最も簡単な圧縮は、モールス信号

文字	モールス符号 i	発生確率 P_i	長さ L_i
A	・—	0.064	6
B	—・・・	0.013	10
C	—・—・	0.022	12
Space	←符号長→	0.186	1

【二値化】それぞれの文字を「・」か「ー」で置換する

【 圧 縮 】通報全体の符号長を出来るだけ短くしたい

頻出する文字には短い符号を，
さもなければ長い符号を割り当てる．

圧縮の基本　ハフマン符号化

通報＝{ ABC_AAB_AABA } の場合

文字	自然2進数	ハフマン符号	発生頻度
A	00 [2 bit]	1 [1 bit]（短い）	6（頻出）
B	01 [2 bit]	00 [2 bit] ↑	3 ↑
_	11 [2 bit]	010 [3 bit] ↓	2 ↓
C	10 [2 bit]	011 [3 bit]（長い）	1（希少）

それぞれの文字を0と1の符号で置き換える（二値化する）

圧縮前は…

通報＝{ ABC_AAB_AABA } の場合

文字	自然2進数
A	00 [2 bit]
B	01 [2 bit]
_	11 [2 bit]
C	10 [2 bit]

→ 00BC_AAB_AABA
→ 0001C_AAB_AABA
→ 000110_AAB_AABA
→ 00011011AAB_AABA
→ 000110・・・AB_AABA

→ 000110110000011100000100
【24 bit　圧縮前】

圧縮後は…

通報＝{ ABC_AAB_AABA } の場合

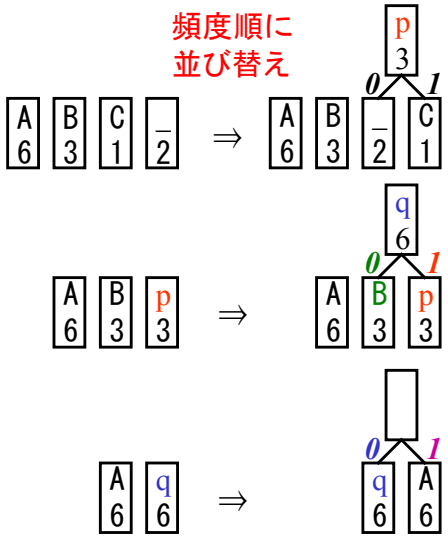
文字	ハフマン符号
A	1 [1 bit]
B	00 [2 bit]
_	010 [3 bit]
C	011 [3 bit]

→ 1BC_AAB_AABA
→ 100C_AAB_AABA
→ 100011_AAB_AABA
→ 100011010AAB_AABA
→ 10001101・・・AB_AABA
→ 1000110101100011011001
【21 bit　圧縮後】

圧縮

→ 000110110000011100000100
【24 bit　圧縮前】

ハフマン符号を 作ってみよう （その1）



A=
B=
C= 1
_= 0

A=
B= 0
C= 11
_= 10

A= 1
B= 00
C= 011
_= 010

各文字の発生頻度が分かれば、圧縮できる。

【問題１】一番圧縮できる通報を選べ.

- 通報①　{ABC_ABC_ABC_}
- 通報②　{AABBBC_AC_C_}
- 通報③　{ABA_ABABCACA}

【問題2】 どのような場合に効率良く圧縮できるか？

【問題3】 通報①は，全く圧縮出来ないか？
次の文字と、「確率的に独立」か？

【問題4】

通報④　{ABC_CAB_ACB_}

- ハフマン符号化を作れ
- 通報④をハフマン符号化せよ。
- 圧縮率を調べよ。

文字	ハフマン符号	発生頻度
A	?	?
B	?	?
_	?	?
C	?	?

圧縮できない場合とは？

通報＝{ ABC_CAB_ACB_ } の場合

圧縮前	00	01	10	11	00	00	01	11	00	00	01	00
圧縮後	10	11	00	01	00	10	11	01	10	00	11	01

どちらも
全　２４　ビット
（平均　２　ビット/文字）

文字	ハフマン符号	発生頻度
A	10	3
B	11	3
_	01	3
C	00	3

圧縮できない!!　→　Why?　→　How?

ハフマン符号を 画像に試してみる

平均符号長

・画素の輝度値は256種類

= 8 [bit/pixel]

= 1 [B /pixel] 1B=8bit

圧縮前は

8 [bpp]

ファイルサイズ

・全部で 256×256 [pixel]

・輝度値は 1 [B/pixel]

= 65536 [B]

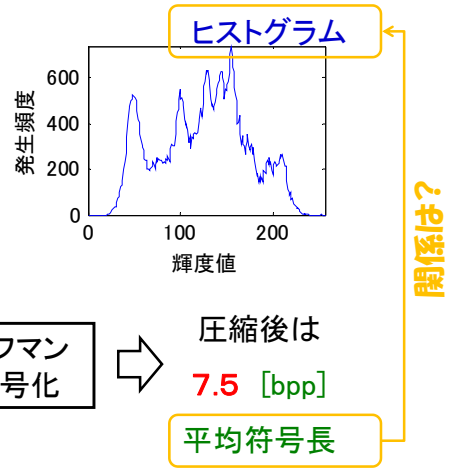
= 64 [KB]

平均符号長

bpp : bit per pixel

ハフマン符号の効果は？

Lena



圧縮前は 8 [bpp] ⇨ ハフマン符号化 ⇨ 圧縮後は 7.5 [bpp]

平均符号長

8.0-7.5= **0.5 [bpp] だけ** (94%に) **圧縮できた!?**

「ハフマン符号で画像圧縮」 ここまでのまとめ

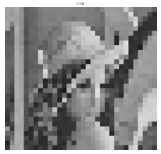
★白黒でもカラーでも、ハフマンでは9割程度にしか圧縮できない。

◎ MPEGやJPEGでは5～10%に圧縮できる。

★画素数を減らせばファイルのサイズは小さくなる (画像サイズも)。

◎ しかし、圧縮率は相変わらず

※ 画質劣化(alias)もある。



★階調を減らせば圧縮できる。

◎ しかし、画質劣化が酷い

さて、どうやって性能を改善するか？

何故、圧縮できるのか？

{ ABC_AAB_AABA }

文字	ハフマン符号	発生頻度
A	1	6
B	00	3
_	010	2
C	011	1

圧縮できる

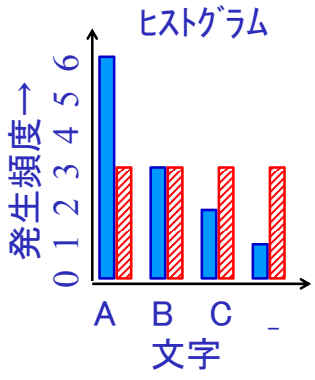
{ ABC_CAB_ACB_ }

文字	ハフマン符号	発生頻度
A	10	3
B	11	3
_	01	3
C	00	3

圧縮できない

発生頻度　＝　ヒストグラム

文字	発生頻度
A	6
B	3
_	2
C	1



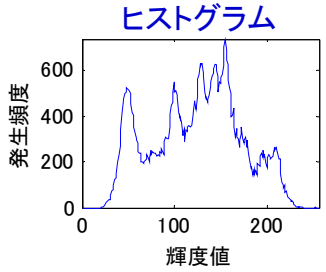
文字	発生頻度
A	3
B	3
_	3
C	3

圧縮できる

圧縮できない

ヒストグラムが　圧縮率を決める

Lena



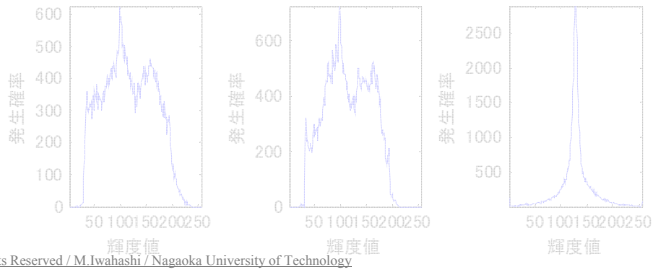
圧縮前は
8 [bpp]
100%

⇨ ハフマン符号化 ⇨

圧縮後は
7.5 [bpp]
94%

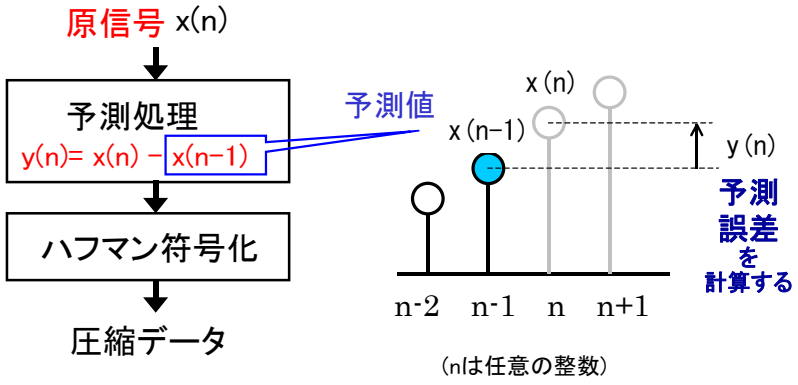
ある処理をすると圧縮率がアップ！

原画像	ぼかし	エンボス
Barbara		
94 %	91%	81%
7.5 bpp	7.3 bpp	6.5 bpp



微分すれば圧縮できる
戻すときは積分する

エンボス ＝ 微分 ＝ 予測



予測の実例　微分と積分で元通り！

原信号

$[x(0)\ x(1)\ x(2)] = [10\ \ 12\ \ 11]$

予測処理（微分）

$y(0) = x(0) - x(-1) = 10 - 0 = 10$

$y(1) = x(1) - x(0) = 12 - 10 = 2$

$y(2) = x(2) - x(1) = 11 - 12 = -1$

予測誤差

再生方法（積分）

$x(0) = y(0) + x(-1) = 10 + 0 = 10$

$x(1) = y(1) + x(0) = 2 + 10 = 12$

$x(2) = y(2) + x(1) = -1 + 12 = 11$

様々な予測（微分）処理を試す

横方向　縦方向　縦横方向

68%　60%　61%

5.4 bpp　**4.8 bpp**　4.9 bpp

微分すれば60%に圧縮できる

積分すれば画質劣化無く戻る

ハフマン符号化だけでは不十分

64 KB （100%）

原画像 x(n)

↓

ハフマン符号化

↓

圧縮データ

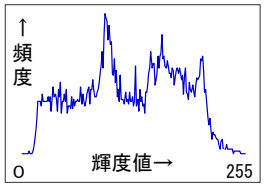
60 KB（94%）

→ m

↓ n

原画像 “Barbara”

x(n,m)



ヒストグラム

微分を併用して効果的に圧縮

64 KB （100%）

原画像 x(n)

↓

微分処理

$y(n) = x(n) - x(n-1)$

↓

ハフマン符号化

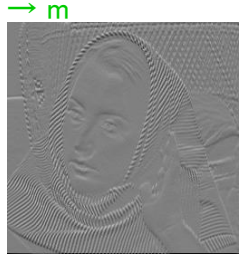
↓

圧縮データ

45 KB（71%）＜ 94%

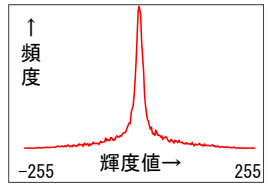
→ m

↓ n



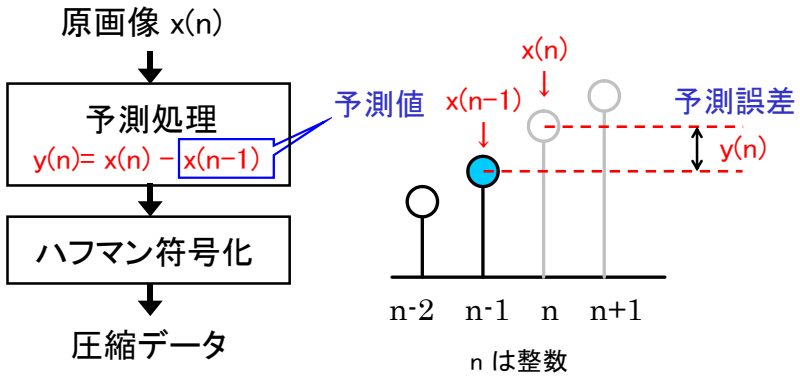
予測誤差 +128

$y(n,m) = x(n,m) - x(n-1,m)$



ヒストグラム

微分 は 予測 （0次の外挿）



予測処理は画像圧縮に効果的！

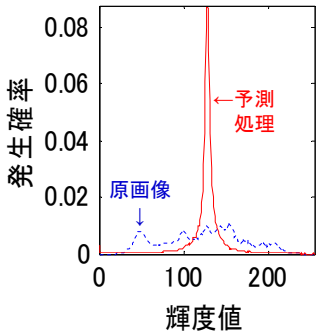
これが Lossless JPEG の基だ！

1位	予測処理＋ハフマン 60% （**4.8** [bpp]）
2位	ハフマンのみ 94% （**7.5** [bpp]）
3位	原画像 100% （ **8** [bpp]）

“Lena”の場合

- **分散が小さい**
- **出現頻度が偏っている**

⇒ **よく圧縮できる**



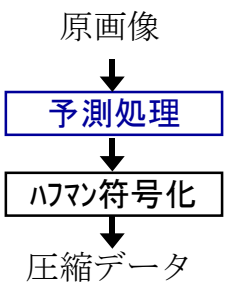
予測とハフマンによる画像圧縮

画像は画素間の相関が強い.

⇒ 予測できる.
⇒ 予測誤差を符号化する.

予測誤差には相関が無い.
しかし，出現頻度は偏っている.

⇒ ハフマン符号化を適用する.



画像圧縮の一般的な処理手順