PHILIPS

MIV tutorial
IEEE VCIP 2021

Part 1

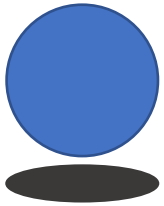
Bart Kroon, bart.kroon@philips.com

What is immersive video?

Immersive video gives 6 degrees of freedom

Viewer position

- The viewer may:
  - Translate left/right
  - Translate up/down
  - Translate forward/back
- Within a *3D viewing space*



Viewer orientation

- The viewer may:
  - Yaw left/right
  - Pitch up/down
  - Roll left/right
- Within the *field of view*

Immersive → *6DoF viewing space*

Video → *time axis*

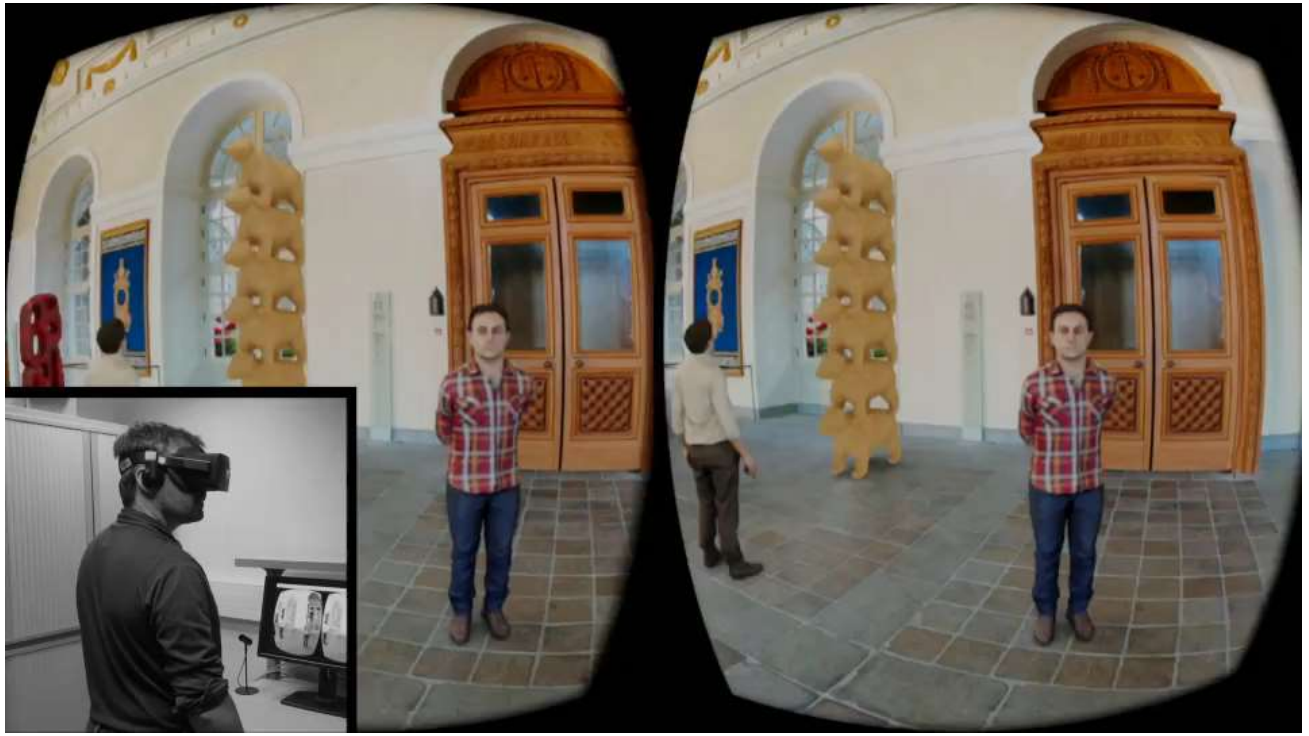
Use case:
Virtual reality

Applications: Cinematic, Educational, Training, Medical, Professional, Telepresence

Level of immersion: Sitting/standing, Some steps

Experience: be somewhere else

Client device: Closed HMD, Eye-tracked 3D



Use case:
Free navigation

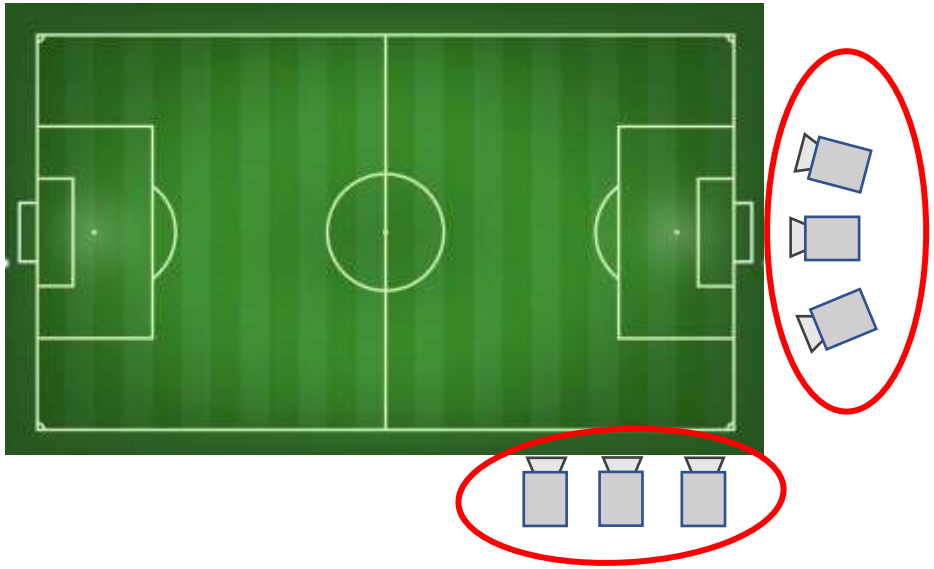
Applications: Sports (live, replays), Educational, Training, Medical, Professional, Conferencing

Level of immersion: Sitting/standing, Virtual window, Outside-in

Experience: Ability for a user to change the viewpoint

Client device: Phone, Tablet, Laptop/PC, TV, Eye-tracked 3D

Possible camera rig on basketball field



Possible camera rig positioning near goal on soccer field

Use case:
Light-field display

Applications: Cinematic, Educational, Training, Medical, Professional, Telepresence, Conferencing

Level of immersion: Virtual window

Experience: Be somewhere else, Ability to change viewpoint

Client device: Eye-tracked 3D, Light-field display

Use cases of immersive video

Application domains

Sports (live, replays)
Cinematic
Educational
Training
Medical
Professional
Telepresence
Conferencing

Level of immersion

Sitting/standing [VR]

Virtual window [VR, FN]
Some steps [VR, FN]
Outside-in [AR, XR, FN]

Full 6DoF [VR, XR, FN]

Experience

Virtual reality (VR)
(be somewhere else)

Augmented reality (AR)
(bring something here)

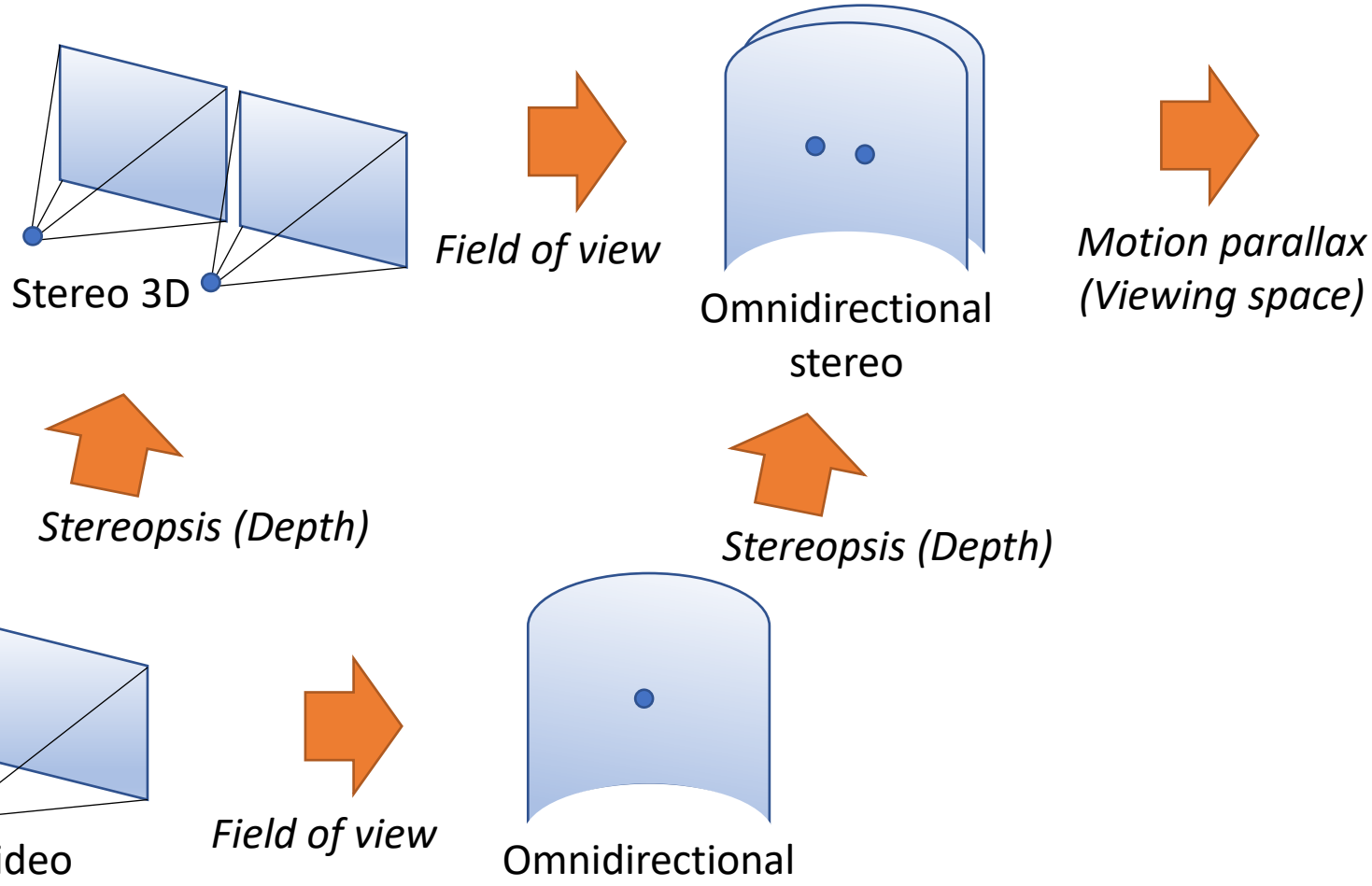
Mixed reality (XR)
(interaction between physical and virtual scene elements)

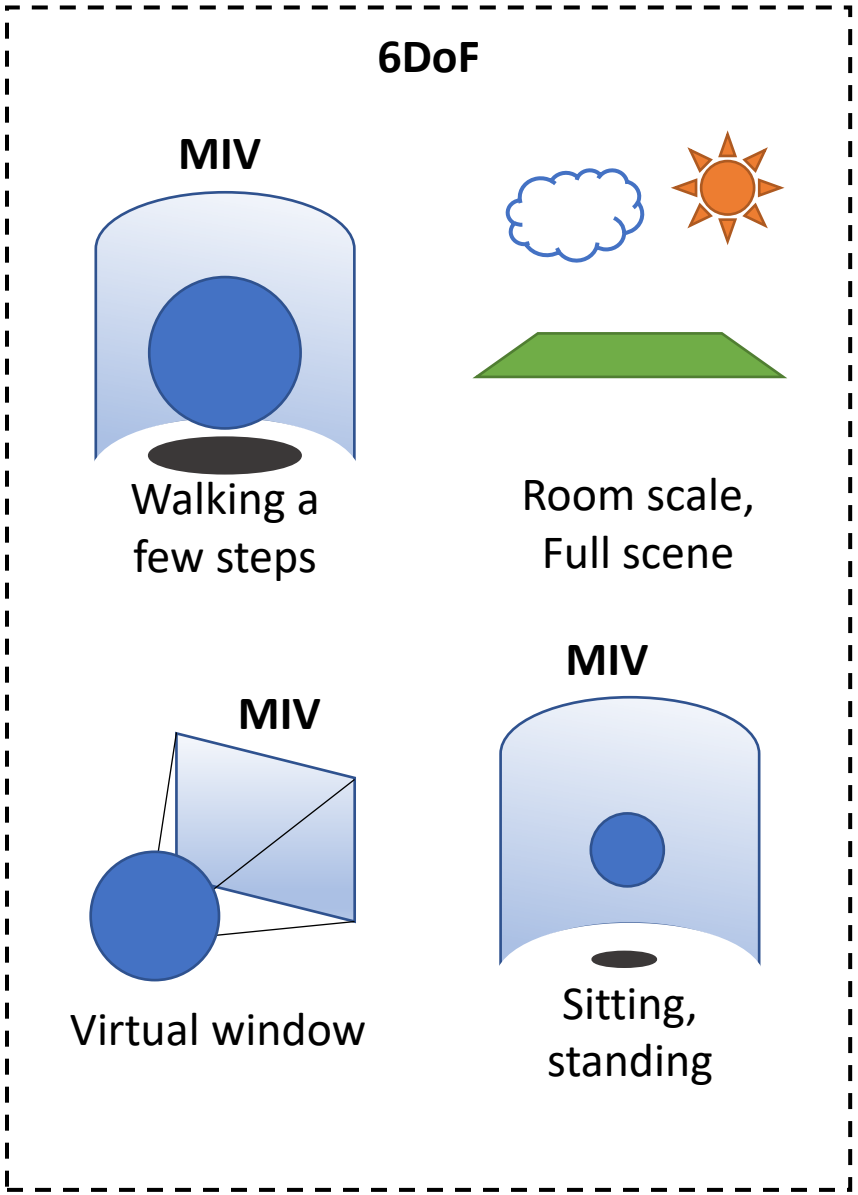
Free navigation (FN)
(ability for a user to change the viewpoint)

Client device

Phone [AR, XR, FN]
Tablet [AR, XR, FN]
Laptop/PC [FN]
Television [FN]
Closed HMD [VR]
See-trough HMD [XR]
Eye-tracked 3D [VR, FN]
Light-field display [VR, FN]

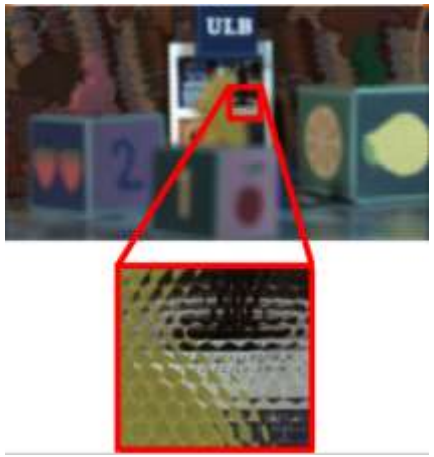
Levels of immersion





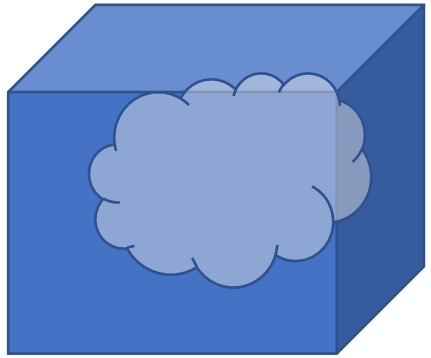
Possible representations for immersive video

Densely sampled light field [rays]



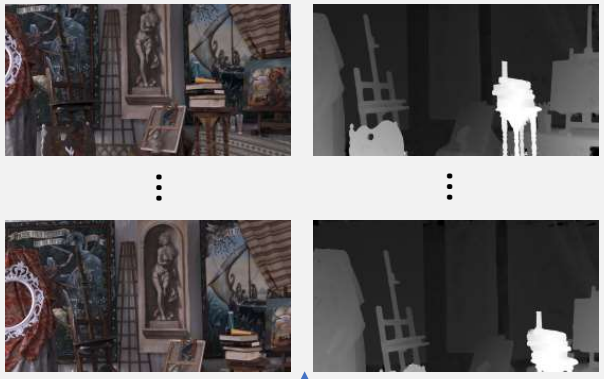
Source: Université Libre Bruxelles

Densely sampled volume [voxels]



MIV

Multiview + depth (MVD) [pixels]

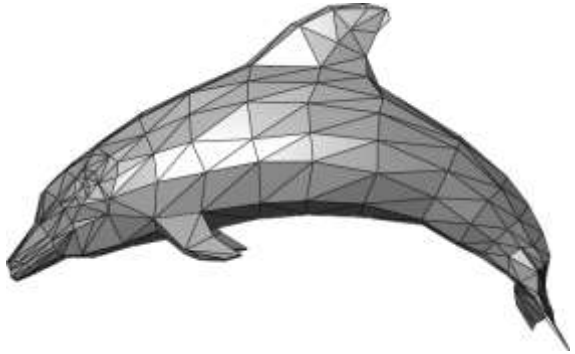


Multi-planar (MPI / MSI) [pixels]

Source: Interdigital

Point cloud [points]

Mesh [polygons]

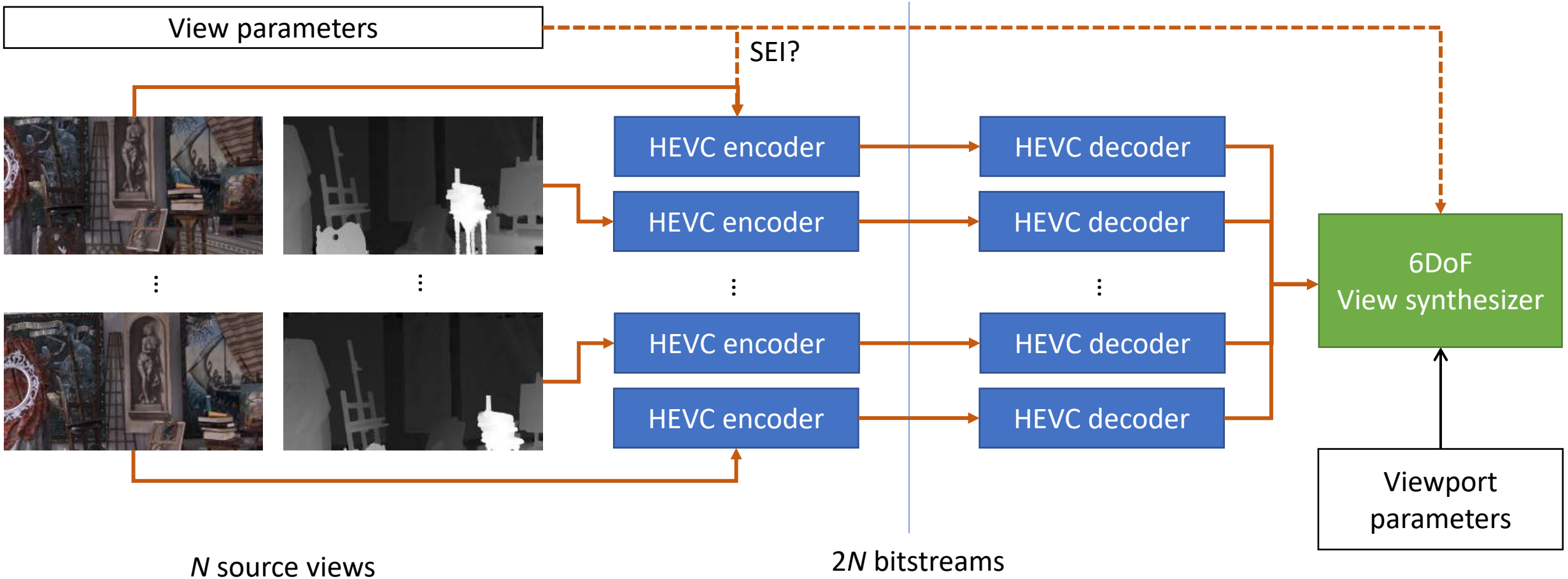


https://en.wikipedia.org/wiki/File:Dolphin_triangle_mesh.png

Existing codecs

Multicast HEVC

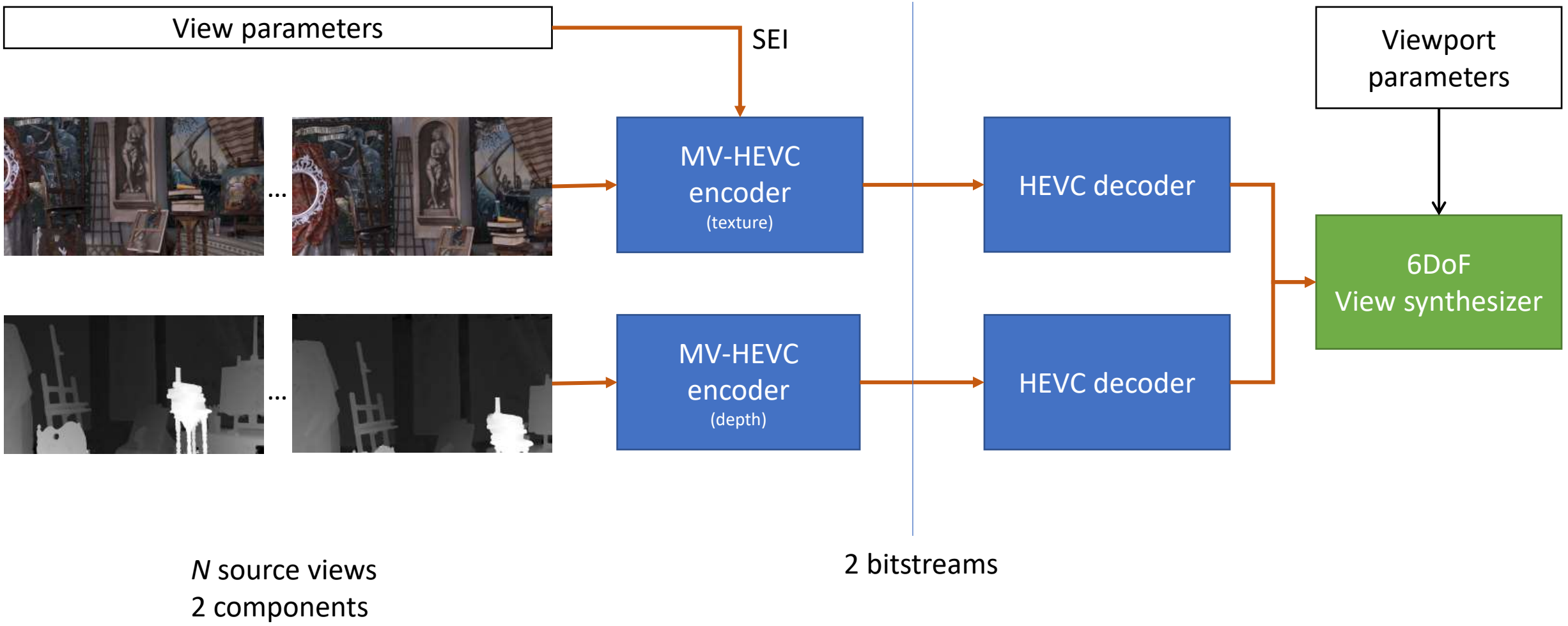
ISO/IEC 23008-2 | ITU-T H.265, also other 2D video codecs



SEI = Supplemental Enhancement Information

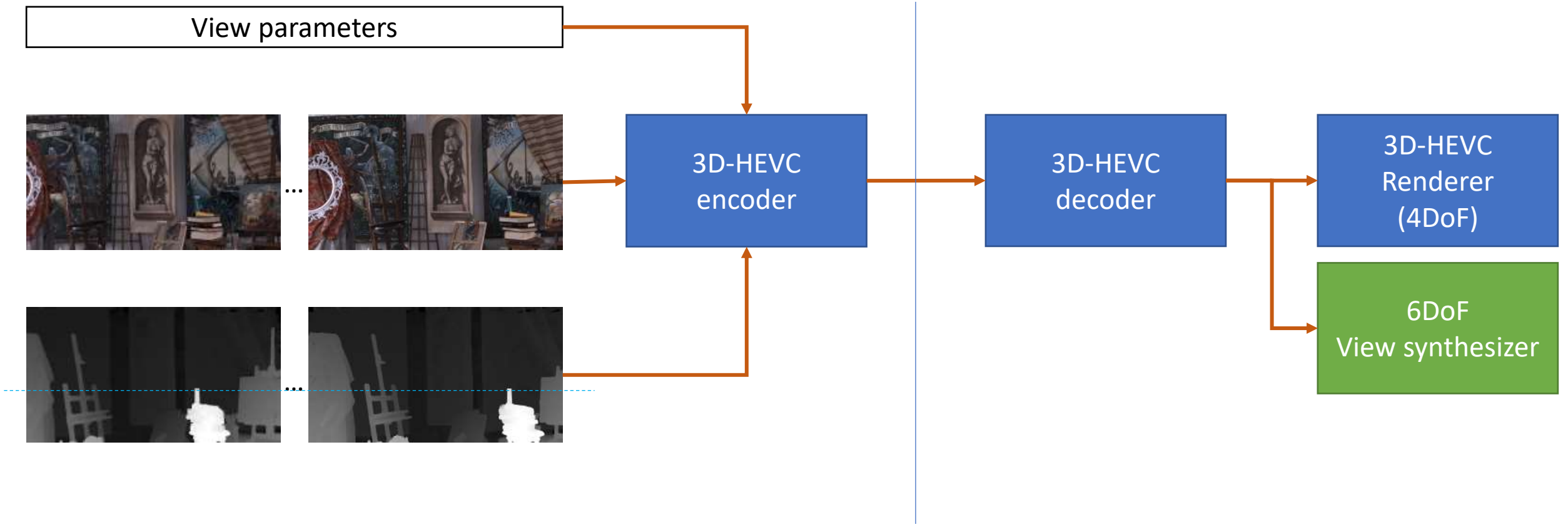
Multiview HEVC (MV-HEVC)

Annex G of ISO/IEC 23008-2 | ITU-T H.265



3D-HEVC

Annex I of ISO/IEC 23008-2 | ITU-T H.265



MPEG Immersive video (MIV)

ISO/IEC 23090-12

How it started…

- MPEG-I Phase 1b requirements: **October 2018**
  - Head motion parallax
  - Small viewing space (sitting/standing)
- “pixels” → Investigation by MPEG video coding **January 2018**
- Creating of an anchor based on multicast HEVC and a purposely built view synthesizer (RVS)
- Call for Evidence → provided by Technicolor
- Call for Proposals → 5 proposals from 7 organizations **March 2019** (ZJU, PUT/ETRI, Technicolor/Intel, Nokia, Philips)

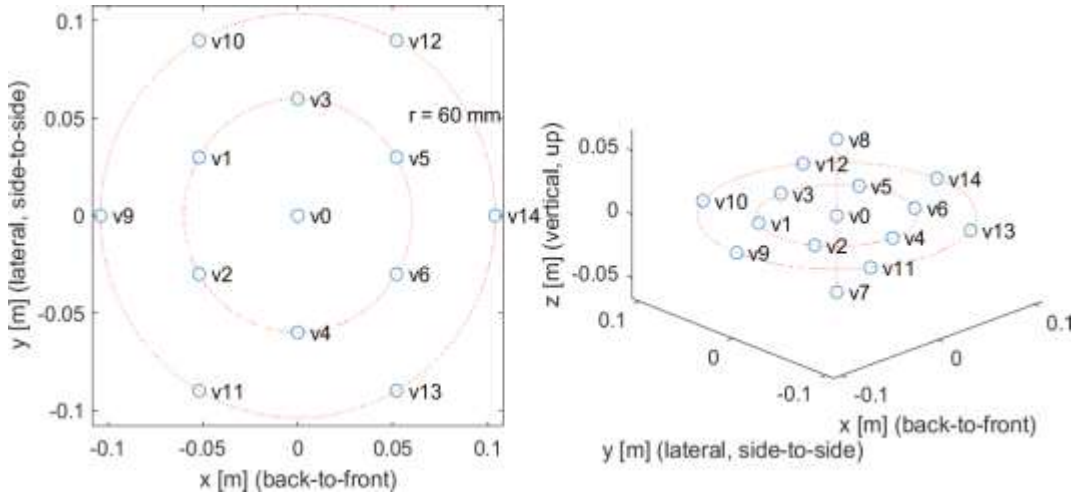
Introduction & scope

Introduction [MIV] was developed to support compression of immersive video content, in which a real or virtual 3D scene is captured by multiple real or virtual cameras. [MIV] enables storage and distribution of immersive video content over existing and future networks, for playback with 6 degrees of freedom of view position and orientation.

Scope [MIV] specifies the syntax, semantics and decoding processes for MPEG Immersive video (MIV), as an extension of [V3C]. It provides support for playback of a three-dimensional (3D) scene within a limited range of viewing positions and orientations, with 6 Degrees of Freedom (6DoF).

Example encoder sources

(Rendering of intermediate views)

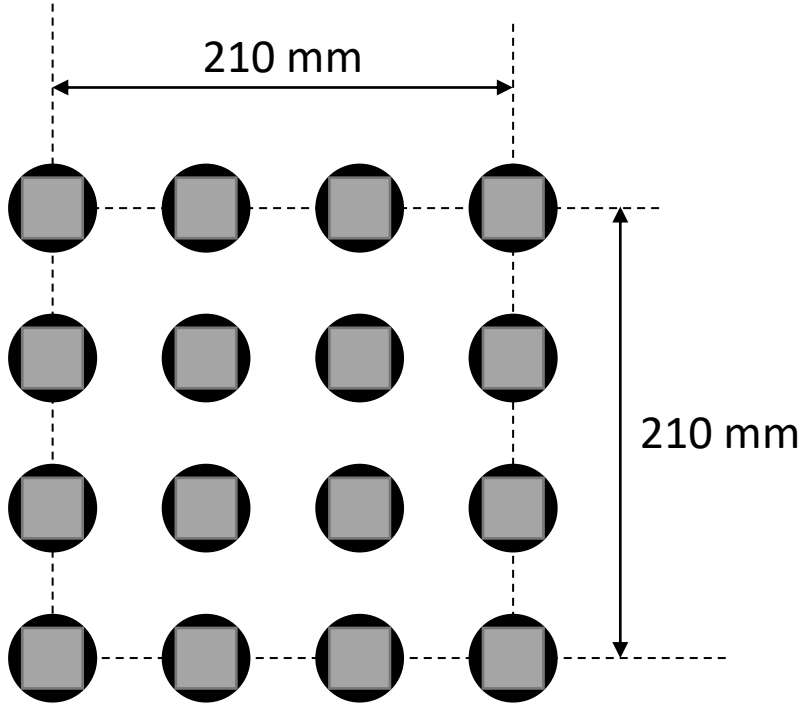


15 views (photorealistic):

- 4K × 2K
- 360° equirectangular projection (ERP)
- Geometry (=depth range)
- Texture attribute (YCbCr)

Example encoder sources

(Rendering of intermediate views)

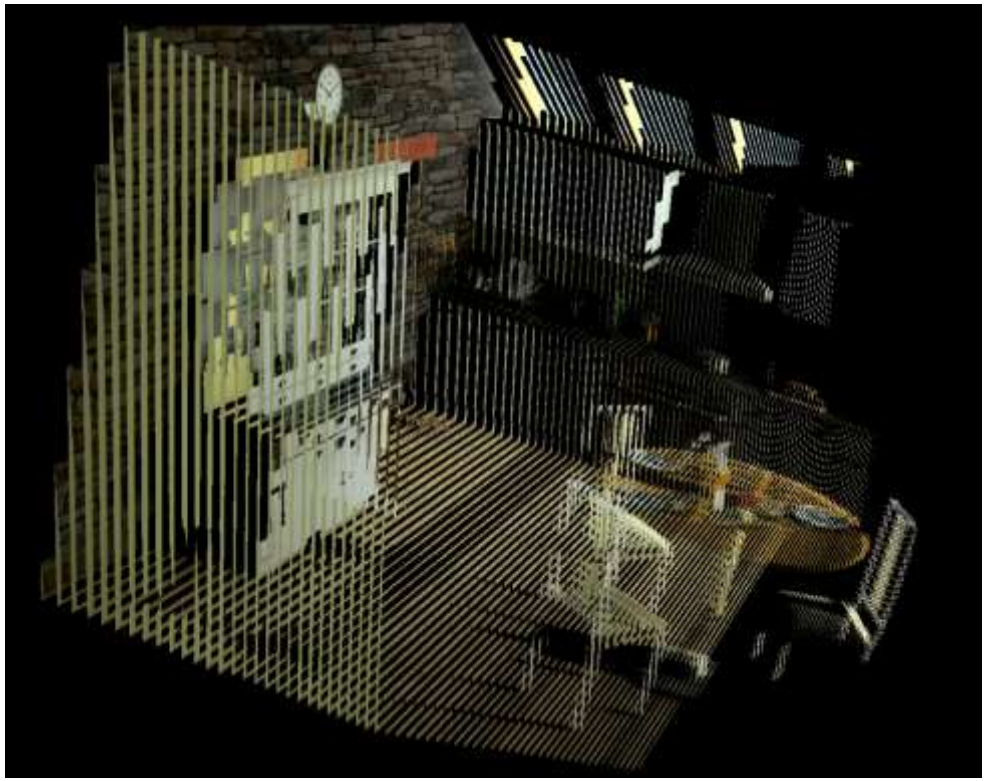


16 physical cameras:

- 2K × 1K
- Perspective projection
- Geometry (=depth range)
- Texture attribute (YCbCr)

Example encoder sources

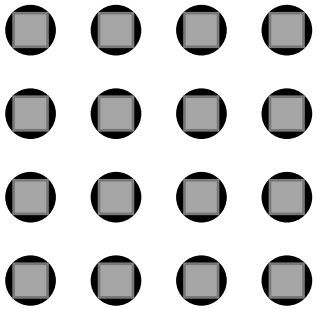
Multi-plane image (MPI)



Multi-sphere image (MSI)

MIV codec model

Transform the source to leverage any existing 2D video codec for 6DoF video



Original

MIV encoder

V3C bitstream

- Video data
- Atlas data
- View params

MIV decoder/ renderer

Reconstruction

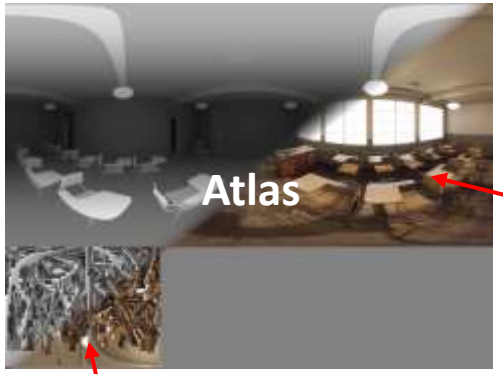
Viewing space

Multi-view or multi-plane video

- Geometry (G) or Transparency
- Texture attribute (T)
- View parameters

T

G



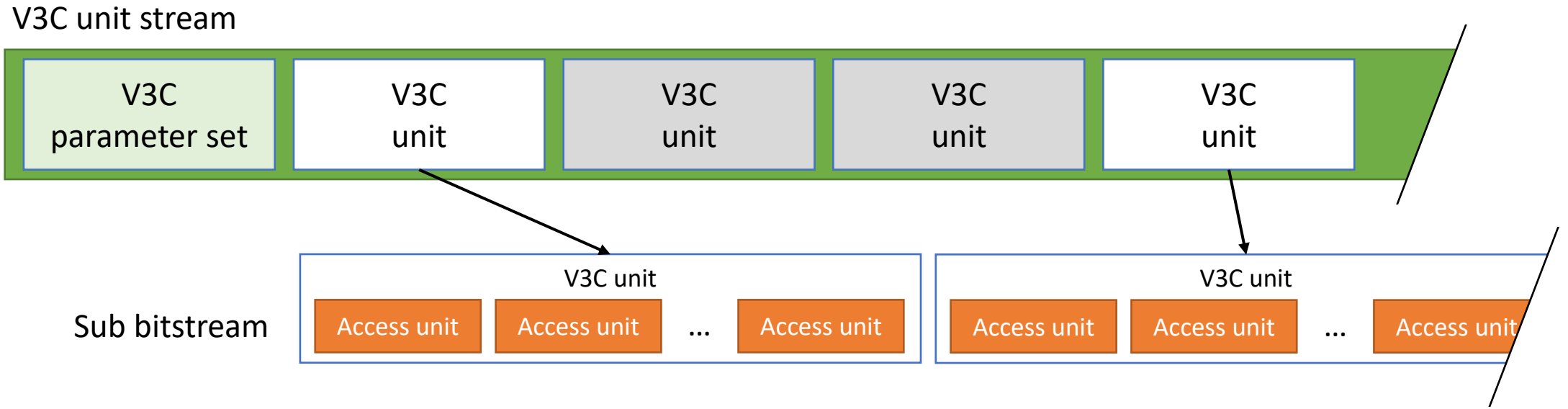
Complete (basic) view

Patches from additional views

Viewport video

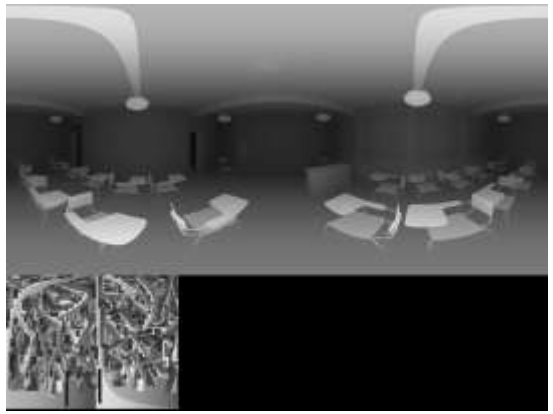
- (Geometry)
- Texture attribute

Bitstream structure



Sub bitstreams:

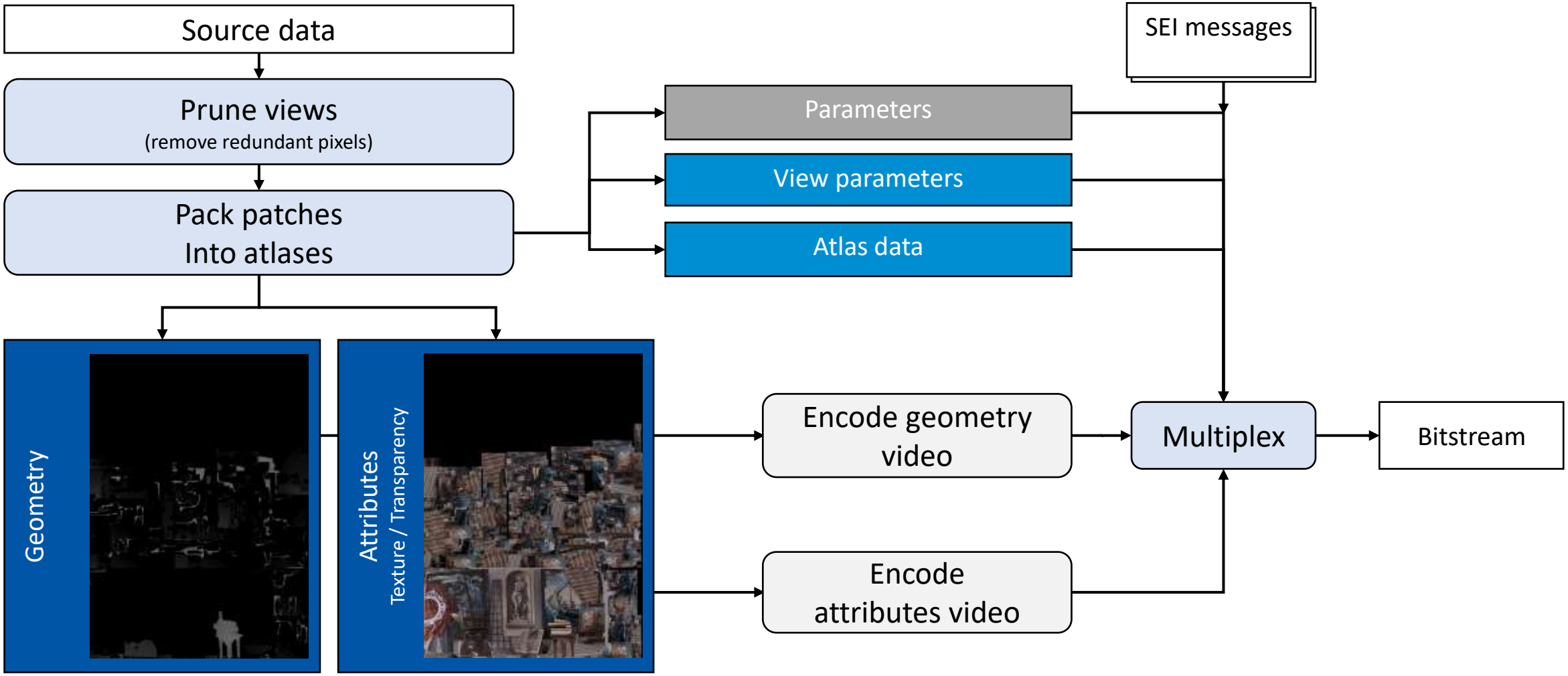
- Common atlas data (has view parameters)
- Multiple atlases:
  - Geometry video data
  - Attribute video data
  - Atlas data (has patch parameters)



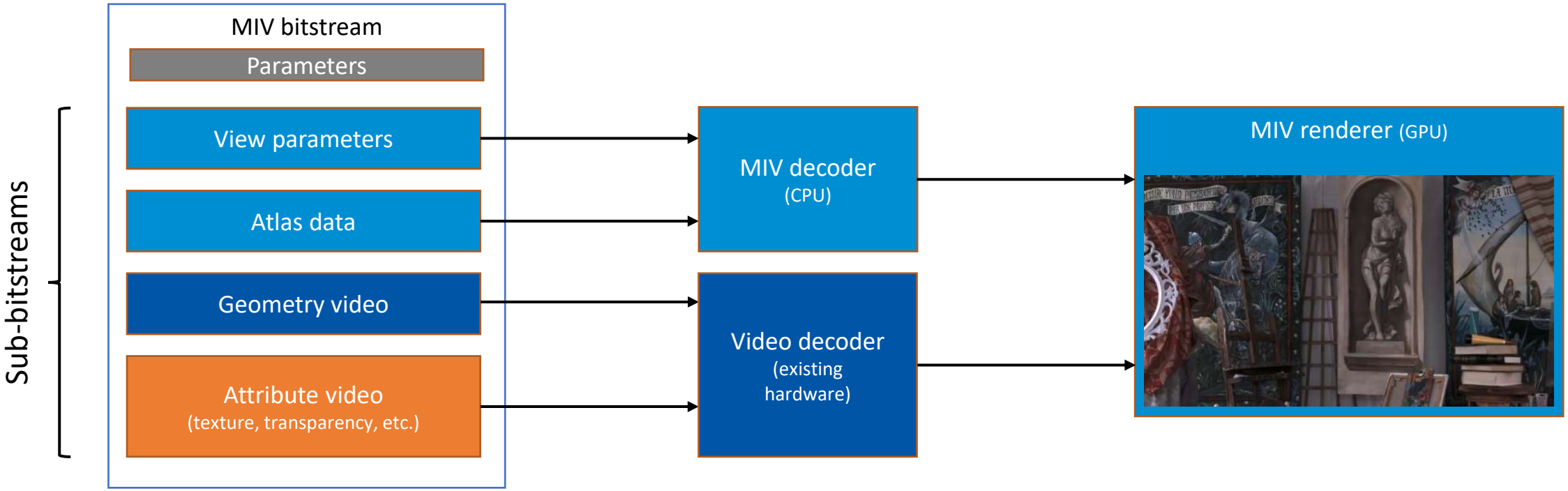
MIV test model

- Test Model for Immersive Video (TMIV)
- Public software: https://gitlab.com/mpeg-i-visual/tmiv
- Includes:
  - *Example* of an MIV encoder
  - Conforming MIV decoder
  - *Example* of an MIV renderer
  - Extra tools: multiplexer, parser, bitrate report
  - Multiple configurations for experimentation purposes

TMIV encoder model



TMIV decoder/renderer model

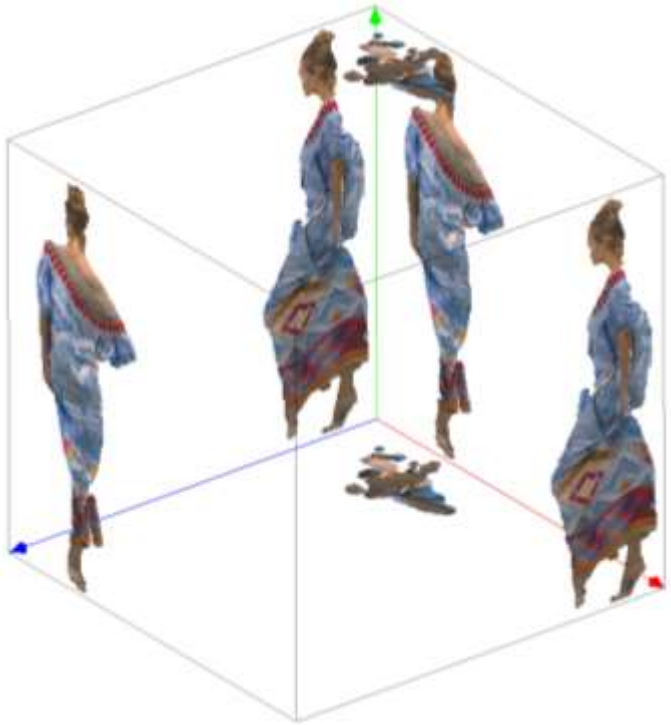


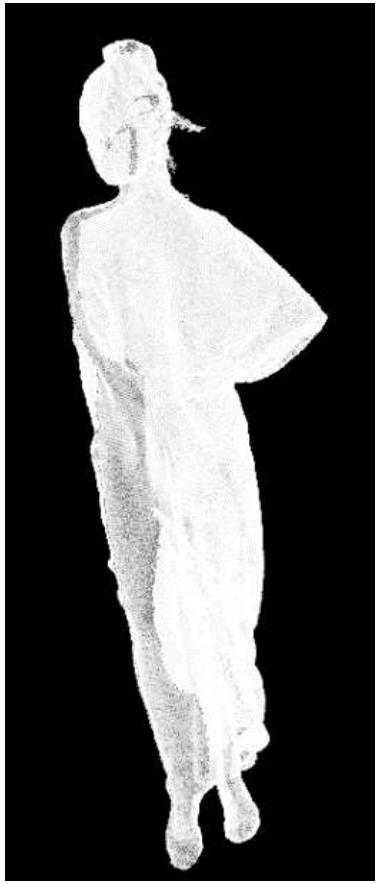
Related standards

Video-based Point Cloud Coding (V-PCC)

ISO/IEC 23090-5 Annex H

- Single atlas, but may be tiled
- Point reconstruction information
- Six orthographic projection planes

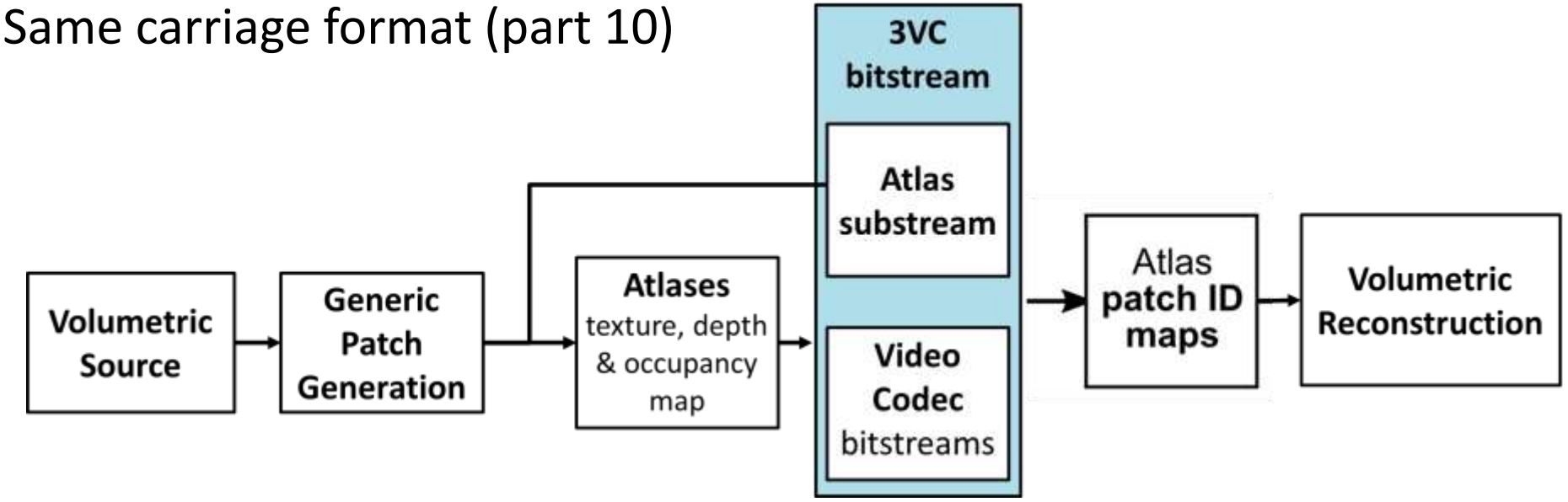




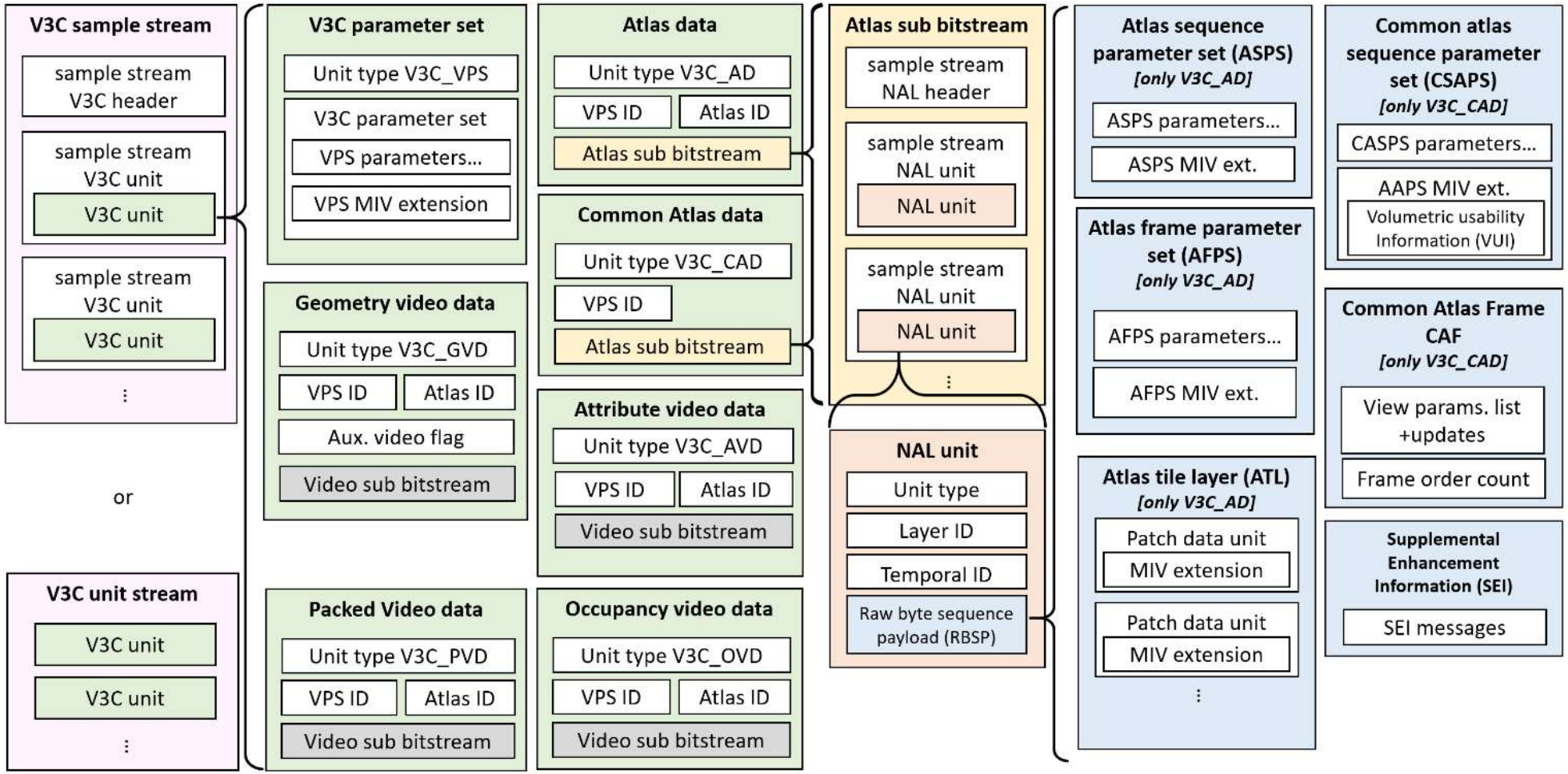
Video-based Visual Volumetric Coding (V3C)

ISO/IEC 23090-5

- V3C is the (large) intersection of MIV and V-PCC
  - Avoid duplicate specification
  - Reduce implementation effort
  - Same bitstream format
  - Same carriage format (part 10)



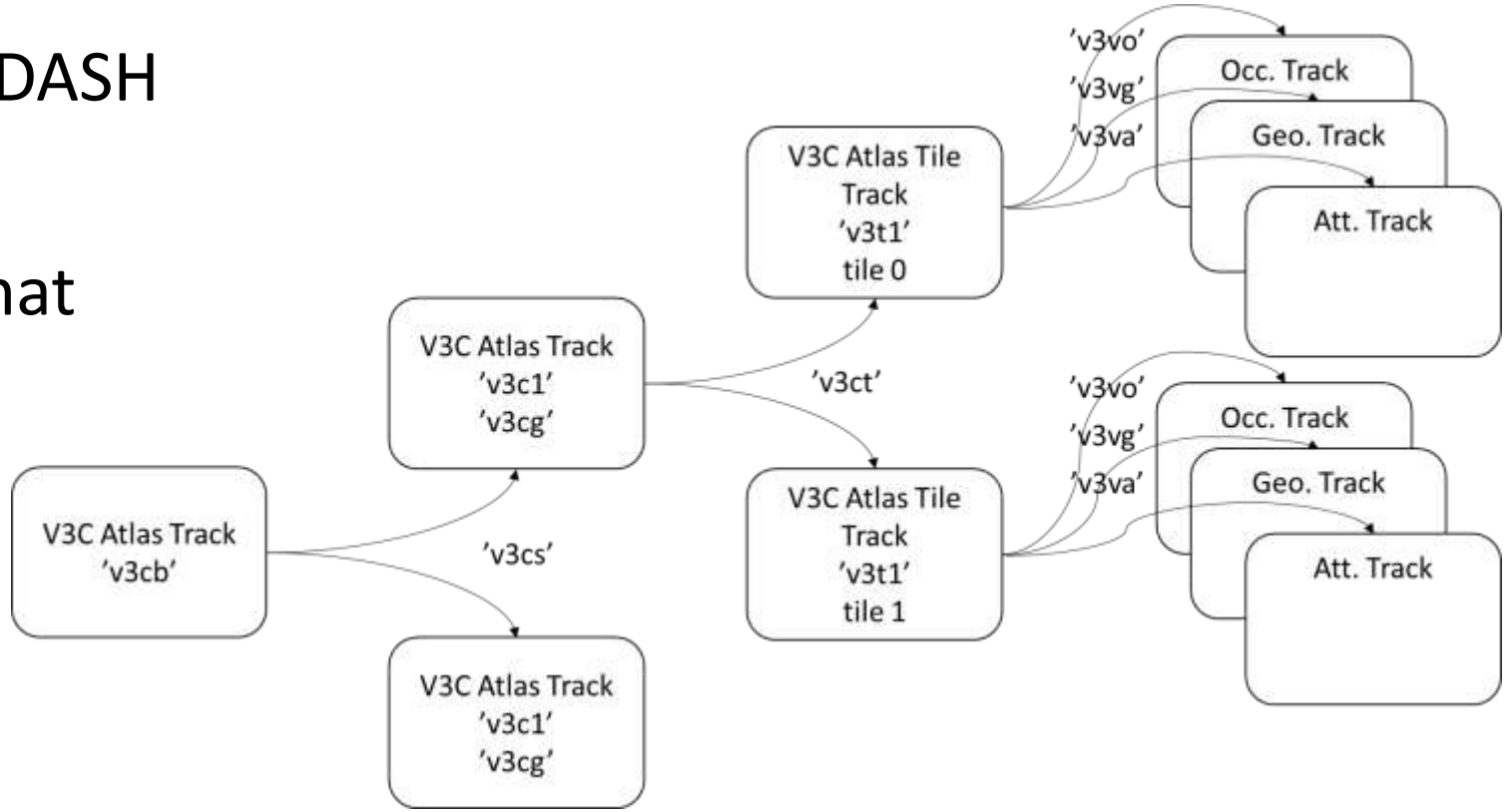
MIV: Restrictions and extensions on V3C



Carriage of V3C Data

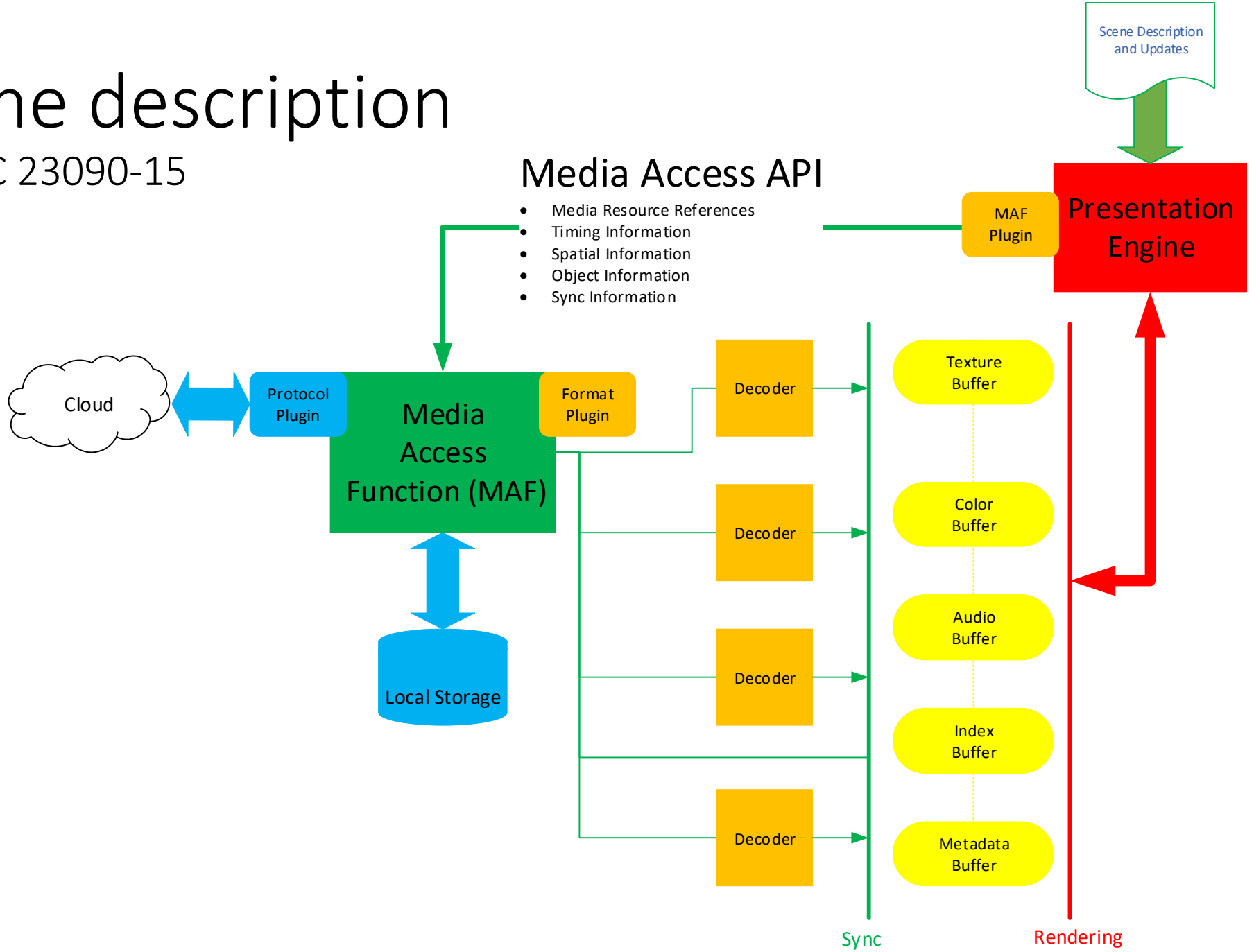
ISO/IEC 23090-10

- Single/multi-track format
- Partial access
- Encapsulation in MPEG DASH
- Encapsulation in MMT
- ISO base media file format



Scene description

ISO/IEC 23090-15



Timeline

MPEG meeting	V3C + V-PCC 1st edition	V3C 2nd edition	MIV 1st edition	Carriage of V3C 1st edition
131: Jul. 2020	Final draft (FDIS)			
132: Oct. 2020				
133: Jan. 2021			Draft (DIS)	
134: Apr. 2021	Publication			Final draft (FDIS)
135: Jul. 2021		Draft (DIS)	Final draft (FDIS)	
136: Oct. 2021	Conformance & Verification tests			Publication
137: Jan. 2022		Final draft (FDIS)	Publication	
138: Apr. 2022			Conformance & Verification tests	

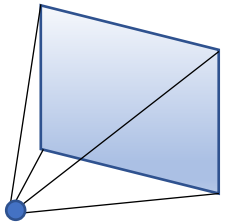
DIS = Draft international standard
FDIS = Final draft international standard

Features of the standard

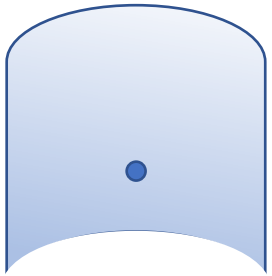
View parameters

Projection plane

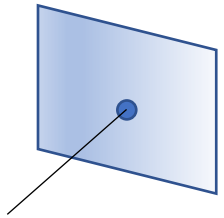
Intrinsics



Perspective projection
(f_u, f_v, p_u, p_v)

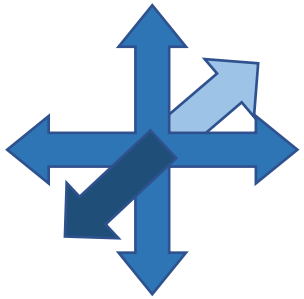


Equirectangular projection
$(\theta_1, \theta_2, \phi_1, \phi_2)$

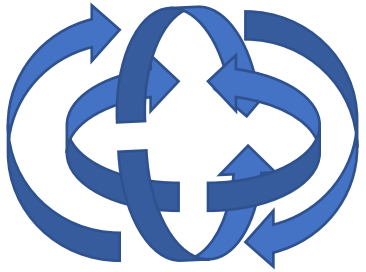


Orthographic projection
(w, h)

Extrinsics

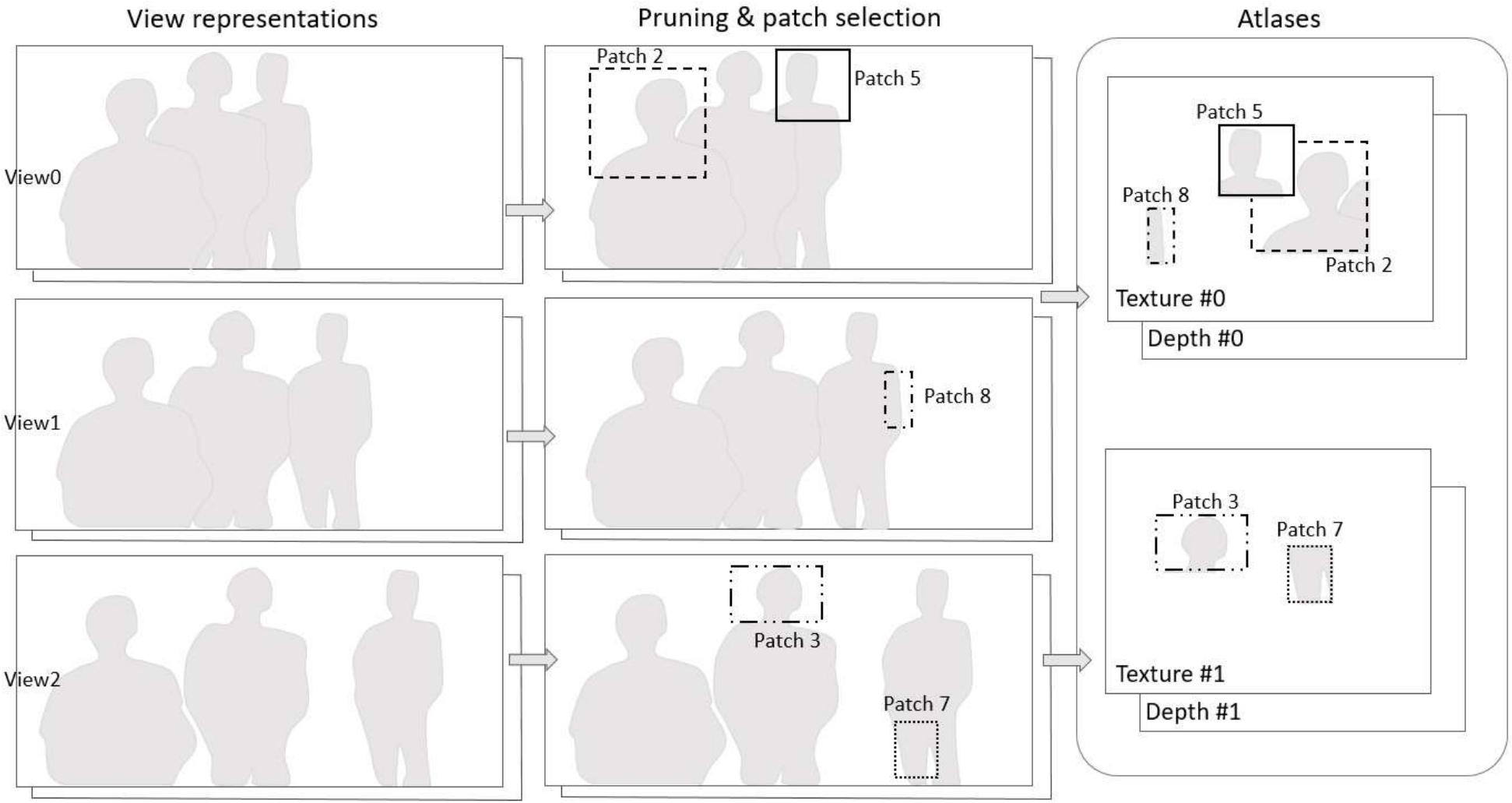


Position
(x, y, z)



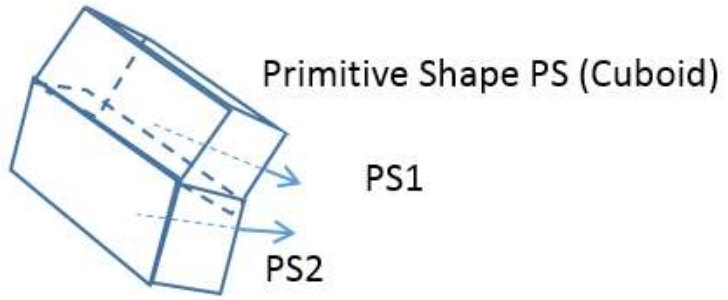
Orientation
(α, β, γ)

Patch parameters

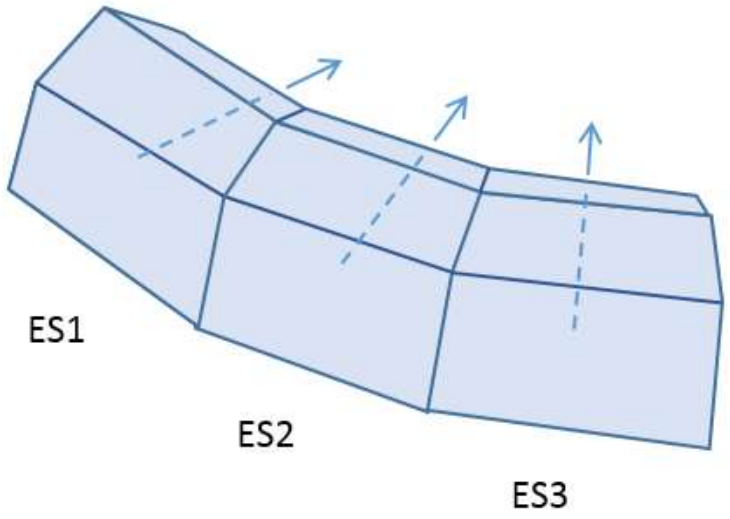


Viewing space definition

- Definition of a viewing space
  - Spatial (3D)
  - Spatial + field of view (5D)
- Can be as simple as a box or a sphere
- Constructive solid geometry (CSG)
- Signed distance function
  - < 0: Inside
  - = 0: Edge
  - > 0: Outside



Elementary Shape ES= PS1 ∪ PS2

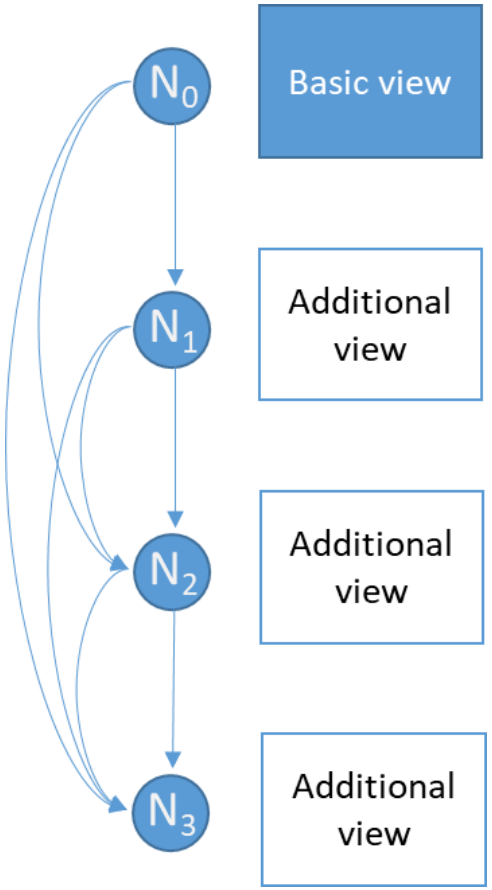


Viewing Space VS= ES1 ∪ ES2 ∪ ES3

Viewing space handling

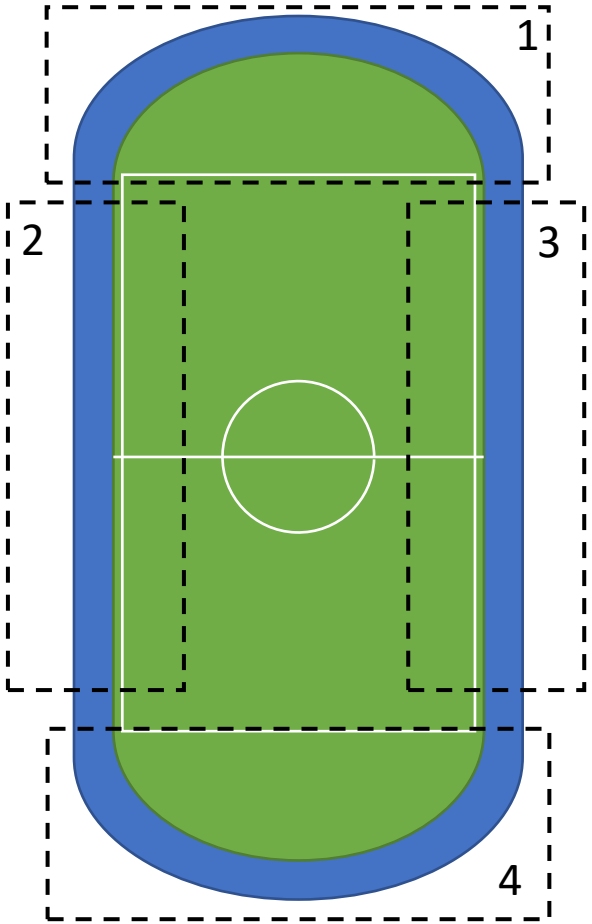
- SEI message to convey the creative intent regarding the viewing space
  - For device: all, HMD, phone, 3D display
  - For application: all, AR, VR, web, scene description
  - What happens when the viewer moves outside the viewing space:
    - Default client behaviour
    - Always render, even when there would be artifacts
    - Fade out
    - Extrapolate scene colours
    - Reset the viewer position
    - Stretch viewer pose to stay within the viewing space
    - Rotate the viewer to stay within the field of view

View pruning graph



- Provide information on which view was used to prune which other views.
- Useful for rendering algorithms that rely on view blending.

Group-based encoding



MIV encoder 1

MIV encoder 2

MIV encoder 3

MIV encoder 4

Merge bitstreams

Transmission over network

Select bitstreams

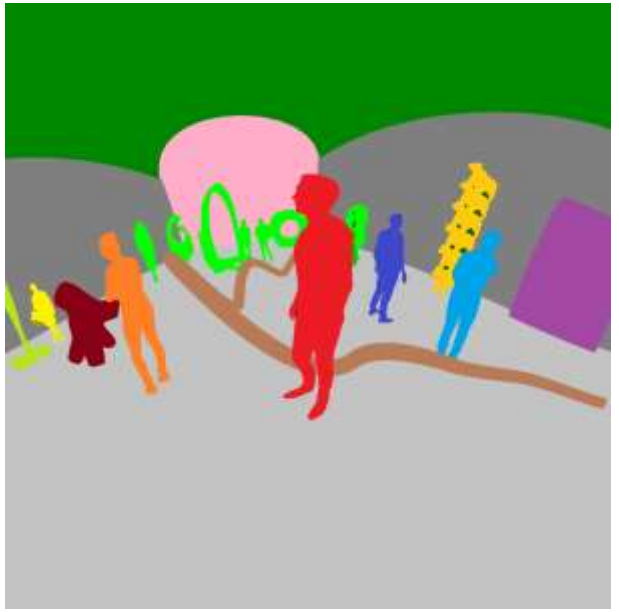
Decode & render

Request video track

Entity-based coding

Texture attribute

Geometry
(Normalized disparity)



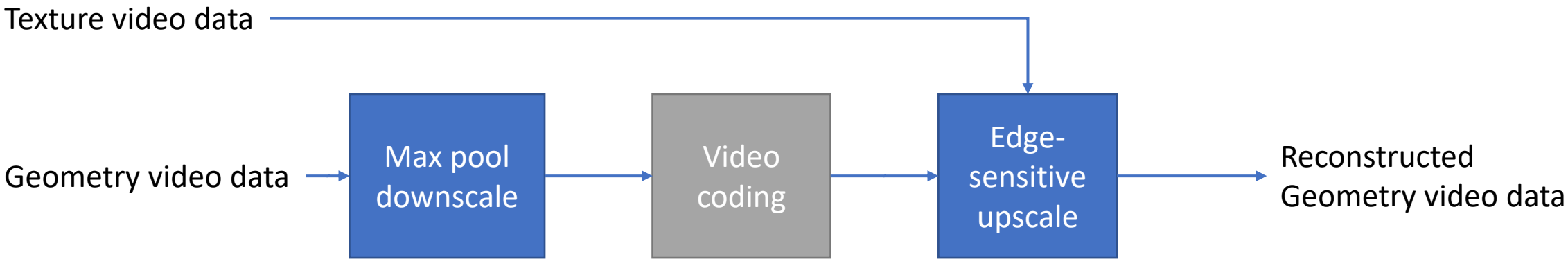
Entity ID map

Not coded as a separate video component → Each patch has an entity ID

Server-side inpainting

- Move inpainting to the server (more info, more resources)
- Inpaint patches or whole views
- The renderer may use them to fill up disocclusion zones.
- TMIV encoder renders a low-res background from the source views

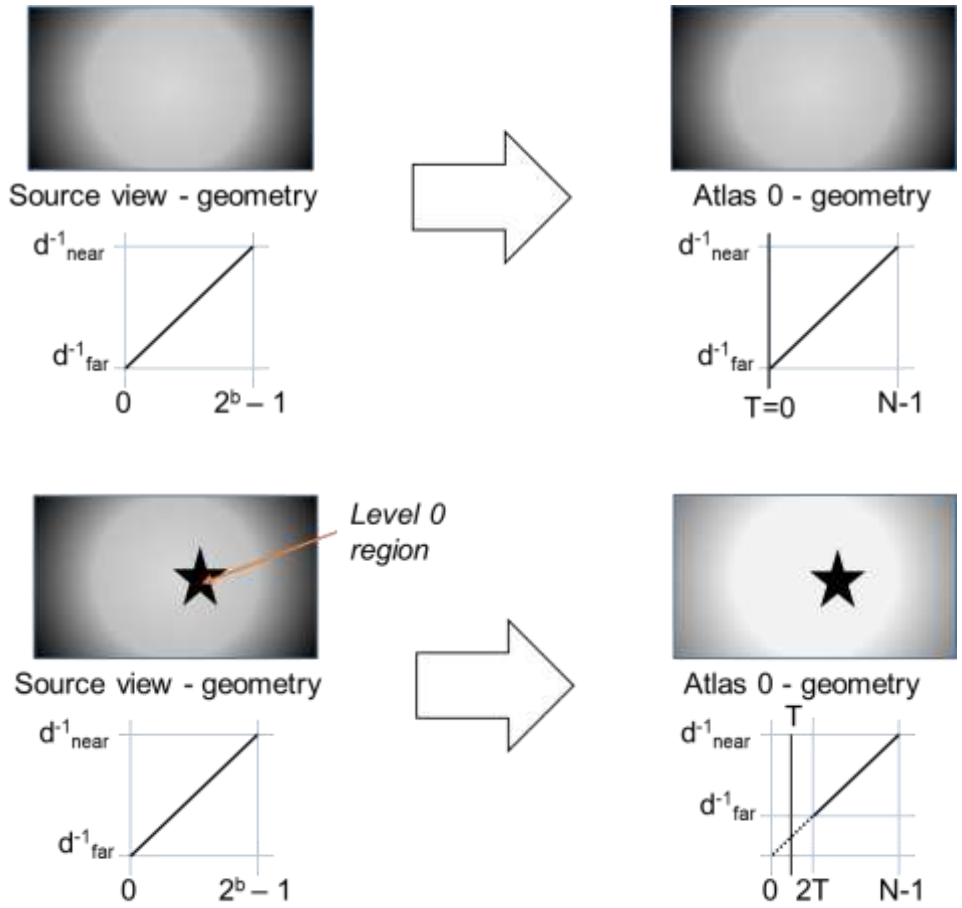
Geometry scaling



- Video encoders are not designed to encode geometry
- Geometry downscaling enables lower quantization (QP) parameters for a given bitrate
- Preserve thin foreground edges

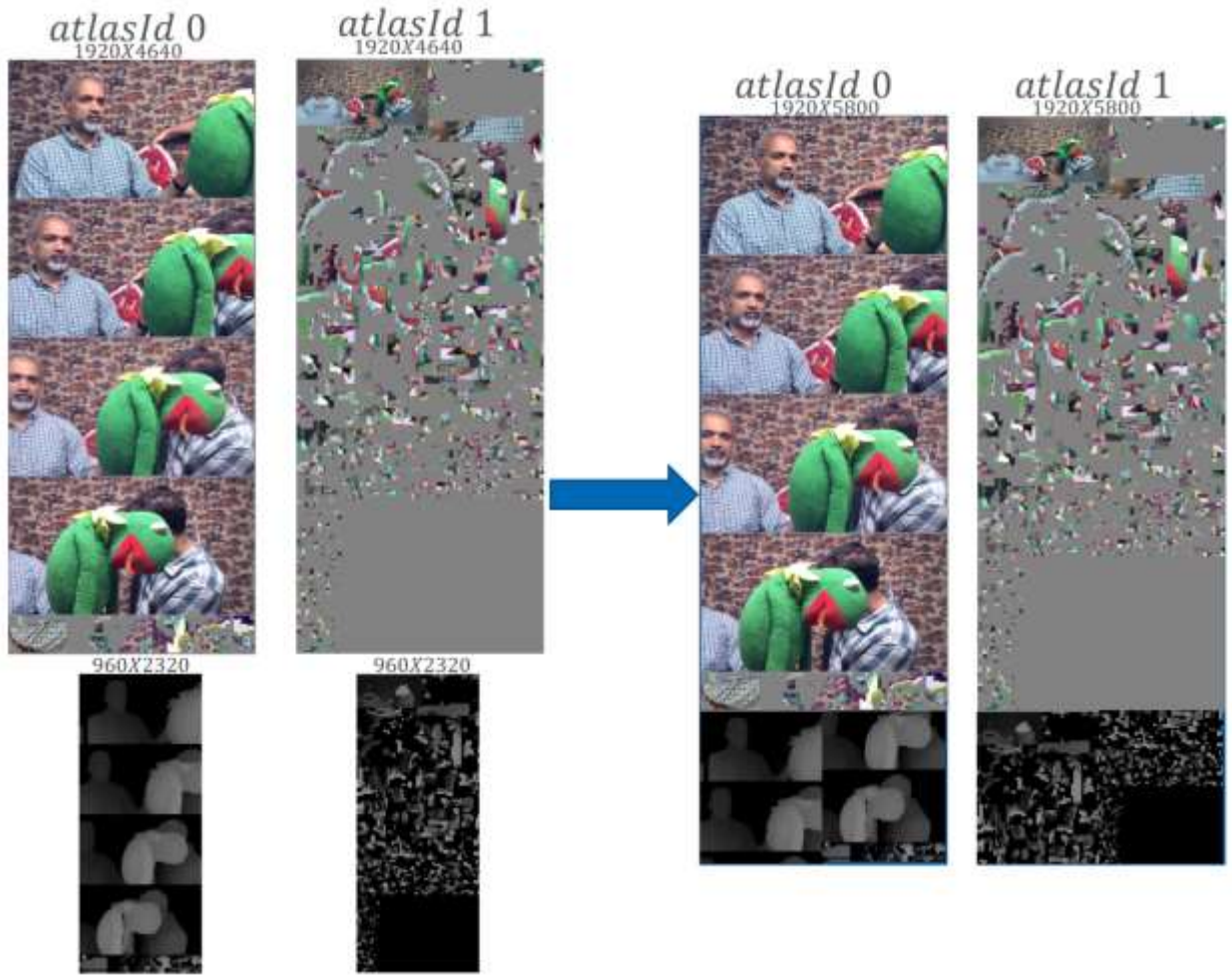
Representation of occupancy information

- **Full patch occupancy**
  All sample values of a patch are occupant
- **Embedded occupancy**:
  Occupancy embedded in geometry video data
- **Explicit occupancy:**
  Occupancy video data
  Full resolution or subsampled



Frame packing

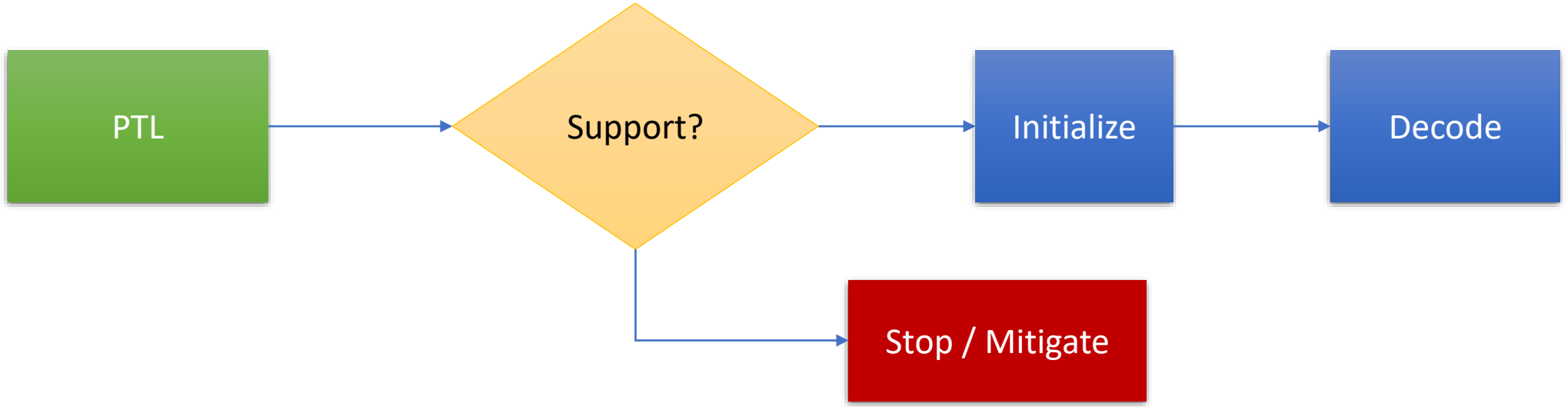
- Reduce the number of video decoder instantiations by packing multiple components into one video frame.
- Works well with (but does not require) HEVC MCTS or VVC sub-pictures.



Profile and sub-profiles

Profile, tier and level information

- Profile = subset of the standard
  - Codec group: MPEG-4 AVC, HEVC, VVC, MP4RA
  - Toolset: V-PCC …, MIV …
- Tier = main (to client device), high (production, archival)
- Level = limits on samples, pixels, bits, etc.



Profile: MIV Main

- Useful for multiview + depth:
  - Computer graphics
  - Natural content with depth est.
- **Geometry video data**
- **Texture video data** (opt.)
- Full or **embedded occupancy**
- Patches and full views

Real-time immersive video on Android phone

Martijn Govers, Bart Sonneveldt, Hugo Habets, Chris Varekamp, Andy Willems, Bart Kroon

- MIV Main
- Two HEVC Main10 video bitstreams
  - Texture
  - Geometry
- Full views + inpainted background
- Computer graphics and natural content
- One V3C sample stream per second

- Phone with interactivity

Profile: MIV Extended

- Full or embedded occupancy, or **occupancy video data**
- **Constant depth per patch** or geometry video data
- Texture video data (opt.)
- **Transparency video data** (opt.)
- **Packed video data** (with any of the above video components)

Immersive video playback on Intel Xe Max

Max Dmitrichenko, Basel Salahieh, Mengyu Chen, Penne Lee, Jill Boyce

- MIV Extended
- Single HEVC Main10 video bitstream
- Packed video with texture and geometry
- Only full views
- Atlas data carried in vendor-specific SEI message

- 2D display with eye tracking

MIV Extended with Restricted Geometry

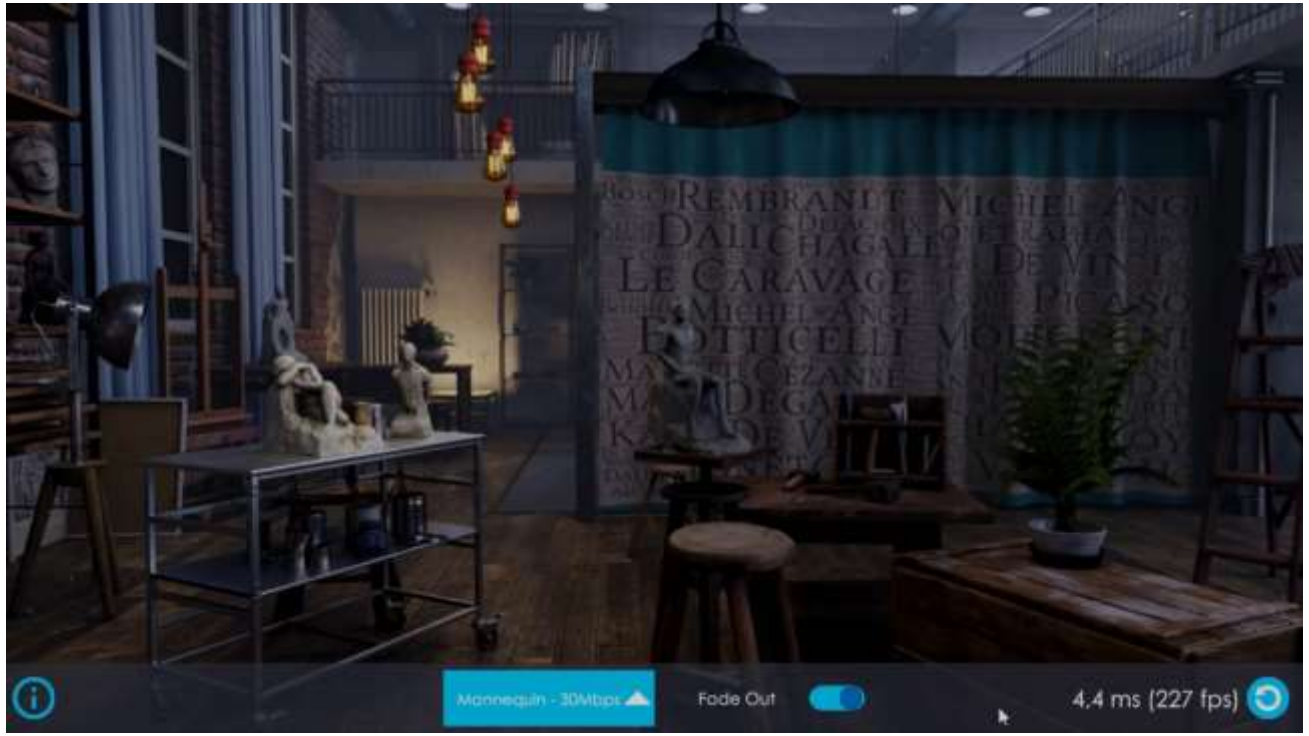
- Simplified rendering due to non-intersecting patches (inverse painters algorithm)
  - Full occupancy
  - Constant depth per patch
  - Texture video data
  - Transparency video data
  - Packed video data (opt.)

Multi-Plane Images for Free Viewpoint

Julien Fleureau, Renaud Doré

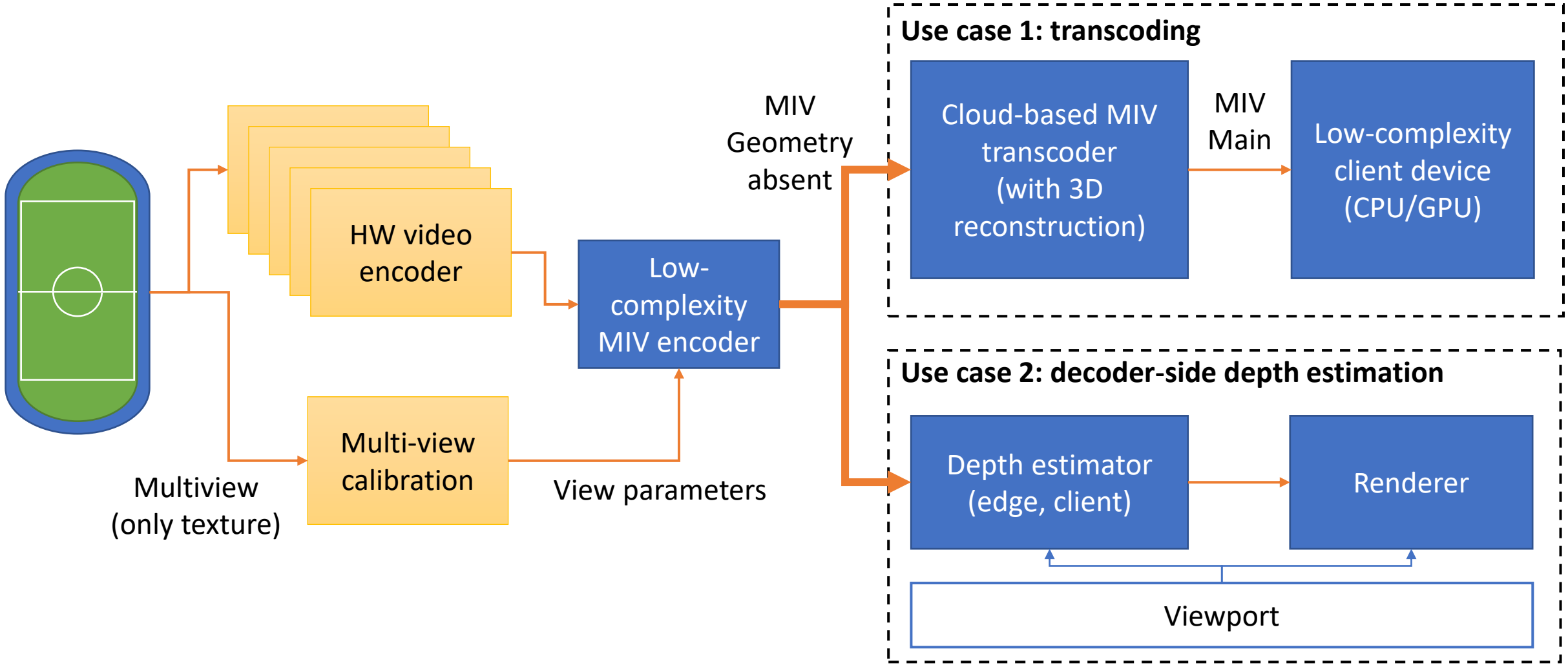
- MIV Extended w. Restricted Geometry
- Two HEVC Main10 video bitstreams
  - Texture
  - Transparency
- Multi-plane image strips packed in atlas

- Mobile, PC



Profile: MIV Geometry absent

Postpone 3D reconstruction



Preparation of MIV edition-2 has started

Please participate

Preliminary schedule

- 21Q4 Draft requirements
- 22Q1 Final requirements (WD)
- 22Q2 138 WD
- 22Q3 139 WD
- 22Q4 140 CD
- 23Q1 141 DIS
- 23Q3 143 FDIS

Draft use cases and requirements

- Evolution of MIV 1
- Larger viewing spaces
- Non-Lambertian content
- Scenes with static and dynamic elements
- Object-based coding
- Bullet time

MIV tutorial

- MIV website: https://mpeg-miv.org
- Public software:
  - Test model: https://gitlab.com/mpeg-i-visual/tmiv
  - IV-PSNR: https://gitlab.com/mpeg-i-visual/ivpsnr
  - IVDE: https://gitlab.com/mpeg-i-visual/ivde
  - RVS: https://gitlab.com/mpeg-i-visual/rvs
- Demo videos:
  - Freeport player: https://youtu.be/UeT_Xm1jBGs
  - Multi-plane images: https://youtu.be/DPMMzj2I6Yw
  - Real-time RVS: https://lisaserver.ulb.ac.be/rvs/
- Test material (14+ sequences) available on request

Q&A

End of part 1

MIV tutorial
IEEE VCIP 2021

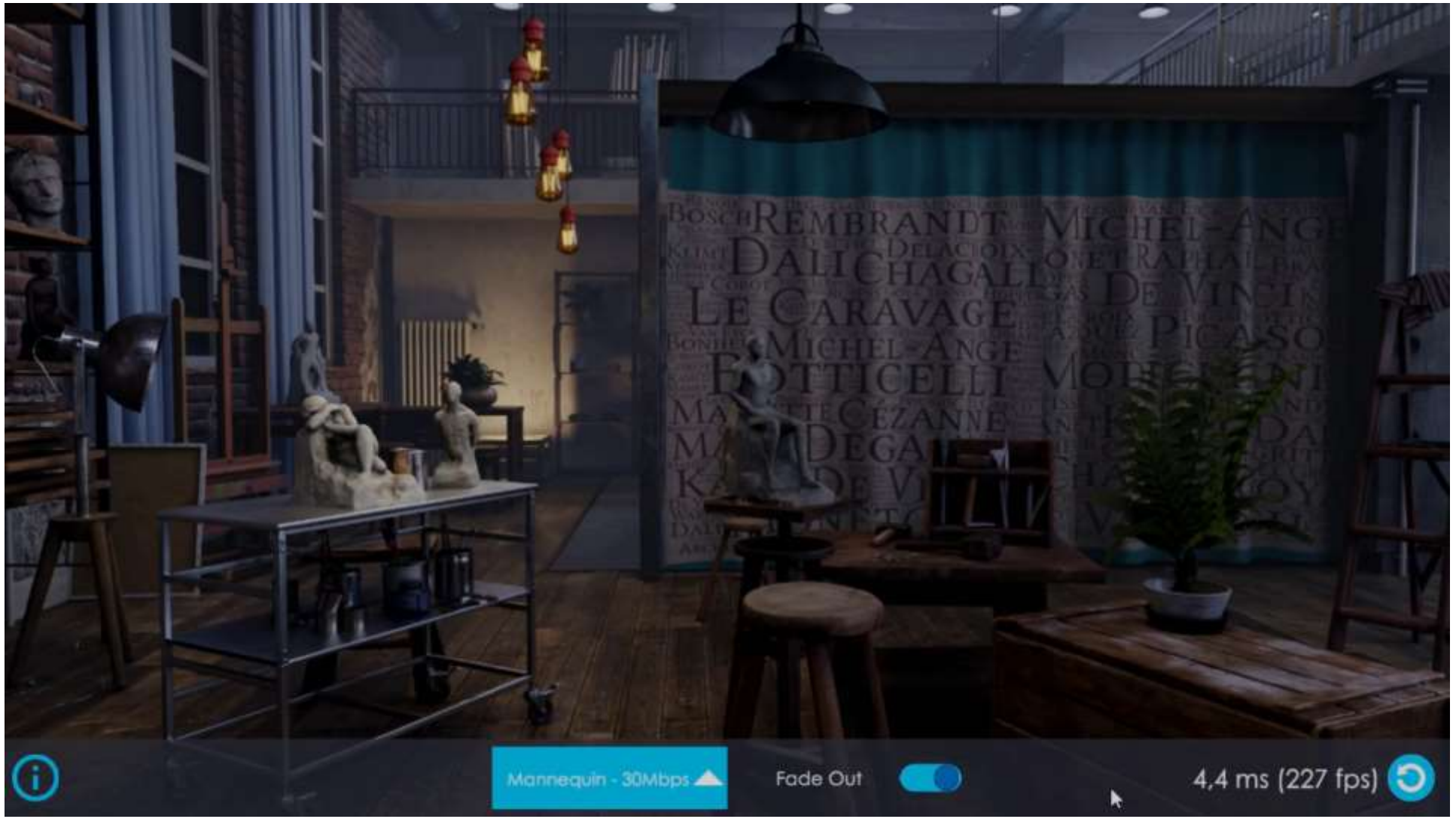
Demo break

Freeport player demo

FROG SEQUENCE

Full video on: https://youtu.be/UeT_Xm1jBGs

Multi-Plane Images for Free Viewpoint



Full video on: https://youtu.be/DPMMzj2I6Yw

Real-time Reference View Synthesis (RVS)

ULB

Interactive demo on: https://lisaserver.ulb.ac.be/rvs/ Source code on: https://gitlab.com/mpeg-i-visual/rvs

MIV tutorial

- MIV website: https://mpeg-miv.org
- Public software:
  - Test model: https://gitlab.com/mpeg-i-visual/tmiv
  - IV-PSNR: https://gitlab.com/mpeg-i-visual/ivpsnr
  - IVDE: https://gitlab.com/mpeg-i-visual/ivde
  - RVS: https://gitlab.com/mpeg-i-visual/rvs
- Demo videos:
  - Freeport player: https://youtu.be/UeT_Xm1jBGs
  - Multi-plane images: https://youtu.be/DPMMzj2I6Yw
  - Real-time RVS: https://lisaserver.ulb.ac.be/rvs/
- Test material (14+ sequences) available on request

The MPEG Immersive video coding standard

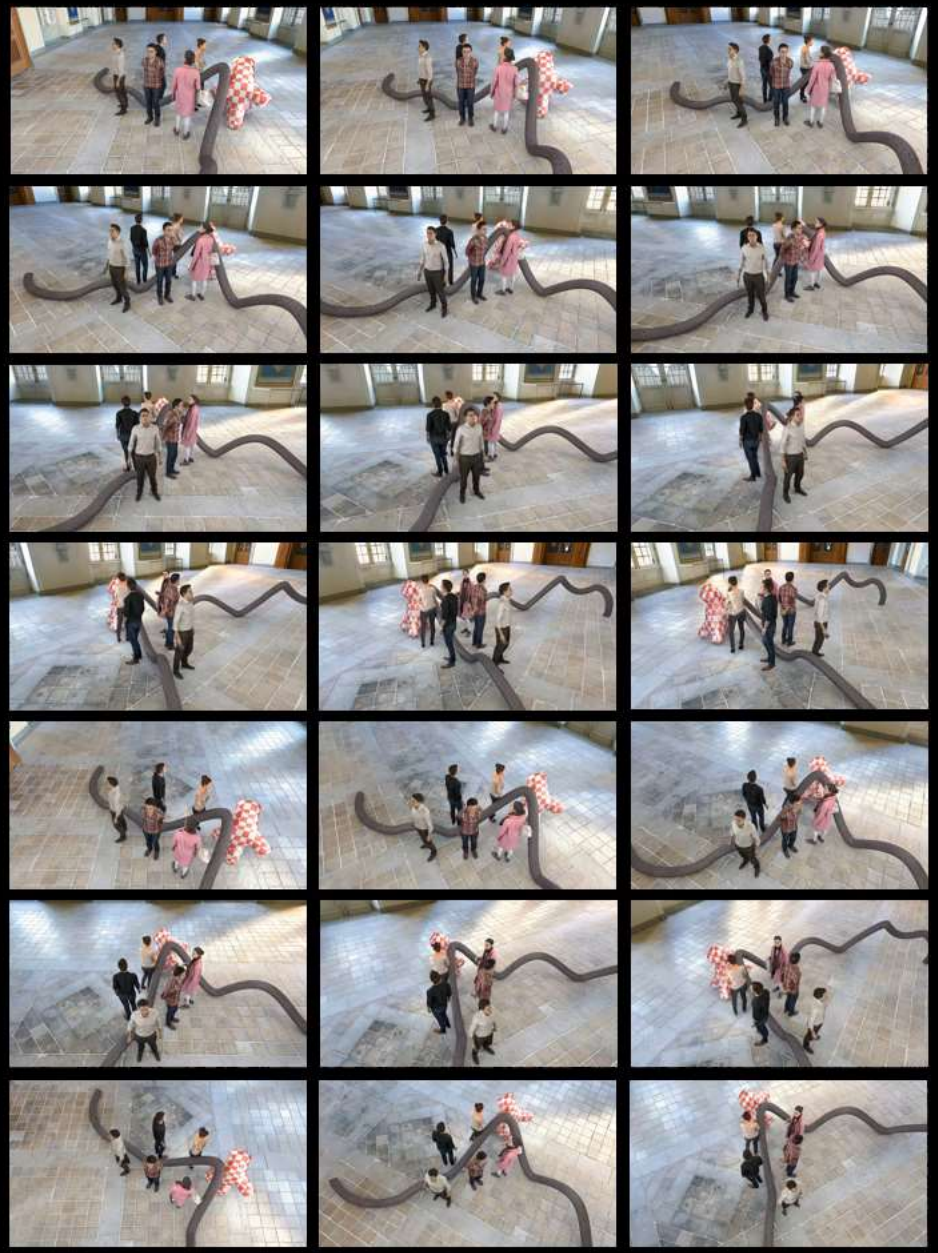
Part II – Source video acquisition and encoder overview

Dawid Mieloch, PhD

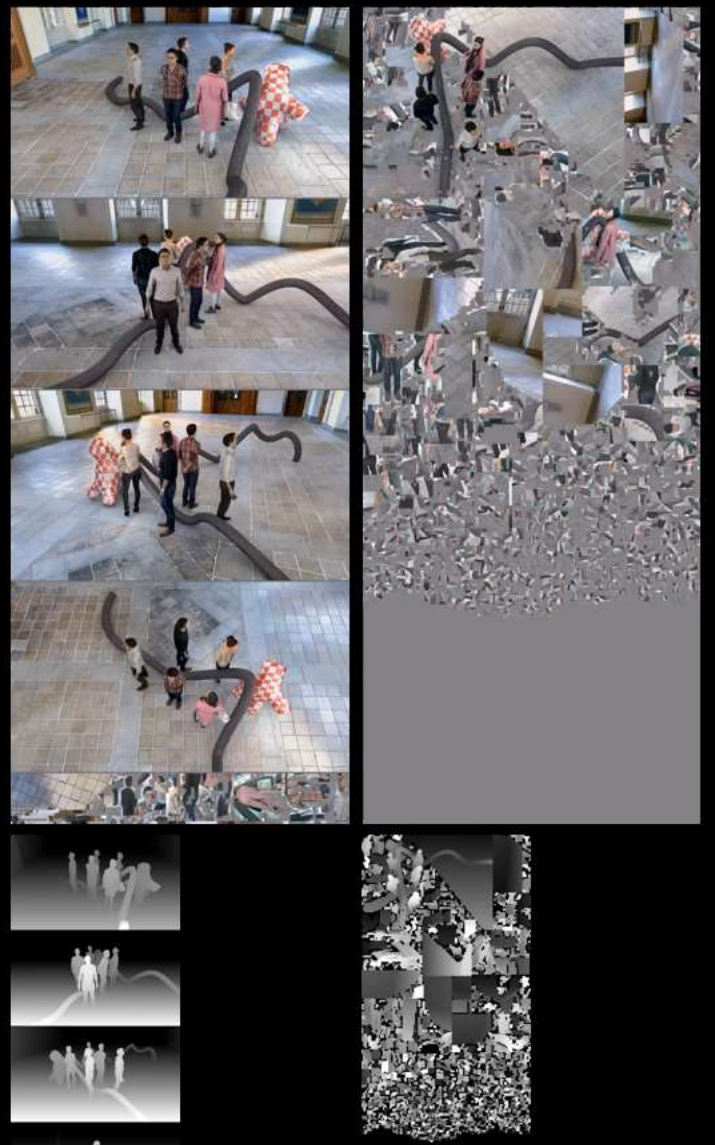
Institute of Multimedia Telecommunications
Poznań University of Technology

itm

input views
attributes + geometry + cam. params

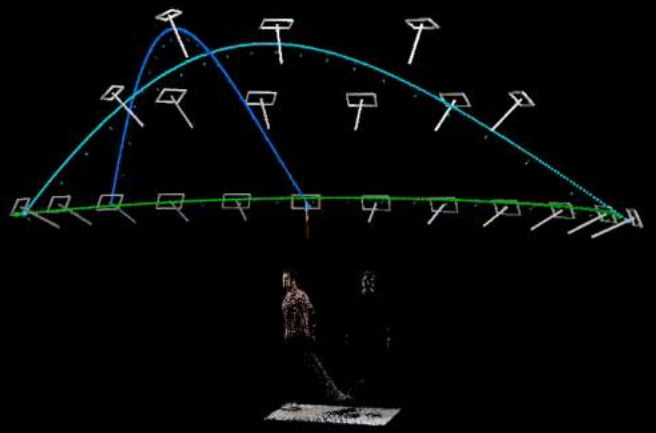


atlases
attributes + geometry + metadata sub-bitstream

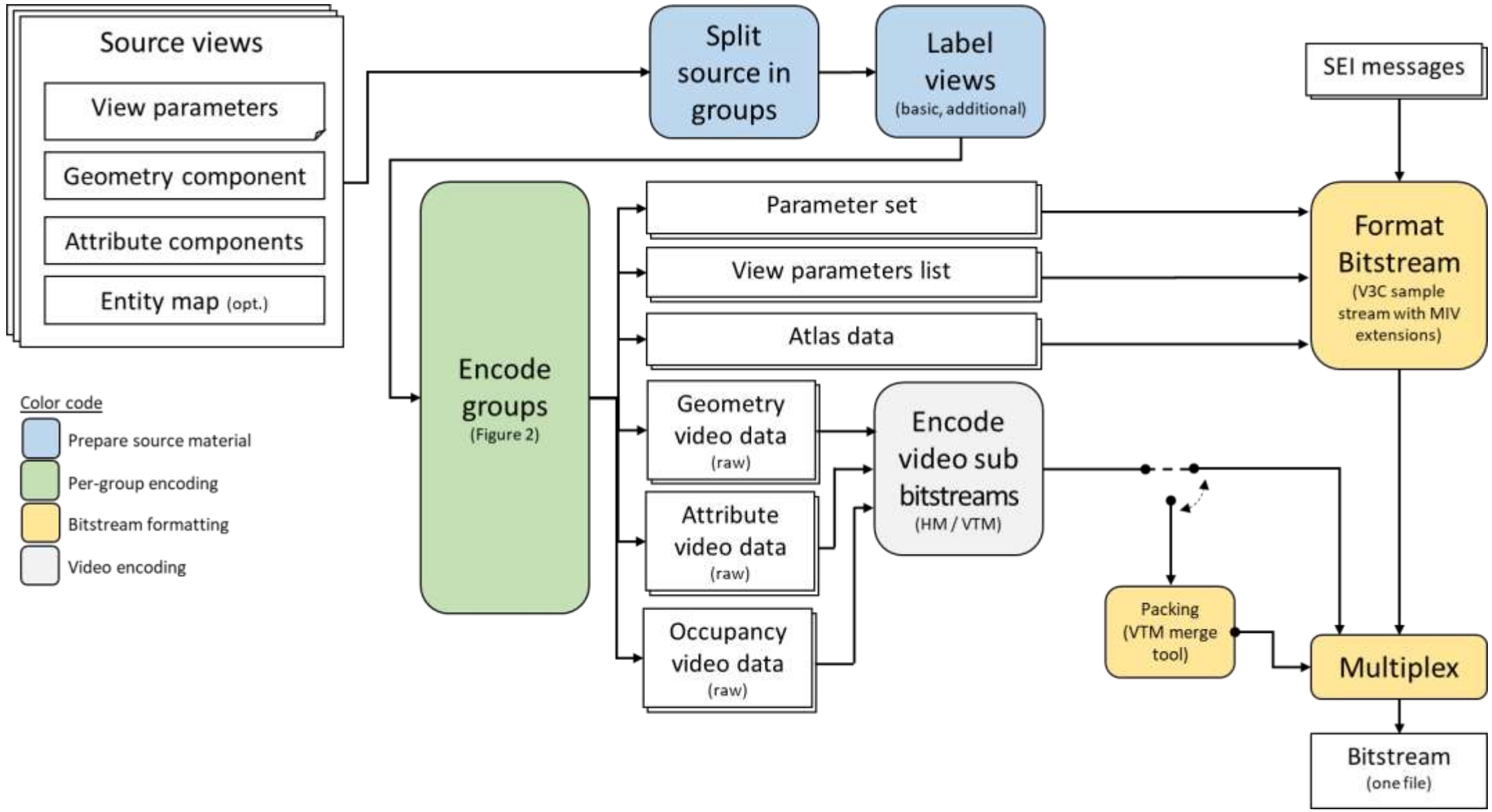


viewport

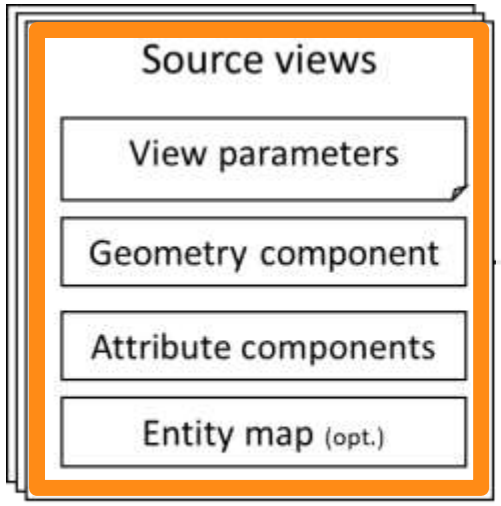
trajectory of the viewer



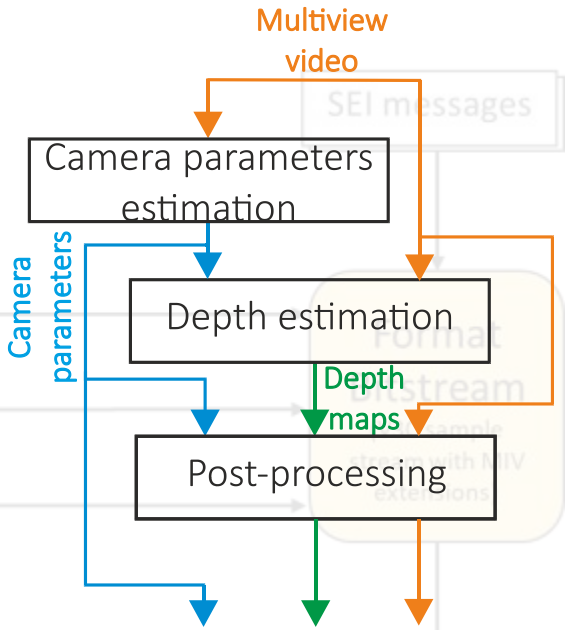
MIV encoder


Source views
View parameters
Geometry component
Attribute components
Entity map (opt.)
Split source in groups
Label views
(basic, additional)
SEI messages
Encode groups
(Figure 2)
Parameter set
View parameters list
Atlas data
Format Bitstream
(V3C sample stream with MIV extensions)
Geometry video data
(raw)
Attribute video data
(raw)
Occupancy video data
(raw)
Encode video sub bitstreams
(HM / VTM)
Packing
(VTM merge tool)
Multiplex
Bitstream
(one file)
Color code
Prepare source material
Per-group encoding
Bitstream formatting
Video encoding

MIV encoder

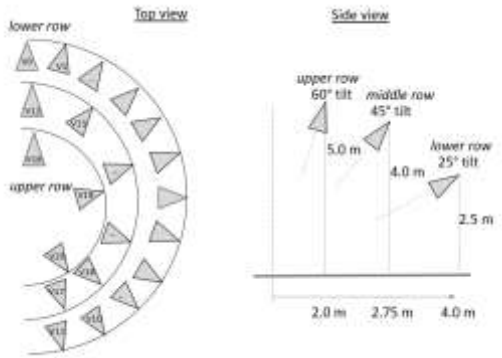


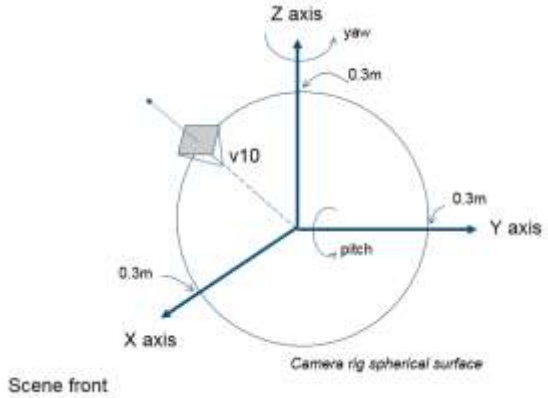
- Acquisition of multiview sequence
- Estimation of view/camera parameters
- Estimation of geometry of acquired scene
- Post-processing of multiview sequence
- Validation of the sequence quality



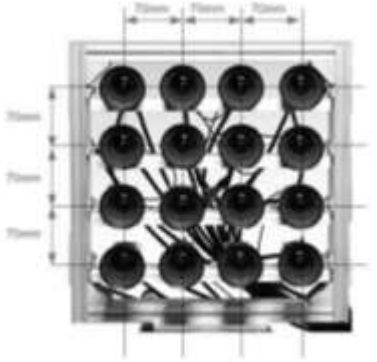
Multiview sequences for immersive video

- Very diverse set of possible camera arrangements
  - Inward, outward, light-field-like, linear
  - No constrains on number of cameras
  - Different camera types





Scene front

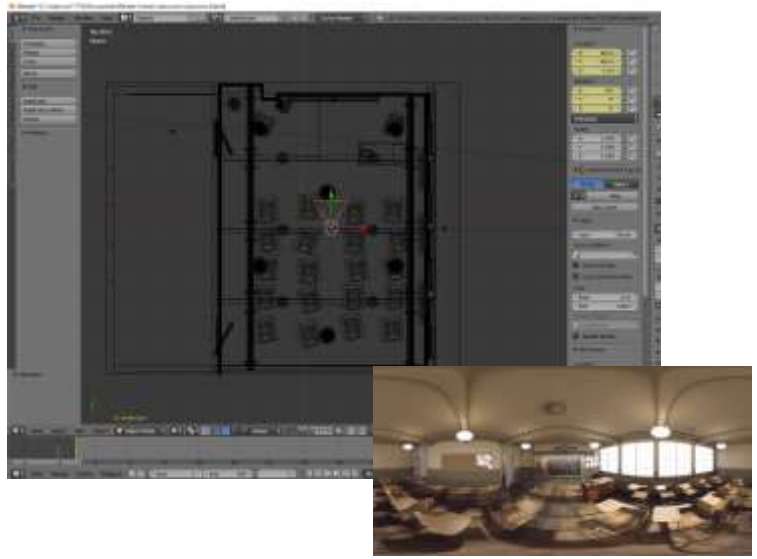


Immersive video acquisition

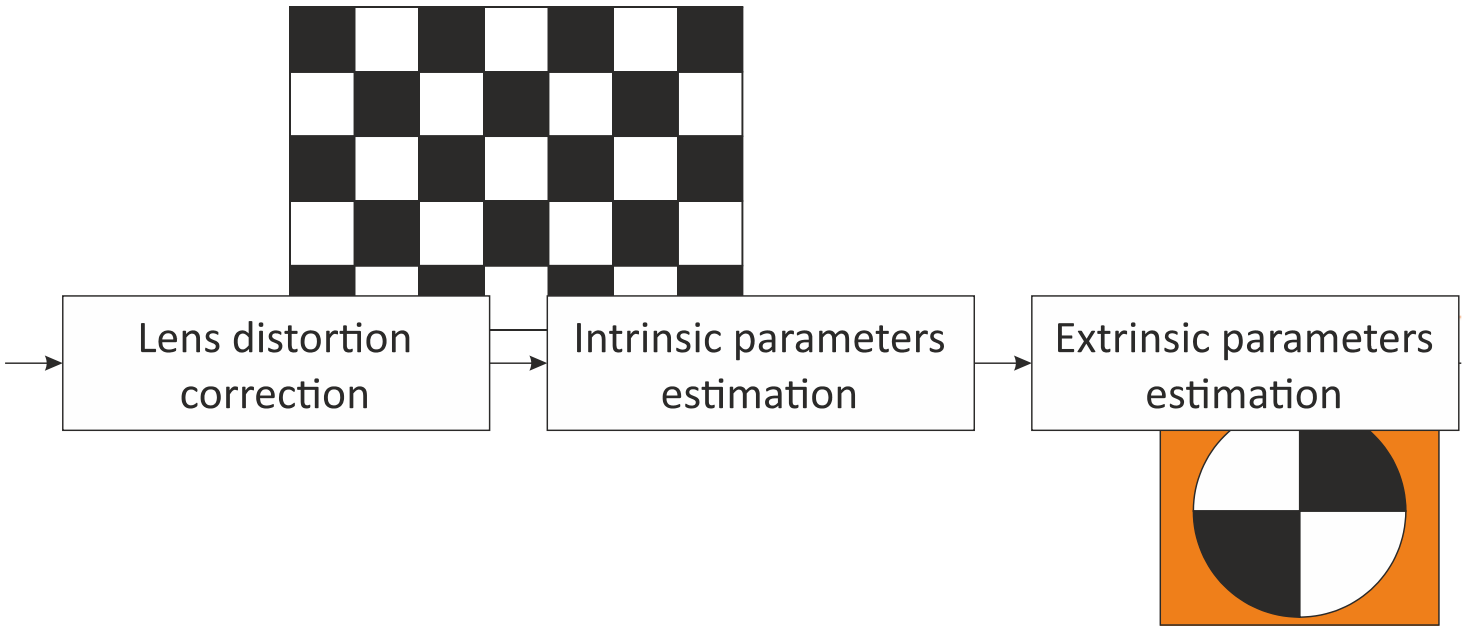
Natural sequences

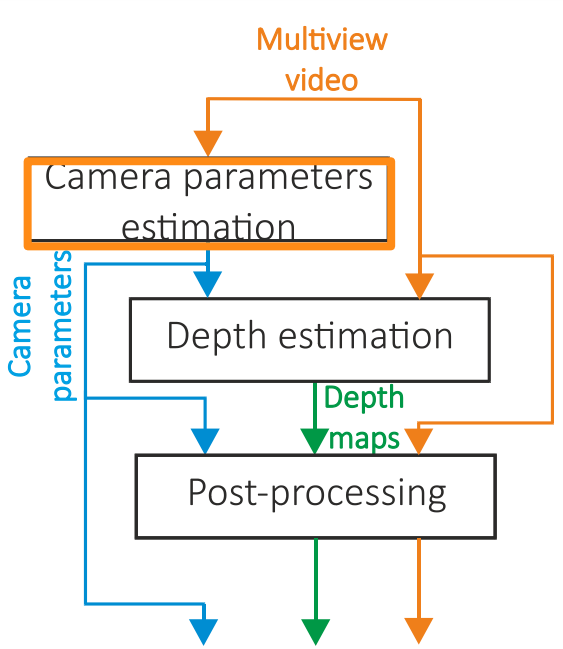
Mixed natural & CGI sequences

CGI sequences

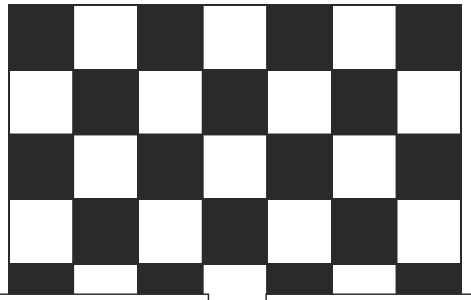


Camera parameters estimation





Camera calibration

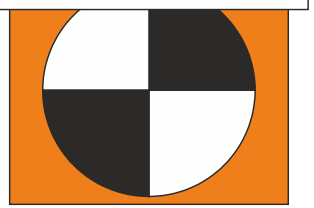


Lens distortion correction → Intrinsic parameters estimation

- Usually performed simultaneously
- Easy to perform using defined pattern, e.g. classical chessboard, pattern of circles
- Typically, 10 diverse views of the pattern are required for each camera
- Available in open source libraries which usually provide sufficient accuracy (e.g. OpenCV)

Camera calibration

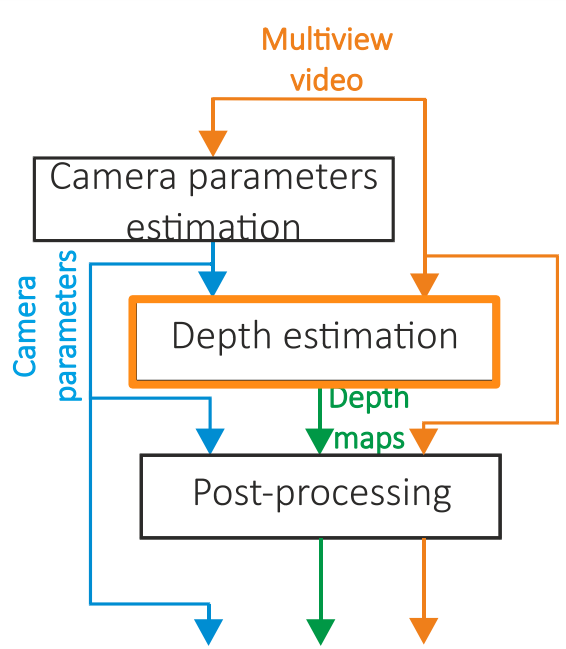
Lens distortion correction → Intrinsic parameters estimation → Extrinsic parameters estimation



- Performed each time after positions of cameras were changed
- Can be done by optimization of re-projection function (minimization of point to epipolar line distance) – bundle adjustment
  - Required: points correspondance, camera intrinsic parameters
- The automatic search for reference points is preferred for practical multicamera systems
- Reference points obtained by automatic feature extraction or from calibration object

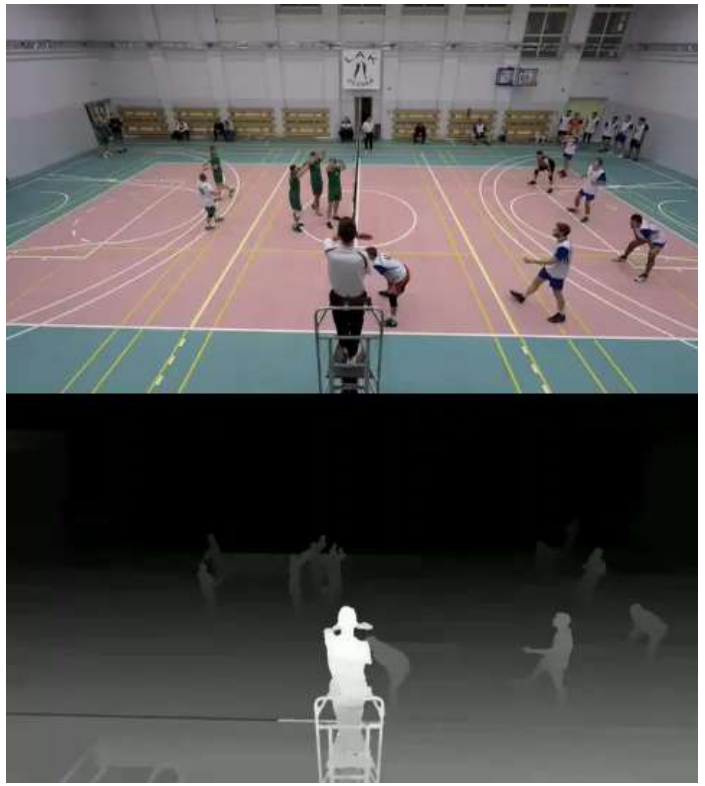
Depth estimation

- Main requirements for the method:
  - High quality of estimated depth maps
    - Particular emphasis on inter-view and temporal consistencies
  - Versatility of estimation process
    - No assumptions about the number and positioning of cameras
    - Can be used for different scenes without any modifications
- Additional requirements for decoder-side depth estimation:
  - Dealing with decoded input views distorted by the compression
  - Automatic calculation of the depth range



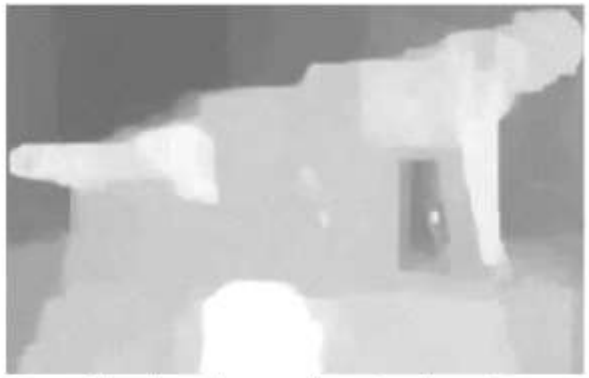
Immersive Video Depth Estimation - IVDE

- Depth estimated for segments for all views simultaneously
- No assumptions about the positioning of views
- Matching method adapted for compressed input views
- Automatic calculation of the depth range
- The support of MIV Geometry Assistance SEI
- Available: https://gitlab.com/mpeg-i-visual/ivde

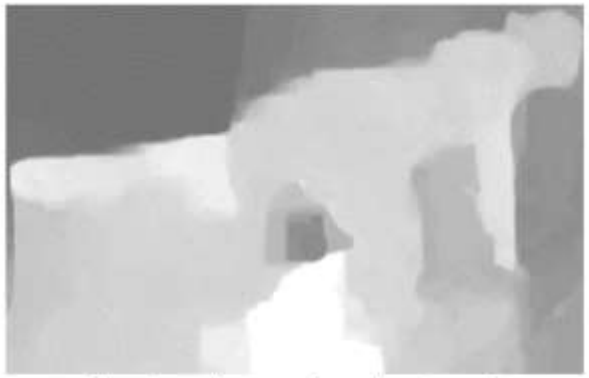


Depth refinement – motivation

- Lack of the inter-view consistency of depth maps strongly deteriorates the quality of generated virtual views



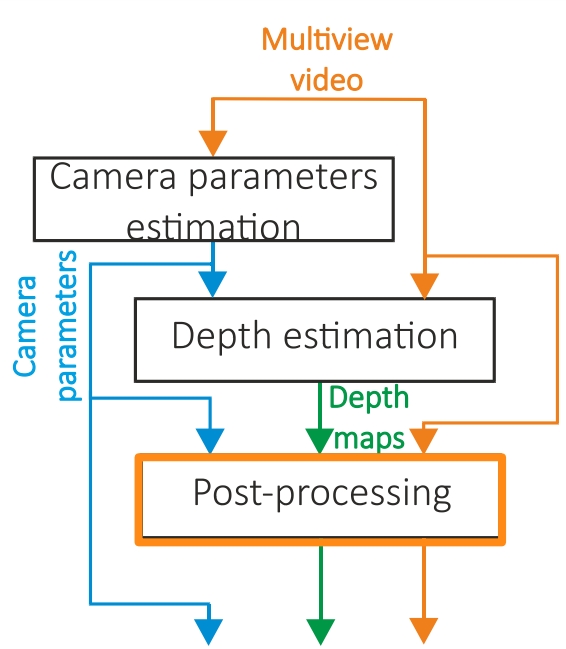
The depth map for the view 0



The depth map for the view 2

The reference view 1

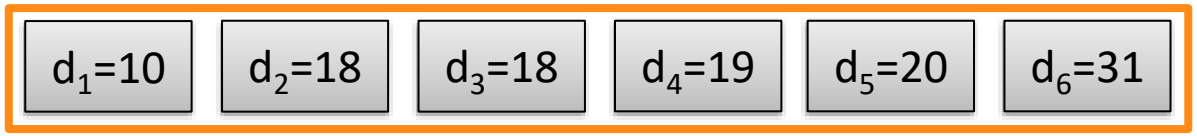
The virtual view in the position of the view 1



Depth refinement – example

- Only information from depth maps is used
  - Texture can introduce errors in the refinement due to inter-view color inconsistencies and noise
- List $\boldsymbol{D}_i(x,y)$ of depth values projected from various input depth maps is created and sorted in ascending order:

$$\boldsymbol{D}_i(x,y) = [d_1(x,y),\, d_2(x,y),\, d_3(x,y), \ldots]$$

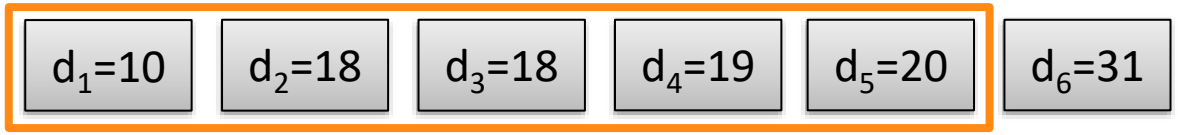


Poznan Depth Refinement – available within MPEG Video Coding and on https://gitlab.com/dmieloch/depth-map-refinement

Depth refinement – example

- Only information from depth maps is used
  - Texture can introduce errors in the refinement due to inter-view color inconsistencies and noise
- List $\boldsymbol{D}_i(x, y)$ of depth values projected from various input depth maps is created and sorted in ascending order:

$$\boldsymbol{D}_i(x, y) = [d_1(x, y),\ d_2(x, y),\ d_3(x, y), \ldots]$$

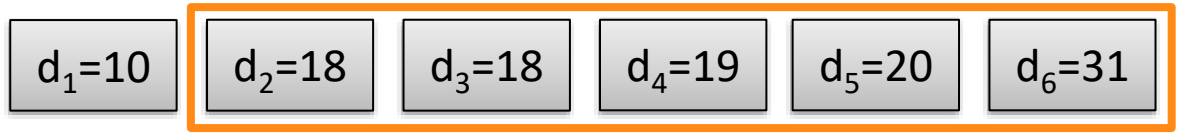


Poznan Depth Refinement – available within MPEG Video Coding and on https://gitlab.com/dmieloch/depth-map-refinement

Depth refinement – example

- Only information from depth maps is used
  - Texture can introduce errors in the refinement due to inter-view color inconsistencies and noise
- List $\boldsymbol{D}_i(x,y)$ of depth values projected from various input depth maps is created and sorted in ascending order:

$$\boldsymbol{D}_i(x,y) = [d_1(x,y),\, d_2(x,y),\, d_3(x,y), \ldots]$$

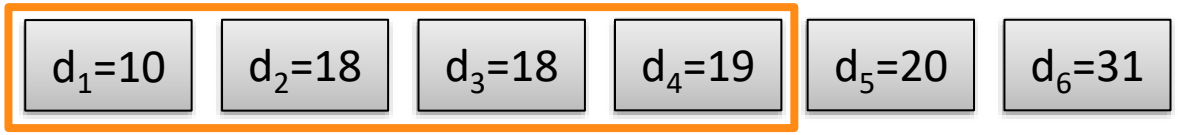


Poznan Depth Refinement – available within MPEG Video Coding and on https://gitlab.com/dmieloch/depth-map-refinement

Depth refinement – example

- Only information from depth maps is used
  - Texture can introduce errors in the refinement due to inter-view color inconsistencies and noise
- List $\boldsymbol{D}_i(x, y)$ of depth values projected from various input depth maps is created and sorted in ascending order:

$$\boldsymbol{D}_i(x, y) = [d_1(x, y),\ d_2(x, y),\ d_3(x, y), \ldots]$$

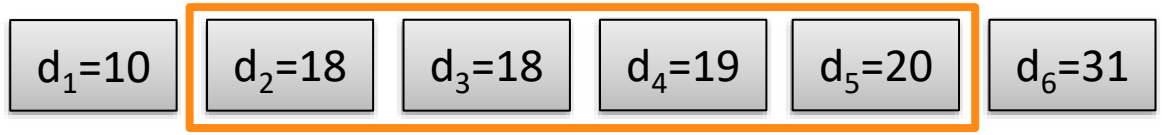


Poznan Depth Refinement – available within MPEG Video Coding and on https://gitlab.com/dmieloch/depth-map-refinement

Depth refinement – example

- Only information from depth maps is used
  - Texture can introduce errors in the refinement due to inter-view color inconsistencies and noise
- List $\boldsymbol{D}_i(x, y)$ of depth values projected from various input depth maps is created and sorted in ascending order:

$$\boldsymbol{D}_i(x, y) = [d_1(x, y),\ d_2(x, y),\ d_3(x, y), \ldots]$$

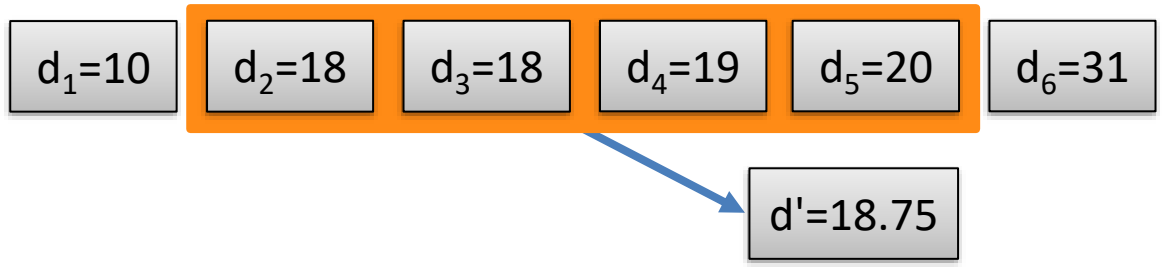


Poznan Depth Refinement – available within MPEG Video Coding and on https://gitlab.com/dmieloch/depth-map-refinement

Depth refinement – example

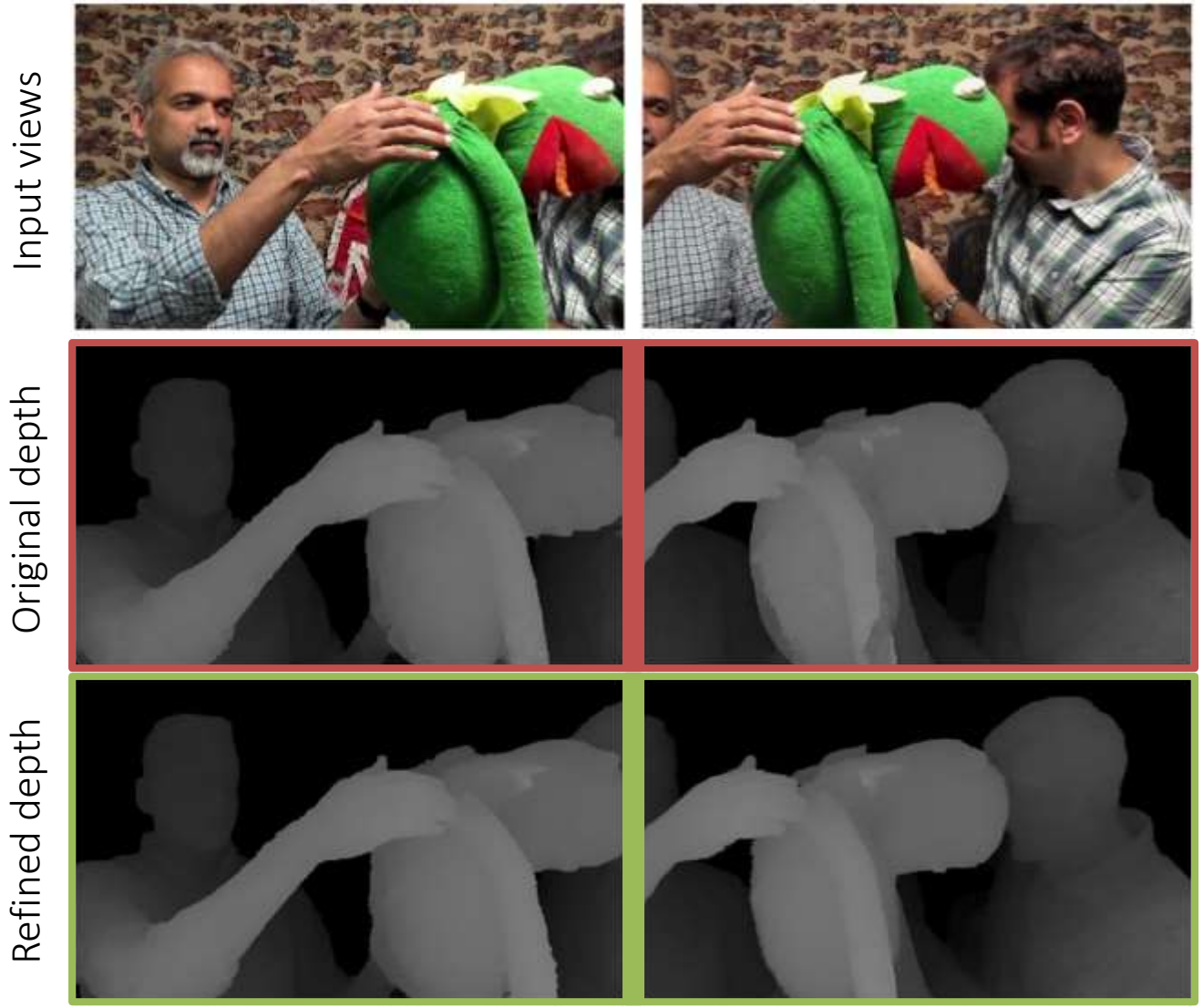
- Only information from depth maps is used
  - Texture can introduce errors in the refinement due to inter-view color inconsistencies and noise
- List $\boldsymbol{D}_i(x,y)$ of depth values projected from various input depth maps is created and sorted in ascending order:

$$\boldsymbol{D}_i(x,y) = [d_1(x,y),\, d_2(x,y),\, d_3(x,y), \ldots]$$



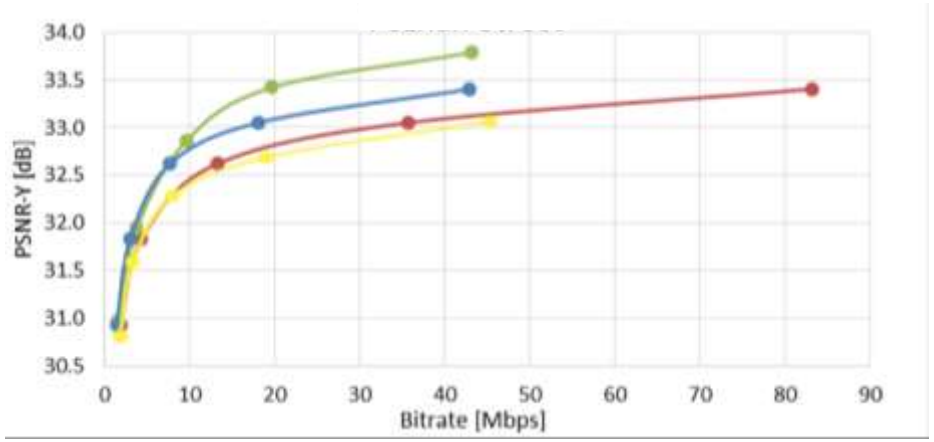
Poznan Depth Refinement – available within MPEG Video Coding and on https://gitlab.com/dmieloch/depth-map-refinement

Depth refinement



PSNR-Y RD-curves for encoding with:

- original depth maps (red curve)
- depth refined with the proposal (green)
- depth refined with synthesis-based refinement (blue)
- depth refined with bilateral filter (yellow)

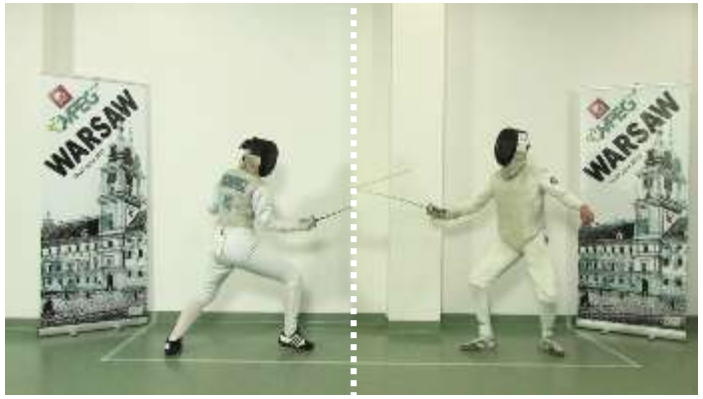


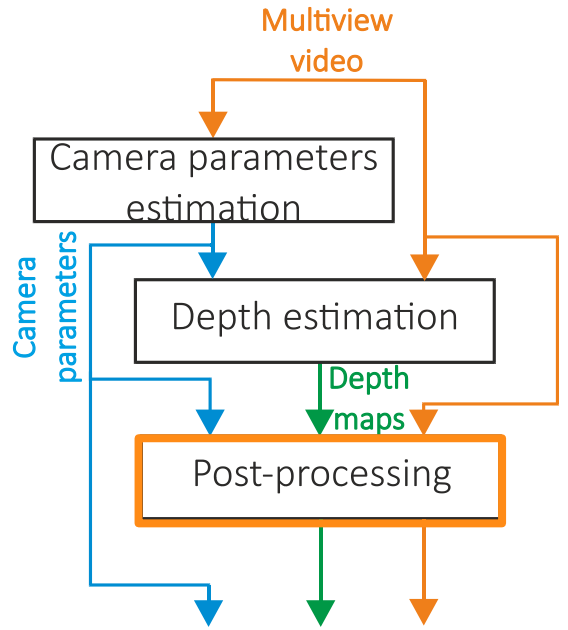
Poznan Depth Refinement – available within MPEG Video Coding and on https://gitlab.com/dmieloch/depth-map-refinement

Color refinement – motivation

- Lack of the temporal consistency (e.g. automatic white balance)

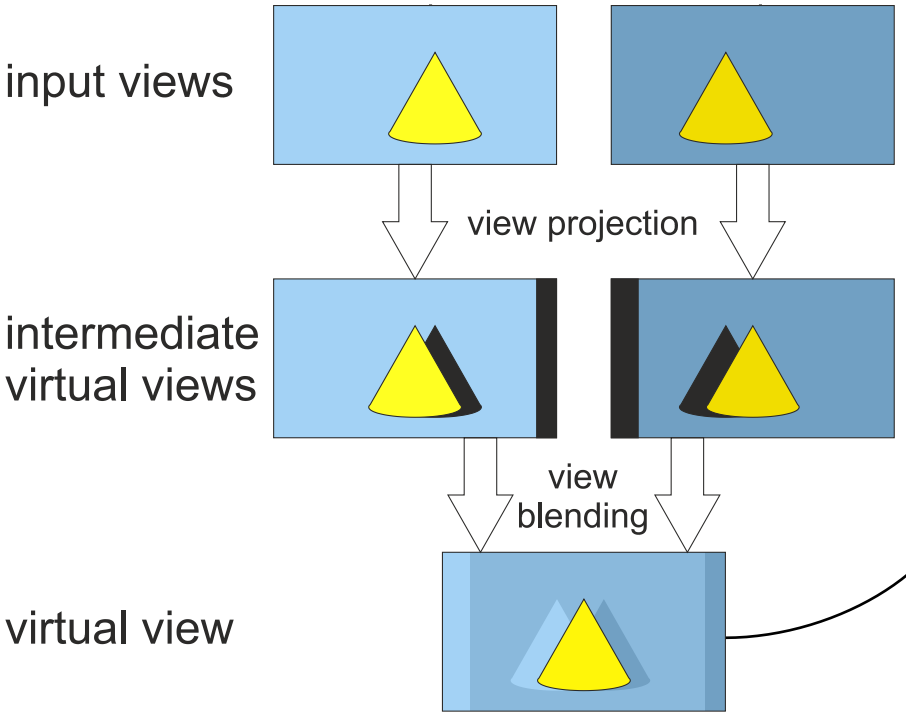
Comparison of the first frame(left side of pictures) and the last frame (right side)





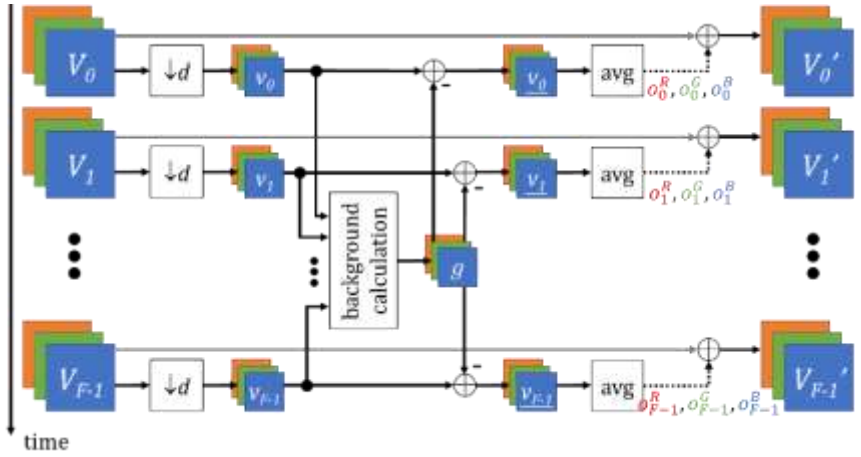
Color refinement – motivation

- Lack of inter-view consistency

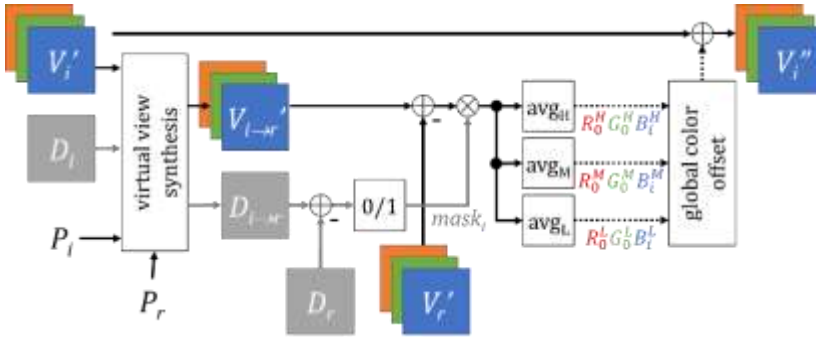


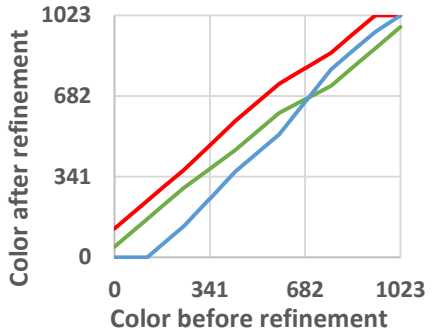
Color refinement – example

1. **Calculate the temporal global offset** between each frame and the time-invariant background frame g:



2. **Calculate the inter-view color mapping** matching the characteristics of central view:





Poznan Color Refinement – available within MPEG Video Coding and on https://gitlab.com/adziembowski/poznancolorrefinement

Color refinement – results

Comparison of the first frame(left side of the picture) and the last frame (right side)

Before color correction

After color correction

Poznan Color Refinement – available within MPEG Video Coding and on https://gitlab.com/adziembowski/poznancolorrefinement

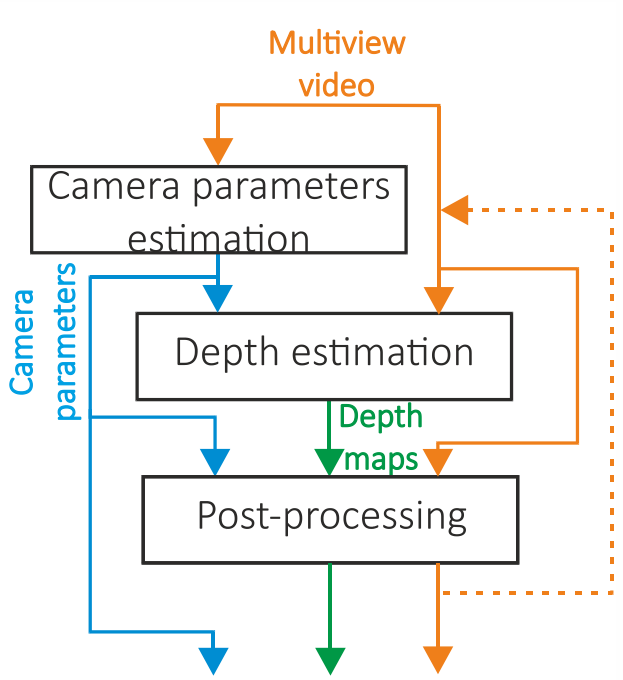
Color refinement – results

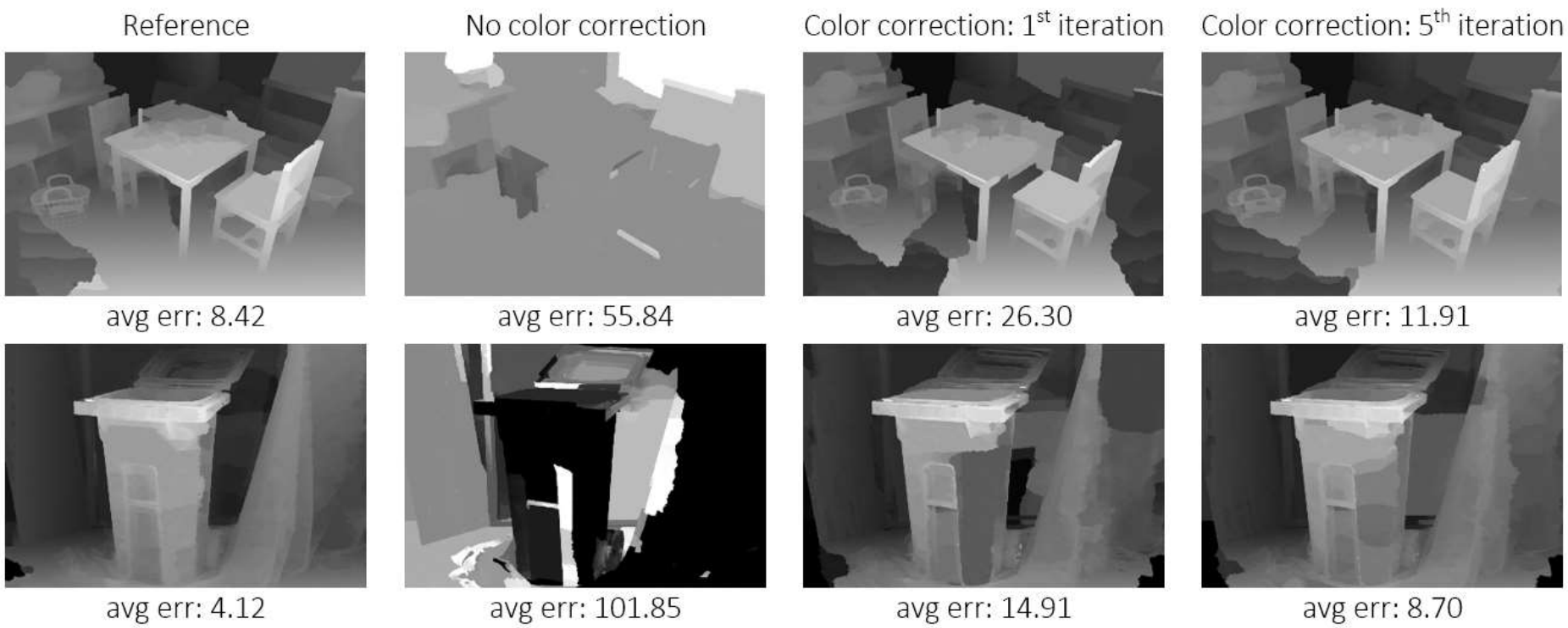
Before color correction

After color correction

Poznan Color Refinement – available within MPEG Video Coding and on https://gitlab.com/adziembowski/poznancolorrefinement

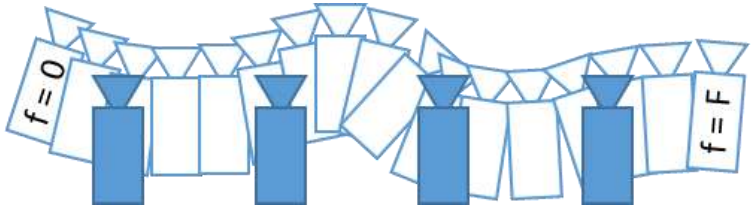
Color refinement – influence on depth estimation





Evaluation of the quality

- Objective quality assessment on the posetrace (white cameras) not possible – no reference

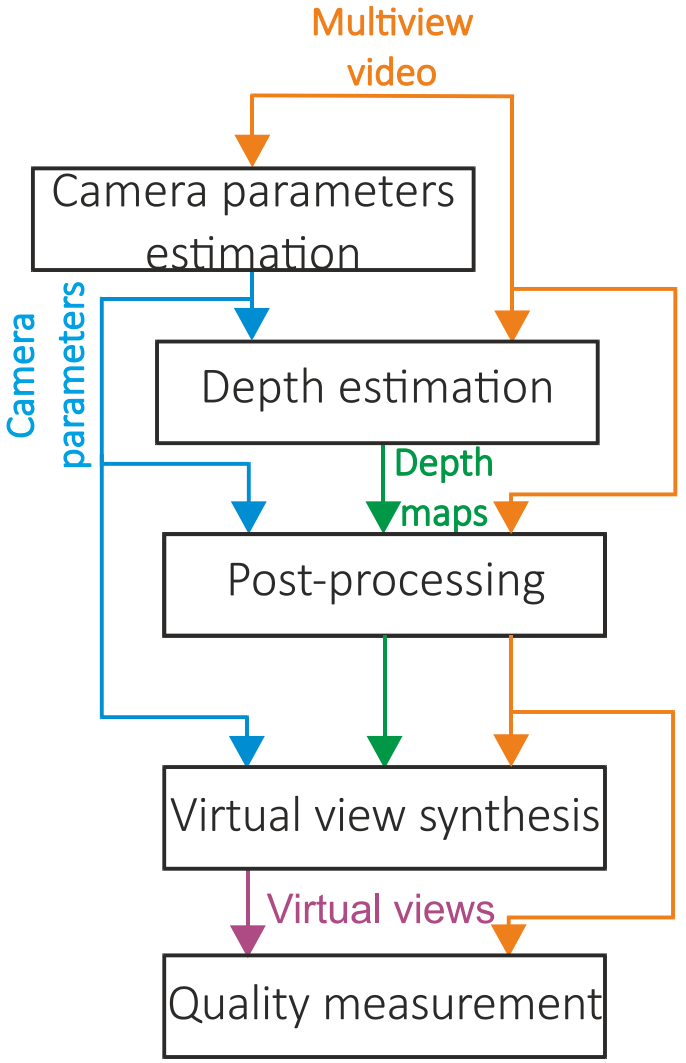


- Evaluation by measurement of quality of views synthesized in the position of input views (blue cameras)

Evaluation of the quality

- Objective quality assessment on the posetrace (white cameras) not possible – no reference

- Evaluation by measurement of quality of views synthesized in the position of input views (blue cameras)



Evaluation of the quality

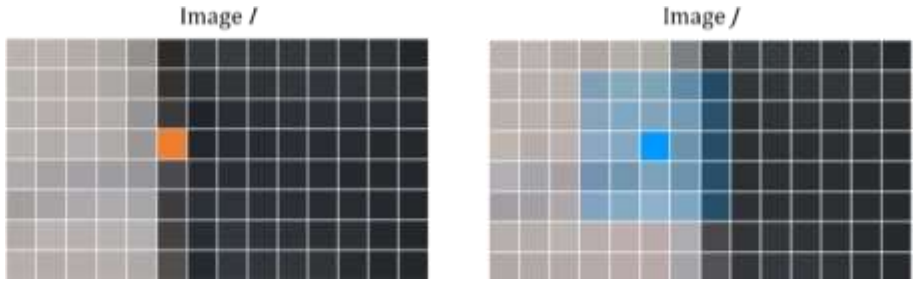
Synthesis performed using e.g. Real-time Reference View Synthesis (RVS)

Interactive demo on: https://lisaserver.ulb.ac.be/rvs/

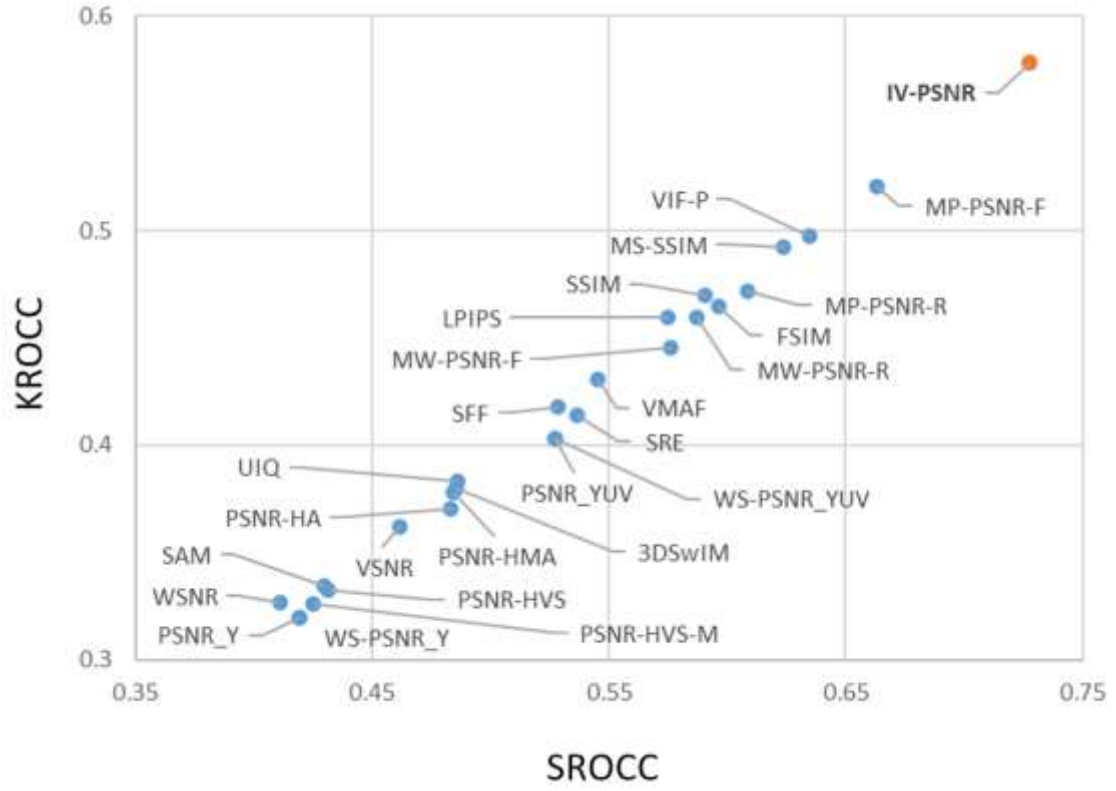
Source code on: https://gitlab.com/mpeg-i-visual/rvs/

Evaluation of the quality

- Quality metric for immersive video – IV-PSNR
  https://gitlab.com/mpeg-i-visual/ivpsnr
- Metric based on PSNR but invariant to small spatial shifts

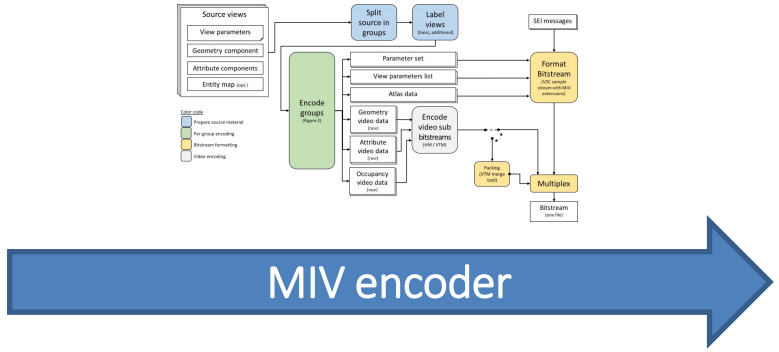


- The effectiveness of the IV-PSNR metric assessed using the results of the “MPEG Call for Proposals on 3DoF+ Visual”

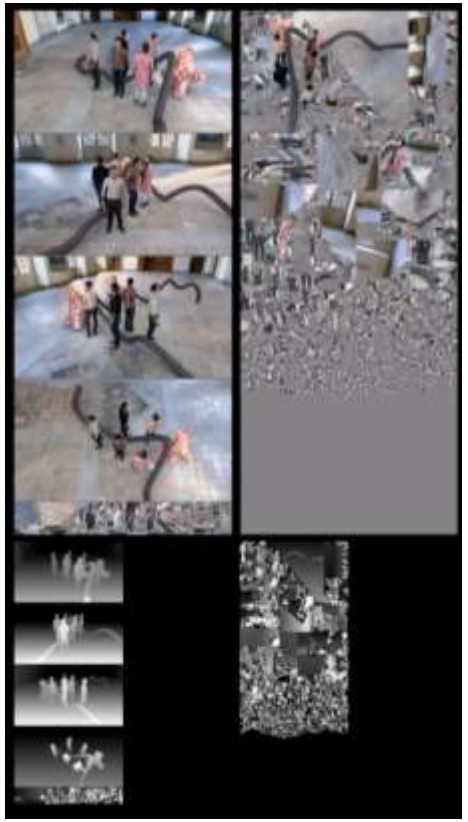


SROCC – Spearman Rank Order Correlation Coefficients
KROCC – Kendall Rank Order Correlation Coefficients

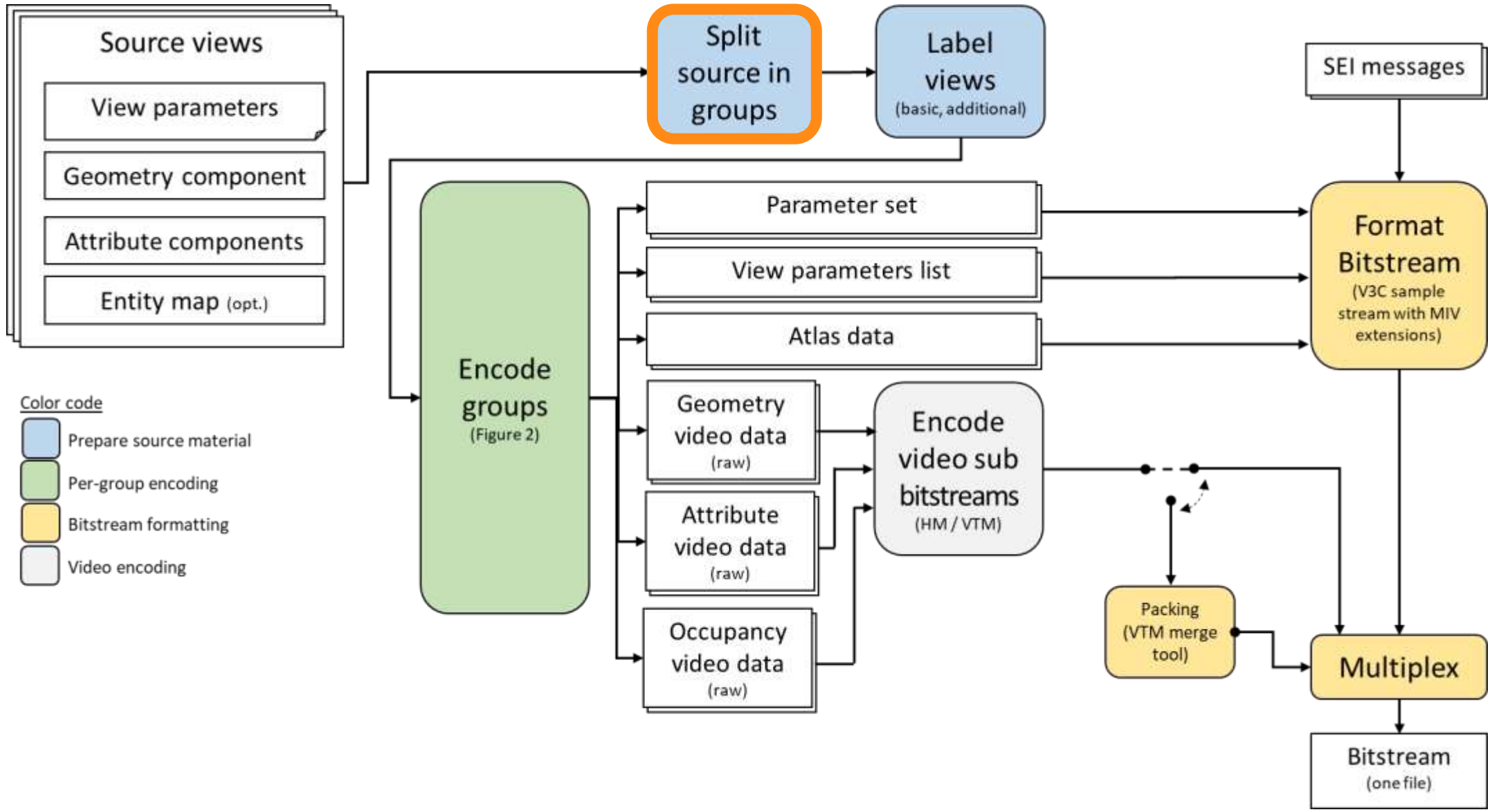
MIV encoder



- All methods and algorithms presented further are just **examples of possible solutions**!
- Encoder is non-normative, shown examples are based on the **Test Model of MPEG immersive video**

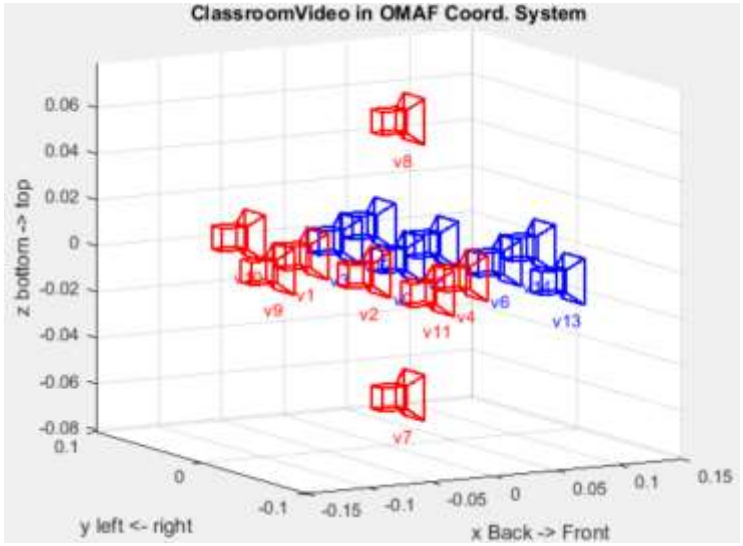


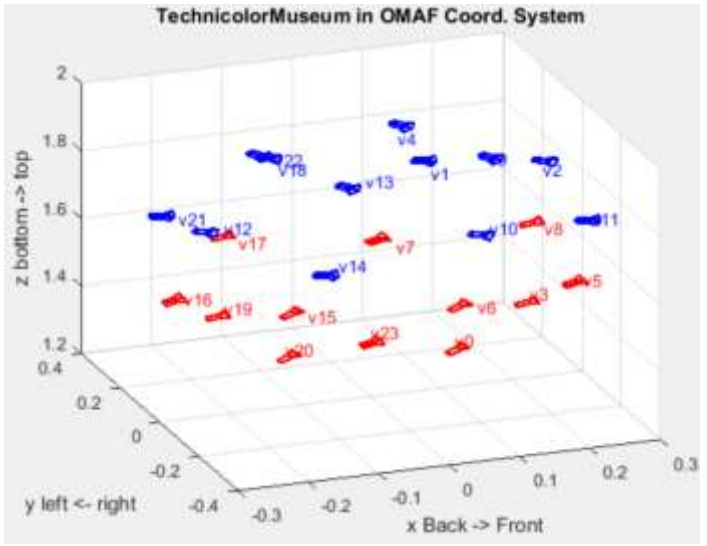
MIV encoder

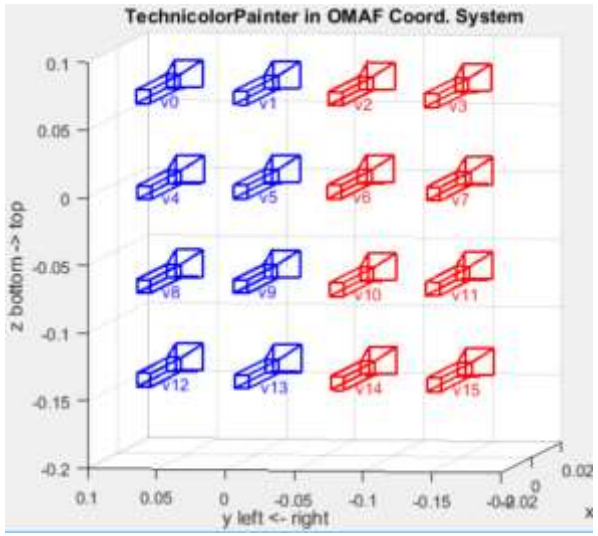

Source views
View parameters
Geometry component
Attribute components
Entity map (opt.)
Split source in groups
Label views
(basic, additional)
SEI messages
Encode groups
(Figure 2)
Parameter set
View parameters list
Atlas data
Format Bitstream
(V3C sample stream with MIV extensions)
Geometry video data
(raw)
Attribute video data
(raw)
Occupancy video data
(raw)
Encode video sub bitstreams
(HM / VTM)
Packing
(VTM merge tool)
Multiplex
Bitstream
(one file)
Color code
Prepare source material
Per-group encoding
Bitstream formatting
Video encoding

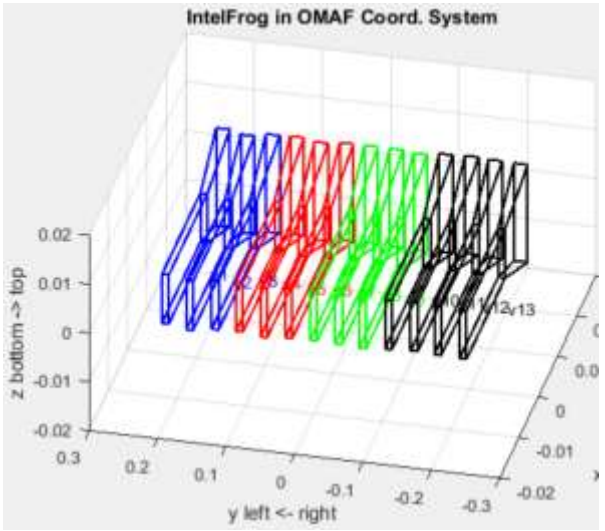
Grouping of source views

Grouping of source views

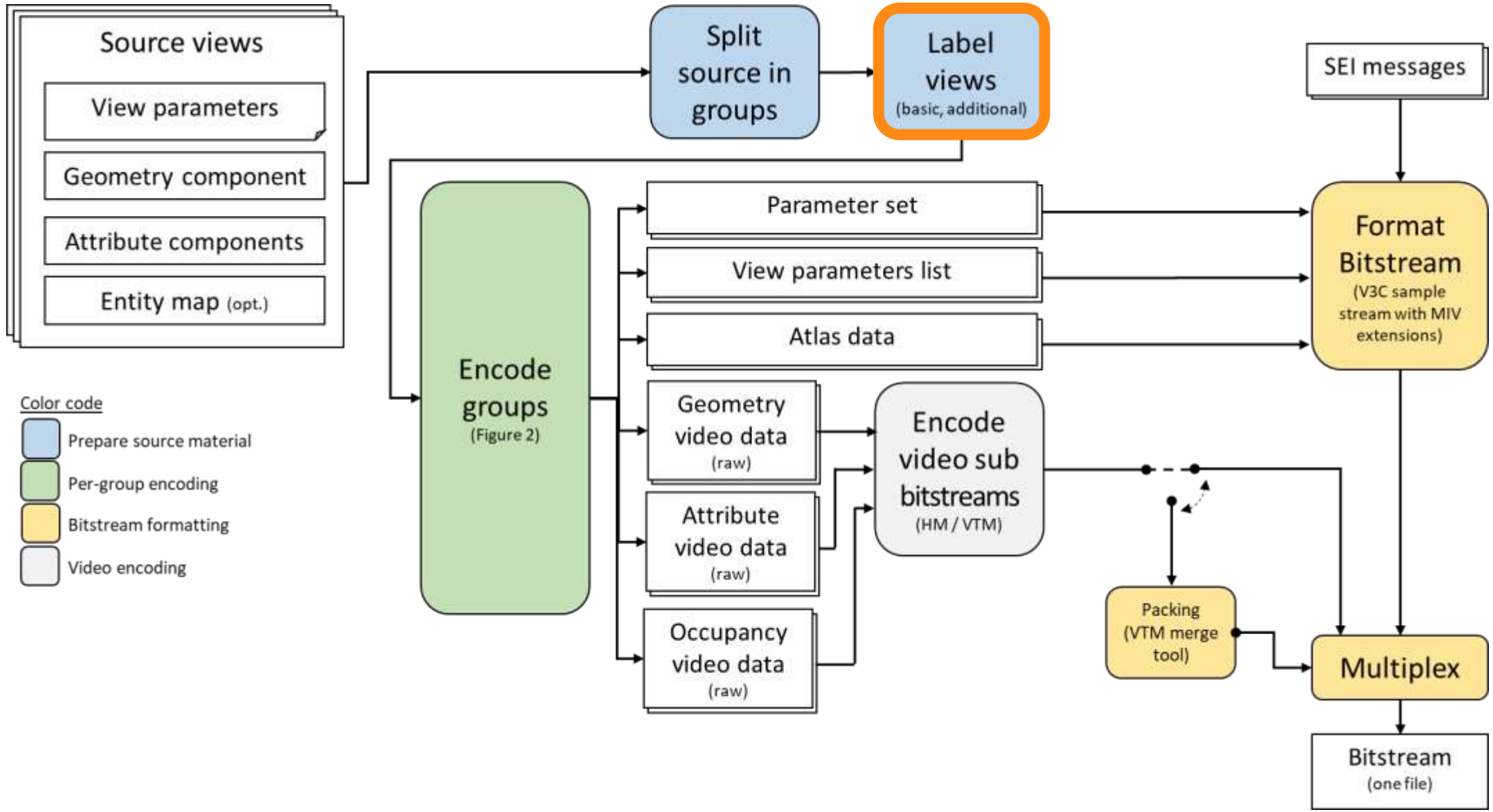








MIV encoder


Source views
View parameters
Geometry component
Attribute components
Entity map (opt.)
Split source in groups
Label views
(basic, additional)
SEI messages
Encode groups
(Figure 2)
Parameter set
View parameters list
Atlas data
Format Bitstream
(V3C sample stream with MIV extensions)
Geometry video data
(raw)
Attribute video data
(raw)
Occupancy video data
(raw)
Encode video sub bitstreams
(HM / VTM)
Packing
(VTM merge tool)
Multiplex
Bitstream
(one file)
Color code
Prepare source material
Per-group encoding
Bitstream formatting
Video encoding

View labeling

Pruning enabled

Without pruning

The following “pixel-rate” constraints imposed on all configurations of TMIV:

- The combined luma sample rate across all decoders does not exceed **1 069 547 520 samples per second** (HEVC Main10 profile level 5.2 used in current widely used decoders, e.g. [1], [2])
- Each coded video picture size does not exceed **8 912 896 pixels** (e.g. 4096 × 2048)
- The number of decoder instances does not exceed **4**

[1] Samsung 2021 TV Video Specifications: https://developer.samsung.com/smarttv/develop/specifications/media-specifications/2021-tv-video-specifications.html
[2] Xilinx H.264/H.265 Video Codec Unit : https://www.xilinx.com/products/intellectual-property/v-vcu.html#overview

View labeling

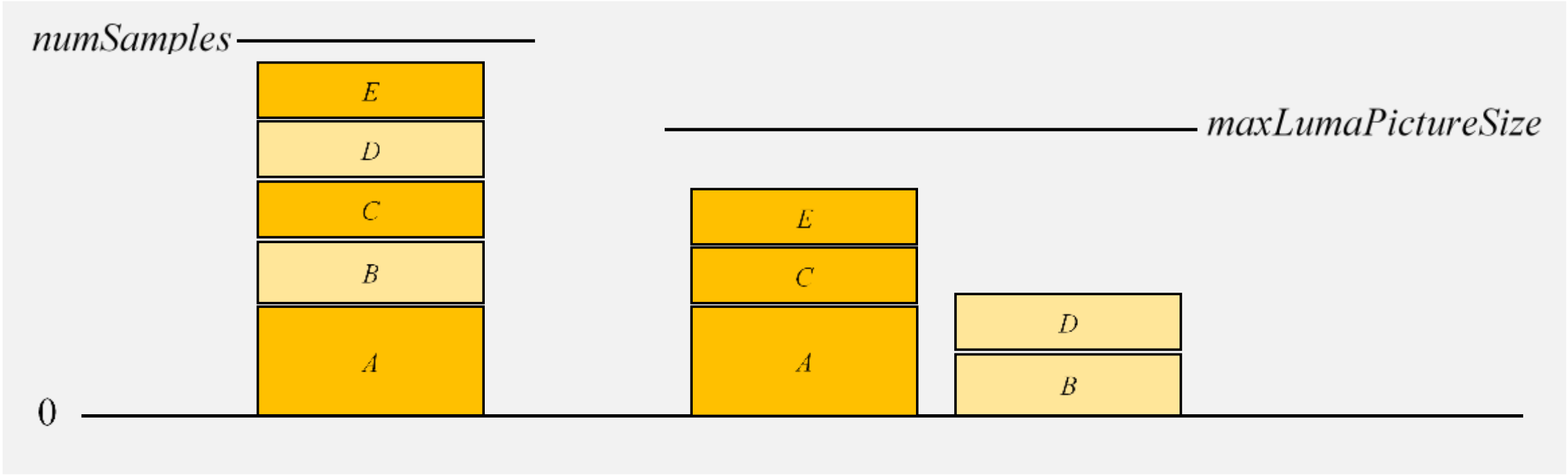
Labeling algorithm:

1. Set the number of basic views
2. Calculate the cost function
3. Choose basic views
4. Update labeling

View labeling

Labeling algorithm:

1. Set the number of basic views
2. Calculate the cost function
3. Choose basic views
4. Update labeling

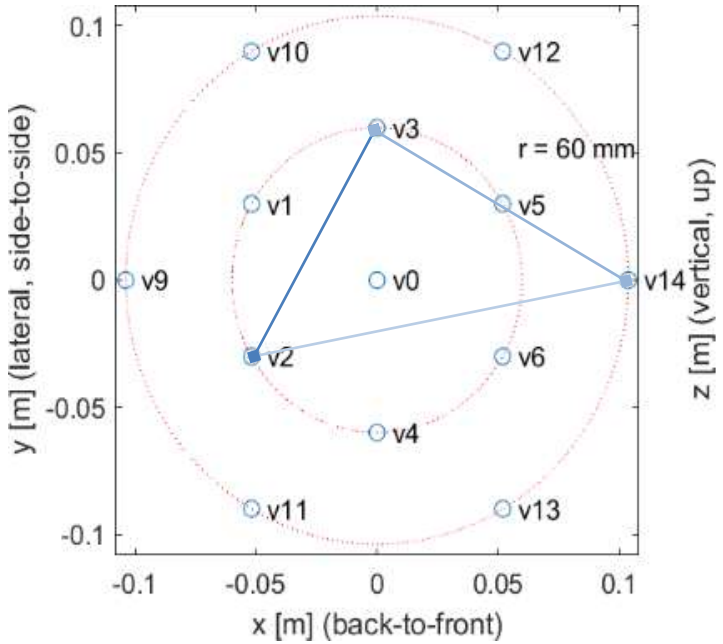


View labeling

Labeling algorithm:

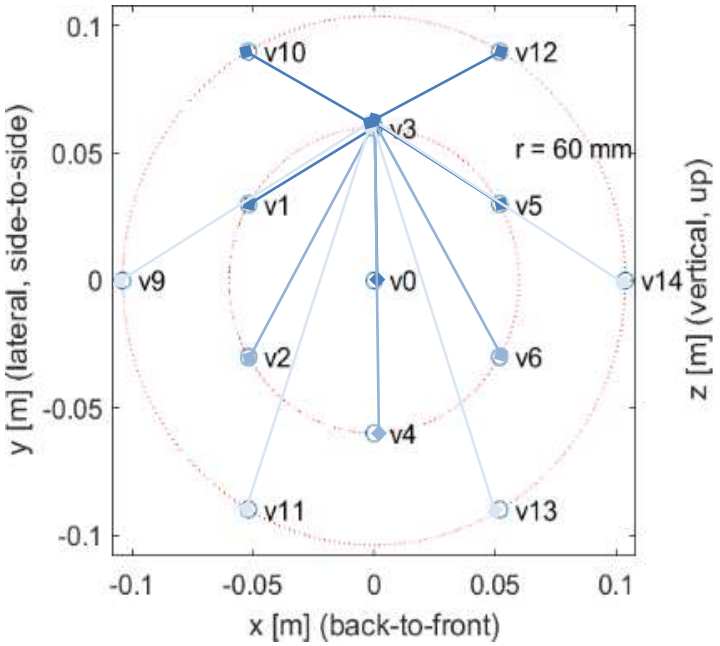
1. Set the number of basic views
2. Calculate the cost function
3. Choose basic views
4. Update labeling

Partitioning Around Medoids (PAM): k-medoids



More than 2 basic views - **repulsion**:

$$J = 2 \sum_{\substack{1 \le i < k \\ i < j \le k}} r_{c_i,c_j}^{-2}$$



Only one basic view - **attraction**:

$$J = - \sum_{1 \le i < c, c < i \le n} r_{c,i}^{-2}$$

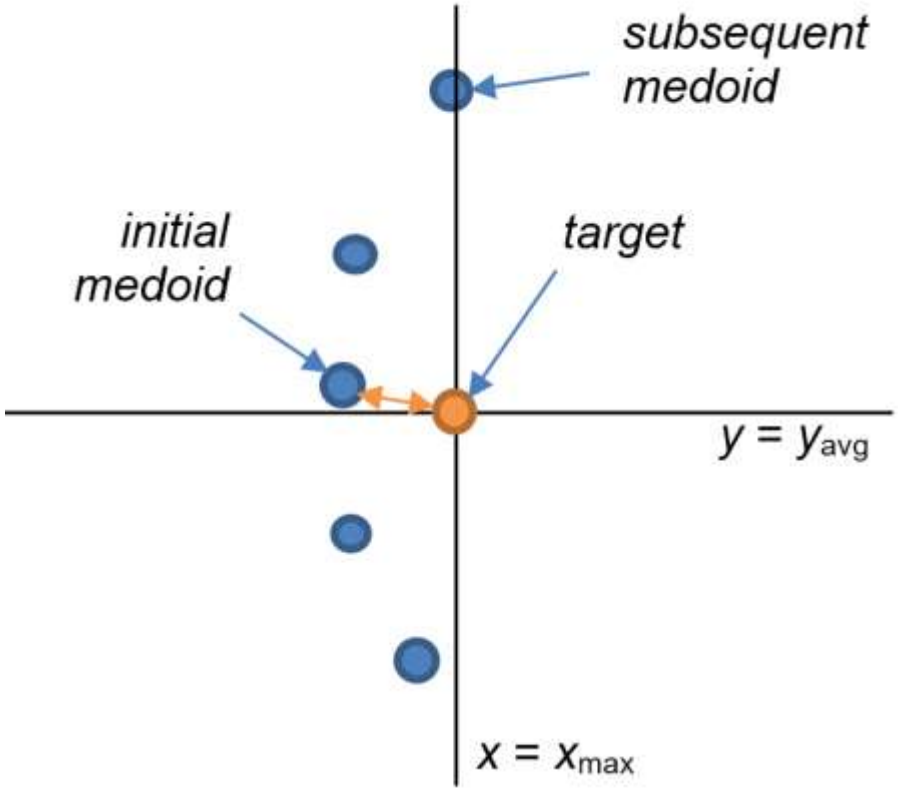
View labeling

Labeling algorithm:

1. Set the number of basic views
2. Calculate the cost function
3. Choose basic views
4. Update labeling

Initial basic view - view closest to the point with:

- x_max (closest to the scene)
- y_avg (center of the system horizontally)
- z_avg (center of the system vertically)



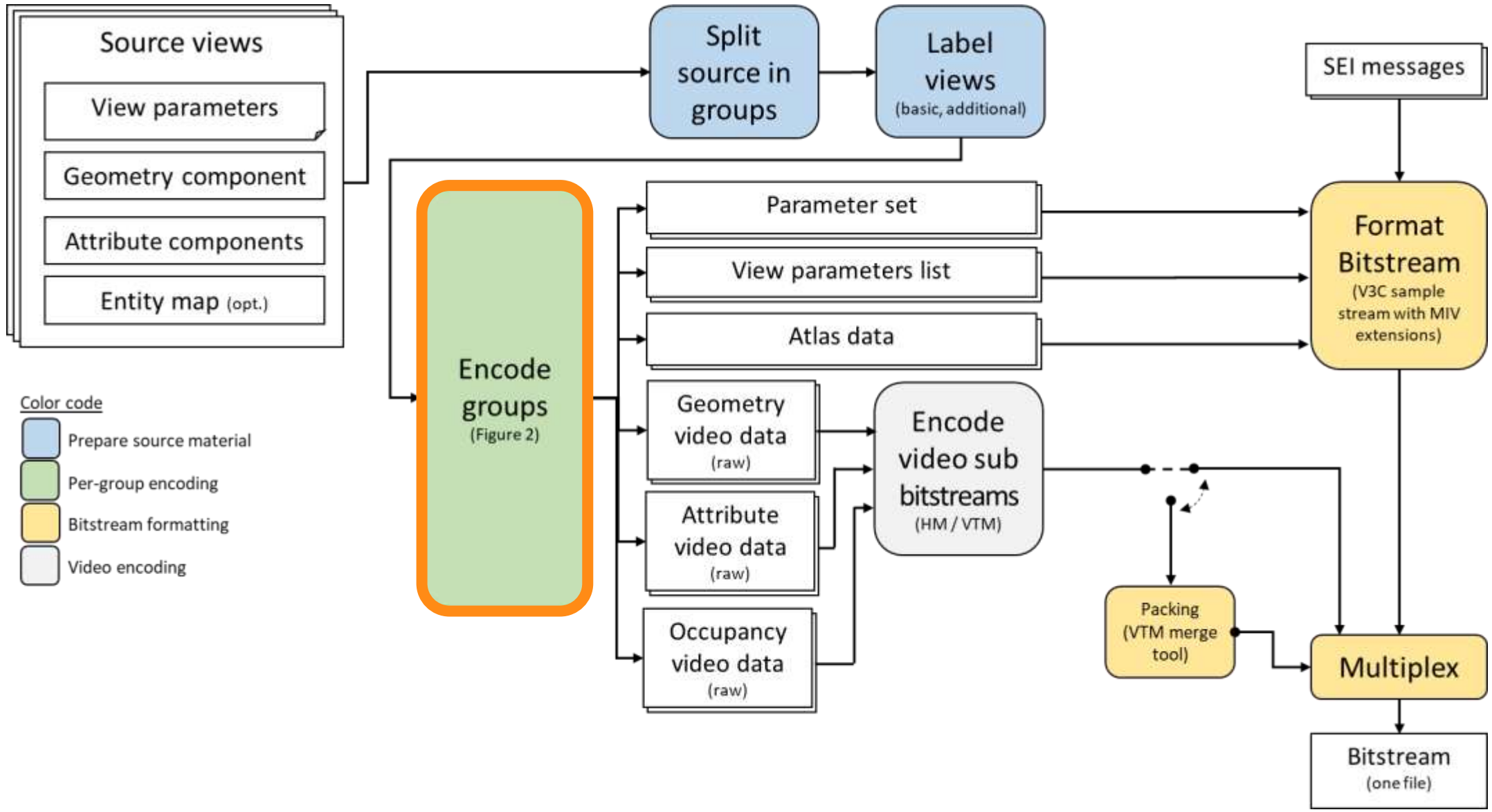
View labeling

Labeling algorithm:

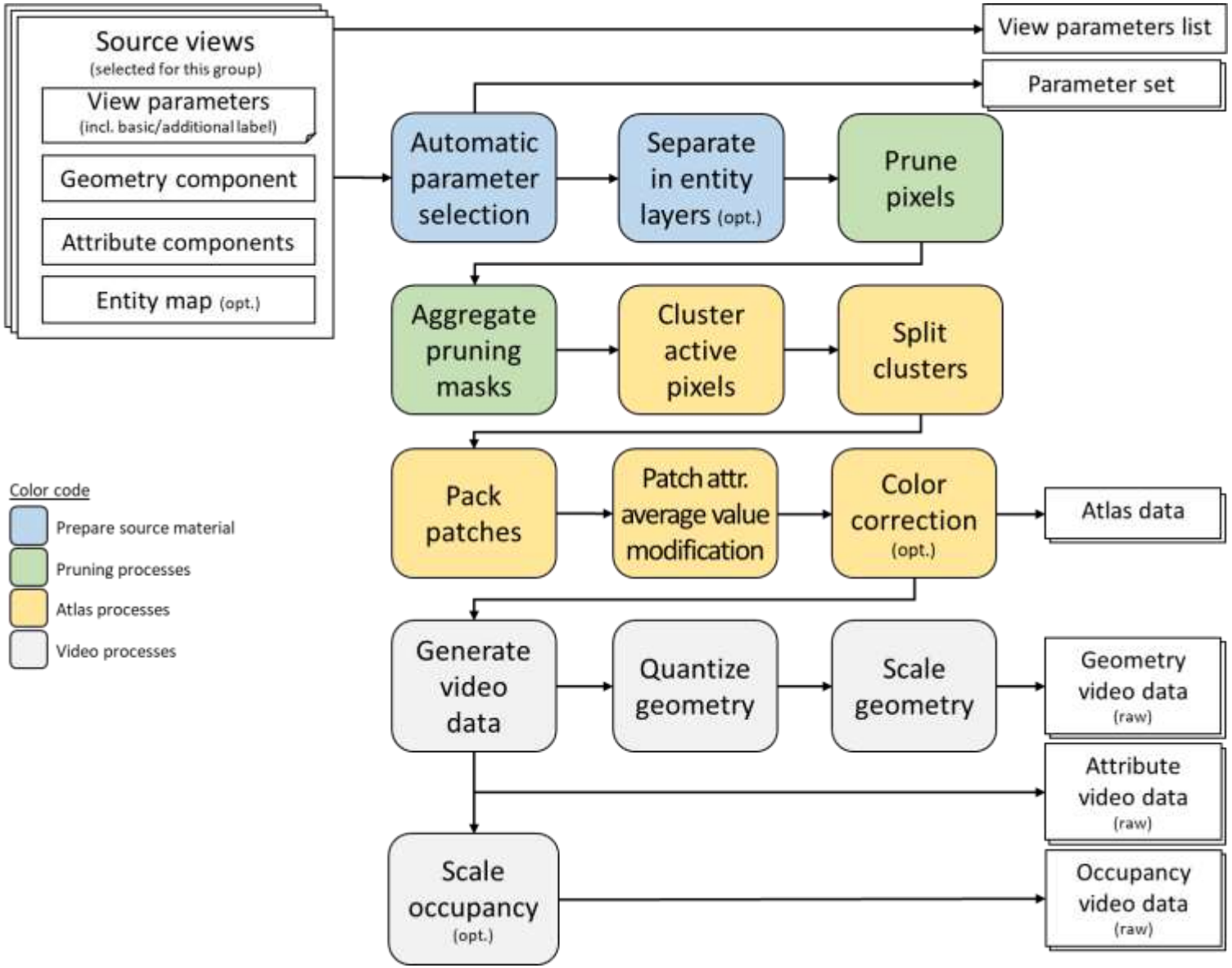
1. Set the number of basic views
2. Calculate the cost function
3. Choose basic views
4. Update labeling

- In each iteration all possible replacements of the base view <-> additional view are considered
- The most cost-reducing change is performed in each iteration
- The algorithm ends when no further cost reduction is possible

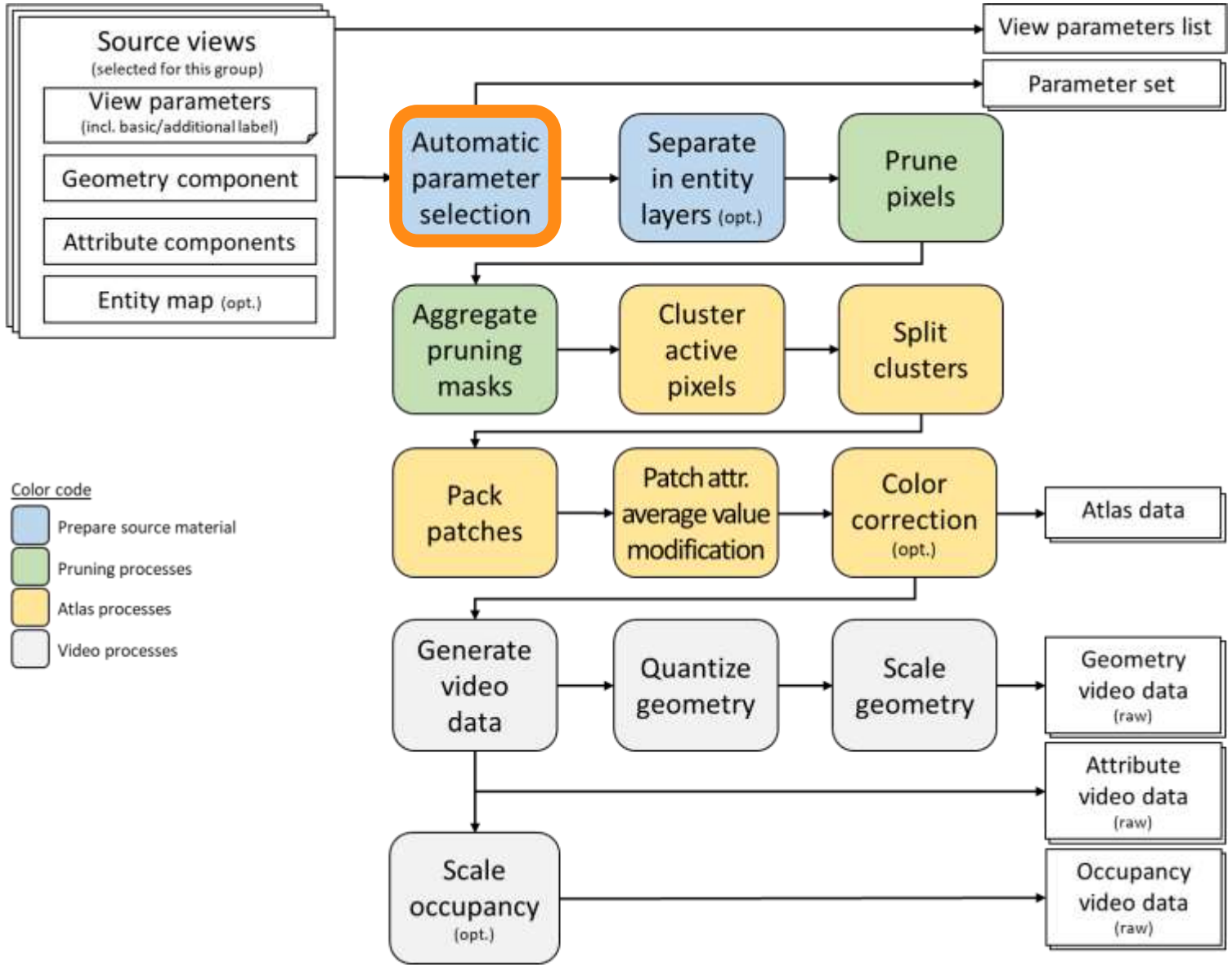
MIV encoder


Source views
View parameters
Geometry component
Attribute components
Entity map (opt.)
Split source in groups
Label views
(basic, additional)
SEI messages
Encode groups
(Figure 2)
Parameter set
View parameters list
Atlas data
Format Bitstream
(V3C sample stream with MIV extensions)
Geometry video data
(raw)
Attribute video data
(raw)
Occupancy video data
(raw)
Encode video sub bitstreams
(HM / VTM)
Packing
(VTM merge tool)
Multiplex
Bitstream
(one file)
Color code
Prepare source material
Per-group encoding
Bitstream formatting
Video encoding

MIV – encoding of one group



MIV – encoding of one group

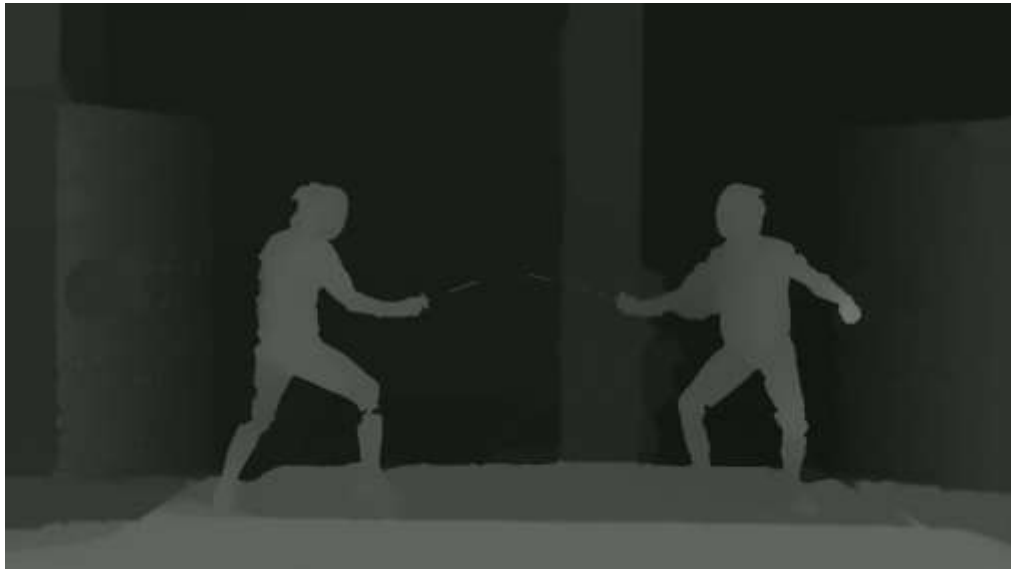


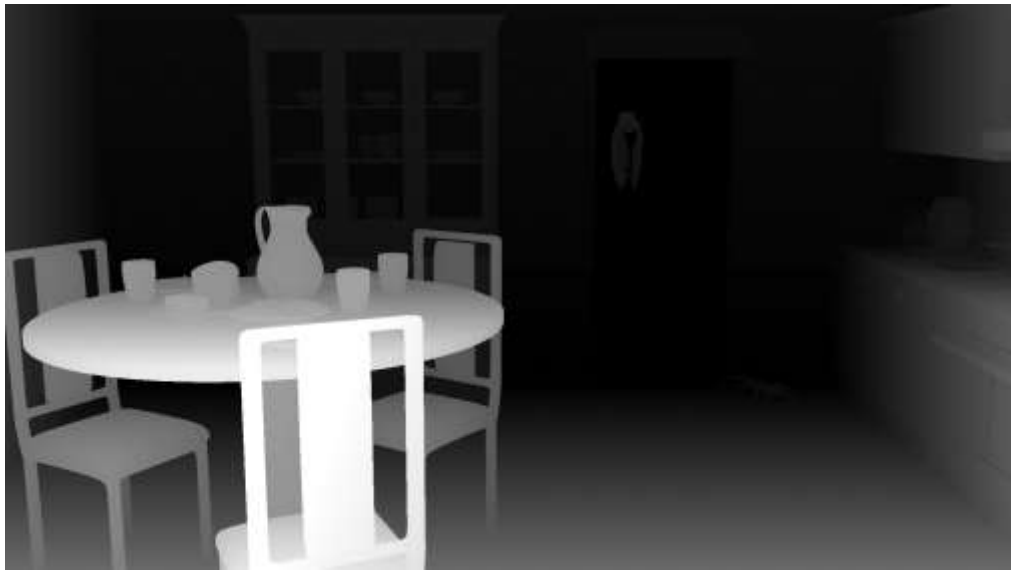
Automatic parameter selection

- Depth quality assessment
- Calculating the size of the atlas

Automatic parameter selection

- Depth quality assessment
- Calculating the size of the atlas





Automatic parameter selection

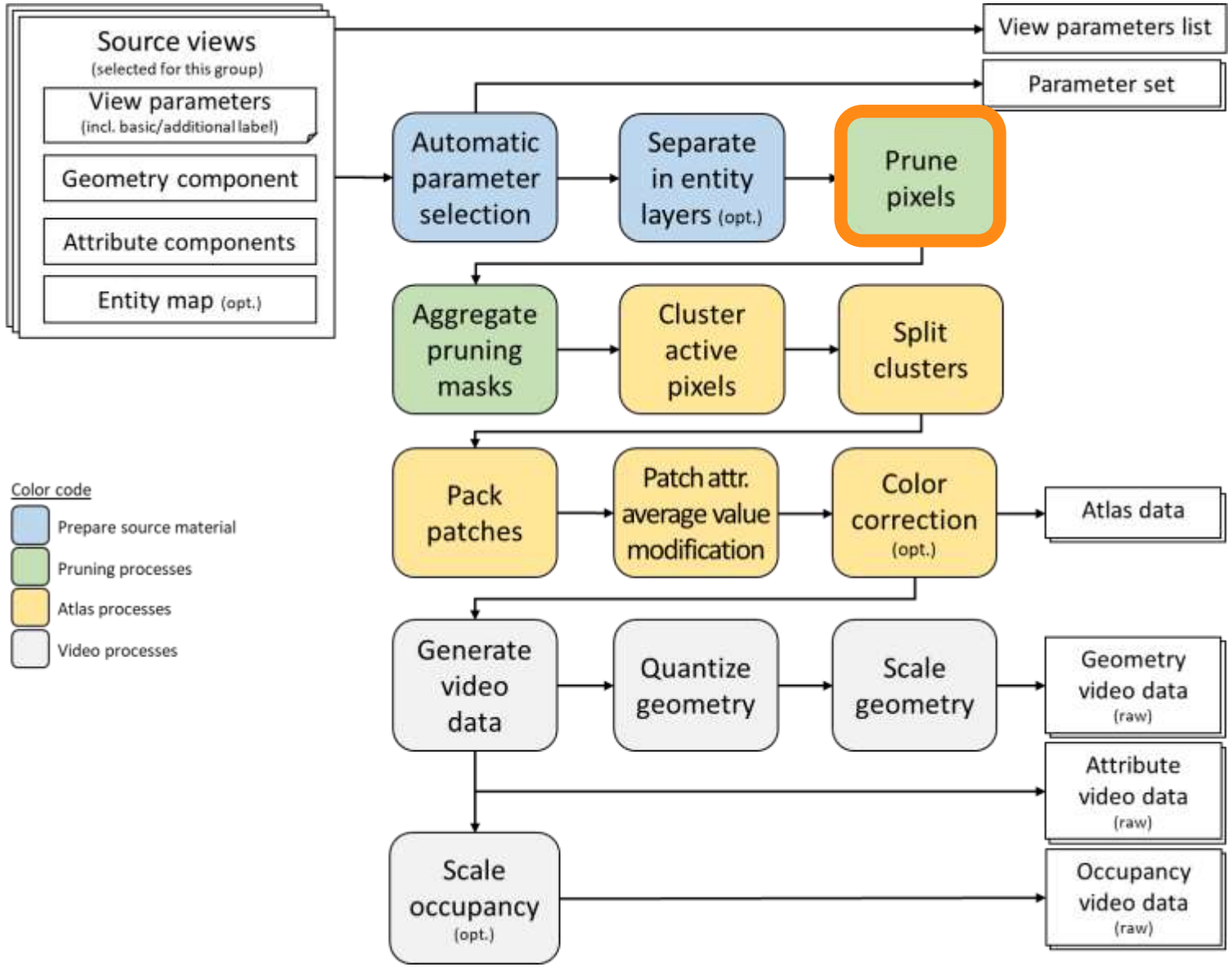
- Depth quality assessment
- Calculating the size of the atlas

Requirements:
- Maximum 4 decoder instances
- Maximum 8 MPix per one video

Solution:
- 2 atlases (2 × attributes + 2 × geometry)
- Atlas width: width of the widest input view
- Height of the atlas: the largest possible (meeting previous requirements)

$$W_{atl} = W_{input} \qquad H_{atl} = \frac{8Mpix}{W_{atl}}$$

MIV – encoding of one group



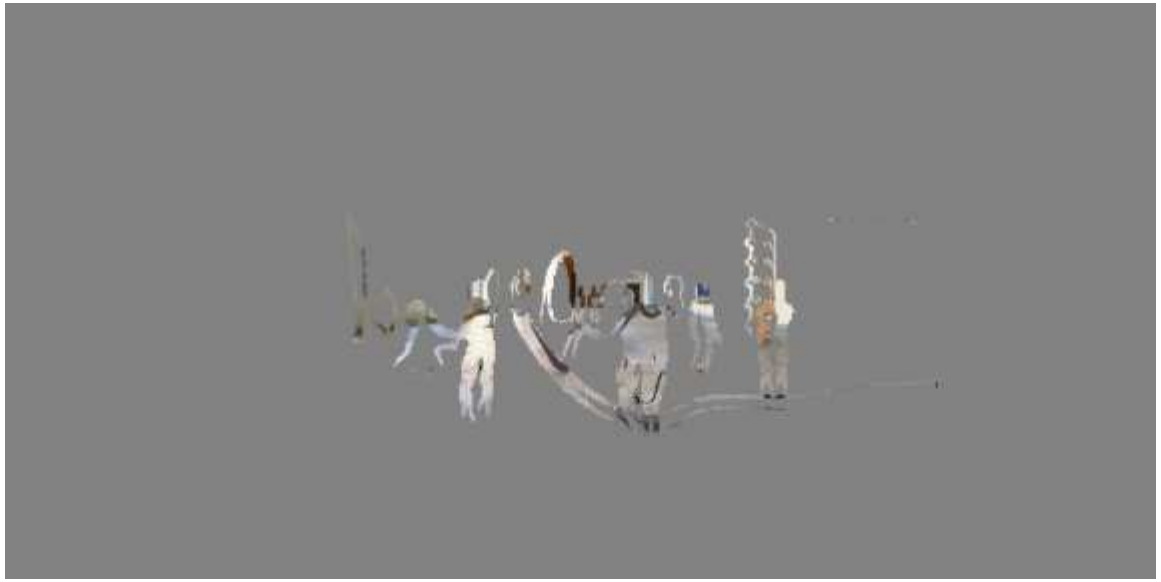
Pixel pruning

- Most of elements visible in a view are also visible in other views

Pixel pruning

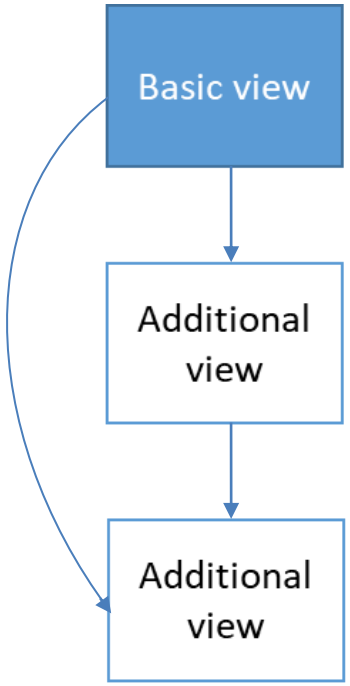
- Most of elements visible in a view are also visible in other views

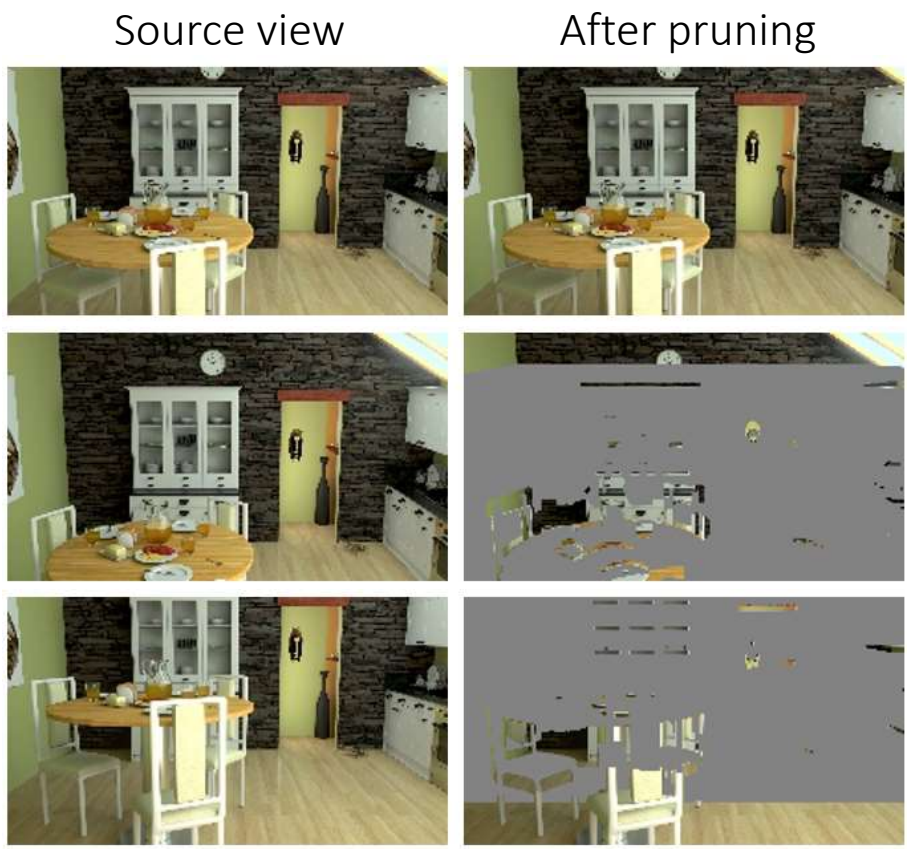
- Pruning – removing inter-view redundancy and selecting the smallest amount of elements required to compress whole scene



Pixel pruning: pruning graph

1. Insert basic views into the pruning graph (as root nodes).
2. Project all pixels of all basic views to each additional view
3. Create the pruning mask for each additional view
4. Select the additional view with maximum number of preserved pixels (to prefer larger patches)
5. Insert selected additional view into the pruning graph (as a child node of all nodes already in graph) and stop if all the views are assigned to nodes in the pruning graph
6. Project all preserved pixels of selected view to remaining additional views
7. Update the pruning mask for each remaining additional view
8. Go to 4



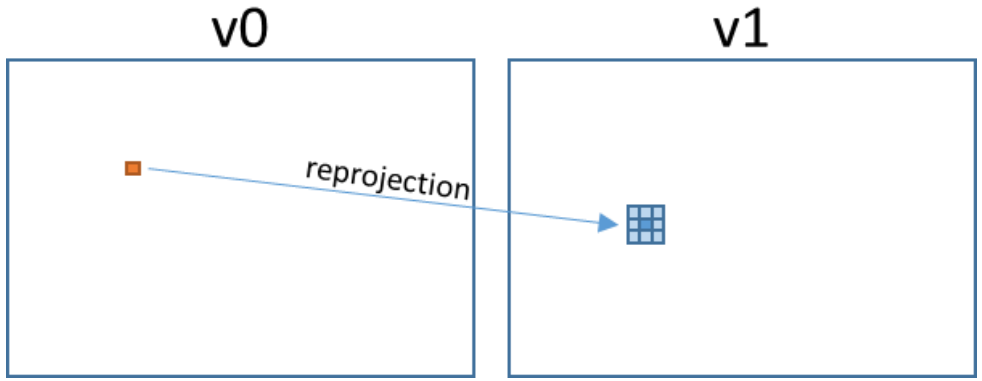


Pixel pruning: pruning mask creation

Conditions for removing points:

- The difference of the **depth** of the transferred point and the point corresponding to it must be higher than a threshold (10%)

- The difference of the **luma** of the transferred point and all points in the corresponding 3 × 3 block must be higher than the threshold
    - Threshold value depends on the noise level in the sequence, the base value of 4% multiplied by the standard deviation of the Gaussian noise in the 1st frame

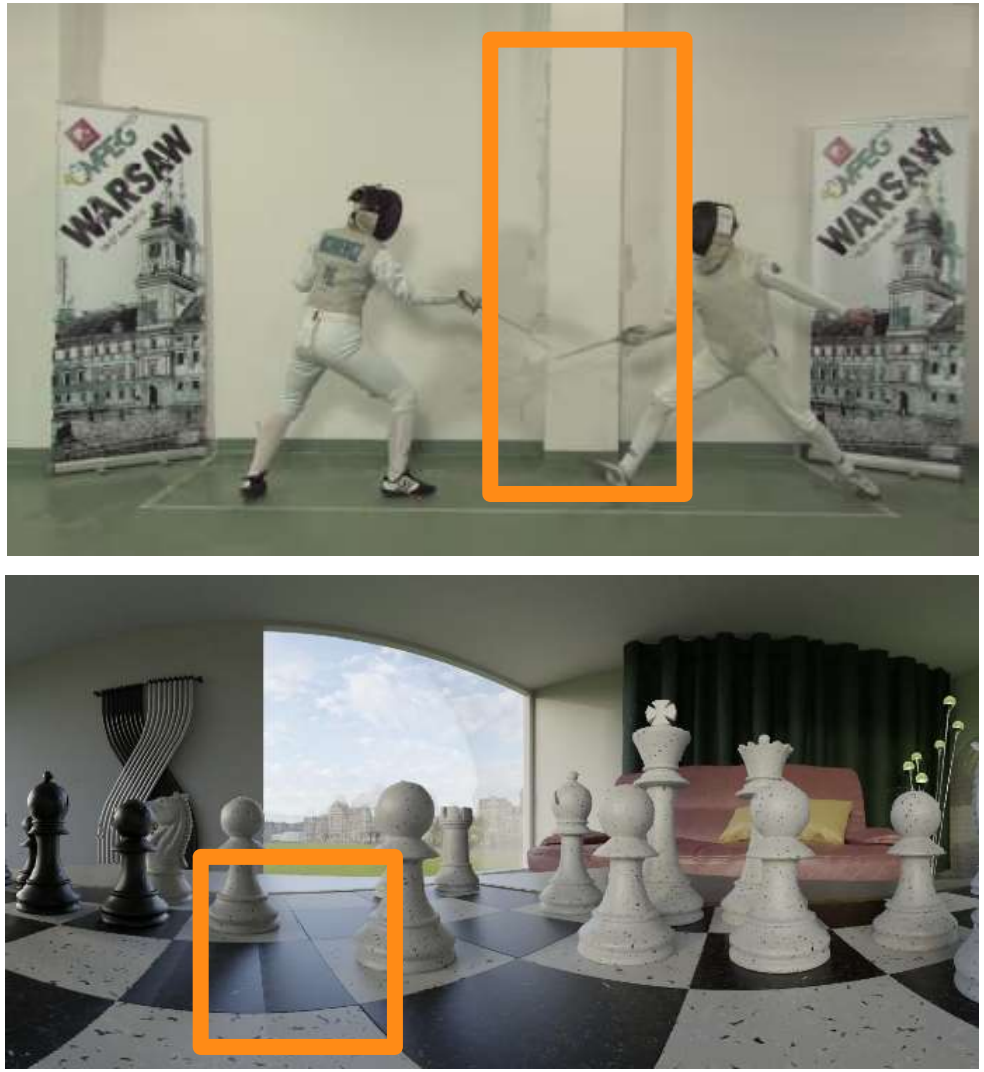


Final binary mask filtration (0: pruned pixel, 1: preserved pixel):

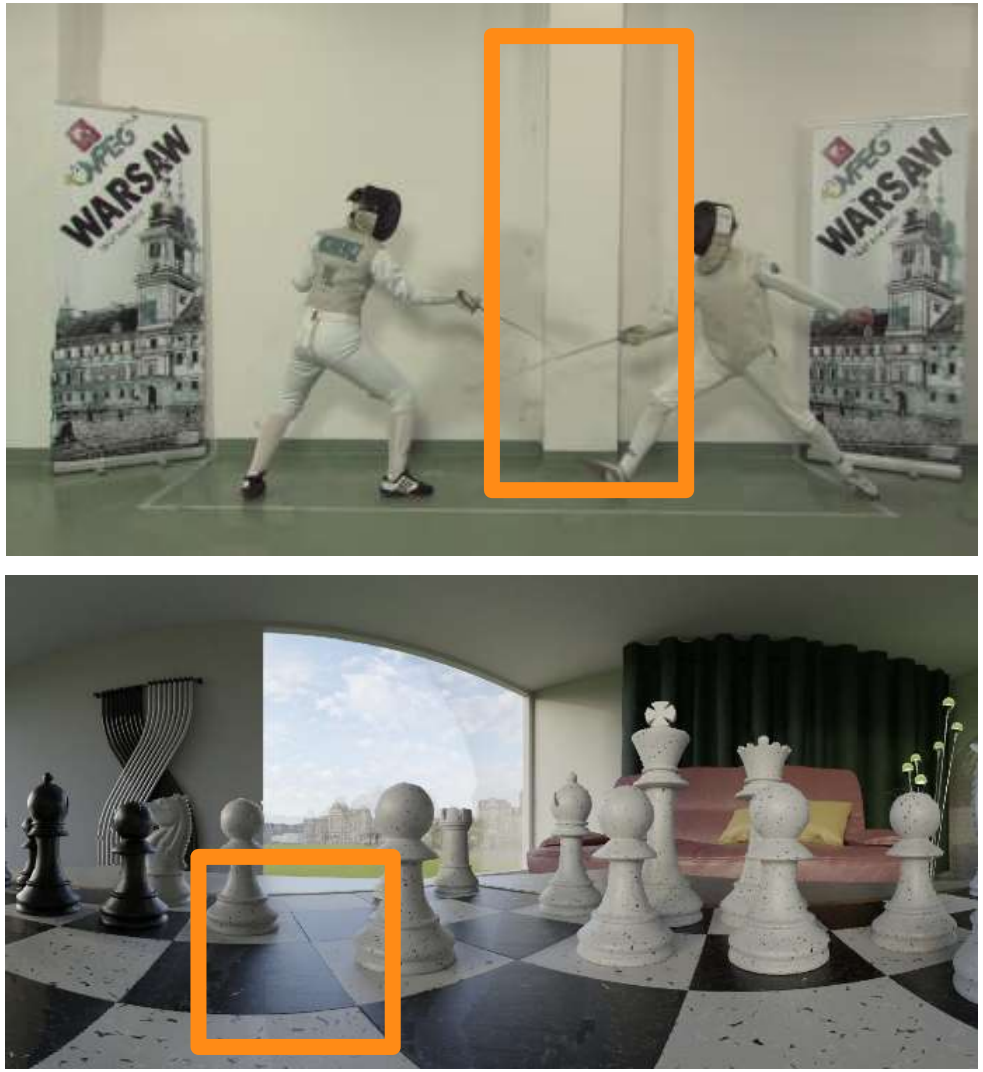
- Iterative erosion (3×3 window)
- Iterative dilation (3×3 window)

Pixel pruning

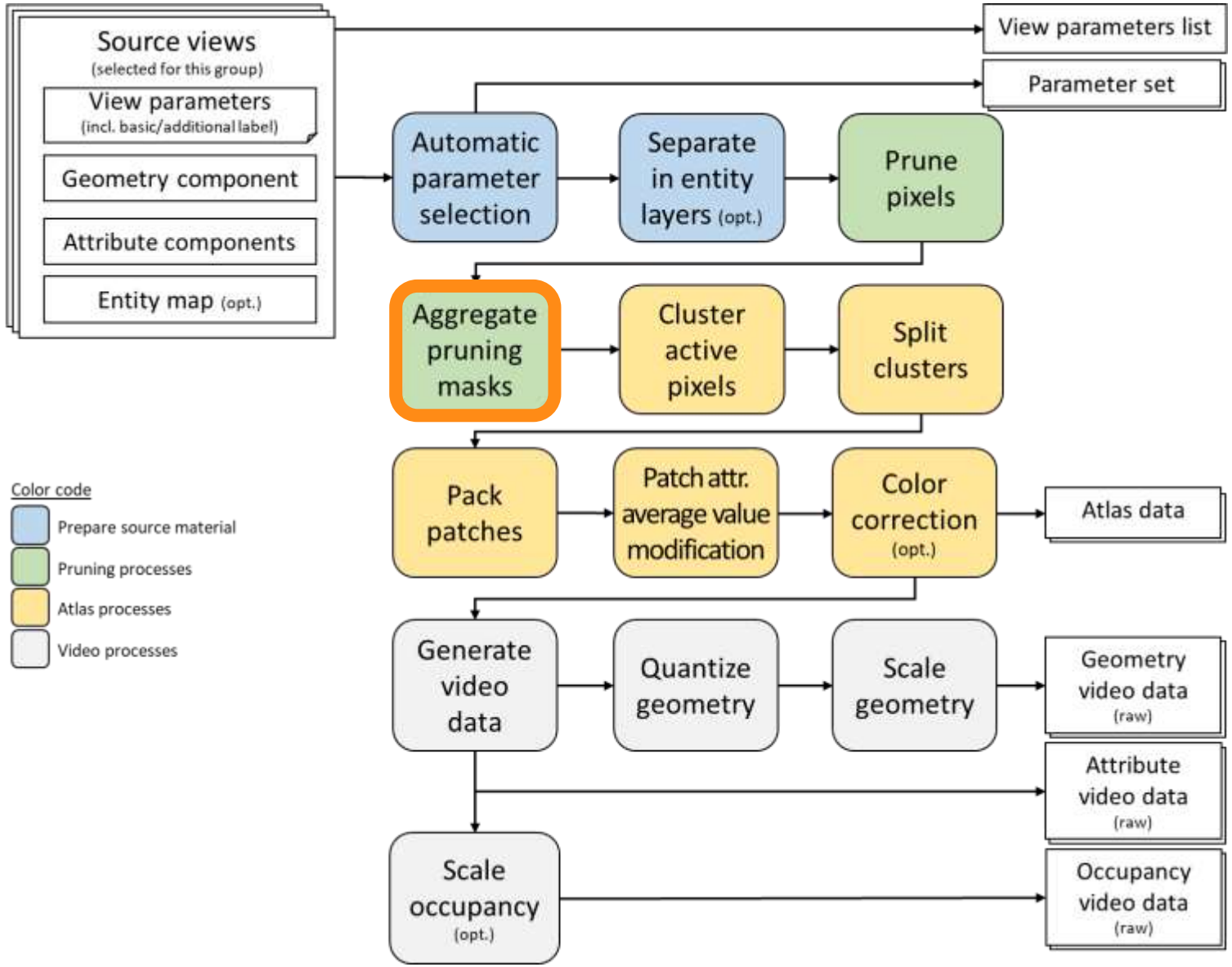
Geometry-based pruning



Texture- and geometry-based pruning

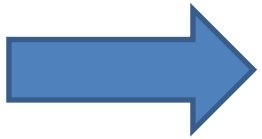


MIV – encoding of one group

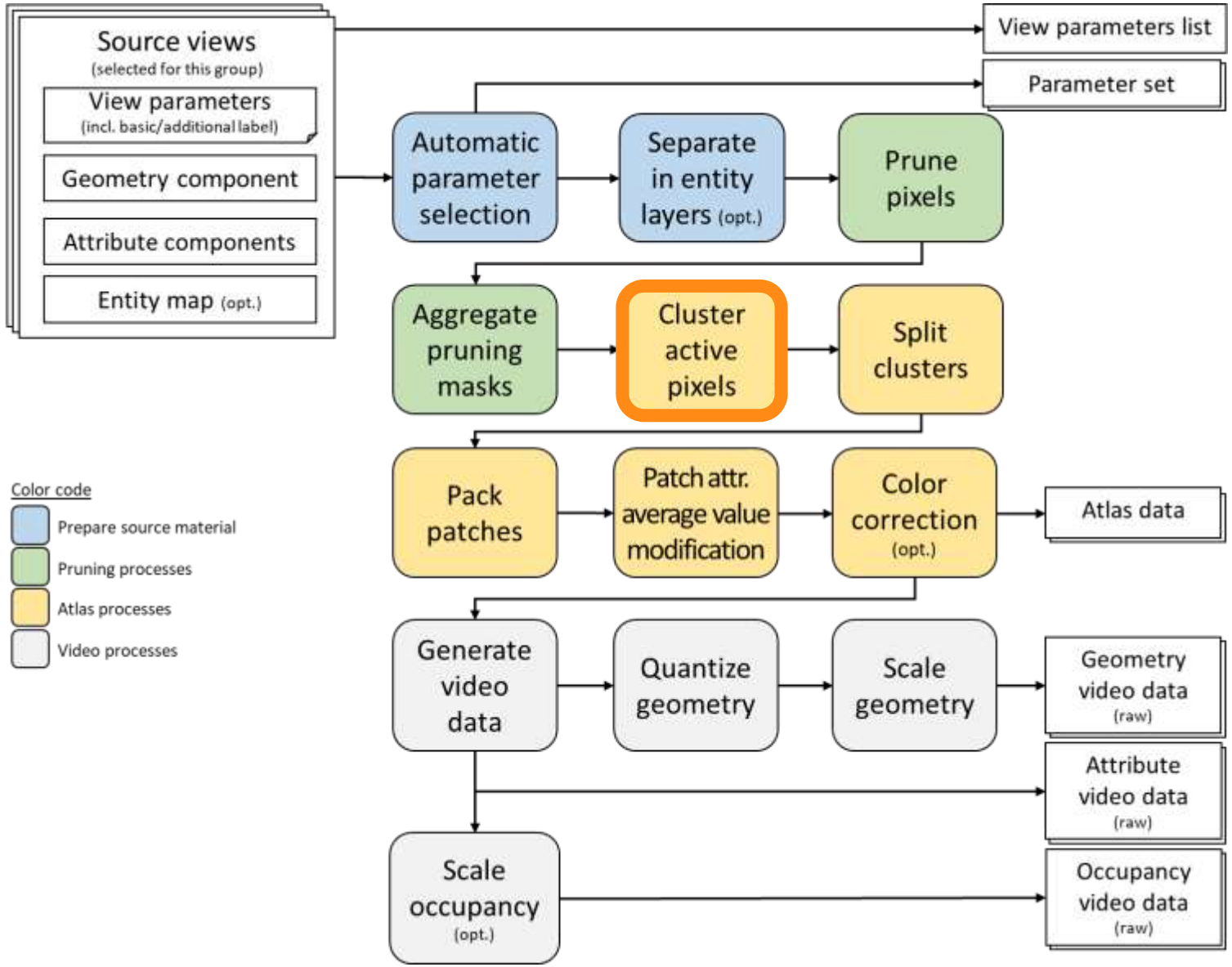


Pruning mask aggregation

Temporal aggregation of pruning masks (in one GOP)



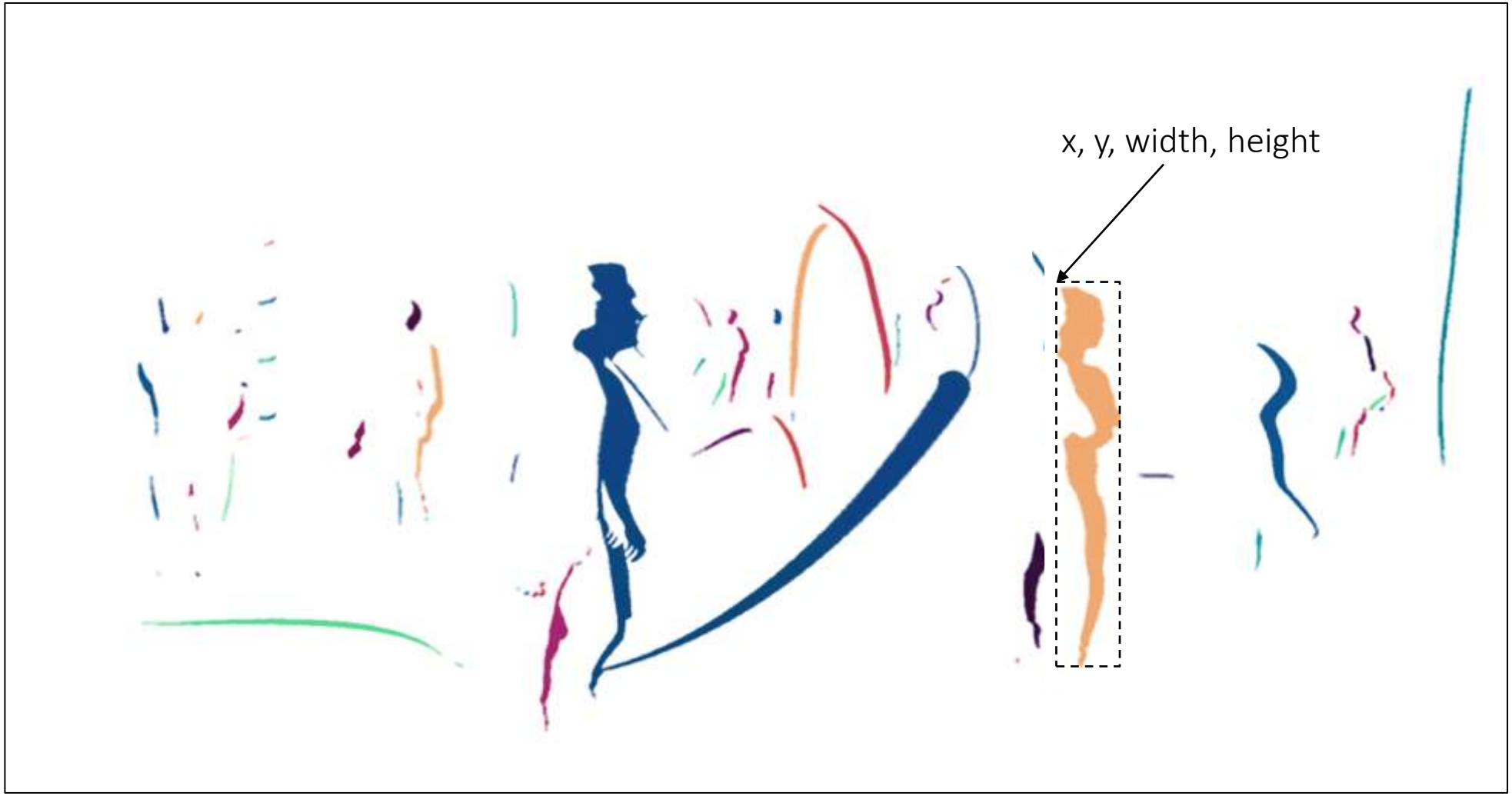
MIV – encoding of one group



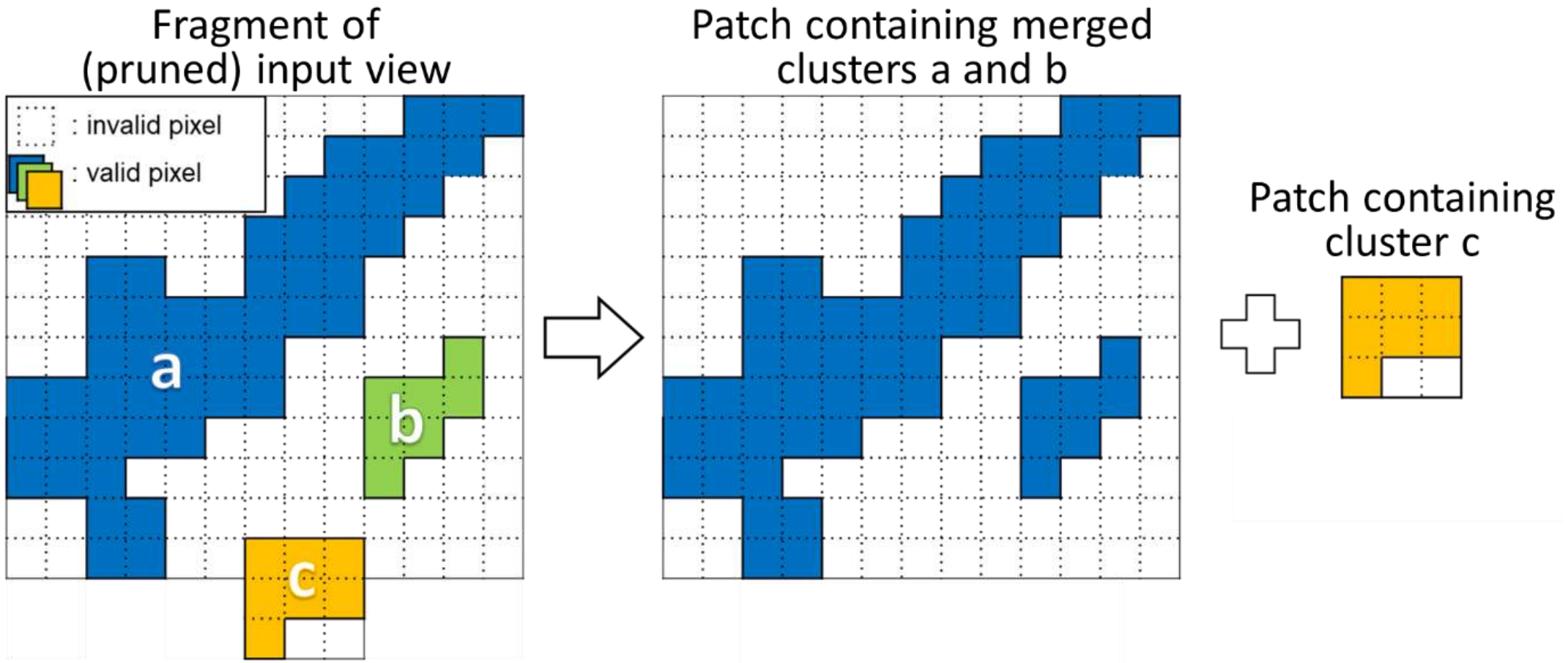
Pixel clustering

Pixel clustering

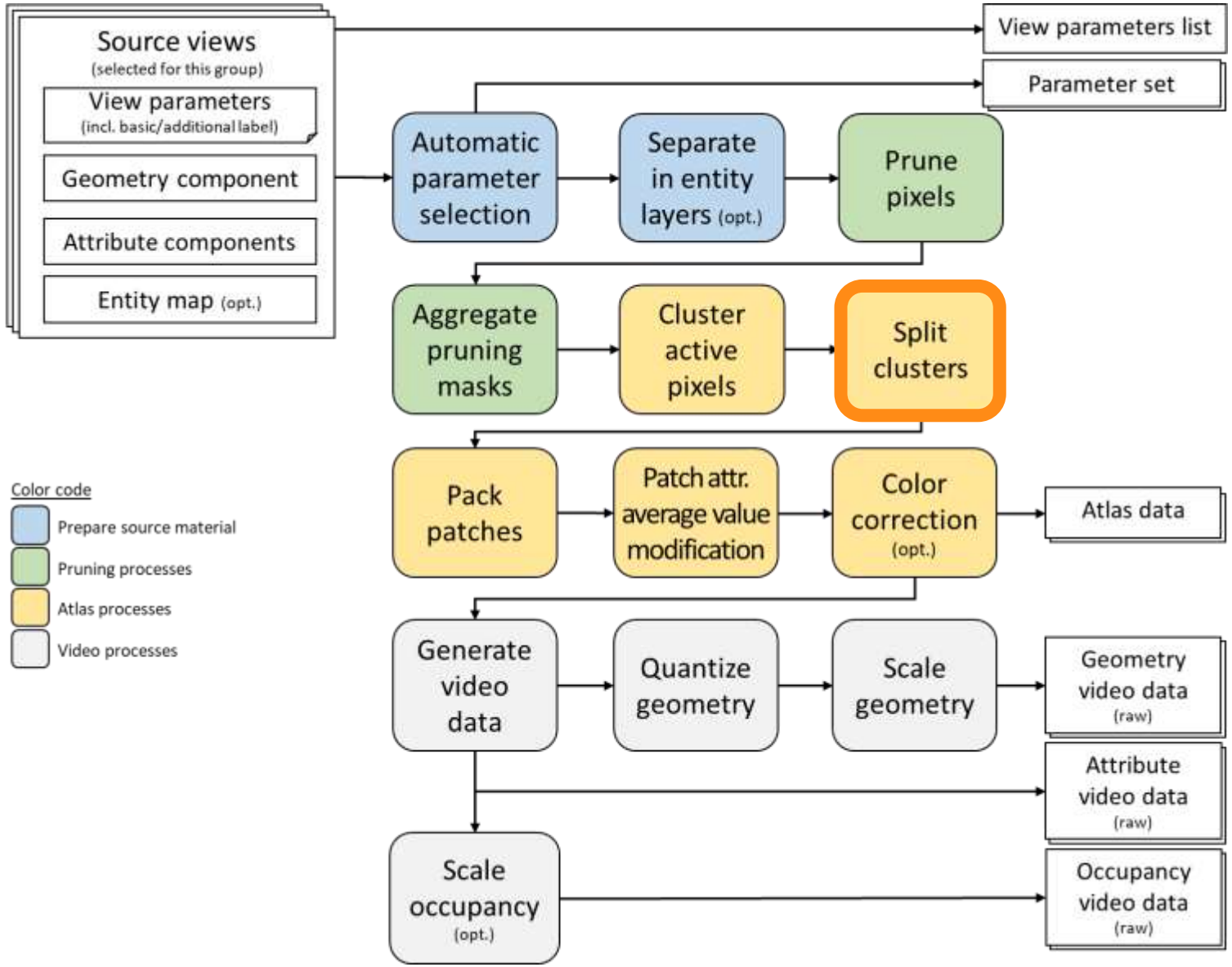
Pixel clustering



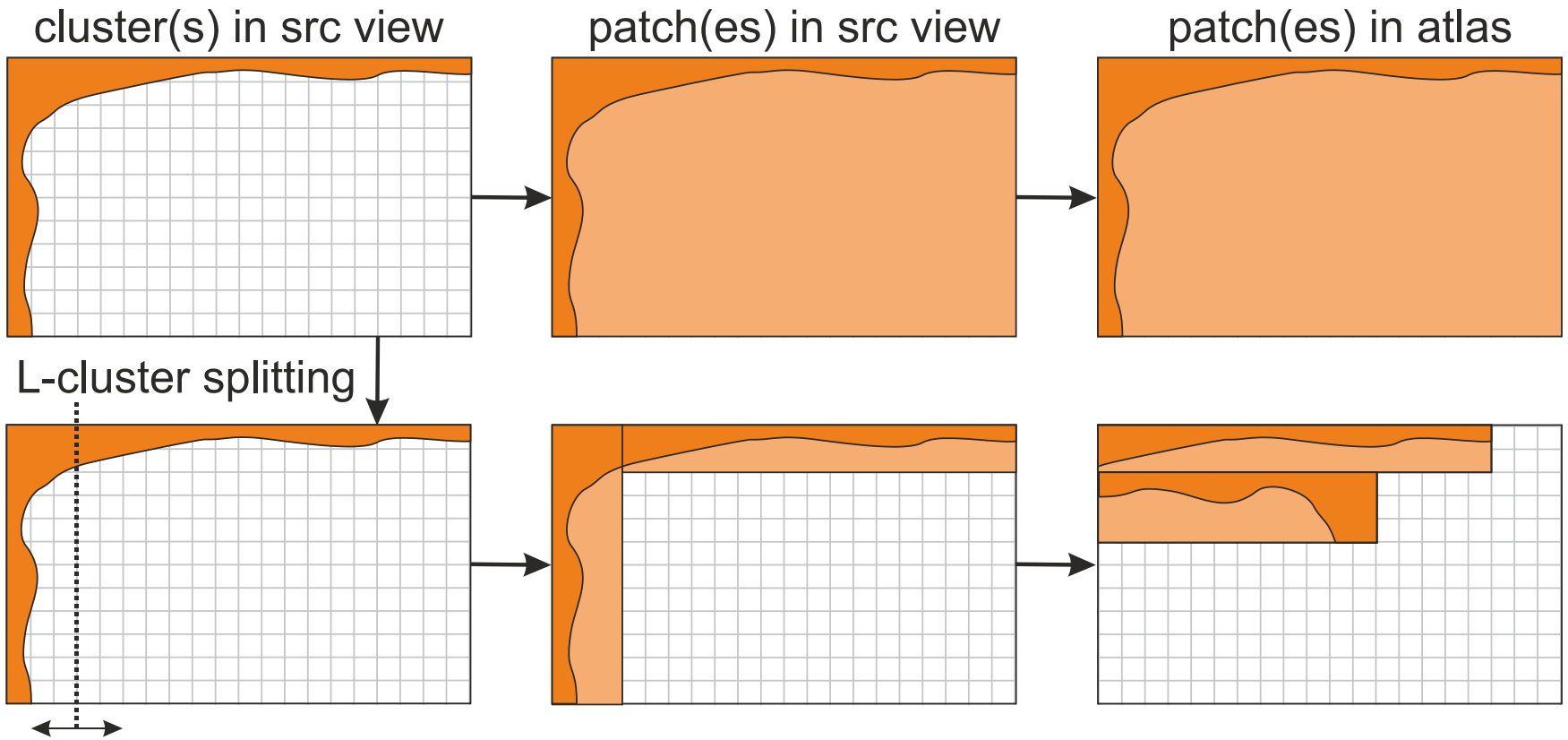
Pixel clustering: cluster merging



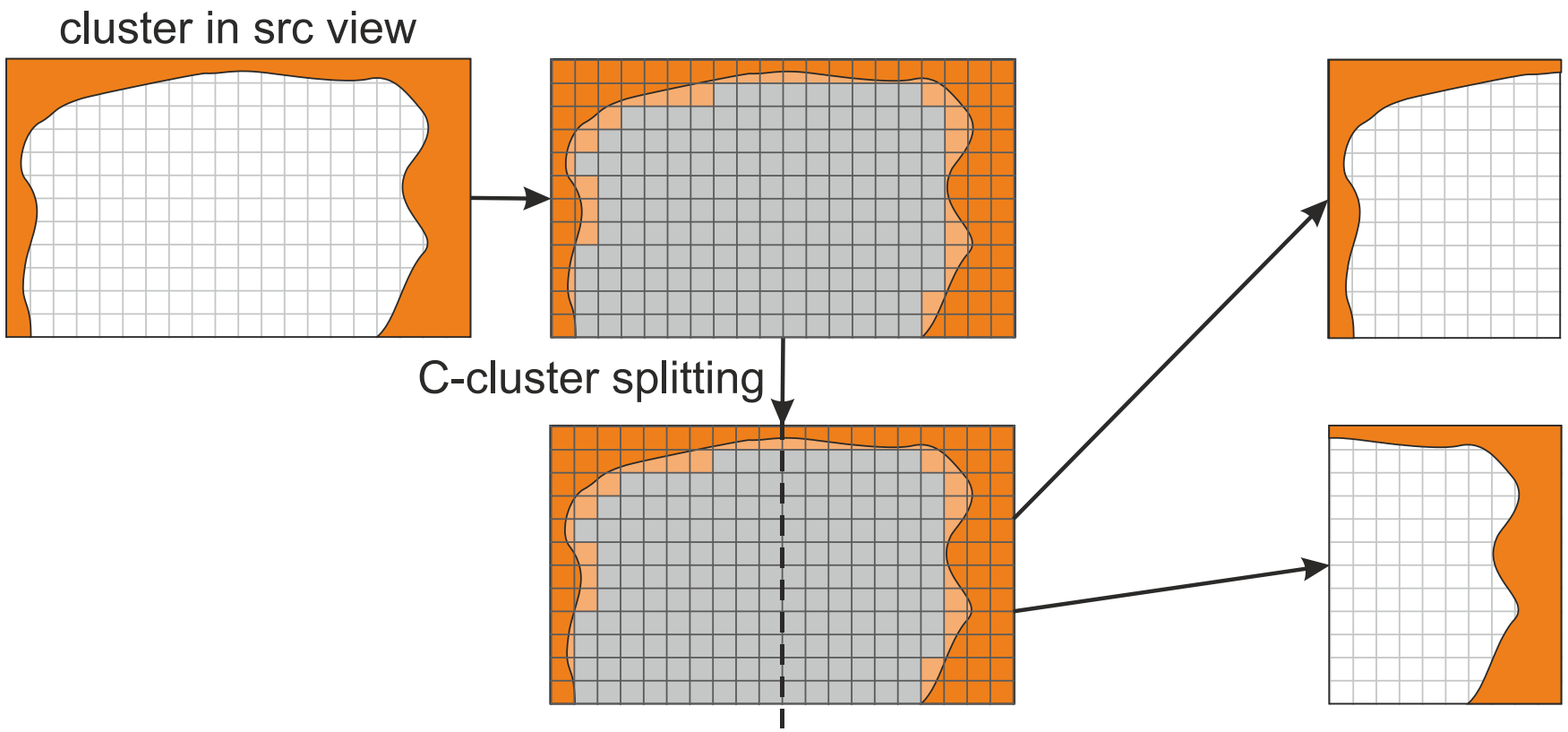
MIV – encoding of one group



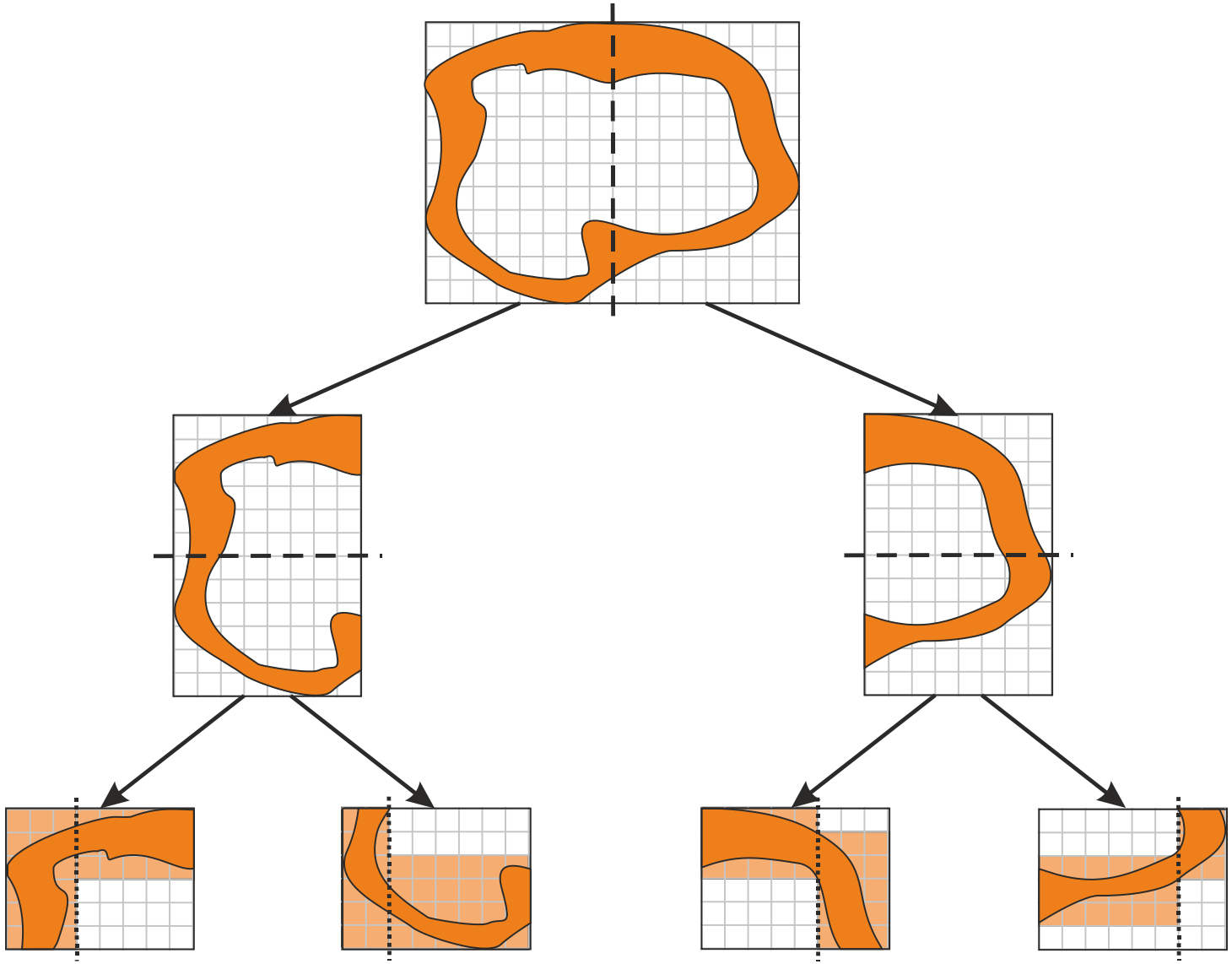
Cluster splitting



Cluster splitting



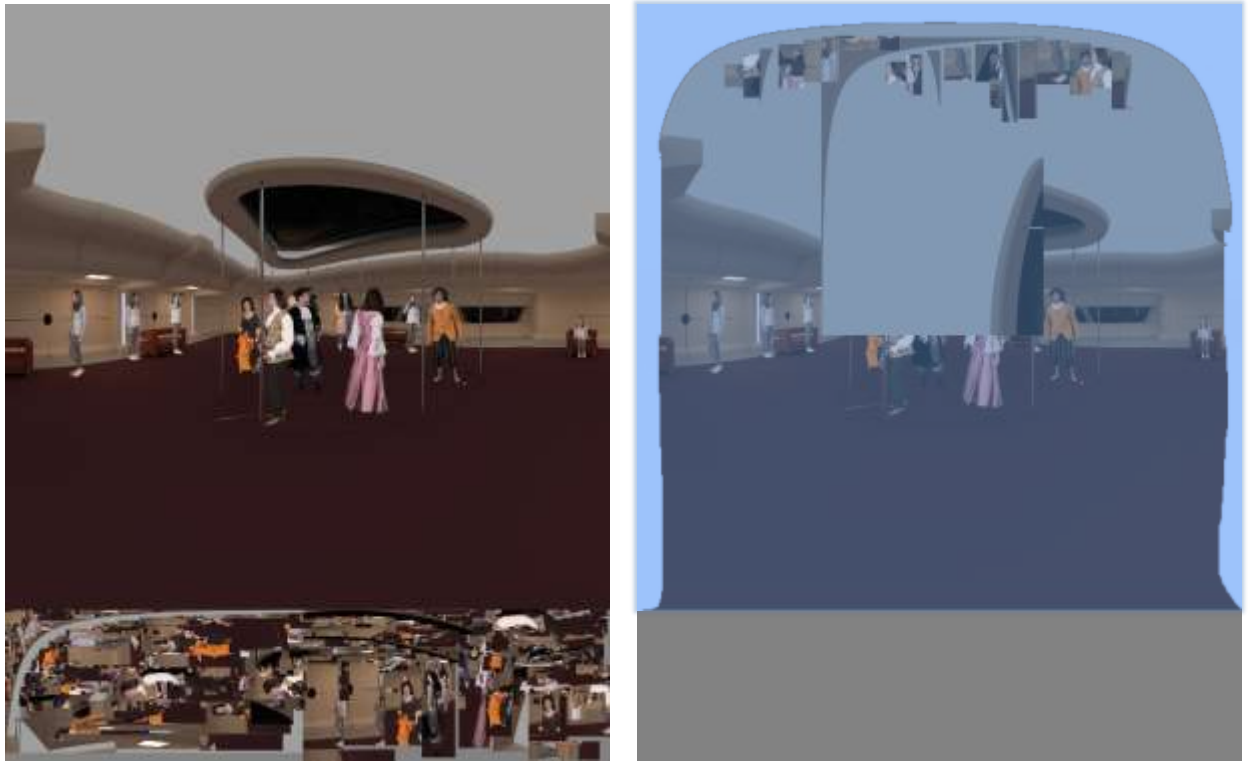
Cluster splitting



Cluster splitting

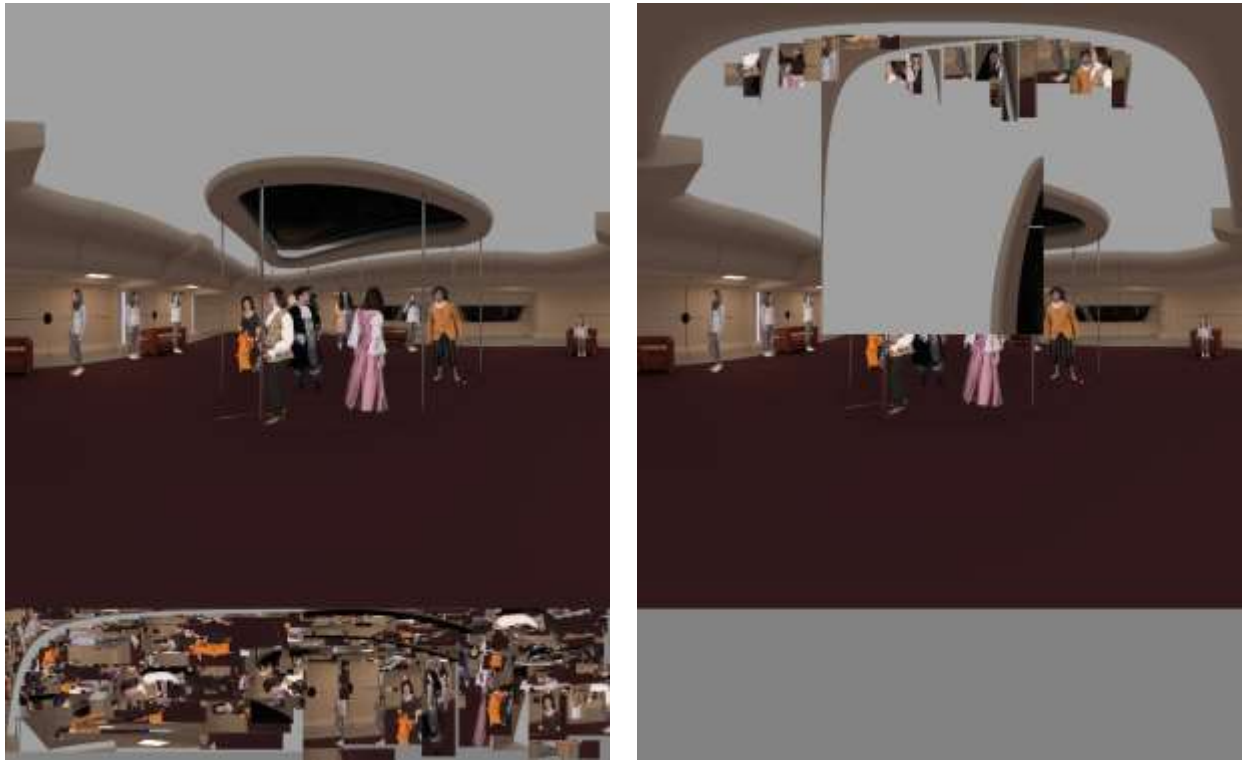
No splitting

Cluster splitting



No splitting

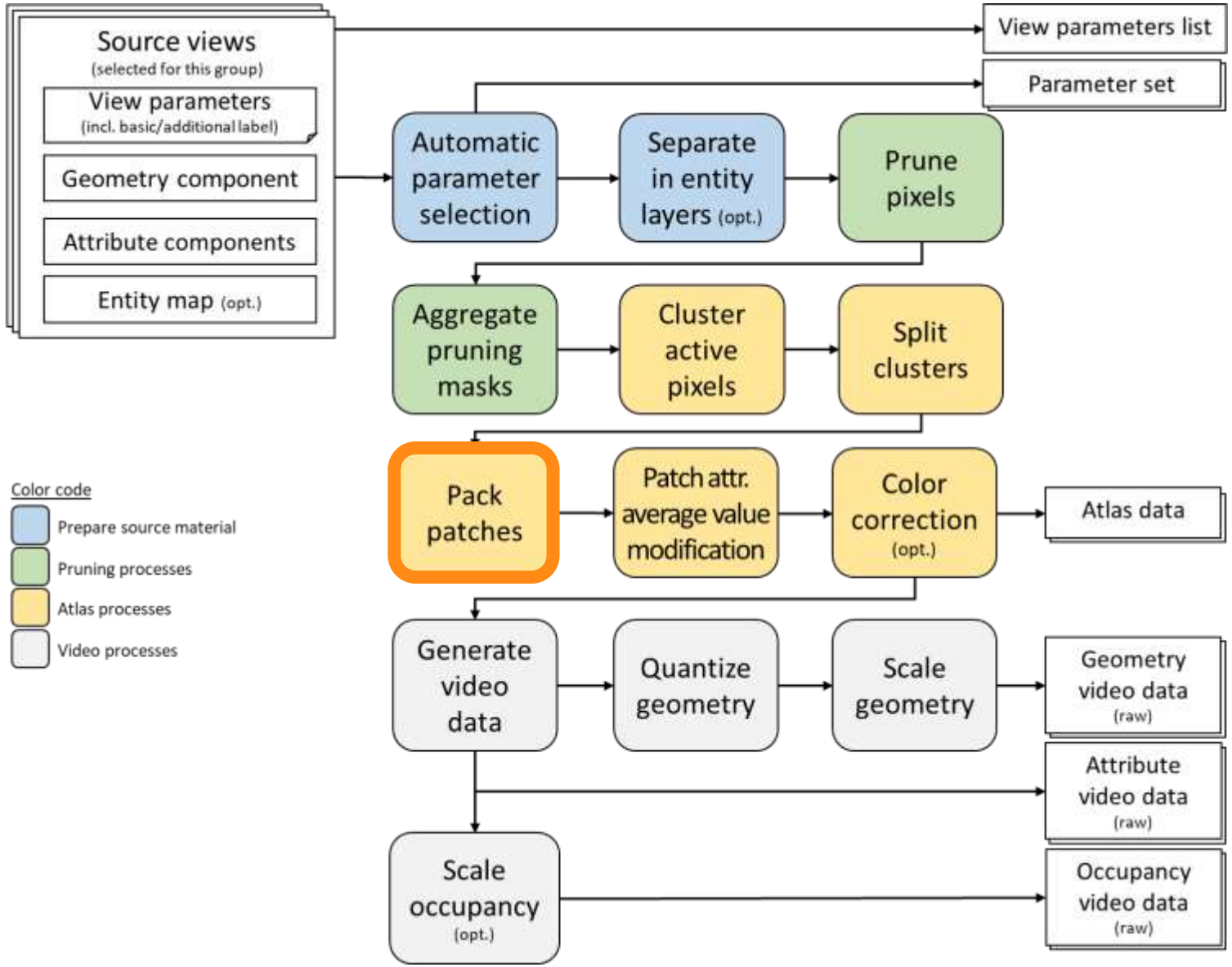
Cluster splitting



No splitting

Splitting enabled

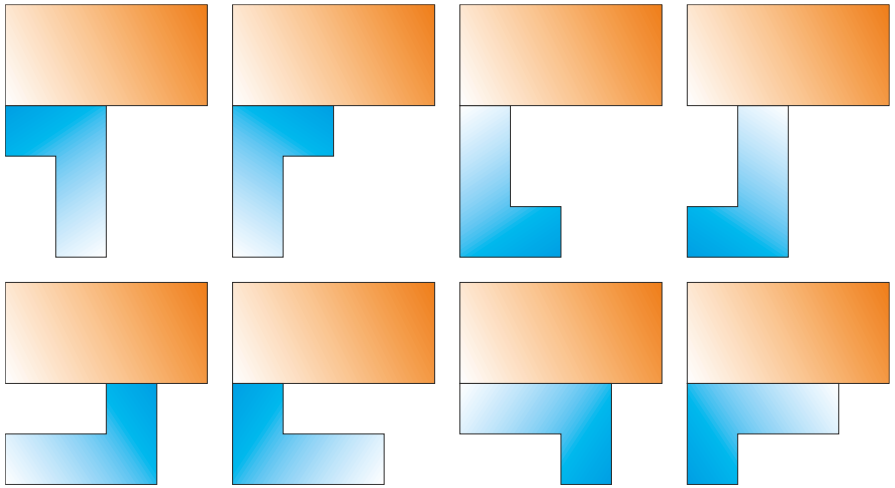
MIV – encoding of one group

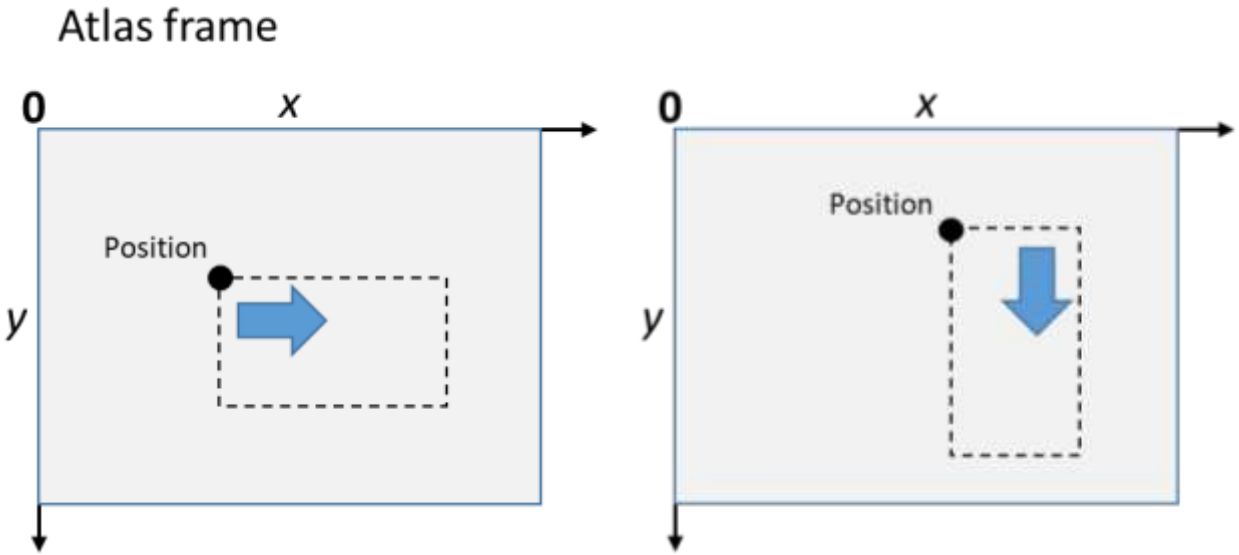


Patch packing

8 possible positioning variants:

- 4 rotations (0°, 90°, 180°, 270°)
- Optional mirroring



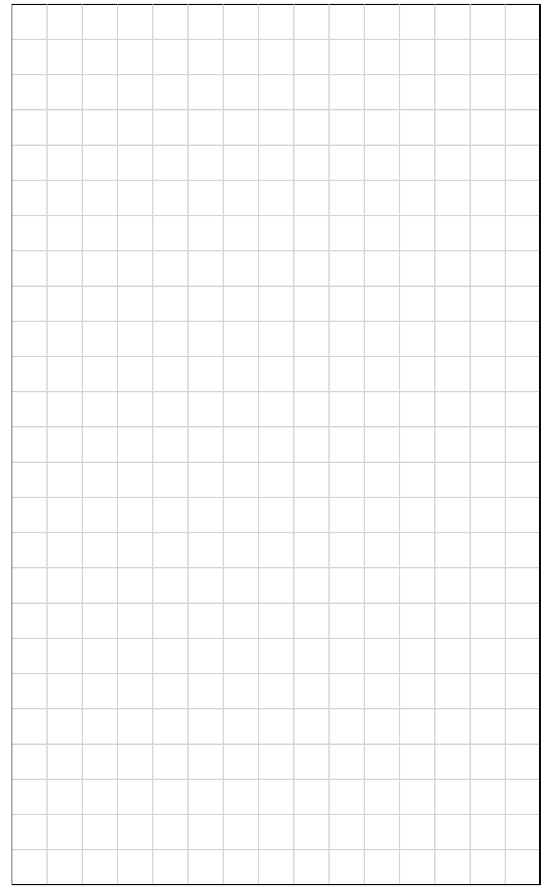


Patch packing

For each cluster:
    For each atlas:
        Push the cluster in “Used Space” (0° rotation first, 90° otherwise)
        If the push failed:
            Push the cluster into “Free Space” (0° rot. first, 90° otherwise)
            If the push failed:
                Split the cluster into 2 parts by its largest border
                For each resulting 2 parts:
                    If smaller than MinPatchSize:
                        Discard the patch
                    Else:
                        Put the part in the cluster priority list

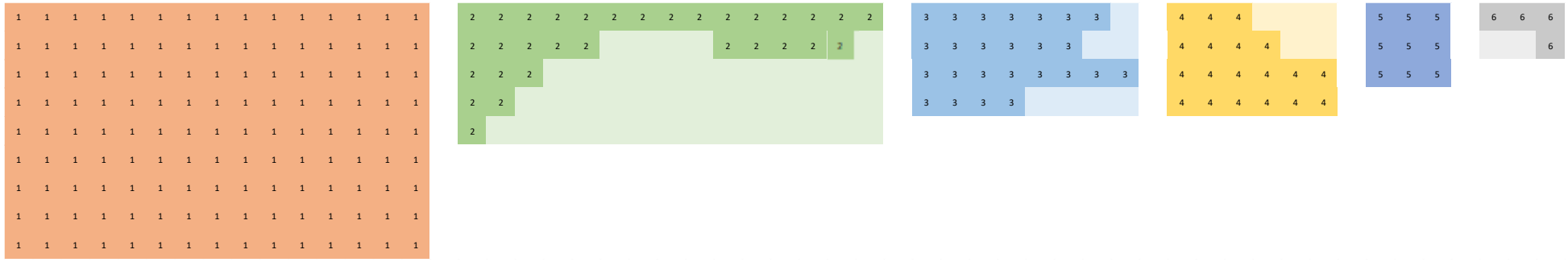
Patch packing

Atlas:



For each cluster:
    For each atlas:
        Push the cluster in “Used Space” (0° rotation first, 90° otherwise)
        If the push failed:
            Push the cluster into “Free Space” (0° rot. first, 90° otherwise)
            If the push failed:
                Split the cluster into 2 parts by its largest border
                For each resulting 2 parts:
                    If smaller than MinPatchSize:
                        Discard the patch
                    Else:
                        Put the part in the cluster priority list

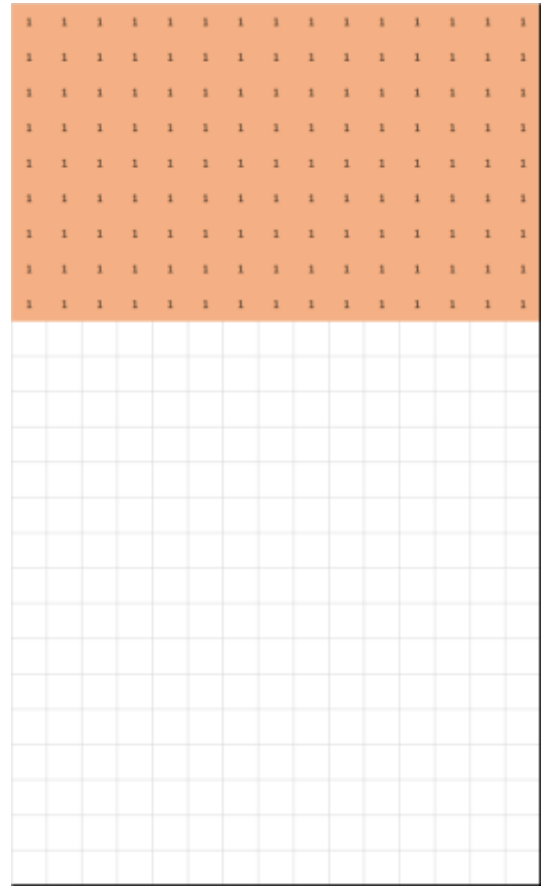
List of clusters sorted by area (1 square: 8 × 8 pixels):



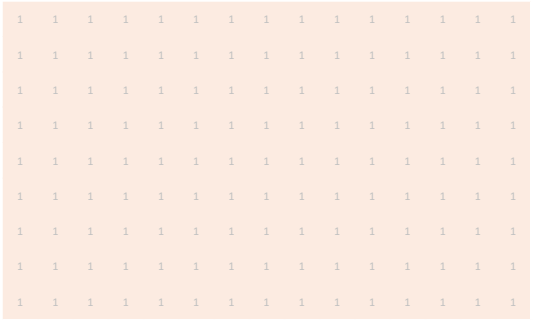
Patch packing

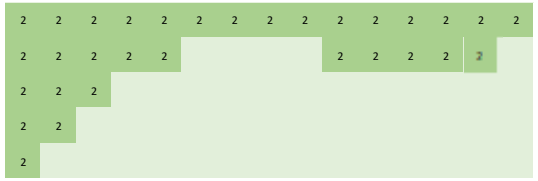
For each cluster:
    For each atlas:
        Push the cluster in “Used Space” (0° rotation first, 90° otherwise)
        If the push failed:
            Push the cluster into “Free Space” (0° rot. first, 90° otherwise)
            If the push failed:
                Split the cluster into 2 parts by its largest border
                For each resulting 2 parts:
                    If smaller than MinPatchSize:
                        Discard the patch
                    Else:
                        Put the part in the cluster priority list

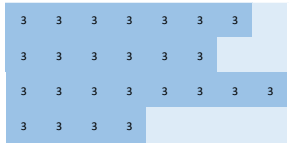
Atlas:

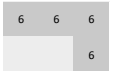


List of clusters sorted by area (1 square: 8 × 8 pixels):





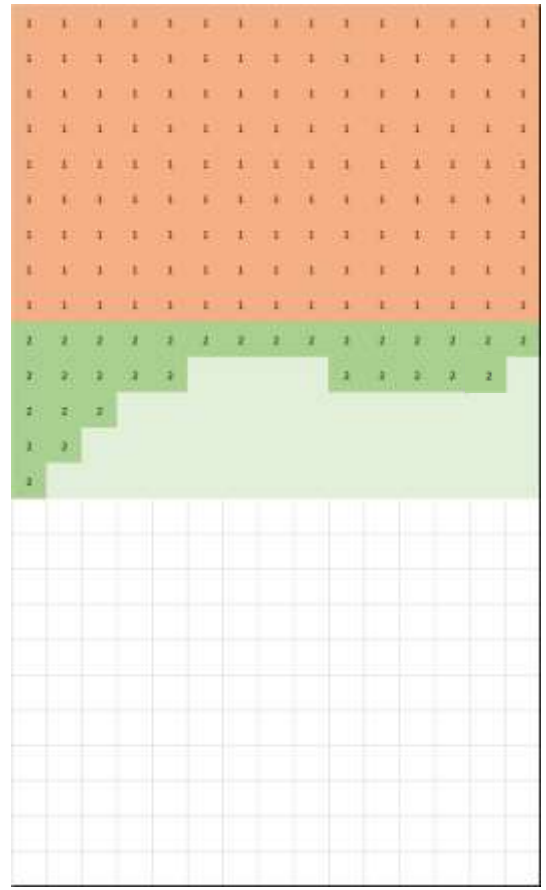




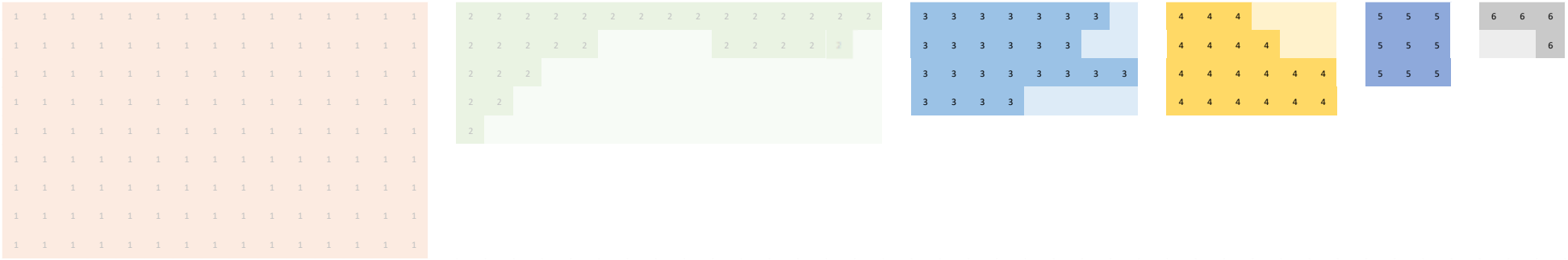
Patch packing

For each cluster:
    For each atlas:
        Push the cluster in “Used Space” (0° rotation first, 90° otherwise)
        If the push failed:
            Push the cluster into “Free Space” (0° rot. first, 90° otherwise)
            If the push failed:
                Split the cluster into 2 parts by its largest border
                For each resulting 2 parts:
                    If smaller than MinPatchSize:
                        Discard the patch
                    Else:
                        Put the part in the cluster priority list

Atlas:



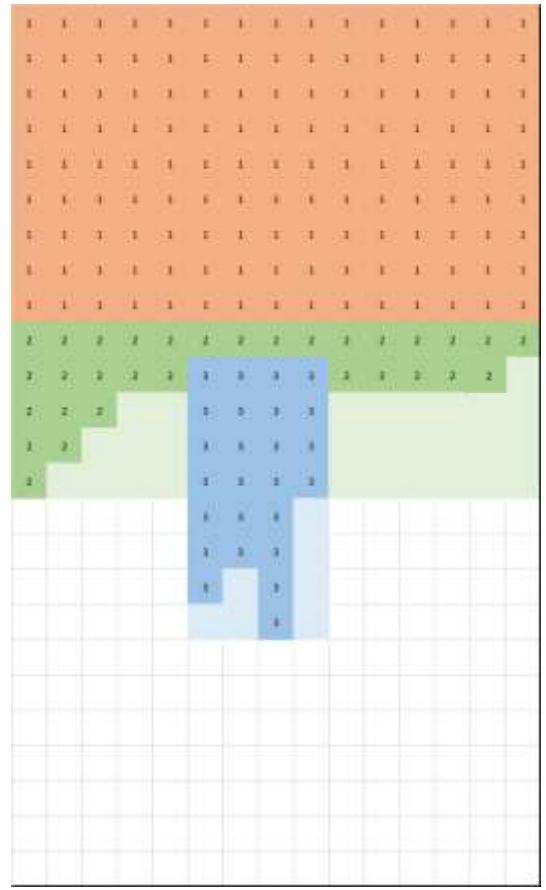
List of clusters sorted by area (1 square: 8 × 8 pixels):



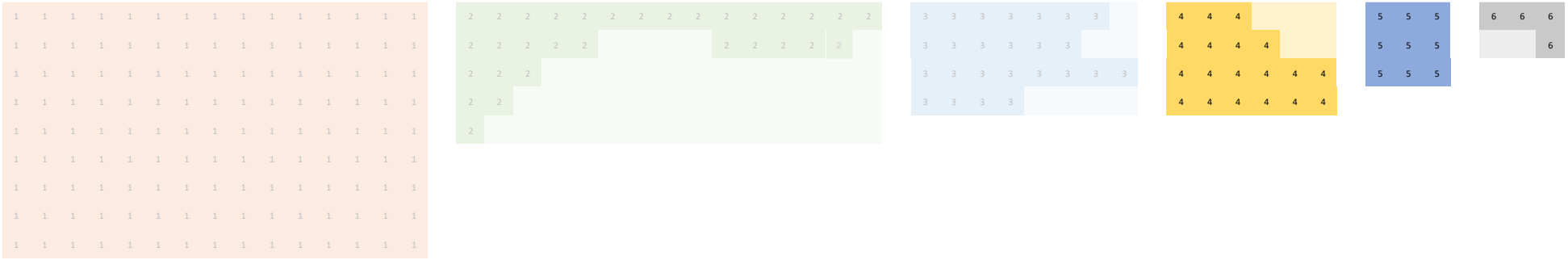
Patch packing

For each cluster:
  For each atlas:
    Push the cluster in "Used Space" (0° rotation first, 90° otherwise)
    If the push failed:
      Push the cluster into "Free Space" (0° rot. first, 90° otherwise)
      If the push failed:
        Split the cluster into 2 parts by its largest border
        For each resulting 2 parts:
          If smaller than MinPatchSize:
            Discard the patch
          Else:
            Put the part in the cluster priority list

Atlas:



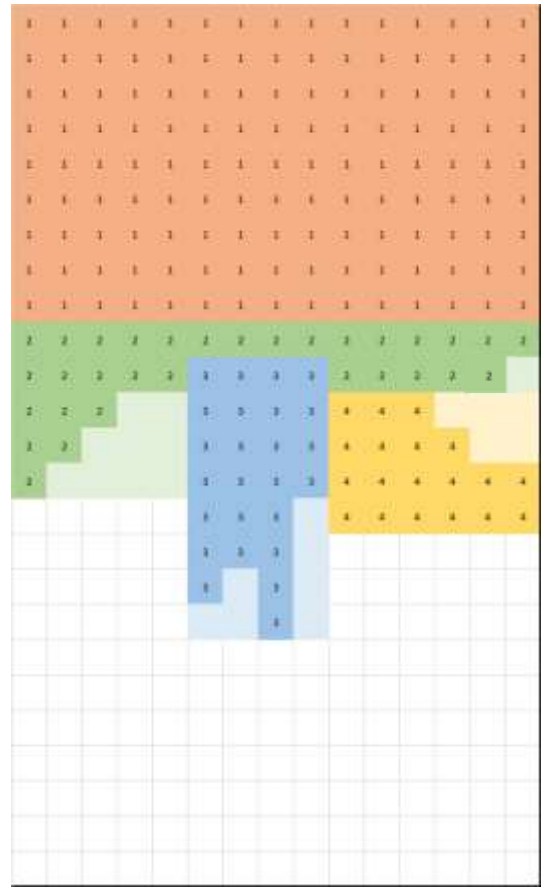
List of clusters sorted by area (1 square: 8 × 8 pixels):



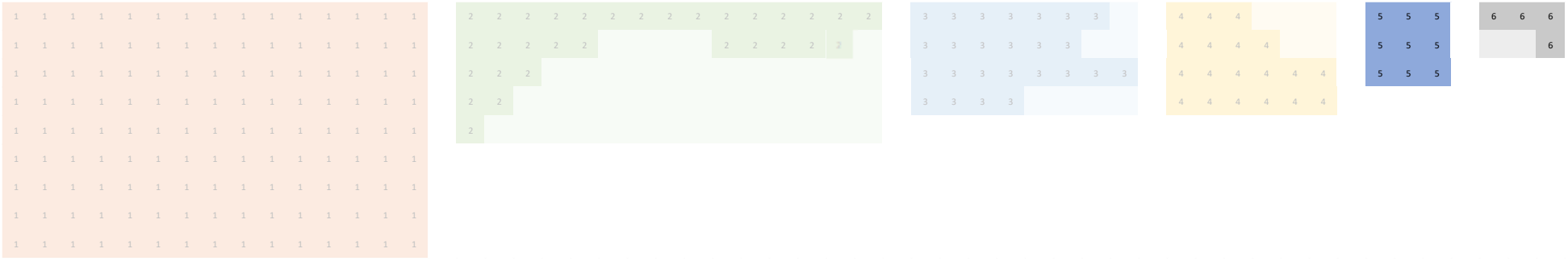
Patch packing

For each cluster:
    For each atlas:
        Push the cluster in "Used Space" (0° rotation first, 90° otherwise)
        If the push failed:
            Push the cluster into "Free Space" (0° rot. first, 90° otherwise)
            If the push failed:
                Split the cluster into 2 parts by its largest border
                For each resulting 2 parts:
                    If smaller than MinPatchSize:
                        Discard the patch
                    Else:
                        Put the part in the cluster priority list

Atlas:



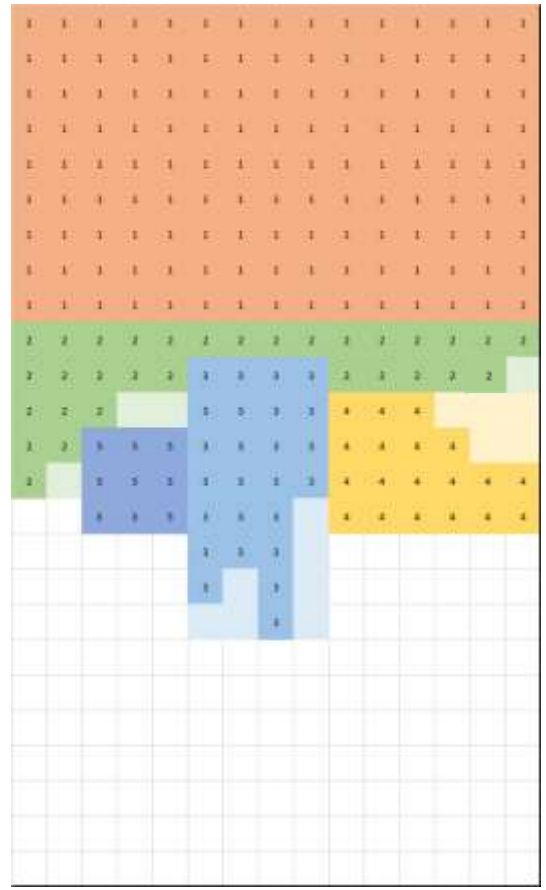
List of clusters sorted by area (1 square: 8 × 8 pixels):



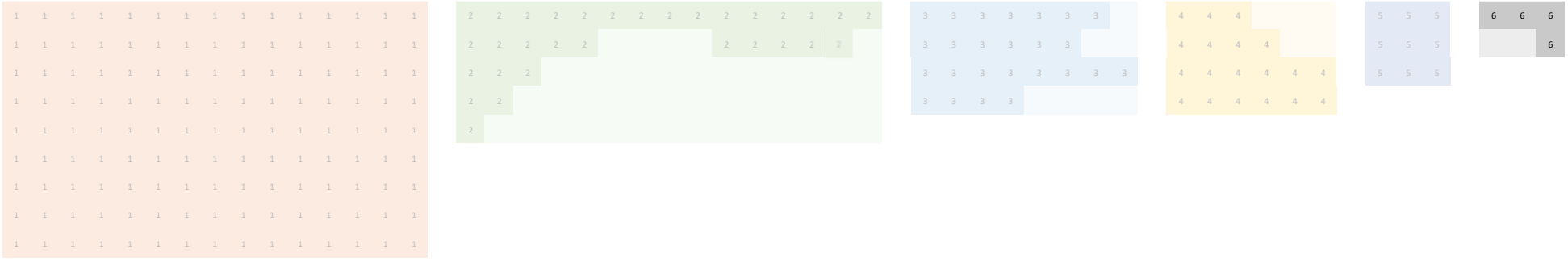
Patch packing

For each cluster:
    For each atlas:
        Push the cluster in "Used Space" (0° rotation first, 90° otherwise)
        If the push failed:
            Push the cluster into "Free Space" (0° rot. first, 90° otherwise)
            If the push failed:
                Split the cluster into 2 parts by its largest border
                For each resulting 2 parts:
                    If smaller than MinPatchSize:
                        Discard the patch
                    Else:
                        Put the part in the cluster priority list

Atlas:



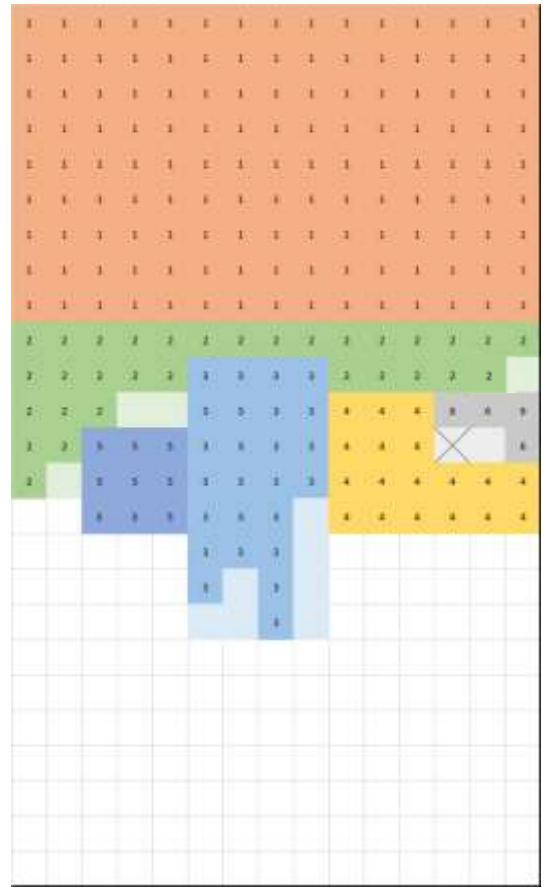
List of clusters sorted by area (1 square: 8 × 8 pixels):



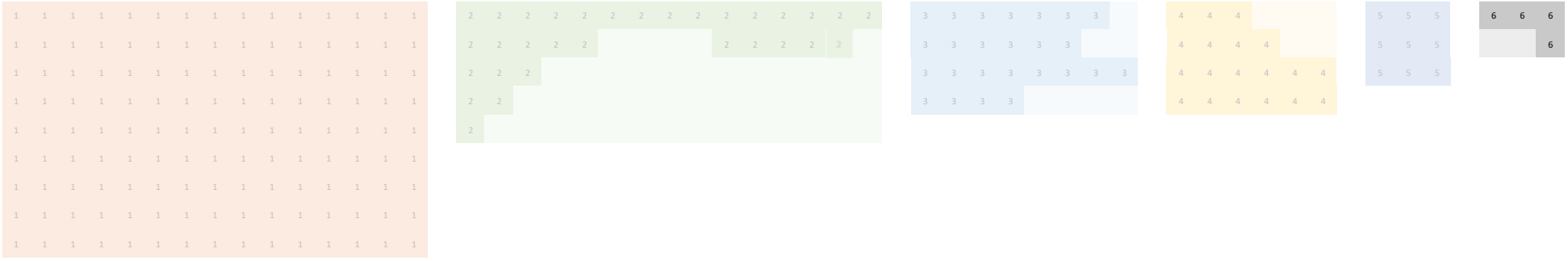
Patch packing

For each cluster:
    For each atlas:
        Push the cluster in "Used Space" (0° rotation first, 90° otherwise)
        If the push failed:
            Push the cluster into "Free Space" (0° rot. first, 90° otherwise)
            If the push failed:
                Split the cluster into 2 parts by its largest border
                For each resulting 2 parts:
                    If smaller than MinPatchSize:
                        Discard the patch
                    Else:
                        Put the part in the cluster priority list

Atlas:



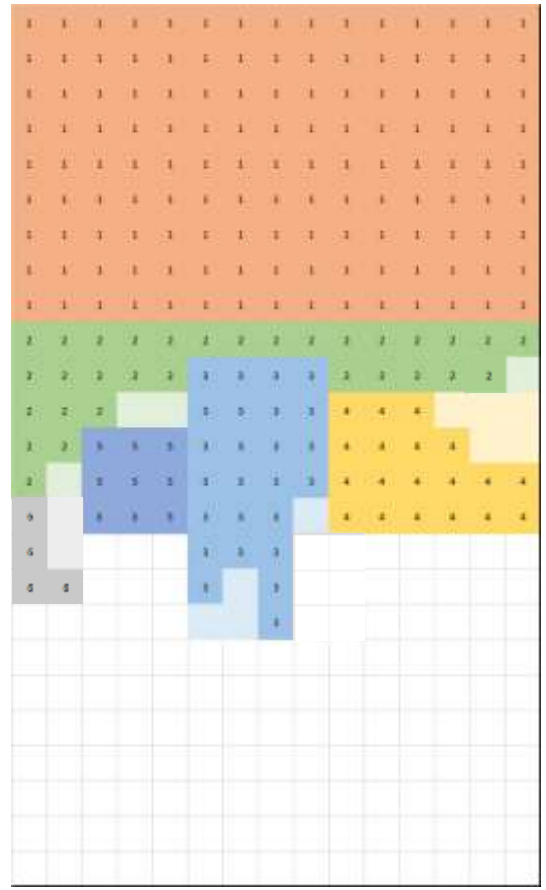
List of clusters sorted by area (1 square: 8 × 8 pixels):



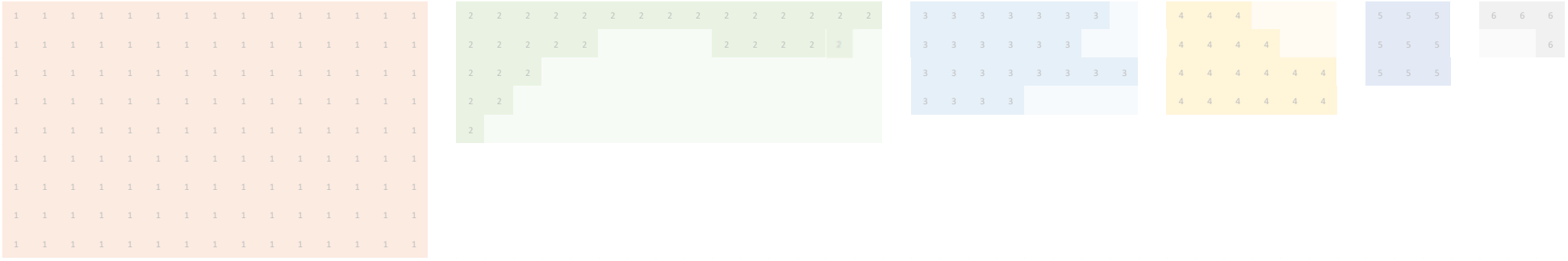
Patch packing

For each cluster:
    For each atlas:
        Push the cluster in "Used Space" (0° rotation first, 90° otherwise)
        If the push failed:
            Push the cluster into "Free Space" (0° rot. first, 90° otherwise)
            If the push failed:
                Split the cluster into 2 parts by its largest border
                For each resulting 2 parts:
                    If smaller than MinPatchSize:
                        Discard the patch
                    Else:
                        Put the part in the cluster priority list

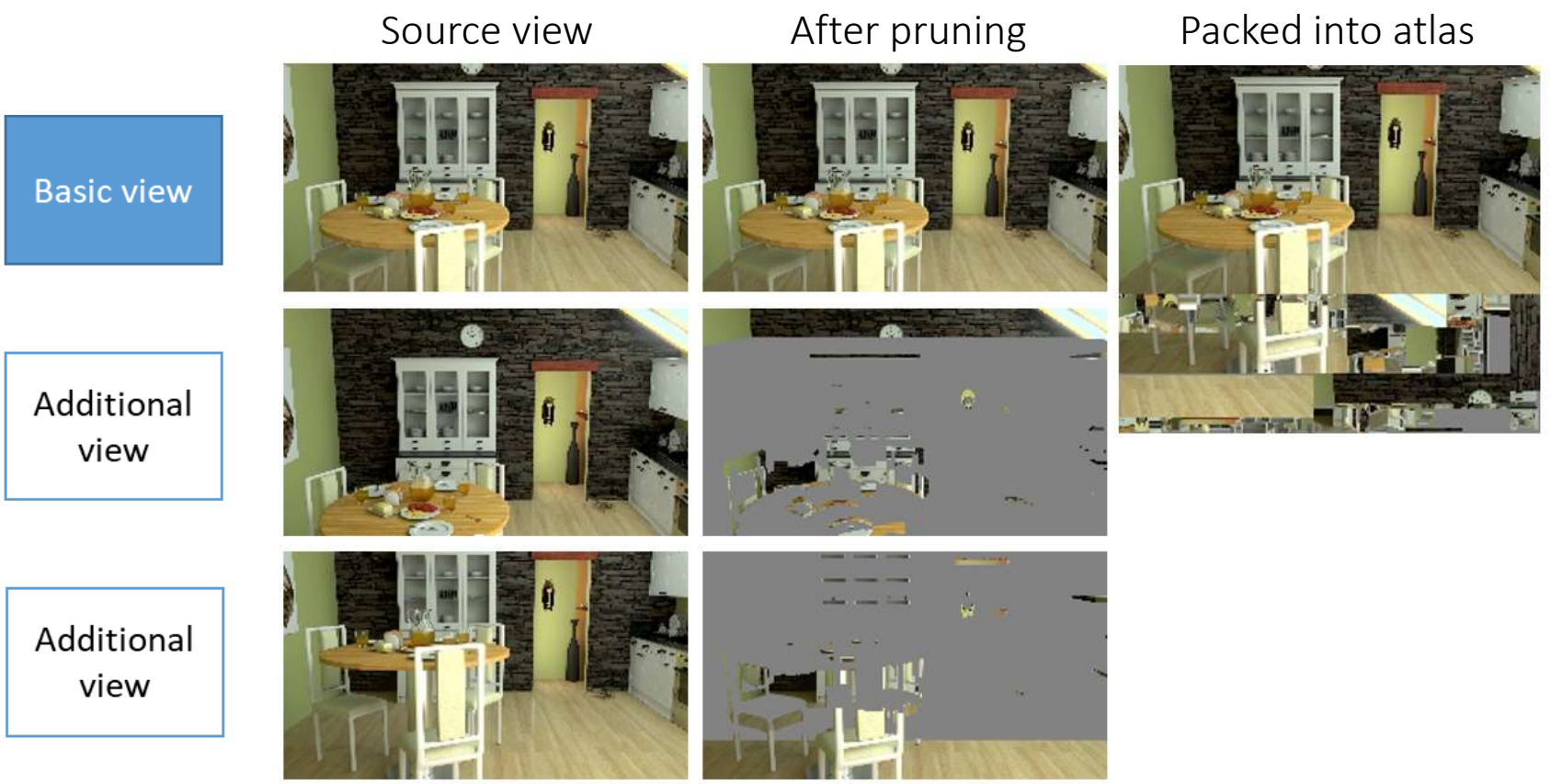
Atlas:



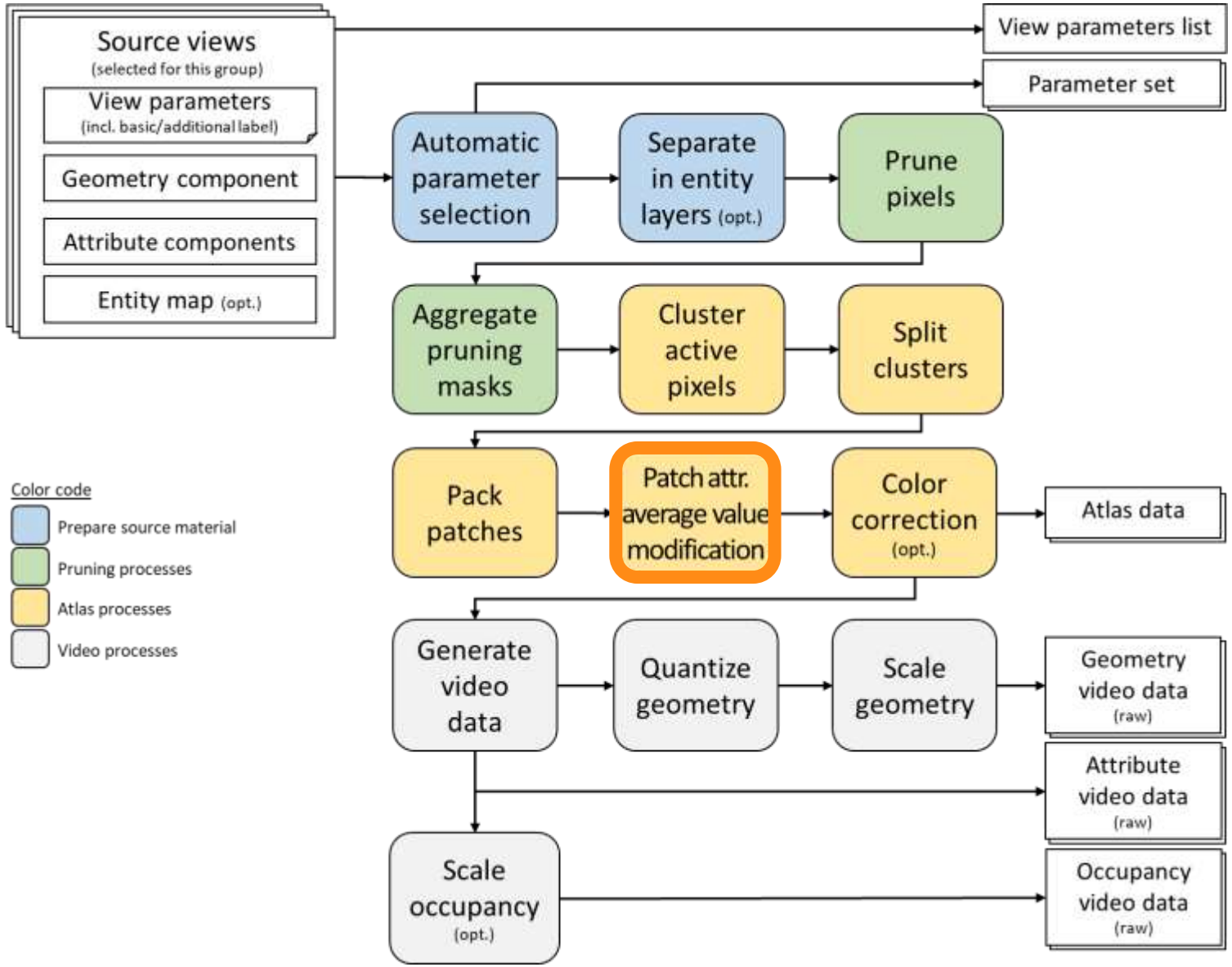
List of clusters sorted by area (1 square: 8 × 8 pixels):



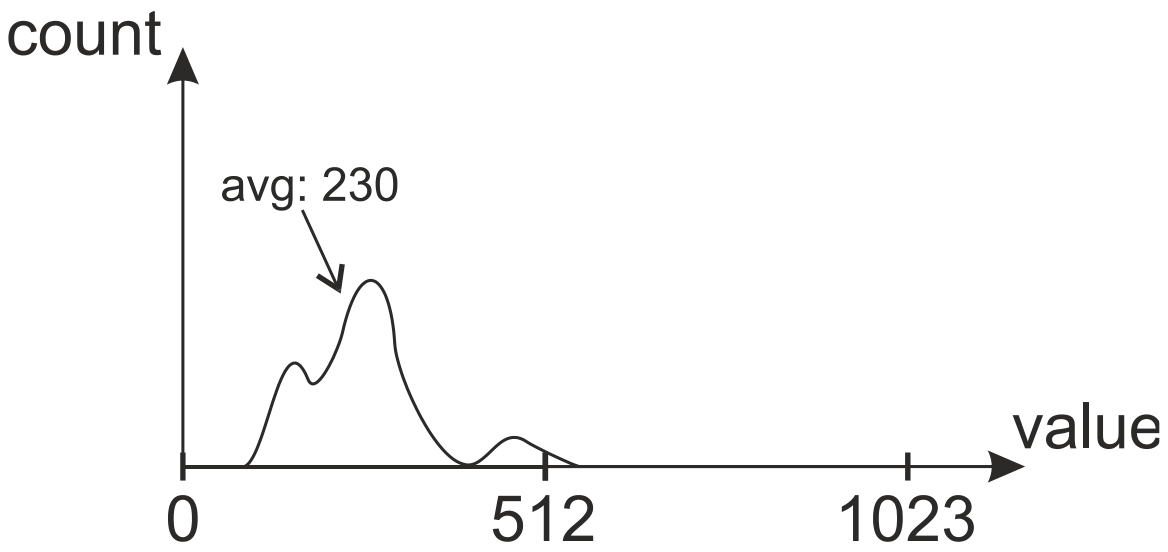
Patch packing



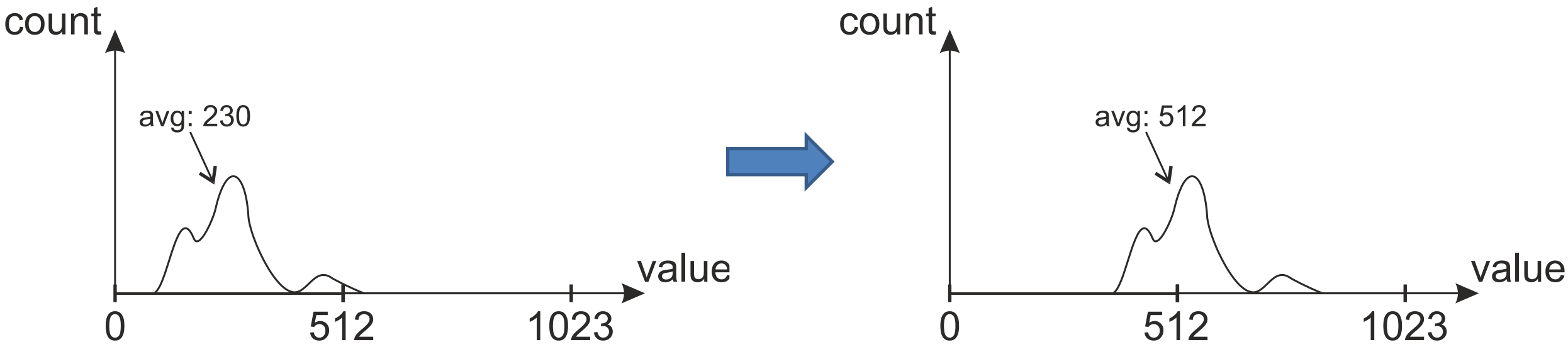
MIV – encoding of one group



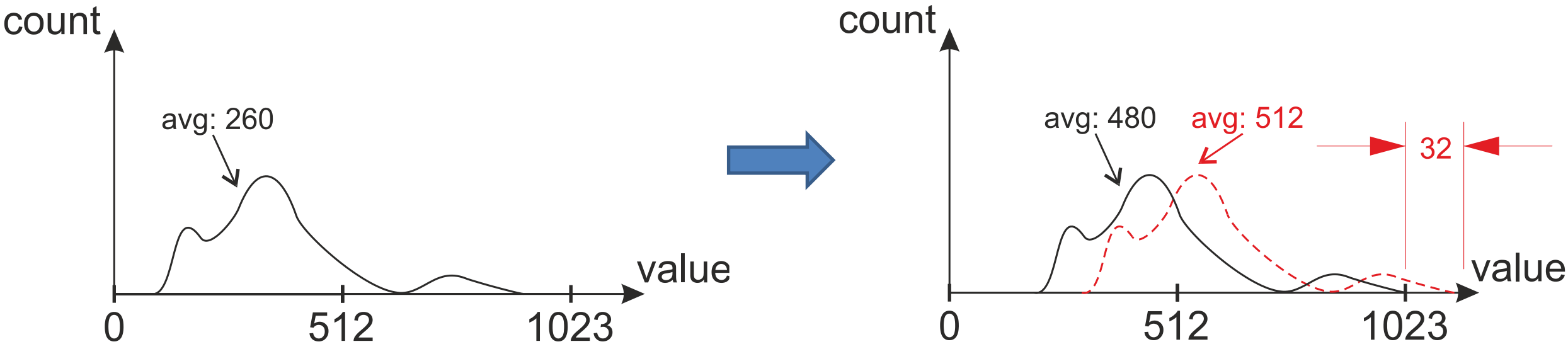
Patch attribute value modification



Patch attribute value modification

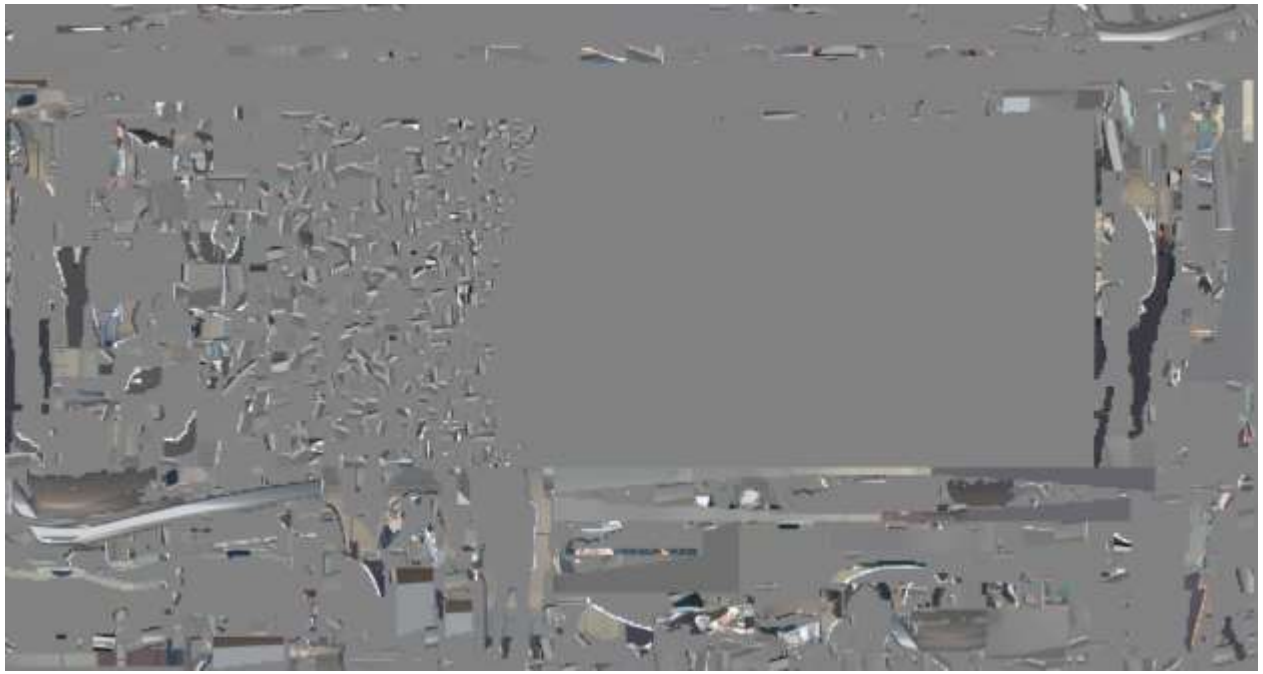


Patch attribute value modification



Patch attribute value modification

Without mean patch value modification



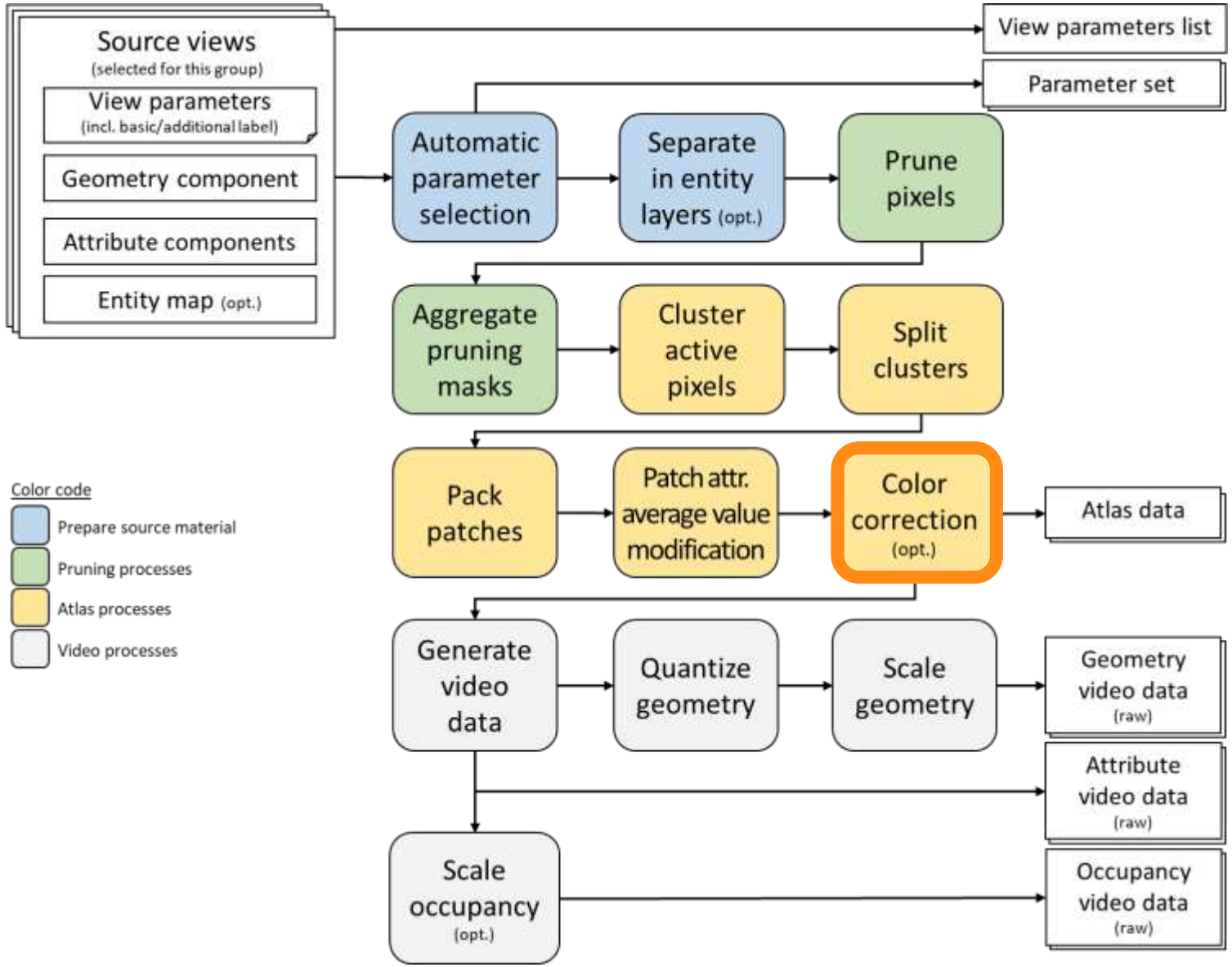
Mean patch value modification enabled

Patch attribute value modification

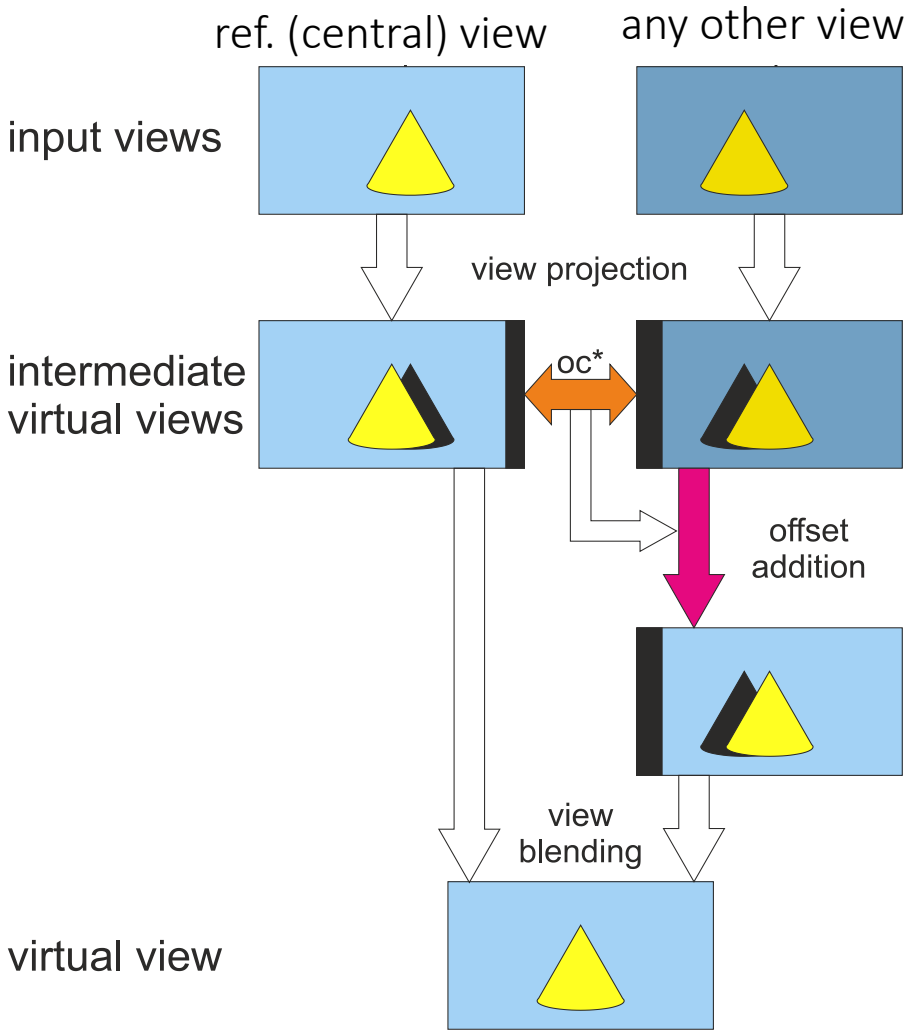
Without mean patch value modification

Mean patch value modification enabled

MIV – encoding of one group



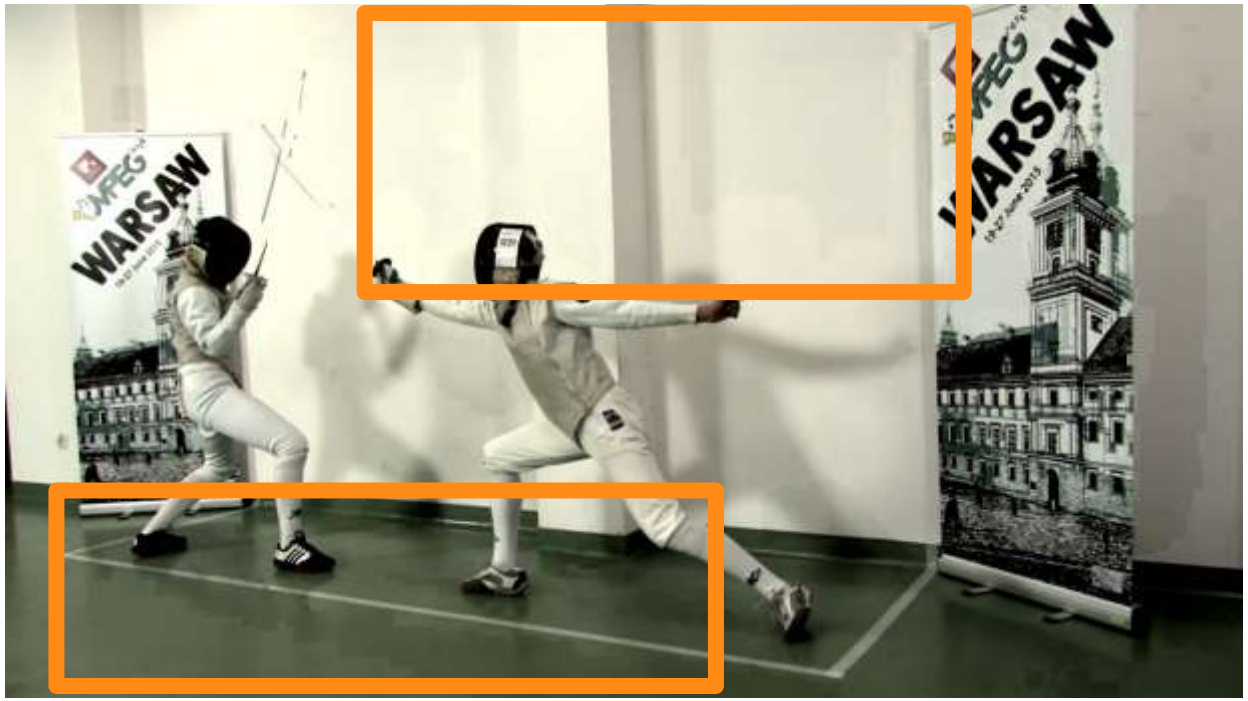
Color correction



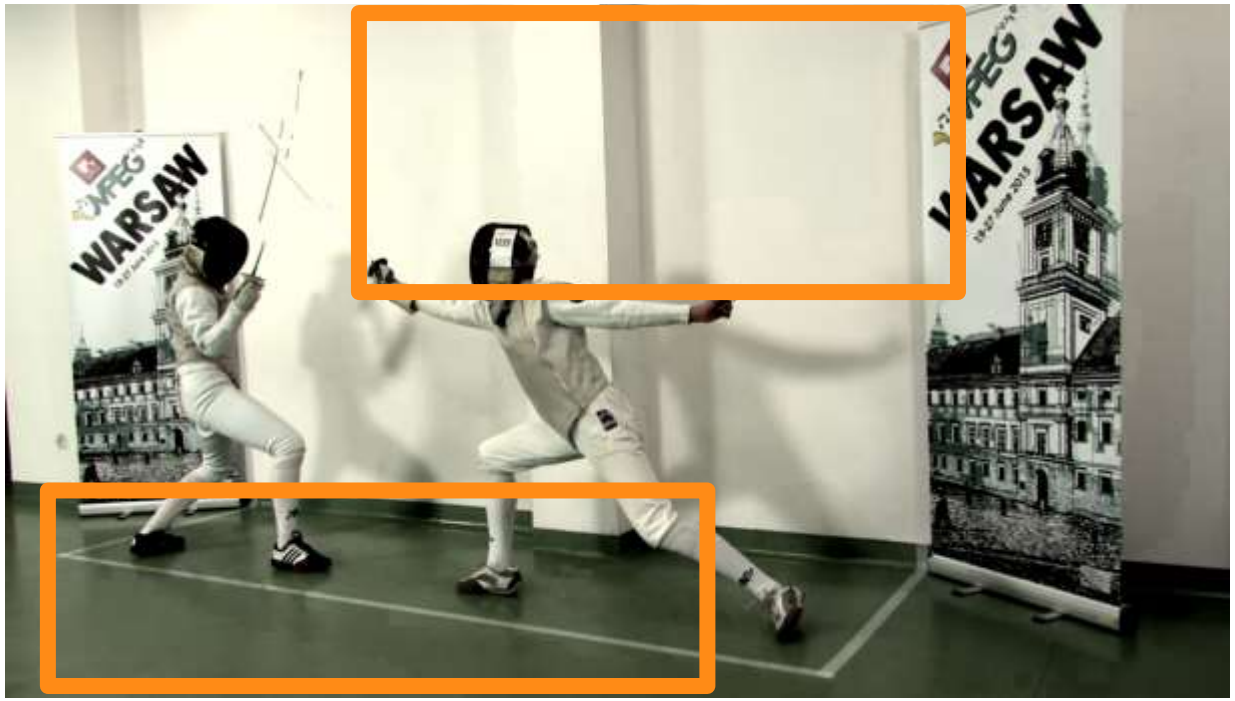
*oc – offset calculation

- Alignment of color characteristics of patches from different source views
- Offset averaged independently for each patch
- The ability to compensate for local changes in lighting

Color correction

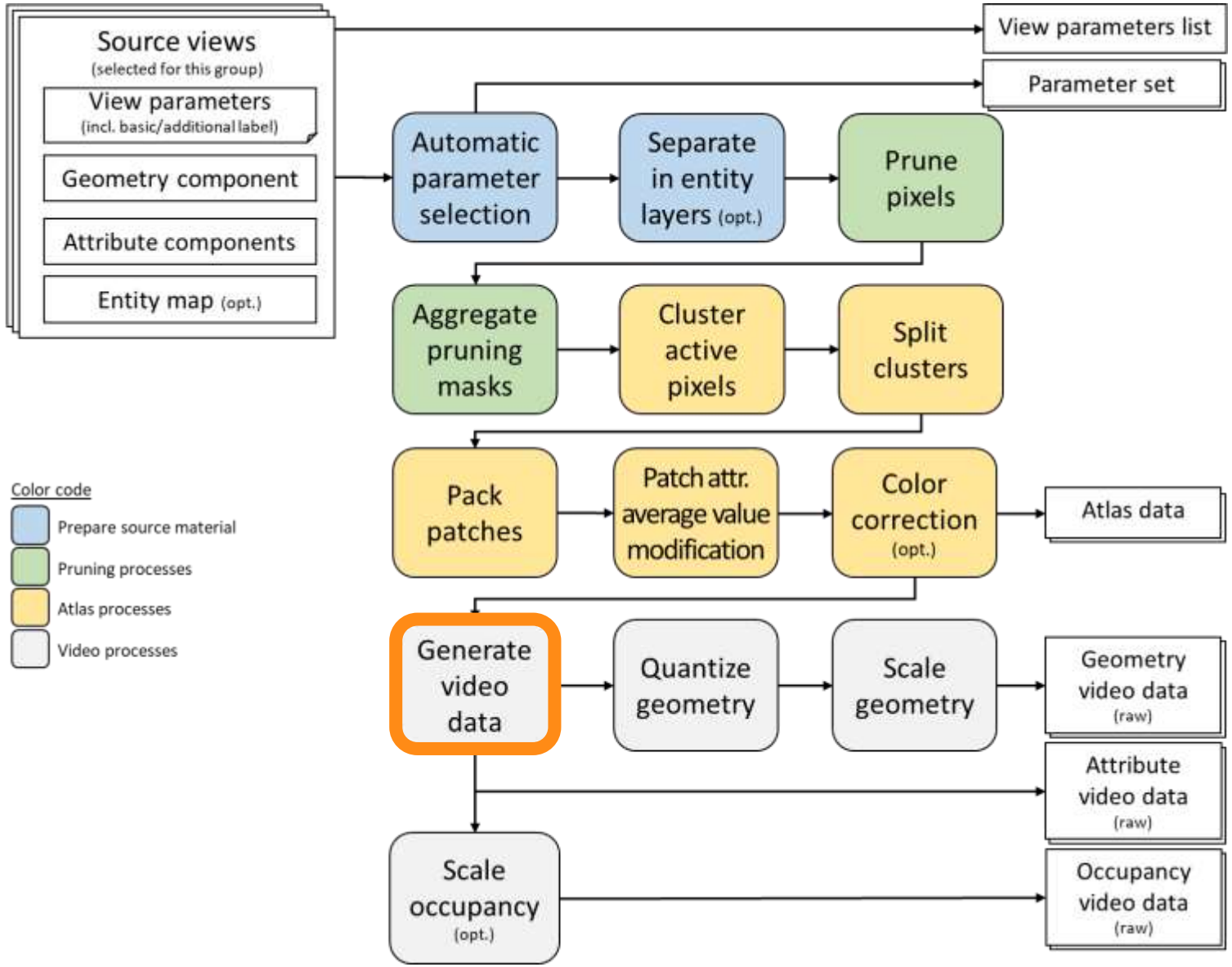


Without color correction



Color correction enabled

MIV – encoding of one group



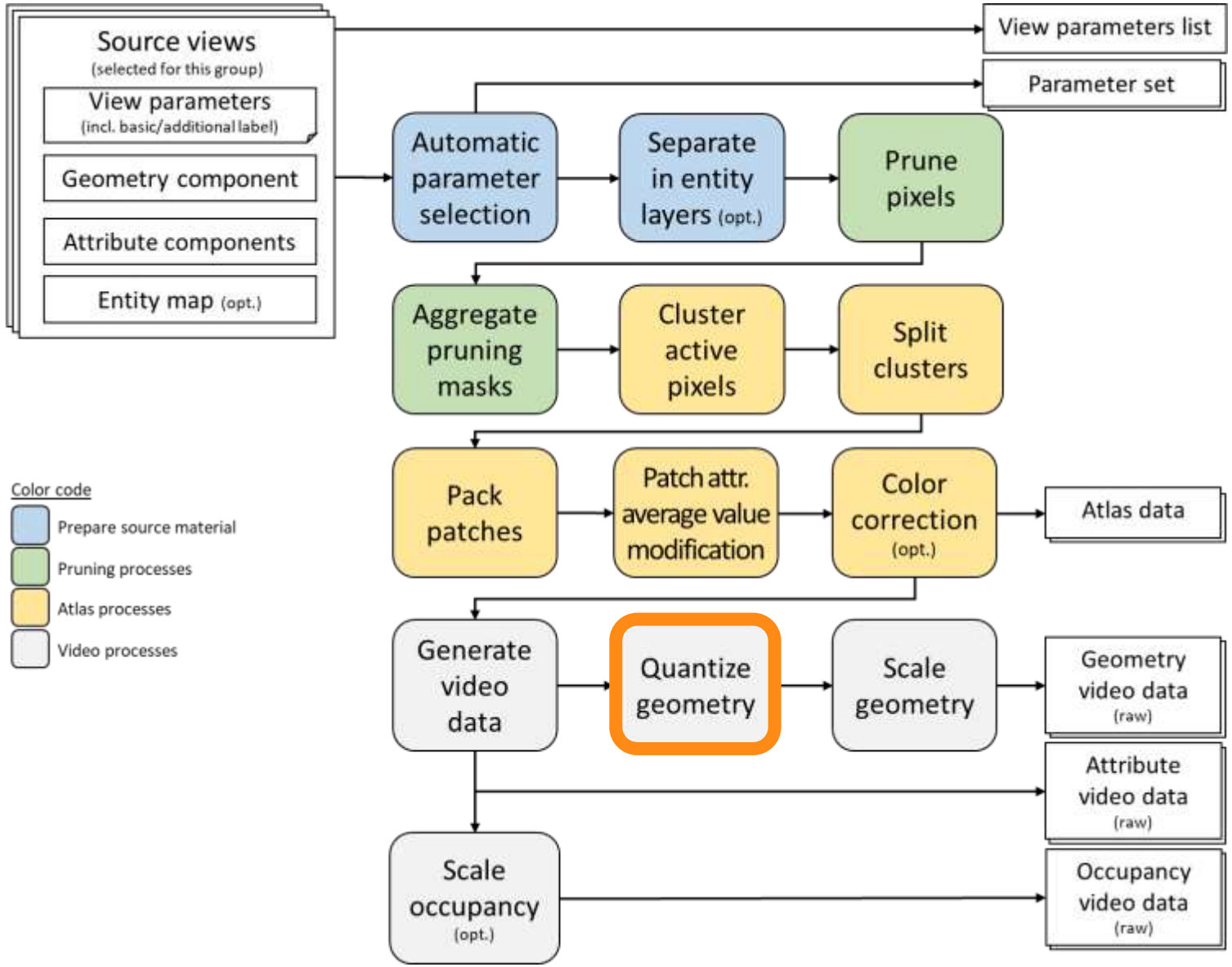
Video data generation: temporal redundancy removal

- Initially area within whole bounding box of a patch is packed into the atlas
- Temporal redundancy removal uses data from pixel pruning to send only data required in the current frame
- Occupancy of patch sent in its geometry

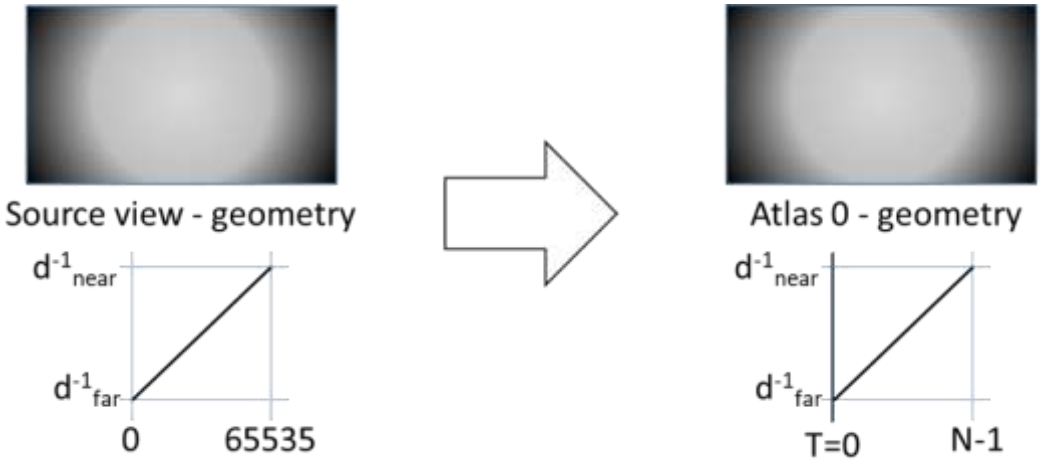
Without temporal redundancy removal

Temporal redundancy removal enabled

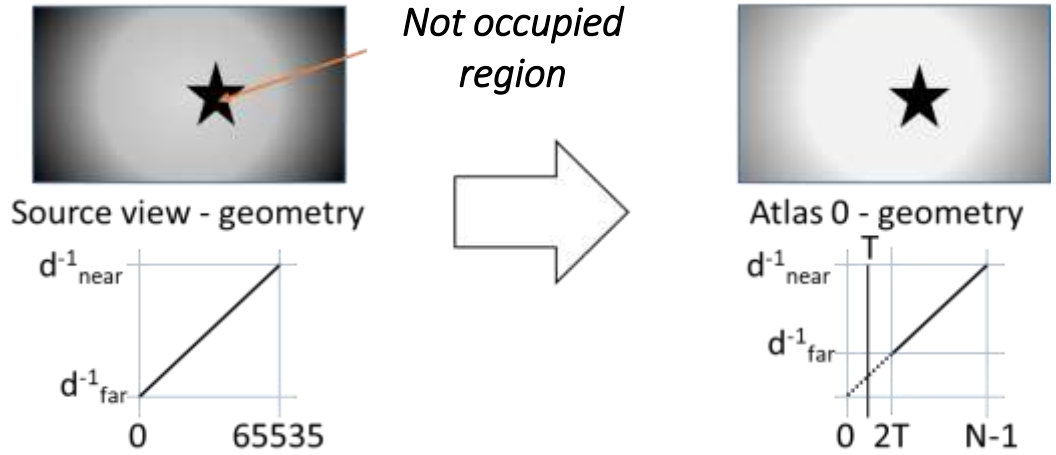
MIV – encoding of one group



Geometry quantization



base view



additional view

Filling the entire available dynamic range independently for each cluster:

$$d_{near}^{-1} = \max_{\substack{all\ pixels\ of \\ cluster\ in\ GOP}} d^{-1}(pixel)$$

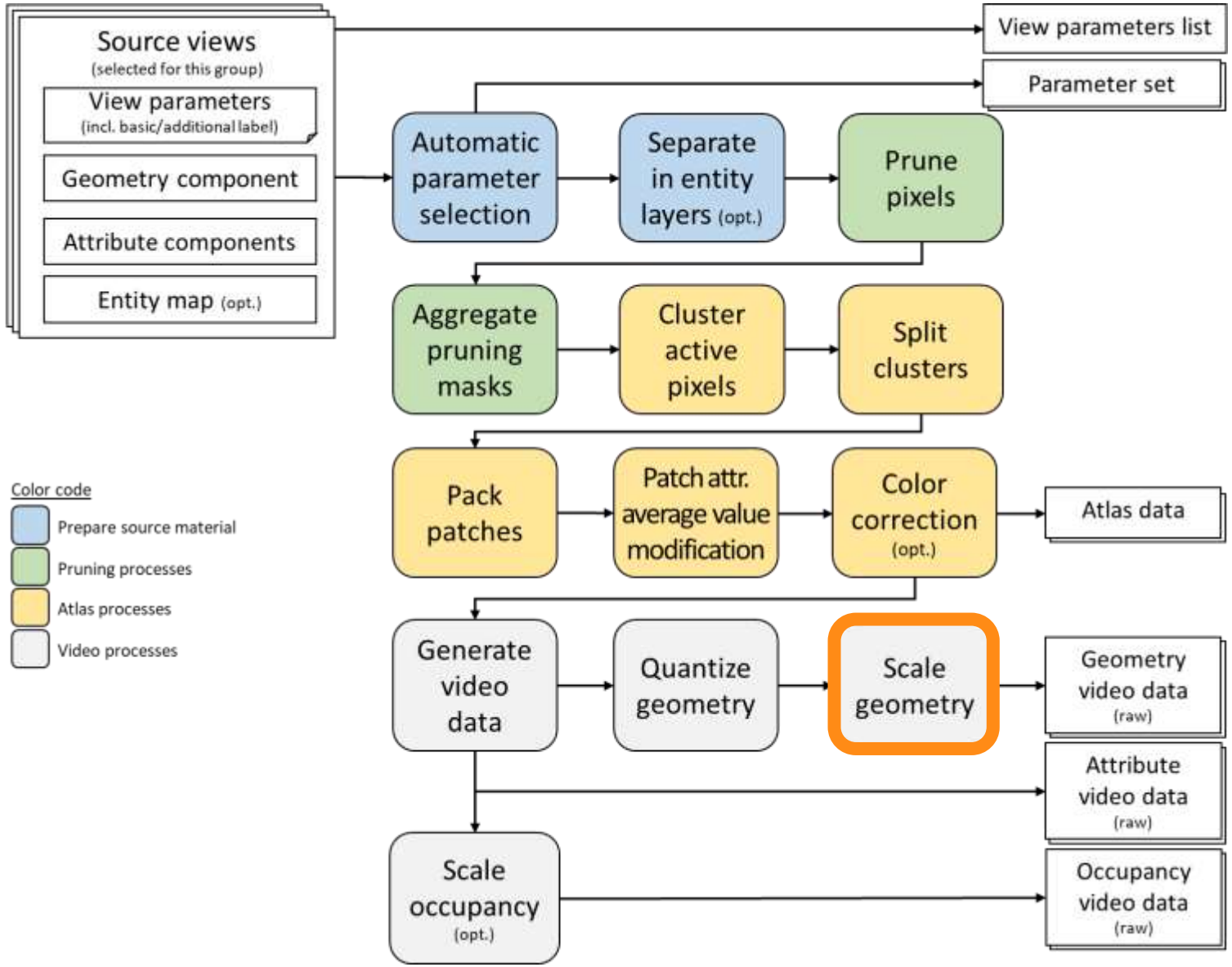
$$d_{far}^{-1} = \min_{\substack{all\ pixels\ of \\ cluster\ in\ GOP}} d^{-1}(pixel)$$

Geometry quantization

Before scaling

After scaling

MIV – encoding of one group



Geometry downscaling

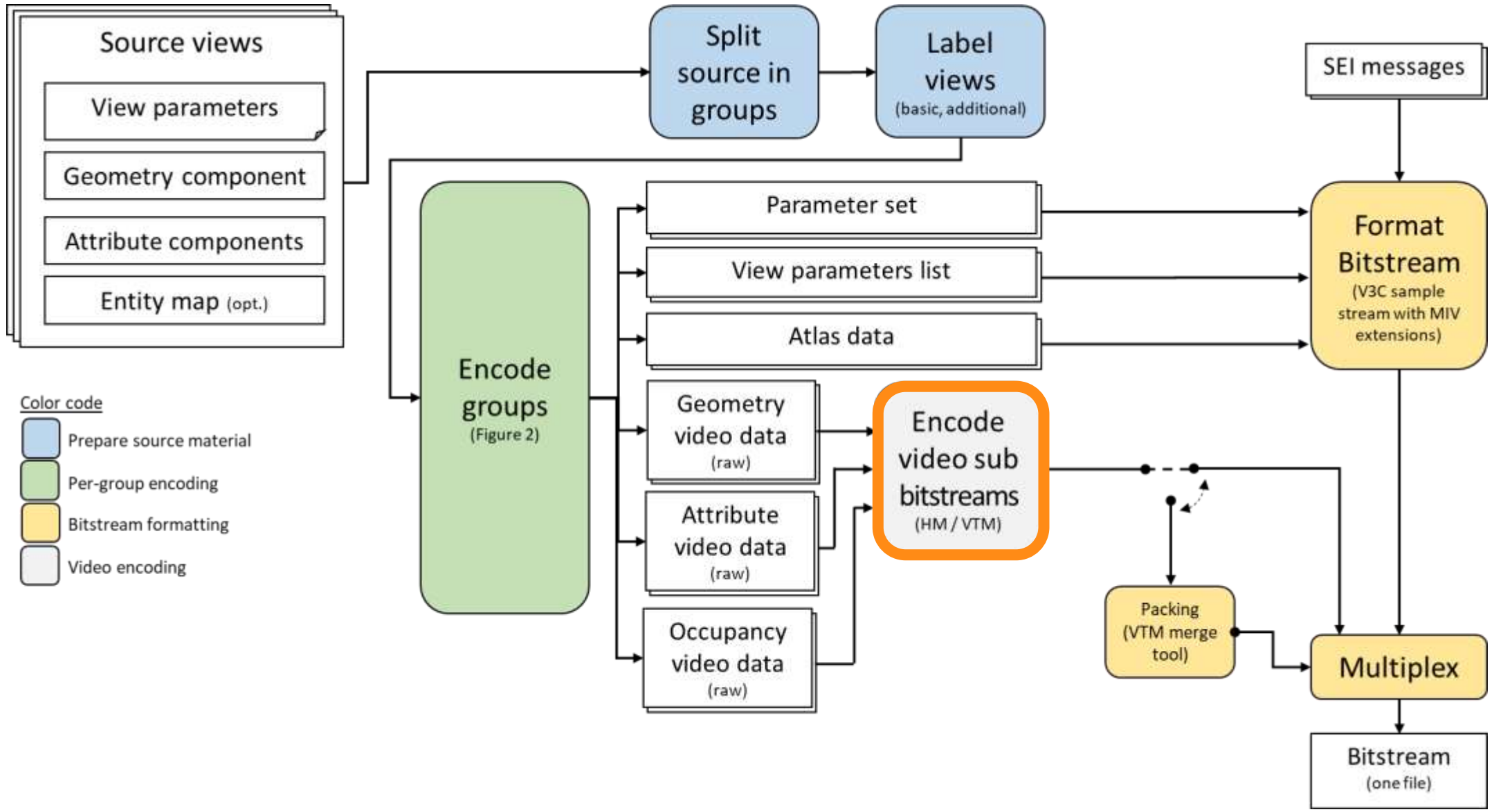
Attribute atlas: 1920 × 4640
Geometry atlas: 1920 × 4640

Geometry downscaling

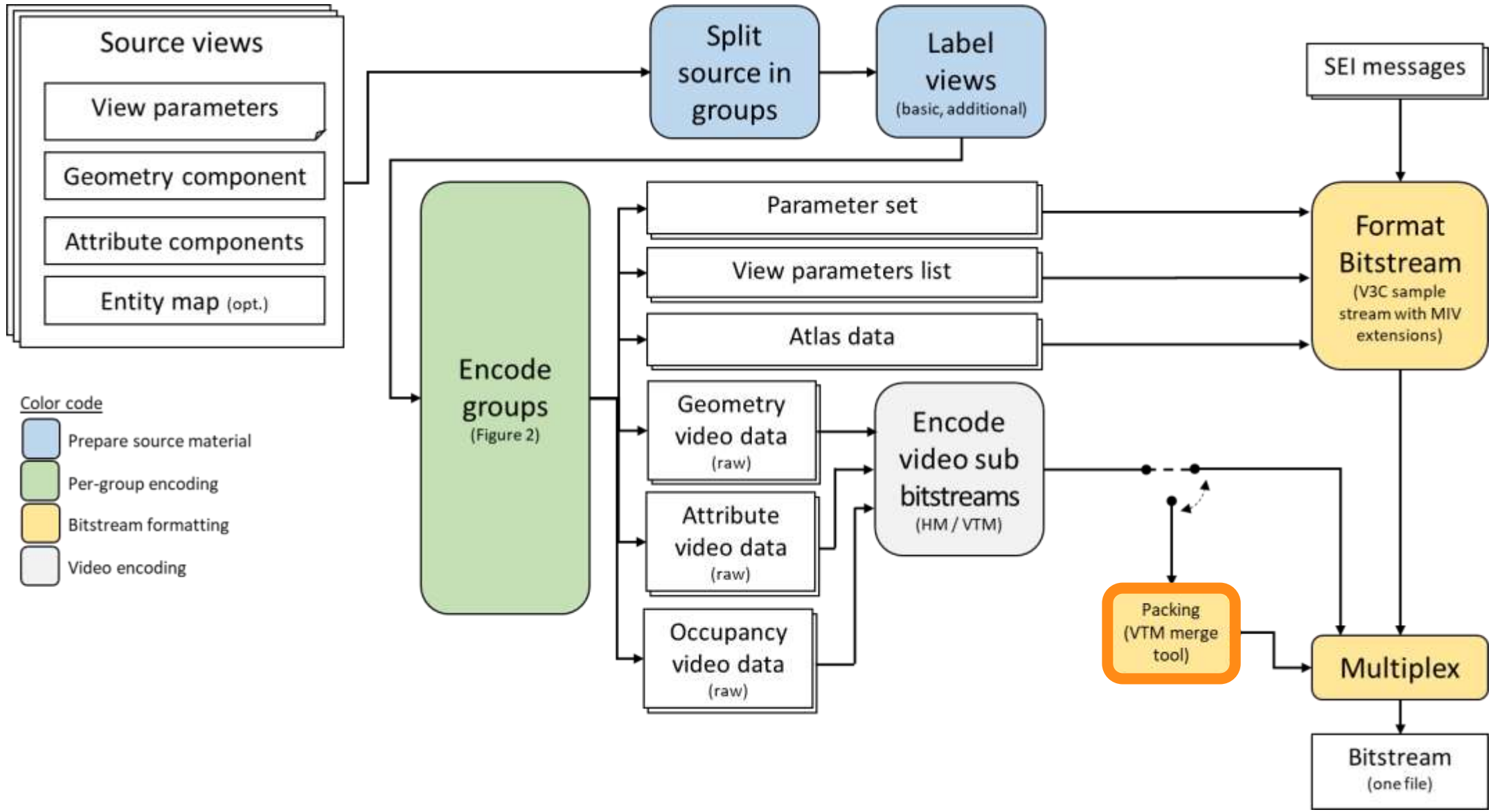
Attribute atlas: 1920 × 4640
Geometry atlas: 960 × 2320

- Less points
- Smaller bitstream
- Smaller QP for texture with the same bitrate

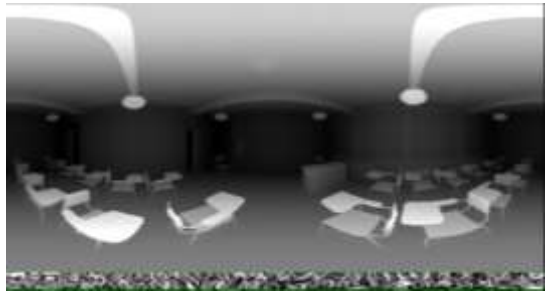
MIV encoder

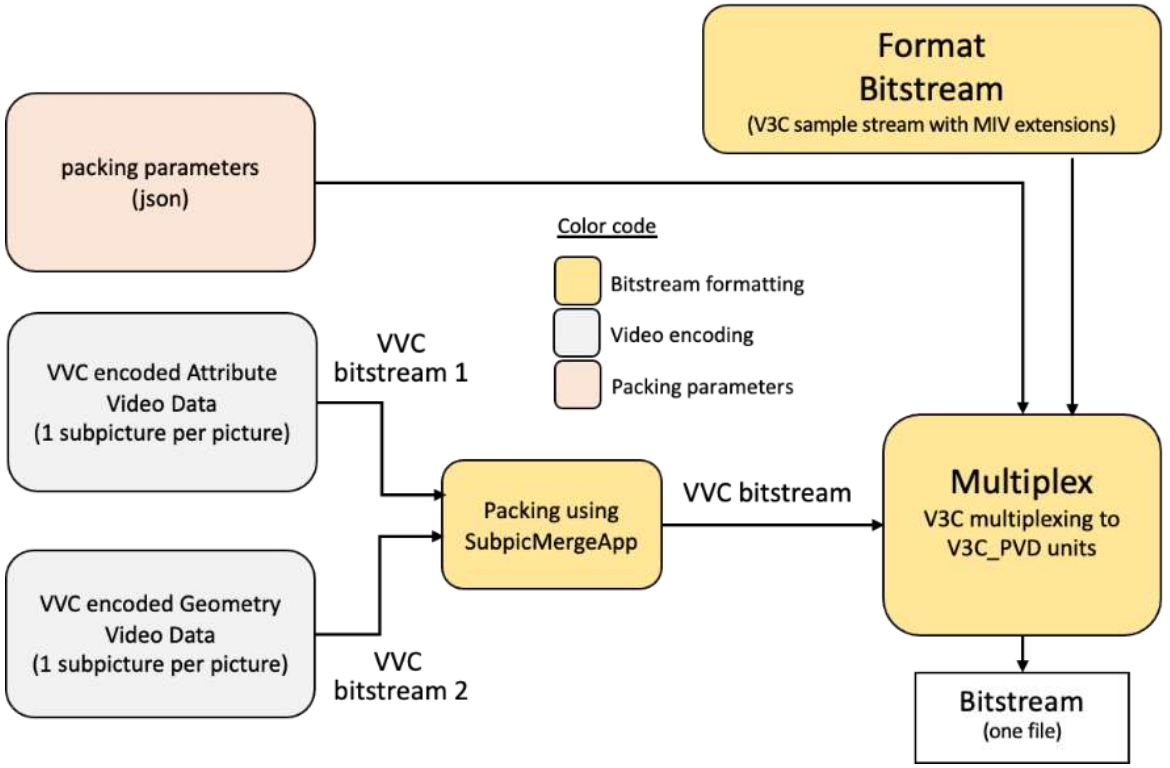

Source views
View parameters
Geometry component
Attribute components
Entity map (opt.)
Split source in groups
Label views
(basic, additional)
SEI messages
Encode groups
(Figure 2)
Parameter set
View parameters list
Atlas data
Format Bitstream
(V3C sample stream with MIV extensions)
Geometry video data
(raw)
Attribute video data
(raw)
Occupancy video data
(raw)
Encode video sub bitstreams
(HM / VTM)
Packing
(VTM merge tool)
Multiplex
Bitstream
(one file)
Color code
Prepare source material
Per-group encoding
Bitstream formatting
Video encoding

Koder MIV

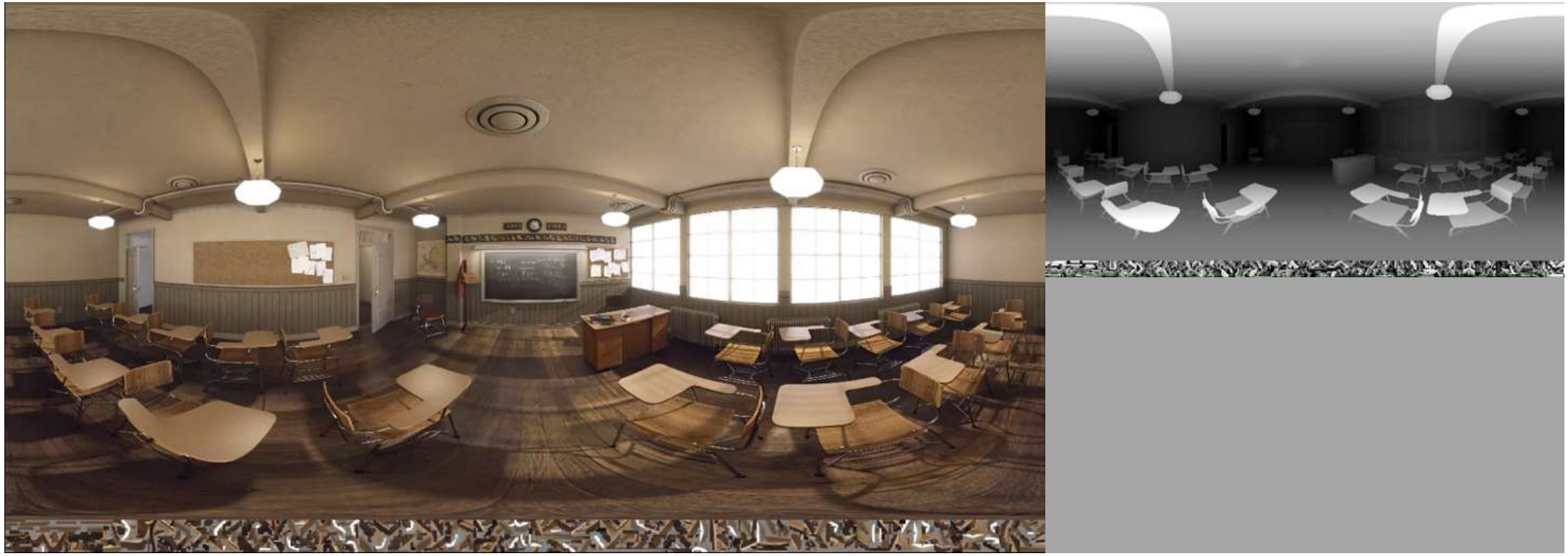

Source views
View parameters
Geometry component
Attribute components
Entity map (opt.)
Split source in groups
Label views
(basic, additional)
SEI messages
Encode groups
(Figure 2)
Parameter set
View parameters list
Atlas data
Format Bitstream
(V3C sample stream with MIV extensions)
Geometry video data
(raw)
Attribute video data
(raw)
Occupancy video data
(raw)
Encode video sub bitstreams
(HM / VTM)
Packing
(VTM merge tool)
Multiplex
Bitstream
(one file)
Color code
Prepare source material
Per-group encoding
Bitstream formatting
Video encoding

Video sub-bitstream packing

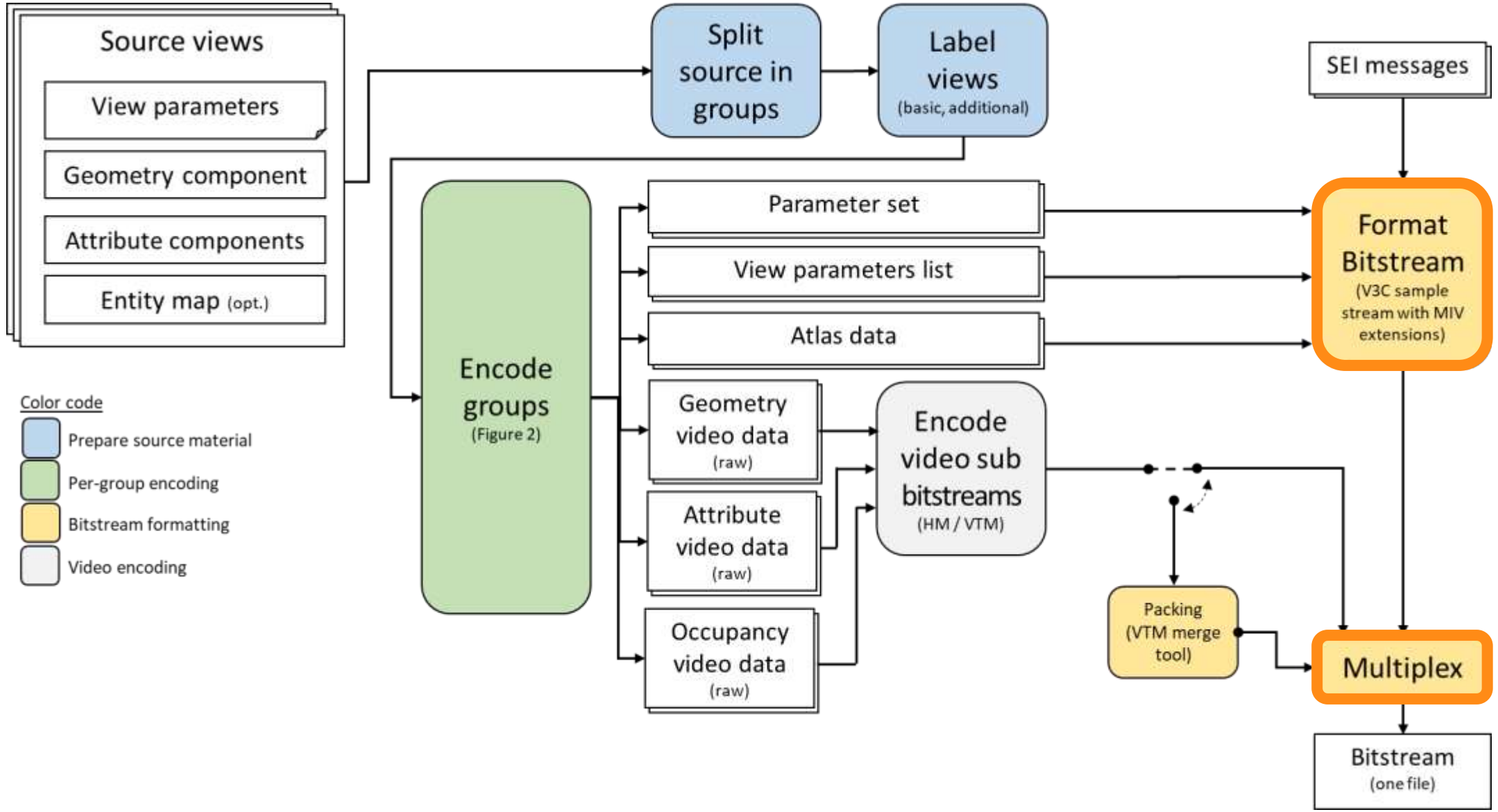




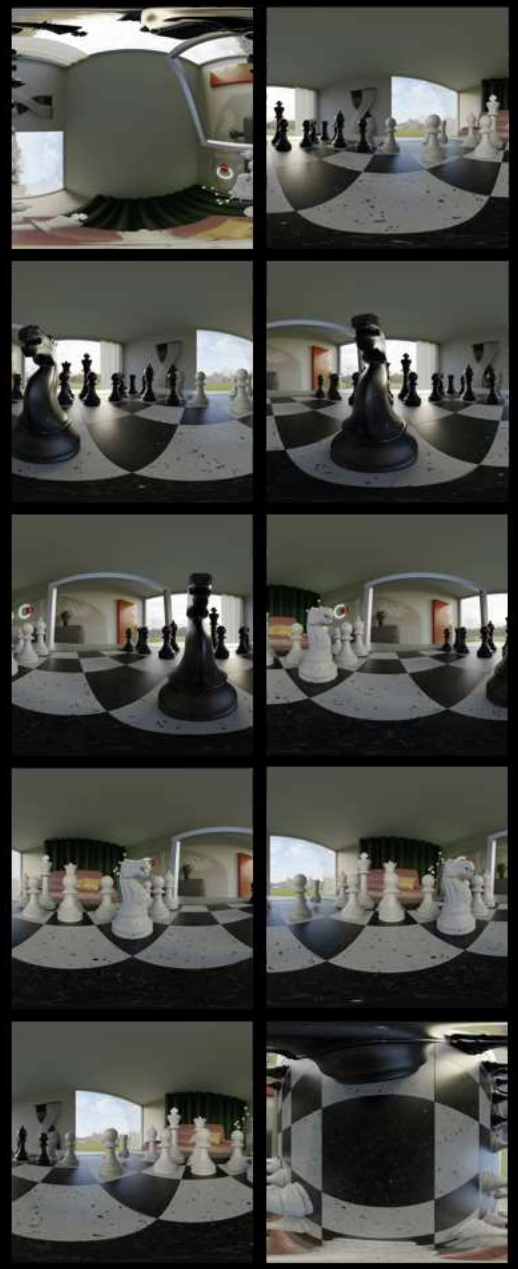
Video sub-bitstream packing



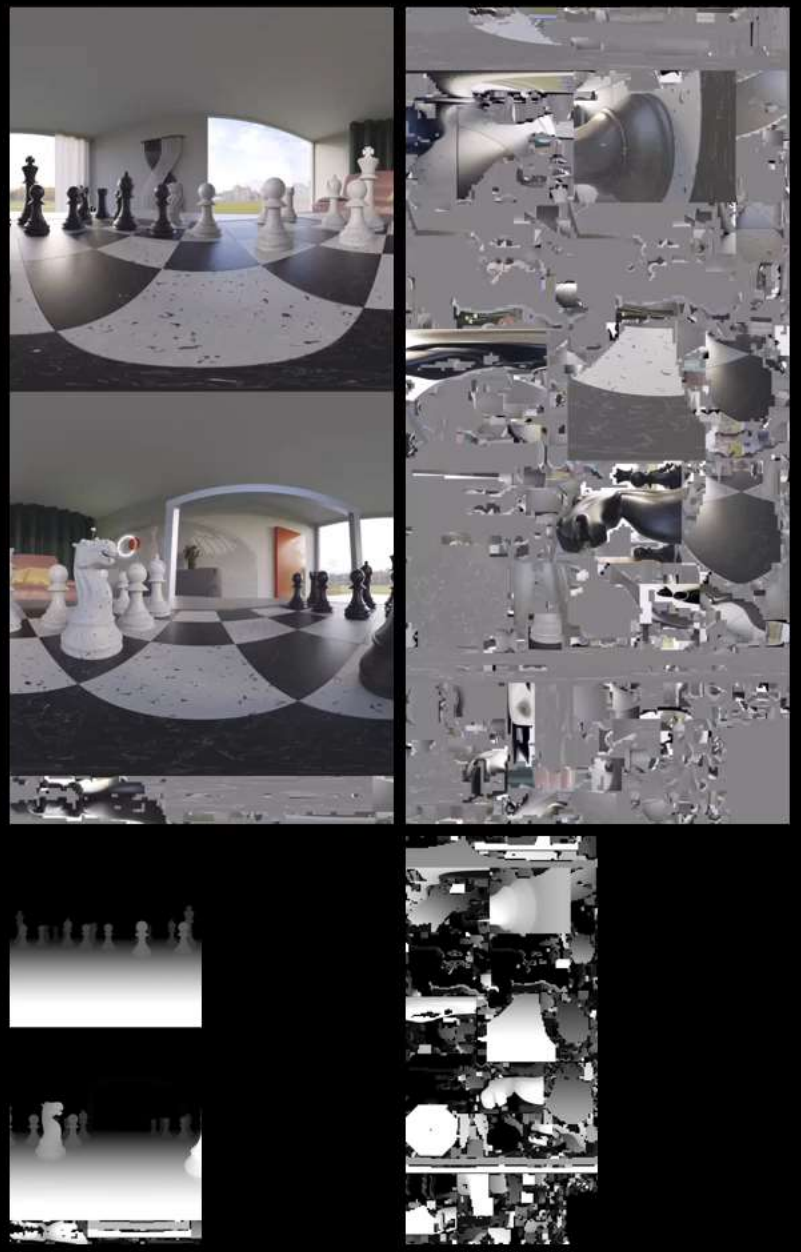
MIV – bitstream generation and multiplexing



input views
attributes + geometry + cam. params



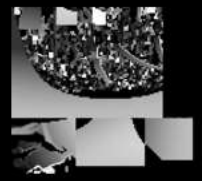
atlases
attributes + geometry + metadata sub-bitstream



viewport

input views
attributes + geometry + cam. params

atlases
attributes + geometry + metadata sub-bitstream



viewport

MIV tutorial

- MIV website: https://mpeg-miv.org
- Public software:
  - Test model: https://gitlab.com/mpeg-i-visual/tmiv
  - IV-PSNR: https://gitlab.com/mpeg-i-visual/ivpsnr
  - IVDE: https://gitlab.com/mpeg-i-visual/ivde
  - RVS: https://gitlab.com/mpeg-i-visual/rvs
- Demo videos:
  - Freeport player: https://youtu.be/UeT_Xm1jBGs
  - Multi-plane images: https://youtu.be/DPMMzj2I6Yw
  - Real-time RVS: https://lisaserver.ulb.ac.be/rvs/
- Test material (14+ sequences) available on request

Q&A

End of tutorial