|  |  |
| --- | --- |
| Joint Collaborative Team on Video Coding (JCT-VC)  of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11  17th Meeting: Valencia, ES, 27 March – 4 April 2014 | Document: JCTVC-Q0065 |

|  |  |  |  |
| --- | --- | --- | --- |
| Title: | **AhG10: Transition copy mode for Palette mode** | | |
| Status: | Input Document to JCT-VC | | |
| Purpose: | Proposal | | |
| Author(s) or Contact(s): | C. Gisquet, G. Laroche, P. Onno  Canon Research Centre France Rue de la Touche Lambert 35510 CESSON-SEVIGNE, FRANCE | Tel: Email: | +33 299876800 [christophe.gisquet@crf.canon.fr](mailto:christophe.gisquet@crf.canon.fr) [patrice.onno@crf.canon.fr](mailto:patrice.onno@crf.canon.fr) |
| Source: | Canon | | |

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

# Abstract

During the previous meeting, a new index prediction method was proposed, using the position of so-called transitions to determine a position from which the prediction copies indexes. This contribution presents a small modification to the update mechanism of the proposition, and also describes a new prediction method that encodes a run of values being the expected transition. It is reported that the use of one such prediction mode provides gains for respectively AI-MT, AI-HT and AI-SHT configurations, on top of AhG10 software, of more than 2.4%/2.8%/2.9% for SC classes, and up to 5.8%/6.0%/6.1% on optional classes. It is also reported that a simpler mode only predicting the index value provides gains for respectively AI-MT, AI-HT and AI-SHT configurations, on top of AhG10 software, of more than 1.9%/2.3%/2.1% for SC classes, and up to 4.9%/5.0%/5.1% on optional classes.

# Introduction

AhG10 proposes 2 index predictions methods that can be seen in the following figure:

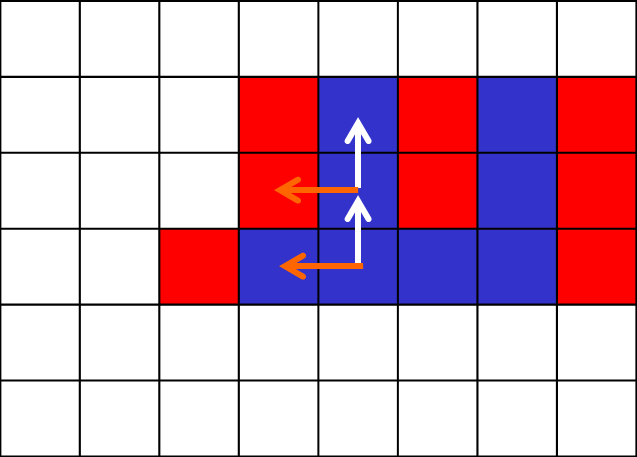


Figure 1 : index prediction modes in RCE4 test2

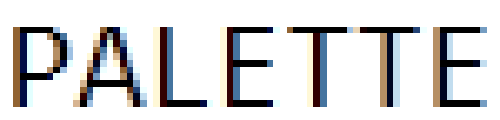
In the above example, we have represented arrows for these 2 prediction modes:

* The orange arrow copies from the left, i.e. represents a repetition of the same value;
* The white arrow copies from the above, i.e. is a repetition of the line above.

In the above example, one can easily deduce that the above mode will be selected for the first case, while the left mode will be used for the second one.

These modes are well suited for strictly vertical or horizontal features. However, there are instances where palette-like content exhibits different behavior. In particular, one such content can be observed below:

Figure 2 : example of specific palette content



This example represents text rendered with antialiasing, which is critical to user reading comfort. In such an example, one can clearly see the existence of so-called transitions. One can easily notice the repetitiveness of these transitions: the red, orange and blue arrows show repetitive transitions across the above figure.

# Proposal

## Concept

We propose adding a new prediction mode, called here transition mode or pattern mode, which takes into account above transitions. The idea is, for each palette index, to record the position of its last occurrence. Then, the new prediction mode will use these positions as start places for copying, depending on the value of the pixel on the left.

The above can be represented in the following figure:

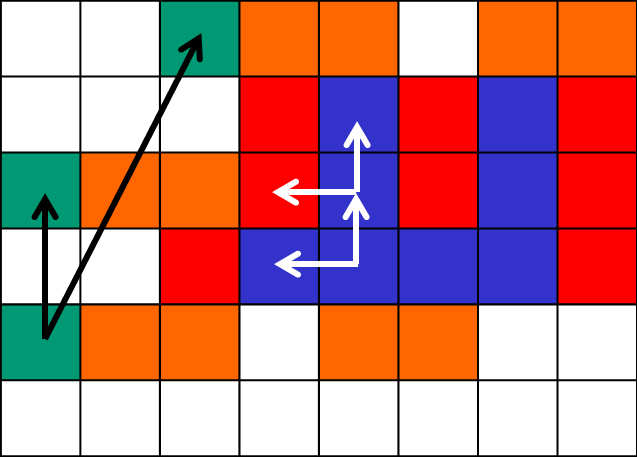


Figure 3 : transition copy mode

Classical prediction modes are identical here, while we have presented the new possible modes with black arrows. In these cases, when encoding the orange index, the green one on the left is an indication of a potential transition. Now, having recorded last positions of the green index, it is possible to copy from these positions. The above example is a simplified version, but Figure 2 easily shows how that can apply.

## Improvements to the update mechanism

When the copy up or copy transition methods are used, this means the content is a repetition of an already coded content, thereby providing no new information. As a consequence, it is more beneficial to only update the transition information when the prediction mode is copy left, as it is the only one conveying new information.

In addition, it may happen that the index coded by the copy left prediction method is the same as the last one: indeed, the copy-up run may end because a new index occurs, while the run of the value is actually longer. Such is the case in Figure 3 where blue and red levels are present on the above line, but the current line contains a run of blue levels. As a consequence, the copy-left mode actually provides no new information, as it does not correspond to a transition.

Last but not least, the copy-left method is the simplest prediction method, as it does not read from the index buffer. It seems better to not burden other, more complex, prediction methods with the added update mechanism.

## Modified encoder algorithm

The modified algorithm for coding the index can be represented as follows:

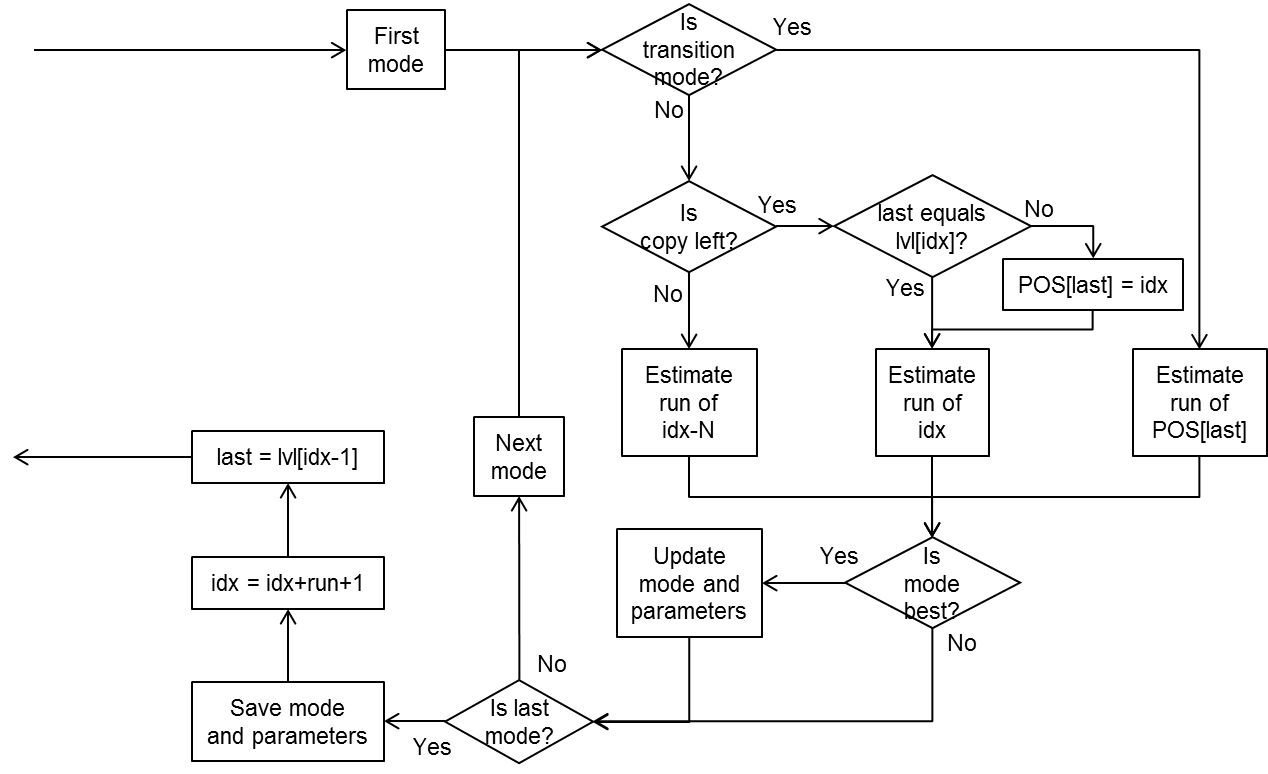


Figure 4 : encoder algorithm

The new transition mode can operate mostly similarly to the copy from above mode, except the position from which to copy is not, for a CU of size NxN, the position idx-N above but a position which depends on the last occurrence of the level at the left of the current position. POS is a table of 32 elements, storing position in the current block, which in the worst case (64x64 CU) is 32\*10 bits. In addition, this requires storing all decoded levels, which in the same worst case is 4095\*5 bits, for a total of around 2.5Kbytes.

## Simplified method

To further reduce storage requirements, it is simpler to only store the index of the transition, for each index: in Figure 3, instead of POS[last] = idx, one can store POS[last] = lvl[idx] instead, and the estimate run is a run of the value of index POS[last]. The worst case memory is therefore 32\*5bits (20 bytes). As a consequence, instead of copying from a position, the transition mode represents a run of the index associated to the transition.

## Entropy coding

It currently uses a flag whose context is based on a neighbor mode. It is more beneficial to use the left neighbor instead of the top one, as is currently done in AhG10 software. We have then extended this by using the following codewords:

|  |  |
| --- | --- |
| Codeword | Mode |
| 0 | Copy left |
| 10 | Copy transition |
| 11 | Copy above |

Table 1 : prediction mode codewords

# Results

## Normal mode

Using AhG10 software as anchor, we have for the AI lossy configurations:



The results are somewhat below what was observed in RCE4Test2 of previous meeting. As this new transition mode increases signaling, the overhead may be amplified by the overall reduction in bitrate due to e.g. IBC PUs.

For lossless, we have:



## Simplified mode

Using the same anchor and configurations, the results are:



The results are a bit below. It can be observed that more palette complex content has a larger loss, while class F now offers improvements. For lossless, we have:



The simple transition mode therefore causes a loss often close to 0.6% for lossy, and 0.3% for lossless, compared to the “normal” transition mode.

# Conclusion

In light of the contained complexity increase and important gains, we suggest further studying the method in any palette study.

# Patent rights declaration(s)

**Canon Research Centre France 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).**