SONY

# On the integration between MIV and V-PCC

m51044

**Danillo Graziosi, Alexandre Zaghetto, Ali Tabatabai**

**US Research Center**

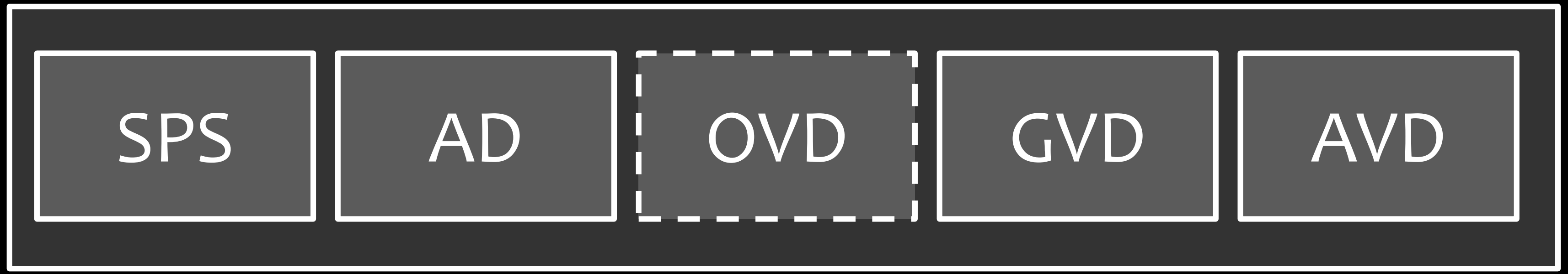
**Oct 6, 2019**

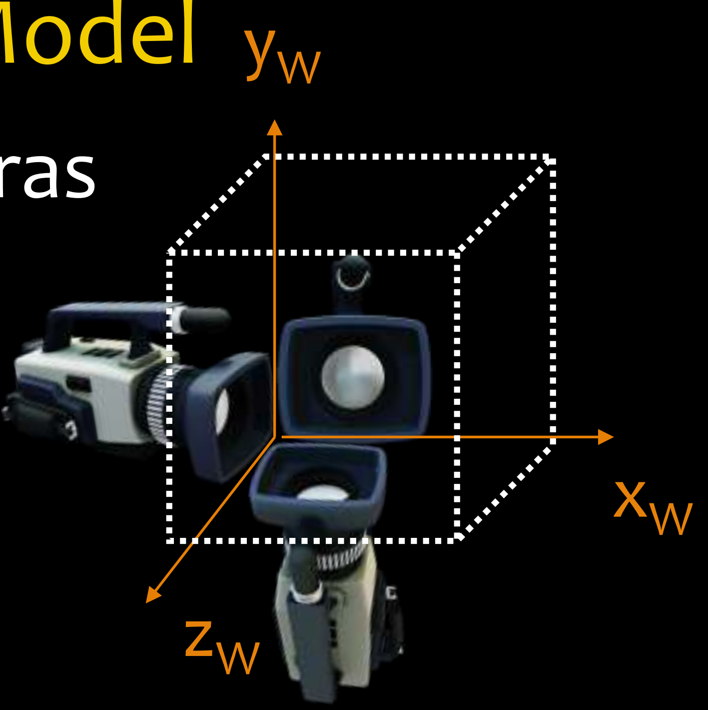
# SUMMARY

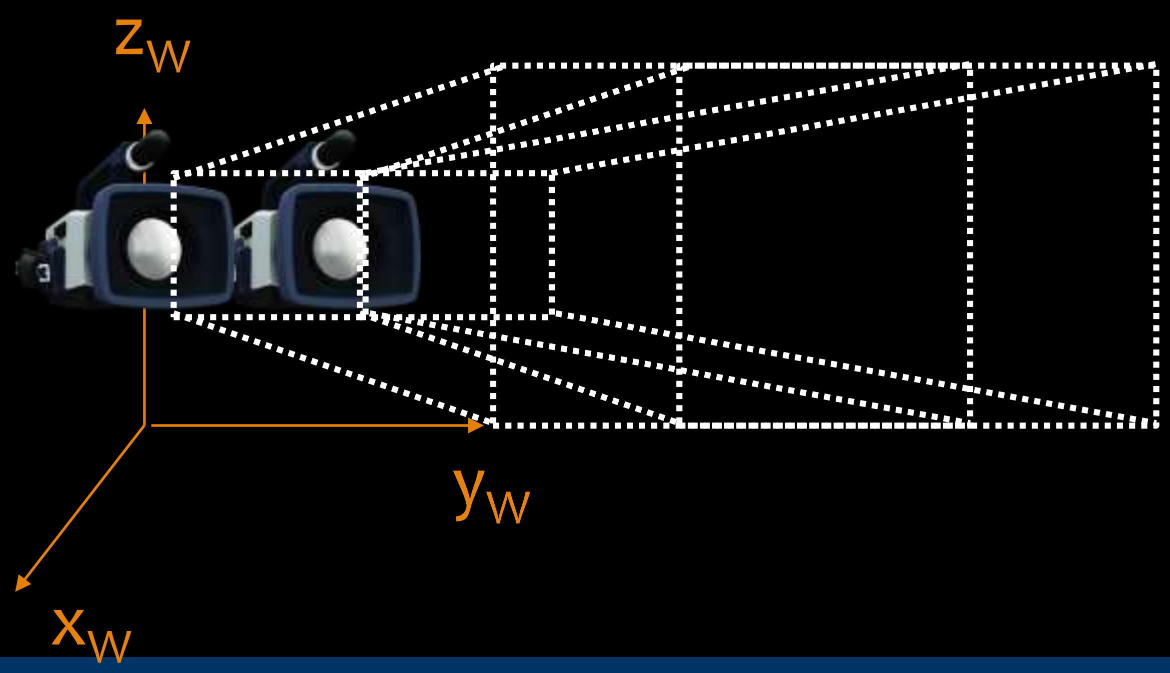
## Integration between V-PCC and MIV

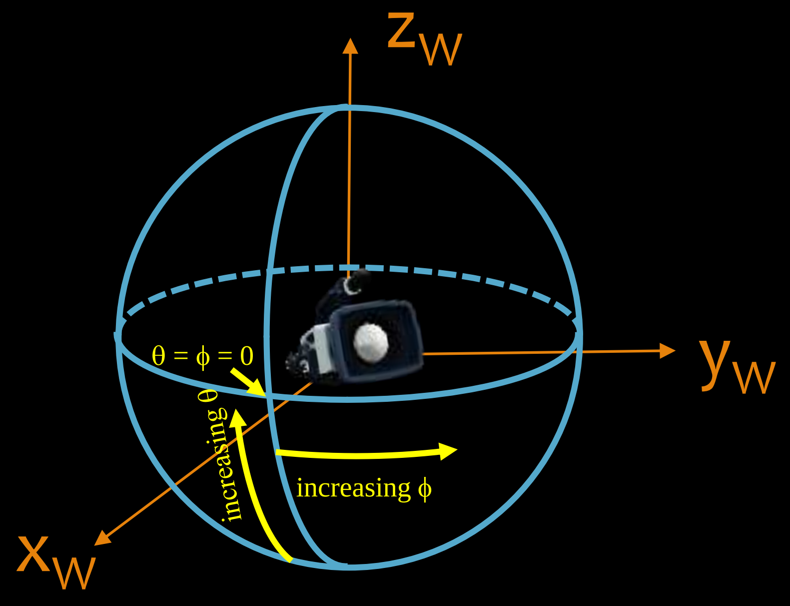
- Syntax changes in V-PCC to accommodate MIV content
- Occupancy Map generation
  - Embedded occupancy map in geometry video
- Geometry Image Conversion
  - 16 → 10 bit conversion
- Unified Camera Model
  - V-PCC x MIV cameras
- Results
- Conclusion

| vpcc_parameter_set( ) { | Descriptor |
|---|---|
| profile_tier_level() | |
| camera_parameters() | |
| **vpcc_sequence_parameter_set_id** | u(4) |
| **vpcc_atlas_count_minus1** | u(6) |
| … | |



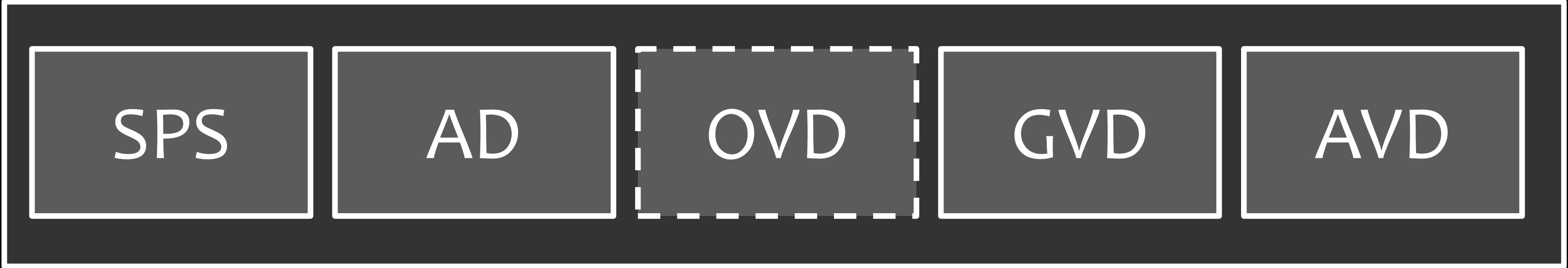



# V-PCC and MIV syntax
# (differences from V-PCC highlighted in green)




| vpcc_parameter_set( ) { | Descriptor |
|---|---|
| profile_tier_level() | |
| camera_parameters() | |
| **vpcc_sequence_parameter_set_id** | u(4) |
| **vpcc_atlas_count_minus1** | u(6) |
| for(j = 0; j < vpcc_atlas_count_minus1 + 1; j++ ) { | |
| **vpcc_frame_width**[ j ] | u(16) |
| **vpcc_frame_height**[ j ] | u(16) |
| **vpcc_avg_frame_rate_present_flag**[ j ] | u(1) |
| if( vpcc_avg_frame_rate_present_flag[ j ] ) | |
| **vpcc_avg_frame_rate**[ j ] | u(16) |
| **vpcc_map_count_minus1**[ j ] | u(4) |
| if( vpcc_map_count_minus1[ j ] > 0 ) | |
| **vpcc_multiple_map_streams_present_flag**[ j ] | u(1) |
| vpcc_map_absolute_coding_enabled_flag[ j ][ 0 ] = 1 | |
| for(i = 0; i < vpcc_map_count_minus1[ j ]; i++ ) { | |
| **vpcc_map_absolute_coding_enabled_flag**[ j ][ i + 1 ] | u(1) |
| if( vpcc_map_absolute_coding_enabled_flag[ j ][ i + 1 ] = = 0 ) { | |
| if( i > 0) | |
| **vpcc_map_predictor_index_diff**[ j ][ i + 1 ] | ue(v) |
| else | |
| vpcc_map_predictor_index_diff[ j ][ i + 1 ] = 0 | |
| } | |
| } | |
| **vpcc_raw_patch_enabled_flag**[ j ] | u(1) |
| if( vpcc_raw_patch_enabled_flag[ j ] ) | |
| **vpcc_raw_separate_video_present_flag**[ j ] | u(1) |
| occupancy_information( j ) | |
| geometry_information( j ) | |
| attribute_information( j ) | |
| } | |
| byte_alignment( ) | |
| } | |

| occupancy_information( atlas_id ) { | Descriptor |
|---|---|
| **oi_occupancy_map_embedded_flag[** atlas_id **]** | u(1) |
| if(**oi_occupancy_map_embedded_flag[** atlas_id **]) {** | |
| **oi_occupancy_map_embedded_threshold[** atlas_id **]** | u(8) |
| } else { | |
| **oi_occupancy_codec_id[** atlas_id **]** | u(8) |
| **oi_lossy_occupancy_map_compression_threshold[** atlas_id **]** | u(8) |
| **oi_occupancy_nominal_2d_bitdepth_minus1[** atlas_id **]** | u(5) |
| **oi_occupancy_MSB_align_flag[** atlas_id **]** | u(1) |
| } | |
| } | |

| geometry_information( atlas_id ) { | Descriptor |
|---|---|
| **gi_geometry_codec_id[** atlas_id **]** | u(8) |
| **gi_geometry_nominal_2d_bitdepth_minus1[** atlas_id **]** | u(5) |
| **gi_geometry_MSB_align_flag[** atlas_id **]** | u(1) |
| **gi_geometry_3d_coordinates_bitdepth_minus1[** atlas_id **]** | u(5) |
| if( vpcc_raw_separate_video_present_flag[ atlas_id ] ) | |
| **gi_raw_geometry_codec_id[** atlas_id **]** | u(8) |
| **gi_geometry_range_compression_flag[** atlas_id **]** | u(1) |
| if(**gi_geometry_range_compression_flag[** atlas_id **])** | |
| **gi_range_convertion_offset[** atlas_id **]** | u(8) |
| } | |

# PCC/MIV occupancy map harmonization

- Added syntax element to indicate if occupancy map is embedded in geometry streams or not

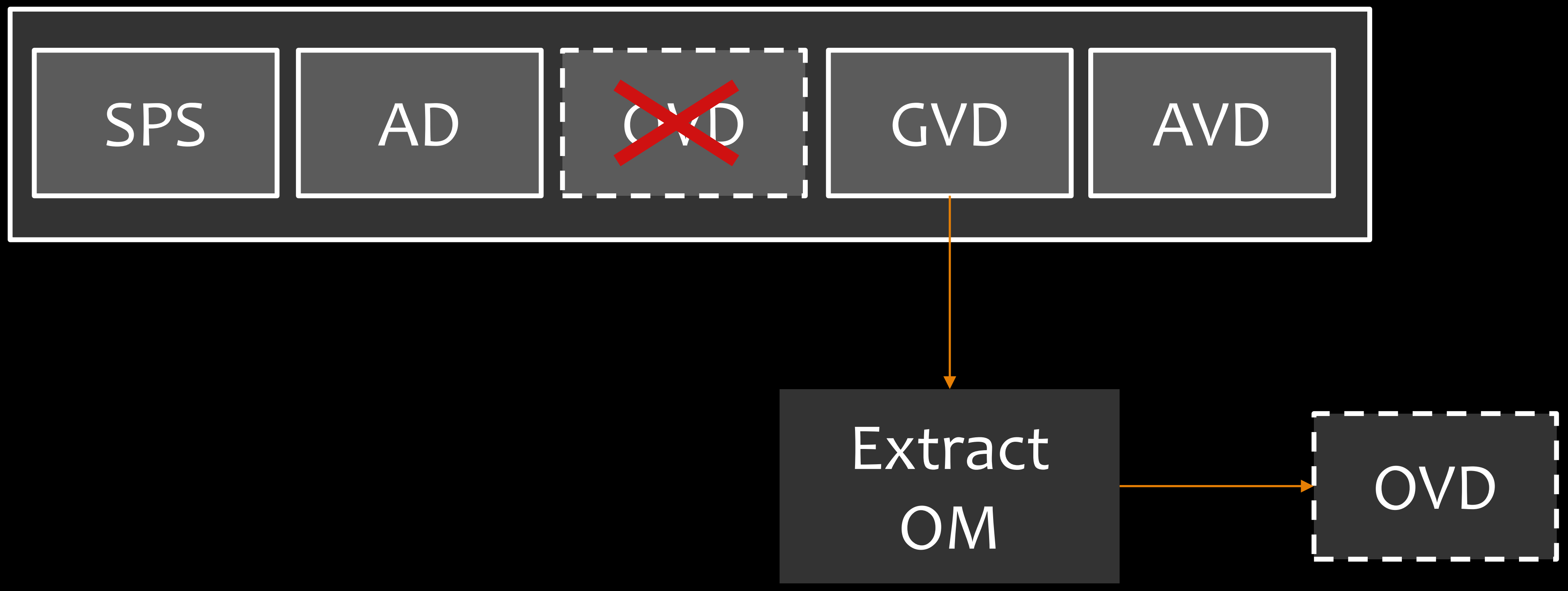
| occupancy_information( atlas_id ) { | **Descriptor** |
|---|---|
| **oi_occupancy_map_embedded_flag[** atlas_id **]** | u(1) |
| if(**oi_occupancy_map_embedded_flag[** atlas_id **]) {** | |
| **oi_occupancy_map_embedded_threshold[** atlas_id **]** | u(8) |
| } else { | |
| **oi_occupancy_codec_id[** atlas_id **]** | u(8) |
| **oi_lossy_occupancy_map_compression_threshold[** atlas_id **]** | u(8) |
| **oi_occupancy_nominal_2d_bitdepth_minus1[** atlas_id **]** | u(5) |
| **oi_occupancy_MSB_align_flag[** atlas_id **]** | u(1) |
| } | |
| } | |

This allows to encode the occupancy map in both methods exactly the same, but MIV might consider the advantages of sending the occupancy map explicitly.

# OVD generation

**oi_occupancy_map_embedded_flag[** atlas_id **] = 0**


**oi_occupancy_map_embedded_flag[** atlas_id **] = 1**

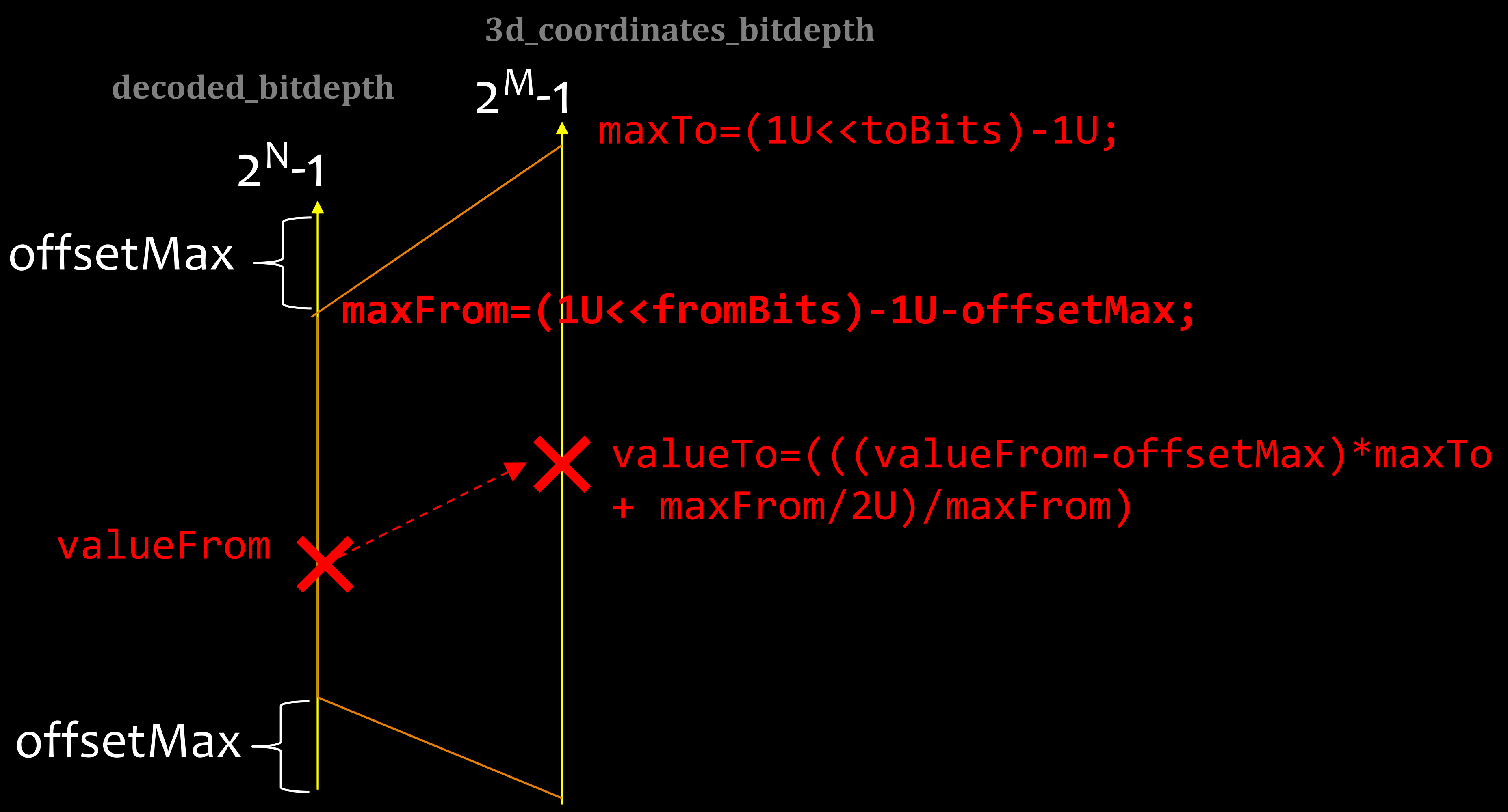



if (GVD[0][x][y] < **oi_occupancy_map_embedded_threshold[** atlas_id **] )**
 OVD[0][x][y] = 0
else
 OVD[0][x][y] = 1

# PCC/MIV geometry conversion harmonization

- Added syntax element to indicate if the range of the depth image was compressed or not (otherwise assume depth slicing)




| geometry_information( atlas_id ) { | **Descriptor** |
|---|---|
| **gi_geometry_codec_id[** atlas_id **]** | u(8) |
| **gi_geometry_nominal_2d_bitdepth_minus1[** atlas_id **]** | u(5) |
| **gi_geometry_MSB_align_flag[** atlas_id **]** | u(1) |
| **gi_geometry_3d_coordinates_bitdepth_minus1[** atlas_id **]** | u(5) |
| if( vpcc_raw_separate_video_present_flag[ atlas_id ] ) | |
| **gi_raw_geometry_codec_id[** atlas_id **]** | u(8) |
| **gi_geometry_range_compression_flag[** atlas_id **]** | u(1) |
| if(**gi_geometry_range_compression_flag[** atlas_id **]**) | |
| **gi_range_convertion_offset[** atlas_id **]** | u(8) |
| } | |

In MIV, depth is represented from 3D → 2D by range compression. Decoded bit depth is always less than the nominal bit depth

Offset is used to reserve values to embed occupancy maps in geometry?

In PCC, depth is represented from 3D → 2D by range slicing.

Range compression can also be done with the current V-PCC syntax (see m51170, based on proposal m39342 from Intel)

# Camera Model (differences from m49590 highlighted in green)

| camera_params_list( ) { | Descriptor |
|---|---|
| **cpl_explicit_projection_info_enabled_flag** | u(1) |
| if (cpl_explicit_projection_info_enabled_flag) { | |
| **cpl_num_cameras_minus1** | u(16) |
| **cpl_cam_pos_x_granularity** | u(32) |
| **cpl_cam_pos_y_granularity** | u(32) |
| **cpl_cam_pos_z_granularity** | u(32) |
| **cpl_yaw_pitch_roll_present_flag** | u(1) |
| **cpl_cam_scaling_flag**[ i ] | u(1) |
| **cpl_cam_id_present_flag** | u(1) |
| for ( i= 0; i <= cpl_num_cameras_minus1; i++) { | |
| if (cam_id_present_flag) | |
| **cpl_cam_view_id**[ i ] | u(16) |
| else | |
| cpl_cam_view_id[ i ] = i | |
| **cpl_cam_use_OMAF_coordinate_system_flag**[ i ] | u(1) |
| **cpl_cam_pos_x**[ i ] | u(32) |
| **cpl_cam_pos_y**[ i ] | u(32) |
| **cpl_cam_pos_z**[ i ] | u(32) |
| if (cpl_yaw_pitch_roll_present_flag) { | |
| **cpl_cam_yaw**[i ] | u(32) |
| **cpl_cam_pitch**[ i ] | u(32) |
| **cpl_cam_roll**[ i ] | u(32) |
| } | |
| if (cpl_cam_scaling_flag) { | |
| **cpl_cam_rot_scale_x**[i ] | u(32) |
| **cpl_cam_rot_scale_y**[ i ] | u(32) |
| **cpl_cam_rot_scale_z**[ i ] | u(32) |
| } | |
| } | |
| **cpl_intrinsic_params_equal_flag** | u(1) |
| camera_intrinsics(cpl_intrinsic_params_equal, cpl_num_ cameras_minus1 ) | |
| **cpl_depth_quantization_params_equal_flag** | u(1) |
| depth_quantization(cpl_depth_quantization_equal_flag, cpl_num_ cameras_minus1) | |
| } | |
| else { | |
| **cpl_45degree_projection_patch_enabled_flag** | u(1) |
| cpl_num_cameras_minus1 =<br>cpl_45degree_projection_patch_enabled_flag ? 6 : 10 | |
| } | |
| } | |

| camera_intrinsics( equalFlag, numCamerasMinus1 ) { | Descriptor |
|---|---|
| for ( v = 0; v <=; v++ ) { | |
| if ( v == 0 \|\| equalFlag == 0) { | |
| **cam_type**[ v ] | u(8) |
| **projection_plane_width**[ v ] | u(32) |
| **projection_plane_height**[ v ] | u(32) |
| **cpl_cam_reverse_depth_flag**[ i ] | u(1) |
| **cpl_cam_projection_mode_flag**[ i ] | u(1) |
| if ( cam_type[ v ] == 0 ) { | |
| **erp_phi_min**[ v ] | u(32) |
| **erp_phi_max**[ v ] | u(32) |
| **erp_theta_min**[ v ] | u(32) |
| **erp_theta_max**[ v ] | u(32) |
| } else if (cam_type[ v ] = = 1) | |
| **cubic_map_type**[ v ] | u(8) |
| else if (cam_type[ v ] = = 2) { | |
| **perspective_focal_hor**[ v ] | u(32) |
| **perspective_focal_ver**[ v ] | u(32) |
| **perspective_center_hor**[ v ] | u(32) |
| **perspective_center_ver**[ v ] | u(32) |
| } | |
| else { | |
| cam_type[ v ] = cam_type[ 0 ] | |
| projection_plane_width[ v ] = projection_plane_width[ 0 ] | |
| projection_plane_height[ v ] = projection_plane_height[ 0 ] | |
| erp_phi_min[ v ] = erp_phi_min[ 0 ] | |
| erp_phi_max[ v ] = erp_phi_max[ 0 ] | |
| erp_theta_min[ v ] = erp_theta_min[ 0 ] | |
| erp_theta_max[ v ] = erp_theta_max[ 0 ] | |
| cubic_map_type[ v ] = cubic_map_type[ 0 ] | |
| perspective_focal_hor[ v ] = perspective_focal_hor[ 0 ] | |
| perspective_focal_ver[ v ] = perspective_focal_ver[ 0 ] | |
| perspective_center_hor[ v ] = perspective_center_hor[ 0 ] | |
| perspective_center_ver[ v ] = perspective_center_ver[ v ] | |
| } | |
| } | |
| } | |

| depth_quantization( equalFlag, numCamerasMinus1 ) { | Descriptor |
|---|---|
| for ( v = 0; v <=; v++ ) { | |
| if ( v == 0 \|\| equalFlag == 0) { | |
| **quantization_law**[ v ] | u(8) |
| if ( quantization_law[ v ] == 0 ) { | |
| **depth_near**[ v ] | u(32) |
| **depth_far**[ v ] | u(32) |
| } | |
| } else { | |
| quantization_law[ v ] = quantization_law[ 0 ] | |
| depth_near[ v ] = depth_near[ 0 ] | |
| depth_far[ v ] = depth_far[ v ] | |
| } | |
| } | |
| } | |

V-PCC camera model syntax elements:
- Cluster Index
- U1,V1,D1
- sizeU1,sizeV1,sizeD1

# Camera Model

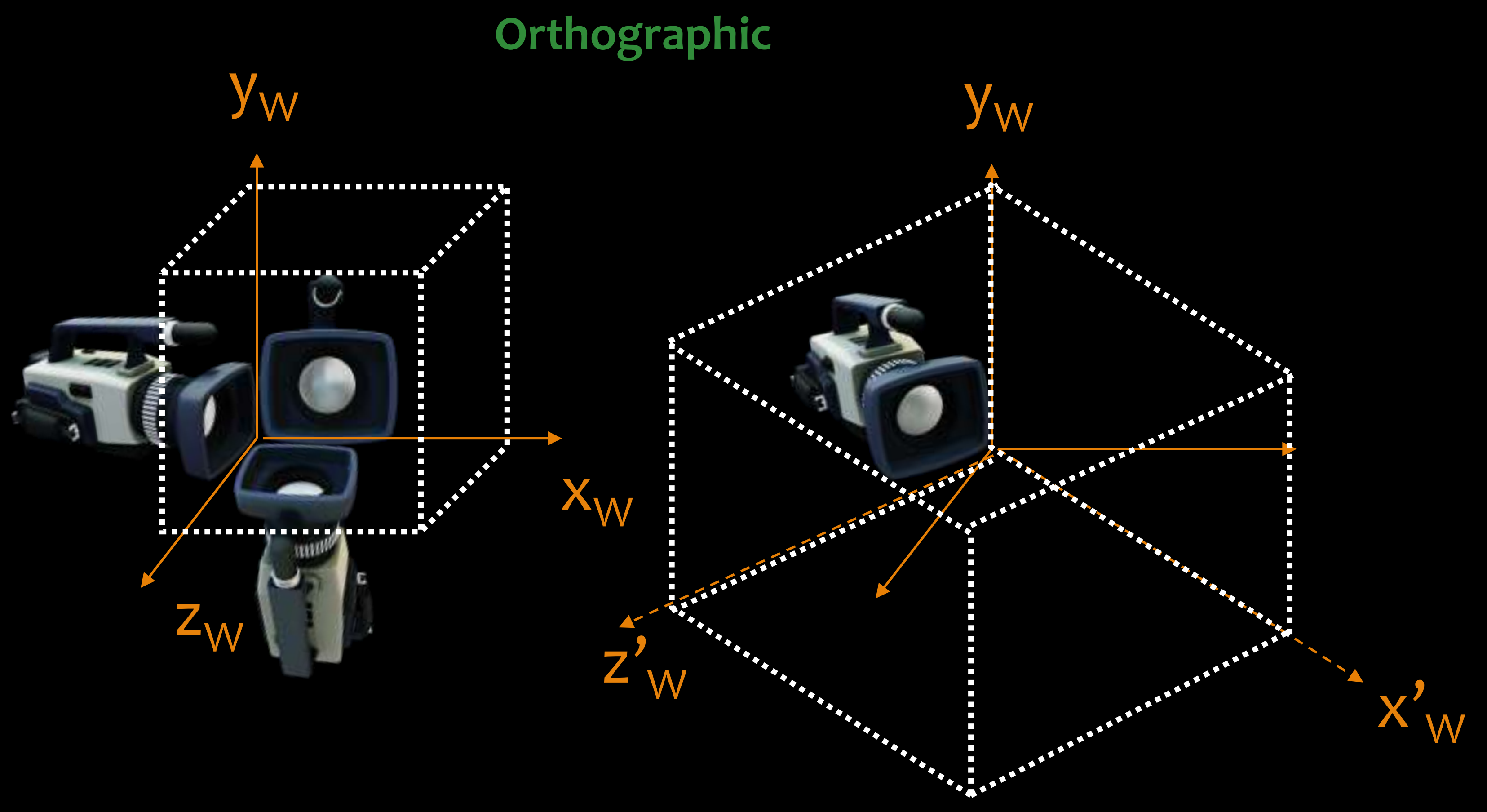
MIV camera model syntax elements:
- Extrinsic parameters (XYZ-Position, Yaw, Pitch, Roll)
- Intrinsic Parameters (focal distance, window size, principal point, etc.)
- Near/far planes

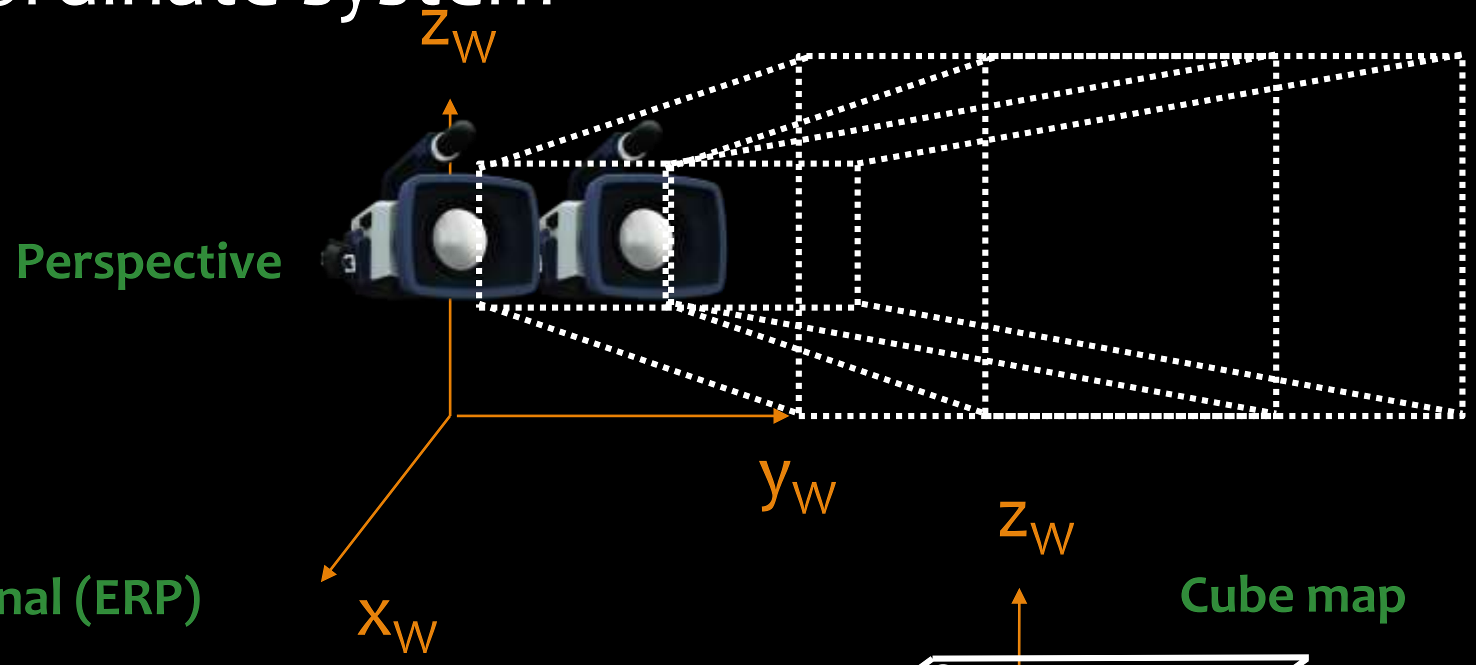
❖ V-PCC (object-centric)

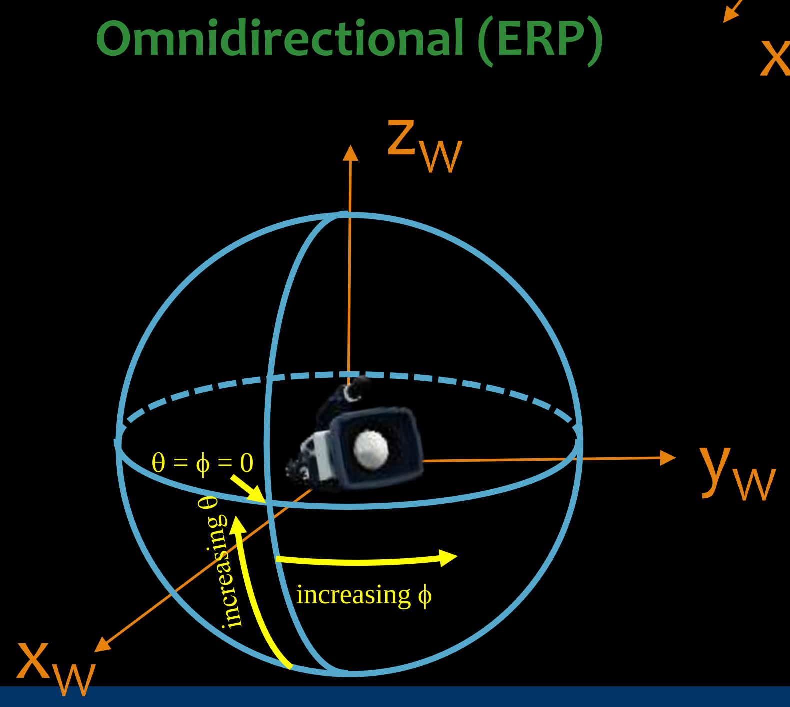
- Orthographic cameras placed at the origin of the world coordinate system (OpenGL model), with screen size equals to the size of the 3D bounding box

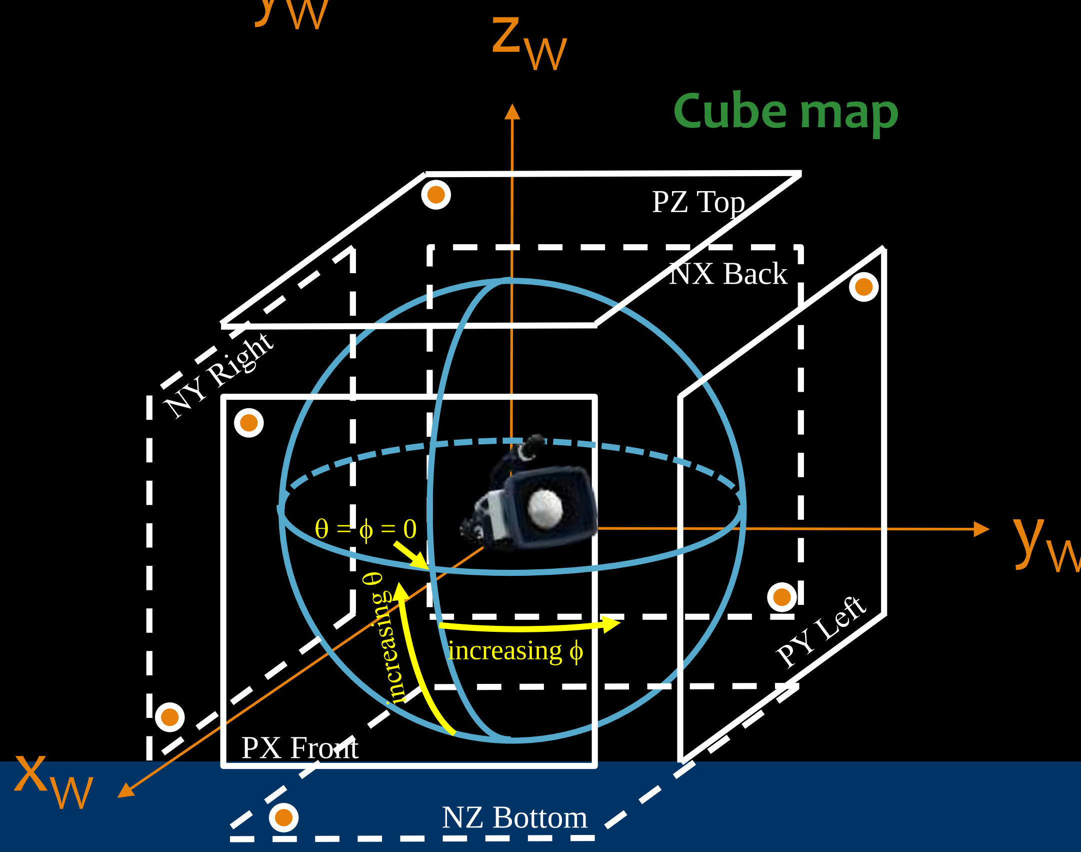
❖ MIV (scene-centric)

- Perspective/omnidirectional cameras placed at any position in the 3D space following OMAF coordinate system



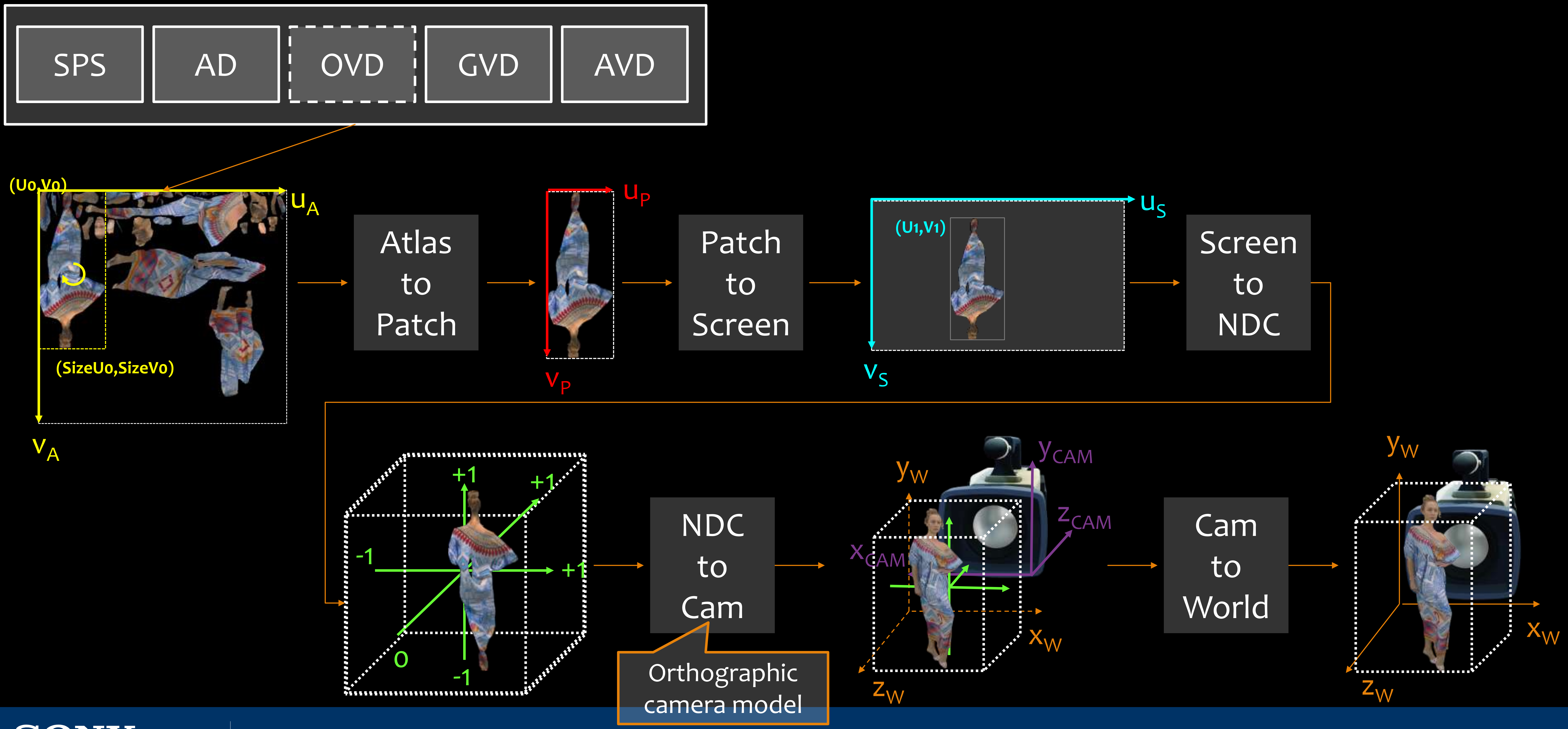



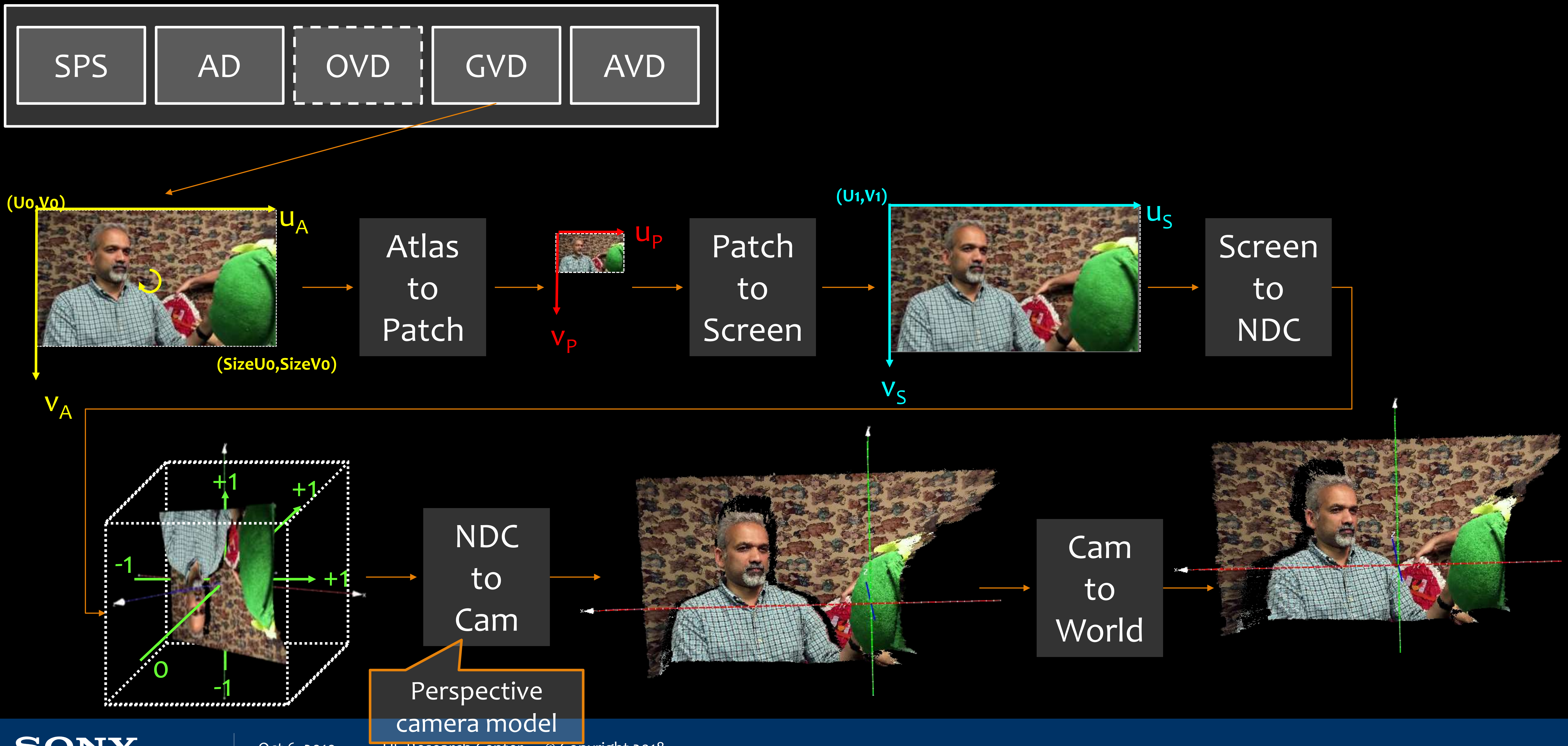



# V-PCC point reconstruction

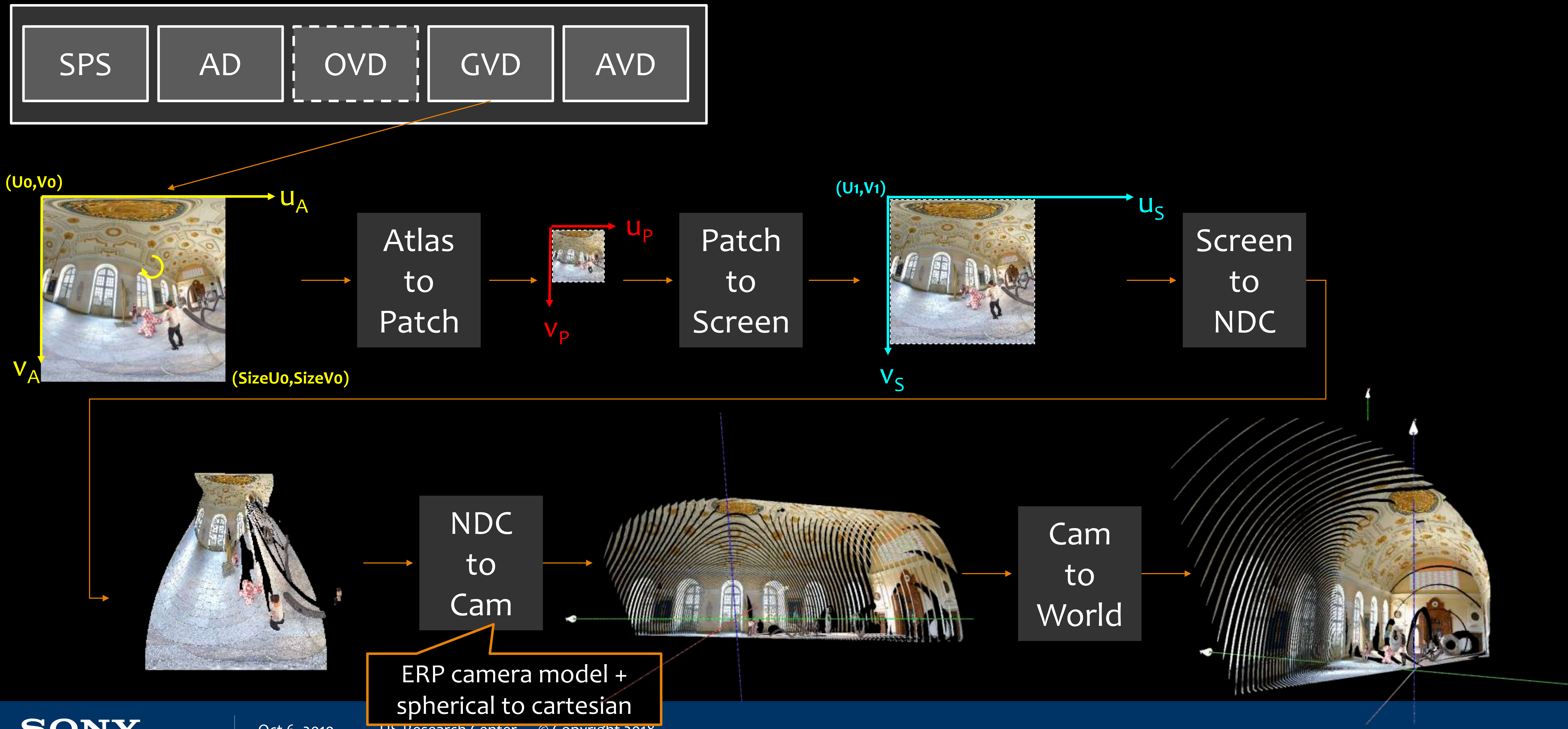


# MIV Atlas point reconstruction (perspective camera)



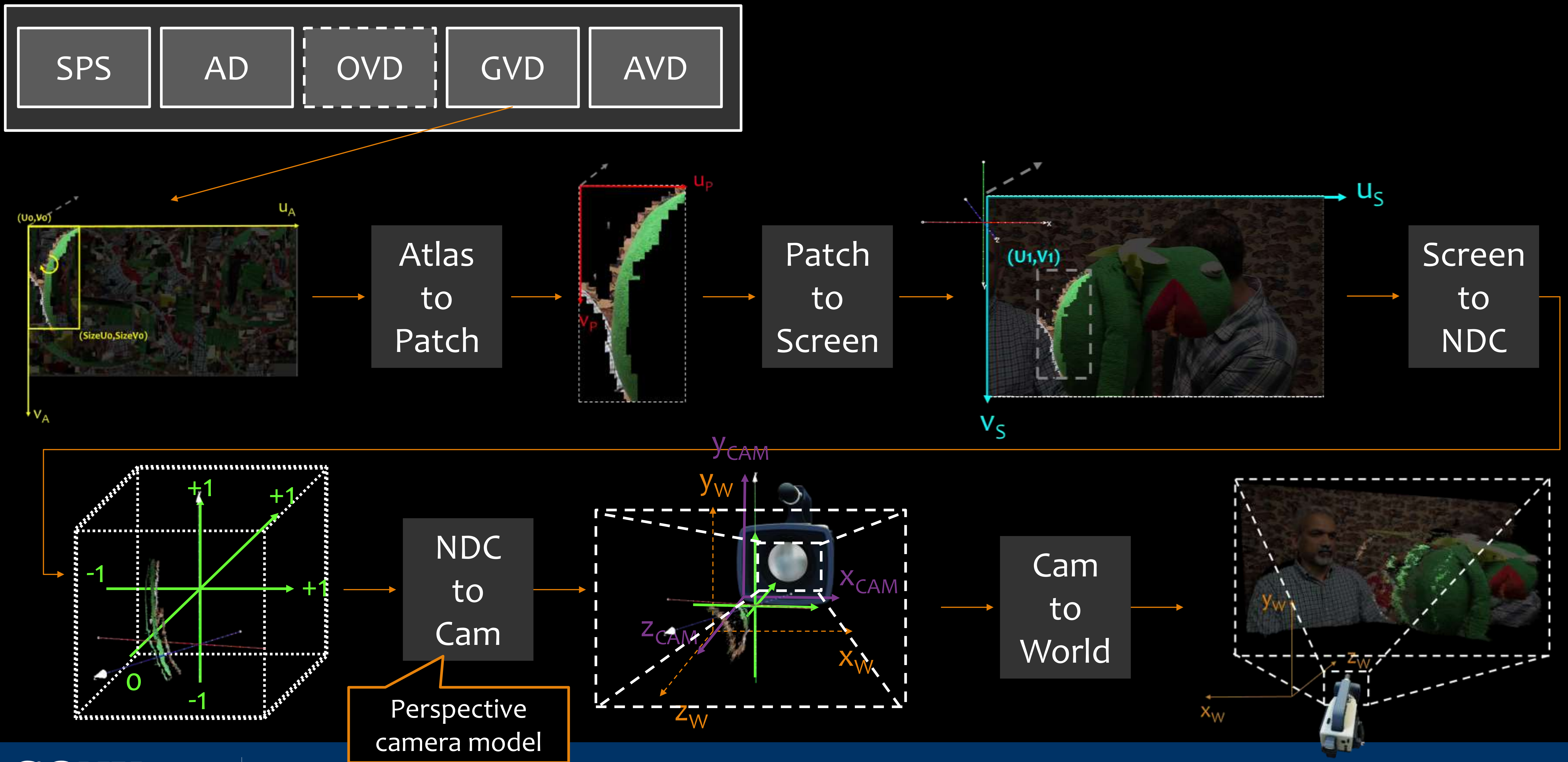

 

# MIV Atlas point reconstruction (ERP camera)



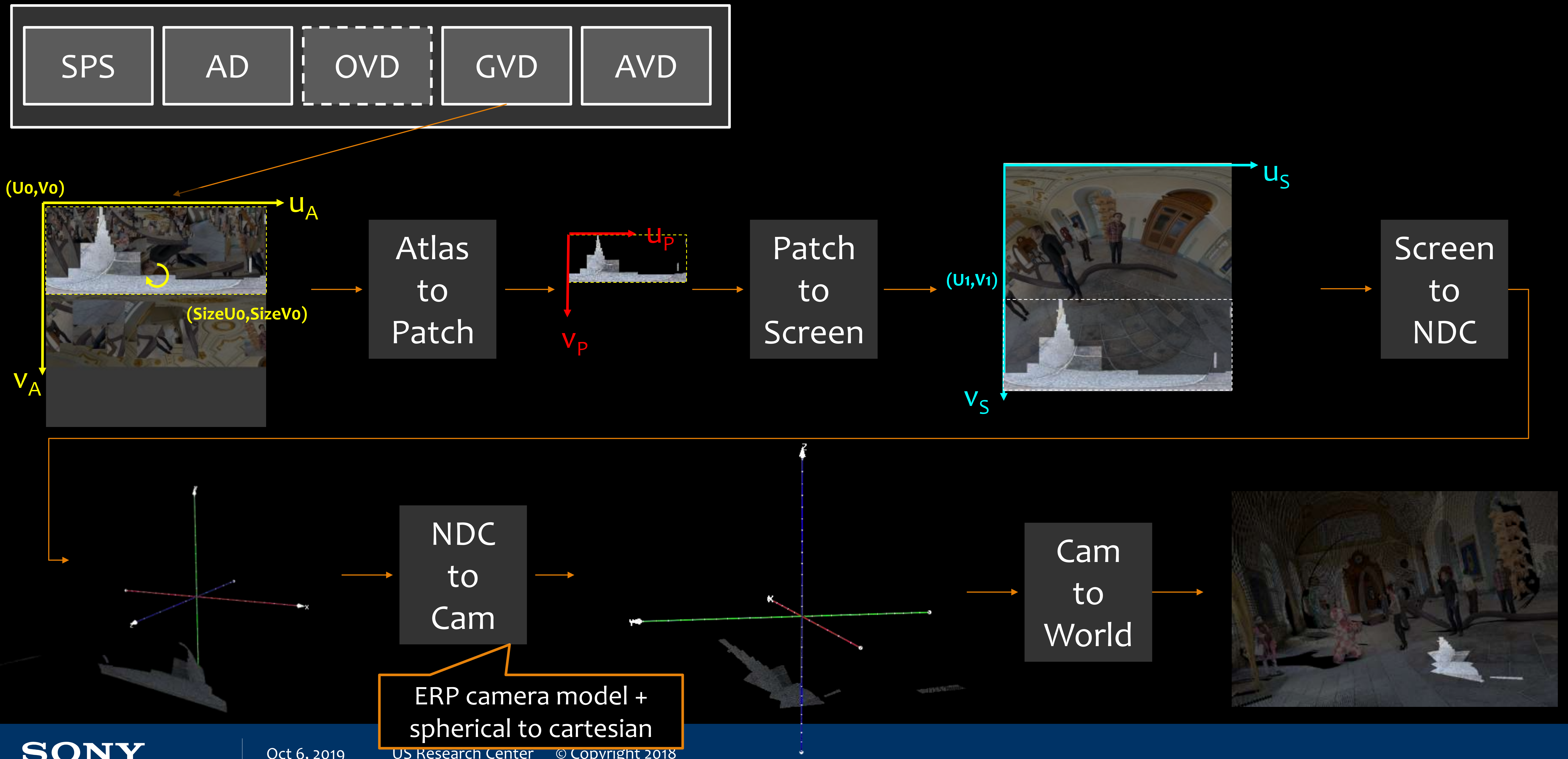

 

# MIV Patch point reconstruction (perspective camera)



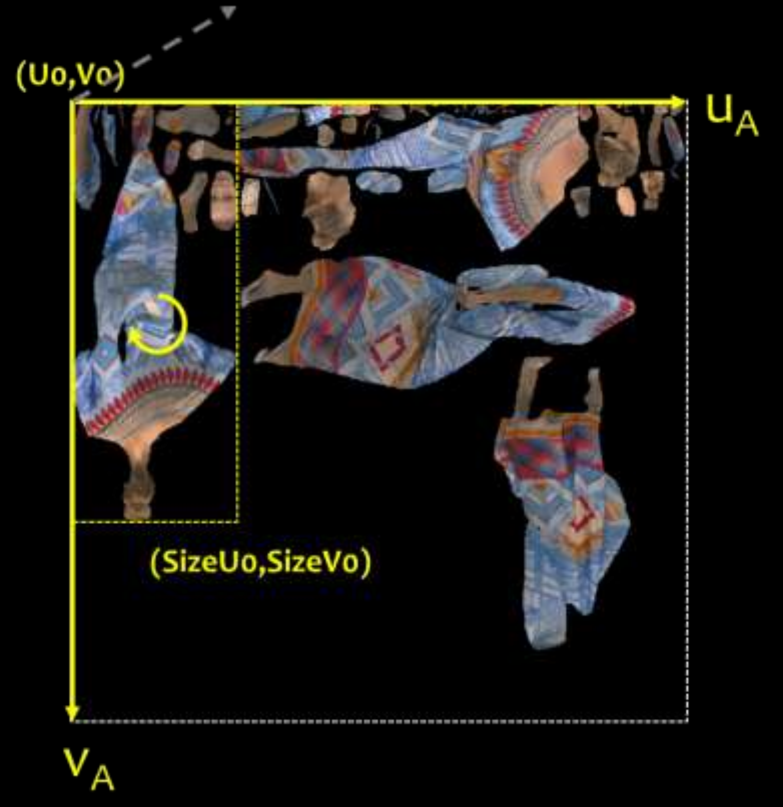

 

# MIV Patch point reconstruction (ERP camera)



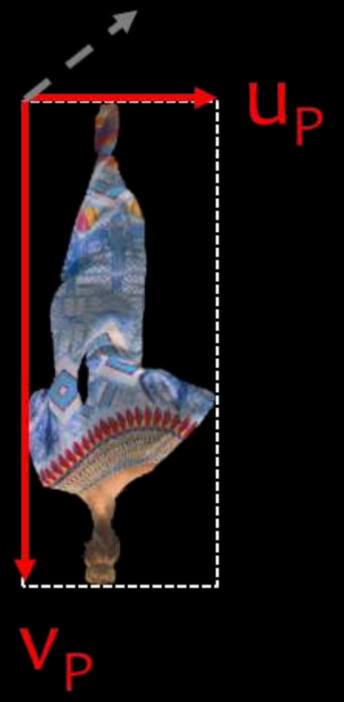
# Atlas → Patch

- Syntax: U0,V0, SizeU0,SizeV0, orientation (rotation+flip)




$$\begin{bmatrix} R^T & 0 & -R^T T \\ 0 \quad 0 & 1 & 0 \\ 0 \quad 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} R & 0 & T \\ 0 \quad 0 & 1 & 0 \\ 0 \quad 0 & 0 & 1 \end{bmatrix}$$




Different syntax elements, this should be aligned if integration occurs

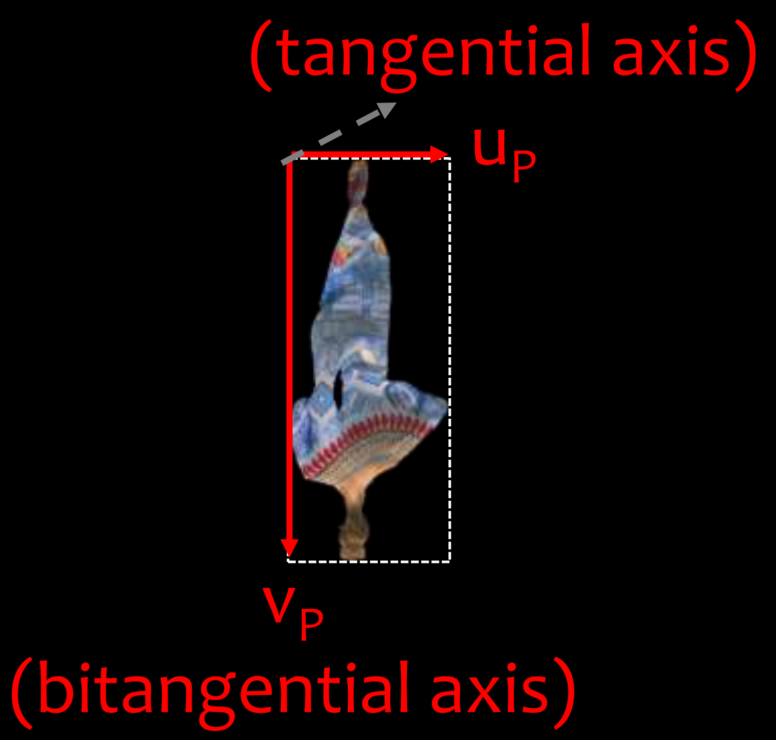
```
vector<size_t> miv2pcc = { PATCH_ORIENTATION_DEFAULT (0)  //no flip, upright (0)
                           PATCH_ORIENTATION_ROT270  (3)      //no flip, ccw (1)
                           PATCH_ORIENTATION_ROT180  (2)       //no flip, ht (2)
                           PATCH_ORIENTATION_ROT90   (7)       //no flip, cw (3)
                           PATCH_ORIENTATION_MROT180 (6)     //flip, upright (4)
                           PATCH_ORIENTATION_SWAP    (1)         //flip, ccw (5)
                           PATCH_ORIENTATION_MIRROR  (4)          //flip, ht (6)
                           PATCH_ORIENTATION_MROT90  (5)          //flip, cw (7)
```

| x | Identifier | Rotation( x ) | Offset( x ) |
|---|---|---|---|
| 0 | FPO_NULL | $\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$ | $\begin{bmatrix}0\\0\end{bmatrix}$ |
| 1 | FPO_SWAP | $\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$ | $\begin{bmatrix}0\\0\end{bmatrix}$ |
| 2 | FPO_ROT90 | $\begin{bmatrix}0 & -1\\1 & 0\end{bmatrix}$ | $\begin{bmatrix}\text{Patch2dSizeY[ p ]}\\0\end{bmatrix}$ |
| 3 | FPO_ROT180 | $\begin{bmatrix}-1 & 0\\0 & -1\end{bmatrix}$ | $\begin{bmatrix}\text{Patch2dSizeX[ p ]}\\\text{Patch2dSizeY[ p ]}\end{bmatrix}$ |
| 4 | FPO_ROT270 | $\begin{bmatrix}0 & 1\\-1 & 0\end{bmatrix}$ | $\begin{bmatrix}0\\\text{Patch2dSizeX[ p ]}\end{bmatrix}$ |
| 5 | FPO_MIRROR | $\begin{bmatrix}-1 & 0\\0 & 1\end{bmatrix}$ | $\begin{bmatrix}\text{Patch2dSizeX[ p ]}\\0\end{bmatrix}$ |
| 6 | FPO_MROT90 | $\begin{bmatrix}0 & -1\\-1 & 0\end{bmatrix}$ | $\begin{bmatrix}\text{Patch2dSizeY[ p ]}\\\text{Patch2dSizeX[ p ]}\end{bmatrix}$ |
| 7 | FPO_MROT180 | $\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$ | $\begin{bmatrix}0\\\text{Patch2dSizeY[ p ]}\end{bmatrix}$ |

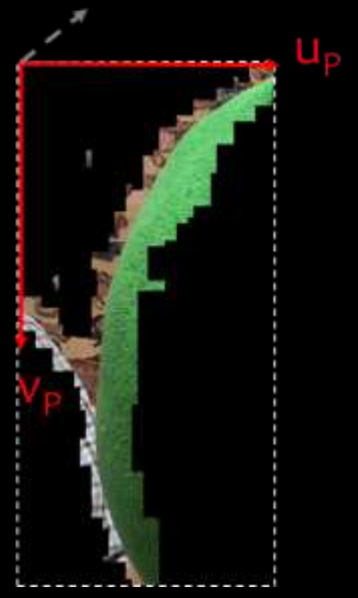
# Patch → Screen

- Syntax: U1, V1

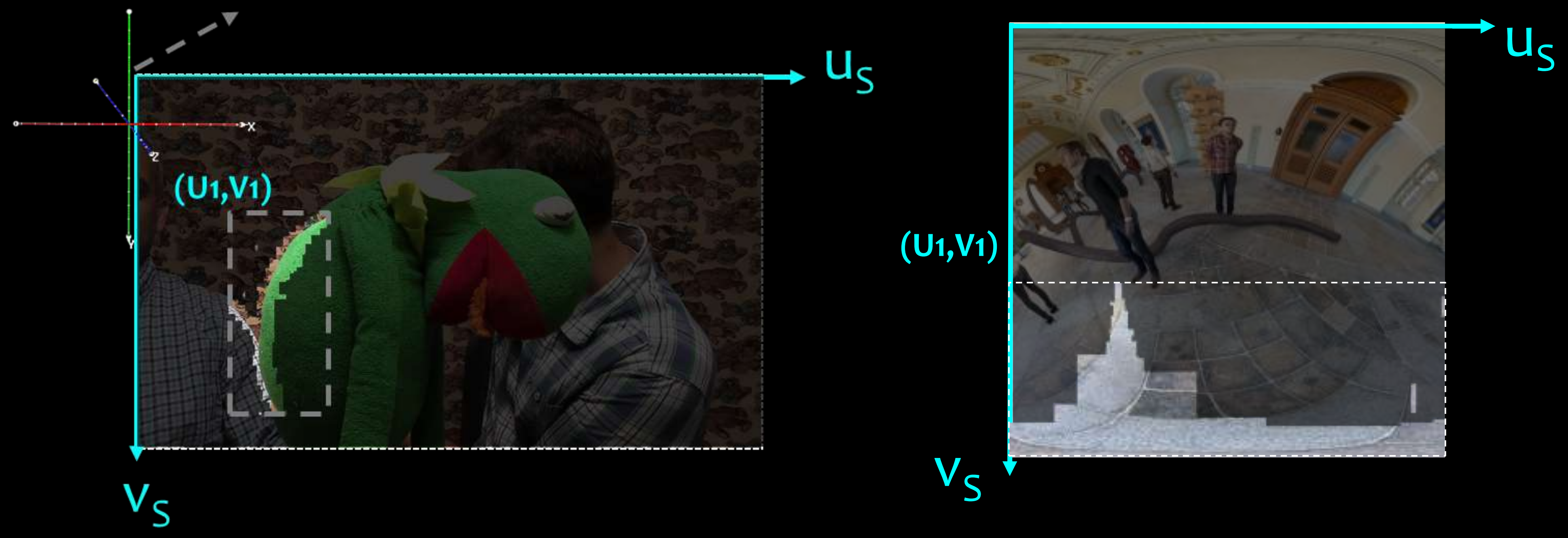



$$\begin{bmatrix} 1 & 0 & 0 & U_1 \\ 0 & 1 & 0 & V_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & -U_1 \\ 0 & 1 & 0 & -V_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



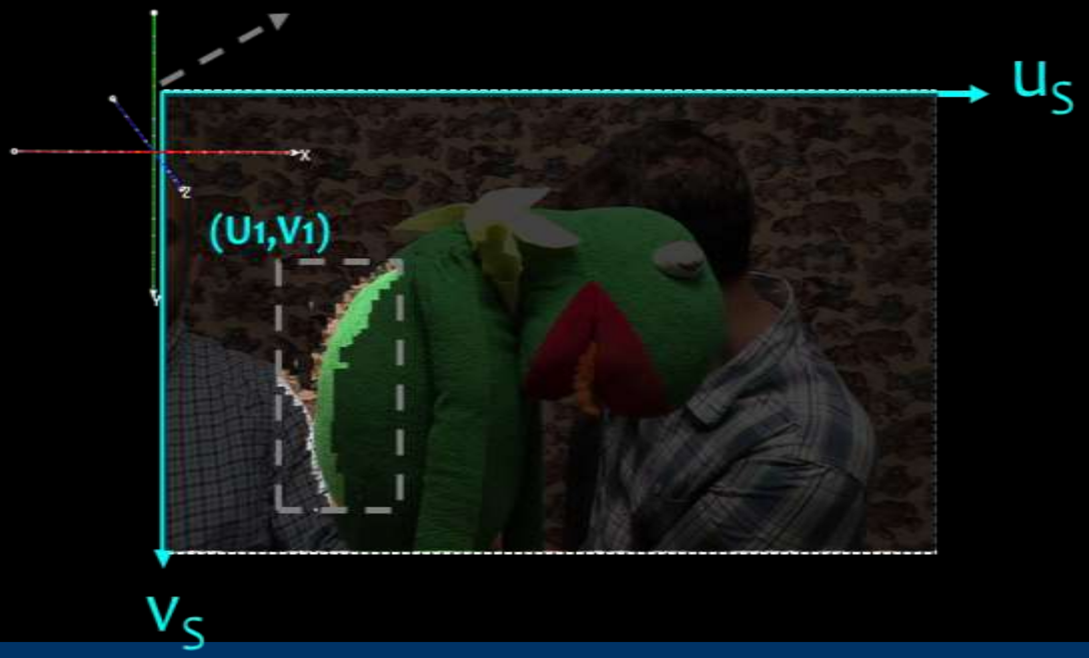




# Screen → Normalized Device Coordinate

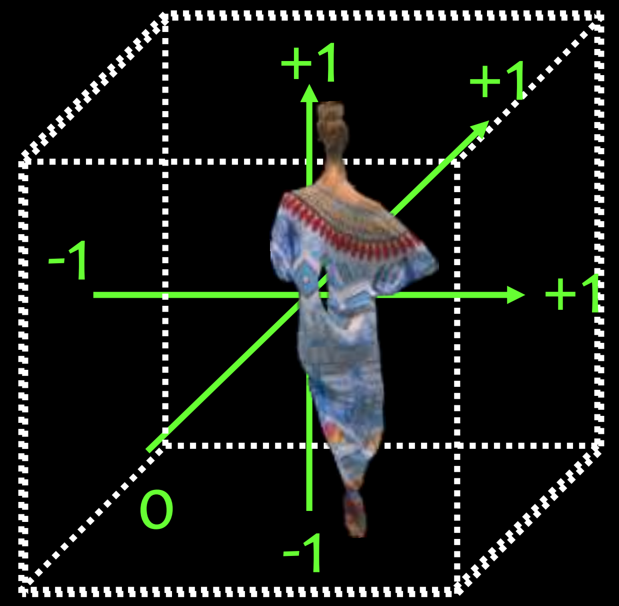
- projection_plane_width, projection_plane_height, gi_geometry_3d_coordinates_bitdepth_minus1, projection_mode, D1

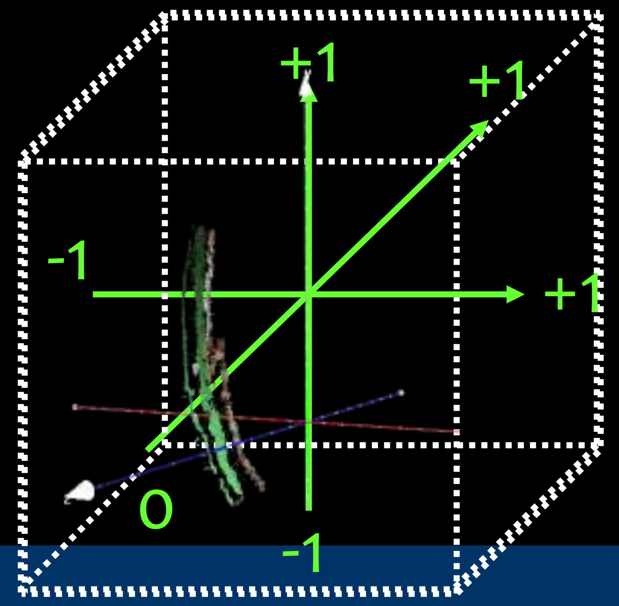




Projection mode

$$\begin{bmatrix} \frac{2}{\text{projection\_plane\_width}} & 0 & 0 & -1 \\ 0 & \frac{2}{\text{projection\_plane\_height}} & 0 & -1 \\ 0 & 0 & \frac{\pm 1}{2^{3d\_bits}-1} & \frac{D_1}{2^{3d\_bits}-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

See contribution m51170 for more details on possible depth scaling, clipping and offset

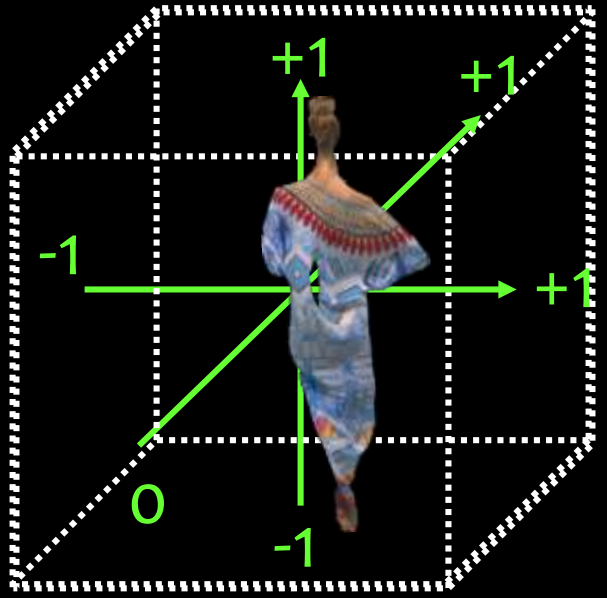


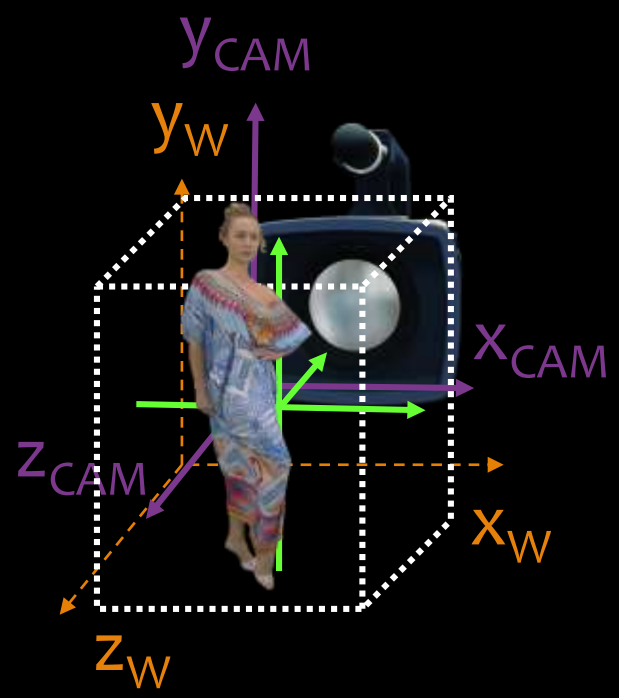



# Normalized Device Coordinate → Camera

- ❖ Camera Intrinsics
  - Orthographic Camera Model

$$\begin{bmatrix} \frac{2^{3D\_bits}-1}{2} & 0 & 0 & \frac{2^{3D\_bits}-1}{2} \\ 0 & \frac{2^{3D\_bits}-1}{2} & 0 & \frac{2^{3D\_bits}-1}{2} \\ 0 & 0 & -2^{3D\_bits}-1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



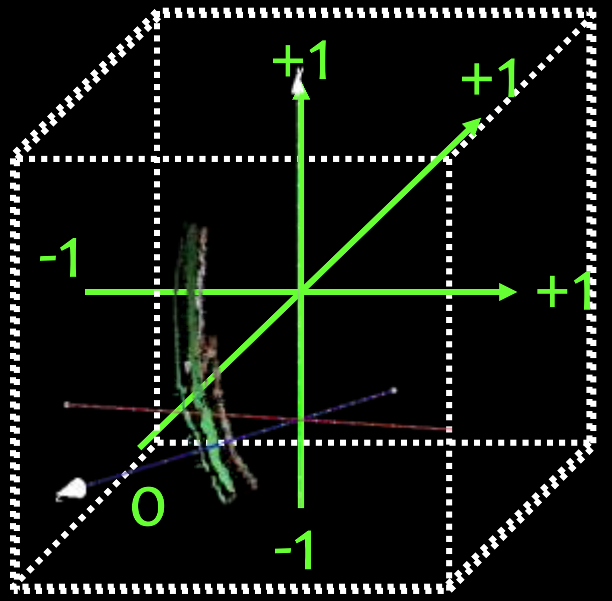

 

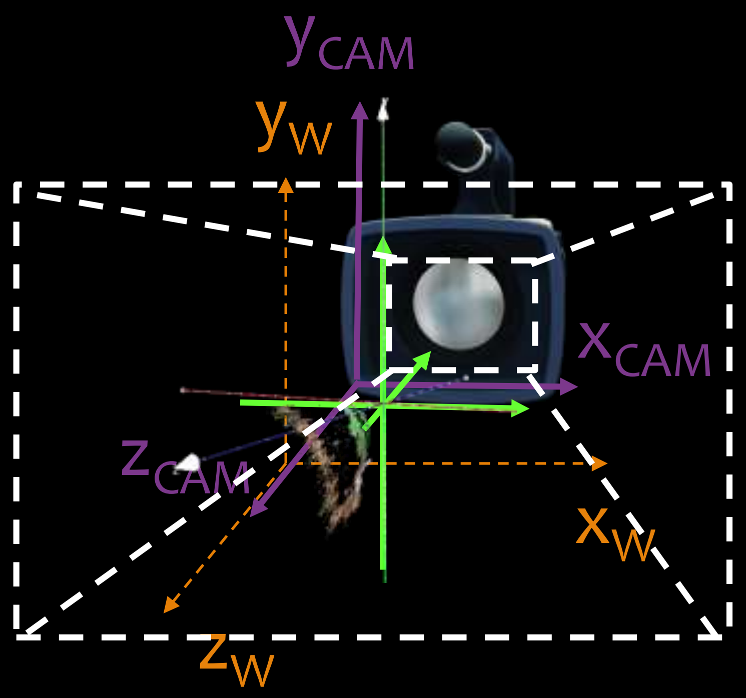
# Normalized Device Coordinate → Camera

## ❖ Camera Intrinsics

- Perspective Camera (with reversed depth/ disparity)

$$\begin{bmatrix} W/2f_x & 0 & 0 & \frac{W/2 - p_x}{f_x} \\ 0 & H/2f_y & 0 & \frac{H/2 - p_y}{f_y} \\ 0 & 0 & 0 & -1 \\ 0 & 0 & (Z_N - Z_F)/(Z_F * Z_N) & -1/Z_F \end{bmatrix}$$

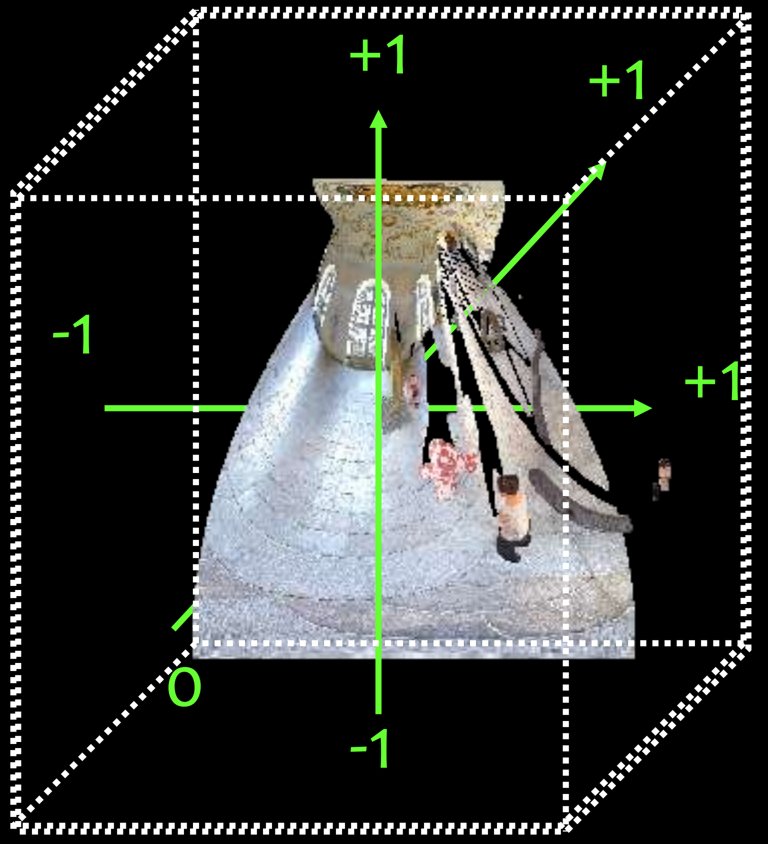


# Normalized Device Coordinate → Camera

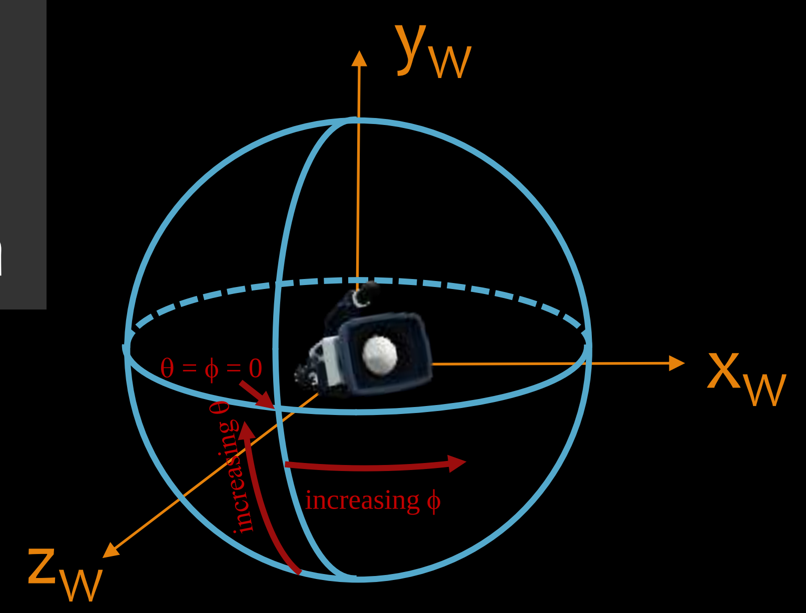
## ❖ Camera Intrinsics

- Omni-directional camera
  - » Orthogonal camera with reversed depth + Spherical to cartesian conversion




$$\begin{bmatrix} \frac{\partial \emptyset}{\partial u}/2 & 0 & 0 & \frac{\partial \emptyset}{\partial u}/2 + \emptyset_{MAX} \\ 0 & \frac{\partial \theta}{\partial v}/2 & 0 & \frac{\partial \theta}{\partial v}/2 + \theta_{MAX} \\ 0 & 0 & 0 & -1 \\ 0 & 0 & (Z_N - Z_F)/(Z_F * Z_N) & -1/Z_F \end{bmatrix}$$

Spherical to Cartesian

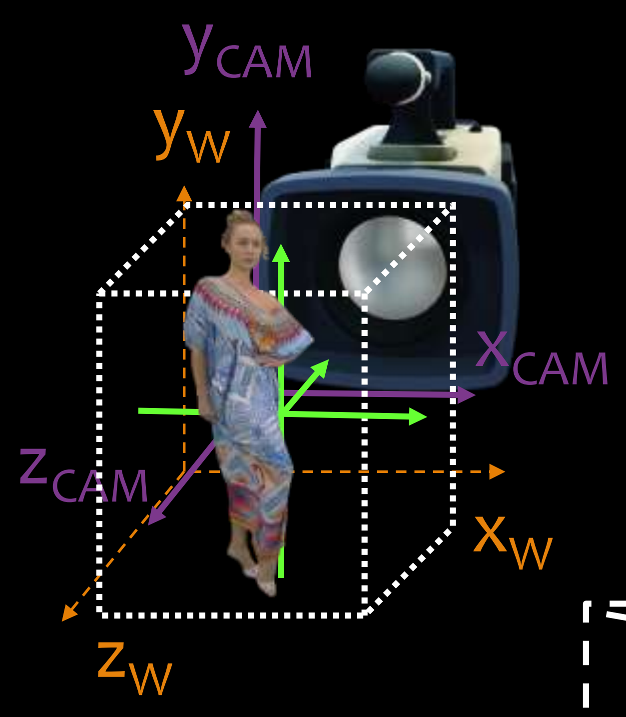


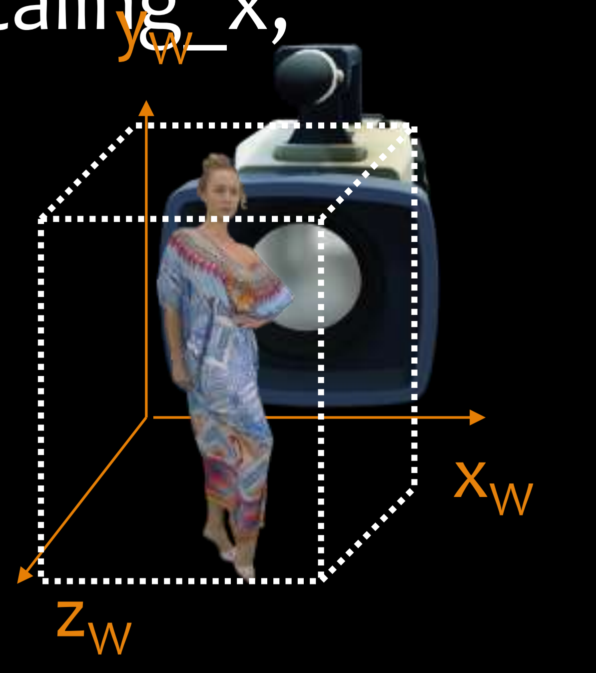
# Camera → World

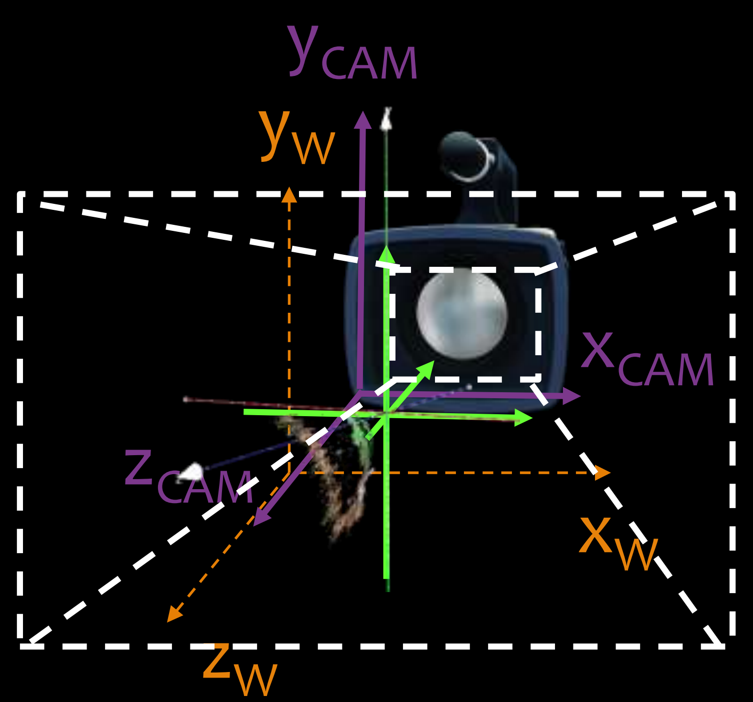
❖ Camera extrinsics:

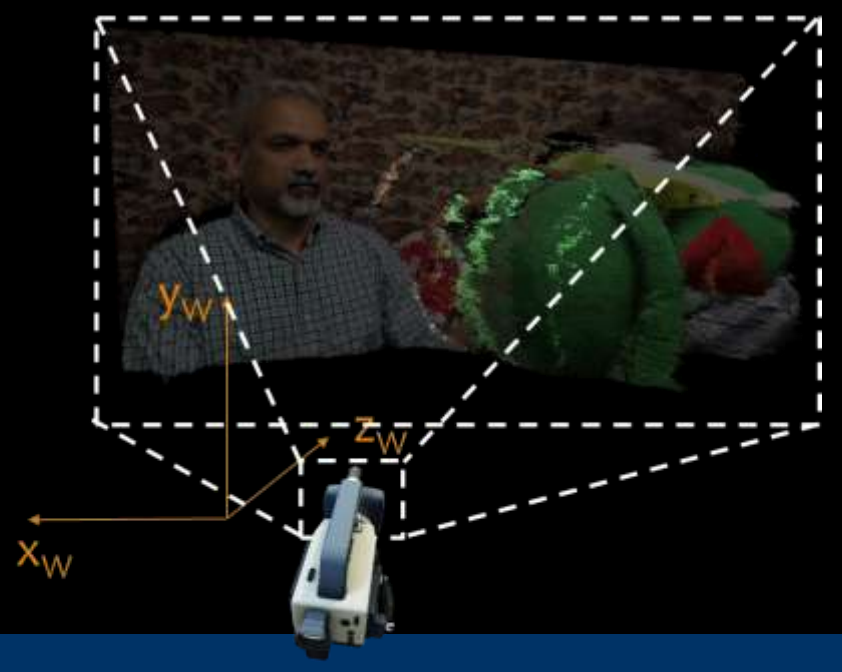
- cpl_cam_use_OMAF_coordinate_system_flag, cpl_cam_pos_x, cpl_cam_pos_y, cpl_cam_pos_z, cpl_cam_yaw, cpl_cam_pitch, cpl_cam_roll, cpl_cam_scaling_x, cpl_cam_scaling_y, cpl_cam_scaling_z




$$\begin{vmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} * \begin{bmatrix} R_{L2W} & T_{L2W} \\ 0 & 1 \end{bmatrix} * \underbrace{\begin{vmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}}_{\text{To OMAF coordinate system,}}$$



# Coding MIV content with EE2.4 (5 frames only)

- Intel Frog sequence


MIV Atlas generation

V-PCC packing/encoding/3D reconstruction

Bitstream stat:

Header: 16 B 128 b
vpccUnitSize[ VPCC_SPS ]: 890 B 7120 b
vpccUnitSize[ VPCC_PDG ]: 7261 B 58088 b
vpccUnitSize[ VPCC_OVD ]: 0 B 0 b ( Ocm video = 0 B )
vpccUnitSize[ VPCC_GVD ]: 1401521 B 11212168 b ( Geo video = 1401441 B + 0 B + 0 B + 0 B )
vpccUnitSize[ VPCC_AVD ]: 2720282 B 21762256 b ( Tex video = 2720202 B + 0 B )
TotalMetadata: 8327 B 66616 b
TotalGeometry: 1401441 B 11211528 b
TotalTexture: 2720202 B 21761616 b
Total: 4129970 B 33039760 b

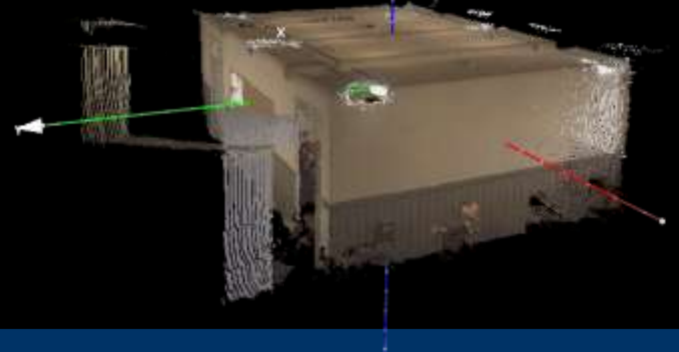
 

# Coding MIV content with EE2.4 (5 frames only)

- Classroom sequence

MIV Atlas generation

V-PCC packing/encoding/3D reconstruction



Bitstream stat:

Header: 16 B 128 b

vpccUnitSize[ VPCC_SPS ]: 874 B 6992 b

vpccUnitSize[ VPCC_PDG ]: 4978 B 39824 b

vpccUnitSize[ VPCC_OVD ]: 0 B 0 b ( Ocm video = 0 B )

vpccUnitSize[ VPCC_GVD ]: 165244 B 1321952 b ( Geo video = 165228 B + 0 B + 0 B + 0 B )

vpccUnitSize[ VPCC_AVD ]: 2532094 B 20256752 b ( Tex video = 2532078 B + 0 B )

TotalMetadata: 5900 B 47200 b

TotalGeometry: 165228 B 1321824 b

TotalTexture: 2532078 B 20256624 b

Total: 2703206 B 21625648 b

# Conclusion

- We showed here that there are several similarities between both MIV and V-PCC, whereas MIV content can be interpreted as perspective or ERP cameras using reversed depth in camera space, while V-PCC cameras can be considered as orthographic cameras with depth in world space.
- V-PCC syntax could be considered in MIV case for a more flexible coding approach. For example, MIV patch generation could be done per frame, occupancy map with pixel precision could also be adopted.
- We recommend to continue the study, and look for more opportunities to improve the unification of both approaches.