EECE5512
Networked XR Systems

Some of the slides credits: MPEG & Groot paper

Last Class - Recap

- 2D Video Compression
- Open3D

Lecture Outline for Today

- Homework2
- Depth Map Compression
- Point Cloud Compression
  - MPEG GPCC
  - MPEG VPCC

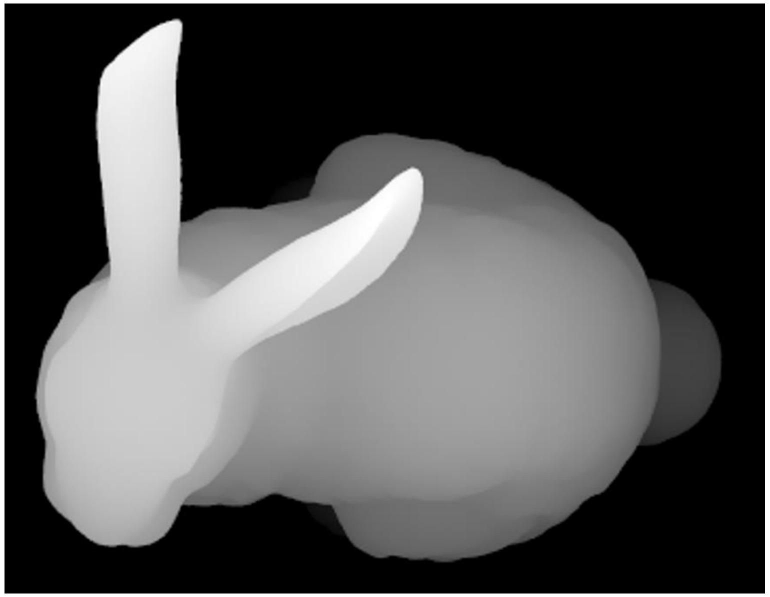
Depth Map Compression

- Why do we need depth compression?
  - For storage
  - For streaming over a network under lower bandwidths
  - For 3D scene reconstruction on client devices

- Compute the date rate for a depth video with a resolution of 1024x1024.

Depth Map Compression

- How to compress depth?
  - Can we use similar ideas that we discussed in case of 2D video codecs?
  - Frame prediction
  - Transform coding & quantization
  - Entropy coding



Depth Map Compression

- Why not just adopt standard video codecs?
  - They have been engineered for decades
  - Probably no need to reinvent similar algorithms if we can directly input the depth videos to color video codecs

Adopting Standard Video Codecs

- Challenge
    - Compression schemes for standard videos are highly tuned for color videos
    - Considering human perception, e.g., by spending fewer bits on color than luminance information, and so forth.
    - These insights do not apply to depth compression.

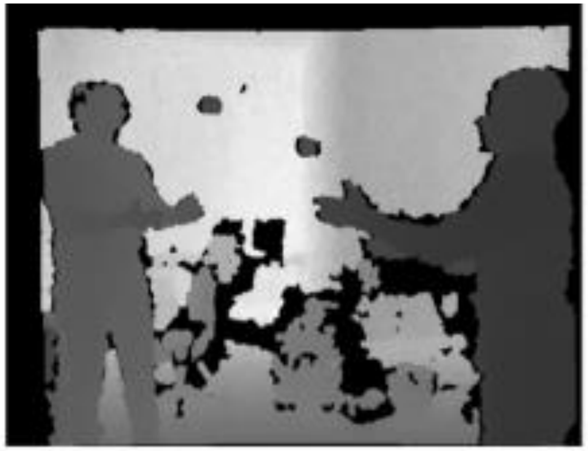
Adopting Standard Video Codecs

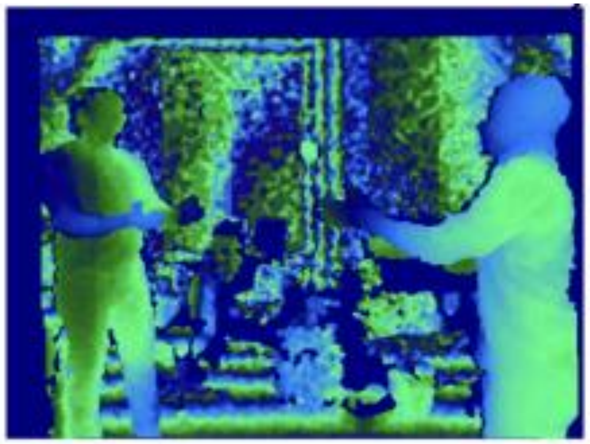
- Challenge
  - Bit-depth and channel inconsistency
  - Depth videos are single channeled
  - Bit-depth of depth videos is larger than color videos in general

  - Example: 8-bit videos can only store coarse-grained depth or short range (trade-off), so typically you need 16-bit or 24-bit or 32-bit-detphs for depth videos

Adopting Standard Video Codecs

- Can we convert the single channel large bit-depths to three channel small bit-depth?





https://jankautz.com/publications/depth-streaming.pdf

Adopting Standard Video Codecs

- Example: How do we pack a single 16 bit-depth depth map into a three channel 8 bit-depth depth map?
- Bit Multiplexing Method
  - Take a chunk of bits from 16 bits, insert them in each in 8-bit channel
  - E.g., first 6 MSB of 16-bits into first 6 MSB of first channel, subsequent 5 bits into first 5 MSB of second channel, last 5 bits into first 5 MSB of third channel
  - Pad with zeros for the remaining bits

Bit Packing

- Consider a 16-bit depth value D, which we split across the three 8-bit channels (R, G, B) using the multiplexing method.
- **Original Depth Value**:
  1. Suppose D = 1010101111001101 (binary) or 0xABCD in hexadecimal.
  2. After packing, let's say we have:
     1. **R Channel**: 10101000 (8 bits, representing the first 6 MSBs plus padding zeros)
     2. **G Channel**: 11110000 (8 bits, representing the next 5 bits plus padding zeros)
     3. **B Channel**: 01101000 (8 bits, representing the last 5 bits plus padding zeros)

Bit Packing

- **Quantization During Encoding**:
- When this packed representation is encoded using a lossy video codec, quantization occurs to reduce the data size.
- Imagine that, during quantization, the codec changes a single bit in the **R Channel** from 10101000 to 10100000. This represents a change in one of the most significant bits, dropping the value from 0xA8 to 0xA0.

Bit Packing

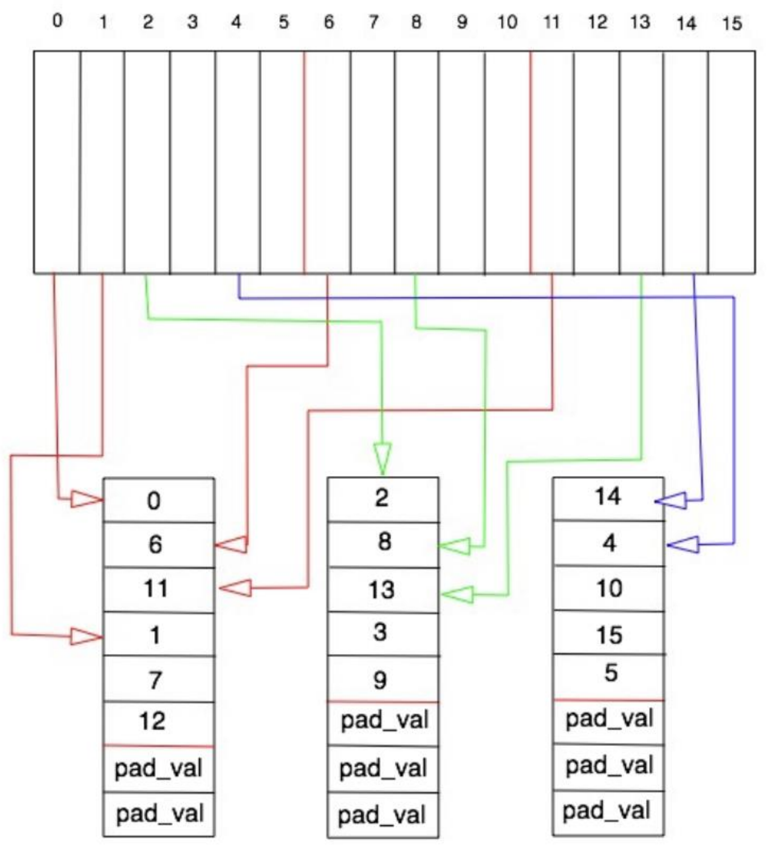
- **Effect on Reconstructed Depth**:
- After decoding, when you try to reconstruct the original 16-bit depth value from the modified R, G, and B channels, you would end up with a different depth value:
- Original **R Channel**: 10101000 (binary) → contributes heavily to the overall depth value.
- Modified **R Channel**: 10100000 (binary) → introduces a significant difference.
- Let's calculate the impact:
  - The original 6 MSBs were 101010, but after quantization, they became 101000.
  - This means the depth value decreased significantly because it is missing 2^5 units in magnitude due to this change.
  - In a depth map context, such a change can represent a substantial shift in the measured distance, potentially several meters if the depth values cover a wide range.

Adopting Standard Video Codecs

- Example: How do we pack a single 16 bit-depth depth map into a three channel 8 bit-depth depth map?
- Bit Multiplexing Method
  - Take a chunk of bits from 16 bits, insert them in each in 8-bit channel
  - Problems?
    - Loss in MSBs can cause large depth discontinuities

Adopting Standard Video Codecs

- Example: How do we pack a single 16 bit-depth depth map into a three channel 8 bit-depth depth map?
- Interleaved Bit Multiplexing
    - The first few bits (e.g., 0, 6, 11, etc.) go into one channel.
    - Another set of bits (e.g., 2, 8, 13, etc.) goes into a different channel.
    - A third set of bits (e.g., 14, 4, 10, etc.) goes into the final channel.



https://jankautz.com/publications/depth-streaming.pdf

Adopting Standard Video Codecs

- Interleaved bit packing
    - Since each channel now carries a mix of different bit positions, the impact of compression artifacts is more evenly distributed, potentially making the entire packed value more resistant to the errors introduced by lossy compression.
    - This means that even if some bits are lost or altered, the chance of catastrophic errors affecting the entire depth value is reduced.

Adapting Standard Video Codecs

- General Limitations
  - Lossy
- Lossless entropy coding
    - Fast Lossless Depth Image Compression, ISS’17
      - Skips frame prediction, transform and quantization to avoid depth discontinuities
      - Applies entropy coding
      - But only for images, not for video
    - Temporal RVL: A Depth Stream Compression Method, IEEE VR’20
      - For videos; computes deltas across frames

Lecture Outline for Today

- Homework2
- Depth Map Compression
- Point Cloud Compression
    - MPEG GPCC
    - MPEG VPCC

Point Clouds

- A point cloud is a discrete set of data points in space.
- Or a set of 3D independent points
- Each Point (X, Y, Z) + Attributes
- Attributes: Color, Alpha, Reflectance

Point Cloud

- Representation
  - Each Point is a floating-point number – 32 bits
  - <X, Y, Z> : 96 bits
  - RGB: 3 channels: 24 bits
  - Also, has other attributes sometimes (light related)

  - Each point: 96 + 24 bits or 15 bytes

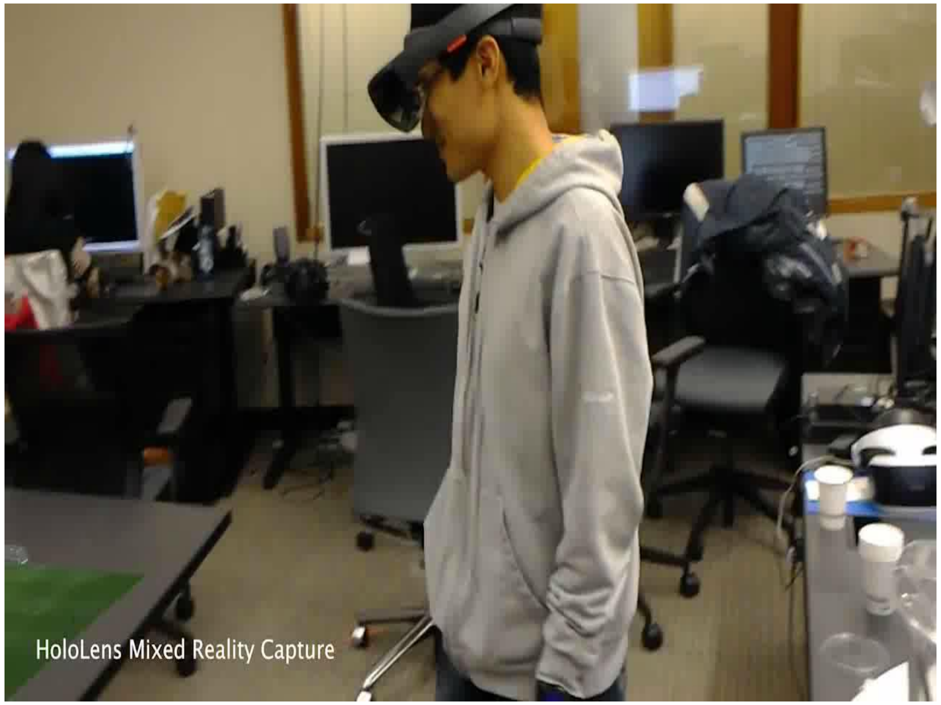
- Typically, a point cloud has thousands to millions of points – guess the data rate numbers

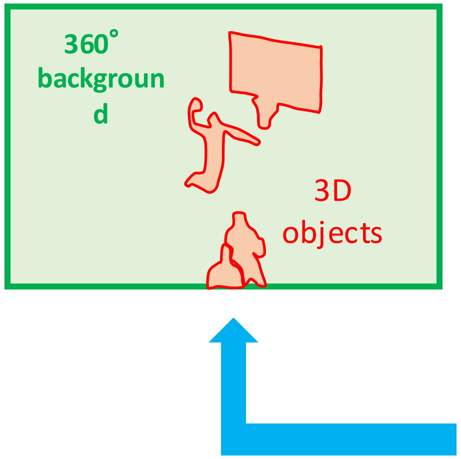
Point Cloud

Sample data numbers

	queen	longdress	loot	redandblack	soldier
Average number of points (in 300 frames)	1,005,000	834,000	794,000	727,000	1,076,000
Bitrates for transmitting uncompressed video (Mbytes/s)	514.47	542.22	490.61	448.21	681.96

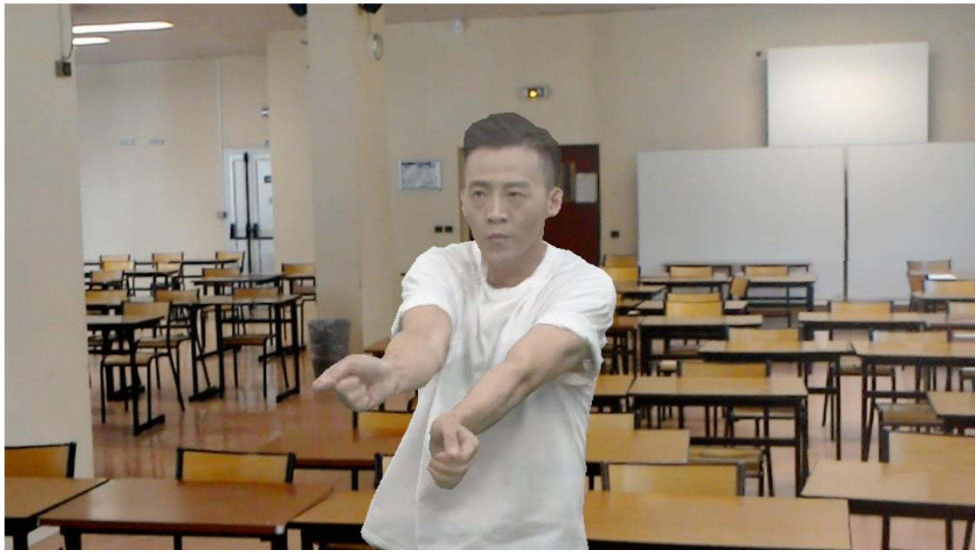
Example Applications





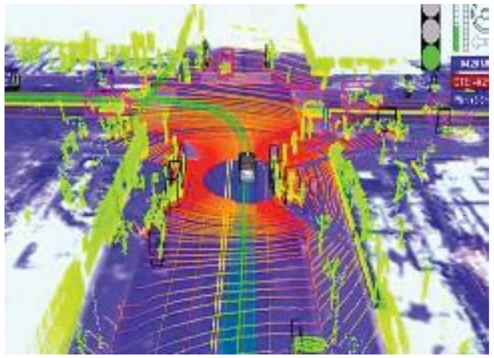
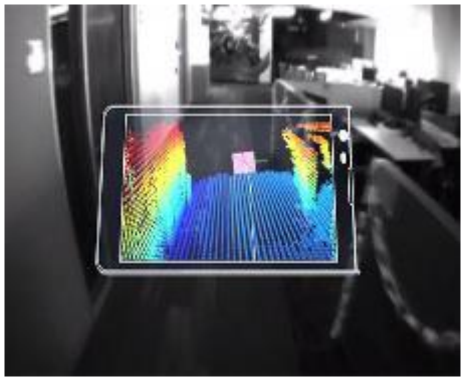
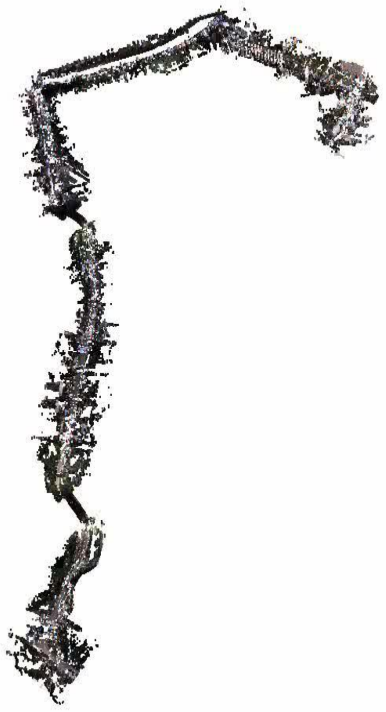
1-3 Gbps per object

Example Applications



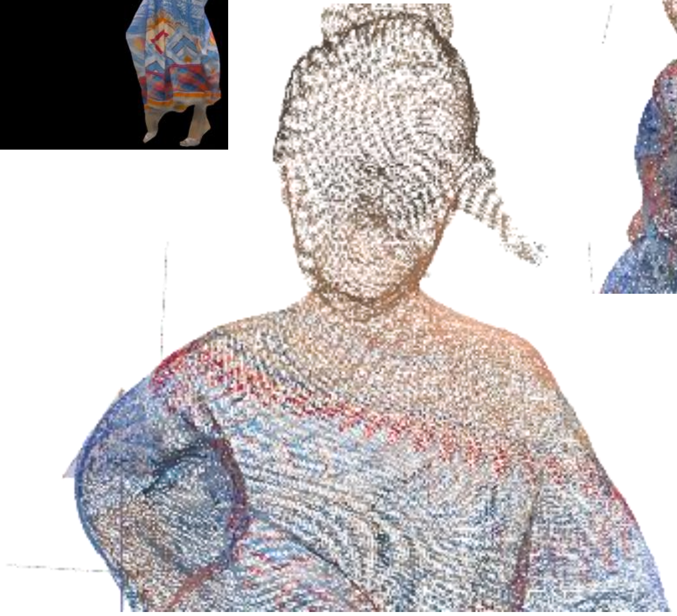
Example Applications

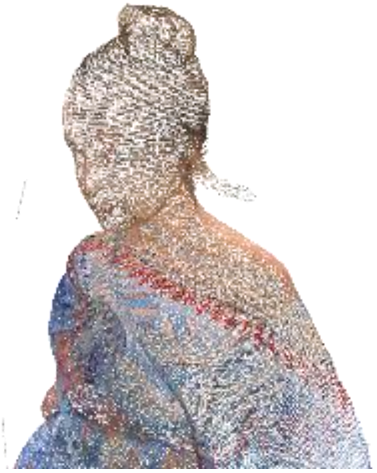
- ~20 million points
  - 2,020,734,515 bytes

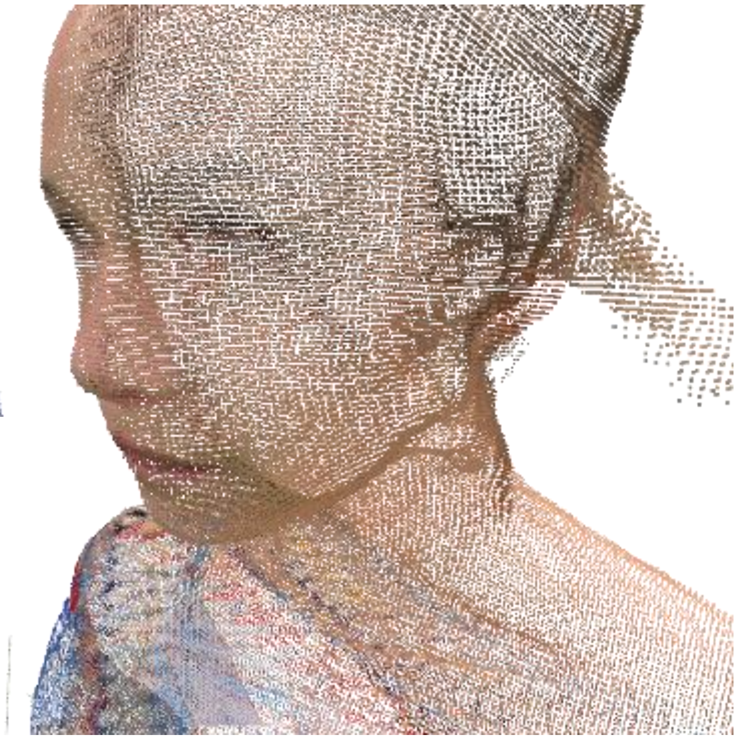




Point Cloud Compression

800,000 points -> 1 000 Mbps (uncompressed)







Compression is required in order to make PC useful

Point Cloud Compression

- Can we use similar block based intra and inter frame prediction and transform coding for point clouds?
  - E.g., is it possible to divide the point cloud into 3D blocks and apply similar techniques that we used in case of 2D videos (block matching algorithms etc.)

MPEG Point Cloud Compression

V-PCC 01/2020

G-PCC 4/2020

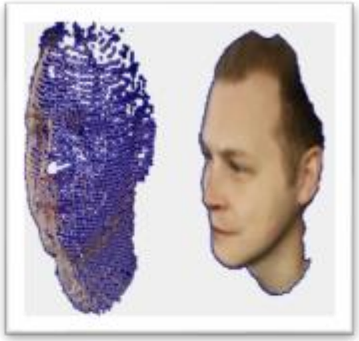
2014 2015 2016 2017 2018 2019 2020

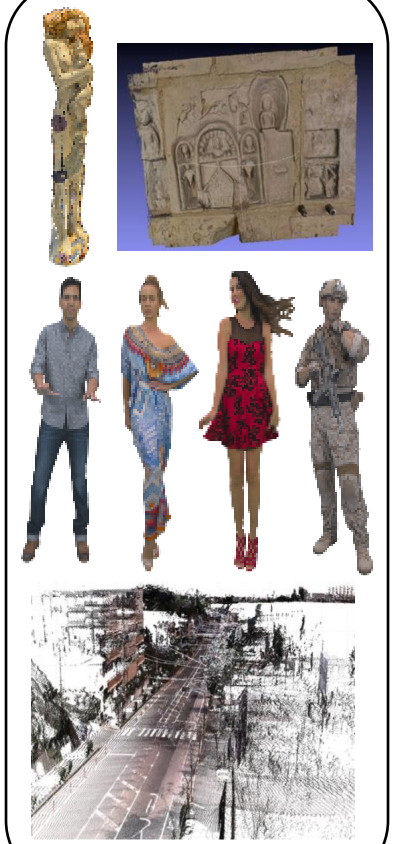
MPEG initiated the work on PCC

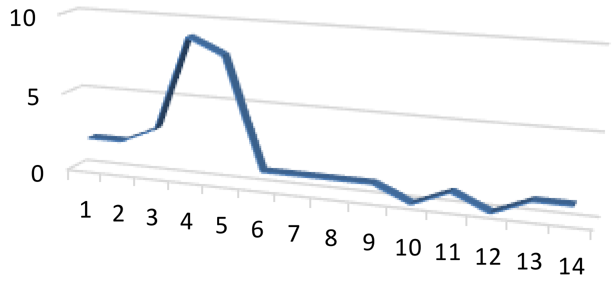
In April 2017 MPEG issued a Call for Proposals

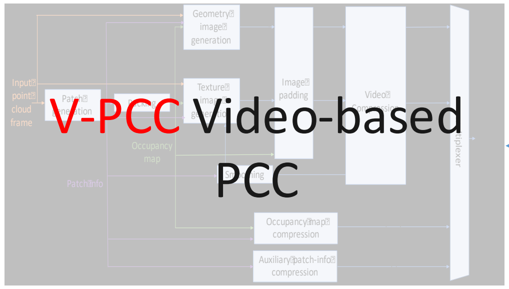
First Committee Draft issued in October 2018

9 technology leading companies responded and MPEG evaluated them in October 2017

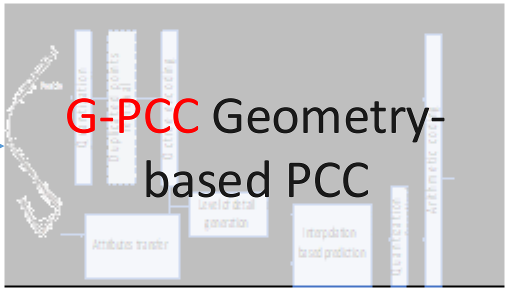








V-PCC Video-based PCC



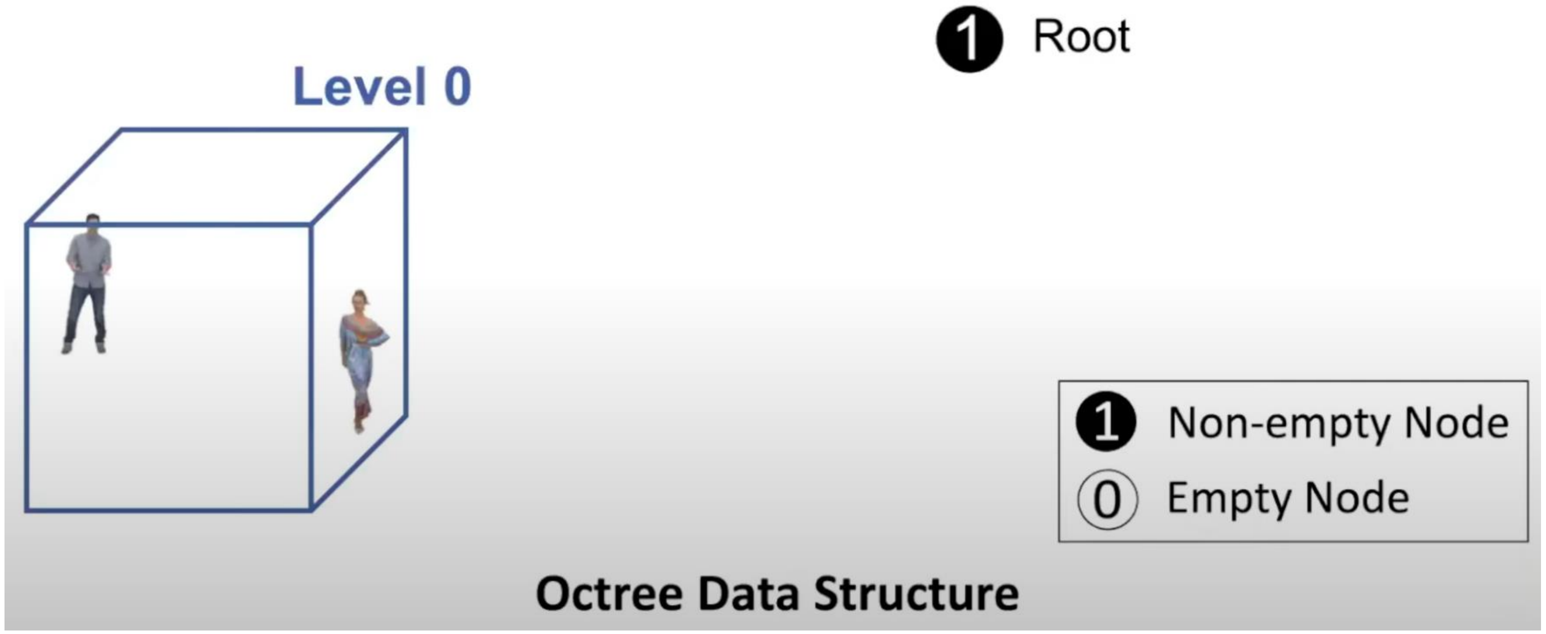
G-PCC Geometry-based PCC

MPEG GPCC

- Geometry based point cloud compression
  - 3D tree data structures (Octree or KD-tree)

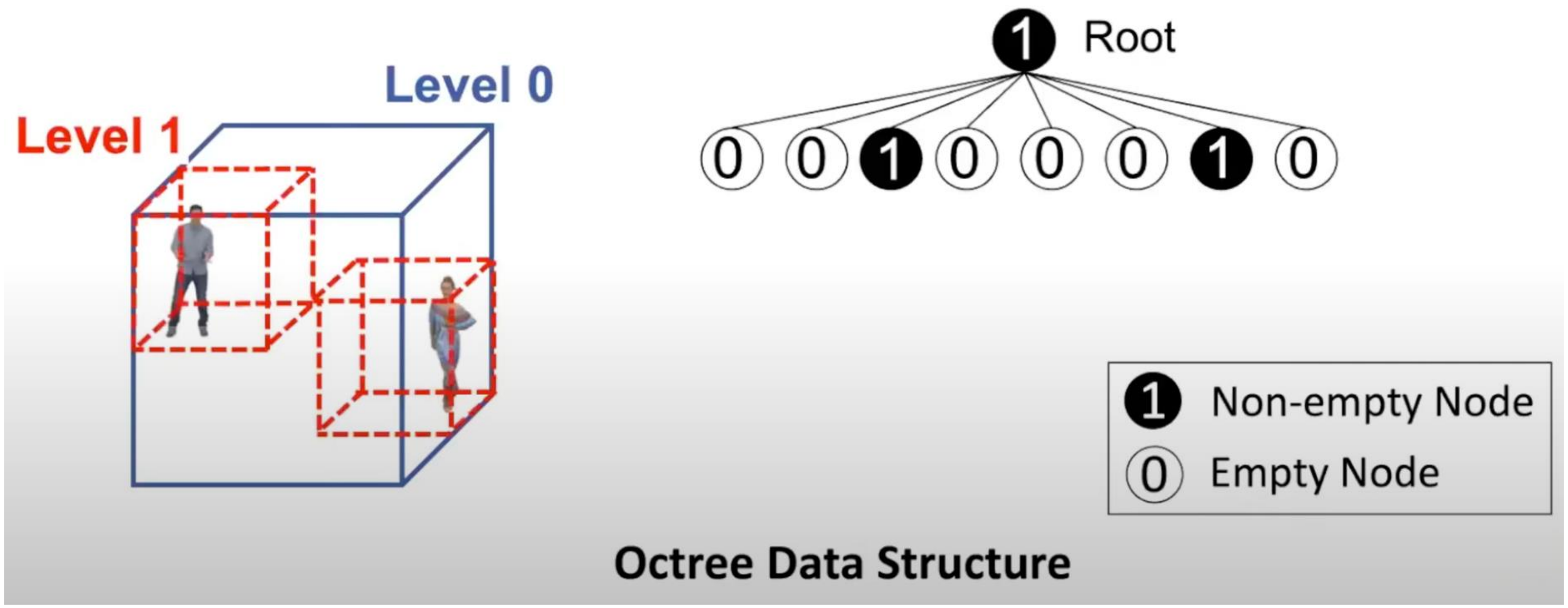
MPEG GPCC

- Geometry based point cloud compression
  - 3D tree data structures (Octree or KD-tree)



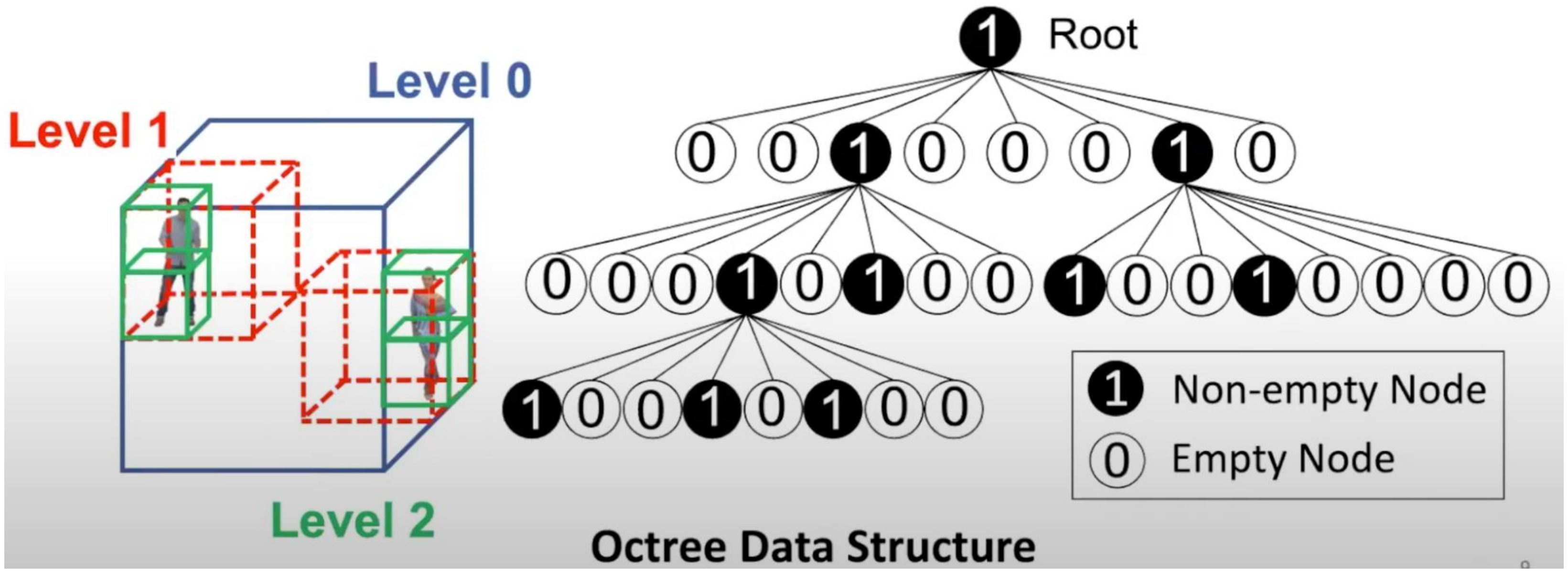
MPEG GPCC

- Geometry based point cloud compression
  - 3D tree data structures (Octree or KD-tree)

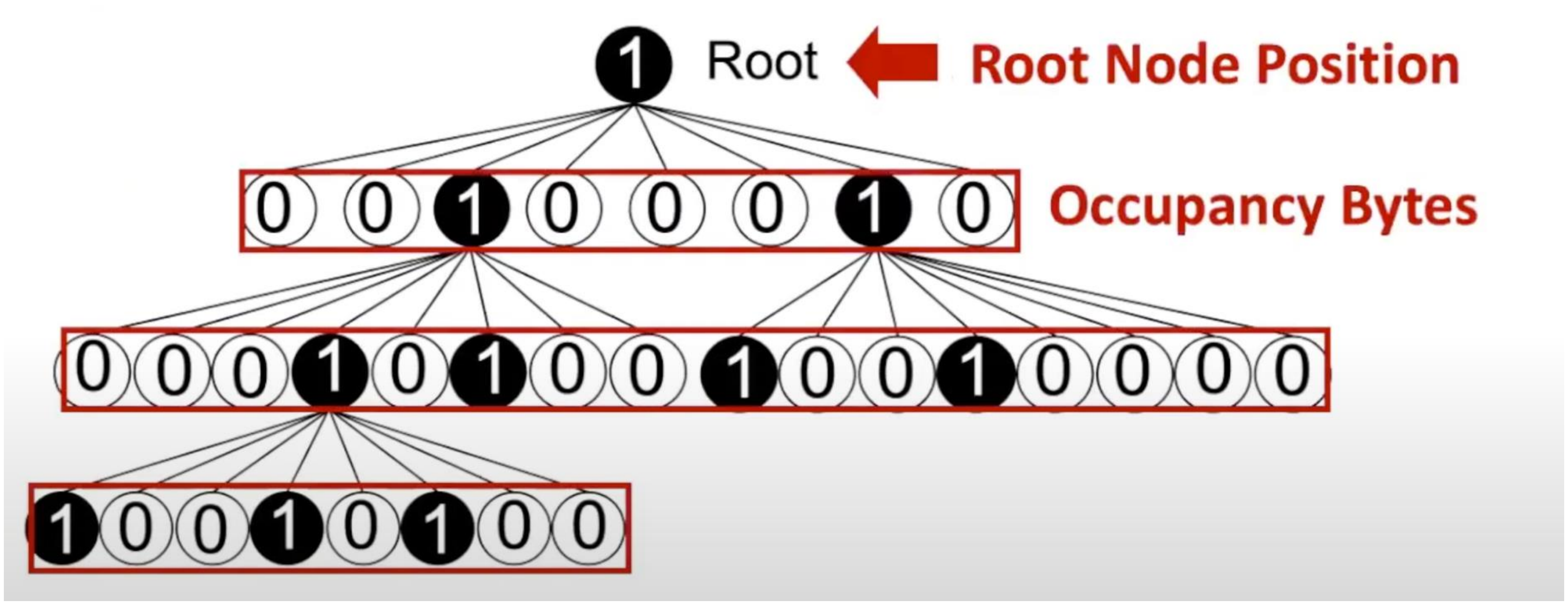


MPEG GPCC

- Geometry based point cloud compression
  - 3D tree data structures (Octree or KD-tree)

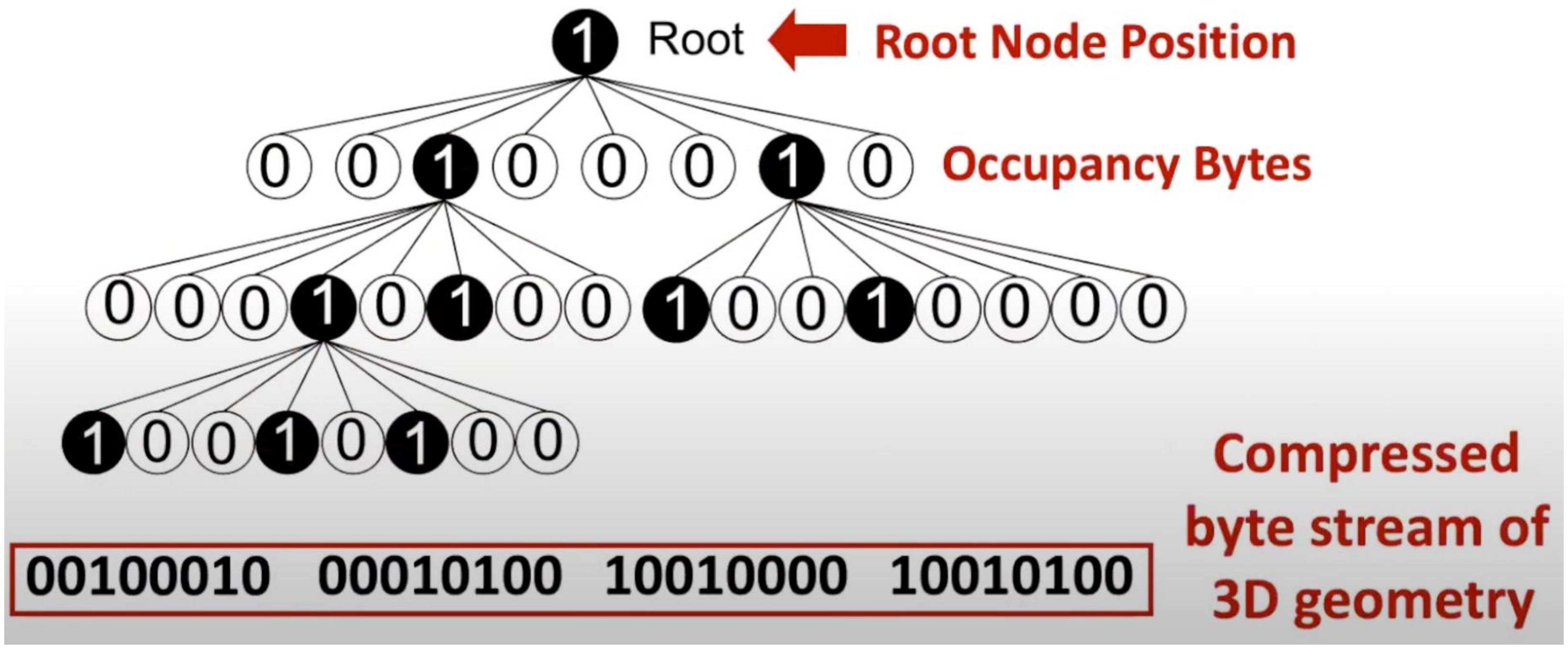


MPEG GPCC



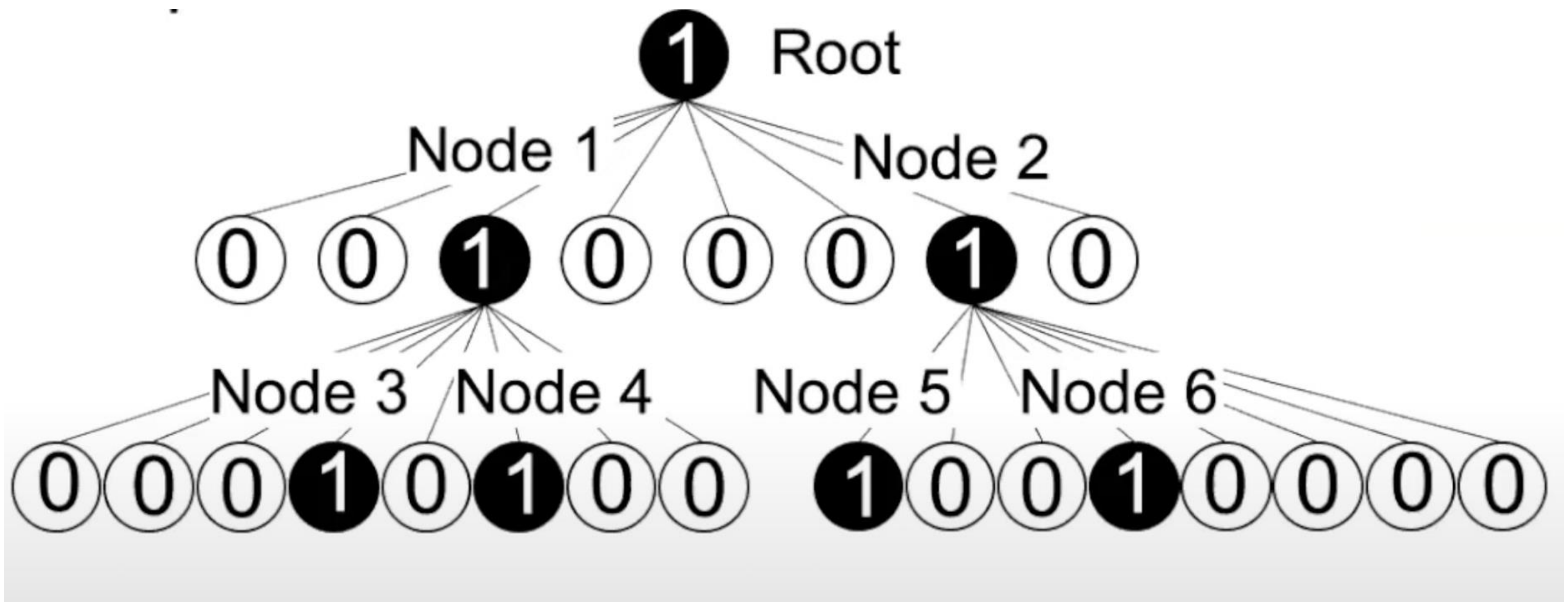
1 Root
Root Node Position
0 0 1 0 0 0 1 0
Occupancy Bytes
0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 0
1 0 0 1 0 1 0 0

MPEG GPCC

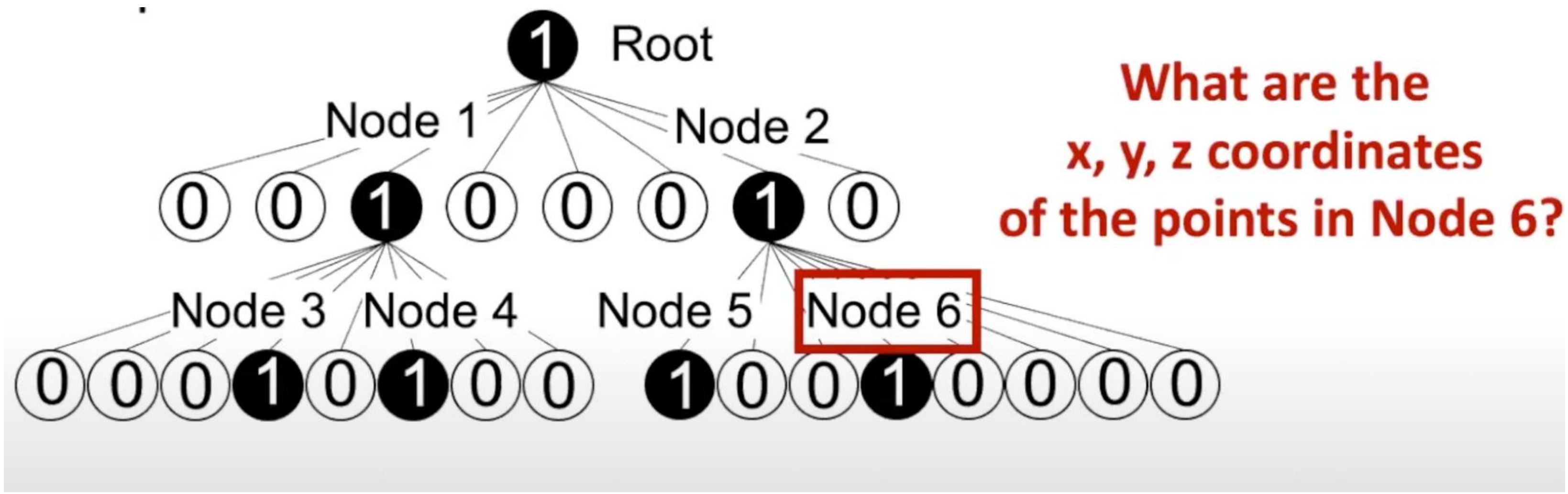


Root
Root Node Position
Occupancy Bytes
Compressed byte stream of 3D geometry
00100010 00010100 10010000 10010100

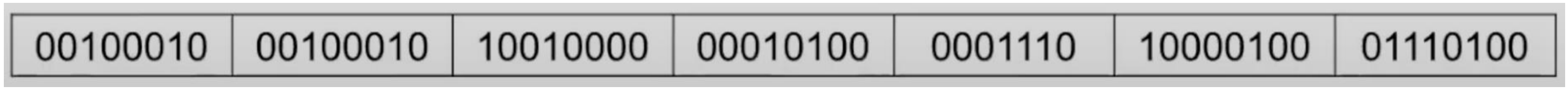
MPEG GPCC


1 Root
Node 1
Node 2
0 0 1 0 0 0 1 0
Node 3
Node 4
Node 5
Node 6
0 0 0 1 0 1 0 0
1 0 0 1 0 0 0 0

MPEG GPCC



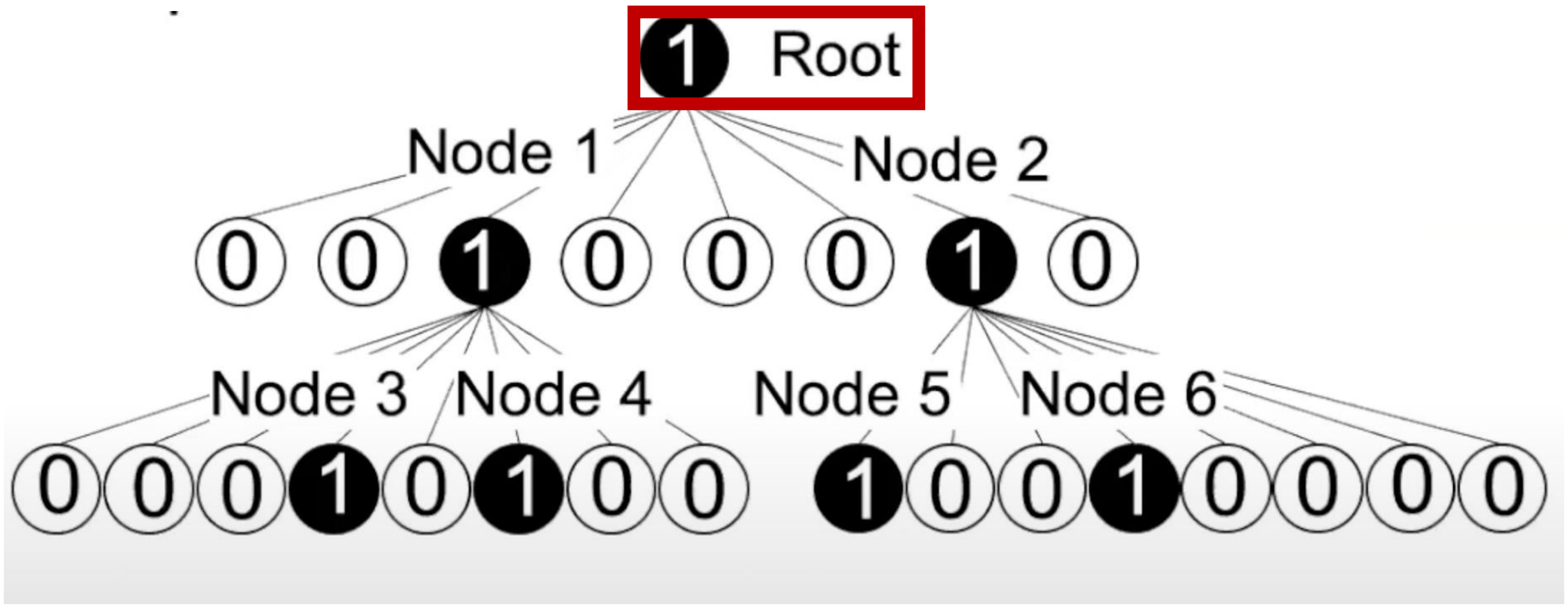
What are the x, y, z coordinates of the points in Node 6?



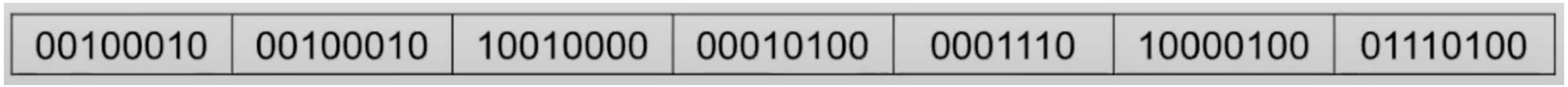
00100010	00100010	10010000	00010100	0001110	10000100	01110100

Compressed Byte Stream

MPEG GPCC



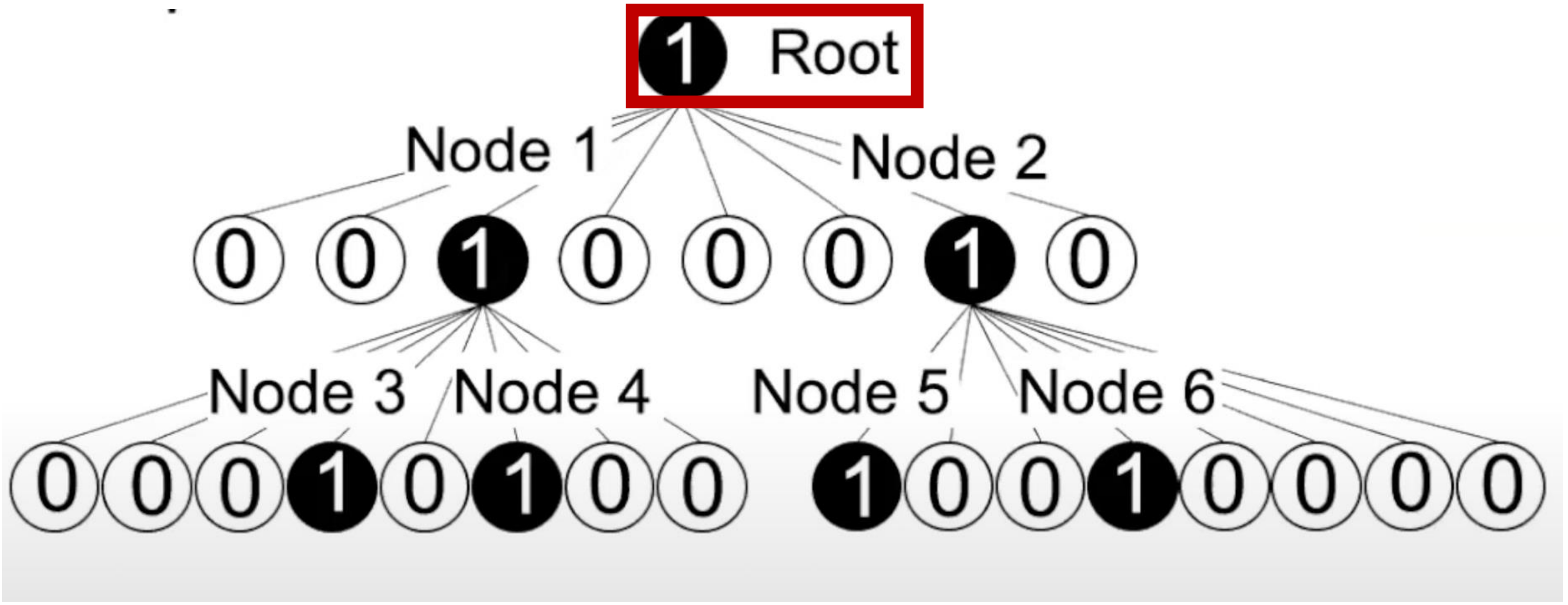
Root Node1 Node2



00100010	00100010	10010000	00010100	0001110	10000100	01110100

Compressed Byte Stream

MPEG GPCC

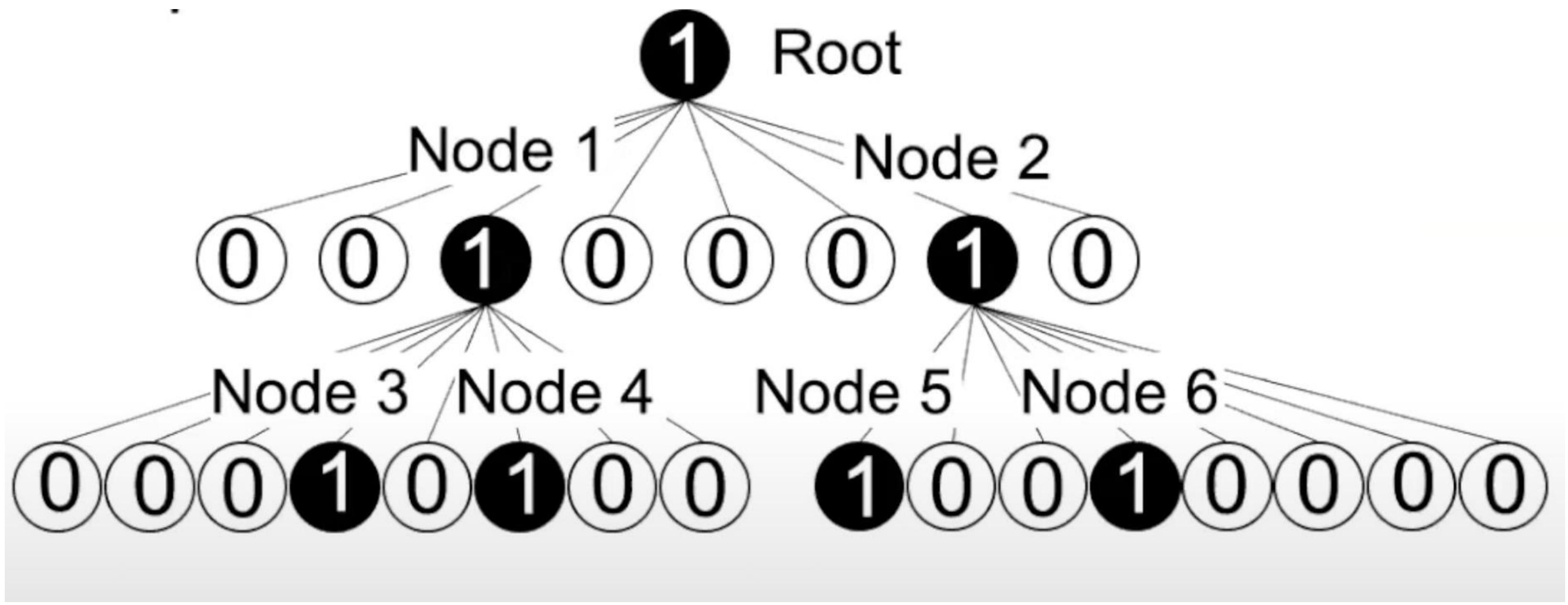


Which one is Node6?

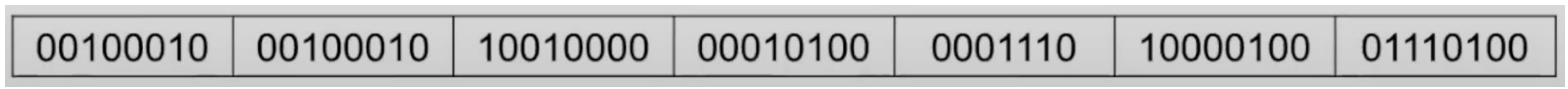
Root	Node1	Node2				
00100010	00100010	10010000	00010100	0001110	10000100	01110100

Compressed Byte Stream

MPEG GPCC



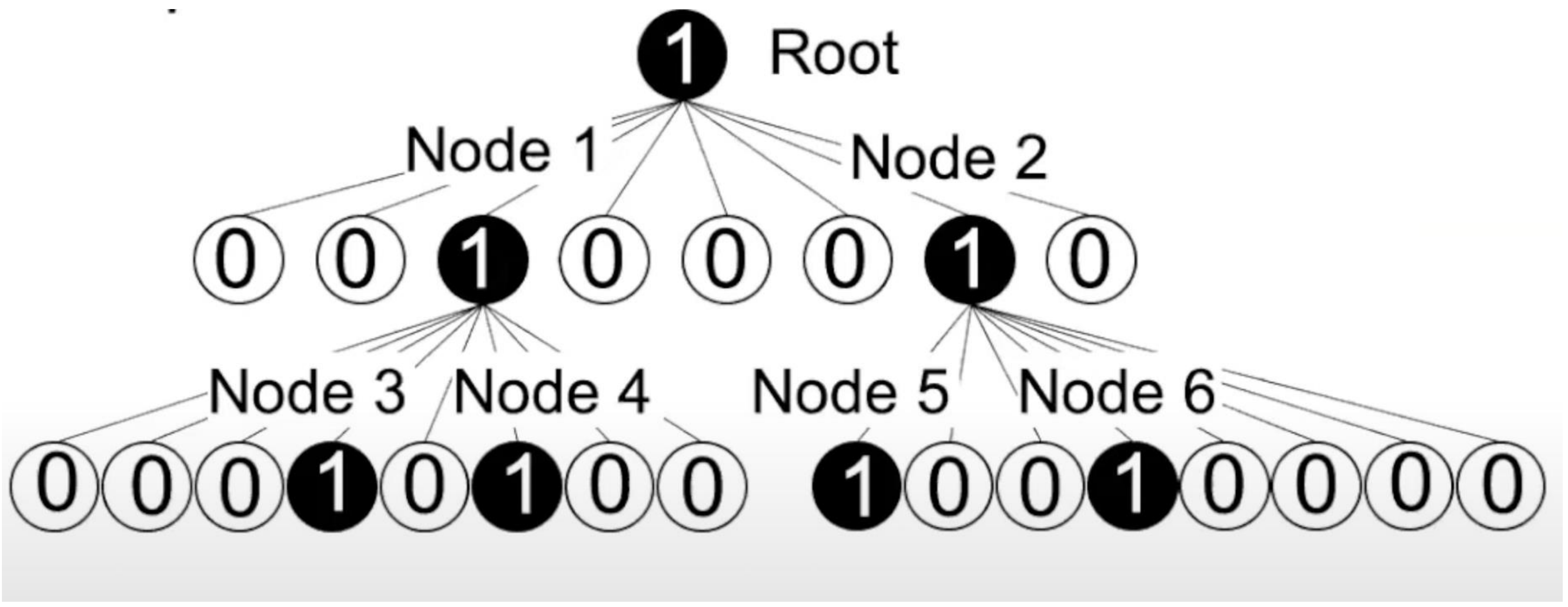
Root Node1 Node2 Node3 Node4



00100010	00100010	10010000	00010100	0001110	10000100	01110100

Compressed Byte Stream

MPEG GPCC

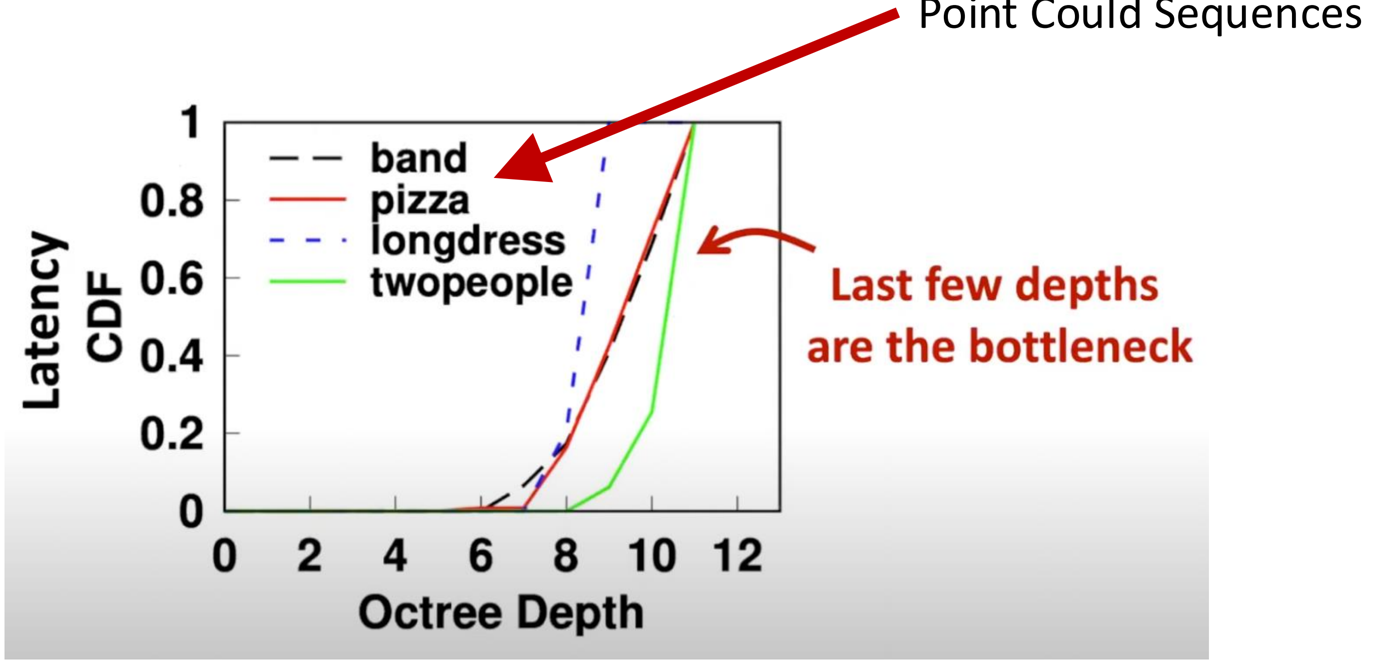


Root	Node1	Node2	Node3	Node4	Node5	Node6
00100010	00100010	10010000	00010100	0001110	10000100	01110100

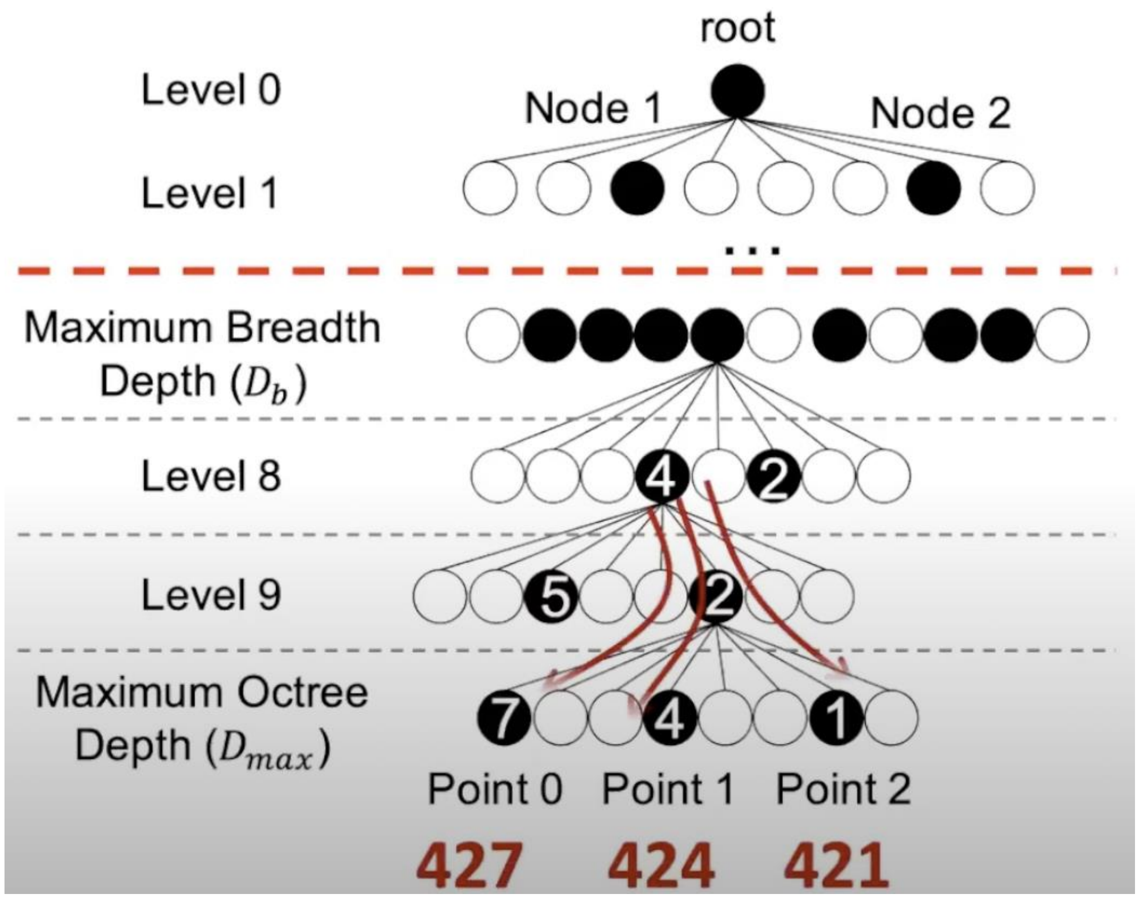
Compressed Byte Stream

MPEG GPCC

- Problem
  - Generates a dependency between the points – makes parallel processing difficult
  - Computationally expensive



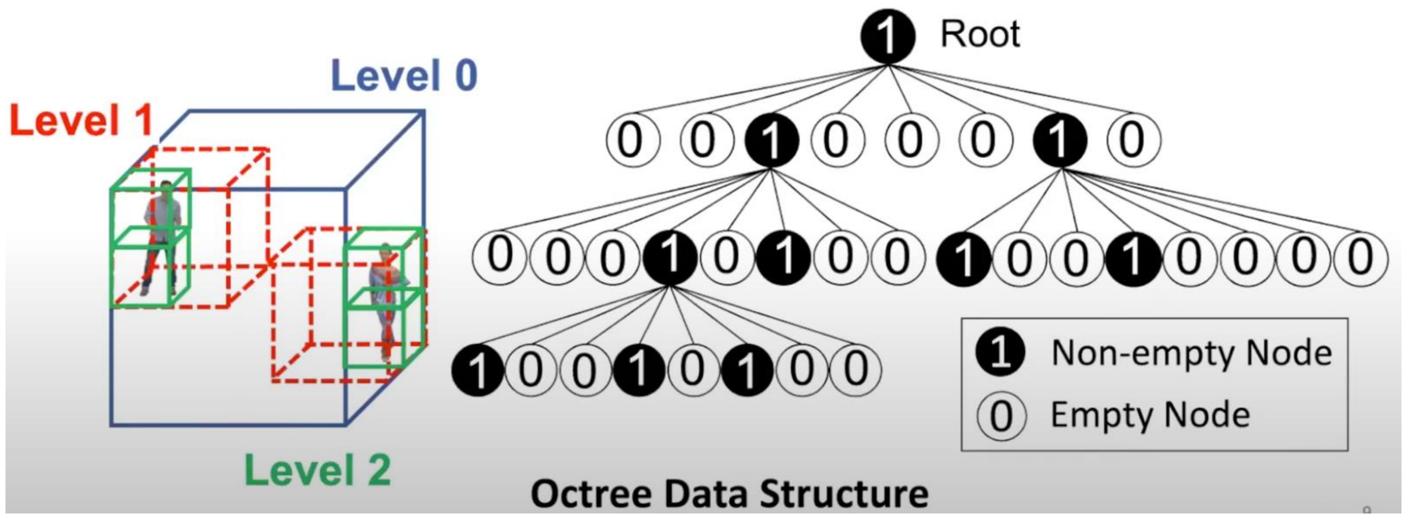
Parallel Decodable Tree



Break the dependency between points for parallel decoding

MPEG GPCC

- Problem
  - Points jump from one branch of the tree to another even with small motion or due to sensor noise
  - Great for static point cloud frames, but problematic temporally



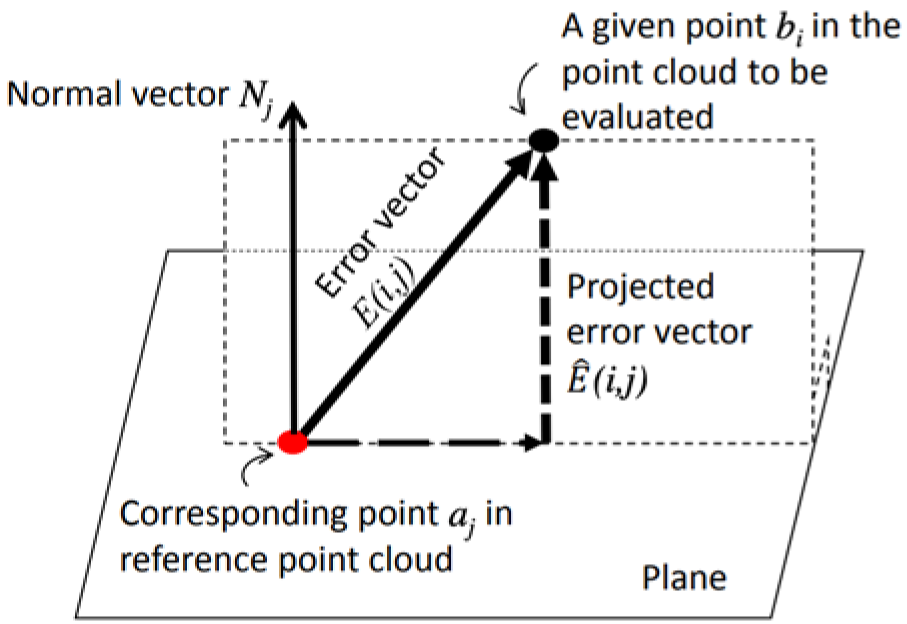
Point Cloud Error Metrics

- Point-to-Point
  - Error between nearest neighbor points

$$\frac{1}{N_A} \sum_{\forall a_j \in \mathbf{A}} \|E(i,j)\|_2^2$$

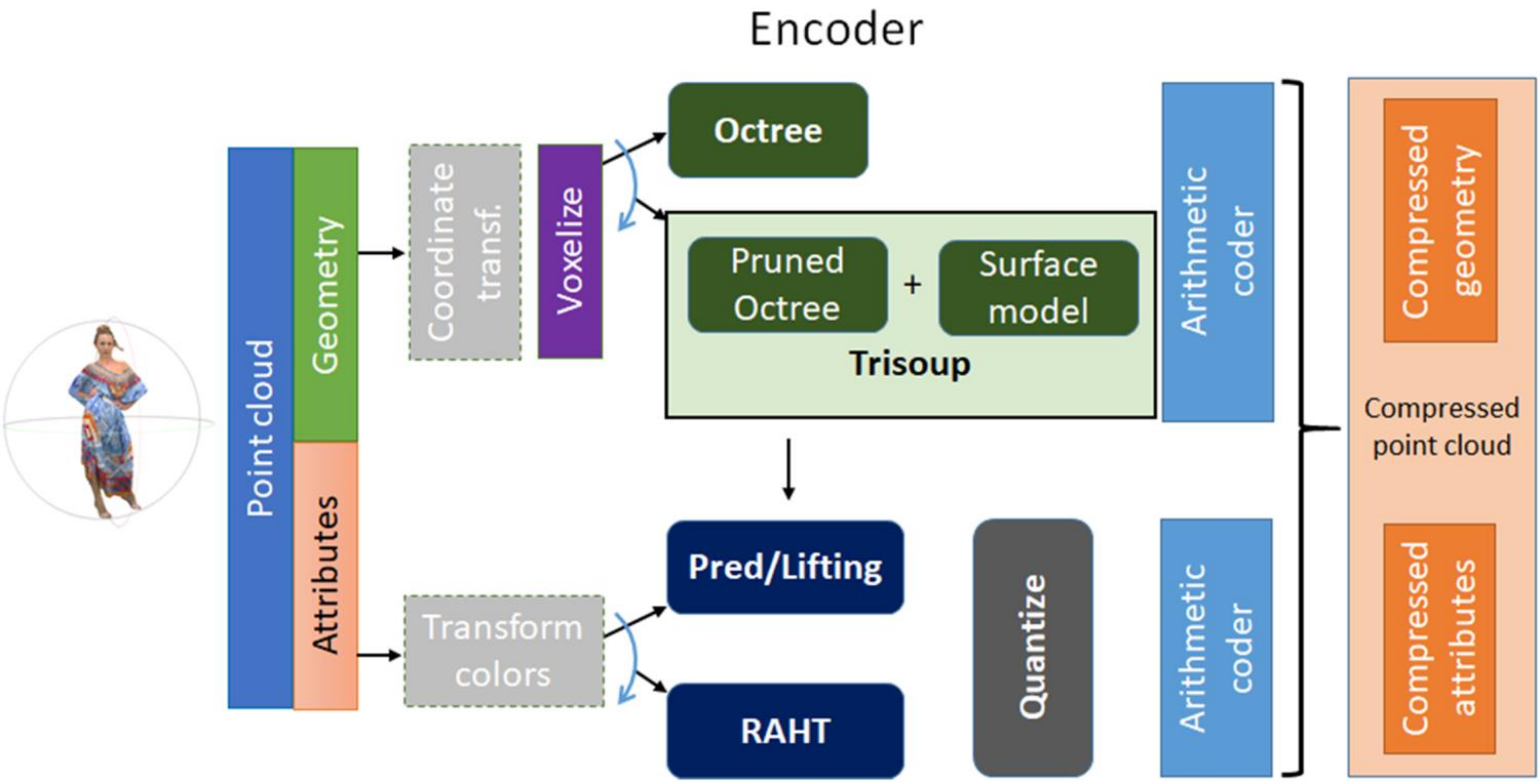
Point Cloud Error Metrics

- Point-to-Plane
  - Measures error along normal directions
  - More penalty on error that are away from surface



A given point b_i in the point cloud to be evaluated
Normal vector N_j
Error vector $E(i,j)$
Projected error vector $\hat{E}(i,j)$
Corresponding point a_j in reference point cloud
Plane

MPEG GPCC

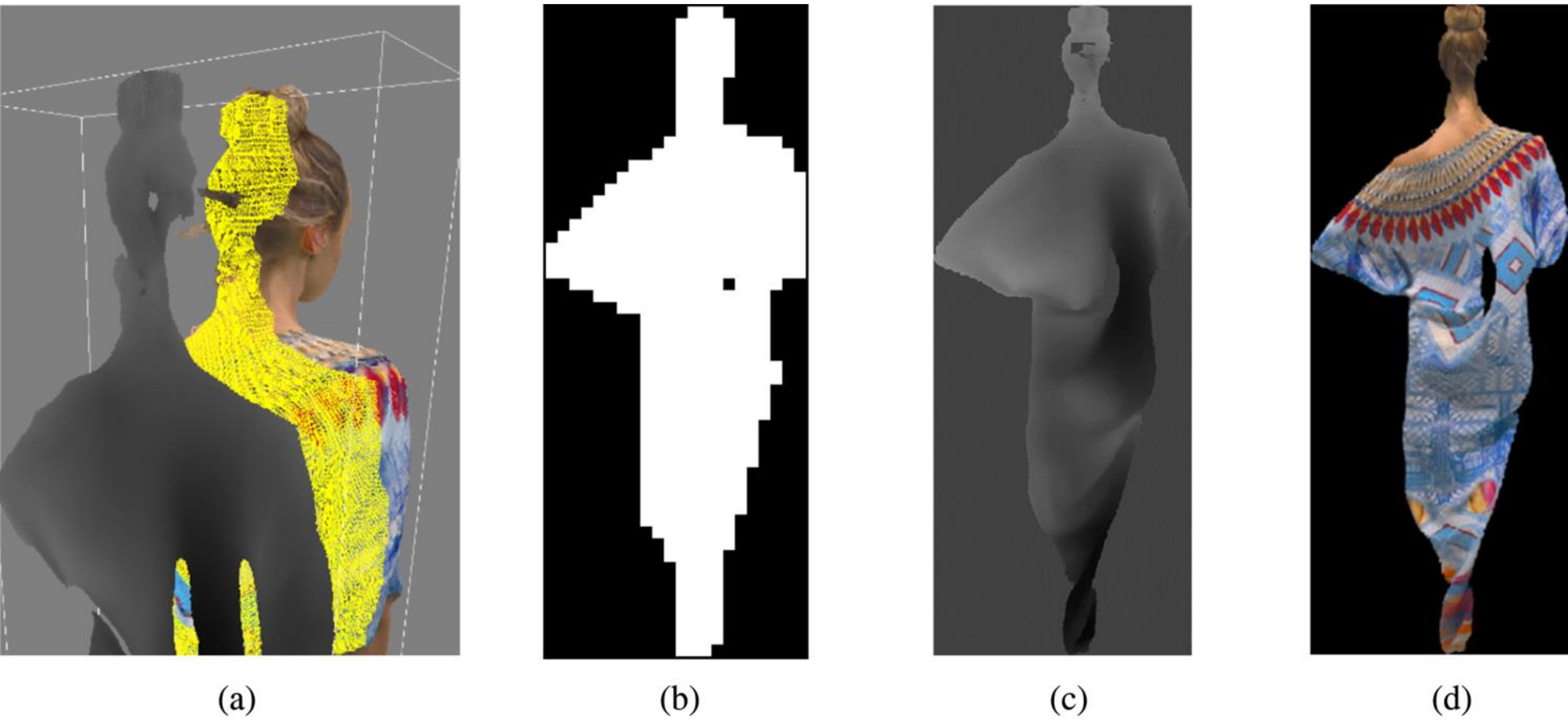


High level architecture

MPEG VPCC

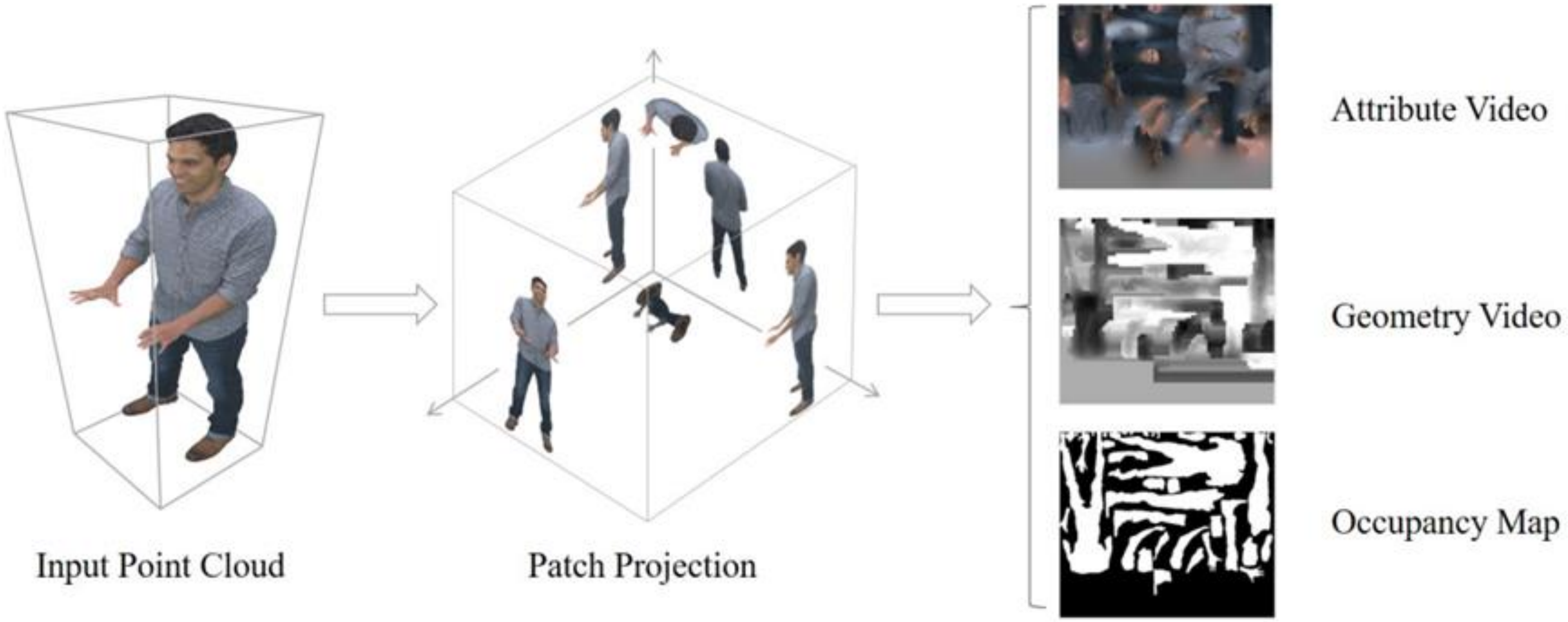
- Video based point cloud compression
  - Projection based coding – from 3D to 2D
  - Idea: Take advantage of existing video codecs to compress 2D projections

MPEG VPCC



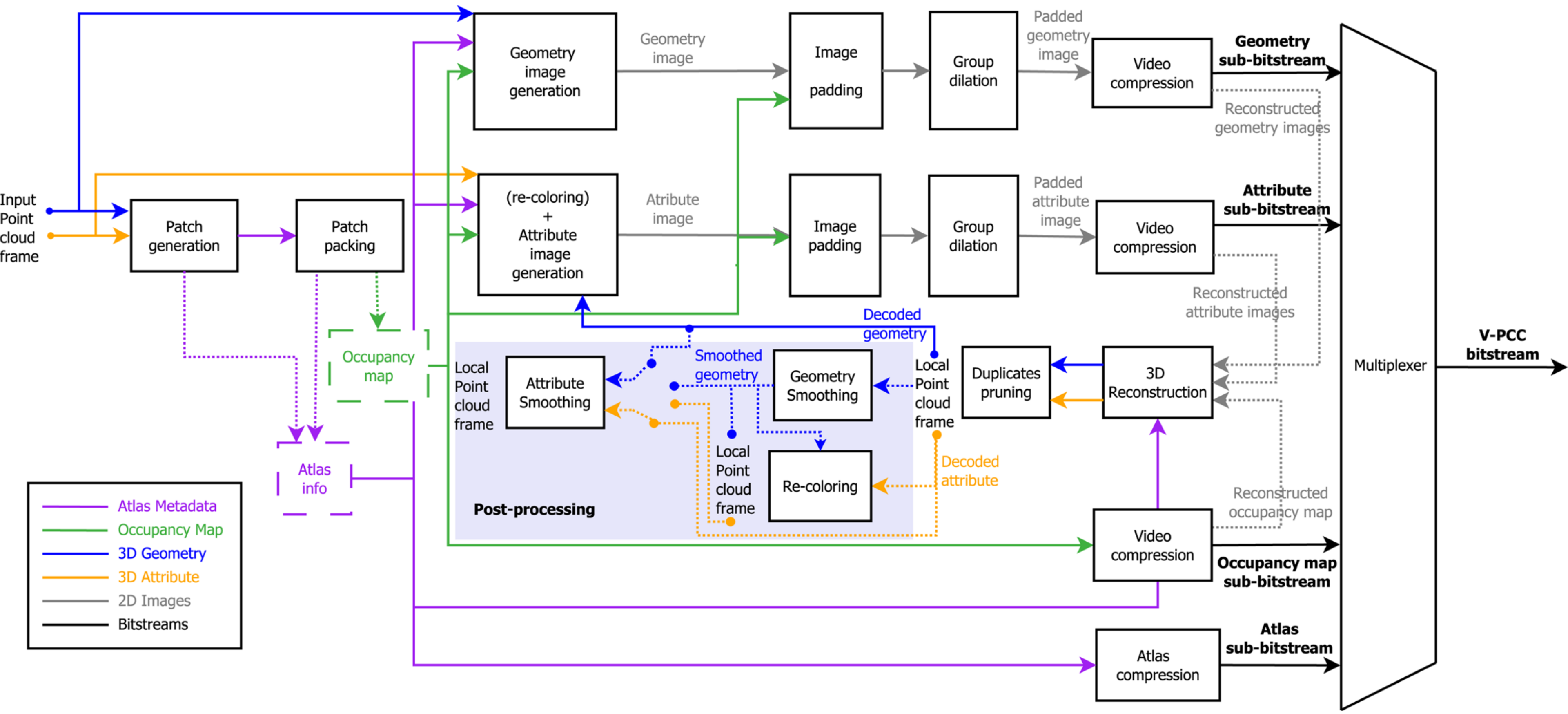
3D Patch projection and respective occupancy map, geometry, and attribute 2D images, (a) 3D patch, (b) 3D Patch Occupancy Map, (c) 3D Patch Geometry Image, (d) 3D Patch Texture Image.

MPEG VPCC



3 Video streams + 1 additional meta data stream

MPEG VPCC



High level architecture

MPEG VPCC

- Problem
  - Computationally expensive – Patch generation, packing, video generation, compression (4 streams)

Compressing Large Scale Point Clouds

- Both GPCC and VPCC suffer
  - Need to rely on the other forms of data structures for efficiency

Summary of the Lecture

- Depth Map Compression
- MPEG GPCC
- MPEG VPCC
  - Both are computationally expensive – unusable at this point for practical purposes

Next up: Mesh Compression