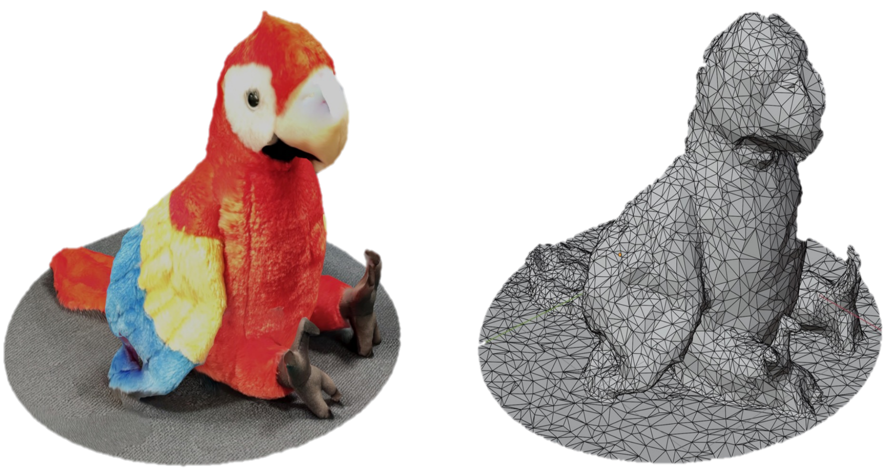
EECE5698
Networked XR Systems

Lecture Outline for Today

- Mesh Streaming
- Homework3

Recap: Mesh

- Data representation
  - Each frame has vertices and connectivity
  - Color texture is stored independently, so there is also mapping information from texture to polygons

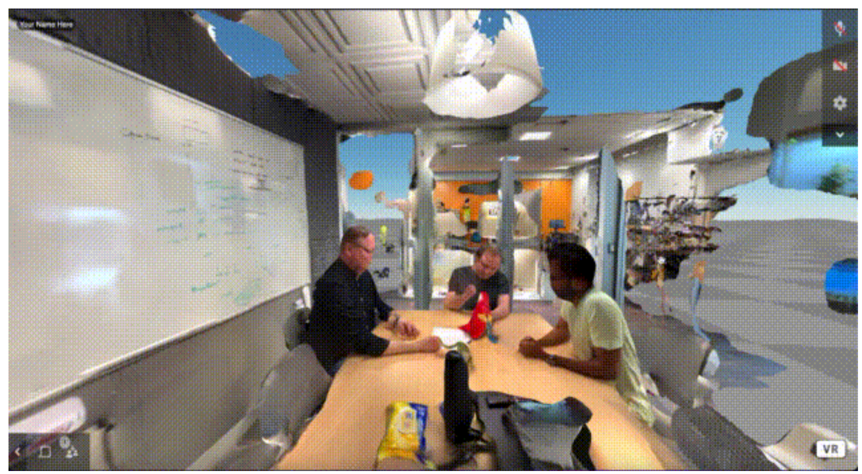


Challenges in Mesh Streaming

- **Low Latency**: For interactive live streaming, we expect latencies on the order of 2D video conferencing systems (<100ms).
- **Scalable**: 3D video quality is often a function of number of cameras and scene size. An ideal capture solution should support dozens of sensors with commodity hardware.
- **Adaptive Streaming**: The system must operate given practical bitrates for Internet streaming, and the quality of the system should adapt to bandwidth availability.

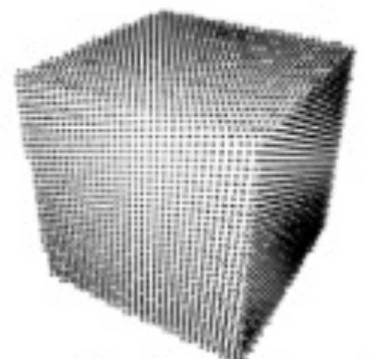
Mesh Streaming

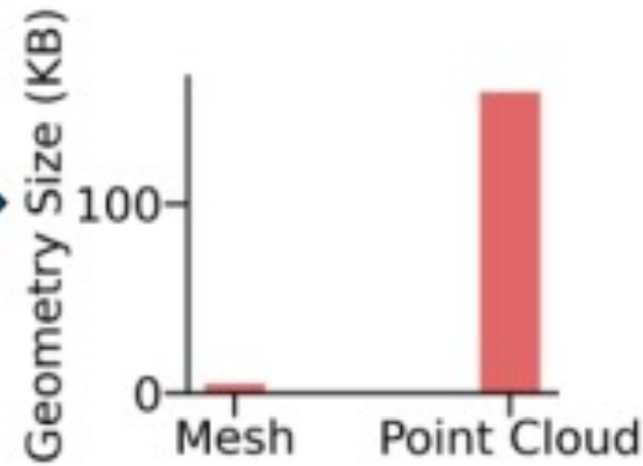
- Bandwidth and latency are also a function of the scale of the scene captured.



Implications of 3D Representations

Point Cloud Representation



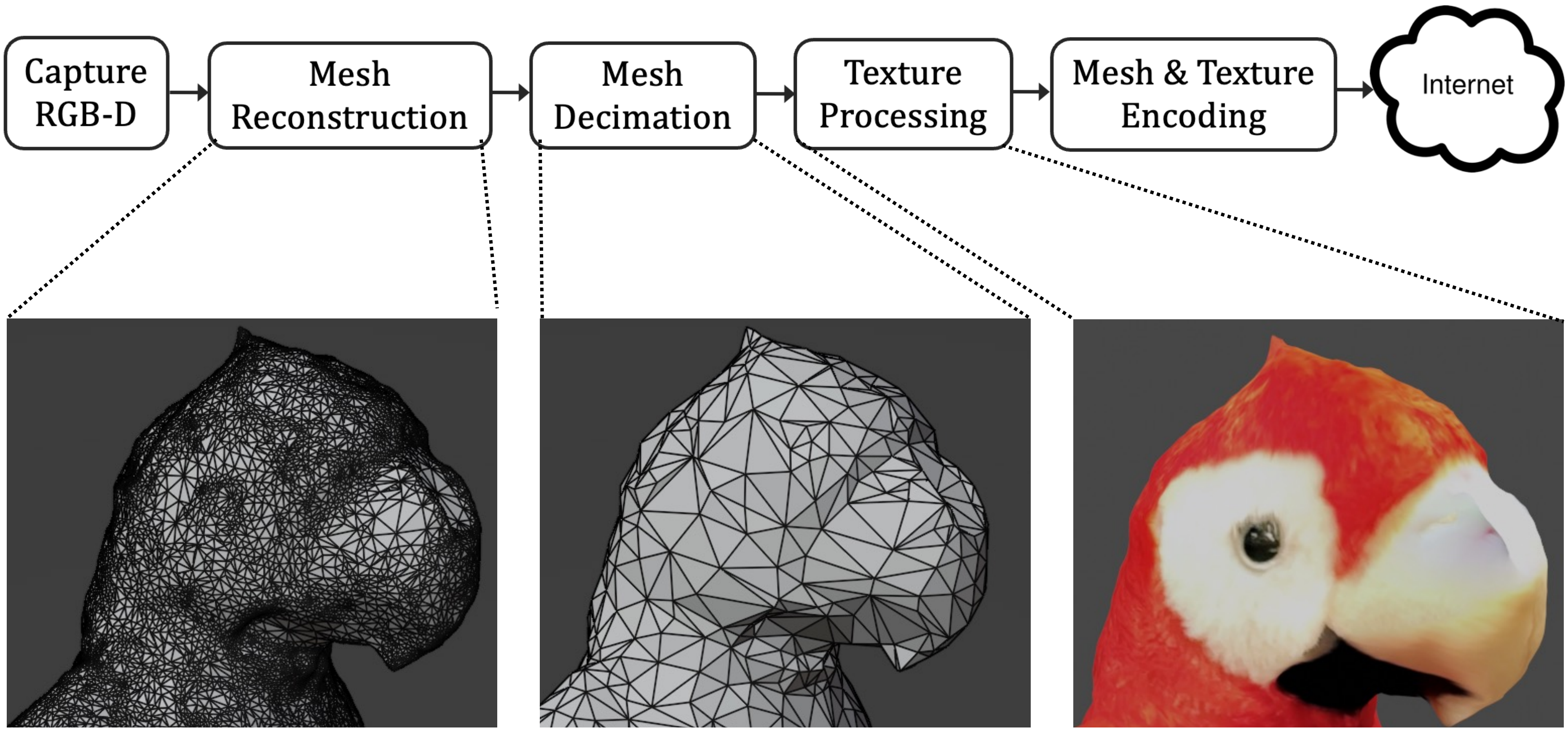


Mesh Representation

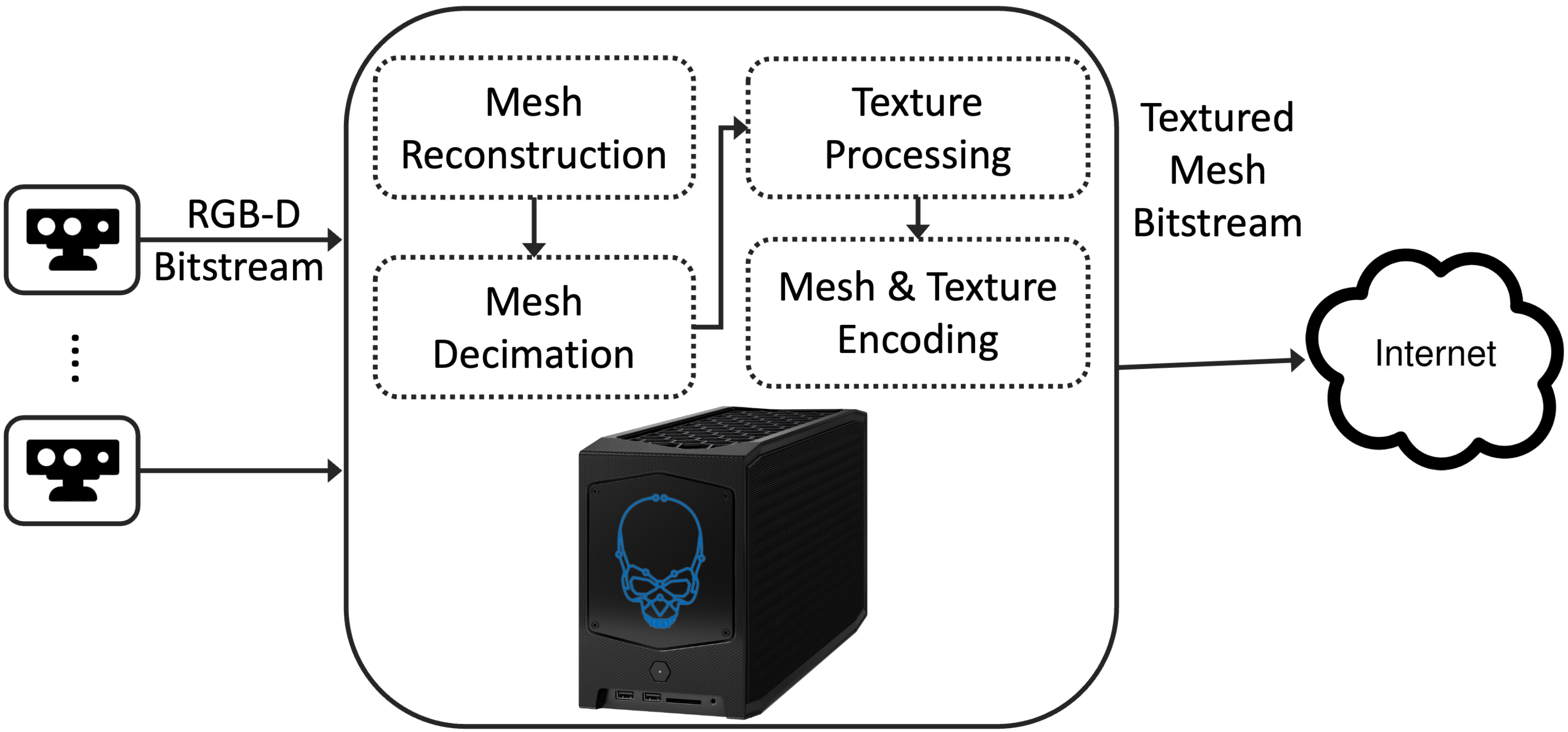
Mesh is not natively available in sensors

Mesh requires significantly less data rate but poses numerous computational challenges

Mesh Capture

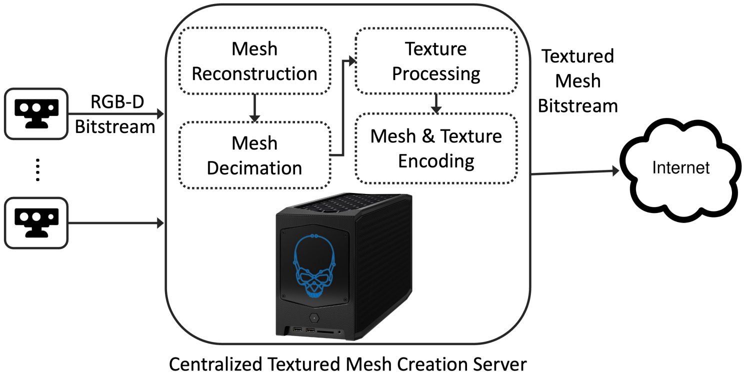


Mesh Capture

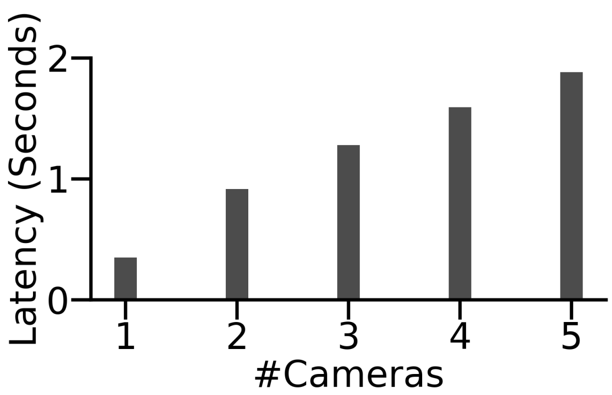


Centralized Textured Mesh Creation Server

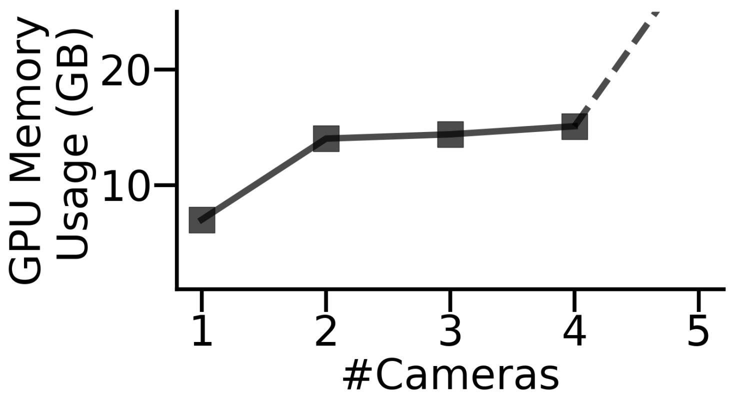
Mesh Capture



Decimation is computationally expensive task

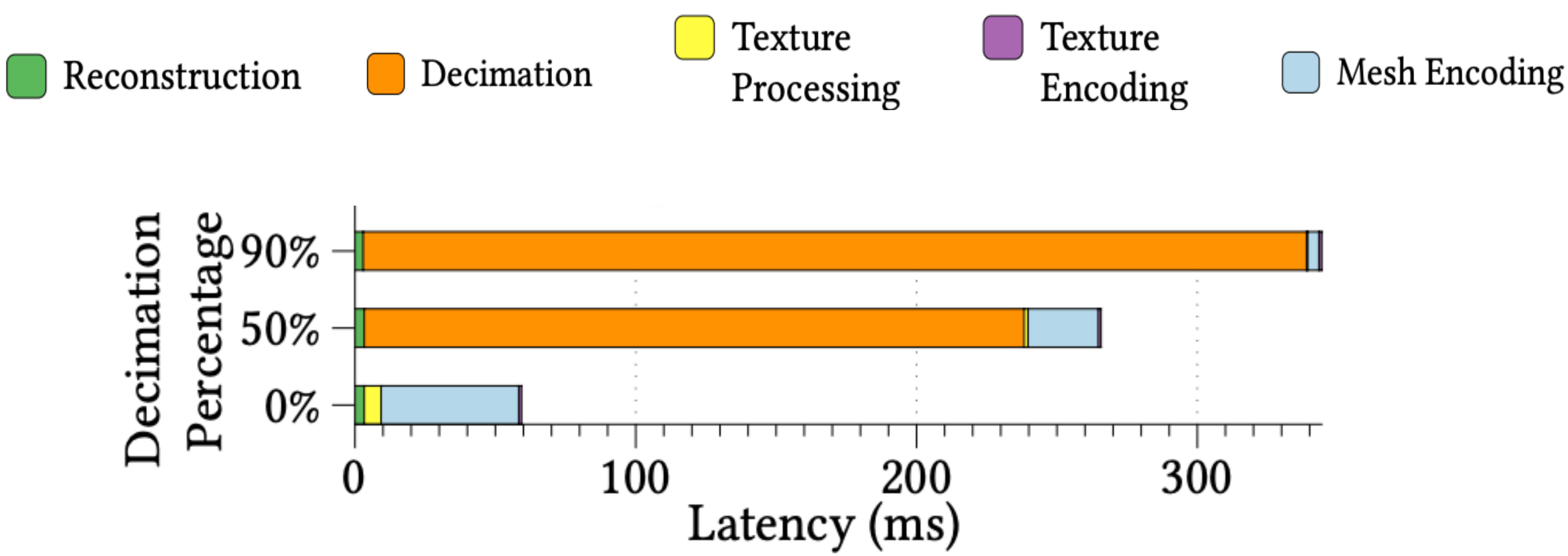


Reconstruction is GPU memory intensive



Not scalable with #cameras, scene size, and high quality

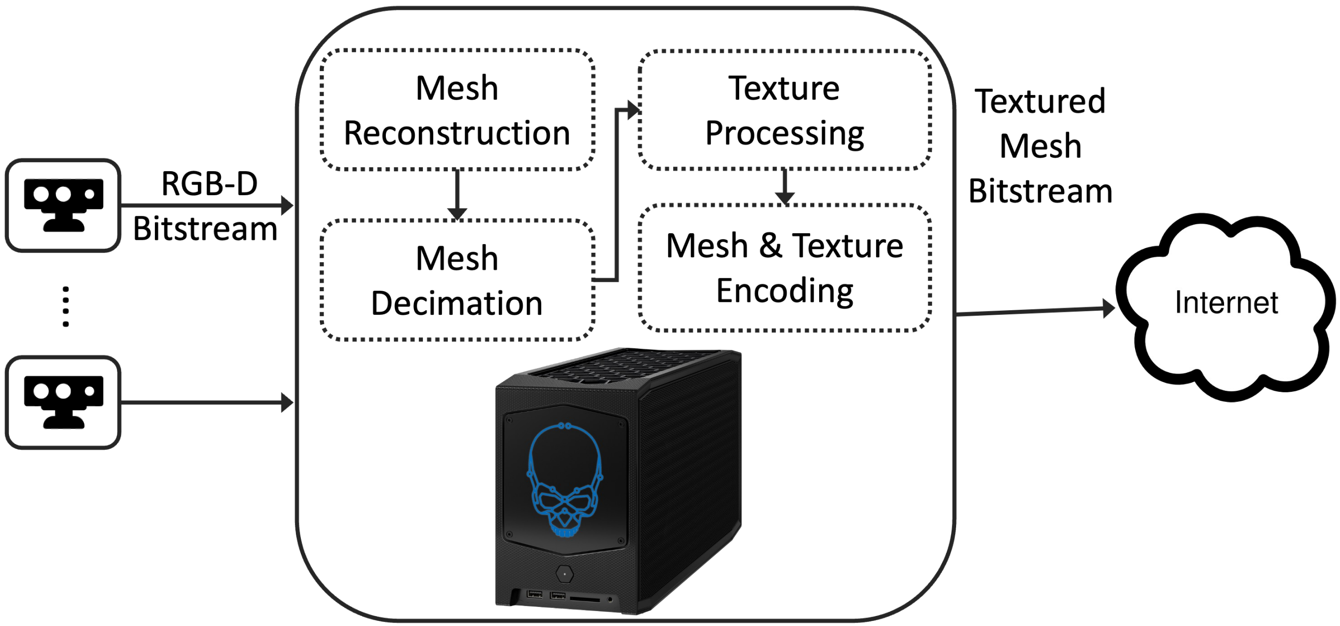
Root Cause Analysis



Decimation is computationally expensive task

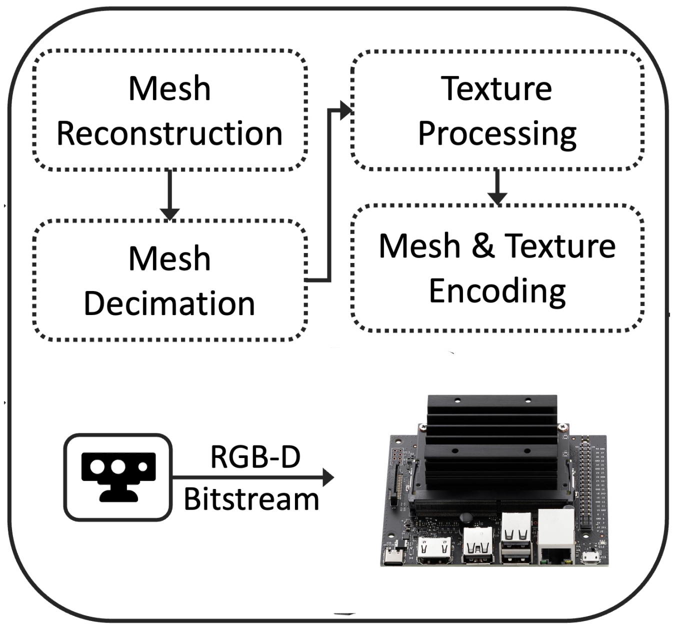
Reconstruction is fast but GPU memory intensive

Distributed Capture



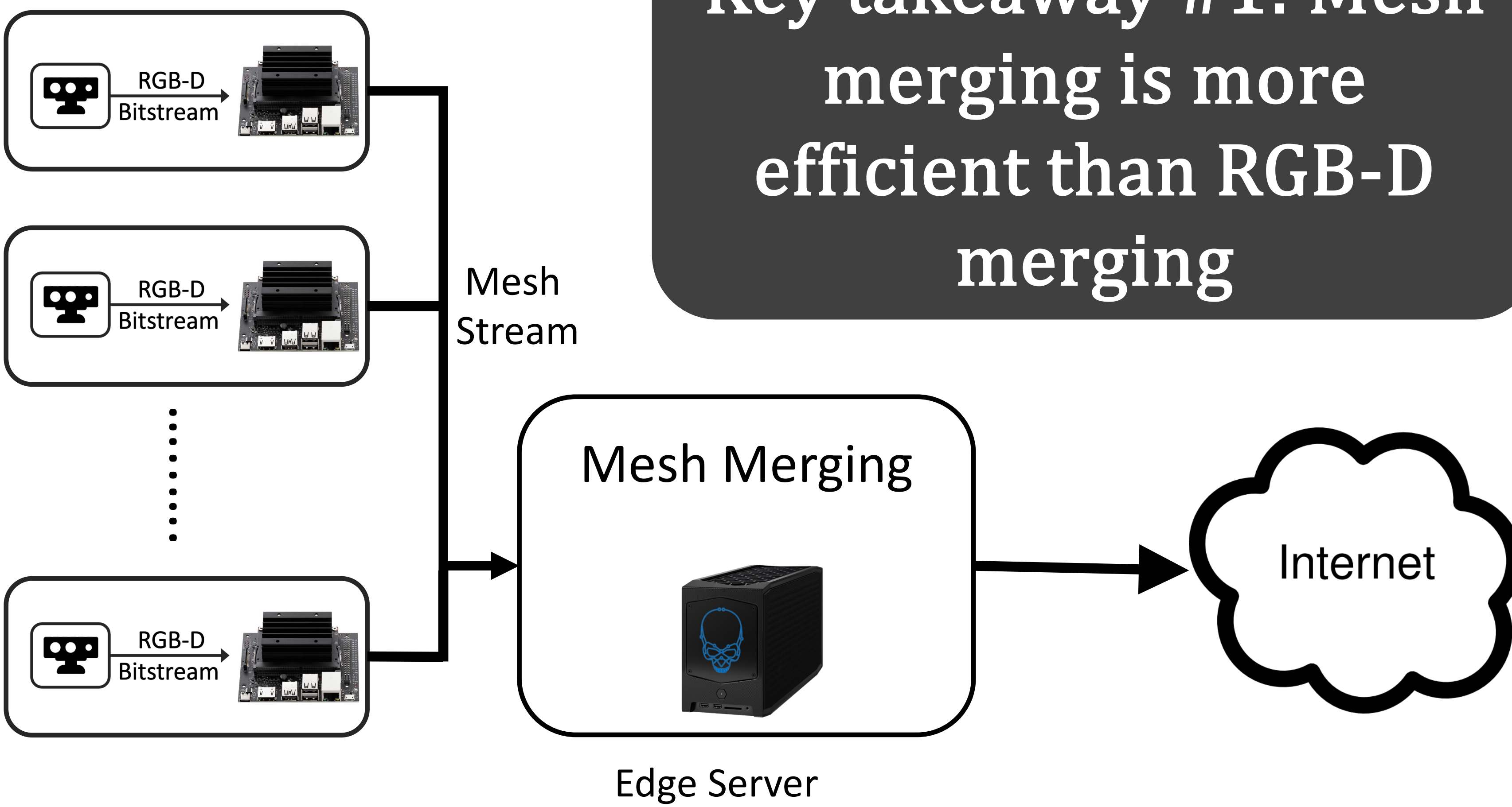
Centralized Textured Mesh Creation Server

Distributed Capture

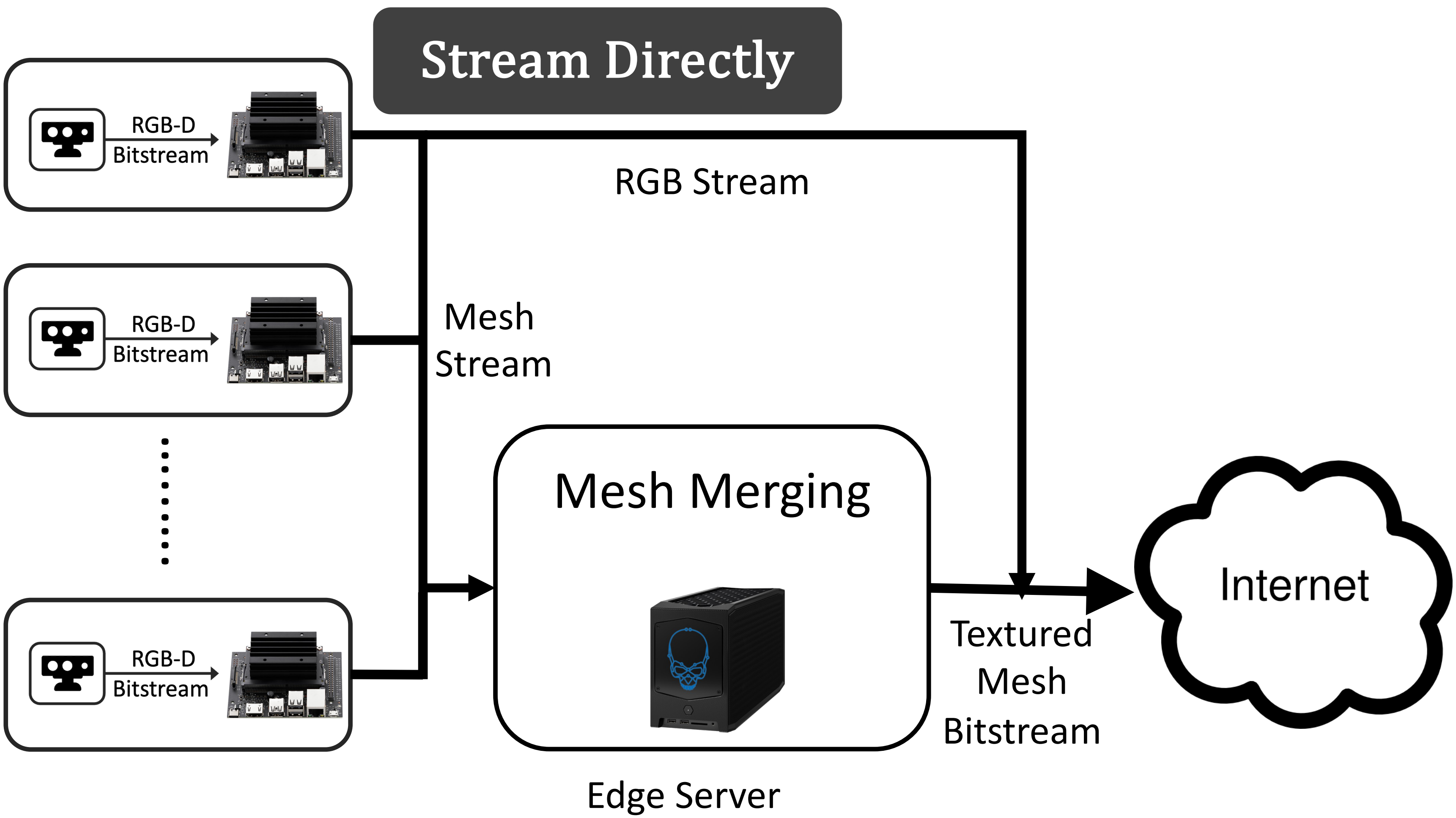


Distributed Capture

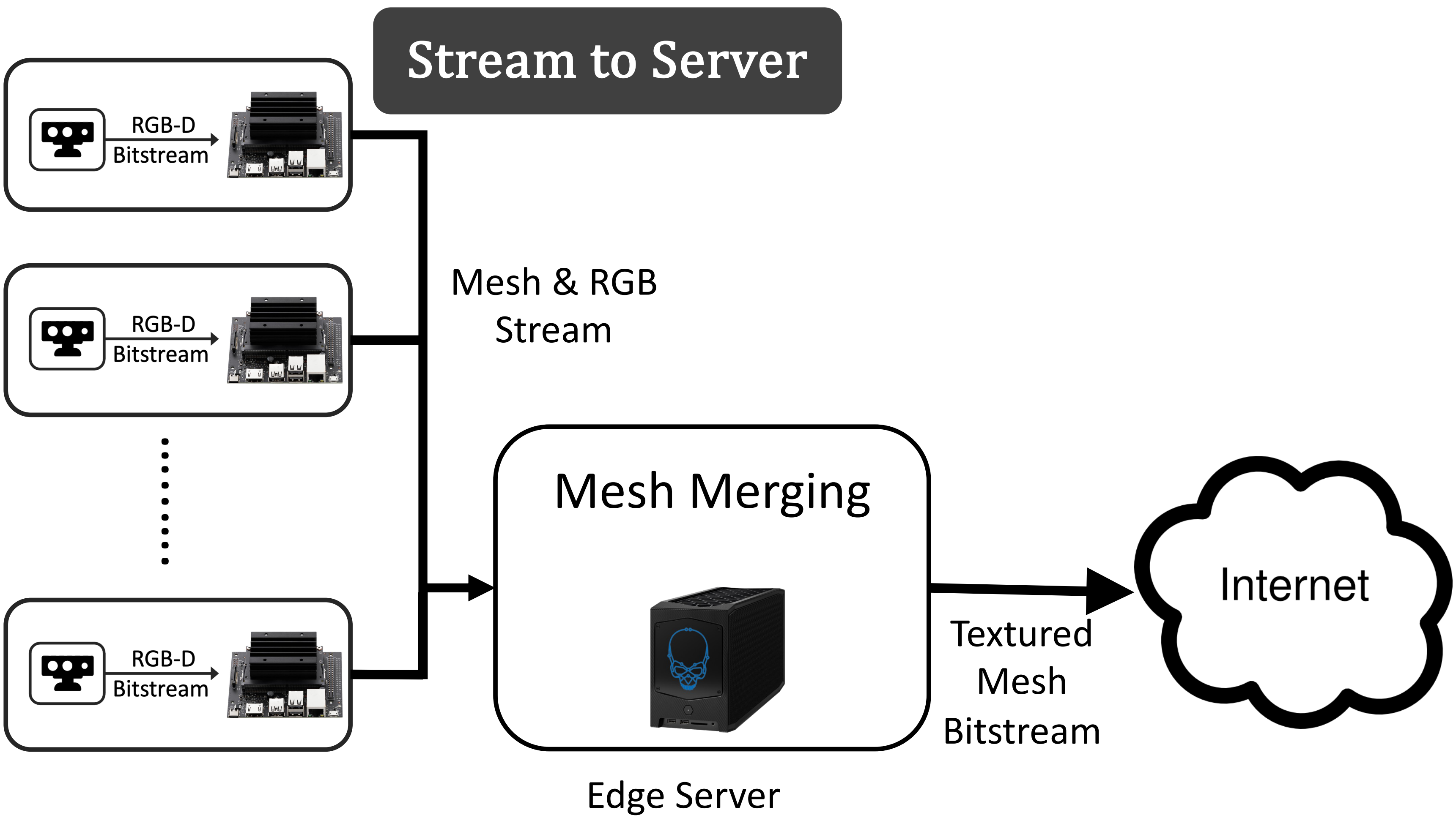
Key takeaway #1: Mesh merging is more efficient than RGB-D merging



Distributed Capture



Distributed Capture



Texture Processing

NxN

NxN

Compact Packing

Texture Processing

Compact Packing

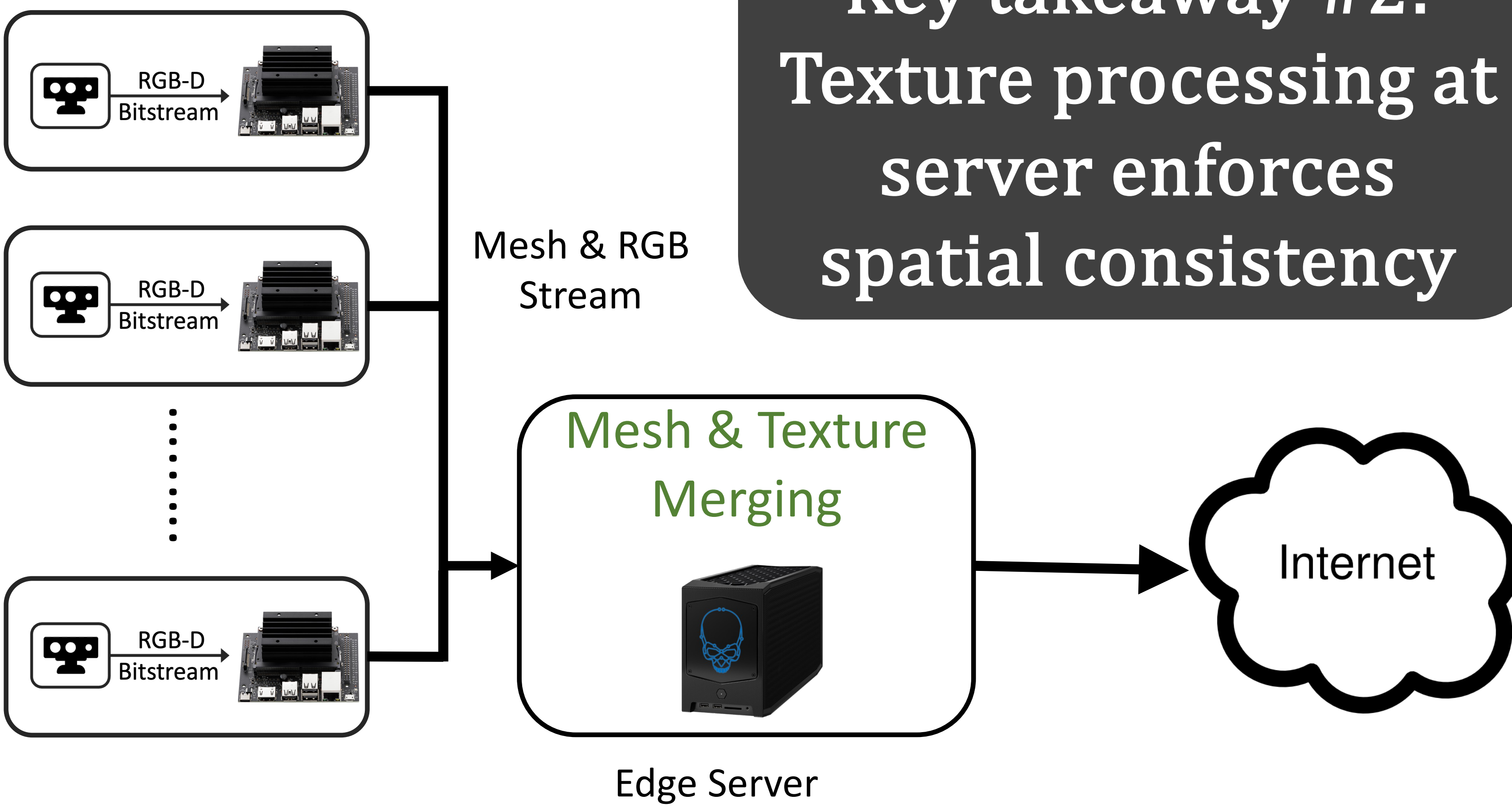
Nx2N

< Nx2N

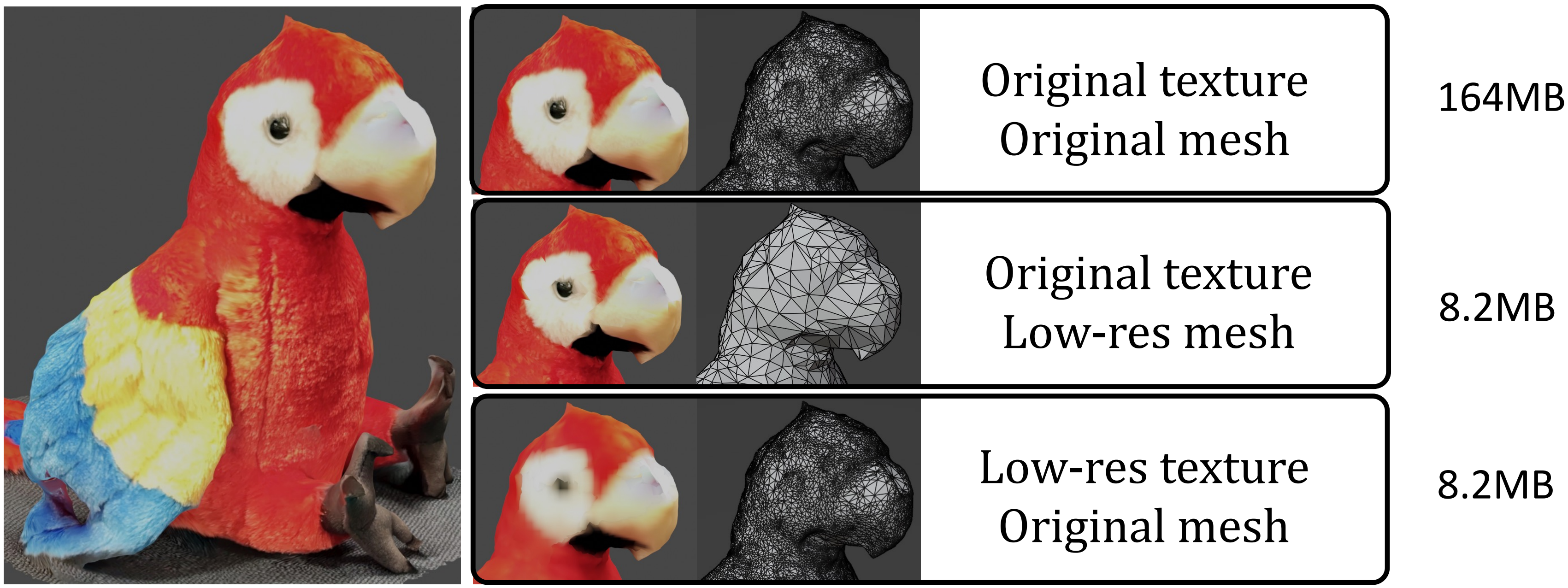
Codec/Bandwidth Efficient Packing

Distributed Capture

Key takeaway #2: Texture processing at server enforces spatial consistency



Streaming Texture vs. Mesh Geometry



Texture has more impact on final rendered quality. It should be allocated more bits.

Streaming Texture vs. Mesh Geometry

Rate control Problem

$$\mathcal{L} = \mathcal{D} + \underbrace{\lambda_t \mathcal{R}_t + \lambda_g \mathcal{R}_g}$$

Rate distortion Function

Distortion

Bitrates for texture and mesh geometry with tuning parameters

How to split the bandwidth between texture and mesh?

How to select optimal coding parameters for both texture & mesh?

Streaming Texture vs. Mesh Geometry

How to split the bandwidth between texture and mesh?

How to select optimal coding parameters for both texture & mesh?

RGB-D frames → Predictive Model

Bandwidth history → Predictive Model

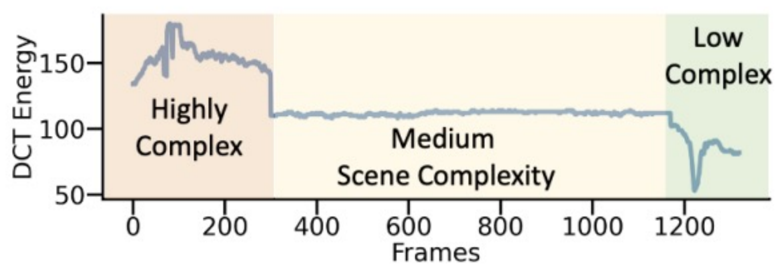
Predictive Model → Coding Parameters for Texture & Mesh

Scene Complexity Metric

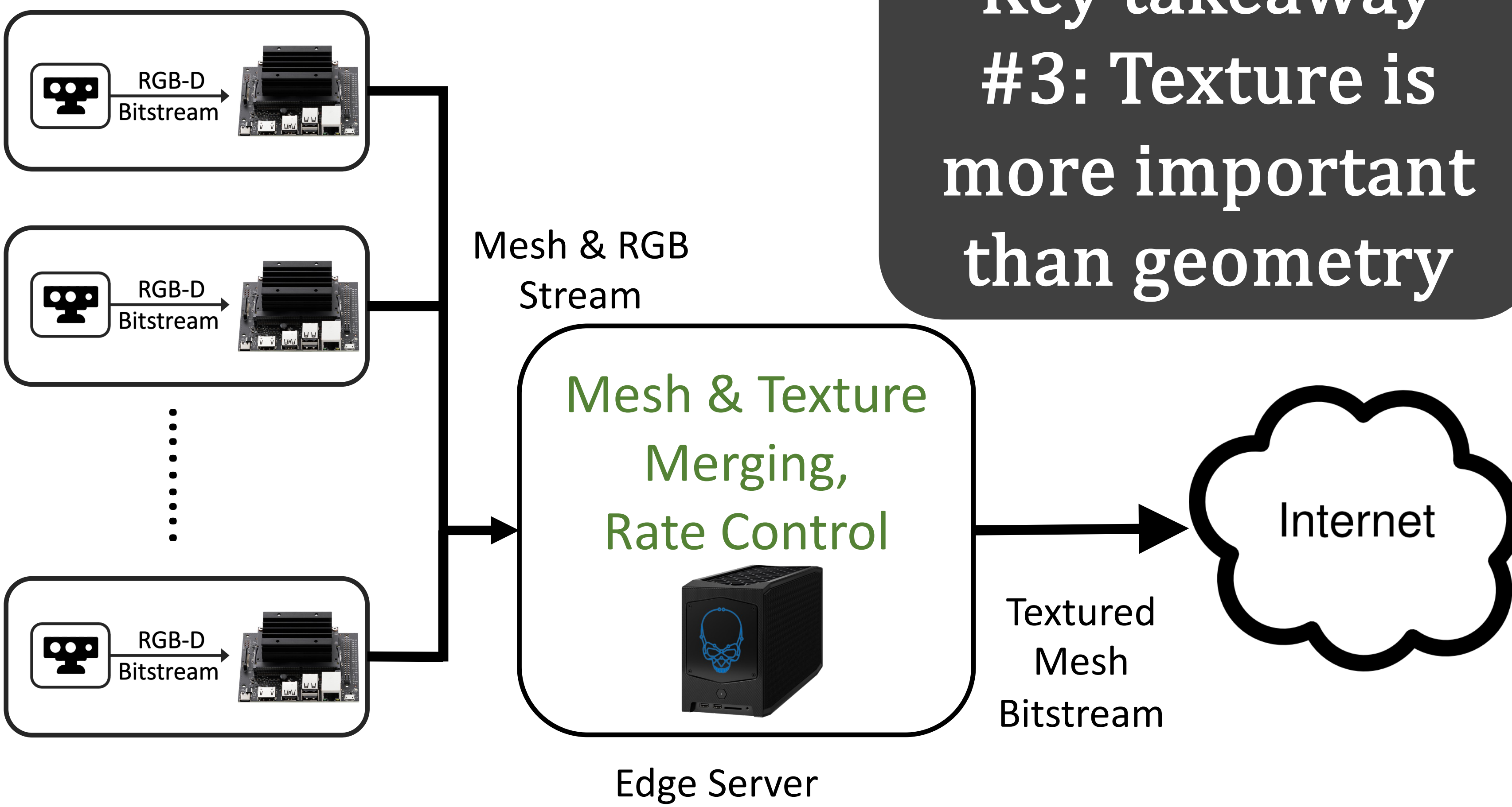
- DCT Energy

$$\mathscr{E}_{dct} = \sum_{i=w}^{j=h} e^{[(\frac{ij}{wh})^2-1]} |DCT(i-1, j-1)|$$

where, where w and h are the width and height of each block, and $DCT(i, j)$ is the $(i, j)^{th}$ DCT component when $i + j > 2$, and 0 otherwise

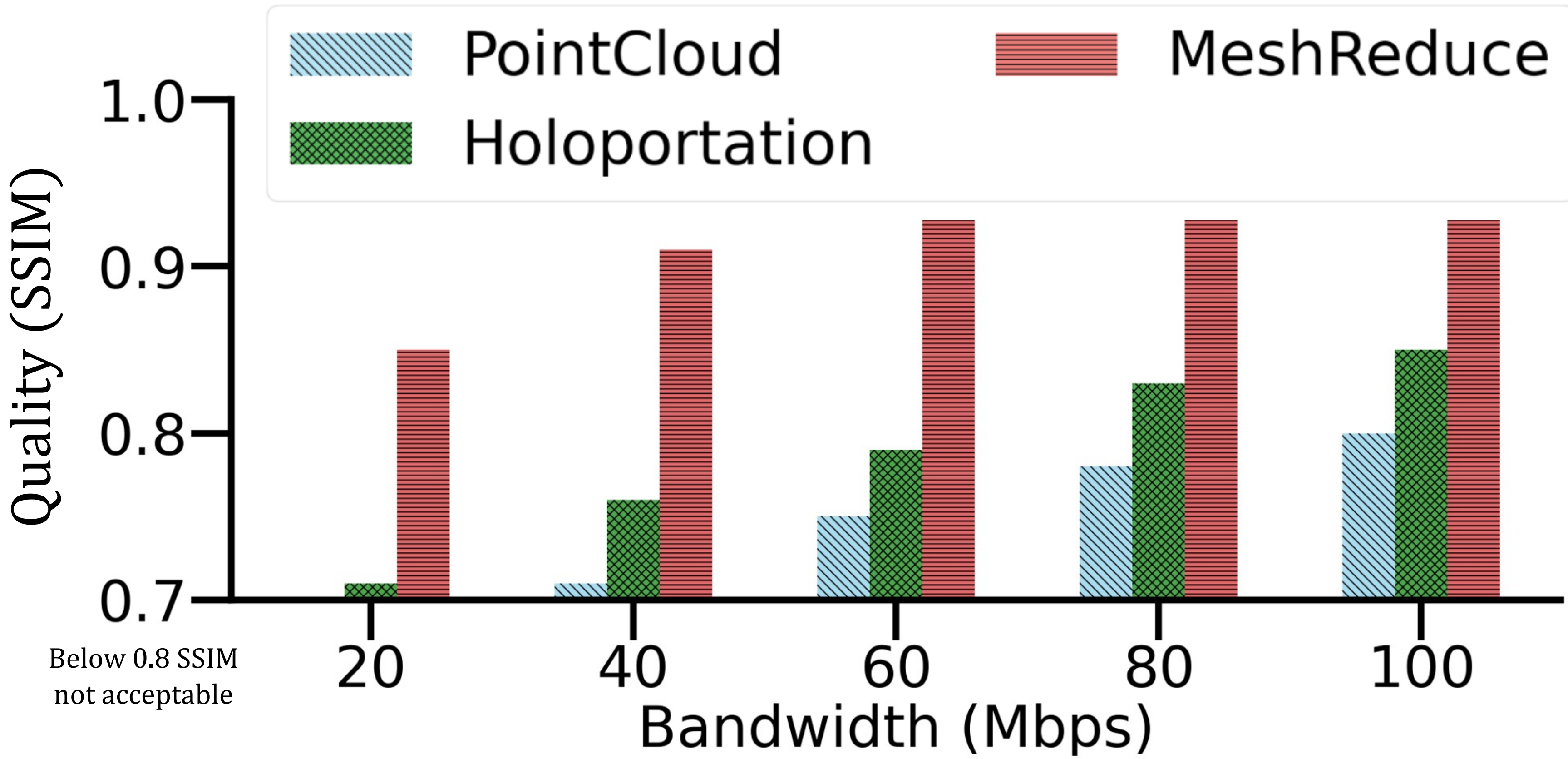


Distributed Capture



Key takeaway #3: Texture is more important than geometry

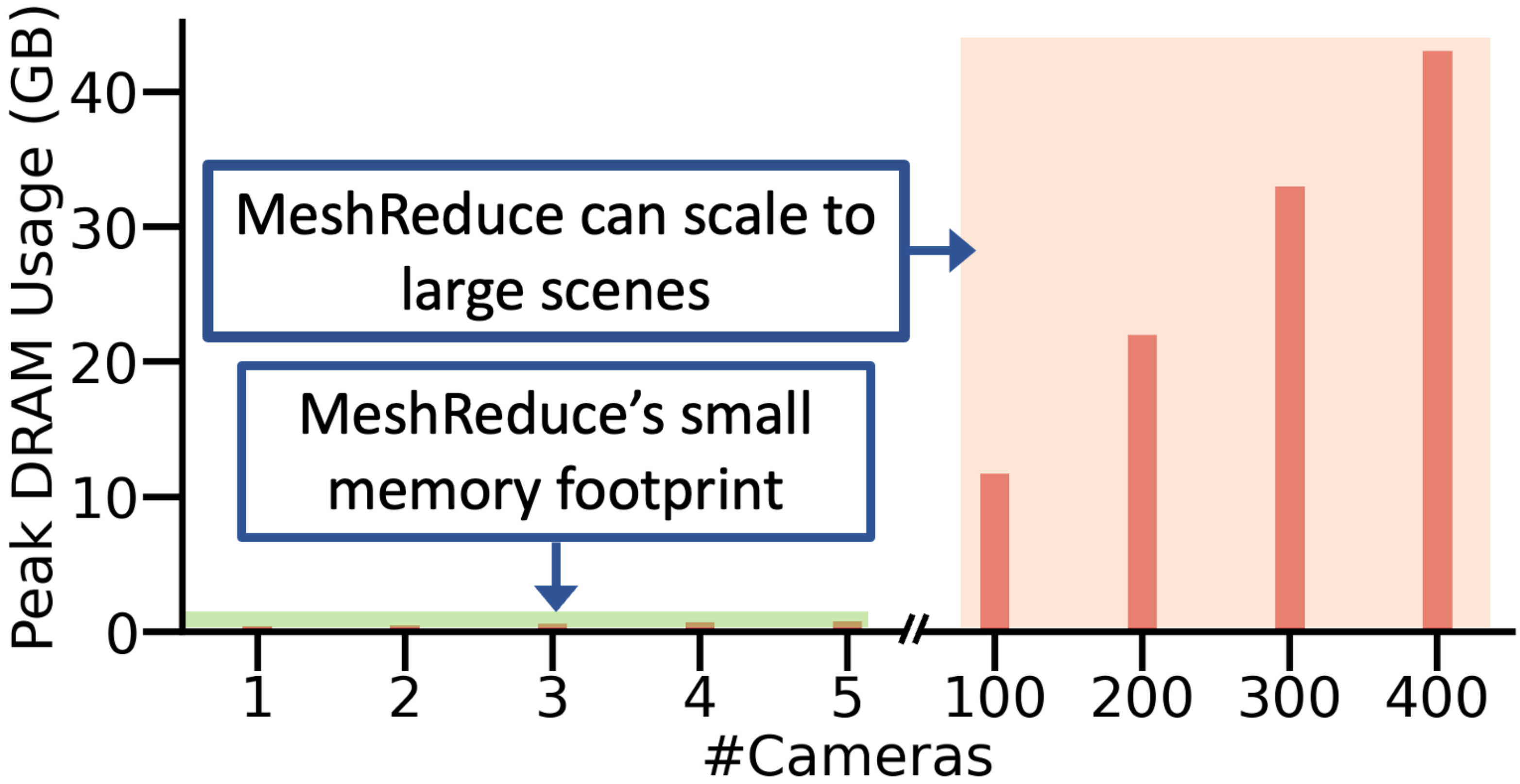
Distributed Capture Performance



Below 0.8 SSIM
not acceptable

Significantly better quality under lower bitrates

Scalability of Distributed Capture



Not GPU bottlenecked; DRAM is abundant and scalable

Adapting Texture and Mesh Streams

The scene is segmented into N objects: $1...N$.

q_i is the perceived quality of of the object i.

the quality q is a function of the texture video bitrate t_r, mesh polygon distortion ratio (m_r), and the texture mapping distortion (t_d).

We can model the function q in the following way.

$q = f(t_r \oplus m_r) - t_d$

where, $f(t_r \oplus m_r)$ is the perceived quality of rendered quality using the texture quality at t_r bitrate, and mesh quality at m_r bitrate.

Adapting Texture and Mesh Streams

$f(t_r \oplus m_r)$ can be computed in the following ways.

1. A linear function: $f(t_r \oplus m_r) = t_r + m_r$
2. A ratio: $f(t_r \oplus m_r) = \frac{t_r + m_r}{max_{t_r} + max_{m_r}}$
3. An index into a table of $t_r + m_r$ estimated through rate distortion curves of t_r and m_r

Adapting Texture and Mesh Streams

Given the above quality function for each object in the scene, we can model the overall expected perceptual quality of rendered final scene as

$E(Q) = \sum_{i=1}^{N} p_i \times r_{i,d} \times q_i$

where p_i is the view probability of object i, and $r_{i,d}$ is an integer variable denoting object being downloaded.

If an object is missed i.e., not downloaded, it results in loss of quality. We represent this loss in quality as

$E(O_m) = \sum_{i=1}^{N} p_i \times r_{i,m}$

where $r_{i,m}$ is an integer variable denoting object being missed.

Thus, the overall QoE can be modeled as

QoE = $E(Q) - \sigma E(O_m)$

Our objective is to maximize the above QoE function subject to the following constraints.

Adapting Texture and Mesh Streams

Constraints.

The following constraints ensure the feasibility of the solution. The quality of an object can only be positive if the object is downloaded, i.e.,

$q_i \geq r_{r,d}$

also, every object must be either downloaded or missed.

$r_{i,d} + r_{i,m} = 1, \forall i = 1...N$

Finally, the downloading must complete before some time δ before the playback time P_t. Let $d(q_i)$ be the download time of object i at quality q_i. Then this constraint can be represented as.

$\sum_{i=1}^{N} d(q_i) \times r_{i,d} + \delta \leq P_t$

This represents the bandwidth constraint, which can also be written as follows, $t_r + m_r \leq C$, where C is the predicted bandwidth.

Summary of the Lecture

- Mesh streaming
- Distributed Capture
- Adaptation of texture and mesh streams

Next up: Progressive transmission of XR content