INTERNATIONAL ORGANIZATION FOR STANDARDIZATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC 1/SC 29/WG 04 MPEG VIDEO CODING

ISO/IEC JTC 1/SC 29/WG 04 m63700

July 2023, Geneva, Switzerland

Title **[MIV] Chroma dynamic range modification and MIV ed. 2**

Source **PUT, ETRI**

Authors **Adrian Dziembowski, Dawid Mieloch, Gwangsoon Lee, Jun Young Jeong**

Abstract

The document presents the updated syntax for chroma dynamic range modification (m63397). The syntax is compatible with casps_miv_2_extension (m63655, m63656).

Syntax changes (compared to m63397):

- moved casme_chroma_scaling_present_flag: casps_miv_extension => casps_miv_2_extension,
- cs_u_min, cs_u_max, cs_v_min, and cs_v_max use u(v) instead of u(16).

Ver2:

- added **mvp_chroma_scaling_bit_depth_minus_1**
- added H.4 section

Ver3

1 Syntax & semantics

Syntax and semantics are analogous to the attribute/texture offset already adopted to MIV.

8.3.2.9 Common atlas sequence parameter set MIV edition 2 extension syntax

casps_miv_2_extension( ) {	**Descriptor**
casme_reserved_zero_8bits	u(8)
casme_decoder_side_depth_estimation_flag	u(1)
casme_chroma_scaling_present_flag	u(1)
}	

casme_chroma_scaling_present_flag equal to 1 indicates that the chroma scaling parameters are present in the syntax structure. casme_chroma_scaling_present_flag equal to 0 indicates that the chroma scaling parameters are not present in the syntax structure. When not present, the value of casme_chroma_scaling_present_flag is inferred to be equal to 0.

Remaining part of the proposed syntax did not change since m63397, except for green parts.

8.3.2.6 Common atlas frame

8.3.2.6.1 MIV extension syntax

caf_miv_extension( ) {	Descriptor
if(nal_unit_type == NAL_CAF_IDR) {	
miv_view_params_list( )	
} else {	
came_update_extrinsics_flag	u(1)
came_update_intrinsics_flag	u(1)
if(casme_depth_quantization_params_present_flag)	
came_update_depth_quantization_flag	u(1)
if(casme_chroma_scaling_present_flag)	
came_update_chroma_scaling_flag	u(1)
if(came_update_extrinsics_flag)	
miv_view_params_update_extrinsics( )	
if(came_update_intrinsics_flag)	
miv_view_params_update_intrinsics( )	
if(came_update_depth_quantization_flag)	
miv_view_params_update_depth_quantization( )	
if(came_update_chroma_scaling_flag)	
miv_view_params_update_chroma_scaling( )	
}	
}	

came_update_chroma_scaling_flag equal to 1 indicates that the miv_view_params_update_chroma_scaling( ) syntax structure is present in this syntax structure. came_update_chroma_scaling_flag equal to 0 indicates that the miv_view_params_update_chroma_scaling ( ) syntax structure is not present in this syntax structure. When not present, the value of came_update_chroma_scaling_flag is inferred to be equal to 0.

8.3.2.6.2 MIV view parameters list syntax

miv_view_params_list( ) {	Descriptor
mvp_num_views_minus1	u(16)
mvp_explicit_view_id_flag	u(1)
if(mvp_explicit_view_id_flag)	
for(v = 0; v <= mvp_num_views_minus1; v++)	
mvp_view_id[v]	u(16)
for(v = 0; v <= mvp_num_views_minus1; v++) {	
camera_extrinsics(v)	
mvp_inpaint_flag[v]	u(1)
}	
mvp_intrinsic_params_equal_flag	u(1)
for(v = 0; v <= mvp_intrinsic_params_equal_flag ? 0 : mvp_num_views_minus1; v++)	
camera_intrinsics(v)	
if(casme_depth_quantization_params_present_flag) {	
mvp_depth_quantization_params_equal_flag	u(1)

for(v = 0; v <= mvp_depth_quantization_equal_flag ? 0 : mvp_num_views_minus1; v++)	
depth_quantization(v)	
}	
mvp_pruning_graph_params_present_flag	u(1)
if (mvp_pruning_graph_params_present_flag)	
for(v = 0; v <= mvp_num_views_minus1; v++)	
pruning_parents(v)	
if (casme_chroma_scaling_present_flag) {	
mvp_chroma_scaling_bit_depth_minus1	u(5)
for(v = 0; v <= mvp_num_views_minus1; v++)	
chroma_scaling(v)	
}	
}	

mvp_chroma_scaling_bit_depth_minus1 plus 1 specifies the number of bits used to represent the chroma scaling syntax elements.

NOTE – The value of mvp_chroma_scaling_bit_depth_minus_1 is expected to be equal to or larger than the maximum of ai_attribute_2d_bit_depth_minus1[*aspsAtlasID*][attrIdx] values, for all values of *aspsAtlasID* and attrIdx where ai_attribute_type_id[*aspsAtlasID*][attrIdx] is equal to ATTR_TEXTURE, inclusive.

8.3.2.6.10 MIV view parameters update chroma scaling syntax

miv_view_params_update_chroma_scaling( ) {	**Descriptor**
mvpucs_num_view_updates_minus1	u(16)
for(i = 0; i <= mvpucs_num_view_updates_minus1; i++) {	
mvpucs_view_idx[i]	u(16)
chroma_scaling(mvpucs_view_idx[i])	
}	
}	

mvpucs_num_view_updates_minus1 plus 1 specifies the number of chroma_scaling(v) syntax structures that are present within this syntax structure. The value of mvpucs_num_view_updates_minus1 shall be in the range of 0 to mvp_num_views_minus1, inclusive.

mvpucs_view_idx[i] specifies the view index for which updated chroma scaling parameters will be signalled. The value of mvpucs_view_idx[i] shall be in the range of 0 to mvp_num_views_minus1, inclusive.

8.3.2.6.11 Chroma scaling syntax

chroma_scaling(v) {	**Descriptor**
cs_u_min[v]	u(v)
cs_u_max[v]	u(v)
cs_v_min[v]	u(v)
cs_v_max[v]	u(v)
}	

cs_u_min[v] specifies in scene the minimum value of the first chroma of the view with index equal to v, calculated over entire frame and entire group of pictures. The number of bits used for the representation of cs_u_min[v] is equal to mvp_chroma_scaling_bit_depth_minus1 + 1.

cs_u_max[v] specifies in scene the maximum value of the first chroma of the view with index equal to v, calculated over entire frame and entire group of pictures. The number of bits used for the representation of cs_u_min[v] is equal to mvp_chroma_scaling_bit_depth_minus1+ 1.

cs_v_min[v] specifies in scene the minimum value of the second chroma of the view with index equal to v, calculated over entire frame and entire group of pictures. The number of bits used for the representation of cs_u_min[v] is equal to mvp_chroma_scaling_bit_depth_minus1+ 1.

cs_v_max[v] specifies in scene the maximum value of the second chroma of the view with index equal to v, calculated over entire frame and entire group of pictures. The number of bits used for the representation of cs_u_min[v] is equal to mvp_chroma_scaling_bit_depth_minus1+ 1.

H.4 ~~Patch texture offset process~~ Texture reconstruction

H.4.1 Patch texture offset process

No changes here

H.4.2 Chroma range restoration

The chroma range restoration process changes the range of two chroma component values of each atlas sample.

Inputs to this process are:

— the variable *patchIdx*, corresponding to the atlas patch index of the atlas sample;

— the variable *atlasID*, the atlas ID;

— the variable *attrIdx*, corresponding to the texture attribute index of the atlas sample;

— the 2D array *chromaUSample*, corresponding to the second component value of the sample texture attribute;

— the 2D array *chromaVSample*, corresponding to the third component value of the sample texture attribute.

The output of this process is:

— the 2D array *chromaUSample*, corresponding to the modified second component value of the sample texture attribute;

— the 2D array *chromaVSample*, corresponding to the modified third component value of the sample texture attribute.

The chroma range restoration process is defined as follows:

```
v = AtlasPatchViewIndex[ patchIdx ]
chromaUSample = ( chromaUSample *
    ( cs_u_max[ v ] - cs_u_min[ v ] ) + cs_u_min[ v ] ) >>
    ai_attribute_2d_bit_depth[ atlasID ][ attrIdx ]
chromaVSample = ( chromaVSample *
    ( cs_v_max[ v ] - cs_v_min[ v ] ) + cs_v_min[ v ] ) >>
    ai_attribute_2d_bit_depth[ atlasID ][ attrIdx ]
```

2 Recommendation

We recommend:

- adopting the proposed syntax,
- integrating the change u(16) -> u(v) into TMIV17.

3 Acknowledgement

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2018-0-00207, Immersive Media Research Laboratory).