Content-Based Adaptive Binary Arithmetic Coding (CABAC)

Li Li
2017/2/9
Self-introduction

• Name: Li Li          Email: lil1@umkc.edu
• Education
  – 2007-2011       Bachelor USTC
  – 2011-2016       PhD          USTC Houqiang Li
  – 2016-           Postdoc UMKC Zhu Li
• Research interests
  – Video coding
    • Rate control for High Efficiency Video Coding and its extension
    • Affine motion model for Future Video Coding
  – Light field image compression
  – 360-degree video compression
Outline

• **Basic framework of CABAC**
  – Binarization
  – Context modeling
  – Arithmetic coding

• **Residue coding**
  – H.264/AVC
  – H.265/HEVC
Basic framework of CABAC (1)

- CABAC: Content-Based Adaptive Binary Arithmetic Coding
  - Content-Based Adaptive: Adjust the possibility of each bin according to the context or previous coded bins
  - Binary Arithmetic Coding: Arithmetic Coding based on 0 and 1
Basic framework of CABAC (2)

- Step 1: Binarization
- Step 2: Context model selection
- Step 3: Regular/Bypass coding engine
- Step 4: Context model update
Basic framework of CABAC (3)

- Step 1: Context model selection
- Step 2: Arithmetic decoding
- Step 3: Context model updating
- Step 4: De-binarization
Outline

• Basic framework of CABAC
  – Binarization
  – Context modeling
  – Arithmetic coding

• Residue coding
  – H.264/AVC
  – H.265/HEVC
Binarization (1)

• Why Binarization? (One premise and two reasons)
  – Binarization is a lossless operation
  – Adaptive $m$-order ($m>2$) arithmetic coding is complex operation requiring at least two multiplications for each symbol to encode while binary arithmetic coding can be performed in a multiplication-free way
  – Binarization enables context modeling on a sub-symbol level
    • MSB uses context model
    • LSB uses bypass coding mode (no context)
Binarization (2)

• Basic binarization schemes (1)
  – Unary Binarization
    
    | n | Codeword |
    |---|---------|
    | 0 | 0       |
    | 1 | 10      |
    | 2 | 110     |
    | 3 | 1110    |
    | 4 | 11110   |
    | 5 | 111110  |

  – Truncated Unary (TU) Binarization
    • If the max number $S$ is 5 then 5 can be binarized as 11111, the 0 in the last is not needed
Binarization (3)

• Basic binarization schemes (2)
  – Fix Length (FL) binarization
    • N is to the Lth power of 2: log_2 N
    • N is not to the Lth power of 2: example

  – Kth order Exp-Golomb Binarization (EGB)
    • A combination of unary binarization and fix length binarization
    • Refer to Prof. Li slides for more information
Binarization (4)

• Concatenation of basic binarization scheme
  – Coded block pattern: 4-bit FL prefix and a TU suffix with max number S equal to 2
  – TU prefix and Kth order Exp-Golomb

Binarization

• Absolute motion vector difference: TU prefix with max value of 9 and 3-order Exp-Golomb

• Transform coefficient levels: TU prefix with max value of 14 and 0-order Exp-Golomb
Binarization (5)

- After binarization, all the symbols are converted into bin string
  - Reference index 3: 111 (TU)
- For the binary symbol, the binarization process is ignored (skip flag)
- Then for each bin, we will choose a suitable context model and finish the arithmetic coding
Outline

• Basic framework of CABAC
  – Binarization
  – Context modeling
  – Arithmetic coding

• Residue coding
  – H.264/AVC
  – H.265/HEVC
Context modeling (1)

• What is context modeling
  – Context modeling is used to estimate an accurate possibility of each bin
  – Context
    • The coded results of the previous blocks
    • The coded results of the previous bins
    • The position of the current block
  – Different context models mean different possibilities for various bins
Context modeling (2)

• Context modeling type (1)
  – Just one context model to represent the possibility

• Context modeling type (2)
  – the neighboring element to the left and on top of the current syntax element
  – Examples for Skip flag
Context modeling (3)

• Context modeling type (3)
  – The values of prior coded bins are used for the choice of a model for a given bin

• Context modeling type (4)
  – The position in the scanning path
  – The accumulated number of encoded levels
Context modeling (4)

• Context model initialization
  – Based on two parameters (Slice type and Quantization parameters)
  – The initialized context model is based on some training data
  – According to the AVS2 (audio video standard), the initialized context model will not lead to significant BD-rate difference
Outline

• Basic framework of CABAC
  – Binarization
  – Context modeling
  – Arithmetic coding

• Residue coding
  – H.264/AVC
  – H.265/HEVC
Arithmetic Coding (1)

- Arithmetic coding
  - Update the Low Bound and Range

\[
\begin{align*}
  \text{HIGH} & \leftarrow \text{LOW} + \text{RANGE} \times \text{CDF}(n) \\
  \text{LOW} & \leftarrow \text{LOW} + \text{RANGE} \times \text{CDF}(n-1)
\end{align*}
\]

LOW = 0, HIGH = 1

LOW = 0, HIGH = 0.6

LOW = 0.36, HIGH = 0.6

LOW = 0.504, HIGH = 0.6

LOW = 0.504, HIGH = 0.5616

Prob(0)=0.6. Sequence: 0110
Arithmetic Coding (2)

• Multiplication operation

\[ R_{LPS} = R \cdot p_{LPS} \]

• Multiplication free for easy implementation
  – R is quantized with 4 values only when calculating the \( R_{LPS} \)
  – \( p_{LPS} \) is quantized with 64 values
  – A look-up table with 64 by 4 entries will be used to replace the multiplication operation
Arithmetic Coding (3)

- $R$ is quantized with 4 values
  - $2^8 \leq R < 2^9$ (corresponding to $[0.5, 1]$)
  - The 4 parts: $[256, 320]$, $[320, 384]$, $[384, 448]$, $[448, 512]$ ($\rho=0, 1, 2, 3$)
  - Quantization: replaced using the median value of the zone: $288, 352, 416, 480$

- $P_{LPS}$ is quantized using 64 values
  - $P_{LPS}$ is within $[0.01875, 0.5]$
  - $P_{\sigma} = \alpha \times P_{\sigma-1}$ ($\alpha$ is approximated as 0.95)

$$\alpha = \left(\frac{0.01875}{0.5}\right)^{1/63}$$
Arithmetic Coding (4)

- Probability updating

\[ p_{\text{new}} = \begin{cases} 
\max(\alpha \cdot p_{\text{old}}, p_{62}), & \text{if a MPS occurs} \\
\alpha \cdot p_{\text{old}} + (1 - \alpha), & \text{if a LPS occurs} 
\end{cases} \]
Arithmetic Coding (4)

- Encoding process
Arithmetic Coding (5)

• Encoding Process
  – Calculate $R_{LPS}$ according to the look-up table
  – If the current bin is MPS, update $R$ and possibility model
  – If the current bin is LPS, update $R$, Lower Bound, and possibility model
  – Renorm Process is used to make sure $R$ is in a suitable range ($2^8 \leq R < 2^9$)
    • Refer to the three scaling operations in the slides provided by Prof. Li
    • E1, E2, E3 conditions
Arithmetic Coding (6)

- Need E1 scaling
- Need E2 scaling
- Need E3 scaling
- No scaling is required. Continue to encode/decode the next symbol.
Arithmetic coding (7)

- Bypass coding (No need to update context model)
  - Both the possibilities of LPS and MPS are 0.5
  - Double decision threshold

```plaintext
L = L << 1

Yes

binVal != 0

L = L + R

No

RenormE
(one loop using doubled decision thresholds; no doubling of L and R)

Done
```
Outline

• Basic framework of CABAC
  – Binarization
  – Context modeling
  – Arithmetic coding

• Residue coding
  – H.264/AVC
  – H.265/HEVC
Residue coding for H.264/AVC (1)

• Scan process to convert the block from 2D to 1D: Zig-Zag scan

• Significance map
  – Significant coefficient flag
  – Last significant coefficient flag

• Level information
  – Absolute level minus 1
  – Coefficient sign flag
Residue coding for H.264/AVC (2)

- Examples (4x4 block)

```
<table>
<thead>
<tr>
<th>9</th>
<th>0</th>
<th>0</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

Zig-zag Scan

<table>
<thead>
<tr>
<th>significant_coeff_flag</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>last_siginificant_coeff_flag</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>coeff_abs_level_minus1</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>coeff_sign_flag</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Signaling order

- Significant map
- Coefficient level and sign
Outline

• Basic framework of CABAC
  – Binarization
  – Context modeling
  – Arithmetic coding

• Residue coding
  – H.264/AVC
  – H.265/HEVC
Residue coding for H.265/AVC (1)

- For larger blocks such as 8x8, 16x16, 32x32
  - Divide into 4x4 sub-blocks (Coefficient group)
  - For the inter blocks, intra 16x16 and 32x32, Diagonal scan is used
  - For the intra 8x8 blocks, mode dependent scan is chosen from diagonal, horizontal, and vertical

![Diagonal, Horizontal, Vertical Scans](image)
Residue coding for H.265/HEVC (2)

- 2D-1D: mostly diagonal scan (less data dependency compared with the zig-zag)

- Significance map
  - Last significant coefficient position
  - Significant coefficient flag

- Level information
  - Absolute coefficient level greater than 1 flag (8)
  - Absolute coefficient level greater than 2 flag (1)
  - Coefficient sign flag
  - Absolute coefficient remaining
Residue coding for H.265/HEVC (3)

- **Examples (4x4 block)**

![Diagonal Scan](image)

<table>
<thead>
<tr>
<th>last_significant_coeff_x</th>
<th>last_significant_coeff_y</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>significant_coeff_flag</th>
<th>1 0 1 0 0 0 1 0 1 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>coeff_abs_level_greater1_flag</td>
<td>0 0 1 1 1</td>
</tr>
<tr>
<td>coeff_abs_level_greater2_flag</td>
<td>1</td>
</tr>
<tr>
<td>coeff_abs_level_remaining</td>
<td>0 3 7</td>
</tr>
<tr>
<td>coeff_sign_flag</td>
<td>0 1 0 0 1 0</td>
</tr>
</tbody>
</table>

**Context coded bins**

| 3 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |

**Bypass coded bins**

| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 3 | 7 |
Residue coding for H.265/HEVC (4)

- Sign takes about 10% to 20% of the total bits
- Sign data hiding
  - The sign of the last zero coefficients in each 4x4 block (Coefficient group) is simply omitted when using sign data hiding
  - The sign value is embedded in the parity of the sum of the levels of the 4x4 block using a predefined convention
    - Even corresponds to “+”
    - Odd corresponds to “-”
    - Determined in the encoder side (RDO, change coefficients)
Reference


4. HEVC reference software. 
   https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/
Thank you!