t\boldsymbol{L} = \boldsymbol{I}$（正規直交）
- $\boldsymbol{y}^{(k)} = \boldsymbol{L}^t\boldsymbol{x}^{(k)}$ で変換 $\Rightarrow$ $\boldsymbol{y}^{(k)} = \begin{bmatrix} y_1^{(k)} & y_2^{(k)} & y_3^{(k)} & \cdots & y_N^{(k)} \end{bmatrix}^t$ ：変換係数

$$\begin{aligned} \boldsymbol{C_{yy}} &= E_k\left[\left(\boldsymbol{y}^{(k)}\right)^t \cdot \left(\boldsymbol{y}^{(k)}\right)\right] = E_k\left[\left(\boldsymbol{L}^t\boldsymbol{x}^{(k)}\right)^t \cdot \left(\boldsymbol{L}^t\boldsymbol{x}^{(k)}\right)\right] \\ &= \boldsymbol{L}^t E_k\left[\left(\boldsymbol{x}^{(k)}\right)^t \cdot \left(\boldsymbol{x}^{(k)}\right)\right]\boldsymbol{L}^t = \boldsymbol{L}^t\boldsymbol{C_{xx}}\boldsymbol{L} = \text{対角行列} \Rightarrow \text{無相関} \end{aligned}$$

- $\boldsymbol{L}\boldsymbol{y}^{(k)} = \boldsymbol{L}\boldsymbol{L}^t\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k)} \Rightarrow$ 逆変換：$\boldsymbol{x}^{(k)} = \boldsymbol{L}\boldsymbol{y}^{(k)} \Rightarrow$ 完全復元
- 変換係数 $y_i^{(k)}$ を量子化 $\tilde{y}_i^{(k)} \Rightarrow$ 逆変換 $\tilde{\boldsymbol{x}}^{(k)} = \boldsymbol{L}\tilde{\boldsymbol{y}}^{(k)}$
- あらゆる変換 $\boldsymbol{x} \to \boldsymbol{y}$ の中で量子化による誤差が最小

## ただし，実際には使い難い ...

- $\boldsymbol{C_{xx}}$ は画像依存 $\Rightarrow$ $\boldsymbol{L}$ も画像依存（画像を変えると計算し直し）
- 変換行列 $\boldsymbol{L}$ は符号側，復号側で共有 $\Rightarrow$ オーバヘッド

## そこで ... 離散コサイン変換 (DCT) へ

- 画素間相関モデル $C_{x_i x_j} = \sigma^2 \rho^{|i-j|}$ を利用

$$\boldsymbol{C}_{\boldsymbol{xx}}^{\text{model}} = \sigma^2 \begin{bmatrix} 1 & \rho & \rho^2 & \rho^3 & \rho^4 & \rho^5 & \rho^6 & \rho^7 \\ \rho & 1 & \rho & \rho^2 & \rho^3 & \rho^4 & \rho^5 & \rho^6 \\ \rho^2 & \rho & 1 & \rho & \rho^2 & \rho^3 & \rho^4 & \rho^5 \\ \rho^3 & \rho^2 & \rho & 1 & \rho & \rho^2 & \rho^3 & \rho^4 \\ \rho^4 & \rho^3 & \rho^2 & \rho & 1 & \rho & \rho^2 & \rho^3 \\ \rho^5 & \rho^4 & \rho^3 & \rho^2 & \rho & 1 & \rho & \rho^2 \\ \rho^6 & \rho^5 & \rho^4 & \rho^3 & \rho^2 & \rho & 1 & \rho \\ \rho^7 & \rho^6 & \rho^5 & \rho^4 & \rho^3 & \rho^2 & \rho & 1 \end{bmatrix}$$

- $\boldsymbol{C}_{\boldsymbol{xx}}^{\text{model}}$ の固有ベクトルは解析的に解ける
- $\rho$ は 1 に近いが画像依存 $\Rightarrow$ $\rho \to 1$ の極限 $\Rightarrow$ 離散コサイン変換 (DCT)

### $N$ 点離散コサイン変換 (DCT)

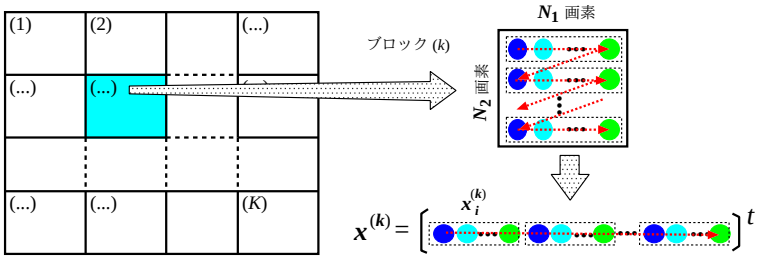
$$F(u) = \frac{2C(u)}{N} \sum_{i=0}^{N-1} f(i) \cdot \cos \frac{(2i+1)u\pi}{2N}$$

$$f(i) = \frac{2}{N} \sum_{u=0}^{N-1} C(u)F(u) \cdot \cos \frac{(2i+1)u\pi}{2N}$$

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



## (2 次元の) 画像に適用するには (1)

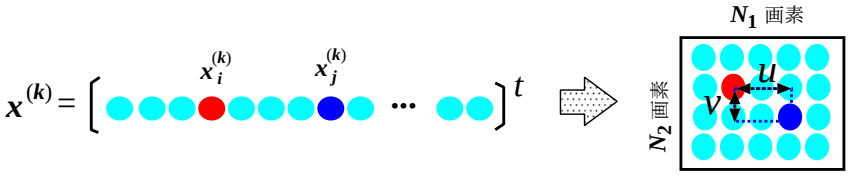



- $N_1 \times N_2$ のブロックに分割 ($K$ : ブロック数)
- $k$ 番目のブロック内の画素値を縦ベクトルに配置

$$\boldsymbol{x}^{(k)} = \begin{bmatrix} x_1^{(k)} & x_2^{(k)} & x_3^{(k)} & \cdots & x_N^{(k)} \end{bmatrix}^t$$

- $\boldsymbol{C}_{\boldsymbol{xx}}$ を計算 $\Leftarrow$ 画素間相関モデル $C_{x_i x_j} = \sigma^2 \rho_H^{|u|}, \rho_V^{|v|}$ を利用

  $u =$ 画素 $x_i$ と $x_j$ の水平距離, $u =$ 画素 $x_i$ と $x_j$ の垂直距離




- $N_1 \times N_2 = 3 \times 2$ のブロックで $\boldsymbol{C}_{\boldsymbol{xx}}$ を計算してみると ...

## （2 次元の）画像に適用するには (2)

$$\boxed{\begin{matrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \end{matrix}} \quad \Rightarrow \quad \boldsymbol{x}^{(k)} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{bmatrix}$$

$$\boldsymbol{C_{xx}} = \begin{bmatrix} 1 & \rho_H & \rho_H^2 & \rho_V & \rho_V\rho_H & \rho_V\rho_H^2 \\ \rho_H & 1 & \rho_H & \rho_V\rho_H & \rho_V & \rho_V\rho_H \\ \rho_H^2 & \rho_H & 1 & \rho_V\rho_H^2 & \rho_V\rho_H & \rho_V \\ \rho_V & \rho_V\rho_H & \rho_V\rho_H^2 & 1 & \rho_H & \rho_H^2 \\ \rho_V\rho_H & \rho_V & \rho_V\rho_H & \rho_H & 1 & \rho_H \\ \rho_V\rho_H^2 & \rho_V\rho_H & \rho_V & \rho_H^2 & \rho_H & 1 \end{bmatrix} = \begin{bmatrix} 1 & \rho_H & \rho_H^2 & \rho_V & \rho_V\rho_H & \rho_V\rho_H^2 \\ \rho_H & 1 & \rho_H & \rho_V\rho_H & \rho_V & \rho_V\rho_H \\ \rho_H^2 & \rho_H & 1 & \rho_V\rho_H^2 & \rho_V\rho_H & \rho_V \\ \rho_V & \rho_V\rho_H & \rho_V\rho_H^2 & 1 & \rho_H & \rho_H^2 \\ \rho_V\rho_H & \rho_V & \rho_V\rho_H & \rho_H & 1 & \rho_H \\ \rho_V\rho_H^2 & \rho_V\rho_H & \rho_V & \rho_H^2 & \rho_H & 1 \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} 1 & \rho_V \\ \rho_V & 1 \end{bmatrix}}_{\boldsymbol{C}_V} \otimes \underbrace{\begin{bmatrix} 1 & \rho_H & \rho_H^2 \\ \rho_H & 1 & \rho_H \\ \rho_H^2 & \rho_H & 1 \end{bmatrix}}_{\boldsymbol{C}_H} \quad （\otimes：クロネッカー積）$$

### クロネッカ積で与えられる行列の固有値と固有ベクトル

- $\boldsymbol{C} = \boldsymbol{A} \otimes \boldsymbol{B}$

  $\boldsymbol{A}$ ： 固有値 $\alpha_i$，固有ベクトル $\boldsymbol{a}_i$ $(i = 1, 2, \cdots, N_A)$

  $\boldsymbol{B}$ ： 固有値 $\beta_j$，固有ベクトル $\boldsymbol{b}_j$ $(j = 1, 2, \cdots, N_B)$

  の固有値は $\alpha_i\beta_j$，固有ベクトルは $\boldsymbol{a}_i \otimes \boldsymbol{b}_j$



## （2 次元の）画像に適用するには (3)

- 画素間相関モデル $\sigma^2 \rho_H^{|u|} \rho_V^{|u|}$ $\Rightarrow$ 水平 $u$ と垂直 $v$ に分離できると，
- 2 次元の KLT $\Rightarrow$ 水平と垂直方向の 1 次元 KLT の「ある種の積」
- 2 次元 DCT $\Rightarrow$ 水平と垂直の 1 次元 DCT の「ある種の積」 $\Rightarrow$ 分離型

## $N$ 点離散コサイン変換 (DCT)

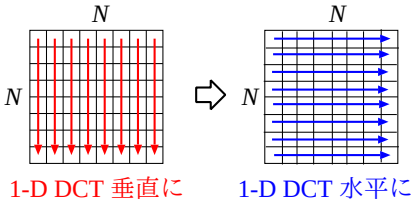
$$F(u,v) = \frac{2C(u)C(v)}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j) \cos\frac{(2i+1)u\pi}{2N} \cos\frac{(2j+1)v\pi}{2N}$$

$$f(i,j) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)F(u,v) \cos\frac{(2i+1)u\pi}{2N} \cos\frac{(2j+1)v\pi}{2N}$$

$$0 \le i, j, u, v \le N-1, \text{ および } C(n) = \begin{cases} 1/\sqrt{2} & (n = 0) \\ 1 & (n \neq 0) \end{cases}$$

## 分離型の利点

- 計算回数； $N^4 \Rightarrow 2N^2$
- $\boldsymbol{C_{xx}}$ ：画像全体で平均 $\Rightarrow$ 平均的に分離型が良い $\Rightarrow$ 局所的にはわからない




1-D DCT 垂直に　　1-D DCT 水平に

## DCT の性質 — なぜ良いか？(1)

$$F(u,v) = \sum_{i=0}^{7}\sum_{j=0}^{7} f(i,j) \underbrace{\frac{2C(u)C(v)}{N} \cos\frac{(2i+1)u\pi}{2N} \cos\frac{(2j+1)v\pi}{2N}}_{\text{2 次元 DCT 基底 } W_{ij}^{uv}}$$

$$F(u,v) = \sum_{i,j=0}^{N-1} f(i,j)\cdot W_{ij}^{uv}, \quad f(i,j) = \sum_{u,v=0}^{N-1} F(i,j)\cdot W_{ij}^{uv} \quad \text{（内積計算）}$$

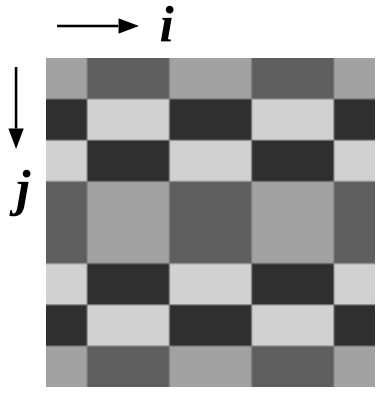
【$u=4$, $v=6$ の $W_{ij}^{46}$ の例】

$\rightarrow i$

$$\begin{bmatrix}
+0.07 & -0.07 & -0.07 & +0.07 & +0.07 & -0.07 & -0.07 & +0.07 \\
-0.16 & +0.16 & +0.16 & -0.16 & -0.16 & +0.16 & +0.16 & -0.16 \\
+0.16 & -0.16 & -0.16 & +0.16 & +0.16 & -0.16 & -0.16 & +0.16 \\
-0.07 & +0.07 & +0.07 & -0.07 & -0.07 & +0.07 & +0.07 & -0.07 \\
-0.07 & +0.07 & +0.07 & -0.07 & -0.07 & +0.07 & +0.07 & -0.07 \\
+0.16 & -0.16 & -0.16 & +0.16 & +0.16 & -0.16 & -0.16 & +0.16 \\
-0.16 & +0.16 & +0.16 & -0.16 & -0.16 & +0.16 & +0.16 & -0.16 \\
+0.07 & -0.07 & -0.07 & +0.07 & +0.07 & -0.07 & -0.07 & +0.07
\end{bmatrix}$$

$\downarrow j$

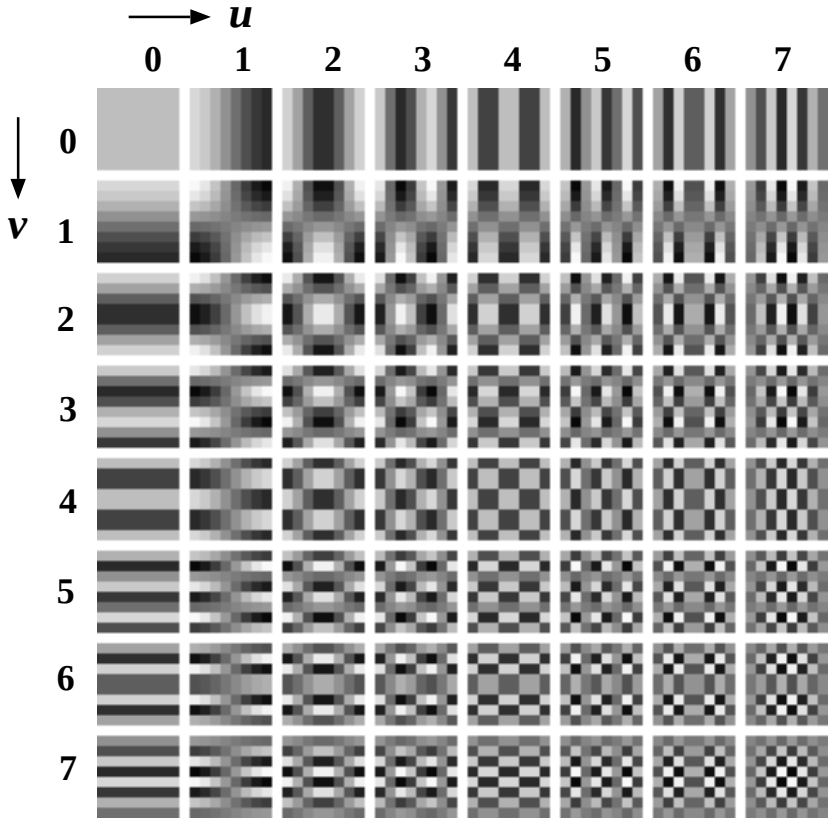
数値表示



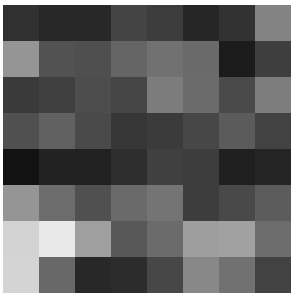

輝度表示



## $N=8$ の場合のすべての基底 $W_{ij}^{uv}$】

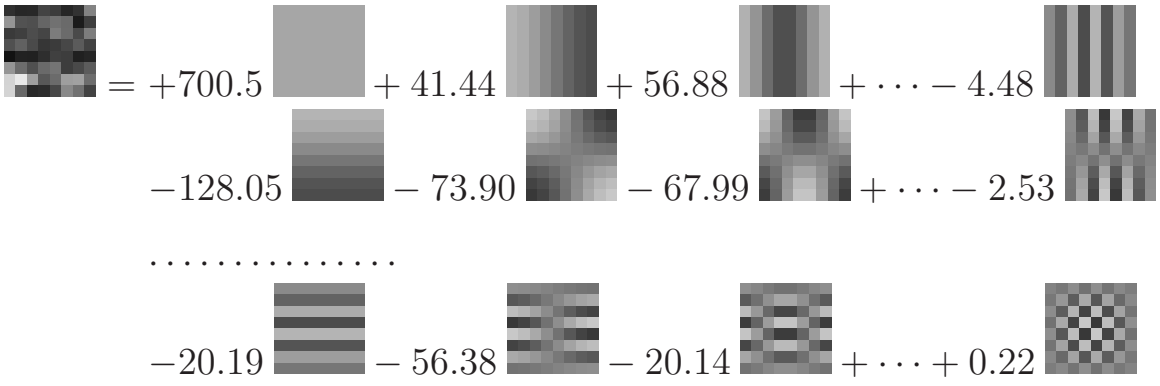



$F(0,0)$ のみ ：**DC 係数**，$F(0,0)$ 以外 ：**AC 係数**



$$F(u,v) = \begin{bmatrix} 700.50 & 41.44 & 56.88 & 70.64 & 25.25 & -26.49 & 11.89 & -4.48 \\ -128.05 & -73.90 & -67.99 & -75.39 & 61.17 & -25.38 & 23.55 & -2.53 \\ 87.34 & 28.06 & 63.91 & 42.95 & 16.48 & 16.06 & 16.39 & -13.64 \\ -39.34 & 8.13 & -15.61 & -57.53 & 31.97 & -3.43 & -20.21 & -1.66 \\ -156.25 & -51.13 & 14.90 & -3.25 & -7.50 & 21.49 & -7.80 & -9.58 \\ 93.47 & -10.39 & 47.00 & -49.29 & -57.52 & -9.78 & -28.01 & 10.11 \\ -52.41 & -63.91 & -2.61 & -20.42 & 56.77 & -2.78 & -2.91 & 11.73 \\ -20.19 & -56.38 & -20.14 & -18.60 & -26.23 & -27.34 & 8.66 & 0.22 \end{bmatrix}$$

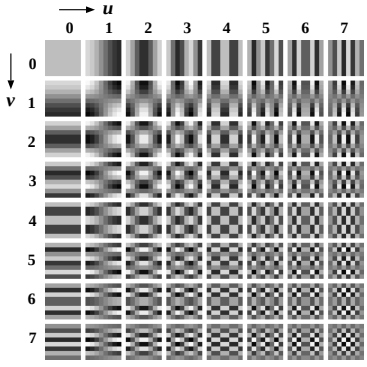
$\Downarrow$




基底 $W_{ij}^{uv}$ の重み和で表現（分解） ⇒ 重み：変換係数



## DCT の性質 — なぜ良いか？(2)




- KLT の性質受継ぐ（誤差最小，直交性）
- 画像依存性なし
- 変換係数 $F(u,v)$ ⇒ ほぼ無相関化
- 基底は周波数（パターンの細かさ）に基づいて構成（cos の簡単な関数）
- 直交変換 ⇒ パーセバルの関係 ⇒ これを用いると ...

  – $\tilde{F}(u,v)$：量子化した変換係数，$\tilde{f}(i,j)$：$\tilde{F}$ を逆変換した画素値

$$\underbrace{\sum_{u,v=0}^{N-1}\left(F(u,v)-\tilde{F}(u,v)\right)^2}_{\text{量子化誤差}} = \underbrace{\sum_{i,j=0}^{N-1}\left(f(i,j)-\tilde{f}(i,j)\right)^2}_{\text{量子化に伴う平均 2 乗誤差}}$$

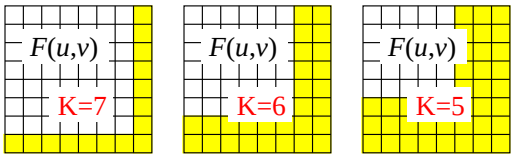
  – つまり，変換係数の誤差がそのまま画素領域の誤差になる

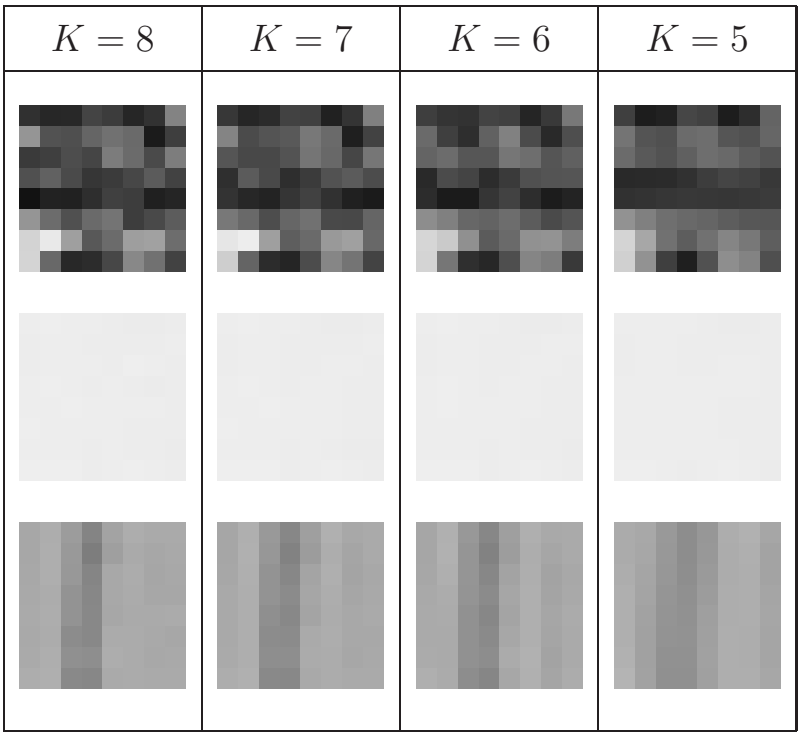
- 人間の視覚特性 ⇒ 細かいパターンに対する感度低い

  ⇒ $u$，$v$ の大きな（細かな）基底を「無視」,「粗く量子化」

## 部分的逆変換 — 細かいパターン（高周波成分）を用いずに逆変換

$$f(i,j) = \sum_{u,v=0}^{K-1} F(u,v)\, W_{ij}^{uv}$$






## 実画像で部分的逆変換すると

DCT $K = 7$ (PSNR=29.30 [dB])

KLT $K = 7$ (PSNR=29.35 [dB])

DCT $K = 5$ (PSNR=23.14 [dB])

KLT $K = 5$ (PSNR=23.23 [dB])

- $u$，$v$の大きな変換係数（高周波成分）を完全に無視するのはひどいので
- 実際には，$u$，$v$が大きくなるにつれて粗く量子化する

## 符号化への応用 (1) —基本方針

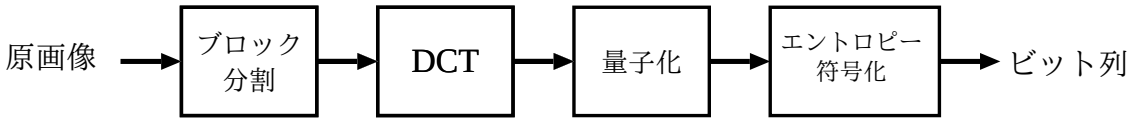
- $N = 8$ 点 DCT を用いる ⇒ 効率，計算量，リンギングによる劣化

$N = 4$ (0.4[bpp], 21.2[dB])

$N = 8$ (0.4[bpp], 21.5[dB])

$N = 32$ (0.4[bpp], 21.1[dB])

- 画像全体を $8 \times 8$ のブロックに分割
- ブロック単位に DCT，変換係数単位に量子化しエントロピー符号化






## DCT 係数の量子化

- 係数 $F(u,v)$ 毎に量子化ステップ幅を定めた量子化テーブルを利用

$$Q(u,v) = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

$$F'(u,v) = \left\lfloor \frac{F(u,v)}{Q(u,v)} + 0.5 \right\rfloor \quad \text{によって一様量子化}$$

画質と符号量は量子化テーブルの値によって制御可能

実際には，パラメータ $q$ を用いて，$Q(u,v)$ を一様に $q$ 倍

【例】

$$f(i,j) = \begin{bmatrix} 167 & 173 & 156 & 132 & 163 & 173 & 170 & 170 \\ 167 & 174 & 153 & 125 & 159 & 170 & 167 & 169 \\ 168 & 174 & 152 & 133 & 168 & 172 & 166 & 168 \\ 169 & 172 & 147 & 135 & 168 & 171 & 167 & 167 \\ 170 & 173 & 147 & 136 & 167 & 170 & 170 & 171 \\ 169 & 172 & 140 & 136 & 171 & 171 & 169 & 167 \\ 170 & 174 & 144 & 142 & 173 & 171 & 168 & 169 \\ 173 & 175 & 137 & 135 & 170 & 171 & 170 & 170 \end{bmatrix}$$

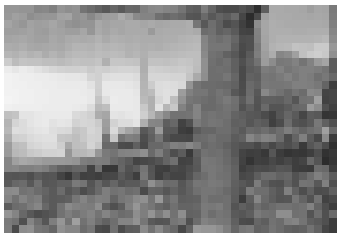
$$F(u,v) = \begin{bmatrix} 1299.50 & -21.87 & 60.31 & 58.40 & -20.25 & -52.35 & -11.18 & 13.72 \\ -3.14 & 2.58 & 3.12 & -9.67 & -13.93 & -0.01 & 11.79 & 5.45 \\ 0.75 & -0.49 & 3.30 & -0.23 & -2.54 & -0.51 & 0.47 & -0.43 \\ 1.12 & -2.13 & -1.29 & -1.65 & -0.41 & 2.50 & -0.30 & 0.07 \\ 1.75 & -1.30 & 2.27 & 0.30 & 0.50 & 2.02 & -0.40 & -0.52 \\ 4.01 & 0.84 & -7.56 & -0.39 & 2.76 & 0.88 & -0.20 & -0.65 \\ 0.50 & -1.17 & -1.53 & 3.30 & 1.05 & -0.72 & -1.30 & 0.03 \\ 5.02 & 0.56 & -1.72 & -3.46 & 0.52 & 0.42 & 0.97 & 0.18 \end{bmatrix}$$



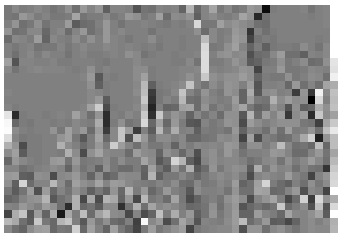
$$F'(u,v) = \begin{bmatrix} 81 & -2 & 6 & 4 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \tilde{F}(u,v) = \begin{bmatrix} 1296 & -22 & 60 & 64 & -24 & -40 & 0 & 0 \\ 0 & 0 & 0 & -19 & -26 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\tilde{f}(i,j) = \begin{bmatrix} 162 & 176 & 155 & 134 & 155 & 176 & 171 & 167 \\ 163 & 175 & 154 & 135 & 156 & 176 & 171 & 167 \\ 166 & 174 & 151 & 135 & 158 & 176 & 170 & 167 \\ 169 & 172 & 148 & 136 & 161 & 176 & 168 & 167 \\ 172 & 169 & 144 & 137 & 164 & 176 & 167 & 167 \\ 175 & 167 & 141 & 137 & 166 & 176 & 166 & 167 \\ 178 & 166 & 139 & 138 & 168 & 176 & 165 & 168 \\ 179 & 165 & 137 & 138 & 169 & 176 & 164 & 168 \end{bmatrix}$$

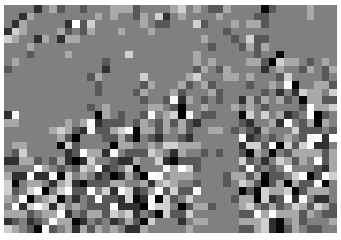
## 量子化後の DCT 係数のエントロピー符号化



$F'(0,0)$



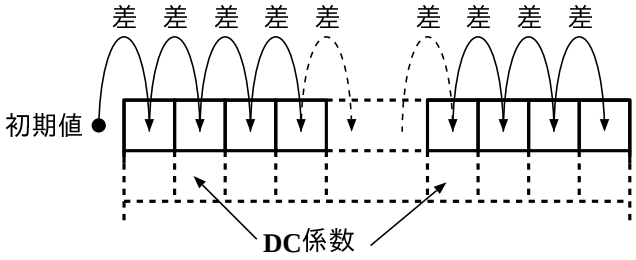
$F'(1,0)$



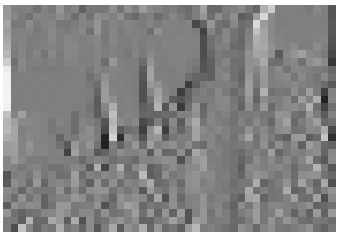
$F'(3,2)$

## DC 係数の符号化 — ブロック間に相関が残っているので差分符号化

- 左隣のブロックとの差分を取り，差分値をエントロピー符号化




DC 係数の差分符号化

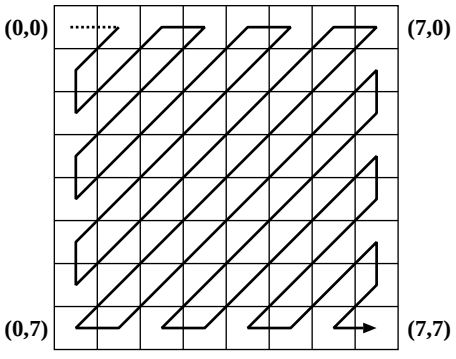


差分値の分布



## AC 係数の符号化 — ブロック内でのみ符号化

- AC 係数をジグザグスキャン




- 絶対値の大きなものから小さなものへと並ぶ傾向
- AC 係数の中にはゼロが多い

  $\cdots$ 非ゼロ係数 0 $\cdots$ 0 非ゼロ係数 0 $\cdots$ 0 非ゼロ係数 $\cdots$

- ラン長（ゼロ並びの数）: ランを切る非ゼロ係数 $\Rightarrow$ エントロピー符号化
- 最後の非ゼロ係数から後がすべてゼロであることを示す $\Rightarrow$ 効率化

  $\Rightarrow$ エンド・オブ・ブロック符号 (EOB)

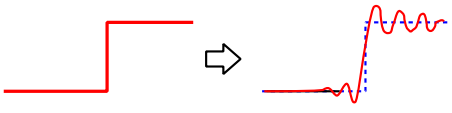
## さまざまな（正規）直交変換

- 最も簡単な（正規）直交基底 ⇒ 自然基底 $[1,0,0]$，$[0,1,0]$，$[0,0,1]$

- （正規）直交変換 ⇒ 対応する正規直交基底で定まる
- どんな複雑な（正規）直交基底，変換も互いに回転したもの

  KLT，DCT，離散アダマール変換 (DHT)，整数 DCT，$\cdots$
- 直交変換 ⇒ 線形変換 $\boldsymbol{y} = \boldsymbol{A}\,\boldsymbol{x}$ ⇐ 2 乗ノルムで誤差評価し最小化
- 2 乗誤差の最小化 ⇒ 量子化等で係数のバランスが崩れると

  ⇒ リンギング，モスキートノイズ



- ブロック直交変換 ⇒ 高圧縮時にブロックノイズが顕著

  ⇒ 重複直交変換，ウェーブレット変換



## 「前半：基礎の基礎」の内容

1. 画像と符号化 — 準備
2. 画像の性質と符号化 — 画素間相関と圧縮
3. エントロピー符号化 — 簡単に
4. 量子化
5. 相関除去法 (1) — 予測による方法
6. 相関除去法 (2) — 変換による方法
7. 動画像の性質と符号化 — 時間方向の相関除去
8. 以上を組み合わせて

# 7. 動画像の性質と符号化 — 時間方向の相関除去

## 動画像とは...

- 動画像：一定時間間隔でサンプリングした静止画像の集合
- フレーム：動画像を構成する 1 枚の画像
- フレームレート：毎秒あたりのフレーム数 [frame/s]
- 第 $n$ フレームを $f_n(i,j)$ と表す

$f_0$ $f_1$ $f_2$ $f_3$

$f_4$ $f_5$ $f_6$ $f_7$



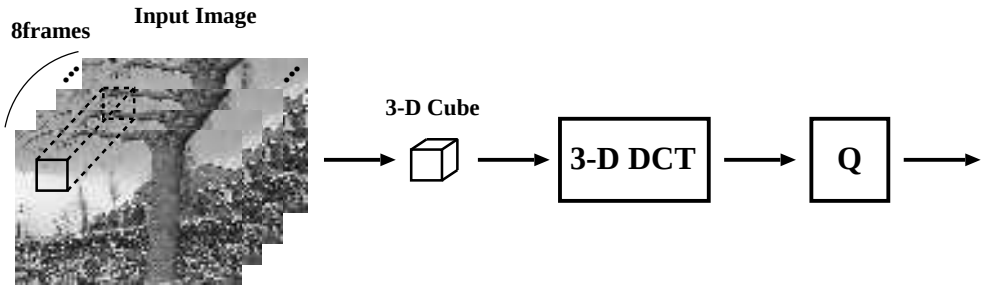
## 動画像符号化の諸方式

- 最も単純な方式 ⇒ フレーム内 (Intra) 符号化（フレーム単位に符号化）

  ⇒ 時間方向にも高い相関 ⇐ 似ているから
  ⇒ フレーム内符号化では時間相関の除去は不可能

- フレーム間符号化の必要性

  例：⇒ $x$–$y$–$t$ の 3 次元直交変換



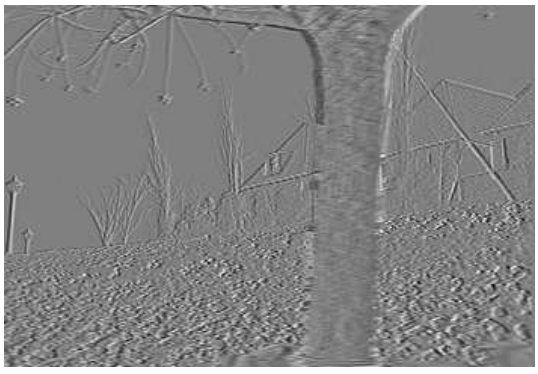

⇒ 動部分と静止部分が混在し，動部分の効率が悪い
⇒ 動きが大きい部分はサンプリング定理を満たさない

⇓

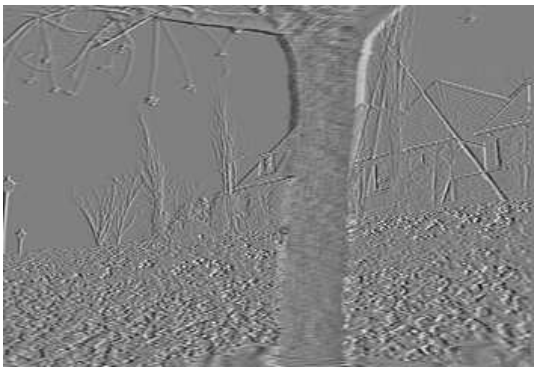
**動き補償予測 (MC) 方式 — 予測符号化を応用**

## フレーム間単純差分 — 準備

$d_n(i,j) = f_n(i,j) - f_{n-1}(i,j)$　前フレーム $f_{n-1}$ を予測値に



$d_1 = f_1 - f_0$



$d_4 = f_4 - f_3$

$\Downarrow$

各部は「動いている」ため，同一撮影対象であっても
差分値 $d_n$ はゼロ近くになるとは限らない



## フレーム差分の改善 — 動き補償の導入

### 仮定

- $f_{n-1}$ に対して適当な幾何変換 $F(\boldsymbol{p}_n,\ \cdot\ )$ を施す $\Rightarrow$
- $f_n$ の予測画像 $f'_n$ を作成可能（$\boldsymbol{p}_n$：変換（動き）パラメータ

$$f'_n = F(\boldsymbol{p}_n,\ f_{n-1}) \simeq f_n \quad \rightarrow \quad d_n = f_n - f'_n = f_n - F(\boldsymbol{p}_n, f_{n-1})$$

$\Downarrow$

- 「予測がうまくいっている」という仮定の下で $d_n \simeq 0$

$\Downarrow$

時間方向の相関が除去可能

---

- $f_{n-1}$ から $f_n$ の予測画像を作る操作 $\Rightarrow$ 動き補償 (MC)
- $f_{(n-1)}$ と $f_n$ から動きパラメータ $\boldsymbol{p}_n$ を抽出する操作 $\Rightarrow$ 動き推定 (ME)

## 実際の ME/MC

- フレームレートはある程度高い ⇒ フレーム間の動き，変形小
  ⇒ 部分的に並行移動として近似可
- 画像内の動きは局所的 ⇒ ローカルな動きにも対応する必要

⇓

1. 並行移動のみを補償対象とする
2. $16 \times 16$ のブロック (MB) 単位に MC を行なう

⇓

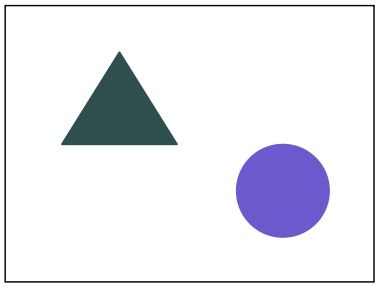
- 動きパラメータ：$16 \times 16$ の MB について動きベクトル (MV) 1 つ

- 符号化対象：対象フレーム，動き補償の元になるフレーム：参照フレーム
- 参照フレームは符号/復号側で同一のフレームを用いる（ドリフト防止）

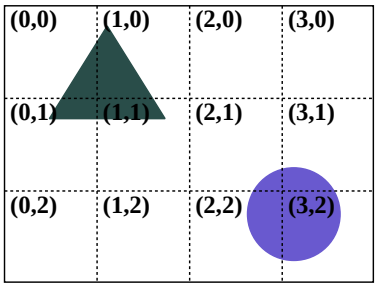


【ME の手順】

1. $f_n$ を重なり合わない MB に分割

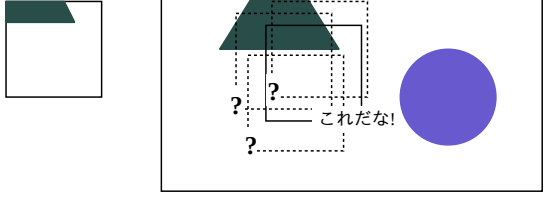


前フレーム $f_{n-1}$

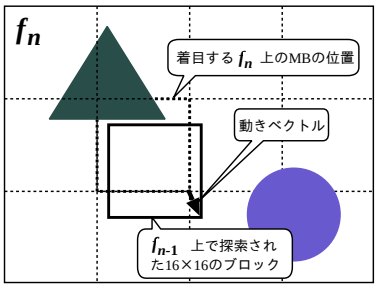



現フレーム $f_n$

2. 各 MB に最も近い $16 \times 16$ ブロックを $f_{n-1}$ 上で探索（ブロックマッチング）



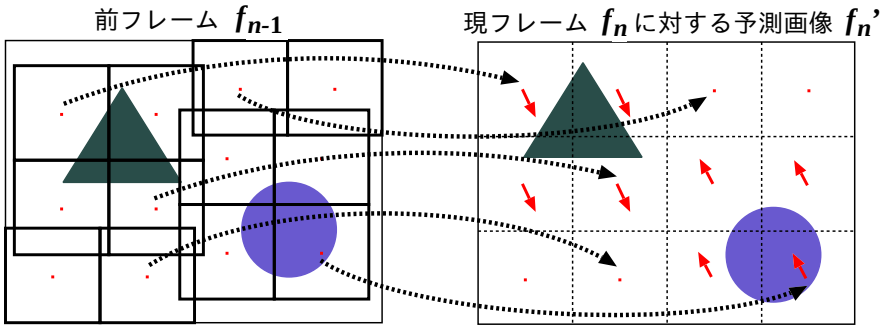

ブロックの探索



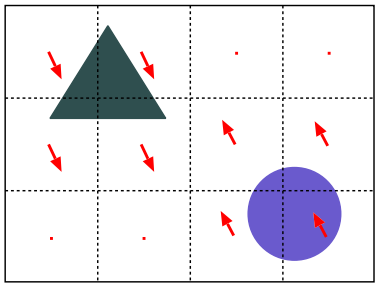

得られる MV

【MC の手順（予測画像と差分画像の生成）】

1. 各 MB の動きベクトル MV の指す $16 \times 16$ ブロックを $f_{n-1}$ 上で切り出す
2. 予測画像 $f'_n$ 上の対応位置に張り付ける




3. 現フレーム $f_n$ と動き補償予測画像 $f'_n$ に対する差分画像 $d_n = f_n - f'_n$ を得る．

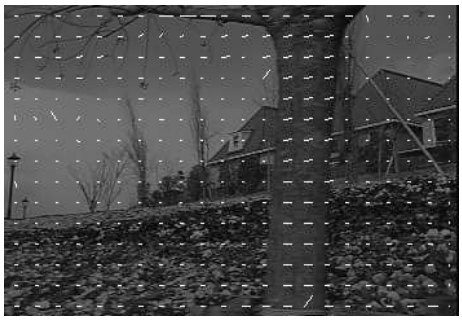


全 MV

差分画像

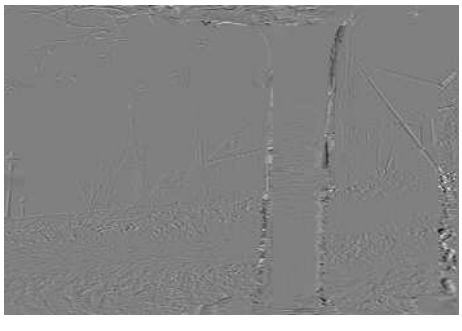


【実画像での例】

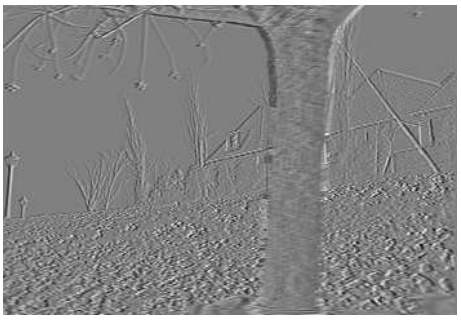


MV

予測画像



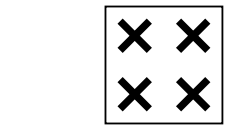
差分画像



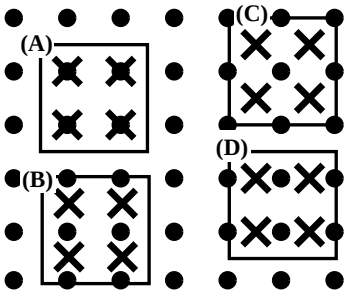
単純差分画像

## ME/MC の精度

- 整数画素精度の動き推定 (Full-pel ME)
- 半画素精度の動き推定 (Half-pel ME) ⇒ 画素補間を用いて 1/2 画素精度まで探索



× --- 対象ブロック上の画素




● --- 参照フレーム上の画素

2 × 2 の対象ブロック　　半画素精度の動き推定

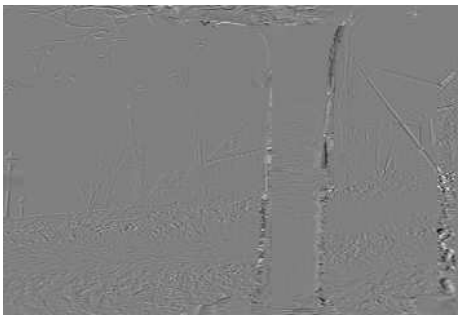
- 画素補間法 — 規格毎に決められている（用いる画素と係数，丸め）

$$f(x,y) = \frac{f(i,j) + f(i+1,j) + f(i,j+1) + f(i+1,j+1)}{4}$$ （2x2 タップの FIR フィルタ（遅延は 1/2 画素）

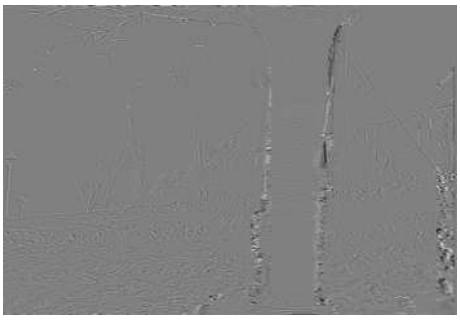
- 1/4 画素精度の動き推定 (Quarter-pel ME) ....
- 基本的に総当たり法で探索 ⇒ さまざまな高速化，高精度化手法が検討されている



【実画像での例】



整数画素精度



半画素精度

## MC で予測誤差の分散 $\sigma_e^2$ はどの程度低減するか？

- 対象画像：平均 $\mu_f$ 分散 $\sigma_f^2$，水平垂直画素間相関 $\rho$ を持つと仮定
- $1/n$ 画素精度の探索 ⇒ 各 MB 内で探索誤差は縦横とも $1/2n$ と仮定

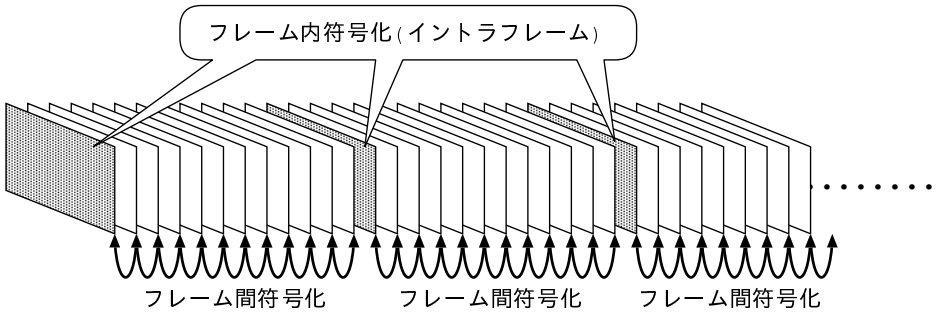
$$\sigma_e^2 = E_B\left[\left(f(x,y) - f(x+\frac{1}{2n}, y+\frac{1}{2n})\right)^2\right]$$

$$= E_B\left[\left((f(x,y)-\mu_f) - (f(x+\frac{1}{2n}, y+\frac{1}{2n})-\mu_f)\right)^2\right] = 2\sigma_f^2(1-\rho^{1/n})$$

- $\rho = 0.96$，半画素精度 $(n=2)$ の場合，$\sigma_e^2/\sigma_f^2 = 2(1-0.96^{1/2}) = 0.04$
- 実際には並進以外の動きや変形 ⇒ 1/4 精度未満での改善は見込めない

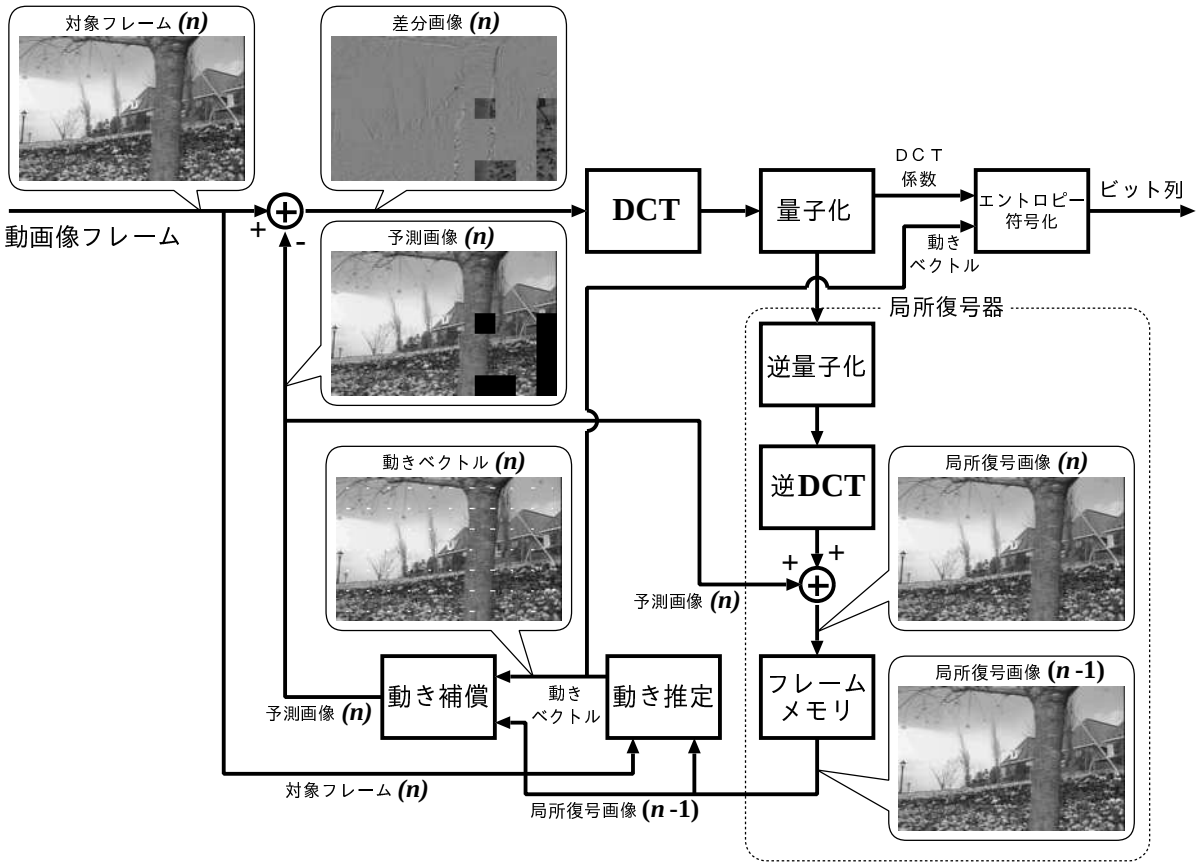
## MC を用いた符号化法の問題点

- 初期フレームには参照フレームが存在しない
  ⇒ 初期フレームのみフレーム内（イントラ）符号化（リフレッシュ）
- 参照フレームが符号側と復号側で異なる ⇒ ドリフト
  ⇒ ローカルデコーダの利用
- 対象フレーム上には ME/MC が不可能な MB が存在
  ⇒ MB 単位でイントラ符号化

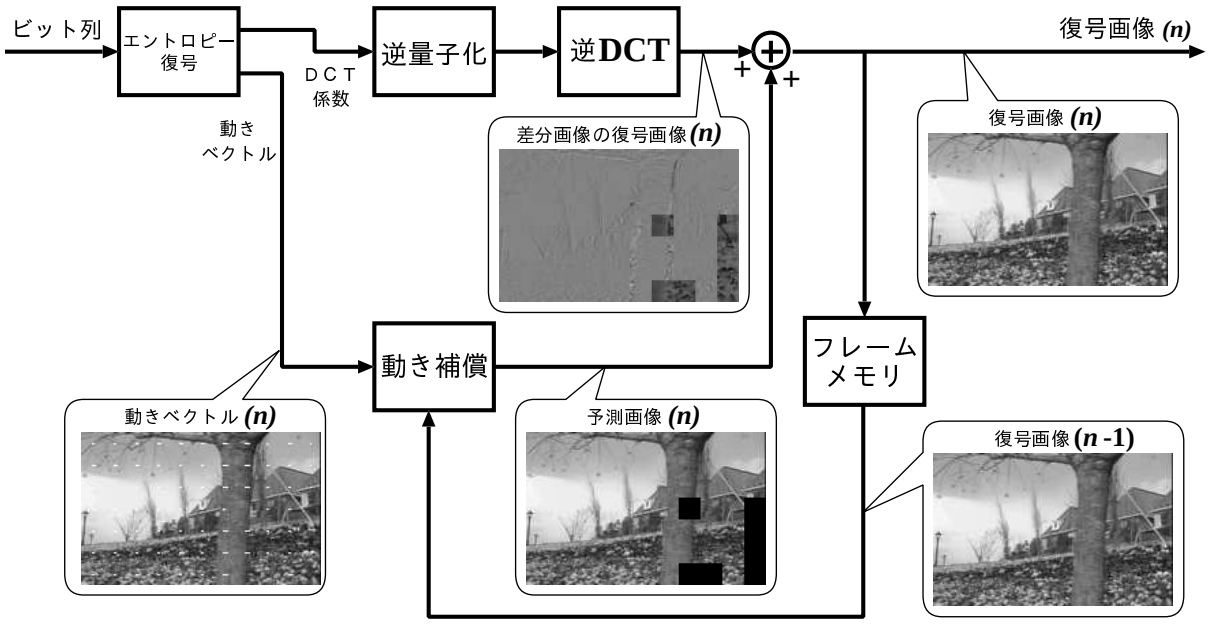





【実際の符号化器の構成】

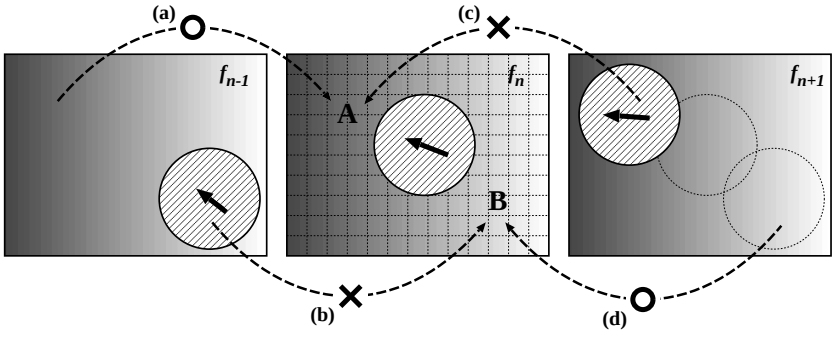


【実際の復号器の構成】






## 逆方向動き補償について




- $f_n$：対象，$f_{n-1}$：参照 $\Rightarrow$ **順方向 MC**
- A 点付近：ME 可能，B 点付近：ME 不可能

$\Downarrow$

### 仮定

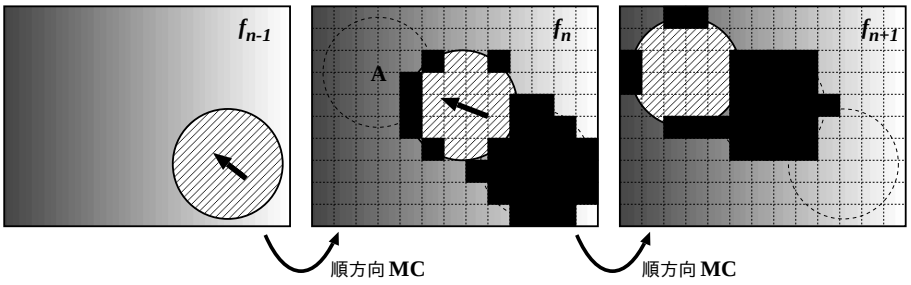
- $f_n$ に対して，過去 $f_{n-1}$，および未来 $f_{n+1}$ の符号化が完了
- 復号側では両フレームを参照可能

$\Downarrow$

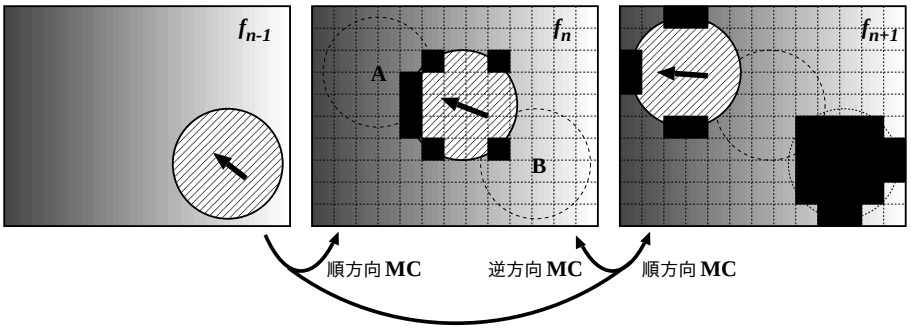
$f_n$：対象，$f_{n+1}$：参照 $\Rightarrow$ **逆方向 MC**

## 双方向動き補償へ

- $f_{n-1}$ を参照フレームとして $f_{n+1}$ の順方向 ME/MC
- $f_n$ は過去 $f_{n-1}$，未来 $f_{n+1}$ の参照可能 $\Rightarrow$ 双方向 MC




順方向 MC のみ（ME に失敗する MB ■ = 58 個）

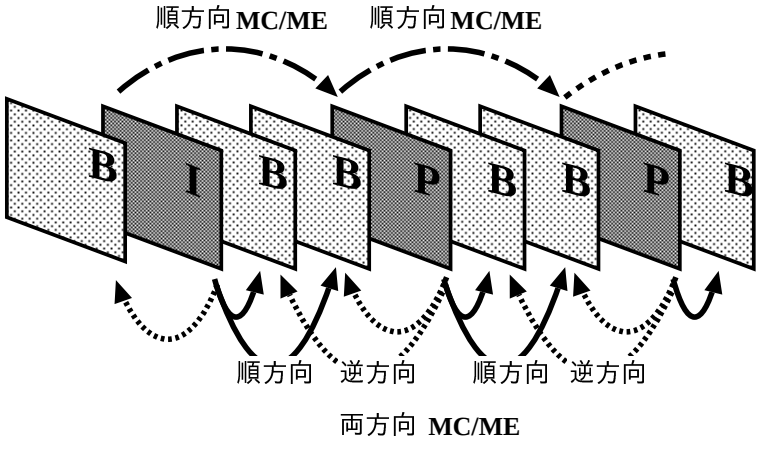



双方向 MC を用いると（ME に失敗する MB ■ = 33 個）



## 実際の双方向動き補償

- **I ピクチャ**(Intra-coded picture)：全体がフレーム内（イントラ）符号化
- **P ピクチャ**(Predictive-coded picture)：順方向 ME/MC のみ
- **B ピクチャ**(Bidirectionally predictive-coded picture)：双方向 ME/MC




- $f_m$ から $f_{m+M}$ へ順方向 MC，その間に $(M-1)$ 枚の B ピクチャ
- $M = 3$ 程度が実用的
- 早送りやランダムアクセス，リフレッシュにも対応．

## MC に基づく動画像符号化の効率向上

- 予測符号化が基本 ⇒ いかに効率的に予測画像を作るか？
  - ブロックサイズを適応的に分割・選択（動き境界等で有利）
  - 参照可能なピクチャを増やす ⇒ 計算量も増大
  - 柔軟な双方向予測
- 各ブロックの MV，参照ピクチャ番号等のサイド情報の圧縮
- 計算量も相応に実現可能である必要



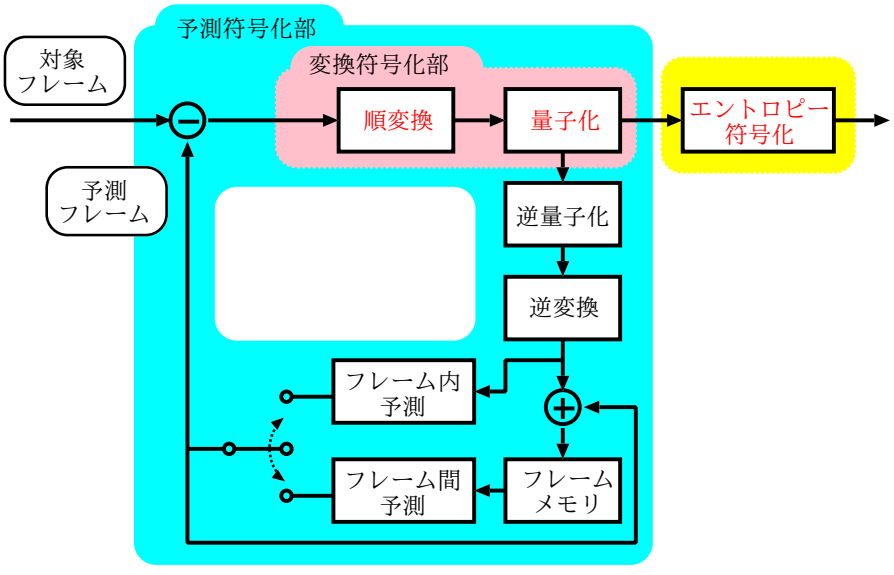
## 「前半：基礎の基礎」の内容

1. 画像と符号化 — 準備
2. 画像の性質と符号化 — 画素間相関と圧縮
3. エントロピー符号化
4. 量子化
5. 相関除去法 (1) — 予測による方法
6. 相関除去法 (2) — 変換による方法
7. 動画像の性質と符号化 — 時間方向の相関除去
8. 以上を組み合わせて

# 8. 以上を組み合わせて

## 前半の内容と現在の動画像符号化法

- エントロピー符号化
- 量子化
- 予測（差分）符号化方法
- 変換符号化法
- 動き補償予測




現在の動画像符号化法