

EECE5512

Networked XR Systems

Last Class - Recap

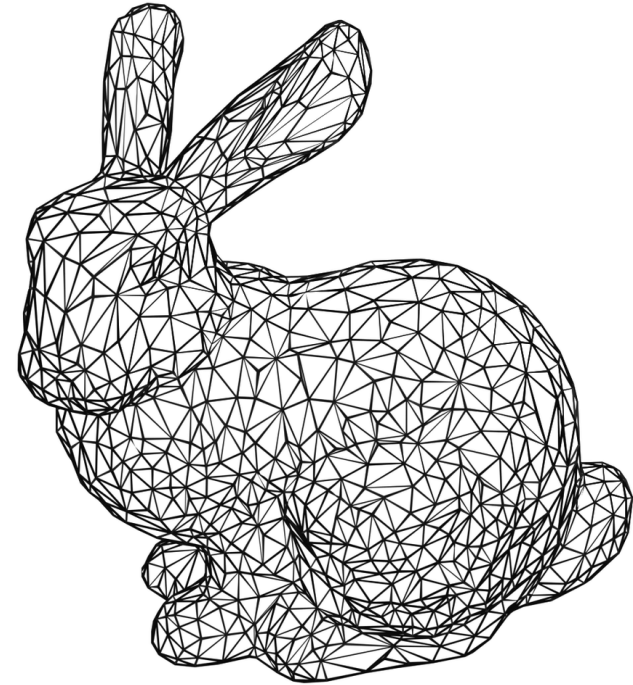
- Homework2
- Depth Map Compression
- Point Cloud Compression
 - MPEG GPCC
 - MPEG VPCC

Lecture Outline for Today

- Mesh Compression

Mesh

- A set of polygons, connected by their common edges or vertices
- Typically represented by triangles
- Meshes are fundamental to rendering scenes in video games, animations, XR, and more.



Mesh

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

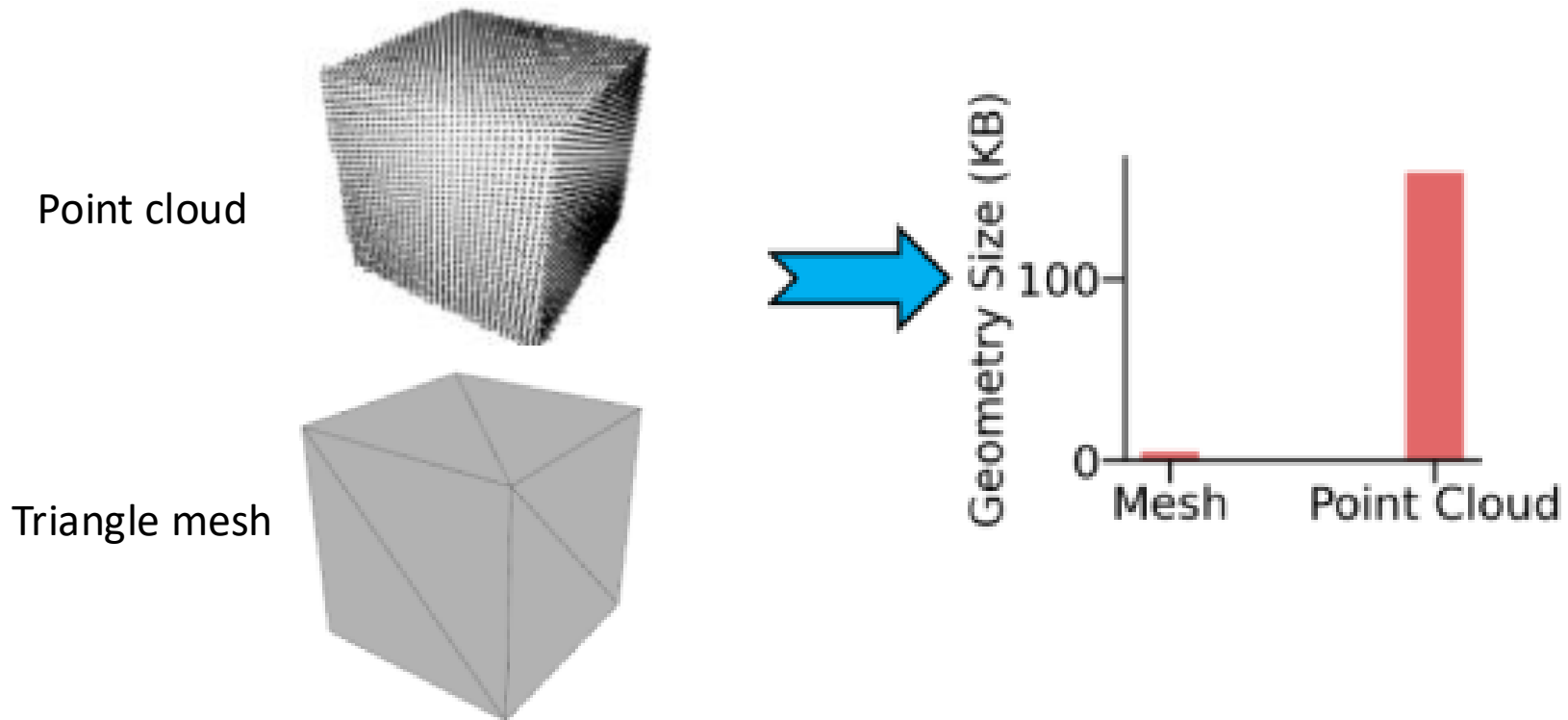


Why Mesh Compression

- **Challenges:** Large meshes consume significant memory and bandwidth, making storage and transmission inefficient.
- **Objectives:** Compress meshes to reduce file size without significantly losing quality, enabling faster loading times and lower storage requirements.
- **Benefits:** Efficient mesh compression improves performance in real-time applications and reduces costs in data transmission and storage.

Mesh vs. Point Cloud

- Meshes are much more compact



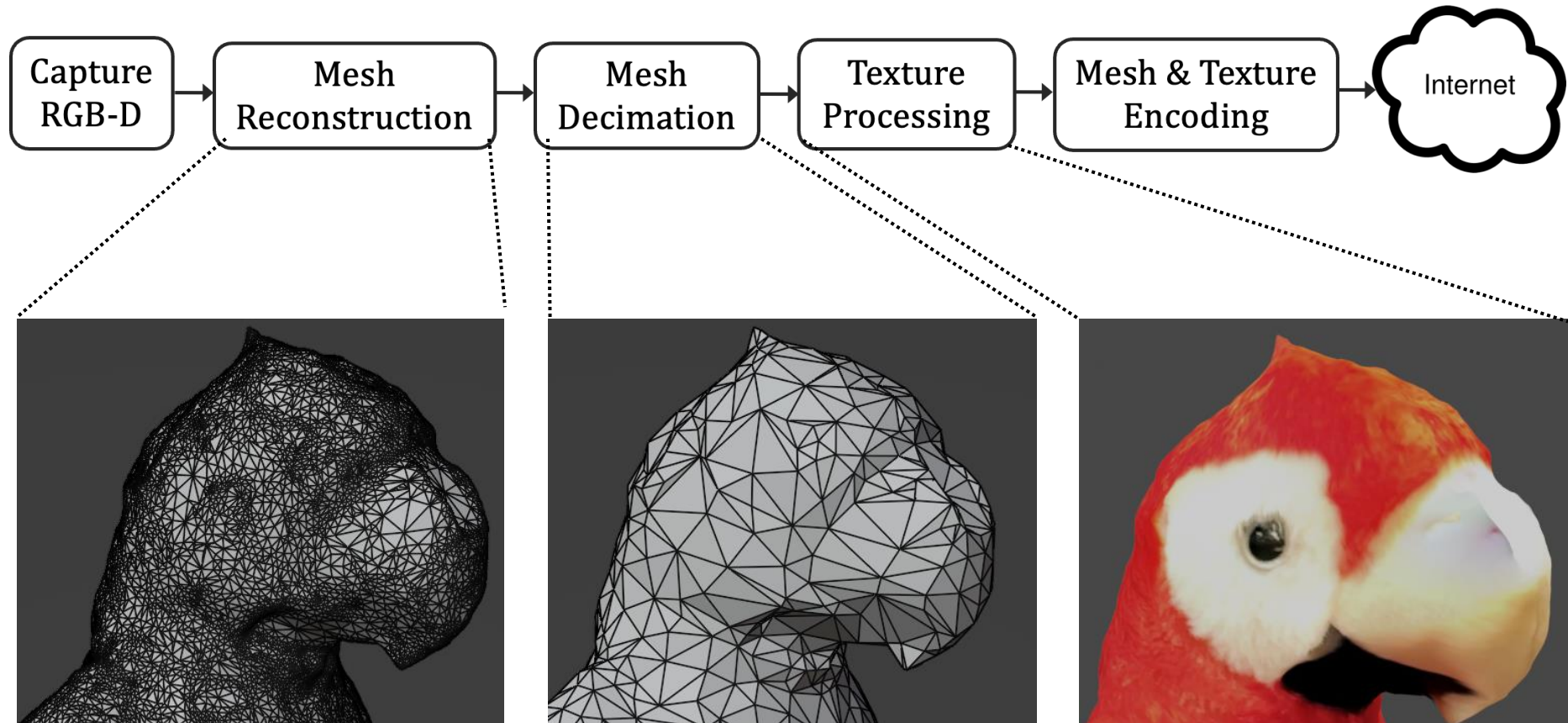
Counter Intuitive from the previous slide?

Mesh Compression

- Mesh Simplification – Vertex clustering or quadratic error decimation
- Vertex Compression
- Connectivity Compression
- Texture Compression

Mesh Compression

Mesh simplification can be a form of compression



Mesh Compression

- Vertex Compression
 - Reduce the size of vertex coordinates while preserving the mesh's geometric detail.
- Techniques
 - **Quantization:** Converts floating-point coordinates to a fixed number of bits, reducing precision but saving space.
 - **Predictive Coding:** Encodes vertex positions as differences from predicted positions based on previous vertices, exploiting spatial coherence.
- Example: Using delta encoding, where each vertex position is stored as the difference from the previous vertex, significantly reducing the range of values.

Mesh Compression

- Vertex Compression
 - Reduce the size of vertex coordinates while preserving the mesh's geometric detail.
- Techniques:
 - Vertices can be considered same as point cloud
 - **Can we use point cloud compression techniques that we discussed in the previous lecture?**

Mesh Compression

- Connectivity Compression
 - Efficiently encoding the mesh topology, which defines how vertices are connected to form faces.
- Techniques – Edgebreaker algorithm
 - The algorithm traverses the mesh, encoding its topology with a sequence of symbols representing the traversal operations.
 - Includes symbols like C (connect), L (left), R (right), E (end), and S (start), which describes how to move from one triangle to the next – CLERS String

Mesh Compression

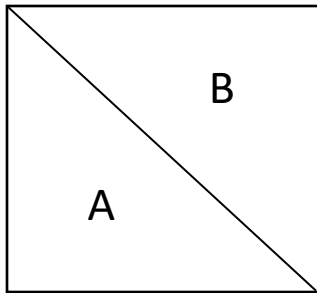
For each new triangle encountered, Edgebreaker records its connectivity relation with previously visited triangles using one of five symbols, collectively called the CLERS code:

| Symbol | Meaning | Description |
|--------|---------|---|
| C | Create | Introduces a new vertex (corner) not seen before. |
| L | Left | The left neighbor of the current triangle is already visited. |
| E | End | The triangle closes a region (no new neighbor). |
| R | Right | The right neighbor is already visited. |
| S | Split | A split occurs — multiple unvisited neighbors share a vertex. |

Mesh Compression

- Edgebreaker Algorithm

- The algorithm starts at an edge of the mesh and follows the edges around the mesh in a systematic way, essentially "breaking" the edges as it goes to avoid retracing its path. This traversal forms a loop around the mesh, visiting each triangle once.



Step 1: Start at the outer edge of **A** (S could denote this start, but it's optional).

Step 2: Move to triangle **B** via the shared edge — since **B** is directly connected without requiring a turn, this move is encoded as "C" (Connect).

Step 3: From **B**, there's no new triangle to visit, so the algorithm would end — this could be marked with "E" for End, but since it's a simple case, the end might be implicit.

Mesh Compression

- Do this mesh as an exercise – Edgebreaker algorithm.



Mesh Compression

- Connectivity Compression
 - Efficiently encoding the mesh topology, which defines how vertices are connected to form faces.
- Techniques – Edgebreaker algorithm
 - Achieves at most 4 bits per vertex
 - Published in 1996 but popular even today
 - Used in Google's Draco Mesh compression code (<https://google.github.io/draco>)

<https://faculty.cc.gatech.edu/~jarek/papers/EdgeBreaker.pdf>

Mesh Compression

- Connectivity Compression
 - Efficiently encoding the mesh topology, which defines how vertices are connected to form faces.
- Techniques – Edgebreaker algorithm
 - Limitations: the algorithm assumes a manifold mesh, which may limit its applicability to meshes with more complex topologies without preprocessing

Definition of manifold mesh: if you were a tiny ant walking on the surface of the 3D model, you could walk all over the model without ever finding a place where the surface doesn't make sense i.e., no holes, no edges hanging in the air, and no overlapping faces.

Mesh Compression

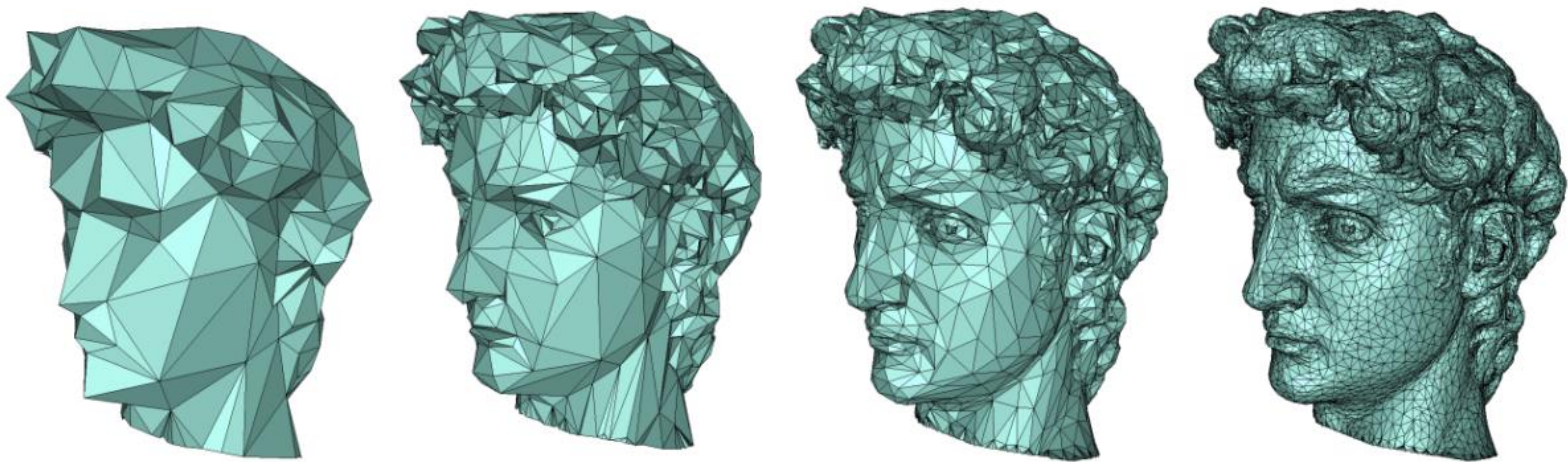
- Connectivity Compression
 - Efficiently encoding the mesh topology, which defines how vertices are connected to form faces.
- Techniques
 - Edgebreaker – lossy for non-manifold meshes
 - TFAN (Triangle fan) algorithm – lossless
 - Valence-driven encoding – based on the number of connected edges

Mesh Compression

- Texture compression
 - How?

Mesh Compression

- Progressive compression
 - Different levels of detail are created by simplifying the original mesh step by step, usually by vertex decimation or edge collapse techniques.

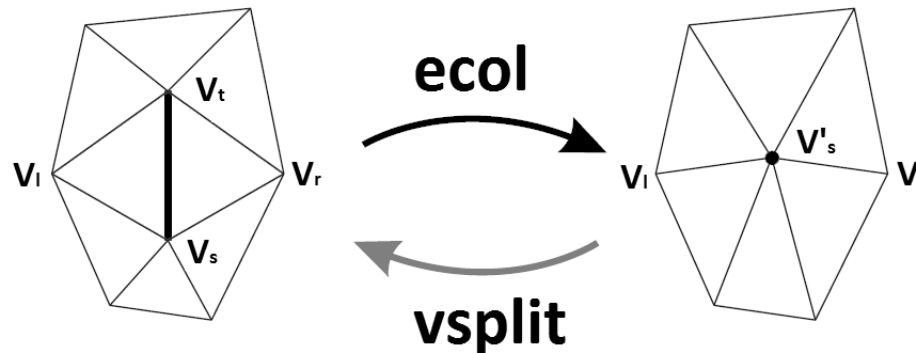


Mesh Compression

- Progressive compression

1. Edge Collapse: In the simplification process, an operation called "edge collapse" is frequently used, where an edge between two vertices is collapsed into a single vertex, reducing the overall count of vertices and faces.

2. Vertex Split: The reverse of edge collapse is "vertex split." To refine the mesh, the algorithm splits a vertex into two and recreates the original edge and associated faces. The vertex split operation is stored as a record of how to refine the mesh from one LOD to the next.



Mesh Compression

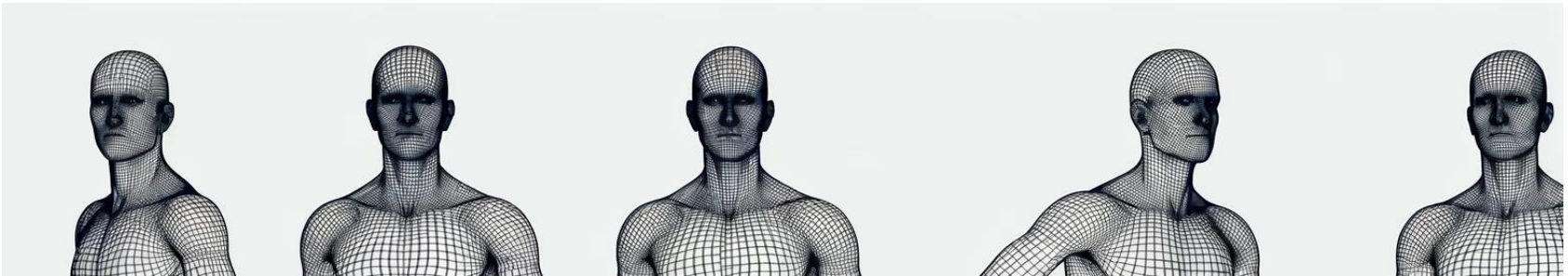
- So far, we talked about only static meshes... What about dynamic meshes?
- Animated meshes
- Sequence of mesh frames

Animated Mesh Compression

- **Sparse Keyframes:** Instead of storing every frame of the animation, only keyframes are stored, and intermediate frames are interpolated. This greatly reduces the amount of data required.
- **Interpolation:** The in-between frames are generated by interpolating the transformations (such as position, rotation, and scaling) from keyframes. Efficient algorithms ensure that this interpolation does not require too much computational power.

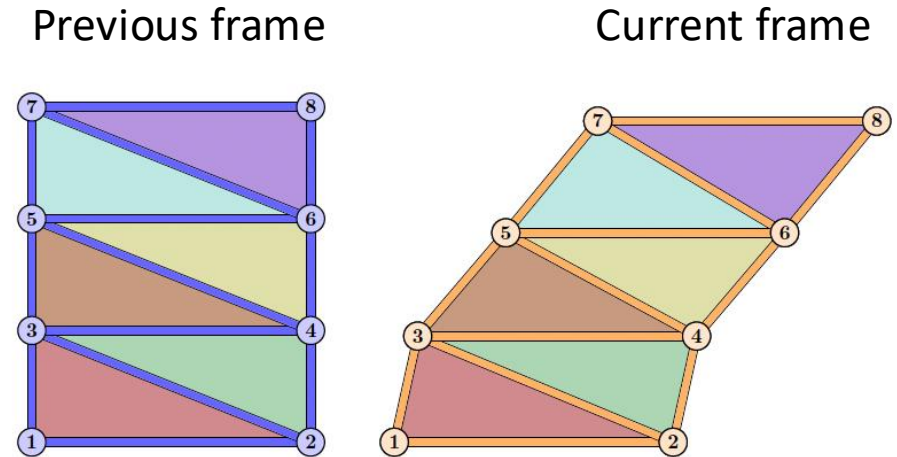
Compressing a mesh sequence

- Recall intra and inter frame prediction for exploiting spatial and temporal redundancy in 2D videos
 - Can we apply similar principles?



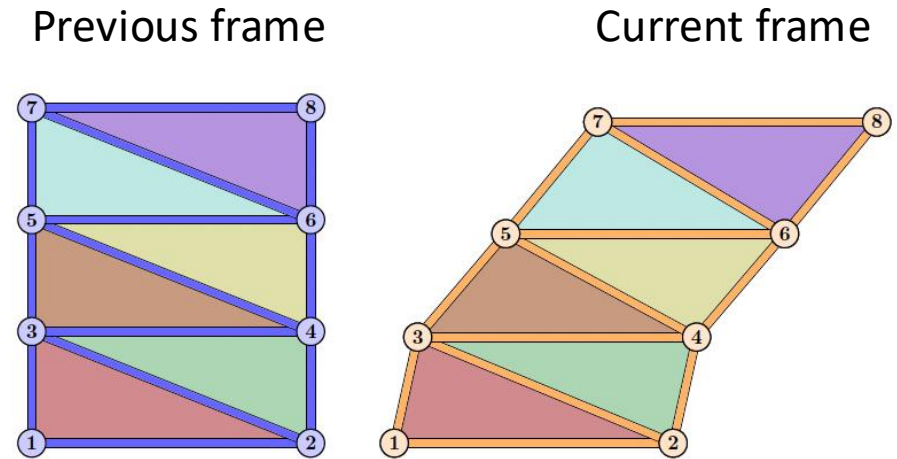
Compressing a mesh sequence

- Compress displacements instead of vertices
 - Displacements are much smaller values and require fewer bits compared to vertices
- Key assumption: vertex correspondence



Compressing a mesh sequence

- Compress displacements instead of vertices
 - Displacements are much smaller values and require fewer bits compared to vertices
- Key assumption: vertex correspondence

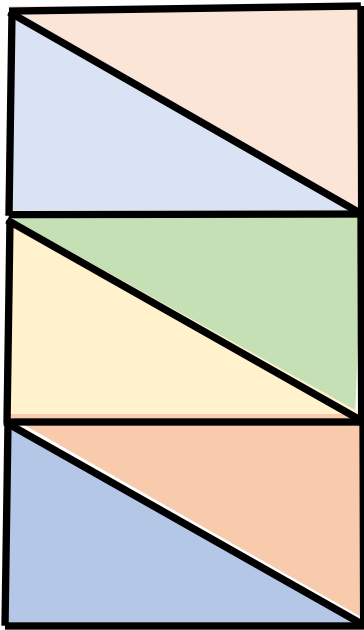


Final Step: Entropy Coding

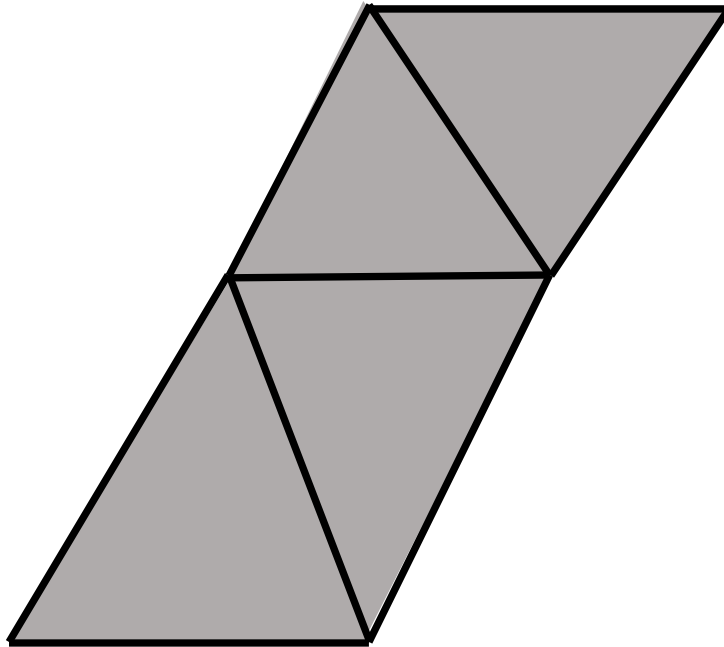
Compressing a mesh sequence

- Topology changes in practice (also called as time varying mesh)

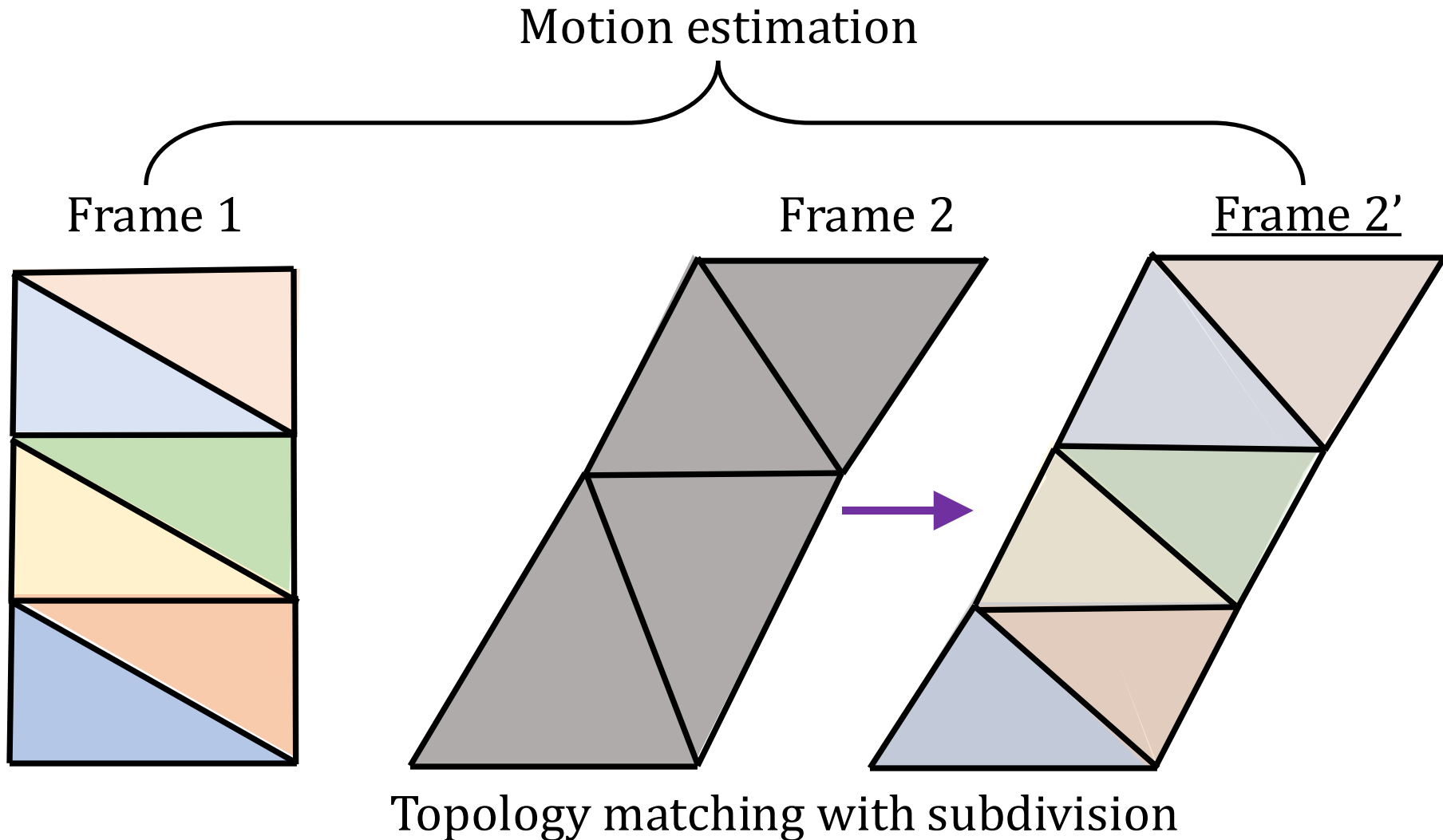
Frame 1



Frame 2



Compressing a mesh sequence



Compressing a mesh sequence

- Extract key points from each mesh
- Establish correspondences
- Apply non-rigid transformation



Open3D

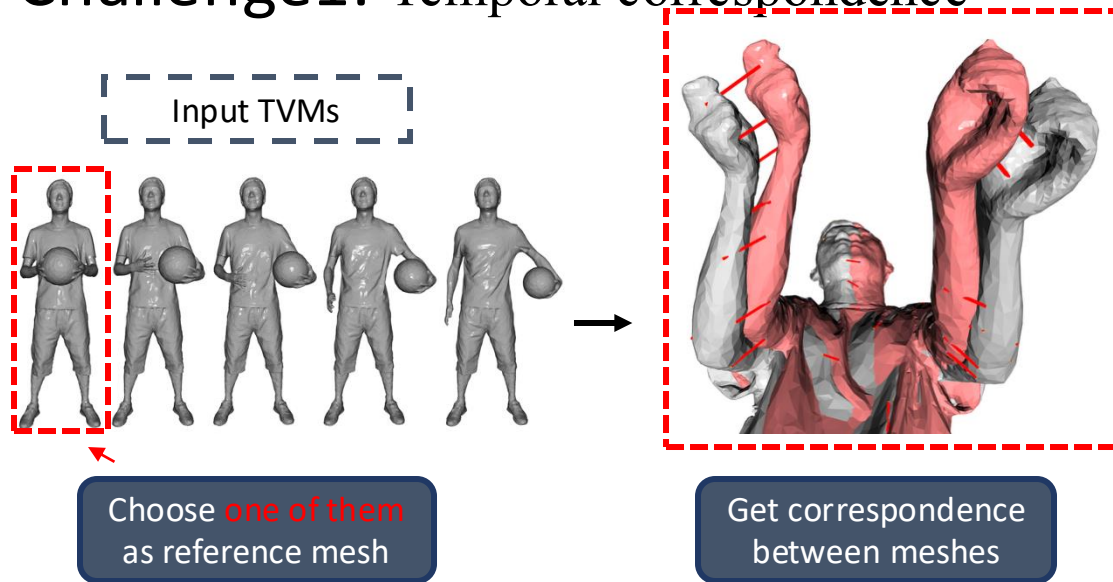
Compressing a mesh sequence

- Challenges
 - Not easy to get a useful reference mesh always – due to self contact or addition or deletion of geometry across time
- Still an active area of research – no open source or very well adopted techniques yet

TVMC: Time-varying mesh compression

TVMC: MMSys 2025

- Challenge1: Temporal correspondence



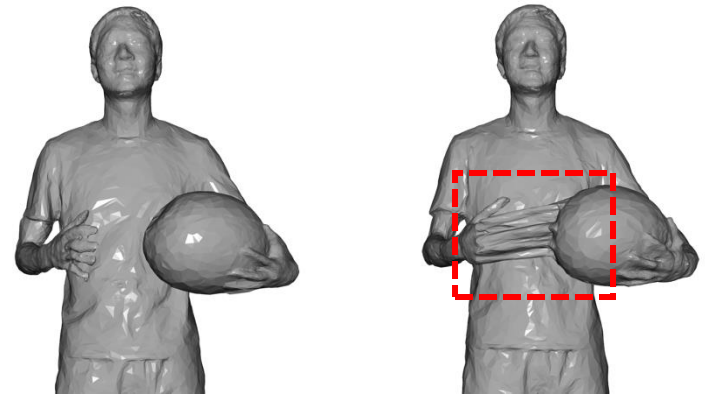
- Intrinsic Shape Signatures
- ICP/optimization

Unstable & unreliable

TVMC: Time-varying mesh compression

- Challenge2: Self contact issue

- Deforming mesh based on the movement of selected keypoints
 - Get the nearest K keypoints
 - Update the vertex position based on these K deformations

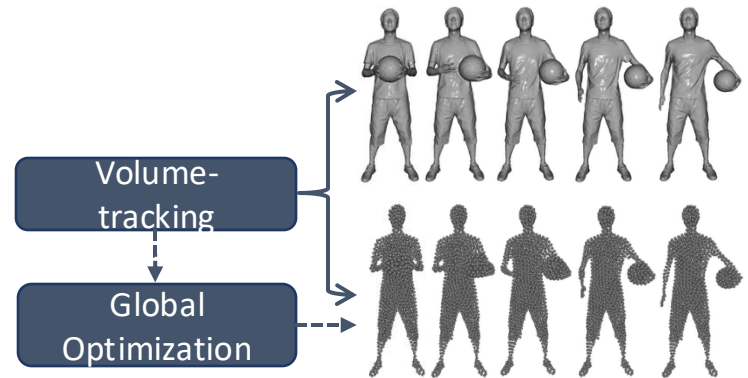


Visual distortion due to self contact issue

TVMC: Time-varying mesh compression

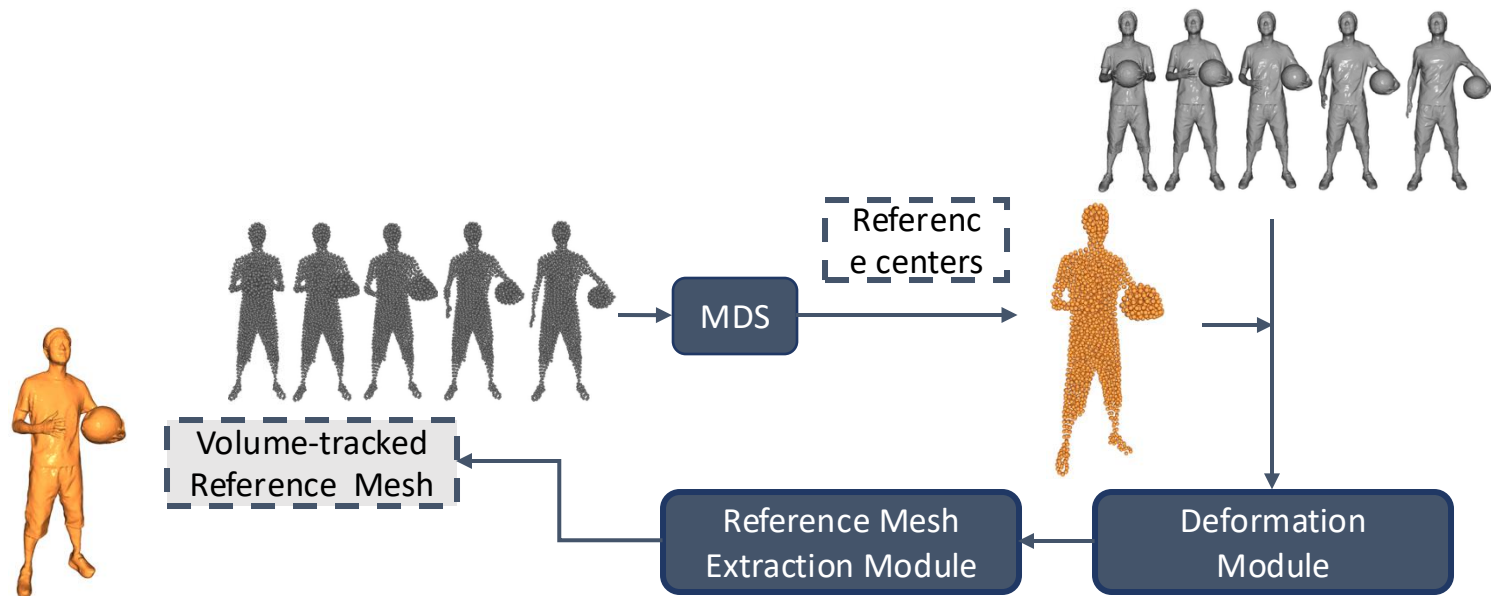
- **Our insight 1:** working on **volume** enclosed by the mesh surface can create more stable and valuable **inter-frame correspondence**.

- Converted into a dense regular square voxel grid
- Fast winding number in-and-out test
- Uniform distribution
- Linear extrapolation and optimization



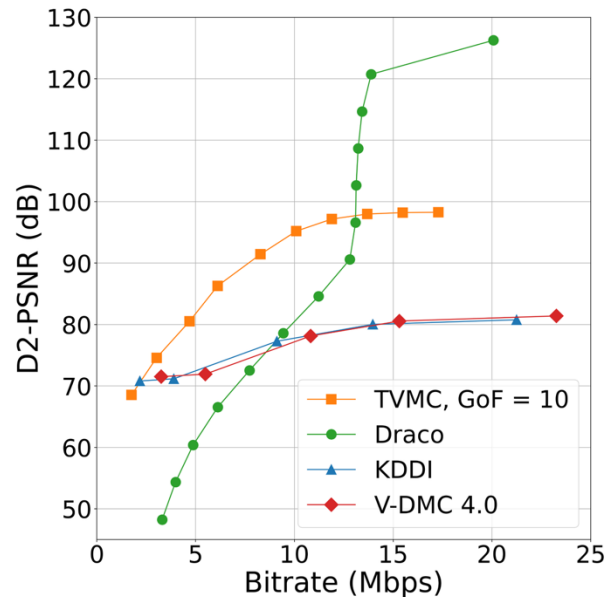
TVMC: Time-varying mesh compression

- **Our insight 2:** create a **self-contact-free** reference mesh based on a group of consecutive frames that can be deformed to get approximations of each mesh frame in the group.



TVMC: Time-varying mesh compression

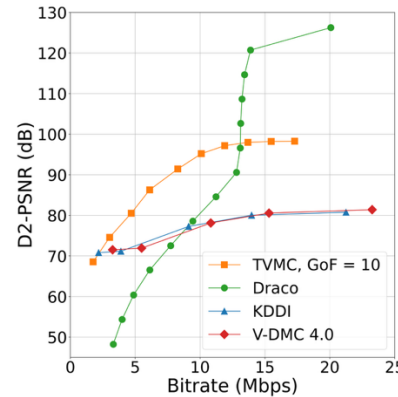
- Baselines
 - Google Draco, V-DMC 4.0 from MPEG, KDDI's work from Japan
- Dataset
 - MPEG TVM Dataset, our own custom dataset
- Metrics
 - Quantitative: D2-PSNR, RMSE, Coding time
 - Qualitative evaluation



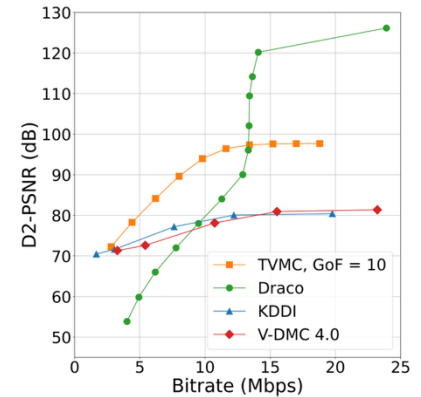
Small objects

TVMC Experiment results

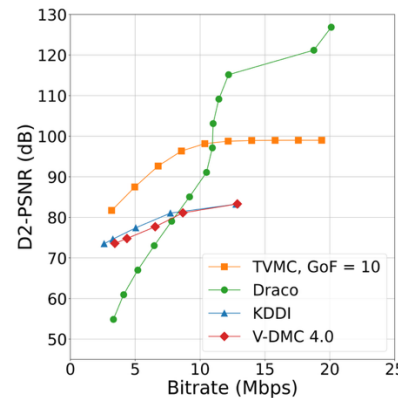
- TVMC outperforms Draco, KDDI, V-DMC 4.0 with bitrates varying from 5 Mbps to 10 Mbps
- Reference mesh with displacements
- Increase the group of frame size to 10 (5 for KDDI)
- More inter-frame predicting compared with V-DMC



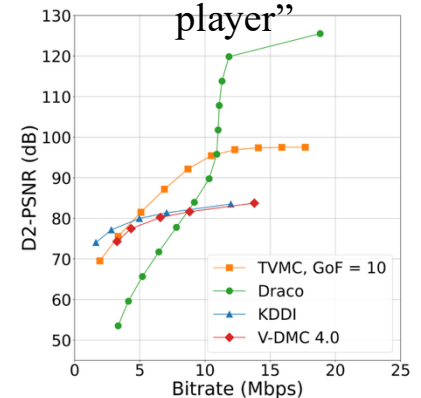
“Dancer”



“Basketball player”



“Mitch”

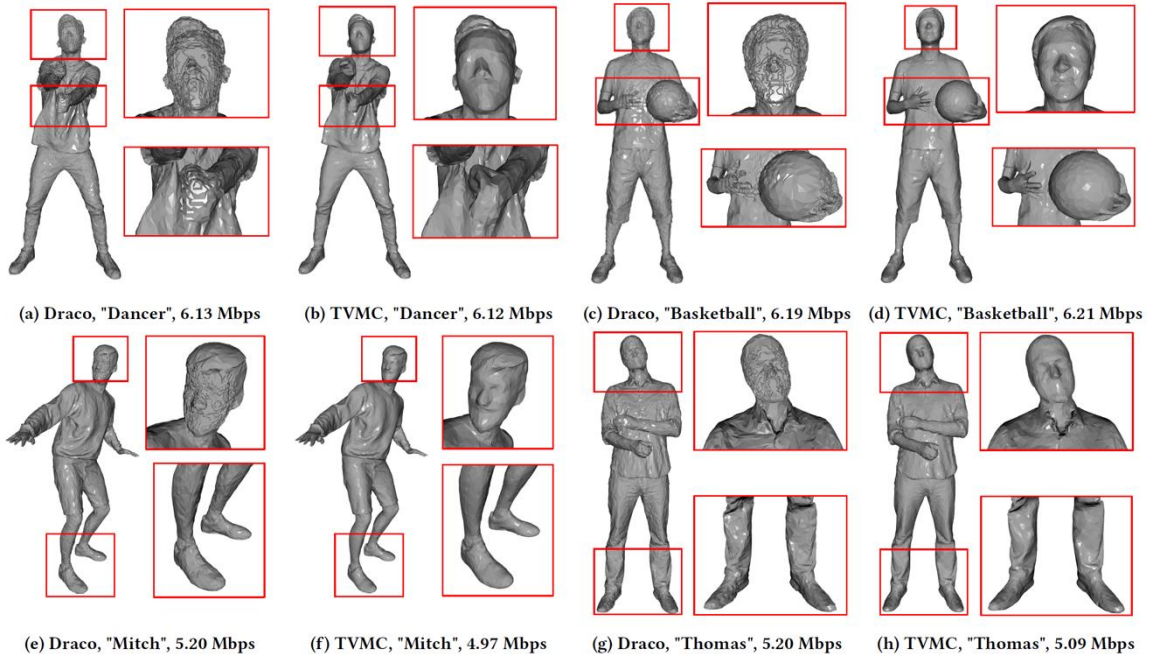


“Thomas”

TVMC Experiment results

- Visual quality
 - distortions on the parts of faces, hands, or feet
- Decoding time
 - TVMC requires on average 13.95 ms, 66.1% reducing compared to Draco.

| TVMs | TVMC | | | | | Draco | | | |
|-------------------|---------|-------|---------|----------|---------|---------|-------|-----------|----------|
| | D2-PSNR | RMSE | Encoder | Decoder | Portion | D2-PSNR | RMSE | Encoder | Decoder |
| Dancer | 84.10 | -3.63 | 13.38 s | 15.20 ms | 93.08% | 66.55 | -2.76 | 181.70 ms | 46.20 ms |
| Basketball player | 84.14 | -2.65 | 21.12 s | 14.60 ms | 93.14% | 66.01 | -1.74 | 229.27 ms | 47.07 ms |
| Mitch | 87.46 | -3.78 | 10.65 s | 13.40 ms | 93.36% | 66.99 | -2.76 | 139.33 ms | 35.40 ms |
| Thomas | 81.50 | -3.54 | 10.88 s | 12.60 ms | 93.52% | 65.65 | -2.75 | 137.93 ms | 35.60 ms |



TVMC Experiment results

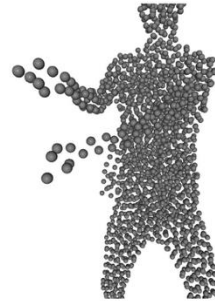
- Number of volume centers can affect TVMC's performance
 - Fine tune experiments may be required to get optimal results



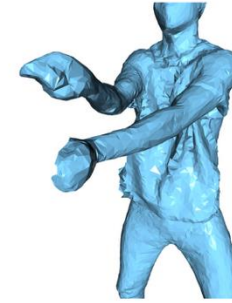
(a) Visualization of 1000 centers



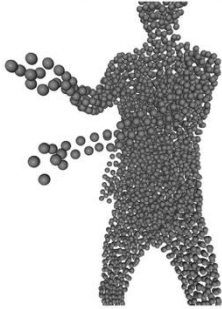
(b) Result with 1000 centers



(c) Visualization of 2000 centers



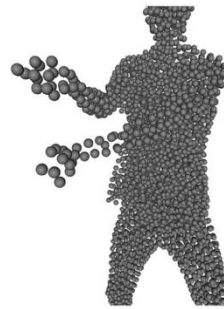
(d) Result with 2000 centers



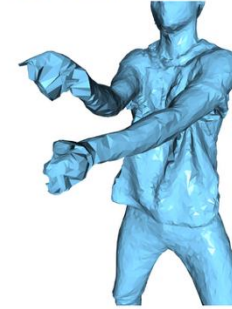
(e) Visualization of 3000 centers



(f) Result with 3000 centers



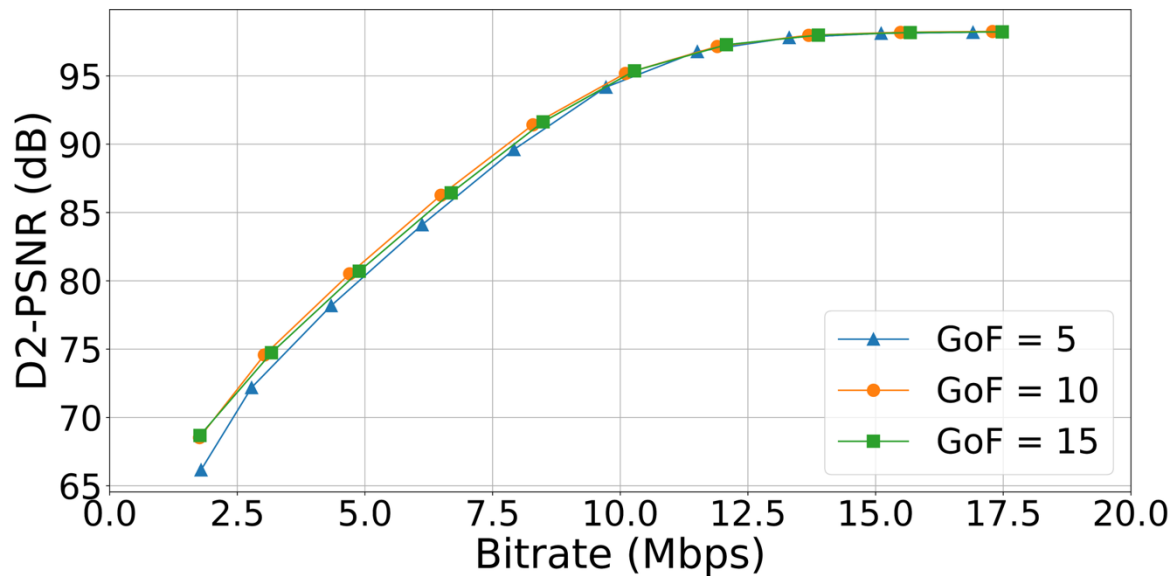
(g) Visualization of 4000 centers



(h) Result with 4000 centers

TVMC Experiment results

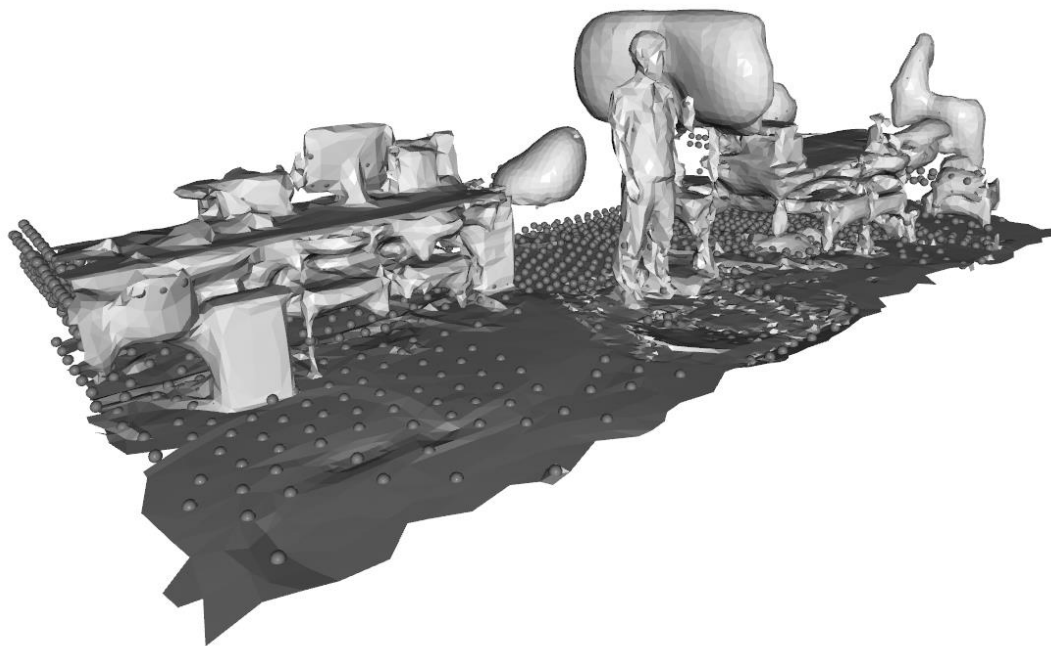
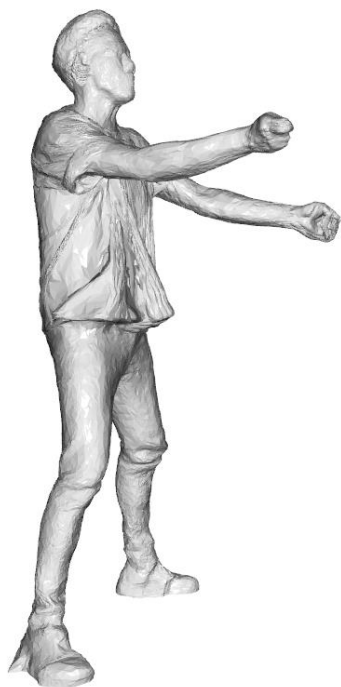
- TVMC can scale the GoF to 15 without obvious quality decrease
 - GoF over 15 can cause distortions because of excessive motion changes



Compressing a mesh sequence

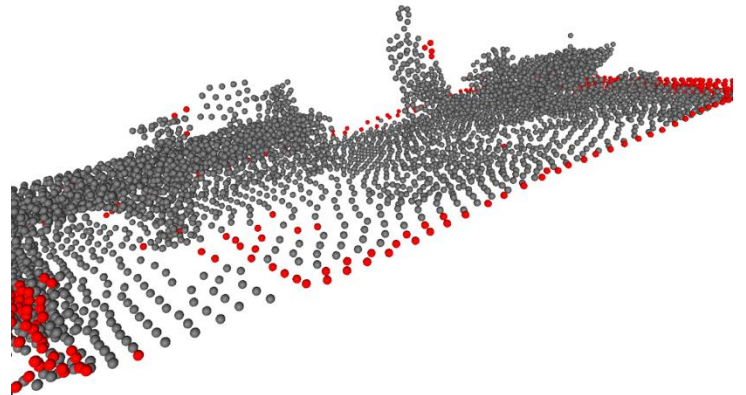
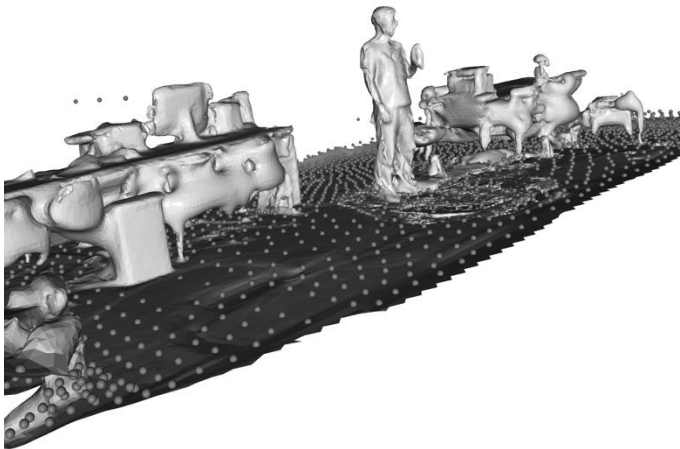
- Small-scale vs. Large scale mesh sequences
- Large meshes often tend to be static most of the regions
 - Divide the mesh into patches
 - Check if the patch is static, if so, don't store, just store a motion vector (note even in static cases topology can change but since we “know” it's a static region it's okay use the lossy reference)
 - If not static, need to store
 - **Question: how to detect if it's static or moving?**

Small vs. Large Meshes



Large Mesh Compression

- Identify dynamic volume centers



Existing Methods & Limitations

| Compression Type | Methods & Tools | Main Idea | Capabilities | Limitations |
|-------------------------------|----------------------|--|---|--|
| Static Mesh Compression | Draco (Google) | Edge-breaker | Single static mesh | Single-frame compression Long decoding time High bandwidth requirement Ignores temporal redundancy |
| Dynamic Mesh Compression | V-DMC (MPEG) | Base mesh + Displacement (Video coding) | Dynamic meshes with consistent topology and connectivity across frames | Cannot handle volumetric video or meshes with varying topologies |
| | VSMC (Apple) | Base mesh + Displacement (Video coding) | Dynamic meshes or some certain time-varying meshes after remeshing | Has constrains on mesh type , unsuitable for volumetric video |
| Time-Varying Mesh Compression | KDDI, 2024 ICASSP | Embedded Deformation, add displacements when decoding | A small group of time-varying meshes (maximum 5) | Self-contact problem Global distortion as every vertex is deformed |
| | UCSD, 2023 ICCVW | Embedded Deformation, use deformation when decoding | A small group of time-varying meshes (maximum 5) | Self-contact problem Global distortion as every vertex is deformed Long decoding time |
| Neural-Based Mesh Compression | Google, 2018 ACM TOG | Truncated Signed Distance Function | Single static mesh | Computational costly ... |
| | UCSD, 2024 SIGGRAPH | Represent the target surface using a coarse set of quadrangular patches , and add surface details using coordinate neural networks by displacing the patches (subdivision) | Single static mesh | Computational costly ... |

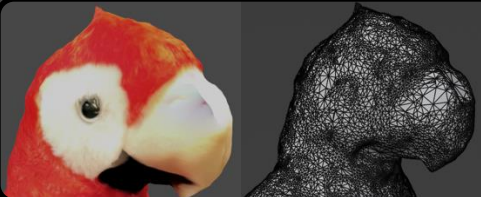
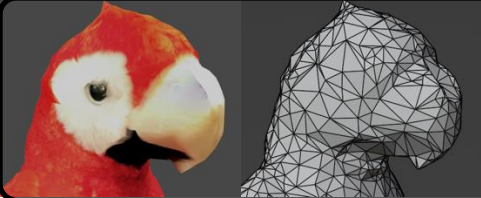
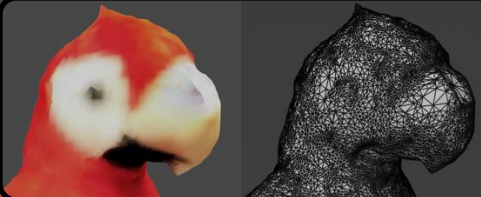
Compressing mesh and texture together

- We care about the final rendered image quality – so we need to optimize a function that compresses both mesh texture and mesh together
 - Need to effectively allocate bits for mesh and texture

Compressing mesh and texture together

- We care about the final rendered image quality – so we need to optimize a function that compresses both mesh texture and mesh together
 - Need to effectively allocate bits for mesh and texture



| | | |
|--|-----------------------------------|-------|
|  | Original texture Original mesh | 164MB |
|  | Original texture Low-res mesh | 8.2MB |
|  | Low-res texture Original mesh | 8.2MB |

Mesh Compression

- This lecture has focused mainly on compression efficiency
 - Almost all of the algorithms are computationally very expensive
 - None of the dynamic or time varying methods can run in real-time – so only suitable for offline stored applications

Summary of the Lecture

- Mesh compression
 - Vertex, connectivity, texture compression
 - Static
 - Dynamic or time varying
 - Progressive