The MPEG Immersive video coding standard

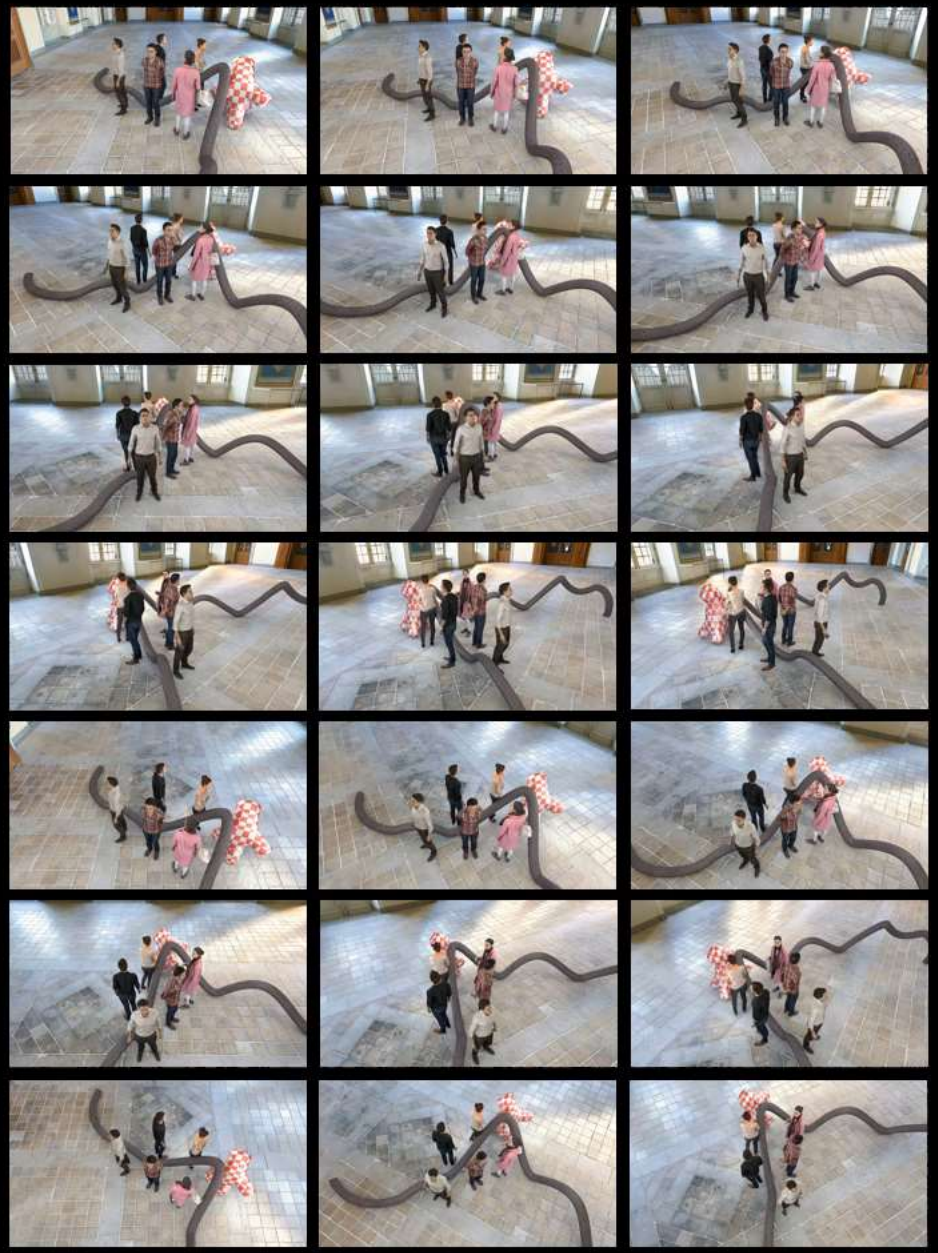
Part II – Source video acquisition and encoder overview

Dawid Mieloch, PhD

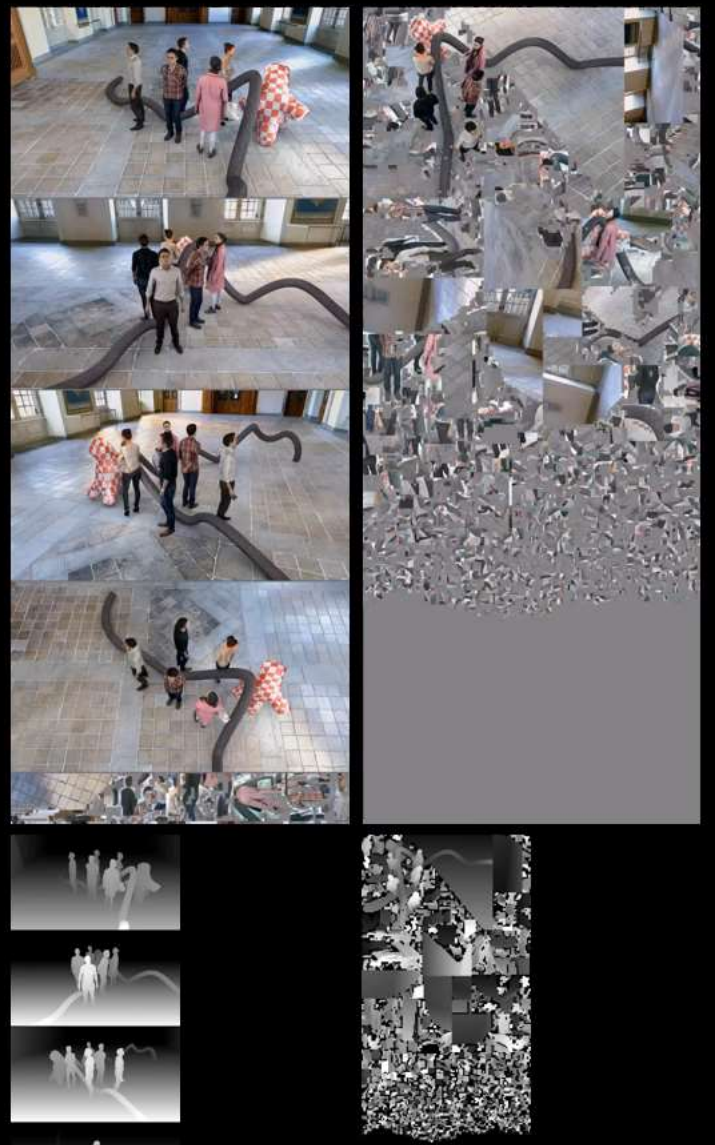
Institute of Multimedia Telecommunications
Poznań University of Technology

itm

input views
attributes + geometry + cam. params

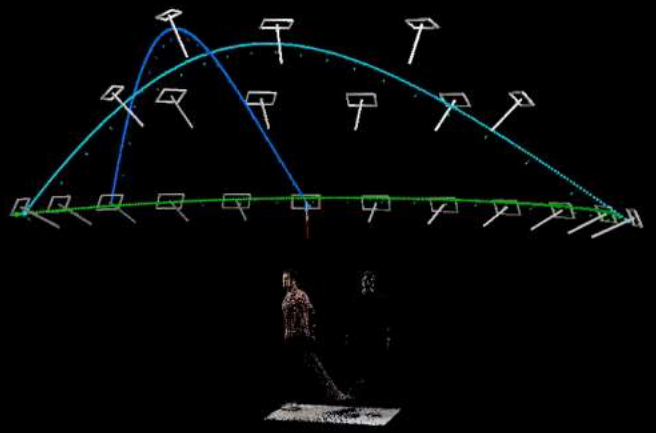


atlases
attributes + geometry + metadata sub-bitstream

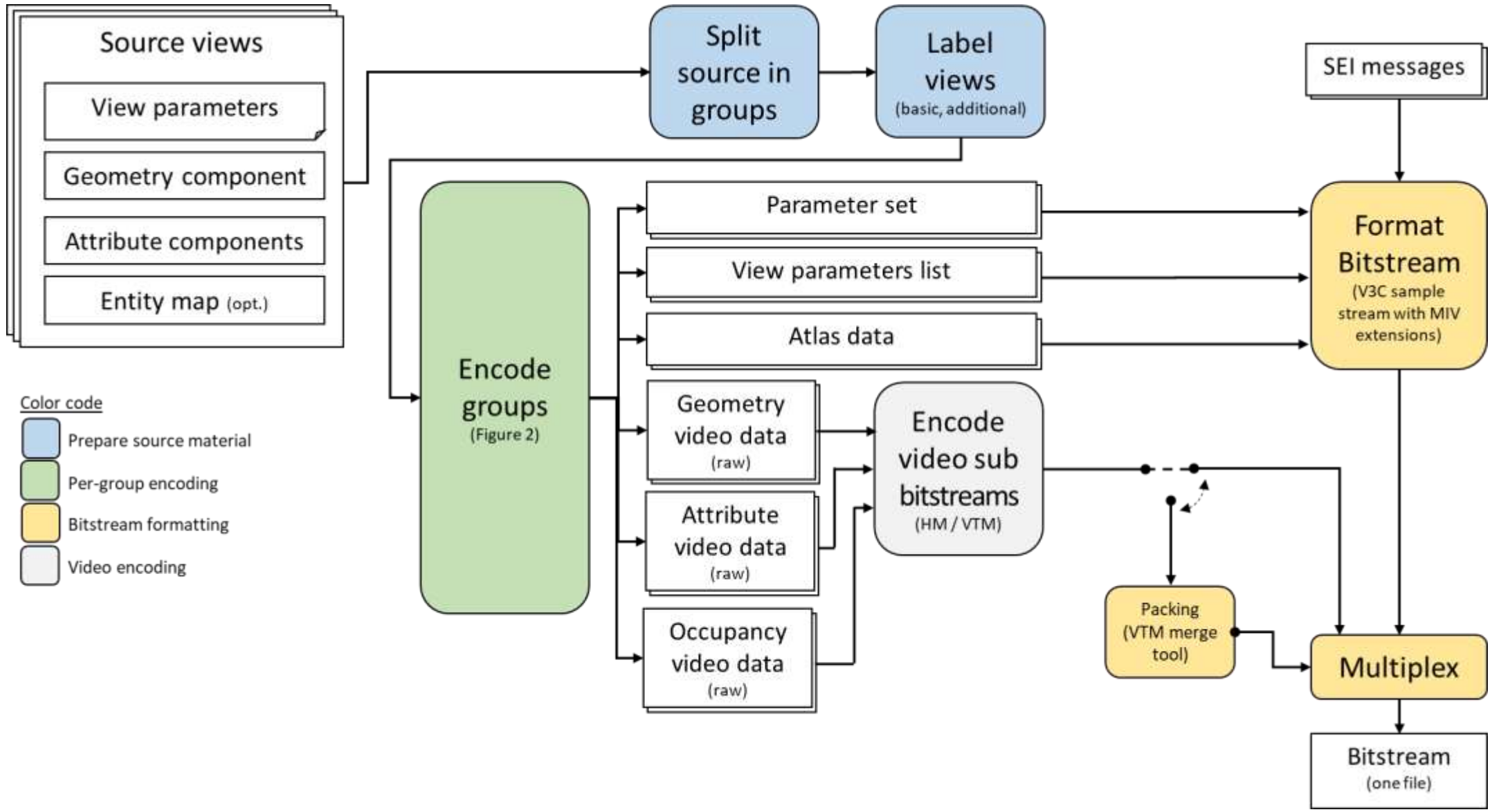


viewport

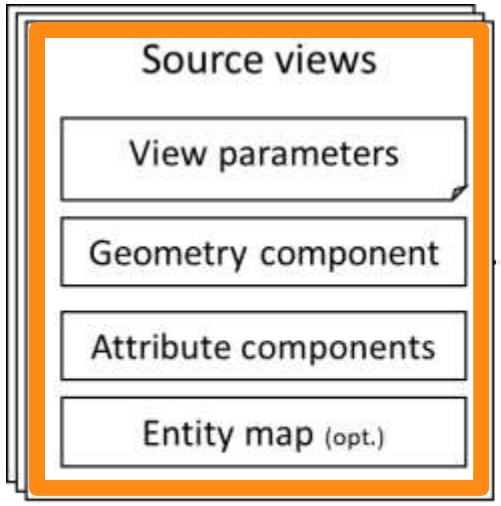
trajectory of the viewer



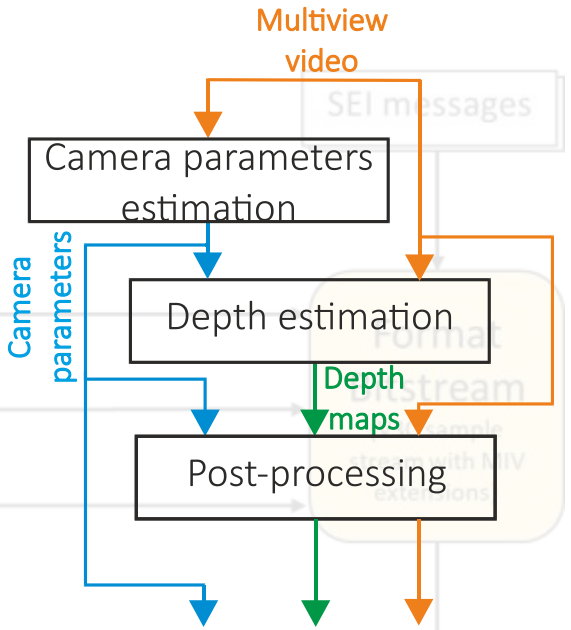
MIV encoder


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

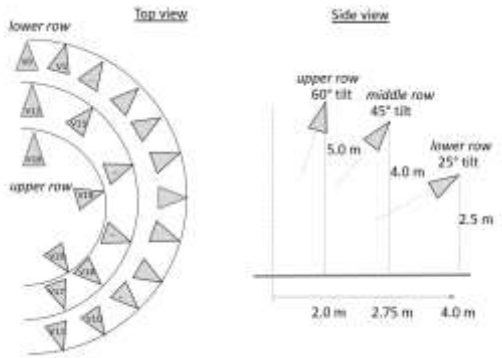


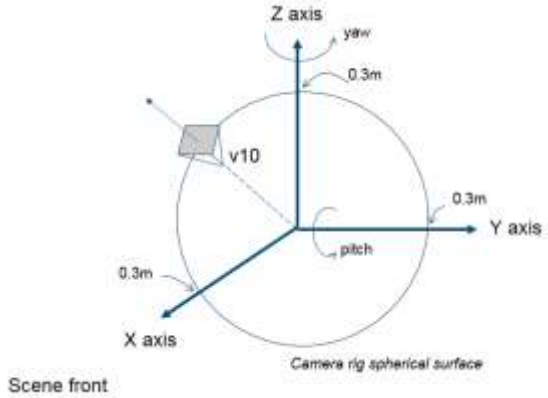
- Acquisition of multiview sequence
- Estimation of view/camera parameters
- Estimation of geometry of acquired scene
- Post-processing of multiview sequence
- Validation of the sequence quality



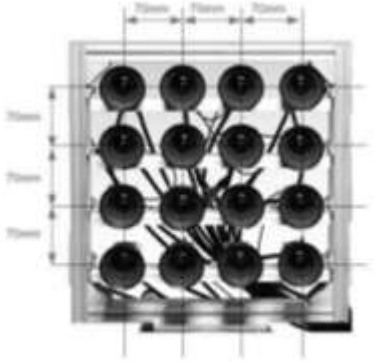
Multiview sequences for immersive video

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





Scene front

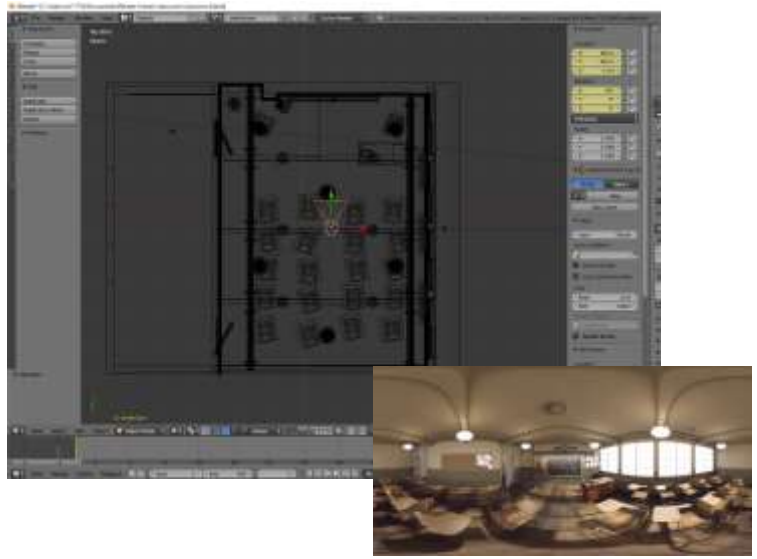


Immersive video acquisition

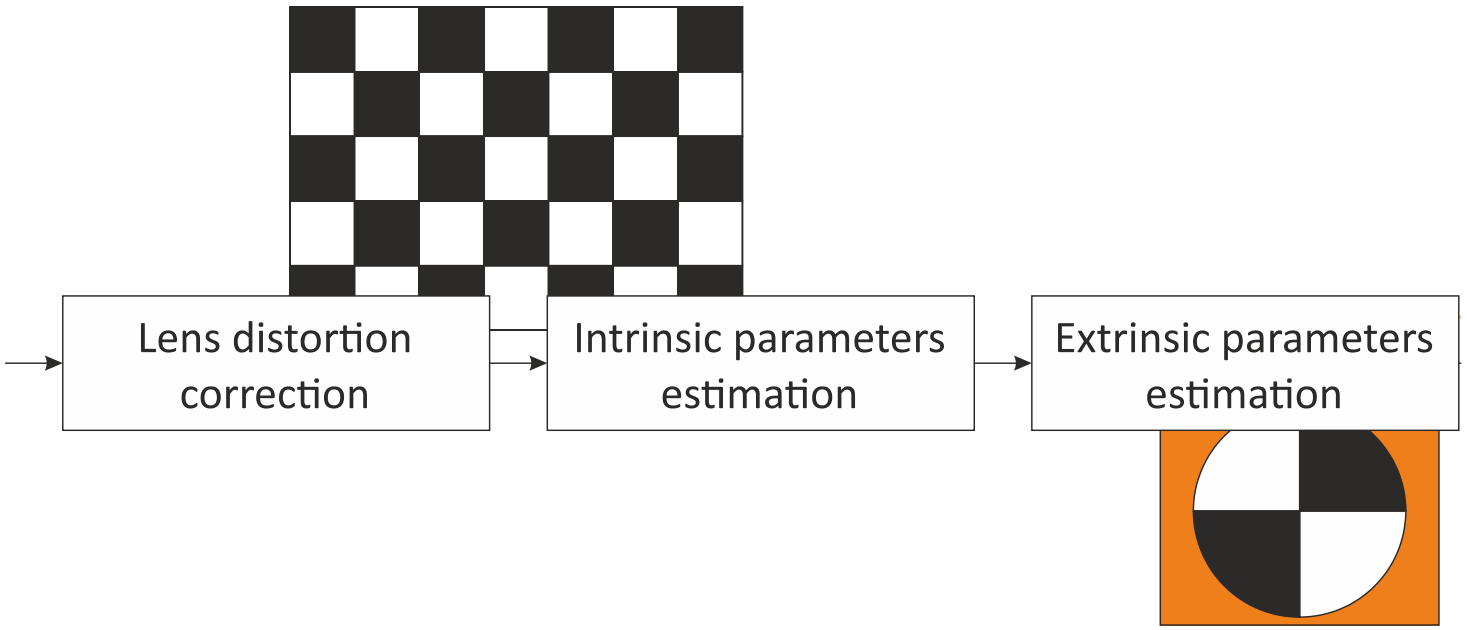
Natural sequences

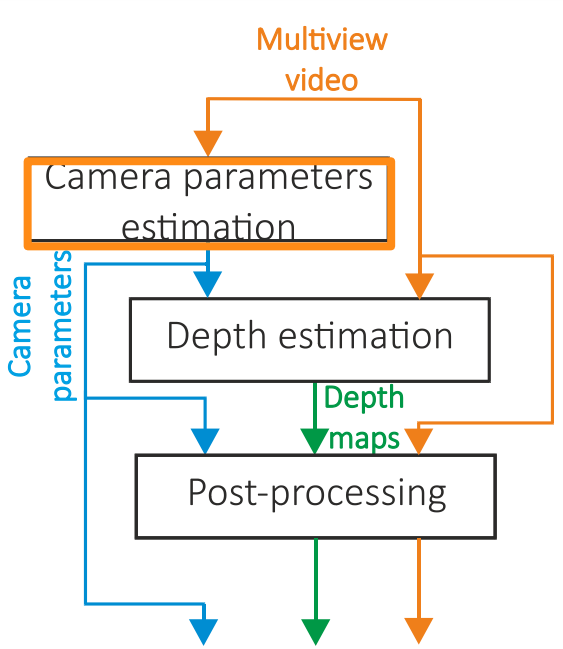
Mixed natural & CGI sequences

CGI sequences

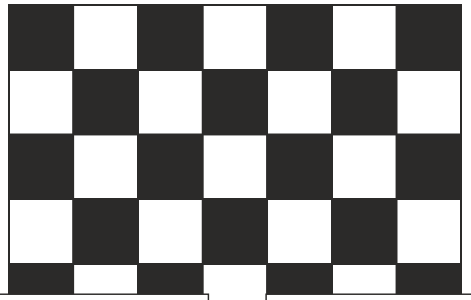


Camera parameters estimation





Camera calibration

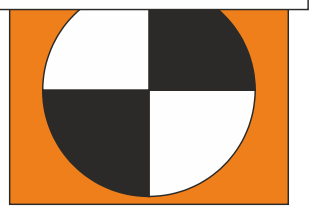


Lens distortion correction → Intrinsic parameters estimation

- 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

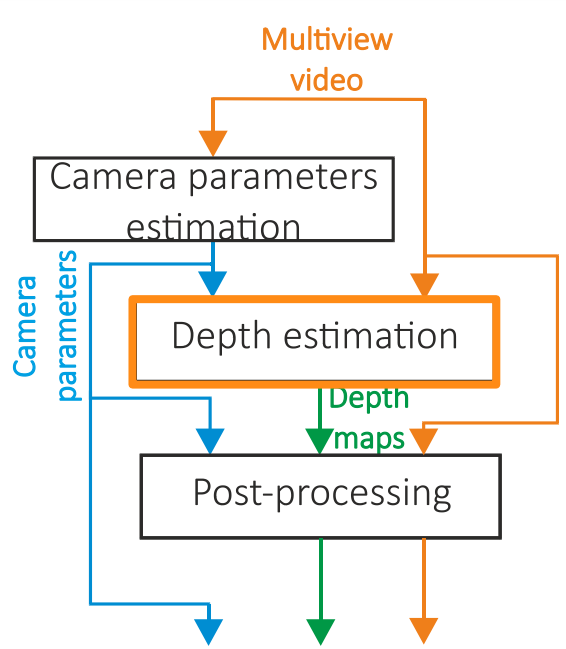
Lens distortion correction → Intrinsic parameters estimation → Extrinsic parameters estimation



- 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

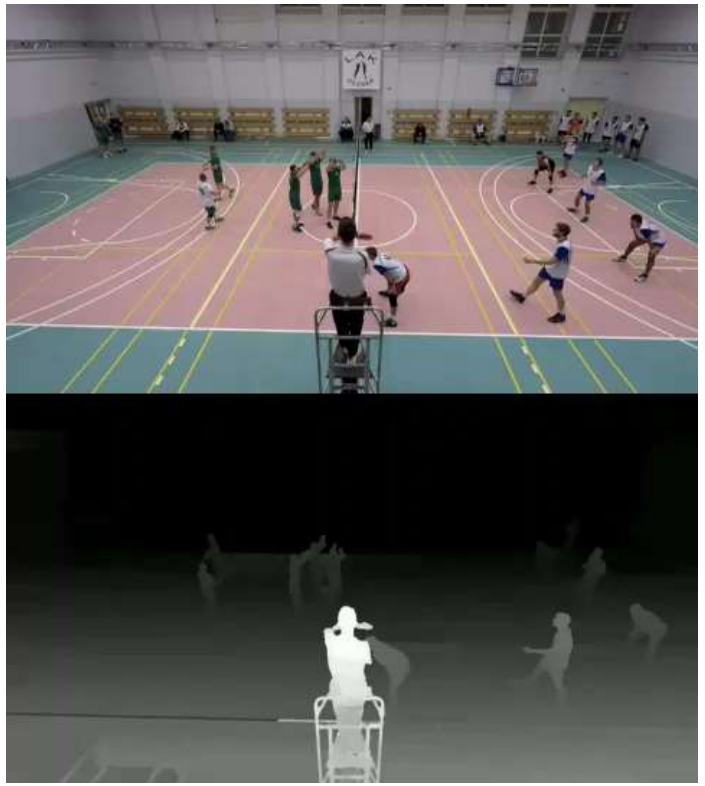
Depth estimation

- 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



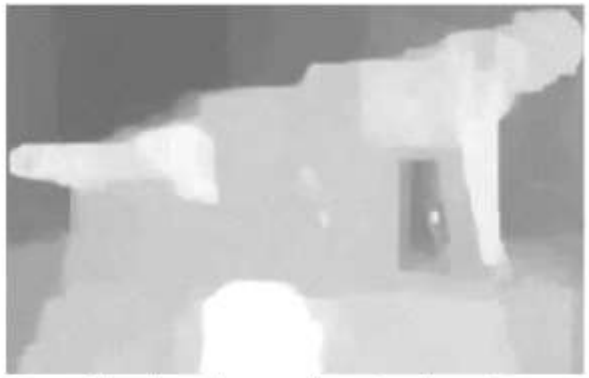
Immersive Video Depth Estimation - IVDE

- 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

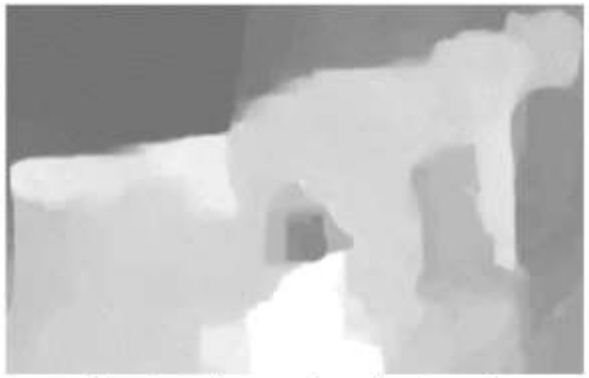


Depth refinement – motivation

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



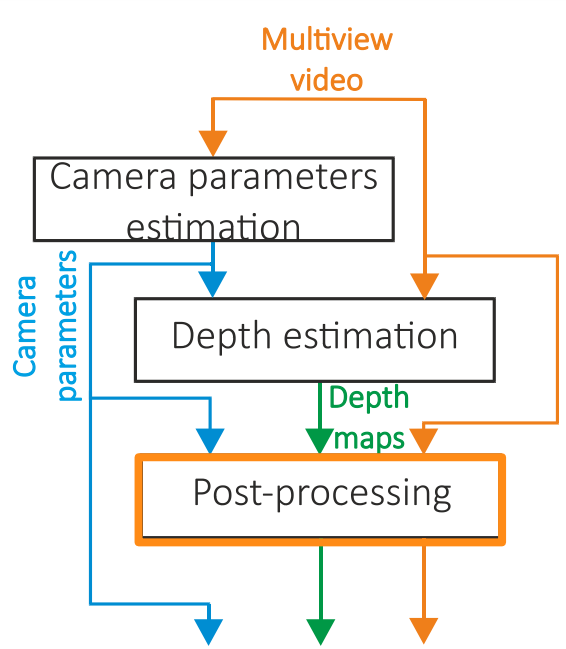
The depth map for the view 0



The depth map for the view 2

The reference view 1

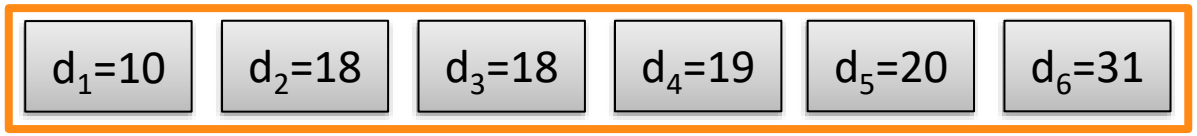
The virtual view in the position of the view 1



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:

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

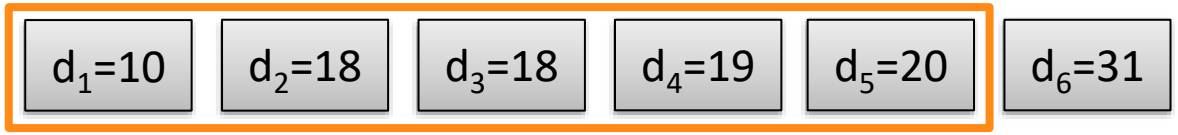


Poznan Depth Refinement – available within MPEG Video Coding and on https://gitlab.com/dmieloch/depth-map-refinement

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:

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

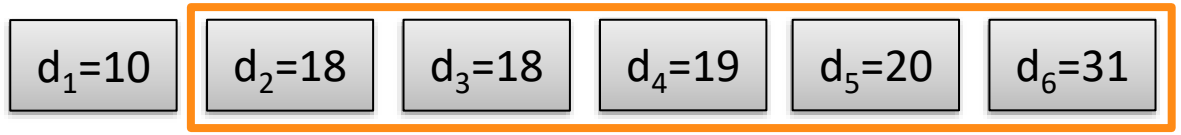


Poznan Depth Refinement – available within MPEG Video Coding and on https://gitlab.com/dmieloch/depth-map-refinement

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:

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

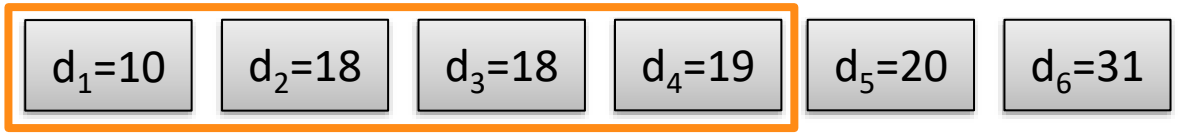


Poznan Depth Refinement – available within MPEG Video Coding and on https://gitlab.com/dmieloch/depth-map-refinement

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:

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

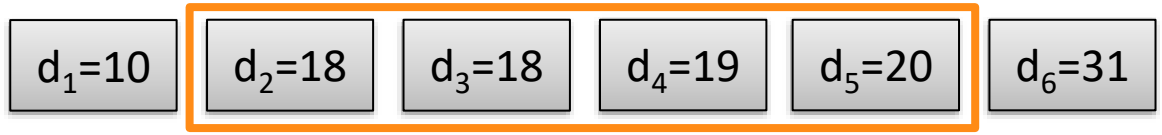


Poznan Depth Refinement – available within MPEG Video Coding and on https://gitlab.com/dmieloch/depth-map-refinement

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:

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

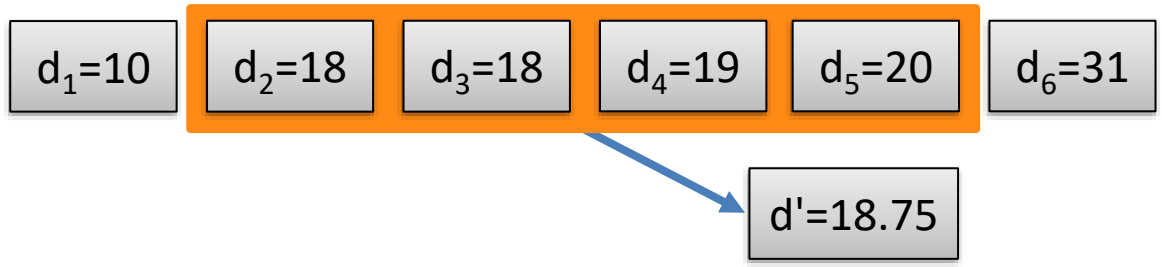


Poznan Depth Refinement – available within MPEG Video Coding and on https://gitlab.com/dmieloch/depth-map-refinement

Depth refinement – example

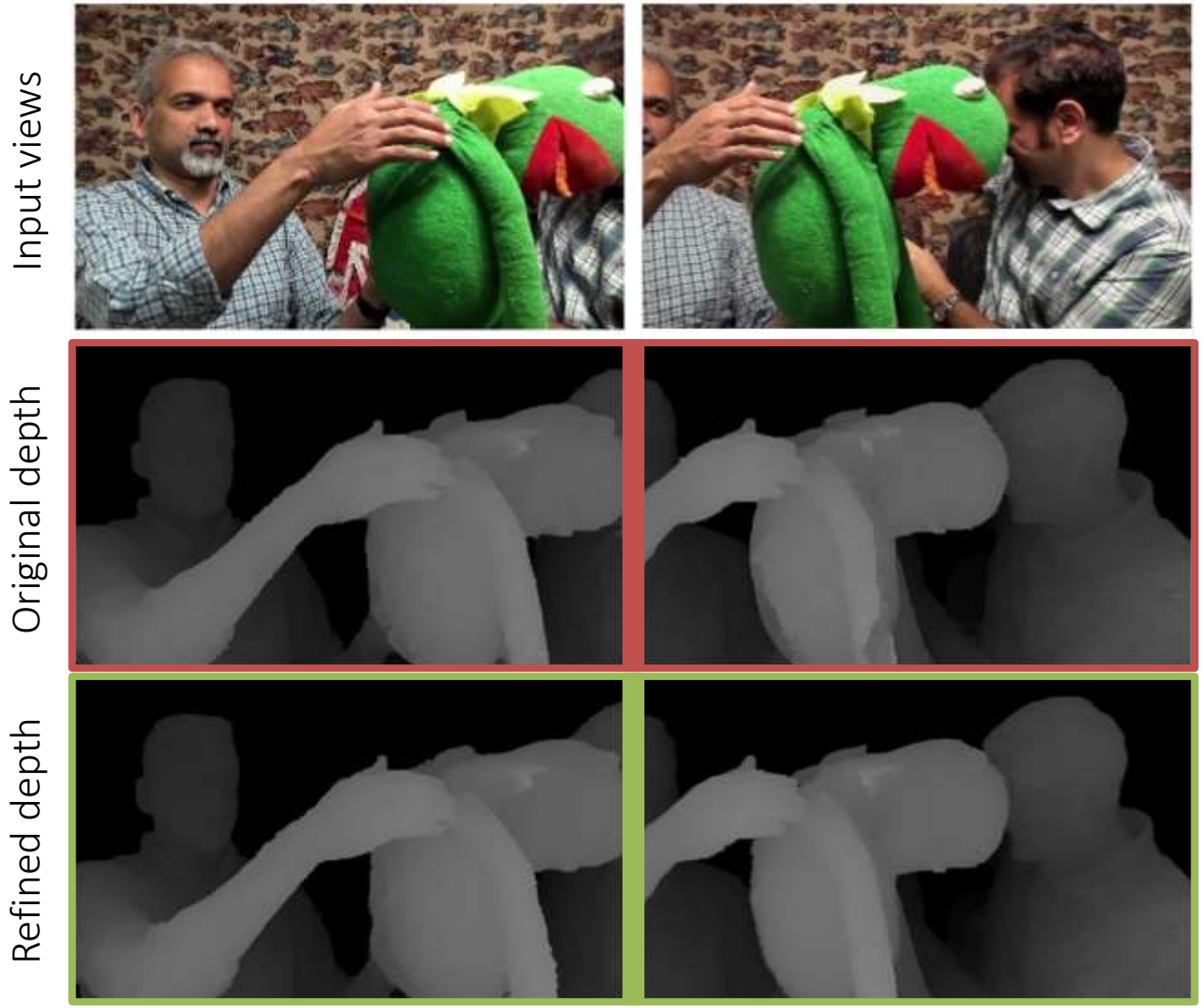
- Only information from depth maps is used
  - Texture can introduce errors in the refinement due to inter-view color inconsistencies and noise
- List $\boldsymbol{D}_i(x,y)$ of depth values projected from various input depth maps is created and sorted in ascending order:

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



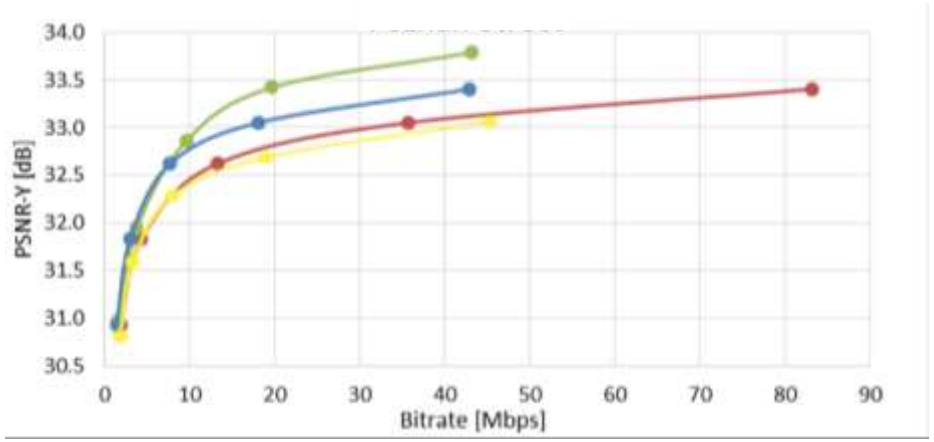
Poznan Depth Refinement – available within MPEG Video Coding and on https://gitlab.com/dmieloch/depth-map-refinement

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)

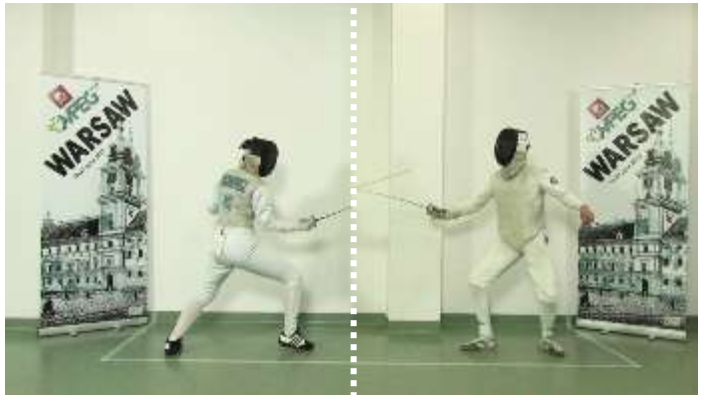


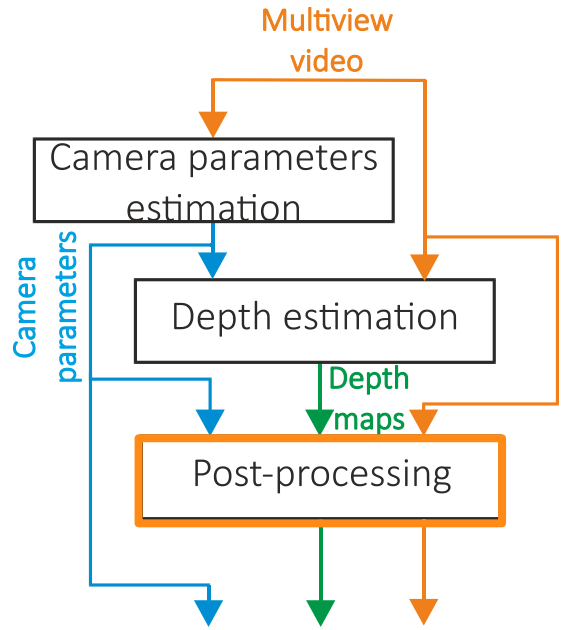
Poznan Depth Refinement – available within MPEG Video Coding and on https://gitlab.com/dmieloch/depth-map-refinement

Color refinement – motivation

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

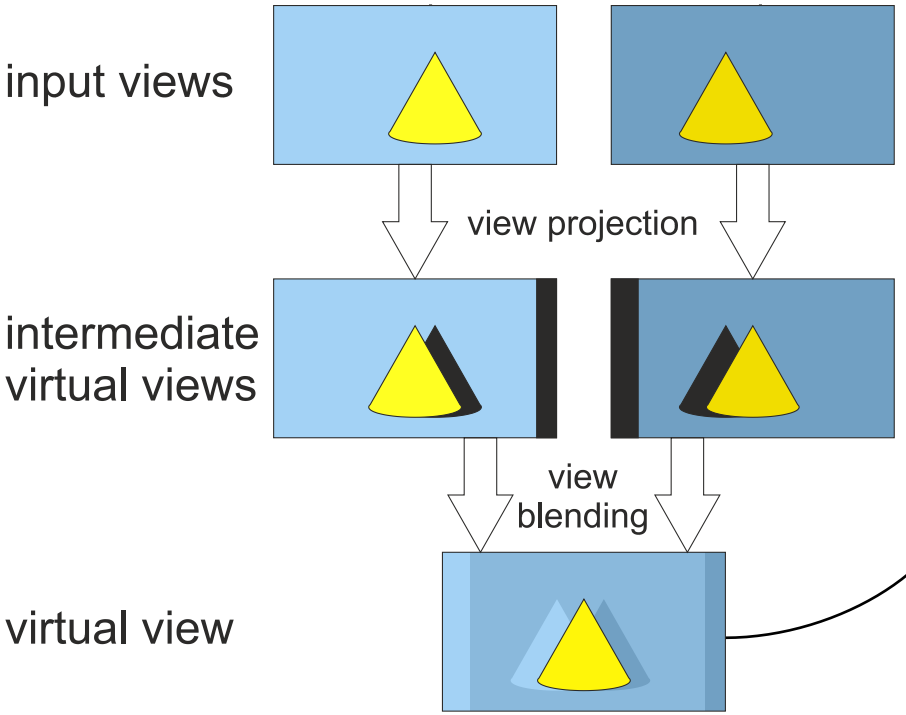
Comparison of the first frame(left side of pictures) and the last frame (right side)





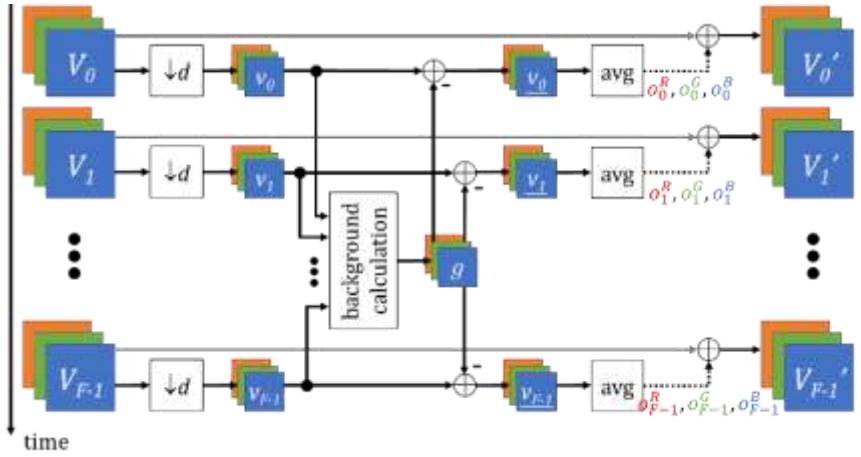
Color refinement – motivation

- Lack of inter-view consistency

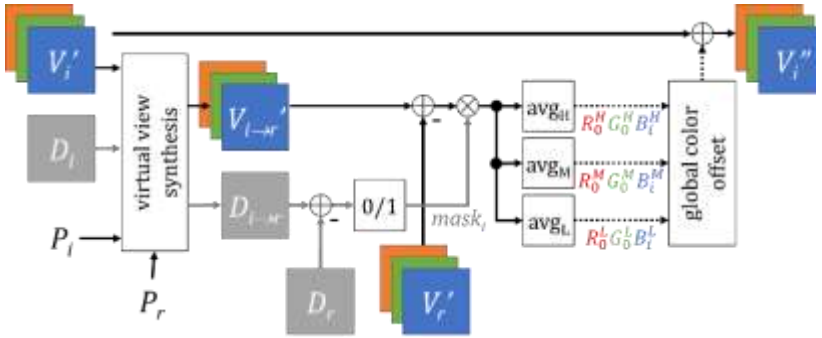


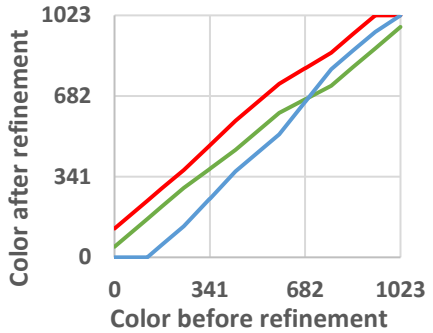
Color refinement – example

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

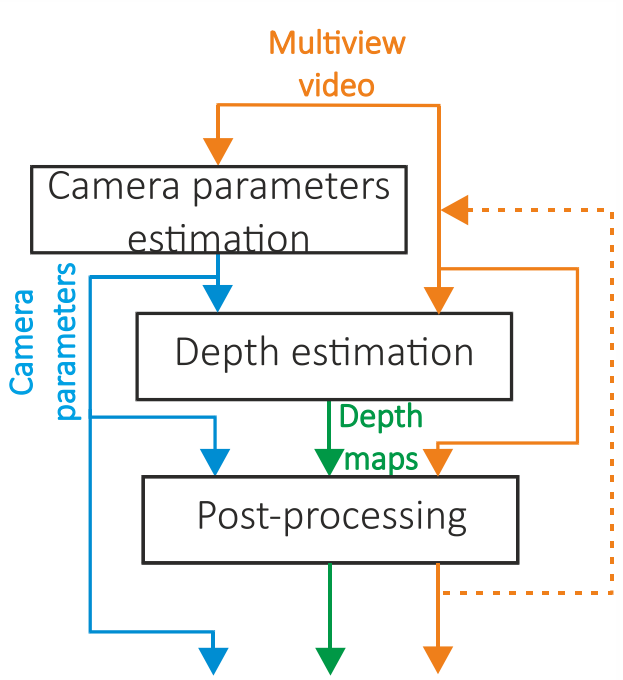
Color refinement – results

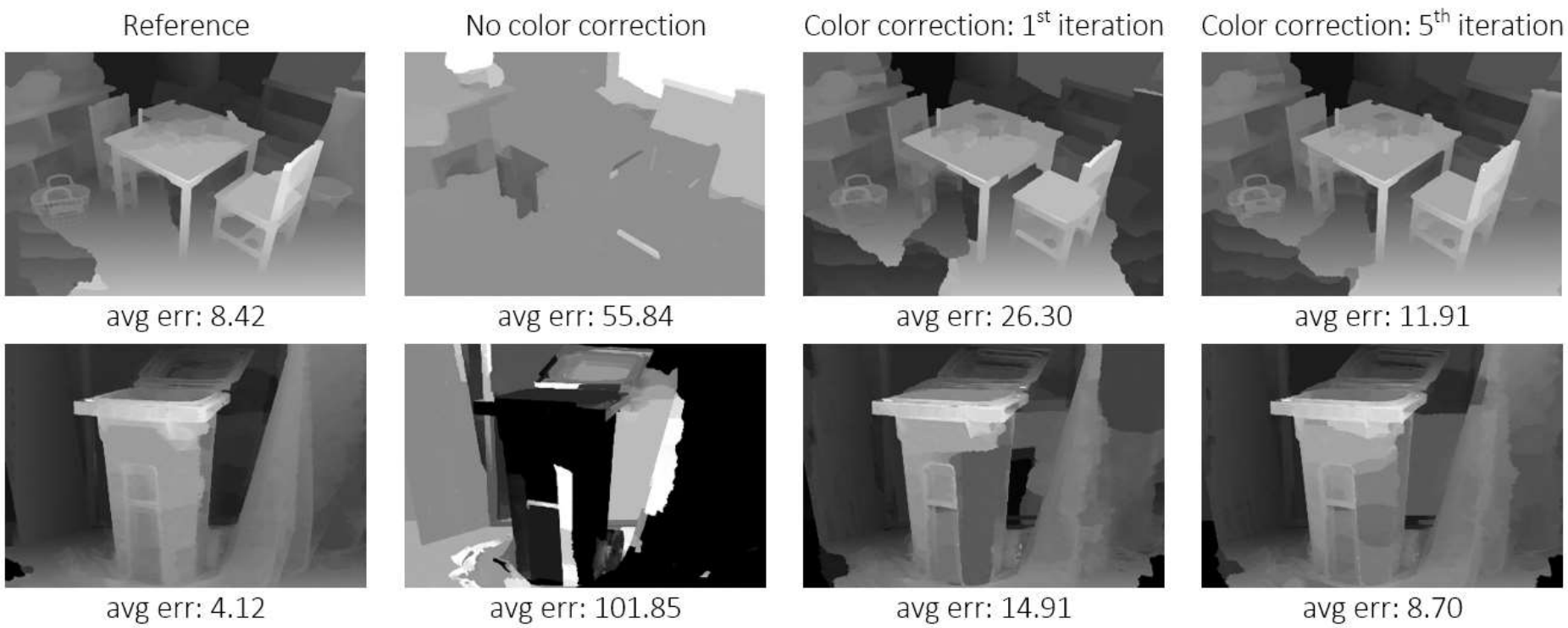
Before color correction

After color correction

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

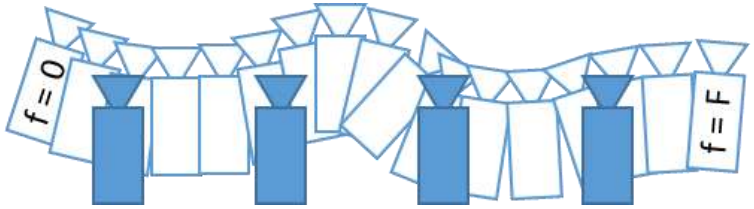
Color refinement – influence on depth estimation





Evaluation of the quality

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

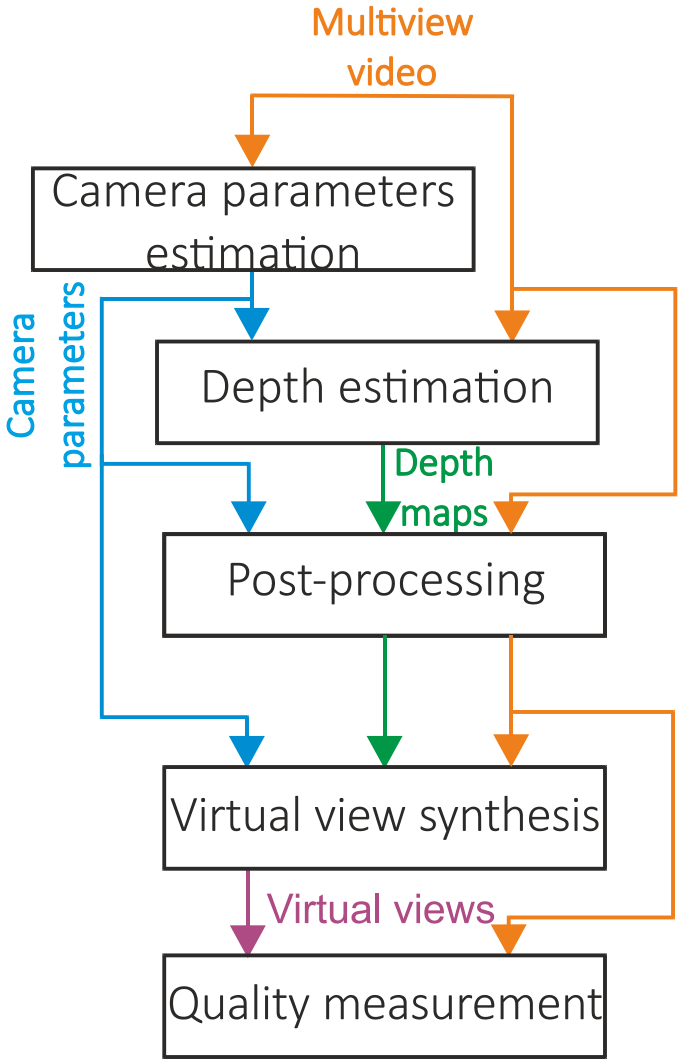


- Evaluation by measurement of quality of views synthesized in the position of input views (blue cameras)

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

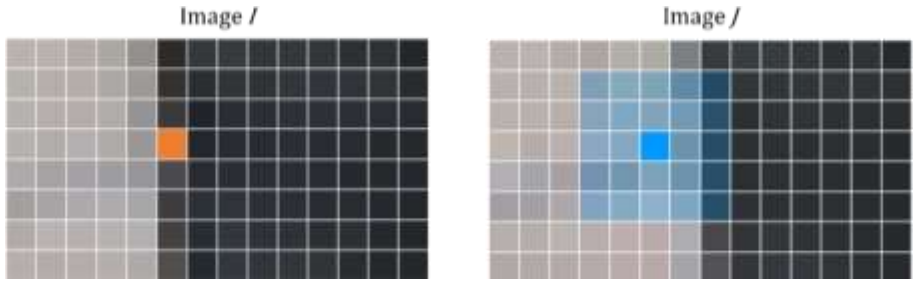
Synthesis performed using e.g. Real-time Reference View Synthesis (RVS)

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

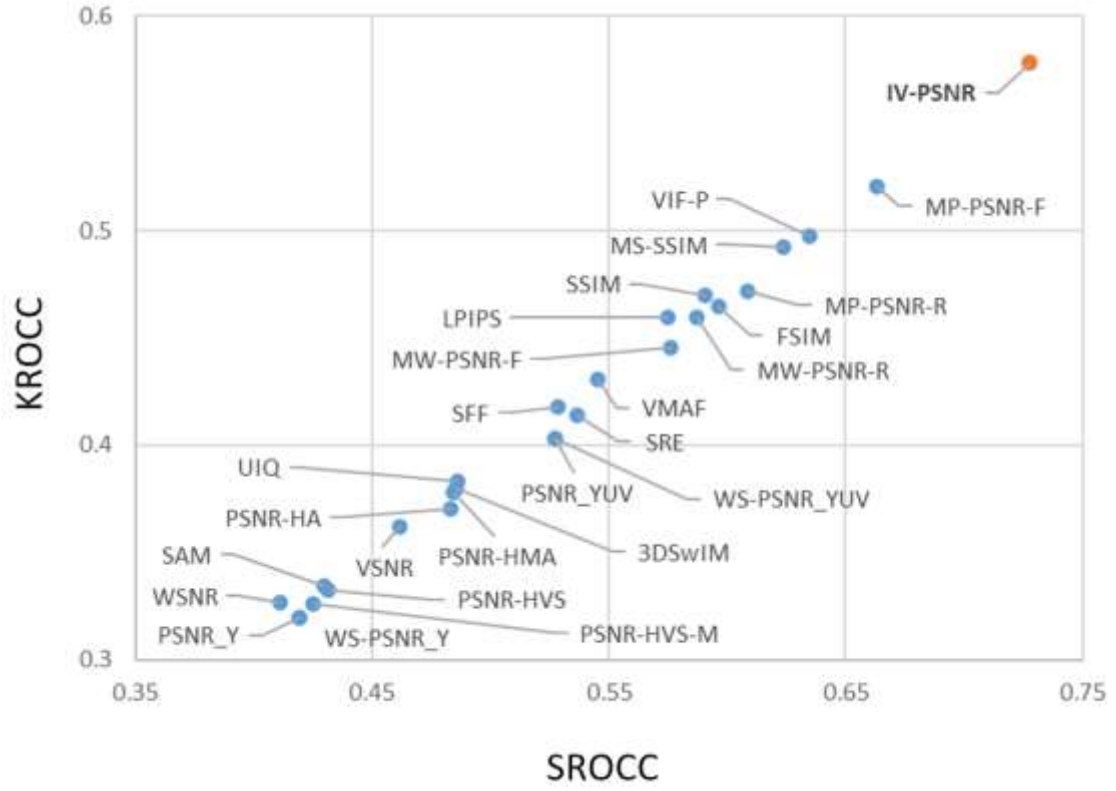
Source code on: https://gitlab.com/mpeg-i-visual/rvs/

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

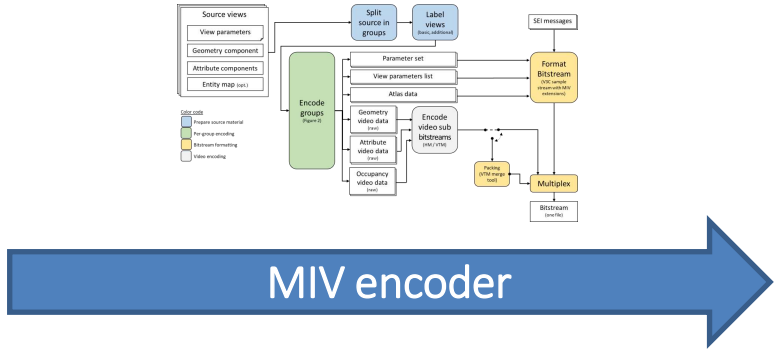


- The effectiveness of the IV-PSNR metric assessed using the results of the “MPEG Call for Proposals on 3DoF+ Visual”

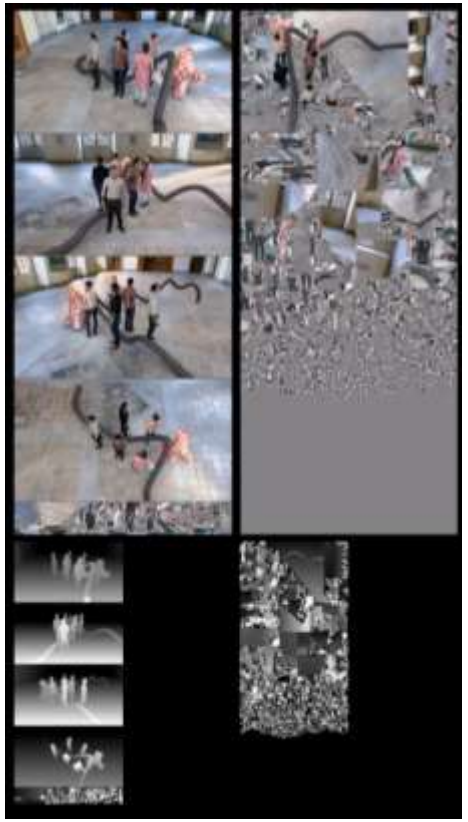


SROCC – Spearman Rank Order Correlation Coefficients
KROCC – Kendall Rank Order Correlation Coefficients

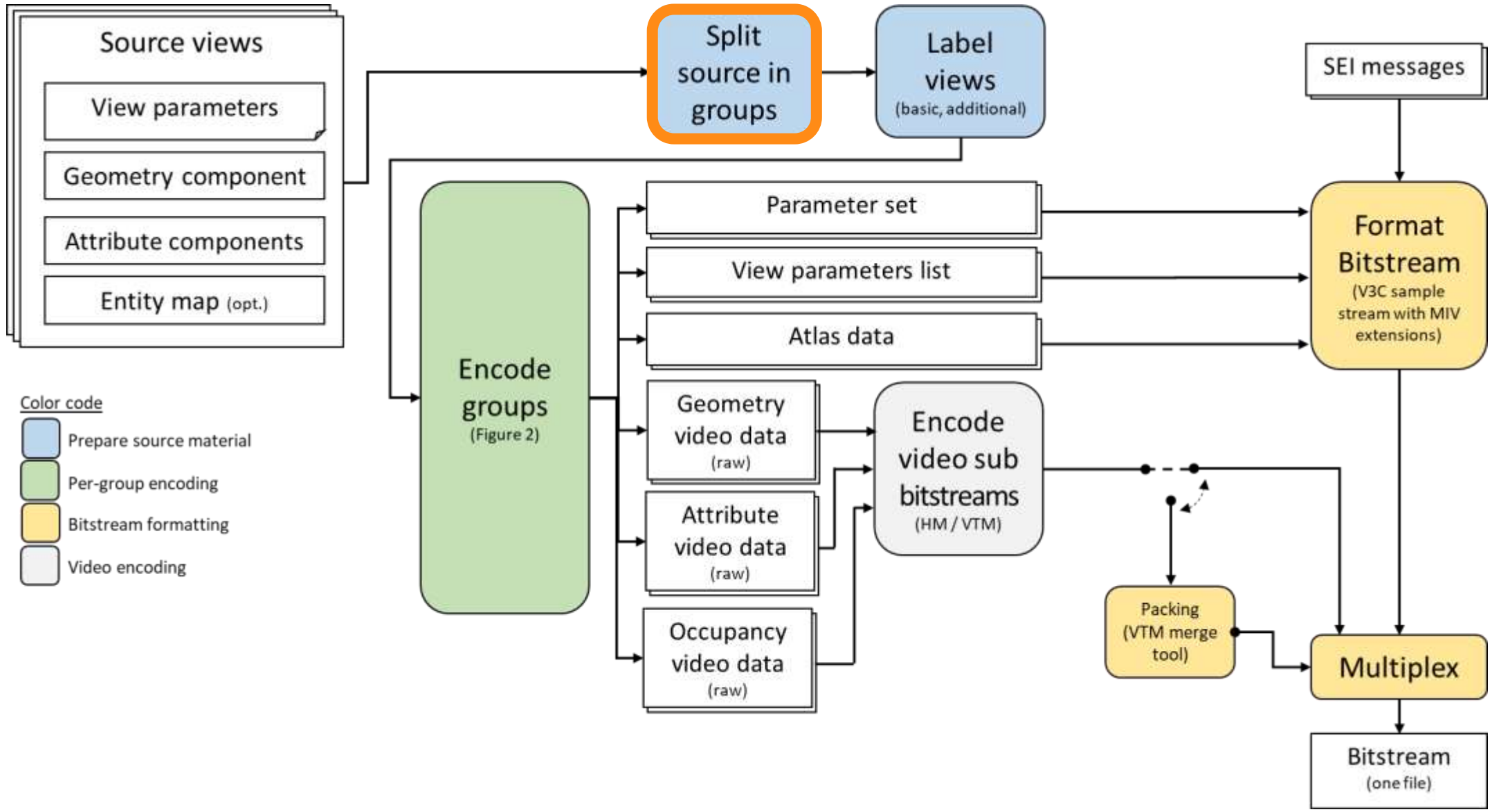
MIV encoder



- 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**

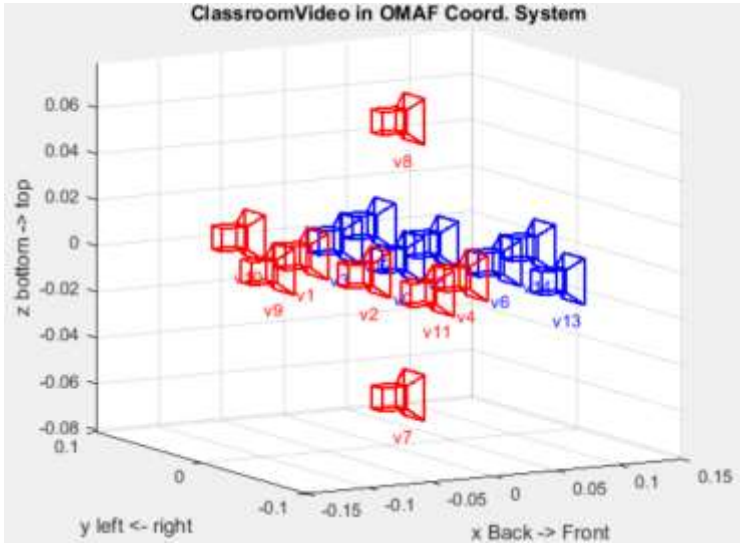


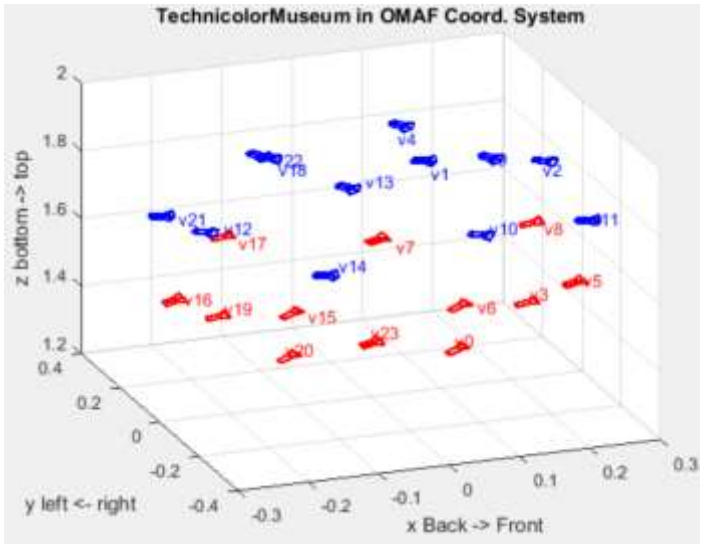
MIV encoder

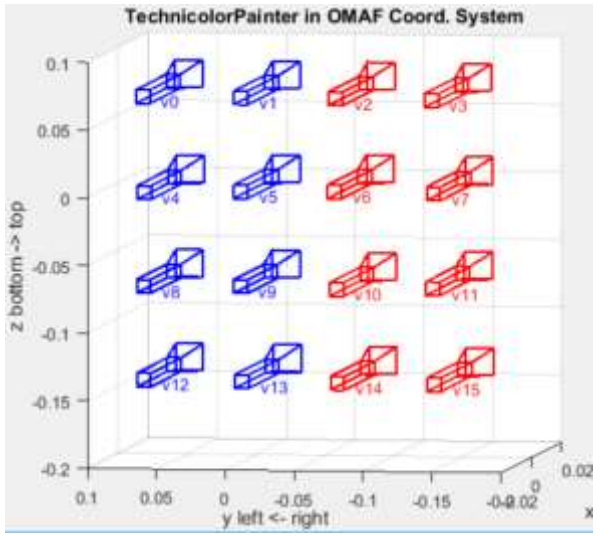

Source views
View parameters
Geometry component
Attribute components
Entity map (opt.)
Split source in groups
Label views
(basic, additional)
SEI messages
Encode groups
(Figure 2)
Parameter set
View parameters list
Atlas data
Format Bitstream
(V3C sample stream with MIV extensions)
Geometry video data
(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

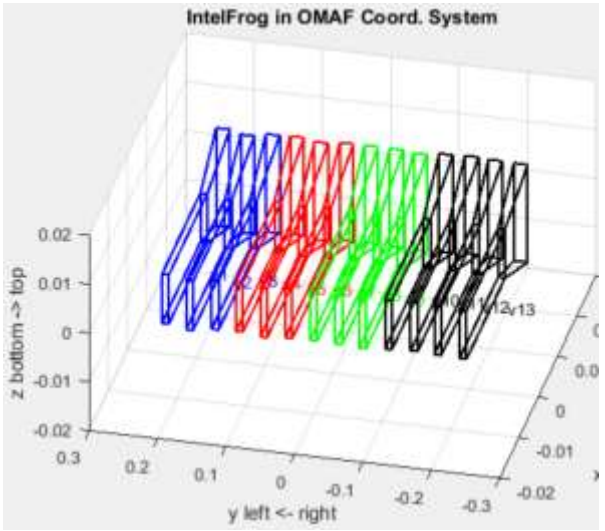
Grouping of source views

Grouping of source views

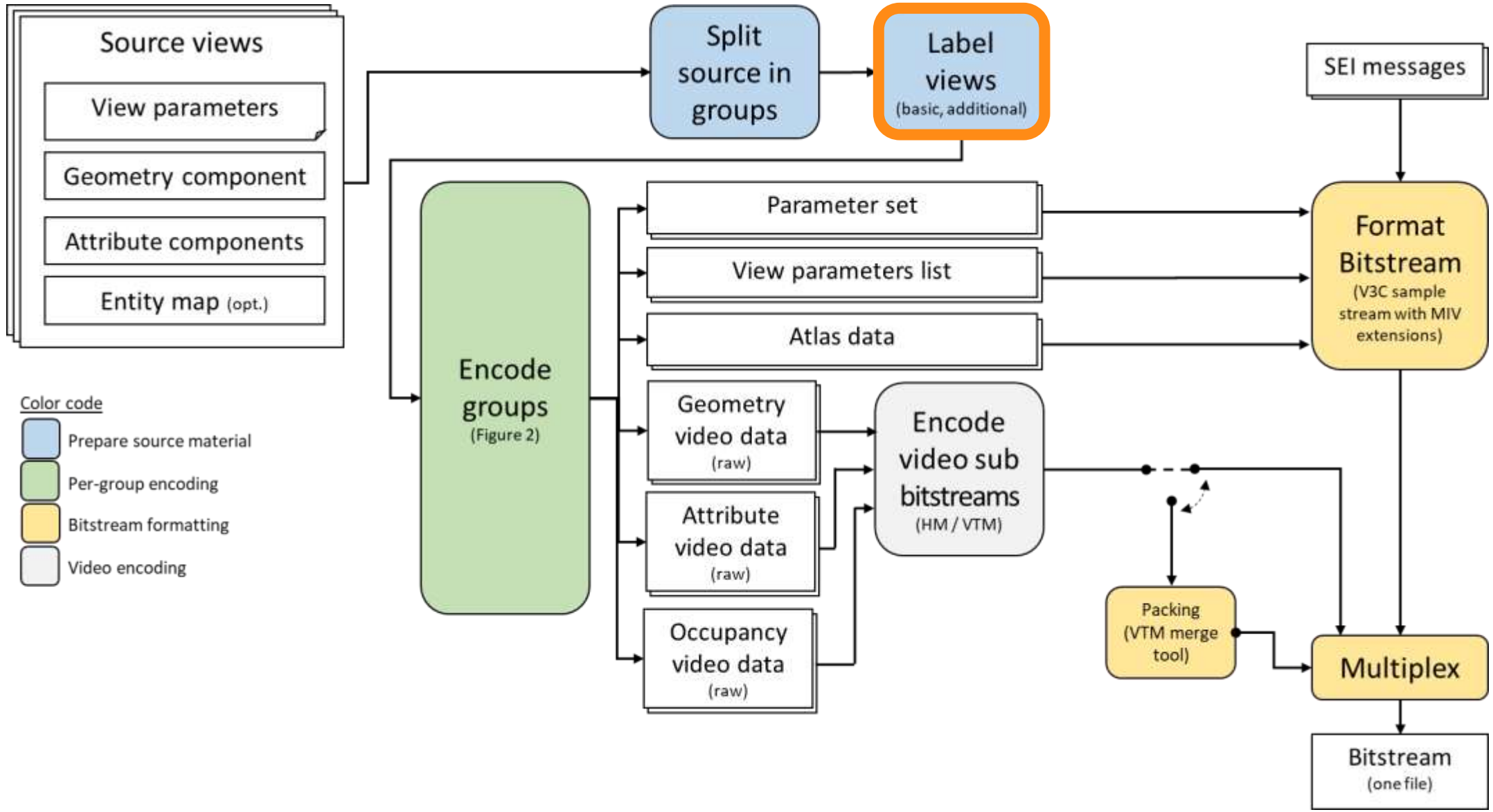








MIV encoder


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

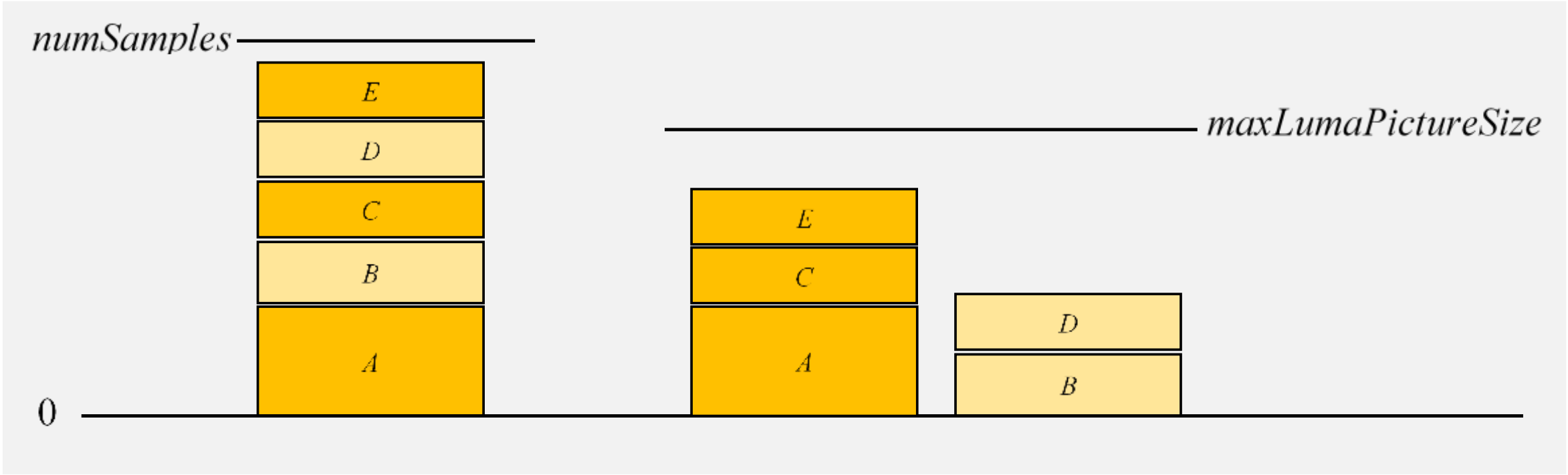
Labeling algorithm:

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

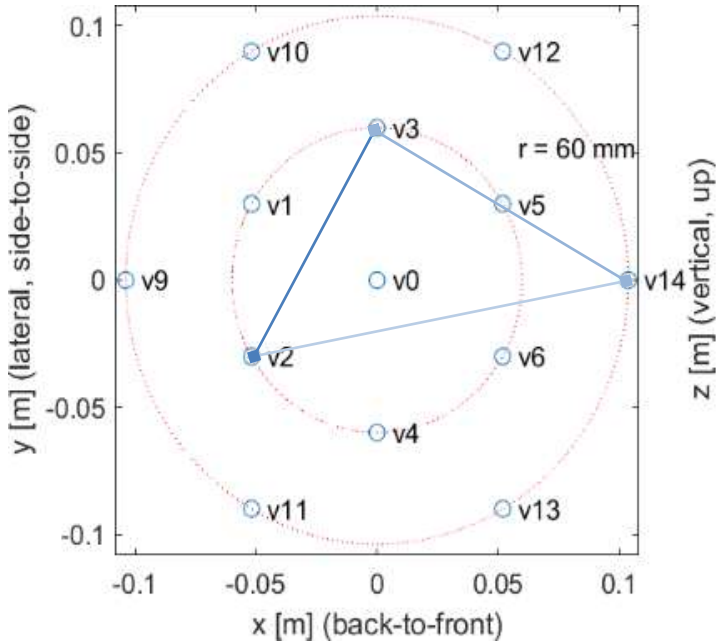


View labeling

Labeling algorithm:

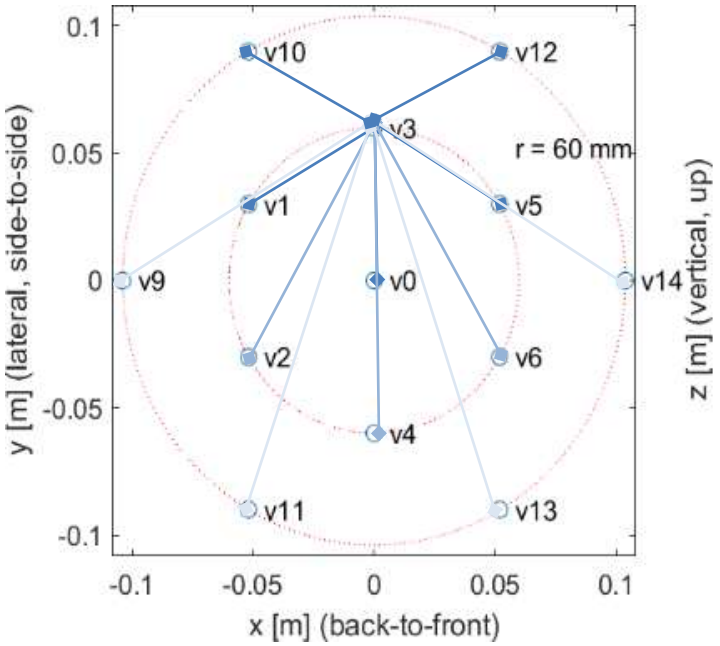
1. Set the number of basic views
2. Calculate the cost function
3. Choose basic views
4. Update labeling

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**:

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

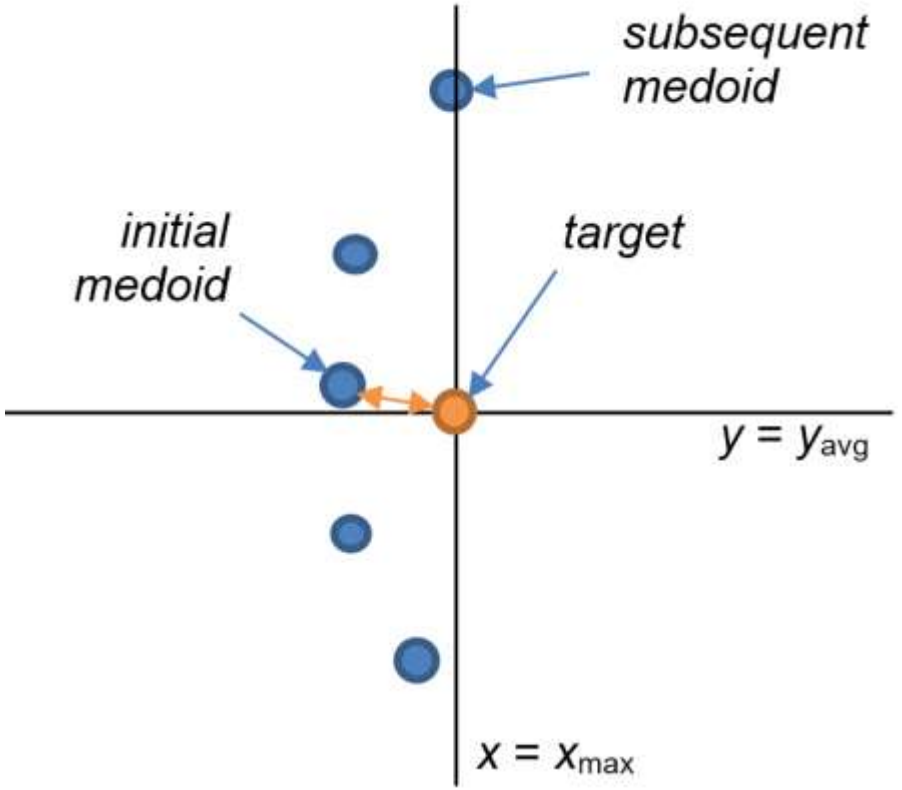
View labeling

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)



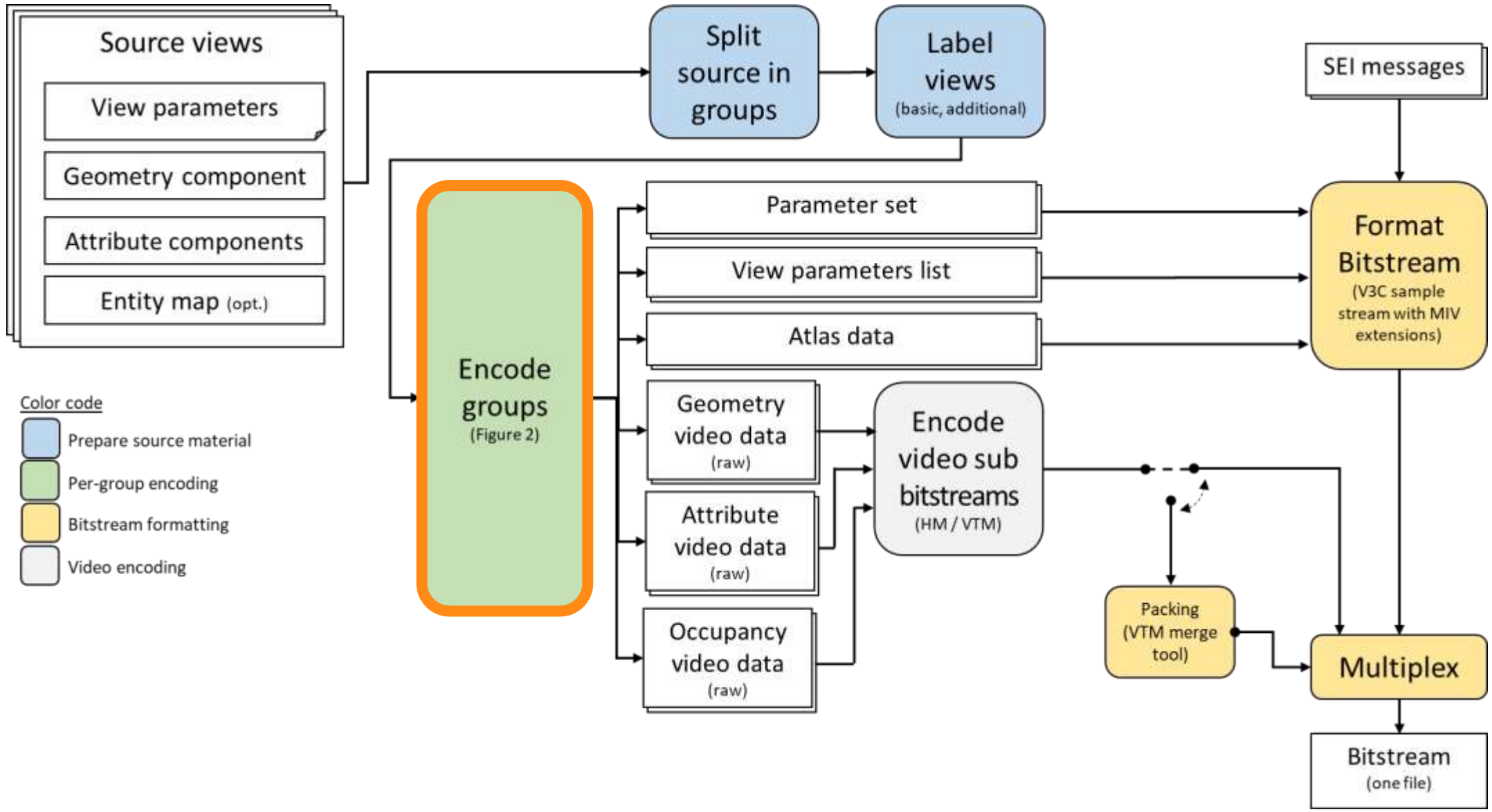
View labeling

Labeling algorithm:

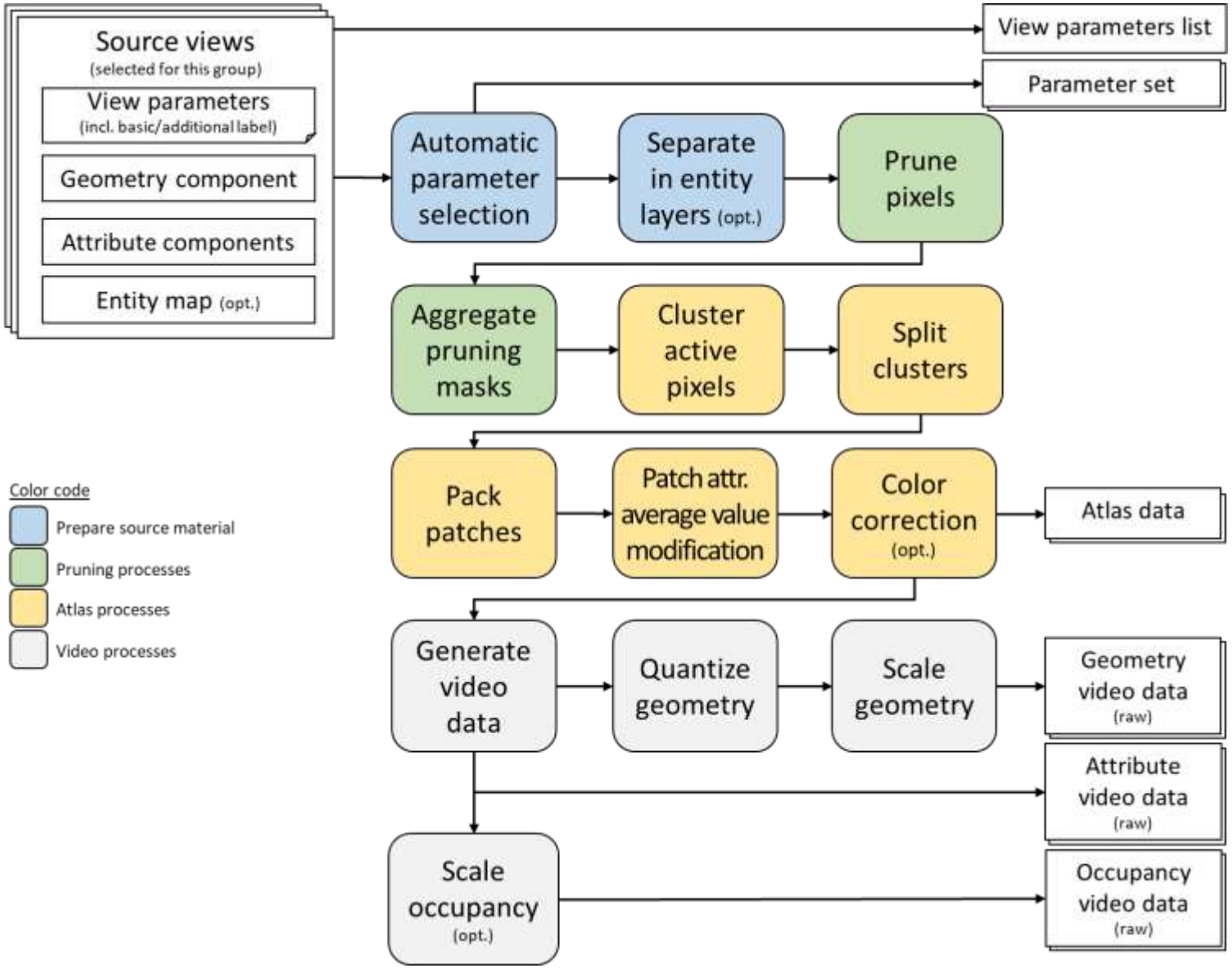
1. Set the number of basic views
2. Calculate the cost function
3. Choose basic views
4. Update labeling

- 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

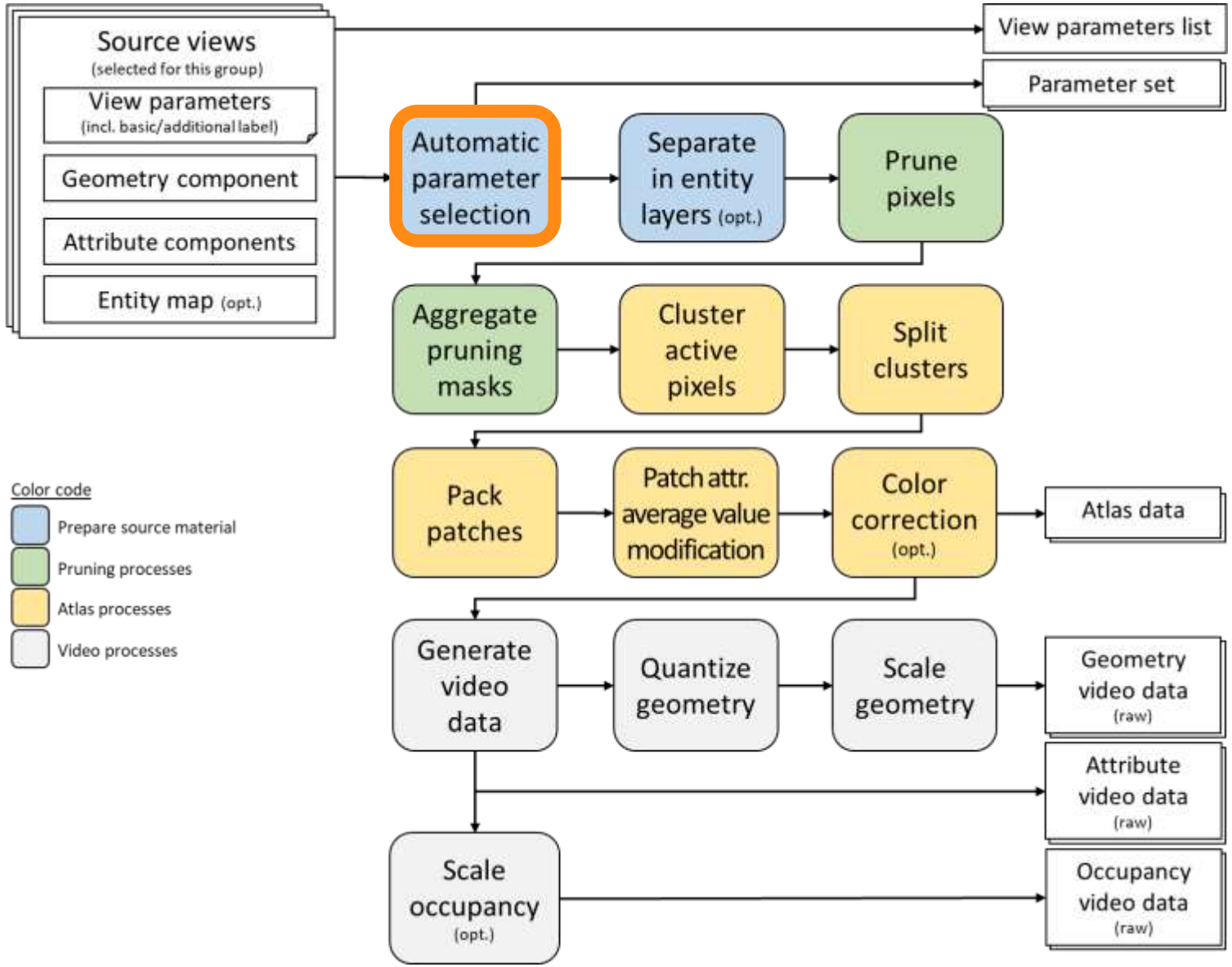
MIV encoder


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