|  |  |
| --- | --- |
| **Joint Collaborative Team on 3D Video Coding Extension Development**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  3rd Meeting: Geneva, CH, 17–23 Jan. 2013 | Document: JCT3V-C0142 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **3D-CE6.h related:** **An efficient coding method for DLT in 3DVC** | | |
| *Status:* | Input Document | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Kai Zhang1, Jicheng An1, Shawmin Lei2  1 MediaTek Beijing North Building 10F, Raycom Infotech Park Tower C, No. 2 Kexueyuan South Rd., Haidian District, Beijing, China 100190  2 MediaTek Hsinchu No. 1, Dusing Rd. 1, Hsinchu Science Park, Hsinchu, Taiwan 30078 | Tel: Email: | Shawmin Lei +886-3-5670766 ext. 25555 {kai.zhang, jicheng.an, shawmin.lei}@mediatek.com |
| *Source:* | MediaTek Inc. | | |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

In the current HTM, a depth lookup table (DLT) is adopted to improve the intra coding efficiency for the depth map. All values in the DLT are coded with exp-Golomb codes, which take more than 65% of data in the sequence parameter set (SPS) averagely. This contribution proposes an efficient coding method for the DLT in HTM. By applying a range constrained bit map (RCBM) coding method, the average number of bits generated by DLT is reduced from 557.33 to 195.25, and the number of bits in the corresponding SPS is reduced from 852.50 to 490.42.

# Introduction

Simplified depth coding (SDC) and a depth lookup table (DLT) are adopted into HTM [1] [2]. For each depth coding unit (CU), if SDC is selected, one of four different prediction modes can be selected. After the prediction, the residual is not coded using transform. As the SDC prediction stage always results in one or two depth segments per coded block, a single residual DC depth value is coded for each of these segments. Moreover, the DLT is used to map coded depth values in SDC to valid depth values of the original depth map. The DLT is constructed based on an initial analysis of the input depth map and is then coded in a sequence parameter set (SPS).

DLT is an optional coding tool. In the current HTM, the encoder will not use DLT if more than half the values from 0 to MAX\_DEPTH\_VALUE (*e.g.*, 255 for 8-bit depth samples) appear in the original depth map at the analysis step. Otherwise, DLT will be coded in SPS. In order to code DLT, the number of valid depth values is coded with Exp-Golomb code first. Then each valid depth value is also coded with an exp-Golomb code as depicted in Table 1.

Exp-Golomb codes are efficient when shorter codes possess higher probabilities of occurrence. However, that is not the case for valid depth values and the number of valid depth values. As a result, exp-Golomb codes usually generate long code lengths, and thus DLT requires a large amount of bits to represent. In the common test condition, four sequences *Balloons*, *Kendo*, *Newspapercc*, and *PoznanHall* are determined to use DLT, while the other three are not. Statistics are done on the four sequences. As depicted in Table 2, DLT requires 557.33 bits on average and accounts for more than 65% bits of SPS in depth components. In other words, the DLT coding makes SPS in depth components much larger than the one without DLT. Under the HTM encoder strategy, the number of bits generated by DLT achieves the maximum when there are 128 values in DLT, which are 128, 129…., 254, 255. In such a case, DLT generates 1937 bits.

**Table 1. DLT coding method in the current HTM**

|  |  |
| --- | --- |
| **dlt\_flag**[ i ] | u(1) |
| if( dlt\_flag[ i ] ) { |  |
| **num\_depth\_values\_in\_dlt**[ i ] | ue(v) |
| for ( j = 0; j < num\_depth\_values\_in\_dlt ; j++) { |  |
| **dlt\_depth\_value**[ i ][ j ] | ue(v) |
| } |  |
| } |  |

**Table 2. Number of bits in DLT and SPS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | DLT Size (bits) | SPS Size (bits) | D/S (%) |
| Balloons | V0\_depth | 555 | 845 | 65.68% |
| V1\_depth | 641 | 937 | 68.41% |
| V2\_depth | 617 | 915 | 67.43% |
| Kendo | V0\_depth | 485 | 775 | 62.58% |
| V1\_depth | 637 | 933 | 68.27% |
| V2\_depth | 649 | 947 | 68.53% |
| Newspapercc | V0\_depth | 606 | 896 | 67.63% |
| V1\_depth | 621 | 917 | 67.72% |
| V2\_depth | 696 | 994 | 70.02% |
| PoznanHall2 | V0\_depth | 420 | 712 | 58.99% |
| V1\_depth | 380 | 678 | 56.05% |
| V2\_depth | 381 | 681 | 55.95% |
| Average | | 557.33 | 852.50 | 65.38% |

# Proposed method

To code the DLT more efficiently, a range constrained bit map (RCBM) coding method is proposed. With RCBM, DLT is coded in two steps.

First, min\_dlt\_value and diff\_max\_dlt\_value are coded as unsigned integer to constrain the value range of DLT. The smallest value in DLT is min\_dlt\_value, and the largest is MaxDltValue, which equals to min\_dlt\_value + diff\_max\_dlt\_value.

Then, the bit map, bit\_map\_flag[i] for i = 0 .. (MaxDltValue - min\_dlt\_value – 2), is coded. If bit\_map\_flag[i] = 1, then depth value (i + min\_dlt\_value + 1) belongs to DLT. Otherwise, i.e., bit\_map\_flag[i] = 0, depth value (i + min\_dlt\_value + 1) does not belong to DLT.

A flag is used to signal whether to code min\_dlt\_value and diff\_max\_dlt\_value or not. When the range is not coded, min\_dlt\_value = -1, MaxDltValue = MAX\_DEPTH\_VALUE + 1, and a full bit map without a constrained range is coded. Obviously, the length of the full bit map is MAX\_DEPTH\_VALUE + 1 bits. The encoder can decide whether to code the range or not depending on which operation generates fewer bits. Hence, the proposed method generates MAX\_DEPTH\_VALUE + 2 (*e.g.* 257 for 8-bit samples) bits at most.

# Experimental results

The proposed methods are integrated on HTM-5.0.1[3]. In Table 3, the DLT sizes and SPS sizes of the anchor and the proposed method are compared using the four sequences that enable DLT. On average, the proposed method generates only 34% bits as many as the anchor to code DLT. The size of SPS is reduced to 58%. The common test conditions [4] are obeyed, and the results are listed in Table 4. It can be seen that the coding performance is similar because SPS only takes a small percentage in the entire bitstream.

**Table 3. Comparison between anchor and the proposed method on DLT and SPS sizes**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | DLT Size (bits) | | | SPS Size (bits) | | |
|  |  | Anchor | Proposed | P/A (%) | Anchor | Proposed | P/A (%) |
| Balloons | V0\_depth | 555 | 213 | 38.38% | 845 | 503 | 59.53% |
| V1\_depth | 641 | 257 | 40.09% | 937 | 553 | 59.02% |
| V2\_depth | 617 | 255 | 41.33% | 915 | 553 | 60.44% |
| Kendo | V0\_depth | 485 | 191 | 39.38% | 775 | 481 | 62.06% |
| V1\_depth | 637 | 255 | 40.03% | 933 | 551 | 59.06% |
| V2\_depth | 649 | 257 | 39.60% | 947 | 555 | 58.61% |
| Newspapercc | V0\_depth | 606 | 191 | 31.52% | 896 | 481 | 53.68% |
| V1\_depth | 621 | 195 | 31.40% | 917 | 491 | 53.54% |
| V2\_depth | 696 | 213 | 30.60% | 994 | 511 | 51.41% |
| PoznanHall2 | V0\_depth | 420 | 111 | 26.43% | 712 | 403 | 56.60% |
| V1\_depth | 380 | 101 | 26.58% | 678 | 399 | 58.85% |
| V2\_depth | 381 | 104 | 27.30% | 681 | 404 | 59.32% |
| Average | | 557.33 | 195.25 | 34.39% | 852.50 | 490.42 | 57.68% |

**Table 4. Performance of test based on HTM5.0.1**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **video 0** | **video 1** | **video 2** | **video only** | **synthesized only** | **coded & synthesized** | **enc time** | **dec time** | **ren time** |
| Balloons | 0.00% | 0.00% | 0.00% | 0.00% | -0.02% | -0.02% | 87.3% | 94.1% | 92.0% |
| Kendo | 0.00% | 0.00% | 0.00% | 0.00% | -0.02% | -0.02% | 90.2% | 93.0% | 91.2% |
| Newspapercc | 0.00% | 0.00% | 0.00% | 0.00% | -0.02% | -0.02% | 88.8% | 98.7% | 93.2% |
| GhostTownFly | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 77.1% | 72.2% | 81.7% |
| PoznanHall2 | 0.00% | 0.00% | 0.00% | 0.00% | -0.03% | -0.03% | 94.7% | 94.6% | 94.2% |
| PoznanStreet | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 63.2% | 72.5% | 71.6% |
| UndoDancer | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% | 74.9% | 70.1% | 82.4% |
| 1024x768 | 0.00% | 0.00% | 0.00% | 0.00% | -0.02% | -0.02% | 88.8% | 95.2% | 92.2% |
| 1920x1088 | 0.00% | 0.00% | 0.00% | 0.00% | -0.01% | -0.01% | 76.7% | 76.7% | 82.1% |
| average | 0.00% | 0.00% | 0.00% | 0.00% | -0.01% | -0.01% | 81.7% | 84.2% | 86.3% |

# Working draft modifications

The proposed working draft modifications are as follows.

|  |  |
| --- | --- |
| vps\_extension( ) { |  |
| … |  |
| **dlt\_flag**[ i ] | u(1) |
| if( dlt\_flag[ i ] ) { |  |
| **~~num\_depth\_values\_in\_dlt~~**~~[ i ]~~ | ~~ue(v)~~ |
| ~~for ( j = 0; j < num\_depth\_values\_in\_dlt ; j++) {~~ |  |
| **~~dlt\_depth\_value~~**~~[ i ][ j ]~~ | ~~ue(v)~~ |
| **code\_full\_bit\_map\_flag**[i] | u(1) |
| if(!**code\_full\_bit\_map\_flag**[i]){ |  |
| **min\_dlt\_value**[i] | u(v) |
| **diff\_max\_dlt\_value**[i] | u(v) |
| } |  |
| for(j=0;j< MaxDltValue[i] - min\_dlt\_value[i] – 1;j++) |  |
| **bit\_map\_flag**[i][j] | u(1) |
| } |  |

**Video parameter set extension semantics**

**~~num\_depth\_values\_in\_dlt~~**~~[ i ] specifies the number of different depth values and the number of elements in the depth lookup table for depth view components of the current layer with layer\_id equal to i.~~

**~~dlt\_depth\_value[~~**~~i~~**~~]~~**~~[ j ] specifies the j-th entry in the depth lookup table for depth view components with layer\_id equal to i.~~

**code\_full\_bit\_map\_flag**[i] specifies whether to code the full bit map or not for depth view components with layer\_id equal to i.

**min\_dlt\_value**[i] specifies the smallest value in the depth lookup table for depth view components with layer\_id equal to i. The number of bits used to represent it is log2(MAX\_DEPTH\_VALUE + 1). When min\_dlt\_value[i] is not present, it shall be inferred to be -1.

**diff\_max\_dlt\_value**[i] specifies the difference between the largest and smallest value in the depth lookup table for depth view components with layer\_id equal to i. MaxDltValue[i] is computed as follows: MaxDltValue[i] = min\_dlt\_value[i] + diff\_max\_dlt\_value[i]. The number of bits used to represent it is log2(MAX\_DEPTH\_VALUE+1-min\_dlt\_value[i]). When diff\_max\_dlt\_value[i] is not present, MaxDltValue[i] shall be inferred to be MAX\_DEPTH\_VALUE+1.

**bit\_map\_flag**[i][j]specifies the j-th entry in the bit map for depth view components with layer\_id equal to i.

**Decoding process for a depth lookup table**

* ~~For i = 0..num\_depth\_values\_in\_dlt –1 the elements in Idx2DepthValue are derived as follows.~~
  + ~~Idx2DepthValue[ i ] is set equal to dlt\_depth\_value[ i ]~~
* The elements in Idx2DepthValue are derived as follows.
  + Set Idx= 0;
    - If min\_dlt\_value >=0, then Idx2DepthValue[ 0 ] is set equal to min\_dlt\_value and Idx++;
  + If MaxDltValue- min\_dlt\_value>1, then for i=0… MaxDltValue- min\_dlt\_value – 2
    - * If bit\_map\_flag[i]==1, then Idx2DepthValue[Idx] = i + min\_dlt\_value + 1 and Idx++;
  + If MaxDltValue <= MAX\_DEPTH\_VALUE and MaxDltValue != min\_dlt\_value, then Idx2DepthValue[Idx] = MaxDltValue and Idx++;
  + num\_depth\_vaules\_indlt=Idx;

# Conclusion

In this proposal, an efficient coding method for the depth lookup table (DLT) is proposed. The proposed method can reduce 2/3 of the original DLT size, and the size of SPS in a depth component is also reduced significantly. Experimental results reveal that the proposed method provides almost the same coding performance as the current HTM for the entire multi-view video.

# Patent rights declaration (s)

**MediaTek Inc. may have current or pending patent rights relating to the technology described in this contribution and, conditioned on reciprocity, is prepared to grant licenses under reasonable and non-discriminatory terms as necessary for implementation of the resulting ITU-T Recommendation | ISO/IEC International Standard (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).**

# References

1. G. Tech, K. Wegner, Y. Chen, S. Yea, “3D-HEVC test model 2,” Document of Joint Collaborative Team on 3D Video Coding Extension Development, JCT3V-B1005, October, 2012.
2. F. Jager, “3D-CE6.h: Simplified depth coding with an optional depth lookup table,” Document of Joint Collaborative Team on 3D Video Coding Extension Development, JCT3V-B0036, October, 2012.
3. HTM-5.0.1, <https://hevc.hhi.fraunhofer.de/svn/svn_3DVCSoftware/tags/HTM-5.0.1/>
4. D. Rusanovskyy, K. Müller, A. Vetro, “Common test conditions of 3DV core experiments,” Document of Joint Collaborative Team on 3D Video Coding Extension Development, JCT3V-A1100, July, 2012.