<m55368> Mesh Patch Data

Danillo Graziosi, Alexandre Zaghetto, and Ali Tabatabai

<Problem statement>

- V3C is about **projections**
- Mesh is about **triangles** (or quads?)
- So let's encode meshes by **triangle projection**!

V3C Mesh Coding

Mesh Voxelization (see m55367)

Patch Creation

V3C Image creation

Vertices (U,V) location + connectivity

Mesh coding

V3C bitstream

1,938,384 points

Point Cloud Reconstruction

Mesh Reconstruction

27,942 vertices
39,240 faces

Normal Calculation

Adjacency Calculation

Initial Segmentation

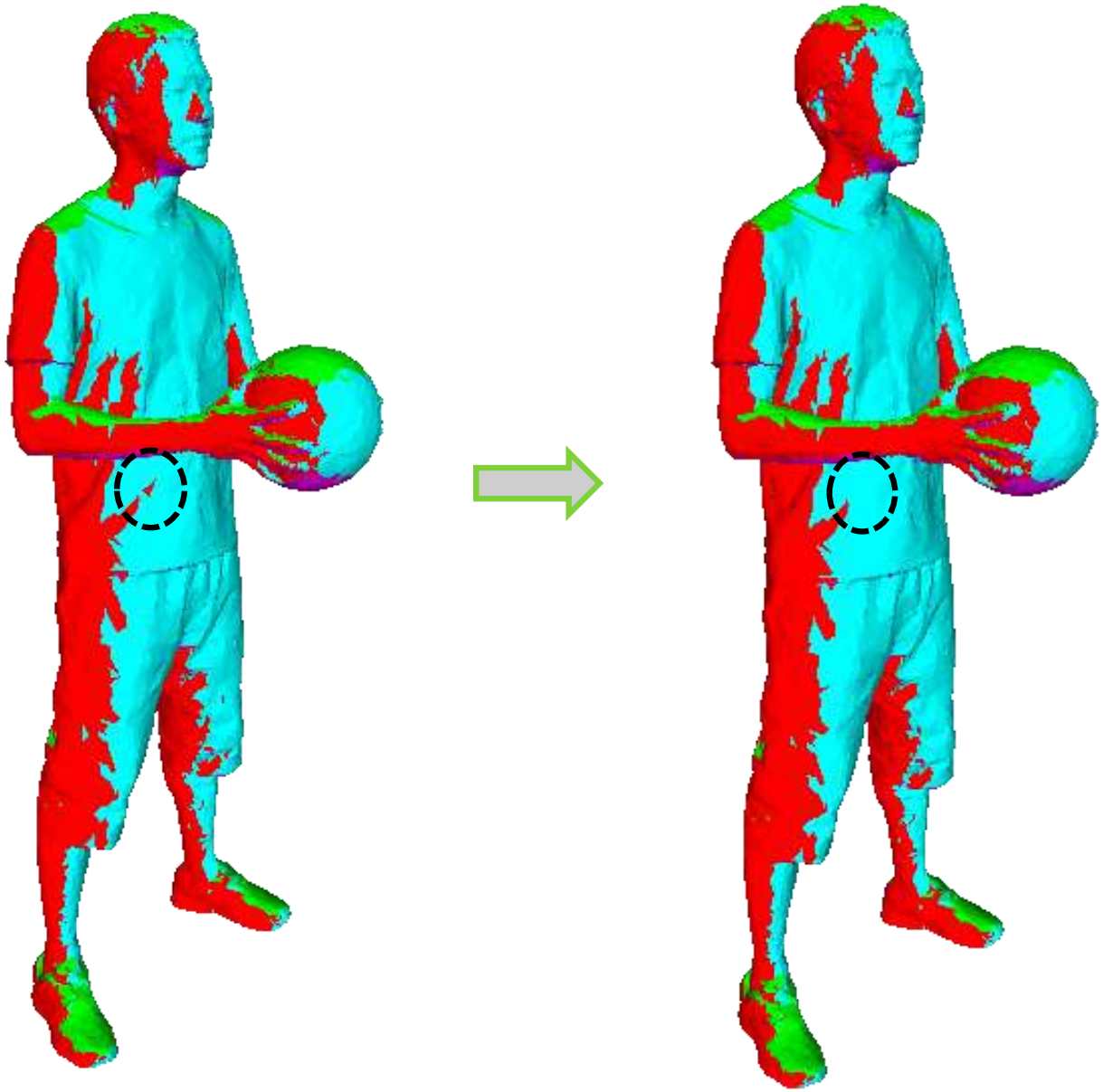
Refinement

Patch Projection

Patch Rasterization

Patch generation (1/2)

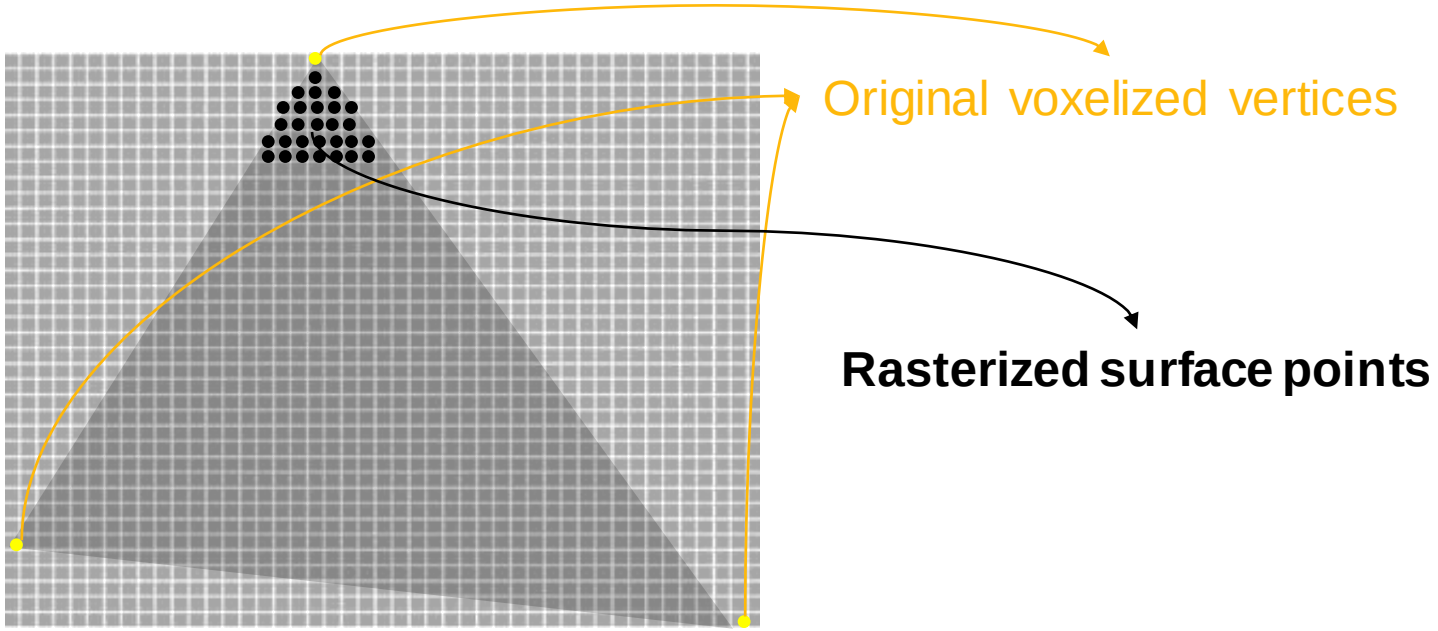
- Determine adjacency neighborhood of each triangle
- Calculate normal of each triangle
  - Cross-product of triangle edges
- Categorize the triangle according to the normal
- Refine category according to neighboring values



Patch creation (2/2)

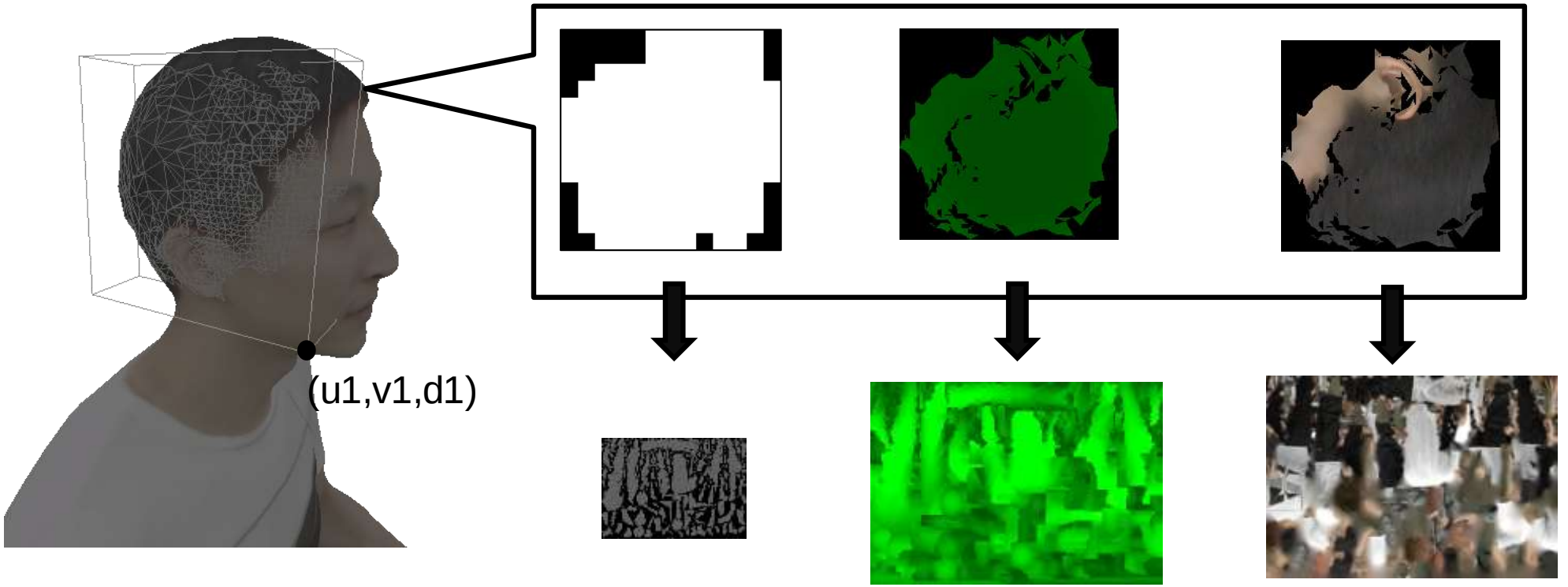
Until all triangles are projected

- Create connected components of triangles
  - Triangles with same category sharing at least one vertex
- If bounding box of the CC is smaller than a certain pre-defined area, the **triangles are moved to a separate list** for independent triangle coding (see contribution m55370)
  - These unprojected triangles will not be rasterized, but coded as vertices with associated color per vertex
- Otherwise, Project each triangle to the patch
  - If projected position of a vertex is already occupied, triangle must be coded in another patch. So it goes to a missing triangles list, to be processed again later (see the loop).
- Rasterize the triangle to generate the points for the geometry/attribute images (point cloud representation)



V3C image generation

- Occupancy Map and Geometry are generated the same way as before
- Attribute Map is generated from the uncompressed geometry
  - Attribute transfer using meshes can be implemented
  - From Meshlab → Texture: Transfer: Vertex Attributes to Texture



Note: Occupancy Map can be skipped for mesh

Mesh Patch Data Syntax

mesh_patch_data_unit(tileID, patchIdx) {	Descriptor
mpdu_2d_pos_x[tileID][patchIdx]	u(v)
mpdu_2d_pos_y[tileID] [patchIdx]	u(v)
mpdu_2d_size_x_minus1[tileID] [patchIdx]	se(v)
mpdu_2d_size_y_minus1[tileID] [patchIdx]	se(v)
mpdu_3d_offset_u[tileID][patchIdx]	u(v)
mpdu_3d_offset_v[tileID][patchIdx]	u(v)
mpdu_3d_offset_d[tileID][patchIdx]	u(v)
if(asps_normal_axis_max_delta_value_enabled_flag)	
mpdu_3d_range_d[tileID][patchIdx]	u(v)
mpdu_projection_id[tileID][patchIdx]	u(v)
mpdu_orientation_index[tileID][patchIdx]	u(v)
if(afps_lod_mode_enabled_flag)	
mpdu_lod_enabled_flag[tileID][patchIdx]	u(1)
if(mpdu_lod_enabled_flag[tileID][patchIdx] > 0) {	
mpdu_lod_scale_x_minus1[tileID][patchIdx]	ue(v)
mpdu_lod_scale_y_idc[tileID][patchIdx]	ue(v)
}	
}	

if(asps_mesh_binary_coding_enabled_flag)	
mpdu_binary_object_present_flag[tileID][patchIdx]	u(1)
if(mpdu_binary_object_present_flag[tileID][patchIdx]) {	
mpdu_mesh_binary_object_size_bytes[tileID][patchIdx]	ue(v)
for(i = 0; i < mpdu_mesh_payload_size_bytes[tileID][patchIdx]; i++)	
mpdu_mesh_binary_object[tileID][patchIdx][i]	b(8)
} else {	
mpdu_vertex_count_minus3[tileID][patchIdx]	ue(v)
mpdu_face_count [tileID][patchIdx]	ue(v)
for(i = 0; i < mpdu_faces_count [tileID][patchIdx]; i++) {	
mpdu_face_vertex[tileID][patchIdx][i][0]	u(v)
mpdu_face_vertex[tileID][patchIdx][i][1]	u(v)
mpdu_face_vertex[tileID][patchIdx][i][2]	u(v)
if(asps_mesh_quad_face_flag) {	
mpdu_face_vertex[tileID][patchIdx][i][3]	u(v)
}	
if(!asps_mesh_vertices_in_vertex_video_data) {	
for(i = 0; i < mpdu_vertex_count_minus3[tileID][patchIdx] +3 ; i++) {	
mpdu_vertex_pos_x[tileID][patchIdx][i]	u(v)
mpdu_vertex_pos_y[tileID][patchIdx][i]	u(v)
}	
}	
}	
}	

2D, 3D, orientation and lod are the same as used for point clouds

Mesh Patch Data Unit Semantics

- Semantics of the new elements (marked in green)

mpdu_binary_object_present_flag[tileID][p] equal to 1 specifies that the syntax elements mpdu_mesh_binary_object_size_bytes[tileID][p] and mpdu_mesh_binary_object[tileID][p][i] are present for the patch with index p of the current atlas tile, with tile ID equal to tileID. If mpdu_binary_object_present_flag[tileID][p] is equal to 0, the syntax elements mpdu_mesh_binary_object_size_bytes[tileID][p] and mpdu_mesh_binary_object[tileID][p][i] are not present for the current patch. If mpdu_binary_object_present_flag[tileID][p] is not present, its value shall be inferred to be equal to 0.

mpdu_mesh_binary_object_size_bytes[tileID][p] specifies the number of bytes used to represent the mesh information in binary form for the patch with index p of the current atlas tile, with tile ID equal to tileID.

mpdu_mesh_binary_object[tileID][p][i] specifies the i-byte of the binary representation of the mesh for patch with index p of the current atlas tile, with tile ID equal to tileID.

mpdu_vertex_count_minus3[tileID][p] plus 3 specifies the number of vertices present in the patch with index p of the current atlas tile, with tile ID equal to tileID.

mpdu_face_count[tileID][p] specifies the number of triangles present in the patch with index p of the current atlas tile, with tile ID equal to tileID. When not present, the value of mpdu_face_count[tileID][p] shall be zero.

mpdu_face_vertex[tileID][p][i][k] specifies the k-th value of the vertex index for the i-th triangle or quad for the current patch with index p of the current atlas tile, with tile ID equal to tileID. The value of mpdu_face_vertex[tileID][p][i][k] shall be in the range of 0 to mpdu_vert_count_minus3[tileID][p] + 2, inclusive.

mpdu_vertex_pos_x[tileID][p][i] specifies the value of the x-coordinate of the i-th vertex for the current patch with index p of the current atlas tile, with tile ID equal to tileID. The value of mpdu_vertex_pos_x[p][i] shall be in the range of 0 to mpdu_2d_size_x_minus1[tileID][p], inclusive.

mpdu_vertex_pos_y[tileID][p][i] specifies the value of the y-coordinate of the i-th vertex for the current patch p of the current atlas tile, with tile ID equal to tileID. The value of mpdu_vertex_pos_y[tileID][p][i] shall be in the range of 0 to mpdu_2d_size_y_minus1[tileID][p], inclusive.

Patch information data syntax

patch_information_data(tileID, patchIdx, patchMode) {	Descriptor
if(ath_type == P_TILE) {	
if(patchMode == P_SKIP)	
skip_patch_data_unit( )	
else if(patchMode == P_MERGE)	
merge_patch_data_unit(tileID, patchIdx)	
else if(patchMode == P_INTRA)	
patch_data_unit(tileID, patchIdx)	
else if(patchMode == P_MESH)	
mesh_patch_data_unit(tileID, patchIdx)	
else if(patchMode == P_TRIANGLE)	
triangle_patch_data_unit(tileID, patchIdx)	
else if(patchMode == P_TRACKED_MESH)	
tracked_mesh_patch_data_unit(tileID, patchIdx)	
else if(patchMode == P_INTER)	
inter_patch_data_unit(tileID, patchIdx)	
else if(patchMode == P_RAW)	
raw_patch_data_unit(tileID, patchIdx)	
else if(patchMode == P_EOM)	
eom_patch_data_unit(tileID, patchIdx)	
}	
else if(ath_type == I_TILE) {	
if(patchMode == I_INTRA)	
patch_data_unit(tileID, patchIdx)	
if(patchMode == I_MESH)	
mesh_patch_data_unit(tileID, patchIdx)	
else if(patchMode == P_TRIANGLE)	
triangle_patch_data_unit(tileID, patchIdx)	
else if(patchMode == I_RAW)	
raw_patch_data_unit(tileID, patchIdx)	
else if(patchMode == I_EOM)	
eom_patch_data_unit(tileID, patchIdx)	
}	
}	

atdu_patch_mode[tileID][p]	Identifier	Description
0	I_INTRA	Non-predicted patch mode
1	I_RAW	RAW Point Patch mode
2	I_EOM	EOM Point Patch mode
3	I_MESH	MESH Patch mode
4	I_TRIANGLE	TRIANGLE Patch mode
5-13	I_RESERVED	Reserved modes for future use by ISO/IEC
14	I_END	Patch termination mode

atdu_patch_mode[tileID][p]	Identifier	Description
0	P_SKIP	Patch Skip mode
1	P_MERGE	Patch Merge mode
2	P_INTER	Inter predicted Patch mode
3	P_INTRA	Non-predicted Patch mode
4	P_RAW	RAW Point Patch mode
5	P_EOM	EOM Point Patch mode
6	P_MESH	MESH Patch mode
7	P_TRIANGLE	TRIANGLE Patch mode
8	P_TRACKED_MESH	TRACKED MESH Patch mode
9-13	P_RESERVED	Reserved modes for future use by ISO/IEC
14	P_END	Patch termination mode

See m55370 and m55372 for triangle and tracked mesh patch data units

ASPS extension for meshes

- Some elements of the mesh patch data are controlled by parameters defined in ASPS. We propose to create a new extension for the ASPS for meshes, with the following content:

atlas_sequence_parameter_set_rbsp( ) {	Descriptor
...	
asps_extension_present_flag	u(1)
if(asps_extension_present_flag) {	
asps_vpcc_extension_present_flag	u(1)
asps_mesh_extension_present_flag	u(1)
asps_extension_6bits	u(6)
}	
if(asps_vpcc_extension_present_flag)	
asps_vpcc_extension( ) /* Specified in Annex H*/	
if(asps_mesh_extension_present_flag)	
asps_mesh_extension( )	
if(asps_extension_6bits)	
while(more_rbsp_data( ))	
asps_extension_data_flag	u(1)
rbsp_trailing_bits( )	
}	

asps_mesh_extension_present_flag equal to 1 specifies that the asps_mesh_extension( ) syntax structure is present in the atlas_sequence_parameter_set_rbsp syntax structure. asps_mesh_extension_present_flag equal to 0 specifies that this syntax structure is not present. When not present, the value of asps_mesh_extension_present_flag is inferred to be equal to 0.

asps_extension_6bits equal to 0 specifies that no asps_extension_data_flag syntax elements are present in the ASPS RBSP syntax structure. When present, asps_extension_6bits shall be equal to 0 in bitstreams conforming to this version of this document. Values of asps_extension_6bits not equal to 0 are reserved for future use by ISO/IEC. Decoders shall allow the value of asps_extension_6bits to be not equal to 0 and shall ignore all asps_extension_data_flag syntax elements in an ASPS NAL unit. When not present, the value of asps_extension_6bits is inferred to be equal to 0.

asps_mesh_extension( ) {	Descriptor
asps_mesh_binary_coding_enabled_flag	u(1)
if(asps_mesh_binary_coding_enabled_flag)	
asps_mesh_binary_codec_id	u(8)
asps_mesh_quad_face_flag	u(1)
asps_mesh_vertices_in_vertex_video_data_flag	u(1)
}	

MeshCodec	asps_mesh_binary_codec_idc	Descriptor
SC3DM	0	SC3DM (MPEG) codec
Draco	1	Draco (Google) codec
Reserved	4..255	–

asps_mesh_binary_coding_enabled_flag equal to 1 indicates that vertex and connectivity information associated to a patch is present in binary format. asps_mesh_binary_coding_enabled_flag equal to 0 specifies that the mesh vertex and connectivity data is not present in binary format. When not present, asps_mesh_binary_coding_enabled_flag is inferred to be 0.

asps_mesh_binary_codec_id indicates the identifier of the codec used to compress the vertex and connectivity information for patch. asps_mesh_binary_codec_id shall be in the range of 0 to 255, inclusive. This codec may be identified through the profiles defined in Annex A, or through means outside this document.

asps_mesh_quad_face_flag equal to 1 indicates that quads are used for the polygon representation. asps_mesh_quad_face_flag equal to 0 indicates that triangles are used for the polygon representation of meshes. When not present, the value of asps_mesh_quad_flag is inferred to be equal to 0.

asps_mesh_vertices_in_vertex_video_data_flag equal to 1 indicates that the vertex information is present in vertex video data. asps_mesh_vertices_in_vertex_flag equal to 0 indicates that vertex information is present in the patch data. When not present, the value of asps_mesh_vertices_in_vertex_map_flag is inferred to be equal to 0.

Vertex Video Sub-bitstream (V3C_VVD)

- New V3C video data unit carries information of the location of vertices
  - Can contain binary values indicating the location of the projected vertices or can also contain multi-level information to be used for connectivity reconstruction.
  - Allow for level-of-detail construction by coding vertices in multiple layers
  - Using VPS extension to define additional parameters (see next slide)

v3c_unit_header( ) {	Descriptor
vuh_unit_type	u(5)
if(vuh_unit_type == V3C_AVD || vuh_unit_type == V3C_GVD || vuh_unit_type == V3C_OVD || vuh_unit_type == V3C_AD || vuh_unit_type == V3C_VVD) {	
vuh_v3c_parameter_set_id	u(4)
vuh_atlas_id	u(6)
}	
if(vuh_unit_type == V3C_AVD) {	
vuh_attribute_index	u(7)
vuh_attribute_partition_index	u(5)
vuh_map_index	u(4)
vuh_auxiliary_video_flag	u(1)
} else if(vuh_unit_type == V3C_GVD) {	
vuh_map_index	u(4)
vuh_auxiliary_video_flag	u(1)
vuh_reserved_zero_12bits	u(12)
} else if(vuh_unit_type == V3C_VVD) {	
vuh_lod_index	u(4)
vuh_reserved_zero_13bits	u(13)
} else if(vuh_unit_type == V3C_OVD || vuh_unit_type == V3C_AD)	
vuh_reserved_zero_17bits	u(17)
else	
vuh_reserved_zero_27bits	u(27)
}	

v3c_unit_payload(numBytesInV3CPayload) {	Descriptor
if(vuh_unit_type == V3C_VPS)	
v3c_parameter_set(numBytesInV3CPayload)	
else if(vuh_unit_type == V3C_AD)	
atlas_sub_bitstream(numBytesInV3CPayload)	
else if(vuh_unit_type == V3C_OVD || vuh_unit_type == V3C_GVD || vuh_unit_type == V3C_AVD || vuh_unit_type == V3C_VVD)	
video_sub_bitstream(numBytesInV3CPayload)	
}	

vuh_lod_index when present, indicates the lod index of the current vertex stream. When not present, the lod index of the current vertex sub-bitstream is derived based on the type of the sub-bitstream and the operations described in subclause X.X for vertex video sub-bitstreams respectively. The value of vuh_lod_index, when present, shall be in the range of 0 to vms_lod_count_minus1[vuh_atlas_id], inclusive.

vuh_reserved_zero_13bits, when present, shall be equal to 0 in bitstreams conforming to this version of this document. Other values for vuh_reserved_zero_13bits are reserved for future use by ISO/IEC. Decoders shall ignore the value of vuh_reserved_zero_13bits.

VPS mesh extension

vps_mesh_extension( ) {	Descriptor
for(k = 0 ; k <= vps_atlas_count_minus1; k++) {	
vme_lod_count_minus1[k]	u(4)
if (vme_num_lod_minus1[k] == 0)	
vme_embed_vertex_in_occupancy_flag[k]	u(1)
if (!vme_embed_vertex_in_occupancy_flag[k]){	
vme_multiple_lod_streams_present_flag[k]	u(1)
vme_lod_absolute_coding_enabled_flag[j][0] = 1	
vme_lod_predictor_index_diff[k][0] = 0	
for(i = 1; i <= vme_num_lod_minus1[k]; i++) {	
if(vme_multiple_lod_streams_present_flag[k])	
vme_lod_absolute_coding_enabled_flag[k][i]	u(1)
else	
vme_lod_absolute_coding_enabled_flag[k][i] = 1	
if(vme_lod_absolute_coding_enabled_flag[k][i] == 0) {	
vme_lod_predictor_index_diff[k][i]	ue(v)
}	
}	
vme_vertex_video_present_flag[k]	u(1)
if (vme_vertex_video_present_flag[k])	
vertex_information(vps_atlas_id[k])	
}	
}	

vme_lod_count_minus1[k] plus 1 indicates the number of lods used for encoding the vertex data for the atlas with atlas ID k. vme_lod_count_minus1[j] shall be in the range of 0 to 15, inclusive.

vme_embed_vertex_in_occupancy_flag[k] equal to 1 specifies that vertex information is derived from occupancy map as specified in clause XX for the atlas with atlas ID k.

vme_embed_vertex_in_occupancy_flag[k] equal to 0 specifies that the vertex information is not derived from the occupancy video. When vme_embed_vertex_in_occupancy_flag[k] is not present, it is inferred to be equal to 0.

vme_multiple_lod_streams_present_flag[k] equal to 0 indicates that all lods for the atlas with atlas ID k are placed in a single vertex video stream, respectively.vme_multiple_lod_streams_present_flag[k] equal to 1 indicates that all lods for the atlas with atlas ID k are placed in separate video streams. When vme_multiple_lod_streams_present_flag[k] is not present, its value shall be inferred to be equal to 0.

vme_lod_absolute_coding_enabled_flag[j][i] equal to 1 indicates that the lod with index i for the atlas with atlas ID k is coded without any form of map prediction. vme_lod_absolute_coding_enabled_flag[k][i]equal to 0 indicates that the lod with index i for the atlas with atlas ID k is first predicted from another, earlier coded, map prior to coding. If vme_lod_absolute_coding_enabled_flag[j][i] is not present, its value shall be inferred to be equal to 1.

vme_lod_predictor_index_diff[k][i] is used to compute the predictor of the lod with index i for the atlas with atlas ID k when vps_map_absolute_coding_enabled_flag[j][i] is equal to 0. More specifically, the map predictor index for lod i, LodPredictorIndex[i], shall be computed as:

$$\text{LodPredictorIndex}[i] = (i - 1) - \text{vme\_lod\_predictor\_index\_diff}[j][i] \quad (15)$$

The value of vme_lod_predictor_index_diff[j][i] shall be in the range from 0 to i – 1, inclusive. When vme_lod_predictor_index_diff[j][i] is not present, its value shall be inferred to be equal to 0.

vme_vertex_video_present_flag[k] equal to 0 indicates that the atlas with ID k does not have vertex data. vms_vertex_video_present_flag[k] equal to 1 indicates that the atlas with ID k has vertex data. When vms_vertex_video_present_flag[j] is not present, it is inferred to be equal to 0.

Vertex information

vertex_information(atlasID) {	Descriptor
vi_vertex_codec_id[atlasID]	u(8)
vi_lossy_vertex_compression_threshold[atlasID]	u(8)
vi_vertex_2d_bit_depth_minus1[atlasID]	u(5)
vi_vertex_MSB_align_flag[atlasID]	u(1)
}	

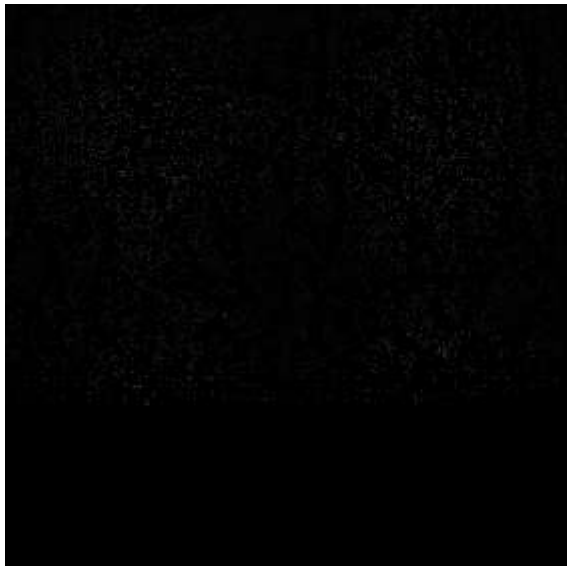
vi_vertex_codec_id[j] indicates the identifier of the codec used to compress the vertex information for the atlas with atlas ID j. vi_vertex_codec_id[j] shall be in the range of 0 to 255, inclusive. This codec may be identified through the profiles defined in Annex A, a component codec mapping SEI message, or through means outside this document.
vi_lossy_vertex_compression_threshold[j] indicates the threshold to be used to derive the binary vertex from the decoded vertex video for the atlas with atlas ID j. vi_lossy_vertex_compression_threshold[j] shall be in the range of 0 to 255, inclusive.
vi_vertex_2d_bit_depth_minus1[j] plus 1 indicates the nominal 2D bit depth to which the vertex video for the atlas with atlas ID j shall be converted to. vi_vertex_2d_bit_depth_minus1[j] shall be in the range of 0 to 31, inclusive.
vi_vertex_MSB_align_flag[j] indicates how the decoded vertex video samples associated with an atlas with atlas ID j are converted to samples at the nominal vertex bit depth, as specified in Annex B.

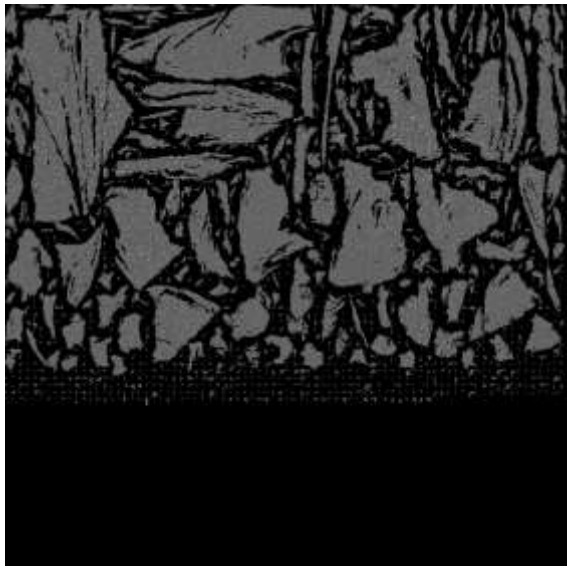
Mesh Patch Data Coding Options

- The syntax allows for 4 different types of vertex/connectivity coding
  1. [DIRECT] Sending the vertex and connectivity information directly in the patch
  2. [VERTEX VIDEO + CONNECTIVITY] Sending the vertex information in the **vertex map** (see definition in the next slide) and the face connectivity in the patch
  3. [VERTEX VIDEO ONLY] Sending the vertex information in the **vertex map** and deriving face connectivity at the decoder side (e.g., using ball pivoting or Poisson reconstruction)
  4. [EXTERNAL] Coding the vertex and connectivity information using an external mesh encoder (e.g., SC3DM, Draco, etc.)
     - Unlike regular 3D meshes, we need to encode a 2D mesh

Vertex Video Data

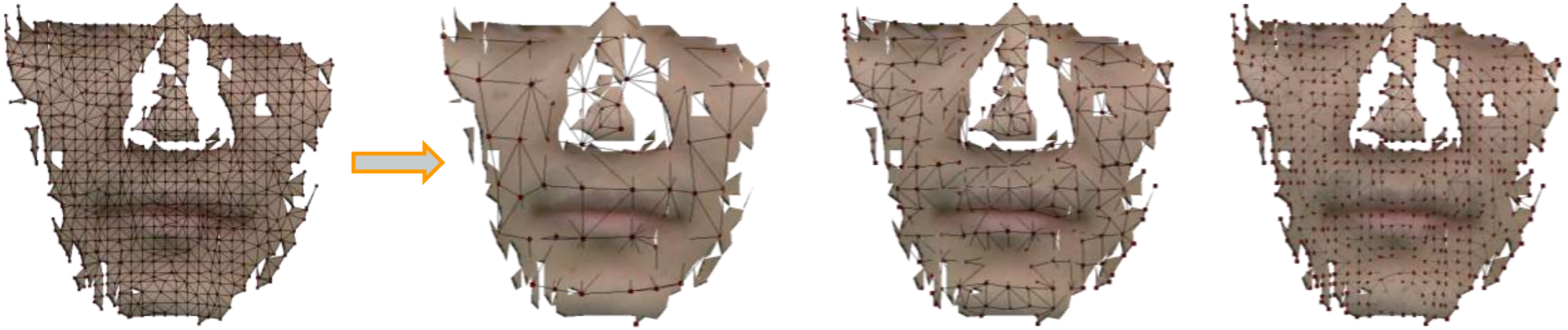
- Use multiple layers to indicate level-of-details. This is useful for progressive mesh coding
- In case only one layer is used, video data can be embedded in the occupancy map to avoid creation of several decoding instances
- Connectivity information can be generated using surface reconstruction algorithms (see Poisson Surface reconstruction and Ball Pivoting)





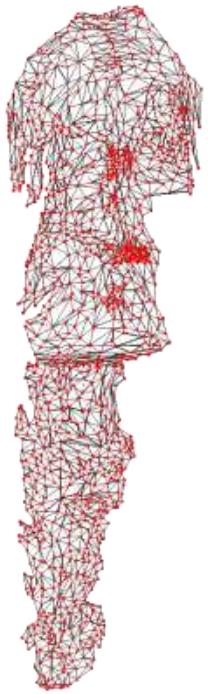
Level-of-detail generation

- Vertices can be combined in layers to generate the multi-level representation of the mesh
  - Generation of level-of-detail is specified in VPS

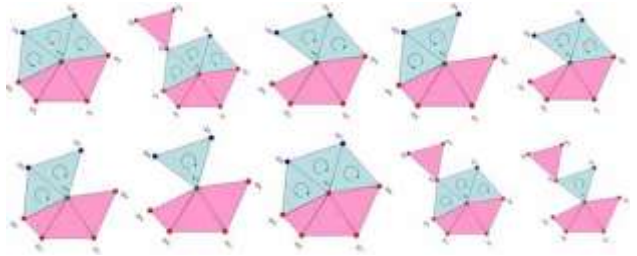


Binary Mesh Coding (MPEG)

- Use SC3DM (MPEG) to encode mesh information per patch
  - Use SC3DMC to encode connectivity and (u,v) information
    - SC3DMCEncoder.exe patch_sc3dm tfan.cfg



```
#VRML V2.0 utf8
Group {
  children [
    Shape {
      geometry IndexedFaceSet {
        ccw TRUE
        solid TRUE
        convex TRUE
        colorPerVertex TRUE
        normalPerVertex TRUE
        coord DEF co Coordinate {
          point [
            10.000000 11.000000 0.000000,
            5.000000 10.000000 0.000000,
            6.000000 13.000000 0.000000,
          ]
        }
        coordIndex [
            0, 1, 2, -1,
        ]
      }
    }
  ]
}
```

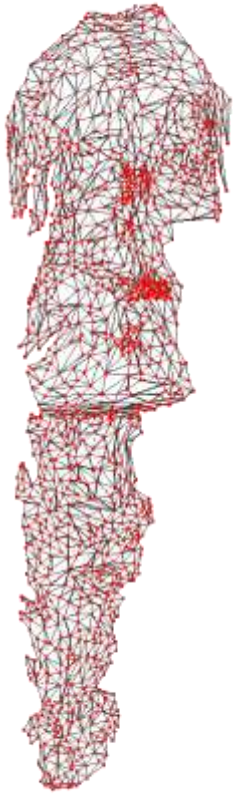


```
!! SC3DMC !!
(EncodingMode -> QBCR = 0  SVA = 1  TFAN = 2)
EncodingMode = 2
----------
(BinarizationMode -> FL = 0  BPC = 1  4C = 2 AC = 3
AC/EGC = 4)
BinarizationMode = 0
(QuantizationParameter -> 0 ~ 31 )
QuantizationParameter.Coord = 0
QuantizationParameter.Normal = 12
QuantizationParameter.Color = 12
QuantizationParameter.TexCoord = 12
QuantizationParameter.OtherAttributes = 12
----------
(For TexCoord -> Height , Width)
TexCoord.Height = 16
TexCoord.Width  = 16
----------
(PredictionMode -> No Prediction = 0, Differential = 1,
XOR = 2, Adaptive = 3, Circular =4, Fast Parallelogram =
5)
PredictionMode.Coord = 5
PredictionMode.Normal = 5
PredictionMode.Color = 5
PredictionMode.TexCoord = 0
PredictionMode.OtherAttribute = 5
PredictionMode.IndexCoord = 3
PredictionMode.IndexNormal = 3
PredictionMode.IndexColor = 3
PredictionMode.IndexTexCoord = 3
PredictionMode.IndexOtherAttribute = 3
(PredictionMode -> TRUE = 1 FALSE = 0 )
PredictionMode.VertexOrder = 0
PredictionMode.TriangleOrder = 0
```

Reconstructed object has issues because of (x,y) **lossy** coding of vertices
SC3DM header is also an issue (92 bytes/patch)

Binary Mesh Coding (Draco)

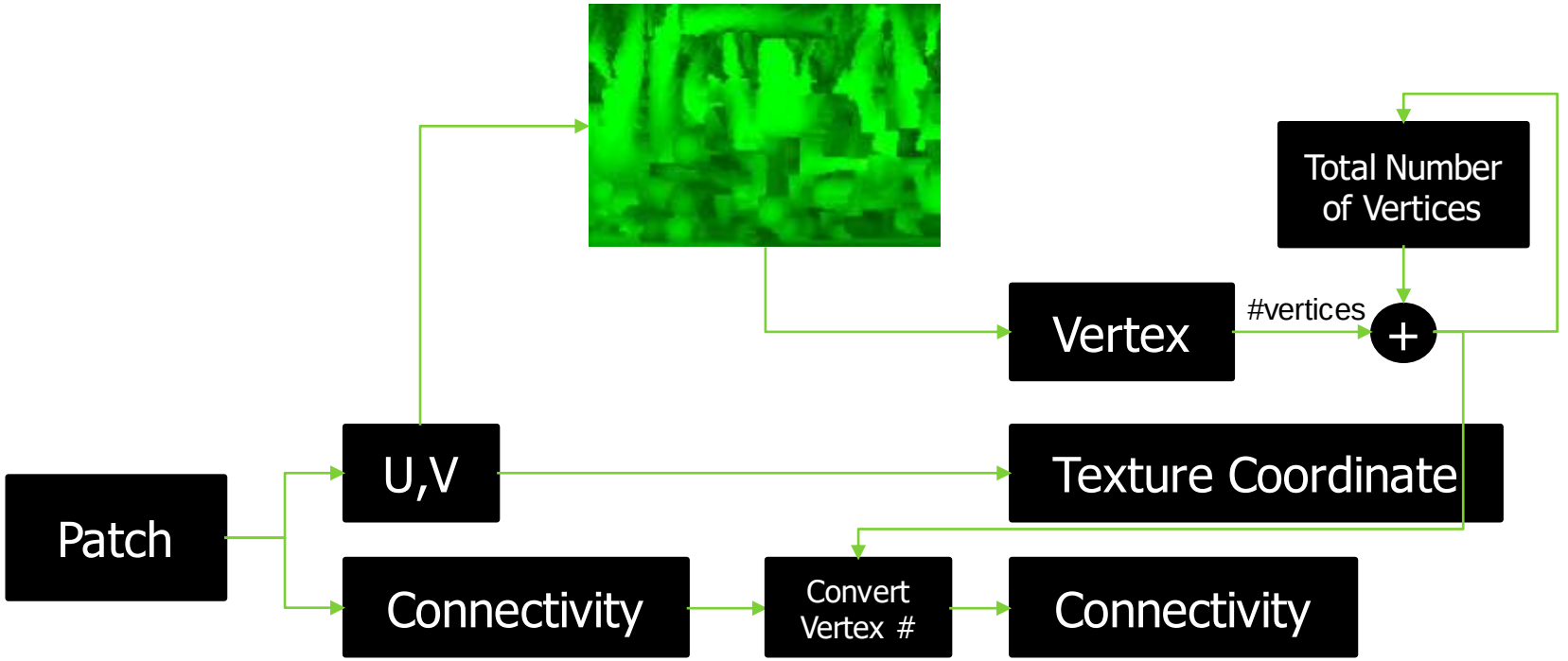
- Use Draco to encode mesh information per patch
  - Use Draco to encode connectivity and (u,v) information
    - draco_encoder.exe –i patch_#num.ply –o patch_#num.drc –qp 0 –cl 10
      - Ply has integer vertices, qp 0 indicates no quantization



```
ply
format ascii 1.0
element vertex #NUM_VERT
property int x
property int y
property int z
element face #NUM_FACE
property list uchar int vertex_indices
end_header
0 1 0
0 5 0
5 4 0
5 0 0
0 0 0
...
3 0 1 2
3 3 0 2
3 4 0 3
...
```

Mesh Reconstruction

- Connectivity will need to use the new vertex numbering, but patches can just be added together
  - Issue: Vertices at the seams may not match because of compression
    - See contribution m55374



<EXPERIMENTAL RESULTS>

- Patch coding

```
Bitstream stat:
  Header:                              16 B       128 b
    V3CUnitSize[ V3C_VPS ]:            31 B       248 b
    V3CUnitSize[ V3C_AD  ]:        199917 B   1599336 b
    V3CUnitSize[ V3C_OVD ]:         16118 B    128944 b
    V3CUnitSize[ V3C_GVD ]:         26709 B    213672 b
    V3CUnitSize[ V3C_AVD ]:        171713 B   1373704 b
    V3CUnitSize[ V3C_VVD ]:             0 B         0 b
  TotalMetadata:                   216098 B   1728784 b
  TotalGeometry:                    26701 B    213608 b
  TotalTexture:                    171705 B   1373640 b
  Total:                           414504 B   3316032 b
```

<EXPERIMENTAL RESULTS>

- VVD + patch connectivity

```
Bitstream stat:
  Header:                              19 B       152 b
    V3CUnitSize[ V3C_VPS ]:            36 B       288 b
    V3CUnitSize[ V3C_AD  ]:        148773 B   1190184 b
    V3CUnitSize[ V3C_OVD ]:         16118 B    128944 b
    V3CUnitSize[ V3C_GVD ]:         26709 B    213672 b
    V3CUnitSize[ V3C_AVD ]:        171713 B   1373704 b
    V3CUnitSize[ V3C_VVD ]:         31276 B    250208 b
TotalMetadata:                     196238 B   1569904 b
  TotalGeometry:                    26701 B    213608 b
  TotalTexture:                    171705 B   1373640 b
  Total:                           394644 B   3157152 b
```

<EXPERIMENTAL RESULTS>

- VVD + ball pivoting

```
Bitstream stat:
  Header:                              19 B       152 b
    V3CUnitSize[ V3C_VPS ]:            36 B       288 b
    V3CUnitSize[ V3C_AD  ]:          8470 B     67760 b
    V3CUnitSize[ V3C_OVD ]:         16118 B    128944 b
    V3CUnitSize[ V3C_GVD ]:         26709 B    213672 b
    V3CUnitSize[ V3C_AVD ]:        171713 B   1373704 b
    V3CUnitSize[ V3C_VVD ]:         31276 B    250208 b
TotalMetadata:                      55935 B    447480 b
  TotalGeometry:                     26701 B    213608 b
  TotalTexture:                     171705 B   1373640 b
  Total:                            254341 B   2034728 b
```

<EXPERIMENTAL RESULTS>

- SC3DM (MPEG)

```
Bitstream stat:
  Header:                              16 B       128 b
    V3CUnitSize[ V3C_VPS ]:            31 B       248 b
    V3CUnitSize[ V3C_AD  ]:         98393 B    787144 b
    V3CUnitSize[ V3C_OVD ]:         16118 B    128944 b
    V3CUnitSize[ V3C_GVD ]:         26709 B    213672 b
    V3CUnitSize[ V3C_AVD ]:        171713 B   1373704 b
    V3CUnitSize[ V3C_VVD ]:             0 B         0 b
  TotalMetadata:                   114574 B    916592 b
  TotalGeometry:                    26701 B    213608 b
  TotalTexture:                    171705 B   1373640 b
  Total:                           312980 B   2503840 b
```

<EXPERIMENTAL RESULTS>

- Draco

```
Bitstream stat:
  Header:                              16 B       128 b
    V3CUnitSize[ V3C_VPS ]:            31 B       248 b
    V3CUnitSize[ V3C_AD  ]:         66807 B    534456 b
    V3CUnitSize[ V3C_OVD ]:         16118 B    128944 b
    V3CUnitSize[ V3C_GVD ]:         26709 B    213672 b
    V3CUnitSize[ V3C_AVD ]:        171713 B   1373704 b
    V3CUnitSize[ V3C_VVD ]:             0 B         0 b
  TotalMetadata:                    82988 B    663904 b
  TotalGeometry:                    26701 B    213608 b
  TotalTexture:                    171705 B   1373640 b
  Total:                           281394 B   2251152 b
```

<EXPERIMENTAL RESULTS>

- Draco + Triangle Patch + Zippering (32 frames)

```
Bitstream stat:
  Header:                              16 B       128 b
    V3CUnitSize[ V3C_VPS ]:            31 B       248 b
    V3CUnitSize[ V3C_AD  ]:       2516188 B  20129504 b
    V3CUnitSize[ V3C_OVD ]:        624026 B   4992208 b
    V3CUnitSize[ V3C_GVD ]:        956195 B   7649560 b
    V3CUnitSize[ V3C_AVD ]:       2840888 B  22727104 b
    V3CUnitSize[ V3C_VVD ]:             0 B         0 b
TotalMetadata:                    3140277 B  25122216 b
  TotalGeometry:                   956187 B   7649496 b
  TotalTexture:                   2840880 B  22727040 b
  Total:                          6937344 B  55498752 b
```

Total bitstream size 6,937,344 B

~50 Mpbs

<EXPERIMENTAL RESULTS>

- Dense mesh results
  - Vertex position coded in occupancy map
  - Connectivity reconstructed using ball pivoting
  - Rate R5

```
Bitstream stat:
  Header:                              16 B       128 b
    V3CUnitSize[ V3C_VPS ]:            33 B       264 b
    V3CUnitSize[ V3C_AD  ]:          3306 B     26448 b
    V3CUnitSize[ V3C_OVD ]:         50530 B    404240 b
    V3CUnitSize[ V3C_GVD ]:        485275 B   3882200 b
    V3CUnitSize[ V3C_AVD ]:       1138279 B   9106232 b
    V3CUnitSize[ V3C_VVD ]:             0 B         0 b
  TotalMetadata:                    53901 B    431208 b
  TotalGeometry:                   485267 B   3882136 b
  TotalTexture:                   1138271 B   9106168 b
  Total:                          1677439 B  13419512 b
```

Total bitstream size 1,677,439 B

Ball pivoting reconstruction may generate triangles with wrong winding

Holes remaining (room for improvement in automatic connectivity generation)

<EXPERIMENTAL RESULTS>

- Dense mesh results
  - Vertex position coded in occupancy map
  - Connectivity reconstructed using ball pivoting
  - Rate R5

```
Bitstream stat:
  Header:                              16 B       128 b
    V3CUnitSize[ V3C_VPS ]:            31 B       248 b
    V3CUnitSize[ V3C_AD  ]:        518792 B   4150336 b
    V3CUnitSize[ V3C_OVD ]:         11975 B     95800 b
    V3CUnitSize[ V3C_GVD ]:        485275 B   3882200 b
    V3CUnitSize[ V3C_AVD ]:       1138279 B   9106232 b
    V3CUnitSize[ V3C_VVD ]:             0 B         0 b
  TotalMetadata:                   530830 B   4246640 b
  TotalGeometry:                   485267 B   3882136 b
  TotalTexture:                   1138271 B   9106168 b
  Total:                          2154368 B  17234944 b
```

Total bitstream size 2,154,368 B

Holes from lossy triangle coding (auxiliary video coding and triangle strips could help)

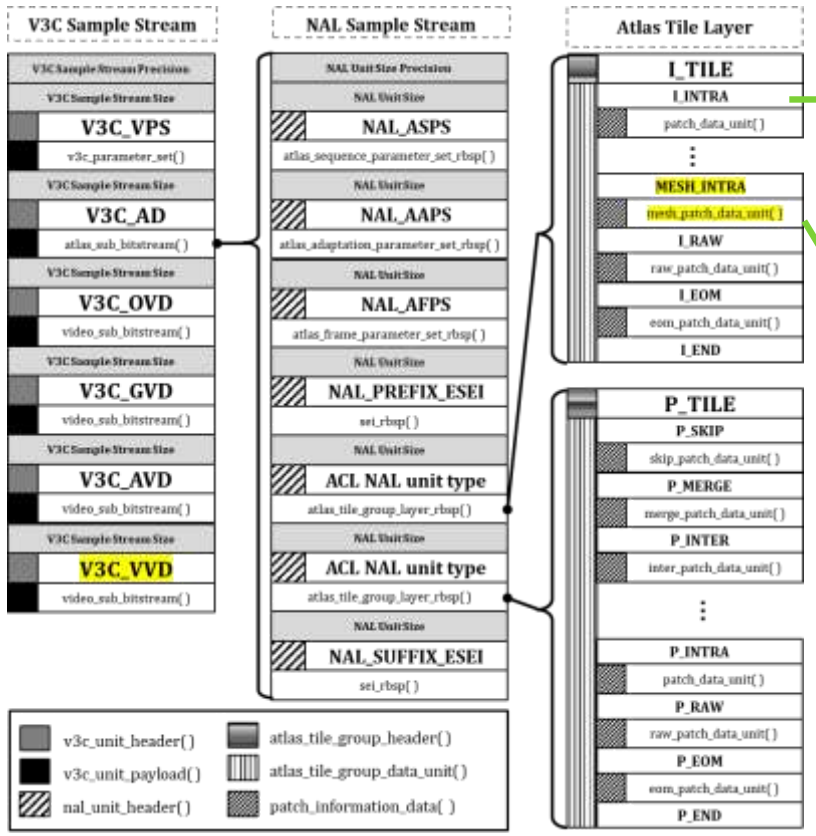
<Conclusion>

- We have shown that meshes can be encoded following the same paradigm stablished in V3C, to project volumetric content.
- The projection of meshes open several new possibilities to develop tool for triangle content (similar to what happened for point cloud)
- We would like the group to consider the new mesh patch data for mesh encoding

Appendix

Combining Point Cloud and Mesh information

- The high-level syntax allows to send a mixture of point cloud and mesh patches. For example, point cloud only patches could be used for hair, while mesh patches could be used for flat areas



Head is a point cloud

Body is a mesh