Lossless Video Coding

Dr. Seishi Takamura
NTT Corporation
Japan
“Lossy” vs. “Lossless”

- **Lossy image/video coding**
  - Input != Output
  - High compression
    (~1/10 - 1/1000)
  - Possibly be “perceptually lossless”

- **Lossless image/video coding**
  - Input == Output
  - Low compression
    (~1/2 - 1/3)
  - “mathematically lossless”
  - This is what’s meant “lossless” in this talk
Is It Worthy to Sacrifice Compression Ratio?

- Yes, for many practical reasons
  - Medical diagnosis, fine art archive, military purposes
  - Image/video analysis, automatic product examination
    The observer is not a human — “perceptually lossless” does not make sense
  - Contribution transmission, studio applications
    Subject to be re-encoded/edited repeatedly — coding error accumulates, if lossy

- and Yes for some emotional reasons
  - A CG artist said, “Just do not lose even one bit from my artwork!”
  - Provides an upper-bound of image quality (can feel at ease)
  - Fed up with drawing R-D curves, evaluating Lagrange cost function, subjective impairment measurement, HVS…
Outline

- Lossless image coding
  - Typical schemes and technology
  - FYI: Lossless image codecs comparison

- Lossless video coding
  - Lossless variants of lossy H.264/AVC
    - Using reversible transforms
    - Using DCT residual packing
  - FYI: Lossless video codecs comparison

- Summary
Outline

- Lossless image coding
  - Typical schemes and technology
  - FYI: Lossless image codecs comparison

- Lossless video coding
  - Lossless variants of lossy H.264/AVC
    - Using reversible transforms
    - Using DCT residual packing
  - FYI: Lossless video codecs comparison

- Summary
# Typical Lossless Image Codecs

<table>
<thead>
<tr>
<th>Method</th>
<th>Algorithm</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIF</td>
<td>—</td>
<td>LZW up to 256 colors</td>
</tr>
<tr>
<td>TIFF</td>
<td>—</td>
<td>LZW</td>
</tr>
<tr>
<td>PNG</td>
<td>Linear prediction</td>
<td>DEFLATE (LZ77+Huffman)</td>
</tr>
<tr>
<td>JPEG*</td>
<td>Linear prediction</td>
<td>Huffman *lossless mode</td>
</tr>
<tr>
<td>JPEG 2000</td>
<td>reversible (5,3) DWT</td>
<td>Arithmetic coding (EBCOT)</td>
</tr>
<tr>
<td>JPEG-LS</td>
<td>Median prediction</td>
<td>context-based Golomb</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(run-length in flat region)</td>
</tr>
<tr>
<td>CALIC</td>
<td>Context-based linear prediction + nonlinear correction</td>
<td>Huffman / Arithmetic coding</td>
</tr>
</tbody>
</table>
Prediction Schemes

- Linear prediction
  - Fixed, encoder-optimized and online-update predictors
  - Using causal pixels (extrapolative DPCM)
  - Using non-causal pixels (interpolative DPCM)
  - Block transform
    - Reversible DWT
    - Reversible DCT
    - R/G/B decorrelation

- Median prediction

- Markov model
  - Build a stochastic model of the image
  - Encode the pixel occurrence probability: \( \text{Prob}(x|c) \)
  - cf. minimum MSE prediction = \( \sum_x (\text{Prob}(x|c) \times x) \)

- Pixelwise residual ("Lossy plus lossless") approach
  - Consider another lossy coder (e.g., JPEG) as a predictor
  - Entropy code the pixel-wise residual

(And motion compensation for video)
Outline

- Lossless image coding
  - Typical schemes and technology
  - FYI: Lossless image codecs comparison

- Lossless video coding
  - Lossless variants of lossy H.264/AVC
    - Using reversible transforms
    - Using DCT residual packing
  - FYI: Lossless video codecs comparison

- Summary
ARTest Codec Comparison (full-color)

Top 14 codecs out of 26
Average of 627 full-color (24-bit) images, weighted by image size (#pixels)
Also provides encoding/decoding time comparison

[The art of lossless data compression, 2002]
MRP — A Gray-Scale Lossless Image Codec

- Context-based 2D-linear prediction
  - $M=20-56$ contexts (predictors)
  - $P=30-72$ pixels from current frame
  - All $M \cdot P$ coefficients are optimized for minimum rate, not for minimum MSE, and downloaded
- Encoding time: 10-25 minutes per frame
- Decoding time: 0.2 seconds per frame

[Matsuda et al., 2005]
Average of 13 gray-scale (8-bit) images
Weighted by image size (#pixels)
CALICa: CALIC with arithmetic coding
CALICh: CALIC with Huffman coding

Note: Conditions are different from previous comparison
Outline

- Lossless image coding
  - Typical schemes and technology
  - FYI: Lossless image codecs comparison

- Lossless video coding
  - Lossless variants of lossy H.264/AVC
    - Using reversible transforms \cite{Takamura, ICASSP2005}
    - Using DCT residual packing
  - FYI: Lossless video codecs comparison

- Summary
- H.264 FRExt skips the transform and quantization

```
Video → Inter/Intra prediction → 4x4/8x8 transform → Quantization → Zigzag scan → Entropy coding
```

- Entropy reduction at each stage:
  - Inter/Intra prediction
  - 4x4/8x8 transform
  - Quantization

- Entropy remains the same through FRExt: Entropy remains the same
Enhancement of FRExt

- Instead of no-operation (FRExt), we apply reversible transform to reduce entropy
- We compared four transforms
Reversible Transforms Tested

- 4x4 Transform of H.264 (reversible if not quantized)
- Piecewise Scaled Transform
- Reversible DCT
- Linear Prediction
  - Non-adaptive version
  - Adaptive version
4x4 Reversible Transform

H.264 4x4 Transform

\[
\begin{bmatrix}
A & B & C & D \\
E & F & G & H \\
I & J & K & L \\
M & N & O & P \\
\end{bmatrix}
= \begin{bmatrix}
1 & 1 & 1 & 1 \\
2 & 1 & -1 & -2 \\
1 & -1 & -1 & 1 \\
1 & -2 & 2 & -1 \\
\end{bmatrix}
\begin{bmatrix}
1 & 2 & 1 & 1 \\
1 & 1 & -1 & -2 \\
1 & -1 & -1 & 2 \\
1 & -2 & 1 & -1 \\
\end{bmatrix}
\]

- Obviously reversible (if not quantized)
- Signal correlation is reduced...
- However, the signal space is expanded
  - 2.66 bit/pixel increment
  - Corrupts coding performance
Decompose H.264’s 4x4 transform into equivalent four 2x2 transforms, and compensate the expansion at each stage.

H.264 transform (decomposed)

\[
\begin{bmatrix}
1 & 1 \\
1 & -1
\end{bmatrix}
\begin{bmatrix}
1 & 1 \\
1 & -1
\end{bmatrix}
\begin{bmatrix}
1 & 1 \\
1 & -1
\end{bmatrix}
\begin{bmatrix}
1 & 1 \\
1 & -1
\end{bmatrix}
\]

Piecewise scaled transform

\[
\begin{bmatrix}
1 & 1 \\
1 & -1
\end{bmatrix}
\begin{bmatrix}
2 & 1 \\
1 & -2
\end{bmatrix}
\begin{bmatrix}
1 & 1 \\
1 & -1
\end{bmatrix}
\begin{bmatrix}
2 & 1 \\
1 & -2
\end{bmatrix}
\]

Output is still reversible!

2 Divide by 2 and round
5 Divide by 5 and round
4x4 Reversible Transform

Reversible DCT

\[ X_0 = R \left( \frac{x_0 + x_1 + x_2 + x_3}{4} \right) \]
\[ X_1 = x_0 - x_3 + R \left( \frac{x_1 - x_2}{C} \right) \]
\[ X_2 = R \left( \frac{x_0 - x_1 - x_2 + x_3}{2} \right) \]
\[ X_3 = R \left( \frac{x_0 - x_3}{C} \right) - x_1 + x_2 \]

where \( R(x) = \left\lfloor x + 0.5 \right\rfloor \)

- This is parallel to
  - H.264’s 4x4 transform, when \( C=2 \)
  - Parallel to mathematical DCT, when \( C = 1 + \sqrt{2} \)
    (recommended by the paper)
- We tried \( C = 1 + \sqrt{2}, 3 \) and 3.5
4x4 Reversible Transform

Linear Prediction (non-adaptive)

- **Horizontal:** Apply following transform to four rows

\[
\begin{align*}
\alpha_H x_0 & \quad \text{pred.} \\
\alpha_H x_1 & \quad \text{pred.} \\
\alpha_H x_2 & \quad \text{pred.}
\end{align*}
\]

\[
\begin{align*}
x_0 & \quad \downarrow \\
R(x_1 - \alpha_H x_0) & \quad \downarrow \\
R(x_2 - \alpha_H x_1) & \quad \downarrow \\
R(x_3 - \alpha_H x_2) & \quad \downarrow \\
x_0 &
\end{align*}
\]

- **Vertical:** transform the columns in the same manner, using \(\alpha_V\)

- \(\alpha_H = \alpha_V = 0.7\) works good for I frames
- \(\alpha_H = \alpha_V = 0.4\) works good for P and B frames
Optimal coefficient $\alpha$ is image dependent

In energy concentration sense, autocorrelation factor gives an optimal $\alpha$. (But not known to the decoder unless explicitly transmitted)

We estimate $\alpha$ from neighboring decoded blocks

$\rho_H, \rho_V$: horizontal and vertical autocorrelation factors in a block

If **horizontally** correlated

If **vertically** correlated

If **not** correlated
## Transform Comparison

<table>
<thead>
<tr>
<th>Transform</th>
<th>Determinant</th>
<th>Norm of each row</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FRExt</strong></td>
<td>1</td>
<td>(1,1,1,1)</td>
</tr>
<tr>
<td>H.264 Transform</td>
<td>40</td>
<td>(2, √10, 2, √10)</td>
</tr>
<tr>
<td>Piecewise Scaled DCT</td>
<td>1</td>
<td>(2, √10/2, 1, √10/10)</td>
</tr>
<tr>
<td>Reversible DCT</td>
<td>1 + 1/C^2</td>
<td>(1/2, √2 + 2C^2 / C, 1, √2 + 2C^2 / C)</td>
</tr>
<tr>
<td>Linear Prediction</td>
<td>1</td>
<td>(1, √1 + α^2, √1 + α^2, √1 + α^2)</td>
</tr>
</tbody>
</table>

- The smaller, the better
- More constant, the better

The table above compares different transforms used in lossless video coding. Each transform is represented by a matrix, and the determinant and the norm of each row are calculated for comparison purposes.

*FRExt* and *H.264 Transform* have the same determinant of 1, indicating they have the same scale factor. However, the norm of each row for H.264 Transform is less constant compared to FRExt.

*Piecewise Scaled DCT* has a determinant of 1, and the norm of each row is more constant compared to the other transforms, making it the better choice for this context.

*Reversible DCT* has a determinant of 1 + 1/C^2, indicating it is reversible and has a more constant norm of each row, making it a suitable choice for lossless video coding.

*Linear Prediction* is non-orthogonal, as indicated by the note in the bottom right corner of the table.
Simulation Conditions

- 4 QCIF sequences and 5 CIF sequences
- I frame performance
  - Encoded 100 frames (all intra)
  - Compared with Motion JPEG 2000 (kakadu codec)
- P frame performance
  - Encoded 100 frames in IPPP… structure
  - Coding performance is obtained from 99 P frames
- B frame performance
  - Encoded 298 frames in IBBPBBP… structure
  - Coding performance is obtained from 198 B frames
Results (average of 9 sequences)

- H.264 trans. w/o quant.
- Piecewise scaled
- Reversible DCT $C = 1 + \sqrt{2}$
- Reversible DCT $C = 3$
- Reversible DCT $C = 3.5$
- FRExt
- Linear pred. non-adaptive
- Linear pred. adaptive
- JPEG 2000

FRExt is not good at intra

Almost no compression
Summary for Reversible Transform Coding

- Compared FRExt with four different reversible transforms
  - Non-orthogonal transform (linear prediction) yields better compression than other orthogonal transforms
  - Adaptive linear prediction works the best
    - 12% better than FRExt (intra)
    - 3% better than FRExt (inter)
    - 2% worse than JPEG 2000

- Note
  Our method deserves yet another ~1% improvement [Sano et al, 2006]
Outline

- Lossless image coding
  - Typical schemes and technology
  - FYI: Lossless image codecs comparison

- Lossless video coding
  - Lossless variants of lossy H.264/AVC
    - Using reversible transforms
    - Using DCT residual packing [Takamura, ICIP2005]
  - FYI: Lossless video codecs comparison

- Summary
Motivation

- H.264/AVC — promising lossy video coding standard
  - About twice better compression than other standards
  - More and more H.264-compliant bit streams will be created and distributed
- What if H.264 bit stream + additional bits = lossless?
  - Useful for many applications
  - Good interoperability with existing H.264 systems
  - Two-layered scalability
- Existing technologies
  - Conventional pixel-wise residual coding: inefficient
  - H.264 High 4:4:4 Profile (FRExt): single-layered, incompatibility
Transform and Quantization

An image (2D array of pixels)

2D mapping

(x, y)

Transform

y = A x + B

Quantization bin

Quantization step size

8-bit border

original

x (and y): 8-bit integer

x

y

A

B

0

255

45 deg. rotated & magnified

v

u

11

11

[45° rotated & magnified]
Recovering Original from Q-Bin

White grid:
Impossible signal

Inverse transform
Pixel-wise residual

No conversion
(simplified FGS)

No conversion
(general FGS)

12 Choices
(optimal)

To specify original(●)…

20 choices

25 choices

A lot of choices

12 Choices
(optimal)
### H.264’s 4x4 Transform

**Transformed coefficients**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>-1</td>
<td>-2</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>-2</td>
<td>2</td>
<td>-1</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\begin{array}{cccc}
  A & B & C & D \\
  E & F & G & H \\
  I & J & K & L \\
  M & N & O & P \\
\end{array}
\end{align*}
\] = \[
\begin{align*}
\begin{array}{cccc}
  1 & 1 & 1 & 1 \\
  2 & 1 & -1 & -2 \\
  1 & -1 & -1 & 1 \\
  1 & -2 & 2 & -1 \\
\end{array}
\end{align*}
\]

**Key Points:**

- Above matrices are all integer-valued
- To have all 16 coefficient values (A to P) = To have exact 4x4 residual values = To have original image!
Scan all the integer grids ($\bullet$, $\bigcirc$, $\cdot$) within 16-dimensional quantization bin 
$\Omega^{16}=(A|A=A_{\text{min}}\ldots A_{\text{max}}, B|B=B_{\text{min}}\ldots B_{\text{max}}, \ldots, P|P=P_{\text{min}}\ldots P_{\text{max}})$ using 16-nested for-loop to obtain:

- **cases** - total number of $\bullet$ (possible candidates)
- **index** - the serial number of $\bigcirc$ (original image)

Encode **index** using $\log_2(\text{cases})$ bits

However...

- Loop count (=volume of $\Omega^{16}$) becomes $800,000,000,000,000,000,000,000$ when QP=12

It takes 24 million YEARS if each loop is done in 1 nsec, only for one 4x4 block!
H.264 Basis Functions

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P
Nifty Properties of H.264 Coefficients

- For arbitrary input, $A+C$ is always even
  - One bit reduction of $C$, if $A$ is known (can be ‘pack’ed half)
  - So are $A+I$, $B+J$, $E+G$, $M+O$ etc.
- $A+C+I+K$ is always a multiple of four.
More Nifty Properties

- **B+(C>>1)+(A>>1):** Always even
  - If A and C are known, B can be discretized with interval of 2.
  - Same for F+(E>>1)+(G>>1), E+(A>>1)+(I>>1), G+(C>>1)+(K>>1), etc.

- **B–J+E–G:** Multiple of 4

- **2.5(A+C)+2B+D:** Multiple of 10

- **2.5(A+C+I+K)+2(E+G)+M+O:** Multiple of 20

- **6.25(A+C+I+K)+5(B+E+G+J)+4F+2.5(D+L+M+O)+2(H+N)+P:** Multiple of 100
1. Entropy-code the coefficient $A$ using $\log_2(A_{\text{max}} - A_{\text{min}} + 1)$ bits
2. ‘Pack’ each of seven coefficients and entropy-code in the same manner
3. For-loop the rest eight coefficients ($\Omega^8$) to find out index and cases
4. Entropy-code index using $\log_2(\text{cases})$ bits

At 2 and 3, numerical properties are exploited

Loop count=400 (2·10^{18} times speed-up!)
Quantization and Bounds

Coefficients $A \ldots P$

Quantization

Levels $level_A \ldots level_P$

Estimation

Bounds

$A_{\min, A_{\max}}$
$B_{\min, B_{\max}}$

$P_{\min, P_{\max}}$

level$_A = (A*\alpha + \beta) >> \gamma$

$x = level_A << \gamma$

$y = x + (1 << \gamma) - 1$

$A_{\min} = (x - \beta + \alpha - 1) / \alpha$

$A_{\max} = (y - \beta) / \alpha$

Then

$A_{\min} <= A <= A_{\max}$

The bounds can be calculated/shared by both encoder and decoder.
Coding Simulation with JM-BL

- Implemented on JM8.3 (H.264 reference codec)

- Two mode selection strategies regarding the base layer (BL) and enhancement layer (EL):
  1. **OverallOpt**: chooses a mode that minimizes \(\text{total}\) (BL + EL) bit amount
  2. **BaseOpt**: chooses a mode that *RD-optimizes* the BL, *no care for the EL*

- Compared the performance with:
  - Motion JPEG 2000
  - FRExt-based lossless coding
  - Pixelwise residual entropy
Rate Trade-Off

Total (EL+BL)

OverallOpt  BaseOpt

Better

Rate (% of uncompressed)

Pixel-wise residual entropy

EL

BL

Silent, IPP

QP(P)
Performance Comparison

- Motion JPEG2000
- OverallOpt (EL/BL)
- BaseOpt (EL/BL)
- FRext

Rate (% of uncompressed)
Coding Simulation with JSVM-BL

- Implemented on JSVM 4 (H.264 scalable extension reference codec)
- 14 test sequences (QCIF/CIF/4CIF, 15/30/60Hz)
- BL structures: III..., IPP..., IBP... and IBBBP...
- Other settings
  - FastSearch: used
  - ME search range: 96
  - Reference frame #: 1
  - 8x8 transform: disabled
- Compared with:
  - Ordinary coding of Motion JPEG2000 (lossless)
  - Residual coding of Motion JPEG2000 (lossless) for EL
  - Residual entropy
Bit Rate Evaluation

Original video → JSVM encoder → JSVM decoder → Arithmetic coder

Proposed BL (III, IPP, etc.)

Pixel-wise residual video

Residual packer

Entropy measurement

Motion JPEG 2000 Lossless Coder

“MJ2K (total)”

“EL Entropy”

Motion JPEG 2000 Lossless Coder

“MJ2K EL”
Bit Rate Breakout (CITY 4CIF, BL=III/IPP)

- **Total (III)**
- **Base (III)**
- **Enhance (III)**
- **Total (IPP)**
- **Base (IPP)**
- **Enhance (IPP)**

- **EL**
- **BL**
- **EL+BL**

Motion JPEG2000

Rate (% of uncompressed) vs. QP

- QP: 3, 5, 7, 9, 11, 13
- Rate (% of uncompressed): 0, 10, 20, 30, 40, 50, 60
EL Bit Rate (BL=III...)

rate (% of uncompressed)

- --- MJ2K
- --- Entropy
- --- Proposed

Bit rate ratio (%)

0 5 10 15 20 25 30

3 5 7 9 11 13 QPi
EL Bit Rate (BL=IPP...)

- MJ2K
- Entropy
- Proposed

Rate (% of uncompressed)

Qp

3 5 7 9 11 13
Intra BL compression ratio is 3 points worse than Motion JPEG2000.

Inter BL is 7 points better than Motion JPEG2000.
~1/3 bits are spent for EL, regardless of BL GOP structure and sequence
Summary of Residual Packing

- Scalable lossless coding having H.264-compliant BL
  - Drastic speed-up without losing efficiency
  - 20% less bits than conventional pixel-wise residual coding
  - Intra BL coding (III)
    - ~55% compression (compared to original)
    - ~3 points worse than Motion JPEG2000
    - 1-3% worse than Motion JPEG2000
    - 14% less bits than FRExt
  - Inter BL coding (IPP, IBP, IBBBP)
    - ~44% compression (compared to original)
    - 5~7 points better than Motion JPEG2000
    - 14% more bits than FRExt
- BL can be safely RD-Optimized (i.e., BaseOpt works good)
- Optimal EL amount: ~1/3 of total bit rate
Outline

- Lossless image coding
  - Typical schemes and technology
  - FYI: Lossless image codecs comparison

- Lossless video coding
  - Lossless variants of lossy H.264/AVC
    - Using reversible transforms
    - Using DCT residual packing
  - FYI: Lossless video codecs comparison

- Summary
VBS — A Gray-Scale Lossless Video Codec

- Video extension of MRP (see page 10)
- Variable block-sized motion compensation
  - Quad-tree based, 32x32 to 8x8 sizes
  - Integer-pel accuracy
- Context-based 3D-linear prediction
  - M=24 contexts (predictors)
  - P=20 pixels from current frame
  - Q=25 pixels from previous frame
  - All M(P+Q) coefficients are optimized for minimum rate, not for minimum MSE, and downloaded for each frame
- Encoding time: 3-4 minutes per frame
- Decoding time: 0.02 seconds per frame

[Shiodera et al., 2006]
SUT Codec Comparison (gray-scale)

Average of 6 gray-scale (8-bit) video sequences (all CIF, 25 frames)

[Shiodera et al., 2006]
Average of 9 YUV 4:2:0 (YV12) video sequences
Also provides encoding time, RGB&YUV4:2:2 coding comparison
Weighted by image size (#pixels, not by #frames)
Outline

- Lossless image coding
  - Typical schemes and technology
  - FYI: Lossless image codecs comparison

- Lossless video coding
  - Lossless variants of lossy H.264/AVC
    - Using reversible transforms
    - Using DCT residual packing
  - FYI: Lossless video codecs comparison

- Summary
Summary

- Presented lossless coding technologies
  - H.264 variants: FRExt extension and Scalable coding
  - Introduced MRP, VBS and other state-of-the-art lossless codec comparisons (only quantitative comparison, since the conditions are not the same)
- Lossless image compression
  - About 1/2 for gray image
  - About 1/3 for RGB24 image
- Lossless video compression
  - About 1/2.9 for both gray and YUV4:2:0 video
Last But Not Least…

- **Drawbacks**
  - High data rate
  - No rate control possible
  - Difficult to enclose users (re-encoding with another codec is not a big deal)

- **Merits**
  - No need to locally-decode the signal
  - Can predict blocks/frames in parallel
  - No subjective viewing necessary
  - Independent of monitors, viewers and viewing conditions
References

- **MRP**

- **The art of Lossless Data Compression**
  [http://artest1.tripod.com/graphics25.html](http://artest1.tripod.com/graphics25.html)

- **Reversible DCT**

- **H.264 + reversible transform (and its follower)**

- **H.264 + residual packing**

- **VBS**

- **MSU codecs comparison 2007**
Thank you