<m55370> Triangle Patch Data

Danillo Graziosi, Alexandre Zaghetto, and Ali Tabatabai

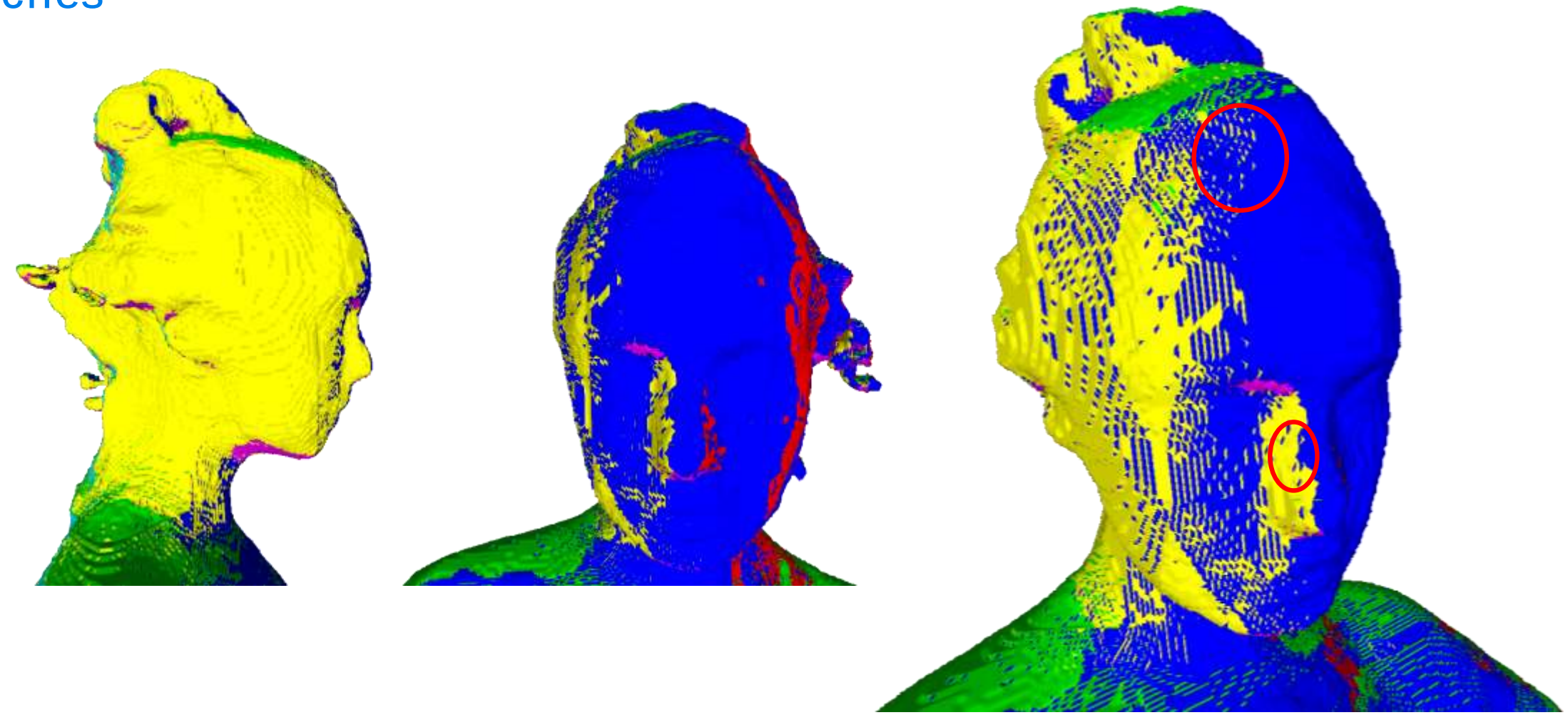
<Problem statement>

- Coding of meshes using projections may have some limitations. For instance, the projection of dense meshes creates many overlapped triangles, that will need to be coded in separate patches. This increases the number of patches and the image size, since each patch might have very small area (maybe 3 pixels only), but the block packing uses at least N x N pixels.

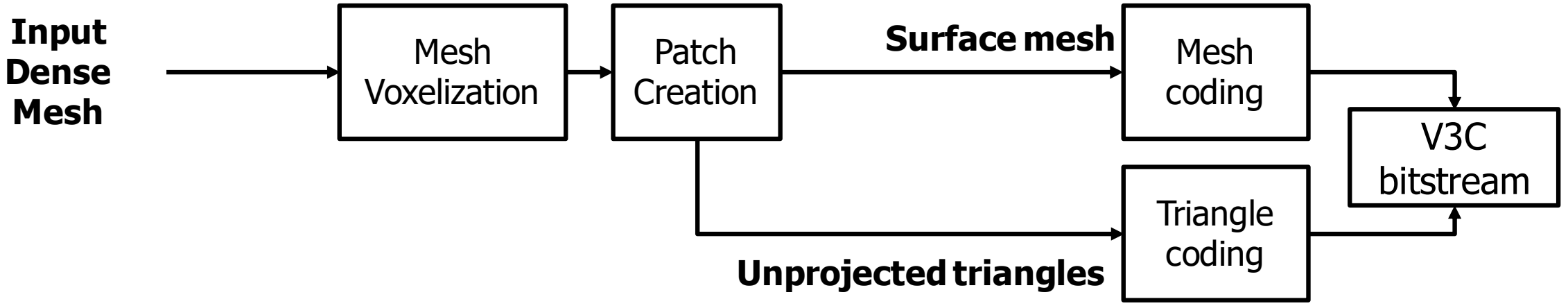
- In this contribution, we propose a new patch for coding of isolated triangles called TRIANGLE PATCH DATA. Similar to the RAW patch, we encode the vertex position directly on the video signal. Syntax elements in the triangle patch data indicate how to reconstruct the mesh connectivity.

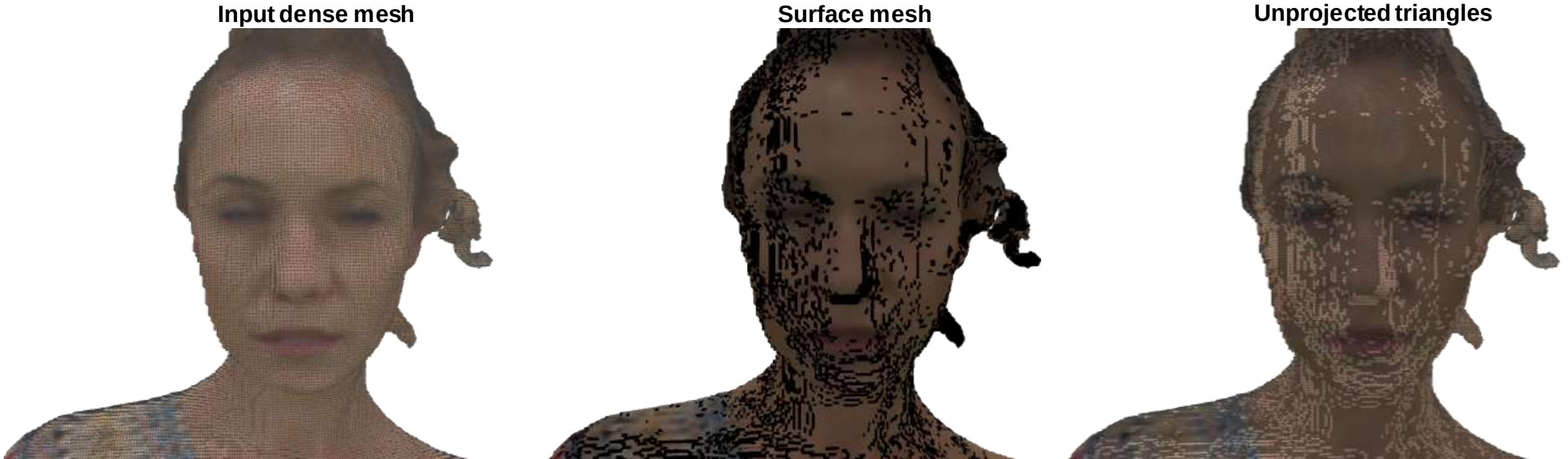
Dense mesh projection issue

- Not all triangles have the same classification, and are projected in patches



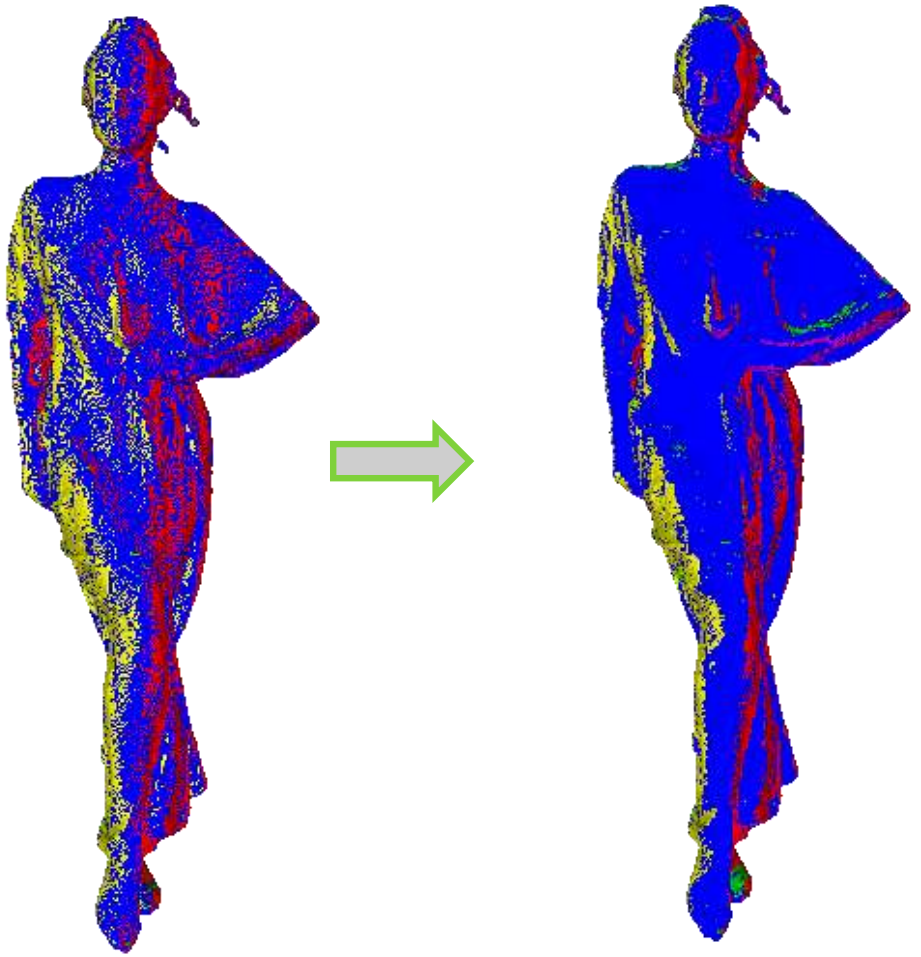
<NEW PROPOSAL: TRIANGLE PATCH >





Patch creation (1/2)

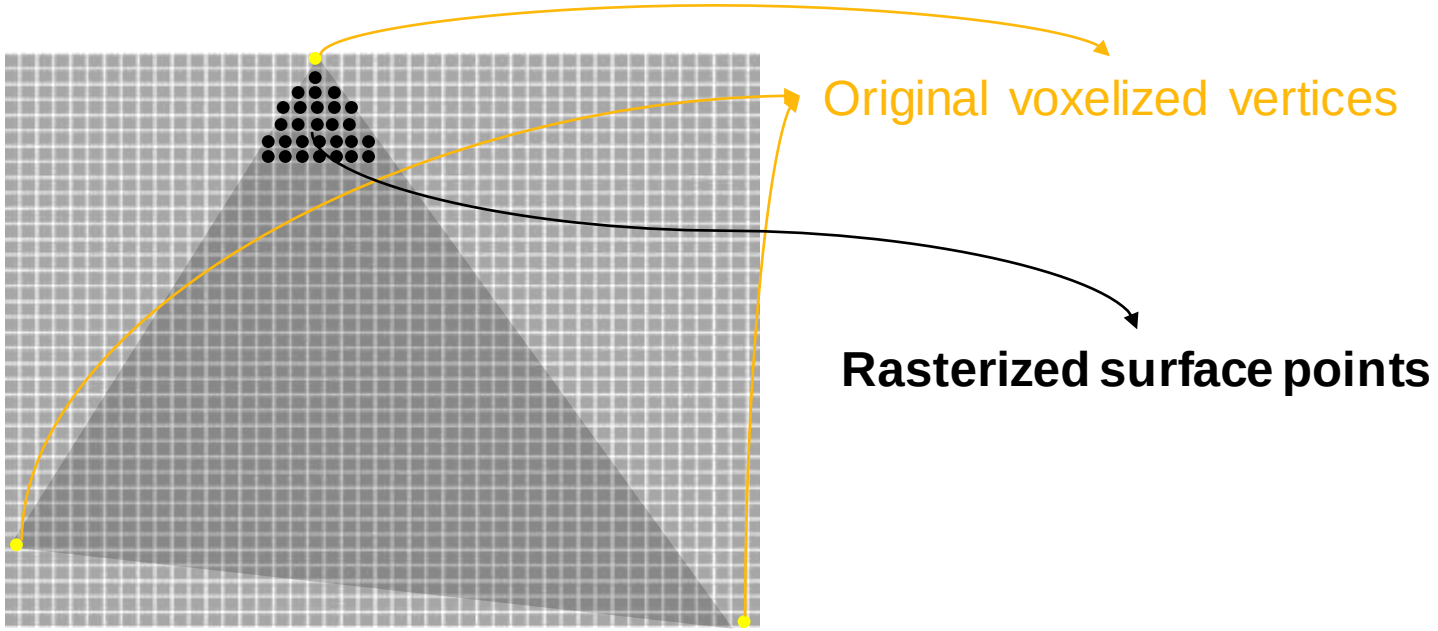
- Determine adjacency neighborhood of each triangle
- Calculate normal of each triangle
- Categorize the triangle according to the normal
- Refine category according to neighboring values



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)



Triangle Patch Data Unit syntax/semantics

triangle_patch_data_unit(tileID, patchIdx) {	**Descriptor**
if(AuxTileHeight[TileIDToIndex[tileID]] > 0)	
tpdu_patch_in_auxiliary_video_flag[tileID][patchIdx]	u(1)
tpdu_2d_pos_x[tileID][patchIdx]	ue(v)
tpdu_2d_pos_y[tileID][patchIdx]	ue(v)
tpdu_2d_size_x_minus1[tileID][patchIdx]	ue(v)
tpdu_2d_size_y_minus1[tileID][patchIdx]	ue(v)
tpdu_3d_offset_u[tileID][patchIdx]	u(v)
tpdu_3d_offset_v[tileID][patchIdx]	u(v)
tpdu_3d_offset_d[tileID][patchIdx]	u(v)
tpdu_vertices_minus3[tileID][patchIdx]	ue(v)
tpdu_primitive_idc[tileID][patchIdx]	u(8)
tpdu_color_expansion_flag[tileID][patchIdx]	u(1)
}	

tpdu_vertices_minus3[tileID][p] plus 3 specifies the number of vertices present in the TRIANGLE coded patch with index p in the current atlas tile, with tile ID equal to tileID. The value of tpdu_vertices_minus3[tileID][p] shall be in the range of 0 to ((tpdu_2d_size_x_minus1[tileID][p] + 1) * (tpdu_2d_size_y_minus1[tileID][p] + 1)) / 3 − 3, inclusive.

tpdu_primitive_idc[tileID][p] indicates the geometry primitive that defines how triangles are obtained from the vertices present in the coded patch with index p in the current atlas tile, with tile ID equal to tileID. If tpdu_primitive_idc[tileID][p] is not present, its value shall be inferred to be equal to 0.

tpdu_color_expansion_flag[tileID][p] equal to 1 specifies that the coordinates of the vertices are packed line-interleaved, and the color values are expanded for the current patch p of the current atlas tile, with tile ID equal to tileID. If tpducolor_expansion_flag[tileID][p] is equal to 0, the coordinates of the vertices are packed sequentially, and the color is not expanded for the current patch. If tpdu_color_expansion_flag[tileID][p] is not present, its value shall be inferred to be equal to 0.

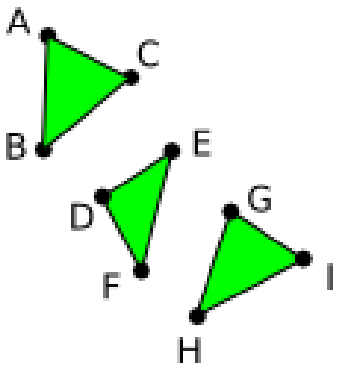
Triangle Patch Data

- List of points in patch are the vertices of the triangles, which are formed according to the primitive IDC.

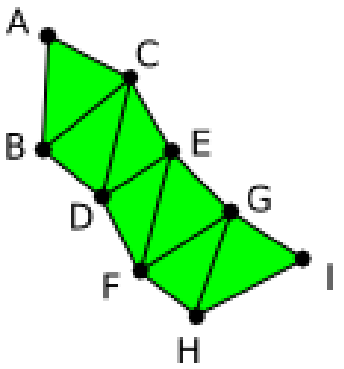
(0) **separate triangles**:
- (0 1 2),(3 4 5),(6 7 8)

(1) **triangle strip**:
- (0 1 2),(2 1 3),(2 3 4)

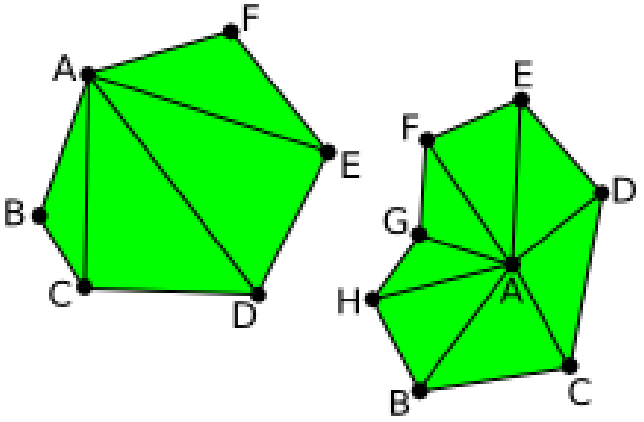
(2) **triangle fan**:
- (0 1 2), (0 2 3), (0 3 4)



GL_TRIANGLES



GL_TRIANGLE_STRIP

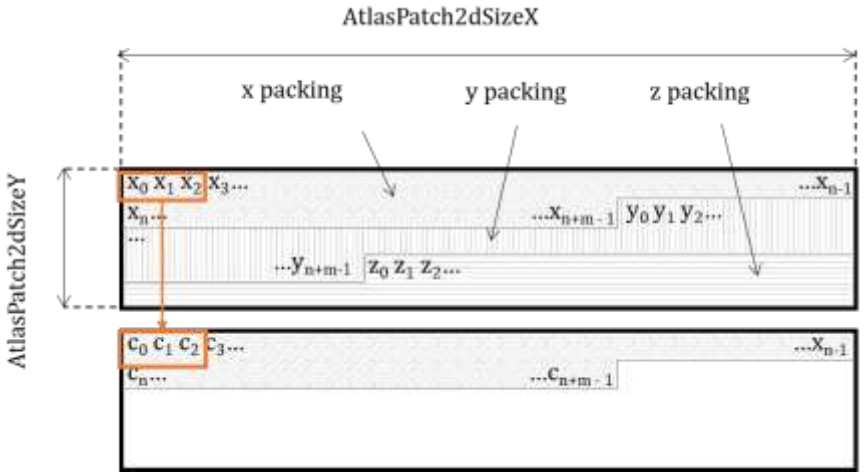


GL_TRIANGLE_FAN

OBS: It is possible to create triangle strips of unconnected triangles by vertex repetition

Triangle Patch Data

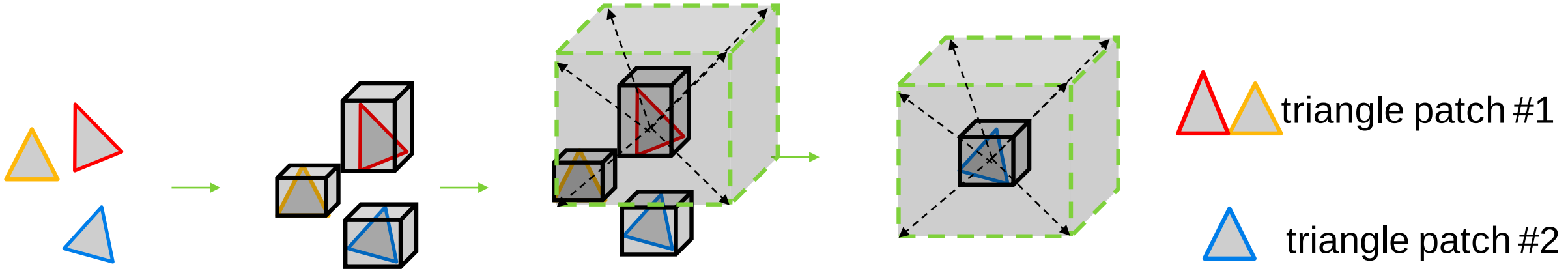
- Two methods for packing data
  - (0) **Component packing**
    - similar packing to RAW patches
  - (1) **Line interleaved**
    - a line is assigned for each component. This option also indicates that the color will be expanded. This might be necessary for textured meshes, because of UV interpolation artifacts (see next slides).



How to collect triangles for the patch

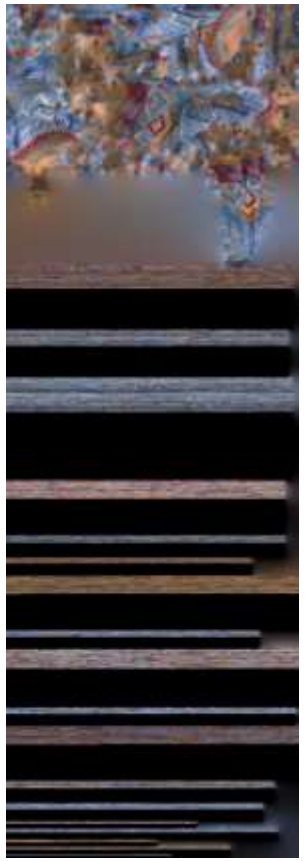
Here we describe a method to create triangle patches from disconnected triangles. The method also guarantees that the bit depth used to then represent the vertex is within a certain value (for example, segmenting 10 bit data into 8 bit segments)

1. Calculate the bounding box of the triangles, if the size of the bounding box is larger than the pre-defined bit depth, then the triangle is removed and not coded
2. While the list of triangles is not empty, do
    1. Create a "selection bounding box" with pre-defined size (2^numBits-1) centered at the center of the bounding box of the first triangle
    2. Now loop over all the triangles and for each triangle, check if the selection bounding box contains the triangle's bounding box. If it contains, the triangle is added to the triangle patch and removed from the list of triangles.



EXPERIMENTAL RESULTS <1/4>

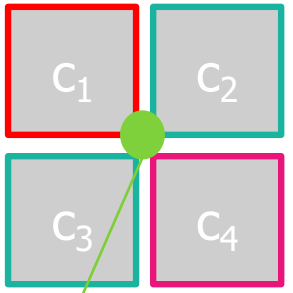
- Lossless coding of dense meshes without using color expansion



4,463,742 B

UV Texture

Color Artifacts due to UV interpolation



$$\mathrm{RGB}(u,v) = \sum_{n=1}^{4}(c_n * w_n)$$

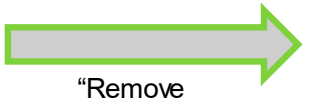
Using Vertex Color

```
numberOfProjectedPatches 298
numberOfTrianglePatches = 18
 # resampled 617338
 # projected triangles 938286
 # raw triangles 345337
 # degenerate triangles 0
 # missed triangles 0
```

Bitstream stat:

Header:	16 B	128 b
V3CUnitSize[V3C_VPS]:	31 B	248 b
V3CUnitSize[V3C_AD]:	6913415 B	55307320 b
V3CUnitSize[V3C_OVD]:	5005 B	40040 b
V3CUnitSize[V3C_GVD]:	1443815 B	11550520 b
V3CUnitSize[V3C_AVD]:	4463742 B	35709936 b
TotalMetadata:	6918483 B	55347864 b
TotalGeometry:	1443807 B	11550456 b
TotalTexture:	4463734 B	35709872 b
Total:	12826024 B	102608192 b

1,653,056 vertices
1,283,623 faces



"Remove duplicate vertices"

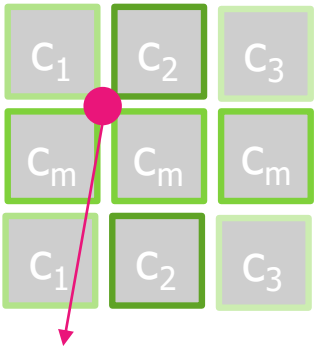
641,829 vertices
1,283,623 faces

EXPERIMENTAL RESULTS <2/4>

- Lossless coding of dense meshes using with color expansion

7,783,919 B

UV Texture



$$\mathrm{RGB}(u,v) = \sum_{n=1}^{4}(c_n * w_n)$$

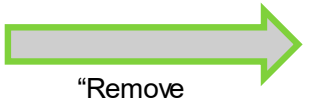
Using Vertex Color

```
numberOfProjectedPatches 298
numberOfTrianglePatches = 18
 # resampled 617338
 # projected triangles 938286
 # raw triangles 345337
 # degenerate triangles 0
 # missed triangles 0
```

Bitstream stat:

Header:	16 B	128 b
V3CUnitSize[V3C_VPS]:	31 B	248 b
V3CUnitSize[V3C_AD]:	6913416 B	55307328 b
V3CUnitSize[V3C_OVD]:	5005 B	40040 b
V3CUnitSize[V3C_GVD]:	1466361 B	11730888 b
V3CUnitSize[V3C_AVD]:	7783919 B	62271352 b
TotalMetadata:	6918484 B	55347872 b
TotalGeometry:	1466353 B	11730824 b
TotalTexture:	7783911 B	62271288 b
Total:	16168748 B	129349984 b

1,653,056 vertices
1,283,623 faces



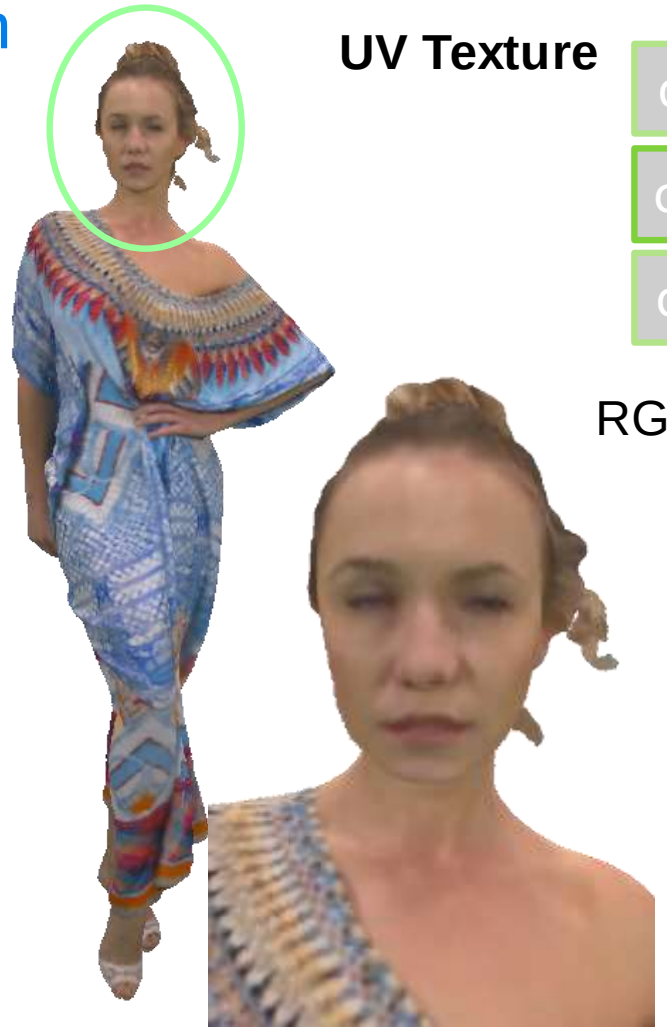
"Remove duplicate vertices"

641,829 vertices
1,283,623 faces

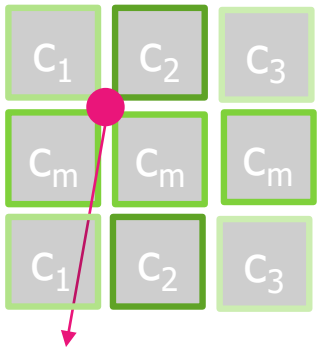
EXPERIMENTAL RESULTS <3/4>

- Lossy texture coding (420, r5) of dense meshes using with color expansion

1,138,271 B



UV Texture

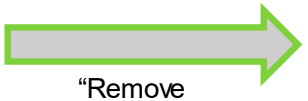


$$\mathrm{RGB}(u,v) = \sum_{n=1}^{4}(c_n * w_n)$$

Using Vertex Color

```
                    Bitstream stat:
 Header:                                16 B       128 b
    V3CUnitSize[ V3C_VPS ]:             31 B       248 b
    V3CUnitSize[ V3C_AD  ]:        6913416 B  55307328 b
    V3CUnitSize[ V3C_OVD ]:          11975 B     95800 b
    V3CUnitSize[ V3C_GVD ]:        1465757 B  11726056 b
    V3CUnitSize[ V3C_AVD ]:        1138279 B   9106232 b
 TotalMetadata:                    6925454 B  55403632 b
 TotalGeometry:                    1465749 B  11725992 b
 TotalTexture:                     1138271 B   9106168 b
 Total:                            9529474 B  76235792 b
Total bitstream size 9529474 B16168748 B 129349984 b
```

1,653,056 vertices
1,283,623 faces



"Remove duplicate vertices"

641,829 vertices
1,283,623 faces

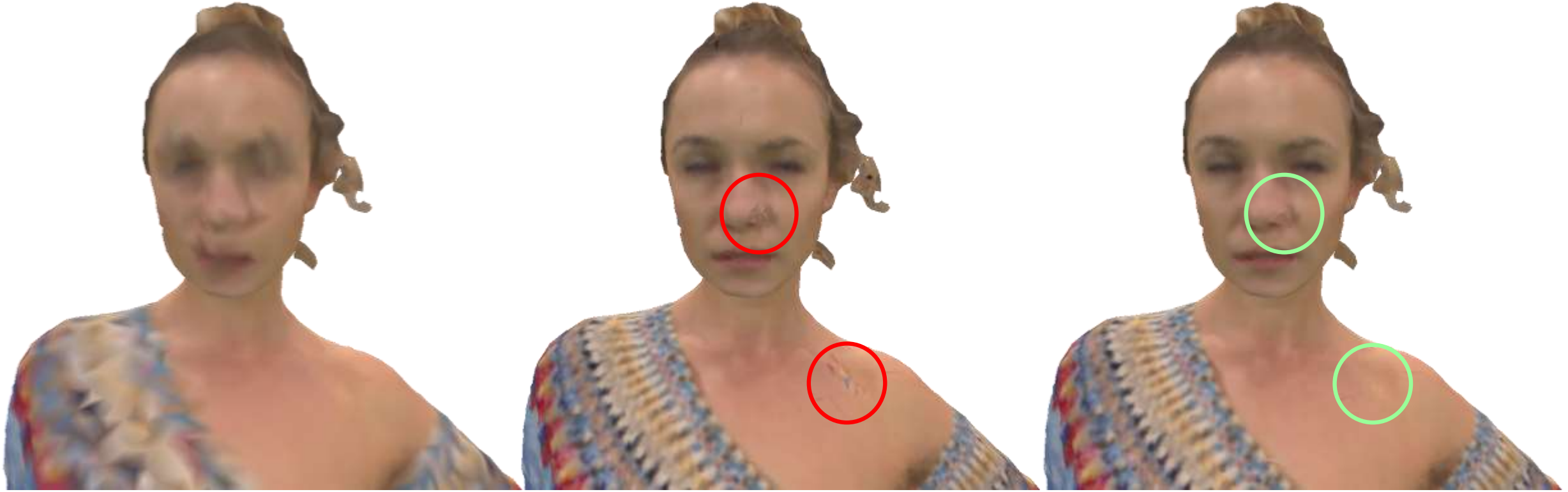
EXPERIMENTAL RESULTS <4/4>

- Triangle patch applied to sparse meshes
  - The triangle patch can be applied to sparse meshes, but the color suffers, since the whole surface will be represented by the interpolation of three colors. However, the color technique expansion can be used to improve texture quality.

Vertex color

Triangle patch without expansion

Triangle patch with expansion



<Conclusion>

- We described here a tool for mesh coding that is suitable for smaller triangles with little color variation between the vertices, commonly the case for dense meshes.
- The new proposed patch has characteristics similar to existing RAW patches, and can be coded lossless or lossy, as well as coded in regular or auxiliary video.
- Additionally, we presented a method to improve color representation of triangle patches, allowing for good reconstruction quality, even using 420 lossy color coding.
- Coding of triangle patch data can be further improved by ordering triangles according to the centroid color value (implementation is on-going)

- We suggest the group to investigate this tool further.