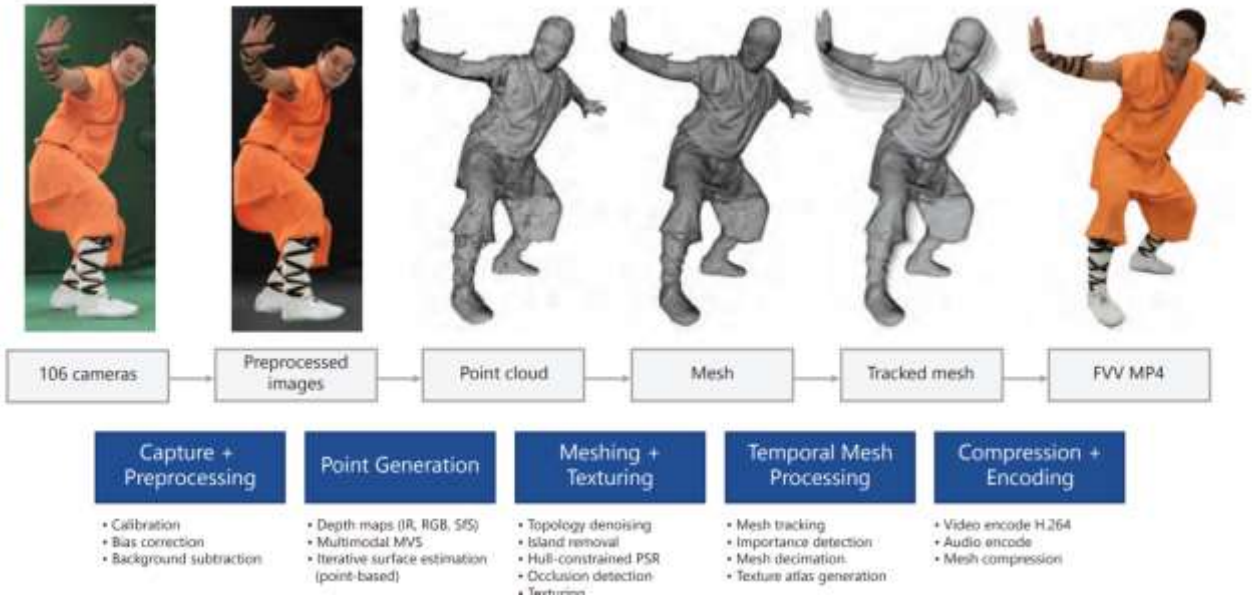
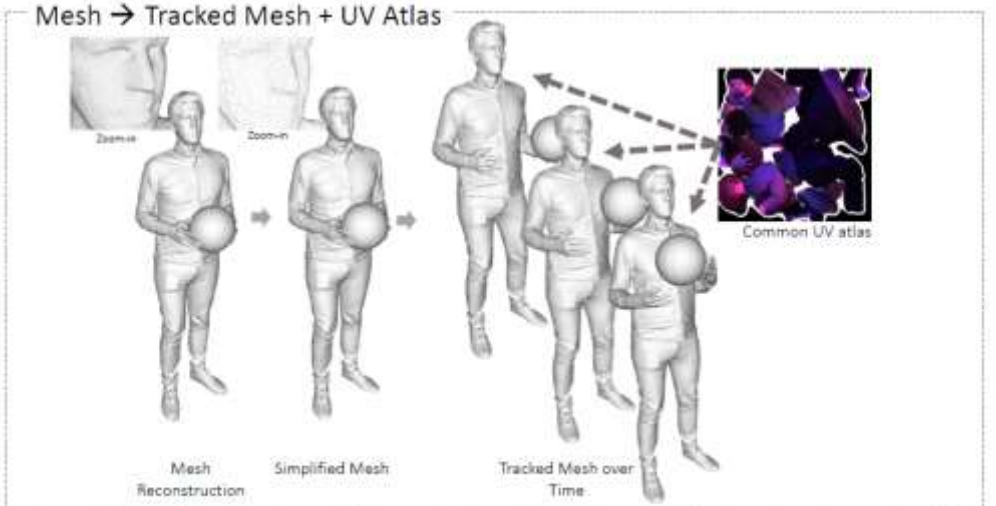
<m55372> Tracked Patch Data

Danillo Graziosi, Alexandre Zaghetto, and Ali Tabatabai

<Problem statement>

- Mesh tracking is part of several volumetric capture pipeline.
- Provide higher compression, since the mesh connectivity does not need to be sent every frame.
  - Nevertheless, vertex motion and possibly surface texture may be updated every frame, generating an increased amount of data for coding.
  - Furthermore, mesh tracking may suffer with topology changes, and schemes to “reset” the tracking by introducing keyframes are commonly used.





The Relightables Pipeline

<PROPOSAL>

- Tracked Mesh Patch Data Unit
  - We propose to define a new patch type that indicates patches that are tracked, or "matched" to patches in previous frames.
    - TMC2 already has INTER_PATCH, but they only indicate match of 2D atlas parameters. The proposed patch extends the functionality of INTER patches to indicate match of connectivity parameters as well.
  - In the case of tracked patches, the connectivity doesn't change from one frame to another, but the position of vertices may change due to global motion or surface motion.
    - The global motion is modeled by the 3D bounding box position (offset U,V,D) and rotation (newly introduced syntax elements using quaternions)
    - The surface motion can either be explicitly sent in the patch information as delta positions, or derived from the video data, in case the reference patch is using the vertex video data (V3C_VVD, see contribution m55368 [4]).

TRACKED PATCH DATA UNIT SYNTAX

tracked_mesh_patch_data_unit(tileID, patchIdx) {	**Descriptor**
if(NumRefIdxActive > 1){	
tmpdu_ref_index[tileID][patchIdx]	ue(v)
RefIdx = **tmpdu_ref_index**[tileID][patchIdx]	
} else	
RefIdx = 0	
tmpdu_patch_index[tileID][patchIdx]	se(v)
tmpdu_2d_pos_x[tileID][patchIdx]	se(v)
tmpdu_2d_pos_y[tileID][patchIdx]	se(v)
tmpdu_2d_delta_size_x[tileID][patchIdx]	se(v)
tmpdu_2d_delta_size_y[tileID][patchIdx]	se(v)
tmpdu_3d_offset_u[tileID][patchIdx]	se(v)
tmpdu_3d_offset_v[tileID][patchIdx]	se(v)
tmpdu_3d_offset_d[tileID][patchIdx]	se(v)
if(asps_normal_axis_max_delta_value_enabled_flag)	
ipdu_3d_range_d[tileID][patchIdx]	se(v)
tmpdu_rotation_present_flag[tileID][patchIdx]	u(1)
if(tmpdu_rotation_present_flag[tileID][patchIdx]) {	
tmpdu_3d_rotation_qx[tileID][patchIdx]	i(16)
tmpdu_3d_rotation_qy[tileID] [patchIdx]	i(16)
tmpdu_3d_rotation_qz[tileID] [patchIdx]	i(16)
}	
tmpdu_connectivity_changed_flag[tileID][patchIdx]	u(1)
if(!tmpdu_connectivity_changed_flag[tileID][patchIdx]) {	
if(!asps_vertices_in_vertex_video_data[tileID][patchIdx]){	
tmpdu_vertices_change_position_flag[tileID][patchIdx]	u(1)
if(tmpdu_vertices_change_position_flag){	
for(i = 0; i < VertexCount[tileId][RefIdx] ; i++) {	
tmpdu_vertex_delta_pos_x[tileID][patchIdx][i]	se(v)
tmpdu_vertex_delta_pos_y[tileID][patchIdx][i]	se(v)
}	
}	
}	
} else {	

if(asps_mesh_binary_coding_enabled_flag)	
tmpdu_binary_object_present_flag[tileID][patchIdx]	u(1)
if(tmpdu_binary_object_present_flag[tileID][patchIdx]) {	
tmpdu_mesh_binary_object_size_bytes[tileID][patchIdx]	ue(v)
for(i = 0; i < tmpdu_mesh_payload_size_bytes[tileID][patchIdx]; i++)	
tmpdu_mesh_binary_object[tileID][patchIdx][i]	b(8)
} else {	
tmpdu_vertex_count_minus3[tileID][patchIdx]	ue(v)
tmpdu_face_count[tileID][patchIdx]	ue(v)
for(i = 0; i < tmpdu_faces_count[tileID][patchIdx]; i++) {	
tmpdu_face_vertex[tileID][patchIdx][i][0]	u(v)
tmpdu_face_vertex[tileID][patchIdx][i][1]	u(v)
tmpdu_face_vertex[tileID][patchIdx][i][2]	u(v)
if(asps_mesh_quad_face_flag) {	
tmpdu_face_vertex[tileID][patchIdx][i][3]	u(v)
}	
if(!asps_mesh_vertices_in_vertex_video_data) {	
for(i = 0; i < tmpdu_vertex_count_minus3[tileID][patchIdx] +3 ; i++) {	
tmpdu_vertex_pos_x[tileID][patchIdx][i]	u(v)
tmpdu_vertex_pos_y[tileID][patchIdx][i]	u(v)
}	
}	
}	
}	

TRACK PATCH DATA UNIT SEMANTICS

tmpdu_rotation_present_flag[t][p] equal to 1 indicates that rotation parameters for the patch with index p and tile with tile ID t are present. tmpdu_rotation_present_flag[t][p] equal to 0 indicates that rotation parameters for the patch patch with indexp and tile with tile ID t are not present. When tmpdu_rotation_present_flag[t][p] is not present, it shall be inferred to be equal to 0.

tmpdu_3d_rotation_qx[t][p] specifies the x component, qX, for the geometry rotation of the patch with index p and tile with tile ID t using the quaternion representation. The value of tmpdu_3d_rotation_qx[t][p] shall be in the range of -2^{14} to $2^{14} - 1$, inclusive. When tmpdu_3d_rotation_qx[t][p] is not present, its value shall be inferred to be equal to 0. The value of qX is computed as follows:

qX = tmpdu_3d_rotation_qx ÷ 2^{14}

tmpdu_3d_rotation_qy[t][p] specifies the y component, qY, for the geometry rotation of the patch with index p and tile with tile ID t using the quaternion representation. The value of tmpdu_3d_rotation_qy[t][p] shall be in the range of -2^{14} to $2^{14} - 1$, inclusive. When tmpdu_3d_rotation_qy[t][p] is not present, its value shall be inferred to be equal to 0. The value of qY is computed as follows:

qY = tmpdu_3d_rotation_qy ÷ 2^{14}

tmpdu_3d_rotation_qz[t][p] specifies the z component, qZ, for the geometry rotation of the patch with index p and tile with tile ID t using the quaternion representation. The value of tmpdu_3d_rotation_qz[t][p] shall be in the range of -2^{14} to $2^{14} - 1$, inclusive. When tmpdu_3d_rotation_qz[t][p] is not present, its value shall be inferred to be equal to 0. The value of qZ is computed as follows:

qZ = tmpdu_3d_rotation_qz ÷ 2^{14}

The fourth component, qW, for the geometry rotation of the patch with indexp and tile with tile ID t using the quaternion representation is calculated as follows:

qW = Sqrt(1 – (qX^2 + qY^2 + qZ^2))

A unit quaternion can be represented as a rotation matrix R as follow:

$$\text{RotationMatrix} = \begin{bmatrix} 1-2*(qY^2+qZ^2) & 2*(qX*qY-qZ*qW) & 2*(qX*qZ+qY*qW) & 0 \\ 2*(qX*qY+qZ*qW) & 1-2*(qX^2+qZ^2) & 2*(qY*qZ-qX*qW) & 0 \\ 2*(qX*qZ-qY*qW) & 2*(qY*qZ+qX*qW) & 1-2*(qX^2+qY^2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

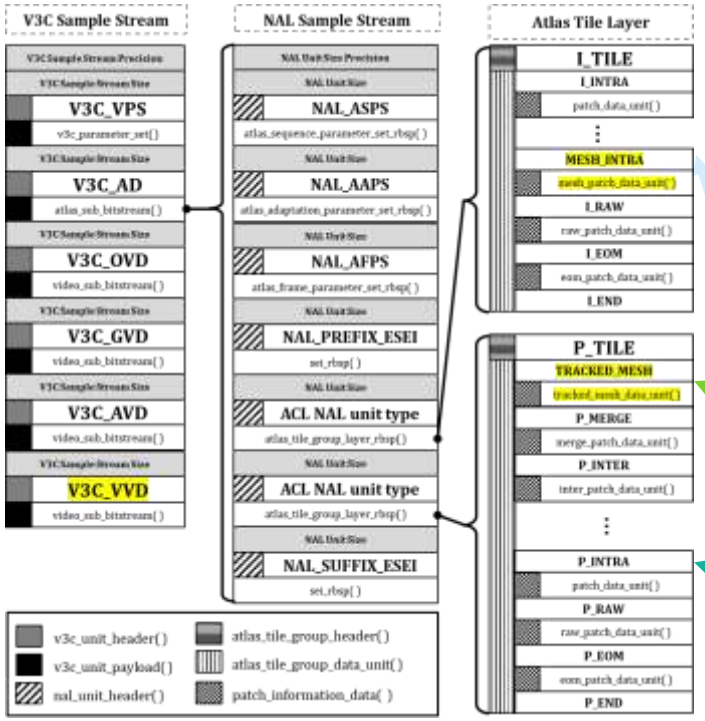
tmpdu_vertices_changed_position_flag[t][p] equal to 1 indicates the vertices displacement for the patch with index p and tile with tile ID t are present. tmpdu_vertices_changed_position_flag[t][p] equal to 0 indicates that the vertices displacement for the patch with indexp and tile with tile ID t are not present. When tmpdu_vertices_changed_position_flag[t][p] is not present, it shall be inferred to be equal to 0.

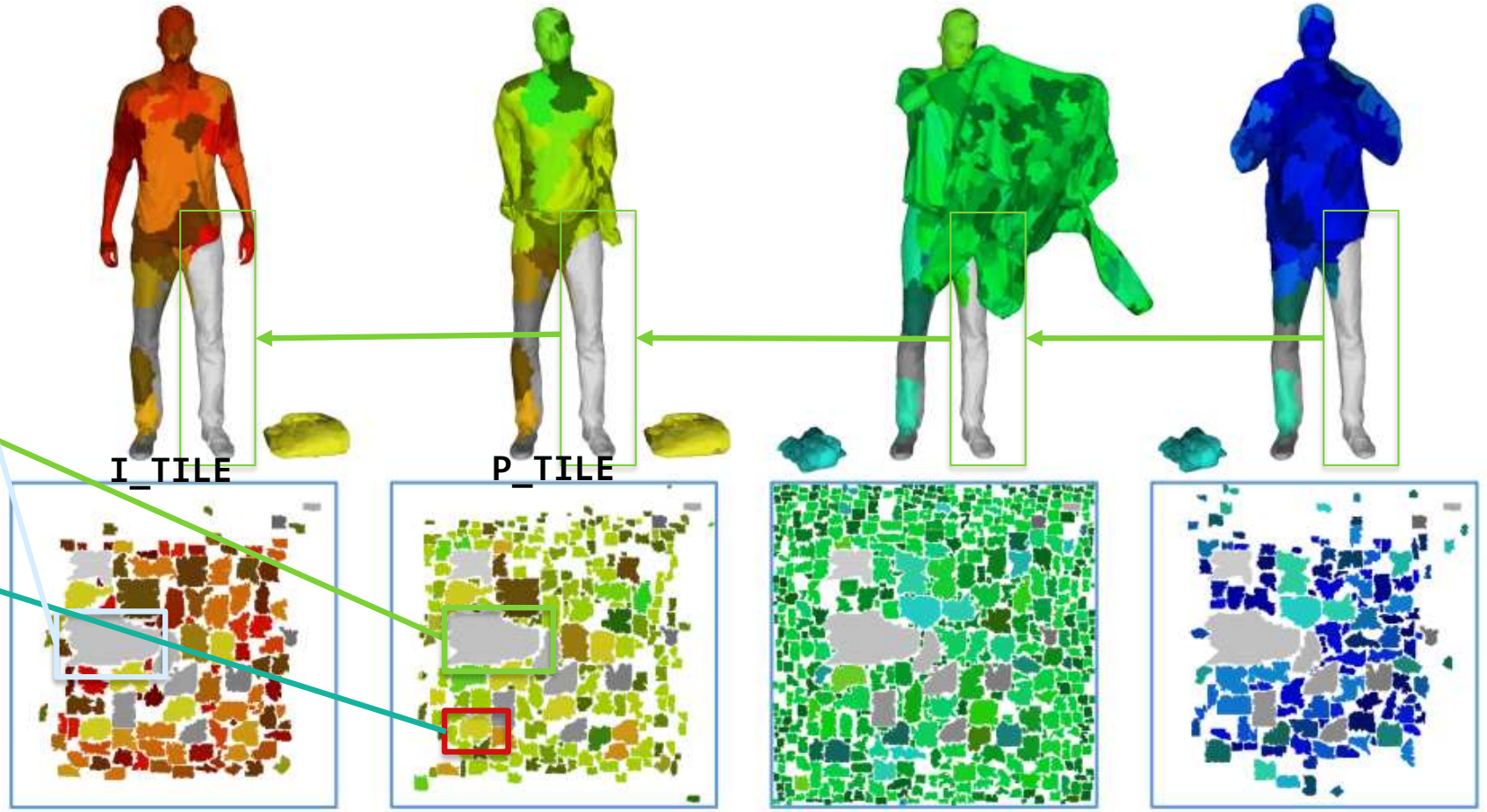
tmpdu_vertex_delta_pos_x[t][p][i] specifies the difference of the x-coordinate values of the i-th vertex of patch with index p and tile with tile ID t and the matched patch indicated by tmpdu_ref_index[t][p]. The value of tmpdu_vertex_pos_x[t][p][i] shall be in the range of 0 to $2^{\text{afps\_num\_bits\_vertex\_delta\_x}}$ - 1, inclusive.

tmpdu_vertex_delta_pos_y[t][p][i] specifies the difference of the y-coordinate values of the i-th vertex of patch with index p and tile with tile ID t and the matched patch indicated by tmpdu_ref_index[t][p]. The value of tmpdu_vertex_pos_y[t][p][i] shall be in the range of 0 to $2^{\text{afps\_num\_bits\_vertex\_delta\_y}}$ - 1, inclusive.

Combining Untracked and Tracked Mesh information

- "Spatiotemporal Atlas Parameterization for Evolving Meshes" by Prada *et al*) segment the mesh into tracked parts and untracked parts.
  - Tracked parts are consistent in time and can be represented by the proposed **tracked_mesh_patch_data_unit(),**
  - Untracked parts are new each frame and can be represented by **mesh_patch_data_unit().**





<EXPERIMENTAL RESULTS>

- Under development → let's get tracked content for our dataset!!!

<Conclusion>

- We believe that tracked meshes are an important dataset that should also be consider for the MPEG mesh coding
- The proposed tracked patch data unit follows the philosophy of V3C and allows for the combination of several different data types: tracked meshes, untracked meshes, and even point clouds.
- We suggest to the group to consider tracked meshes as a future use case for V3C mesh compression, and further investigate the proposed tracked patch data unit.