

EECE5698

Networked XR Systems

Lecture Outline for Today

XR Visual Data Structures

- 2D Videos
- Stereo Videos
- 2D 360 Degree Videos
- 3D Videos
- Multi-view 2D Videos
- Implicit Neural Representations

2D Videos

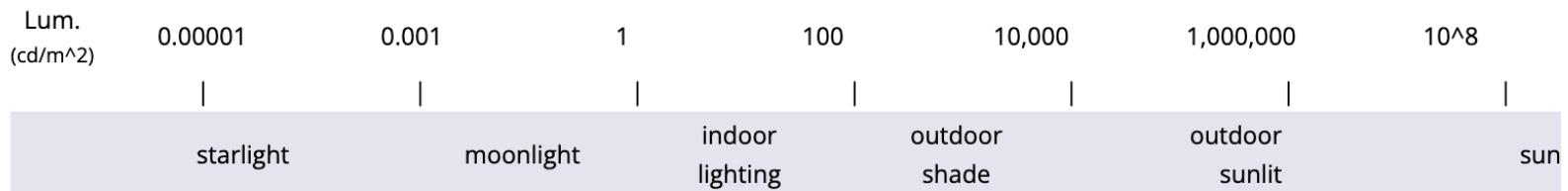
- Fixed data structure
- Color attributes
 - 3 Channels
 - 8 bits each channel
- Color Space Formats
 - RGB – 24 bits
 - YUV (Luma & Chroma)
 - YUV420 – 12 bits
 - YUV422 – 16 bits
 - YUV444 – 24 bits



1080x1920

2D HDR Videos

- High dynamic range – here range is light intensity
- This means that bright objects and dark objects on the same screen can be shown to high degrees of brightness and darkness if the display supports it
- High data rate: 96 bits per pixel, 32 bits per channel
- Dolby vision vs. HDR10
 - Dynamic & Static HDR – for display



2D Videos

- Limitations
 - Not immersive (enough)
 - Cannot pan, tilt or zoom
 - Not interactive
 - Users are passive observers
 - Limited FoV
 - Viewers can only see what the camera captures

Stereo Videos

- Captured using two cameras
 - one per each eye
- Provides depth perception
- More realistic than regular (monocular) 2D videos



2D 360 Degree Videos

- Pixels are similar to 2D videos
 - RGB or YUV channels
 - Captured using an omnidirectional camera or a collection of cameras.
 - Typically, 360 degrees horizontal, 180 degrees vertical
 - During the playback, the user views a particular viewport

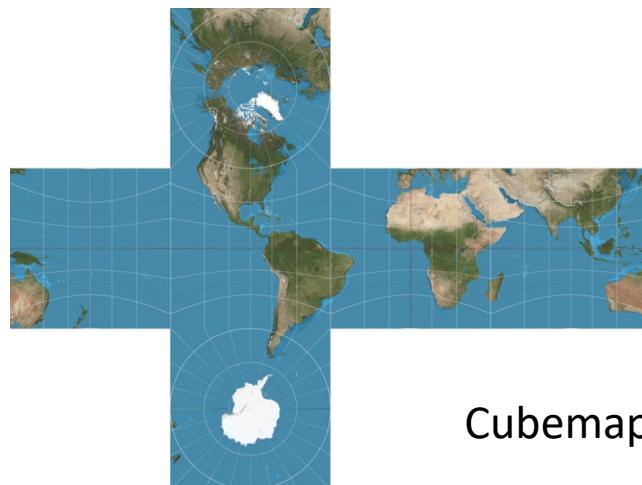
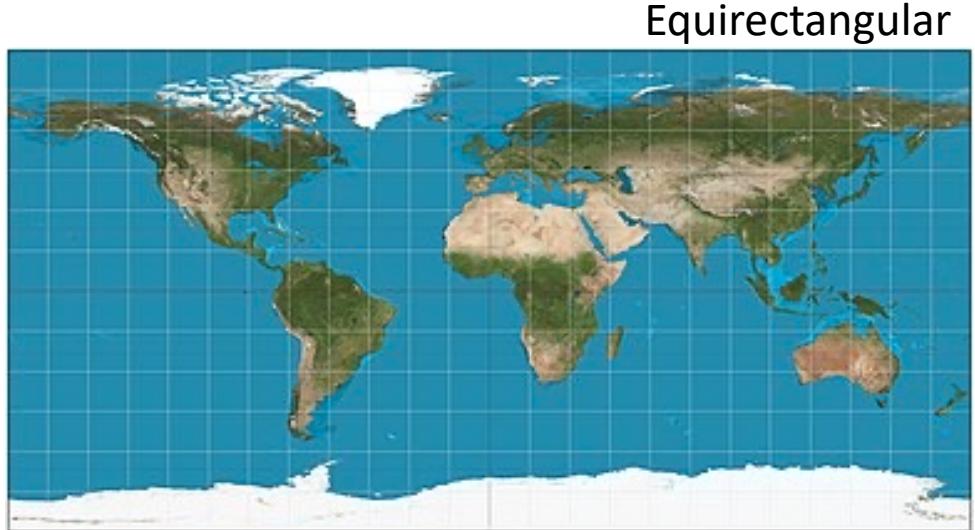


3 degrees of freedom

2D 360 Degree Videos

- Format

- Cubmaps are efficient for hardware
- A lot of pixels are repeated (e.g., top row) for equirectangular format

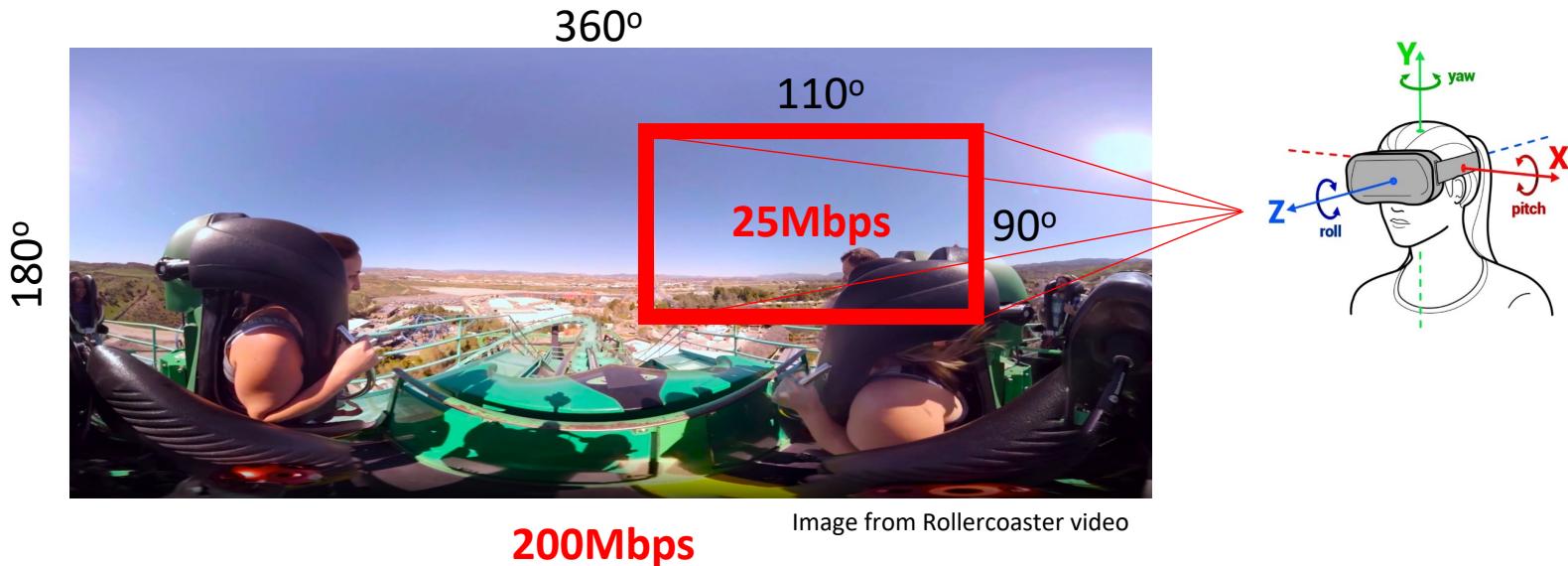


2D 360 Degree Videos

- Display devices
 - Most displays like personal computers, phones, headsets
 - Phones use gyroscope to move around the scene

2D 360 Degree Videos

- Limitations
 - Limited interaction – no translation
 - A well-known problem – bandwidth inefficient



3D 360 Degree Videos

- Allows true 6 degrees of freedom
 - Translation, and Rotation
- Also known as 4D scene – $x, y, z + t$
- Objects or spaces
- Typically requires multiple cameras to capture 3D videos
- Allows interaction

3D Videos

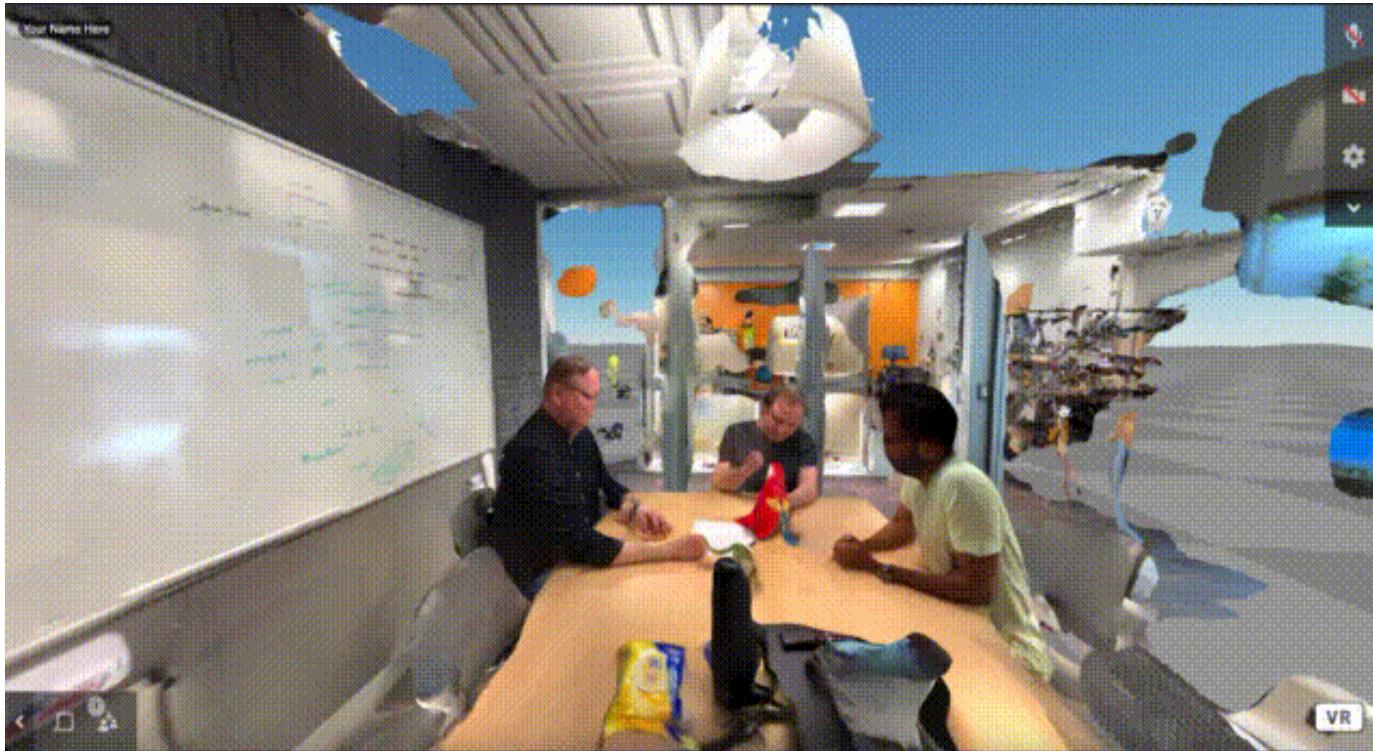
- Objects



Hologram

3D Videos

- Spaces



3D Videos

- Data Representations
 - Depth Maps
 - Point Clouds
 - Meshes

Depth Maps

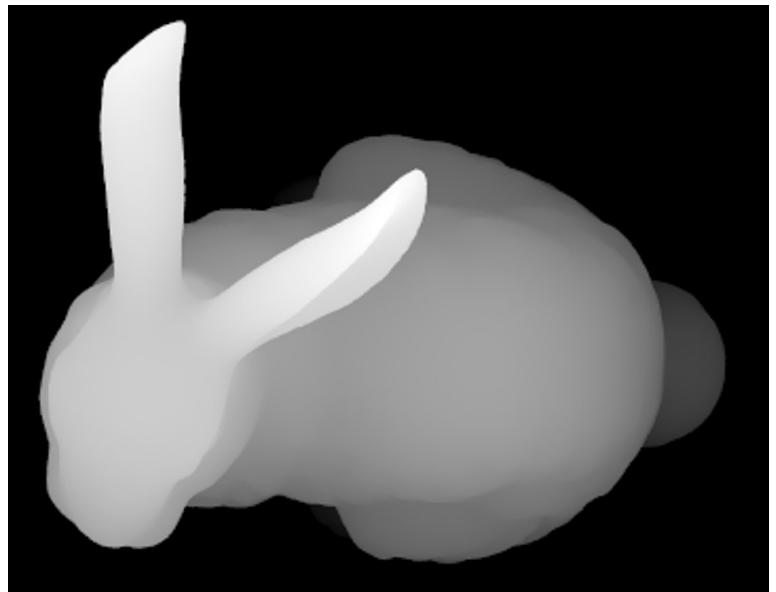
- Depth maps contain information about the distance of objects from a specific perspective or reference point (like a camera lens).
- Each pixel is assigned a value to represent the distance of that pixel from the reference point which creates a 3D representation of the scene for its RGB image or virtual scene.

Depth Maps

- Captured using
 - Depth sensors
 - Stereo Triangulation
 - ToF, Structure light
 - 3D modelling
 - Computer Vision or ML (depth estimation)

Depth Maps

- Typically, the white pixels represent the part of the scene that is closest to the camera lens, and the black pixels represent the part of the scene that is furthest.
- But there's no set standard how to represent the map



Bunny depth map

Depth Maps

- Images – but how many channels?
 - Do we need 3 channels like RGB?
- Bit depth
 - One channel - depth
 - 8 bits – range up to 256 (any unit like meters or cm/mm)
 - 16 bits – range up to 2^{16} units etc

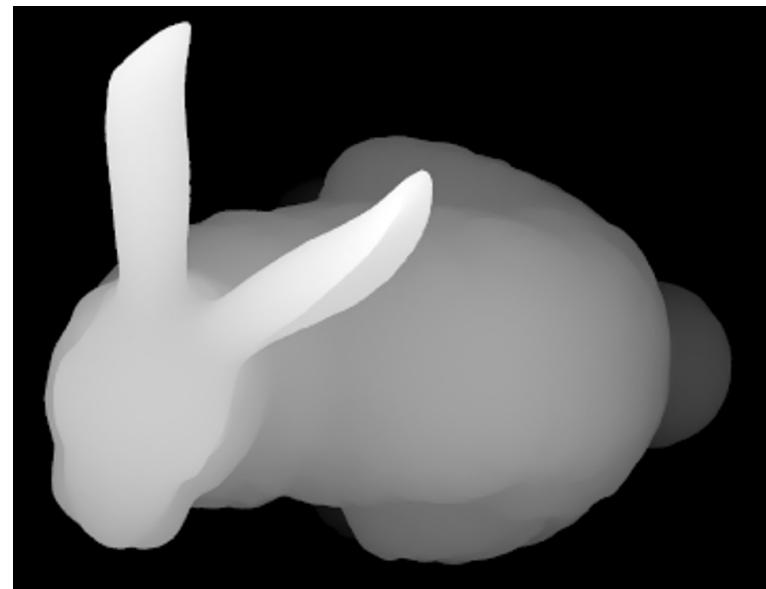
Depth Maps

- Popular Depth Camera or Sensor
 - Depth can be captured at longer ranges, up to 20m.
 - Frame rate of depth capture can be as high as 100 FPS.
 - Field of view, up to 110° (H) x 70° (V).
 - The camera works indoors and outdoors, contrary to active sensors such as structured-light or time of flight.
 - Stereo triangulation



Depth Maps

- Limitations of this representation
 - Fixed size data structure (i.e., image representation)
 - Inefficient storage of depth
 - Most pixels are not occupied



Point Cloud

- 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

- Captured using
 - Regular 2D camera array - Photogrammetry
 - Depth sensing - LiDAR scanning, Time of Flight
 - 3D modelling

Point Cloud

- File format (how it is stored in a file)
- .ply

The diagram illustrates the structure of a .ply file, specifically for a point cloud. It shows the header, vertex properties, face properties, vertex coordinates, and face definitions.

Header begins: ply
format ascii 1.0

Number of vertices: element vertex 6
property float x
property float y
property float z
property uchar red
property uchar green
property uchar blue

Number of faces: element face 8
property list uchar int vertex_index
end_header

Vertex coordinates (x,y,z): 1.000000 0.000000 0.000000 255 0 0
0.000000 1.000000 0.000000 0 255 0
0.000000 0.000000 1.000000 0 0 0
-1.000000 0.000000 0.000000 0 0 255
0.000000 -1.000000 0.000000 255 0 255
0.000000 0.000000 -1.000000 255 255 255

Number of vertices that define this face: 3 0 1 2
3 1 3 2
3 3 4 2
3 4 0 2
3 1 0 5
3 3 1 5
3 4 3 5
3 0 4 5

Annotations:

- Vertex coordinates data type
- Data type for color property
- How the vertex indices are listed to define faces
- RGB triplet to define vertex colors (0 to 255)
- Definition of faces from vertex indices (1st vertex is 0)

Point Cloud

- Representation
 - Each Point is a floating-point number – 32 bits
 - $\langle X, Y, Z \rangle$: 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

- What is the file size of the point cloud that has 5 million points?

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

Point Cloud

- Popular sensor – laser scanning
 - 830-grams
 - 100m Range
 - 300,000 Points per Second
 - 360° Horizontal FOV
 - $30^\circ \pm 15^\circ$ Vertical FOV
- Costly (>\$10,000)



Point Cloud vs Depth Map

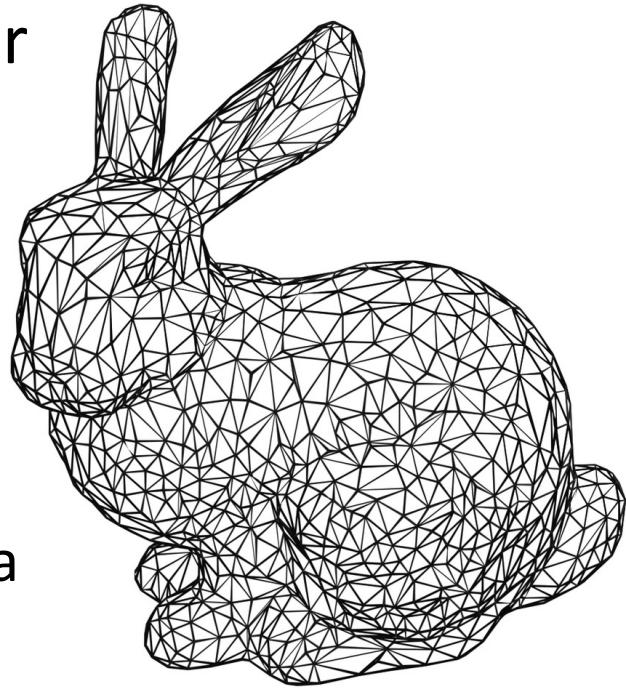
- Both are depth data structures
- A depth map is a 2d image with depth. It only shows the nearest point for each pixel from the direction its oriented.
- A point cloud is a bunch of xyz points, they can be in front of other points.
- Depth map size is fixed while point cloud size varies over time
- Depth map is depth only, while point clouds are often baked with color texture information

Point Cloud

- Limitations
 - Arbitrary data structure
 - Changes number of points in two consecutive frames
 - Creates problems during compression
 - Requires high bandwidth to represent objects or spaces
 - Lacks knowledge of surfaces or requires huge number of points to represent a surface

Mesh

- A set of polygons, connected by their common edges or vertices
- Typically represented by triangles
- Why triangle? Why not other polygons?
 - By definition, a triangle always lies on a single plane, providing a flat surface.
 - This planarity ensures there aren't any distortions when rendering a triangle, making it reliable for building complex 3D shapes.



Mesh

- Capturing mesh data structure
 - No native support from sensors
 - Need to extract mesh polygons from depth maps or point clouds

Mesh

- Data representation
 - Each frame has vertices and connectivity
 - Size depends on file format – next slide
 - Color texture is stored independently, so there is also mapping information from texture to polygons



Mesh

- File format - .ply

The diagram illustrates the structure of a .ply file, which is a plain text-based mesh format. The file begins with a header section containing metadata about the vertex and face elements, followed by vertex coordinates and their corresponding colors, and finally the definitions of the faces.

Header begins: ply
format ascii 1.0

Number of vertices: element vertex 6
property float x
property float y
property float z
property uchar red
property uchar green
property uchar blue

Number of faces: element face 8
property list uchar int vertex_index
end_header

Vertex coordinates (x,y,z): 1.000000 0.000000 0.000000 255 0 0
0.000000 1.000000 0.000000 0 255 0
0.000000 0.000000 1.000000 0 0 0
-1.000000 0.000000 0.000000 0 0 255
0.000000 -1.000000 0.000000 255 0 255
0.000000 0.000000 -1.000000 255 255 255

Number of vertices that define this face: 3 0 1 2
3 1 3 2
3 3 4 2
3 4 0 2
3 1 0 5
3 3 1 5
3 4 3 5
3 0 4 5

Annotations:

- Vertex coordinates data type:** Arrows point from the 'float' and 'uchar' properties to this label.
- Data type for color property:** Arrows point from the 'red', 'green', and 'blue' properties to this label.
- How the vertex indices are listed to define faces:** Arrows point from the 'list uchar int vertex_index' property to this label.
- RGB triplet to define vertex colors (0 to 255):** Arrows point from the color values (e.g., 255 0 0) to this label.
- Definition of faces from vertex indices (1st vertex is 0):** Arrows point from the face vertex index lists to this label.

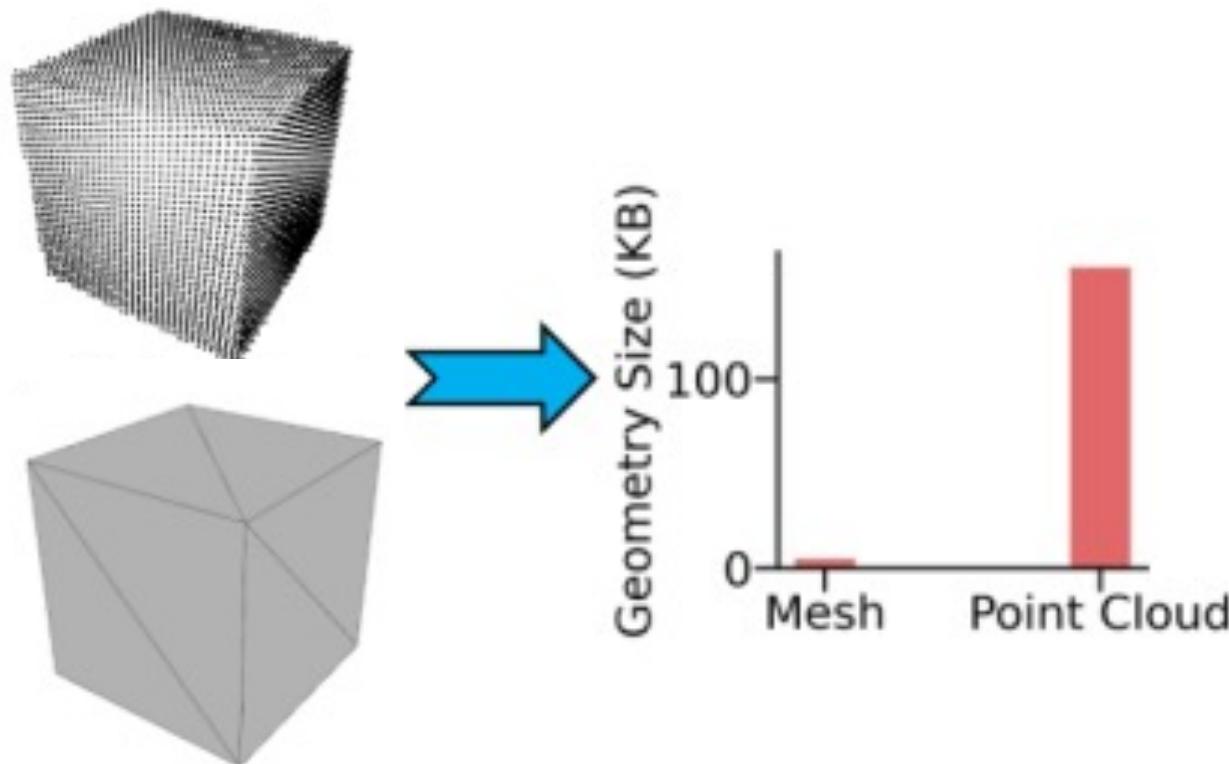
Mesh

- File format - .obj

```
# List of geometric vertices, with (x, y, z, [w]) coordinates, w is optional and defaults to 1.0.
v 0.123 0.234 0.345 1.0
v ...
...
# List of texture coordinates, in (u, [v, w]) coordinates, these will vary between 0 and 1. v, w are optional
# and default to 0.
vt 0.500 1 [0]
vt ...
...
# List of vertex normals in (x,y,z) form; normals might not be unit vectors.
vn 0.707 0.000 0.707
vn ...
...
# Parameter space vertices in (u, [v, w]) form; free form geometry statement (see below)
vp 0.310000 3.210000 2.100000
vp ...
...
# Polygonal face element (see below)
f 1 2 3
f 3/1 4/2 5/3
f 6/4/1 3/5/3 7/6/5
f 7//1 8//2 9//3
f ...
...
# Line element (see below)
l 5 8 1 2 4 9
```

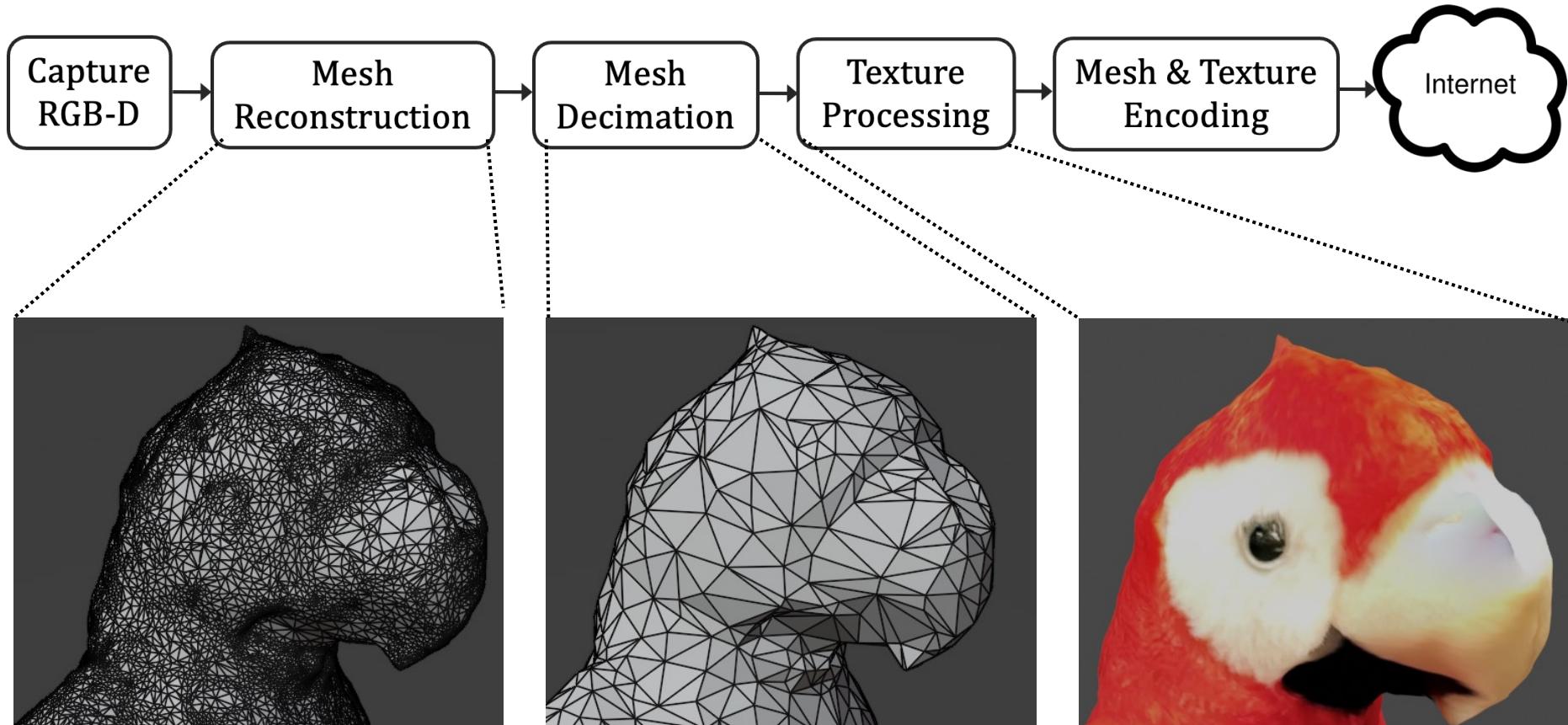
Mesh vs. Point Cloud

- Meshes are much more compact



Mesh

- But extracting meshes is computationally intensive task, unlike point clouds that are readily available



Recap: 3D Data Structures

- Depth Map vs. Point Cloud vs. Mesh
 - Depth maps and point clouds are simple, easy to manipulate, quickly available
 - Meshes are compact and requires significantly less bandwidth, but are computationally heavy to extract
 - Depth maps are fixed in size while the other two have arbitrary sizes
 - Meshes define surfaces while the other two not
 - Meshes are approximate 3D data structures while the other two represent accurate points

View Immersion

- Monocular
- Stereoscopic
- Multi-view

View Immersion

- Mono or monocular
 - Single camera
 - Simple, low cost
- Limitations
 - No depth perception



View Immersion

- Stereo or Stereoscopic
 - 2 cameras
- Depth perception depends on the baseline
- Limited by small field of view



Apple spatial videos

View Immersion

- Multi-view videos
 - Typically, tens to hundreds of cameras are deployed to get full 3D 360° view of the scene of interest
 - Highest level of immersion
 - Costly
 - Very infra heavy
 - Bandwidth heavy
 - Compute heavy
 - Hard (almost impossible) to get in real-time/live



Implicit Neural Representation

- A fully-connected neural network that can generate novel views of complex 3D scenes, based on a partial set of 2D images.
- Set of weights
- To render a view, need to query the neural network by inputting the pose info

<https://www.matthewtancik.com/nerf>

Summary of the Lecture

- XR Data Structures
 - 2D videos
 - 360° videos
 - 3D videos
 - Depth Map
 - Point Cloud
 - Mesh
 - View-immersion
 - Mono
 - Stereo
 - Multi-view
 - Implicit neural representations