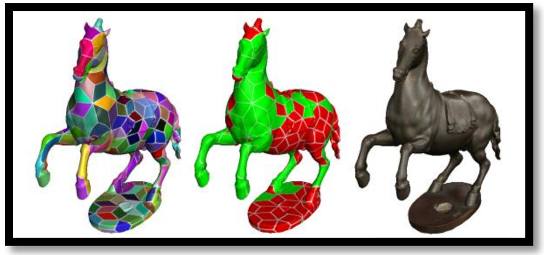
Part 4.1

Scalable Mobile Visualization: Introduction

Enrico Gobbetti, CRS4

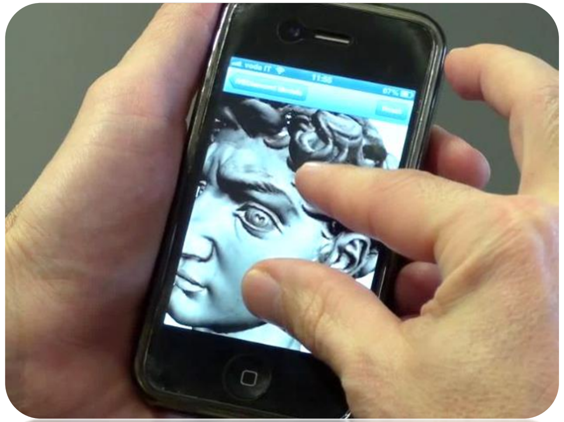
Scalable mobile visualization

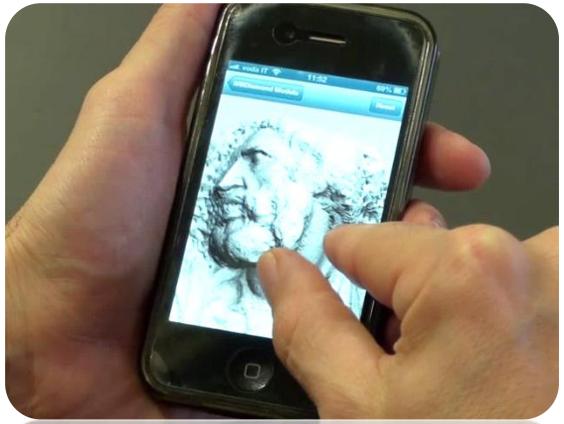
- **Goal is high quality interactive rendering of complex scenes…**
  – Large data, shading, complex illumation, …
- **… on mobile platforms …**
  – Mostly smartphones or tablets
  – Similar considerations can apply to other settings (e.g., embedded systems)
- **Wide variety of applications**
  – Gaming, visualization, cultural heritage…



Mobile platforms scenario

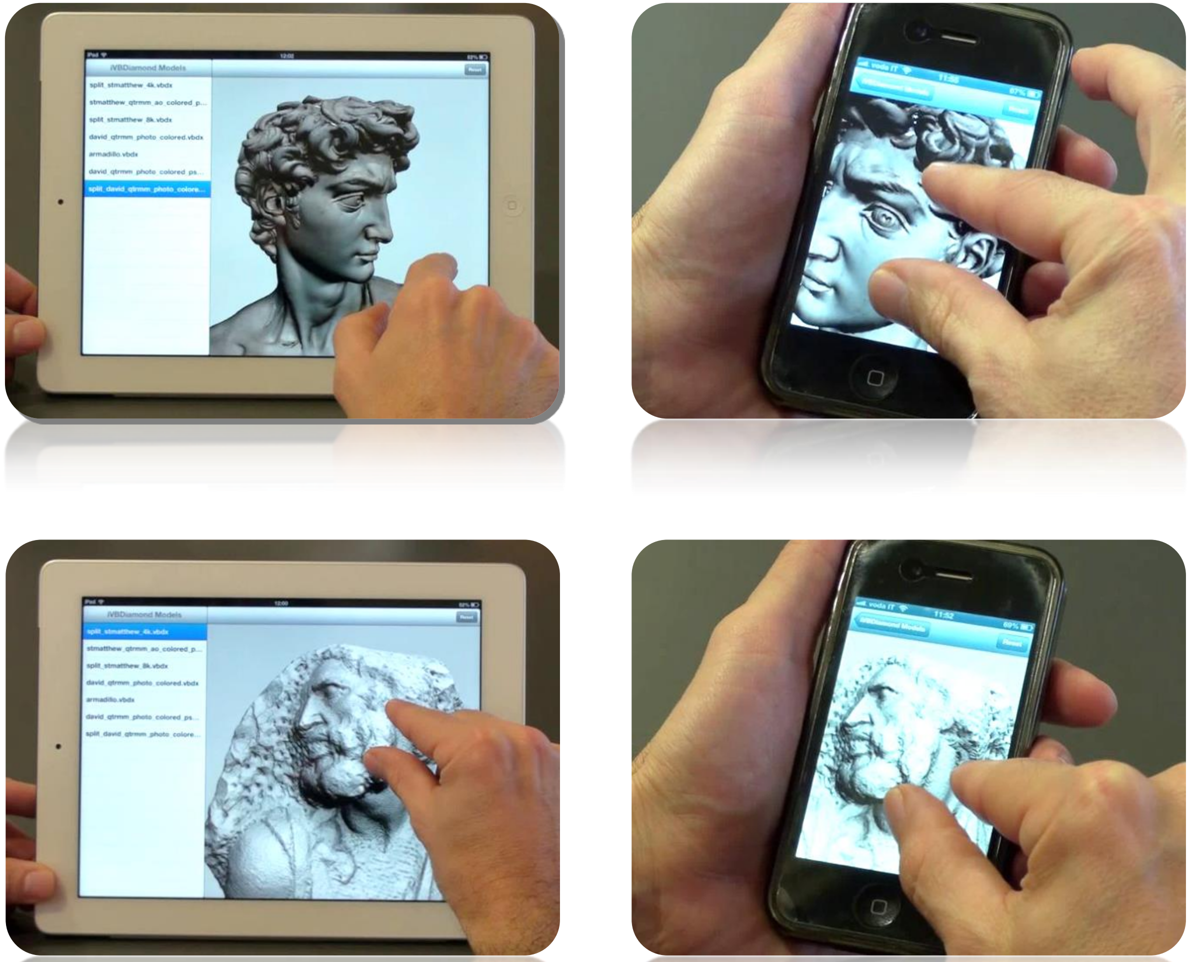
- **Typical scalable rendering problem, but with some specific constraints wrt standard (desktop settings)**
- **… screen resolutions are often extremely large (2 – 6 Mpix)**
  - Lots of pixels to generate!
- **… mobile 3D graphics hardware is powerful but still constrained**
  - Reduced computing powers, memory bandwidths, and amounts of memory wrt desktop graphics systems
  - Limited power supply!





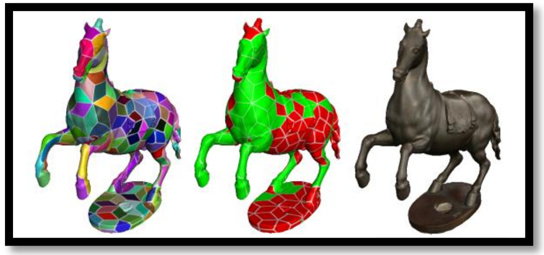
Mobile rendering scenario

- **No brute force method applicable**
  - Need for “smart methods” to perform interactive rendering
  - Exploit at best reduced rendering power
- **Proposed solutions**
  - **Render only necessary data**: adaptive multiresolution
  - **Limit required CPU/GPU work**: full or partial precomputation
  - **Limit data requirements**: streaming approaches
  - **Exploit at best available bandwidth**: data compression



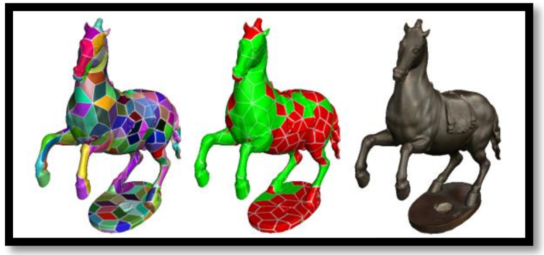
Related Work on mobile visualization

- ***(See previous session for details)***
- **Remote Rendering**
  - …..
- **Local Rendering**
  - **Model based**
    - Original models
    - Multiresolution models
    - Simplified models
      - Line rendering
      - Point cloud rendering
  - **Image based**
    - Image impostors
    - Environment maps
    - Depth images
  - **Smart shading**
  - **Volume rendering**



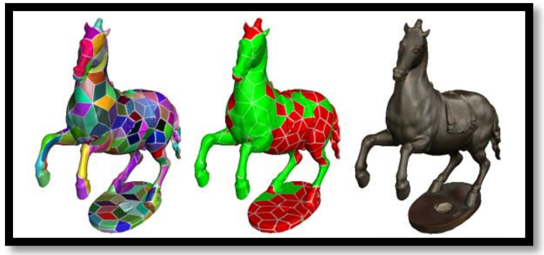
Related Work on mobile visualization

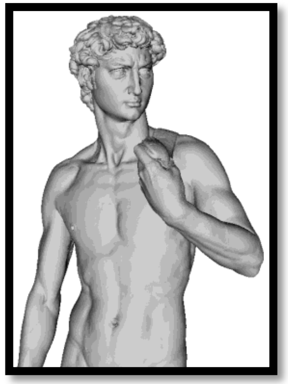
- ***(See previous session for details)***
- **Remote Rendering**
  - …..
- **Local Rendering**
  - **Model based**
    - Original models
    - **Multiresolution models**
    - Simplified models
      - Line rendering
      - Point cloud rendering
  - **Image based**
    - Image impostors
    - **Environment maps**
    - Depth images
  - **Smart shading**
  - **Volume rendering**



Scalable Mobile Visualization

- **Big/complex models:**
  - Detailed scenes from modeling, capturing..
    - Output sensitive: adaptive multiresolution
    - Compression / simple decoding
- **Complex rendering**
  - Global illumination
    - Pre-computation
    - Smart shading
  - Volume rendering
    - Compression / simple decoding



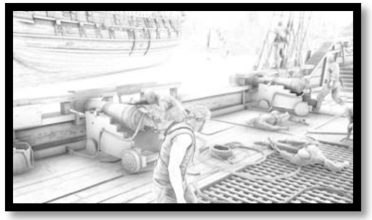


Scalable Mobile Visualization. Outline

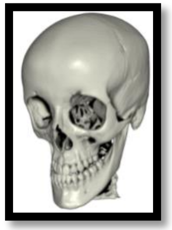
Large meshes

High quality illumination: full precomputation

High quality illumination: smart computation



Volume data



Part 4.2

Scalable Mobile Visualization: Large Meshes

Fabio Marton, CRS4

Scalable Mobile Visualization

Extremely

Massive

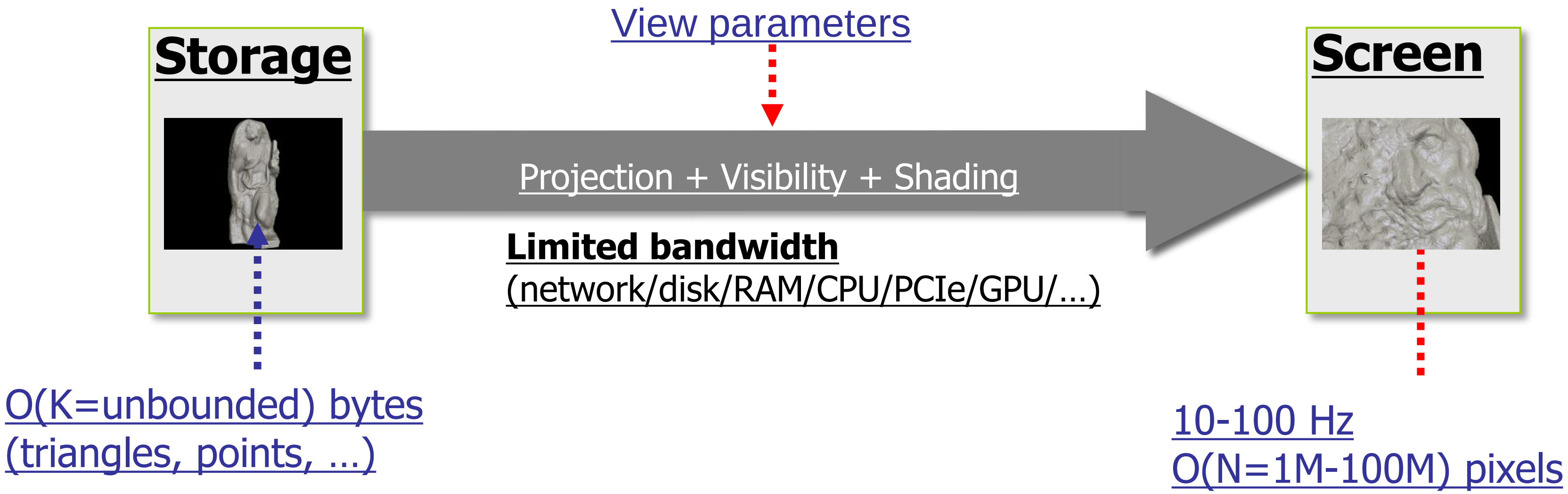
3D Models

Scalable Mobile Visualization

Itty bitty living space!

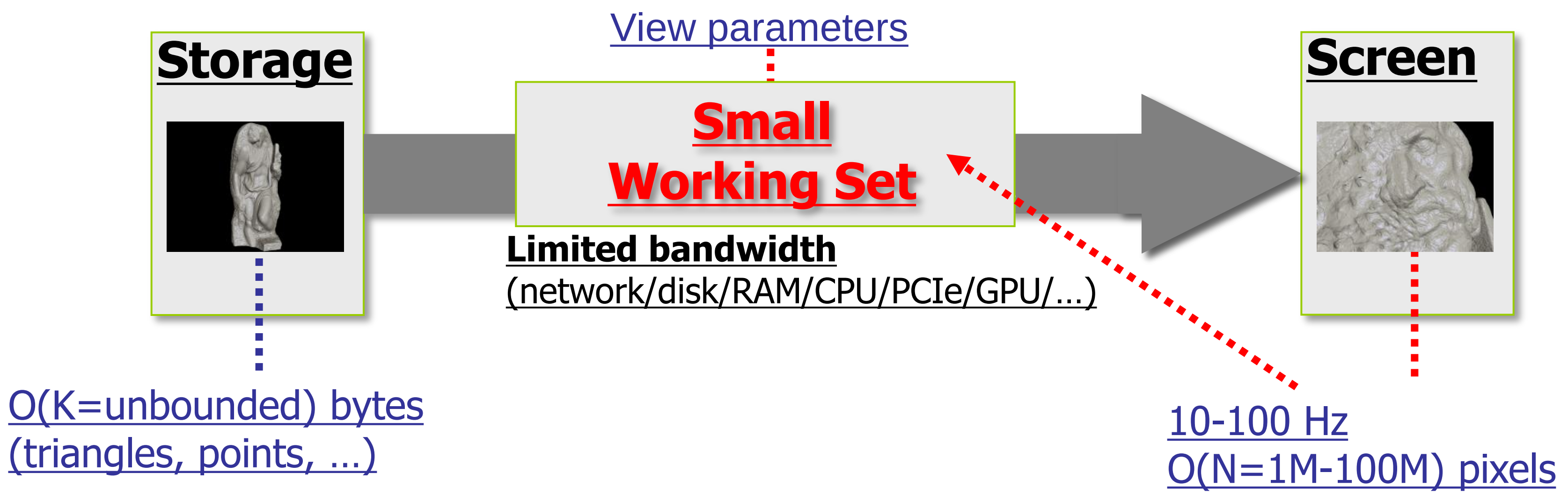
A real-time data filtering problem!

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



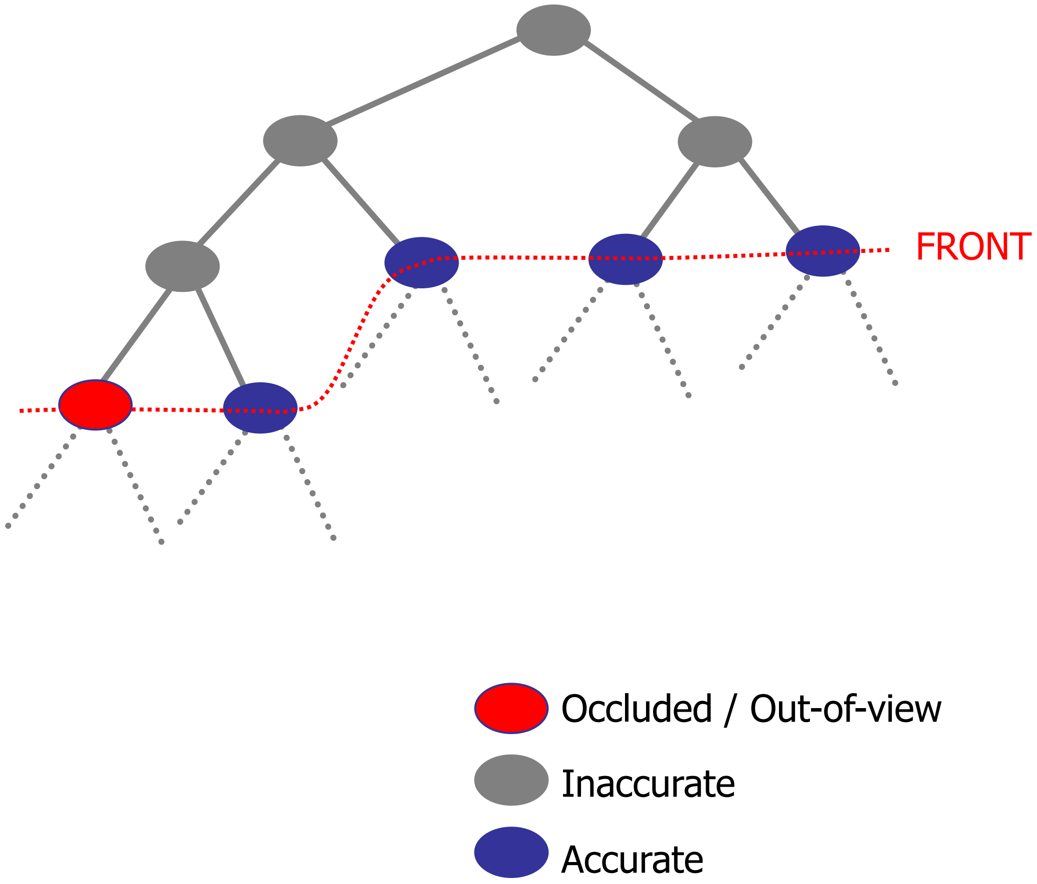
A real-time data filtering problem!

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



Output-sensitive techniques

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

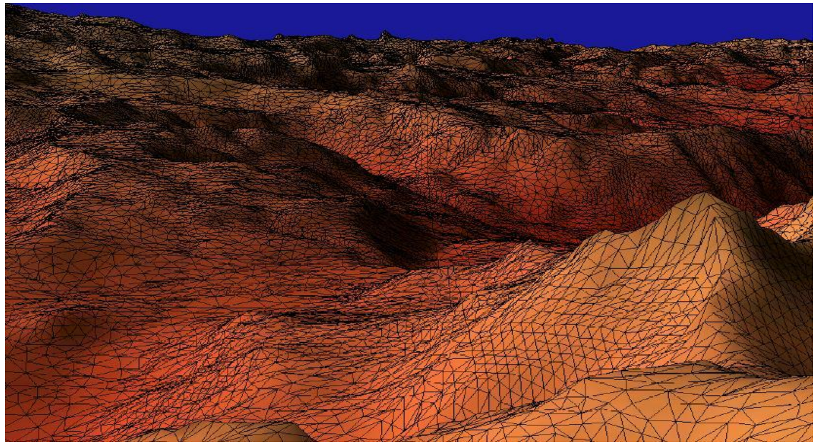


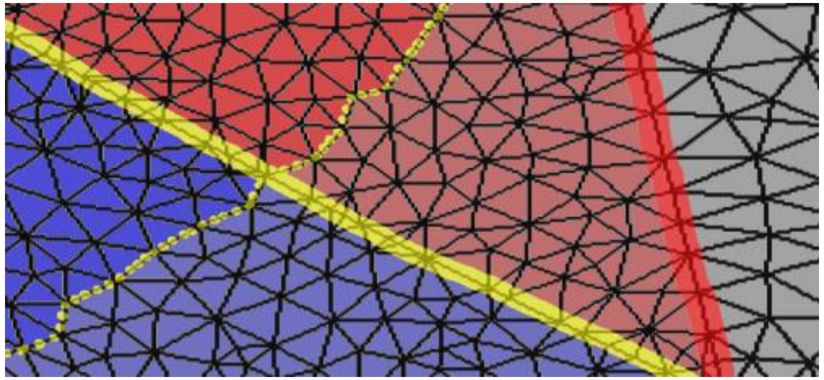
Related work

- **Long history, starting with general solutions**
  - View dependent LOD and progressive streaming [Hoppe 1997]
    - Compute view dependent triangulation each frame -> CPU bound
  - Surface patches [CRS4+ISTI CNR, SIGGRAPH'04]
    - Effective in terms of speed
    - Require non-trivial data structures and techniques for decompression
  - General solutions available for Desktop environments [Cignoni et al, 2005, Yoon et al. 2008]
- **Mesh compression – MPEG-4 [Jovanova et al. 2008]**
- **Light 3D model rendering [MeshPad, PCL]**
- **Gigantic point clouds on mobile devices [Balsa et al. 2012]**
- **… and much more**

Our Contributions: chunked multiresolution structures

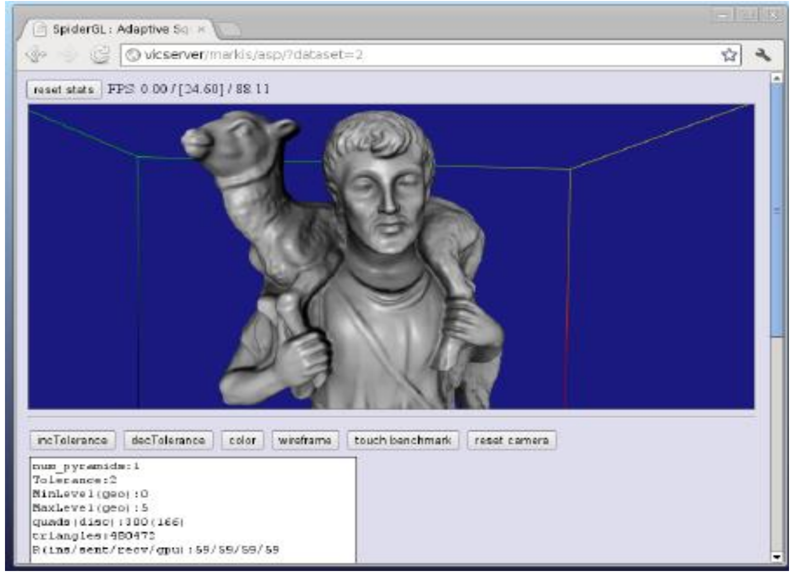
- **Efficient view-dependent meshes**
  - Approximate original surface
  - Seamless
- **Mix and match chunks**
  - Amortize CPU work!
- **Two approaches**
  - **Fixed coarse subdivision**
    - Adaptive QuadPatches
  - **Adaptive coarse subdivision**
    - Compact Adaptive TetraPuzzles





Adaptive Quad Patches
Simplified Streaming and Rendering for Mobile & Web

- **Represent models as fixed number of multiresolution quad patches**
  - Image representation allows component reuse!
  - Natural multiresolution model inside each patch
  - Adaptive rendering handled totally within shaders!
- **Works with topologically simple models**



Javascript!

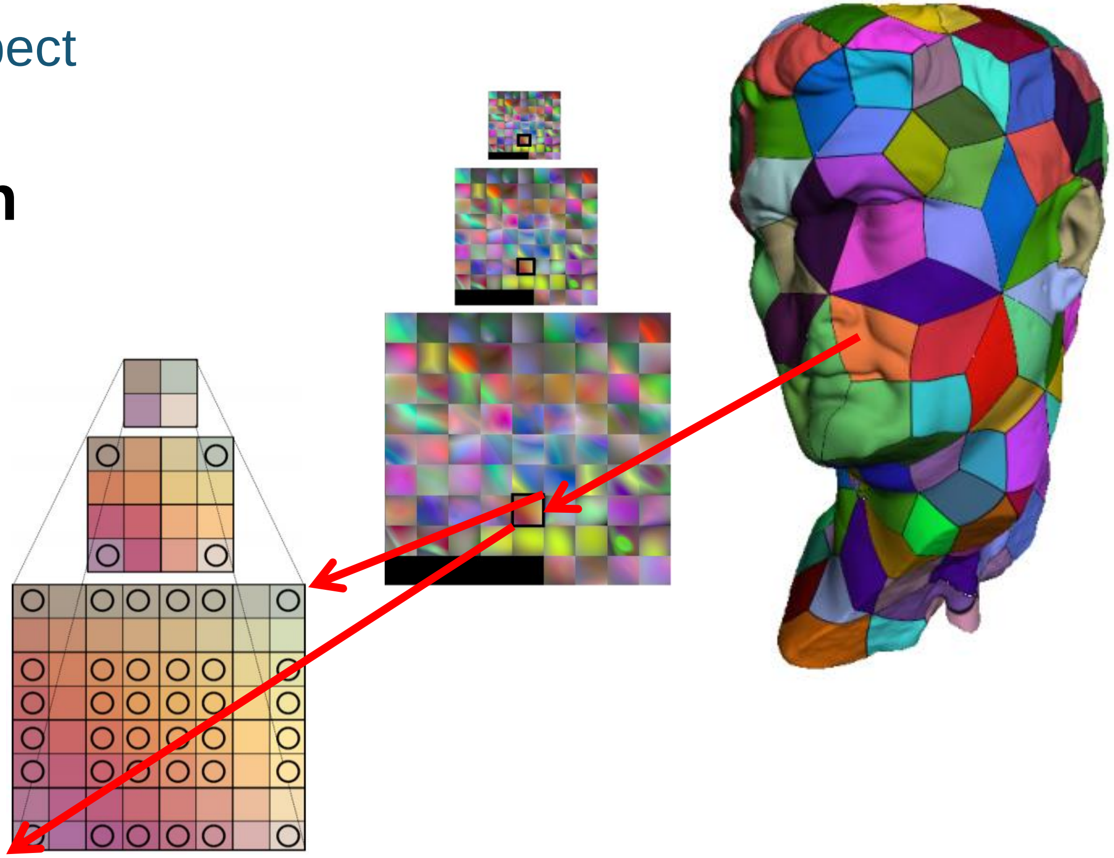
Best paper, WEB3D2012

Related work Adaptive Quad Patches

- **Geometry images [Gu et al. 2002]**
  - Exploit current GPU capabilities / optimized libraries for compression and streaming of images
- **Quad remeshing**
  - Single-disk parametrization [Floater and Hormann 2005]
  - Base mesh to parametrize the model [Petroni et al. 2010]
- **Detail rendering**
  - GPU raycasting [Oliveira et al. 2000]
  - Displacement mapping in GPU [Shiue et al. 2005]

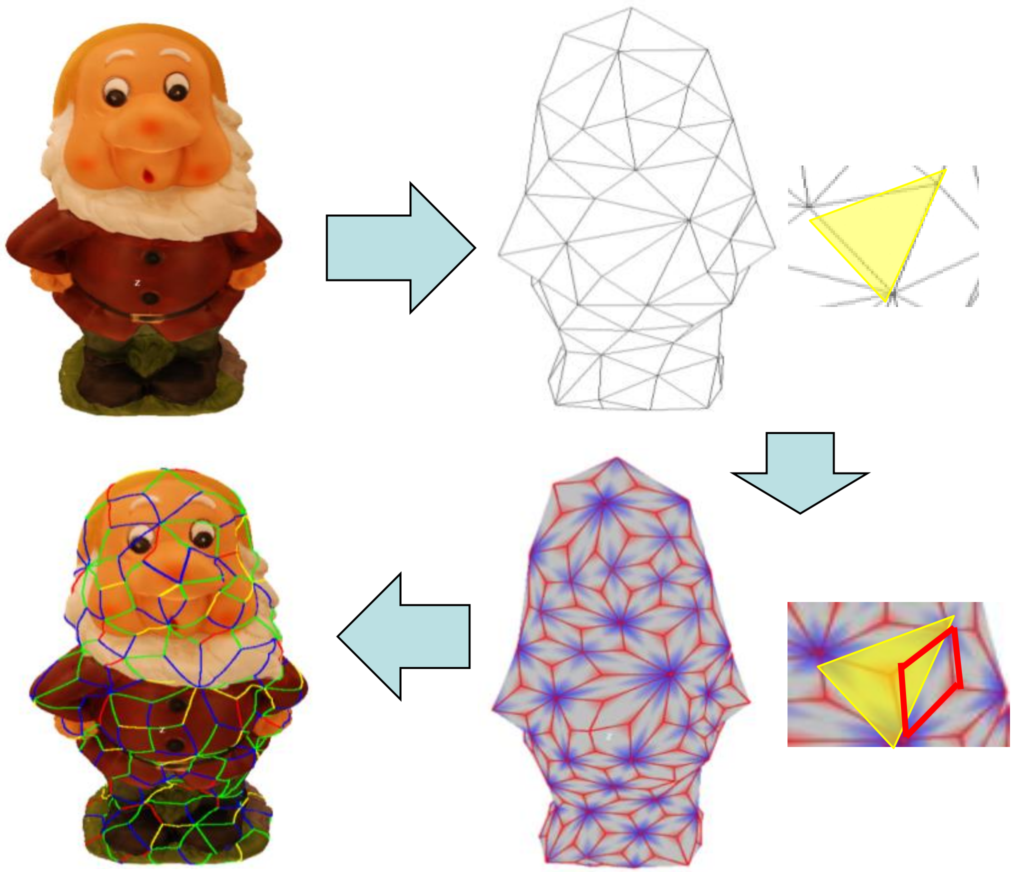
AQP Approach

- **Models partitioned into fixed number of quad patches**
  - Geometry encoded as detail with respect to the 4 corners interpolation
- **For each quad: 3 multiresolution pyramids**
  - Detail geometry
  - Normals
  - Colors
- **Data encoded as images**
  - Exploit .png (lossless compression)
- **Ensure connectivity**
  - Duplicated boundary information



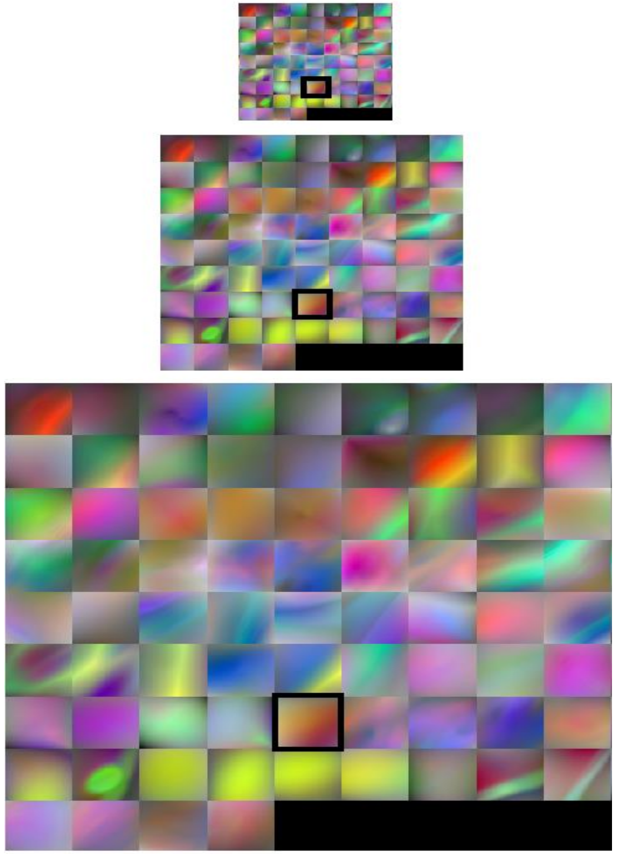
Pre-processing (Reparameterization)

- **Generate clean manifold triangle mesh**
  - –Poisson reconstruction [Kazhdan et al. 2006]
  - –Remove topological noise
    - Discard connected components with too few triangles
- **Parameterize the mesh on a quad-based domain**
  - –Isometric triangle mesh parameterization
    - Abstract domains [Pietroni et al. 2010]
  - –Remap into a collection of 2D square regions
- **Resample each quad from original geometry**
  - –Associates to each quad a regular grid of samples (position, color and normal)



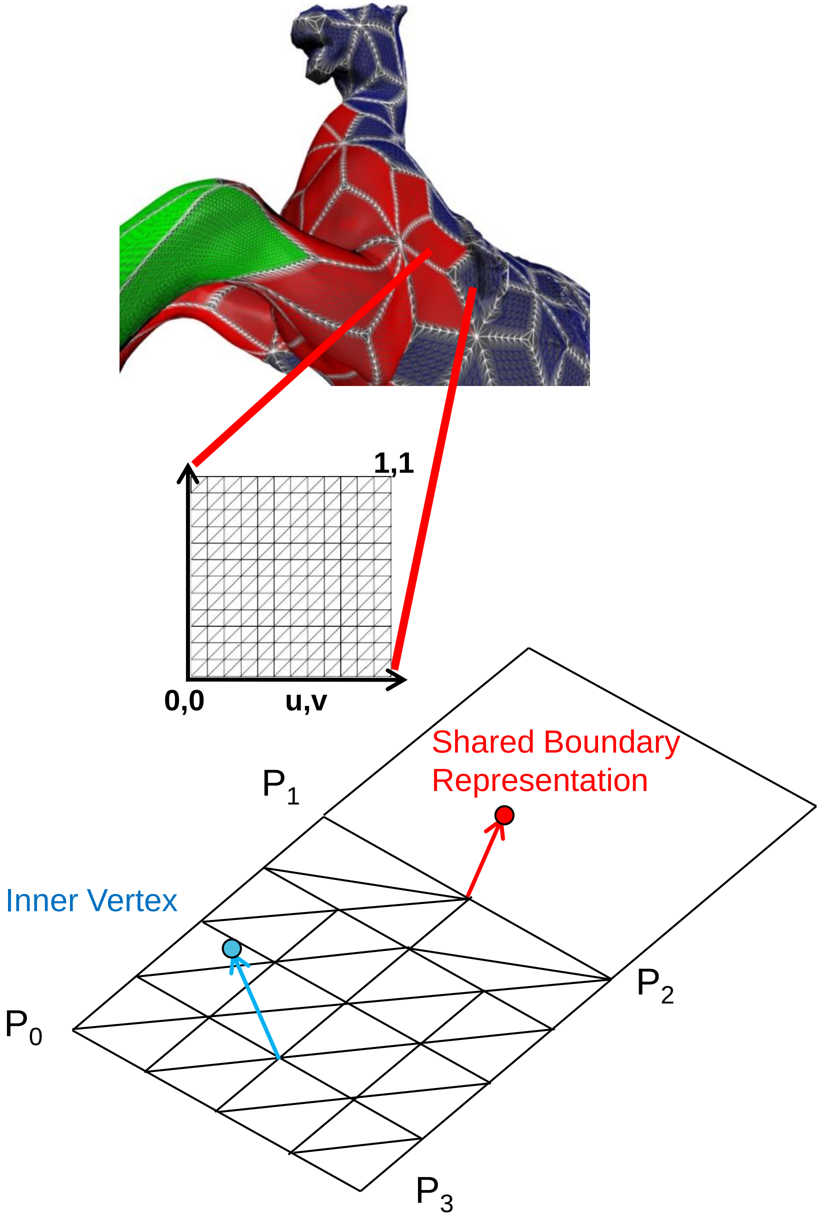
Pre-processing (Multiresolution)

- **Collection of variable resolution quad patches**
  – Coarse representation of the original model
- **Multiresolution pyramids**
  – Detail geometry
  – Color
  – Normals
- **Shared border information**
  – Ensure connectivity



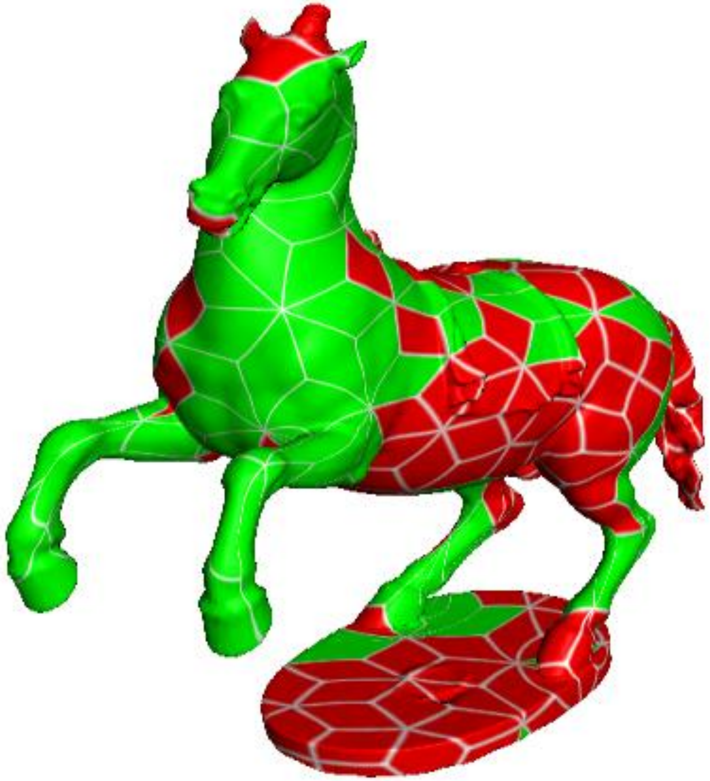
Adaptive rendering

- **1. CPU LOD Selection**
  - Find edge LODs
  - Quad LOD = max edge LODs
  - If data available use it, otherwise
    - Query data for next frames
    - Use best available representation
  - Send VBO with regular grid (1 for each LOD)
- **2. GPU: Vertex Shader**
  - Snap vertices on edges (match neighbors)
  - Base position = corner interpolation (u,v)
  - Displace VBO vertices
    - normal + displacement (dequantized)
- **3. GPU: Fragment Shader**
  - Texturing & Shading



Rendering example

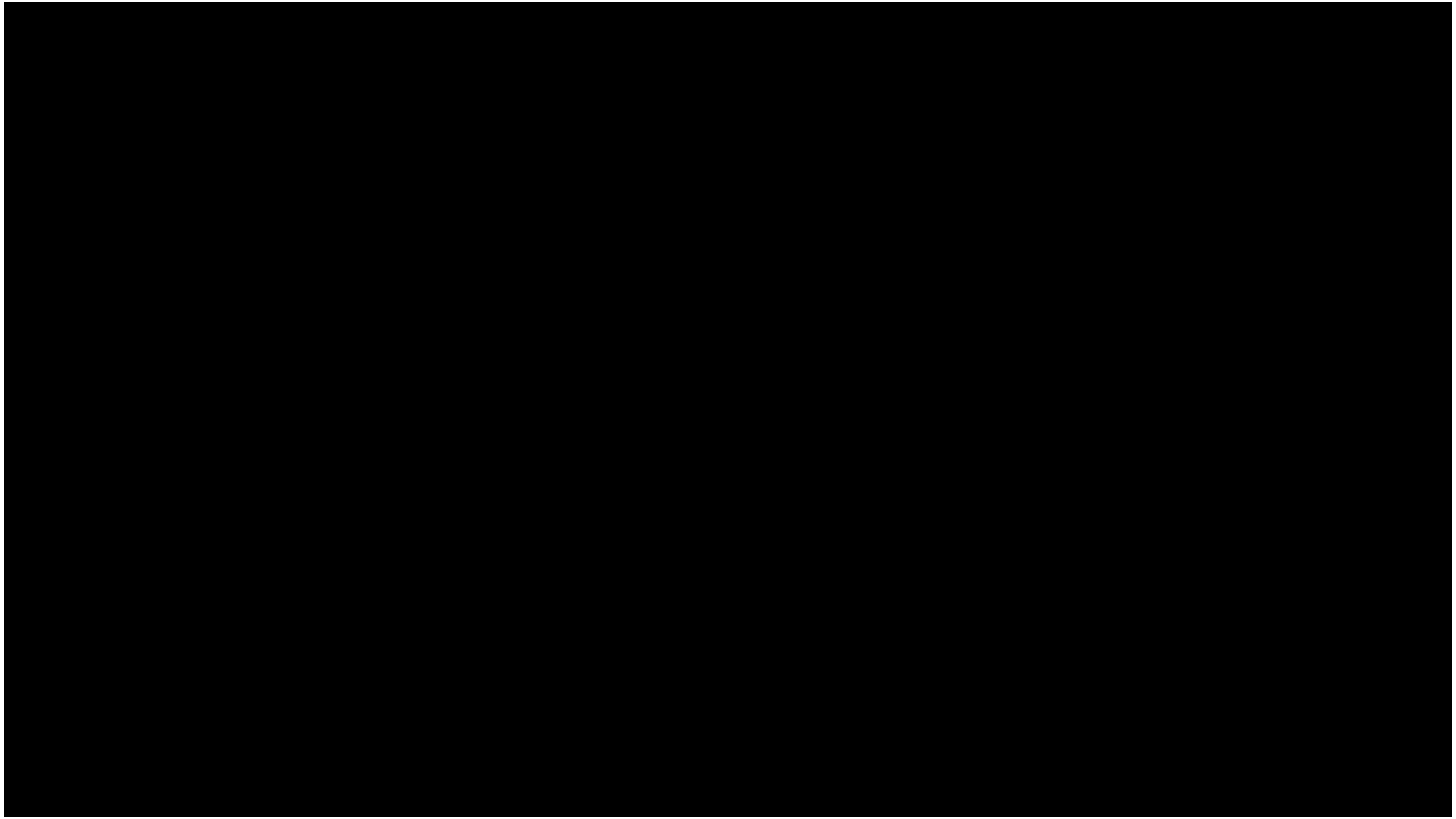
Patches



Levels

Shading

Results



St. Matthew	374 M Tri
Avg bps	24.3 (6.3 + 9.5 + 8.5) (pos + color + normal)
Pixel Accuracy	1
FPS avg	37
FPS min	13
ADSL 8Mbps refine time	2s for model from scratch

Adaptive Quad Patches Conclusions

- **Effective creation and distribution system**
  - –Fully automatic
  - –Compact, streamable and renderable 3D model representations
  - –Low CPU overhead
  - –WebGL
    - Desktop
    - Mobile

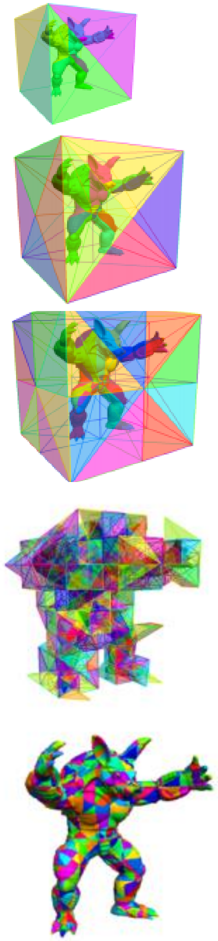
- **Next: More general solution based on full multiresolution structure**

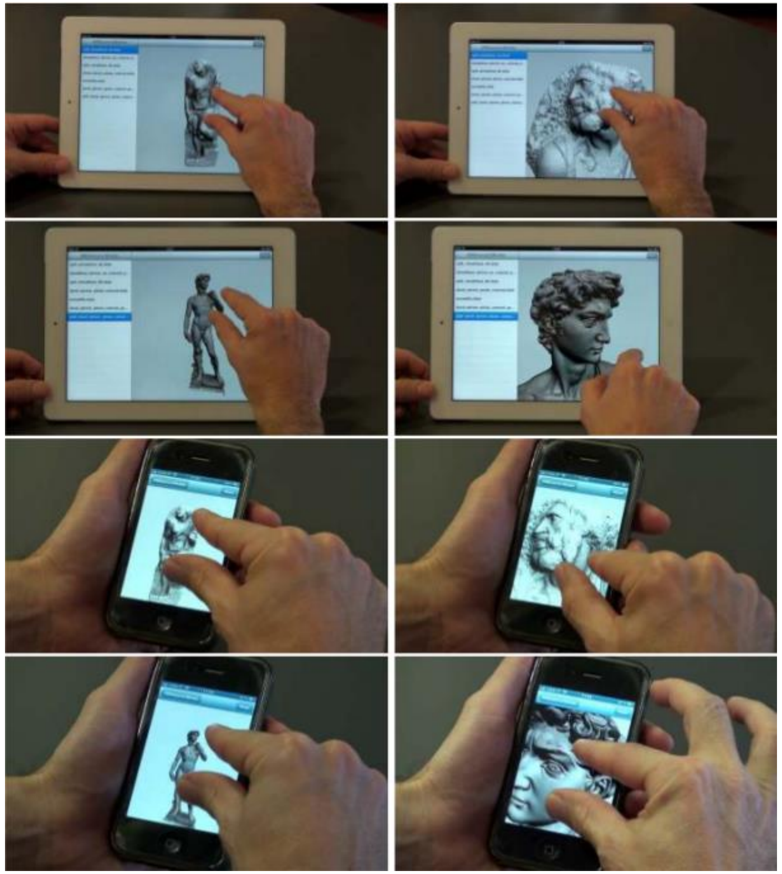
- **Limitations**
  - –Closed objects with large components
  - –Visual approximation (lossy)
- **Extensions**
  - –Explore more aggressive compression techniques
  - –Occlusion culling
  - –More sophisticated shading/shadowing techniques

Compact Adaptive TetraPuzzles

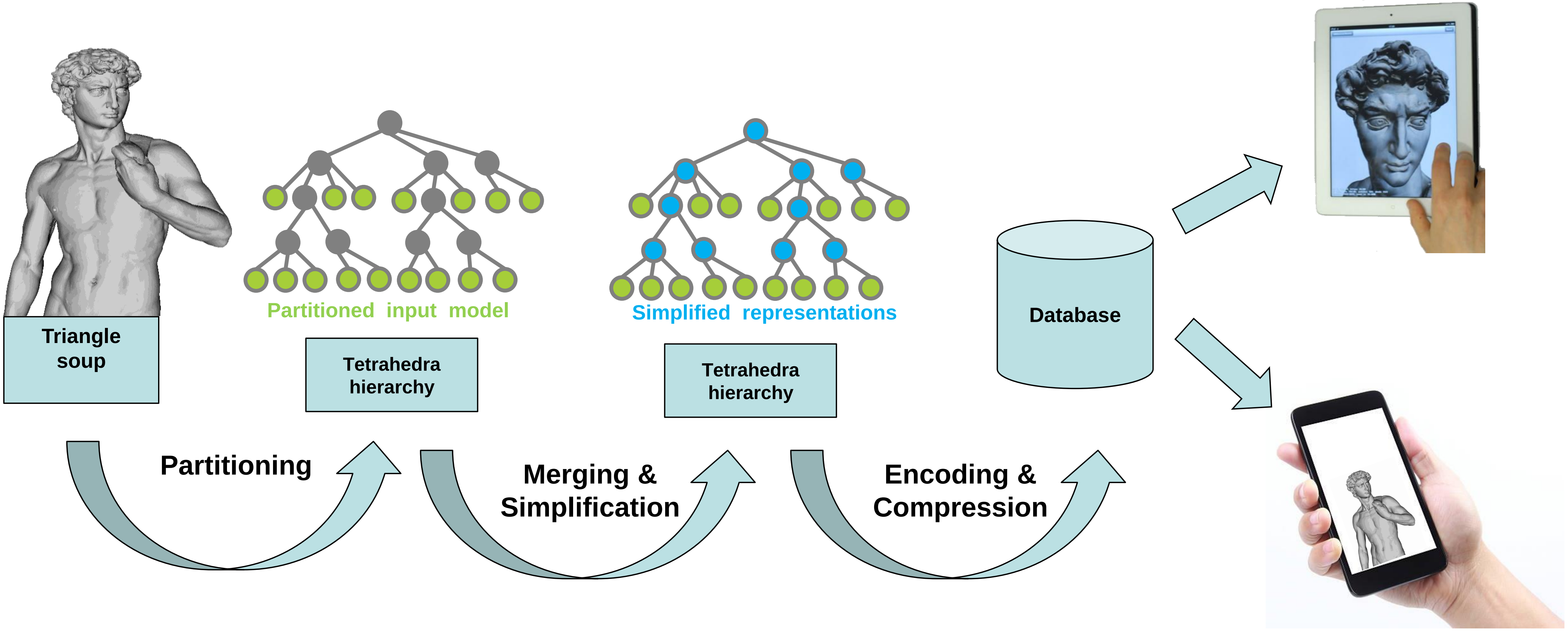
Adaptive multiresolution solution with compression-domain rendering

- **Multiresolution structure with variable number surface patches embedded in a hierarchy of tetrahedra**
  - Fully adaptive and seamless 3D mesh
  - Geometry clipped against containing tetrahedra
  - Local quantization with barycentric coordinates
  - GPU friendly compact data representation
- **Works with general surface models**





Compact Adaptive Tetra Puzzles



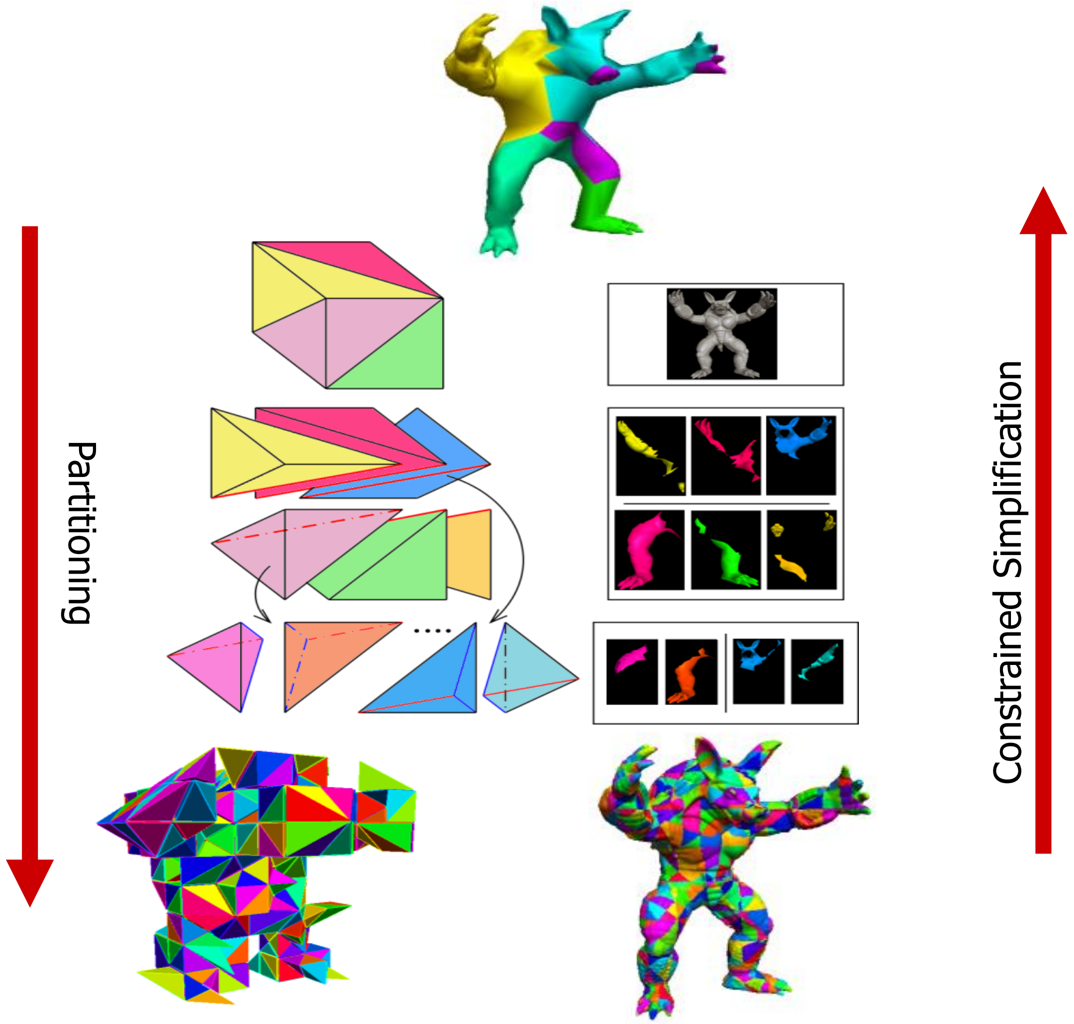
Related work (Compression)

- **Topology coding**
  - Theoretical minimum [Rossignac 2001]
    - 1.62 bits/triangle, 3.24 bits/vertex
  - 8 bpt/16 bpv [Chhugani et al. 2007]
    - HW-implementation
  - 5 bpt/10 bpv [Meyer et al. 2012]
    - CUDA implementation
- **Attribute quantization**
  - Global position quantization [Lee et al. 2009]
  - Local quantization techniques [Lee et al. 2010]
  - Normal compression using octahedral parametrization [Meyer et al. 2010]
- **Our goal is to balance compression rate and decoding+rendering performance by using a GPU-friendly compact representation**

Data Pre-processing

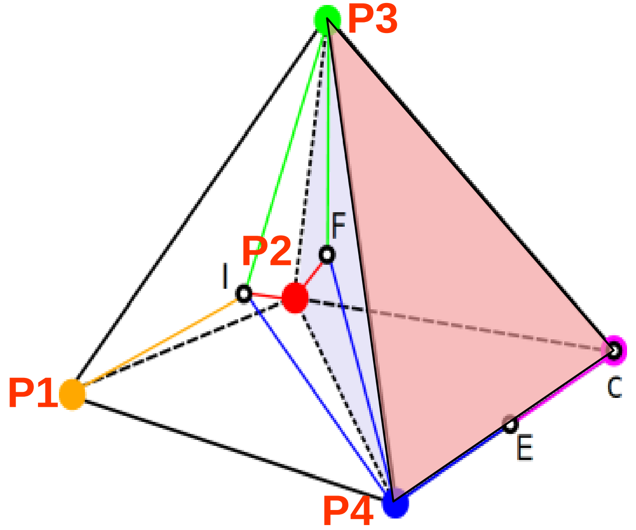
- **Start with hires triangle soup**
- **Partition model using a conformal hierarchy of tetrahedra**
  - Subdivide tetrahedra along longest edge until containing less than N O(10^3) triangles
- **Construct non-leaf cells by lower level cells**
  - bottom-up recombination
  - simplification

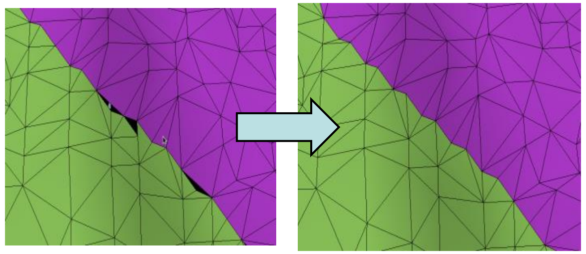
Ensure continuity → Shared information on borders



Data Encoding

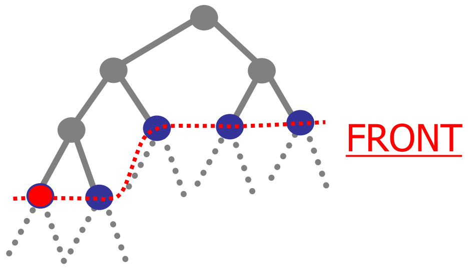
- **Geometry clipped against containing tetrahedra**
- **Vertices: tetrahedra barycentric coordinates**
  - $Pbarycentric = \lambda1*P1+\lambda2*P2+\lambda3*P3+\lambda4*P4$
- **Seamless local quantization**
  - Inner vertices (I): 4 corners
  - Face vertices (F): 3 corners
  - Edge vertices (E): 2 corners
- **GPU friendly compact data representation**
  - 8 bytes = position (3 bytes) + color (3 bytes)+ normal(2 bytes)
  - Normals encoded with the octahedron approach [Meyer et al. 2012]
- **Further compression with entropy coding**
  - exploiting local data coherence

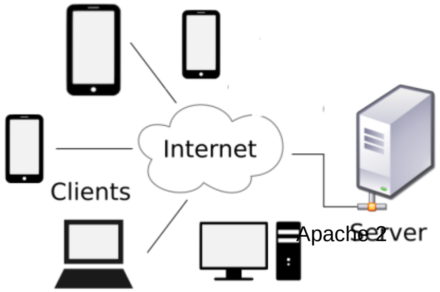


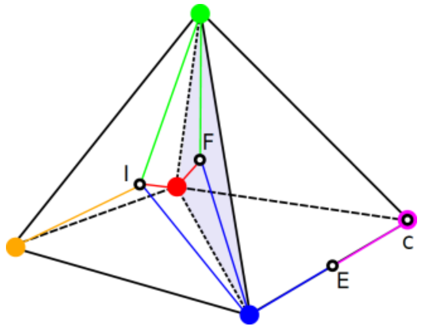


Rendering process

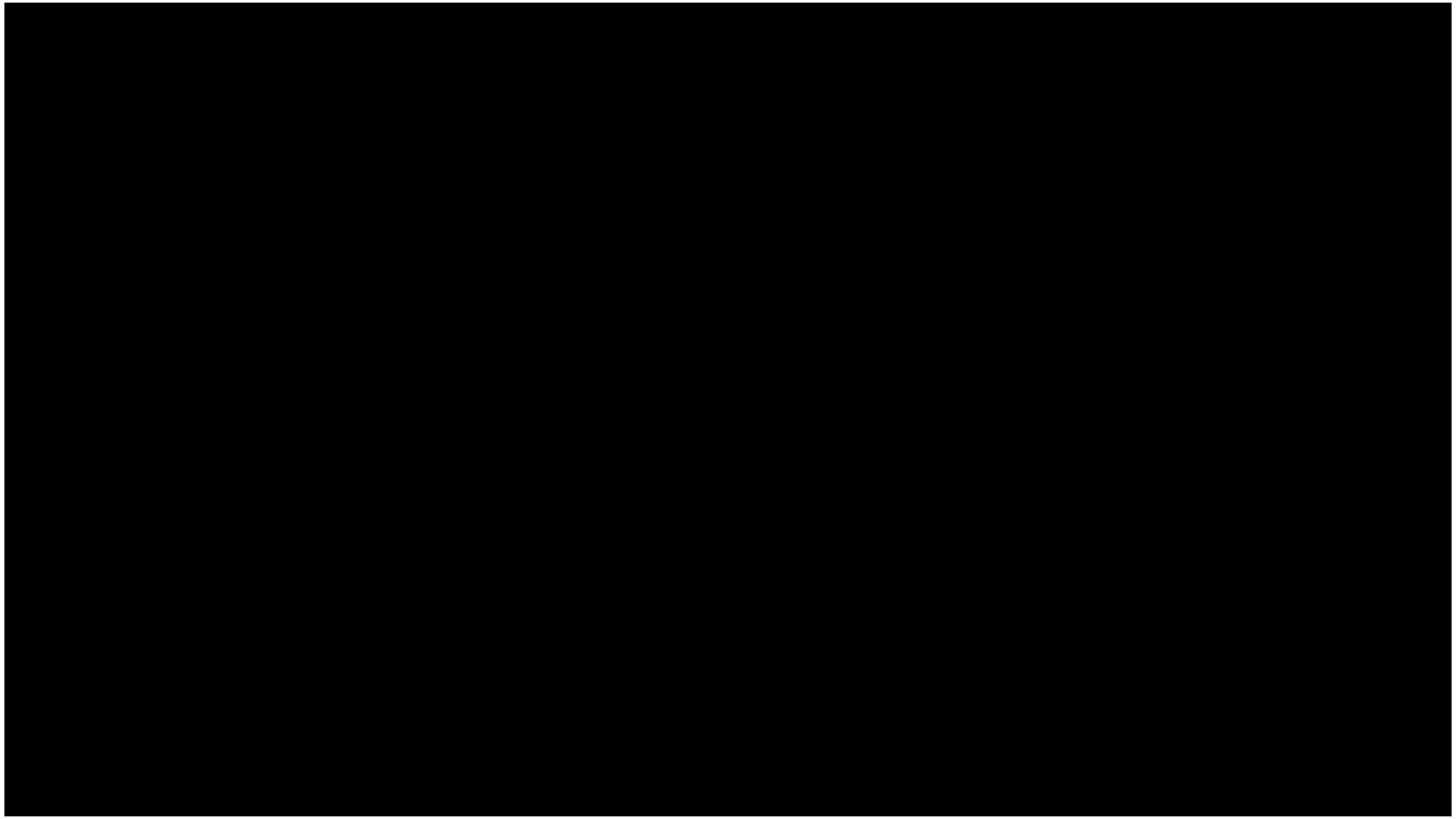
- **Extract view dependent diamond cut (CPU)**
- **Request required patches to server**
  - Asynchronous multithread client
  - Apache 2 based server (data repository, no processing)
- **CPU entropy decoding of each patch**
- **For each node (GPU Vertex Shader):**
  - VBO with barycentric coordinates, normals and colors (64 bpv)
  - Decode position : P = MV * [C0 C1 C2 C3] * [Vb]
    - Vb is the vector with the 4 barycentric coords
    - C0..C3 are tetrahedra corners
  - Decode normal from 2 bytes encoding [Meyers et al. 2012]
  - Use color coded in RGB24







Results

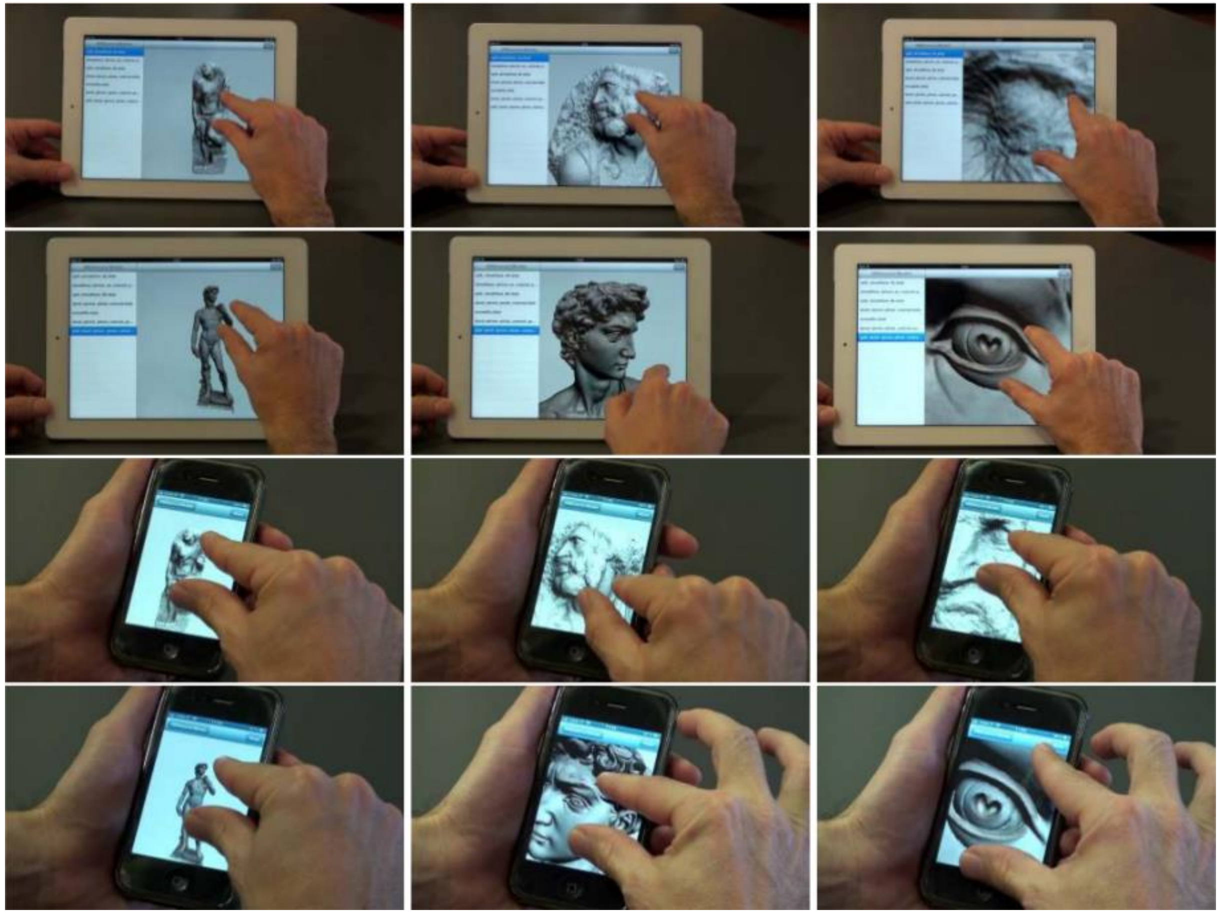


- **Input Models**
  - St. Matthew 374 MTri
  - David 1GTri
- **Compression:**
  - 40 to 50 bits/vertex
- **Streaming full screen view**
  - 30s on wireless,
  - 45s on 3G
  - David 14.5MB (1.1 Mtri)
  - St. Matthew 19.9MB (1.8 Mtri)

Rendering	iPad gen3	iPhone 4
Pixel tolerance	3	3
Triangle throughput	30 Mtri/s	2.8 Mtri/s
FPS avg	35	10
FPS refined views	15	2.8
Triangle Budget	2 M	1 M

Conclusions: Compact ATP

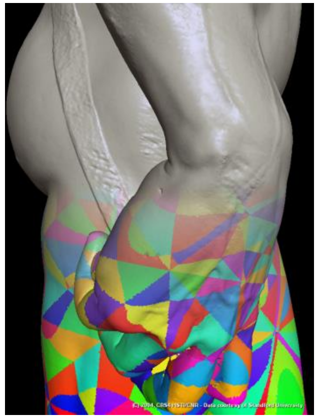
- **Generic gigantic 3D triangle meshes on common handheld devices**
  - Compact, GPU friendly, adaptive data structure
    - Exploiting the properties of conformal hierarchies of tetrahedra
    - Seamless local quantization using barycentric coordinates
  - Two-stage CPU and GPU compression
    - Integrated into a multiresolution data representation
- **Limitations**
  - Requires coding non-trivial data structures
  - Hard to implement on scripting environments



Conclusions: large meshes

- **Various solutions for large meshes**
- **Constrained solution: Adaptive Quad Patches**
  - Simple and fast
  - Good compression
  - Works on topologically simple models

- **General solution: Compact Adaptive Tetra Puzzles**
  - Compact data representation
  - More complex code



15 MINUTES BREAK!

Next Session: Part 4.4

SCALABLE MOBILE VISUALIZATION: INTRODUCTION TO COMPLEX LIGHTING

Part 4.3

Scalable Mobile Visualization: Introduction to complex lighting

Enrico Gobbetti, CRS4

Complex scenes

- **We have seen how to deal with complex meshes O(Gtri)**
  - Similar solutions for point clouds…
- **Problem tackled was size**
  - Solution proposed: adaptive multiresolution chunk-based approaches
  - Various optimized solutions to select chunks, compose them, …
- **Rendering was simple, though**
  - One pass streaming, direct illumination
- **How to deal with more complex illumination and shading?**

Complex scenes

- **Complex illumination/shading introduce data and computation problems**
  - Non-local effects (global illumination, shadows, …) require scattered information
  - Illumination/shading is costly (CPU/GPU time) and requires data-intensive algorithms
- **Proposed solutions in the mobile world**
  - **Full precomputation**
    - Images computed off-line
    - Removes real-time timing constraints, but introduces other problems (which images to compute? How to navigate in an image-based scene?)
  - **Smart computation**
    - Partial precomputation of some intermediate results, approximation tricks
    - Not general solution but improves quality!
- **Next session illustrates examples of full/smart computation in mobile graphics**

Part 4.4

Scalable Mobile Visualization:
Full precomputation of complex lighting

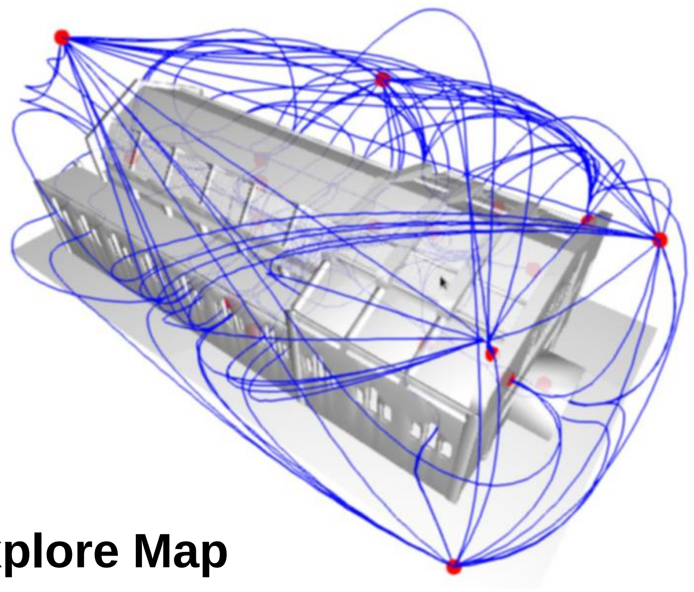
Fabio Marton, CRS4

Ubiquitous exploration of scenes with complex illumination

- **Real-time requirement: ~30Hz**
  - Difficulties handling complex illumination on mobile/web platforms with current methods
- **Image-based techniques**
  - Constraining camera movement to a set of fixed camera positions
  - Enable pre-computed photorealistic visualization
- **Explore-Maps: technique for**
  - Scene representation as set of probes and arcs
  - Precomputed rendering for probes and transitions

Scene Discovery

- **ExploreMaps: Automatic best view/best path methods for generating**
  - Set of probes providing full model coverage
    - Probe = 360° panoramic point of view
  - Set of arcs connecting probes
    - Enable full scene navigation



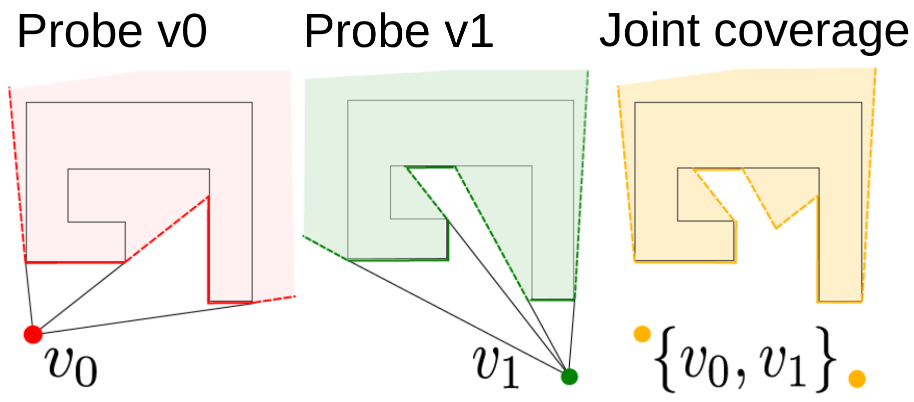
Explore Map

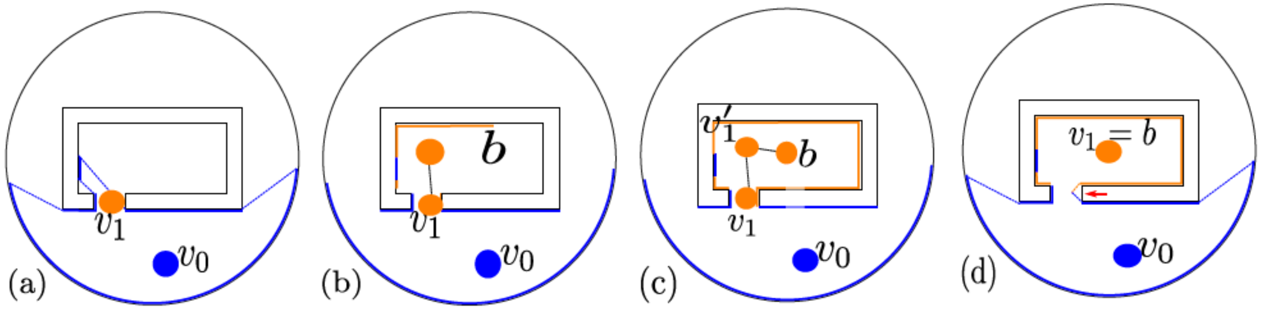
Di Bendeetto et al. Eurographics 2014

ExploreMaps: Efficient Construction and Ubiquitous Exploration of Panoramic View Graphs of Complex 3D Environments.

Best viewpoints computation

- **Position set of probes inside the scene**
  - Probes provide a 360 degree view
  - Greedy algorithm that places probes at the barycenter of newly seen geometry until all the scene is visible
  - Final clustering pass reduces number of probes

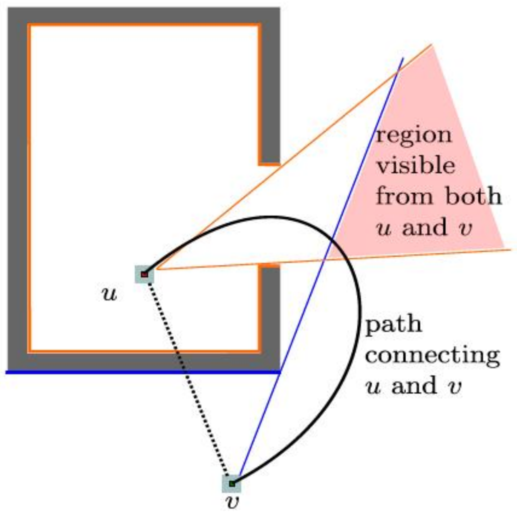


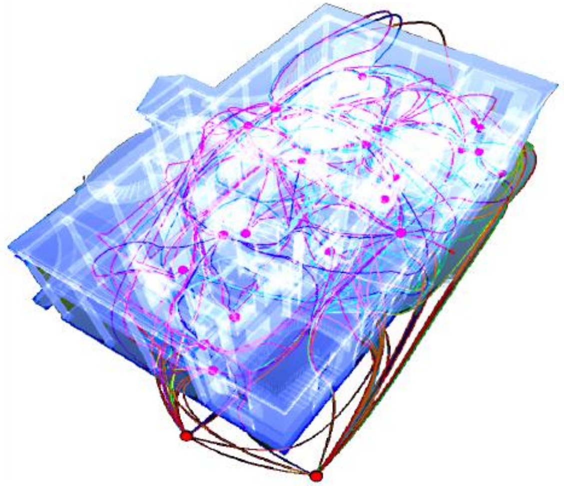


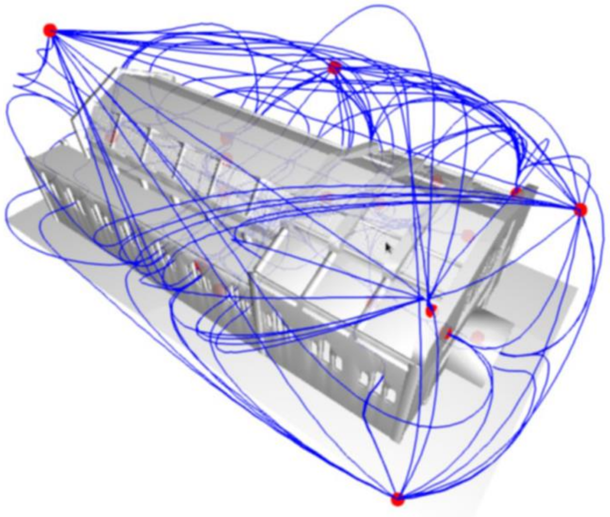
Coverage optimization, by moving to the barycenter of seen geometry

Best path computation

- **Connect probes which have a common visible region**
  – Creates a graph of probes
- **For each pair of mutually visible probe**
  – Create first path going through the closest point in the mutually visible region
  – Optimize and smooth the path using a mass-spring system

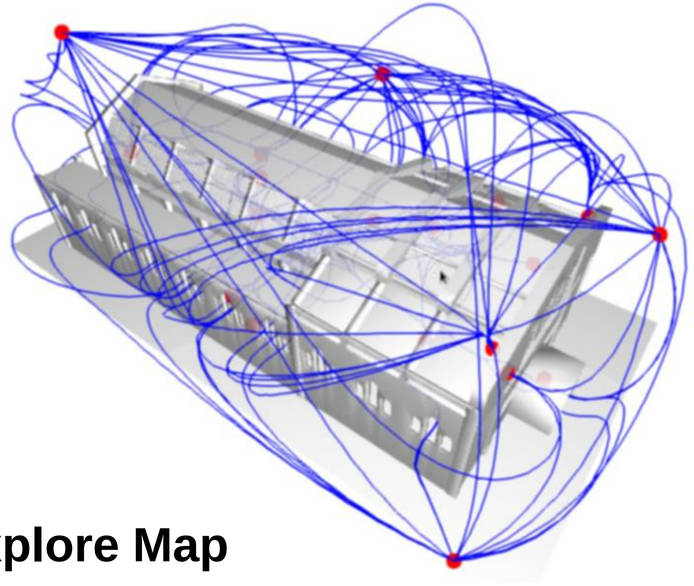






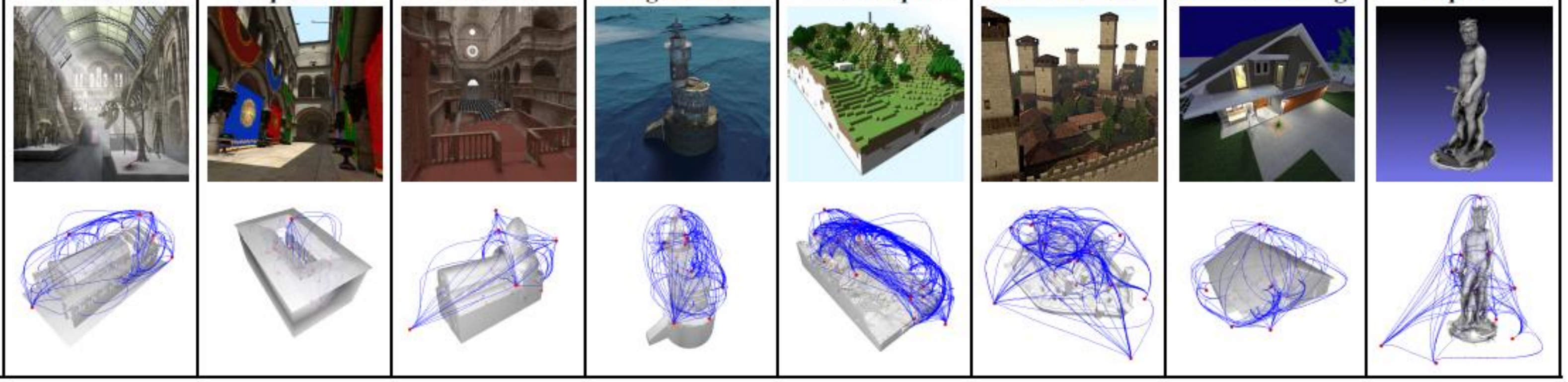
Precomputation of probe images

- **Compute panoramic views for probes and frames of transition arcs**
  - Photorealistic rendering (using Blender 2.68a)
    - panoramic views both for probes and transition arcs
  - 1024^2 probe panoramas
  - 256^2 transition video panoramas
  - 32 8-core PCs,
  - Rendering times ranging from 40 minutes to 7 hours/model



Explore Map

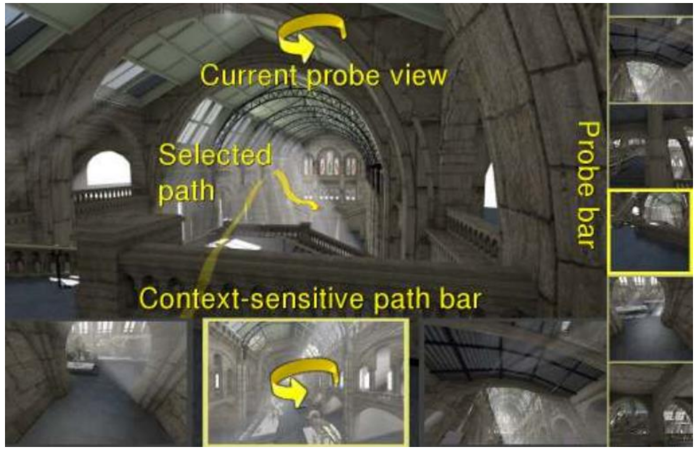
Explore Maps – Processing Results



	Museum	Sponza	Sibenik	Lighthouse	Lost Empire	Medieval Town	German Cottage	Neptune
Input								
#tri	1,468,140	262,267	69,853	48,940	157,136	14,865	79,400	2,227,359
Output								
#probes	70	36	92	57	74	78	140	79
#clusters	17	10	21	17	25	30	23	19
#paths	127	29	58	81	206	222	102	93
Time (s)								
Exploration	154	23	63	15	41	34	163	38
Clustering	17	3	27	8	13	14	118	14
Synthesis	144	35	449	453	284	395	427	279
Path	7	1	31	12	22	80	23	13
Path smoothing	3,012	122	81	89	482	199	185	150
Thumbn.	11	3	7	5	8	10	7	6
Thumbn. pos	2	2	1	1	4	4	2	1
Total	3,347	189	659	583	854	736	925	501
Storage (MB)								
Probes	59	28	72	59	86	103	79	43
Paths	248	146	113	159	371	376	390	120

Interactive Exploration

- **UI for Explore Maps**
  - WebGL implementation + JPEG + MP4
  - Panoramic images: probes + transition path
- **Closest probe selection**
  - Path alignment with current view
- **Thumbnail goto**
  - Non-fixed orientation



Conclusion: Interactive Exploration

- **Interactive exploration of complex scenes**
  - Web/mobile enabled
  - Pre-computed rendering
    - state-of-the-art Global Illumination
  - Graph-based navigation → guided exploration
- **Limitations**
  - Constrained navigation
    - Fixed set of camera positions
  - Limited interaction
    - Exploit panoramic views on paths → less constrained navigation
- **Next part of the talk:**
  - A dynamic solution for complex illumination with smart computation