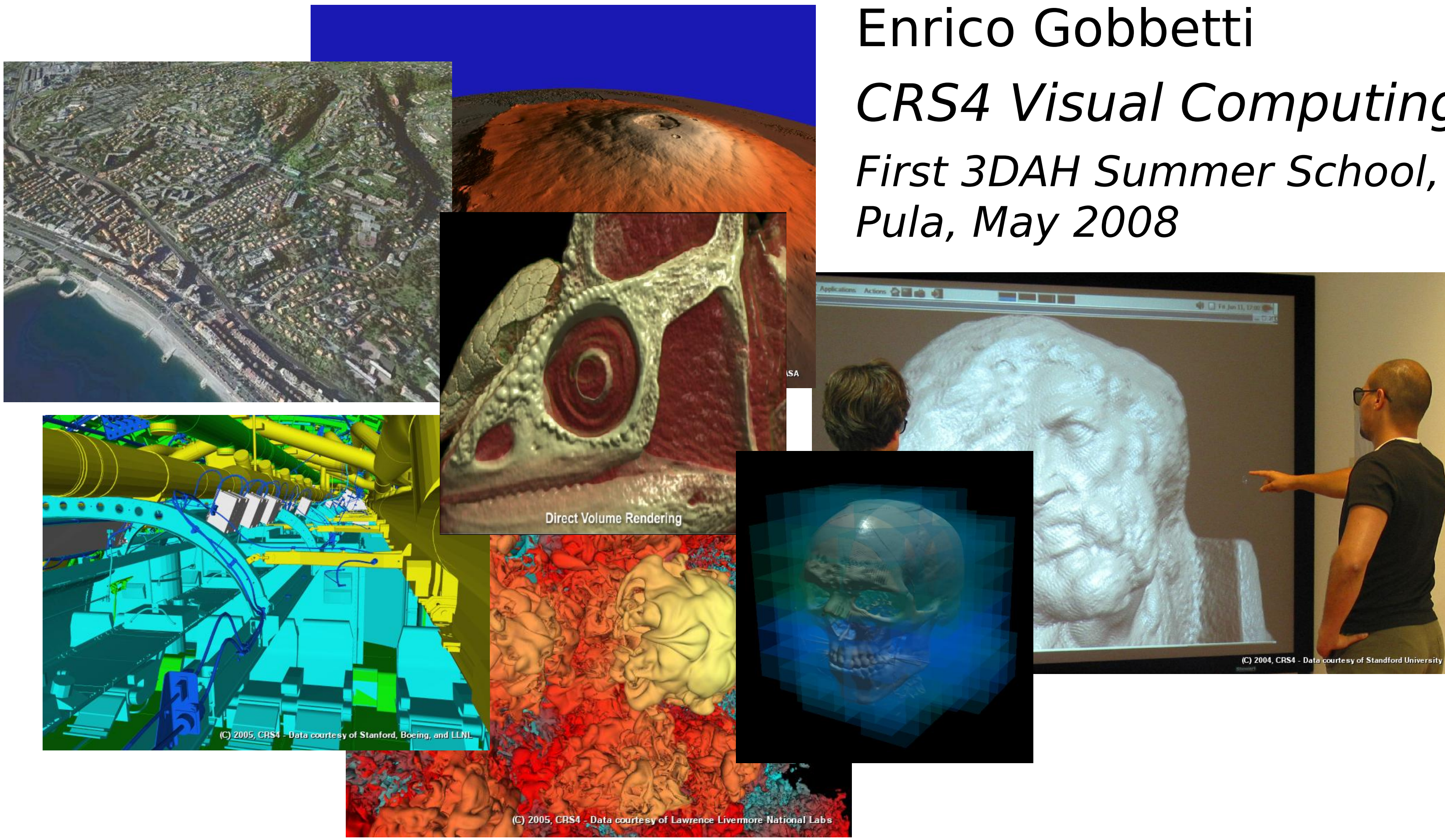
Technical strategies for massive model visualization

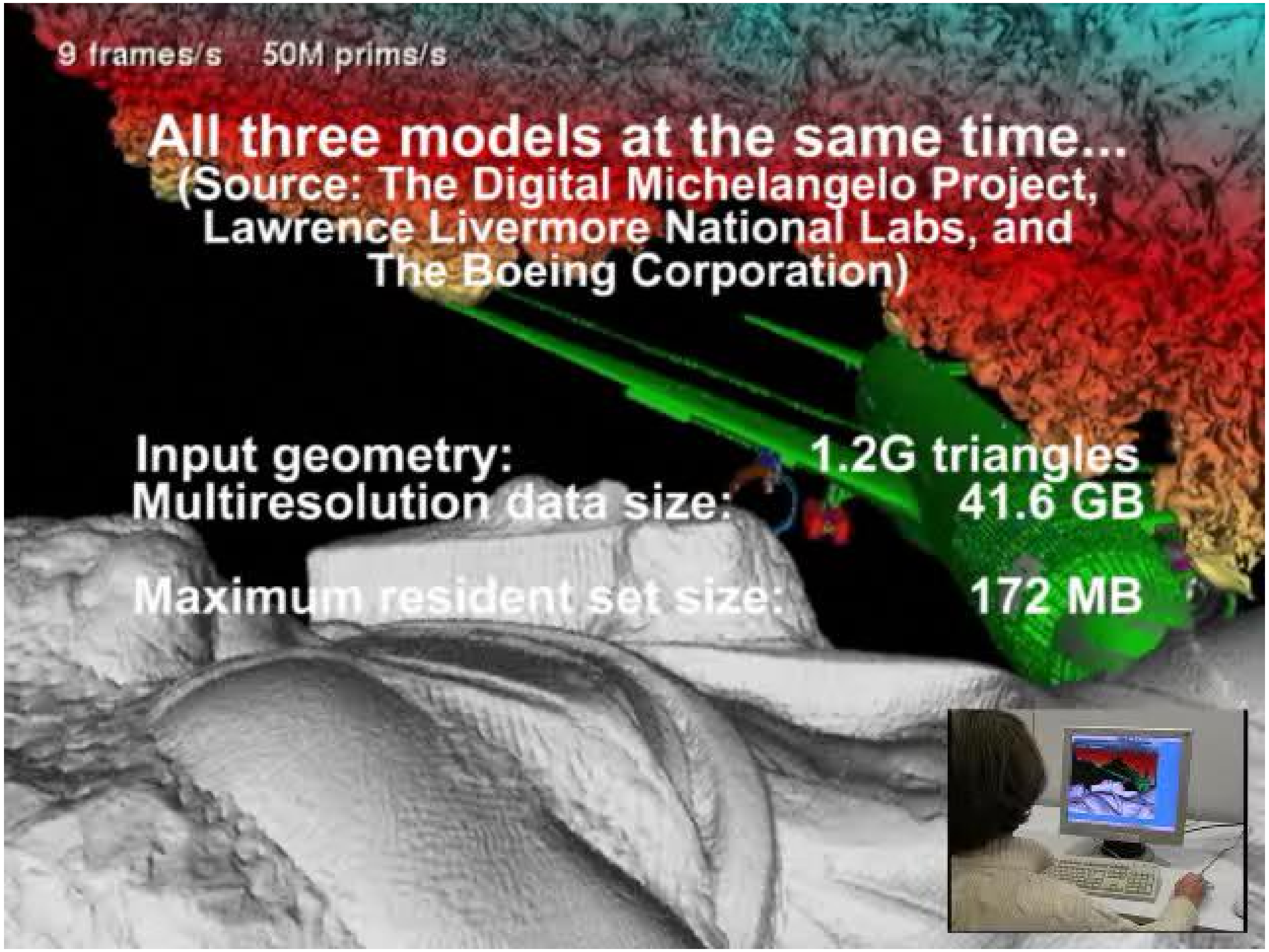
Enrico Gobbetti

CRS4 Visual Computing

First 3DAH Summer School, Pula, May 2008

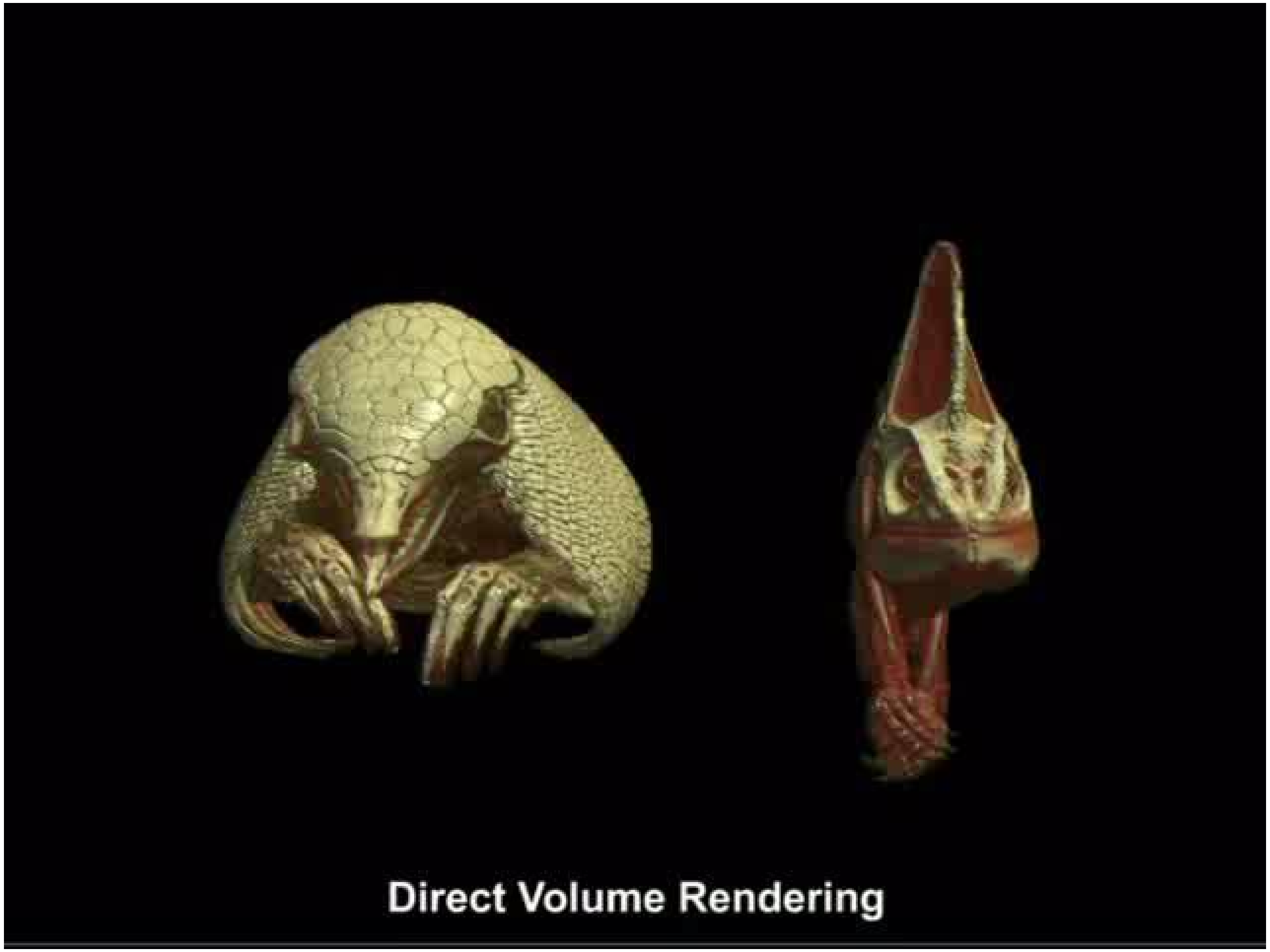


Goal: interactive inspection of massive models on PC platforms...



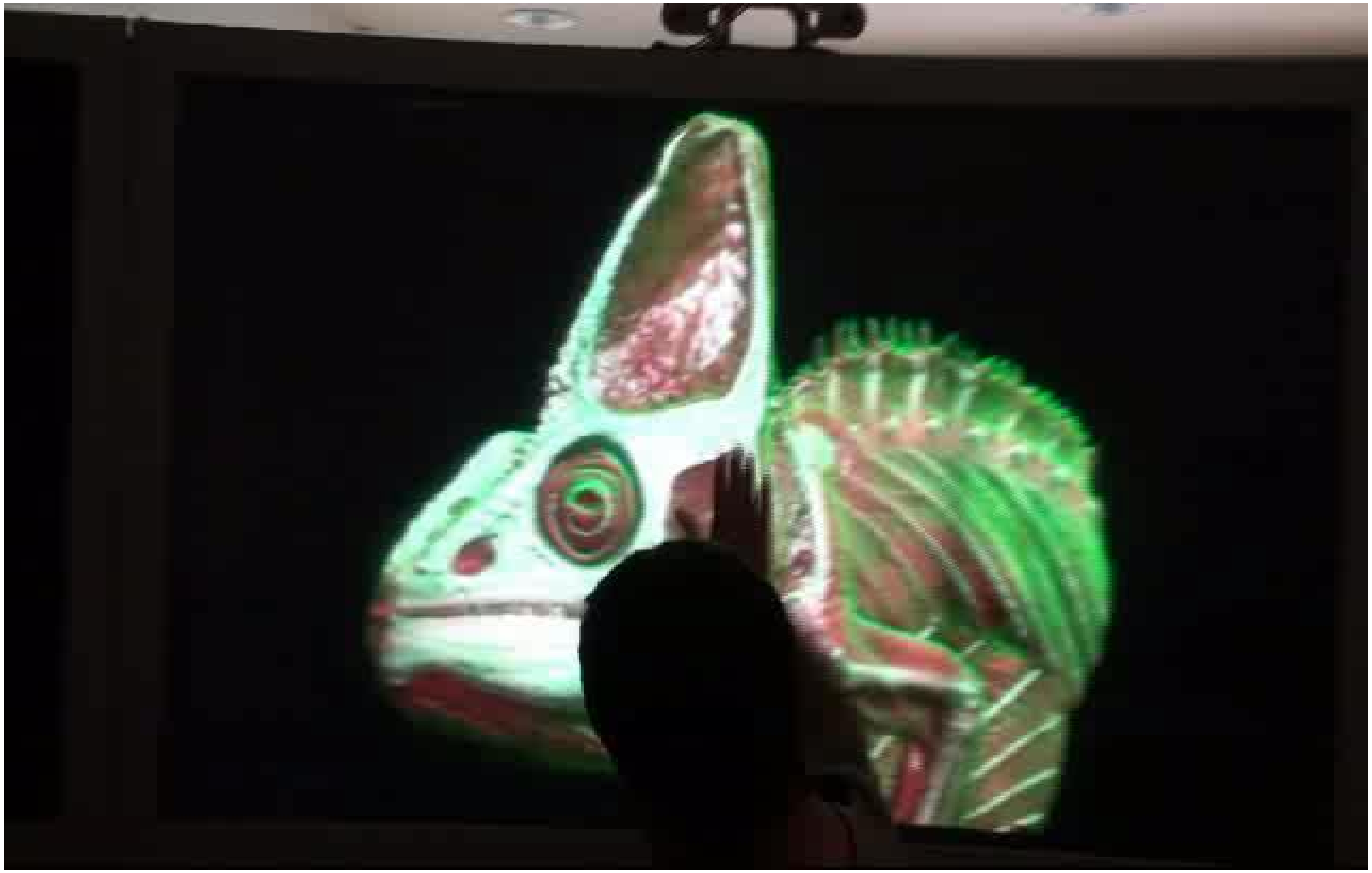
Xeon 2.4GHz / 1GB RAM / 70GB SCSI 320 Disk / NVIDIA 6800GTS

Goal: interactive inspection of massive models on PC platforms...



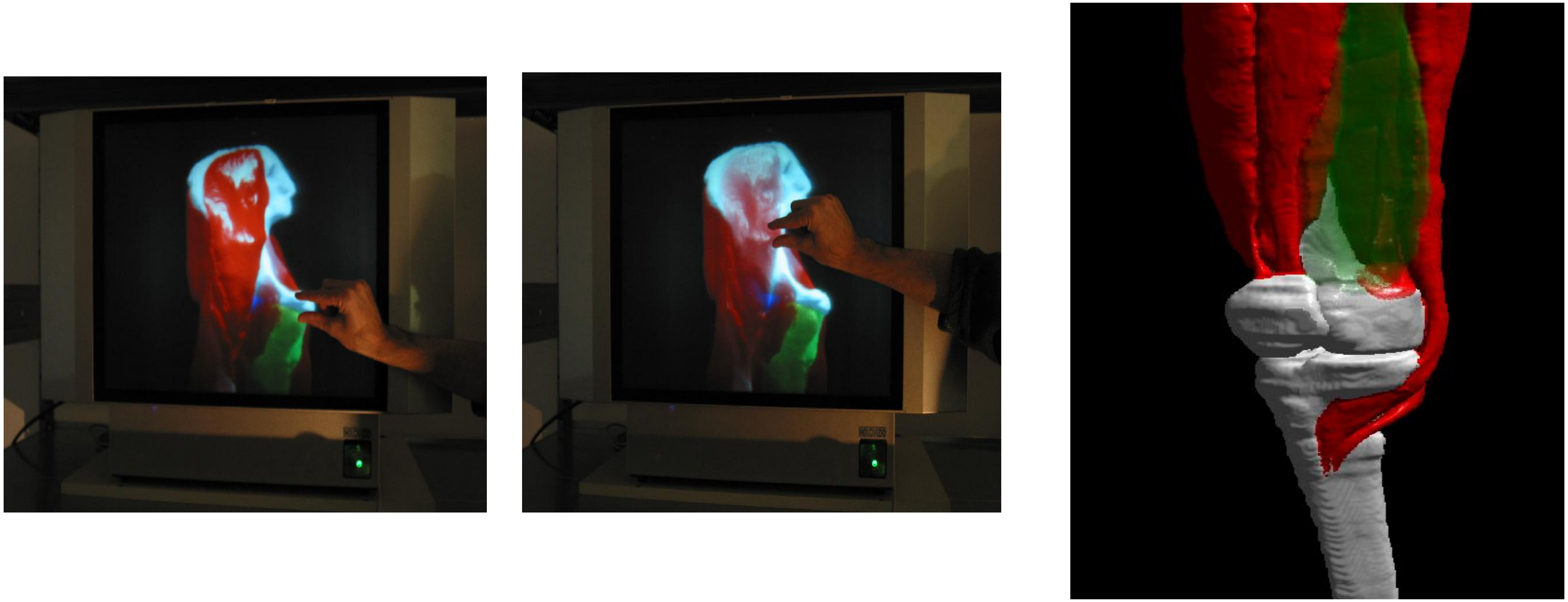
Xeon 2.4GHz / 1GB RAM / 70GB SCSI 320 Disk / NVIDIA 8800GTX

Goal: interactive inspection of massive models on PC platforms...



1GVoxel datataset rendered on a 33Mpixel light field display powered by 36 x NVIDIA 8800 GTS

Goal: interactive inspection of massive models on PC platforms...

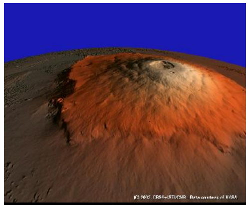


Volumetric renderings of 3DAH segmented leg dataset

Application domains / data sources

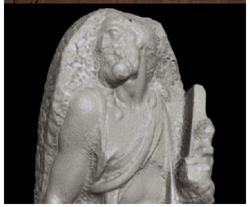
Local Terrain Models

2.5D – Flat – Dense regular sampling



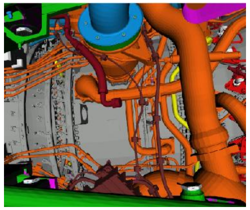
Planetary terrain models

2.5D – Spherical – Dense regular sampling



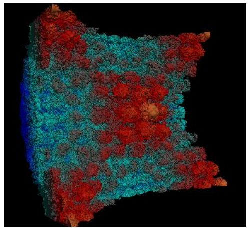
Laser scanned models

3D – Moderately simple topology – low depth complexity - dense



CAD models

3D – complex topology – high depth complexity – structured - ‘ugly’ mesh



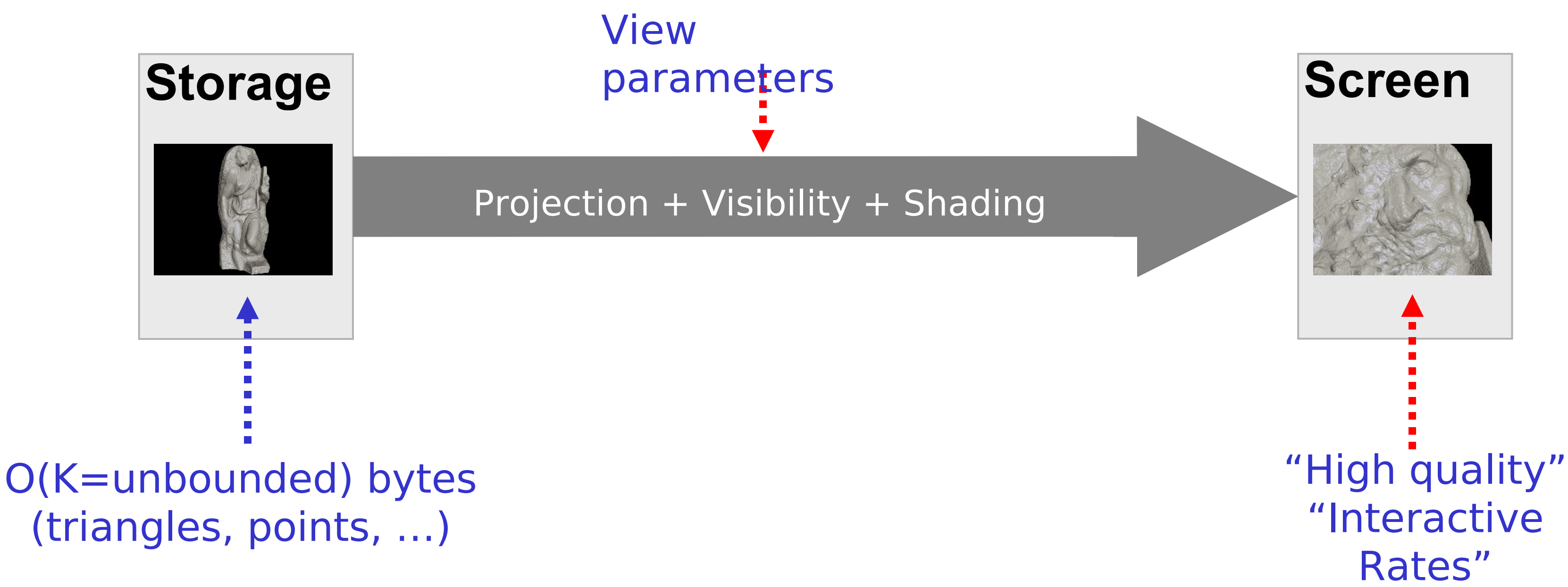
Natural objects / Simulation results

3D – complex topology + high depth complexity + unstructured/high frequency details

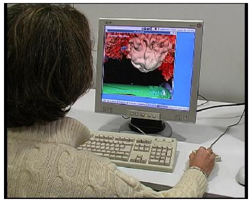
- Many important application domains
- Today’s models exceed
  - $O(10^{8}$-$10^{10})$ samples
  - $O(10^{9})$ bytes
- Varying
  - Dimensionality
  - Topology
  - Sampling distribution

The (minimal) challenge

- Explore very large models at interactive rates
  - Update screen at "interactive rates" as viewpoint changes

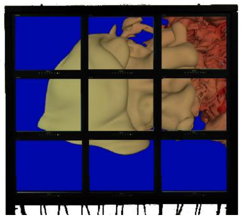


Interactive rendering constraints

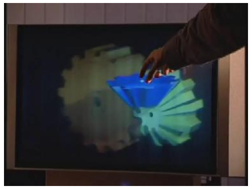


Regular desktop displays
~1M pixels

Geowall-type displays
~1-10M pixels, stereo



Tiled high resolution displays
~10-100M pixels



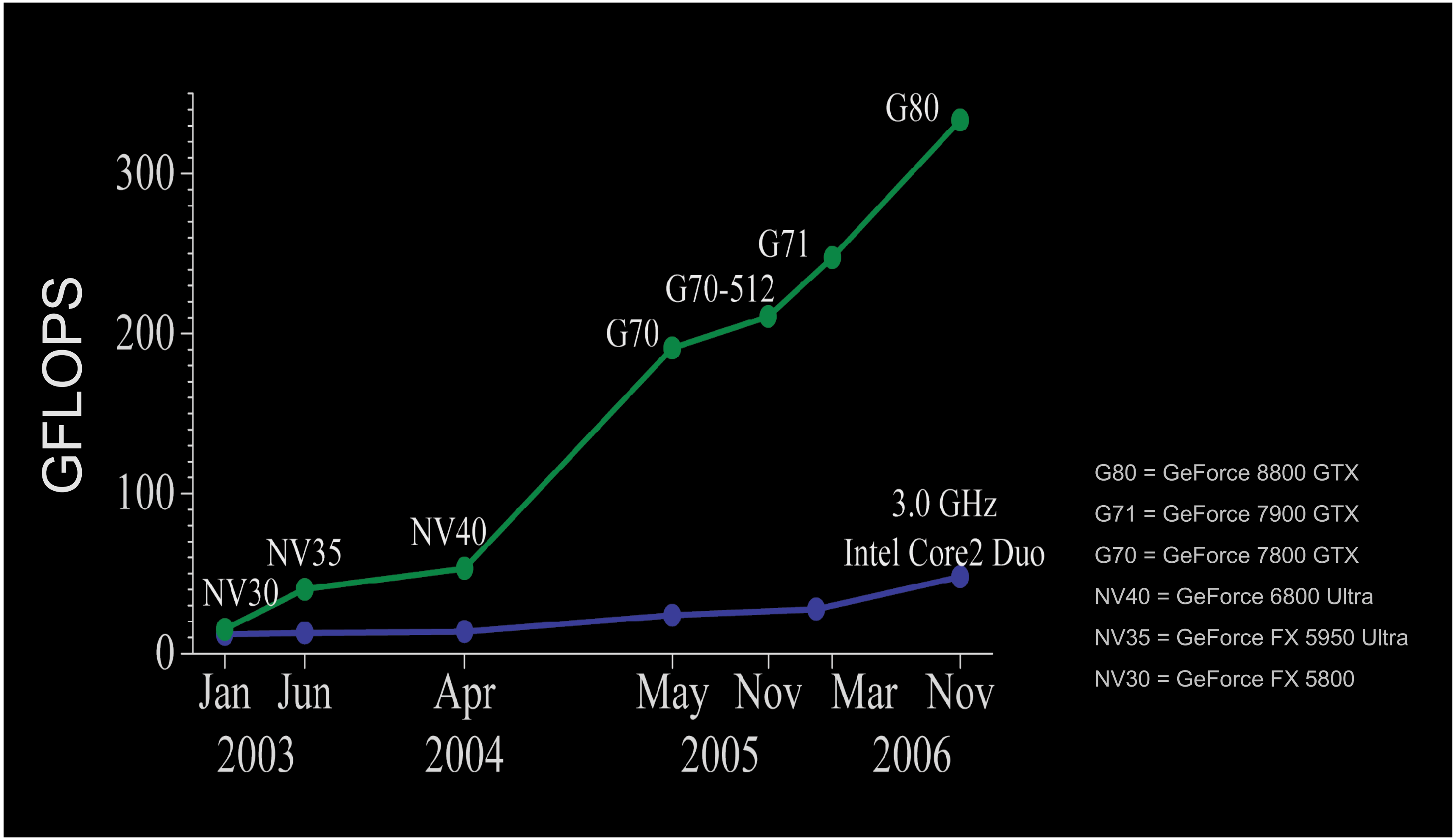
3D displays
~10-100M pixels, holo

- Frequency, latency, resolution, should match human capabilities…
  - … or at least output device's ones!
- On today's displays
  - Frequency: 10-100Hz
  - Latency: ~0.1s
  - Resolution: $O(10^6\text{-}10^7)$ pix

Powerful Hardware: a Solution?

- CPUs/GPUs are now amazingly fast!
  - Single dual-core 3GHz Opteron: ~20GFLOPS
  - Playstation 3's CELL: ~180GFLOPS
  - NVIDIA 8800 GPUs: ~340GFLOPS
- Exponential growth is continuing!
  - … mainly because of increased parallelism:
    - Multi-core CPUs / Multi-pipe GPUs
    - Generalized parallel graphics architectures
  - Multi-core 1TFLOP CPUs already on the horizon…
  - … not to talk about 1TFLOP GPUs…

Powerful Hardware: a Solution?



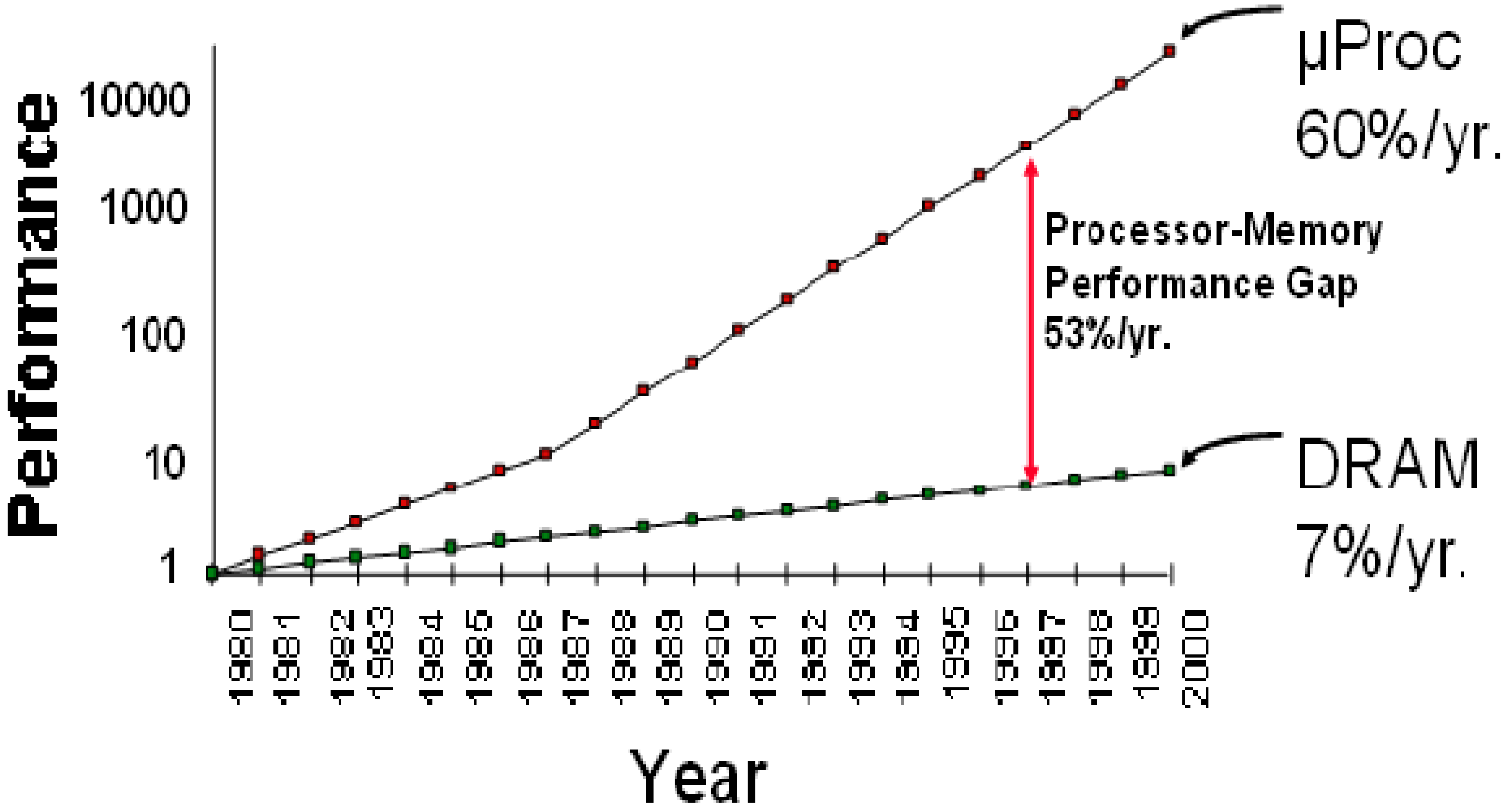
Powerful Hardware: a Solution? No!

- Exponential growth in model complexity outpaces hardware performance growth
  - Current large model complexity is minimal compared to real world complexity
    - … models in film industry are far more complex than those used in real-time apps
  - CPUs are also used to **generate** models
    - … today's large models are tomorrow's small ones…

Powerful Hardware: a Solution? No!

- Hardware excels at computational tasks with good memory locality
  - … the gap between computational performance and bandwidth throughout the memory hierarchy is growing!
  - … work from cache!
- The main problem of massive models is that they require huge amount of memory!
  - … memory locality??
  - … cannot cache an entire large model!

Powerful Hardware: a Solution? No!

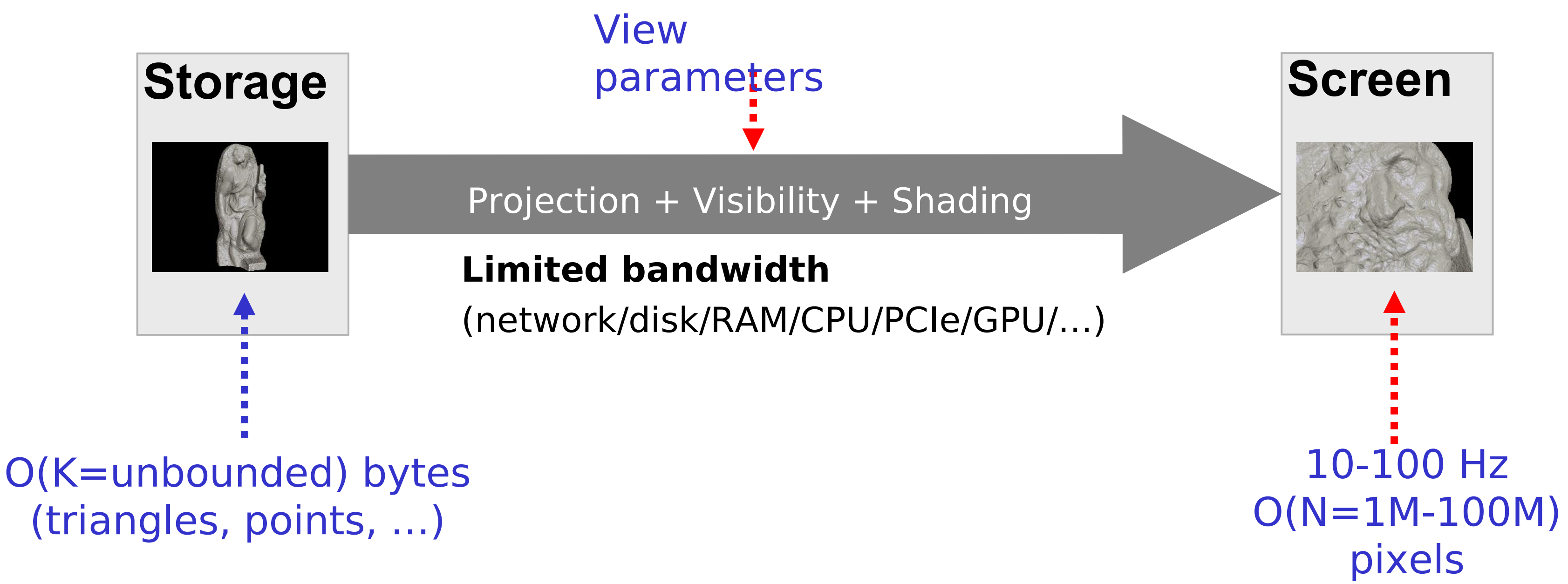


Powerful Hardware: a Solution?

- The challenge is to find methods able to capture as much performance growth as possible
  - … transform the problem into forms that are handled well by current hardware
- Hardware is not a solution by itself, but it dictates which solutions are good in practice and which ones are doomed to be inefficient!

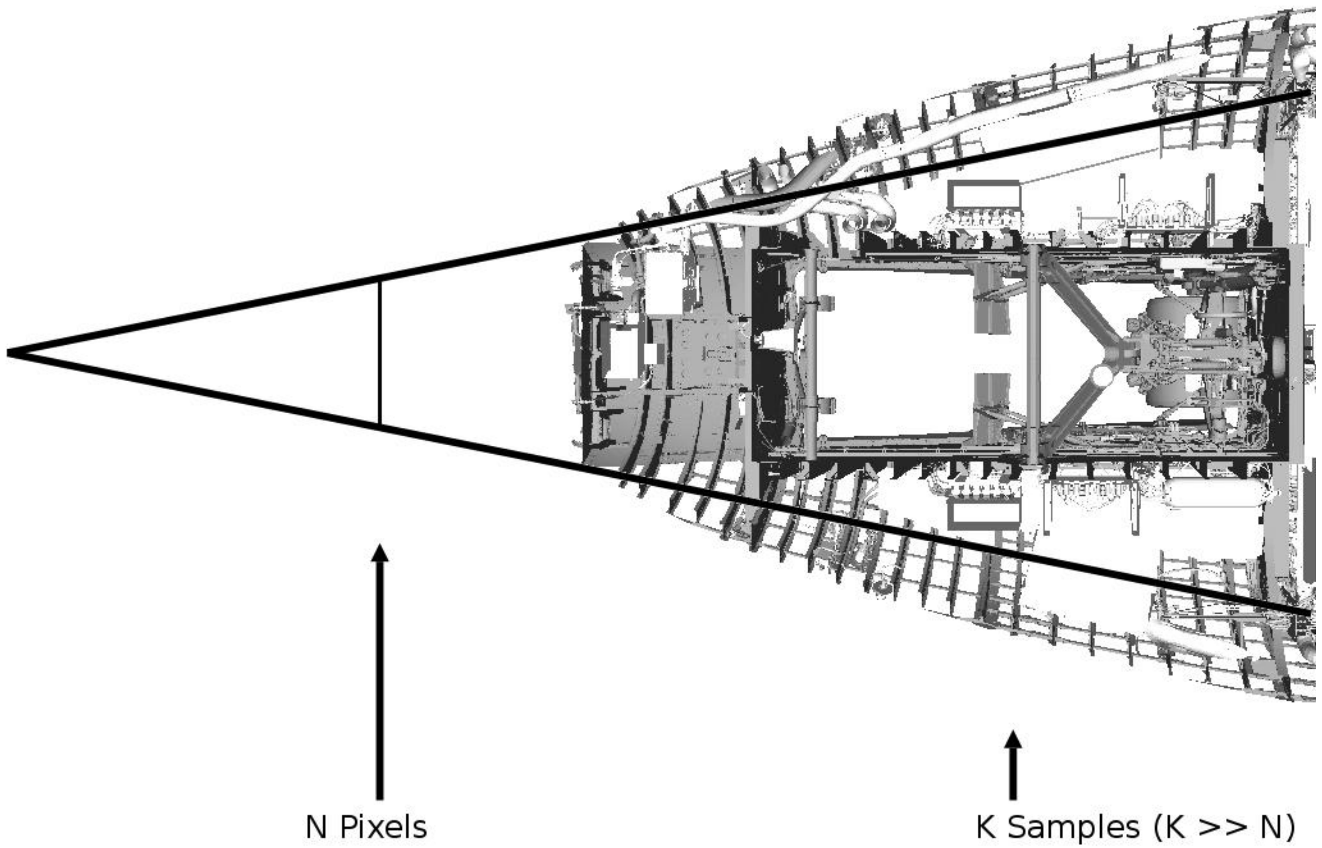
A real-time data filtering problem!

- Models of unbounded complexity on limited computers
  - We assume **less data on screen (N) than in model (K →∞)**
  - Need for **output-sensitive** techniques (O(N), not O(K))
  - Need for **memory-efficient** techniques (maximize cache hits!)



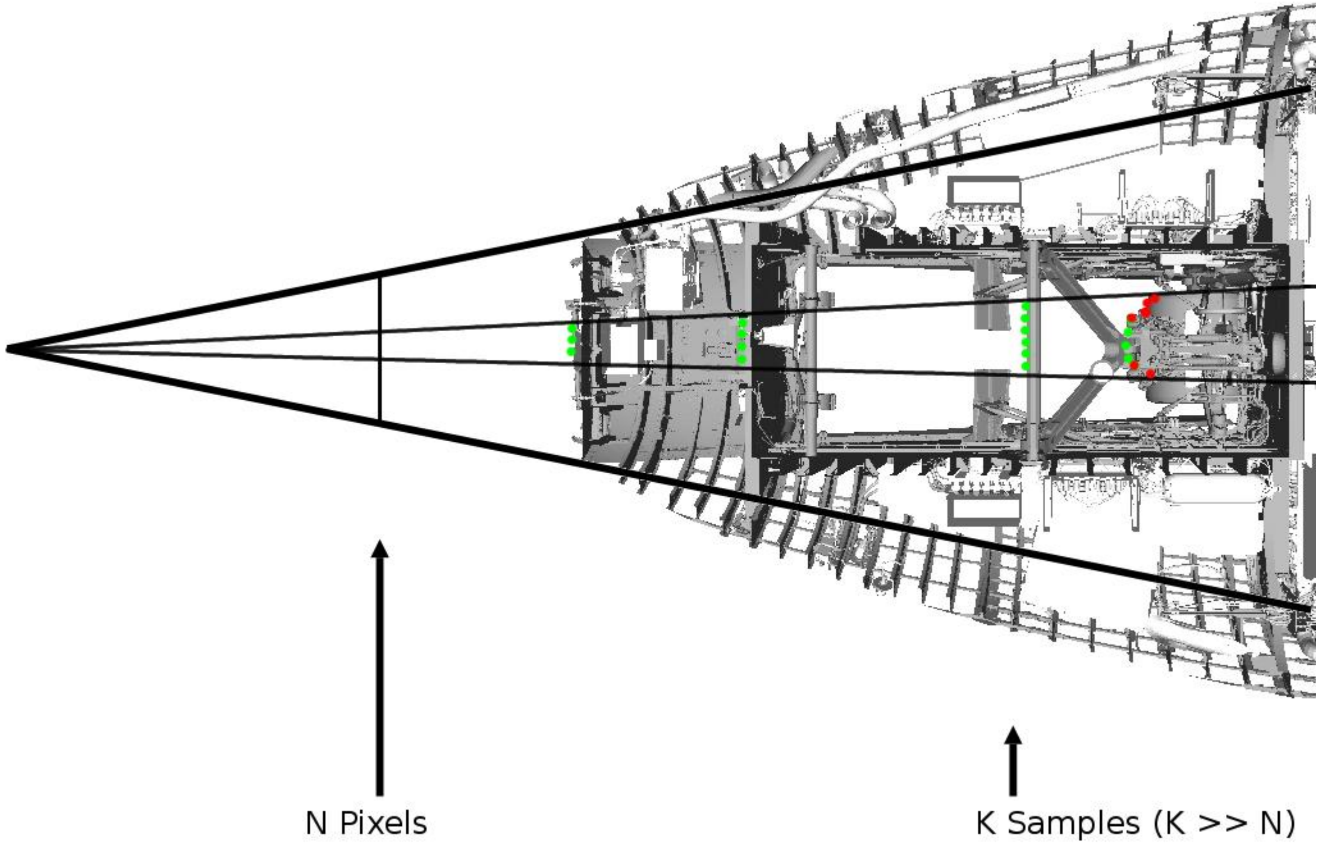
Output-sensitive techniques

Goal: Time/Memory Complexity = O(N) (independent of K)



Output-sensitive techniques

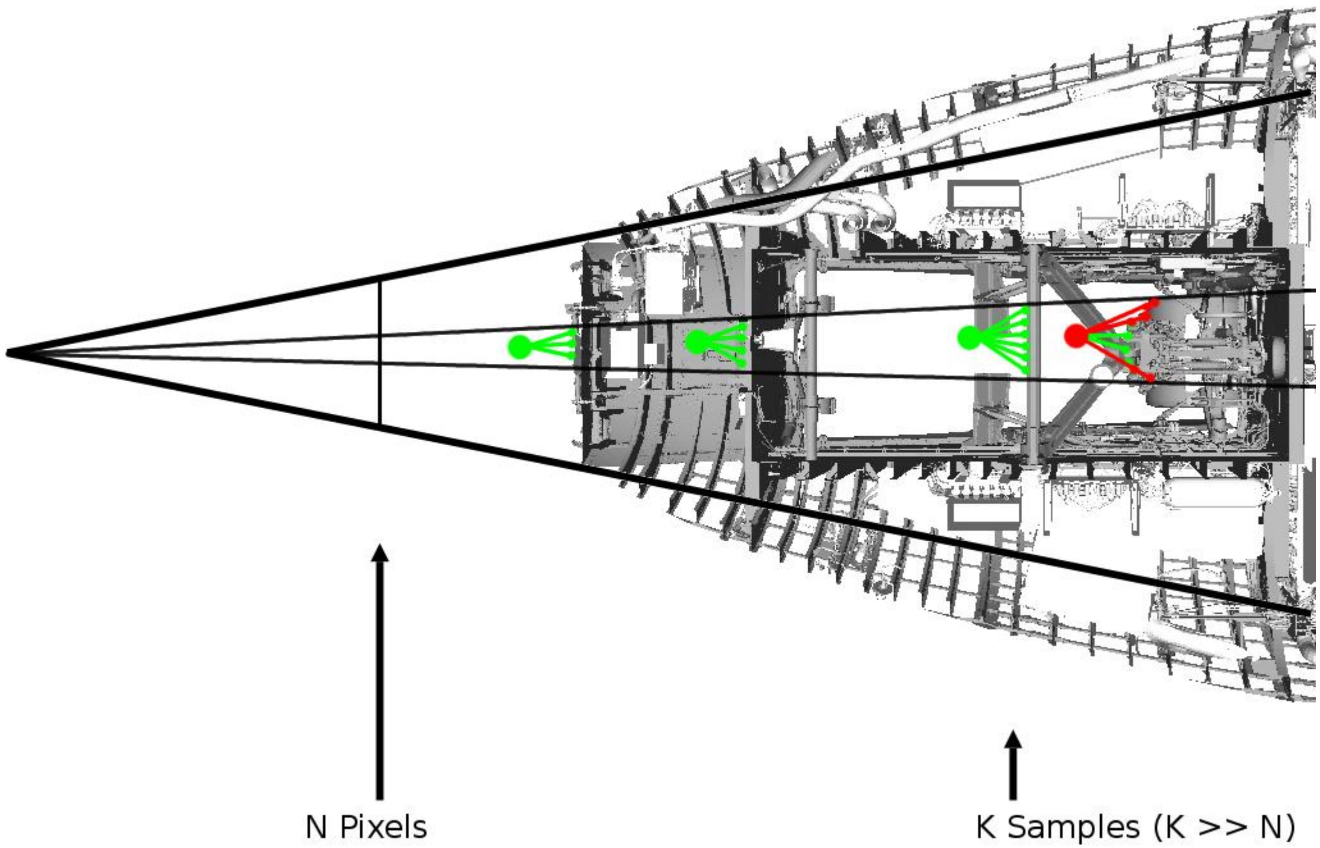
Goal: Time/Memory Complexity = O(N) (independent of K)



Output-sensitive techniques

Goal: Time/Memory Complexity = O(N) (independent of K)

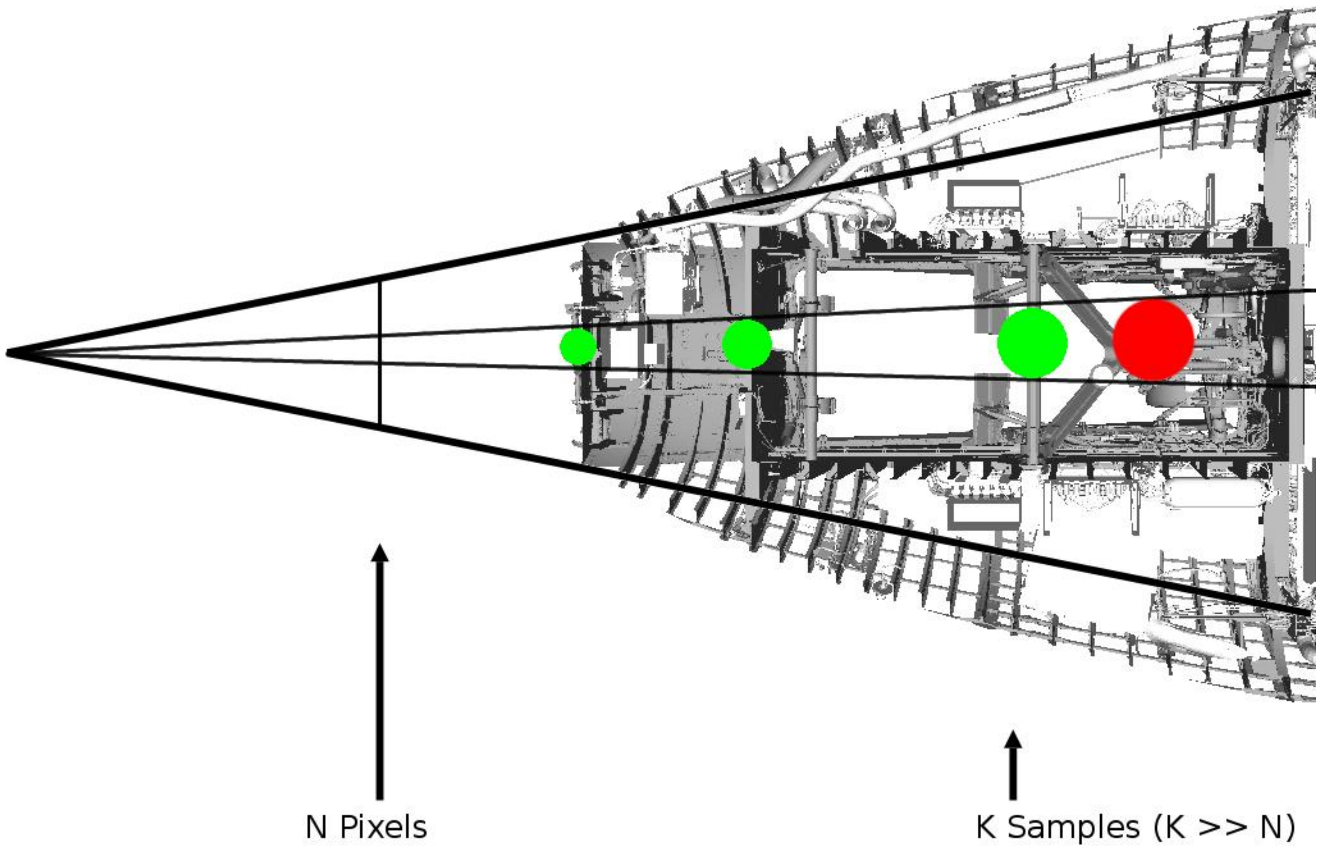
Multiresolution + ...



Output-sensitive techniques

Goal: Time/Memory Complexity = O(N) (independent of K)

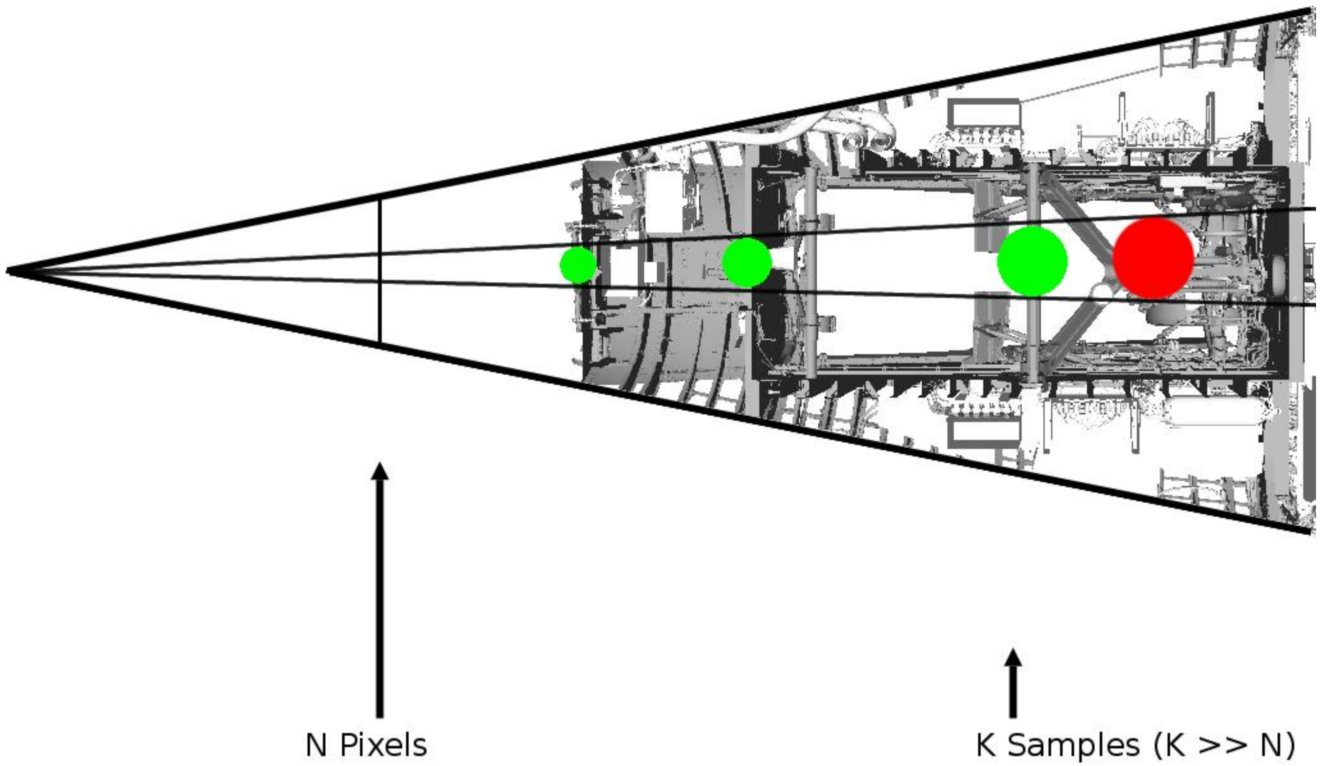
Multiresolution + View dependent LOD selection + ...



Output-sensitive techniques

Goal: Time/Memory Complexity = O(N) (independent of K)

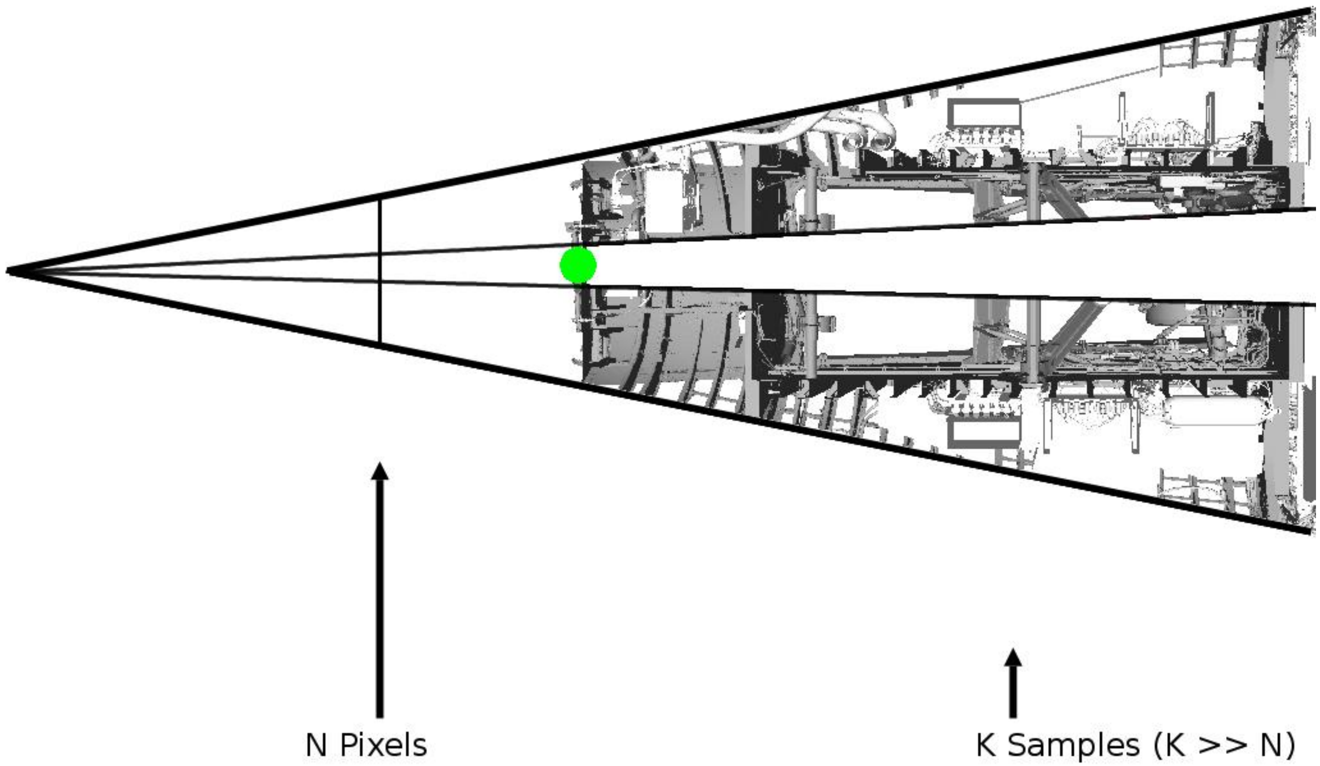
Multiresolution + View dependent LOD selection + View culling +



Output-sensitive techniques

Goal: Time/Memory Complexity = O(N) (independent of K)

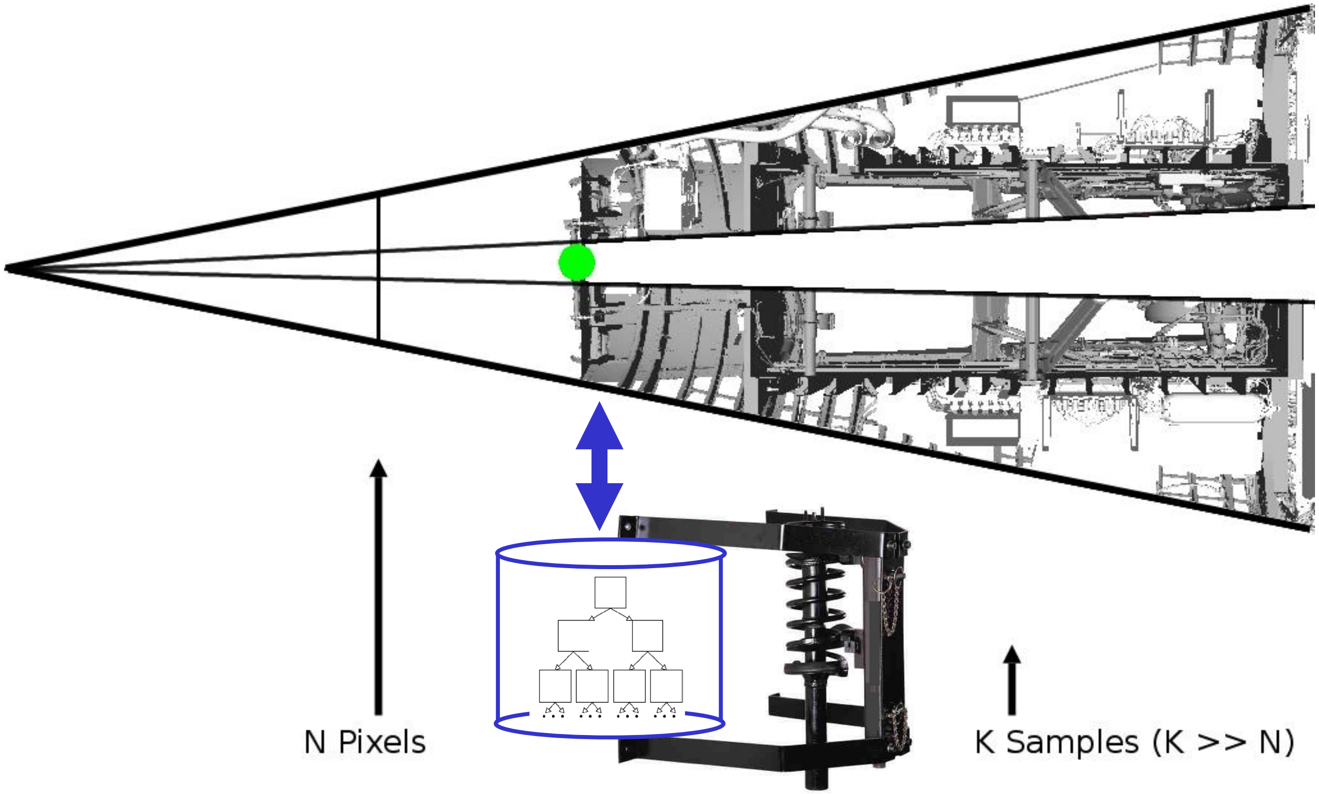
- Multiresolution + View dependent LOD selection + View culling + Occlusion culling + …



Output-sensitive techniques

Goal: Time/Memory Complexity = O(N) (independent of K)

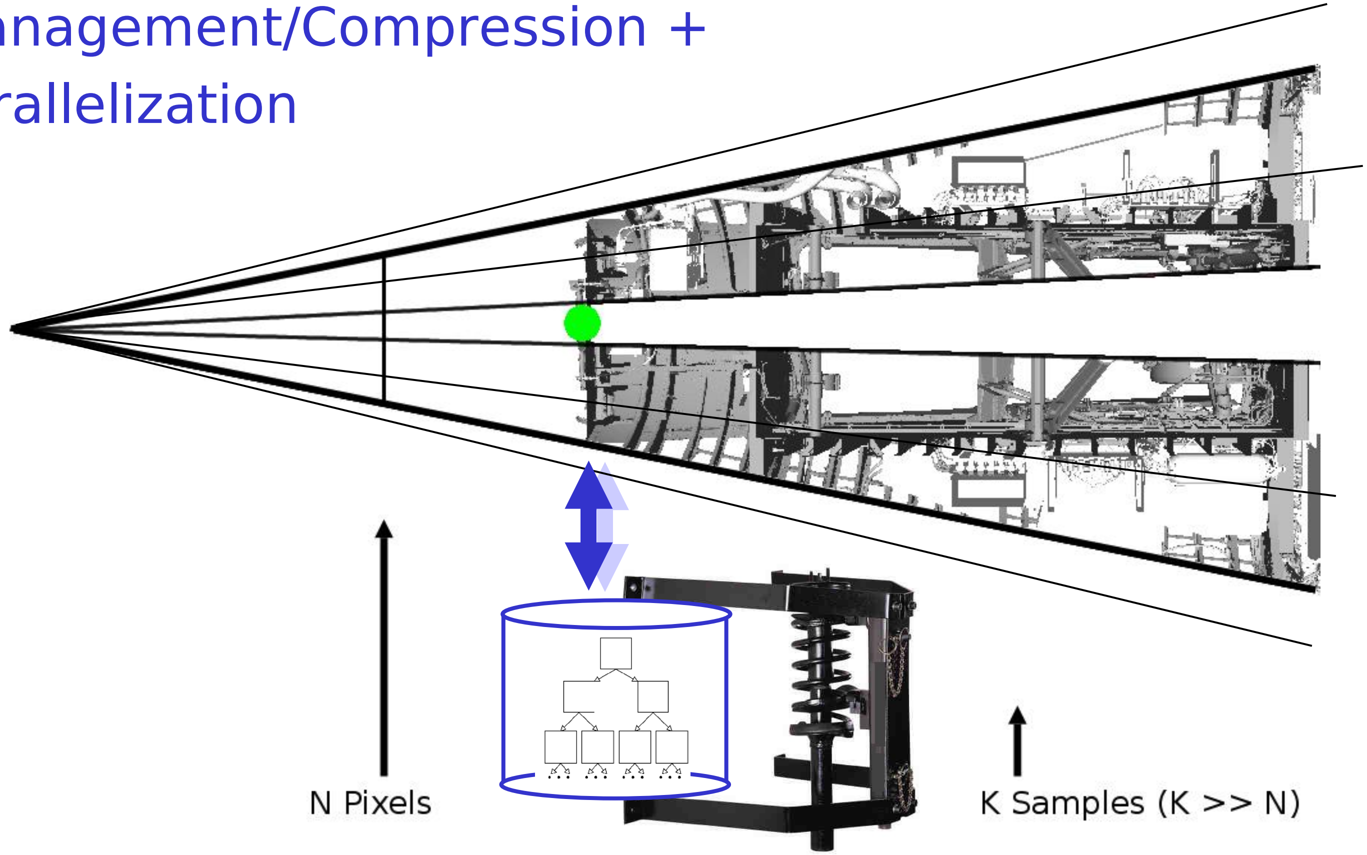
- Multiresolution + View dependent LOD selection + View culling + Occlusion culling + External memory management/Compression



Output-sensitive techniques

Goal: Time/Memory Complexity = O(N) (independent of K)

- Multiresolution + View dependent LOD selection + View culling + Occlusion culling + External memory management/Compression + Parallelization

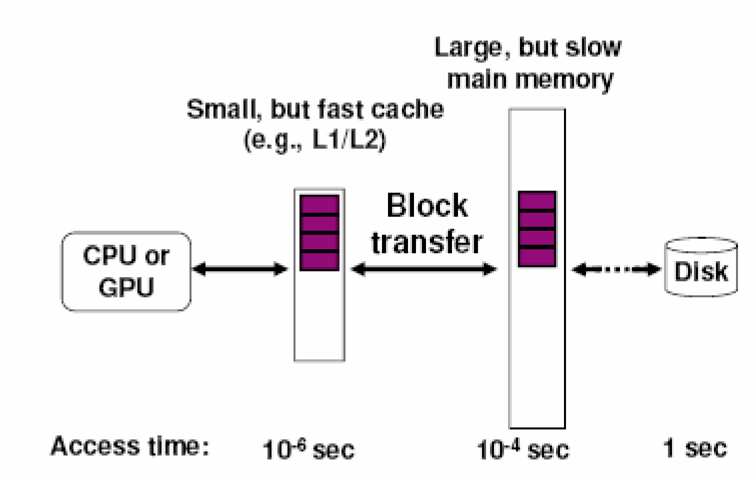


Ouput sensitive techniques

- All techniques must combine
  - An efficient memory management subsystem
  - A multiresolution and spatial subdivision structure
    - Visual/geometric approximations
    - Spatial indexing
  - A view-dependent renderer
    - LOD culling
    - Visibility culling
- Relative weight of components varies depending on model kind
  - E.g., LODs more important than occlusion for flight simulators

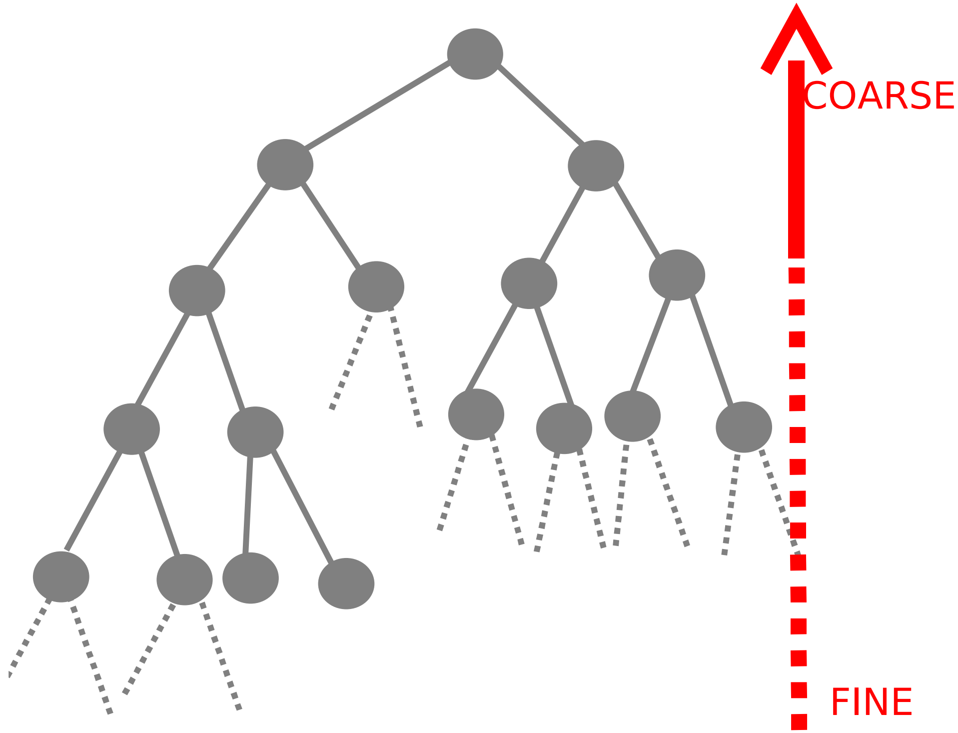
Efficient memory management

- Goal is to reduce/avoid memory access latency
  - Maximize fast cache utilization
- Techniques
  - Reorder computations
    - Streaming, multiresolution
  - Reorder data structures
    - Cache efficient layouts
  - Manage memory in blocks
    - Model partitioning, streamlined low-level I/O
  - Reduce memory consumption
    - Multiresolution, compressed representations



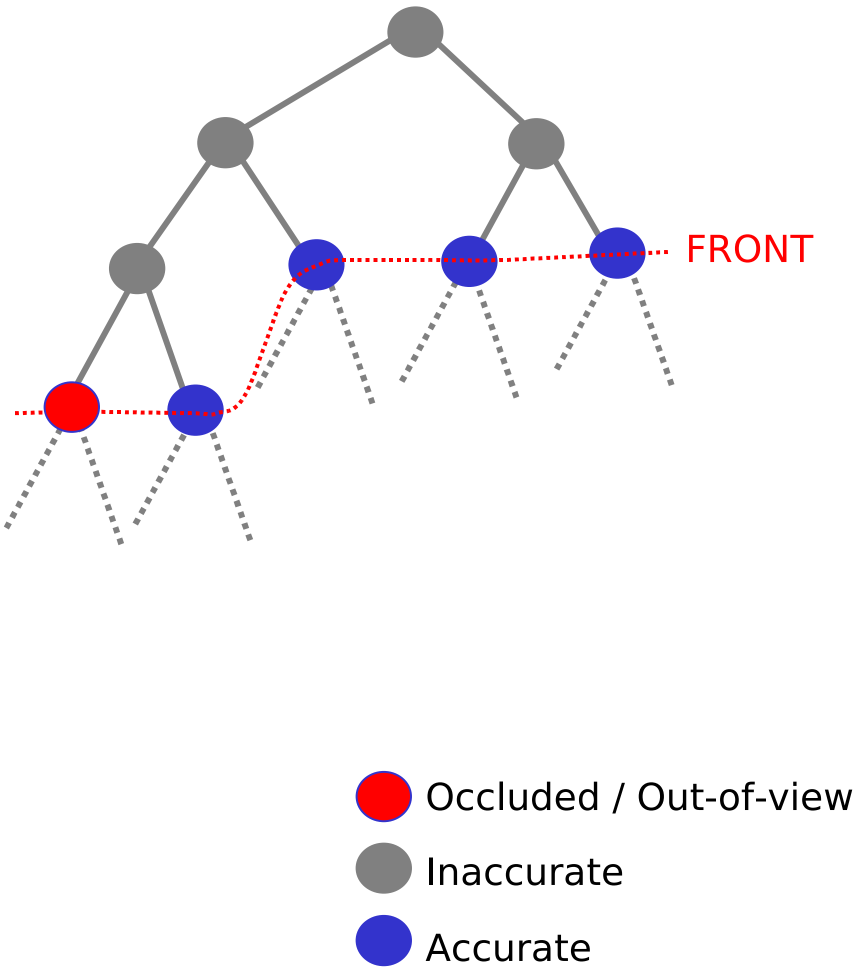
Output-sensitive techniques

- At **preprocessing** time: build MR hierarchy
  - Data prefiltering!
  - Visibility + simplification



Output-sensitive techniques

- At **preprocessing** time: build MR hierarchy
  - Data prefiltering!
  - Visibility + simplification
  - Not output sensitive
- At **run-time**: selective view-dependent refinement from out-of-core data
  - Must be output sensitive
  - Access to prefiltered data under real-time constraints
  - Visibility + LOD



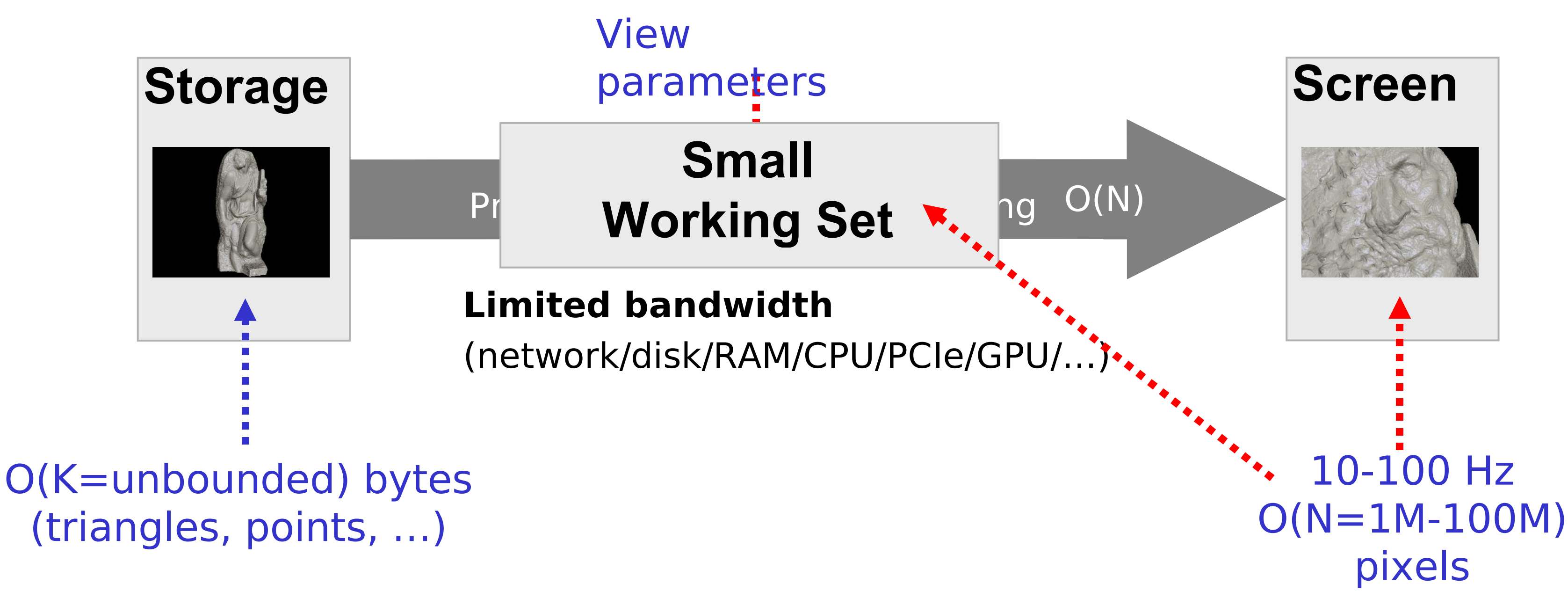
Two main rendering techniques

- Rasterization + Z-buffering (GPU)
  - Start from model
- Ray-tracing (CPU/RPU/GPU)
  - Start from screen

- For large models, methods share many common points
  - Similar hierarchical structures
  - Need for approximate representations to build multiresolution hierarchies
  - Similar memory management subsystem, typically exploiting spatial/temporal coherence

Wrap-up: A real-time data filtering problem!

- Models of unbounded complexity on limited computers
  - We assume **less data on screen (N) than in model (K →∞)**
  - Need for **output-sensitive** techniques (O(N), not O(K))
  - Need for **memory-efficient** techniques (maximize cache hits!)

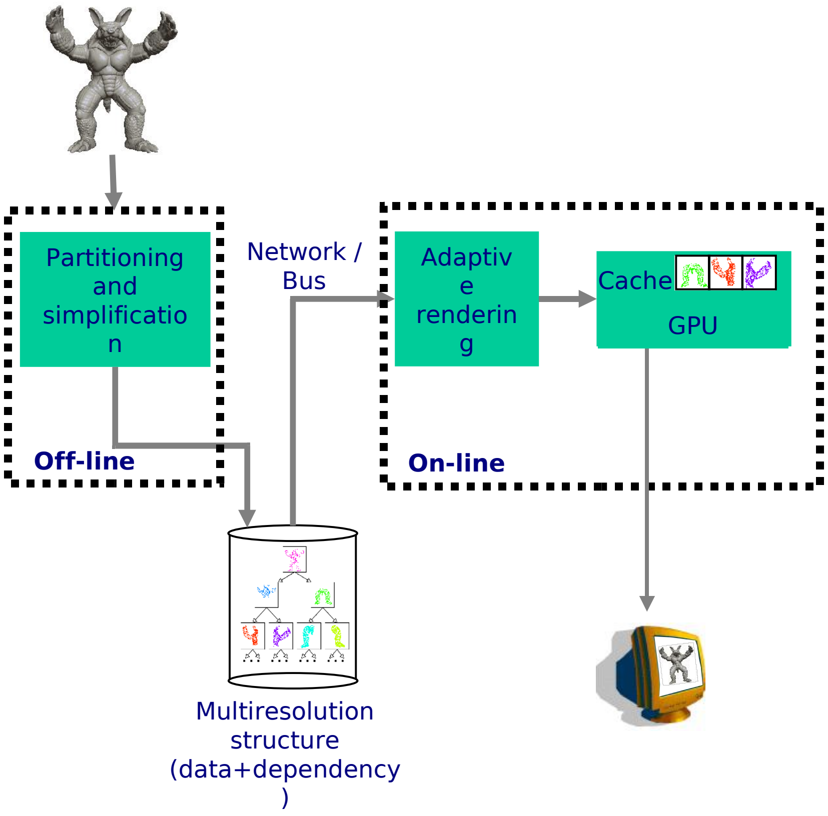


Ouput sensitive technique

- All techniques must combine
  - An efficient memory management subsystem
  - A multiresolution and spatial subdivision structure
    - Visual/geometric approximations
    - Spatial indexing
  - A view-dependent renderer
    - LOD culling
    - Visibility culling

Our contributions
GPU-friendly output-sensitive techniques

- Underlying ideas
  - **Chunk-based multiresolution structures**
    - Combine space partitioning + level of detail
    - Same structure used for visibility and detail culling
  - **Seamless combination of chunks**
    - Dependencies ensure consistency at the level of chunks
  - **Complex rendering primitives**
    - GPU programming features
    - Curvilinear patches, view-dependent voxels, …
  - **Chunk-based external memory management**
    - Compression/decompression, block transfers, caching



Our contributions

GPU-friendly output-sensitive techniques

***-BDAM – Local and Global Terrain Models**

Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Ponchio/Scopigno (CNR)

EG 2003, IEEE Viz 2003, EG 2005

Adaptive Tetrapuzzles – Dense meshes

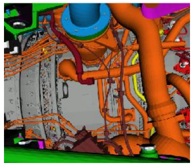
Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Ponchio/Scopigno (CNR)

SIGGRAPH 2004

Layered Point Clouds – Dense clouds

Gobbetti/Marton (CRS4)

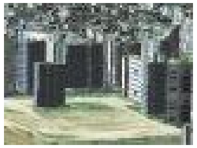
SPBG 2004 / Computers & Graphics 2004



Far Voxels – General

Gobbetti/Marton (CRS4)

SIGGRAPH 2005



Blockmaps – Hybrid volumetric city model

Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Di Benedetto/Scopigno (CNR)

EG 2007

MOVR – Volumetric models

Gobbetti/Marton/Iglesias Guitian (CRS4)

CGI 2008

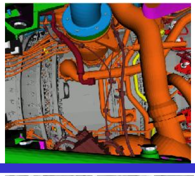
Our contributions

GPU-friendly output-sensitive techniques

***-BDAM – Local and Global Terrain Models**
Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Ponchio/Scopigno (CNR)
EG 2003, IEEE Viz 2003, EG 2005

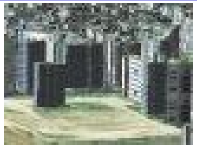
Adaptive Tetrapuzzles – Dense meshes
Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Ponchio/Scopigno (CNR)
SIGGRAPH 2004

Layered Point Clouds – Dense clouds
Gobbetti/Marton (CRS4)
SPBG 2004 / Computers & Graphics 2004



Far Voxels – General
Gobbetti/Marton (CRS4)
SIGGRAPH 2005

RASTERIZATION



Blockmaps – Hybrid volumetric city model
Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Di Benedetto/Scopigno (CNR)
EG 2007

MOVR – Volumetric models
Gobbetti/Marton/Iglesias Guitian (CRS4)
CGI 2008

RAYCASTING

Our contributions

GPU-friendly output-sensitive techniques

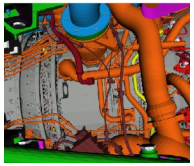
MESH-BASED FRAMEWORK

***-BDAM – Local and Global Terrain Models**
Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Ponchio/Scopigno (CNR)
EG 2003, IEEE Viz 2003, EG 2005

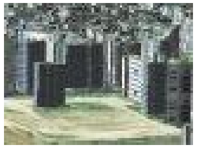
Adaptive Tetrapuzzles – Dense meshes
Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Ponchio/Scopigno (CNR)
SIGGRAPH 2004

MESH-LESS FRAMEWORK

Layered Point Clouds – Dense clouds
Gobbetti/Marton (CRS4)
SPBG 2004 / Computers & Graphics 2004



Far Voxels – General
Gobbetti/Marton (CRS4)
SIGGRAPH 2005



Blockmaps – Hybrid volumetric city model
Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Di Benedetto/Scopigno (CNR)
EG 2007

MOVR – Volumetric models
Gobbetti/Marton/Iglesias Guitian (CRS4)
CGI 2008

Our contributions

GPU-friendly output-sensitive techniques

***-BDAM – Local and Global Terrain Models**

Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Ponchio/Scopigno (CNR)

EG 2003, IEEE Viz 2003, EG 2005

Adaptive Tetrapuzzles – Dense meshes

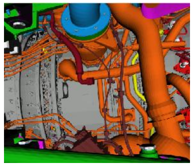
Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Ponchio/Scopigno (CNR)

SIGGRAPH 2004

Layered Point Clouds – Dense clouds

Gobbetti/Marton (CRS4)

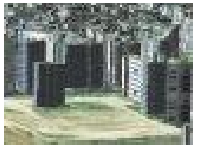
SPBG 2004 / Computers & Graphics 2004



Far Voxels – General

Gobbetti/Marton (CRS4)

SIGGRAPH 2005



Blockmaps – Hybrid volumetric city model

Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Di Benedetto/Scopigno (CNR)

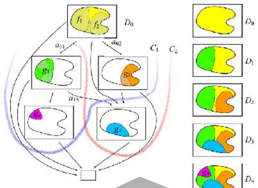
EG 2007

MOVR – Volumetric models

Gobbetti/Marton/Iglesias Guitian (CRS4)

CGI 2008

Specialize

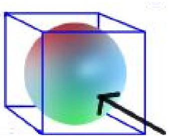


Chunked Multi-Triangulations

Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Ponchio/Scopigno (CNR) *IEEE Viz 2005*

Generalize

Specialize



View-dep. Volumetric Model

In progress

Generalize

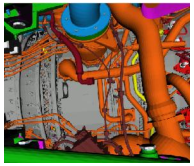
Our contributions

GPU-friendly output-sensitive techniques

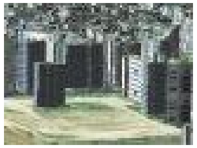
***-BDAM – Local and Global Terrain Models**
Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Ponchio/Scopigno (CNR)
EG 2003, IEEE Viz 2003, EG 2005

Adaptive Tetrapuzzles – Dense meshes
Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Ponchio/Scopigno (CNR)
SIGGRAPH 2004

Layered Point Clouds – Dense clouds
Gobbetti/Marton (CRS4)
SPBG 2004 / Computers & Graphics 2004



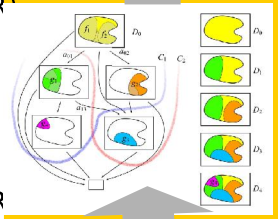
Far Voxels – General
Gobbetti/Marton (CRS4)
SIGGRAPH 2005



Blockmaps – Hybrid volumetric city model
Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Di Benedetto/Scopigno (CNR)
EG 2007

MOVR – Volumetric models
Gobbetti/Marton/Iglesias Guitian (CRS4)
CGI 2008

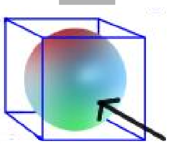
Specialize



Chunked Multi-Triangulations
Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Ponchio/Scopigno (CNR) *IEEE Viz 2005*

Generalize

Specialize



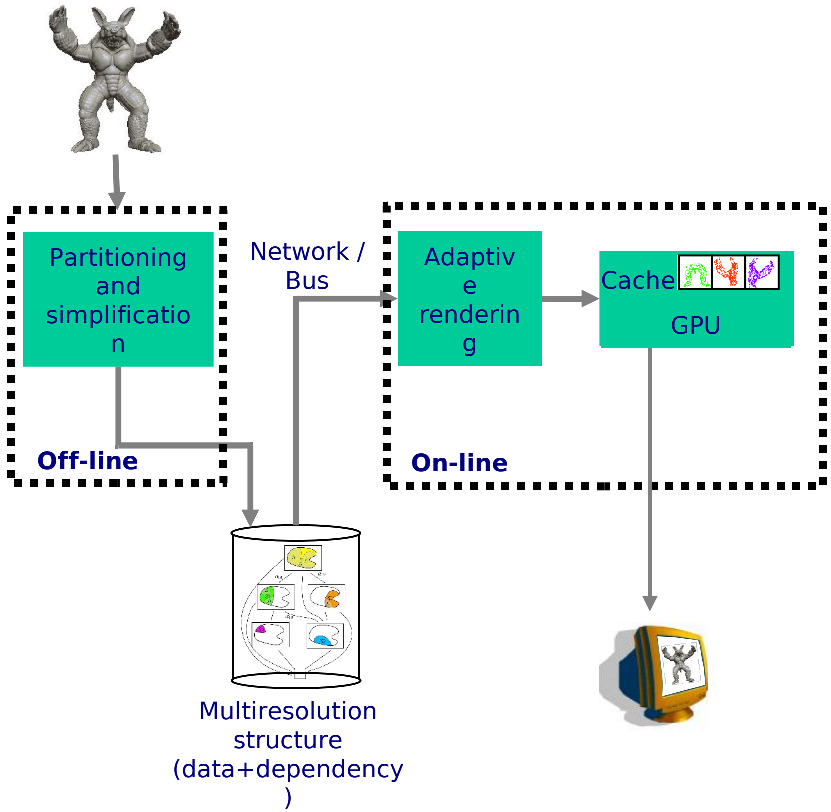
View-dep. Volumetric Model
In progress

Generalize

Chunked Multi Triangulations

The Multi Triangulation Framework

- Theoretical basis
  - MT multiresolution framework (Puppo 1996)
- Our contribution
  - GPU friendly implementation based on surface chunks with boundary constraints
  - Optimized implicit specializations (TetraPuzzles/V-Partitions)
  - Parallel out-of-core pre-processing and out-of-core run-time



Cignoni, Ganovelli, Gobbetti, Marton, Ponchio, and Scopigno.
Batched Multi Triangulation.
In *Proc. IEEE Visualization*. Pages 207-214. October 2005.

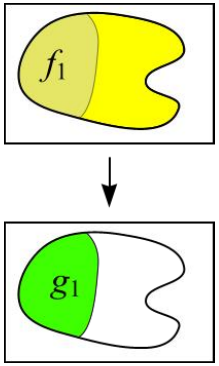
Chunked Multi Triangulations

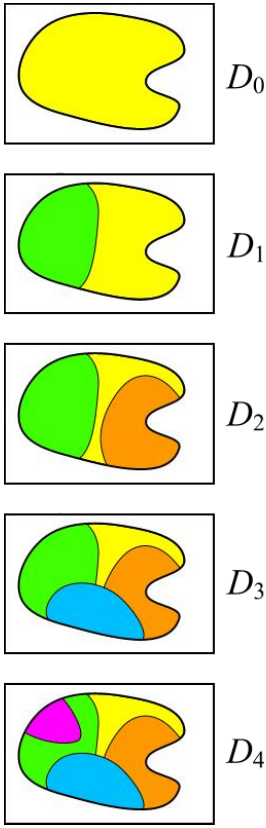
The Multi Triangulation Framework

- Consider a sequence of local modifications over a given description *D*
  - Each modification replaces a portion of the domain with a different conforming portion (simplified)
  - f_1 floor
  - g_1 the new fragment

$$D' = D \setminus f \cup g$$

$$D_{i+1} = D_i \oplus g_{i+1}$$

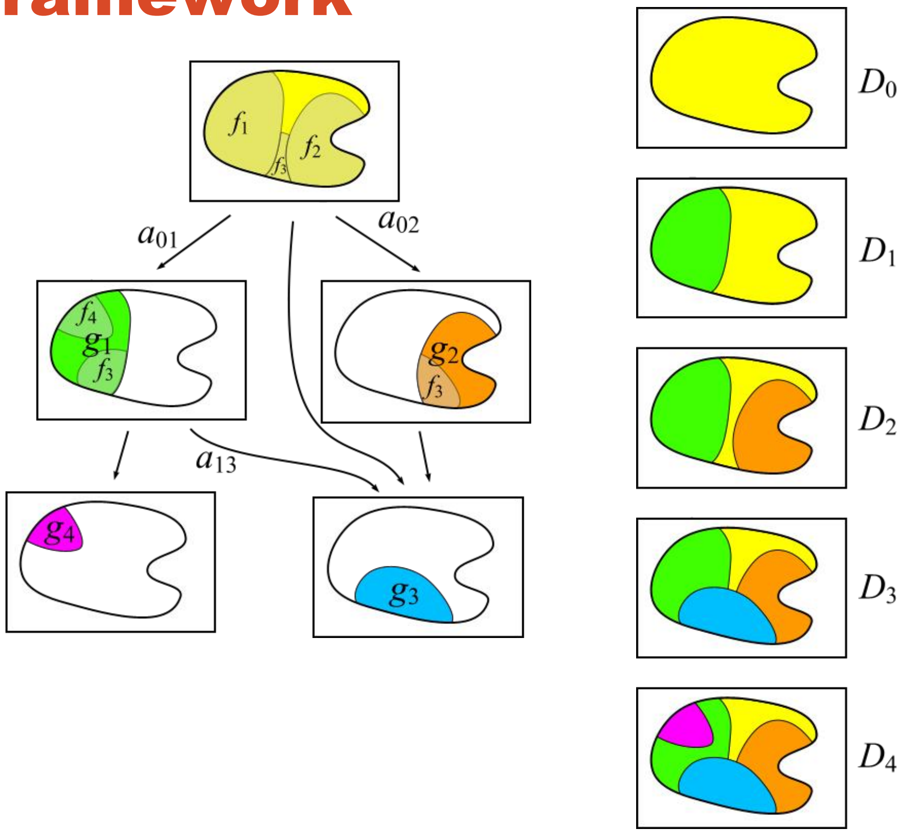




Chunked Multi Triangulations

The Multi Triangulation Framework

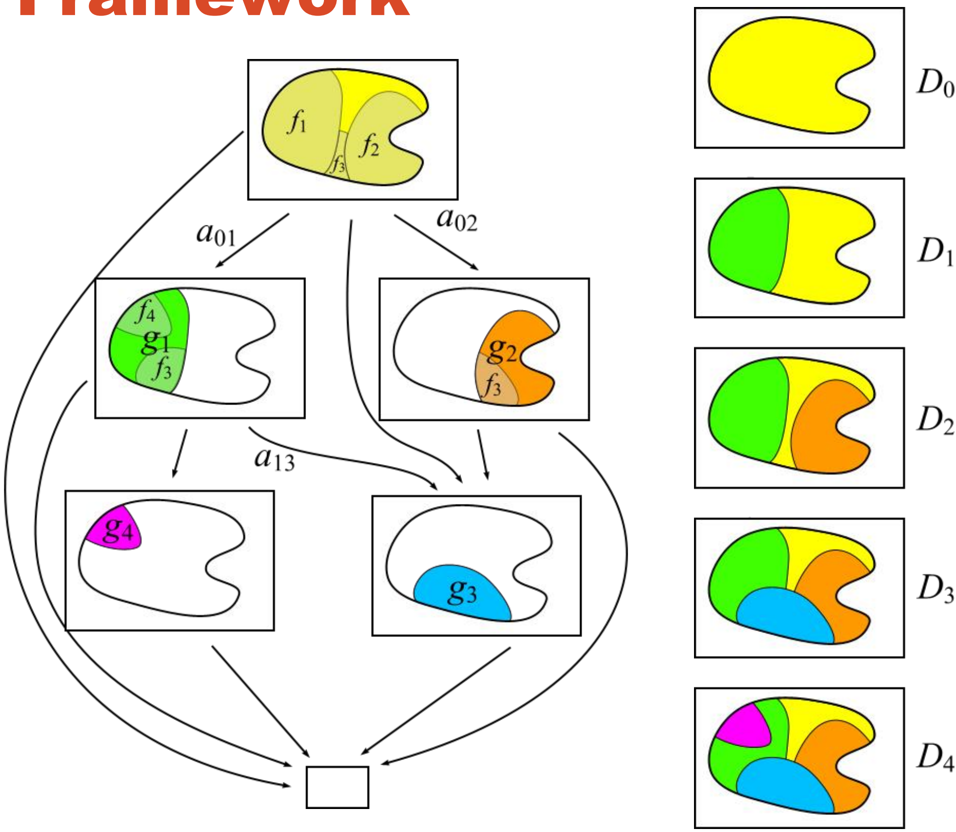
- Dependencies between modifications can be arranged in a DAG



Chunked Multi Triangulations

The Multi Triangulation Framework

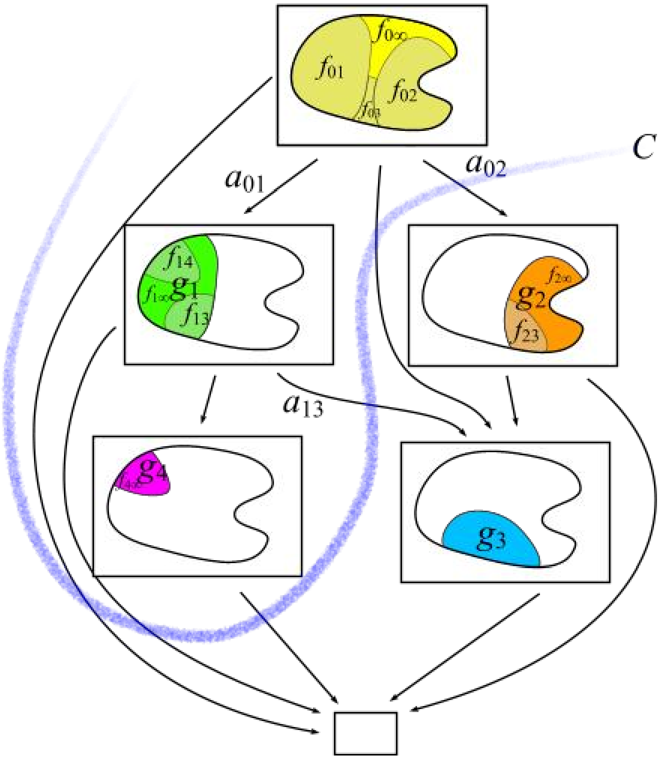
- Dependencies between modifications can be arranged in a DAG
  - Adding a sink to the DAG we can associate each fragment to an arc leaving a node

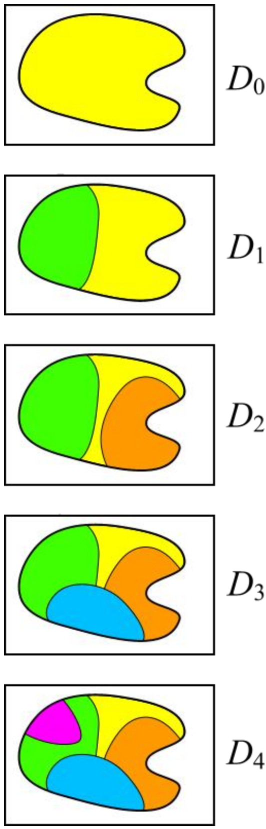


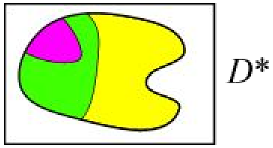
Chunked Multi Triangulations

MT Cuts

- A cut of the DAG defines a new representation
  - Just paste all the fragments above the cut





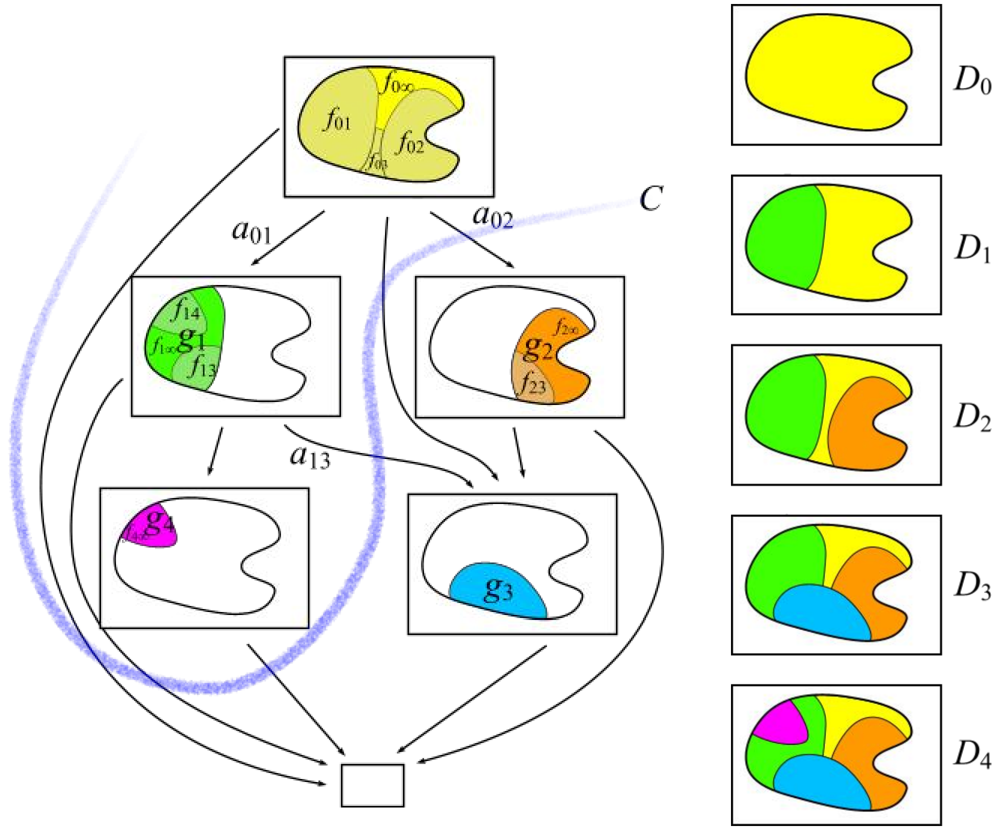


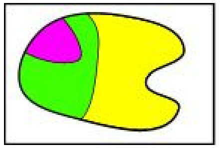
$$D^* = D_0 \oplus g_1 \oplus g_4$$

Chunked Multi Triangulations

MT Cuts

- A cut of the DAG defines a new representation
  - Collect all the fragment floors of cut arcs and you get a new conforming mesh





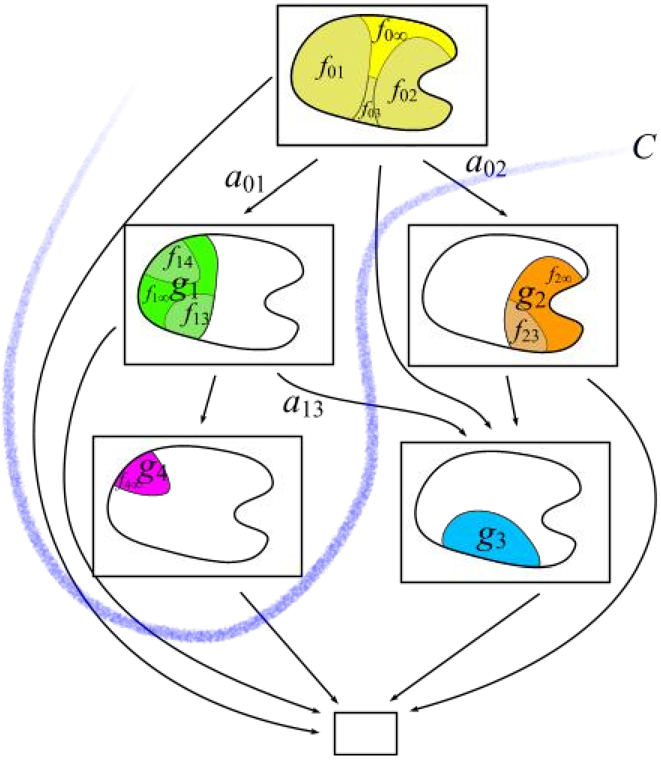
D*

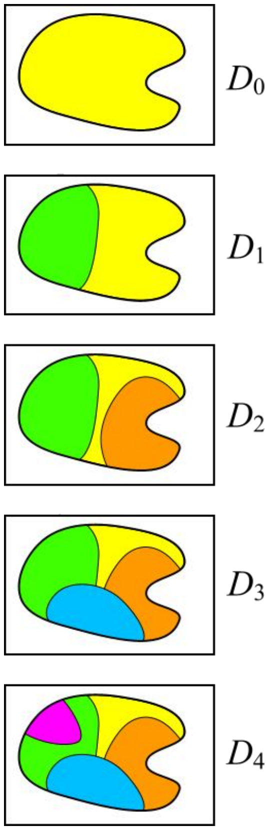
$$D^* = D_0 \oplus g_1 \oplus g_4 = f_{0\infty} \cup f_{02} \cup f_{03} \cup f_{13} \cup f_{1\infty} \cup f_{4\infty}$$

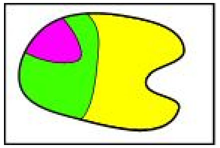
Chunked Multi Triangulations

GPU Friendly MT

- Chunked MT assume fragments are triangle patches with proper boundary constraints
  - DAG << original mesh (patches composed by thousands of tri)
  - Structure memory + traversal overhead amortized over thousands of triangles
  - Per-patch optimizations





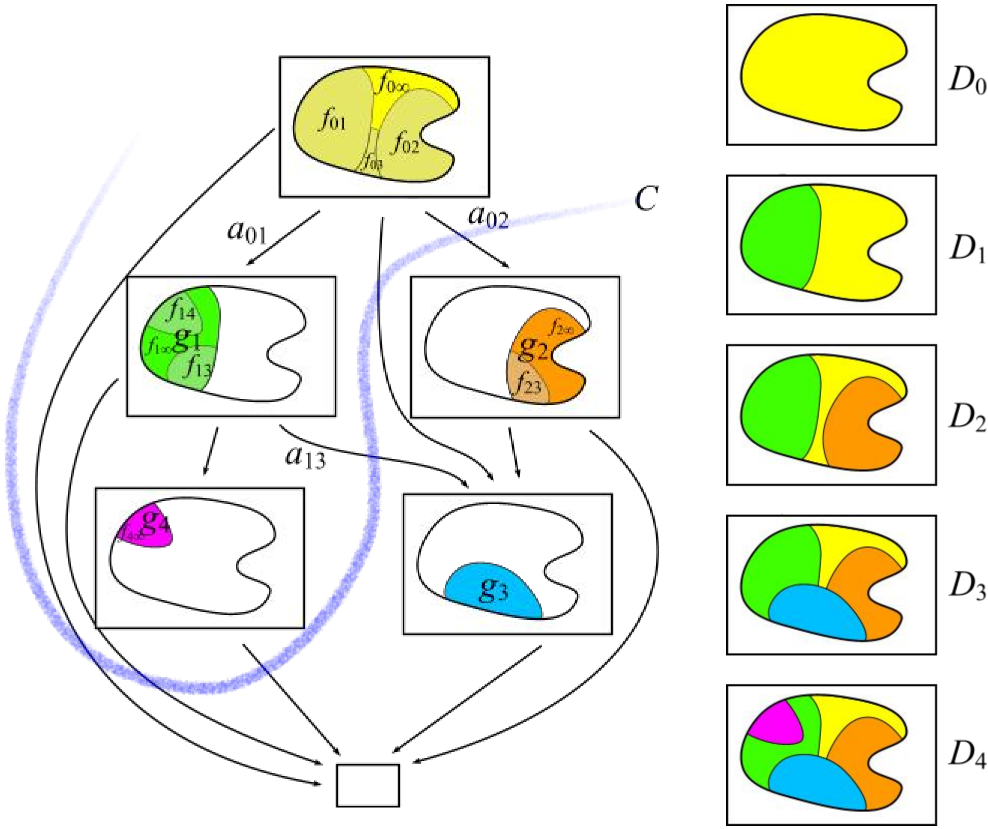


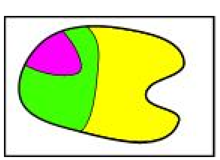
$$D^* = D_0 \oplus g_1 \oplus g_4 = f_{0\infty} \cup f_{02} \cup f_{03} \cup f_{13} \cup f_{1\infty} \cup f_{4\infty}$$

Chunked Multi Triangulations

GPU Friendly MT

- Chunked MT assume regions provide good hierarchical space-partitioning
  - Compact
    - Close-to-spherical
  - Used for computing fast projected error upper bounds
  - Used for visibility queries

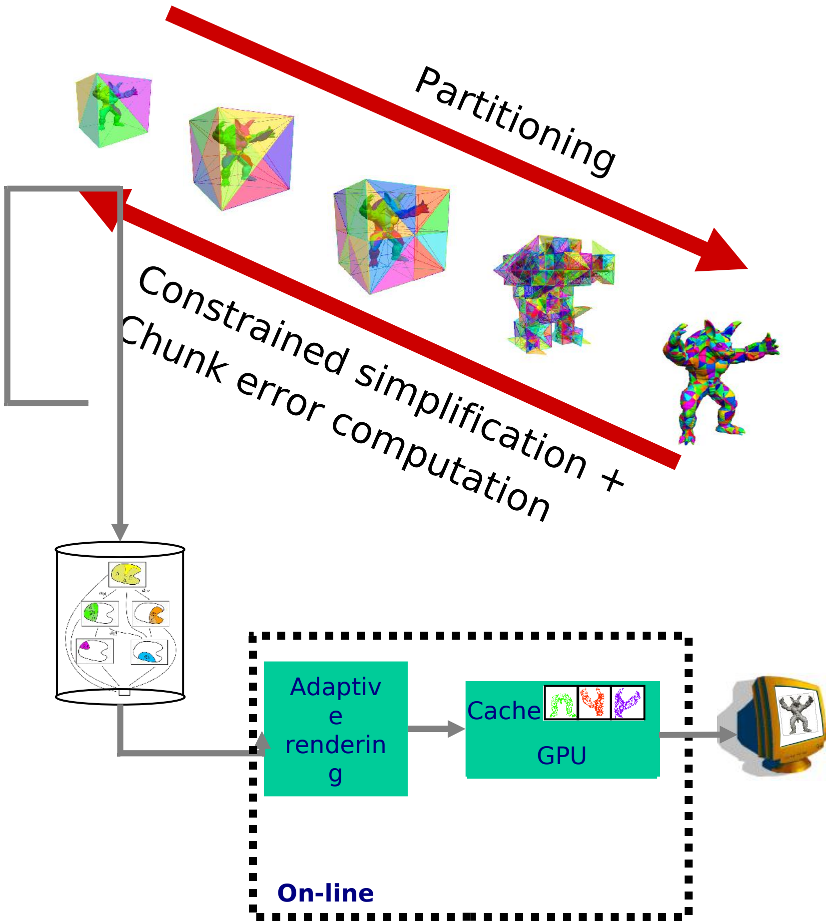



D^*

$$D^* = D_0 \oplus g_1 \oplus g_4 = f_{0\infty} \cup f_{02} \cup f_{03} \cup f_{13} \cup f_{1\infty} \cup f_{4\infty}$$

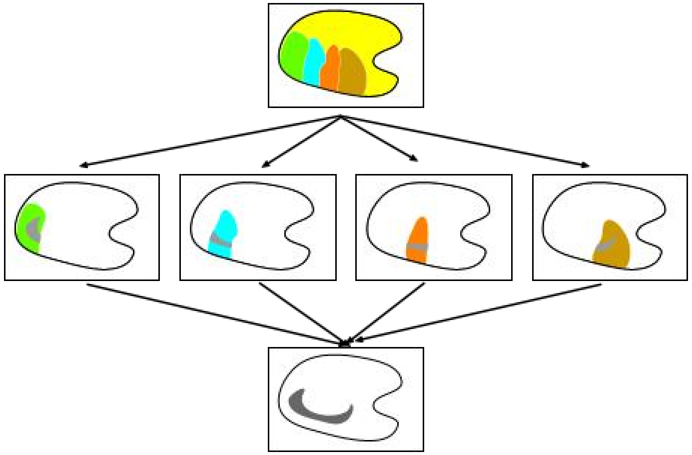
Chunked Multi Triangulations
GPU Friendly MT

- Construction
  - Start with hires triangle soup
  - Partition model using a **hierarchical space partitioning** scheme
  - Construct non-leaf cells by bottom-up **recombination and simplification** of lower level cells
  - Assign **model space errors** to cells
- Rendering
  - Refine conformal hierarchy, render selected precomputed cells
  - Project errors to screen
  - Dual queue

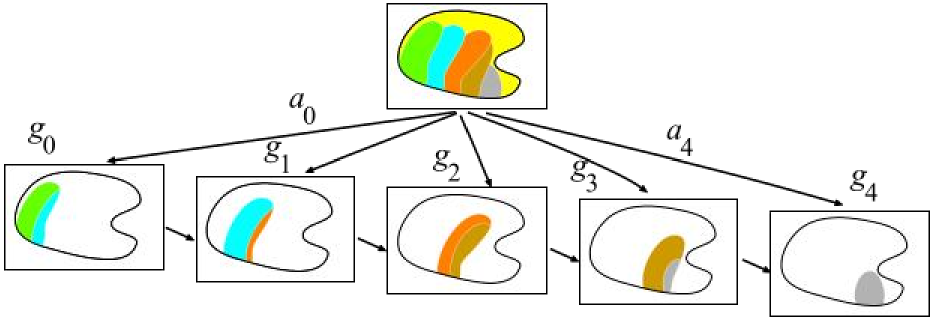


Chunked Multi Triangulations

DAG problems



- Not all MTs are good MTs!
  - The **topology of dependencies** may lower the adaptivity of the multiresolution structure
    - Cascading dependencies are BAD!!!
  - The **geometry of DAG regions** may cause problems in view-dependent rendering
    - Compact (close-to-spherical) regions for good constant error bounds
    - Long+thin regions are BAD!

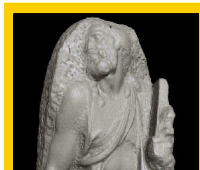


- Proposed solutions:
  - SIGGRAPH 2004: Efficient constrained technique (TetraPuzzles)
  - IEEE Viz 2005: General construction technique (V-Partition)

Our contributions

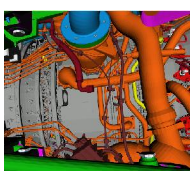
GPU-friendly output-sensitive techniques

***-BDAM – Local and Global Terrain Models**
Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Ponchio/Scopigno (CNR)
EG 2003, IEEE Viz 2003, EG 2005



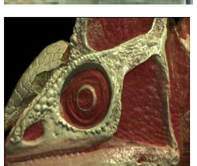
Adaptive Tetrapuzzles – Dense meshes
Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Ponchio/Scopigno (CNR)
SIGGRAPH 2004

Layered Point Clouds – Dense clouds
Gobbetti/Marton (CRS4)
SPBG 2004 / Computers & Graphics 2004



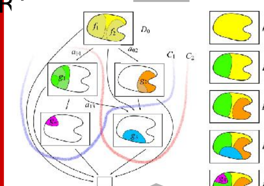
Far Voxels – General
Gobbetti/Marton (CRS4)
SIGGRAPH 2005

Blockmaps – Hybrid volumetric city model
Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Di Benedetto/Scopigno (CNR)
EG 2007



MOVR – Volumetric models
Gobbetti/Marton/Iglesias Guitian (CRS4)
CGI 2008

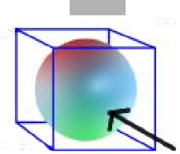
Specialize



Chunked Multi-Triangulations
Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Ponchio/Scopigno (CNR) *IEEE Viz 2005*

Generalize

Specialize



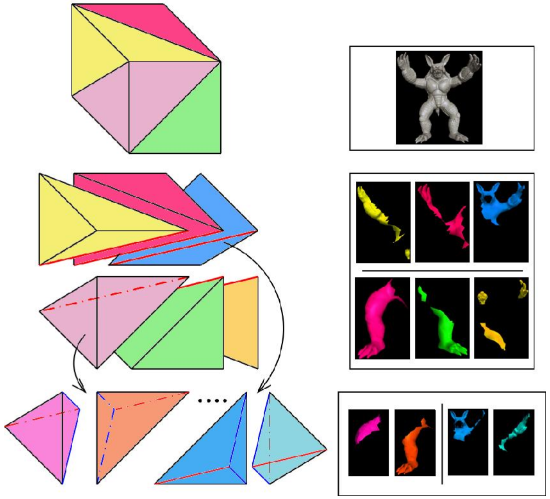
View-dep. Volumetric Model
In progress

Generalize

Adaptive TetraPuzzles

Multiresolution Model for Dense 3D meshes

- **Adaptive TetraPuzzles**: High performance visualization of dense 3D meshes
  - Two-level multiresolution model based on volumetric decomposition
  - Implicit MT based on tetrahedra hierarchy



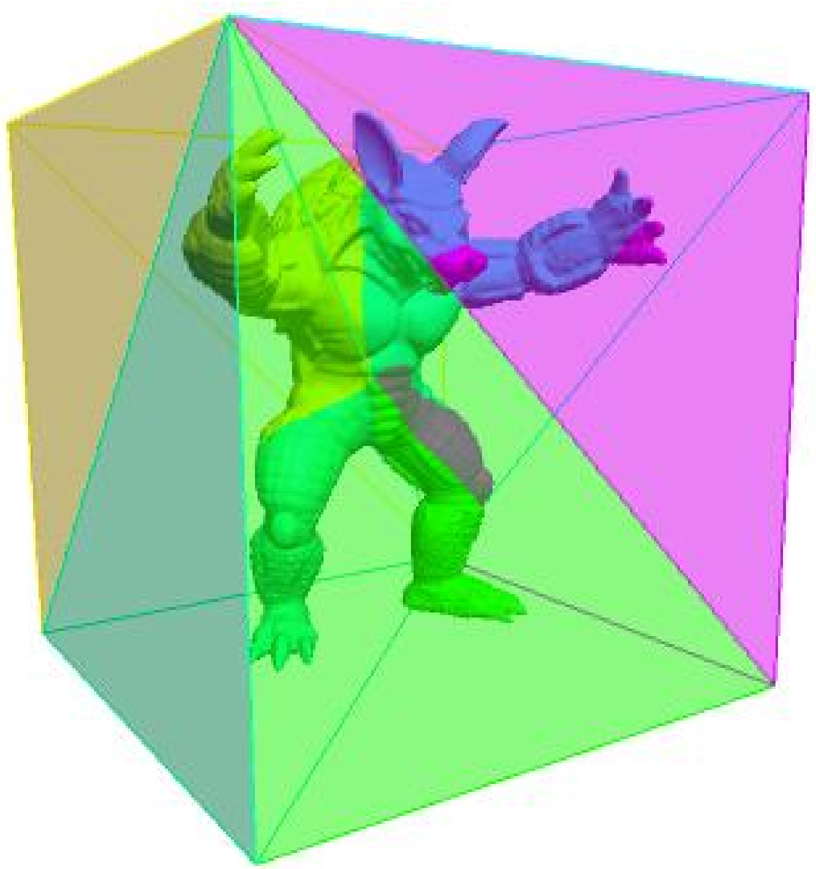
Cignoni, Ganovelli, Gobbetti, Marton, Ponchio, and Scopigno.
Adaptive TetraPuzzles - Efficient Out-of-core Construction and Visualization of Gigantic Polygonal Models.
ACM Transactions on Graphics, 23(3), August 2004
(Proc. SIGGRAPH 2004).

Adaptive TetraPuzzles
Overview

- **Construction**
  - Start with hires triangle soup

Adaptive TetraPuzzles

Overview

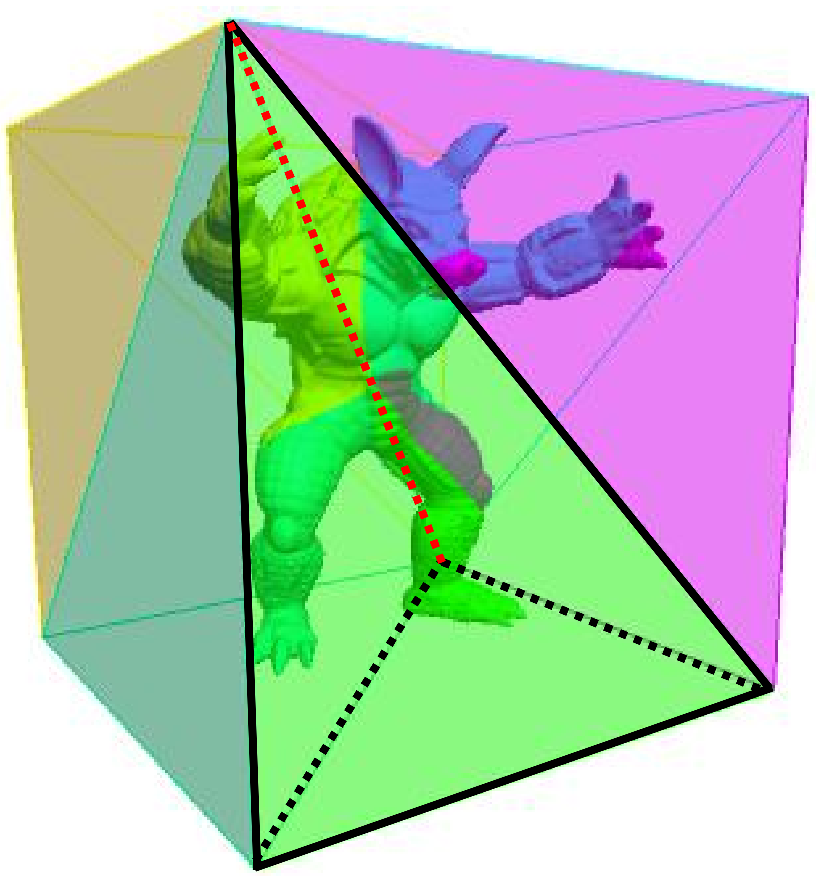


Target = *k* triangles/chunk

- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra

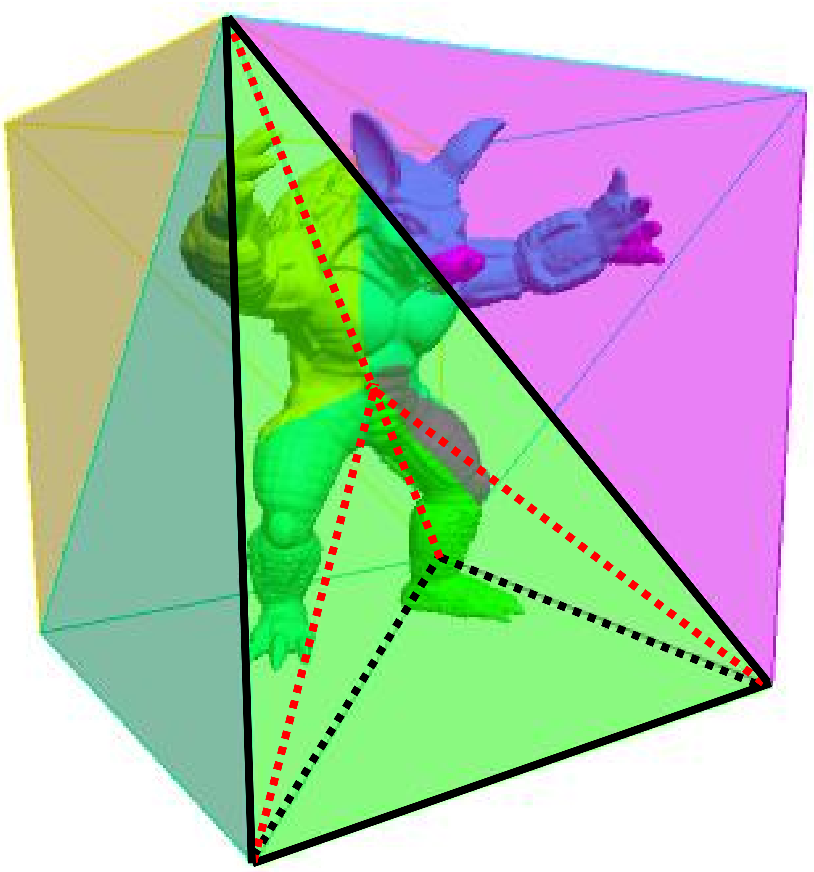
Adaptive TetraPuzzles

Overview



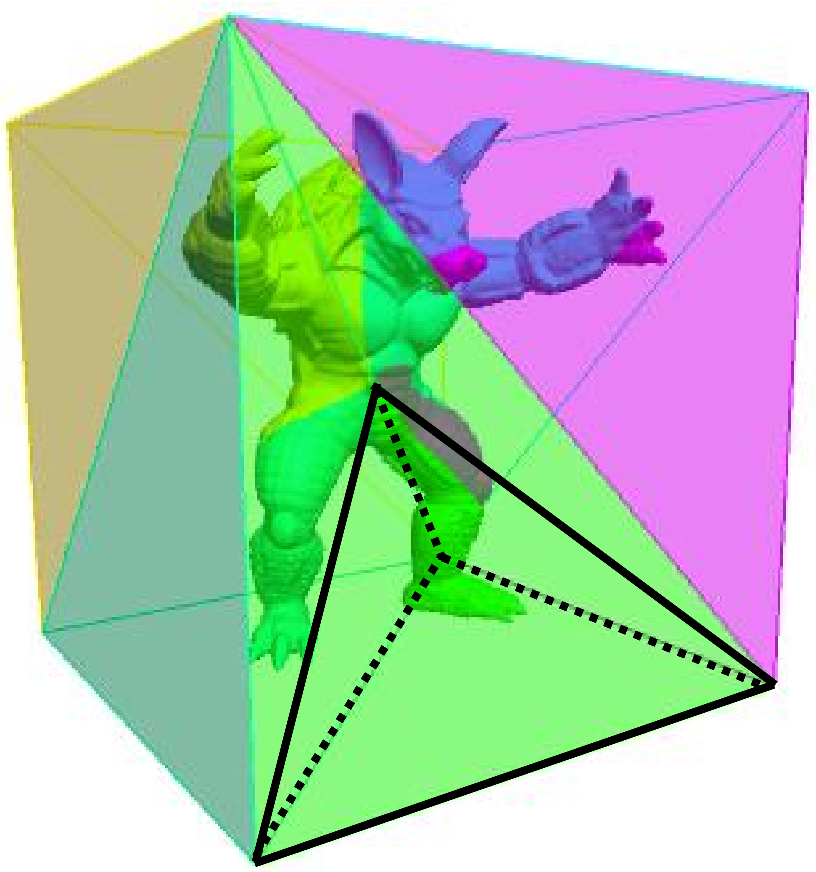
- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra

Adaptive TetraPuzzles
Overview



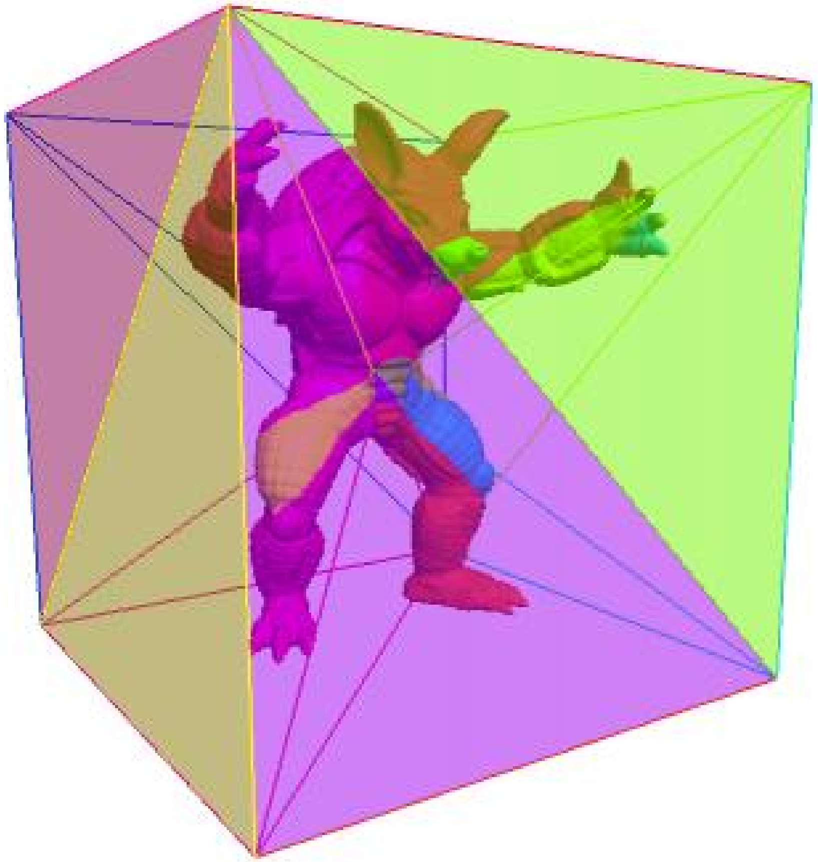
- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra

Adaptive TetraPuzzles
Overview



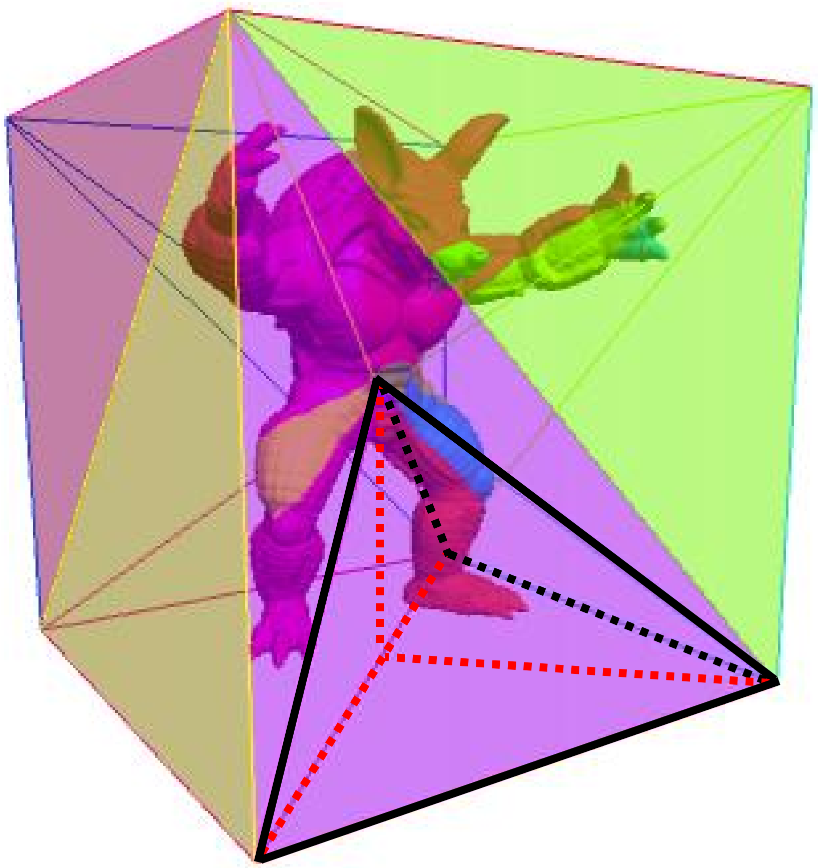
- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra

Adaptive TetraPuzzles
Overview



- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra

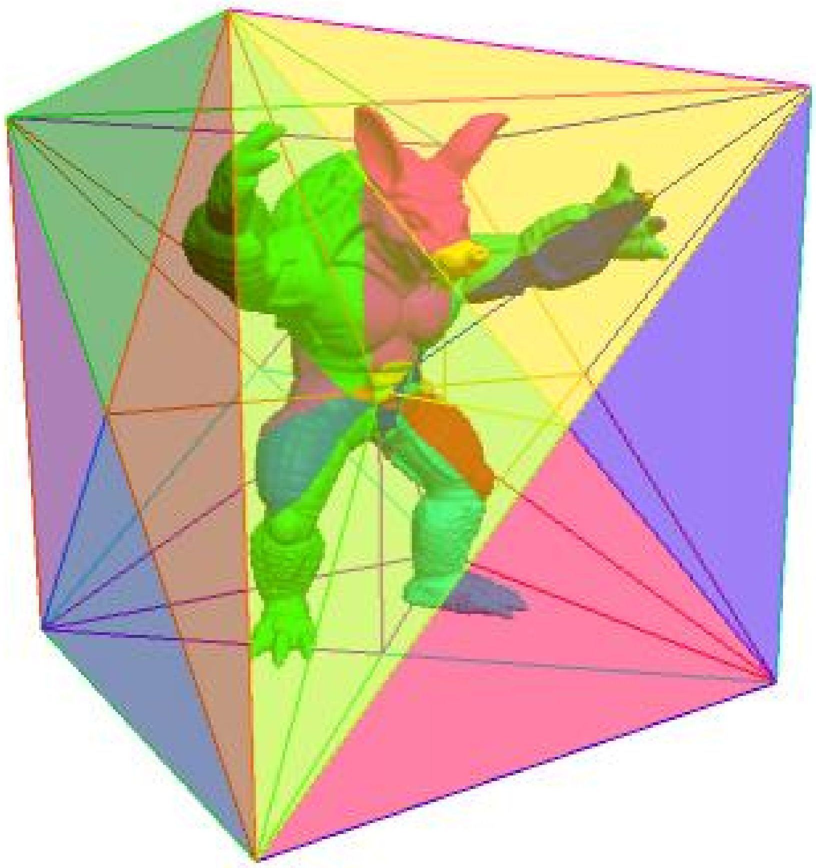
Adaptive TetraPuzzles
Overview



- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra

Adaptive TetraPuzzles

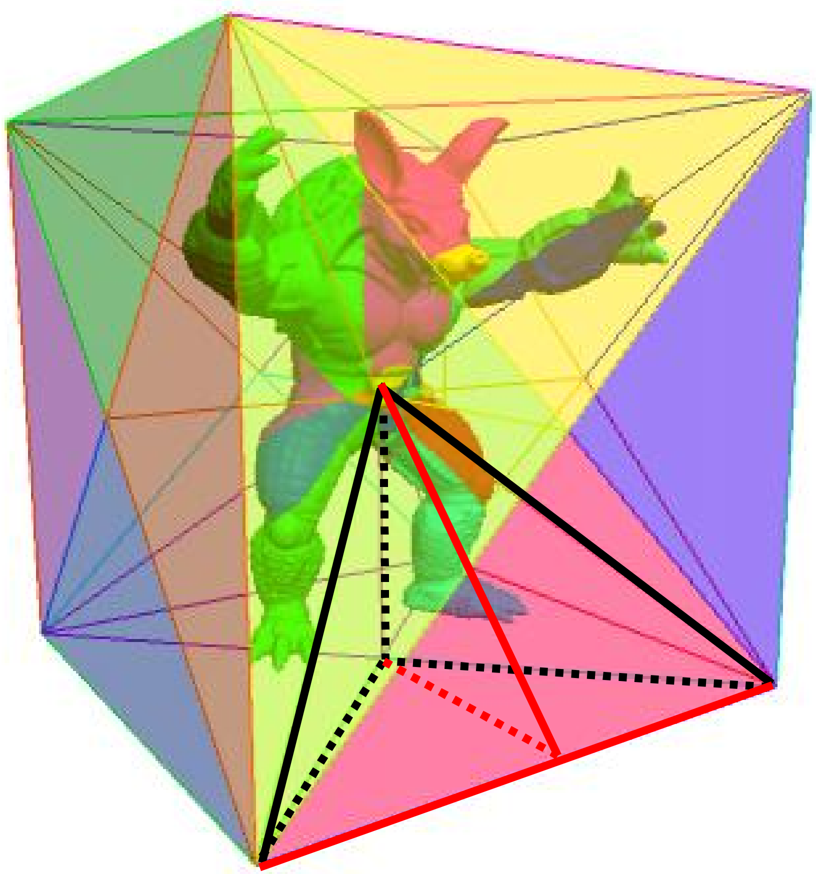
Overview



- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra

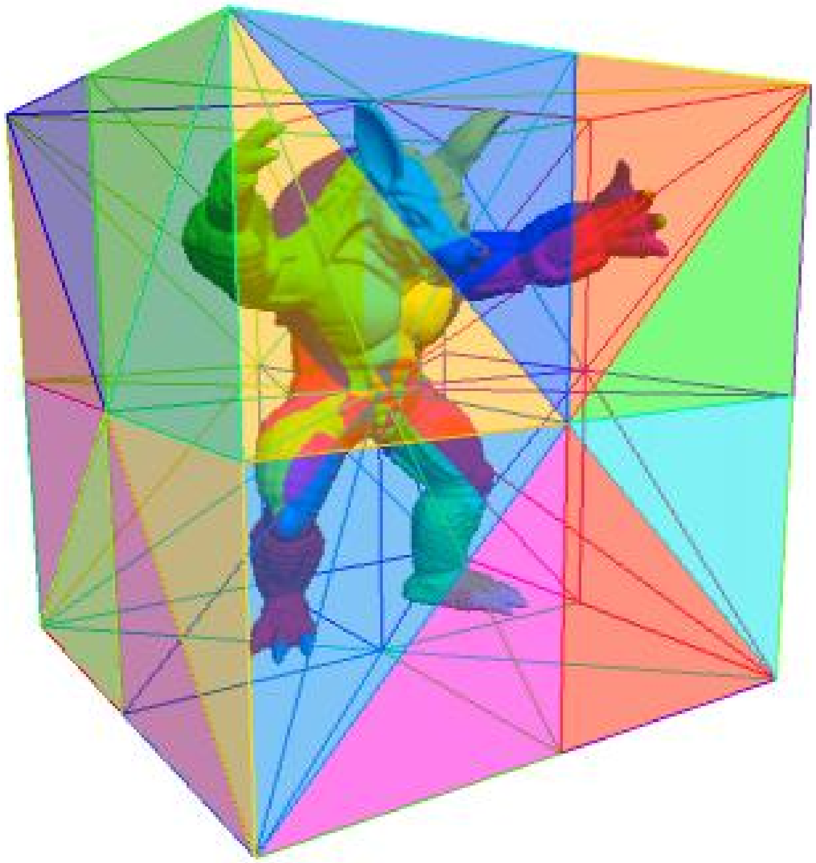
Adaptive TetraPuzzles

Overview



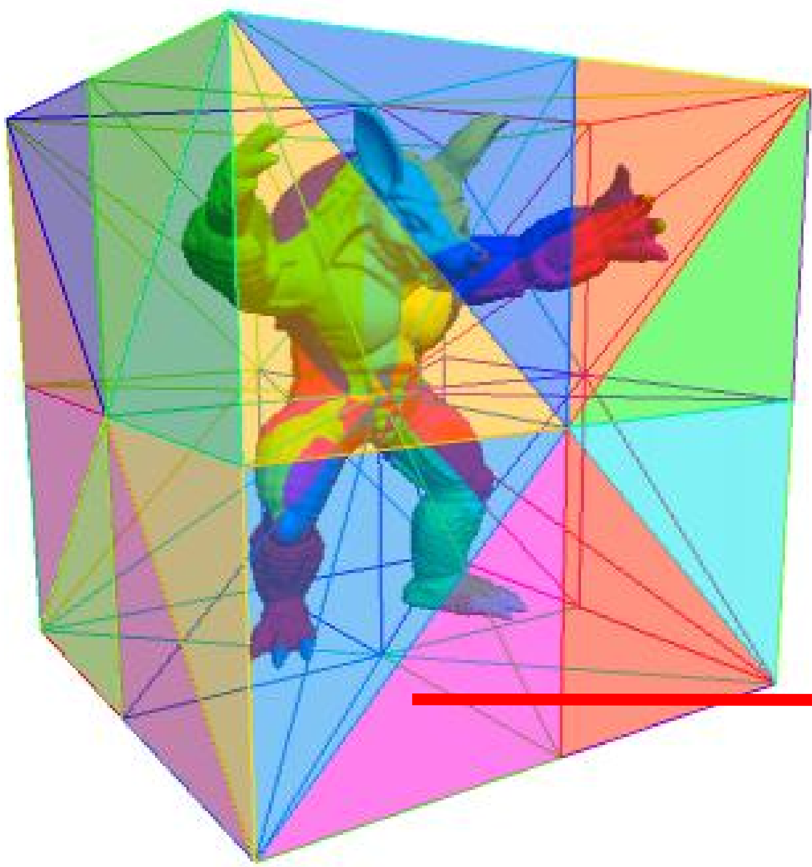
- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra

Adaptive TetraPuzzles
Overview

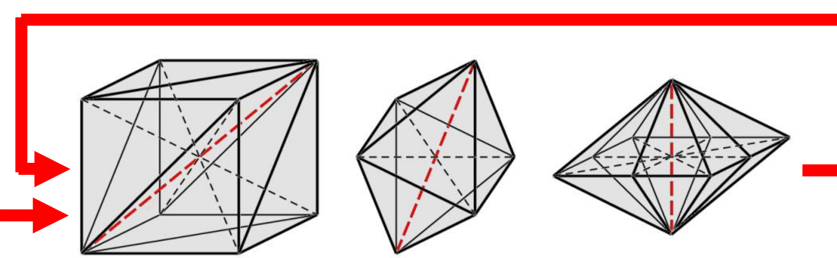


- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra

Adaptive TetraPuzzles
Overview

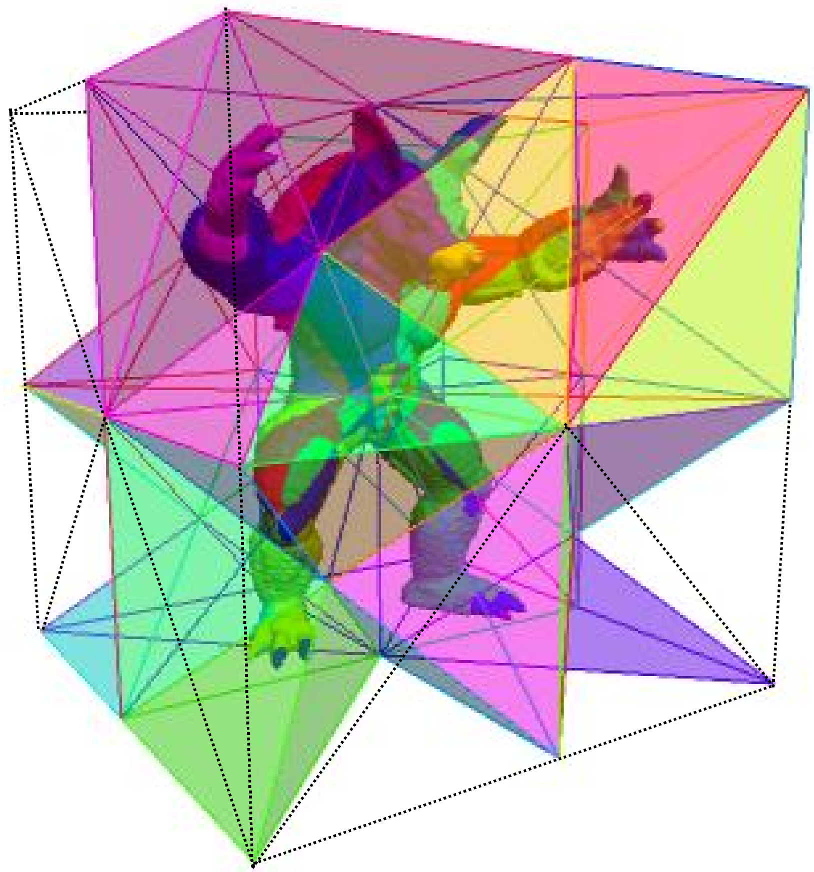


- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra



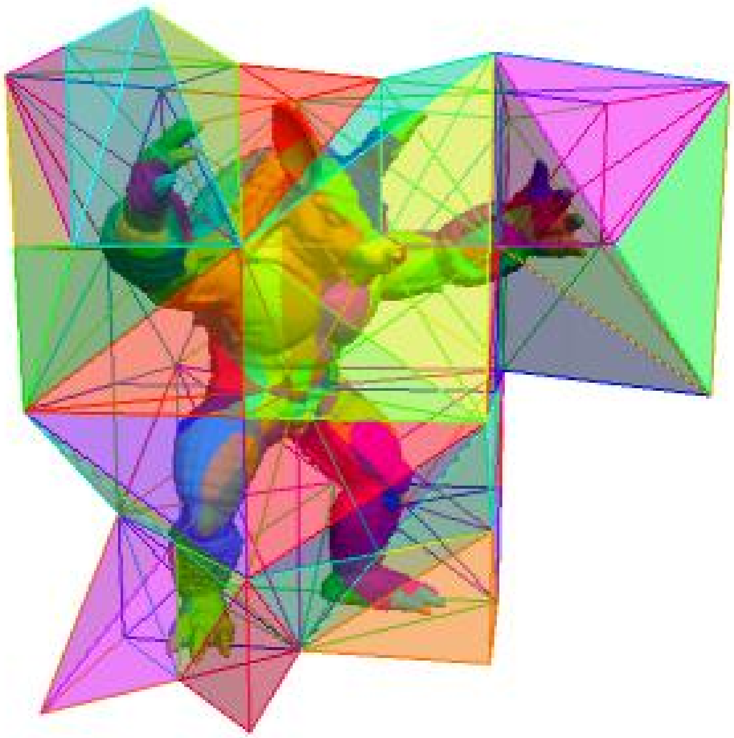
(6 tetra / diamond) (4 tetra / diamond) (8 tetra / diamond)

Adaptive TetraPuzzles
Overview



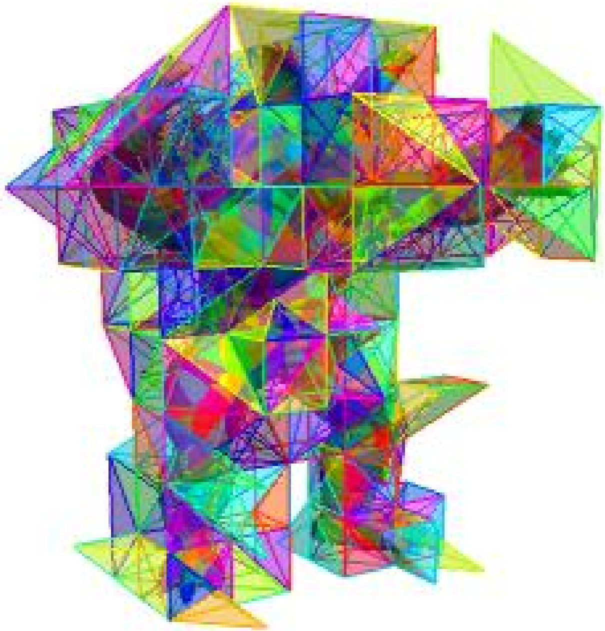
- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra

Adaptive TetraPuzzles
Overview



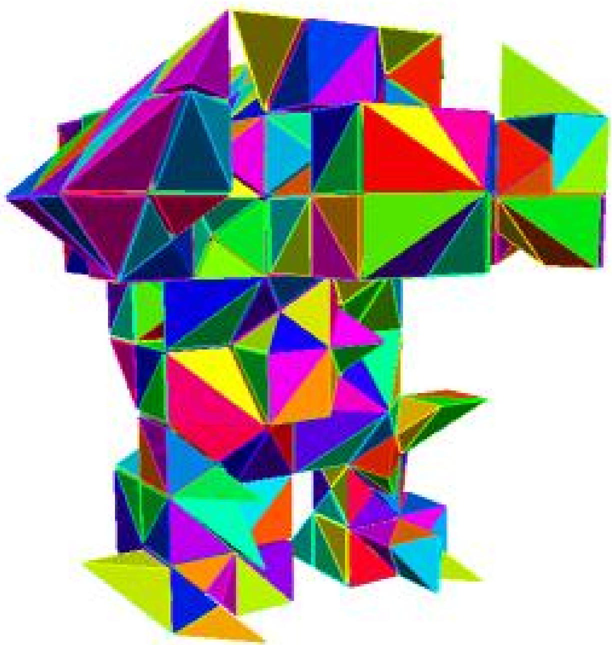
- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra

Adaptive TetraPuzzles
Overview



- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra

Adaptive TetraPuzzles
Overview



- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra

Adaptive TetraPuzzles

Overview

k triangles/chunk

- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra

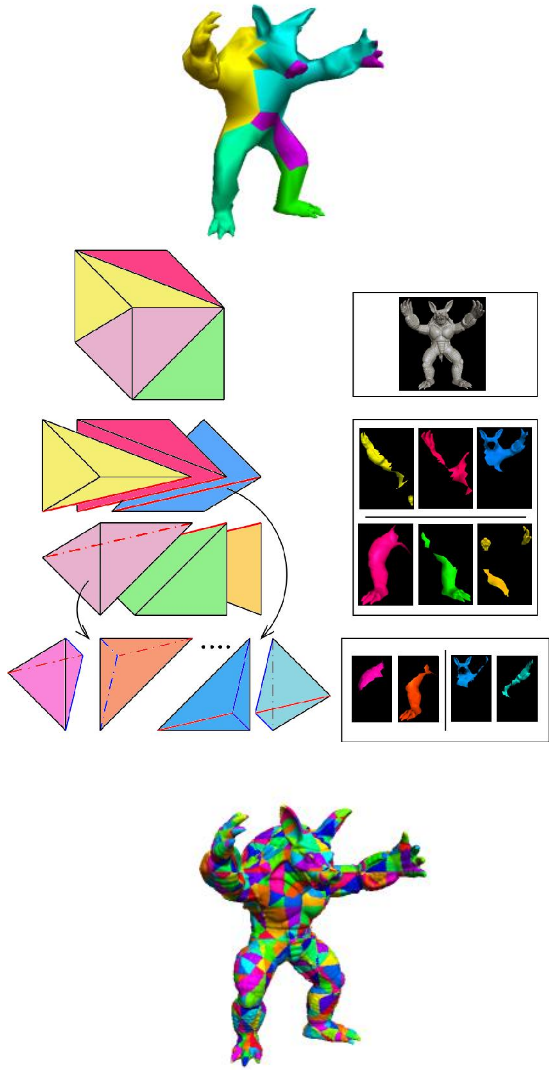
Adaptive TetraPuzzles
Overview

k triangles/chunk

- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra

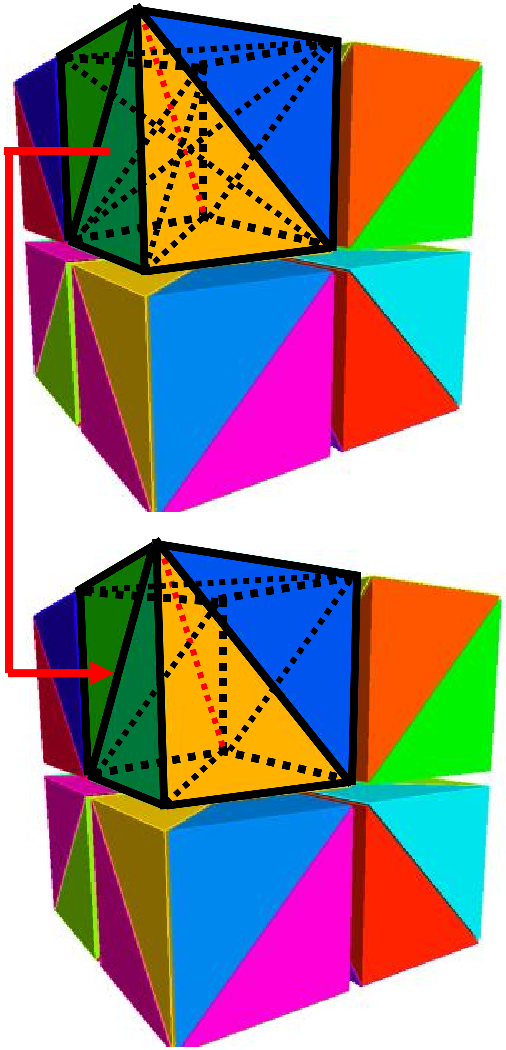
Adaptive TetraPuzzles

Overview



- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra
  - Construct non-leaf cells by bottom-up recombination and simplification of lower level cells

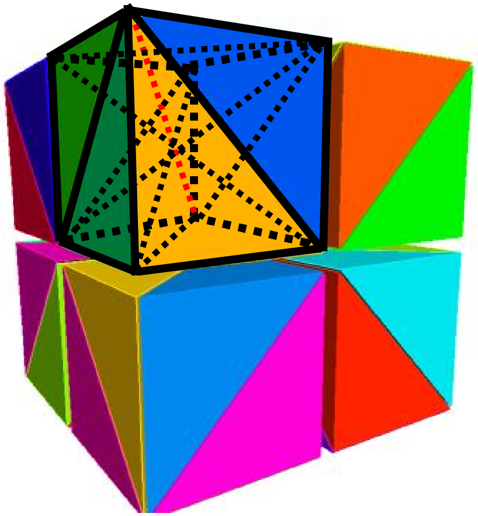
Adaptive TetraPuzzles
Overview

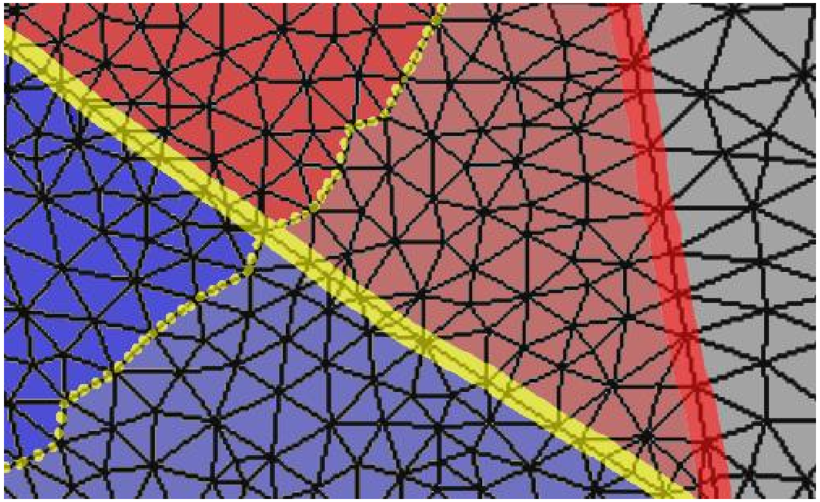


- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra
  - Construct non-leaf cells by bottom-up recombination and simplification of lower level cells

Adaptive TetraPuzzles

Overview



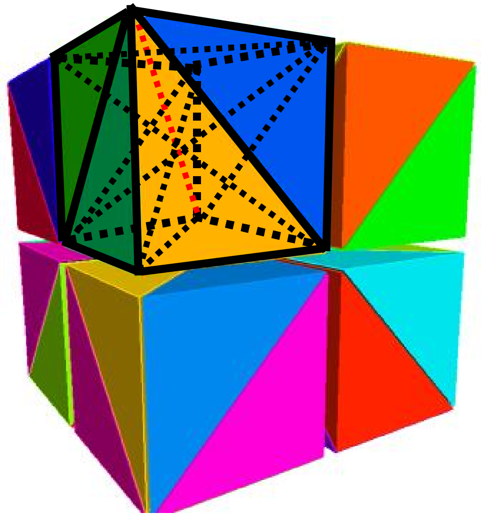


- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra
  - Construct non-leaf cells by bottom-up recombination and simplification of lower level cells

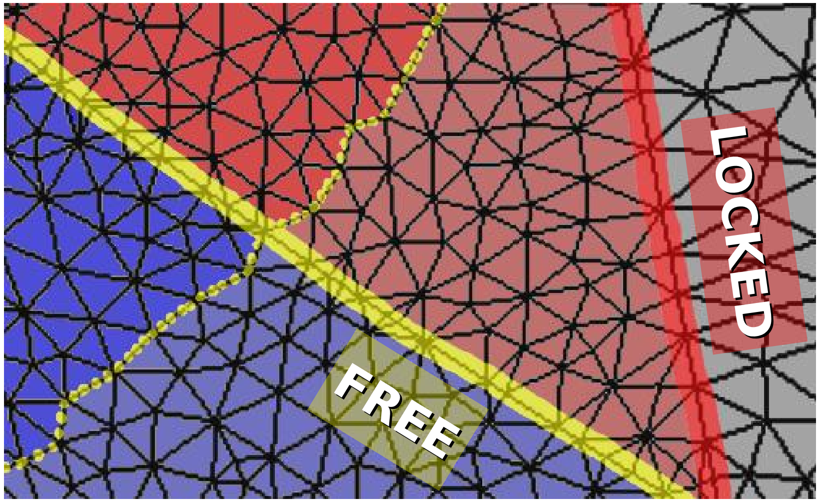
Diamond external boundary
Diamond internal boundary
Child tetrahedra boundary

Adaptive TetraPuzzles

Overview



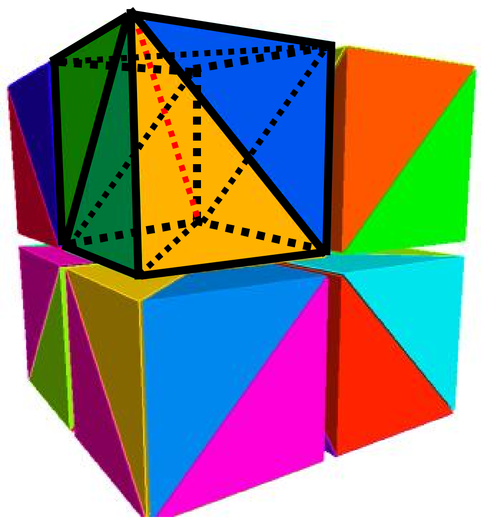
- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra
  - Construct non-leaf cells by bottom-up recombination and simplification of lower level cells

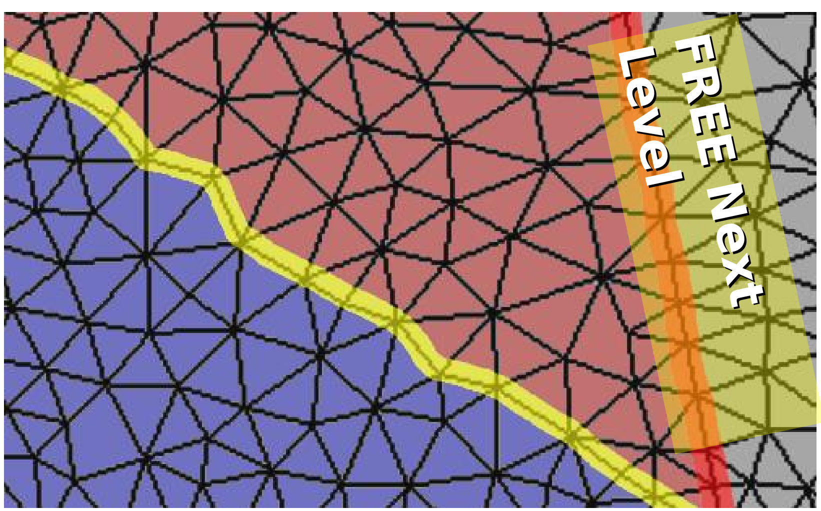


Diamond external boundary
Diamond internal boundary
Child tetrahedra boundary

Adaptive TetraPuzzles

Overview





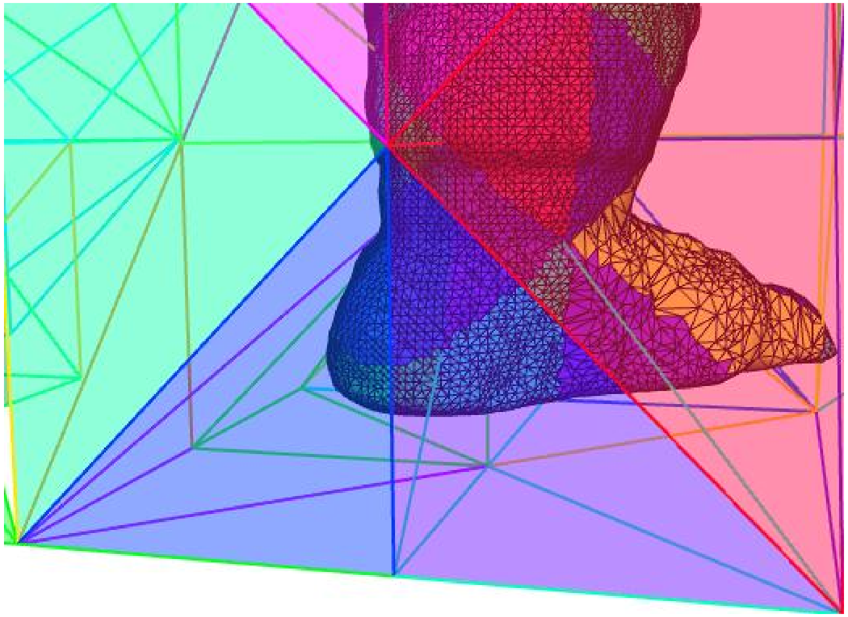
- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra
  - Construct non-leaf cells by bottom-up recombination and simplification of lower level cells

Diamond external boundary

Diamond internal boundary

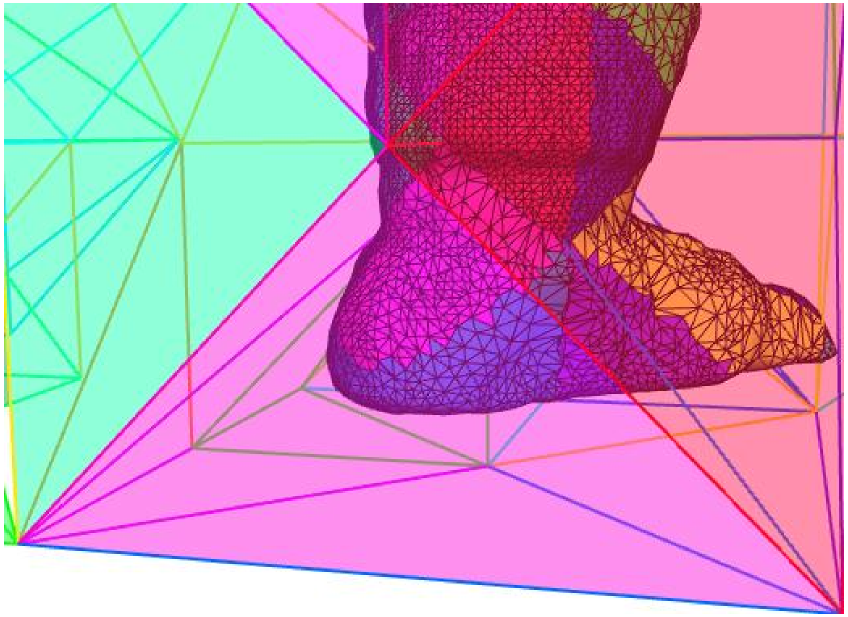
Adaptive TetraPuzzles

Overview



- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra
  - Construct non-leaf cells by bottom-up recombination and simplification of lower level cells

Adaptive TetraPuzzles
Overview

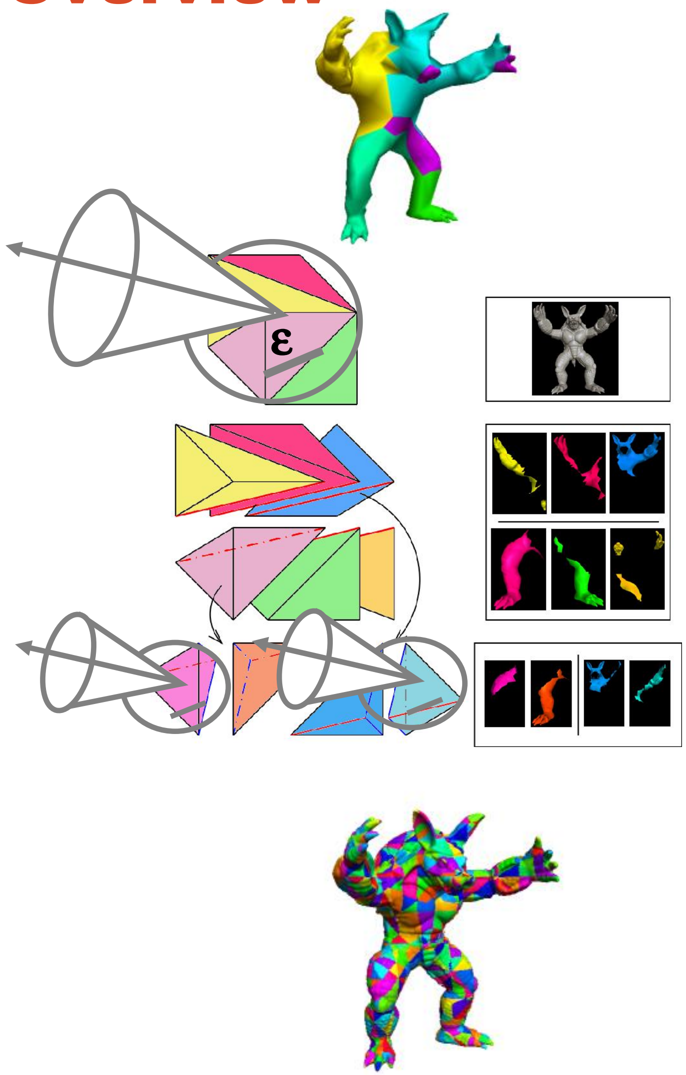


- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra
  - Construct non-leaf cells by bottom-up recombination and simplification of lower level cells

NO CRACKS / NO GLOBALLY LOCKED BOUNDARY!

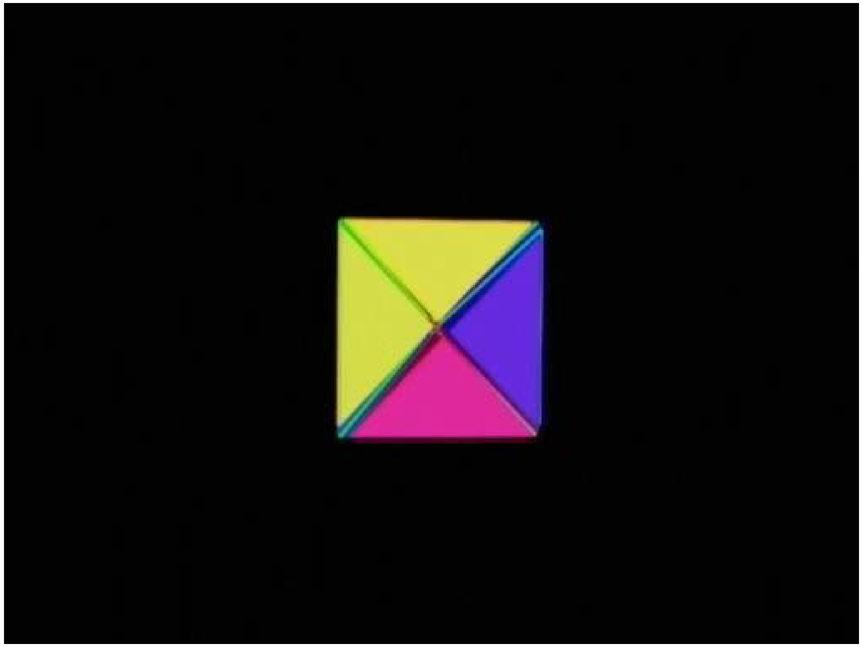
Adaptive TetraPuzzles

Overview



- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra
  - Construct non-leaf cells by bottom-up recombination and simplification of lower level cells
- **Rendering**
  - Refine conformal hierarchy, render selected precomputed cells

Adaptive TetraPuzzles
Overview


View dependent mesh refinement

- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra
  - Construct non-leaf cells by bottom-up recombination and simplification of lower level cells
- **Rendering**
  - Refine conformal hierarchy, render selected precomputed cells

Adaptive TetraPuzzles
Overview

Independent diamond processing

For each mesh chunk:
Simplify + stripify +
compress + eval bounds/error

Out-of-core + parallel

- **Construction**
  - Start with hires triangle soup
  - Partition model using a conformal hierarchy of tetrahedra
  - Construct non-leaf cells by bottom-up recombination and simplification of lower level cells

Out-of-core cull+refine traversal / GPU cached optimized meshes

- **Rendering**
  - Refine conformal hierarchy, render selected precomputed cells

Adaptive TetraPuzzles
Results

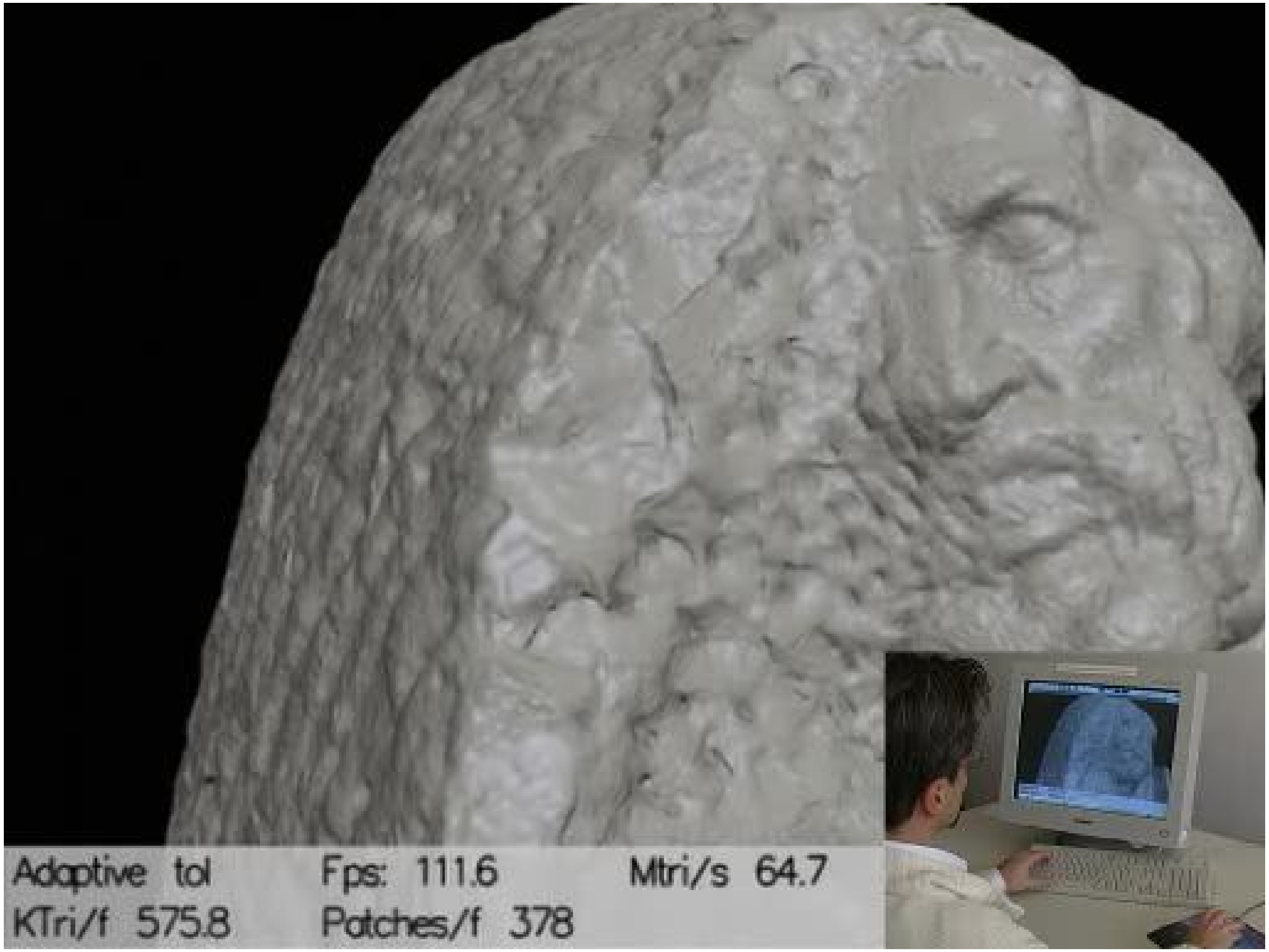
Michelangelo's St. Matthew

Source: Digital Michelangelo Project

Data: 374M triangles

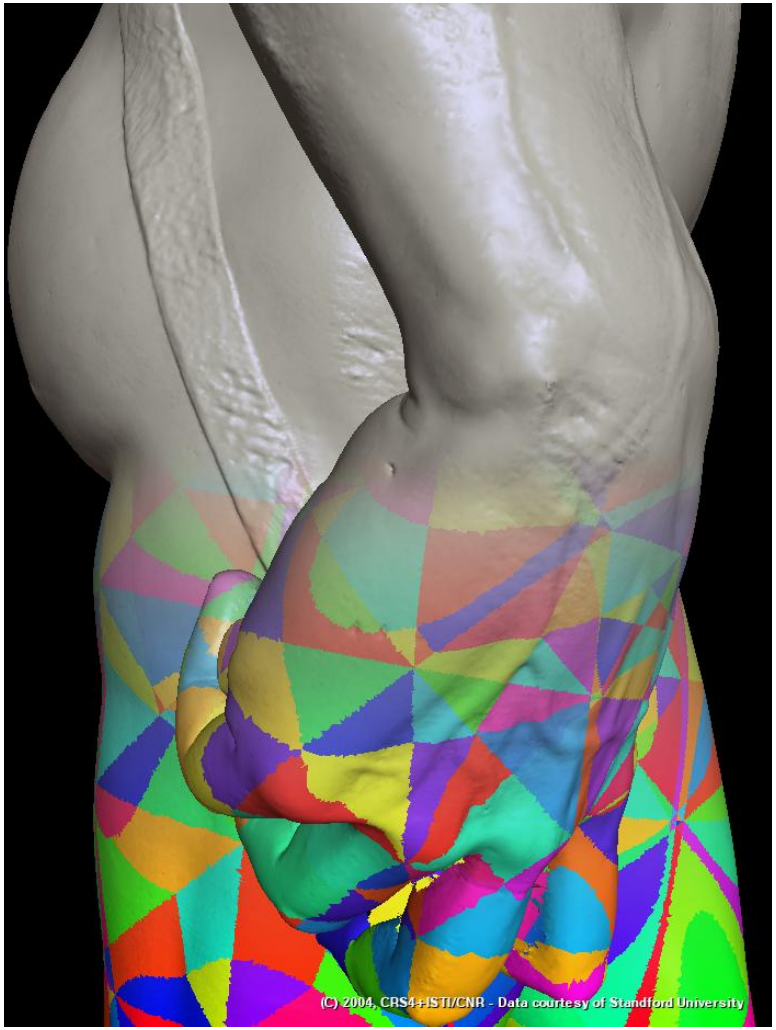
Intel Xeon 2.4GHz 1GB

GeForce FX 5800U AGP8X



Adaptive TetraPuzzles
Conclusions

- Yet another multiresolution algorithm for rendering large static meshes
  - First GPU bound method for very large meshes
  - State of the art performance
    - GPU bound
    - >4Mtri/frame at >30 fps on modern GPUs
  - Tuned for large dense models with "well behaved" surface
  - Special case of general MT framework



Our contributions

GPU-friendly output-sensitive techniques

***-BDAM – Local and Global Terrain Models**

Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Ponchio/Scopigno (CNR

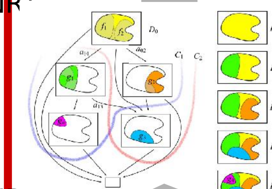
EG 2003, IEEE Viz 2003, EG 2005

Adaptive Tetrapuzzles – Dense meshes

Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Ponchio/Scopigno (CNR

SIGGRAPH 2004

Specialize



Chunked Multi-Triangulations

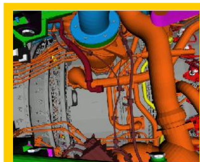
Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Ponchio/Scopigno (CNR) *IEEE Viz 2005*

Generalize

Layered Point Clouds – Dense clouds

Gobbetti/Marton (CRS4)

SPBG 2004 / Computers & Graphics 2004



Far Voxels – General

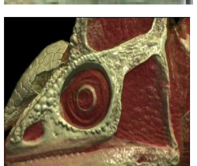
Gobbetti/Marton (CRS4)

SIGGRAPH 2005

Blockmaps – Hybrid volumetric city model

Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Di Benedetto/Scopigno (CNR)

EG 2007

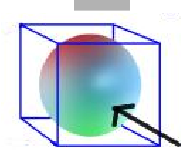


MOVR – Volumetric models

Gobbetti/Marton/Iglesias Guitian (CRS4)

CGI 2008

Specialize



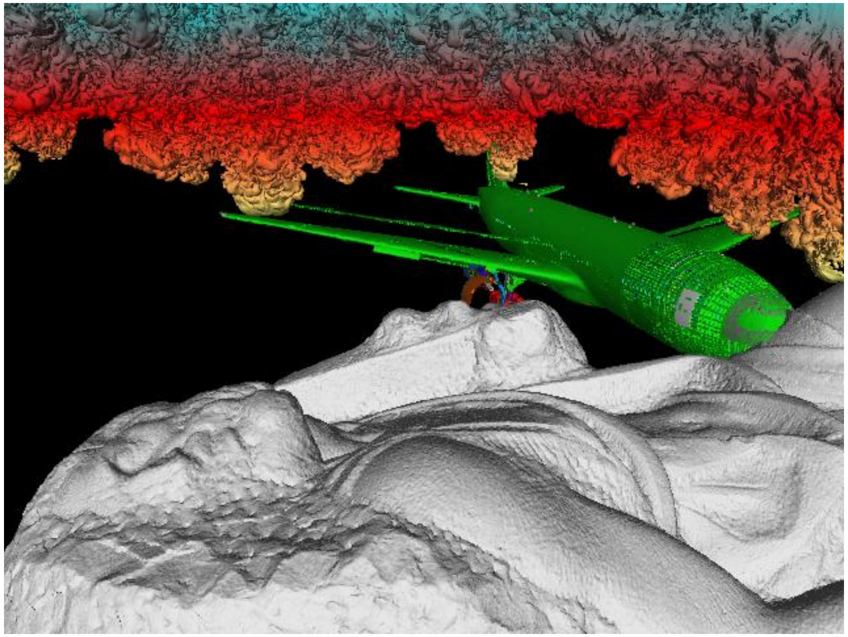
View-dep. Volumetric Model

In progress

Generalize

Our contributions
Far Voxels – General 3D models

- Far Voxels: High performance visualization of arbitrary 3D models
  - Mixed model
  - Seamless integration of occlusion culling with out-of-core data management and multiresolution rendering



Gobbetti and Marton.
Far Voxels – A nultiresolution Framework for Interactive Rendering of Huge Complex 3D Models on Commodity Graphics Platforms.
ACM Transactions on Graphics, 23(4), August 2005
(Proc. SIGGRAPH 2005).

Far Voxels

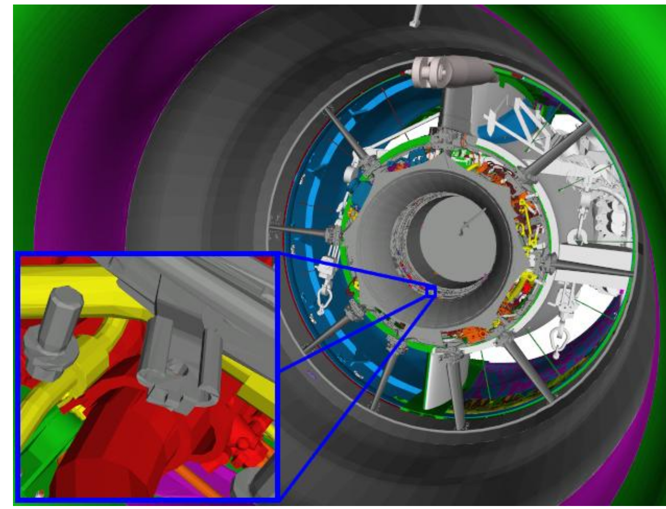
Real-time inspection of huge complex models on commodity graphics platforms

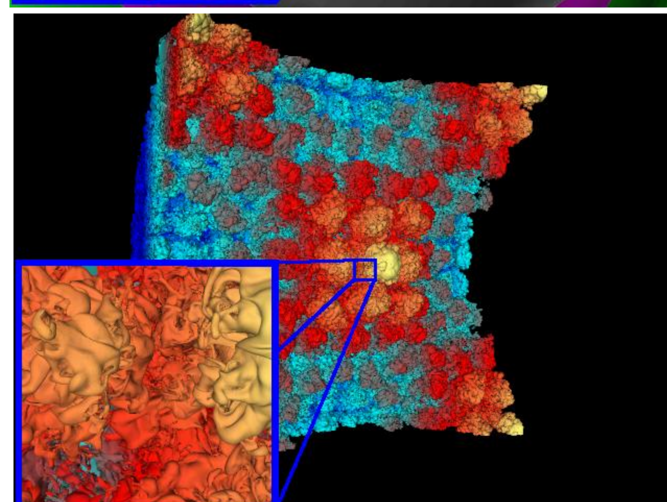
- Huge
  - $O(10^9)$ triangles/bytes
- Complex
  - Heterogeneous materials
  - High topological genus
  - Highly variable depth complexity
  - Fine geometric details
  - “Bad” tessellations

Xeon 2.4GHz / 1GB RAM / 70GB SCSI 320 Disk
NVIDIA 6800GT AGP 8X

Far Voxels
Handling Huge Complex 3D models

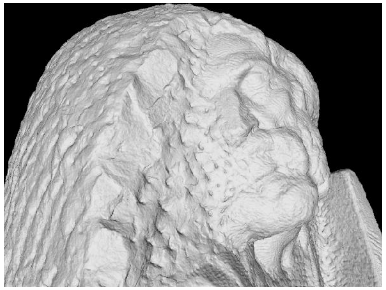
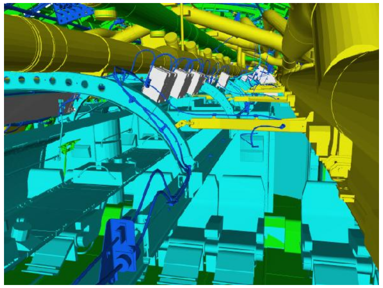
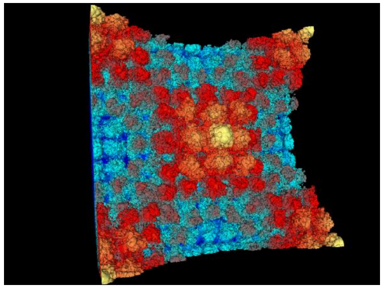
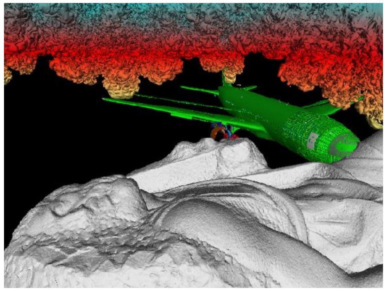
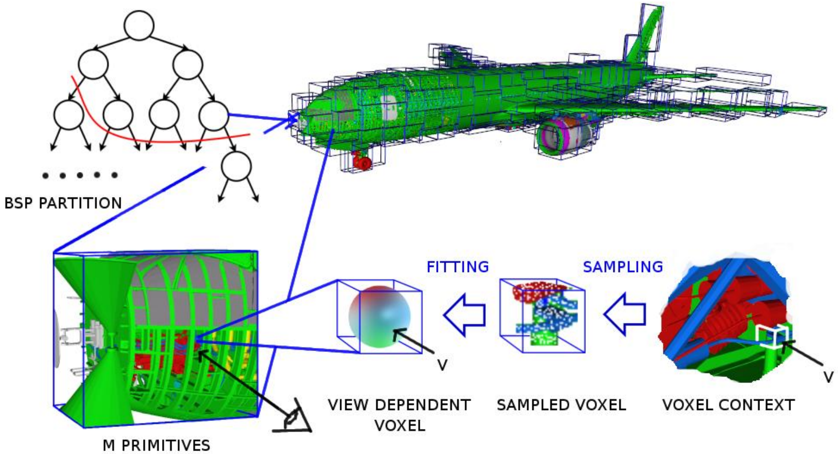
- Classic multiresolution models
  - Error measured on boundary surfaces
  - LOD construction based on local surface coarsening/simplification operations
  - Visibility culling decoupled from multiresolution
- Hard to apply to models with high detail and complex topology and high depth complexity!





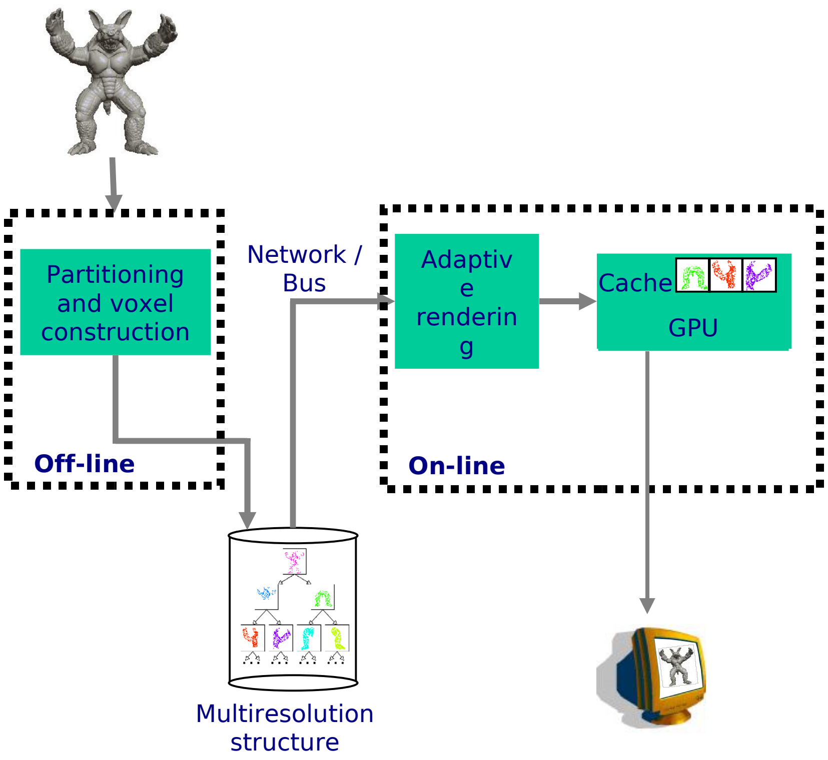
Far Voxels
Handling Huge Complex 3D models

- General purpose technique that targets many model kinds
- Underlying ideas
  - Multi-scale modeling of appearance rather than geometry
  - Volume-based rather than surface-based
  - Tight integration of visibility and LOD construction
  - GPU accelerated (programmabilty + batching)

Far Voxels
Overview

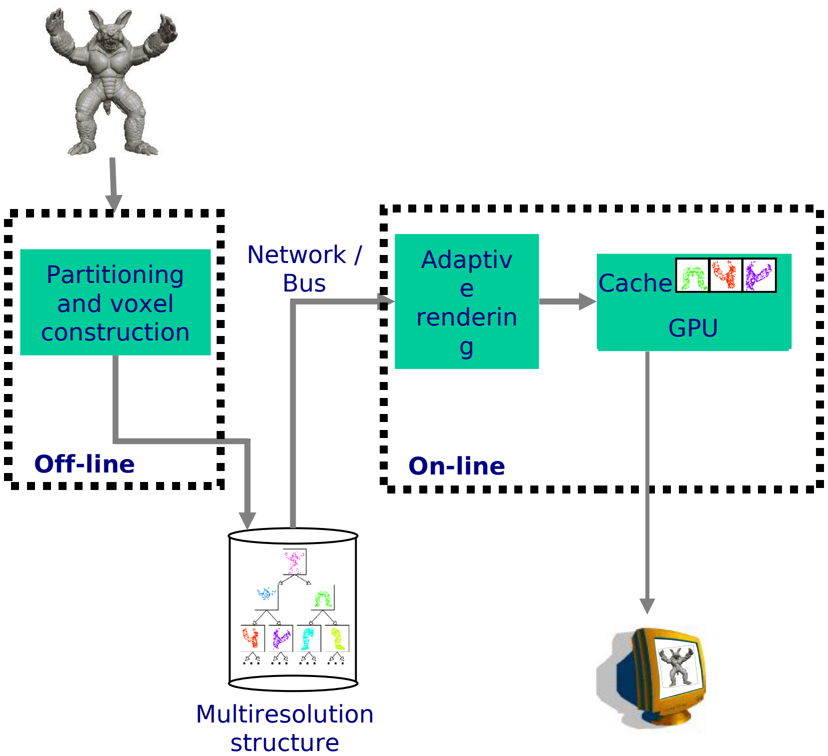
- Basic building block
  - Far voxel primitive
- Construction
  - BSP of the input model
  - Multiresolution structure
  - Far voxel
- Rendering
  - Selective refinement
  - Occlusion culling
  - Far voxel rendering
- Results
  - Preprocessing
  - Rendering



Far Voxels

Overview

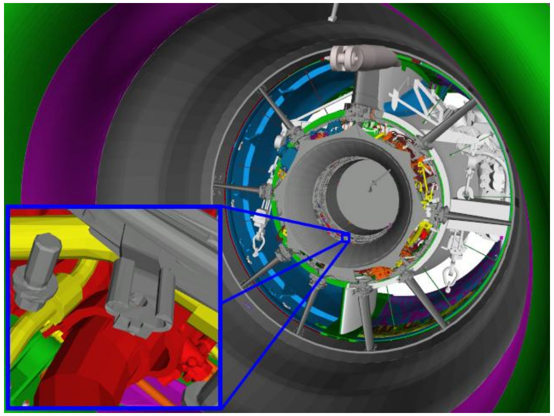
- Basic building block
  - Far voxel primitive
- Construction
  - BSP of the input model
  - Multiresolution structure
  - Far voxel
- Rendering
  - Selective refinement
  - Occlusion culling
  - Far voxel rendering
- Results
  - Preprocessing
  - Rendering

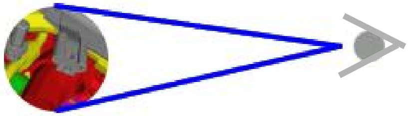


Far Voxels

The Far Voxel Concept

- Assumption: opaque surfaces, non participating medium
- Goal is to represent the appearance of complex far geometry
  - Near geometry can be represented at full resolution



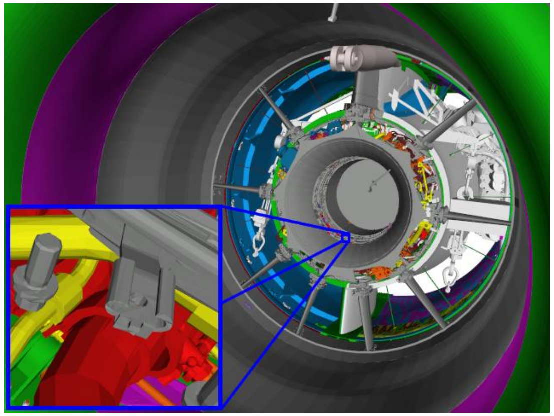


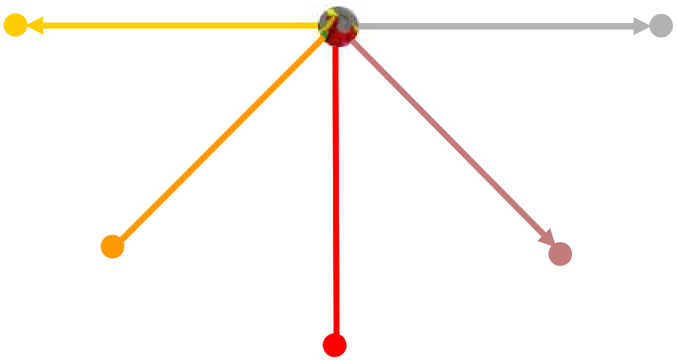
Far Voxels

The Far Voxel Concept

- Assumption: opaque surfaces, non participating medium
- Goal is to represent the appearance of complex far geometry
  - Near geometry can be represented at full resolution
- Idea is to discretize a model into many small volumes located in the neighborood of surfaces
  - Approximates how a small subvolume of the model reflects the incoming light

=> View-dependent voxel

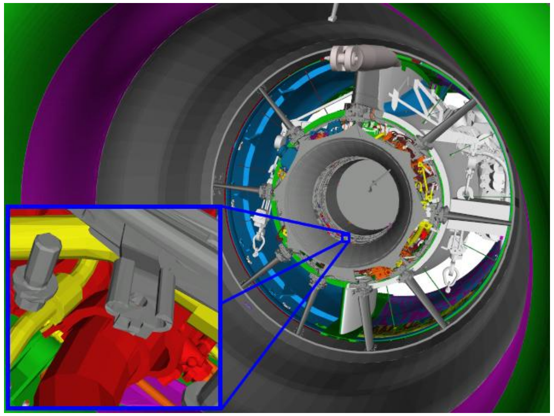




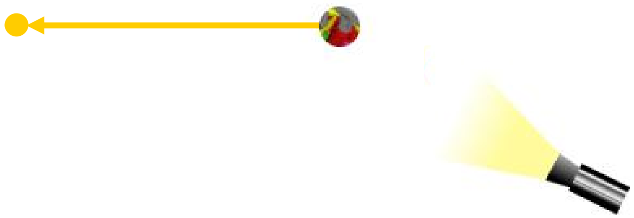
Far Voxels

The Far Voxel Concept

- A far voxel returns color attenuation given
  - View direction
  - Light direction



$$Shader_i(v,l) = BRDF_i(v,l)(n(v).l)_+$$



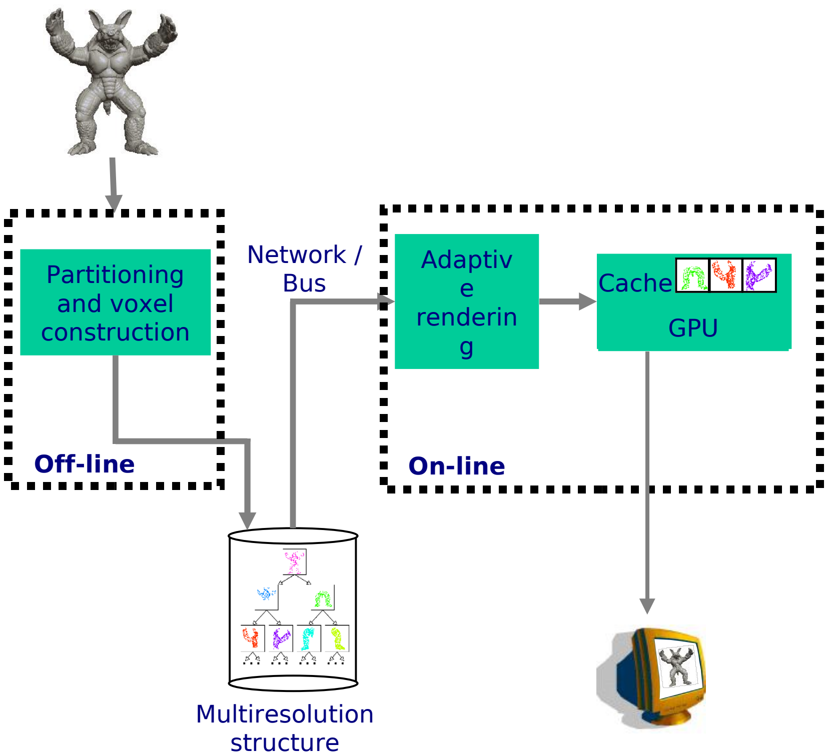
Shader = f (view direction, light direction)

- Rendered using a customized vertex shader executed on the GPU

Far Voxels

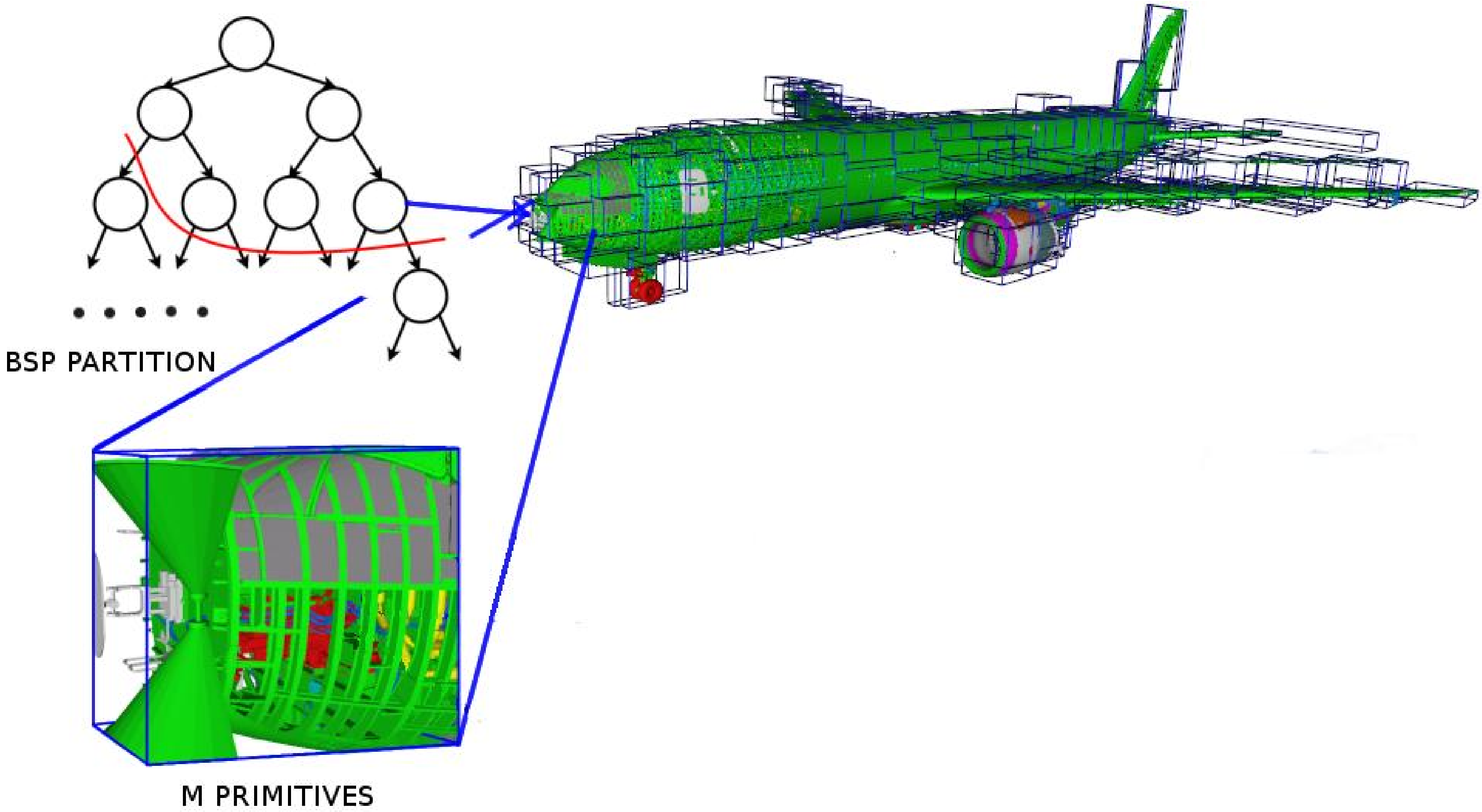
Overview

- Basic building block
  - Far voxel primitive
- Construction
  - BSP of the input model
  - Multiresolution structure
  - Far voxel
- Rendering
  - Selective refinement
  - Occlusion culling
  - Far voxel rendering
- Results
  - Preprocessing
  - Rendering

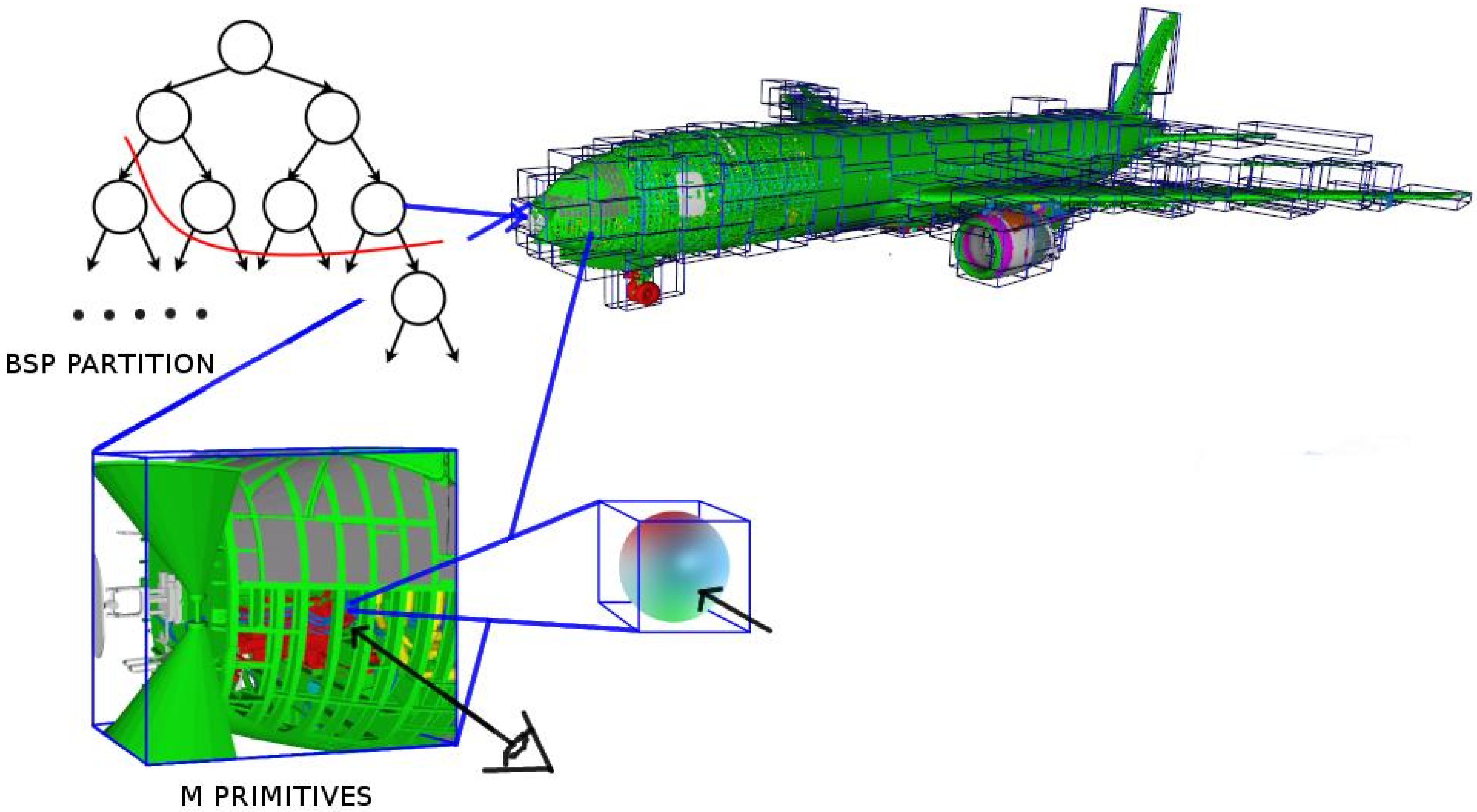


Far Voxels

Construction overview

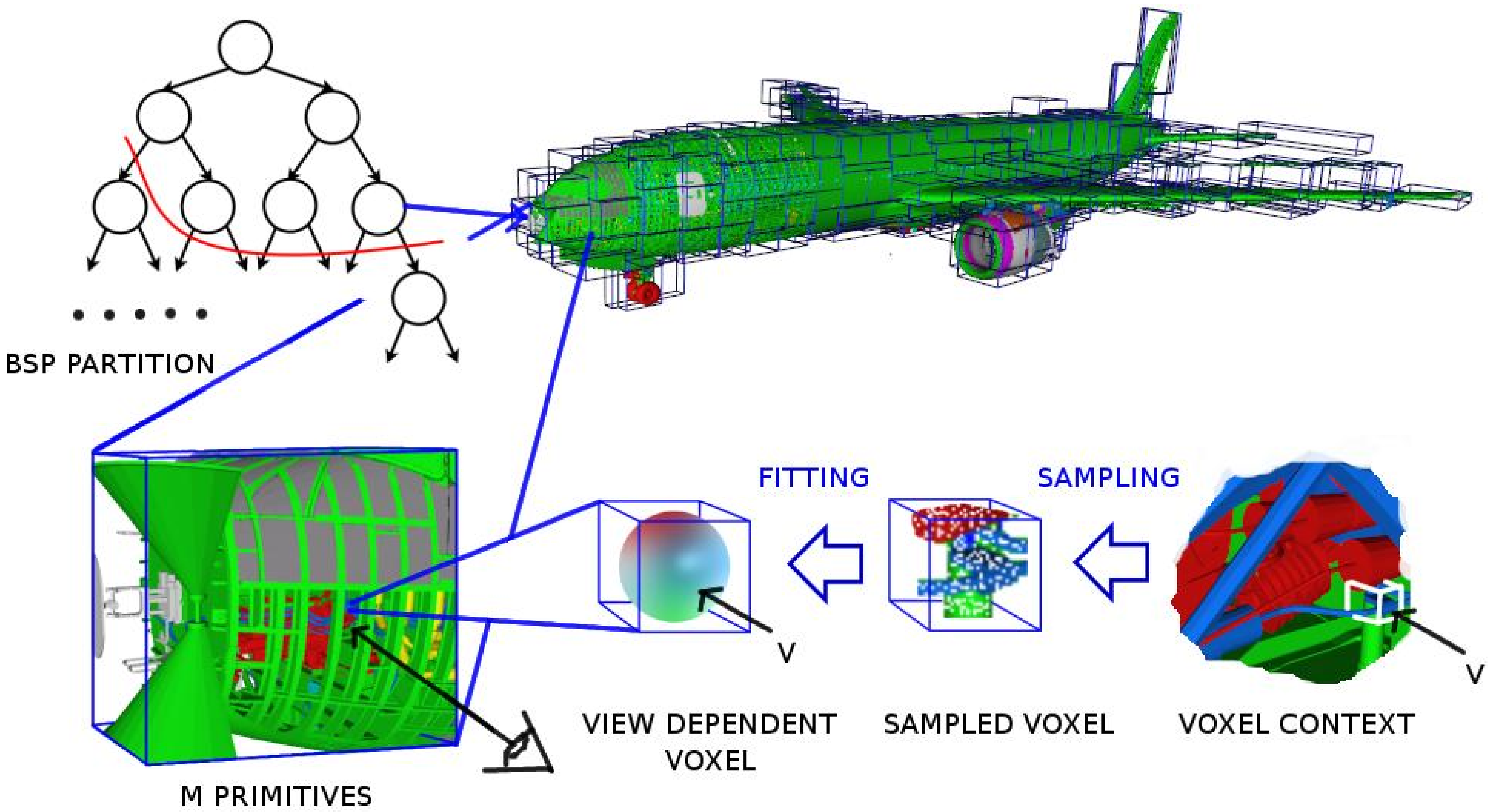


Far Voxels
Construction overview



Far Voxels

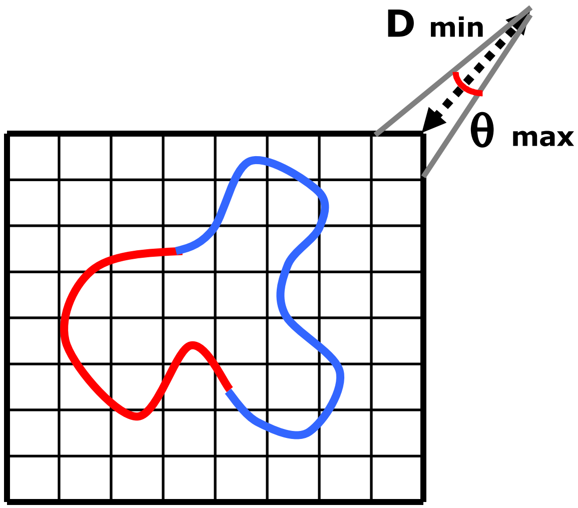
Construction overview



Far Voxels

Construction overview: Inner nodes

- Sample a model subvolume to build a grid of far voxels
- Voxels are far
  - Project to worst case θ_{max}
  - Viewed not closer than d_{min}

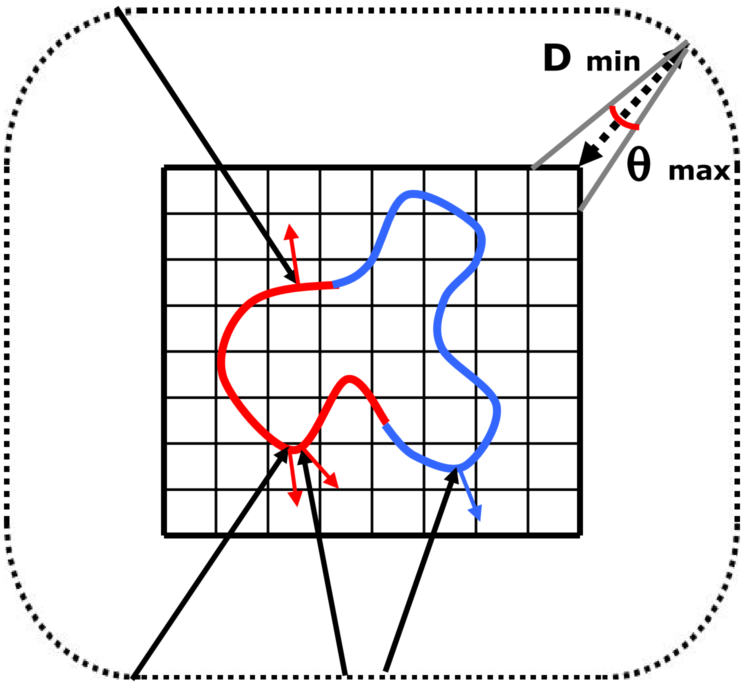


Section of the 3D grid of far voxels

Far Voxels

Construction overview: Inner nodes

- Sample a model subvolume to build a grid of far voxels
- Voxels are far
  - Project to worst case θ_{max}
  - Viewed not closer than d_{min}
- Raycasting samples original model and identifies visible voxels

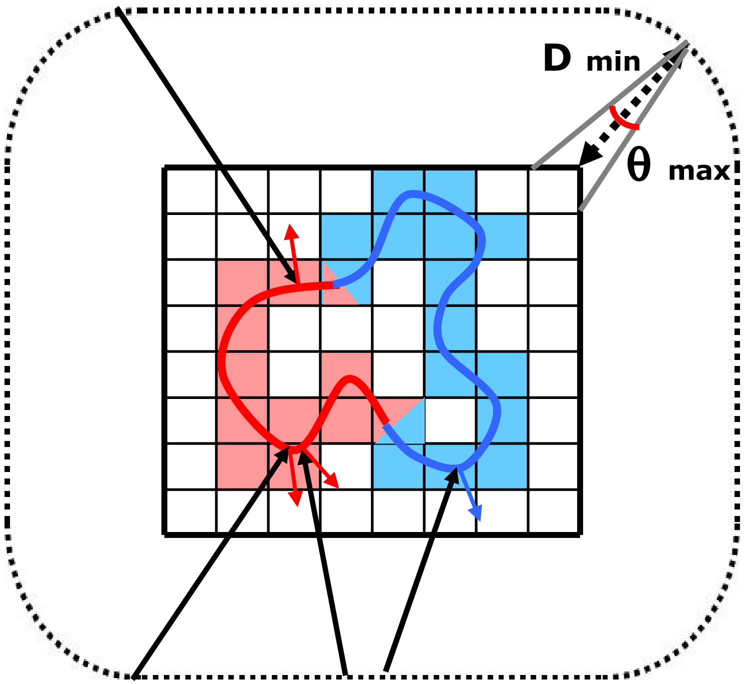


Section of the 3D grid of far voxels

Far Voxels

Construction overview: Inner nodes

- Sample a model subvolume to build a grid of far voxels
- Voxels are far
  - Project to worst case θ_{max}
  - Viewed not closer than d_{min}
- Raycasting samples original model and identifies visible voxels

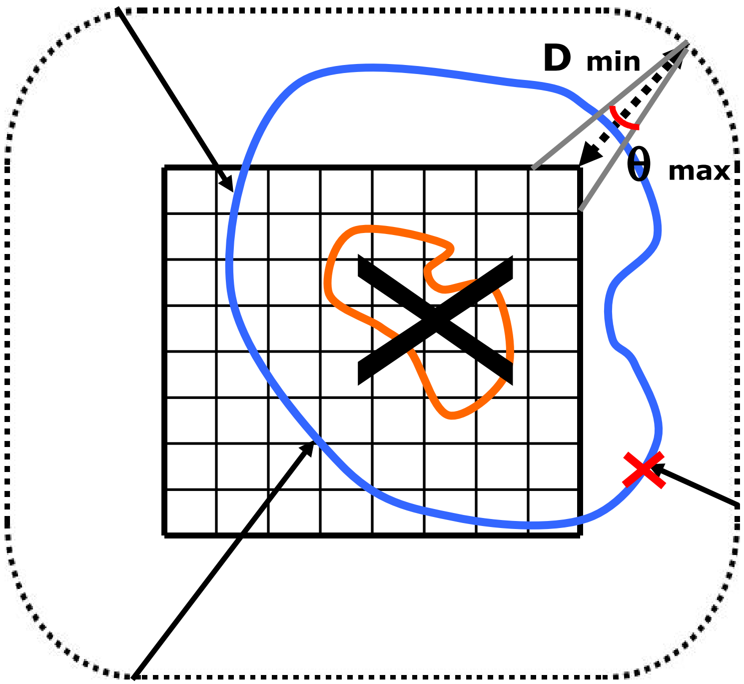


Section of the 3D grid of far voxels

Far Voxels

Construction overview: Object Space Occlusion

- Environment occlusion
- Cull interior part of grid of far voxels

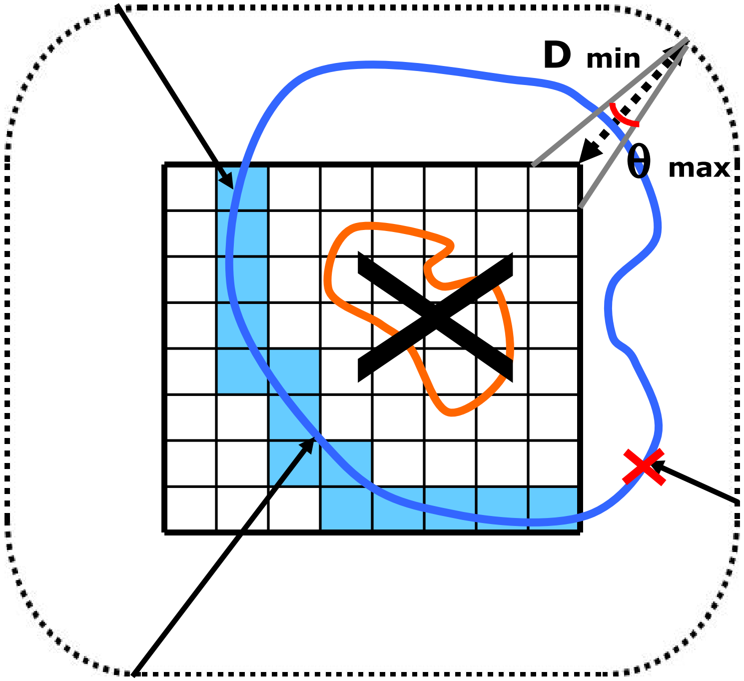


Section of the 3D grid of far voxels

Far Voxels

Construction overview: Object Space Occlusion

- Environment occlusion
- Cull interior part of grid of far voxels

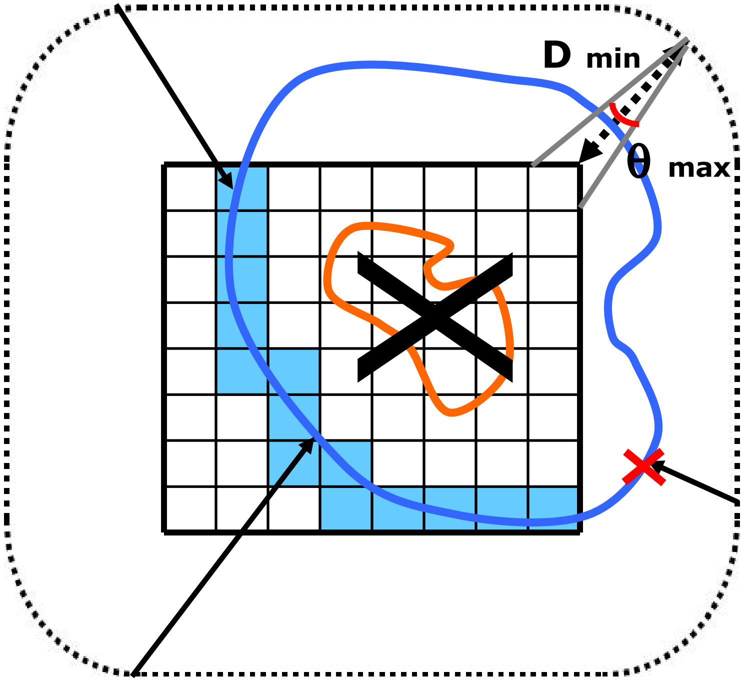


Section of the 3D grid of far voxels

Far Voxels

Construction overview: Object Space Occlusion

- Environment occlusion
- Cull interior part of grid of far voxels
- Culls 40% of the high depth complexity Boeing 777 model,
  - worst case θ_{max} = 0.5 deg (~10 pixel tolerance for 1024x1024 viewport using 50deg FOV)
- Minimize artifacts due to leaking of occluded parts of different colors

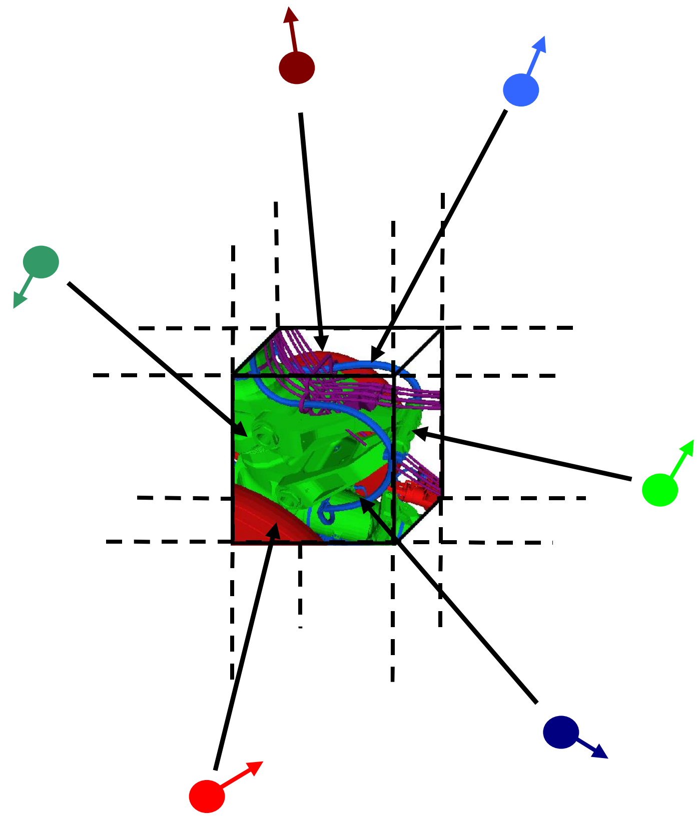


Section of the 3D grid of far voxels

Far Voxels

Construction overview: Far Voxel

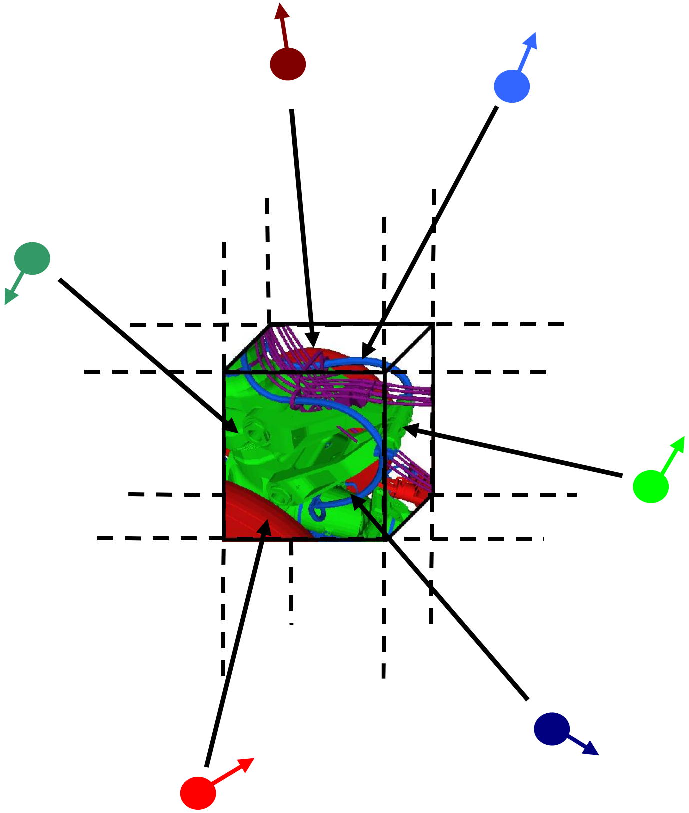
- Consider voxel subvolume
- Samples gathered from unoccluded directions
  - Sample:
    - (BRDF, **n**) = f(view direction)



Far Voxels

Construction overview: Far Voxel

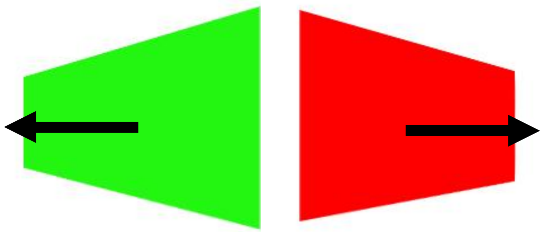
- Consider voxel subvolume
- Samples gathered from unoccluded directions
  - Sample:
    - (BRDF, **n**) = f(view direction)
- Compress shading information by fitting samples to a compact analytical representation



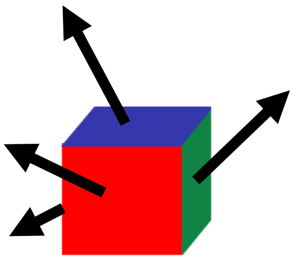
Far Voxels

Construction overview: Far Voxel Shaders

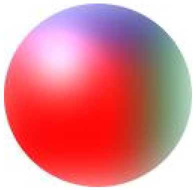
- Phenomenological shader types
  - Flat shader
    - Normal
    - Front and back material
  - Smooth shader
    - 6 normals
    - 6 materials
    - associated with ± x, ± y, ± z



Flat shader: front and back material



Smooth shader: stored info

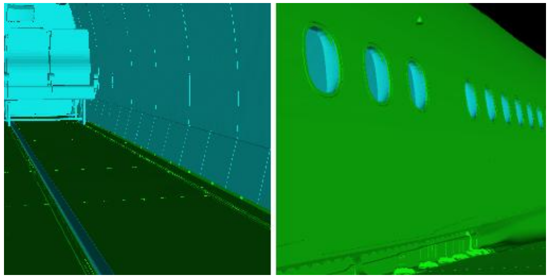


view dep. representation

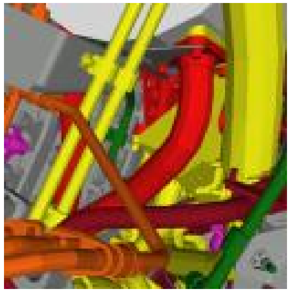
Far Voxels

Construction overview: Far Voxel Shaders

- Phenomenological shader types
  - Flat shader
    - Normal
    - Front and back material
  - Smooth shader
    - 6 normals
    - 6 materials
    - associated with ± x, ± y, ± z



Flat shader: front and back material

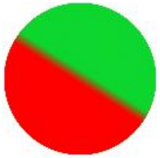


Smooth shader: complex geometry

Far Voxels

Construction overview: Far Voxel Shaders

- Build all the K different far voxels representations
  - K = flat, smooth..
  - Principal component analysis
- Evaluate each representation error
  - Compare real values (samples) with the voxel approximations from the sample direction



Flat proxy:
2 components

Smooth proxy:
6 components

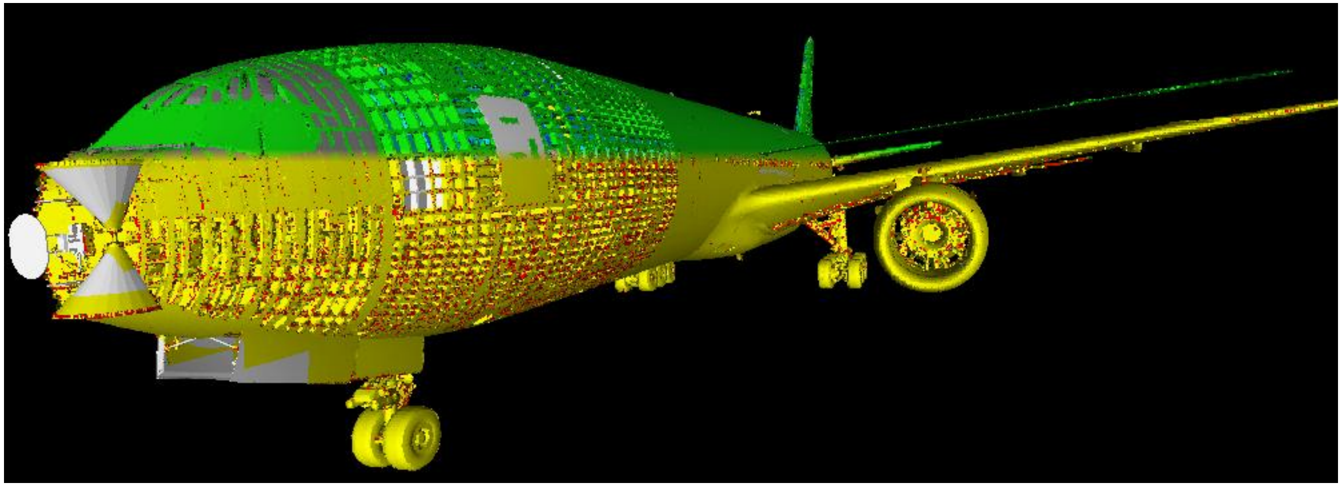
Others...

$$\mathrm{Err}_{(\mathbf{k})} = \sum_i \sum_j \left(BRDF_i^{(sampled)}(\mathbf{v}_i, \mathbf{l}_j) \max(\mathbf{n}_i \cdot \mathbf{l}_j, 0) - Shader^{(k)}(\mathbf{v}_i, \mathbf{l}_j) \right)^2$$

- Choose approximation with lowest error

Far Voxel Distribution
on a perspective view of the Boeing 777

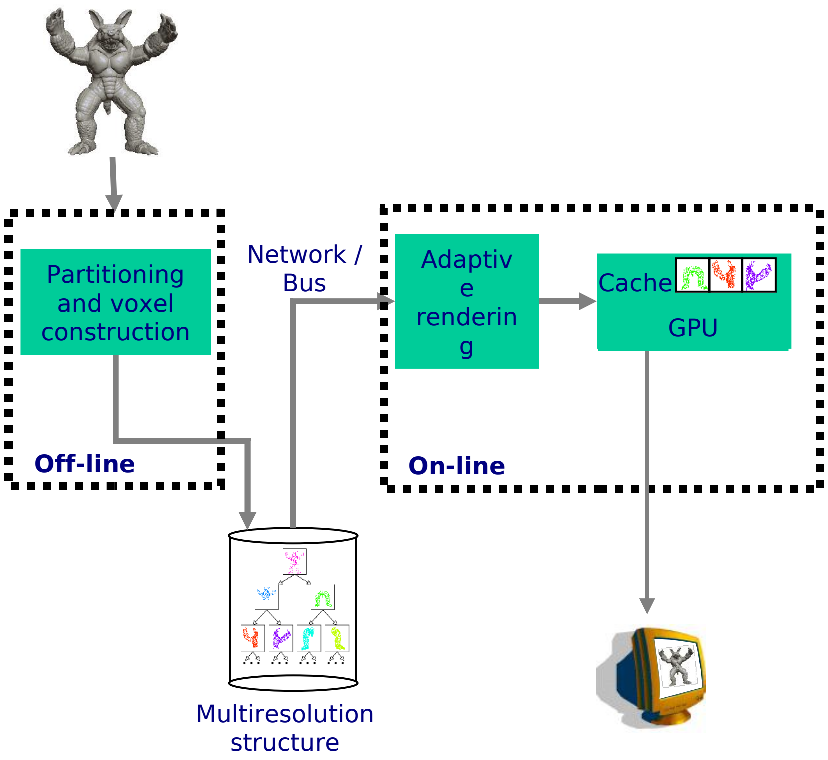
- Flat shaders
- Smooth shaders (complex local geometry)
- Triangles



Far Voxels

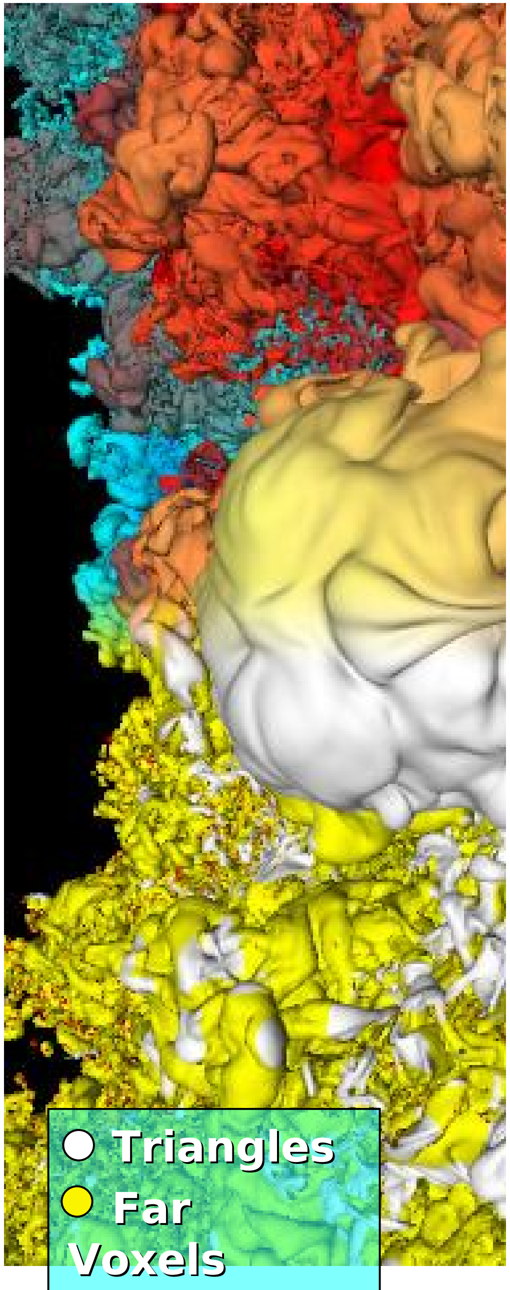
Overview

- Basic building block
  - Far voxel primitive
- Construction
  - BSP of the input model
  - Multiresolution structure
  - Far voxel
- Rendering
  - Selective refinement
  - Occlusion culling
  - Far voxel rendering
- Results
  - Preprocessing
  - Rendering



Far Voxels
Rendering

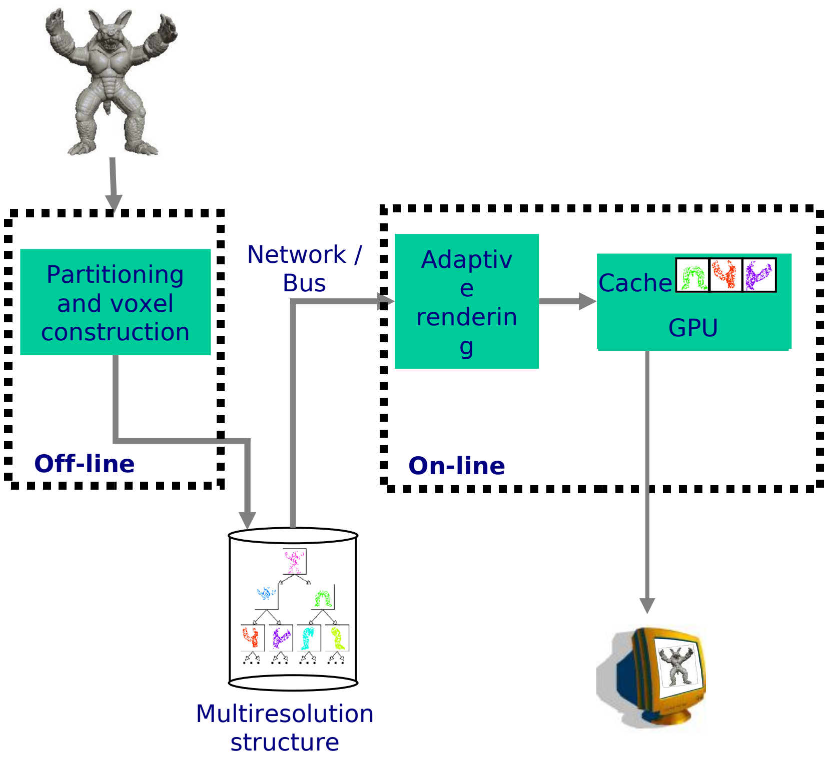
- Hierarchical traversal with coherent culling
  - Stop when out-of view, occluded (GPU feedback), or accurate enough
- Leaf node: Triangle rendering
  - Draw the precomputed triangle strip
- Inner node: Voxel rendering
  - For each far voxel type
    - Enable its shader
    - Draw all its view dependent primitives using glDrawArrays
  - Splat voxels as antialiased point primitives
  - Limits
    - Does not consider primitive opacity
    - Rendering quality similar to one-pass point splat methods (no sorting/blending)



Far Voxels

Overview

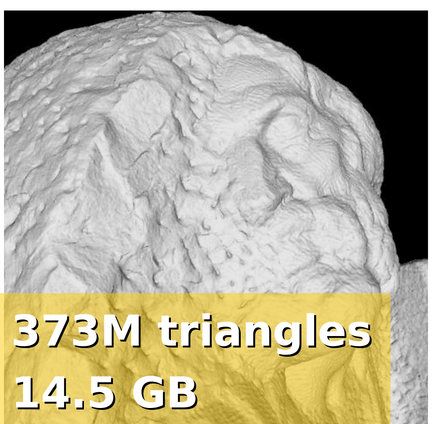
- Basic building block
  - Far voxel primitive
- Construction
  - BSP of the input model
  - Multiresolution structure
  - Far voxel
- Rendering
  - Selective refinement
  - Occlusion culling
  - Far voxel rendering
- Results
  - Preprocessing
  - Rendering

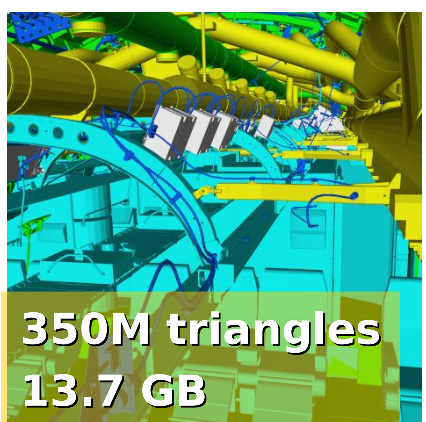


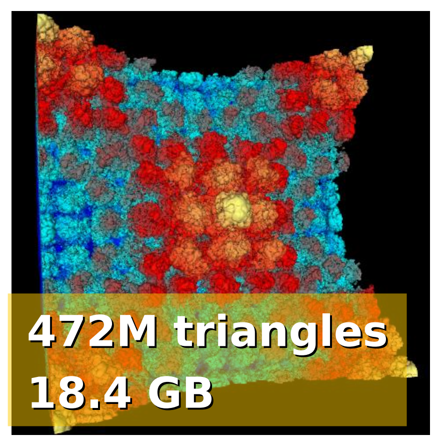
Far Voxels

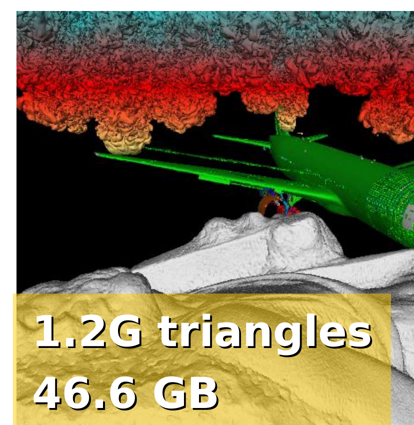
Results

- Tested on extremely complex heterogeneous surface models
  - St.Matthew, Boeing 777, Richtmyer Meshkov isosurf., all at once
- Tested in a number of situations
  - Single processor / cluster construction
  - Workstation viewing, large scale display





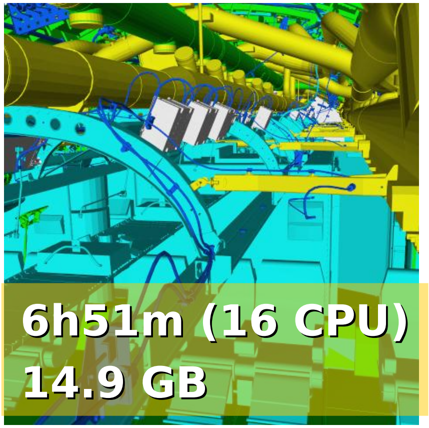


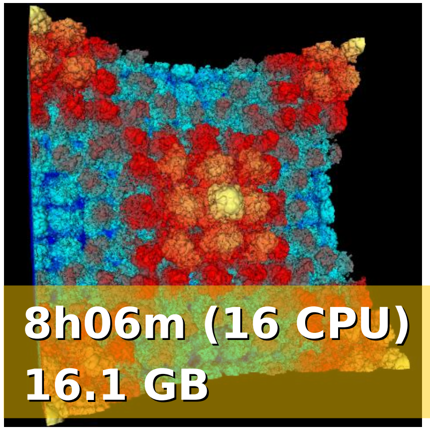


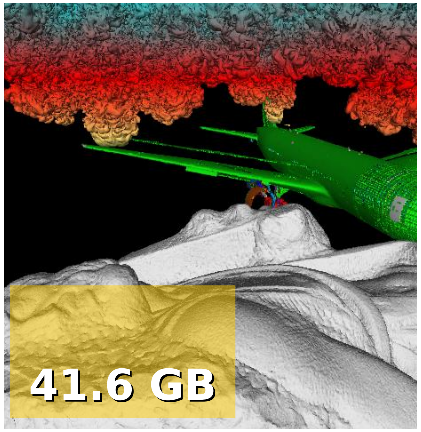
Far Voxels

Results

- 1-16 Athlon 2200+ CPU, 3 x 70GB ATA 133 Disk (IDE+NFS)
- 1-20K triangles/sec
  - Scales well, limited by slow disk I/O for large meshes
  - Slow!! (but similar to recent adaptive tessellation methods)
- Avg. triangles per leaf 5K
- Avg. voxels per inner node 2.5K



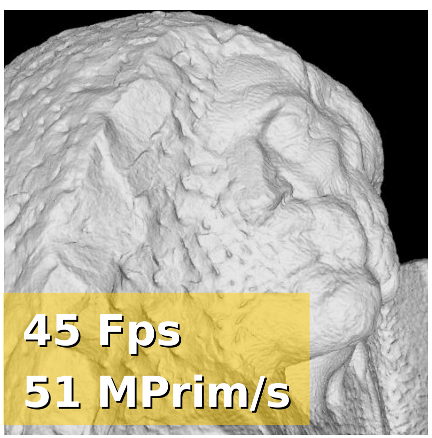


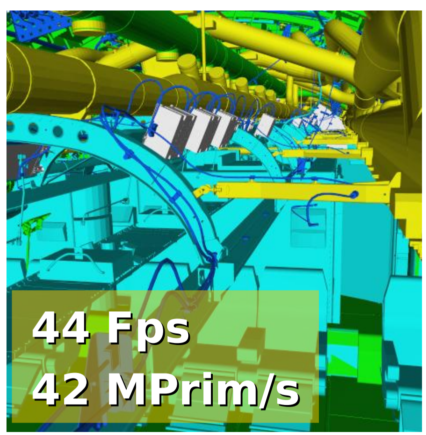


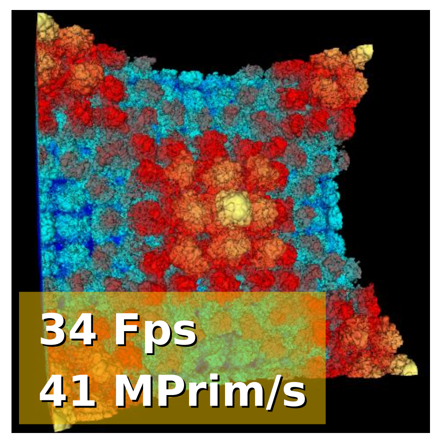
Far Voxels

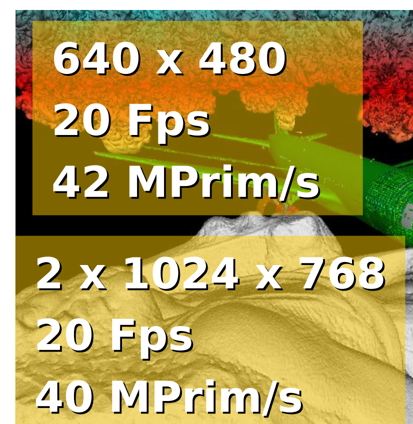
Results

- Xeon 2.4GHz, 70GB SCSI 320 Disk, GeForce FX6800GT AGP 8x
- Window size: from video resolution to stereo projector display
  - St.Matthew, Boeing, Isosurface: 640 x 480
  - All at once: 640 x 480 and Stereo 2 x 1024 x 768
- Pixel tolerance: [Target 1 | Actual ~0.9 | Max ~10]
- Resident set size limited to ~200 MB





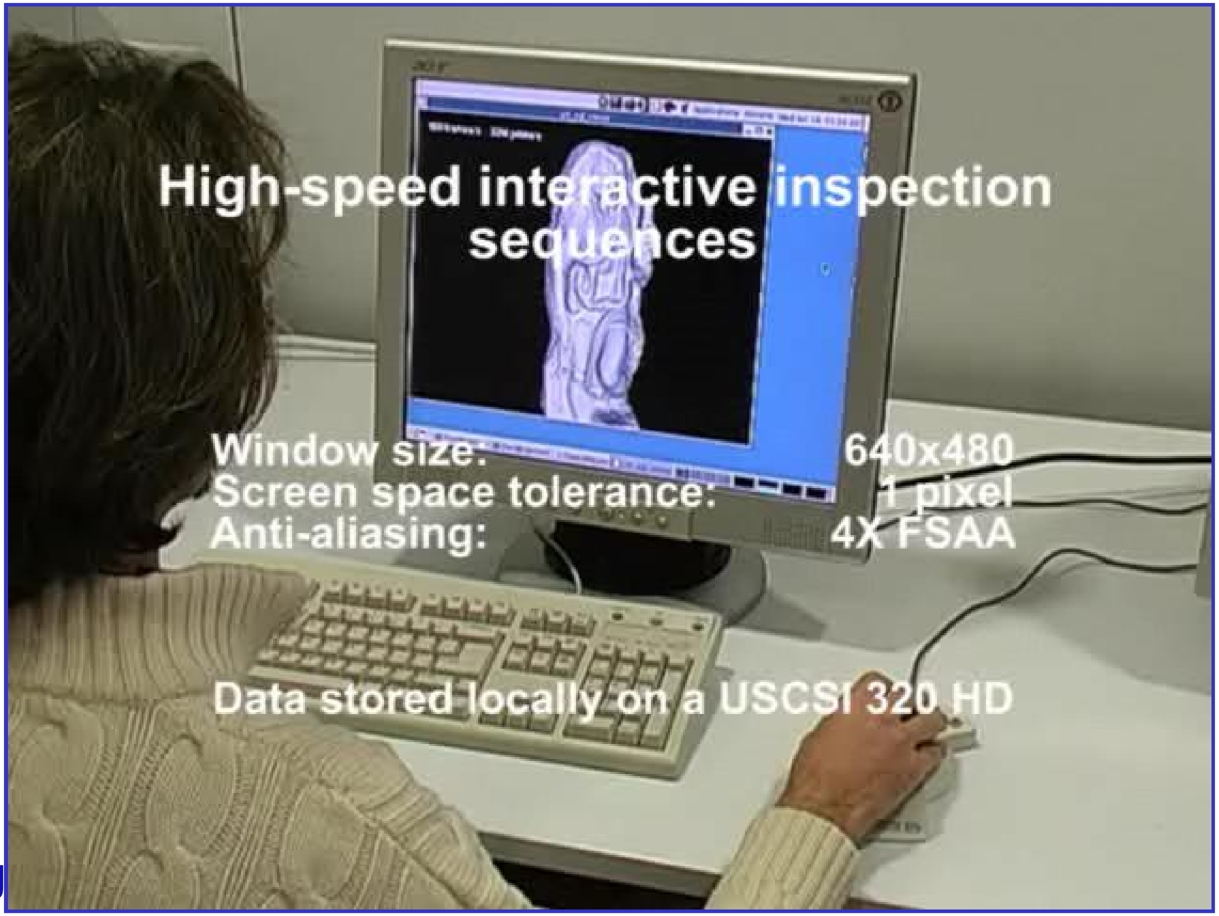




Far Voxels

Conclusions

- General purpose technique that targets many model kinds
  - Seamless integration of
    - multiresolution
    - occlusion culling
    - out-of-core data management
  - High performance
  - Scalability
- Main limitations
  - Slow preprocessing
  - Non-photorealistic rendering quality



Intel Xeon 2.4GHz 1GB, GeForce 6800GT AGP8X

Our contributions

GPU-friendly output-sensitive techniques

***-BDAM – Local and Global Terrain Models**

Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Ponchio/Scopigno (CNR)

EG 2003, IEEE Viz 2003, EG 2005

Adaptive Tetrapuzzles – Dense meshes

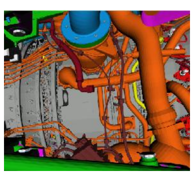
Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Ponchio/Scopigno (CNR

SIGGRAPH 2004

Layered Point Clouds – Dense clouds

Gobbetti/Marton (CRS4)

SPBG 2004 / Computers & Graphics 2004



Far Voxels – General

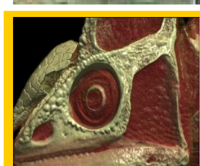
Gobbetti/Marton (CRS4)

SIGGRAPH 2005

Blockmaps – Hybrid volumetric city model

Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Di Benedetto/Scopigno (CNR)

EG 2007

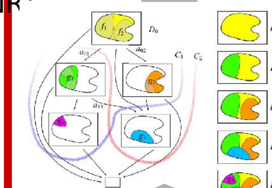


MOVR – Volumetric models

Gobbetti/Marton/Iglesias Guitian (CRS4)

CGI 2008

Specialize

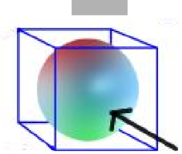


Chunked Multi-Triangulations

Gobbetti/Marton (CRS4), Cignoni/Ganovelli/Ponchio/Scopigno (CNR) *IEEE Viz 2005*

Generalize

Specialize



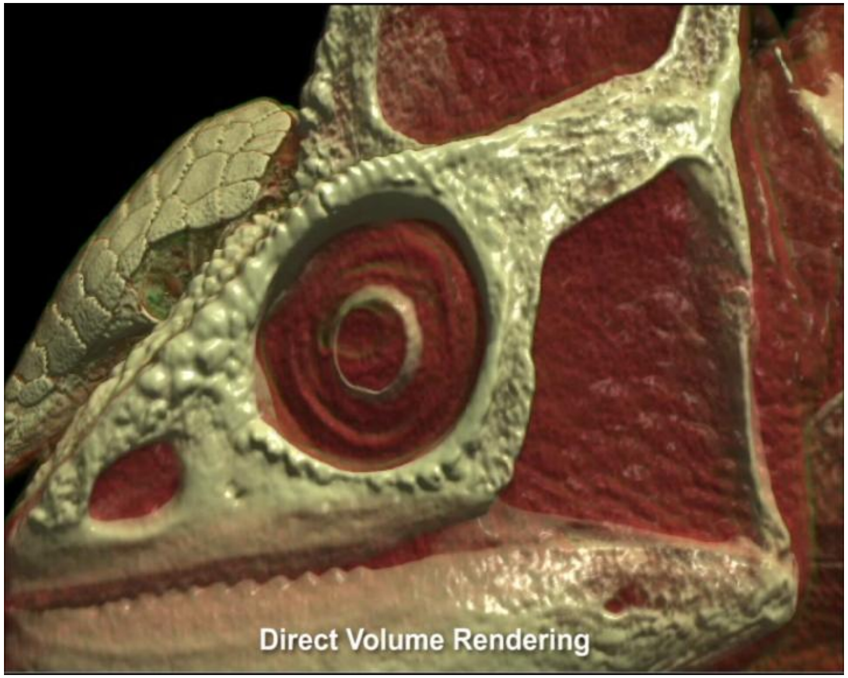
View-dep. Volumetric Model

In progress

Generalize

Our contributions
MOVR – Massive volumetric datasets

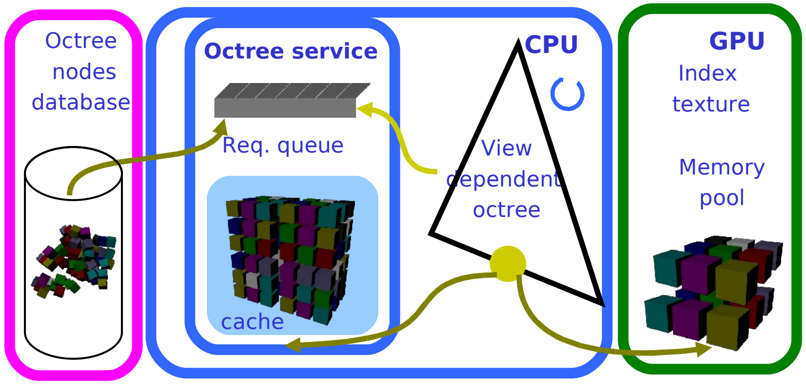
- High quality visualization of massive out-of-core datasets
  - CPU/GPU cooperation
  - Adaptive generation of view-dependent working set in GPU memory
  - Rendering via single-pass GPU ray casting

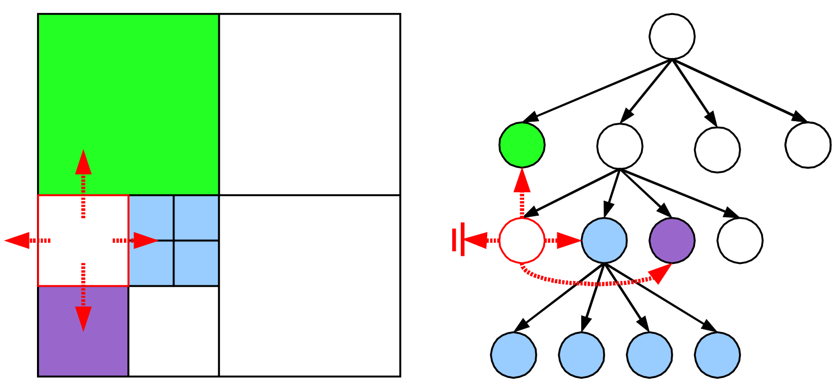


Enrico Gobbetti, Fabio Marton, and José Antonio Iglesias Guitián.
A single-pass GPU ray casting framework for interactive out-of-core rendering of massive volumetric datasets.
The Visual Computer, 24, 2008. Proc. CGI 2008, to appear.

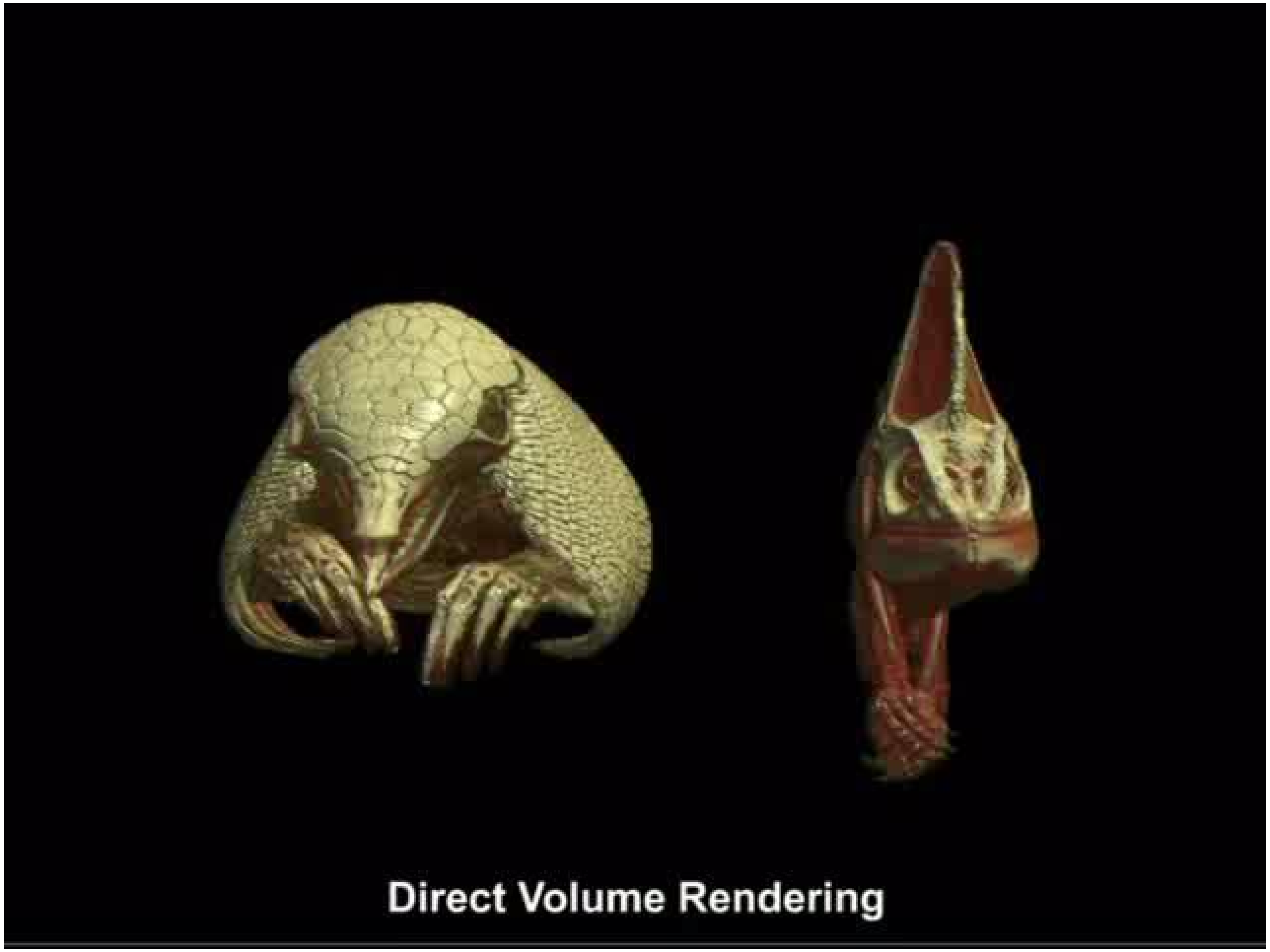
Our contributions
MOVR – Massive volumetric datasets

- Overview
  - Use a CPU runtime loader that updates a view and transfer function – dependent **working set of bricks**
  - **Asynchronously** mantain bricks on both CPU and GPU memory fetching data from the out-of-core octree
  - **Adaptive refinement** method guided by priority:
    - Sorted by decreasing projected screen-space size of voxels
    - Use feedback from occlusion queries
  - **Spatial Index Texture**
  - **Stackless GPU raycaster**
  - **Visibility & Occlusion culling**





Our contributions
MOVR – Massive volumetric datasets



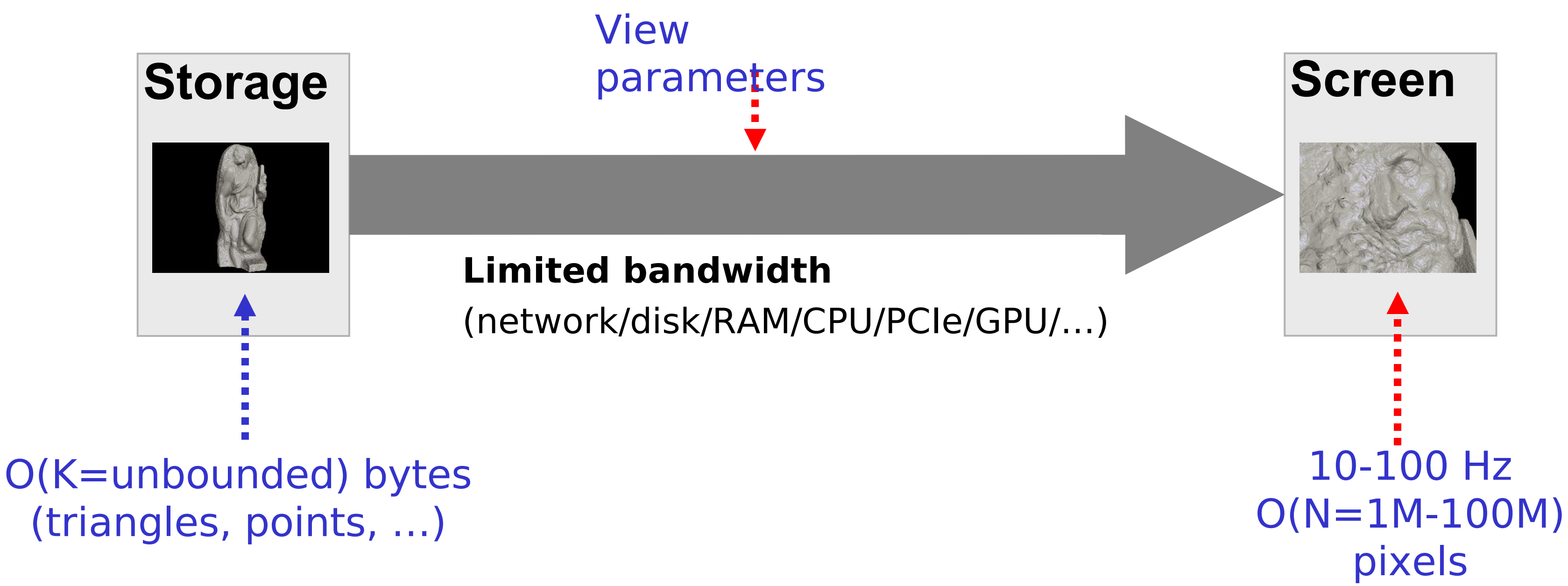
Xeon 2.4GHz / 1GB RAM / 70GB SCSI 320 Disk / NVIDIA 8800GTX

Time for a conclusion, right?

Size matters! Or does it?

A real-time data filtering problem!

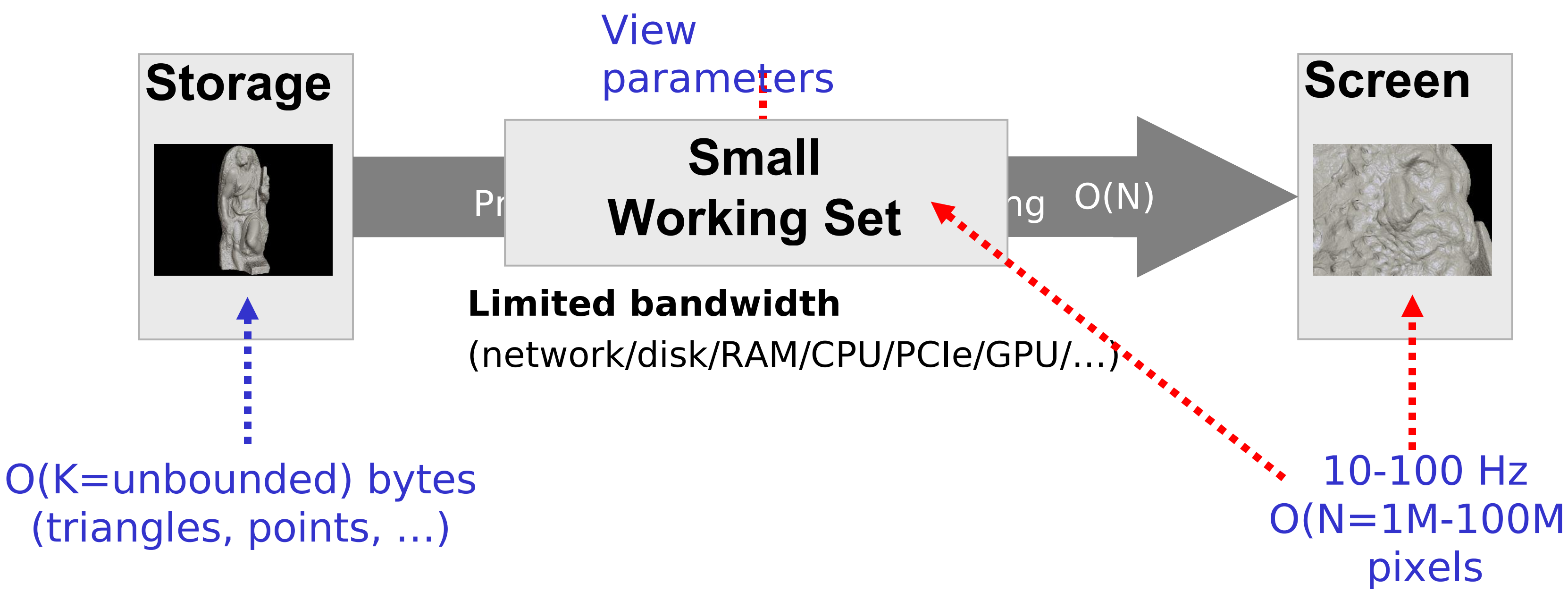
- Models of unbounded complexity on limited computers
  - We assume **less data on screen (N) than in model (K $\to\infty$)**
  - Need for **output-sensitive** techniques (O(N), not O(K))



Size matters! Or does it?

A real-time data filtering problem!

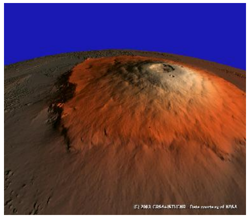
- Models of unbounded complexity on limited computers
  - We assume **less data on screen (N) than in model (K $\rightarrow\infty$)**
  - Need for **output-sensitive** techniques (O(N), not O(K))



Application domains / data sources

Local Terrain Models

2.5D – Flat – Dense regular sampling

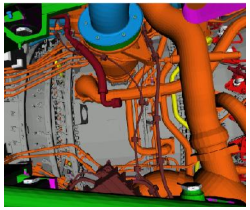


Planetary terrain models

2.5D – Spherical – Dense regular sampling

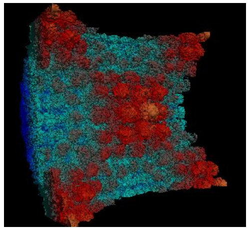
Laser scanned models

3D – Moderately simple topology – low depth complexity - dense



CAD models

3D – complex topology – high depth complexity – structured - 'ugly' mesh



Natural objects / Simulation results

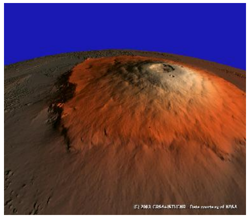
3D – complex topology + high depth complexity + unstructured/high frequency details

- Many important application domains
- Models exceed
  - $O(10^8\text{-}10^9)$ samples
  - $O(10^9)$ bytes
- Varying
  - Dimensionality
  - Topology
  - Sampling distribution

Application domains / data sources

Local Terrain Models

2.5D – Flat – Dense regular sampling

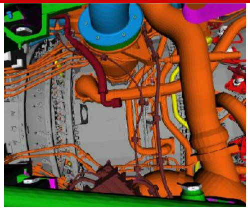


Planetary terrain models

2.5D – Spherical – Dense regular sampling

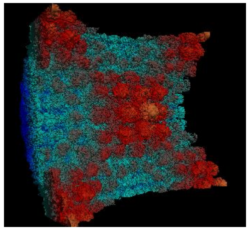
Laser scanned models

3D – Moderately simple topology – low depth complexity - dense



CAD models

3D – complex topology – high depth complexity – structured - 'ugly' mesh

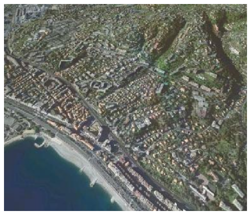


Natural objects / Simulation results

3D – complex topology + high depth complexity + unstructured/high frequency details

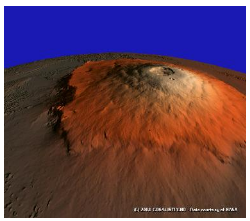
- "Well behaved" surfaces
- Multiresolution dominates visibility
- Good results with surface based methods based on sequences of local modifications
- GPU-MT / TetraPuzzles / … already fast/good enough

Application domains / data sources



Local Terrain Models

2.5D – Flat – Dense regular sampling

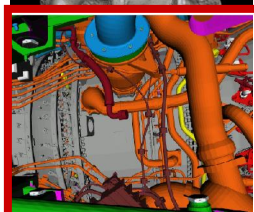


Planetary terrain models

2.5D – Spherical – Dense regular sampling

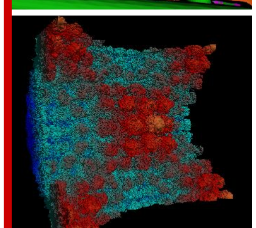
Laser scanned models

3D – Moderately simple topology – low depth complexity - dense



CAD models

3D – complex topology – high depth complexity – structured - 'ugly' mesh



Natural objects / Simulation results

3D – complex topology + high depth complexity + unstructured/high frequency details

- Highly complex surfaces / volumes
- Visibility needs to be tightly combined with LODs
- Need to go to volumetric models
- Far Voxels/MOVR are state-of-the-art solution
- Still not the final word…

So many things, so little time...

- More info:
  http://www.crs4.it/vic/
- Q&A: Your turn…